Modèle Dynamique Temps-Réel pour l'Animation d'Objets Poly-Articulés dans les Environnements Contraints, Prise en Compte des Contacts Frottants et des Déformations Locales : Application en Robotique Humanoïde et aux Avatars Virtuels

Jean-Rémy Chardonnet

ACADÉMIE DE MONTPELLIER

UNIVERSITÉ MONTPELLIER II

- SCIENCES ET TECHNIQUES DU LANGUEDOC -

THÈSE

pour obtenir le grade de :

Docteur de l'Université Montpellier II

par

Jean-Rémy CHARDONNET

Discipline : Génie Informatique, Automatique et Traitement du signal

Formation Doctorale : Systèmes Automatiques et Microélectroniques

École Doctorale : Information, Structures et Systèmes

Titre de la thèse :

Modèle dynamique temps-réel pour l'animation d'objets poly-articulés dans des environnements contraints, prise en compte des contacts frottants et des déformations locales : application en robotique humanoïde et aux avatars virtuels

Soutenue publiquement le :

23 Juin 2009

JURY

Mme Marie-Paule CANI	Professeur à l'INPG, Grenoble	Présidente
M. François PIERROT	Directeur de recherche au CNRS	Directeur de thèse
M. Abderrahmane KHEDDAR	Directeur de recherche au CNRS	Directeur de thèse
M. Martin BUSS	Professeur à TUM, Munich, Allemagne	Rapporteur
M. Fethi BEN OUEZDOU	Professeur à l'Université de Versailles	Rapporteur
M. Eiichi YOSHIDA	Senior Researcher à l'AIST, Japon	Examinateur
M. André CROSNIER	Professeur à l'Université de Montpellier II	Examinateur

Remerciements

Je tiens à remercier d'abord François Pierrot et Abderrahmane Kheddar, mes directeur et co-directeur de thèse respectivement, pour leur soutien, leur confiance et leurs conseils tout au long de cette thèse. Je leur suis très reconnaissant de ce que cette thèse a pu être faite en grande partie au JRL au Japon et au LIRMM à Montpellier.

Je remercie également les membres du jury, Martin Buss et Fethi Ben Ouezdou, pour avoir rapporté sur ma thèse, Marie-Paule Cani, André Crosnier et Eiichi Yoshida, pour l'avoir examinée.

Je veux remercier également toutes les personnes avec lesquelles j'ai travaillé durant cette thèse, François Keith, Anthony David, Hitoshi Arisumi, Adrien Escande, Sylvain Miossec, Nicolas Mansard, Sylvain Garsault et Paul Evrard. Leur aide et leurs conseils ont rendu les collaborations très fructueuses.

Le fait de travailler dans un environnement aussi varié qu'un laboratoire international a été très enrichissant humainement. Etant moi-même bi-culturel, j'ai pu redécouvrir aussi bien la France que, un peu plus, le Japon. Je voudrais remercier tous les membres du JRL, avec qui j'ai passé de très bons moments. J'espère, en échange, leur avoir été utile dans la découverte et la compréhension du Japon.

Je tiens à remercier aussi tous ceux qui, de près ou de loin, ont contribué au bon déroulement de cette thèse, tous mes amis, les familles et amis des membres du JRL, ainsi que le LIRMM.

Je remercie encore une fois Marie-Paule Cani, car, en plus d'avoir examiné ma thèse, elle est ma nouvelle supérieure de travail et elle m'a très gentiment accepté au sein de son équipe, EVASION, avant même ma soutenance, pour travailler sur un sujet passionnant. A ce titre, je remercie aussi toutes les personnes d'EVASION pour leur bienveillance.

Enfin, je voudrais exprimer ma très profonde gratitude envers mes parents qui m'ont donné la chance de faire cette thèse, et envers ma famille, plus particulièrement mes grands-parents japonais qui m'ont accueilli régulièrement chez eux malgré leur âge avancé et les difficultés que cela implique. Le support moral qu'ils m'ont apporté est immense et je leur dédie ce mémoire.

心より皆さんに感謝致します。

Abstract

This thesis deals with the dynamic simulation of multibodies and more specifically humanoid robots. Simulation is a major step to validate physical models on real robots. Indeed these robots are expensive and are aimed at being used in various environments shared with humans and for which physical interaction is of primary concern. Interaction can also be done with virtual environments to realize for example collaborative tasks or to test the robustness of command laws against external perturbations. Therefore real-time simulations must be provided, implying fast and accurate methods. The thesis proposes an overview of the different methods that exist to solve this problem, from the computation of the free dynamics, i.e. without constraint, to the consideration of contact forces and of different non smooth phenomena such as friction, impact, etc. In the frame of a new modular framework called AMELIF developed at the Joint Robotics Laboratory (Japan) and aimed at virtual prototyping, fast and reliable dynamics algorithms taking into account contact with friction are implemented. Especially constraint-based methods are used : they explicitly integrate non-penetration constraints in the dynamics equations and formulate the problem in the operational space as a linear complementarity problem. Compared to other approaches such as penalty-based methods that are implemented for example in OpenHRP, this approach does not need any tuning of parameters and is stable, but needs the computation of the so-called Delassus operator which is the operational space inertia matrix. When adding friction, typically Coulomb's dry friction which is a non-linear law, discretization is needed as the problem becomes non-linear, leading to an increase of the size of the system to be solved and an increase of computation time with a loss of accuracy. To avoid these problems and keep exact friction, iterative methods are implemented. As the thesis deals with simulation, models are numerical, implying numerical integration problems to be considered. The thesis gives an overview of the two main integrators which are the so-called event-driven schemes and the time-stepping schemes. Event-driven schemes are precise but are not suited to situations with a large number of contact points as all events must be detected, whereas time-stepping schemes integrate the whole dynamics on the time step, unifying contact and impact, but need small step sizes to keep an acceptable accuracy as small penetrations are allowed.

The most time consuming parts of the proposed simulator are the collision detection algorithm and the computation of the operational space matrix. The collision detection algorithm detects pairs of contact points, implying specific processes to track these points, that can be time consuming in the case of a large number of contact points. This part can be improved using for example minimum distance

calculation algorithms but is out of scope of the subject. The thesis presents some improvements of the dynamics calculation, more specifically a new algorithm for computing the operational space matrix and introducing collision groups that allow to work with reduced Delassus operators. The proposed simulator is not limited to humanoids and can be extended to multibodies with multiple dofs joints such as human avatars. Among these, spherical joints which are generally decomposed into three pivot joints are considered. They can be described by several representations : rotation matrices, rotation vectors and quaternions. Studies on these different representations in the thesis show that the quaternions are the most trouble-free representation and can easily be integrated into the dynamics computation algorithm. Since the algorithms used in the proposed simulator are fast, interaction with the virtual environment is possible, allowing a user to achieve collaborative tasks or to apply perturbations to the avatars, while sensing force feedback, through a haptic device.

Humanoid robots, especially the HRP-2 robot, have flexibilities in the ankle joints consisting in rubber bushes that can absorb shocks. Indeed, for instance, when the robot is walking, vibrations are generated by the collisions between the foot and the floor, that can damage the mechanical structure of the robot and can represent a non-negligible noise for some specific tasks such as manipulation tasks or vision tasks, if they are not reduced. Flexibilities could also be found in the future as external compliant skins that better fit asperities of the environment and can thus better absorb shocks. These flexibilites could also ensure the safety of humans sharing environments with humanoid robots as well as protect the mechanical structure of the robot. The thesis proposes a way to include these flexibilities in simulation. Internal flexibilities are easily integrated by choosing a simple kinematic representation of the mechanism, whereas external flexibilities are integrated using finite element models. The contact model in the rigid case is then augmented by the deformation component, leading to a formulation that can be solved as for the rigid case. The resulting method is applied, in a first study, to external soles mounted under HRP-2 robot's feet, as the long-term goal is to design a human-like walking pattern that is robust to perturbations caused by a user or by the asperities of the environment. Therefore different parameters such as shapes and materials of the soles and control laws must be considered to meet the requirements of such patterns. The thesis gives some directions to optimize the shape of the soles.

One way to evaluate the quality of a simulator is to make a precise comparison between the values given by real forces sensors and the values of the forces computed in simulation. This requires to identify parameters such as the coefficients of friction between all the bodies, the model of the force sensors, etc., which is a highly time-consuming process and does not necessarily prove the reliability of a simulator. The thesis gives a very simple example of such validation but also shows that the real validation of the proposed simulator, which makes it reliable, is not to make such comparison but to use it in various applications such as extreme manipulation tasks or posture generation, for which simulations, and thus experiments, could not be realized if the physical models that are implemented were not correct.

Table des matières

Table des figures

Liste des tableaux

Chapitre 1

Introduction

L'essor de l'informatique a favorisé le développement de ce qu'il est convenu d'appeler la mécanique numérique. Ce domaine trouve beaucoup d'applications, en particulier on peut citer les jeux vidéos et les films d'animation, mais aussi la robotique et la simulation numérique. Certaines applications même peuvent conditionner les développements de nouveaux produits. Pour obtenir des rendus réalistes et donc fiables, il est nécessaire de bien comprendre les phénomènes physiques, en particulier il est important de veiller à respecter les lois issues de la mécanique. Les simulations de bonne qualité physique sont souvent très longues à réaliser en raison de la complexité des méthodes mises en œuvre. Cependant, pour des applications pour lesquelles le rendu visuel est le plus important, comme les films d'animation, le temps de calcul n'est pas une priorité. Par ailleurs, généralement, ces applications ne requièrent pas d'être interactives.

A l'inverse, pour les jeux vidéos, le temps de calcul est une contrainte majeure car l'utilisateur doit interagir avec l'environnement virtuel, sans nécessairement ressentir des forces ; aussi le réalisme physique n'est pas un élément essentiel, on demande juste que le comportement des objets de la scène virtuelle paraisse réaliste. Aussi les méthodes de calcul de la dynamique mises en œuvre seront généralement simples et rapides. Cependant, depuis quelques années, on observe de plus en plus de contributions de la part de la communauté des jeux vidéos dans le domaine de la simulation.

Pour des applications robotiques ou médicales, mais aussi dans l'aviation ou l'automobile, on doit pouvoir obtenir des simulations physiques réalistes et rapides pour permettre une bonne interactivité avec l'environnement virtuel. En effet, des lois de commande sont développées pour faire se mouvoir les avatars virtuels, et il est nécessaire que l'utilisateur ressente des forces qui l'aident à réaliser correctement diverses tâches comme la manipulation collaborative. La fidélité des efforts ressentis par rapport à la réalité est donc tributaire de la qualité du simulateur. L'interactivité avec retour d'efforts suppose des calculs en temps-réel, pour que l'utilisateur ait la sensation de ressentir des forces continues. Cela

entraîne de revoir les modèles physiques, de mettre au point des nouvelles méthodes de résolution et d'avoir une bonne implémentation informatique, avec toujours la contrainte qui est que ces méthodes doivent respecter le plus possible les lois physiques. Il est important de noter cependant qu'il faut utiliser des lois physiques adaptées au type d'applications recherché et qu'il est possible de s'autoriser quelques approximations pour ne pas trop complexifier les calculs. Ainsi, dans le domaine de la robotique, nous n'avons pas besoin de connaître les phénomènes physiques au niveau microscopique pour obtenir des comportements réalistes des robots, en l'occurrence les lois de la mécanique classique de Newton suffiront.

Ainsi, ce travail a pour but de créer un environnement de simulation dynamique interactive pour des avatars robotiques avec calcul des forces de contact et de frottement, et retour haptique. Nous organiserons notre travail comme suit :

- Dans un premier chapitre, nous introduirons le cadre de cette étude. Nous présenterons les travaux réalisés par d'autres équipes dans le domaine de la simulation. Nous décrirons les différents composants nécessaires pour concevoir un simulateur physique : nous parlerons d'abord des méthodes de calcul de la dynamique non contrainte (en l'absence de forces de contact par exemple), puis nous présenterons les travaux sur la prise en compte de phénomènes physiques issus de la mécanique non régulière comme le contact avec frottements. Nous décrirons entre autres les méthodes par pénalités et les méthodes par contraintes qui sont les méthodes les plus courantes. Une fois la dynamique complète calculée, la configuration du système est mise à jour et de fait, nous aborderons l'intégration numérique des systèmes mécaniques non réguliers.
- Dans un deuxième chapitre, nous présenterons notre simulateur dynamique interactif qui est une partie intégrante d'AMELIF, logiciel destiné au prototypage virtuel. Nous exposerons les différents algorithmes de calculs utilisés dans ce simulateur et nous montrerons des premiers exemples de simulations. Voyant que les temps de calcul sont encore élevés pour réaliser des simulations interactives, nous verrons qu'il est possible d'améliorer les méthodes de calcul de la dynamique, en particulier nous nous concentrerons sur l'amélioration du calcul de la matrice d'inertie dans l'espace opérationnel (l'espace des contacts). Notre simulateur est principalement orienté vers les humanoïdes mais il est également conçu pour simuler tous types d'avatars virtuels ; aussi nous étendrons notre simulateur à des corps possédant des articulations sphériques tels que les avatars humains. Enfin, grâce à ces algorithmes très rapides de calcul de la dynamique, nous pourrons intégrer un dispositif haptique permettant de réaliser des simulations interactives avec retour d'effort.
- Dans un troisième chapitre, nous nous intéresserons à la prise en compte des flexibilités dans les robots humanoïdes. Nous aborderons deux types de flexibilités : (i) interne telle que celle présente dans les chevilles du robot HRP-2, à savoir un système de bagues en caoutchouc qui absorbe les chocs lors de la marche sur un sol, et (ii) externe telle qu'une peau flexible. Après avoir modélisé chacune des flexibilités, la première par des articulations passives, la seconde par la méthode des éléments finis, nous effectuerons une comparaison entre ces deux types pour déterminer laquelle des deux flexibilités absorbe le mieux les chocs. Nous nous baserons pour cela sur la marche du

robot HRP-2 et dans le cas de la flexibilité externe nous placerons une semelle compliante simple sous les pieds du robot. Cela nous amènera à nous intéresser davantage à la forme de la semelle à placer sous les pieds.

– Dans un quatrième et dernier chapitre, nous nous intéresserons à la validation de notre simulateur. En particulier, nous présenterons un scénario simple avec le robot HRP-2 pour lequel nous avons identifié différents paramètres physiques comme le coefficient de frottement entre le robot et les objets de l'environnement, les modèles des capteurs et des actionneurs présents dans le robot HRP-2. Par ailleurs, nous montrerons quelques applications ayant fait usage de notre simulateur pour la réalisation de tâches extrêmes et la génération de postures. A chaque fois, nous confronterons les résultats obtenus en simulation et ceux données par l'expérimentation réelle. Enfin, nous montrerons un exemple d'interfaçage avec AMELIF, en l'occurrence la librairie de simulation de mécanique non régulière SICONOS développée à l'INRIA Rhône-Alpes.

Chapitre 2

Etat de l'art, simulation dynamique, contact frottant

Ce chapitre introduit le problème abordé dans cette thèse et le contexte dans lequel nous l'avons abordé. Il s'agit de réaliser un simulateur robotique, plutôt dédié aux humanoïdes, avec des modèles permettant, d'une part, une simulation fidèle de la réalité capteur, et, d'autre part, répondant aux critères d'interactivités requis pour la simulation de différentes perspectives d'usage des humanoïdes dans des espaces co-localisés avec l'homme afin de réaliser des tâches physiques collaboratives. Forcément, le sujet va bien au-delà de la robotique, il couvre aussi la simulation interactive des mannequins digitaux et peut trouver des applications dans l'industrie. Aussi, nous présentons les travaux réalisés par d'autres équipes dans le domaine de ce qui est désormais convenu d'appeler la "simulation physique"; cet état de l'art n'est certainement pas exhaustif dans les références bibliographiques, tant le domaine est bien fourni, mais nous pensons l'avoir été plutôt dans les idées clés sous-jacentes.

Sommaire

Résumé

Pour introduire notre travail, nous expliquons d'abord l'intérêt de la simulation dynamique dans la robotique humanoïde et nous montrons quelques exemples de simulateurs existants, en particulier la plateforme OpenHRP2 [Kanehiro *et al.*, 2004]. Nous prenons un exemple de simulation montrant les limites de ce simulateur. Nous exposons ensuite les différents modèles dynamiques pour simuler des corps, articulés ou non. La partie suivante est consacrée à la résolution du contact, d'abord sans prendre en compte le phénomène de frottement. Nous introduisons ensuite le frottement sec de Coulomb et nous voyons comment l'appliquer aux différentes méthodes de résolution du contact. Enfin, sur la base de travaux récents, nous présentons les différentes méthodes d'intégration numérique permettant d'obtenir la nouvelle configuration du système.

2.1 Motivation

2.1.1 Intérêt des simulations dynamiques

Actuellement présentés pour le divertissement comme le robot Asimo de Honda ou encore ceux joueurs de musique de Toyota (voir figure 2.1), les robots humanoïdes sont en réalité destinés dans le futur à fournir divers services à l'homme et à être intégrés comme une composante à part entière des systèmes à base de technologies d'information et de communication, et dans des environnements de travail collaboratifs divers [1]. L'anthropomorphisme qu'offre ce type de robot entraîne quelques réticences, par rapport à l'éthique ou à la religion ; de même l'idée qu'un jour ce type de robots puisse surpasser l'homme alimente quelques craintes, ce qui est normal. En revanche cela permet tout de même de ne pas avoir à restructurer les infrastructures de l'environnement d'utilisation qui ont été conçues et façonnées par et pour l'homme. Plus important encore, n'importe quelle personne non spécialiste de la robotique, pourra facilement deviner ce dont sera capable ou pas cette machine, car elle a été conçue à notre image fonctionnelle. A l'évidence, l'usage des humanoïdes posera moins de problème d'acceptation sociale dès lors qu'ils seront utilisés dans le contexte d'applications hostiles à l'homme ou pour faire des tâches collaboratives en partenariat avec l'homme (voir figure 2.2). Ces robots pourraient donc être vus comme une aide précieuse pour l'être humain et l'arrivée de ces robots pourrait provoquer un changement culturel de la société.

Cependant, les robots ne peuvent pas réaliser ces tâches évoluées et collaboratives s'ils n'acquièrent pas une certaine connaissance de l'environnement, car elles induisent une interaction physique avec l'environnement, voire avec l'homme, et donc le traitement du contact. Ces capacités seront de plus en plus développées avec l'arrivée de nouvelles théories et de modèles. Néanmoins, vu le coût et la complexité de ce type de robot, il est prudent d'envisager systématiquement de tester ces modèles en simulation avant de les porter sur la plate-forme réelle : la simulation est donc un outil essentiel des développements envisagés. Par ailleurs, avec le développement des techniques de réalité virtuelle, des jeux vidéos de plus en plus réalistes, la simulation physique se banalise de plus en plus. En particulier,

1. `http://www.robot-at-cwe.eu`

figure 2.1: *Exemples de robots humanoïdes (à gauche : les robots joueurs de musique de Toyota. Au centre : Asimo de Honda. A droite : HRP-2 de Kawada).*

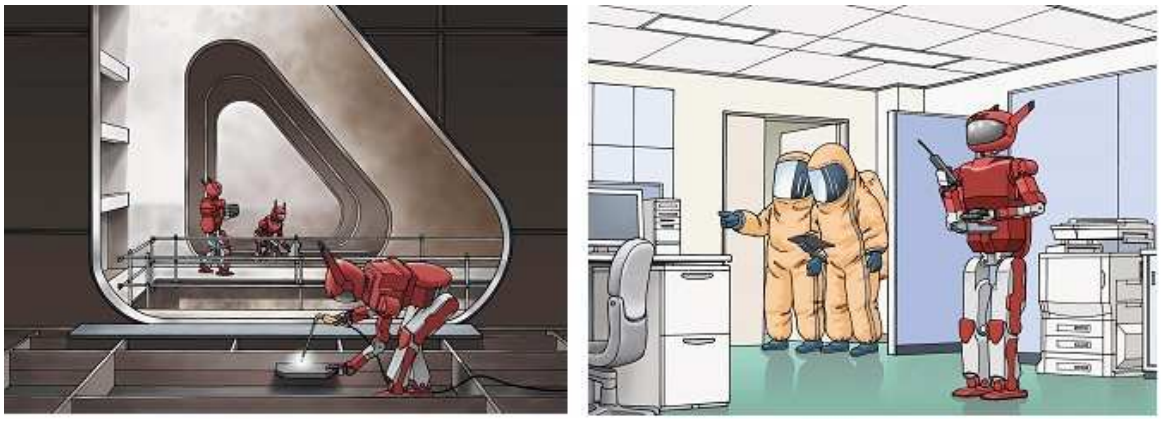


figure 2.2: *Exemples de tâche collaborative par des robots humanoïdes (à gauche : réalisation de travaux dans un immeuble en construction. A droite : inspection d'un environnement contaminé).*

l'interactivité dans la simulation permet d'enrichir les données fournies par la simulation, donc de permettre une meilleure réalisation des tâches physiques, et d'augmenter la sensation d'immersion dans le monde virtuel.

2.1.2 Déroulement d'un pas de temps de simulation

Nous présentons ici le schéma de fonctionnement d'un simulateur interactif (voir figure 2.3). Les objets présents dans la scène sont soumis aux lois physiques de la mécanique usuelle de Newton, à diverses interactions qui peuvent se présenter comme des perturbations, sans considérer dans notre travail celles au niveau microscopique telles que les forces entre atomes. Ces lois physiques permettent de calculer le mouvement des objets et les forces qui s'exercent sur eux. Les efforts d'interaction ne sont calculés que s'il y a collision entre les objets de l'environnement. Il est donc nécessaire de détecter s'il y a collision ou non, et dans le cas où il y a collision, de déterminer quels sont les corps en collision ainsi que les zones de collision. Avec ces informations, les lois du contact sont prises en compte avec

frottements éventuellement, pour obtenir les forces d'interaction. Ces forces sont ensuite ajoutées à la dynamique du système en mouvement libre. Quand l'utilisateur interagit avec la scène virtuelle, il applique des mouvements ou des forces connues car données généralement par le dispositif haptique grâce auquel il interagit avec la simulation. Ces forces sont également ajoutées à la dynamique. Une fois la dynamique complète calculée, la configuration globale du système est mise à jour et ensuite rendue à l'écran.

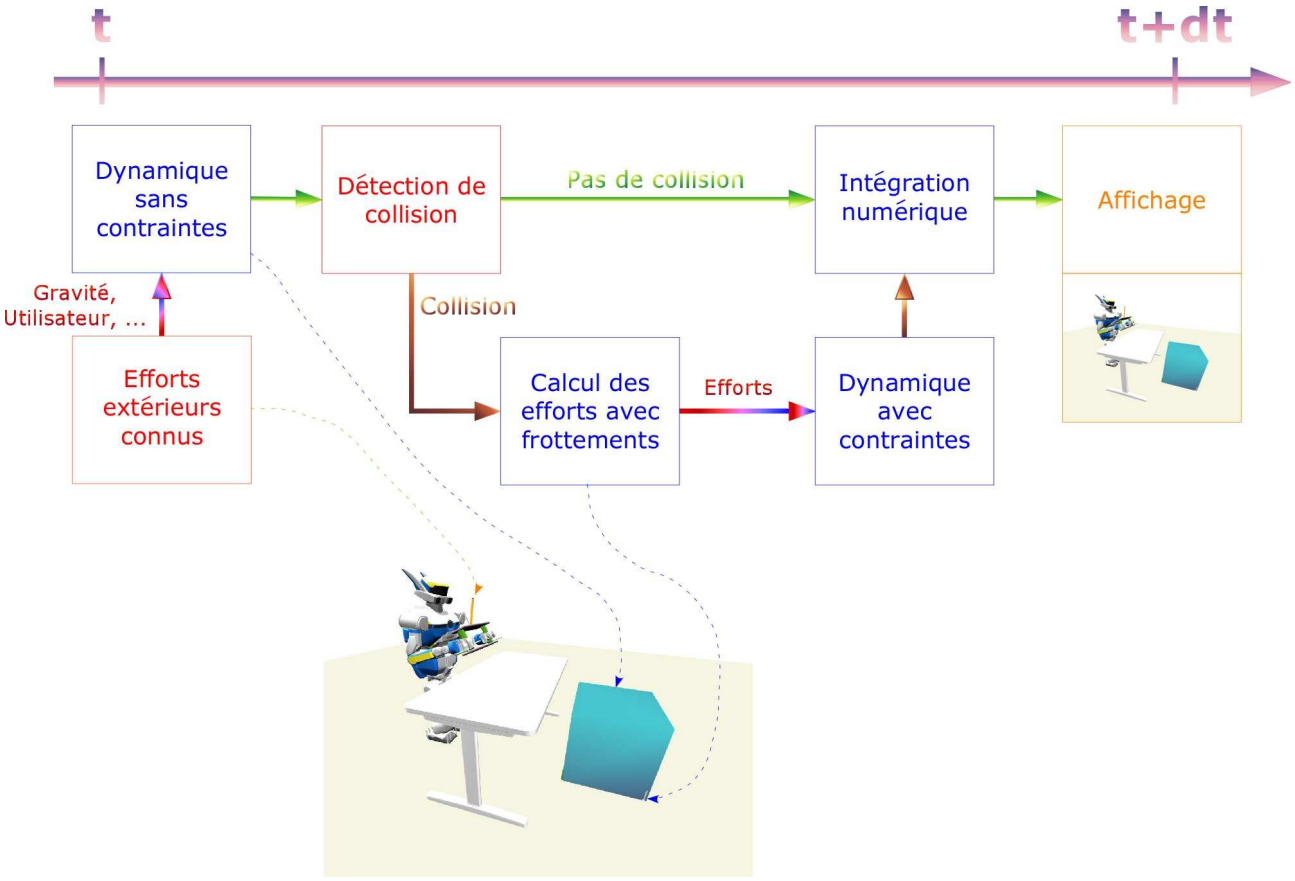


figure 2.3: *Déroulement d'un pas de temps de simulation.*

Le calcul des efforts d'interaction est la partie la plus coûteuse d'un pas de temps de simulation et aussi la partie faisant l'objet de plus d'investigations. Grâce aux différents travaux menés, le temps de calcul de cette partie a pu être réduite considérablement et a pu permettre de réaliser des simulations en temps réel. Pour obtenir une simulation en temps réel, il est nécessaire que le temps de calcul d'un pas de temps soit inférieur ou égale au pas de temps choisi.

2.1.3 Simulateurs existants

Une bonne partie des simulateurs proposés dans la littérature ont généralement été développés pour l'animation ou la génération de mouvements d'acteurs numériques [Baraff, 1994, Hodgins et Wooten, 1998, Weinstein *et al.*, 2006, Shapiro *et al.*, 2007]. L'interactivité avec retour d'effort ne semble pas avoir été une préoccupation majeure malgré la disponibilité sur le marché

des dispositifs haptiques. Dans la communauté roboticienne les simulateurs ont été proposés pour la commande ou la planification [Ruspini et Khatib, 1999, Kanehiro *et al.*, 2004, Son *et al.*, 2004, Hale *et al.*, 2008, Nagasaka *et al.*, 2008]. Nous présentons une sélection de quelques simulateurs particuliers avec leurs avantages et leurs inconvénients. Vu le contexte dans lequel s'est effectué notre travail, nous nous attacherons plus particulièrement au simulateur OpenHRP car il a servi de modèle pour notre simulateur et d'interface pour des expérimentations sur le robot réel HRP-2.

2.1.3.1 OpenHRP2

L'AIST [2] a développé une plate-forme de simulation dédiée aux robots, plus particulièrement centrée sur le robot humanoïde HRP-2 [Kanehiro *et al.*, 2004] [3], appelée OpenHRP2 [4]. Ce simulateur permet d'effectuer des simulations dynamiques et d'exécuter ensuite les actions simulées sur le robot réel. Il est constitué de plusieurs modules : un module de chargement des modèles, un module de détection de collision, un module de calcul dynamique, et un module de commande. Dans la partie dynamique, ce simulateur utilise une méthode par pénalités basée sur les travaux de Yamane et Nakamura [Yamane et Nakamura, 2006] pour le calcul des efforts de contact. Dans cette méthode, comme nous l'expliquerons dans la partie 2.3.1, on ne calcule une force que lorsqu'un objet pénètre dans un autre. Cette pénétration, une fois quantifiée, est pénalisée par une force qui s'écrit, dans le cas courant où cette force correspond à un modèle "ressort-amortisseur" :

$$f = K\delta x + C\delta\dot{x} \tag{2.1}$$

où δx et $\delta\dot{x}$ sont, respectivement, la distance et la variation de la distance de pénétration, et K et C sont les paramètres du ressort-amortisseur virtuel.

Ce simulateur figure parmi les plus développés et fiables, cependant, il possède des inconvénients majeurs.

D'une part le modèle de contact n'est pas satisfaisant : il est inhérent au choix de la méthode de calcul des efforts car la méthode d'intégration numérique utilise l'algorithme de Runge Kutta d'ordre 4. Un des désavantages réside dans le choix des gains de pénalisation K et C : ils ont été optimisés pour la marche de manière à ce que le comportement soit le plus proche possible de celui du robot réel et surtout conforme au capteur d'effort mis sous le pied. La raideur K est relativement élevée pour : (i) empêcher que le pied ne pénètre dans le sol et aussi pour avoir un impact pied/sol cohérent en amplitude avec la réalité, (ii) rendre compte de la flexibilité de la cheville (mécanisme d'absorption de choc de HRP-2), en reportant ses effets sur l'interaction pied/sol. Il nous a été reporté que le réglage des gains de la simulation de la marche s'est fait sur une année en bouclant entre les expérimentations réelles et les données de la simulation ; alors cela pose problème pour l'interaction de tout autre corps. Nous illustrons notre propos par un exemple de simulation réalisée avec OpenHRP2. Puisque ces robots

2. National Institute of Advanced Industrial Science and Technology, Japon
3. `http://www.is.aist.go.jp/humanoid/openhrp/`
4. La version 3 inclut nos idées et est basée sur la simulation du contact par une méthode par contraintes.

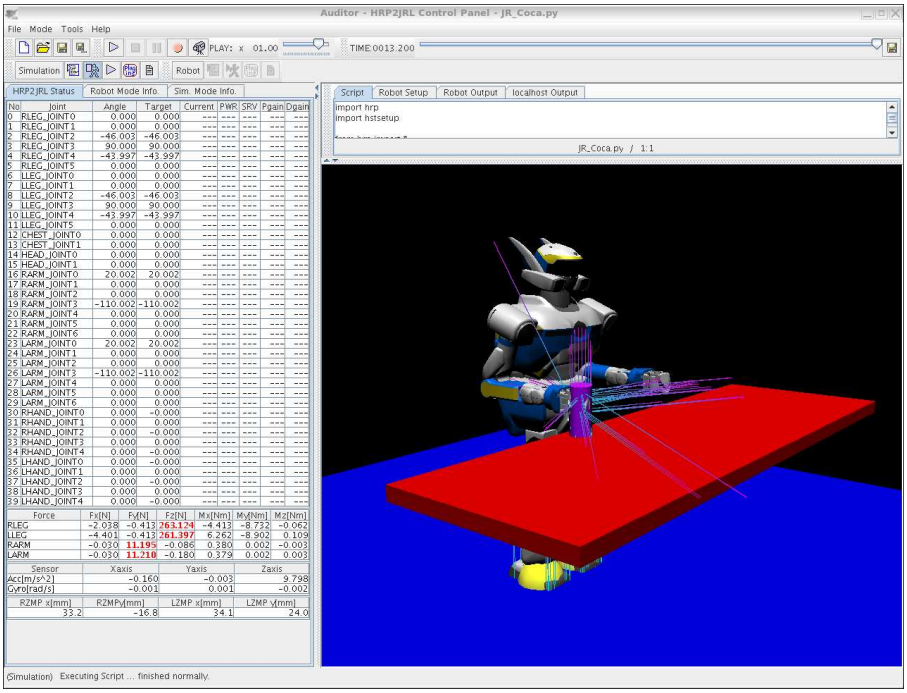


figure 2.4: *Simulateur OpenHRP2.*

n'ont pas été conçus que pour marcher, nous demandons au robot de prendre une canette posée sur une table (voir figure 2.4). Sans surprise, nous pouvons constater dans la figure 2.5, que lorsque la pince du robot se ferme, celle-ci pénètre complètement dans la canette avec des forces de plus en plus grandes et la simulation finit par diverger. Cette instabilité est due d'une part à un choix des paramètres K et C qui n'est pas adapté à cette situation et d'autre part parce que le système pince-canette devient une boucle fermée qui crée des oscillations numériques.

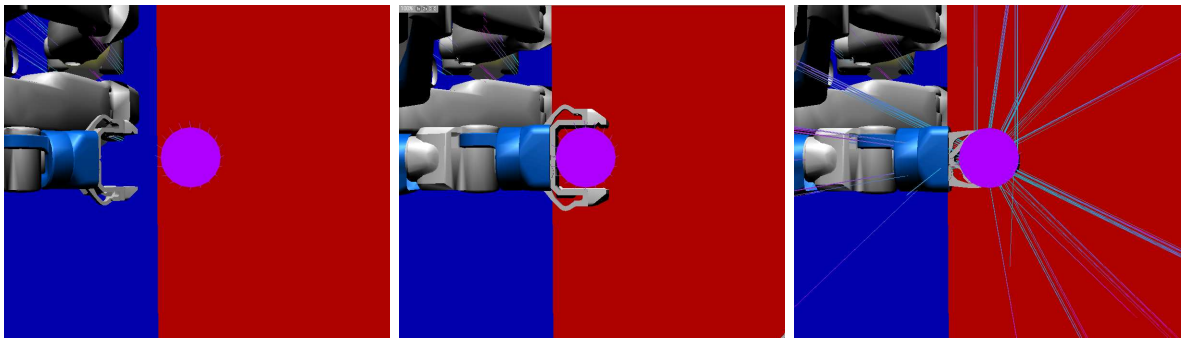


figure 2.5: *Tâche de saisie d'une canette avec OpenHRP2.*

En plus du problème précédent, on peut vouloir appliquer des forces externes sur le robot ou encore simuler une interaction homme-robot avec un dispositif à retour d'effort. Le simulateur OpenHRP2 est dépourvu de toute fonctionnalité d'interaction utilisateur-environnement virtuel ou utilisateur-robot et le seul moyen d'ajouter des forces extérieures au système est d'ajouter manuellement dans le code le point d'application et la valeur de la force.

Enfin, comme nous l'avons mentionné, le robot réel possède un mécanisme d'absorption des chocs

au niveau des chevilles. Ce mécanisme introduit des flexibilités qu'il faut intégrer dans le modèle dynamique. OpenHRP2 prend en compte ces flexibilités en les modélisant par des ressorts-amortisseurs virtuels dont les paramètres ont été identifiés expérimentalement. Nous pensons que le modèle adopté pour OpenHRP est cependant trop approximatif et n'est pas suffisant pour simuler fidèlement l'absorption de chocs.

Dans cette thèse et vu la proximité que nous avons avec OpenHRP, nous nous sommes attelés à pallier ces trois limitations, et aussi à la simulation réaliste des phénomènes d'interactions non linéaires tels que les frottements secs et dynamiques.

2.1.3.2 SAI

Le simulateur SAI de l'équipe de Khatib [Ruspini et Khatib, 1999] [5] (voir figure 2.6) est un des premiers simulateurs à utiliser un retour haptique pour une interaction avec des avatars virtuels. Pour résoudre les forces d'interaction, les méthodes par contraintes, qui incluent explicitement les conditions de non-pénétration dans les équations de la dynamique comme nous l'expliquerons dans la partie 2.3.2, et la matrice d'espace opérationnel introduite par Khatib [Khatib, 1987] sont utilisées ; notons que les bases scientifiques de ces travaux ont été jetées déjà par Gauss et ensuite par Delassus [Delassus, 1923] depuis bien longtemps. Ce simulateur permet également d'effectuer la commande de robots humanoïdes ou de robots manipulateurs redondants avec le formalisme de pile des tâches ordonnancées par priorités. L'inconvénient majeur de SAI est l'absence de détails sur la prise en compte des frottements.

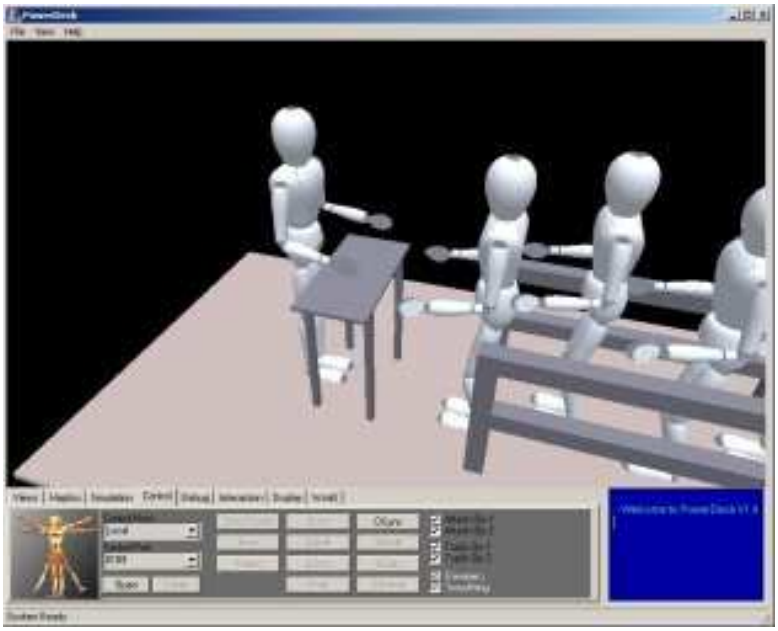


figure 2.6: *Simulateur SAI. Ce simulateur permet d'interagir avec l'environnement virtuel.*

5. `http://ai.stanford.edu/groups/manips/projects/sai/index.html`

2.1.3.3 Simulateur de SONY

Dans la même catégorie que SAI, Nagasaka *et al.* [Nagasaka *et al.*, 2008] ont développé une plate-forme pour le contrôle de robots humanoïdes avec la possibilité d'interagir avec la simulation via des dispositifs haptiques. Cette plate-forme détermine les efforts dans les articulations qui satisfont les contraintes imposées. Pour cela, deux étapes sont nécessaires : d'abord, des forces virtuelles sont calculées en résolvant un problème de complémentarité linéaire (voir partie 2.3.2), afin d'éviter des instabilités numériques. Ces forces sont ensuite transformées en efforts dans les articulations et en efforts d'interaction. Ce simulateur propose aussi un stabilisateur utilisant un modèle de contrôle prédictif. Il a été développé de manière modulaire sous la forme d'un réseau de fonctions qui communiquent entre eux.

2.1.3.4 GraspIt !

Avec OpenHRP2 la saisie d'objets est quasiment impossible. Certains simulateurs comme GraspIt ! [Miller et Christensen, 2003] sont dédiés à la saisie d'objets. La différence majeure avec OpenHRP2 provient du modèle de contact. GraspIt ! utilise des méthodes par contraintes pour la résolution. Pour le calcul de la dynamique les corps sont décomposés en groupes qui peuvent être liés par un contact ou une articulation, et des contraintes additionnelles sont intégrées pour les limites articulaires. Leur simulateur s'applique bien à des robots manipulateurs. Cependant la complexité de l'algorithme par rapport au nombre de degrés de libertés et de points de contact n'est pas mentionnée. Mais comme les auteurs ont opté pour une représentation algébro-différentielle matricielle des équations de la dynamique, il faut aussi satisfaire les contraintes des liaisons par la simulation, ce qui en général pose deux problèmes : la taille des matrices et leur inversion est coûteuse en temps de calcul, en mémoire et aussi en erreurs numériques, et tester à chaque pas de temps que les contraintes de liaisons sont bien satisfaites est une opération complexe.

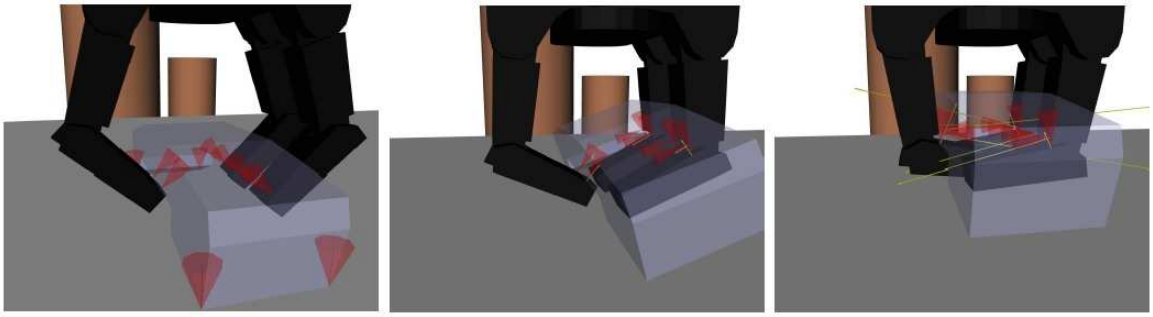


figure 2.7: *Simulateur GraspIt ! (tiré de [Miller et Christensen, 2003]). Contrairement à OpenHRP2, ce simulateur utilise des méthodes par contraintes, ce qui empêche les objets de pénétrer entre eux.*

2.1.3.5 Simulateur de Hale *et al.*

Ce simulateur [Hale *et al.*, 2008][6] (voir figure 2.8) est dédié aux robots humanoïdes et seul le contact entre les pieds du robot et le sol est considéré. A chaque pas de temps de la simulation, la configuration optimale de contact qui respecte les contraintes (non-pénétration, frottement) est déterminée. Pour déterminer la meilleure configuration de contact, une comparaison entre deux états de contact possibles est effectuée en fonction d'une liste de contraintes à respecter en priorité. L'état qui respecte le plus de contraintes prioritaires est choisi. Cependant ce processus est très dépendant de l'intégration numérique et donc sujet à des erreurs numériques. Par ailleurs, les auteurs ne montrent pas l'efficacité de leur méthode dans le cas de simulations complexes multi-contacts. En effet, comme indiqué plus haut, seul le contact pied-sol est considéré ce qui simplifie le traitement du contact. Enfin, ici encore, il n'est pas possible d'interagir avec l'environnement via une interface haptique.

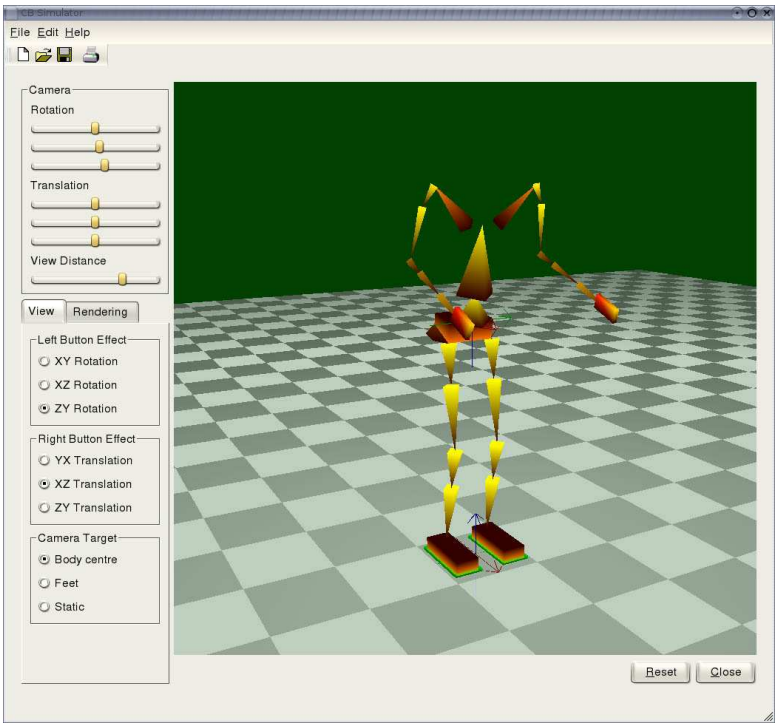


figure 2.8: *Simulateur de Hale* et al.

2.1.3.6 Remarques et but

Les différents exemples de simulateurs présentés ci-dessus font apparaître principalement deux méthodes pour résoudre le contact : les méthodes par pénalités et les méthodes par contraintes. Avec les méthodes par pénalités, la simulation peut devenir très instable et ne permet pas de modéliser correctement les frottements. Nous développerons ces deux méthodes dans la partie 2.3.

La plupart des simulateurs présentés n'intègrent pas la possibilité d'interagir avec l'environnement virtuel. En interfaçant un dispositif haptique à retour d'effort, il est possible de ressentir le contact

6. `http://www.cns.atr.jp/~jhale/academic_research.html`

avec l'environnement virtuel et les efforts appliqués, donc d'enrichir les informations retournées par la simulation. L'interaction avec l'environnement virtuel permet de réaliser par exemple diverses tâches collaboratives ou bien perturber en ligne le robot lors d'un mouvement afin de vérifier la robustesse d'un contrôle prédictif.

Nos objectifs sont d'avoir un simulateur dynamique réaliste interactif complet, autrement dit qui résoud les problèmes de contact avec frottement de manière physiquement plausible, qui permet d'interagir avec l'environnement virtuel et qui simule tous types de robots, y compris ceux à peaux déformables comme les androïdes ou les futurs robots humanoïdes. Le simulateur OpenHRP est pour l'heure incapable de remplir ces fonctions. Enfin, la plupart des simulateurs existants ne sont pas disponibles librement. Nous avons donc créé notre propre simulateur, que nous présenterons en détail dans le chapitre suivant. Ce simulateur a permis le développement de la nouvelle version d'OpenHRP, OpenHRP3, qui intègre les méthodes par contraintes mais qui n'est toujours pas interactif et qui ne permet pas de simuler des corps à peaux flexibles [Nakaoka *et al.*, 2007].

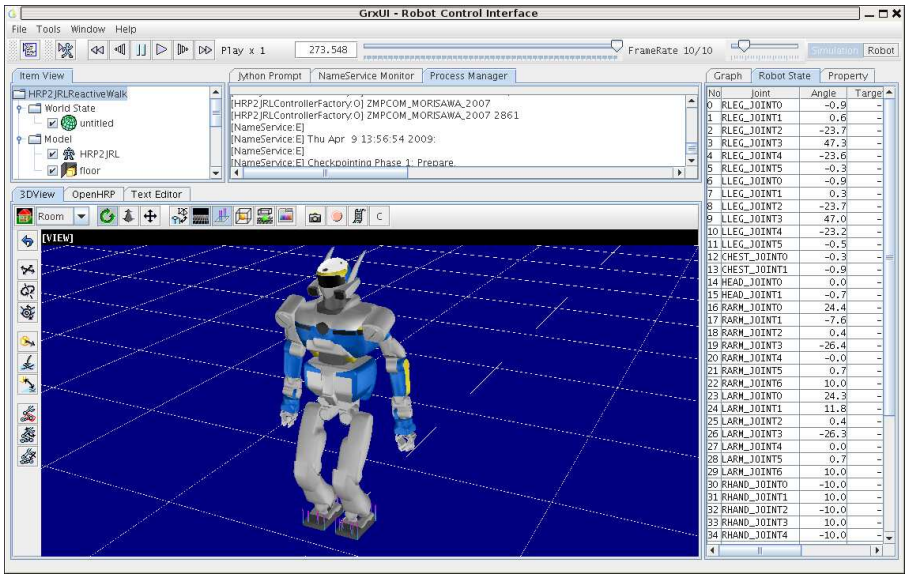


figure 2.9: *Simulateur OpenHRP3.*

Nous présentons maintenant les différents outils pour la simulation dynamique : le modèle dynamique, le calcul des efforts d'interaction avec la prise en compte du frottement et l'intégration numérique.

2.2 Modèle dynamique

2.2.1 Introduction

Nous pouvons distinguer deux types de modèles en robotique selon ce que l'on souhaite calculer :

- le modèle direct,
- le modèle inverse.

Le modèle direct prend en entrée les couples appliqués aux articulations (on omet pour l'instant les forces externes appliquées sur le robot qui se rajoutent aux couples par une simple projection) et calcule les accélérations articulaires. Il est généralement écrit sous la forme lagrangienne suivante :

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}(\mathbf{q}) \left(\mathbf{\Gamma}(\mathbf{q}) - \mathbf{b}\left(\mathbf{q}, \dot{\mathbf{q}}\right) - \mathbf{g}(\mathbf{q}) \right) \tag{2.2}$$

où $\mathbf{M}(\mathbf{q})$ représente la matrice d'inertie globale du système, $\mathbf{\Gamma}(\mathbf{q})$ est le couple donné en entrée à chaque articulation, $\mathbf{b}\left(\mathbf{q}, \dot{\mathbf{q}}\right)$ représente les forces centrifuges et celles dûes aux effets de Coriolis et enfin $\mathbf{g}$ est l'effort dû à la gravité.

A l'inverse, le modèle inverse calcule les couples à appliquer aux articulations connaissant les accélérations articulaires. Il s'écrit donc :

$$\mathbf{M}(\mathbf{q})\ddot{\mathbf{q}} + \mathbf{b}\left(\mathbf{q}, \dot{\mathbf{q}}\right) + \mathbf{g}(\mathbf{q}) = \mathbf{\Gamma}(\mathbf{q}) \tag{2.3}$$

Dans cette thèse, nous ne considèrerons que le modèle direct car ce qui nous intéresse est de suivre l'évolution de la configuration du robot dans le temps. Il existe alors plusieurs formulations pour le calculer, en particulier les formulations de Newton-Euler, Lagrange ou encore Hamilton (il y en a d'autres). Nous présenterons les méthodes de Newton-Euler et Lagrange mais nous n'utilisons que la méthode de Newton-Euler qui est très utilisée en simulation car cette formulation se prête bien à une écriture algorithmique aisée des équations qui permet une programmation facile sur un ordinateur. L'avantage de cette méthode surtout réside dans sa simplicité et sa rapidité d'implémentation. La méthode de Lagrange, plus analytique et plus compacte, est beaucoup plus complexe à mettre en œuvre. Par ailleurs, nous ne traitons ici que le cas, simple, des chaînes cinématiques ouvertes. Nous n'aurons à traiter en effet que des systèmes poly-articulés avec une structure arborescente ouverte. Les chaînes cinématiques fermées sont plus complexes car les variables articulaires ne sont pas indépendantes à cause des contraintes de fermeture. Le lecteur pourra se référer à [Featherstone, 1987] pour plus de détails. Quoiqu'il soit, notre contribution à ce volet du calcul est minime tant le domaine est bien fourni en robotique. En réalité tous les algorithmes de calcul de la dynamique des corps poly-articulés se valent ; le problème n'est pas tant la forme d'où dérive le modèle dynamique, mais sa capacité à être facilement programmable et à être scindé en processus pouvant s'exécuter en parallèle.

2.2.2 Méthode de Newton-Euler

2.2.2.1 Algorithme de base

Les équations de Newton-Euler pour un système s'écrivent pour chacun des corps qui le compose sur la base des deux équations suivantes :

$$\sum \mathbf{f} = m\mathbf{a} \tag{2.4}$$

$$\sum \mathbf{\Gamma} = \mathbf{I}\dot{\omega} + \omega \wedge \mathbf{I}\omega \tag{2.5}$$

où m est la masse du corps rigide en question, $\mathbf{a}$ est l'accélération du centre de gravité, $\mathbf{I}$ est la matrice d'inertie ramenée au centre de gravité et ω est la vitesse angulaire. Nous pouvons réécrire ces équations à l'origine du repère de référence sous forme matricielle, voir par exemple dans [Kajita *et al.*, 2005] :

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{\Gamma} \end{bmatrix} = \underline{\mathbf{I}} \begin{bmatrix} \mathbf{a}_O \\ \dot{\omega} \end{bmatrix} + \begin{bmatrix} \mathbf{v}_O \\ \omega \end{bmatrix} \underline{\mathbf{I}} \begin{bmatrix} \mathbf{v}_O \\ \omega \end{bmatrix} = \underline{\mathbf{I}}\dot{\varphi} + \underline{\mathbf{b}} \tag{2.6}$$

avec

$$\underline{\mathbf{I}} = \begin{bmatrix} m\mathbf{I}_{3\times 3} & m\mathbf{a}^{\times T} \\ m\mathbf{a}^{\times} & m\mathbf{a}^{\times}\mathbf{a}^{\times T} + \mathbf{I} \end{bmatrix}, \ \mathbf{a}^{\times} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix}, \ \underline{\mathbf{b}} = \varphi \wedge \underline{\mathbf{I}}\varphi \tag{2.7}$$

L'équation 2.6 est la notation spatiale des équations de Newton-Euler.

Lorsqu'on considère un système poly-articulé, il est nécessaire de prendre en compte les effets de tout le système sur le corps que l'on traite à un moment donné du parcours : en particulier les vitesses et les accélérations cartésiennes du corps en question vont dépendre de celles du corps précédent. Par exemple dans le cas de deux corps articulés entre eux, les vitesses des deux corps sont liées par :

$$\varphi_2 = \varphi_1 + \begin{bmatrix} \mathbf{d}_2 \wedge \alpha_2 \\ \alpha_2 \end{bmatrix} \dot{q}_2 \tag{2.8}$$

où $\mathbf{d}_2$ représente la position du corps 2 par rapport au repère de référence et α_2 est le vecteur unitaire porté par l'axe de rotation du corps 2 (voir figure 2.10).

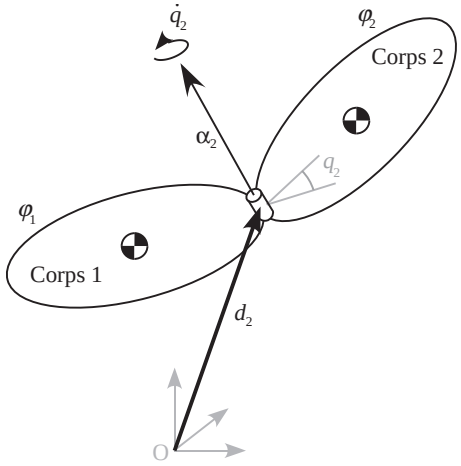


figure 2.10: *Un système de deux corps articulés entre eux.*

En dérivant cette expression, en utilisant la relation liant le couple aux efforts ($\mathbf{\Gamma} = \mathbf{J}^T\mathbf{f}$ avec $\mathbf{J}$ la Jacobienne au point d'application de la force $\mathbf{f}$) et avec l'équation 2.6, on obtient l'accélération articulaire du corps 2.

Pour étendre cet exemple à un nombre n de corps, il existe plusieurs algorithmes récursifs [Walker et Orin, 1982, Featherstone, 1987, Mirtich, 1996, Featherstone et Orin, 2000, Featherstone, 2007, Siciliano et Khatib, 2008]. On ne présente ici que deux d'entre eux, les plus

utilisés en simulation.

2.2.2.2 Algorithme de composition des corps rigides ou *Composite Rigid-Body Algorithm*

Cet algorithme, initialement proposé par Walker et Orin [Walker et Orin, 1982], est souvent utilisé comme une partie d'un algorithme de calcul du modèle dynamique et permet surtout de calculer la matrice d'inertie $\mathbf{M}$ de manière récursive. Le calcul de la matrice $\mathbf{M}$ est de complexité quadratique $\mathcal{O}(n^2)$, mais l'algorithme général (de base) est en complexité cubique $\mathcal{O}(n^3)$.

On peut montrer que la matrice $\mathbf{M}$ peut s'écrire sous la forme générale :

$$M_{ij} = \begin{cases} \mathbf{J}_i^T \hat{\mathbf{I}}_i \mathbf{J}_j & \text{if } i \in s(j) \\ \mathbf{J}_i^T \hat{\mathbf{I}}_j \mathbf{J}_j & \text{if } j \in s(i) \\ 0 & \text{sinon} \end{cases} \tag{2.9}$$

où $s(i)$ est l'ensemble des corps compris entre la base et le corps i et :

$$\hat{\mathbf{I}}_i = \sum_{j \in s(i)} \underline{\mathbf{I}}_j \tag{2.10}$$

est l'inertie de l'ensemble des corps appartenant à $s(i)$. Connaissant cette matrice $\mathbf{M}$ et les autres termes de l'équation 2.2 ($\mathbf{\Gamma}(\mathbf{q}) - \mathbf{b}(\mathbf{q}, \dot{\mathbf{q}}) - \mathbf{g}(\mathbf{q})$), on peut déterminer les accélérations articulaires en utilisant une factorisation de Cholesky.

2.2.2.3 Algorithme des corps articulés ou *Articulated-Body Algorithm*

Cet algorithme a été proposé par Featherstone [Featherstone, 1987, Mirtich, 1996] et calcule le modèle dynamique en trois récursions :

- calcul des paramètres géométriques et cinématiques,
- calcul des inerties dans le repère de référence et des forces extérieures connues,
- calcul des accélérations articulaires.

Cet algorithme fait intervenir deux grandeurs, l'inertie du corps articulé qui est l'inertie que le corps possède lorsqu'il fait partie d'un système de corps rigides, et la force qui donne une accélération nulle au corps considéré et qui contient les effets de Coriolis et ceux des forces extérieures connues. Ces deux grandeurs, respectivement notées $\overline{\mathbf{I}}$ et $\overline{\mathbf{b}}$ sont reliées par une relation linéaire :

$$\begin{bmatrix} \mathbf{f} \\ \mathbf{\Gamma} \end{bmatrix} = \overline{\mathbf{I}} \varphi + \overline{\mathbf{b}} \tag{2.11}$$

Contrairement à l'équation 2.6, $\overline{\mathbf{I}}$ et $\overline{\mathbf{b}}$ ne sont pas intrinsèques au corps considéré i, mais prennent en compte l'effet des corps suivants et s'expriment sous la forme récursive suivante :

$$\overline{\mathbf{I}}_i = \underline{\mathbf{I}}_i + \sum_{j \in \mu_i} (\overline{\mathbf{I}}_j - \overline{\mathbf{I}}_j \mathbf{J}_j (\mathbf{J}_j^T \overline{\mathbf{I}}_j \mathbf{J}_j)^{-1} \mathbf{J}_j^T \overline{\mathbf{I}}_j) \tag{2.12}$$

$$\overline{\mathbf{b}}_i = \underline{\mathbf{b}}_i + \sum_{j \in \mu_i} (\hat{\mathbf{b}}_j + \overline{\mathbf{I}}_j \mathbf{J}_j (\mathbf{J}_j^T \overline{\mathbf{I}}_j \mathbf{J}_j)^{-1} (\mathbf{\Gamma}_i - \mathbf{J}_j^T \hat{\mathbf{b}}_j)) \tag{2.13}$$

avec

$$\hat{\mathbf{b}}_j = \mathbf{b}_j + \overline{\mathbf{I}}_j \dot{\mathbf{J}}_j \dot{q}_j \tag{2.14}$$

et μ_i est l'ensemble des fils du corps i. Finalement, en notant λ_i le parent du corps i, l'accélération articulaire est :

$$\ddot{q}_i = (\mathbf{J}_i \overline{\mathbf{I}}_i \mathbf{J}_i)^{-1} (\mathbf{\Gamma}_i - \mathbf{J}_i^T (\overline{\mathbf{I}}_i \dot{\varphi}_{\lambda_i} + \overline{\mathbf{b}}_i)) \tag{2.15}$$

Cet algorithme est de complexité linéaire $\mathcal{O}(n)$, avec n le nombre de corps. L'avantage de cet algorithme réside dans le calcul de chaque accélération articulaire pour un couple articulaire donné sans avoir à calculer explicitement chaque terme de l'équation 2.2 et en particulier la matrice $\mathbf{M}^{-1}$.

2.2.3 Méthode de Lagrange

La méthode de Lagrange est plus complexe à dériver et à implémenter ; de forme compacte, elle se prête plutôt bien à l'automatique pour la synthèse de lois de commande. On peut bien sûr programmer les dérivations des équations des énergies cinétique et potentielle, comme cela est fait dans HuMAnS [7]. Nous en présentons ici les grandes lignes.

Contrairement à la formulation de Newton-Euler qui lie les efforts et les moments pour chaque corps d'un système, la formulation de Lagrange décrit la dynamique de tout le système en termes de travail et d'énergie. Les contraintes dynamiques qui apparaissent dans la méthode de Newton-Euler disparaissent ici.

Les équations de la dynamique de Lagrange s'écrivent :

$$\frac{d}{dt} \frac{\partial \mathbf{L}}{\partial \dot{\mathbf{q}}} - \frac{\partial \mathbf{L}}{\partial \mathbf{q}} = \mathbf{Q}, \tag{2.16}$$

où $\mathbf{L}$ est le lagrangien défini par

$$\mathbf{L} = \mathbf{T} - \mathbf{E}_P - \mathbf{V}, \tag{2.17}$$

avec $\mathbf{T}$ l'énergie cinétique de tout le système, $\mathbf{E}_P$ l'énergie potentielle et $\mathbf{V}$ l'énergie potentielle associée aux champs de forces extérieures connues, et $\mathbf{Q}$ est l'ensemble des travaux virtuels engendrés par des forces non conservatives. Dans le cas où on n'a pas de forces non conservatives, $\mathbf{Q} = 0$. Le lecteur pourra se référer par exemple à [Hale, 2004] pour plus de détails.

7. `http://bipop.inrialpes.fr/software/humans/index.html`

2.2.4 Calcul parallèle

Lorsque le système est constitué d'un grand nombre de corps (plusieurs milliers), le calcul dynamique peut s'avérer coûteux en temps. Typiquement, pour calculer la dynamique d'une chaîne cinématique, il est possible de calculer séparément et en même temps la dynamique de sous-ensembles de la chaîne puis d'assembler ces sous-ensembles. Il s'agit donc de paralléliser le calcul. Plusieurs travaux ont été menés dans ce domaine, avec des applications dans l'étude du corps humain ou de simulations de molécules [Anderson et Duan, 2000, Fijany *et al.*, 1995, Featherstone, 1999]. En particulier, Featherstone a proposé un algorithme dans lequel la dynamique de chaque sous-corps poly-articulé est calculée, puis l'assemblage des sous-corps se fait en considérant les contraintes cinématiques et celles en forces entre les sous-corps (voir figure 2.11).

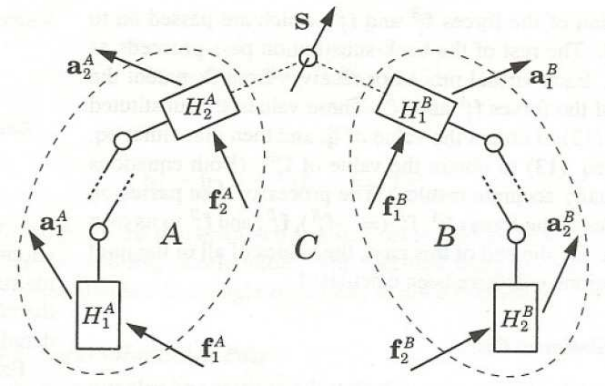


figure 2.11: *Assemblage de deux sous-corps poly-articulés (tiré de [Featherstone, 1999]).*

Connaissant la manière dont les corps sont assemblés entre eux, cet algorithme se déroule en quatre récursions :

- les deux premières servent à calculer les positions et vitesses des corps à partir des positions et vitesses articulaires,
- la troisième récursion est la boucle principale dans laquelle la dynamique de chaque sous-ensemble de corps poly-articulés est calculée,
- la dernière récursion assemble les sous-ensembles et calcule les accélérations articulaires des liaisons entre les sous-ensembles.

Cet algorithme a une complexité en $\mathcal{O}(\log(n))$ et est particulièrement efficace si le nombre de processeurs est élevé. L'inconvénient de cette méthode est une perte de précision numérique quand le nombre de corps augmente.

Yamane et Nakamura [Yamane et Nakamura, 2002] ont proposé un autre algorithme dit d'assemblage-désassemblage ("Assembly-Disassembly Algorithm" (ADA)) composé de deux étapes :

- l'assemblage : les liaisons sont ajoutées une à une et les inerties des ensembles de corps formés sont calculées,
- le désassemblage : partant de la chaîne poly-articulée complète, les liaisons sont enlevées une à une et les accélérations articulaires de chaque liaison sont calculées.

Les méthodes proposées ont cependant un inconvénient : le choix de la décomposition des corps

en sous-ensembles est arbitraire, autrement dit la répartition des calculs à faire en parallèle n'est *a priori* pas homogène. Yamane et Nakamura [Yamane et Nakamura, 2007] ont alors proposé une méthode automatique pour déterminer de quelle manière doit être décomposé un corps poly-articulé pour minimiser le temps de calcul. Ils ont montré qu'ils pouvaient réduire les temps de calcul de 35%.

Dans notre cas, comme nous nous sommes restreints à travailler qu'avec un seul processeur, l'utilisation de ces méthodes n'apportera aucune amélioration intéressante.

2.3 Prise en compte du contact

La simulation dynamique d'un système complexe comme les robots humanoïdes fait intervenir dans le calcul principalement deux inconnues : le calcul des forces extérieures et des accélérations. En reprenant l'équation 2.2 et en y rajoutant les forces extérieures on a alors :

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}(\mathbf{q}) \left(\mathbf{\Gamma}(\mathbf{q}) - \mathbf{b}\left(\mathbf{q}, \dot{\mathbf{q}}\right) - \mathbf{g}(\mathbf{q}) \right) + \mathbf{M}^{-1}(\mathbf{q}) \mathbf{J}^{T}(\mathbf{q}) \mathbf{f} \tag{2.18}$$

où $\mathbf{f}$ représente les efforts extérieurs et $\mathbf{J}$ la Jacobienne aux points d'application des efforts. En simulation, ceux-ci peuvent être connus, s'il s'agit par exemple de perturbations intentionnelles et mesurables exercées sur le robot (via une interface haptique par exemple), ou inconnus, s'il s'agit par exemple des forces de contact. Sans inclure le calcul des forces de contact, la dynamique est assez simple à calculer car, si on considère comme connus les paramètres extérieurs comme les couples qui peuvent être mesurés ou bien une force exercée par un utilisateur, l'équation 2.18 ne possède qu'une seule inconnue qui est l'accélération articulaire $\ddot{\mathbf{q}}$ et qui peut de fait être résolue rapidement en utilisant les algorithmes cités précédemment.

Le cas plus délicat est évidemment celui où les forces $\mathbf{f}$ ne sont pas connues, en particulier les forces d'interaction. On se retrouve alors avec une seule équation à deux inconnues. C'est la partie qui fait l'objet d'un plus grand nombre de travaux car elle n'est pas triviale. Plusieurs méthodes ont été proposées pour résoudre ce problème : les méthodes par pénalités, les méthodes par contraintes, les méthodes à base d'impulsions. Dans cette partie, nous ne considérerons pas le frottement. Celui-ci sera traité dans la partie suivante.

2.3.1 Méthode par pénalités

Cette méthode est très répandue et utilisée dans la plupart des simulateurs [Kanehiro *et al.*, 2004, Hwang *et al.*, 2003, Hasegawa et Sato, 2004, Yamane et Nakamura, 2006, Turk et Wyeth, 2006]. Le principe est de ne calculer une force que lorsqu'il y a pénétration. Quand un point de contact pénètre, la distance de pénétration et sa variation sont mesurées, et cette pénétration est ensuite pénalisée par une force que l'on modélise généralement par un ressort-amortisseur virtuel. Si le contact se fait de manière unilatérale, cette force de contact, selon la normale au point de contact, est écrite généralement

sous la forme simplifiée suivante :

$$f_n = \begin{cases} K\delta x + C\delta\dot{x} & \text{si } \delta x \geq 0 \\ 0 & \text{sinon} \end{cases} \tag{2.19}$$

où K et C sont la raideur du ressort et le facteur d'amortissement respectivement, δx est la distance d'interpénétration et $\delta\dot{x}$ est sa variation (voir figure 2.12). D'autres variantes, plus subtiles pour pénaliser la violation de contraintes de non pénétration existent.

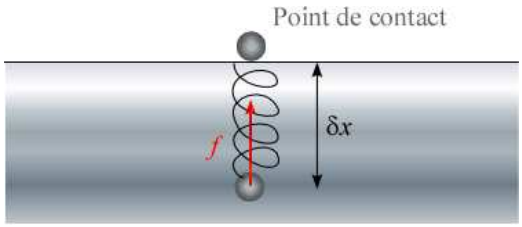


figure 2.12: *Méthode par pénalités (point de contact soumis à une réaction de type ressort). Le point pénètre dans l'objet et alors une force est appliquée.*

L'avantage de cette méthode réside dans sa simplicité d'implémentation et dans sa rapidité. D'autre part, comme les forces calculées sont fonction de l'interpénétration, cette approche, locale, donne relativement une bonne estimation de la répartition des forces de réaction en quasi-statique, ce qui n'est pas forcément le cas avec les méthodes par contraintes qui sont des méthodes globales (voir partie 2.3.2). Par ailleurs, elle simplifie le problème. En effet, le calcul de cette force est donné localement et simplement par l'équation 2.19, autrement dit, dans l'équation de départ 2.18, la force $\mathbf{f}$ devient connue. On se retrouve alors dans le cas où l'équation 2.18 ne possède qu'une seule inconnue, l'accélération articulaire.

Cependant cette méthode possède des inconvénients majeurs. En effet, l'équation 2.19 fait intervenir deux paramètres, *a priori* inconnus, K et C. Ces paramètres, à régler, sont souvent des facteurs d'instabilités numériques, en particulier, un choix élevé de la raideur K peut conduire à des instabilités dans la simulation, suivant le schéma d'intégration numérique choisi. Généralement, la détermination de K et C n'est pas mentionnée dans la littérature car il n'existe pas de formules permettant de les calculer en fonction par exemple des propriétés des matériaux, et le réglage de ces paramètres est fait expérimentalement pour un cas bien particulier de tâche et se trouvent n'être plus valables dans d'autres situations [Kanehiro *et al.*, 2004]. Un autre désavantage de cette méthode est sa justification physique car elle est basée sur les pénétrations. Par ailleurs, la prise en compte du frottement (voir partie 2.4.1) n'est pas simple. En effet, l'estimation des pénétrations dans le plan tangent n'est pas triviale.

2.3.2 Méthode par contraintes

On a vu que dans les méthodes par pénalités, les corps doivent s'interpénétrer pour qu'il y ait une force de réaction non nulle. Dans les méthodes par contraintes, les contraintes de non-pénétration

sont explicitement intégrées aux équations de la dynamique, ce qui accroît la précision des calculs [Baraff, 1994]. Elles sont de plus en plus répandues, en particulier dans la communauté des jeux vidéos car elles permettent des rendus graphiques plus réalistes [Kokkevis, 2004].

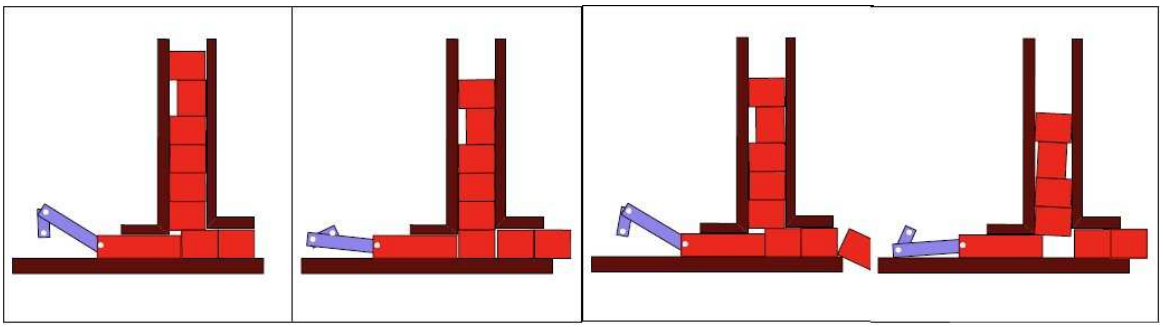


figure 2.13: *Simulation de remplissage de blocs utilisant les méthodes par contraintes (tiré de [Baraff, 1994]).*

Nous partons de l'équation de la dynamique de départ 2.18 :

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}\mathbf{J}^T\mathbf{f} + \mathbf{M}^{-1}(\mathbf{\Gamma} - \mathbf{b} - \mathbf{g}) \tag{2.20}$$

En passant cette équation dans l'espace cartésien où l'on définit chaque contact par une paire de points les plus proches sur chaque objet (en présence d'une infinité de points on suppose l'existence d'un sous-ensemble dénombrable (c'est-à-dire fini) de paires représentatives). On définit aussi pour chaque paire de points une normale commune sur laquelle on projettera les accélérations relatives $\mathbf{a}$ et la force de contact $\mathbf{f}$. On commence par établir les vitesses relatives ($\mathbf{v} = \mathbf{J}\dot{\mathbf{q}}$) puis leurs dérivées ($\mathbf{a} = \mathbf{J}\ddot{\mathbf{q}} + \dot{\mathbf{J}}\dot{\mathbf{q}}$), nous obtenons donc :

$$\mathbf{a} = \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T\mathbf{f} + \mathbf{J}\mathbf{M}^{-1}(\mathbf{\Gamma} - \mathbf{b} - \mathbf{g}) + \dot{\mathbf{J}}\dot{\mathbf{q}} \tag{2.21}$$

que l'on peut réécrire :

$$\mathbf{a} = \mathbf{\Lambda}\mathbf{f} + \mathbf{c} \tag{2.22}$$

avec $\mathbf{\Lambda} = \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$ et $\mathbf{c} = \mathbf{J}\mathbf{M}^{-1}(\mathbf{\Gamma}-\mathbf{b}-\mathbf{g})+\dot{\mathbf{J}}\dot{\mathbf{q}}$. La matrice $\mathbf{\Lambda}$, carrée de taille m, m étant le nombre de points de contact, est la matrice d'inertie projetée au niveau des contacts et est aussi appelée opérateur de Delassus [Delassus, 1923], le vecteur $\mathbf{c}$ est la dynamique sans contraintes du système projetée au niveau des contacts. A cette équation nous imposons plusieurs contraintes :

- les accélérations normales relatives des points de contact empêchent les corps de s'interpénétrer : $a_n \geq 0$,
- les forces normales de contact éloignent les corps les uns des autres : $f_n \geq 0$,
- ou bien le contact disparaît (la force de contact est donc nulle) et l'accélération relative du point de contact n'est pas nulle (elle est positive d'après la première contrainte), ou bien il y a contact (la force de contact est donc positive d'après la deuxième contrainte) et alors l'accélération relative est nulle.

Ces trois contraintes forment un problème de complémentarité linéaire (LCP) [Murty et Yu, 1999],

c'est-à-dire :

$$\begin{cases} \mathbf{a} = \mathbf{\Lambda f} + \mathbf{c} \\ \mathbf{\Lambda f} + \mathbf{c} \geq 0 \\ \mathbf{f} \geq 0 \\ \mathbf{f}^T(\mathbf{\Lambda f} + \mathbf{c}) = 0 \end{cases} \tag{2.23}$$

ou encore sous la forme compacte :

$$0 \leq \mathbf{\Lambda f} + \mathbf{c} \perp \mathbf{f} \geq 0 \tag{2.24}$$

En l'absence de frottement aux points de contact, cette équation est couramment appelée condition de Signorini. L'inclusion des frottements est traité dans la partie 2.4. Nous pouvons remarquer que les accélérations relatives sont une fonction linéaire des forces de contact, ce qui constitue une formulation élégante pour résoudre les problèmes de contact. De plus la convergence vers une solution est assurée par différentes méthodes de résolution directe comme la méthode de Lemke qui utilise des pivots [Lloyd, 2005, Anitescu et Potra, 2002]. Toutes les contraintes en chaque point de contact sont regroupées dans la matrice $\mathbf{\Lambda}$, permettant ainsi une gestion globale des contacts avec la prise en compte de leurs influences mutuelles, ce qui n'est pas le cas avec les méthodes par pénalité (locales).

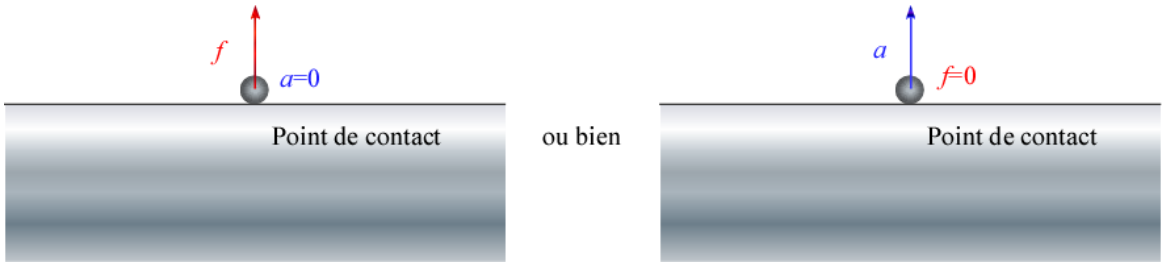


figure 2.14: *Méthode par contraintes. S'il y a contact, la force normale est positive, si le contact décolle, elle devient nulle.*

L'équation 2.23 peut être représentée graphiquement (voir figure 2.15) ce qui rend plus facilement compte de la condition de complémentarité qui est non linéaire. Il est possible de régulariser cette loi de contact en autorisant des petites interpénétrations qui sont justifiables par des petites déformations de la surface [Anitescu et Potra, 2002].

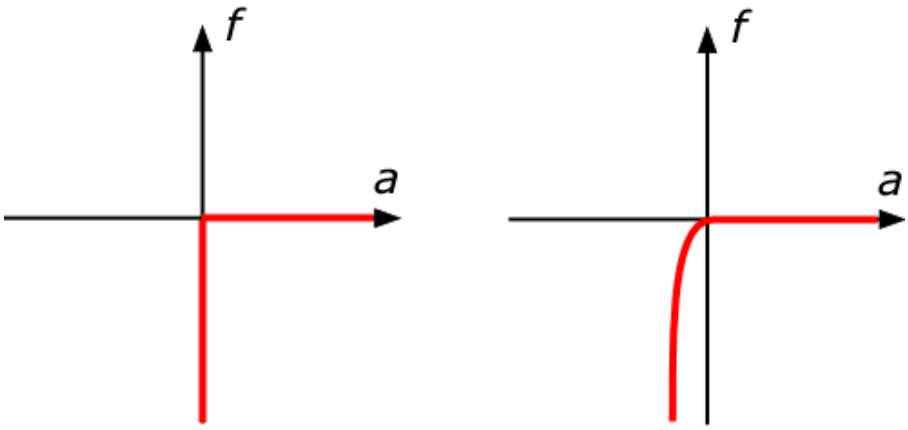


figure 2.15: *Loi de contact (à gauche, la condition de complémentarité énoncée dans l'équation 2.23 est non-linéaire. A droite, une régularisation de la loi).*

Ruspini et Khatib [Ruspini et Khatib, 1999, Ruspini et Khatib, 2000] ont étendu ce formalisme pour résoudre le contact et l'impact en utilisant la formulation spatiale de la dynamique multi-corps dans l'espace opérationnel, introduite plus tôt par Khatib [Khatib, 1987] et qui est en fait une formulation employant des coordonnées généralisées. Nous développerons ce point dans le chapitre suivant.

Cette méthode apparaît idéale pour traiter le contact de manière physiquement plausible et robuste pour la simulation. Même dans des situations complexes, la stabilité est assurée. Cette méthode est cependant bien adaptée pour des systèmes de petites tailles. Les autres inconvénients de cette approche sont : d'une part la résolution du problème ayant plusieurs contacts en hyperstatisme ne garantit pas l'unicité des solutions. Ainsi, pour une situation donnée, il se peut qu'il y ait une infinité de solutions, typiquement, une table à quatre pieds posée à même un sol plat. En supposant que la table est parfaitement équilibrée et que la masse m_{table} est également répartie dans chaque pied, la solution réelle donne une force normale de contact à chaque pied égale à $\frac{m_{\text{table}}g}{4}$. La formulation LCP ne donnera qu'une seule solution parmi cette infinité de solutions qui respectera la dynamique mais qui ne sera pas nécessairement la solution réelle, en l'occurence en prenant une table à quatre pieds de masse $m_{\text{table}} = 1\text{kg}$, nous avons obtenu en simulation la répartition $(2,275\text{N}; 2,625\text{N}; 2,275\text{N}; 2,625\text{N})$ [8]. Ensuite il s'en suit un temps de calcul beaucoup plus élevé du fait de la construction de la matrice $\mathbf{\Lambda}$, ce qui peut poser problème dans le cas de simulations interactives. Par ailleurs, cette matrice $\mathbf{\Lambda}$ doit être bien conditionnée pour que le problème soit bien formulé, en particulier elle doit être définie positive. Enfin la continuité des accélérations des corps n'est préservée qu'à condition de connaître l'état des contacts au pas précédent.

2.3.3 Méthode impulsionnelle

Dans la méthode par impulsion [Mirtich, 1996], dès qu'il y a contact, une impulsion est appliquée instantanément. De fait, il n'y a pas de contraintes d'interpénétration à ajouter. Comme son nom l'indique, cette méthode ne calcule pas des forces mais des impulsions, même si celles-ci peuvent être

8. Nous aurions très bien pu obtenir une autre répartition qui respecte l'équilibre statique.

considérées comme des forces agissant pendant un temps infinitésimal. Ainsi, tout contact, qu'il soit un impact ou un contact glissant ou en adhérence, est considéré comme une collision. Typiquement un contact continu est considéré comme une série d'impacts. La méthode se décompose alors en trois étapes :

- on calcule le temps maximal pendant lequel il ne peut pas y avoir de collision. Dès qu'une paire de points de contact apparaît, les points sont considérés comme points critiques,
- la dynamique de tous les corps est calculée,
- on applique une impulsion aux points critiques. On utilise alors un coefficient de restitution e compris entre 0 et 1 qui traduit l'élasticité de l'impact et qui lient les vitesses pré-impact $\mathbf{v}^-$ et post-impact $\mathbf{v}^+$ de manière linéaire par la loi de Newton :

$$\mathbf{v}^+ = -e\mathbf{v}^- \tag{2.25}$$

L'avantage de cette méthode est que son implémentation est simple. L'inconvénient est la stabilité, en particulier, dans le cas de contacts statiques, lorsque les points se mettent à bouger, la simulation peut devenir instable et le système diverger. Cette méthode est particulièrement inappropriée pour des systèmes robotiques poly-articulés.

2.3.4 Résumé

Les trois méthodes que nous avons présentées font apparaître les différences suivantes :

- la rapidité de calcul : dans le cas de méthodes par pénalités, la force normale de contact se calcule aisément par une relation simulant un ressord-amortisseur virtuel, tandis qu'avec les méthodes par contraintes, le calcul de la force de contact requiert d'abord le calcul de la matrice de Delassus qui peut être très coûteux dans le cas de nombreux points de contact, puis la résolution d'un problème de complémentarité linéaire par des méthodes appropriées exigeantes au niveau du conditionnement de la matrice de Delassus,
- la formulation du contact : comme indiqué juste avant, les méthodes par pénalités simulent un système de pénalisation, donc une dynamique virtuelle qui interfère avec la dynamique propre du système robotique à simuler, tandis que dans les méthodes par contraintes, les contraintes de contact sont explicitement intégrées dans les équations dynamiques. Les méthodes par impulsion, en revanche, proposent de formuler tout contact par une série d'impacts,
- la précision et la stabilité : les méthodes par pénalités sont très dépendantes des réglages des paramètres et des méthodes d'intégration numérique et donc la précision en est affectée. Les méthodes par contraintes sont beaucoup plus précises et plus stables grâce aux contraintes de non-pénétration mais sont plus lentes, sujettes aussi aux erreurs numériques et posent dans quelques cas, des problèmes d'unicité de la solution. Les méthodes par impulsion ne sont pas stables dans le cas où le contact est maintenu ou lorsque les points de contact évoluent en mode contraint.

2.4 Prise en compte du frottement

Jusqu'à présent, nous avons négligé dans la prise en compte du contact, un phénomène physique important qui continue à alimenter beaucoup de travaux en mécanique non régulière : les frottements. Cependant il est nécessaire de les prendre en compte afin d'obtenir une simulation réaliste. Notre but est maintenant de résoudre l'équation 2.18 dont l'accélération et la force de contact ne sont projetées que selon la normale définie au point de contact ; il s'agit maintenant de considérer le plan tangent avec une contrainte supplémentaire liée au frottement. Il existe différents types de frottement dont le frottement sec/dynamique et le frottement visqueux. Nous ne considérerons pas les frottements visqueux qui ne posent pas de problème particulier du fait qu'ils sont proportionnels à la vitesse relative de déplacement.

Différents modèles mathématiques de frottement sec ont été introduits [Olsson *et al.*, 1998] et en particulier, le plus connu et le plus utilisé est le modèle de Coulomb, qui stipule que la force de frottement en un point de contact doit être compris à l'intérieur d'un cône (appelé cône de Coulomb) dont le rayon dépend d'un coefficient intrinsèque au frottement appelé coefficient de frottement et dont la hauteur est égale à l'effort normal. Mathématiquement, ce modèle se traduit par :

$$\|\mathbf{f}_t\| \leq \mu \, |f_n| \,, \tag{2.26}$$

où $\mathbf{f}_t$ est le vecteur composant la force de frottement dans le plan tangent au contact, f_n est la composante normale et μ est le coefficient de frottement. On distingue alors deux cas (voir figure 2.16) :

- ou bien le point de contact adhère (on parle de frottements secs), c'est-à-dire sa vitesse tangentielle $\mathbf{v}_t$ est nulle et on a :

$$\|\mathbf{f}_t\| < \mu \, |f_n| \,, \tag{2.27}$$

- ou bien le point de contact glisse (on parle de frottements dynamiques), c'est-à-dire que sa vitesse tangentielle est non nulle, et la force de frottement s'oppose au mouvement dans la direction de celui-ci, de manière proportionnelle à la composante normale de la force de contact :

$$\|\mathbf{f}_t\| = -\mu \, |f_n| \, \frac{\mathbf{v}_t}{\|\mathbf{v}_t\|}, \tag{2.28}$$

Notons qu'en réalité, le coefficient de frottement μ, pour une paire de matériaux donnés, n'est pas le même dans le cas d'adhérence et de glissement. Un modèle plus fin comporterait donc une non-linéarité supplémentaire avec deux μ différents.

Nous voyons que la prise en compte des phénomènes de frottement introduit une non-linéarité supplémentaire dans la dynamique en raison de la forme de l'équation qui le régit. Par ailleurs, la direction de la composante tangentielle des frottements est déterminée selon la vitesse du mouvement (elle s'y oppose). Or la dynamique 2.18 nous donne des accélérations et comprend toute la difficulté à résoudre le problème : en bref, l'accélération n'est donnée qu'une fois les forces connues, et dans ce cas, pour connaître la direction de la force il faut intégrer au moins une fois l'accélération. Il est possible d'exprimer le problème de frottements en termes d'accélérations, comme l'a

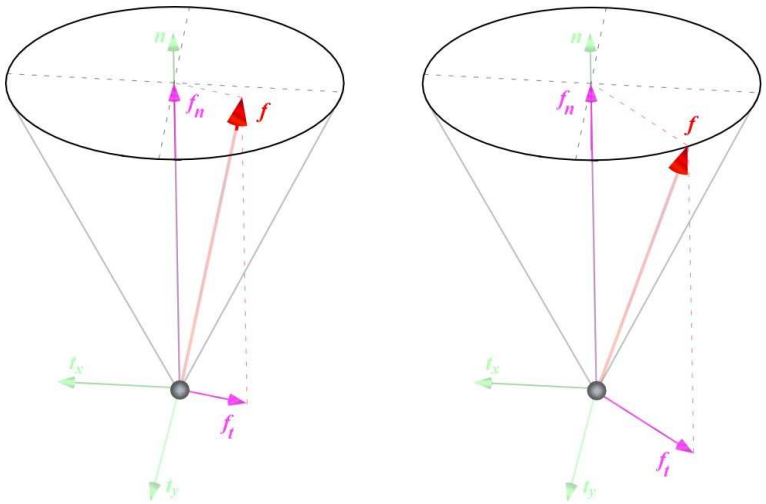


figure 2.16: *Le cône de frottement de Coulomb (à gauche : cas d'adhérence. A droite : cas de glissement).*

fait Baraff [Baraff, 1994], cependant il peut y avoir des cas où aucune solution n'existe pour les forces de frottement. Baraff applique alors des impulsions pour permettre des discontinuités de la vitesse. Le problème sous-jacent qui apparaît est le caractère fortement non-régulier de la dynamique [Brogliato *et al.*, 2002, Acary et Brogliato, 2008]. Simplement dit, si un point de contact passe du cas d'adhérence au cas de glissement, on ne connaît ni la force de frottement ni la direction du glissement tangent.

Dans la suite, nous reprenons les méthodes par pénalités et les méthodes par contraintes pour inclure le frottement.

2.4.1 Méthode par pénalités

Pour traiter les problèmes de frottements avec les méthodes par pénalités, il existe plusieurs approches, nous avons retenu les suivantes :

– on rajoute une fonction de poids w dans le cas du glissement [Yamane et Nakamura, 2006] :

$$\|\mathbf{f}_t\| = -\mu\,|f_n|\,\frac{w(\|\mathbf{v}_t\|)}{\|\mathbf{v}_t\|}\mathbf{v}_t \tag{2.29}$$

Cette fonction w est choisie de manière à être nulle quand la vitesse de glissement est nulle et convergeante vers 1 quand la vitesse de glissement augmente, et permet de réduire la discontinuité des vitesses dans l'intégration numérique qui apparaît lors d'une collision,

– on écrit les efforts tangentiels comme on le fait pour les efforts normaux, c'est-à-dire en intégrant virtuellement un système masse-ressort dans le plan tangent [Hwang *et al.*, 2003, Turk et Wyeth, 2006] :

$$\begin{cases} f_x = K_x\delta x + C_x\delta\dot{x} \\ f_y = K_y\delta y + C_y\delta\dot{y} \end{cases} \tag{2.30}$$

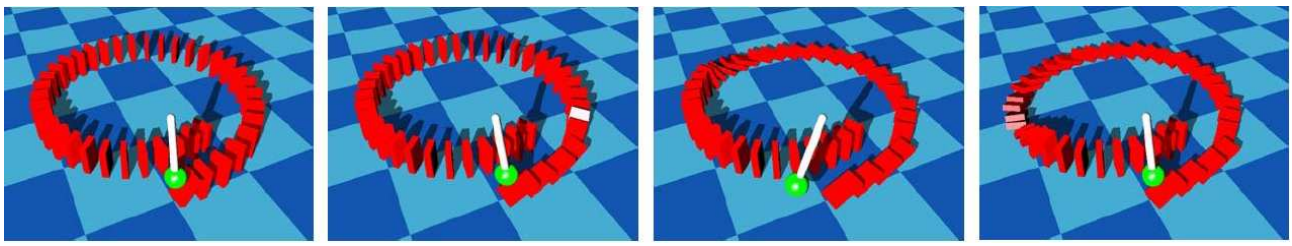


figure 2.17: *Simulation de dominos avec prise en compte du frottement par la méthode par pénalités (tiré de [Yamane et Nakamura, 2006]).*

Lorsqu'il y a glissement les forces sont calculées par

$$\begin{cases} f_x = -\mu f_z \frac{\delta x}{|\delta x|} \\ f_y = -\mu f_z \frac{\delta y}{|\delta y|} \end{cases} \tag{2.31}$$

Le problème ici est de quantifier la pénétration tangentielle.

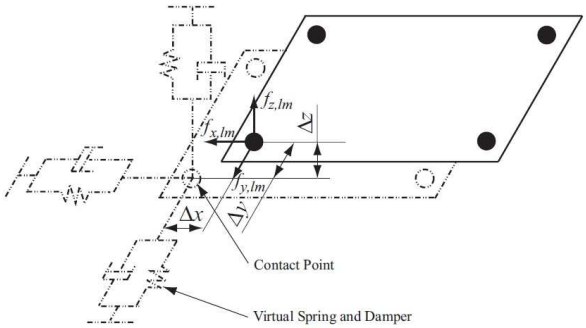


figure 2.18: *Prise en compte du frottement avec la méthode par pénalités (d'après [Hwang* et al.*, 2003]).*

Comme pour le cas sans frottement, cette méthode est très rapide, et de fait adaptée dans le cas de simulations temps-réel, mais le problème est encore une fois le choix des différents paramètres des ressorts et amortisseurs virtuels et la quantification de la pénétration tangentielle. De fait, pour éviter des grandes instabilités numériques, il est nécessaire de prendre un pas de temps très petit ce qui n'est pas adapté pour des simulations interactives.

2.4.2 Méthode par contraintes

Dans un cas simple uni-contact et en écrivant le problème en vitesse (en intégrant une fois l'équation dynamique), l'équation 2.23 se réécrit :

$$0 \leq \begin{bmatrix} \Lambda_{nn} & \Lambda_{nt_1} & \Lambda_{nt_2} \\ \Lambda_{t_1 n} & \Lambda_{t_1 t_1} & \Lambda_{t_1 t_2} \\ \Lambda_{t_2 n} & \Lambda_{t_2 t_1} & \Lambda_{t_2 t_2} \end{bmatrix} \begin{pmatrix} f_n \\ f_{t_1} \\ f_{t_2} \end{pmatrix} + \begin{pmatrix} v_n \\ v_{t_1} \\ v_{t_2} \end{pmatrix} \perp \begin{pmatrix} f_n \\ f_{t_1} \\ f_{t_2} \end{pmatrix} \geq 0 \tag{2.32}$$

2.4.2.1 Formulation LCP

La non-linéarité de la loi de frottement est un problème dans le cas où on travaille avec une formulation de type LCP. Pour pouvoir garder ce formalisme, une des méthodes employées est de discrétiser le cône de frottement en facettes (voir figure 2.19) [Stewart et Trinkle, 1996, Anitescu et Potra, 1997, Trinkle *et al.*, 1997, Sauer et Schömer, 1998]. Le cône de frottement est décomposé en facettes, c'est-à-dire en autant de plans que souhaité, qui s'opposent ou se complètent, et le cône peut être vu comme une pyramide à k faces. Ainsi si le cône possède k facettes, le nouveau système à résoudre pour un point de contact sera de la forme [Duriez, 2004] :

$$\begin{cases} 0 \leq v_n \perp f_n \geq 0 \\ 0 \leq \lambda + v_{t_1} \perp f_{t_1} \geq 0 \\ \vdots \\ 0 \leq \lambda + v_{t_k} \perp f_{t_k} \geq 0 \\ 0 \leq \mu f_n - \sum_{i=1}^{k} f_{t_i} \perp \lambda \geq 0 \end{cases} \tag{2.33}$$

avec λ un multiplicateur qui mesure le déplacement tangent. La première équation est la condition de Signorini exprimée en vitesse, les autres équations indiquent qu'il n'y a de force tangentielle possible que dans la direction opposée au mouvement en cas de glissement, la dernière équation reprend la loi de Coulomb. Ce système peut s'écrire sous la forme compacte suivante comme dans [Stewart et Trinkle, 1996, Anitescu et Potra, 1997, Stewart et Trinkle, 2000, Potra *et al.*, 2006] :

$$\begin{cases} 0 \leq \lambda \mathbf{u} + \mathbf{D}^T \mathbf{v} \perp \mathbf{f}_t \geq 0 \\ 0 \leq \mu f_n - \mathbf{u}^T \mathbf{f}_t \perp \lambda \geq 0 \end{cases} \tag{2.34}$$

où $\mathbf{f}_t = (f_{t_1}, \cdots, f_{t_k})^T$, $\mathbf{u} = (1, \cdots, 1)^T$, $\mathbf{D} = (d_1, \cdots, d_k)$ est une matrice de vecteurs unitaires de direction du cône discrétisé tels que $d_i = -d_j$. Alors l'opérateur de Delassus peut se réécrire :

$$0 \leq \begin{bmatrix} \Lambda_{nn} & \Lambda_{nt_1} & \dots & \Lambda_{nt_k} & 0 \\ \Lambda_{t_1 n} & \Lambda_{t_1 t_1} & \dots & \Lambda_{t_1 t_k} & 1 \\ \vdots & \vdots & \ddots & \vdots & \vdots \\ \Lambda_{t_k n} & \Lambda_{t_k t_1} & \dots & \Lambda_{t_k t_k} & 1 \\ \mu & -1 & \dots & -1 & 0 \end{bmatrix} \begin{pmatrix} f_n \\ f_{t_1} \\ \vdots \\ f_{t_k} \\ \lambda \end{pmatrix} + \begin{pmatrix} v_n \\ v_{t_1} \\ \vdots \\ v_{t_k} \\ 0 \end{pmatrix} \perp \begin{pmatrix} f_n \\ f_{t_1} \\ \vdots \\ f_{t_k} \\ \lambda \end{pmatrix} \geq 0 \tag{2.35}$$

Nous voyons bien les inconvénients qui en résultent, à savoir des contraintes additionnelles pour chaque point de contact (autant que de facettes) qui augmentent donc la taille du LCP de manière substantielle, et une relative perte de précision si les cônes ne sont pas suffisamment discrétisés. Le manque de robustesse de la simulation et l'augmentation des temps de calculs en sont les conséquences, en particulier dans la construction des matrices car leur taille dépend de la discrétisation $(km + 2m)$, ce qui en fait une méthode qui peut s'avérer inefficace pour des simulations interactives. Par ailleurs, l'opérateur de Delassus n'est plus symétrique et peut être mal conditionné, ce qui peut

poser problème pour des solveurs de LCP basés sur des méthodes par pivot comme indiqué dans la partie 2.3.2 [Renouf *et al.*, 2005].

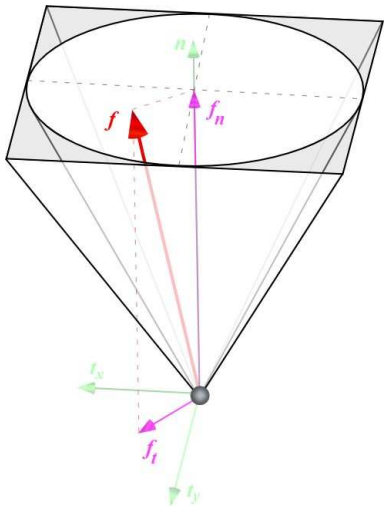


figure 2.19: *Discrétisation du cône de frottement. Ici nous avons pris un cône à quatre facettes, autrement dit une pyramide, extérieur au cône original. Il est possible de prendre le cône discrétisé à l'intérieur du cône original.*

Kaufman *et al.* [Kaufman *et al.*, 2005, Kaufman *et al.*, 2008] proposent une autre méthode qui n'utilise pas des méthodes classiques de résolution de LCP mais qui résoud deux problèmes d'optimisation :

$$\mathbf{f}_t^{t+\delta t} = \min_{\mathbf{f}_t}(\mathbf{f}_t^T \mathbf{D}\dot{\mathbf{q}}^{t+\delta t} \ : \ \mathbf{u}^T\mathbf{f}_t \leq \mu f_n^{t+\delta t}, \mathbf{f}_t \geq 0) \tag{2.36}$$

$$\dot{\mathbf{q}}_t^{t+\delta t} = \min_{\mathbf{y}} \left(\frac{1}{2}\mathbf{y}^T\mathbf{M}\mathbf{y} - \mathbf{y}^T(\mathbf{M}\dot{\mathbf{q}}^{t*} + \mathbf{D}\mathbf{f}^{t+\delta t}) \ : \ \mathbf{N}^T\mathbf{y} \geq 0 \right) \tag{2.37}$$

où δt est le pas de temps, $\dot{\mathbf{q}}^{t*}$ est une estimation de $\dot{\mathbf{q}}$ à un instant $t*$ prédit et $\mathbf{N}$ est la direction normale au contact. Cette formulation utilise une discrétisation du cône de frottement comme dans [Stewart et Trinkle, 2000]. Par ailleurs la dynamique est discrétisée et cette méthode est à mettre en parallèle avec les méthodes d'intégration *Time-stepping* [Moreau, 1988] (voir partie 2.5). Il est à noter que résoudre le problème 2.23 est équivalent, sous certaines conditions, à résoudre un problème d'optimisation :

$$0 \leq \mathbf{f} \perp \mathbf{\Lambda}\mathbf{f} + \mathbf{c} \geq 0 \iff \min_{\mathbf{f}} \mathbf{f}^T\mathbf{\Lambda}\mathbf{f} + \mathbf{f}^T\mathbf{c} \tag{2.38}$$

2.4.2.2 Méthodes itératives

Liu et Wang [Liu et Wang, 2005] ont appliqué une approche itérative de Gauss-Seidel à la robotique, dans des cas cependant d'objets simples et non dans des cas complexes mettant en scène des robots humanoïdes que nous désirons traiter. Cette approche, bâtie sur les travaux de Jean-Jacques Moreau, a déjà été introduite antérieurement pour la simulation de matériaux granulaires [Moreau, 2003]. La méthode de Gauss-Seidel est une méthode classique pour résoudre des systèmes linéaires. En considérant le système 2.32 ($\mathbf{v} = \mathbf{\Lambda}\mathbf{f} + \mathbf{v}_{\text{libre}}$), de taille maintenant $3m$ (car il y a $3m$ inconnues en forces),

le principe est de visiter chaque inconnue f_i une à une et de résoudre chaque équation i en bloquant toutes les autres inconnues f_j, $j \neq i$ à la valeur la plus récemment calculée :

$$f_i = [\Lambda_{ii}]^{-1}_{(3\times3)} \left(v_{\text{libre}i} - \sum_{j=1}^{i-1} [\Lambda_{ij}]_{(3\times3)} f_j - \sum_{j=i+1}^{m} [\Lambda_{ij}]_{(3\times3)} f_j \right) \tag{2.39}$$

et ce, itérativement, jusqu'à la convergence qui est assurée si la matrice $\mathbf{\Lambda}$ est à diagonale dominante ($|\Lambda_{ii}| > \sum_{j\neq i} |\Lambda_{ij}|$). La convergence est prouvée dans [Jourdan *et al.*, 1998]. Dans la pratique, la convergence relative de l'ensemble du système peut être utilisée.

La méthode de type Gauss-Seidel est à mettre en parallèle avec la méthode de Jacobi [9] dont la différence majeure est de ne pas utiliser les valeurs déjà calculées de f_j, autrement dit :

$$f_i = [\Lambda_{ii}]^{-1}_{(3\times3)} \left(v_{\text{libre}i} - \sum_{j=1, j\neq i}^{m} [\Lambda_{ij}]_{(3\times3)} f_j \right) \tag{2.40}$$

Jean [Jean, 1993] et Alart [Alart et Curnier, 1991] ont développé un algorithme couplant l'approche de Gauss-Seidel avec une méthode de Newton pour résoudre un à un les efforts de contact en chaque point de contact. Cette méthode est adaptée pour résoudre des cas d'un contact unique, mais elle est a été étendue à des cas multi-contacts [Duriez *et al.*, 2006]. Nous la développerons dans le chapitre suivant.

L'intérêt majeur des méthodes itératives est de pouvoir s'affranchir de discrétiser les cônes de frottement tout en convergeant de manière garantie vers une solution, à condition que la matrice de Delassus soit à diagonale dominante, sans aucune régularisation, et donc de travailler avec des matrices de taille égale au nombre de points de contact. Le temps de calcul est ainsi considérablement réduit et la précision est améliorée. On peut néanmoins remarquer que la convergence peut être plus rapide et donc les calculs accélérés si les valeurs initiales choisies sont proches de la solution.

2.4.2.3 Remarques

Renouf *et al.* [Renouf *et al.*, 2005, Renouf et Acary, 2006] ont réalisé une comparaison entre les différentes méthodes de résolution du contact avec frottement par contraintes. Ils ont utilisé un schéma d'intégration de type *Time-stepping* de Moreau (voir partie 2.5) et ont considéré une résolution par blocs de matrices, ce qui permet d'accélérer la résolution. Ils ont montré que les méthodes itératives étaient plus rapides que les méthodes par pivot (voir figure 2.21). Ces résultats peuvent se vérifier en utilisant la plate-forme logicielle SICONOS de l'équipe BIBOP de l'INRIA [Acary et Perignon, 2007] [10] qui propose une vaste gamme de solveurs numériques, que ce soit en 2D ou en 3D, avec ou sans discrétisation des cônes de frottement.

Duriez *et al.* [Duriez, 2004, Duriez *et al.*, 2006] utilisent aussi l'approche de Gauss-Seidel pour calculer en temps réel le retour d'efforts avec des objets déformables. Ils ont en particulier montré des

9. On pourra se référer à `http://en.wikipedia.org/wiki/Jacobi_method`.
10. `http://siconos.inrialpes.fr/`

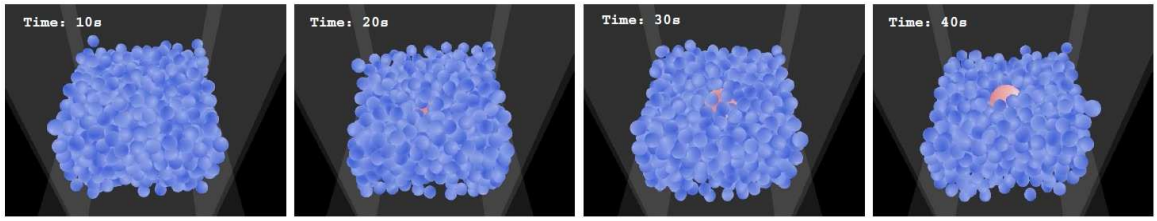


figure 2.20: *Simulation de matériaux granulaires (tiré de [Renouf et Acary, 2006]).*

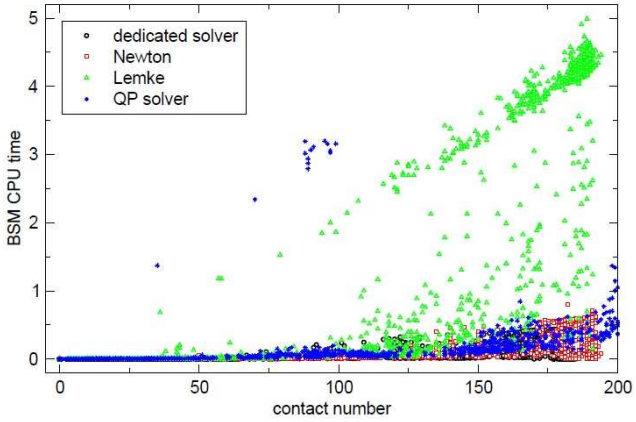


figure 2.21: *Comparaison de différentes méthodes de résolution de contact frottant par contraintes dans le cadre d'une résolution par blocs (tiré de [Renouf et Acary, 2006]). Le solveur de Lemke est utilisé pour résoudre les problèmes de type LCP. Le solveur QP (Quadratic Programming) est utilisé pour résoudre des problèmes d'optimisation. La formulation LCP peut être écrite de manière équivalente sous la forme d'un problème d'optimisation (voir plus haut).*

résultats intéressants qui sont (voir figure 2.22) :

- dans des configurations où il y a plus de dix points de contact, l'approche de Gauss-Seidel est beaucoup plus performante que l'approche classique LCP,
- avec une approche classique LCP, un minimum de huit facettes est requis dans la discrétisation des cônes de frottement pour avoir 10% d'écart de précision avec l'approche de Gauss-Seidel.

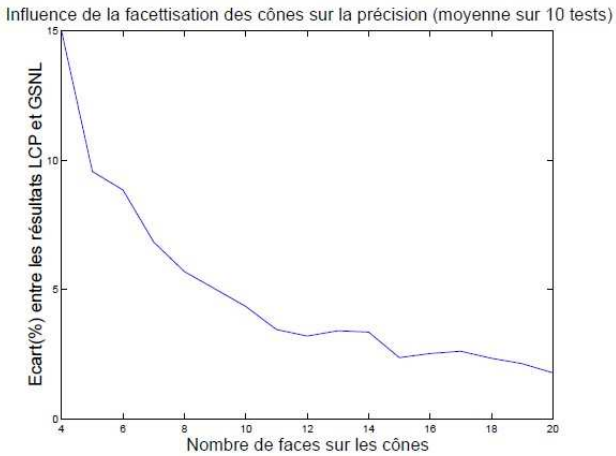


figure 2.22: *Ecart de précision entre l'approche LCP et Gauss-Seidel en fonction du nombre de facettes (tiré de [Duriez, 2004]).*

Un autre avantage d'utiliser des méthodes itératives est qu'il n'est pas nécessaire d'attendre la fin des calculs pour obtenir une solution. En effet on peut obtenir une estimation de la solution et donc arrêter la résolution au moment désiré, ce qui n'est pas le cas avec les méthodes classiques de type Lemke. Le lecteur pourra se référer à [Brogliato *et al.*, 2002, Acary et Brogliato, 2008] pour une dissertation approfondie sur les systèmes mécaniques non-réguliers.

2.4.3 Résumé

Les méthodes par pénalités sont simples à implémenter mais font apparaître des inconvénients majeurs, en particulier pour la paramétrisation des gains, la stabilité et la précision. Les méthodes par contraintes peuvent être précises à condition de ne pas discrétiser le cône de frottement. Comme nous souhaitons réaliser des simulations interactives, nous souhaitons avoir des temps de calcul petits et par conséquent nous allons nous diriger vers une approche itérative de type Gauss-Seidel pour résoudre les problèmes de contact avec frottement. Par ailleurs, une approche itérative permet plus facilement de régler le compromis entre la précision du calcul et les contraintes de temps. En effet, en rendu haptique, on pourra toujours arrêter la résolution du problème aux temps critiques en se contentant d'une solution approchée à une précision que l'on pourra connaître.

D'une manière générale, la robustesse des approches proposées dans des situations complexes avec un grand nombre de points de contact et de contraintes bilatérales n'est pas prouvée et les exemples donnés dans la littérature sont généralement très simples [11]. De plus, leur validité n'est en générale pas démontrée expérimentalement. Nous pensons que les implémentations effectuées dans les différents travaux pourraient cacher des problèmes pour lesquels les hypothèses de départ seront à remettre en cause.

2.5 Intégration numérique

2.5.1 Méthodes explicites et implicites

Une fois les accélérations articulaires calculées, il devient possible de mettre à jour l'état (vitesses et positions) global des acteurs composant la simulation, en particulier pour le rendu graphique. Comme l'équation 2.18 est une équation différentielle, nous avons donc besoin d'intégrer le système. Cependant le choix de la méthode d'intégration est important, en particulier elle peut ou non créer des instabilités et peux nécessiter aussi quelques ajustements de pas ou de gains. Nous pouvons classer les méthodes d'intégration en deux catégories : les méthodes explicites et les méthodes implicites. On note δt le pas de temps.

11. Nous pouvons cependant noter les travaux de Jean-Jacques Moreau dans lesquels les milieux granulaires ont un grand nombre de points de contact.

2.5.1.1 Méthodes explicites

Les méthodes explicites estiment le mouvement courant à partir des pas précédents. Nous citons quelques exemples de méthodes explicites :

- Euler explicite (ordre 1) : $\mathbf{q}^{t+\delta t} = \mathbf{q}^t + \delta t\dot{\mathbf{q}}^t$ et $\dot{\mathbf{q}}^{t+\delta t} = \dot{\mathbf{q}}^t + \delta t\ddot{\mathbf{q}}^t$,
- Verlet (ordre 2) : $\mathbf{q}^{t+\delta t} = \mathbf{q}^t + \delta t\dot{\mathbf{q}}^t + \frac{\delta t^2}{2}\ddot{\mathbf{q}}^t$ et $\dot{\mathbf{q}}^{t+\delta t} = \dot{\mathbf{q}}^t + \delta t\frac{\ddot{\mathbf{q}}^t+\ddot{\mathbf{q}}^{t+\delta t}}{2}$,
- Runge-Kutta d'ordre 2 : $\mathbf{q}^{t+\delta t} = \mathbf{q}^t + \delta t f\left(\mathbf{q}^t + \frac{k}{2}, t + \frac{\delta t}{2}\right)$ et $k = \delta t f(\mathbf{q}^t, t)$,
- Runge-Kutta d'ordre 4 : $\mathbf{q}^{t+\delta t} = \mathbf{q}^t + \frac{k_1}{6} + \frac{k_2}{3} + \frac{k_3}{3} + \frac{k_4}{6}$ et $k_1 = \delta t f(\mathbf{q}^t, t)$, $k_2 = \delta t f\left(\mathbf{q}^t + \frac{k_1}{2}, t + \frac{\delta t}{2}\right)$, $k_3 = \delta t f\left(\mathbf{q}^t + \frac{k_2}{2}, t + \frac{\delta t}{2}\right)$, $k_4 = \delta t f(\mathbf{q}^t + k_3, t + \delta t)$.

Il est à noter que plus l'ordre d'intégration est élevé, plus la méthode est précise et robuste. Cependant des méthodes d'ordre élevé de type Runge-Kutta requiert un plus grand nombre de calculs (dans le cas de la méthode Runge-Kutta d'ordre 4, il faut calculer quatre fois la dynamique) et donc ne sont pas appropriées pour des simulations en temps réel.

2.5.1.2 Méthodes implicites

Les méthodes implicites estiment le mouvement courant à partir du mouvement final. Ces méthodes sont plus stables que les méthodes explicites mais requièrent des calculs supplémentaires qui peuvent s'avérer complexes. Nous citons quelques exemples de méthodes implicites :

- Euler implicite : $\mathbf{q}^{t+\delta t} = \mathbf{q}^t + \delta t\dot{\mathbf{q}}^{t+\delta t}$ et $\dot{\mathbf{q}}^{t+\delta t} = f(\mathbf{q}^{t+\delta t}, t + \delta t)$,
- θ-méthode : $\mathbf{q}^{t+\delta t} = \mathbf{q}^t + \delta t\theta\dot{\mathbf{q}}^{t+\delta t} + \delta t(1-\theta)\dot{\mathbf{q}}^t$, avec θ compris entre 0 et 1.

On remarque que dans le cas de la θ-méthode (qui semble être une homotopie entre deux méthodes d'intégration numérique), si θ est égal à 0, on se retrouve avec une méthode d'Euler explicite et si θ est égal à 1, on se retrouve avec une méthode d'Euler implicite. Il est préférable de prendre θ compris entre $\frac{1}{2}$ et 1 pour assurer une meilleure stabilité [Barclay *et al.*, 2000].

2.5.1.3 Remarques

Dans le contexte temps réel, les méthodes explicites peuvent être utilisées car elles sont rapides et les forces de contact sont déterminées par les calculs des pas de temps précédents. Cependant, pour des simulations de corps rigides, les méthodes implicites sont à privilégier car les problèmes de contact sont non-réguliers et font apparaître des discontinuités en force, ce qui peut rendre le système instable. Un autre avantage des méthodes implicites est qu'il est possible de prendre un pas de temps plus élevé, qui convient dans un contexte temps réel.

2.5.2 *Event-driven* et *Time-stepping*

Pour intégrer la dynamique complète d'un système mécanique non régulier sur un pas de temps, il existe deux classes *d'intégrateurs*. Il ne s'agit pas de méthodes d'intégration telles qu'on l'entend couramment, même si elles sont dénommées de la même manière, ce sont plutôt deux façons d'initier une intégration dans une simulation, et qui peuvent utiliser les méthodes d'intégration présentées précédemment.

2.5.2.1 Schéma d'intégration évènementiel ou *Event-driven*

Dans ce schéma [Pfeiffer et Glocker, 1996], la dynamique est décomposée en modes, c'est-à-dire en intervalles de temps où la dynamique est régulière, et en évènements discrets où la dynamique n'est pas régulière. D'un point de vue numérique, cette approche utilise la décomposition temporelle de la dynamique pour détecter et résoudre la dynamique à des instants particuliers comme les impacts. Plus concrètement, les instants discrets de contact sont détectés (un à un), la simulation est arrêtée à chacun de ces instants, les forces de contact et les forces d'impact sont calculées, puis la dynamique est réinitialisée et enfin la simulation régulière repart avec les nouvelles conditions données par la prise en compte particulière de ces évènements. Dans les intervalles où la dynamique est régulière, les méthodes d'intégration classiques peuvent être utilisées. Cette méthode n'est pas adaptée aux systèmes avec beaucoup de points de contact car elle nécessite la détection des évènements et tend donc à résoudre chaque évènement. Dans ce cas, la gestion de ces évènements peut devenir très complexe. Des traitements plus particuliers de la collision et de retour arrière sont à prévoir car il est nécessaire de détecter les instants (et leurs temps) où un évènement se produit, des algorithmes de détection de collision continus (aussi dits 4D) sont nécessaires dans ce cas [Redon *et al.*, 2002]. De fait cette méthode est très précise.

2.5.2.2 Schéma d'intégration pas à pas ou *Time-stepping*

Reprenons l'équation de la dynamique que nous réécrivons de la manière suivante :

$$\mathbf{M}\ddot{\mathbf{q}} = \mathbf{B} + \mathbf{F} \tag{2.41}$$

$\mathbf{M}$ est la matrice de masse du système, $\mathbf{q}$ est la position, $\mathbf{B}$ sont les forces de Coriolis, la gravité et les forces extérieures connues ($\mathbf{B} = -\mathbf{b} - \mathbf{g}$), et $\mathbf{F} = \sum_i \mathbf{N}_i^T \mathbf{f}_i$ représente les efforts extérieurs inconnus tels que les forces d'interaction avec $\mathbf{N}_i$ la direction généralisée des efforts et qui peut se décomposer en direction normale $\mathbf{N}_{ni}$ et en direction tangentielle $\mathbf{N}_{ti}$. Nous y ajoutons des contraintes de non-pénétration qui s'écrivent de la manière générale suivante :

$$\phi_i(\mathbf{q}) \geq 0,\ i = 1, ..., m \tag{2.42}$$

avec m le nombre de contraintes. Par ailleurs nous notons $\mathbf{v}_r$ la vitesse relative entre deux corps en contact, et enfin $\cdot^t \approx \cdot(t, \mathbf{q}^t, \dot{\mathbf{q}}^t)$, $\cdot^{t*} = \cdot(t + \delta t, \mathbf{q}^t, \dot{\mathbf{q}}^t)$.

Le principe général d'un schéma de type *Time-stepping* est d'intégrer l'équation de la dynamique sur un pas de temps $[t, t + \delta t]$ avec δt l'échantillonnage du pas, petit de préférence, et de discrétiser l'équation obtenue. Partons de l'équation 2.41, intégrée sur le pas de temps δt, elle s'écrit :

$$\int_{[t,t+\delta t]} \mathbf{M}\ddot{\mathbf{q}} dt = \int_{[t,t+\delta t]} \mathbf{B} dt + \int_{[t,t+\delta t]} \sum_i \mathbf{N}_i^T \mathbf{f}_i dt \tag{2.43}$$

Cette équation est discrétisée de la manière suivante :

$$\int_{[t,t+\delta t]} \mathbf{M}\ddot{q}dt \approx \tilde{\mathbf{M}}(\dot{\mathbf{q}}^{t+\delta t} - \dot{\mathbf{q}}^t) \tag{2.44}$$

$$\int_{[t,t+\delta t]} \mathbf{B}dt \approx \tilde{\mathbf{B}}\delta t \tag{2.45}$$

$$\int_{[t,t+\delta t]} \sum_i \mathbf{N}_i^T \mathbf{f}_i dt \approx \sum_i \tilde{\mathbf{N}}_i^T \mathbf{f}_i \tag{2.46}$$

où $\tilde{\mathbf{M}}$, $\tilde{\mathbf{B}}$ et $\tilde{\mathbf{N}}_i$ sont des approximations de $\mathbf{M}$, $\mathbf{B}$ et $\mathbf{N}_i$. Nous obtenons alors :

$$\tilde{\mathbf{M}}(\dot{\mathbf{q}}^{t+\delta t} - \dot{\mathbf{q}}^t) = \tilde{\mathbf{B}}\delta t + \sum_i \tilde{\mathbf{N}}_i^T \mathbf{f}_i \tag{2.47}$$

Par ailleurs, la mise à jour de la position peut s'écrire de manière générale [Studer, 2008] :

$$\mathbf{q}^{t+\delta t} = \mathbf{q}^t + ((1-\xi)\dot{\mathbf{q}}^t + \xi\dot{\mathbf{q}}^{t+\delta t})\delta t \tag{2.48}$$

avec ξ une constante comprise entre 0 et 1. Cette équation n'est rien d'autre que la θ-méthode. De cette formulation générale, plusieurs schémas d'intégration ont été proposés [Moreau, 1988, Jean, 1999, Förg *et al.*, 2005, Stewart et Trinkle, 1996, Stewart et Trinkle, 2000, Anitescu et Potra, 1997, Anitescu et Potra, 2002, Anitescu et Hart, 2004, Potra *et al.*, 2006, Gavrea *et al.*, 2008, Studer, 2008, Chakraborty *et al.*, 2007, Song *et al.*, 2004, Paoli et Schaztman, 2002]. Nous passons rapidement en revue quelques schémas puis nous ferons quelques remarques sur ces méthodes. Il faut noter que la description de ces méthodes a été remarquablement fait par [Studer, 2008] récemment et nous résumons ici la synthèse de ses résultats qui nous ont paru très pertinents comme état de l'art.

La méthode de Moreau propose de discrétiser la dynamique de la manière suivante [Moreau, 1988] :

$$\mathbf{M}^{t+\frac{\delta t}{2}}(\dot{\mathbf{q}}^{t+\delta t} - \dot{\mathbf{q}}^t) = \mathbf{B}^{t+\frac{\delta t}{2}}\delta t + \sum_i (\mathbf{N}_i^{t+\frac{\delta t}{2}})^T \mathbf{f}_i \tag{2.49}$$

$$\mathbf{q}^{t+\delta t} = \mathbf{q}^t + \frac{\dot{\mathbf{q}}^{t+\delta t} + \dot{\mathbf{q}}^t}{2}\delta t \tag{2.50}$$

$\mathbf{M}$, $\mathbf{B}$ et $\mathbf{N}_i$ sont approximés au point médian $\mathbf{q} = \mathbf{q}^t + \frac{\delta t}{2}\dot{\mathbf{q}}^t$. Les contraintes sont écrites en vitesse et font apparaître la loi d'impact de Newton :

$$\mathbf{v}_r^{t+\delta t} + e\mathbf{v}_r^t \geq 0 \tag{2.51}$$

On rappelle que e est le coefficient de restitution, compris entre 0 et 1. Cette formulation intègre aussi bien le contact que l'impact. En effet, si la vitesse $\mathbf{v}_r^t$ est suffisamment élevée, alors il y a impact et la loi d'impact s'applique et $\mathbf{v}_r^{t+\delta t} = -e\mathbf{v}_r^t$, ce qui provoque un décollement du contact. En revanche si la vitesse $\mathbf{v}_r^t$ est suffisamment petite tel que sur un pas de temps complet l'état du contact ne varie pas, alors il y a contact. Ce modèle d'impact a cependant été validé dans des cas très simples, par exemple

une bille sur un sol plat.

La méthode de Jean, appelée aussi θ-méthode modifiée, discrétise la dynamique de la manière suivante [Jean, 1999] :

$$\mathbf{M}^t(\dot{\mathbf{q}}^{t+\delta t} - \dot{\mathbf{q}}^t) = ((1-\theta)\mathbf{B}^t + \theta\mathbf{B}_{t+\delta t}\delta t + \sum_i (\mathbf{N}_i^t)^T \mathbf{f}_i \tag{2.52}$$

$$\mathbf{q}^{t+\delta t} = \mathbf{q}^t + ((1-\theta)\dot{\mathbf{q}}^t + \theta\dot{\mathbf{q}}^{t+\delta t})\delta t \tag{2.53}$$

avec θ compris entre 0 et 1. On peut remarquer que la dernière équation est la même que 2.48 avec $\xi = \theta$. En choisissant $\theta \geq \frac{1}{2}$, la méthode d'intégration est à dominante implicite et est par conséquent stable. Ici $\mathbf{M}$, $\mathbf{B}$ et $\mathbf{N}_i$ sont calculés au début du pas de temps par commodité mais il est possible de choisir un autre instant intermédiaire. Les contraintes sont écrites en vitesse, respectivement pour le contact et le frottement :

$$\frac{\tilde{\phi}_i}{\delta t} - \mathbf{v}_r^{t+\delta t} \geq 0,\ \mathbf{N}_i^T \dot{\mathbf{q}}^{t\delta t} \geq 0 \tag{2.54}$$

où $\tilde{\phi}_i$ est une approximation de la contrainte au point $\mathbf{q} = \mathbf{q}^t + (1-\theta)\dot{\mathbf{q}}^t\delta t$. En réalité, la contrainte pour le contact est une contrainte en position linéarisée. De fait, celle-ci est exprimée implicitement en vitesse. Il est cependant important de ne pas choisir une contrainte indépendamment de la mise à jour de la position, auquel cas le traitement des impacts devient imprédictible. Par ailleurs, les erreurs numériques peuvent faire osciller le système. Pour que les contraintes soient respectées pendant un contact, il est nécessaire d'avoir une vitesse relative nulle.

La méthode proposée par Förg *et al.* discétise la dynamique de la manière suivante [Förg *et al.*, 2005] :

$$\mathbf{M}^t(\dot{\mathbf{q}}^{t+\delta t} - \dot{\mathbf{q}}^t) = \mathbf{B}^t\delta t + \sum_i \mathbf{N}_i^T \mathbf{f}_i \tag{2.55}$$

$$\mathbf{q}^{t+\delta t} = \mathbf{q}^t + \dot{\mathbf{q}}^{t+\delta t}\delta t \tag{2.56}$$

Les auteurs traitent les impacts inélastiques et le multi-contact est résolu par une approche itérative de type Gauss-Seidel. Il est montré par ailleurs l'efficacité de leur méthode par rapport à une méthode de résolution de LCP (en l'occurence la méthode de Lemke) et une approche lagrangienne. Les contraintes sont écrites en positions mais sont linéarisées.

Stewart et Trinkle proposent de discrétiser le modèle de la manière suivante [Stewart et Trinkle, 1996, Stewart et Trinkle, 2000] :

$$\tilde{\mathbf{M}}(\dot{\mathbf{q}}^{t+\delta t} - \dot{\mathbf{q}}^t) = \tilde{\mathbf{B}}\delta t + \sum_i \mathbf{N}_i^T \mathbf{f}_i \tag{2.57}$$

$$\mathbf{q}^{t+\delta t} = \mathbf{q}^t + \dot{\mathbf{q}}^{t+\delta t}\delta t \tag{2.58}$$

$\tilde{\mathbf{M}}$ et $\tilde{\mathbf{B}}$ sont calculés à $\mathbf{q} = \mathbf{q}^t + \dot{\mathbf{q}}^t\delta t$. La méthode proposée ne s'applique qu'à des impacts inélastiques. Les contraintes de contact sont écrites en position avec une tolérance ϵ, le frottement étant écrit en

vitesse :

$$(\mathbf{N}_{ni}^t)^T \mathbf{q}^{t+\delta t} + \epsilon \geq 0,\ (\mathbf{N}_{ti}^t)^T \dot{\mathbf{q}}^{t+\delta t} \geq 0 \tag{2.59}$$

Anitescu et Potra ont proposé plusieurs méthodes. Une première méthode, basée sur les travaux de Stewart et Trinkle, est de discrétiser de la manière suivante [Anitescu et Potra, 1997] :

$$\tilde{\mathbf{M}}(\dot{\mathbf{q}}^{t+\delta t} - \dot{\mathbf{q}}^t) = \tilde{\mathbf{B}}\delta t + \sum_i \tilde{\mathbf{N}}_i^T \mathbf{f}_i \tag{2.60}$$

$$\mathbf{q}^{t+\delta t} = \mathbf{q}^t + \dot{\mathbf{q}}^t \delta t \tag{2.61}$$

$\tilde{\mathbf{M}}$ et $\tilde{\mathbf{B}}$ sont calculés à $\mathbf{q} = \mathbf{q}^t + \dot{\mathbf{q}}^t \delta t$. Dans le cas où il y a une collision, Anitescu et Potra proposent d'estimer le temps de collision, la position au moment de la collision et la vitesse juste avant la collision puis de résoudre l'impact en deux phases : une phase de compression et une phase de décompression, prenant ainsi en compte les collisions partiellement élastiques. Cette résolution de l'impact en deux phases est aussi appelée impact de Poisson. Dans cette méthode, les contraintes sont écrites en vitesse, respectivement pour le contact et le frottement :

$$(\mathbf{N}_{ni}^t)^T \dot{\mathbf{q}}^{t+\delta t} \geq 0,\ (\mathbf{N}_{ti}^t)^T \dot{\mathbf{q}}^{t+\delta t} \geq 0 \tag{2.62}$$

Une seconde méthode est basée sur une méthode d'Euler linéaire implicite et discrétise les équations de la dynamique de la manière suivante [Anitescu et Potra, 2002] :

$$\mathbf{M}^t(\dot{\mathbf{q}}^{t+\delta t} - \dot{\mathbf{q}}^t) = \mathbf{B}^t \delta t + \nabla_{\mathbf{q}}\mathbf{B}^t(\mathbf{q}^{t+\delta t} - \mathbf{q}^t) + \nabla_{\dot{\mathbf{q}}}\mathbf{B}^t(\dot{\mathbf{q}}^{t+\delta t} - \dot{\mathbf{q}}^t) + \sum_i (\mathbf{N}_i^t)^T \mathbf{f}_i \tag{2.63}$$

$$\mathbf{q}^{t+\delta t} = \mathbf{q}^t + \dot{\mathbf{q}}^{t+\delta t}\delta t \tag{2.64}$$

avec $\nabla_{\mathbf{q}}\mathbf{B} = \frac{\partial \mathbf{B}}{\partial \mathbf{q}}$ et $\nabla_{\dot{\mathbf{q}}}\mathbf{B} = \frac{\partial \mathbf{B}}{\partial \dot{\mathbf{q}}}$. Dans cette méthode, seuls les impacts inélastiques sont considérés mais il est possible de résoudre l'impact comme dans [Anitescu et Potra, 1997]. Ici aussi les contraintes sont écrites en vitesse, respectivement pour le contact et le frottement :

$$(\mathbf{N}_{ni}^t)^T \dot{\mathbf{q}}^{t+\delta t} \geq 0,\ (\mathbf{N}_{ti}^t)^T \dot{\mathbf{q}}^{t+\delta t} \geq 0 \tag{2.65}$$

Nous pouvons remarquer que dans la première méthode, la mise à jour de la position se fait avec une intégration explicite d'Euler tandis que la seconde méthode emploie une méthode implicite.

Une variante de cette méthode est proposée dans [Anitescu et Hart, 2004] :

$$\mathbf{M}^t(\dot{\mathbf{q}}^{t+\delta t} - \dot{\mathbf{q}}^t) = \mathbf{B}^t \delta t + \sum_i (\mathbf{N}_i^t)^T \mathbf{f}_i \tag{2.66}$$

$$\mathbf{q}^{t+\delta t} = \mathbf{q}^t + \dot{\mathbf{q}}^{t+\delta t}\delta t \tag{2.67}$$

Les contraintes de contact, qui ne sont incluses dans la résolution que dans le cas $\phi_i(\mathbf{q}^t) < \epsilon$ avec ϵ suffisamment petit, c'est-à-dire dans le cas où la contrainte i devient active, sont écrites en vitesse et

résultent d'une linéarisation des contraintes :

$$(\mathbf{N}_{ni}^t)^T \dot{\mathbf{q}}^{t+\delta t} + \frac{\phi_i(\mathbf{q}^t)}{\delta t} \geq 0 \tag{2.68}$$

Potra *et al.* proposent de discrétiser la dynamique de la manière suivante [Potra *et al.*, 2006] :

$$\tilde{\mathbf{M}}(\dot{\mathbf{q}}^{t+\delta t} - \dot{\mathbf{q}}^t) = \tilde{\mathbf{B}}\delta t + \sum_i (\mathbf{N}_i^{t+\frac{\delta t}{2}})^T \mathbf{f}_i \tag{2.69}$$

$$\mathbf{q}^{t+\delta t} = \mathbf{q}^t + (\dot{\mathbf{q}}^{t+\delta t} + \dot{\mathbf{q}}^t)\frac{\delta t}{2} \tag{2.70}$$

$\tilde{\mathbf{M}}$ et $\tilde{\mathbf{B}}$ sont calculés de la manière suivante :

$$\tilde{\mathbf{M}} = \mathbf{M}^t - \frac{\delta t}{2}\tilde{\mathbf{B}}_{\dot{\mathbf{q}}}^t - \frac{\delta t^2}{4}\tilde{\mathbf{B}}_q^t,\ \tilde{\mathbf{B}} = \frac{\delta t}{2}\left(\mathbf{B}^t + \mathbf{B}^{t*} + \frac{\delta t}{2}\tilde{\mathbf{B}}_q^t \dot{q}^t\right) \tag{2.71}$$

avec $\tilde{\mathbf{B}}$ est une approximation de $\mathbf{B}^{t*}$. Il s'agit d'un schéma d'intégration du deuxième ordre utilisant une méthode trapézoïdale implicite contrairement aux autres méthodes qui sont d'ordre 1. Par ailleurs, cette méthode propose un moyen de déterminer les évènements de manière robuste afin de préserver l'ordre d'intégration. En particulier, les contraintes actives sont recherchées et la détection de l'évènement se fait en réalisant une interpolation cubique de la position $\mathbf{q}$ et en recherchant le plus petit $t*$ tel que les contraintes soient des égalités, c'est-à-dire $\phi_i(\mathbf{q}^{t*}) = 0$ dans l'intervalle du pas de temps. L'impact est ensuite traité par une loi de Poisson. Les contraintes sont écrites en vitesse, respectivement pour le contact et le frottement :

$$(\mathbf{N}_{ni}^t)^T \frac{\dot{\mathbf{q}}^{t+\delta t} + \dot{\mathbf{q}}^t}{2} \geq 0,\ (\mathbf{N}_{ti}^t)^T \frac{\dot{\mathbf{q}}^{t+\delta t} + \dot{\mathbf{q}}^t}{2} \geq 0 \tag{2.72}$$

Gavrea *et al.* proposent de discrétiser la dynamique de la manière suivante [Gavrea *et al.*, 2008] :

$$\tilde{\mathbf{M}}(\dot{\mathbf{q}}^{t+\delta t} - \dot{\mathbf{q}}^t) = \tilde{\mathbf{B}}\delta t + \sum_i (\mathbf{N}_i^t)^T \mathbf{f}_i \tag{2.73}$$

$$\mathbf{q}^{t+\delta t} = \mathbf{q}^t + ((1-a)\dot{\mathbf{q}}^t + a\dot{\mathbf{q}}^{t+\delta t})\delta t \tag{2.74}$$

$\tilde{\mathbf{M}}$ et $\tilde{\mathbf{B}}$ sont calculés de la manière suivante :

$$\tilde{\mathbf{M}} = \mathbf{M}^t - \zeta\delta t(\mathcal{F}^t + \tilde{\mathbf{k}}_{\dot{q}}^{t*}) - \xi\zeta\delta t^2 \tilde{\mathbf{k}}_q^{t*},\ \tilde{\mathbf{B}} = \delta t((1-\zeta)\mathbf{k}^t + \zeta\mathbf{k}^{t*}) + (1-\zeta)\delta t\mathcal{F}^t\dot{\mathbf{q}}^t + \zeta\delta t^2\tilde{\mathbf{k}}_q^t\dot{q}^t \tag{2.75}$$

où ζ est compris entre 0 et 1, $\mathcal{F}$ et $\mathbf{k}$ sont tels que $\mathbf{B} = \mathcal{F}\dot{\mathbf{q}} + \mathbf{k}$, et $\tilde{\mathbf{k}}$ est une approximation de $\mathbf{k}^{t*}$. Ici aussi l'instant d'impact est déterminé et l'impact est traité avec une loi de Poisson, comme dans [Potra *et al.*, 2006]. Les contraintes sont écrites en vitesse, respectivement pour le contact et le frottement :

$$(\mathbf{N}_{ni}^t)^T(\zeta\dot{\mathbf{q}}^{t+\delta t} + (1-\zeta)\dot{\mathbf{q}}^t) \geq 0,\ (\mathbf{N}_{ti}^t)^T(\zeta\dot{\mathbf{q}}^{t+\delta t} + (1-\zeta)\dot{\mathbf{q}}^t) \geq 0 \tag{2.76}$$

Studer [Studer, 2008] propose deux méthodes pour discrétiser la dynamique. La première est basée sur les travaux de Gear, Gupta et Leimkuhler :

$$\mathbf{M}(\dot{q}^{t+\delta t} - \dot{\mathbf{q}}^t) = \mathbf{B}\delta t + F_{\dot{q}} \tag{2.77}$$

$$\mathbf{q}^{t+\delta t} = \mathbf{q}^t + \frac{\delta t}{2}(\dot{\mathbf{q}}^{t+\delta t} - \dot{\mathbf{q}}^t) + F_q \tag{2.78}$$

où $F_{\dot{q}}$ et F_q sont des multiplicateurs de Lagrange intervenant dans les contraintes en vitesse et en position respectivement :

$$\dot{\mathbf{q}}^{t+\delta t} + e\dot{\mathbf{q}}^t \geq 0,\ \mathbf{q}^{t+\delta t} + \epsilon \geq 0 \tag{2.79}$$

On remarque une ressemblance avec la méthode de Moreau. La différence avec celle-ci est la présence des multiplicateurs $F_{\dot{q}}$ et F_q qui sont évalués en effectuant une projection. Les impacts élastiques et inélastiques peuvent être traités. L'inconvénient de cette méthode est qu'il est nécessaire d'autoriser des petites pénétrations pour éviter des oscillations.

La seconde méthode est une approche par préconditionnement :

$$\mathbf{M}(\hat{\dot{\mathbf{q}}}^{t+\delta t}) - \hat{\dot{\mathbf{q}}}^t) = \mathbf{B}\delta t^2 + \sum_i \mathbf{N}_i^T \hat{\mathbf{f}}_i \tag{2.80}$$

avec $\hat{\cdot} = \cdot\delta t$. La multiplication par δt empêche un mauvais conditionnement du problème dans le cas où les contraintes sont écrites en position.

Dans [Studer, 2008], seul le cas 1D est traité pour ces deux méthodes.

2.5.2.3 Remarques

La première remarque que l'on peut faire est la différence de traitement de la dynamique avec le schéma *Event-driven*. En effet, puisque la dynamique est intégrée sur un pas de temps (on travaille sur l'intégrale des forces et non sur les forces instantanées), on ne s'occupe pas de déterminer les évènements où il y a un changement d'état. De fait, des petites pénétrations des corps sont tolérées. On peut cependant noter l'exception de Potra *et al.* [Potra *et al.*, 2006] et Gavrea *et al.* [Gavrea *et al.*, 2008] qui effectuent une recherche d'évènements afin de préserver l'ordre d'intégration. Une conséquence de ne pas chercher le temps d'impact est que cette méthode n'est pas très précise. Plusieurs solutions sont alors proposées comme diminuer le pas de temps ou augmenter l'ordre d'intégration des schémas. La méthode a cependant l'avantage d'être simple et robuste.

L'instant de calcul des différents paramètres $\mathbf{M}$, $\mathbf{B}$, $\mathbf{N}_i$ diffère selon les méthodes, ce qui peut modifier la précision mais pas la stabilité [Studer, 2008]. En pratique, le choix du point médian comme dans le schéma de Moreau se révèle être le plus adapté.

Une autre remarque vient de la formulation des contraintes. En effet celles-ci changent en fonction des schémas et s'expriment ou bien en position [Förg *et al.*, 2005, Anitescu et Hart, 2004, Studer, 2008] ou bien en vitesse [Moreau, 1988, Anitescu et Potra, 1997, Anitescu et Potra, 2002, Potra *et al.*, 2006, Gavrea *et al.*, 2008]. Plusieurs problèmes peuvent apparaître : des problèmes de

dérive, de mauvais conditionnement, de non-linéarité des contraintes et de traitement des impacts. Dans le cas où les contraintes sont écrites en position, les problèmes de dérive sont absents, mais les équations sont mal conditionnées et les contraintes ne sont pas linéaires. Pour résoudre le mauvais conditionnement, on effectue une opération de préconditionnement, en utilisant les variables $\dot{\mathbf{q}}\delta t$ et $\mathbf{f}\delta t$ [Studer, 2008]. Pour résoudre la non-linéarité des contraintes, on linéarise les contraintes. On a alors des contraintes exprimées implicitement en vitesse. Il est cependant nécessaire de faire attention à ce que cette discrétisation soit cohérente avec la mise à jour de la position, comme pour la méthode de Jean. Pour traiter les impacts, on peut les détecter en effectuant une extrapolation [Potra *et al.*, 2006, Gavrea *et al.*, 2008]. Cependant il s'agit d'un effort supplémentaire dans le calcul dynamique et qui ne permet pas de traiter beaucoup de points. Un autre moyen est de recalculer un pas de temps contenant un impact en utilisant la méthode de Moreau.

L'écriture des contraintes en vitesse est intéressante dans la mesure où il est possible d'unifier les contraintes, géométriques et cinématiques, en une seule formulation et de prendre en compte l'impact élastique et inélastique. La discrétisation de la dynamique est alors simple, en particulier les méthodes de Moreau et d'Anitescu *et al.* sont prouvées simples et robustes. L'inconvénient est l'apparition de dérives en particulier lors d'un contact continu. Il est possible de stabiliser ces dérives à condition de linéariser les contraintes en position. On obtient alors une écriture implicite en vitesse mais qui ne permet pas de prendre en compte les impacts élastiques.

Les méthodes *Time-stepping* sont des schémas d'intégration d'ordre 1 pour la plupart. Potra *et al.* introduisent un schéma d'ordre 2. Comme indiqué précédemment, le schéma *Time-stepping* souffre d'un manque de précision qu'il est possible d'augmenter en adaptant le pas de temps et en effectuant une extrapolation. Dans les parties régulières, le pas de temps peut être grand tandis que dans les parties non-régulières, le pas de temps peut être réduit de manière à déterminer les moments où il y a un changement d'état. L'extrapolation permet une augmentation de l'ordre d'intégration. Elle n'est utilisée que dans les parties régulières, autrement dit dans les parties où le pas de temps est élevé, car dans les parties où il y a des changements d'états l'évolution de la position et de la vitesse n'est pas régulière [Studer, 2008]. Cette méthode sera de fait plus précise mais plus coûteuse en calcul.

Le lecteur pourra se référer à [Studer, 2008] pour une dissertation exhaustive et remarquablement synthétique sur les méthodes d'intégration *Time-stepping*.

2.5.3 Résumé

Pour résumer, la méthode *Time-stepping* est simple et robuste car elle permet de traiter le contact et l'impact en une seule formulation. Contrairement à la méthode *Event-driven*, le *Time-stepping* permet de ne pas avoir à rechercher les instants d'impacts à condition de choisir un pas de temps petit, et d'intégrer sur les parties régulières avec des méthodes classiques. Le *Time-stepping* tolère des petites pénétrations, ce que l'*Event-driven* ne permet pas.

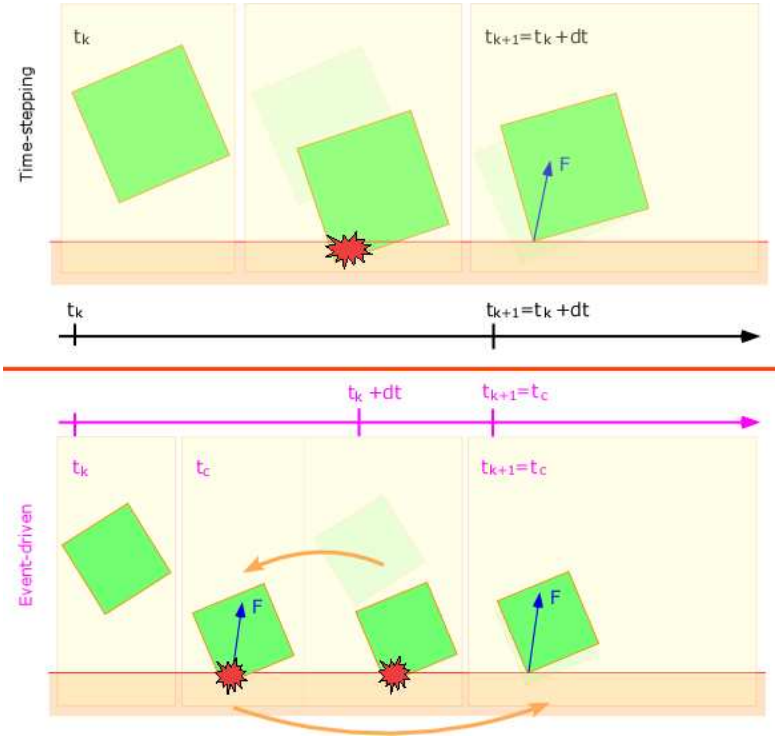


figure 2.23: Time-stepping *(en haut) et* Event-driven *(en bas) : dans le* Time-stepping *le mouvement libre est calculé, la détection de collision est faite puis le mouvement contraint est obtenu en résolvant les forces de contact ; dans l'*Event-driven *quand il y a un impact, la simulation est arrêtée, les forces sont calculées, le système est mis à jour et la simulation repart.*

2.6 Conclusion

Nous avons présenté dans ce chapitre la problématique de notre sujet. Nous avons montré plusieurs travaux pertinents du domaine permettant de traiter le calcul dynamique d'un corps poly-articulé. Nous avons vu que le traitement du contact est particulèrement délicat car il nécessite des méthodes de résolution spécifiques, plus ou moins rapides et précises, pour empêcher les corps dans un environnement de pénétrer entre eux. Les principales méthodes pour résoudre le contact sont, d'une part les méthodes par pénalités, présentes dans la plupart des simulateurs existants et qui calculent une force à partir de la violation des contraintes géométriques, et d'autre part les méthodes par contraintes, qui intègrent les contraintes de non-pénétration dans les équations de la dynamique et permettent d'écrire le système sous une forme linéaire. La prise en compte de phénomènes non réguliers tels que le frottement sec complexifie davantage le problème. En effet, dans le cas des méthodes par pénalités, la quantification de la pénétration tangentielle n'est pas triviale, tandis que dans le cas des méthodes par contraintes, le problème devient non-linéaire et nécessite une discrétisation de la loi de frottement pour garder la linéarité. Les méthodes itératives permettent de s'affranchir de ce problème et de trouver un bon compromis entre la rapidité et la précision des calculs, qui est un point important pour réaliser des simulations interactives. Nous avons enfin présenté les méthodes d'intégration numérique qui sont nécessaires car les modèles utilisés sont numériques.

Maintenant que nous avons passé en revue le problème que nous souhaitons aborder et la manière

dont il a été traité par plusieurs travaux pertinents du domaine, le chapitre suivant est consacré à la présentation des composants de notre simulateur.

Chapitre 3

Simulateur dynamique

Nous présentons dans ce chapitre le simulateur que nous avons développé à l'aide des différents outils introduits dans le chapitre précédent. Nous indiquons diverses améliorations permettant d'accélérer les calculs du modèle dynamique et d'obtenir des simulations très rapides, dans le but d'interagir avec l'environnement virtuel. Nous montrons des exemples de simulation interactive réalisés avec un dispositif haptique du commerce. Nous étendons aussi notre simulateur aux articulations à plusieurs degrés de liberté, ce qui permet d'intégrer par exemple des avatars humains.

Sommaire

3.1 Architecture du simulateur

Dans le chapitre précédent, nous avons introduit les modules nécessaires à la simulation dynamique d'un système robotique. Nous avons intégré tous ces modules dans un logiciel de prototypage et de développement nommé AMELIF, et que nous présentons. En particulier, nous avons utilisé l'algorithme de Featherstone [Featherstone, 1987] pour le calcul de la dynamique en mouvement libre ; quand au mouvement contraint par le contact, nous avons opté pour les méthodes à bases de contraintes pour le calcul des forces d'interactions. Nous nous sommes inspirés des travaux de Ruspini et Khatib [Ruspini et Khatib, 1999] que nous avons étendus au traitement des frottements grâce à une approche itérative de type Gauss-Seidel.

La contribution de ce chapitre porte sur l'intégration des modules dans le cadre d'un formalisme logiciel robuste pour traiter des cas complexes de simulation plutôt que sur un plan théorique. En réalisant ce simulateur, et relativement à OpenHRP2, nous voulions montrer (i) que les méthodes par contraintes étaient viables et plus adaptées à la simulation des contacts que les méthodes par pénalités, et (ii) qu'il est possible d'avoir des temps de calcul qui permettent d'interagir en ligne avec l'environnement virtuel via une interface haptique et qu'il est plus que possible d'unifier le calcul du rendu haptique et la simulation dynamique.

3.1.1 Le logiciel AMELIF

AMELIF a été conçu au JRL dans le but de pérenniser les développements et connaissances des outils de base de la simulation et contrôle d'un humanoïde. Dans un deuxième temps, on souhaite également l'utiliser pour pouvoir charger une simulation sur un robot réel comme le fait OpenHRP et d'autres simulateurs robotiques. Le cahier des charges que nous nous sommes imposé pour notre simulateur est le suivant :

- la modularité : on souhaite que l'utilisateur puisse implémenter n'importe quel algorithme d'un module donné de la simulation, très rapidement et très efficacement sans avoir à connaître les détails d'implémentation d'autres modules, en particulier le noyau. On garantit ainsi une robustesse dans l'évolution des différentes versions, la portabilité, et une interopérabilité entre les composants modifiés,
- on souhaite également pouvoir charger n'importe quel univers (c'est-à-dire contexte de simulation) avec ses propriétés en quelques lignes sans avoir à tout recompiler,
- simuler des interactions haptiques avec l'utilisateur moyennant des dispositifs à retour d'effort du commerce, et sans spécificité dans le calcul des forces rendues : unifier dans le même formalisme, le calcul des forces rendues à l'opérateur et le calcul des forces de contact issus de la dynamique des objets simulés ; par conséquent pouvoir réaliser des simulations temps réel,
- simuler tous types de corps poly-articulés ou non, robots ou humains virtuels, rigides ou composés de parties flexibles (c'est-à-dire déformables),
- fournir des informations de capteurs simulés proches des signaux réels, et donc offrir la possibilité d'inclure des modèles de bruit aux capteurs simulés. Ceci est nécessaire pour pouvoir confronter

la simulation à la réalité physique et fiabiliser le succès du passage des scénarios de simulations au robot réel,

– disponible à la communauté et donc téléchargeable sur Internet, ce que peu de simulateurs proposent actuellement.

L'architecture globale d'AMELIF est orientée objet et est programmée en C++. La figure 3.1 illustre les principaux composants et leurs dépendances :

– le noyau gère les données des objets présents dans la scène, en particulier la position, l'orientation, la masse, l'inertie, la position dans l'arbre du système s'il s'agit d'un système poly-articulé ou pas, etc. Le noyau permet d'effectuer la boucle de simulation et réaliser le rendu multimodal de l'environnement,

– différents modules utilisant le noyau, plus spécifiquement, ceux existants tels que la dynamique, la détection de collision, le contrôle et l'interaction :

  – Le module de dynamique calcule le modèle dynamique complet des objets (dynamique libre et contrainte avec prise en compte des phénomènes non linéaires tels que l'impact et le contact frottant) : c'est celui que nous avons le plus développé lors de cette thèse et que nous présentons en détail dans ce manuscrit.

  – Le module de détection de collision gère toutes les collisions dans l'environnement et utilise pour cela une librairie spécifique. L'utilisateur peut choisir les paires de corps qui nécessitent d'être traquées si besoin est ; par défaut, tous les objets mobiles sont traqués. Pour des besoins de test ou de validation, l'opérateur peut également associer une (ou plusieurs) méthode(s) de détection à chaque paire de corps (distance de proximité, détection d'interpénétration binaire ou quantifiée, détection continue temporelle, etc.) [van den Bergen, 2003, Ericson, 2005].

  – Le module de contrôle calcule les couples à envoyer aux corps poly-articulés grâce à des lois de commande appropriées.

  – Enfin le module d'interaction permet d'interfacer l'environnement de la simulation avec des évènements externes, notamment l'interfaçage de la simulation avec des dispositifs externes (à bras à retour d'effort, souris classique ou 3D, etc.). Nous détaillerons ce module dans la partie 3.4.

  Chaque module peut faire appel aux autres (voir figure 3.1), en particulier certaines méthodes d'un module peuvent faire appel à des méthodes d'un autre module. L'utilisateur-développeur peut créer un module indépendant des autres ou un module pour lequel il spécifie les dépendances avec un ou plusieurs modules existants ou en développement. Il peut également implémenter un nouvel algorithme pour un module existant (c'est-à-dire le redéfinir) s'il juge que ceux implémentés par défaut ne lui conviennent pas. Chaque module est compilé en une librairie indépendante des autres, ce qui exempte l'utilisateur d'avoir à installer tous les modules. Ainsi, par exemple un utilisateur n'est pas obligé de compiler ou de faire usage du module d'interaction s'il n'a pas besoin d'interagir avec l'environnement virtuel. La compilation en librairies permet également d'avoir un logiciel qui n'est pas lourd et utilisable rapidement.

– un programme principal contenant l'initialisation de la scène et la boucle de simulation. Ce

programme est écrit par l'utilisateur. Nous expliquons ci-dessous rapidement le contenu de ce programme. Celui-ci est compilé en une librairie dynamique chargée ensuite par l'application AMELIF.

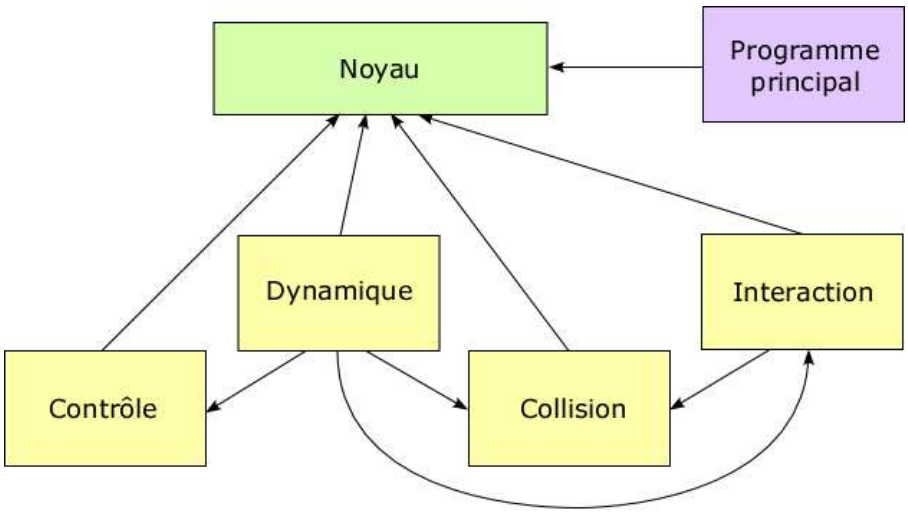


figure 3.1: *L'architecture globale de la plate-forme AMELIF. Tous les modules (en jaune clair) dépendent du noyau (en vert clair). Il peut y avoir des dépendances entre les différents modules. Le programme principal (en mauve clair) est ce que l'utilisateur écrit et contient les spécifications données par l'utilisateur.*

Si on ne modifie pas les modules, l'utilisation de cette plate-forme logicielle est très simple. Dans le cas d'une utilisation sous Windows avec le logiciel Microsoft Visual C++, l'utilisateur crée un projet avec le programme principal. Ce programme doit contenir :

- la création d'une classe contenant les objets composants la simulation (l'univers, la simulation, les corps mobiles présents dans l'environnement, la détection de collision, le dispositif d'interaction si on veut interagir avec l'environnement),
- l'initialisation de l'univers : l'univers est décrit dans un fichier XML [1]. L'utilisateur doit spécifier le type de système à simuler (corps simple ou corps poly-articulé), fournir les données géométriques et physiques du système, autrement dit la position et l'orientation de départ, la masse, le centre de masse, l'inertie et le fichier VRML [2] des points et des triangles qui composent la géométrie des objets à créer. L'utilisateur spécifie également l'angle de vue de la caméra et les lumières pour la scène (voir figure 3.2).

  Dans le programme, il spécifie la scène à charger et les corps à simuler. Il peut choisir ensuite les paires de corps qui peuvent entrer en collision (à défaut tous les corps sont pris). Ces paires sont envoyées au module de détection de collision. Il ajoute ensuite à l'objet de simulation les corps qu'il veut simuler, et s'il veut interagir avec l'environnement virtuel il ajoute également le dispositif d'interaction. Il peut initialiser aussi le contrôleur s'il veut contrôler un corps par des

1. *Extensible Markup Language* : format de fichier utilisé pour stocker des données de type texte structuré en champs arborescents.
2. *Virtual Reality Modeling Language* : format de fichier utilisé pour représenter une scène virtuelle, en particulier inclut des listes de coordonnées de points et des informations de couleurs des objets.

```
<Universe>
  <MultiBody id="0">
    <File>AFResources/xml/robot_vrml.xml</File>
    <Rotation angle="0.0" x="0.0" y="1.0" z="0.0"/>
    <Translation x="-0.6" y="0.7" z="0.725" />
  </MultiBody>

  <Body id="0">
    <File>AFResources/vrml/decor_vrml/groundBlueLight.wrl</File>
    <Mass>0.0</Mass>
    <CoM>0.0 0.0 0.0</CoM>
    <Inertia>0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0 0.0</Inertia>
    <Translation x="1.0" y="1.0" z="0.02" />
    <Label>Ground1</Label>
  </Body>

  <Configurations>
    <Camera>
      <Rotation angle="120.0" x="-0.577" y="-0.577" z="-0.577"/>
      <Translation x="0.0" y="1.0" z="4.0" />
    </Camera>

    <Light>
      <LightAttribute type="position" x="0.5" y="0.5" z="0.1" w="0.0" />
      <LightAttribute type="ambient" r="0.15" g="0.15" b="0.25" a="1.0" />
      <LightAttribute type="diffuse" r="0.95" g="0.95" b="1.0" a="1.0" />
      <LightAttribute type="specular" r="1.0" g="1.0" b="1.0" a="1.0" />
    </Light>

    <Light>
      <LightAttribute type="position" x="0" y="-5" z="3.0" w="0.0" />
      <LightAttribute type="ambient" r="0.2" g="0.2" b="0.6" a="1.0" />
      <LightAttribute type="diffuse" r="0.825" g="0.66" b="0.165" a="1.0" />
      <LightAttribute type="specular" r="1.0" g="1.0" b="1.0" a="1.0" />
    </Light>

  </Configurations>
</Universe>
```

figure 3.2: *Exemple de fichier XML.*

lois spécifiques,

- la boucle de simulation : il fait appel aux différents modules présentés plus haut. S'il veut faire suivre une trajectoire à un corps poly-articulé, il inclue cette trajectoire dans cette boucle. Un pas de temps est défini par défaut, il peut être modifié manuellement dans cette partie du programme,
- la fin de la simulation : la mémoire est libérée.

Après la compilation, une librairie dynamique est créée et est chargée par l'utilisateur lors du lancement de l'application pour démarrer la simulation.

Cette architecture est utilisable aussi bien sous Windows que sous Linux Ubuntu. L'affichage de la scène se fait grâce à la librairie graphique OpenGL et constitue un *thread*[3] à part entière, la simulation dynamique constituant un autre *thread*. L'affichage de l'interface de simulation se fait grâce à la librairie wxWidgets[4]. L'utilisateur n'est cependant pas obligé de connaître cette librairie pour utiliser AMELIF. Les potentialités et les extensions possibles de l'architecture d'AMELIF sont détaillées dans [Evrard *et al.*, 2008].

Nous présentons maintenant le contenu du module de dynamique, en particulier le calcul des efforts de contact et de collision.

3. En français, processus léger, c'est la duplication de l'espace code d'un programme sans duplication de l'espace de données. De fait, plusieurs portions de code peuvent tourner en parallèle en partageant le même espace de données.

4. `http://www.wxwidgets.org/`

figure 3.3: *Fenêtre d'application d'AMELIF.*

3.1.2 Modèle d'impact

Nous reprenons ici les travaux de Ruspini et Khatib [Ruspini et Khatib, 1999]. On rappelle que dans leurs travaux, les frottements ne sont pas considérés. Pour calculer les efforts d'impact, normaux au contact, nous travaillons en vitesse. La loi d'impact est aussi non régulière. En effet, lors d'un choc, pour éviter que deux corps ne s'interpénètrent, il faut respecter les conditions complémentaires suivantes :

– ou bien la vitesse relative des points de contact est nulle et alors l'impact $\mathbf{f}_i$ doit être positif,

– ou bien la vitesse relative des points de contact est non nulle et alors l'impact $\mathbf{f}_i$ doit être nul.

De plus, l'impact est défini par :

$$\mathbf{f}_i = \mathbf{\Lambda}^{-1} \left(\mathbf{v}^+ - \mathbf{v}^- \right) \tag{3.1}$$

où $\mathbf{\Lambda}$ est la matrice d'inertie dans l'espace opérationnel, présentée dans le chapitre précédent et définie par

$$\mathbf{\Lambda} = \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T \tag{3.2}$$

avec $\mathbf{J}$ la Jacobienne aux points de contact, $\mathbf{v}^-$ et $\mathbf{v}^+$ sont les vitesses des points de contact avant et après collision respectivement. Par ailleurs, pour un point de contact actif j (l'impact est non nul), on utilise la loi d'impact de Newton reliant $\mathbf{v}^-$ et $\mathbf{v}^+$:

$$\mathbf{v}_j^+ = -e\mathbf{v}_j^- \tag{3.3}$$

où e est le coefficient de restitution, compris entre 0 et 1 et déjà défini dans le chapitre précédent. Pour que cette loi soit valable, il faut contraindre le mouvement à tous les points de contact restants. On a alors pour l'ensemble des points :

$$\mathbf{v}^+ \geq -e\mathbf{v}^- \tag{3.4}$$

Cette contrainte prend en compte tous les points de contact, actifs et non-actifs. Si un point n'est pas actif (l'impact est nul), alors sa vitesse doit être au moins aussi élevée que si le point était actif. Les contraintes sur le système s'écrivent alors [Studer, 2008] :

$$0 \leq \mathbf{f}_i \perp \mathbf{v}^+ + e\mathbf{v}^- \geq 0 \tag{3.5}$$

En injectant l'équation 3.4 dans l'équation 3.1, nous obtenons :

$$\mathbf{\Lambda f}_i \geq -(1+e)\mathbf{v}^- \tag{3.6}$$

Considérant les conditions complémentaires précédentes, on obtient l'équation suivante à résoudre :

$$0 \leq \mathbf{f}_i \perp \mathbf{\Lambda f}_i + (1+e)\mathbf{v}^- \geq 0 \tag{3.7}$$

La matrice $\mathbf{\Lambda}$ est symétrique et définie positive, le problème peut être résolu avec un algorithme classique de résolution de LCP, par exemple l'algorithme de Lemke [Lloyd, 2005, Anitescu et Potra, 2002].

Une fois les efforts d'impact calculés, le système est mis à jour par :

$$\mathbf{v}^+ = \mathbf{\Lambda f}_i + \mathbf{v}^- \tag{3.8}$$

et comme $\mathbf{v} = \mathbf{J}\dot{\mathbf{q}}$, les vitesses articulaires sont pareillement mises à jour :

$$\dot{\mathbf{q}}^+ = \mathbf{M}^{-1}\mathbf{J}^T\mathbf{f}_i + \dot{\mathbf{q}}^- \tag{3.9}$$

3.1.3 Modèle de contact

De la même manière que pour le modèle d'impact et sans considérer pour le moment les frottements, lorsqu'il y a contact, les conditions à respecter pour éviter que deux corps ne s'interpénètrent sont :

- ou bien l'accélération relative des points de contact est nulle et alors la force de contact $\mathbf{f}_c$, on parle ici de sa composante normale au contact, doit être positive,
- ou bien l'accélération relative des points de contact est non nulle et alors la force de contact normale doit être nulle.

Nous avons déjà montré ces conditions dans le chapitre précédent. Ici, contrairement au modèle d'impact, les contraintes sont exprimées en accélération.

L'équation à résoudre est :

$$0 \leq \mathbf{f}_c \perp \mathbf{\Lambda f}_c + \mathbf{a}_{\text{libre}} \geq 0 \tag{3.10}$$

où $\mathbf{a}_{\text{libre}}$ est l'accélération relative libre des points de contact, c'est-à-dire issue du calcul de la dynamique sans contraintes (elle traduit l'accélération qu'auraient eue les points choisis pour traduire le contact s'ils n'étaient pas contraints, en d'autres termes, s'il n'y avait pas de contact) :

$$\mathbf{a}_{\text{libre}} = \mathbf{J}\mathbf{M}^{-1}(\mathbf{\Gamma} - \mathbf{b} - \mathbf{g}) + \dot{\mathbf{J}}\dot{\mathbf{q}} \tag{3.11}$$

Ce problème se résout aussi avec un algorithme classique de résolution de LCP, comme l'algorithme de Lemke.

Une fois les efforts de contact calculés, les accélérations cartésiennes sont mises à jour par :

$$\mathbf{a} = \mathbf{\Lambda}\mathbf{f}_c + \mathbf{a}_{\text{libre}} \tag{3.12}$$

que nous réécrivons par la suite

$$\mathbf{a} = \mathbf{a}_{\mathbf{f}\neq 0} + \mathbf{a}_{\mathbf{f}=0} \tag{3.13}$$

avec

$$\mathbf{a}_{\mathbf{f}\neq 0} = \mathbf{\Lambda}\mathbf{f}_c \text{ et } \mathbf{a}_{\mathbf{f}=0} = \mathbf{a}_{\text{libre}} \tag{3.14}$$

En utilisant la relation $\mathbf{a} = \mathbf{J}\ddot{\mathbf{q}} + \dot{\mathbf{J}}\dot{\mathbf{q}}$, les accélérations articulaires sont mises à jour par :

$$\ddot{\mathbf{q}} = \ddot{\mathbf{q}}_{\mathbf{f}\neq 0} + \ddot{\mathbf{q}}_{\mathbf{f}=0} \tag{3.15}$$

avec

$$\ddot{\mathbf{q}}_{\mathbf{f}\neq 0} = \mathbf{M}^{-1}\mathbf{J}^T\mathbf{f}_c \text{ et } \ddot{\mathbf{q}}_{\mathbf{f}=0} = \mathbf{M}^{-1}(\mathbf{\Gamma} - \mathbf{b} - \mathbf{g}) \tag{3.16}$$

Ici, nous étendons l'approche de Ruspini et Khatib en incluant les frottements de Coulomb. La loi de frottement est écrite en vitesses et notre problème est exprimé en accélérations. Il est possible de garder une formulation en vitesses mais il peut y avoir des cas où il n'y a pas de solutions comme indiqué dans le chapitre précédent. Nous écrivons donc notre problème en vitesses en intégrant l'équation 3.10. Par exemple, un schéma simple d'intégration d'Euler explicite produirait alors :

$$\mathbf{v}^{t+\delta t} = \delta t\mathbf{a} + \mathbf{v}^t \tag{3.17}$$

L'équation 3.10 s'écrit alors :

$$0 \leq \mathbf{f}_c \perp (\delta t\mathbf{\Lambda})\mathbf{f}_c + (\delta t\mathbf{a}_{\mathbf{f}=0} + \mathbf{v}^t) \geq 0 \tag{3.18}$$

où δt est le pas d'intégration et $\mathbf{v}^t$ est la vitesse relative des points de contact à l'instant présent t.

En utilisant les équations 3.13, 3.15 et 3.17, les mises à jour des vitesses cartésiennes et articulaires sont alors respectivement :

$$\mathbf{v}^{t+\delta t} = (\delta t\mathbf{\Lambda})\mathbf{f}_c + (\delta t\mathbf{a}_{\mathbf{f}=0} + \mathbf{v}^t) \tag{3.19}$$

$$\dot{\mathbf{q}}^{t+\delta t} = \left(\delta t\mathbf{M}^{-1}\mathbf{J}^T\right)\mathbf{f}_c + (\delta t\ddot{\mathbf{q}}_{\mathbf{f}=0} + \dot{\mathbf{q}}^t) \tag{3.20}$$

avec $\dot{\mathbf{q}}^t$ la vitesse articulaire à l'instant présent.

3.1.4 Dynamique avec contact

3.1.4.1 Calcul de la matrice $\mathbf{\Lambda}$

Pour pouvoir résoudre les équations 3.7 et 3.18, on a besoin de déterminer la matrice $\mathbf{\Lambda}$ et les vitesses relatives des points de contact. Les vitesses relatives sont faciles à déterminer si l'on connaît les vitesses des corps en contact. Il suffit de projeter ces vitesses dans l'espace de contact et d'en faire la différence algébrique.

Pour déterminer $\mathbf{\Lambda}$, il est possible de calculer de manière brute $\mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T$ – c'est-à-dire, la Jacobienne et sa transposée, la matrice d'inertie puis son inverse –, mais cette méthode est d'une complexité élevée et donc inefficace pour les simulations interactives que nous désirons réaliser. Nous présentons une méthode simple pour calculer cette matrice ; nous verrons dans la partie 3.2 d'autres méthodes plus rapides. Nous partons de l'équation de la dynamique :

$$\ddot{\mathbf{q}} = \ddot{\mathbf{q}}_{\mathbf{f}=0} + \ddot{\mathbf{q}}_{f\neq 0} = \mathbf{M}^{-1}(\mathbf{\Gamma} - \mathbf{b} - \mathbf{g}) + \mathbf{M}^{-1}\mathbf{J}^T\mathbf{f}_c \tag{3.21}$$

Considérons un instant qu'il n'y a ni couple articulaire ($\mathbf{\Gamma} = 0$), ni vitesse articulaire ($\dot{\mathbf{q}} = 0$), ni gravité ($\mathbf{g} = 0$). Alors l'accélération articulaire libre devient nulle ($\ddot{\mathbf{q}}_{\mathbf{f}=0} = \mathbf{M}^{-1}(\mathbf{\Gamma} - \mathbf{b} - \mathbf{g}) = 0$). L'équation 3.21 devient :

$$\ddot{\mathbf{q}} = \ddot{\mathbf{q}}_{\mathbf{f}\neq 0} = \mathbf{M}^{-1}\mathbf{J}^T\mathbf{f}_c \tag{3.22}$$

Considérons alors une perturbation de force unitaire en chaque point de contact ($\mathbf{f}_c = 1$). En réutilisant l'algorithme de Featherstone et en supposant cette force unitaire comme une force extérieure connue, nous obtenons de nouvelles accélérations articulaires :

$$\ddot{\mathbf{q}}_{\mathbf{f}=1} = \mathbf{M}^{-1}\mathbf{J}^T \tag{3.23}$$

En utilisant la relation $\mathbf{a} = \mathbf{J}\ddot{\mathbf{q}} + \dot{\mathbf{J}}\dot{\mathbf{q}}$ et n'oubliant pas qu'on a considéré les vitesses articulaires nulles, nous en déduisons les accélérations cartésiennes :

$$\mathbf{a}_{\mathbf{f}=1} = \mathbf{J}\ddot{\mathbf{q}}_{\mathbf{f}=1} = \mathbf{J}\mathbf{M}^{-1}\mathbf{J}^T = \mathbf{\Lambda} \tag{3.24}$$

Notons que, dans cette dernière équation, $\mathbf{a}_{\mathbf{f}=1}$ n'est pas un vecteur mais une matrice (voir partie 3.2.1 et algorithme 3).

3.1.4.2 Résolution des forces de contact avec frottement

Une fois la matrice $\mathbf{\Lambda}$ et les vitesses relatives des points de contact calculées, les forces de contact sont résolues en utilisant l'algorithme de Gauss-Seidel. Grâce à cette approche, le cône de frottement ne nécessite pas de discrétisation. Ici on utilise l'approche introduite par [Alart et Curnier, 1991, Alart, 1993, Jean, 1993] et utilisée dans [Duriez, 2004, Acary et Perignon, 2007], qui se sert d'une méthode de Newton pour résoudre précisément les forces de contact avec frottement en chaque point de

contact.

On rappelle l'équation du contact dans l'espace opérationnel :

$$\mathbf{v}^{t+\delta t} = (\delta t \mathbf{\Lambda})\mathbf{f}_c + (\delta t \mathbf{a}_{\mathbf{f}=0} + \mathbf{v}^t) \tag{3.25}$$

que nous allons écrire pour simplifier la notation :

$$\mathbf{v} = \mathbf{W}\mathbf{f}_c + \mathbf{v}_{\text{libre}} \tag{3.26}$$

avec $\mathbf{W} = \delta t \mathbf{\Lambda}$, $\mathbf{v}_{\text{libre}} = \delta t \mathbf{a}_{\mathbf{f}=0} + \mathbf{v}^t$. En décomposant les différents éléments, nous avons :

$$\begin{pmatrix} v_n \\ v_{t_1} \\ v_{t_2} \end{pmatrix} = \begin{bmatrix} W_{nn} & W_{nt_1} & W_{nt_2} \\ W_{t_1 n} & W_{t_1 t_1} & W_{t_1 t_2} \\ W_{t_2 n} & W_{t_2 t_1} & W_{t_2 t_2} \end{bmatrix} \begin{pmatrix} f_n \\ f_{t_1} \\ f_{t_2} \end{pmatrix} + \begin{pmatrix} v_n^{\text{libre}} \\ v_{t_1}^{\text{libre}} \\ v_{t_2}^{\text{libre}} \end{pmatrix} \tag{3.27}$$

On distingue alors trois cas : le cas de non-contact, le cas d'adhérence et le cas de glissement. La distinction entre les trois cas est montrée plus clairement dans l'algorithme 1. Dans le premier cas, la solution est évidente : on a $v_n^{\text{libre}} \geq 0$ donc $\mathbf{f}_c = 0$ par définition du problème de complémentarité. Le deuxième cas est aussi trivial : $\mathbf{v} = 0$ donc $\mathbf{f}_c = -\mathbf{W}^{-1}\mathbf{v}_{\text{libre}}$. Dans le cas du glissement, les solutions sont moins évidentes à obtenir. On bâtit alors une application φ définie de la manière suivante :

$$\begin{cases} \varphi_1(\mathbf{v}, \mathbf{f}_c) = v_n - v_n^{\text{libre}} - W_{nn} f_n - \mathbf{W}_{nt}\mathbf{f}_t \\ \varphi_2(\mathbf{v}, \mathbf{f}_c) = \mathbf{v}_t - \mathbf{v}_t^{\text{libre}} - \mathbf{W}_{tn} f_n - \mathbf{W}_{tt}\mathbf{f}_t \\ \varphi_3(\mathbf{v}, \mathbf{f}_c) = f_n - \text{proj}_n(f_n - \rho_n v_n) \\ \varphi_4(\mathbf{v}, \mathbf{f}_c) = \mathbf{f}_t - \text{proj}_{\mathcal{C}}(\mathbf{f}_t - \rho_t \mathbf{v}_t) \end{cases} \tag{3.28}$$

avec $\mathcal{C}$ le cône de frottement, et ρ_n et ρ_t des constantes choisies de manière à rendre respectivement contractantes les applications [5] [Alart, 1993] :

$$f_n \to f_n - \rho_n v_n, \ \mathbf{f}_t \to \mathbf{f}_t - \rho_t \mathbf{v}_t \tag{3.29}$$

et les valeurs qui rendent ces applications contractantes avec le meilleur coefficient de contraction sont :

$$\rho_n = \frac{1}{W_{nn}}, \ \rho_t = \frac{\lambda_{\min}}{\lambda_{\max}^2} \tag{3.30}$$

avec $\lambda_{\min}$ et $\lambda_{\max}$ les valeurs propres de $\mathbf{W}_{tt}$.

Les trois premières lignes de cette application correspondent à l'équation 3.27. Les trois suivantes correspondent aux contraintes imposées, d'une part la loi de Signorini, d'autre part la loi de Coulomb.

5. Une application contractante ϕ définie d'un ensemble non vide E dans lui-même est une application k-lipschitzienne avec $0 < k < 1$. D'après le théorème du point fixe, une telle application admet un unique point fixe et la suite x, $\phi(x)$, $\phi(\phi(x))$, ... avec $x \in E$ converge vers ce point fixe. La propriété de contraction nous intéresse dans le cas de la résolution du contact car nous voulons que la méthode converge rapidement vers une solution.

Dans le cas où il y a contact et glissement respectivement :

$$f_n - \rho_n v_n > 0 \implies \varphi_3(\mathbf{v}, \mathbf{f}_c) = \rho_n f_n \tag{3.31}$$

$$\mathbf{f}_t - \rho_t \mathbf{v}_t \notin \mathcal{C} \implies \varphi_4(\mathbf{v}, \mathbf{f}_c) = \mathbf{f}_t - \mu f_n \frac{\mathbf{f}_t - \rho_t \mathbf{v}_t}{\|\mathbf{f}_t - \rho_t \mathbf{v}_t\|} \tag{3.32}$$

Il suffit de chercher un couple $(\mathbf{v}, \mathbf{f}_c)$ tel que $\varphi(\mathbf{v}, \mathbf{f}_c) = 0$ pour résoudre le problème.

On notera que cette application φ est continue différentiable presque partout et qu'une méthode de Newton peut s'appliquer pour traiter le problème [Alart, 1993]. On définit alors la dérivée $\partial\varphi$:

$$\begin{array}{llll}
\frac{\partial\varphi_1}{\partial v_n} = I & \frac{\partial\varphi_1}{\partial \mathbf{v}_t} = 0 & \frac{\partial\varphi_1}{\partial f_n} = -W_{nn} & \frac{\partial\varphi_1}{\partial \mathbf{f}_t} = -\mathbf{W}_{nt} \\
\frac{\partial\varphi_2}{\partial v_n} = 0 & \frac{\partial\varphi_2}{\partial \mathbf{v}_t} = \mathbf{I} & \frac{\partial\varphi_2}{\partial f_n} = -\mathbf{W}_{tn} & \frac{\partial\varphi_2}{\partial \mathbf{f}_t} = -\mathbf{W}_{tt} \\
\frac{\partial\varphi_3}{\partial v_n} = \rho_n & \frac{\partial\varphi_3}{\partial \mathbf{v}_t} = 0 & \frac{\partial\varphi_3}{\partial f_n} = 0 & \frac{\partial\varphi_3}{\partial \mathbf{f}_t} = 0 \\
\frac{\partial\varphi_4}{\partial v_n} = 0 & \frac{\partial\varphi_4}{\partial \mathbf{v}_t} = \rho_t \mu f_n M(\mathbf{f}_t - \rho_t \mathbf{v}_t) & \frac{\partial\varphi_4}{\partial f_n} = -\mu \frac{\mathbf{f}_t - \rho_t \mathbf{v}_t}{\|\mathbf{f}_t - \rho_t \mathbf{v}_t\|} & \frac{\partial\varphi_4}{\partial \mathbf{f}_t} = \mathbf{I} - \mu f_n M(\mathbf{f}_t - \rho_t \mathbf{v}_t)
\end{array} \tag{3.33}$$

où M est la dérivée de $V = \frac{V}{\|V\|}$.

Algorithme 1 : Algorithme de calcul des efforts de contact par l'approche de Gauss-Seidel.

Données : $\ddot{\mathbf{q}}_{\mathbf{f}=0}$ (ou $\dot{\mathbf{q}}_{\mathbf{f}=0}$), $\mathbf{W}$, $\mathbf{v}_{\text{libre}}$
Résultat : f_c
début
  tant que *(non convergence)* **faire**
    pour *chaque contact* $i = 1$ **à** m **faire**
      $\mathbf{v}_i$ = GaussSeidelSolver()
      si $v_{in} = 0$ **alors**
        $\mathbf{f}_i = -\mathbf{W}_i^{-1}\mathbf{v}_i$ (on suppose qu'on est en adhérence)
        vérifier $\|\mathbf{f}_{it}\| \leq \mu f_{in}$
        si $\|\mathbf{f}_{it}\| > \mu f_{in}$ **alors**
          Cas de glissement
          $\mathbf{f}_i$ = NewtonContactSolver()
      sinon
        Cas de non-contact
        $\mathbf{f}_i = 0$
  pour *chaque corps* $i = 1$ **à** n **faire**
    $\ddot{\mathbf{q}}_i = \ddot{\mathbf{q}}_{\mathbf{f}=0\ i} + \mathbf{M}^{-1}\mathbf{J}^T\mathbf{f}_i$ ou $\dot{\mathbf{q}}_i = \dot{\mathbf{q}}_{\mathbf{f}=0\ i} + \delta t \mathbf{M}^{-1}\mathbf{J}^T\mathbf{f}_i + \dot{\mathbf{q}}_i^t$
fin

Lorsqu'on est en contact, on suppose qu'on est en adhérence. Si le cas d'adhérence n'est pas vérifié, on traite ce cas par la méthode de Newton [Duriez, 2004]. On s'arrange également pour démarrer la méthode de Newton avec une valeur de force proche de la solution (par exemple en utilisant la valeur du pas de temps précédent). En effet, la méthode de Newton a l'avantage de converger très rapidement si la valeur initiale est proche de la solution. On réduit ainsi le temps de calcul.

Le critère de convergence utilisé dans la résolution par la méthode de type Gauss-Seidel est la convergence relative sur l'ensemble des points, qui est plus commode dans le cas où on a beaucoup de

points de contact :

$$\sum_{i=1}^{m} \frac{\|\mathbf{f}_i^{k+1} - \mathbf{f}_i^k\|}{\|\mathbf{f}_i^{k+1}\|} < \epsilon \tag{3.34}$$

où k est l'itération courante et $k+1$ la suivante.

Algorithme 2 : Algorithme de calcul des efforts de contact en un point avec la méthode de Newton (NewtonContactSolver).

Données : $\mathbf{W}_i$, $\mathbf{v}_i^{\text{libre}}$, $\mathbf{f}_i$
Résultat : $\mathbf{v}_i$, $\mathbf{f}_i$
début
  $(\lambda_{\min}, \lambda_{\max}) = \text{ValeursPropres}(\mathbf{W}_{itt})$
  $\rho_n = 1/W_{inn}$
  $\rho_t = \lambda_{\min}/\lambda_{\max}^2$
  tant que $\|\varphi(\mathbf{v}_i, \mathbf{f}_i)\| \neq 0$ **faire**
    construire φ
    construire $\partial\varphi$
    $\begin{pmatrix} \mathbf{v}_i \\ \mathbf{f}_i \end{pmatrix} = \begin{pmatrix} \mathbf{v}_i \\ \mathbf{f}_i \end{pmatrix} - \partial\varphi(\mathbf{v}_i, \mathbf{f}_i)^{-1}\varphi(\mathbf{v}_i, \mathbf{f}_i)$
fin

L'algorithme global est résumé en figure 3.4.

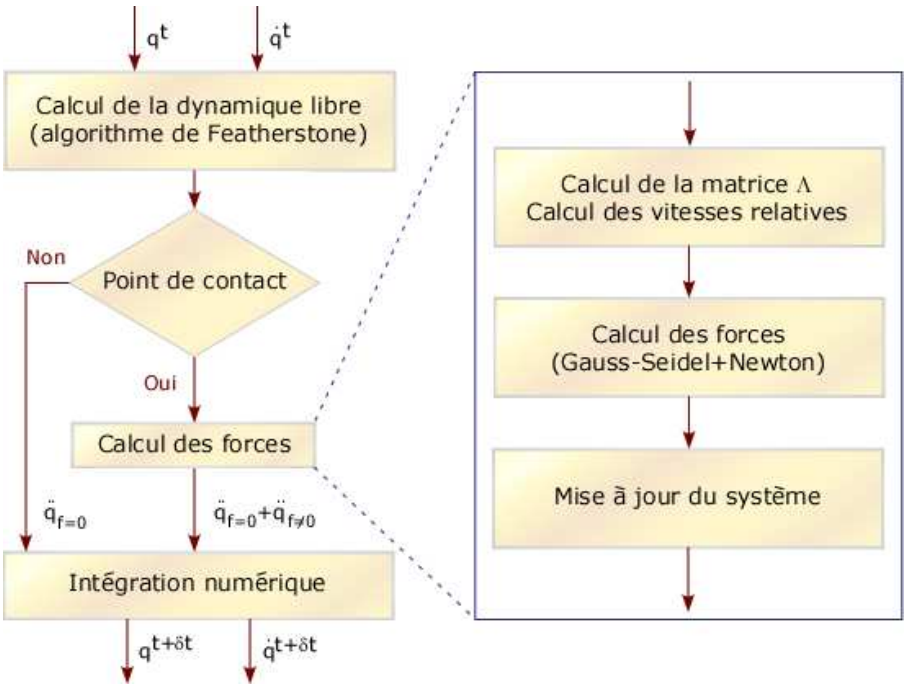


figure 3.4: *Algorithme global de la simulation dynamique.*

3.1.5 Détection de collision

Dans les problèmes de contact avec frottement, la détection de collision (ou plus exatement de proximité) est un composant important : c'est ce module qui désigne les paires de points de contact

potentiels sur chaque objet. Il existe de nombreux algorithmes de détection de collision, et nous avons choisi PQP [Larsen *et al.*, 1999] pour sa rapidité d'intégration et sa facilité d'utilisation dans notre simulateur. En donnant le maillage des corps, il est possible de connaître les paires de triangles en collision lorsque les objets s'interpénètrent et surtout la distance minimale séparant deux corps ainsi que la paire de points (un sur chaque objet) les plus proches. On peut utiliser la détection de collision si l'on tolère une légère pénétration. Dans ce cas, une fois les paires de triangles connues, il faut trouver les points de contact en calculant l'intersection des triangles, grâce à des algorithmes appropriés comme celui de Möller [Möller, 1997]. Toutefois, l'utilisation de cette méthode entraîne une forte redondance des points de contact et donc implique un travail de tri, ce qui ralentit la simulation. La recherche des corps en collision est une caractéristique commune à PQP et à Opcode [6], la librairie de détection de collision de Pierre Terdiman, initialement implémentée dans AMELIF (et aussi OpenHRP2). Cette dernière librairie n'inclut cependant pas le calcul de la distance minimale entre les corps. A titre de comparaison des deux librairies, nous avons fait des tests avec des objets simples (un cube sur un plan) et nous avons remarqué que PQP nous donnait des points plus plausibles relativement à Opcode, sans gain de performance.

Pour l'impact, comme la simulation dynamique se fait dans le domaine discret, il est nécessaire de considérer l'état des points de contact à deux instants successifs t_0 et $t_0 + \delta t$, où δt est le pas d'échantillonnage. Un impact peut se produire entre ces deux instants, à un temps qui doit être déterminé pour que les forces d'impact soient appliquées correctement aux points de contact. On peut pour cela utiliser les méthodes de détection de collision 4D (c'est-à-dire temporelles) qui rendent aussi le premier temps de collision entre deux pas discrets [Redon *et al.*, 2002], ce qui permet d'avoir le premier temps d'impact. Elles procèdent par une interpolation temporelle *a priori* du mouvement entre deux pas discrets ; toutefois elles sont plus coûteuses en temps de calcul et n'existent pratiquement pas en librairie libre. Mais le mouvement exact des corps est inconnu entre deux instants d'échantillonnage. Si on prend un petit pas d'échantillonnage (1ms), on peut considérer que le temps d'impact est t_0. Dans notre cas, nous calculerons toutes les forces d'interaction (impact ou force maintenue) à l'instant t_0.

3.1.6 Applications

Nous montrons deux scénarios de démonstration en simulation. Toutes les simulations ont été réalisées sur un ordinateur fonctionnant sous Windows avec un processeur à $2,5$GHz. Nous avons pris un pas de temps de 1ms et nous avons utilisé la méthode d'intégration explicite d'Euler. Nous avons pris le robot humanoïde HRP-2 de Kawada Industries comme personnage de nos simulations. Ici nous ne cherchons pas à comparer la simulation avec la réalité, parce que les simulations que nous présentons ici sont difficilement réalisables sur le robot réel pour des raisons de sécurité ; nous traiterons cette partie dans le dernier chapitre. Nous voulons montrer ici que nous pouvons traiter des cas multi-contacts et ainsi réaliser des simulations complexes mettant en scène des systèmes complexes dans des environnements complexes avec des scénarios réalistes.

6. `http://www.codercorner.com/Opcode.htm`

3.1.6.1 Exemples de simulation

Le premier exemple consiste à attraper un objet posé loin du robot sur une table (voir figure 3.5). Pour ce faire, le robot se penche en avant vers la table. Il utilise une de ses mains comme support sur la table pour qu'il ne puisse pas glisser et tomber. Cet exemple est en fait inspiré de la thèse d'Adrien Escande qui consiste à planifier des points d'appuis pour faire des mouvements acycliques [Escande, 2008]. OpenHRP ne permet pas de simuler HRP-2 en mouvements acycliques en multicontacts avec ce type de scénarios. Le coefficient de frottement est de $0,4$ entre chaque objet de la scène.

figure 3.5: *Le robot HRP-2 attrape un objet sur une table.*

Comme le montre la figure 3.5, un des mouvements générés par le planificateur permet de se rendre compte que lorsque le robot se penche en avant, ses pieds décollent de l'estrade et glissent vers l'arrière. Ce glissement s'arrête lorsque les jambes du robot entrent en contact avec la table. Le robot est suffisamment proche de celle-ci pour que ses pieds ne puissent pas glisser davantage. Cependant, grâce aux points d'appui sur la table avec sa main, le robot reste globalement stable et ne pivote pas sur le côté. Notre simulateur a donc permis de se rendre compte que le mouvement généré peut être dangereux dans certaines étapes de l'exécution de cette tâche, et un autre mouvement plus satisfaisant a donc été généré, simulé, puis porté sur le robot.

Dans le second exemple de simulation, nous demandons au robot de transporter une grosse boîte posée en travers sur ses bras d'un endroit à un autre (voir figure 3.6). En cours de route, le robot perd l'équilibre et tombe sur le côté. La boîte tombe alors aussi sur le sol ; donc les modèles de marche générés ne sont pas corrects (évidemment, nous n'avons pas porté cette expérience sur le robot).

3.1.6.2 Temps de calcul

Nous avons réalisé des tests de performances en prenant un plus grand nombre de points de contact (plus de 50 points) et nous avons fait une comparaison entre différentes approches des méthodes par contraintes (LCP et itérative). En particulier, avec 67 points de contact, la dimension de la matrice $\mathbf{\Lambda}$ est de taille $67 \times 3 = 201$ avec une approche itérative, alors qu'en gardant une approche LCP, en discrétisant le cône avec huit facettes, nous obtenons une matrice de taille $67 \times (1 + 8 + 1) = 670$. Nous voyons bien l'écart important d'un facteur d'environ huit entre les différentes approches, ce qui induit une augmentation sensible du temps de calcul. L'intérêt d'utiliser une approche itérative de

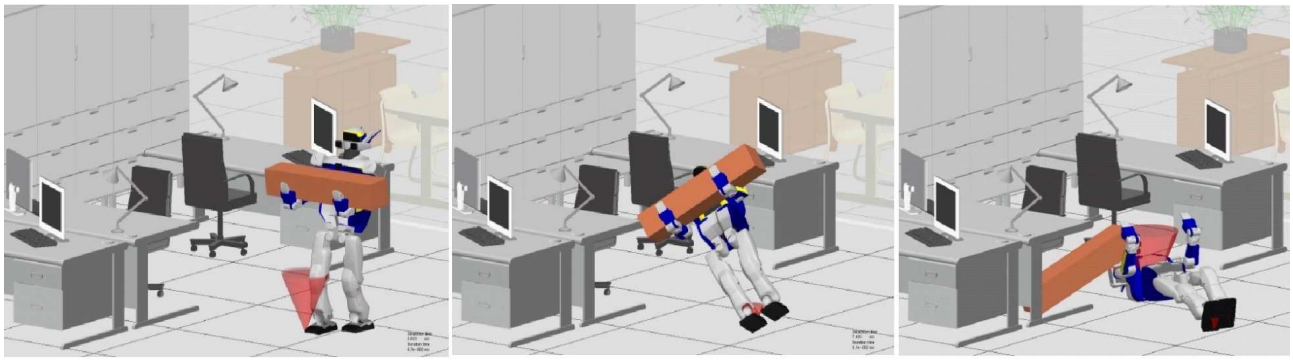


figure 3.6: *Le robot HRP-2 porte une boîte lourde en marchant et perd l'équilibre.*

type Gauss-Seidel pour des simulations interactives est clair. Bien sûr, avec la discrétisation des cônes de frottement on garde une formulation linéaire du problème et on résoud un problème linéaire alors qu'avec la formulation conique analytique des frottements, on résout un problème de petite taille mais non linéaire. L'avantage du second, en réalité, tient plus à la possibilité de régler la précision et le temps consacré à la résolution du problème, qu'à sa taille. En effet, la condition d'arrêt de l'algorithme est lorsque la variation de la solution est plus petite qu'une borne que l'on se fixe, on peut donc sacrifier la précision de la solution vis-à-vis du temps d'exécution, notamment lorsque les aspects temps réels priment (simulation interactive).

Par ailleurs, nous montrons en figure 3.7 le temps moyen pris par le micro-processeur pour traiter un pas de temps de la simulation en fonction du nombre de points de contact (qui donne une tendance de la complexité algorithmique).

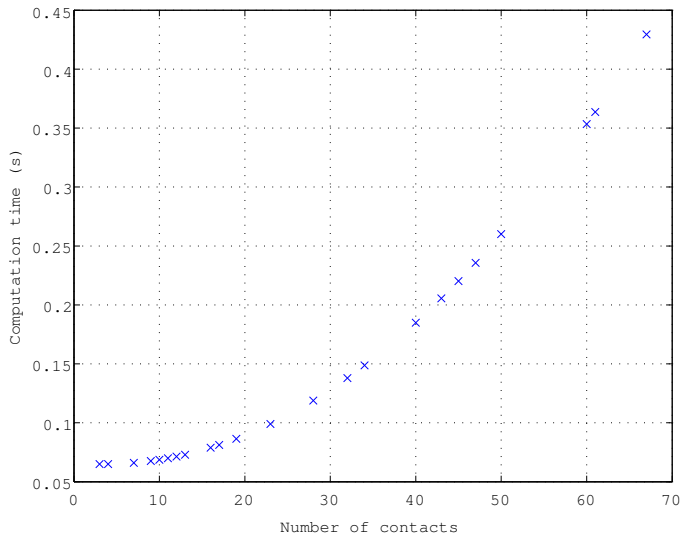


figure 3.7: *Temps moyen de calcul d'un pas de temps en fonction du nombre de points de contact.*

Nous voyons que le temps de calcul augmente de manière quasi-quadratique avec le nombre de points de contact. Plus le nombre de points de contact augmente, moins nous pouvons être proches

du temps réel et par conséquent nous ne pouvons pas prétendre faire véritablement de l'interaction haptique. Il est donc nécessaire d'optimiser le calcul des forces de contact.

En particulier, lorsque nous regardons la répartition du temps de calcul des différents modules de simulation sur un pas de temps (voir figure 3.8), nous nous apercevons que la détection de collision est la partie la plus coûteuse, suivie du calcul des forces de contact. Le problème de la détection de collision est toujours d'actualité dans le domaine de l'animation graphique, des solutions orientées hardware sont prometteuses de solutions définitives à ce problème, dans un futur qui ne nous paraît pas si lointain.

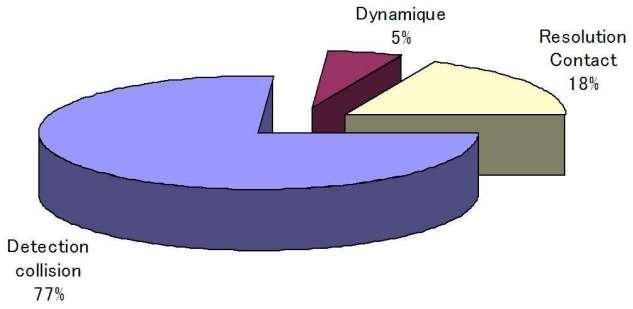


figure 3.8: *Répartition du temps de calcul des différents modules du simulateur.*

En revanche nous pouvons améliorer le temps de calcul des forces de contact. En particulier, la majeure partie du temps est passée dans le calcul de la matrice $\mathbf{\Lambda}$. Nous allons maintenant voir comment nous pouvons accélérer ce calcul.

3.2 Amélioration des calculs du modèle dynamique

Dans la partie précédente, nous nous sommes attachés à optimiser le code pour avoir des temps de calcul très petits, en particulier nous avons fait en sorte de ne travailler qu'avec des vecteurs et des matrices de taille 3 pour lesquels des librairies spécifiques efficaces existent. Ainsi pour la matrice $\mathbf{\Lambda}$ et les vecteurs de forces de contact et de vitesses, nous avons préféré utiliser des tableaux de matrices de taille 3 et des tableaux de vecteurs de taille 3. Cependant, et comme nous l'avons indiqué dans le chapitre précédent, le calcul de l'opérateur de Delassus est très coûteux, en particulier dans les cas où il y a un grand nombre de points de contact. Dans cette partie nous présentons des améliorations algorithmiques du calcul de la dynamique, nous nous focaliserons en particulier sur le calcul de la matrice $\mathbf{\Lambda}$. Nous exposons trois méthodes différentes pour calculer cette matrice et nous montrons leur performance respective : une méthode initiale présentée dans la partie précédente et dans [Chardonnet *et al.*, 2006], une méthode proposée par Chang et Khatib [Chang et Khatib, 2000] et une nouvelle méthode plus rapide que les deux autres et présentée dans [Chardonnet *et al.*, 2008b].

Rappelons d'abord l'équation de la dynamique dans l'espace articulaire :

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}(\mathbf{q})\left(\mathbf{\Gamma}(\mathbf{q}) - \mathbf{b}\left(\mathbf{q}, \dot{\mathbf{q}}\right) - \mathbf{g}(\mathbf{q})\right) + \mathbf{M}^{-1}(\mathbf{q})\mathbf{J}^T(\mathbf{q})\mathbf{f} \tag{3.35}$$

et dans l'espace opérationnel :

$$\mathbf{a} = \mathbf{\Lambda}\mathbf{f} + \mathbf{c} \tag{3.36}$$

avec on le rappelle $\mathbf{\Lambda} = \mathbf{J}(\mathbf{q})\mathbf{M}(\mathbf{q})^{-1}\mathbf{J}(\mathbf{q})^T$ et $\mathbf{c} = \mathbf{J}(\mathbf{q})\mathbf{M}^{-1}(\mathbf{q})\left(\mathbf{\Gamma}(\mathbf{q}) - \mathbf{b}\left(\mathbf{q}, \dot{\mathbf{q}}\right) - \mathbf{g}(\mathbf{q})\right) + \dot{\mathbf{J}}(\mathbf{q})\dot{\mathbf{q}}$.

Dans tout ce qui suit, nous garderons les notations suivantes :

- n le nombre de corps du système,
- m le nombre de points de contact,
- C le nombre de corps en contact,
- l'indice $_c$ désigne un point de contact,
- l'indice $_C$ désigne un corps en contact.

La matrice $\mathbf{\Lambda}$ peut se décomposer de la manière suivante :

$$\mathbf{\Lambda} = \mathbf{X}_c\mathbf{J}_C\mathbf{M}^{-1}\mathbf{J}_C^T\mathbf{X}_c^T \tag{3.37}$$

où $\mathbf{X}_c$ est la matrice de transformation de l'origine d'un corps en contact aux points de contact de ce corps, $\mathbf{J}_C$ est la Jacobienne des corps en contact calculée à l'origine de ces corps. De fait, on a $\mathbf{J} = \mathbf{X}_c\mathbf{J}_C$. Par ailleurs nous travaillerons exclusivement dans le repère du monde, ainsi toutes les grandeurs sont exprimées dans ce repère.

3.2.1 Méthode initiale

Dans cette méthode (voir partie 3.1.4.1), le calcul de la matrice $\mathbf{\Lambda}$ est réalisé en deux parties. D'une part on calcule l'accélération des corps en contact en considérant dans l'espace des contacts une force unitaire $\mathbf{f} = 1$ s'exerçant en chaque point de contact, cette force s'écrit dans le repère du monde :

$$\mathbf{f}_{C,c}[y] = (\mathbf{F}_{C,c}[y], \tau_{C,c}[y])^T = ((n_c[y], t_{1c}[y], t_{2c}[y]), \mathbf{x}_c \wedge \mathbf{F}_{C,c}[y])^T \tag{3.38}$$

avec $\mathbf{x}_c$ la position du contact c, $\mathbf{n}_c$, $\mathbf{t}_{1c}$ et $\mathbf{t}_{2c}$ respectivement les normale et tangentes au contact c, y allant de 1 à 3 (correspondant aux trois axes de forces). Puis on effectue $3m$ fois l'algorithme de Featherstone. Le calcul de cette accélération est décomposé en deux boucles, la première allant des effecteurs à la base, la seconde en sens inverse (voir figure 3.9). Cette accélération vaut :

$$\mathbf{a}_{C\ \mathbf{f}=1} = \mathbf{J}_C\mathbf{M}^{-1}\mathbf{J}_C^T\mathbf{X}_c^T \tag{3.39}$$

et est de taille $6 \times 3m$; elle est obtenue par assemblage des accélérations $\mathbf{a}_{C_i\ \mathbf{f}=1}$ de tous les corps en contact C_i. Dans la pratique, on calcule l'accélération de tous les corps, y compris ceux qui ne sont pas en contact, car on se sert de cette accélération dans la mise à jour de la dynamique. D'autre part on

calcule $\mathbf{X}_c$, puis on effectue le produit des deux pour arriver à l'équation 3.24. Il s'agit en fait d'une simple projection dans l'espace des contacts :

$$\mathbf{a}_{\mathbf{f}=1} = \mathbf{X}_c \mathbf{a}_{C\ \mathbf{f}=1} = \mathbf{\Lambda} \tag{3.40}$$

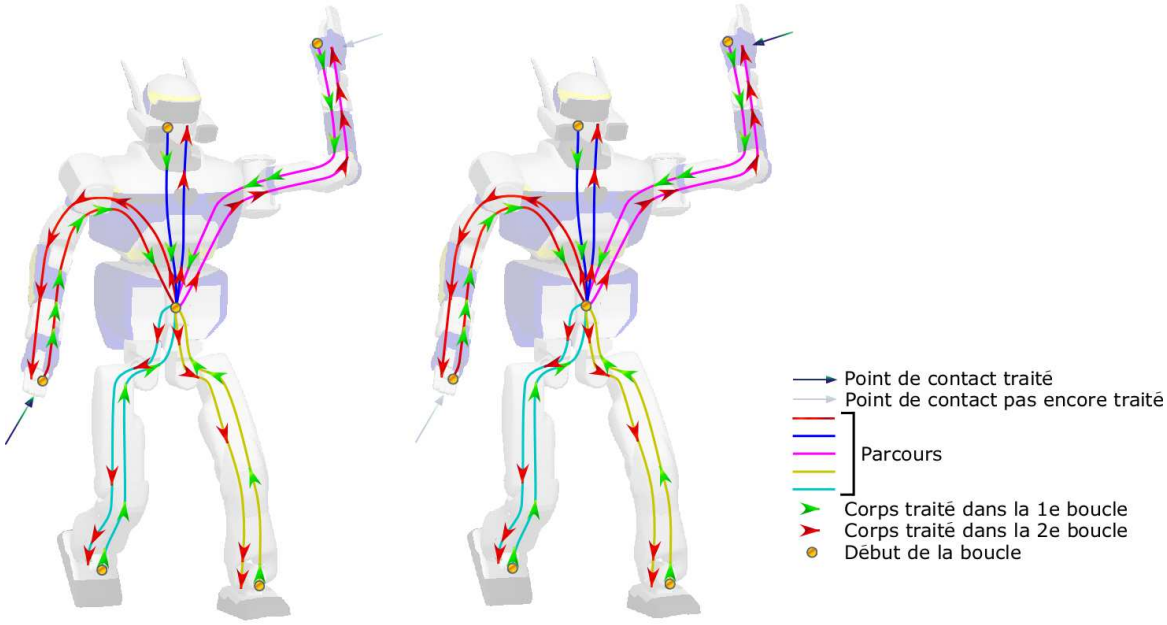


figure 3.9: *Parcours effectué avec la méthode initiale pour le calcul de* $\mathbf{\Lambda}$*. Ici dans le cas où il y a deux points de contact, la figure de gauche correspond au traitement du premier point de contact, celle de droite correspond au traitement du second point de contact. Pour chaque point on visite tous les corps du robot.*

La table 3.1 donne le nombre d'opérations pour le calcul de la matrice d'inertie dans l'espace opérationnel. s désigne le nombre de corps compris entre la base et le corps en contact, $d_{i,j}$ est le nombre d'articulations comprises entre les corps i et j. La complexité de cet algorithme est en $\mathcal{O}(m^2 + nm)$.

TABLE 3.1 – Opérations nécessaires pour calculer $\mathbf{\Lambda}$ avec la méthode initiale.

$\times$ $\div$	$144m$	$+39sm$	$+36\beta_m$	$+27m\frac{m+1}{2}$
$+$ $-$	$99m$	$+36sm$	$+21\beta_m$	$+27m\frac{m+1}{2}$

avec $\beta_m = \sum_{i \leq m} d_{i,0}$, $\beta_m < nm$.

On peut remarquer le grand nombre d'opérations effectuées. En effet on visite tous les corps du système, y compris ceux qui ne sont pas contact et qui ne sont pas dans la branche base-corps en contact. Cette méthode n'est donc pas adaptée aux cas où il y a beaucoup de points de contact ni pour des simulations interactives.

Algorithme 3 : Algorithme de calcul de la matrice $\mathbf{\Lambda}$ avec la méthode initiale. Les notations sont celles de Featherstone : $\mathbf{H} = \mathbf{IS}$ avec $\mathbf{S}$ le vecteur de l'axe de l'articulation, $\mathbf{I}$ l'inertie, et h_i est le parent du corps i.

Données : $\mathbf{f} = 0$, $\mathbf{S}$, $\mathbf{I}$, $\mathbf{H}$, $\mathbf{X}_c$
Résultat : $\mathbf{\Lambda}$
début
  pour *tous les corps i en collision* **faire**
    pour *tous les points j de contact* **faire**
      pour $y = 1$ **à** 3 **faire**
        $\mathbf{F}_{i,j}[y] = (n_j[y], t_{j1}[y], t_{i2}[y])^T$
        $\tau_{i,j}[y] = \mathbf{x}_j \wedge \mathbf{F}_i[y]$
        $\mathbf{f}_{i,j}[y] = (\mathbf{F}_{i,j}[y], \tau_{i,j}[y])^T$
  pour *tous les corps i différents de la base* **faire**
    pour *tous les points j de contact* **faire**
      pour $y = 1$ **à** 3 **faire**
        $\mathbf{U}_i = \mathbf{S}_i f_{i,j}[y]$
        $\mathbf{f}_{h_i,j}[y] = \mathbf{f}_{h_i,j}[y] + \mathbf{f}_{i,j}[y] + \mathbf{H}_i(\mathbf{S}_i^T \mathbf{I}_i \mathbf{S}_i)^{-1} \mathbf{U}_i$
  pour *tous les points j de contact* **faire**
    pour $y = 1$ **à** 3 **faire**
      pour *tous les corps k* **faire**
        $\ddot{q}_{k\ \mathbf{f}_{k,j}} = (\mathbf{S}_k^T \mathbf{I}_k \mathbf{S}_k)^{-1}(\mathbf{U}_k - \mathbf{H}_k^T \mathbf{a}_{h_k\ \mathbf{f}_{k,j}})$
        $\mathbf{a}_{k\ \mathbf{f}_{k,j}} = \mathbf{S}_k \ddot{q}_{k\ \mathbf{f}_{k,j}} + \mathbf{a}_{h_k\ \mathbf{f}_{k,j}}$
  Assembler les $\mathbf{a}_{k\ \mathbf{f}_{k,j}}$ de tous les corps en contact pour former $\mathbf{a}_{C\ \mathbf{f}=1}$
  $\mathbf{\Lambda} = \mathbf{X}_c \mathbf{a}_{C\ \mathbf{f}=1}$
fin

3.2.2 Méthode de Chang et Khatib

Chang et Khatib définissent une matrice $\mathbf{\Omega}$ de taille $6n \times 6n$ qui lie l'accélération spatiale du corps j à la force s'appliquant sur le corps i [Chang et Khatib, 1999, Chang et Khatib, 2000] :

$$\mathbf{a}_j = \begin{matrix} \overline{\mathbf{a}}_j \\ \alpha_j \end{matrix} \Bigg) = \mathbf{\Omega}_{i,j}\mathbf{f}_i \tag{3.41}$$

avec $\overline{\mathbf{a}}$ et α l'accélération linéaire et angulaire respectivement. Cette matrice, calculée de manière récursive, s'écrit, avec les notations de Featherstone [Featherstone, 1987] :

$$\mathbf{\Omega}_{i,j} = \begin{cases} \mathbf{S}_i \left(\mathbf{S}_i^T \mathbf{I}_i \mathbf{S}_i\right)^{-1} \mathbf{S}_i^T + {}_i^{h_i}\mathbf{L}^T \mathbf{\Omega}_{h_i,h_i} {}_i^{h_i}\mathbf{L} & \text{si } i = j = h \\ {}_i^{h_i}\mathbf{L}^T \mathbf{\Omega}_{j,h_i} & \text{si } j = h \\ \mathbf{\Omega}_{i,h_j} {}_j^{h_j}\mathbf{L} & \text{sinon} \end{cases} \tag{3.42}$$

où h est le plus proche ancêtre commun de i et j, h_i est le parent de i, $\mathbf{S}_i$ est le vecteur de taille 6 de l'axe de l'articulation – il s'agit en fait d'une colonne de Jacobienne –, $\mathbf{I}_i$ est l'inertie du corps i et ${}_i^{h_i}\mathbf{L}$ propage l'accélération spatiale d'un corps h_i au corps fils i et est défini par

$${}_i^{h_i}\mathbf{L} = \left(\mathbf{I}^6 - \mathbf{S}_i \left(\mathbf{S}_i^T \mathbf{I}_i \mathbf{S}_i\right)^{-1} \mathbf{H}_i^T\right)^T \tag{3.43}$$

$\mathbf{I}^6$ est la matrice identité de taille 6 et $\mathbf{H}_i = \mathbf{I}_i\mathbf{S}_i$ est une variable intermédiaire de l'algorithme. Nous pouvons noter au passage que $\mathbf{\Omega}_{i,i}$ est symétrique et que :

$$\mathbf{\Omega}_{i,j} = \mathbf{\Omega}_{j,i}^T \tag{3.44}$$

Pour calculer cette matrice, trois récursions sont effectuées :

1. on calcule $\mathbf{L}$, le calcul des autres grandeurs $\mathbf{S}$, $\mathbf{H}$ étant déjà effectué lors du calcul de la dynamique libre. Pour cela un parcours des corps en contact à la base est effectué,
2. on calcule les éléments diagonaux de $\mathbf{\Omega}$, $\mathbf{\Omega}_{i,i}$. Pour cela un parcours du corps de base aux corps en contact est effectué,
3. on calcule enfin les autres éléments de $\mathbf{\Omega}$, $\mathbf{\Omega}_{i,j}$. Pour cela un parcours entre les corps en contact est effectué.

Dans la pratique, on peut se limiter à deux récursions (voir figure 3.10). En effet, le calcul de $\mathbf{L}$ étant local à une articulation, autrement dit indépendant des autres corps, il peut être fait en même temps que celui de $\mathbf{\Omega}_{i,i}$. Par ailleurs, il est possible de limiter des produits de matrices dans le calcul de $\mathbf{\Omega}_{i,i}$.

En effet le produit ${}^{h_i}_{i}\mathbf{L}^T \mathbf{\Omega}_{h_i,h_i} {}^{h_i}_{i}\mathbf{L}$ peut être simplifié de la manière suivante :

$$\begin{aligned}
{}^{h_i}_{i}\mathbf{L}^T \mathbf{\Omega}_{h_i,h_i} {}^{h_i}_{i}\mathbf{L} &= \left(\mathbf{I}^6 - \mathbf{S}_i \left(\mathbf{S}_i^T \mathbf{I}_i \mathbf{S}_i\right)^{-1} \mathbf{H}_i^T\right) \mathbf{\Omega}_{h_i,h_i} \left(\mathbf{I}^6 - \mathbf{H}_i \left(\mathbf{S}_i^T \mathbf{I}_i \mathbf{S}_i\right)^{-T} \mathbf{S}_i^T\right) \\
&= \mathbf{\Omega}_{h_i,h_i} - \mathbf{S}_i \left(\mathbf{S}_i^T \mathbf{I}_i \mathbf{S}_i\right)^{-1} \mathbf{H}_i^T \mathbf{\Omega}_{h_i,h_i} - \mathbf{\Omega}_{h_i,h_i} \mathbf{H}_i \left(\mathbf{S}_i^T \mathbf{I}_i \mathbf{S}_i\right)^{-T} \mathbf{S}_i^T \\
&\quad + \mathbf{S}_i \left(\mathbf{S}_i^T \mathbf{I}_i \mathbf{S}_i\right)^{-1} \mathbf{H}_i^T \mathbf{\Omega}_{h_i,h_i} \mathbf{H}_i \left(\mathbf{S}_i^T \mathbf{I}_i \mathbf{S}_i\right)^{-T} \mathbf{S}_i^T \\
&= \mathbf{\Omega}_{h_i,h_i} - \mathbf{\Psi}_1 - \mathbf{\Psi}_2 + \mathbf{\Psi}_3 \\
&= \mathbf{\Omega}_{h_i,h_i} - \mathbf{\Psi}_1 - \mathbf{\Psi}_1^T + \mathbf{\Psi}_3
\end{aligned}$$

Une fois la matrice $\mathbf{\Omega}$ calculée, la matrice d'inertie dans l'espace opérationnel est obtenue en effectuant un changement de repère du corps en collision aux points de contact, autrement dit en projetant dans l'espace des contacts :

$$\mathbf{\Lambda}_{c_i,c_j} = \mathbf{X}_{c_i} \mathbf{\Omega}_{i,j} \mathbf{X}_{c_j}^T \tag{3.45}$$

L'assemblage des $\mathbf{\Lambda}_{c_i,c_j}$ donne la matrice $\mathbf{\Lambda}$. Dans la pratique, la matrice $\mathbf{\Lambda}$ est symétrique, on ne se limite qu'au calcul de la partie supérieure de la matrice.

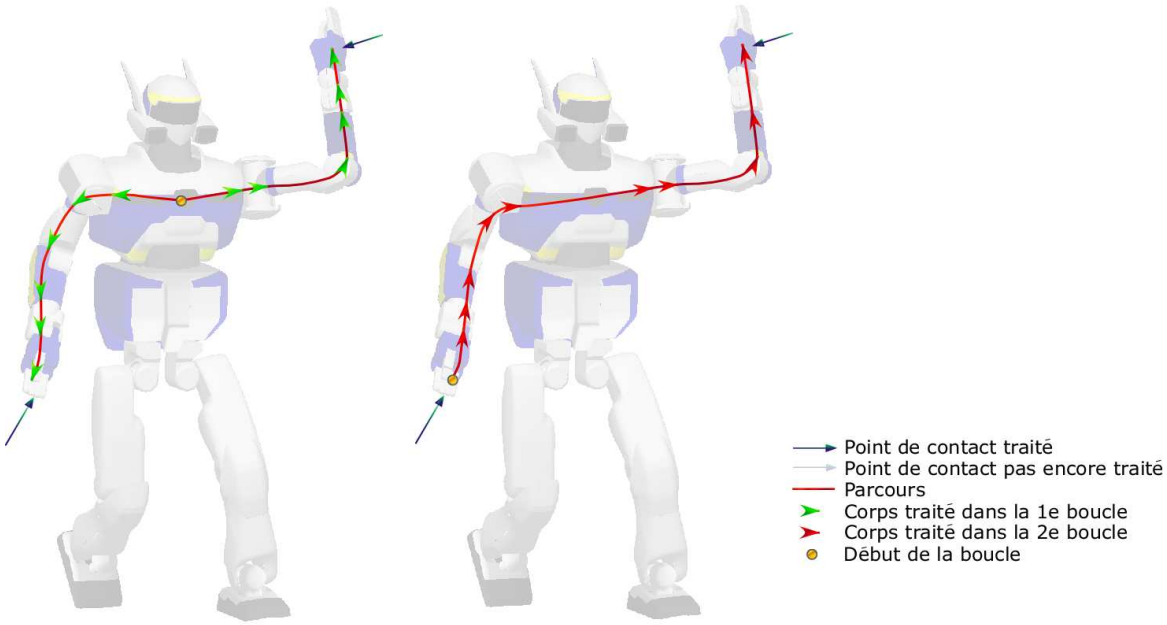


figure 3.10: *Parcours effectué avec [Chang et Khatib, 2000] pour le calcul de* $\mathbf{\Lambda}$*. Ici dans le cas où il y a deux corps en contact, à gauche, le parcours effectué pour calculer* $\mathbf{\Omega}_{i,i}$*, à droite, le parcours effectué pour calculer* $\mathbf{\Omega}_{i,j}$*.*

Cette méthode donne une complexité générale en $\mathcal{O}(m^2 + Cm)$. La complexité du calcul de $\mathbf{\Omega}$ dépend du nombre de corps en collision C. La projection dans l'espace des contacts est toujours en $\mathcal{O}(m^2 + nm)$. Cette méthode est de complexité moindre que la méthode initiale, mais le calcul de $\mathbf{\Omega}$ est plus fastidieux à implémenter car elle nécessite de calculer une nouvelle grandeur ($\mathbf{L}$) et la recherche de l'ancêtre commun le plus proche. Le tableau 3.2 donne le nombre d'opérations nécessaires pour cette

méthode. Nous notons ici encore s le nombre de corps compris entre la base et un corps en contact, $d_{i,j}$ le nombre d'articulations comprises entre le corps i et le corps j.

TABLE 3.2 – Opérations nécessaires pour calculer $\mathbf{\Lambda}$ avec la méthode de Chang et Khatib.

$\times \div$	286	$+153s$	$+72\alpha_C$	$+54Cm + 27m\frac{m+1}{2}$
$+ -$	235	$+179s$	$+66\alpha_C$	$+54Cm + 27m\frac{m+1}{2}$

avec $\alpha_C = \sum_{i,j \leq C \mid j>i} d_{i,j}$.

Cette méthode, comparée à la méthode précédente nécessite moins d'opérations car on n'a pas à parcourir tous les corps du système. Seules les branches contenant les corps en contact sont visitées.

Algorithme 4 : Algorithme de calcul de la matrice $\mathbf{\Lambda}$ avec la méthode de Chang et Khatib.

Données : $\mathbf{S}$, $\mathbf{I}$, $\mathbf{H}$, $\mathbf{X}_c$
Résultat : $\mathbf{\Lambda}$
début
  $\mathbf{\Omega}_{\text{base,base}} = 0$
  pour *tous les corps i en collision* **faire**
    pour *tous les corps j allant de la base à i* **faire**
      $\mathbf{L}_j = \left(\mathbf{I}^6 - \mathbf{S}_i \left(\mathbf{S}_i^T \mathbf{I}_i \mathbf{S}_i\right)^{-1} \mathbf{H}_i^T\right)^T$
      $\mathbf{\Omega}_{j,j} = \mathbf{S}_i \left(\mathbf{S}_i^T \mathbf{I}_i \mathbf{S}_i\right)^{-1} \mathbf{S}_i^T +_i^{h_i} \mathbf{L}^T \mathbf{\Omega}_{h_i,h_i} {}_i^{h_i}\mathbf{L}$
  Rechercher le plus proche ancêtre commun h aux corps i et j en collision
  pour *tous les corps j allant d'un corps i en collision à h* **faire**
    $\mathbf{\Omega}_{i,j} =_i^{h_i} \mathbf{L}^T \mathbf{\Omega}_{h,h_i}$
  pour *tous les corps i allant de h à un corps j en collision* **faire**
    $\mathbf{\Omega}_{i,j} = \mathbf{\Omega}_{h,h_j} {}_j^{h_j}\mathbf{L}$
  pour *tous les contacts i et j* **faire**
    $\mathbf{\Lambda}_{i,j} = \mathbf{X}_i \mathbf{\Omega}_{C_i,C_j} \mathbf{X}_j^T$
fin

3.2.3 Nouvelle méthode

Nous proposons ici une méthode qui reprend le principe de Chang et Khatib, à savoir calculer $\mathbf{\Lambda}$ en deux temps en introduisant d'abord une matrice $\mathbf{\Omega}$ indiquant ici l'influence de chaque membre du système sur les autres, puis en faisant une projection dans l'espace des contacts. L'idée est d'adapter la première méthode de façon à limiter la dépendance par rapport au nombre de points de contact et de n'avoir une dépendance que par rapport au nombre de corps en contact. Nous obtenons une méthode plus simple à implémenter que la méthode de Chang et Khatib.

Nous souhaitons éviter le calcul de nouvelles grandeurs. Pour calculer $\mathbf{\Omega}$, nous réutilisons comme pour la méthode initiale l'algorithme de Featherstone en mouvement libre pour chaque corps en contact cette fois-ci, en supposant qu'il n'y a ni gravité ($\mathbf{g} = 0$), ni couples articulaires ($\mathbf{\Gamma} = 0$), ni vitesses articulaires ($\dot{\mathbf{q}} = 0$). Cet algorithme se fait aussi en deux boucles, la première du corps en contact à la base, la seconde de la base au corps en contact (voir figure 3.11). De fait, l'accélération articulaire

libre devient nulle et on a :

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}\mathbf{J}^T\mathbf{f} \tag{3.46}$$

Considérons cette fois-ci que nous appliquons six fois (correspondant aux six composantes) un effort unitaire $\mathbf{f}_i = (\mathbf{F}_i\ \tau_i)^T$ à l'origine d'un corps en collision i et défini par :

$$\mathbf{F}_i = \begin{matrix}(1,0,0)^T\\(0,1,0)^T\\(0,0,1)^T\\(0,0,0)^T\\(0,0,0)^T\\(0,0,0)^T\end{matrix} \quad \tau_i = \begin{matrix}(0,0,0)^T\\(0,0,0)^T\\(0,0,0)^T\\(1,0,0)^T\\(0,1,0)^T\\(0,0,1)^T\end{matrix} \quad \begin{matrix}\text{première exécution}\\\text{deuxième exécution}\\\text{troisième exécution}\\\text{quatrième exécution}\\\text{cinquième exécution}\\\text{sixième exécution}\end{matrix}$$

ou bien sous forme condensée ${}_u\mathbf{f}_i[y] = 1$ si $u = y$, 0 sinon et $y = \{1, 2, 3, 4, 5, 6\}$. Alors nous obtenons :

$$\ddot{q}_{Ci} = \mathbf{M}^{-1}\mathbf{J}_{Ci}^T \tag{3.47}$$

où $\mathbf{J}_{Ci}$ est la Jacobienne du corps en collision i, puis en passant en accélération spatiale

$$\mathbf{a}_i = \mathbf{J}_{Ci}\mathbf{M}^{-1}\mathbf{J}_{Ci}^T =: \mathbf{\Omega}_{i,i} \tag{3.48}$$

On rappelle que l'accélération libre est nulle. Avec l'équation 3.41 ($\mathbf{a}_j = \mathbf{\Omega}_{i,j}\mathbf{f}_i$) et $\mathbf{f}_i = 1$ on en déduit :

$$\mathbf{a}_j = ({}_0\mathbf{a}_j \ldots {}_5\,\mathbf{a}_j) = \mathbf{\Omega}_{i,j} \tag{3.49}$$

où ${}_u\mathbf{a}_j$ est l'accélération spatiale du corps j associée à la force unitaire ${}_u\mathbf{f}_i$ exercée sur le corps i. Nous projetons ensuite cette équation dans l'espace des contacts pour obtenir la matrice $\mathbf{\Lambda}$ en utilisant l'équation 3.45. Dans la pratique, seule la partie supérieure de $\mathbf{\Lambda}$ est calculée.

La complexité globale de notre méthode est la même que celle de Chang et Khatib. Le calcul de $\mathbf{\Omega}$ est dans le pire des cas en $\mathcal{O}(n^2)$, autrement il est en $\mathcal{O}(C^2)$. Le nombre d'opérations effectuées dans cette méthode est donnée dans le tableau 3.3.

TABLE 3.3 – Opérations nécessaires pour calculer $\mathbf{\Lambda}$ avec la nouvelle méthode.

$\times\ \div$	$252C$	$+78Cs$	$+72\beta_C$	$+54Cm + 27m\frac{m+1}{2}$
$+\ -$	$180C$	$+72Cs$	$+42\beta_C$	$+54Cm + 27m\frac{m+1}{2}$

avec $\beta_C = \sum_{i \le C} d_{i,0}$, $\beta_C < n\frac{n+1}{2}$

Comme la méthode de Chang et Khatib, notre méthode ne visite que les branches contenant les corps en contact. Comme pour la méthode initiale, on réutilise l'algorithme de Featherstone, mais au lieu de l'appliquer $3m$ fois, on l'applique $6C$ fois. On peut noter que cette méthode a également été appliquée dans [Nagasaka *et al.*, 2008].

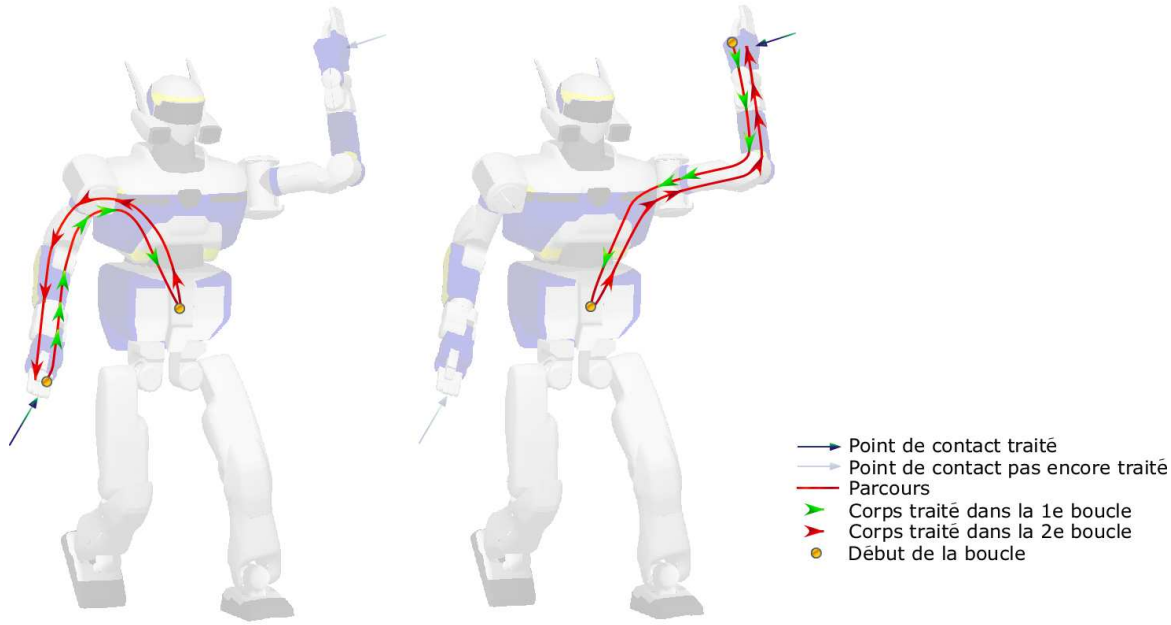


figure 3.11: *Parcours effectué avec la nouvelle méthode pour le calcul de* $\mathbf{\Lambda}$*. Ici dans le cas où il y a deux corps en contact, à gauche, le parcours effectué pour le premier corps en contact, à droite, le parcours effectué pour le second corps en contact.*

3.2.4 Performances et comparaison entre les méthodes

Nous avons réalisé des tests de simulation pour comparer ces méthodes. Nous avons utilisé le robot HRP-2 dans une configuration simple où il est debout. Le seul contact avec l'environnement est avec le sol, donc il y a 2 corps en contact (les deux pieds) et 8 points de contacts au total (quatre pour chaque pied [7]). Le pas d'intégration est de 1ms et les simulations ont été réalisées avec un PC tournant sous un bi-AMD$^{\text{TM}}$64 2,5GHz. Les résultats sont portés sur la figure 3.12.

En reprenant les tableaux 3.1, 3.2, 3.3, on a $C = 2$, $s = 2 \times 6 = 12$ (dans l'arborescence du robot HRP-2, le pied est le 6$^{\text{e}}$ corps après la base (les hanches)), $m = 8$, $\beta_m = 8 \times 6 = 48$, $\alpha_C = 12$ (entre les deux pieds il y a 12 corps), $\beta_C = 6 + 6 = 12$ (il y a deux pieds en contact). Alors le nombre total d'opérations effectuées est de 13824 pour la méthode initiale, 9833 pour la méthode de Chang et Khatib et 9504 pour la nouvelle méthode.

Les deux dernières méthodes sont beaucoup plus performantes que la première méthode, un gain en temps d'un facteur 6,5 environ est réalisé. L'écart entre la méthode de Chang et Khatib et notre méthode est en revanche moins substantiel. Dans la plupart des cas, notre nouvelle méthode sera plus rapide ; cependant dans le cas où deux corps en contact sont situés sur une même branche, la méthode de Chang et Khatib est plus rapide que notre méthode. En effet avec notre méthode nous passerons par la base pour atteindre un corps en contact, par construction de l'algorithme de Featherstone, tandis que la méthode de Chang et Khatib ne visite que la branche liant les corps en contact sans avoir à passer

7. On notera que, physiquement, ce n'est pas crédible si l'objet (ici le pied) est supposé indéformable. On peut soit considérer que l'on a trois points de contact, soit admettre que l'on est dans le cas de systèmes hyperstatiques et faire des hypothèses. C'est un des inconvénients de travailler avec des forces ponctuelles. Même si, dans cette thèse, nous ne nous étendons pas sur ce sujet, nous sommes conscients de ce problème et nous réfléchissons à des solutions.

Algorithme 5 : Algorithme de calcul de la matrice $\mathbf{\Lambda}$ avec la nouvelle méthode. Les notations sont celles de Featherstone et sont rappelées dans la méthode de Chang et Khatib.

Données : $\mathbf{f} = 0$, $\mathbf{S}$, $\mathbf{I}$, $\mathbf{H}$, $\mathbf{X}_c$
Résultat : $\mathbf{\Lambda}$
début
  pour *tous les corps i en collision* **faire**
    pour $y = 1$ **à** 6 **faire**
      $\mathbf{f}_i[y] = (\mathbf{F}_i[y], \tau_i[y])$
    tant que *i différent de la base* **faire**
      pour $y = 1$ **à** 6 **faire**
        $\mathbf{U}_i = \mathbf{S}_i f_i[y]$
        $f_{h_i}[y] = f_i[y] + \mathbf{H}_i(\mathbf{S}_i^T \mathbf{I}_i \mathbf{S}_i)^{-1}\mathbf{U}_i$
        $i = h_i$
  pour *tous les corps i en collision* **faire**
    pour $y = 1$ **à** 6 **faire**
      pour *tous les corps j allant de la base à i* **faire**
        $\ddot{q}_j = (\mathbf{S}_j^T \mathbf{I}_j \mathbf{S}_j)^{-1}(\mathbf{U}_j - \mathbf{H}_j^T \mathbf{a}_{h_j})$
        $\mathbf{a}_j = \mathbf{\Omega}_{i,j} = \mathbf{S}_j \ddot{q}_j + \mathbf{a}_{h_j}$
  pour *tous les contacts i et j* **faire**
    $\mathbf{\Lambda}_{i,j} = \mathbf{X}_{c_i} \mathbf{\Omega}_{i,j} \mathbf{X}_{c_j}^T$
fin

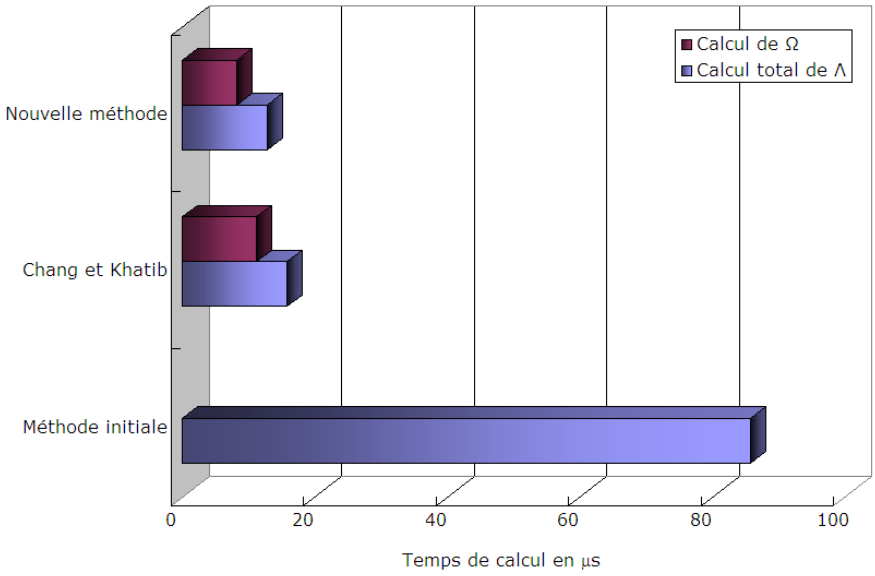


figure 3.12: *Comparaison des trois méthodes en temps de calcul. Les barres bleues représentent le temps de calcul total de la matrice $\mathbf{\Lambda}$ (calcul de la matrice $\mathbf{\Omega}$ et projection dans l'espace des points de contact, pour la méthode de Chang et Khatib et notre nouvelle méthode ; dans la méthode initiale, il n'y a pas de variable intermédiaire introduite, d'où l'absence de barre rose).*

par la base. Prenons l'exemple des figures 3.13 et 3.14, la main et l'avant-bras droits du robot sont en contact. La méthode de Chang et Khatib ne visite la base qu'une fois, tandis que notre méthode visite

la base deux fois. On aura alors dans ce cas 6323 opérations avec la méthode de Chang et Khatib et 7590 avec notre méthode. Dans la pratique, si l'on considère les corps potentiellement en contact dans le cas d'un humanoïde, on passera souvent par la base pour connecter les corps en contact et notre méthode s'avèrera donc plus performante. Toutefois, rien n'empêche d'utiliser les deux méthodes (ce que nous faisons), avec le choix sur la plus performante, selon les cas de contact en cours.

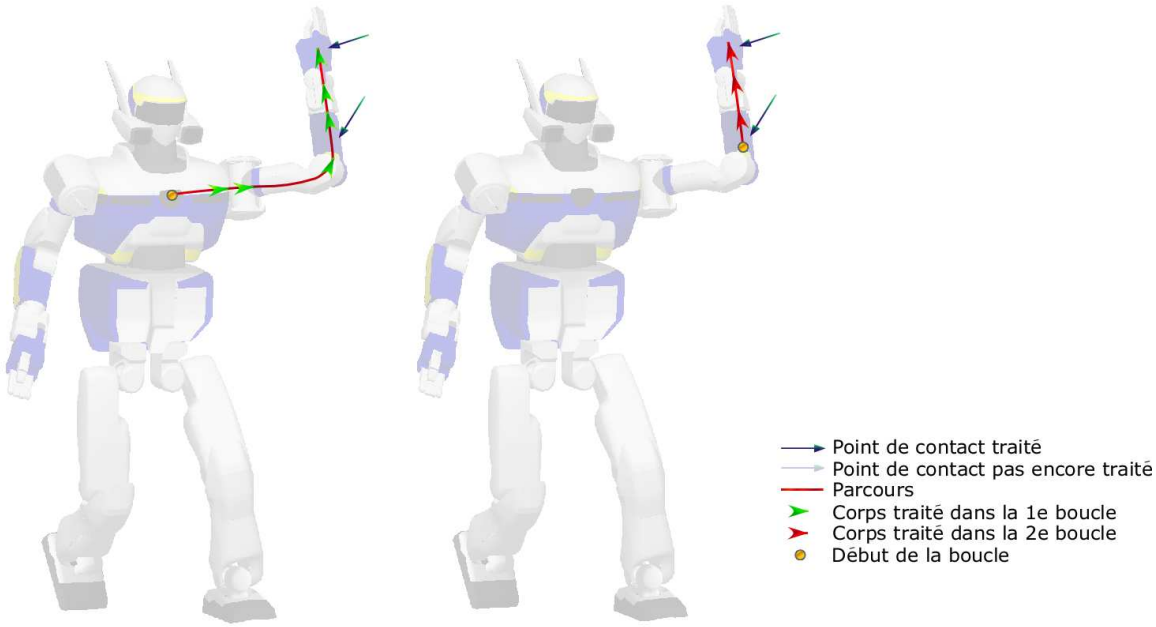


figure 3.13: *Parcours effectués avec [Chang et Khatib, 2000] pour le calcul de* $\mathbf{\Lambda}$. *Ici dans le cas où deux corps en contact sont sur la même branche, à gauche le calcul de* $\mathbf{\Omega}_{i,i}$, *à droite le calcul de* $\mathbf{\Omega}_{i,j}$.

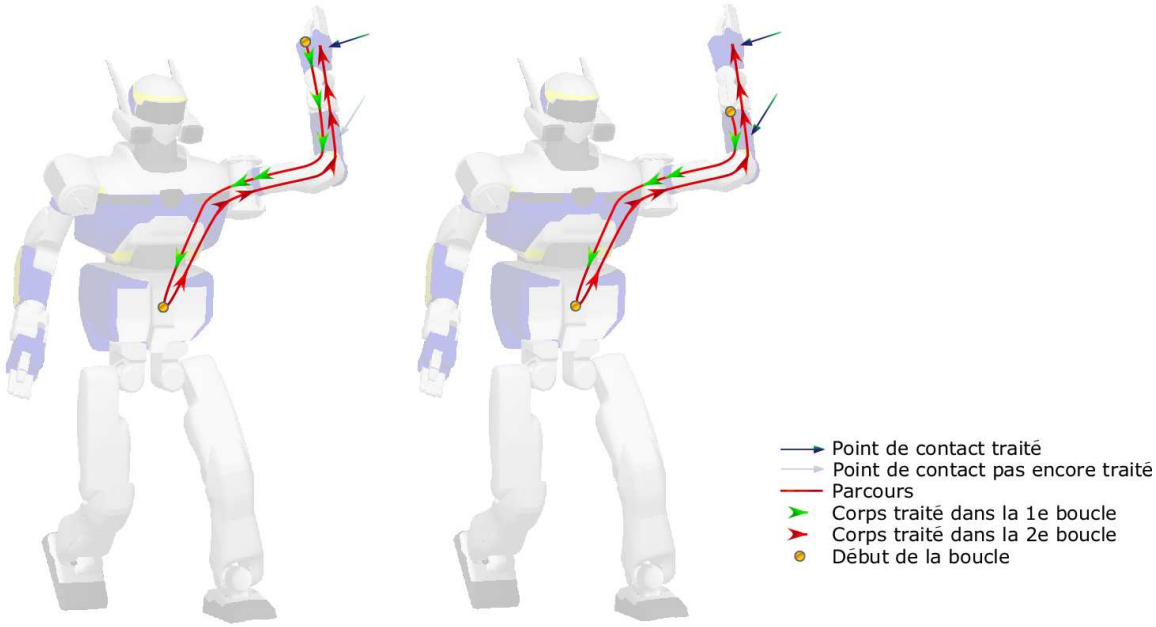


figure 3.14: *Parcours effectués avec notre méthode pour le calcul de* $\mathbf{\Lambda}$. *Ici dans le cas où deux corps en contact sont sur la même branche, on calcule d'abord pour le premier corps en contact puis pour le second.*

3.2.5 Groupes de collision

La taille de la matrice $\mathbf{\Lambda}$ dépend du nombre de points de contact. En présence de frottements et sans discrétisation du cône de frottement, cette matrice est de taille $3m \times 3m$. Cette matrice peut être diagonale par bloc si les objets ne se touchent pas entre eux, chaque bloc étant la matrice d'inertie de chaque objet. Pour obtenir une matrice de rang plein, nous définissons des groupes de collision. Un groupe de collision est une suite de corps non fixes dans l'environnement et en contact entre eux (voir figure 3.15). Chaque groupe a ainsi sa matrice $\mathbf{\Lambda}$ de taille réduite, ce qui permet d'accélérer son calcul et donc celui des forces de contact.

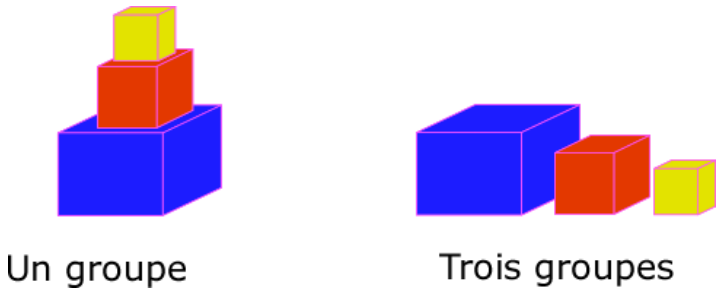


figure 3.15: *Exemples de groupes de collision.*

Pour montrer l'intérêt d'utiliser des groupes de collision, nous avons effectué des tests sur des cubes simples posés sur un sol. Nous avons pris $k = [1, 30]$ cubes et chacun d'eux possède $m = 4$ points de contact [8]. Pour un nombre de cubes k donné, nous avons comparé les deux cas possibles suivants :

- tous les cubes forment un même groupe de collision. Alors la taille de la matrice $\mathbf{\Lambda}$ est $mk \times mk$. Le calcul de la matrice sera donc en $(km)^2$, autrement dit quadratique en nombre de cubes.
- chaque cube est un groupe de collision. Alors la taille de la matrice $\mathbf{\Lambda}$ est $m \times m$. Le calcul de la matrice sera donc en km^2, autrement dit linéaire en nombre de cubes.

Le graphe de la figure 3.16 donne les résultats de simulation. Plus le nombre de cubes augmente, plus le gain de performance est élevé, avec comme prévu une progression du temps de calcul linéaire dans le cas de k groupes de collision et quadratique dans le cas d'un seul groupe. Ce réarrangement préliminaire des corps en contact, réalisé à chaque pas de temps, permet une gestion plus efficace des forces de contact.

3.3 Dynamique des corps poly-articulés à articulations sphériques

Pour que notre simulateur soit plus générique, nous devons pouvoir simuler tous types de corps poly-articulés, en particulier les avatars humains qui possèdent des liaisons sphériques en certains endroits. L'extension aux avatars humains nous permet de simuler des environnements collaboratifs dans lesquels des humains et des robots peuvent travailler ensemble. Pour simuler une articulation sphérique, il est possible de la décomposer en trois liaisons pivot, chaque liaison étant virtuelle, ce qui est une modélisation simplifiée de l'articulation mais qui peut poser problème. En effet, chaque liaison

8. Voir note 7 page 68.

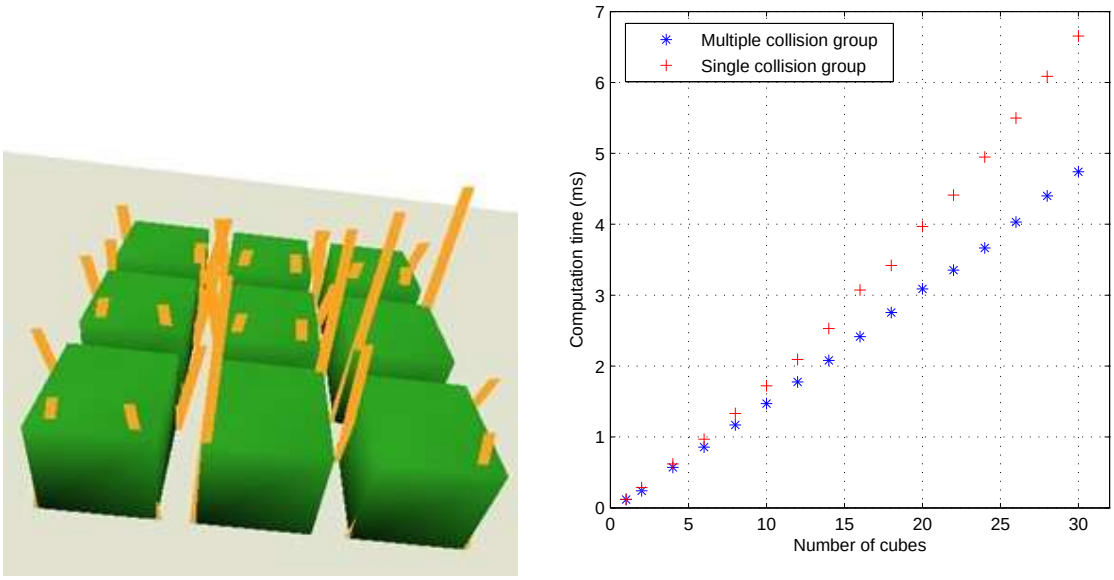


figure 3.16: *Temps de calcul d'un pas de temps en fonction du nombre de cubes.*

virtuelle est attachée à un corps virtuel sans masse, ce qui peut entraîner des problèmes de robustesse numérique. Nous nous proposons ici de considérer les liaisons sphériques telles quelles.

Le traitement de ces liaisons sphériques a été réalisé avec François Keith et a été publié dans [Keith, 2007, Keith *et al.*, 2008, Chardonnet *et al.*, 2008c]. Nous montrons d'abord comment calculer l'accélération articulaire puis la modélisation utilisée pour la position et la vitesse articulaires.

3.3.1 Calcul des accélérations articulaires dans le cas général

Dans le chapitre précédent, lorsque nous avons introduit les équations de Newton-Euler pour un corps poly-articulé, nous avons vu que la vitesse spatiale d'un corps par rapport au précédent s'écrivait sous la forme :

$$\varphi_i = \varphi_{i-1} + \begin{bmatrix} \mathbf{d}_i \wedge \alpha_i \\ \alpha_i \end{bmatrix} \dot{q}_i = \varphi_{i-1} + \mathbf{J}_i \dot{q}_i \tag{3.50}$$

où $\mathbf{d}_i$ représente la position du corps i par rapport au repère de référence et α_i est le vecteur axe de rotation du corps i. Dans le cas des articulations à un seul degré de liberté, $\mathbf{J}_i$ est un vecteur de taille 6 et $\dot{q}_i$ est un scalaire. Avec des articulations à plusieurs degrés de liberté (l degrés de liberté), $\mathbf{J}_i$ devient une matrice de taille $6 \times l$ et $\dot{q}_i$ devient un vecteur de taille l. L'équation précédente se réécrit alors :

$$\varphi_i = \varphi_{i-1} + \mathbf{J}_i \dot{\mathbf{q}}_i \tag{3.51}$$

On reprend ensuite l'algorithme des corps poly-articulés (*Articulated-Body Algorithm*) de Featherstone présenté dans le chapitre précédent pour calculer l'accélération articulaire d'une articulation à l degrés

de liberté :

$$\bar{\mathbf{I}}_i = \underline{\mathbf{I}}_i + \sum_{j \in \mu_i} (\bar{\mathbf{I}}_j - \bar{\mathbf{I}}_j \mathbf{J}_j (\mathbf{J}_j^T \bar{\mathbf{I}}_j \mathbf{J}_j)^{-1} \mathbf{J}_j^T \bar{\mathbf{I}}_j) \tag{3.52}$$

$$\bar{\mathbf{b}}_i = \underline{\mathbf{b}}_i + \sum_{j \in \mu_i} (\hat{\mathbf{b}}_j + \bar{\mathbf{I}}_j \mathbf{J}_j (\mathbf{J}_j^T \bar{\mathbf{I}}_j \mathbf{J}_j)^{-1} (\mathbf{\Gamma}_i - \mathbf{J}_j^T \hat{\mathbf{b}}_j)) \tag{3.53}$$

$$\ddot{\mathbf{q}}_i = (\mathbf{J}_i^T \bar{\mathbf{I}}_i \mathbf{J}_i)^{-1} (\mathbf{\Gamma}_i - \mathbf{J}_i^T (\bar{\mathbf{I}}_i \dot{\varphi}_{\lambda_i} + \overline{\mathbf{b}_i})) \tag{3.54}$$

où on le rappelle, $\bar{\mathbf{I}}$ et $\overline{\mathbf{b}}$ sont respectivement l'inertie du corps articulé et la force qui donne une accélération nulle au corps considéré, $\underline{\mathbf{I}}$ est l'inertie du corps considéré, $\mathbf{\Gamma}$ est le couple appliqué à l'articulation, λ_i est le père du corps i, μ_i est l'ensemble des fils du corps i et :

$$\underline{\mathbf{b}}_i = \varphi_i \wedge \underline{\mathbf{I}}_i \varphi_i, \ \hat{\mathbf{b}}_j = \overline{\mathbf{b}}_j + \bar{\mathbf{I}}_j \dot{\mathbf{J}}_j \dot{\mathbf{q}}_j \tag{3.55}$$

Contrairement au cas des articulations à un seul degré de liberté, $(\mathbf{J}_j^T \bar{\mathbf{I}}_j \mathbf{J}_j)^{-1}$ est une matrice de taille $l \times l$, $\mathbf{\Gamma}_i$ et $\ddot{\mathbf{q}}_i$ sont des vecteurs de taille l. Les autres variables restent les mêmes.

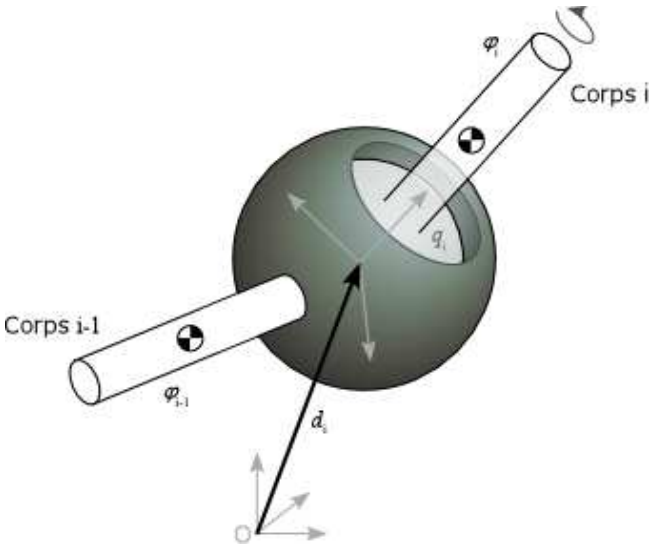


figure 3.17: *Articulation sphérique.*

3.3.2 Modélisation d'une articulation sphérique

Cet algorithme se prête bien à tous types d'articulations, cependant le modèle choisi pour représenter une articulation sphérique peut présenter des avantages et des inconvénients. Dans le cas qui nous intéresse ($l = 3$), il existe différents modèles ou paramétrisations possibles : les matrices de rotation par exemple les angles d'Euler, le vecteur de rotation et les quaternions. Nous rappelons rapidement ces différents modèles.

3.3.2.1 Matrices de rotation

Les matrices de rotation sont obtenues à partir de la composition de rotations successives selon des axes prédéfinis. On utilise trois angles pour représenter la rotation autour de ces axes et il existe plusieurs compositions possibles. Par exemple, pour passer d'un repère $(Oxyz)$ à un repère $(Ox'y'z')$, une composition possible s'obtient par les rotations suivantes : ρ autour de z, θ autour de n, n étant un vecteur porté par l'intersection des plans Oxy et $Ox'y'$ et orienté arbitrairement, et ϕ autour de z', appelées aussi respectivement précession-nutation-rotation propre. Cela veut dire que n'importe quelle orientation peut être obtenue par rapport à un repère de référence à partir de cette composition, et donc il existe au moins un jeu d'angles qui la réalisent. La position articulaire est notée $\mathbf{q} = (\rho, \theta, \phi)^T$. Dans notre exemple, la rotation est calculée de la manière suivante :

$$\mathbf{R} = \mathbf{R}_z(\rho)\mathbf{R}_n(\theta)\mathbf{R}_{z'}(\phi) \tag{3.56}$$

La dérivée de la position articulaire $\dot{\mathbf{q}}$ est liée à la vitesse angulaire ω par :

$$\omega = \mathbf{J}\dot{\mathbf{q}} \tag{3.57}$$

$\mathbf{J}$ est la Jacobienne au point d'origine de l'articulation. Lorsque le déterminant de $\mathbf{J}$ est non nul (pour notre exemple, $\cos(\theta) \neq 0$), cette matrice est inversible. Finalement la vitesse articulaire est donnée par :

$$\dot{\mathbf{q}} = \begin{cases} \mathbf{J}^{-1}\omega & \text{si } \det(\mathbf{J}) \neq 0 \\ \left(\sqrt{\omega_x^2 + \omega_y^2},\ \omega_z + \sin(\theta)\dot{\phi}^2,\ \dot{\phi}\right)^T & \text{sinon} \end{cases} \tag{3.58}$$

Une singularité apparaît lorsque $\theta = i\pi + \frac{\pi}{2}$, i entier. Toutes les représentations (n'importe quelle composition) d'orientation à base de trois angles de rotation présentent des singularités qui ont pour conséquence des erreurs dans le calcul de la position de l'articulation, donc un calcul erroné des vitesses et des accélérations et donc une déviation de la trajectoire normale. Cette déviation est visible par exemple dans le cas d'un pendule lié à une articulation sphérique et que l'on fait tourner autour d'un seul axe. Au lieu d'effectuer un mouvement de balancier, on observe une déviation progressive et un pivotement du pendule sur lui-même.

3.3.2.2 Vecteur de rotation

Dans cette méthode [Grassia, 1998], on utilise un vecteur de rotation $\mathbf{q}$ et un angle de rotation $\theta = \|\mathbf{q}\|$. La direction de la rotation est donnée par le vecteur normé de $\mathbf{q}$, $\overline{\mathbf{q}}$. De fait on a une singularité pour $\theta = 0$. La rotation d'une articulation sphérique est calculée alors par l'exponentielle du vecteur de rotation :

$$\mathbf{R} = \exp(\mathbf{q}) = \exp(\overline{\mathbf{q}}\theta) = \mathbf{I}^3 + \mathbf{q}\sin(\theta) + (1 - \cos(\theta))\overline{\mathbf{q}}^2 \tag{3.59}$$

où $\mathbf{I}^3$ est la matrice identité de taille 3. Cette équation est plus connue sous le nom de formule de Rodriguez [Murray *et al.*, 1994]. En notant ω la vitesse angulaire, la dérivée de $\mathbf{q}$ est donnée par :

$$\dot{\mathbf{q}} = \frac{1}{2}(\gamma\omega + \omega \wedge \mathbf{q} - \eta\mathbf{q}), \begin{cases} \gamma = \theta\cos\left(\frac{\theta}{2}\right),\ \eta = \frac{\omega\mathbf{q}}{\theta}\left(\cot\left(\frac{\theta}{2}\right) - \frac{2}{\theta}\right) & \text{si } \theta > 0 \\ \gamma = 2 - \frac{\theta^2}{6},\ \eta = \omega\mathbf{q}\frac{60+\theta^2}{360} & \text{si } \theta \to 0 \end{cases} \tag{3.60}$$

cot est la cotangente.

3.3.2.3 Quaternion

Un quaternion est généralement représenté par le quadruplet $q = (q_0, \mathbf{q}) = (q_0, q_1, q_2, q_3)$. Si on norme ce quaternion, on obtient la rotation d'un vecteur d'angle q_0 autour de l'axe $\mathbf{q}$ et la matrice de rotation associée s'écrit alors [Eberly, 2003] :

$$\mathbf{R} = \begin{bmatrix} q_0^2 + q_1^2 + \frac{1}{2} & q_1q_2 + q_0q_3 & q_1q_3 - q_0q_2 \\ q_1q_2 - q_0q_3 & q_0^2 + q_2^2 - \frac{1}{2} & q_2q_3 + q_0q_1 \\ q_1q_3 + q_0q_2 & q_2q_3 - q_0q_1 & q_0^2 + q_2^3 - \frac{1}{2} \end{bmatrix} \tag{3.61}$$

Par ailleurs, en notant ω la vitesse angulaire d'un corps i par rapport à un corps $i-1$ et $\overline{\omega}$ le quaternion associé ($\overline{\omega} = (0, \omega)$), on obtient la dérivée du quaternion q par [Schwab, 2002] :

$$\dot{q} = \frac{1}{2}\overline{\omega}q = \frac{1}{2}\begin{bmatrix} -q_1 & q_0 & -q_3 & q_2 \\ -q_2 & q_3 & q_0 & -q_1 \\ -q_3 & -q_2 & q_1 & q_0 \end{bmatrix}\omega \tag{3.62}$$

3.3.2.4 Application au simulateur

Nous avons fait une étude comparative entre ces trois méthodes. Avec les matrices de rotation nous avons des problèmes de divergence numérique autour des positions singulières tandis qu'avec le vecteur de rotation, nous avons observé une augmentation du temps de calcul et une perte de précision numérique. Seuls les quaternions nous ont donné des résultats satisfaisants. D'une part leur représentation est très simple et d'autre part les singularités sont absentes. Nous avons donc utilisé les quaternions pour représenter les articulations sphériques.

Le lecteur pourra se référer à [Grassia, 1998, Schwab, 2002, Eberly, 2003] pour le détail des différentes modélisations.

3.3.3 Exemple de simulation

Avec cette méthode, nous avons pu simuler un avatar humain (voir figure 3.18). Les caractéristiques inertielles ainsi que les constantes diverses du modèle géométrique et dynamique de cet avatar sont données par [Hanavan, 1964, Clauser *et al.*, 1969, Gravez *et al.*, 2005]. Seuls l'épaule, le poignet, la hanche et le pied possèdent des articulations sphériques. Pour simuler les *actionneurs*, nous prenons un modèle très simplifié. En effet, le mécanisme d'actionnement réel est très complexe (il faut modéliser

entre autres les muscles) et ceci dépasse le cadre de cette thèse. Les articulations sphériques sont actionnées en appliquant un couple s'écrivant sous la forme d'un proportionnel-dérivée dans les trois axes de la rotation. Les doigts de la main sont également actionnés.

figure 3.18: *Avatar humain réalisant un mouvement d'étirement des bras. Les épaules, les poignets, la hanche et les pieds sont modélisés par des articulations sphériques.*

Au niveau du temps de calcul, nous n'observons pas de différence notable avec le cas des articulations à un seul degré de liberté.

3.4 Manipulation interactive

Notre souhait est de permettre à des utilisateurs de réaliser des simulations interactives ; par exemple, l'utilisateur pourra réaliser des tâches collaboratives avec des avatars virtuels. Mais l'interactivité est aussi importante dans le cas où l'on veut étudier le comportement d'un avatar virtuel face à une perturbation quelconque, par exemple, si on veut tester la robustesse d'une loi de commande par rapport à des perturbations externes, des imprévues, durant la simulation d'une tâche faisant usage de cette loi. Plus concrètement, on pourrait appliquer une force sur un humanoïde en train de marcher. Dans OpenHRP nous sommes dans l'obligation de programmer ces perturbations à chaque simulation et il est souvent difficile de les paramétrer correctement à l'avance. Pour ce faire, nous avons intégré un dispositif haptique à retour d'effort. Nous avons choisi d'intégrer la famille des PHANTOM© OMNI^{TM} commercialisée par Sensable [9] et qui possède six degrés de liberté en mouvement et trois en retour d'effort. Dans l'architecture d'AMELIF, il est possible d'intégrer d'autres modèles de dispositifs haptiques [Evrard *et al.*, 2008] et d'autres types de dispositif d'interaction.

Nous considérons alors deux manières d'interagir impliquant des considérations particulières dans la manière de procéder. L'opérateur peut utiliser deux modes : (i) dans le premier mode, le point haptique peut pénétrer les objets et être attaché (grâce à un bouton au niveau du poignet du dispositif haptique) et utilisé comme point d'application des forces de manière bilatérale, ce mode permet à l'utilisateur d'appliquer des forces à l'intérieur ou à la surface des corps, et (ii) le mode où le point haptique ne pénètre pas les objets et sert donc de point d'interaction unilatéral, ce mode plus classique

9. `http://www.sensable.com`

figure 3.19: *Le dispositif haptique PHANTOM© OMNI^{TM}.*

considère le point haptique comme point d'exploration tactile. Dans les deux cas, il faut déterminer le point de contact entre le dispositif haptique et l'objet avec lequel on veut interagir. Nous utilisons pour cela la librairie de détection de collision fournie avec le dispositif haptique. Le traitement de ce point sera différent de celui des autres points de contact. En effet il faut fournir au dispositif une liste de triangles correspondant à la zone où se trouve un contact potentiel ainsi que la position relative entre ce point et le pointeur haptique. Pour ne pas surcharger les calculs, on ne donne au dispositif que les triangles proches de lui. Pour cela, nous utilisons la détection de collision de PQP (voir partie 3.1.5) et s'il y a une collision, la géométrie du corps en contact est envoyée au dispositif. Une force de contact est alors calculée.

Dans le cas du toucher, la force peut être calculée par la méthode par contraintes. Cependant, si le point n'a pas de masse, ceci engendre des problèmes numériques. La force à appliquer à l'objet est celle donnée directement par le pointeur haptique, que nous pondérons par un coefficient choisi arbitrairement.

Dans le cas où il s'agit d'attacher un point pour y appliquer des forces (ce qui permet de manipuler et déplacer les objets), la force est calculée à partir d'un modèle masse-ressort 6D liant le point d'attache au point actuel donné par l'utilisateur. La force à appliquer sur l'objet est alors de la forme :

$$\mathbf{f}_e = k_p \left(\mathbf{x}_{\text{objet}} - \mathbf{x}_{\text{pointeur}}\right) + k_v \left(\dot{\mathbf{x}}_{\text{objet}} - \dot{\mathbf{x}}_{\text{pointeur}}\right) \tag{3.63}$$

où $\mathbf{x}_{\text{objet}}$ et $\mathbf{x}_{\text{pointeur}}$ sont les positions de l'objet (point d'attache à l'intérieur ou sur la surface) et du pointeur haptique respectivement, k_p et k_v sont les coefficients du ressort-amortisseur avec $k_v = \sqrt{2m_{\text{objet}}k_p}$. On peut noter qu'il s'agit d'une méthode par pénalités : le choix des coefficients peut aboutir à des instabilités numériques. Par ailleurs le rendu haptique n'est pas nécessairement réaliste pour l'utilisateur et le choix des paramètres est subjectif ; en revanche si la raideur est trop forte, la force résultante devient grande jusqu'à faire diverger le système, et donc peut être dangereuse pour l'utilisateur et pour le dispositif. Pour résoudre ce problème, les méthodes à base de contraintes, plus coûteuses, peuvent aussi être appliquées [Zilles et Salisbury, 1995, Ruspini et Khatib, 1998].

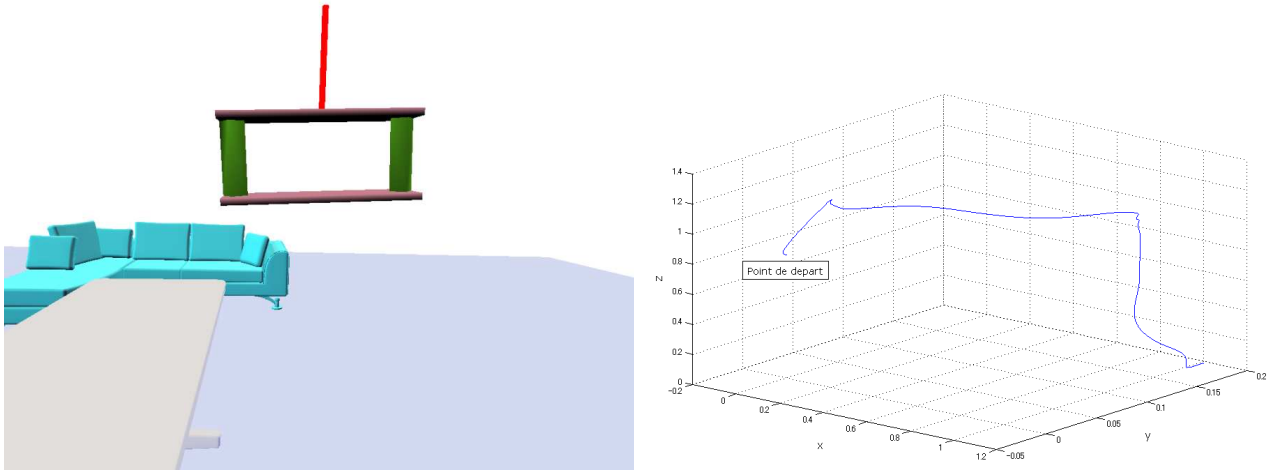
Une fois la force calculée, elle peut être intégrée à la dynamique libre des objets comme une force extérieure connue $\mathbf{f}_e$ au même titre que la gravité :

$$\ddot{\mathbf{q}} = \mathbf{M}^{-1}(\mathbf{q}) \left(\mathbf{\Gamma}(\mathbf{q}) - \mathbf{b}\left(\mathbf{q}, \dot{\mathbf{q}}\right) - \mathbf{g}(\mathbf{q})\right) + \mathbf{M}^{-1}\mathbf{J}_e^T \mathbf{f}_e \tag{3.64}$$

où $\mathbf{J}_e$ est la Jacobienne au point d'application de la force sur le corps en contact avec le dispositif haptique.

3.4.1 Exemples de simulation

Nous testons d'abord l'interaction, moyennant un dispositif à retour d'effort avec un objet simple. Pour cela nous considérons un objet de $1,75$kg posé initialement sur une table et avec le dispositif haptique nous le prenons pour le déplacer à un autre endroit selon une seule direction (l'axe x). Nous poussons ensuite l'objet vers le bas et nous le lâchons un peu au-dessus du sol (voir figure 3.20). Les forces exercées sur l'objet pendant la manipulation sont données en figure 3.21.

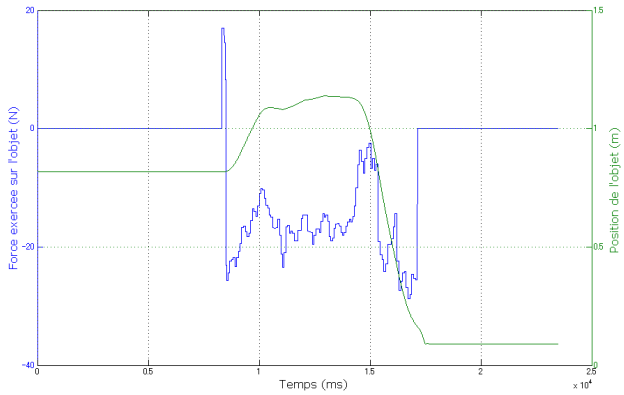


(a) Simulation dans laquelle nous manipulons un objet avec le dispositif haptique.

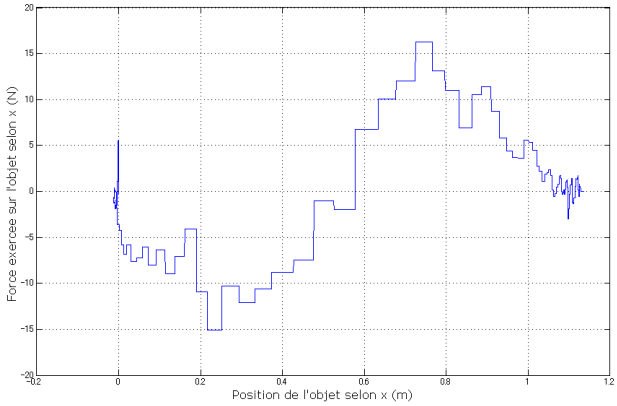
(b) Evolution de la position de l'objet dans l'espace.

figure 3.20: *Simulation interactive avec un objet.*

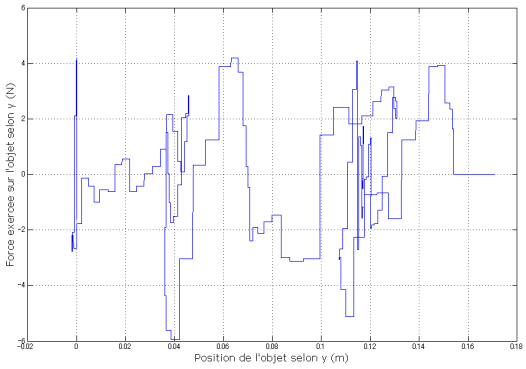
L'évolution des forces se fait par paliers car enregistrée à partir d'AMELIF, le taux de rafraîchissement de l'interface haptique n'est pas le même que celui de la simulation (bien plus élevé). Dans la figure 3.21(a), le pic observé à 8s correspond au moment de la saisie de l'objet. On y attache alors le point à partir duquel les forces de l'opérateur sont appliquées selon un couplage virtuel à base de ressorts-amortisseurs. La force augmente ensuite jusqu'à -25N car nous soulevons l'objet ce qui a pour conséquence, avec le poids de l'objet, d'étirer le ressort-amortisseur virtuel. La force diminue ensuite petit à petit au moment où l'objet a atteint une position en z à peu près constante, autrement dit le ressort se compresse et atteint un état stable, pour ensuite se stabiliser autour de $-17,15$N, qui est le poids de l'objet. Pendant la période où la force oscille autour de $-17,15$N, l'objet est déplacé selon l'axe x. La force diminue ensuite brutalement au moment où l'objet est poussé vers le bas car le ressort-amortisseur virtuel est alors compressé. La force augmente ensuite à nouveau brutalement car nous arrêtons de pousser sur l'objet. Les oscillations qui suivent sont simplement dûes à la stabilisation du ressort-amortisseur. L'objet est ensuite lâché (élimination du point d'attache), la force devient donc nulle. La figure 3.21(b) représente l'évolution de la force en fonction de la position de l'objet selon



(a) Evolution de la position de l'objet et de la force exercée par le dispositif selon l'axe z au cours du temps. En bleu, la force exercée sur l'objet ; en vert, la position de l'objet.



(b) Evolution de la force exercée par le dispositif selon l'axe x. x au cours du déplacement de l'objet selon l'axe x.



(c) Evolution de la force exercée par le dispositif selon l'axe y au cours du déplacement de l'objet selon l'axe y.

figure 3.21: *Evolution de la force exercée par le dispositif selon les axes x, y et z.*

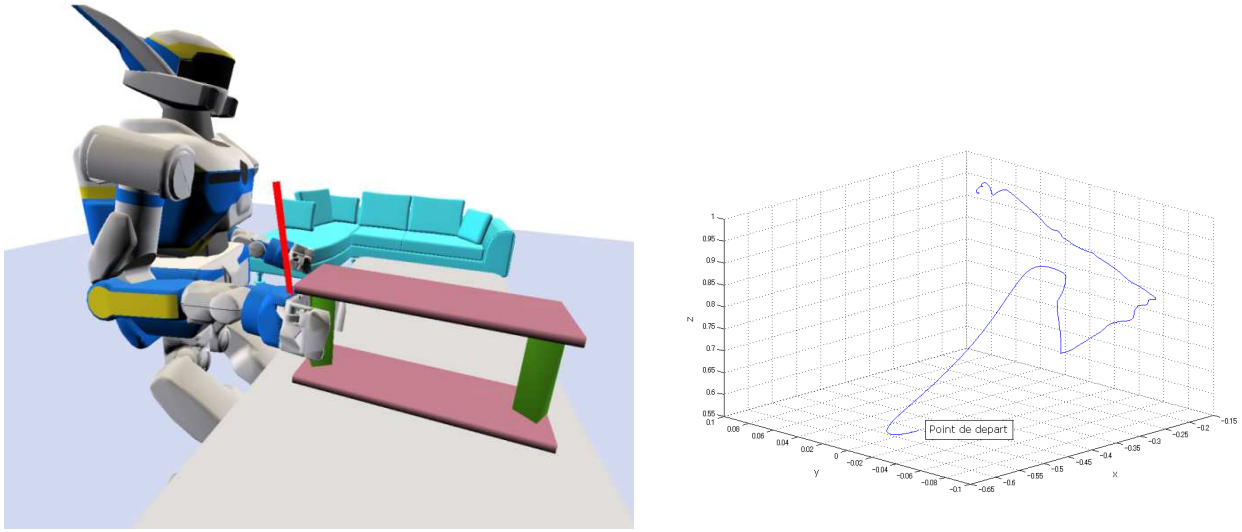
l'axe x. L'objet est déplacé d'environ $1,1$m. La phase $[0;0,58]$m correspond au moment où nous tirons assez fort sur l'objet. La force est de fait négative car le ressort-amortisseur tend à tirer l'objet dans l'autre sens. Pendant la phase $[5,8;1,1]$m, nous tirons beaucoup moins fort sur l'objet. De fait, l'énergie accumulée pendant la première phase est libérée et la force devient positive. Elle devient quasiment nulle au moment où la position en x de l'objet a atteint $1,1$m. La figure 3.21(c) montre l'évolution selon l'axe y. La force oscille car lors du déplacement de l'objet nous n'avons pas bloqué le dispositif haptique dans la direction y.

Nous avons réalisé une deuxième simulation plus complexe. Nous manipulons cette fois-ci le robot HRP-2. Au départ le robot est dans la position debout, puis il plie un peu ses jambes et ses bras. Ici, nous avons rendu le robot compliant en le contrôlant en position et en définissant les positions articulaires désirées $\mathbf{q}_d$ par l'équation différentielle suivante :

$$\mathbf{M}_d\ddot{\mathbf{q}}_d + \mathbf{B}_d\dot{\mathbf{q}}_d = \mathbf{J}_e^T\mathbf{f}_e \tag{3.65}$$

où $\mathbf{M}_d$ et $\mathbf{B}_d$ sont des matrices diagonales positives correspondant à une inertie et un amortissement

virtuels respectivement. La Jacobienne $\mathbf{J}_e$ lie les vitesses articulaires aux vitesses cartésiennes des corps compris entre la base et le corps sur lequel on applique une force $\mathbf{f}_e$. Avec le dispositif haptique, nous prenons la main droite du robot et nous l'approchons d'un objet possédant deux poignées (il s'agit du même objet que précédemment). Nous plaçons la main droite pour que le robot prenne l'objet et avec le dispositif haptique, nous soulevons le bras avec l'objet (voir figure 3.22). Les forces exercées sur le bras du robot pendant la manipulation sont données en figure 3.23.



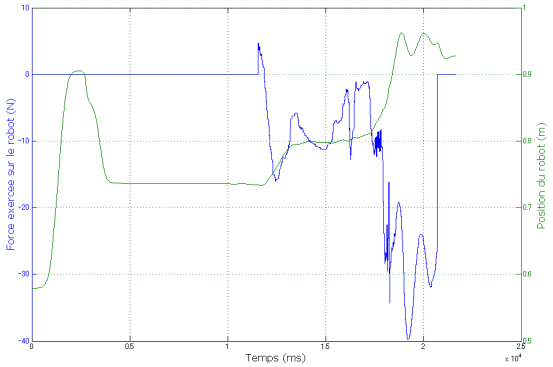
(a) Simulation dans laquelle nous manipulons le bras du robot avec le dispositif haptique.

(b) Evolution de la position du bras du robot dans l'espace.

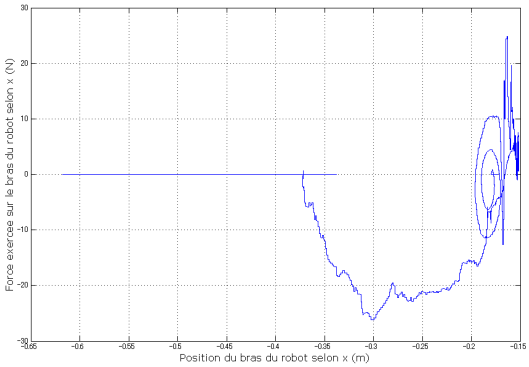
figure 3.22: *Simulation interactive avec le bras du robot HRP-2.*

Dans la figure 3.23(a), la phase $[0; 4]$s correspond au moment où le robot plie ses jambes et ses bras. A $t = 12,5$s, nous prenons la main du robot avec le dispositif haptique et nous la déplaçons pour la positionner en face de l'objet. En regardant les autres figures 3.23(b), (c), ce déplacement est fait principalement selon l'axe x. Comme dans la première simulation, le ressort-amortisseur virtuel tend à tirer le bras dans l'autre sens. A $t = 17,5$s, le robot ferme la pince pour prendre l'objet. Pendant cette phase, nous tenons le bras, d'où les oscillations de courte durée observées sur la figure 3.23(a). Le pic observé sur la figure 3.23(b) coïncide avec la chute de la force à $t = 18,2$s sur la figure 3.23(a) et au pic observé à $x = -0,164$m sur la figure 3.23(c). Ce pic est dû au mauvais placement de la main devant l'objet. En effet la main n'a pas été placée de manière à ce qu'il n'y ait pas de débattement de l'objet dans la main, de fait en soulevant l'objet, celui-ci a bougé pour que la forme de la poignée épouse celle de la pince. A la fin de la manipulation, nous arrêtons de bouger le dispositif haptique ; toutefois le bras vibre lentement sous l'effet de l'inertie de l'objet, de la compliance du robot et du ressort-amortisseur virtuel, d'où la présence d'oscillations qui s'amortissent.

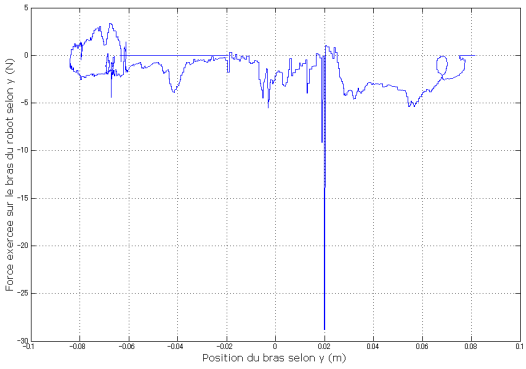
Enfin nous avons réalisé une simulation dans laquelle nous considérons le robot tenant l'objet dans ses deux pinces. Avec le dispositif à retour d'effort, nous tenons une autre partie de l'objet (choix d'un point d'attache sur ou dans l'objet, puis application de forces à souhait) et nous déplaçons avec le robot cet objet (voir figure 3.24).



(a) Evolution de la position du bras du robot et de la force exercée par le dispositif selon l'axe z au cours du temps. En bleu, la force exercée sur le bras du robot ; en vert, la position du bras du robot.



(b) Evolution de la force exercée par le dispositif selon l'axe x au cours du déplacement du bras du robot selon l'axe x.



(c) Evolution de la force exercée par le dispositif selon l'axe y au cours du déplacement du bras du robot selon l'axe y.

figure 3.23: *Evolution de la force exercée par le dispositif selon les axes x, y et z.*

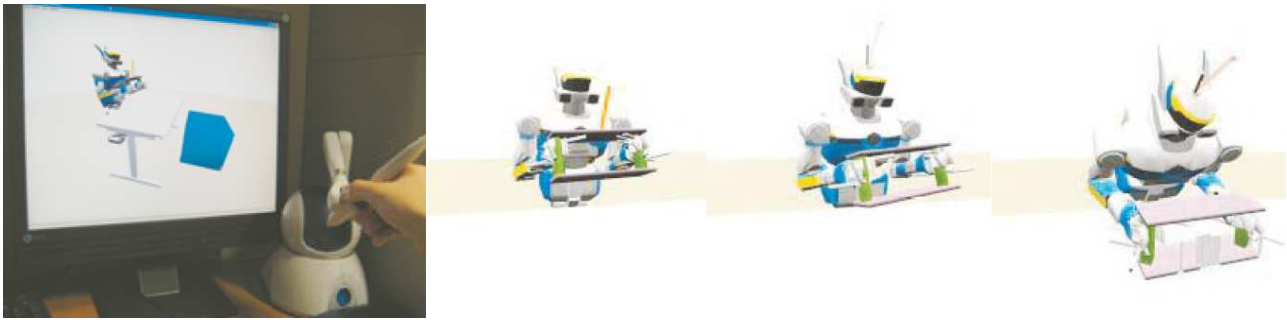


figure 3.24: *Tâche collaborative avec le robot HRP-2.*

Ici le robot n'est pas très compliant, de fait le robot tombe en avant lorsqu'à la fin de la simulation nous prenons la tête du robot avec le dispositif (mode interaction unilatérale).

3.5 Conclusion

Dans ce chapitre nous avons présenté le simulateur développé au cours de cette thèse. Nous avons conçu une architecture modulaire, rapide et permettant de charger n'importe quelle scène, de même qu'il est possible d'implémenter un modèle simplement et rapidement, sans avoir à connaître l'architecture complète du simulateur. Nous avons présenté les modèles physiques implémentés dans notre simulateur. Dans le souci constant d'obtenir des simulations interactives temps-réel et *réalistes*, nous avons opté pour des algorithmes de calcul dynamique très rapides et des méthodes par contraintes pour résoudre les problèmes de contact avec frottement. Ces modèles ont déjà prouvé leur efficacité dans des exemples académiques, ou dans des simulateurs pour d'autres applications, cependant nous avons voulu montrer ici un exemple d'intégration réussie avec des scénarios robotiques complexes.

Les simulations réalisées avec notre simulateur ont montré que deux modules étaient coûteux en temps de calcul : d'une part la détection de collision, d'autre part le calcul des forces de contact, et plus particulièrement le calcul de l'opérateur de Delassus plus connu en robotique comme l'inertie effective dans l'espace opérationnel (ici l'espace des contacts). Nous avons présenté une nouvelle méthode de calcul de cette matrice ainsi que les groupes de collision permettant une amélioration du calcul dynamique.

Nous avons ensuite montré qu'il était possible avec notre simulateur de simuler des corps poly-articulés possédant des articulations à plusieurs degrés de liberté, plus spécifiquement des articulations sphériques, ce qui nous a permis de simuler des avatars humains. Pour cela, nous avons étendu l'algorithme de calcul dynamique de Featherstone et nous avons pris soin de choisir une bonne paramétrisation de l'articulation. Il est important de noter qu'il existe des articulations plus complexes, c'est-à-dire avec plusieurs degrés de liberté et plusieurs actionneurs en parallèle dans des systèmes bouclés et non simplement arborescents. Nous nous sommes contentés dans cette thèse de ne considérer que des systèmes arborescents ; cependant la simulation de telles articulations pourra être envisagée.

Enfin grâce à la modularité de notre simulateur, nous avons pu intégrer facilement un module d'interaction, permettant à l'utilisateur d'interagir avec l'environnement virtuel et de réaliser des tâches collaboratives ou d'étudier le comportement d'un système face à une perturbation. Nous avons intégré un dispositif à retour d'effort permettant à l'utilisateur de ressentir les efforts exercés sur les objets manipulés. Pour manipuler un objet, nous avons attaché entre l'objet et le pointeur haptique un ressort-amortisseur virtuel. Cette méthode est très rapide mais nécessite un paramétrage minutieux pour éviter des instabilités numériques.

L'étape suivante est de rajouter dans notre simulateur la possibilité de simuler des corps flexibles, ce que nous proposons dans le chapitre suivant.

Chapitre 4

Prise en compte des flexibilités

Jusqu'à présent, nous avons considéré que les corps composant le robot étaient rigides. En réalité, les robots possèdent des flexibilités, en particulier articulaires à cause des mécanismes de réductions de vitesse ou, de manière générale, les composants impliqués dans la chaîne de transmission des efforts. Par ailleurs, si nous souhaitons simuler des avatars humains ou les éventuels robots qui seront munis d'une carapace flexible (on dira peaux artificielles par abus de langage), il est nécessaire de considérer ces flexibilités dans le modèle dynamique et particulièrement dans le modèle de contact. Dans ce chapitre nous nous intéressons à la prise en compte de la manière la plus simple possible, de ces flexibilités dans notre simulateur. Nous considérons deux types de flexibilités : la flexibilité articulaire et la flexibilité de carapace. A titre d'étude préliminaire montrant l'intérêt de cette problématique, nous étudions également les formes de semelles qui permettent d'atténuer les vibrations.

Sommaire

4.1 Contexte

4.1.1 Dans la réalité

Un des problèmes majeurs des robots humanoïdes actuels est de pouvoir évoluer dans un environnement humain en toute sécurité ; c'est ce qu'on dénomme communément la collocation homme-robot. Il est donc nécessaire de développer des stratégies permettant d'assurer la sécurité de l'homme mais aussi celle des robots. Par ailleurs le robot doit pouvoir garder le contrôle de ses mouvements lors d'un impact imprévu avec l'environnement. En effet, un impact, se propageant dans la structure du robot provoque des vibrations visibles au niveau des points terminaux ; il convient donc d'atténuer le plus possible ces vibrations souvent perçues comme des saccades de mouvements et qui donnent l'impression d'imperfection de la machine. Ce phénomène est visible lors de la marche ; en particulier plus la marche est rapide, plus les vibrations au points terminaux sont élevées. Ces vibrations peuvent engendrer une instabilité dans la réalisation de la tâche ; dans des cas extrêmes, elles peuvent endommager la structure du robot ou amplifier les défauts de l'assemblage mécanique ainsi que l'usure. Enfin, les robots futurs sont destinés à être dotés d'une *peau flexible* comme c'est le cas pour les robots androïdes. Pour résoudre ces problèmes de vibration indésirables, deux méthodes sont étudiées.

La première est d'utiliser des systèmes internes. Par exemple, des mécanismes d'absorption de chocs sont placés à l'intérieur du pied du robot. C'est le cas du robot ASIMO de Honda ou du robot HRP-2. Ces mécanismes peuvent être réalisés en gomme comme sur le robot HRP-2. Ils absorbent les chocs entre le robot et le sol. Ils agissent également comme filtre passe-bas mécanique pour atténuer les vibrations dues au contrôle de la flexibilité utilisée pour la stabilisation du robot en ligne [Isozumi *et al.*, 2004]. Park *et al.* [Park *et al.*, 2007] proposent un système composé d'un ressort, de glissières et d'un câble. Lorsqu'une force est appliquée, un moment est créé, le câble compresse le ressort. Cette méthode est intéressante lorsque les forces appliquées ne sont pas concourantes avec l'axe de rotation de l'articulation, mais dans un cas de marche sur un sol plat par un robot humanoïde, les forces sont majoritairement concourantes avec l'axe de rotation de l'articulation, par conséquent le moment est proche de zéro et donc le système n'absorbe pas les chocs.

La deuxième méthode est d'utiliser des systèmes externes. L'intérêt d'utiliser de tels systèmes est de permettre une meilleure adaptation aux irrégularités de l'environnement grâce aux déformations induites ; de plus ces systèmes ont de meilleures propriétés d'adhésion au terrain. Par exemple, Yamaguchi *et al.* [Yamaguchi *et al.*, 1995] utilisent des semelles multi-composites. Ce mécanisme permet de détecter la surface d'impact lors de la marche et d'absorber les chocs. La méthode UKEMI [Fujiwara *et al.*, 2002] propose de contrôler le mouvement de chute d'un robot humanoïde pour que l'impact ait lieu sur une partie déformable préalablement définie et installée sur le robot. Ces méthodes ne font cependant à aucun moment appel à un quelconque modèle analytique des systèmes déformables. Or ce qui nous intéresse ici est d'intégrer ces systèmes en simulation afin, par exemple, d'en optimiser la forme et les caractéristiques physiques. Plus généralement la prise en compte de ces systèmes de manière analytique n'est pas courante dans la littérature.

4.1.2 En simulation

Du point de vue de la simulation la prise en compte de telles flexibilités passe par la modélisation et la simulation de corps déformables. Les premiers travaux sur la simulation de corps déformables ont été réalisés dans le domaine de l'animation graphique [Terzopoulos et Fleischer, 1988]. Depuis, l'animation graphique propose un panel de méthodes intéressantes pour la simulation des déformations mais dans le but de la synthèse d'images réalistes et de l'animation virtuelle, mais pas nécessairement à finalité d'ingénierie. Certaines applications portent également sur la simulation dans le domaine médical, comme le propose le logiciel SOFA [Allard *et al.*, 2007], qui inclut les modèles de déformation présentés ci-dessous et la possibilité d'interagir avec l'environnement virtuel.

4.1.2.1 Masse-ressort

Pour simuler des corps déformables, l'approche classique et la plus utilisée en synthèse d'image est le modèle masse-ressort. Un corps est maillé en nœuds représentés par des masses et reliés entre eux par des ressorts. L'équation de mouvement du système est alors représentée de la manière suivante :

$$\mathbf{M}\ddot{\mathbf{x}} + \mathbf{C}\dot{\mathbf{x}} + \mathbf{K}\mathbf{x} = \mathbf{F} \tag{4.1}$$

où $\mathbf{M}$, $\mathbf{C}$ et $\mathbf{K}$ sont respectivement les matrices de masse, d'amortissement et de rigidité, $\mathbf{M}$ et $\mathbf{C}$ sont diagonales. L'avantage, comme pour la résolution du contact par des méthodes par pénalités, est sa facilité d'implémentation et sa rapidité de calcul. En particulier, cette approche est adaptée pour des applications interactives temps-réel et on retrouve beaucoup d'applications dans le domaine médical. Comme le traitement des interactions entre les nœuds se fait de manière locale, le calcul peut être parallélisé. Récemment, cette méthode a été implémentée sur des cartes graphiques qui offrent une puissance de calcul accrue par rapport à une implémentation sur des processeurs classiques, permettant la réalisation de simulations interactives temps-réel sur des exemples appliqués à la médecine [Altomonte *et al.*, 2008]. Les inconvénients sont toujours le choix des paramètres à partir des caractéristiques physiques des matériaux. Cette approche n'est pas utilisée dans la conception et l'ingénierie de produits. Par ailleurs, la déformation du système va dépendre du maillage choisi : en particulier les efforts seront dirigés dans la direction des arêtes du maillage. Enfin, une grande raideur des ressorts peut donner lieu à des divergences numériques et une autre résolution nécessite de redéfinir les paramètres. Dans notre situation, nous désirons obtenir des simulations réalistes et robustes numériquement de systèmes complexes dans lesquelles nous pouvons être amenés à réaliser du contrôle et dont le but est aussi d'atténuer les vibrations. Par conséquent cette approche n'est pas adaptée à notre cas.

4.1.2.2 Eléments finis

La deuxième approche est d'utiliser des solutions empruntées aux méthodes numériques des milieux continus, typiquement les éléments finis. Cette approche donne une approximation de fonctions

continues réalisant l'équilibre du système. Le système est découpé en éléments reliés entre eux par des nœuds. Le choix du type d'élément (triangle, rectangle, tétraèdre...) dépend de la précision désirée. En appliquant des efforts sur un système, celui-ci se déforme. On calcule alors la déformation par l'équation de mouvement suivante :

$$\mathbf{M\ddot{U}} + \mathbf{C\dot{U}} + \mathbf{KU} = \mathbf{F} \tag{4.2}$$

Ici les éléments **M**, **C** et **K** résultent de l'assemblage des matrices élémentaires de chaque élément. Contrairement à l'approche précédente, l'état du système est représenté par l'intégration sur le volume de tous les éléments. Cette méthode présente l'avantage d'être beaucoup plus précise que l'approche de masse-ressort mais est très coûteuse en temps de calcul car, en particulier dans le cas de grandes déformations, il est nécessaire de réévaluer les matrices de masse et de raideur à chaque pas de temps. Puisque le système est discrétisé, l'influence de la méthode d'intégration sur la stabilité, mais aussi sur la rapidité, n'est pas négligeable. On a vu dans le premier chapitre que les méthodes implicites sont particulièrement efficaces et sont couramment utilisées pour l'animation graphique [Galoppo *et al.*, 2006, Sifakis *et al.*, 2007]. Longtemps inappropriée pour des applications temps-réel, l'approche "éléments finis" est de plus en plus utilisée grâce aux progrès réalisés dans la performance des ordinateurs.

Dans la pratique, si on considère des déformations petites par rapport à la taille du système, les matrices seront constantes et donc pré-calculées au début de la simulation. Par ailleurs, pour accélérer le calcul, il n'est pas nécessaire de prendre en compte tous les éléments comme nous le montrerons plus loin.

4.1.2.3 Méthodes hybrides

Une approche hybride a été proposée par [Terzopoulos et Witkin, 1988] et permet de représenter le mouvement d'un système mécanique en séparant en deux composantes la configuration du système [Duriez, 2004, Moissenet, 2007] : une composante de référence et une composante de déformation du système par rapport à celle de référence. De cette configuration sont calculées les déformations du système. Cette méthode a été proposée en particulier dans le cas où l'élasticité du matériau est non-linéaire, une partie de cette non-linéarité étant due aux rotations de l'objet et entraînant un mauvais conditionnement des équations lorsque l'objet tend à être de plus en plus rigide. L'intérêt de cette méthode est de pouvoir considérer un modèle de déformation en élasticité linéaire et de garder un modèle rigide dans le cas où il n'y a pas de déformation, ce qui permet de réaliser de grands mouvements dans l'espace. La condition à cela est que dans le repère de l'objet les déformations doivent rester petites. Une approche locale de cette méthode, nommée corotationnelle, propose de séparer les rotations de chaque élément du maillage des déformations réelles [Hauth et Strasser, 2004]. Le modèle obtenu est intégré implicitement. Cette méthode hybride s'applique exclusivement à des corps déformables. Dans notre cas, nous simulons des corps aussi bien rigides que déformables, nous ne pouvons donc pas appliquer cette méthode telle quelle. Nous pouvons néanmoins l'adapter, ce que nous montrons plus

loin.

Le lecteur pourra se référer à [Gibson et Mirtich, 1997, Duriez, 2004] pour une description plus détaillée de ces méthodes.

Nous présentons dans ce chapitre les développements théoriques et pratiques pour intégrer ces flexibilités à la simulation dynamique. Ces flexibilités seront intégrées au simulateur du robot HRP-2. Dans un premier temps, nous effectuons une comparaison entre les deux méthodes présentées auparavant. Nous nous attachons ensuite à améliorer notre modèle et nous nous intéressons à la marche du robot avec une semelle sur un sol plat. Une partie des résultats a été publiée dans [David *et al.*, 2008, Chardonnet *et al.*, 2008a].

4.2 Modèle d'une articulation déformable

Il s'agit ici de modéliser un système interne passif pour l'absorption de chocs comme celui présent dans le pied du robot HRP-2. En l'occurrence, ce système est composé de trois bagues en caoutchouc avec des amortisseurs et est placé entre le support du pied et la cheville du robot (voir figure 4.1).

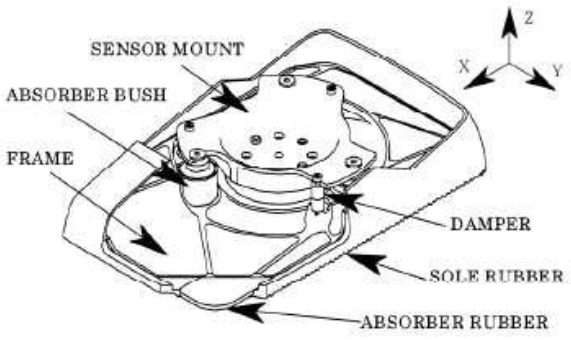


figure 4.1: *Le mécanisme d'absorption de chocs sur le robot HRP-2 (tiré de [Isozumi* et al.*, 2004]).*

Ce mécanisme produit un effet non négligeable sur le comportement dynamique du robot HRP-2, en particulier lors de la marche, par conséquent il est nécessaire de le modéliser dans notre simulateur pour obtenir une simulation plus fidèle du robot humanoïde HRP-2. Nakaoka *et al.* [Nakaoka *et al.*, 2007] utilisent un système ressort-amortisseur par axe articulaire virtuel à trois degrés de liberté comme modèle de ce mécanisme. L'articulation virtuelle consiste en une liaison glissière selon l'axe z et deux liaisons pivot dans les deux autres axes qui s'écrivent sous la forme (voir figure 4.2) :

$$\begin{aligned} f_z &= -k_{zp}q_z - k_{zv}\dot{q}_z \\ \tau_x &= -k_{xp}q_x - k_{xv}\dot{q}_x \\ \tau_y &= -k_{yp}q_y - k_{yv}\dot{q}_y \end{aligned} \tag{4.3}$$

où q_z, q_x, q_y sont les positions des liaisons glissière et pivots respectivement. Les valeurs des coefficients de raideur et d'amortissement ont été obtenues par identification expérimentale avec le robot HRP-2 [Nakaoka *et al.*, 2007]. Ces articulations sont ensuite intégrées dans le modèle dynamique comme

pour les autres articulations du robot.

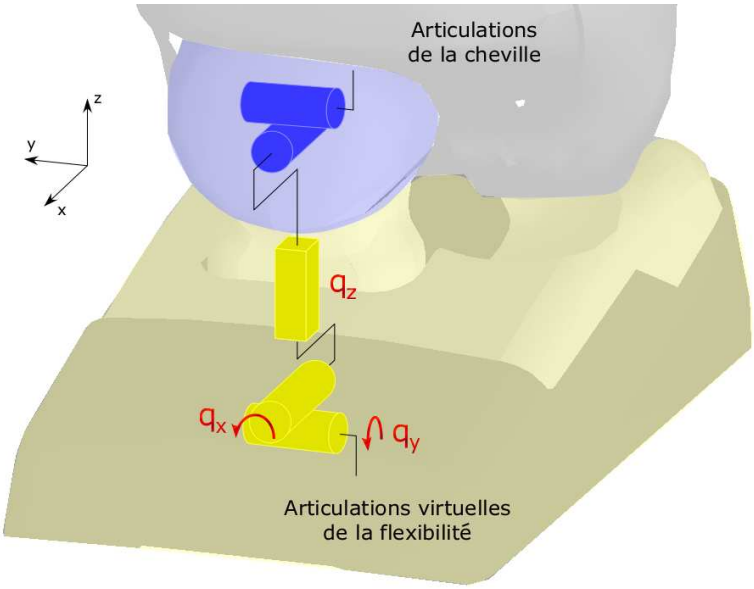


figure 4.2: *L'articulation déformable modélisée dans le pied du robot HRP-2.*

Le robot réel possède également un recouvrement fin en caoutchouc enrobant le pied. Cependant, nous pensons que cette flexibilité est négligeable au regard du mécanisme présent dans le pied. En effet, l'épaisseur de cette couverture est très petite et sa déformation est très faible par rapport au mécanisme de bagues déformables. Nous verrons plus loin en comparant avec un système externe que ce type de flexibilité n'est pas adapté pour absorber les chocs correctement.

4.3 Création d'une semelle déformable et modèle

Nous nous intéresserons plutôt à placer une semelle sous les pieds du robot pour les raisons invoquées plus haut. A la différence de la semelle présentée dans [Yamaguchi *et al.*, 1995], notre semelle, placée sous le pied du robot HRP-2, aura les caractéristiques suivantes :

- une forme simple dans un premier temps,
- absence de capteurs internes comme des potentiomètres,
- interaction directe avec l'environnement (ce n'est pas une liaison déformable entre deux plaques rigides),
- modélisation par un modèle analytique.

Nous utilisons un modèle par éléments finis pour modéliser cette semelle.

4.3.1 Maillage de la semelle

Nous avons utilisé le logiciel GMesh [1] qui permet de concevoir et de mailler très simplement notre semelle en trois dimensions et d'exporter toutes les informations dont nous avons besoin, à savoir les nœuds du maillage, les triangles et les tétraèdres associés.

1. `http://www.geuz.org/gmsh/`

Nous choisissons d'avoir une semelle de forme très simple constituée d'un seul matériau linéaire isotrope (voir figure 4.3). Nous faisons ce choix de la linéarité car les situations que nous désirons simuler ne font pas intervenir des grandes déformations du matériau. Par la même occasion, d'un point de vue purement informatique, cela nous permet de faire des simulations plus rapides.

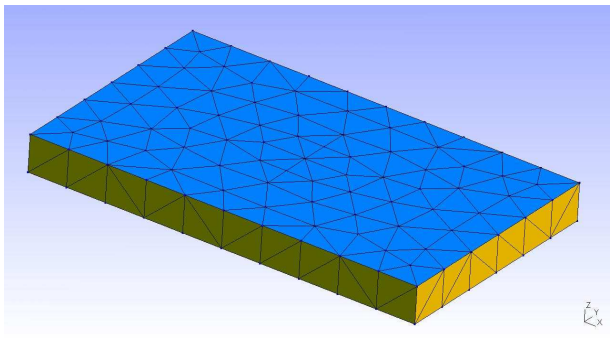


figure 4.3: *Le maillage de la semelle déformable.*

4.3.2 Modèle de la semelle

Le matériau utilisé présente de petites déformations par rapport à la taille de la semelle, par conséquent nous utilisons un modèle en petites déformations. Dans ce cas, nous avons une relation linéaire entre la déformation ϵ et le déplacement $\mathbf{u}$ du point où la déformation est évaluée :

$$\epsilon = \frac{1}{2}(\nabla \mathbf{u} + \nabla^T \mathbf{u}) \tag{4.4}$$

où ∇ est le gradient. La loi de Hooke nous permet d'obtenir la contrainte interne σ, valable dans le cas de petites déformations :

$$\sigma = 2\mu\epsilon + \lambda tr(\epsilon)\mathbf{I}^3 \tag{4.5}$$

où $\mathbf{I}^3$ est la matrice identité de taille 3, μ et λ sont les coefficients de Lamé, calculés à partir du module d'Young (E) et du coefficient de Poisson (ν) qui caractérisent un matériau, et donnés par

$$\mu = \frac{E}{2(1+\nu)},\ \lambda = \frac{E\nu}{(1+\nu)(1-2\nu)} \tag{4.6}$$

Les contraintes σ et les forces volumiques sont liées par :

$$\mathrm{div}(\sigma) + \mathbf{f}_v = 0 \tag{4.7}$$

Enfin, en travaillant dans un cas statique, les déplacements de chaque point du maillage et les forces extérieures sont liés par une relation linéaire :

$$\mathbf{KU} = \mathbf{F} \tag{4.8}$$

où $\mathbf{K}$ est la matrice de raideur de la semelle, $\mathbf{U}$ le vecteur des déplacements de tous les nœuds, $\mathbf{F}$ le vecteur des forces extérieures appliquées sur les nœuds.

Pour calculer la matrice de raideur $\mathbf{K}$, nous calculons les matrices élémentaires $\mathbf{K}_i$ de chaque tétraèdre. En utilisant les notations de [Imbert, 1995], pour un matériau linéaire isotrope nous avons pour chaque tétrahèdre i,

$$\mathbf{K}_i = \mathbf{V}_i \mathbf{B}_i^T \mathbf{C}_i \mathbf{B}_i \tag{4.9}$$

avec $\mathbf{V}_i$ le volume du tétrahèdre i, $\mathbf{B}_i$ la matrice qui lie les déformations et les déplacements ($\epsilon_i = \mathbf{B}_i \mathbf{u}_i$), $\mathbf{C}_i$ est la matrice liant les contraintes et les déformations pour un matériau isotrope ($\sigma_i = \mathbf{C}_i \epsilon_i$). La matrice de raideur de la semelle $\mathbf{K}$ est obtenue en assemblant toutes les matrices de raideur élémentaires $\mathbf{K}_i$.

Dans le cas d'une semelle composée de plusieurs matériaux, les matrices élémentaires $\mathbf{K}_i$ sont calculées avec la relation précédente mais en utilisant des coefficients de Lamé différents (μ_j, λ_j, $j = 1, \cdots, n$ et n le nombre de matériaux).

4.3.3 Réduction aux nœuds de surface

Le maillage de la semelle possède des nœuds fixes, autrement dit avec un déplacement nul dans le repère de la semelle. Ces nœuds sont situés à la frontière entre la semelle et le pied et sont appelés nœuds de Dirichlet. Nous pouvons ainsi réduire la taille de l'équation 4.8 en supprimant les lignes et colonnes de $\mathbf{u}$ et $\mathbf{K}$ correspondant aux nœuds de Dirichlet. Cette réduction est nécessaire pour rendre la matrice $\mathbf{K}$ inversible. Nous obtenons ainsi une relation réduite :

$$\mathbf{K}^r \mathbf{U}^r = \mathbf{F}^r \tag{4.10}$$

d'où

$$\mathbf{U}^r = (\mathbf{K}^r)^{-1} \mathbf{F}^r \tag{4.11}$$

Ce qui nous intéresse est de suivre l'évolution de la semelle, en particulier lorsque la semelle entre en contact avec l'environnement. De fait, il suffit de ne calculer que le déplacement des nœuds qui sont à la surface de la semelle. Cependant, nous ne pouvons pas considérer seulement les nœuds de surface à cause des relations liant chaque nœud, en particulier les nœuds internes au maillage. Pour obtenir une équation ou un système d'équations donnant des relations uniquement entre les nœuds de surface mais en tenant compte de leurs relations avec les nœuds internes, on réarrange l'équation 4.10, en séparant les nœuds de surface s et les nœuds internes i. Ainsi un nœud de surface sera en relation avec un autre nœud de surface qui n'est pas nécessairement voisin mais qui y est lié par un ou plusieurs nœuds internes. Cette opération, proposée dans [Bro-Nielsen et Cotin, 1996], est plus couramment appelée condensation et permet d'accélérer les temps de calcul :

$$\begin{bmatrix} \mathbf{K}_{ss}^r & \mathbf{K}_{si}^r \\ \mathbf{K}_{is}^r & \mathbf{K}_{ii}^r \end{bmatrix} \begin{bmatrix} \mathbf{U}_s^r \\ \mathbf{U}_i^r \end{bmatrix} = \begin{bmatrix} \mathbf{F}_s^r \\ \mathbf{F}_i^r \end{bmatrix} \tag{4.12}$$

En développant cette équation nous obtenons :

$$\begin{aligned} \mathbf{K}^r_{ss}\mathbf{U}^r_s + \mathbf{K}^r_{si}\mathbf{U}^r_i &= \mathbf{F}^r_s \\ \mathbf{K}^r_{is}\mathbf{U}^r_s + \mathbf{K}^r_{ii}\mathbf{U}^r_i &= \mathbf{F}^r_i \end{aligned} \tag{4.13}$$

En isolant $\mathbf{U}^r_i$ dans la deuxième équation et en considérant que nous n'avons pas de forces internes ($\mathbf{F}^r_i = 0$), nous obtenons :

$$\begin{aligned} \left(\mathbf{K}^r_{ss} - \mathbf{K}^r_{si}(\mathbf{K}^r_{ii})^{-1}\mathbf{K}^r_{is}\right)\mathbf{U}^r_s &= \mathbf{F}^r_s \\ \mathbf{K}^r_s\mathbf{U}^r_s &= \mathbf{F}^r_s \end{aligned} \tag{4.14}$$

4.3.4 Mise à jour du modèle de contact

Maintenant que nous avons établi une relation entre les déplacements des nœuds et les forces extérieures, nous devons prendre en compte ce modèle dans le calcul des efforts de contact. Pour cela, nous nous inspirons de l'approche hybride présentée plus haut, en superposant le mouvement rigide et le mouvement de déformation. Nous omettons volontairement l'indice supérieur ^{r} pour plus de clarté. On rappelle le modèle de contact présenté dans le premier chapitre :

$$\mathbf{v} = \mathbf{\Lambda f} + \mathbf{v}_{\text{libre}} \tag{4.15}$$

On augmente cette relation de celle donnant la déformation de la semelle, en ajoutant la vitesse de déformation des nœuds en contact $\dot{\mathbf{U}}^r_c$:

$$\mathbf{v} = \mathbf{\Lambda f} + \mathbf{v}_{\text{libre}} + \dot{\mathbf{U}}_c \tag{4.16}$$

Ce vecteur des vitesses de déformation $\dot{\mathbf{U}}_c$ peut être calculé à partir du vecteur des déformations des nœuds en contact $\mathbf{U}_c$ en utilisant une méthode d'intégration d'Euler :

$$\dot{\mathbf{U}}_c = \frac{\mathbf{U}^{t+dt}_c - \mathbf{U}^t_c}{dt} = \frac{(\mathbf{K}_c)^{-1}\mathbf{F}_c - \mathbf{U}^t_s}{dt} \tag{4.17}$$

où $\mathbf{K}_c$ est la matrice de raideur des noeuds de surface en contact. Pour déterminer la matrice de raideur $\mathbf{K}_c$ et les déplacements $\mathbf{U}_c$, nous séparons les nœuds en contact c de ceux qui ne sont pas en contact n en effectuant comme pour $\mathbf{K}_s$ et $\mathbf{U}_s$ une opération de condensation :

$$\begin{bmatrix} \mathbf{K}_{cc} & \mathbf{K}_{cn} \\ \mathbf{K}_{nc} & \mathbf{K}_{nn} \end{bmatrix} \begin{bmatrix} \mathbf{U}_c \\ \mathbf{U}_n \end{bmatrix} = \begin{bmatrix} \mathbf{F}_c \\ \mathbf{F}_n \end{bmatrix} \tag{4.18}$$

En développant cette équation, nous obtenons :

$$\begin{aligned} \mathbf{K}_{cc}\mathbf{U}_c + \mathbf{K}_{cn}\mathbf{U}_n &= \mathbf{F}_c \\ \mathbf{K}_{nc}\mathbf{U}_c + \mathbf{K}_{nn}\mathbf{U}_n &= \mathbf{F}_n \end{aligned} \tag{4.19}$$

En isolant $\mathbf{U}_n$ dans la deuxième opération et en considérant que les nœuds de surface qui ne sont pas en contact ne sont soumis à aucune force ($\mathbf{F}_n = 0$), nous obtenons :

$$\begin{aligned} \left(\mathbf{K}_{cc} - \mathbf{K}_{cn}\mathbf{K}_{nn}^{-1}\mathbf{K}_{nc}\right)\mathbf{U}_c &= \mathbf{F}_c \\ \mathbf{K}_c\mathbf{U}_c &= \mathbf{F}_c \end{aligned} \tag{4.20}$$

En injectant l'équation 4.17 dans l'équation 4.16, nous obtenons :

$$\begin{aligned} \mathbf{v} &= \left(\mathbf{\Lambda} + \frac{\mathbf{K}_c^{-1}}{dt}\right)\mathbf{f} + \left(\mathbf{v}_{\text{libre}} - \frac{\dot{\mathbf{U}}_c^t}{dt}\right) \\ &= \mathbf{W}\mathbf{f} + \mathbf{V}_{\text{libre}} \end{aligned} \tag{4.21}$$

Nous obtenons une relation semblable au cas rigide. Nous pouvons donc résoudre ce système de la même manière que dans le cas rigide.

Une fois les forces de contact calculées, nous mettons à jour les nœuds en contact puis les nœuds de surface qui ne sont pas en contact :

$$\begin{aligned} \mathbf{U}_c &= \mathbf{K}_c^{-1}\mathbf{F}_c \\ \mathbf{U}_n &= \left(\mathbf{K}_{nn} - \mathbf{K}_{nc}\mathbf{K}_{cc}^{-1}\mathbf{K}_{cn}\right)^{-1}\left(-\mathbf{K}_{nc}\mathbf{K}_{cc}^{-1}\mathbf{F}_c\right) \end{aligned} \tag{4.22}$$

4.3.5 Remarques

Si nous plaçons un corps déformable autour du robot, il n'est pas nécessaire d'avoir un modèle d'impact. En effet, il n'y a pas de discontinuité des vitesses pour la partie rigide. La partie déformable diminue progressivement la vitesse du corps le long de la normale au contact.

L'implémentation de la mise à jour du modèle de contact dans notre simulateur s'est faite aisément grâce à la modularité de notre programme comme indiquée dans le chapitre précédent, en l'occurrence il s'agit d'un simple terme à ajouter au modèle de contact.

Ce modèle traduit la superposition du mouvement rigide et du mouvement de déformation qui s'approche du modèle hybride présenté plus haut.

Il est à noter que l'utilisation du modèle en statique est justifiée car dans notre cas la masse des systèmes que nous simulons (robots humanoïdes) est beaucoup plus grande que celle de la semelle et par conséquent les termes en vitesse et en accélération de déformation de la semelle ont une influence négligeable sur le mouvement global. Dans le cas dynamique, on peut montrer que nous obtenons une forme du modèle de contact similaire au cas statique, au prix de lourds calculs supplémentaires pour une amélioration de la précision négligeable.

4.4 Premières simulations et comparaison

Dans cette partie, nous cherchons à déterminer le matériau de notre semelle simple qui absorbe le mieux les chocs et nous cherchons à connaître les bénéfices et les problèmes de notre semelle déformable et de l'articulation flexible présentée en partie 4.2. Toutes nos mesures sont prises au centre de masse

du torse de HRP-2.

Nous fixons nos semelles déformables sous la plante des pieds du robot. Ces semelles sont de simples parallélépipèdes mesurant 22cm×13cm×2cm (voir figure 4.4).

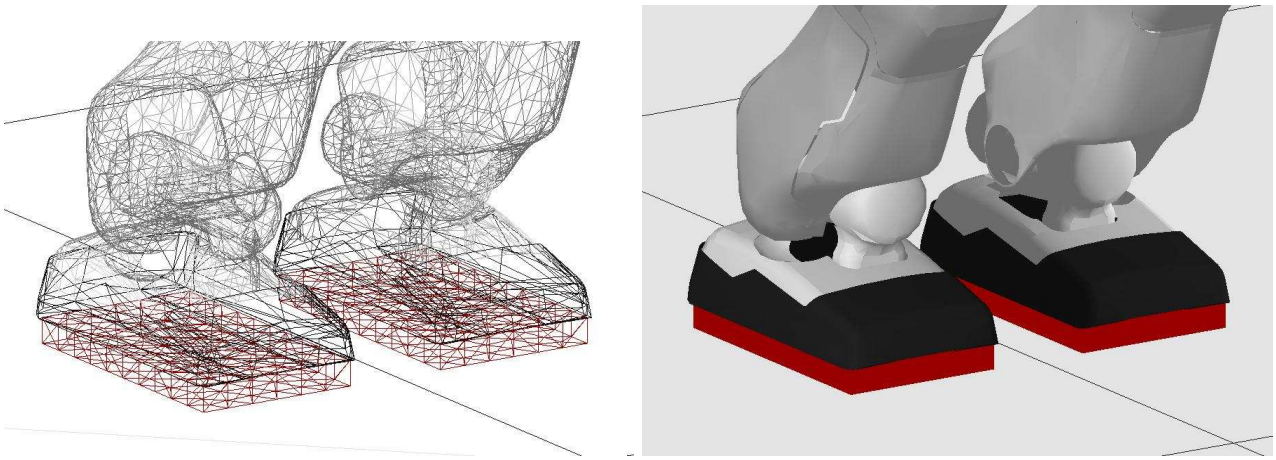


figure 4.4: *Semelle sous les pieds du robot HRP-2.*

4.4.1 Détermination des caractéristiques du matériau

Un matériau est caractérisé par trois paramètres : ses dimensions, son module d'Young E et son coefficient de Poisson ν. Pour déterminer les deux derniers paramètres, nous considérons deux critères :

- l'accélération du point de référence du robot (ici le centre des hanches du robot) selon l'axe vertical. C'est selon cette direction que les effets des impacts et de la déformation de la semelle sont les plus visibles, on cherchera donc à limiter ces effets,
- l'inclinaison latérale du torse qu'on cherchera également à limiter.

Ces deux critères sont antagonistes. En effet, plus la semelle est déformable, plus l'absorption des chocs est forte et donc moins l'accélération du point de référence est élevée, mais plus l'inclinaison du torse est forte (cette inclinaison est due à l'inertie du robot créée par le mouvement de marche du robot). La vitesse de la partie rigide du robot sera amortie progressivement mais si l'inclinaison du torse est trop forte, le robot aura plus de chances de perdre l'équilibre et donc de tomber. Nous allons donc chercher le meilleur compromis entre ces deux critères.

Nous avons choisi une épaisseur de la semelle de 2cm, que nous pensons être suffisant au regard des déformations que la semelle doit subir lors de nos simulations. Il est important de s'assurer que la déformation maximale n'atteint jamais le domaine plastique dans la pratique car dans ce cas le comportement obtenu sera proche de celui d'un corps rigide.

Le coefficient de Poisson d'un matériau est inférieur à $0,5$. Un matériau parfaitement isotrope possède un coefficient de Poisson de $0,25$. Si le coefficient de Poisson vaut $0,5$, le matériau est parfaitement incompressible. Nous avons alors choisi de faire varier ν entre $0,01$ et $0,49$.

De même nous faisons varier le module d'Young. Nous désirons avoir un matériau élastique tel qu'une gomme qui possède un module d'Young compris entre $1,5$ et 5MPa. Nous choisissons alors de

faire varier E entre 1 et 10MPa.

Pour déterminer la réduction de l'accélération du torse du robot selon l'axe vertical, nous lâchons le robot à 1cm au-dessus du sol, avec toutes ses articulations à zéro (tous les membres tendus). Quant à la réduction de l'inclinaison du torse, nous prenons le robot debout sur le sol, et nous déplaçons petit à petit son centre de gravité vers l'avant en inclinant le torse jusqu'à un angle avec la verticale de $15,3$ degrés, les jambes restant à la verticale. Nous mesurons la différence entre l'inclinaison réelle et celle désirée. Les résultats de la simulation sont présentés en figures 4.5 et 4.6. Nous observons l'antagonisme indiqué plus haut. Nous avons bien une diminution de l'accélération du point de référence au fur et à mesure que la semelle devient déformable. Cependant cette déformation (E inférieur à 1MPa et ν proche de zéro) entraîne une inclinaison plus forte du torse, ce qui amène le robot à perdre l'équilibre. A la suite de ces résultats, nous pouvons choisir un matériau qui remplit les critères que nous avons imposés plus haut. Au vu des courbes, nous choisissons $E = 3$MPa et $\nu = 0,4$ qui semble un bon compromis entre une accélération du point de référence et une inclinaison du torse faibles. Notons que ce matériau correspond à une gomme.

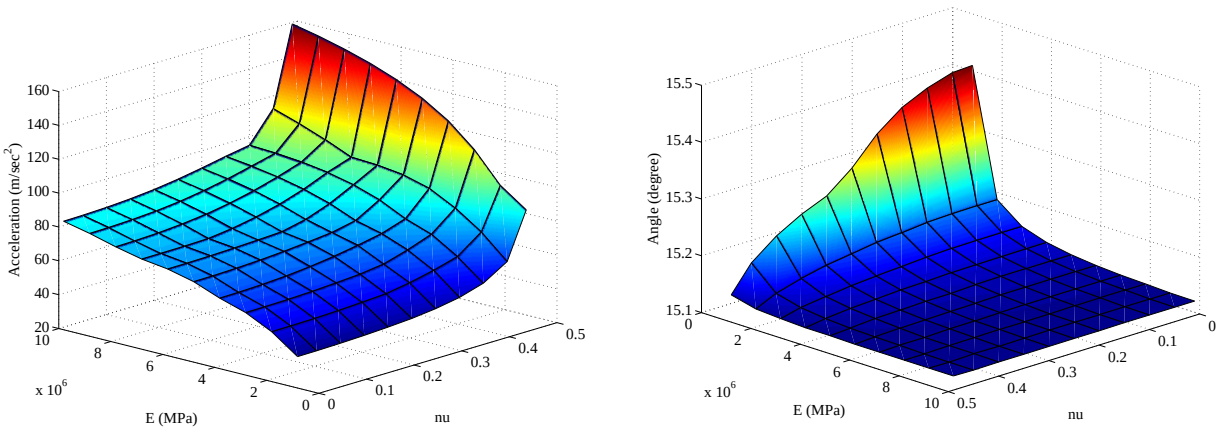


figure 4.5: *A gauche : évolution de l'accélération du centre de masse du robot selon l'axe vertical en fonction des caractéristiques du matériau. A droite : évolution de l'inclinaison latérale du torse. L'antagonisme est bien visible.*

4.4.2 Comparaison

Pour déterminer les avantages et inconvénients des différents systèmes étudiés ici (articulation flexible et semelle), nous effectuons des simulations comparatives dans les cas suivants :

- robot sans aucune flexibilité (autrement dit, le modèle rigide pur tel que décrit dans les deux premiers chapitres),
- robot avec semelle déformable,
- robot avec articulation flexible.

La simulation du robot sans aucune flexibilité nous servira de référence pour notre comparaison. Nos semelles sont les mêmes que précédemment avec les caractéristiques déterminées précédemment. Dans

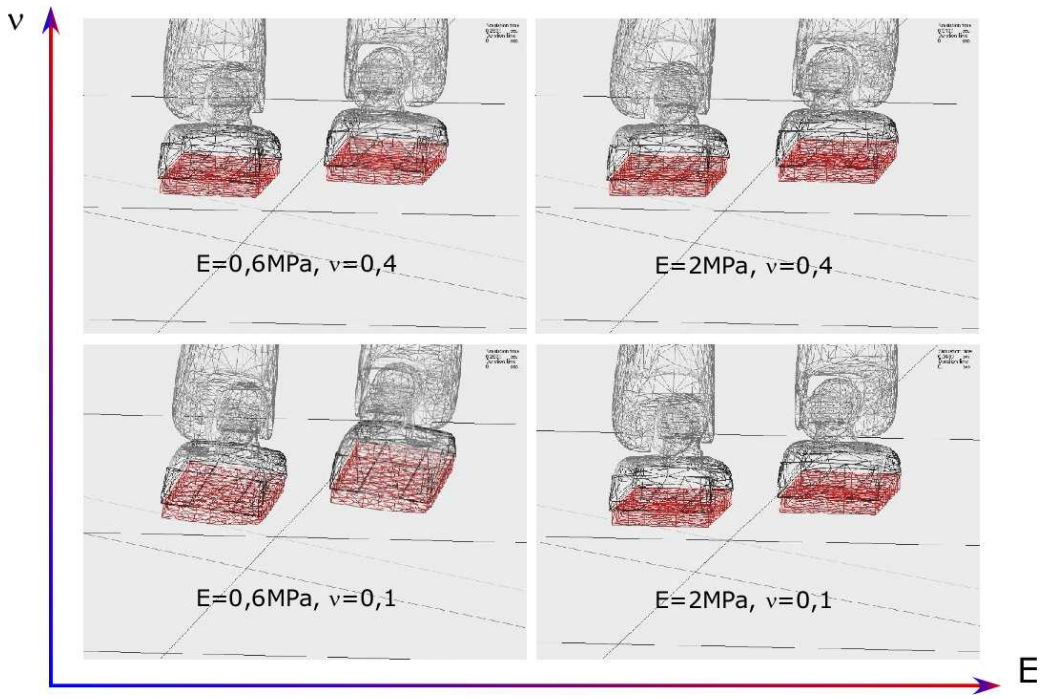


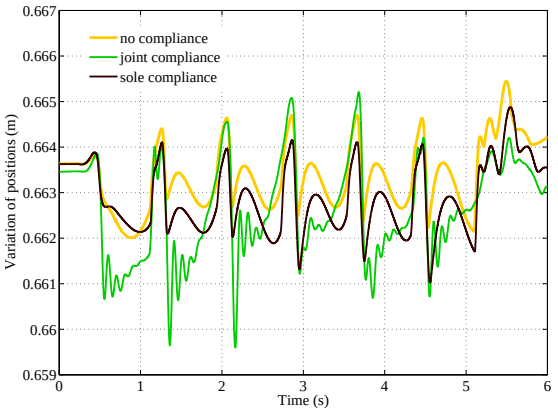
figure 4.6: *Simulations de la marche avec semelles en fonction du module d'Young et du coefficient de Poisson.*

toutes les simulations que nous réalisons ici, nous demandons au robot HRP-2 de faire quelques pas de marche sur un sol plat, dont les positions ont été créées en utilisant le générateur de marche implémenté sur le robot [Stasse *et al.*, 2008], plus précisément la marche est composée de six pas pour une durée de six secondes. La vitesse maximale du point de référence est de $0,3\text{m.s}^{-1}$ et la longueur du pas maximale est de $0,45\text{m}$.

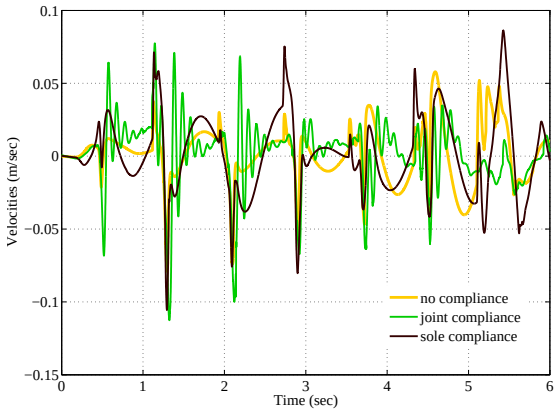
Nous avons tracé pour les trois cas les courbes de l'évolution de la position du centre de masse du robot, sa vitesse, son accélération selon l'axe vertical de la gravité, et les couples articulaires des jambes (voir figure 4.7). Encore une fois, c'est selon l'axe vertical que les effets de la déformation et de l'impact sont les plus visibles.

Nous pouvons remarquer que l'évolution de la position du point de référence est identique dans le cas sans flexibilité et dans celui avec les semelles. L'explication tient dans le fait qu'après un impact et en absence de perturbations, les semelles déformables se comportent comme des corps rigides, car les semelles, en se déformant, absorbent les chocs et elles gardent leur nouvelle forme jusqu'à ce que les pieds décollent à nouveau du sol. Dans ces deux cas, les courbes de positions ne nous apprennent rien de très intéressant et ne montrent pas l'apport d'une semelle déformable. En revanche, le cas avec les articulations flexibles est différent car nous pouvons observer que l'évolution de la position du centre de masse du torse est perturbée par des oscillations. Ces oscillations sont évidemment dues aux ressorts-amortisseurs présents dans les chevilles de HRP-2 et font apparaître les problèmes d'une telle modélisation en simulation. Grâce au facteur d'amortissement, elles diminuent cependant rapidement.

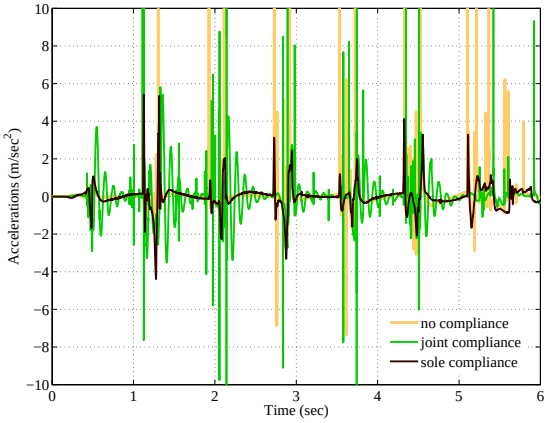
Il en est de même pour l'évolution de la vitesse du point de référence ainsi que les couples articulaires du genou. En revanche pour l'accélération du centre de masse du torse, nous pouvons remarquer l'intérêt



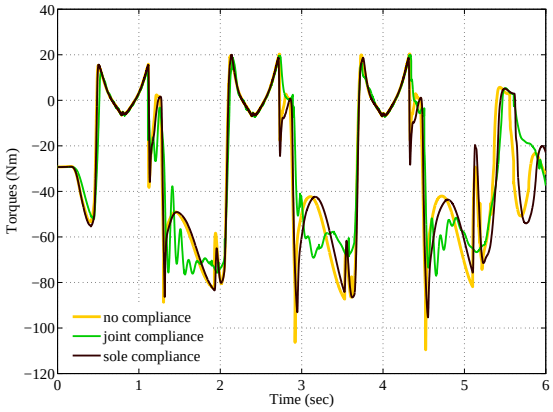
(a) Evolution de la position du centre de masse du robot selon l'axe vertical. La position dans le cas avec semelle a été décalée de l'épaisseur de la semelle (2cm) pour la comparaison.



(b) Evolution de la vitesse du centre de masse selon l'axe vertical.



(c) Evolution de l'accélération du centre de masse selon l'axe vertical. Ici les courbes ont été tronquées pour aller de -10m.s^{-2} à 10m.s^{-2}.



(d) Evolution des couples articulaires des jambes du robot.

figure 4.7: *Evolution de différents données du centre de masse du robot pendant la marche dans les cas sans flexibilité (en jaune), avec articulations flexibles (en vert) et semelles déformables (en noir).*

des semelles. En effet, les courbes font apparaître des pics d'accélération au moment où il y a un impact avec le sol dans les cas sans flexibilité et avec flexibilité dans le pied. L'amplitude des pics est cependant différente selon le cas, trivialement, l'amplitude est plus forte dans le cas où il n'y a aucun mécanisme d'absorption des chocs (environ 50m.s^{-2} contre 30m.s^{-2} avec l'articulation flexible). Il est à noter que la flexibilité dans les articulations n'atténue pas suffisamment ces pics malgré la présence de ressorts-amortisseurs. Dans le cas avec les semelles, l'amplitude de l'accélération est très modérée (de l'ordre de 5m.s^{-2}).

4.4.3 Résumé et remarques

En l'absence de toute flexibilité, le robot est soumis à de grands chocs. La présence de semelles déformables semble atténuer efficacement les chocs sans créer d'oscillations et permet d'obtenir un comportement globalement régulier, contrairement aux articulations flexibles qui, même en limitant l'impact sur le sol, créent de fortes perturbations dans la loi de commande.

Même en plaçant une semelle déformable sous les pieds du robot, nous observons des pics d'accélération non négligeables. Or le but de ces semelles est d'avoir des impacts les plus petits possibles. La semelle que nous avons choisie jusqu'à présent est trop simple pour résoudre ce problème. Nous étudierons plus loin plusieurs formes de semelles pour atténuer ces chocs mais il est tout aussi clair que cela devrait idéalement se faire de pair avec le générateur de marche.

Il faut cependant remarquer que les constantes choisies pour reproduire le mécanisme interne passif réel de HRP-2 n'ont pas été optimisées que pour la marche mais résultent d'un compromis entre différents critères limités par la conception du mécanisme (matériau utilisé, fabrication, intégration, etc.), en particulier le système n'est pas constitué d'articulations ponctuelles. Le mécanisme réel se comporte par ailleurs comme un filtre mécanique passe-bas et répond à d'autres considérations qui ne sont pas traitées ici. Ainsi notre comparaison n'est en réalité pas très équitable, il faut donc la prendre avec prudence et simplement comme une tendance observée.

4.5 Extension du modèle de la semelle

Pour simuler le contact entre la semelle et l'environnement, nous avons jusqu'à présent considéré un sol plat pour lequel la détection de collision est très simple. Contrairement à la détection de collision utilisée habituellement et présentée dans le chapitre précédent, nous avons considéré simplement que lorsqu'un nœud a une position selon l'axe vertical inférieure à la hauteur du sol, alors il est en contact. La force de contact est alors directement appliquée en ce nœud, autrement dit le point de contact et le nœud sont les mêmes. Dans le cas où l'environnement n'est pas plat, typiquement un sol granuleux (voir figure 4.8) [2], cette démarche n'est plus valable et nous devons recourir à nouveau à une détection de collision plus sophistiquée, en particulier les points de contact ne sont plus nécessairement les nœuds du maillage, en particulier cela crée des discontinuités. Or le souci dans la méthode par éléments finis est d'assurer la continuité des fonctions qui réalisent l'équilibre. Comme les forces de contact sont appliquées aux points de contact, le problème est alors de déterminer la déformation associée en ces points.

Pour résoudre ce problème, il est possible d'affiner le maillage autour du point de contact en faisant en sorte que le point de contact et un nœud soient confondus [Debunne *et al.*, 2001, Grinspun *et al.*, 2002]. Cette possibilité permet une résolution beaucoup plus fine du contact. L'inconvénient de cette approche avec les éléments finis est qu'il est alors nécessaire de recalculer la matrice de rigidité du nouveau maillage, entraînant un net ralentissement de la simulation. Il est cependant possible

2. Notons qu'ici notre but immédiat n'est pas de concevoir une marche permettant à un robot de se mouvoir sur un sol non plat, mais qu'à terme, une des utilisations pratiques des semelles est de se mouvoir sur tous types de sols.

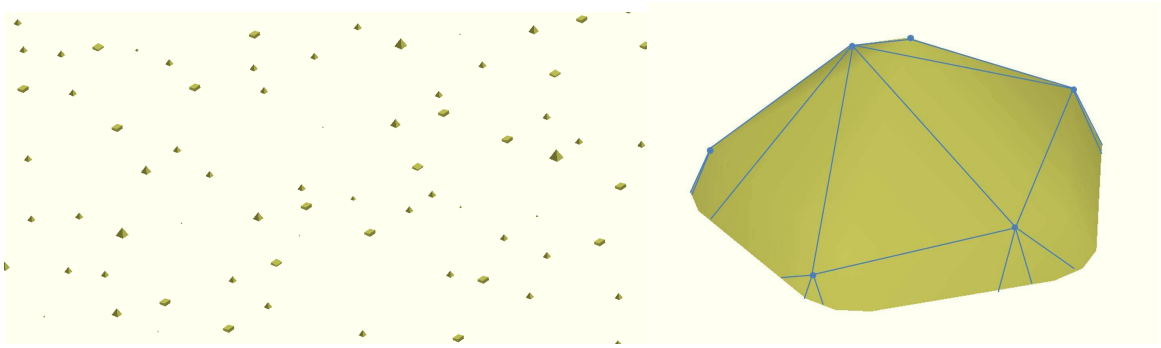


figure 4.8: *Exemple d'environnement non régulier, ici un sol granuleux.*

de faciliter le remaillage en considérant des points arbitraires dans un élément [Sifakis *et al.*, 2007]. Une autre solution, que nous avons adoptée, est de répartir la force de contact sur les nœuds immédiatement voisins au point de contact. Il s'agit évidemment d'une approximation mais qui, quand le maillage est assez fin, n'est pas trop grossière. Pour cela, nous introduisons une matrice de répartition $\mathbf{P}$. Une fois que les forces de contact sont calculées en un point de contact, la matrice $\mathbf{P}$ répartit ces efforts sur les nœuds voisins.

Tous les objets de l'environnement sont représentés par des listes de triangles, les semelles également, par conséquent nous effectuons une détection de collision entre des paires de triangles comme pour le cas rigide en utilisant la librairie PQP (voir le chapitre précédent). Nous obtenons ainsi une paire de points de contact ainsi que les triangles associés.

Les forces de contact sont réparties en fonction de la distance entre le point de contact et les nœuds du triangle en contact associé. Nous expliquons ceci par un exemple simple représenté en figure 4.9.

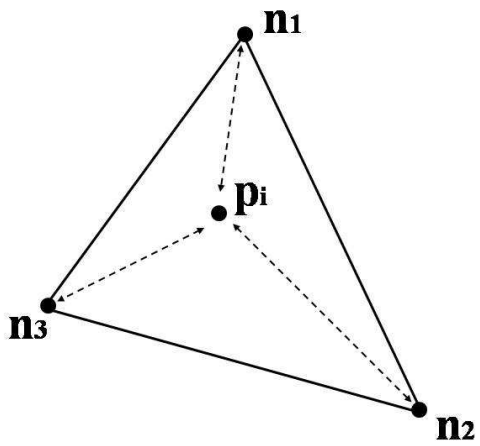


figure 4.9: *Répartition des efforts calculés au point de contact sur les nœuds du triangle associé.*

$\mathbf{p}_i$ est le point de contact, $\mathbf{n}_1$, $\mathbf{n}_2$ et $\mathbf{n}_3$ sont les trois nœuds associés au triangle en contact. $\mathbf{f}_{p_i}$ est la force de contact calculée en $\mathbf{p}_i$, $\mathbf{f}_{n_1}$, $\mathbf{f}_{n_2}$ et $\mathbf{f}_{n_3}$ sont les forces réparties sur les trois nœuds telles que :

$$\mathbf{f}_{p_i} = d_1\mathbf{f}_{n_1} + d_2\mathbf{f}_{n_2} + d_3\mathbf{f}_{n_3} \tag{4.23}$$

avec d_j les coefficients de répartitions pour les nœuds n_j tels que $\sum d_j = 1$, $j = [1, 2, 3]$. Puisque nous utilisons une répartition géométrique des forces, ces coefficients nous donnent également la distance du point $\mathbf{p}_i$ par rapport aux nœuds $\mathbf{n}_j$

$$\mathbf{p}_i = d_1\mathbf{n}_1 + d_2\mathbf{n}_2 + d_3\mathbf{n}_3 \tag{4.24}$$

Connaissant les positions de $\mathbf{p}_i$ et $\mathbf{n}_j$, nous pouvons obtenir les coefficients d_j

$$\begin{pmatrix} d_1 \\ d_2 \\ d_3 \end{pmatrix} = \begin{bmatrix} x_{n_1} & x_{n_2} & x_{n_3} \\ y_{n_1} & y_{n_2} & y_{n_3} \\ z_{n_1} & z_{n_2} & z_{n_3} \end{bmatrix}^{-1} \begin{pmatrix} x_{p_i} \\ y_{p_i} \\ z_{p_i} \end{pmatrix} \tag{4.25}$$

sous réserve que la matrice formée par les coordonnées des nœuds est inversible, ce qui dans notre cas sera toujours vrai car les coordonnées des nœuds ne sont pas nulles. Nous pouvons ensuite écrire l'équation 4.23 de la manière suivante

$$\mathbf{f}_{p_i} = \mathbf{P}_i^T \begin{pmatrix} F_{n_1} \\ F_{n_2} \\ F_{n_3} \end{pmatrix} \tag{4.26}$$

avec $\mathbf{P}_i$ la matrice de répartition élémentaire liée au point de contact $\mathbf{p}_i$ et qui s'écrit

$$\mathbf{P}_i = \begin{bmatrix} I_{d_1} & I_{d_2} & I_{d_3} \end{bmatrix} = \begin{bmatrix} d_1 & 0 & 0 & d_2 & 0 & 0 & d_3 & 0 & 0 \\ 0 & d_1 & 0 & 0 & d_2 & 0 & 0 & d_3 & 0 \\ 0 & 0 & d_1 & 0 & 0 & d_2 & 0 & 0 & d_3 \end{bmatrix} \tag{4.27}$$

La matrice de répartition $\mathbf{P}$ est obtenue par assemblage des matrices $\mathbf{P}_i$. Par exemple pour l'exemple de la figure 4.10, nous avons les matrices élémentaires suivantes

$$\mathbf{P}_1 = \begin{bmatrix} I_{d_1} & I_{d_2} & I_{d_3} \end{bmatrix}, \ \mathbf{P}_2 = \begin{bmatrix} I_{d_4} & I_{d_5} & I_{d_6} \end{bmatrix} \tag{4.28}$$

ce qui donne la matrice $\mathbf{P}$ suivante

$$\mathbf{P} = \begin{bmatrix} I_{d_1} & I_{d_2} & I_{d_3} & 0 \\ I_{d_4} & I_{d_5} & 0 & I_{d_6} \end{bmatrix} \tag{4.29}$$

Maintenant nous devons prendre en compte cette répartition dans le calcul des déformations. Nous reprenons l'équation 4.21

$$\mathbf{v} = \left(\mathbf{\Lambda} + \frac{\mathbf{K}_{cp_i}^{-1}}{dt} \right) \mathbf{f}_{p_i} + \left(\mathbf{v}_{\text{libre}} - \frac{\dot{\mathbf{U}}_{cp_i}^t}{dt} \right) \tag{4.30}$$

avec $\mathbf{K}_{cp_i}$ et $\mathbf{U}_{cp_i}$ sont la matrice de raideur et la déformation du point de contact $\mathbf{p}_i$. On rappelle que nous ne connaissons pas ces grandeurs mais on peut maintenant les exprimer grâce à la matrice de

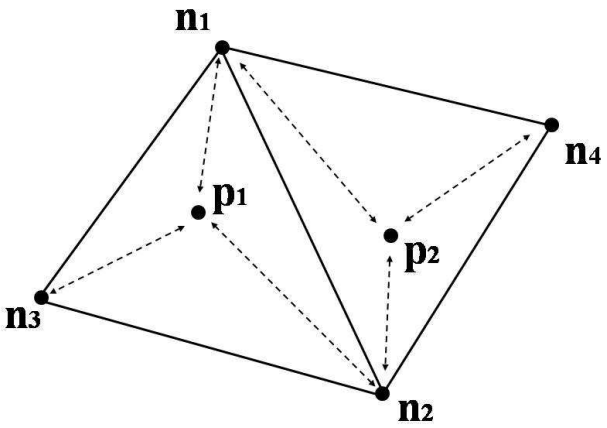


figure 4.10: *Répartition des efforts dans le cas de deux points de contact.*

répartition $\mathbf{P}$ en fonction de la matrice de raideur et de la déformation des nœuds associés

$$\mathbf{K}_{cp_i}^{-1} = \mathbf{P}\mathbf{K}_c^{-1}\mathbf{P}^T,\ \dot{\mathbf{U}}_{cp_i} = \mathbf{P}\dot{\mathbf{U}}_c \tag{4.31}$$

d'où nous obtenons le modèle de contact suivant

$$\begin{aligned} \mathbf{v} &= \left(\mathbf{\Lambda} + \frac{\mathbf{P}\mathbf{K}_c^{-1}\mathbf{P}^T}{dt}\right)\mathbf{f}_{p_i} + \left(\mathbf{v}_{\text{libre}} - \frac{\mathbf{P}\dot{\mathbf{U}}_c^t}{dt}\right) \\ &= \mathbf{W}^n\mathbf{f} + \mathbf{V}_{\text{libre}}^n \end{aligned} \tag{4.32}$$

Avec ce modèle, nous avons réalisé des simulations de marche sur un sol granuleux avec le robot HRP-2. Ici encore nous nous servons de la trajectoire donnée par le générateur de marche de [Stasse *et al.*, 2008] qui n'est pas adaptée dans le cas de marches sur un sol irrégulier. Par conséquent le robot finit par tomber (voir figure 4.11). Rappelons qu'ici nous ne nous intéresserons pas à créer une marche sur des sols irréguliers car elle fait intervenir des problèmes, en particulier de discontinuités et de contrôle, très complexes à résoudre. Ici nous montrons juste que la semelle se déforme bien au contact d'une aspérité. Encore une fois, nous avons réalisé une approximation, ainsi la semelle n'épouse pas parfaitement la forme des aspérités mais cette approximation reste acceptable.

4.6 Formes de semelle

Avec tous ces résultats, nous pouvons aussi simuler des semelles complexes de formes diverses (voir figure 4.12), ceci fait l'objet d'un travail de recherche dans le cadre d'un séjour post-doctoral en cours actuellement au JRL.

Ici nous nous proposons d'aller un peu plus loin dans la conception des semelles, pour atténuer les impacts. Une manière d'observer ces vibrations est de regarder l'évolution de l'accélération du torse, comme nous l'avons fait dans la partie 4.4.2. Or nous avons vu dans cette partie que les semelles simples que nous avons placées sous les pieds du robot HRP-2 atténuaient déjà bien les vibrations. Cependant il reste une accélération non négligeable du torse lorsque les pieds touchent le sol (à chaque

pas, l'accélération maximale en valeur absolue est de l'ordre de 5m.s^{-2}). C'est cette accélération que nous cherchons ici à diminuer en simulation pour obtenir une marche avec des impacts les plus faibles possibles, et sans avoir à changer la loi de commande actuellement implémentée. Nous nous intéresserons donc ici à créer des semelles de formes différentes qui diminuent l'accélération du torse. Nous pourrions également nous intéresser à stabiliser la position du torse. En effet, nous avons observé dans la partie 4.4.2 des légères oscillations dans l'évolution de la position du torse. Or, le générateur de marche de [Stasse *et al.*, 2008] a été conçu pour que la position du torse selon l'axe vertical reste constante. Cependant, en observant les courbes, on remarque que les oscillations sont de l'ordre de 2mm, ce qui est très petit par rapport à la taille du robot. Nous avons jugé que l'intérêt de diminuer ces oscillations était faible par rapport à celui de diminuer les pics d'accélérations. Ici nous considérerons une marche sur un sol plat. Nous ne chercherons pas ici à obtenir une solution finale avec des résultats mais plutôt à donner des pistes pour améliorer les semelles déformables.

Les semelles montrées en figure 4.12 ne sont pas très intéressantes car elles n'ont été dessinées qu'à titre d'exemples, sans considération pour le problème que nous souhaitons étudier maintenant.

Une première solution envisagée est d'avoir des semelles avec des coussinets. L'idée est d'avoir deux couches de matériaux : une première couche, collée au pied du robot, composée d'un matériau qui ne se déforme pas trop, et une seconde couche, qui touche le sol, consistant en deux coussinets de tailles différentes, un plus grand à l'arrière qu'à l'avant, et composés d'un matériau plus mou (voir figure 4.13). Il s'agit en quelque sorte de se rapprocher des semelles de chaussures pour les humains, pour que, lorsqu'un pied arrive au sol, la seconde couche, plus molle, absorbe une bonne partie de l'impact, et la première couche, plus dure, absorbe le résidu.

Nous avons testé en simulation cette semelle sur le robot HRP-2. Nous avons pris les caractéristiques suivantes : pour la première couche, nous avons $E = 3\text{MPa}$ et $\nu = 0,25$ et pour la seconde couche, $E = 1\text{MPa}$ et $\nu = 0,25$. Nous avons représenté l'évolution de l'accélération du torse selon l'axe vertical en fonction du temps (voir figure 4.14).

Nous pouvons observer que les pics sont toujours présents, mais que leur amplitude est réduite : maintenant l'amplitude maximale en valeur absolue est de 2m.s^{-2} contre 5m.s^{-2} auparavant. Nous avons essayé avec d'autres valeurs de modules d'Young et de coefficients de Poisson, nous n'avons pas obtenu de meilleurs résultats.

Une deuxième solution envisagée est d'utiliser une semelle avec des bords arrondis (voir figure 4.15). L'idée est d'obtenir une marche du robot proche de la marche humaine : le pied ne se pose pas sur le sol parallèlement au sol, mais en posant d'abord le talon, puis en basculant progressivement le pied vers l'avant. L'intérêt d'utiliser des bords arrondis est d'avoir une surface de contact grande lors du contact avec le sol, au lieu d'une ligne de contact comme c'est le cas avec des pieds rigides. Pour pouvoir essayer cette semelle, il faut modifier le générateur de marche de [Stasse *et al.*, 2008]. Jusqu'à présent, ce générateur nous a donné un mouvement des pieds parallèles au sol. En utilisant ce mouvement de marche, nous observons que le robot est très instable : dès les premiers pas, le robot bascule vers l'arrière et tombe. Avec ce générateur de marche, il est déjà possible d'obtenir des trajectoires de pied avec un angle d'attaque non nul. Cependant, nous n'avons malheureusement pas pu essayer ces trajectoires en

raison d'erreurs d'implémentation qui n'ont pas pu être résolues faute de temps.

4.6.1 Remarques

Nous avons remarqué que les formes que nous avons choisies pour tenter d'atténuer les chocs ne semblent pas résoudre complètement le problème. Ceci est assez normal puisque nous n'avons considéré ici que la forme des semelles. Or il y a plusieurs autres paramètres à prendre en compte tels que les caractéristiques des matériaux, les dimensions de la semelle, le générateur de marche, etc. En particulier, pour la deuxième solution des semelles à bords arrondis, la manière dont le pied va arriver sur le sol va influer sur la forme de ces bords. En effet, pour augmenter la stabilité du robot, il est préférable de garder une surface de contact qui soit la plus grande possible. Or la stabilité du robot est dépendante de la position du ZMP, qui elle-même va permettre de générer une trajectoire de la marche. Si on a une ligne de contact, on ne peut pas utiliser le ZMP pour générer une trajectoire. En planifiant les déformations de la semelle, nous pouvons obtenir un profil du ZMP qui respecte la stabilité du robot, et à partir de là optimiser la forme des bords.

Nous avons souhaité ne pas modifier la loi de commande. Cependant, si on veut atténuer les vibrations ou les déviations de trajectoire, il est nécessaire de considérer des lois de commande qui corrigent les erreurs de position. En particulier, on peut créer un stabilisateur, inspiré de celui implémenté actuellement sur le robot et qui corrige les erreurs introduites par les articulations flexibles.

Pour la détermination des dimensions de la semelle, nous pouvons considérer la limite d'élasticité, qui est la contrainte à partir de laquelle un matériau commence à se déformer. Dans un cas de déformation élastique, qui est notre cas, cette limite est proportionnelle à la déformation maximale. Cette déformation est dépendante du déplacement maximal des nœuds du maillage et de l'épaisseur minimale de la semelle. Si l'on choisit un matériau avec un module d'Young donné et une limite d'élasticité donnée, nous pouvons déterminer l'épaisseur minimale que la semelle doit avoir pour garder un comportement élastique.

4.7 Conclusion

Dans ce chapitre, nous nous sommes intéressés à la prise en compte des flexibilités dans notre simulateur. Cette prise en compte est une extension naturelle de notre simulateur et est essentielle car les robots actuels présentent des flexibilités qui ne sont pas négligeables pour la réalisation de mouvements ; en particulier le robot HRP-2 possède des flexibilités internes, au niveau des chevilles, représentées dans notre simulateur par des ressorts-amortisseurs virtuels. Ces flexibilités permettent d'atténuer les vibrations observées par exemple lors de la marche. Nous avons confronté ce type de flexibilités avec des semelles déformables placées sous les pieds du robot et modélisées par une méthode d'éléments finis, dans le cas d'une marche sur un sol plat. Nous avons cherché un premier matériau qui réduit le balancement du torse et son accélération. Avec ce matériau, nous avons observé que les semelles déformables absorbent mieux les impacts que la flexibilité interne, que les vibrations sont

moindres et que la stabilité du robot reste assurée.

Un des avantages des flexibilités externes telles que les semelles déformables est son adaptation aux aspérités de l'environnement. En effet, à terme, on souhaite se passer des flexibilités internes et faire évoluer les robots dans tous types d'environnements ; en particulier nous désirons concevoir des lois de commande permettant de faire marcher les robots humanoïdes sur des sols incertains tout en conservant la stabilité. Cependant, dans le cadre de cette thèse qui se réduit à la simulation, nous ne nous sommes pas attardés à ce problème. Nous avons donc étendu notre modèle pour pouvoir simuler des déformations sur des obstacles de type graviers. Nous avons choisi un modèle simplifié dans un premier temps, basé sur des considérations géométriques. Avec ce modèle, nous avons recréé un environnement où le sol est granuleux et fait marcher le robot. Dans le futur nous comptons utiliser d'autres méthodes permettant une résolution plus fine.

L'utilisation de telles semelles permettra également de concevoir des modèles de marche plus proches de la marche humaine. Or nous avons vu que même avec une semelle, on observait des impacts non négligeables. Nous avons donc testé des semelles de formes différentes et composées de plusieurs matériaux pour tenter de diminuer ces chocs. Nous avons présenté deux solutions, la première étant une semelle inspirée de chaussures humaines, et la seconde étant une semelle avec des bords arrondis. La première semelle a permis de réduire de moitié les pics d'accélération, mais ceux-ci restent encore non négligeables. En revanche, nous pensons que la deuxième semelle est prometteuse, même si dans l'immédiat nous n'avons pas obtenu de résultats, car en s'approchant d'une marche humaine, c'est-à-dire avec une marche où les pieds arrivent sur le sol avec un angle, nous pouvons efficacement atténuer les impacts. A l'avenir, nous comptons explorer cette voie et travailler sur l'optimisation de la forme de cette semelle.

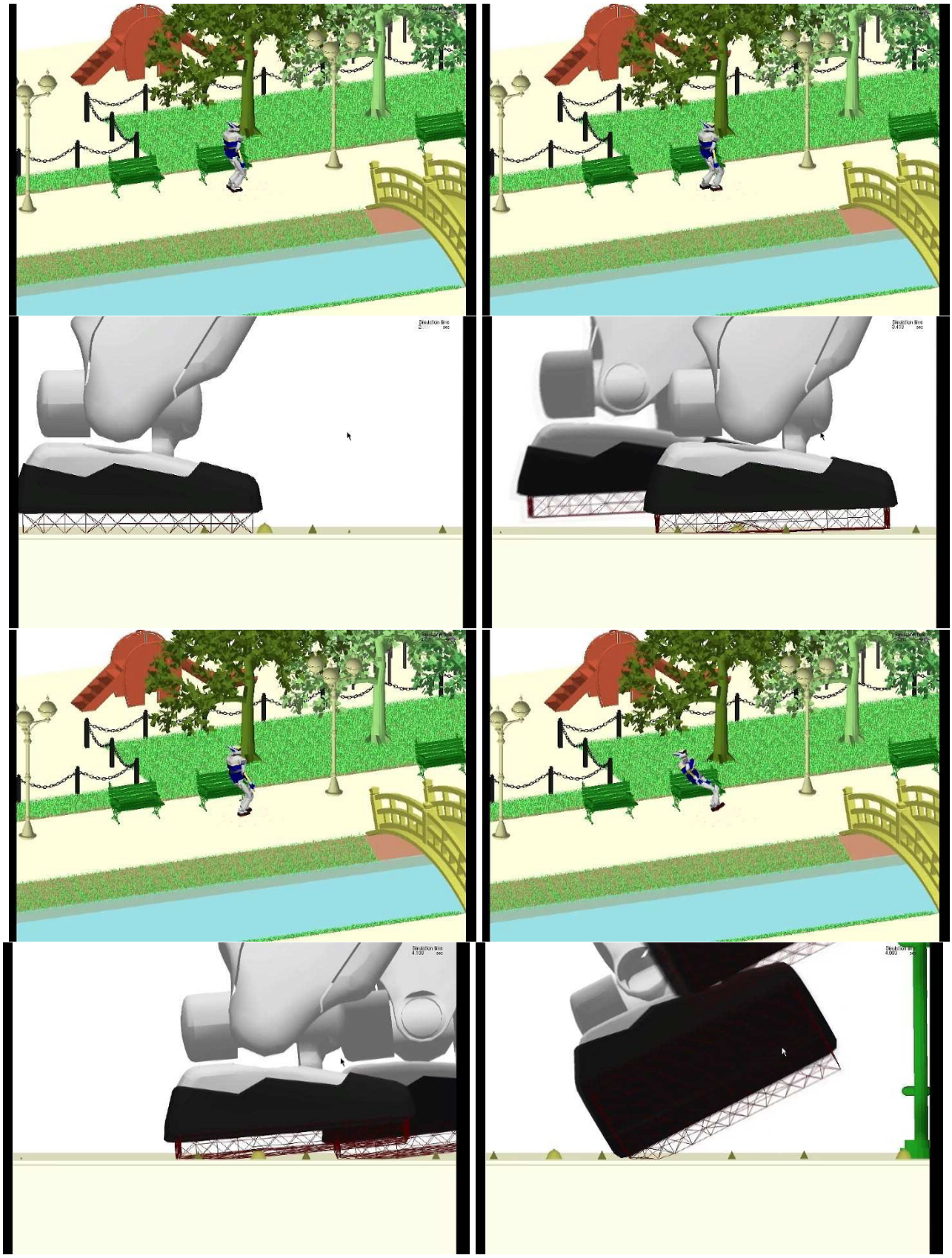


figure 4.11: *Simulation de marche sur un sol granuleux. En haut : vue d'ensemble. En bas : zoom sur les semelles. On observe bien les déformations au contact des grains.*

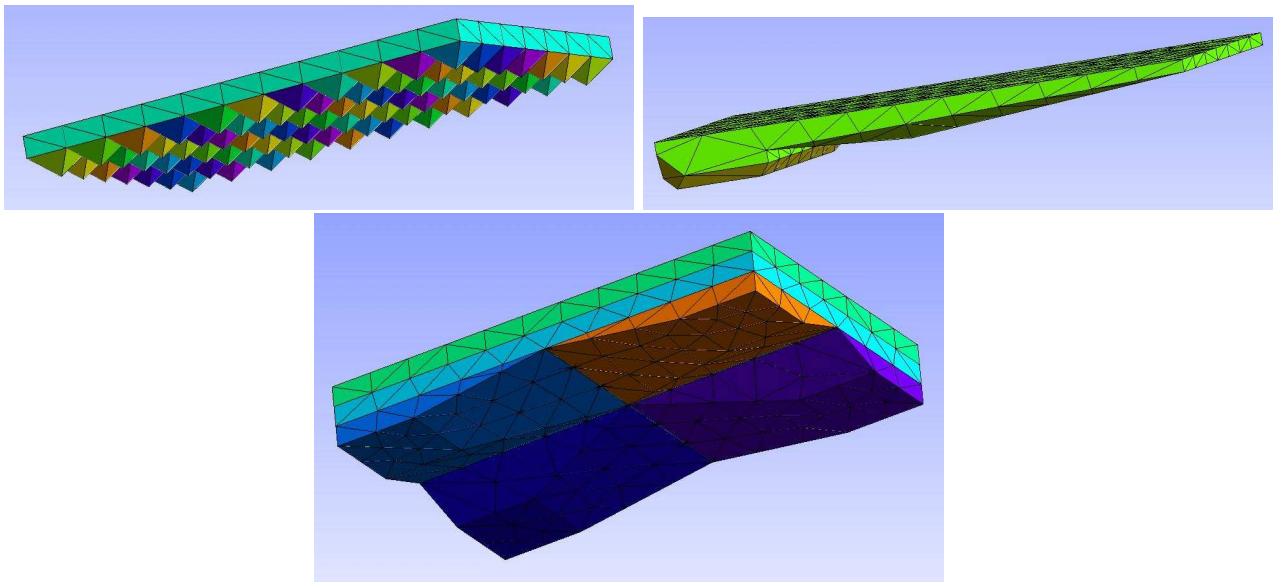


figure 4.12: *Différentes formes de semelles.*

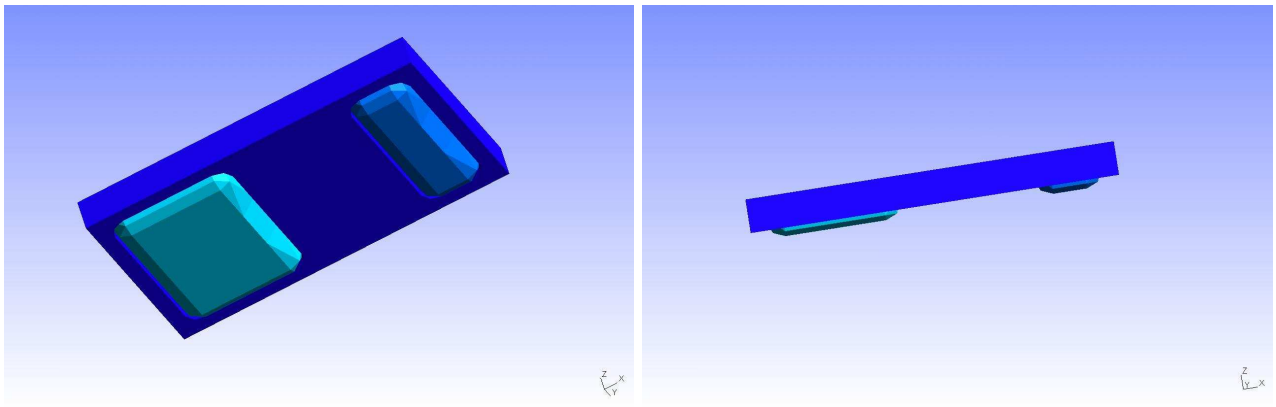


figure 4.13: *Une semelle avec deux couches de matériaux et des coussinets.*

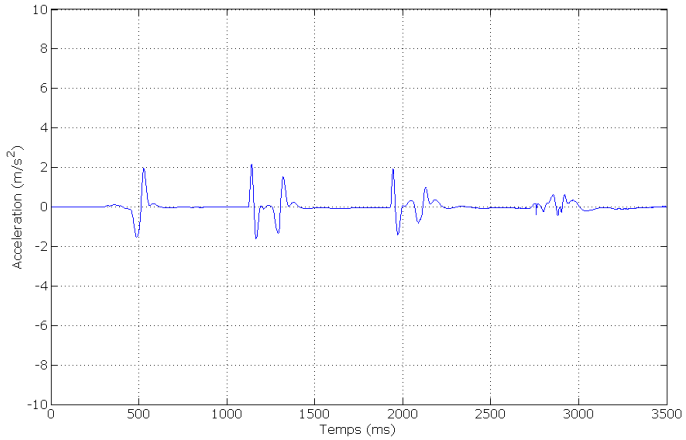


figure 4.14: *Evolution de l'accélération du torse selon l'axe vertical.*

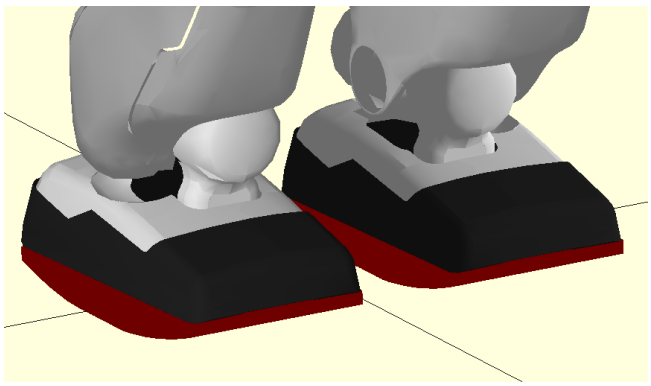


figure 4.15: *Une semelle avec des bords arrondis.*

Chapitre 5

Validation et applications

Dans les chapitres précédents, nous avons présenté les modèles physiques et les modules développés pour réaliser un simulateur dynamique complet. Dans ce chapitre, nous confrontons la simulation à la réalité. Nous proposons d'abord un scénario simple de validation. Ensuite, nous présentons des applications ayant fait usage de notre simulateur, notamment la simulation et l'expérimentation de manipulations et générations de postures. Les applications ont été réalisées avec Hitoshi Arisumi, Sylvain Miossec, Adrien Escande et Sylvain Garsault, sur le robot HRP-2. Enfin, lors d'un séjour à l'INRIA Rhône-Alpes, nous avons tenté d'interfacer AMELIF et SICONOS, avec Vincent Acary, pour étendre les méthodes de résolution du contact frottant.

Sommaire

Avant tout propos, il est important de souligner qu'il n'existe pas à ce jour une méthodologie ou une métrique dont nous ferions usage afin de valider un simulateur robotique, par exemple, des tests sur plateforme ou un comité de validation qui permette de noter qualitativement ou quantitativement la qualité d'une simulation. Certains chercheurs tentent de reproduire les mesures capteurs et avec la plus fine granularité possible, les modèles d'actionnement et les interactions physiques. On serait tenté de dire que la validité d'un simulateur pourrait s'appuyer sur des expérimentations réelles et la confrontation des comportements du robot simulé et réel en se référant aux sorties capteurs. Mais identifier les paramètres réels tels que le coefficient de frottement (entre la carapace du robot et différents objets de l'environnement) ou des réducteurs ou mécanismes de liaisons articulaires, le coefficient de restitution d'impact, les inerties des corps... est quasiment impossible à faire à chaque scénario. Il est d'ailleurs légitime de se poser la question de l'utilité d'une telle démarche. La réponse est évidemment (et heureusement) non. En effet, ce qui importe lorsque l'on simule un robot marchant sur un sol, n'est pas de connaître le coefficient de frottement réel pied/sol, mais de pouvoir modéliser et le simuler correctement de telle sorte que si en situation réelle le sol avait ce frottement, alors on aurait un comportement du robot qui serait à peu près le même. En plus les capteurs sont assez souvent bruités et ce qui est important ce n'est pas tant de simuler ce bruit exactement comme la réalité et de confronter les mesures réelles et simulées, mais plutôt d'avoir un bruit dans la simulation et qui aurait des propriétés globales identiques. En robotique, on développe des lois de commande robustes à ces variations et le but est de s'assurer de la robustesse de ces lois en les confrontant avec divers scénarios de simulation dans le but d'augmenter les chances que les expériences réelles soient réussies dans la majorité des cas avec des paramètres bien souvent différents.

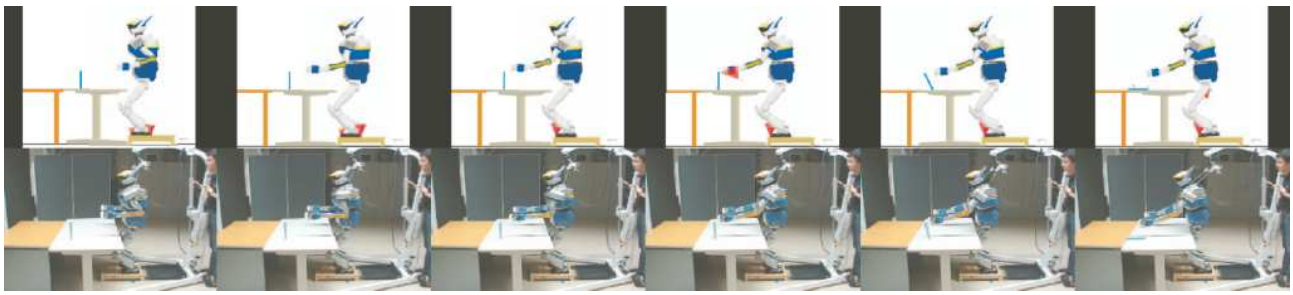
C'est dans cet esprit que nous avons conduit la validation de notre simulateur : nous avons travaillé simplement avec des collègues qui ont fait usage de notre simulateur dans la validation de leurs travaux de recherches et développement en optimisation de tâches extrêmes, ou la planification de points d'appuis, etc.

5.1 Confrontation avec la réalité

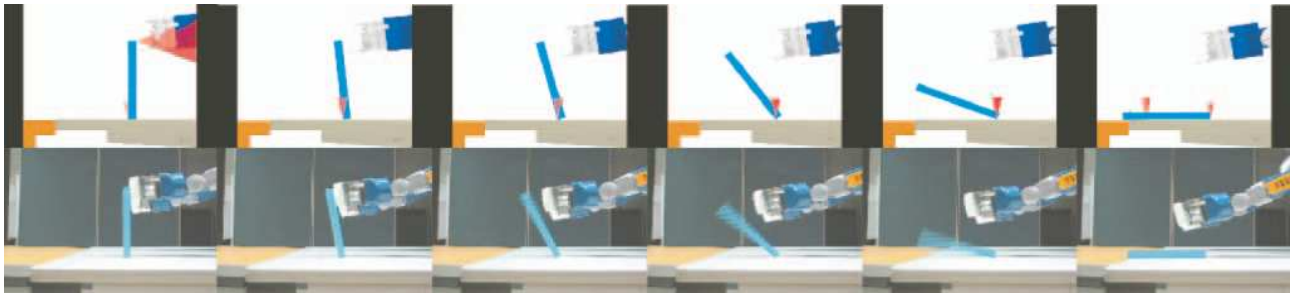
5.1.1 Première comparaison

Nous effectuons ici une première comparaison, purement visuelle, entre notre simulateur et la réalité. Nous demandons au robot HRP-2 de faire tomber un objet posé sur une table. Pour cela il se penche en avant de manière à placer son centre de masse en dehors du polygone de support [Chardonnet *et al.*, 2006]. Le robot est placé sur une estrade.

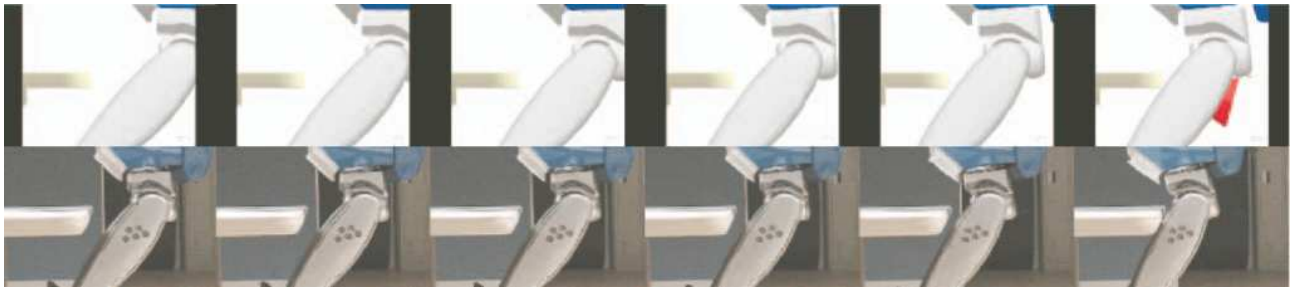
La figure 5.1 nous montre que visuellement la simulation et la réalité sont très proches, en particulier l'objet glisse en tombant aussi bien en simulation qu'expérimentalement. Nous avons mesuré un écart de 2mm dans la distance de glissement de l'objet après être tombé, ce qui est satisfaisant dans le contexte d'usage de cette expérience. Notons toutefois que dans cette expérience, nous nous sommes contentés de mesurer grossièrement les différents coefficients de frottement entre les objets. A cela



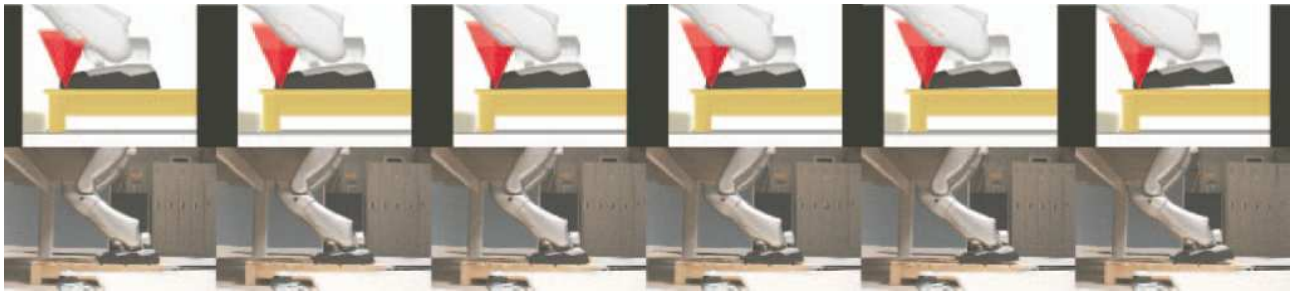
(a) Vue générale.



(b) La pince du robot touche l'objet et le fait tomber sur la table.



(c) Les jambes du robot touchent la table sans rebond apparent.



(d) Le robot se penche en avant, les pieds décollent de l'estrade.

figure 5.1: *Comparaison visuelle entre la simulation et la réalité (en haut : la simulation. En bas : la réalité). Les cônes rouges dans la simulation sont les cônes de frottement, leur hauteur est proportionnelle à la valeur de la force normale.*

s'ajoutent les erreurs potentielles de calibration visuelle d'origines diverses (entre la caméra simulée et la caméra réelle). De fait cette comparaison donne une bonne tendance mais elle ne nous permet pas de valider complètement notre simulateur. Dans la partie qui suit nous allons affiner la comparaison entre notre simulateur et la réalité en allant à contre sens de ce que nous avons dit en préambule de ce chapitre : nous observons les valeurs des efforts calculés en simulation et celles données dans la réalité.

5.1.2 Scénario de validation

Une comparaison visuelle n'est évidemment pas précise, nous avons donc opté pour une comparaison des phénomènes de contact en se basant sur les données des capteurs d'efforts : celles calculées par le simulateur et celles obtenues directement du capteur. Pour cela nous réaliserons un essai très simple en statique impliquant seulement le robot et un objet fixe dans l'environnement. En l'occurrence, nous demandons au robot de plier son bras droit à angle droit et d'appuyer légèrement sur une poutre située juste en face de lui. C'est la partie où le robot appuie sur la poutre qui nous intéresse le plus ici, en particulier les efforts exercés sur la main.

Pour obtenir des résultats les plus proches de la réalité, plusieurs paramètres doivent être pris en compte : le coefficient de frottement entre les objets, le modèle réel des capteurs d'effort et les paramètres réels des actionneurs du robot et enfin les flexibilités au niveau des chevilles.

5.1.2.1 Détermination des coefficients de frottement

Les coefficients de frottement que nous avons à identifier sont entre les pieds et le sol, et entre la main et la poutre. Pour celui entre les pieds et le sol, nous avons pu obtenir le pied seul du robot sur lequel nous avons rajouté un poids ($m = 11,84\text{kg}$ [1]), et grâce à un dynamomètre nous avons mesuré la force nécessaire f pour faire glisser le pied sur le sol (voir figure 5.2). Le coefficient de frottement sec s'obtient simplement par l'équation :

$$f = \mu f_n = \mu mg \Rightarrow \mu = \frac{f}{mg} \tag{5.1}$$

Nous avons réalisé 50 essais afin d'obtenir un résultat plus précis (voir figure 5.3). Nous avons pris alors la moyenne de toutes les valeurs obtenues, à savoir $\mu \approx 0,983$. Il est à noter que nous ne mesurons ici que le coefficient de frottement statique et non dynamique. Dans la situation présente, nous pouvons considérer que ces deux coefficients sont identiques. De même pour le coefficient entre la main et la poutre, nous avons $\mu \approx 0,1$.

5.1.2.2 Modélisation des capteurs d'efforts

La modélisation des capteurs d'efforts est plus complexe. Comme les mesures fournies par le capteur sont données dans le repère du capteur, nous avons besoin de la position et de l'orientation du capteur par rapport au repère du corps dans lequel il se situe. En l'occurrence pour la main, le capteur a une orientation d'environ 90 degrés par rapport à la main et son centre est quasiment confondu avec celui de la main (voir figure 5.4).

Ensuite pour obtenir les véritables valeurs des efforts, les valeurs données par le capteur sont à multiplier par une matrice, carrée pleine, de gains de calibration $\mathbf{G}$ qui lorsque le capteur est parfait est égale à l'identité. Dans notre cas, les capteurs sont un peu anciens, par conséquent cette matrice $\mathbf{G}$

1. En réalité la masse n'a pas d'importance, nous avons pris une masse qui permette d'effectuer des mesures fiables compte tenu de la précision du dynamomètre.

figure 5.2: *Dispositif pour mesurer le coefficient de frottement statique entre le pied et le sol.*

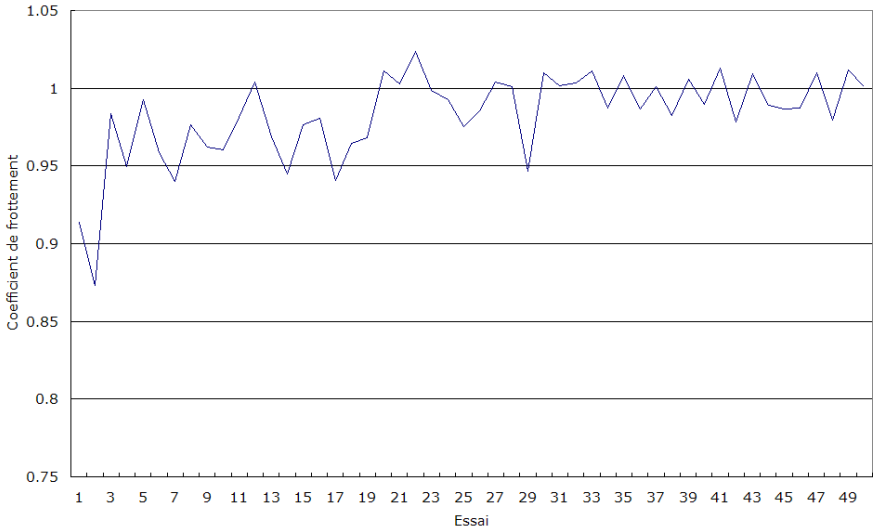


figure 5.3: *Valeurs obtenues lors des mesures du coefficient de frottement statique entre le pied et le sol.*

n'est pas égale à l'identité ; ses paramètres ont été identifiés à partir de mesures hors ligne à diverses postures/orientations de la pince. Nous avons enfin l'équation de la dynamique du capteur qui s'écrit :

$$m\ddot{\mathbf{x}} = \mathbf{P} + \mathbf{f}_e + \mathbf{f}_r \tag{5.2}$$

où m et $\ddot{\mathbf{x}}$ sont respectivement la masse et l'accélération du corps où se trouve le capteur, $\mathbf{f}_e$ et $\mathbf{f}_r$ sont respectivement les forces extérieures s'exerçant sur le capteur et la force exercée par les autres corps du robot sur le capteur, $\mathbf{P}$ est le poids du corps où se trouve le capteur. Cette équation est exprimée dans le repère du capteur. Dans notre cas, nous travaillons principalement en statique car nous voulons

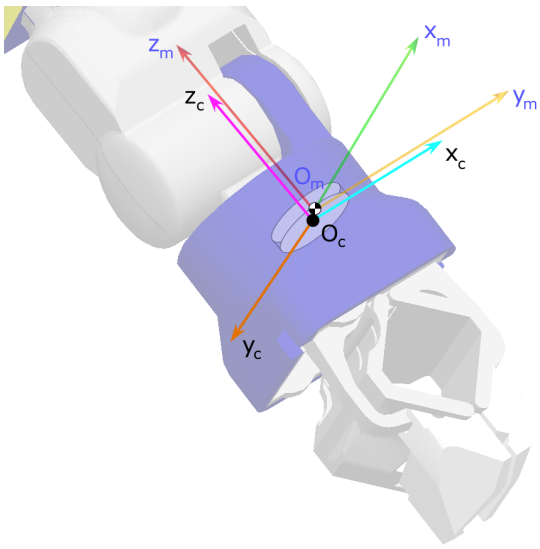


figure 5.4: *Position et orientation du capteur par rapport à la main. L'indice $_c$ représente le capteur, l'indice $_m$ la main.*

mesurer les forces en statique. De fait on a $\ddot{\mathbf{x}} = 0$ et :

$$\mathbf{P} + \mathbf{f}_e + \mathbf{f}_r = 0 \tag{5.3}$$

et

$$\mathbf{f}_e = -\mathbf{P} - \mathbf{f}_r \tag{5.4}$$

La force donnée par le capteur $\mathbf{f}_c$ est en fait égale à $-\mathbf{f}_r$. De fait on a :

$$\mathbf{f}_e = -\mathbf{P} + \mathbf{f}_c = -\mathbf{P} + \mathbf{G}\mathbf{f}_l \tag{5.5}$$

avec $\mathbf{f}_c = \mathbf{G}\mathbf{f}_l$ et $\mathbf{f}_l$ la force directement lue sur le capteur. En réalité, les capteurs sont calibrés lorsque le robot se tient droit (toutes les articulations à zéro), les forces données sur la main sont mises à zéro ! Par conséquent nous rajoutons un décalage $\mathbf{f}_d$ sur le résultat et nous avons alors :

$$\mathbf{f}_e = -\mathbf{P} + \mathbf{G}(\mathbf{f}_l + \mathbf{f}_d) \tag{5.6}$$

5.1.2.3 Loi de commande

Pour actionner les articulations, un couple est envoyé par les moteurs. Généralement dans les simulations, le couple appliqué résulte d'une loi de commande de type proportionnel-dérivée :

$$\mathbf{\Gamma} = K_p(\mathbf{q}_d - \mathbf{q}) + K_v(\dot{\mathbf{q}}_d - \dot{\mathbf{q}}) \tag{5.7}$$

mais nous devons en réalité tenir compte de différents facteurs tels que les frottements secs statiques et visqueux dans les articulations et les coefficients de réduction des moteurs. Avec ces différents

paramètres, le couple alors à envoyer pourrait plutôt résulter d'une loi qui s'écrit :

$$\mathbf{\Gamma} = R\left(K_p(\mathbf{q}_d - \mathbf{q}) + K_v(\dot{\mathbf{q}}_d - \dot{\mathbf{q}})\right) + \mu_v\dot{\mathbf{q}} + \mu_d \text{sign}(\dot{\mathbf{q}}) \tag{5.8}$$

avec R les coefficients de réductions, μ_v et μ_d les coefficients de frottements visqueux et sec statique respectivement. Ces différents paramètres ont été identifiés expérimentalement.

5.1.2.4 Flexibilités

Comme nous l'avons indiqué dans le chapitre précédent, le robot possède des flexibilités au niveau des chevilles. Ces flexibilités entraînent des oscillations sur le robot qui peuvent être réduites en utilisant un stabilisateur. Dans notre cas, nous nous passerons de ce stabilisateur car on ne connaît pas son expression et la façon dont il agit sur les références articulaires ; son usage serait donc clairement source d'incertitude pour la mesure. Par contre la flexibilité des chevilles existe et son influence, elle, n'est certainement pas négligeable. Par conséquent nous devons intégrer ces flexibilités. Nous utilisons pour cela le modèle, simplifié, présenté dans le chapitre précédent, à savoir un ressort-amortisseur virtuel dans chaque direction. Les valeurs des paramètres des ressorts-amortisseurs ont été réglées d'après les valeurs fournies par le groupe HRG [Nakaoka *et al.*, 2007].

5.1.2.5 Simulation et expérience

Avec toutes ces considérations, nous obtenons alors les résultats présentés dans les figures 5.5 et 5.6 pour la simulation et l'expérience respectivement. Les figures 5.7 et 5.8 représentent les données du capteur de la main droite du robot en simulation (en bleu) et expérimentalement (en vert) selon les axes z et y respectivement du capteur. Noter que le robot n'effectuant le mouvement que dans le plan sagittal, on aura toujours une force nulle ou négligeable selon l'axe x au niveau de la main.

Sur les figures 5.7 et 5.8, la période allant de 1s à 3s correspond au moment où le robot plie un peu ses jambes et ses bras, la période allant de 4s à $5,5$s correspond au moment où le robot plie son bras droit pour l'avoir à l'horizontale, le robot avance ensuite son bras de $5,5$s à 7s puis ne bouge plus. Au début (le robot se tient droit) le capteur donne comme valeur selon l'axe z $7,55$N qui est le poids de la main et bien 0 selon l'axe y. Contrairement à ce qui a été dit plus haut nous avons préféré faire en sorte que le capteur donne selon l'axe z le poids de la main lorsque le robot se tient droit, pour que lorsque la main est à l'horizontale, le capteur nous donne 0N ce qui est bien le cas entre $5,5$s et $6,5$s. Le pic observé à $6,5$s correspond au moment où le robot touche la poutre avec sa main. Celle-ci a tendance à ripper légèrement sur la poutre ce qui provoque une chute des efforts de contact avant d'appuyer vraiment sur la poutre et donc obtenir des efforts positifs selon l'axe z. La force selon y évolue peu et reste équivalente au poids de la main. Les oscillations observées à partir de 7s sont dues aux flexibilités des chevilles. Nous pouvons constater que nous obtenons des résultats en simulation très proches de ceux de la réalité. Les légères différences observées entre $6,5$s et 7s sont en particulier dues à des erreurs entre la position, l'orientation réelles du robot avec celles données au simulateur et à des erreurs dans la modélisation des flexibilités aux chevilles.

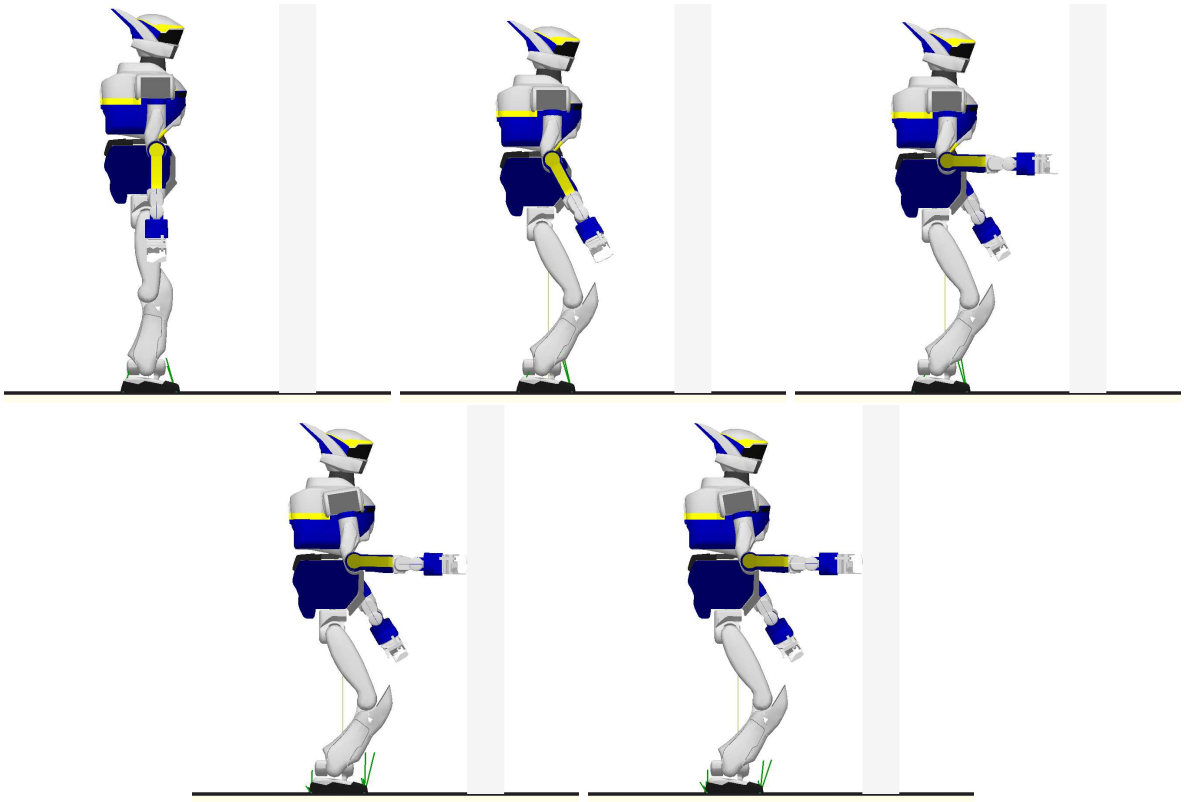


figure 5.5: *Captures d'écran de simulation. De gauche à droite, le robot est d'abord tout droit, il plie ensuite ses jambes et légèrement ses bras, il plie son bras droit pour l'avoir horizontal, il avance son bras, il touche alors la poutre et enfin il reste dans cette position.*

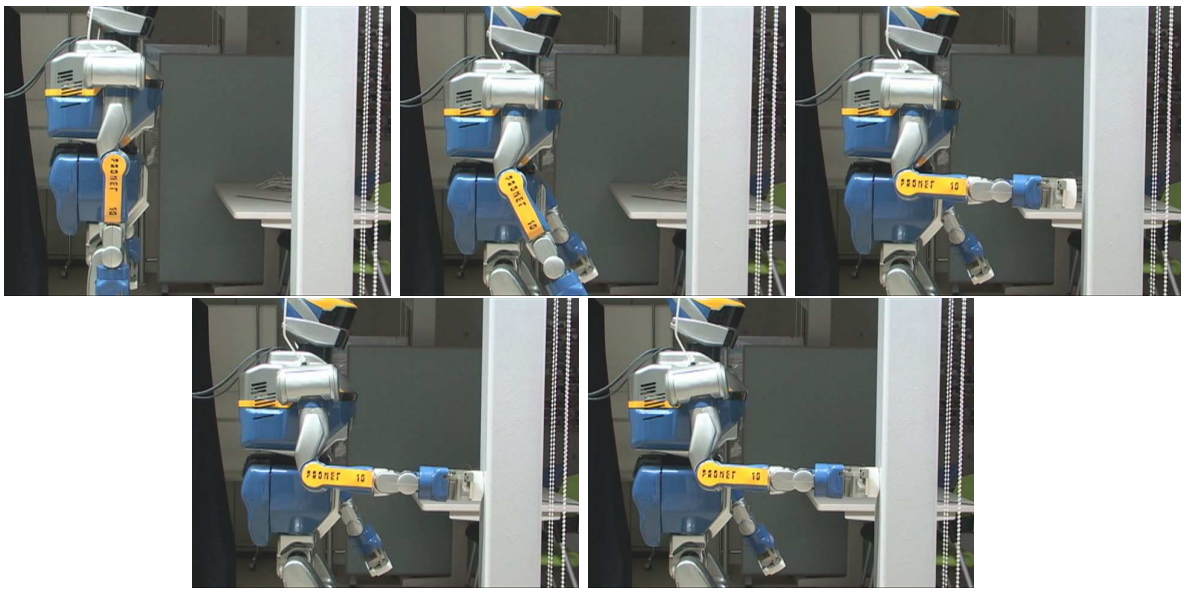


figure 5.6: *Captures d'écran de l'expérience.*

5.2 Applications

Dans cette partie nous montrons trois applications pratiques de notre simulateur au robot HRP-2. Pour les deux premières applications, il s'agit pour le robot d'effectuer des tâches de manipulation

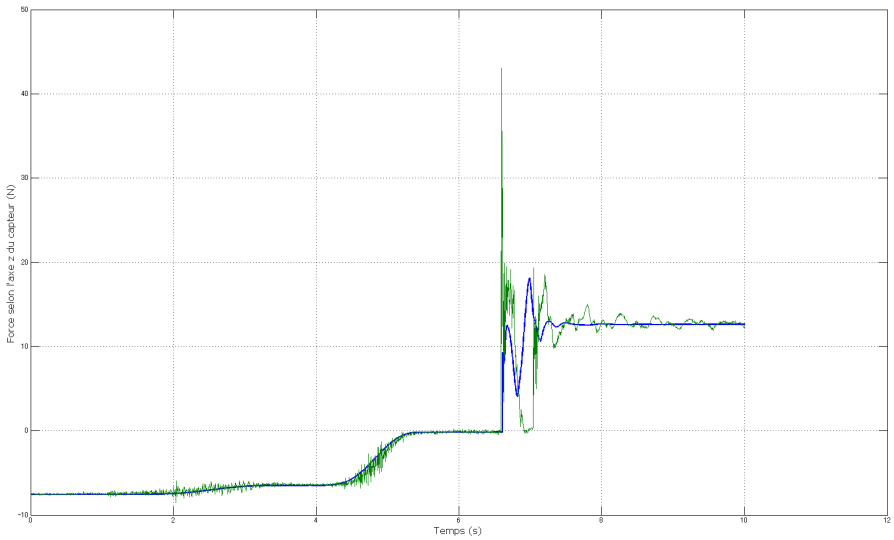


figure 5.7: *Evolution de la force lue sur le capteur selon l'axe z au cours du temps. En bleu, la force calculée en simulation ; en vert, la force donnée par le capteur réel.*

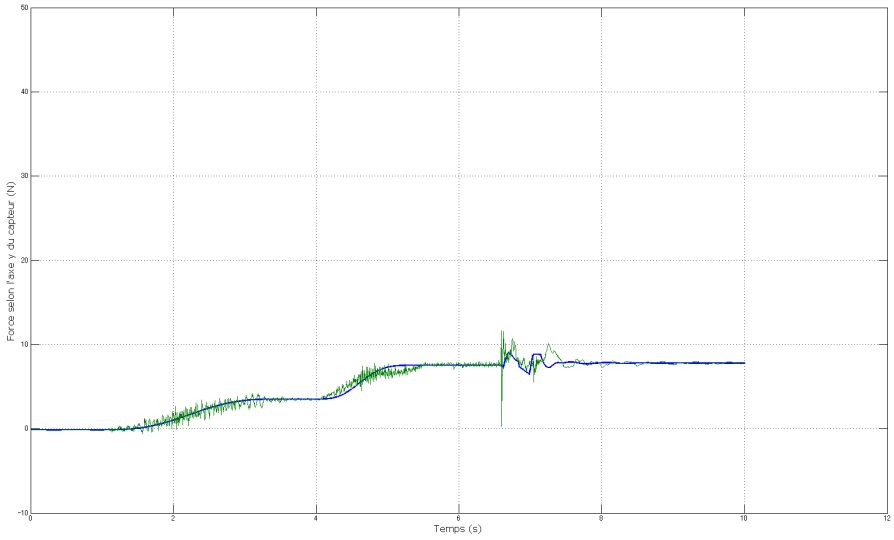


figure 5.8: *Evolution de la force lue sur le capteur selon l'axe y au cours du temps. En bleu, la force calculée en simulation ; en vert, la force donnée par le capteur réel.*

avec des objets. Le premier exemple consiste à soulever de manière dynamique une boîte lourde de $4,5$kg jusqu'à la hauteur des hanches, le deuxième exemple s'inspire de l'haltérophilie et consiste à soulever des objets lourds jusqu'au dessus de la tête du robot. Ces travaux ont été réalisés en collaboration avec Hitoshi Arisumi pour le premier exemple et avec en plus Sylvain Miossec pour le second exemple. En réalité, la seconde tâche que l'on peut qualifier d'extrême, a été réalisée grâce à l'optimisateur de trajectoire développé par Sylvain Miossec [Miossec *et al.*, 2006] et dont la base est le calcul de la dynamique. Les deux applications qui suivent sont tirées de [Arisumi *et al.*, 2007,

Arisumi *et al.*, 2008]. Enfin, la dernière application consiste à générer des postures et des trajectoires dynamiquement stables et à les jouer sur notre simulateur. La génération de postures et de trajectoires a été réalisée dans le cadre du travail de thèse d'Adrien Escande et du stage de master de Sylvain Garsault, nous avons défini les scénarios puis validé leur réalisation sur notre simulateur. La majeure partie des résultats et figures sont tirés de [Garsault, 2008] pour cette partie.

5.2.1 Mouvement de levée dynamique

La plupart des tâches de manipulations extrêmes sont généralement précédées d'un mouvement préliminaire sans lequel il n'est pas possible d'accomplir ces tâches. Par exemple, dans le domaine sportif, pour lancer un javelot il est nécessaire d'étendre son bras vers l'arrière et de bouger tout le corps pour le lancer le plus loin possible (voir figure 5.9 [2]).

figure 5.9: *Exemple de situation où un mouvement préliminaire est réalisé. Pour lancer le javelot, l'athlète doit étendre son bras vers l'arrière et tourner un peu son torse.*

Ici, le mouvement préliminaire est le mouvement du bras et du corps. Ce mouvement préliminaire a pour effet :

- de créer des forces auxiliaires. Ce mouvement augmente le moment cinétique et permet d'appliquer des impulsions sur l'objet. Ceci est particulièrement utile lorsqu'il n'est pas possible d'appliquer une force de manière continue ;
- de préserver la stabilité lors de la manipulation. En l'absence de mouvement préliminaire, nous avons tendance à basculer et à tomber, quelle que soit la force que nous appliquons parce que le centre de masse de l'ensemble personne-objet est en-dehors de soi et que nous n'avons pas les pieds fixés au sol (voir figure 5.10).

Ainsi, un mouvement préliminaire permet d'augmenter nos capacités de manipulation. Les travaux présentés dans la littérature ne prennent pas en compte ce mouvement avec un robot humanoïde car le centre de masse global est toujours considéré à l'intérieur du polygone de support du robot [Harada *et al.*, 2003, Harada *et al.*, 2005], autrement dit l'objet à manipuler est toujours très proche du robot, alors qu'il existe des cas où le centre de masse global ne peut être qu'en dehors du polygone de support du robot (voir figure 5.11). Ici nous nous proposons de faire effectuer au robot HRP-2 une opération de levée dynamique d'objet telle que celle représentée sur la figure 5.12 en lui faisant faire un mouvement préliminaire. Le but de cette application est d'accroître la mobilité et les

2. Tirée de `http://eps.roudneff.com/eps/`.

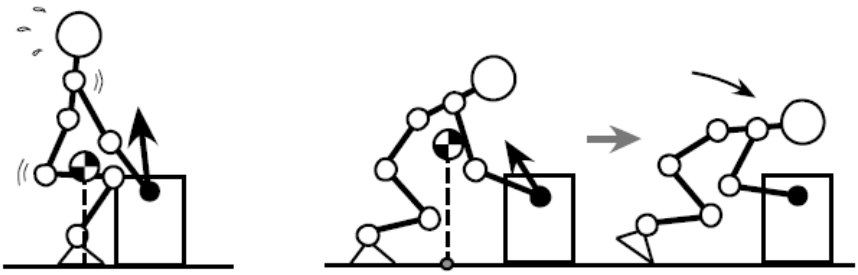


figure 5.10: *Cas où la manipulation d'objets est impossible avec une force appliquée de manière continue. A gauche : la personne ne peut pas soulever l'objet quelle que soit la force qu'elle y met. Au milieu et à droite : la personne bascule en avant car le centre de masse global est en-dehors de la personne.*

capacités du robot HRP-2.

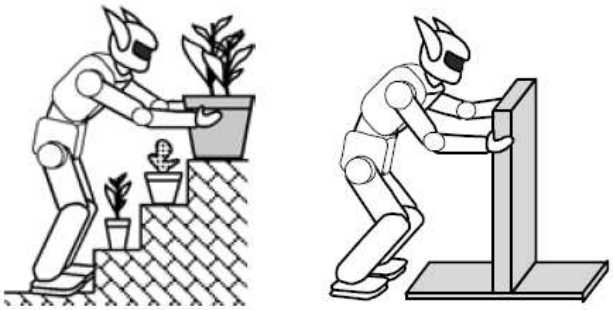


figure 5.11: *Cas où la manipulation d'objets est difficile car l'objet est trop loin du robot. Le centre de masse global est toujours en-dehors du polygone de support du robot.*

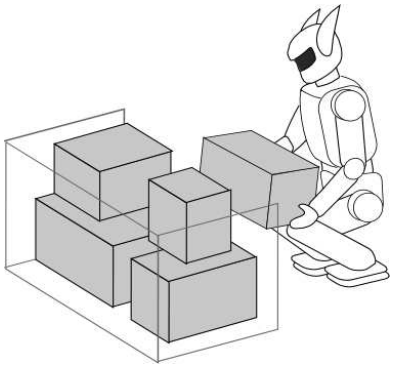


figure 5.12: *Exemple de tâche de manipulation que nous désirons réaliser ici.*

5.2.1.1 Principe

Nous supposons que le mouvement à effectuer est symétrique par rapport au plan médian, de même que le robot et l'objet. De fait, nous pouvons considérer le système robot-objet dans le plan médian, autrement dit un système en deux dimensions. Le robot ici est alors composé de sept articulations à un degré de liberté (voir figure 5.13). Ensuite on définit un espace accessible statique qui est un ensemble de positions de la main du robot pour lesquelles le centre de masse global robot-objet est à l'intérieur

du polygone de support du robot (voir figure 5.14). Si la main du robot est en-dehors de cet espace, le robot perd la stabilité. De même, la taille et la forme de cet espace varient en fonction de la charge imposée à la main. Lorsque la main tient un objet, on peut distinguer deux types d'espace accessible : il s'agit d'un volume lorsque l'objet est soulevé et d'une ligne parallèle au sol lorsque l'objet est posé au sol. L'état initial de l'objet au sol et l'état final de l'objet soulevé sont considérés ici statiques. La transition d'un état à l'autre ne peut se faire en appliquant une force continue sur l'objet car elle passe par un état instable. La question est donc de savoir comment passer du premier état (l'objet posé au sol) au second (l'objet soulevé) sans que le robot perde l'équilibre.

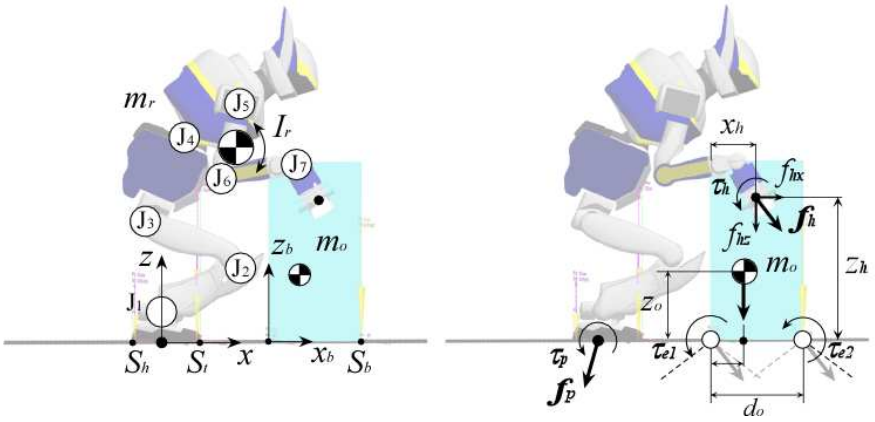


figure 5.13: *Le modèle utilisé. Le robot, l'objet et le mouvement à réaliser étant symétrique, nous pouvons considérer le système en deux dimensions.*

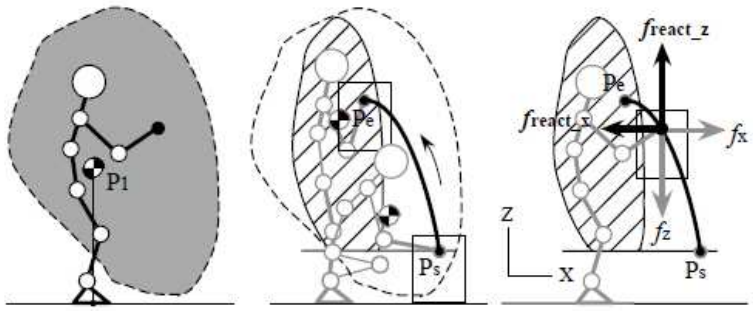


figure 5.14: *A gauche : espace accessible du robot. Au centre : la trajectoire que doit effectuer la main du robot. A droite : les forces s'appliquant sur la main du robot.*

La stratégie adoptée est d'effectuer le mouvement de manière séquentielle. Ce mouvement est décomposé en cinq étapes (voir figure 5.15). On suppose que le centre de masse global du système est initialement à l'intérieur du polygone de sustentation du système robot-objet et non plus à l'intérieur de celui du robot seul. De fait, le ZMP se trouve à l'extérieur du polygone de support du robot, en supposant que dans une configuration statique le centre de masse et le ZMP sont confondus. Pour que le robot puisse soulever l'objet sans tomber vers l'avant, il est donc nécessaire de faire bouger le ZMP à l'intérieur de la zone de stabilité du robot, ce qui demande un couple important au niveau des moteurs. Dans la pratique il est difficile de réaliser un tel mouvement en raison de la puissance

limitée des actionneurs. Une solution, intuitive, est de pousser l'objet vers l'avant et d'utiliser les forces de réaction alors engendrées (représentées par $\mathbf{f}_{\text{react}}$ sur la figure 5.14 droite). Cependant l'objet sera toujours loin de la position finale désirée. Par conséquent, pour résoudre ce problème, nous allons nous servir de ce fameux mouvement préliminaire. Comme indiqué plus haut, lors de ce mouvement des forces auxiliaires seront créées et le moment cinétique du robot sera élargi. Nous ferons en sorte d'engendrer des impulsions de forces (forces de grandes amplitudes dans un très petit intervalle de temps) pour permettre au robot de soulever l'objet, autrement dit du point de vue du robot, ce mouvement préliminaire consistera à créer un impact dans le sens du mouvement de levée.

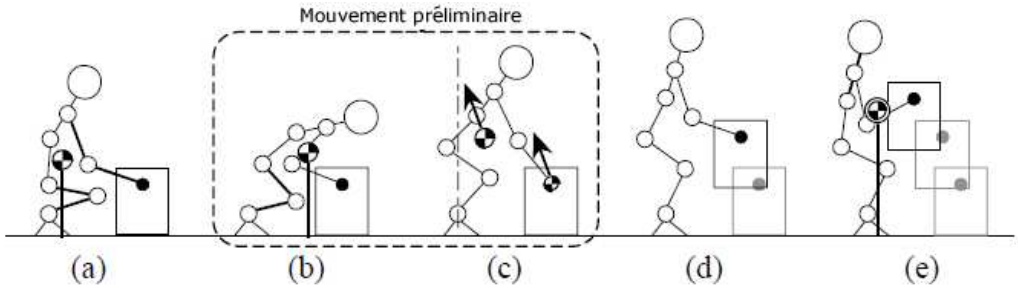


figure 5.15: *Les différentes étapes pour soulever un objet. (a) Le robot est assis. (b) et (c) Mouvement préliminaire, le robot se penche en avant pour accroître son moment cinétique, ce qui crée une force impulsionnelle permettant au robot de lever l'objet. (d) Le robot soulève l'objet jusqu'à sa position finale. (e) Le robot porte l'objet.*

De fait, lorsque le robot va soulever l'objet, celui-ci effectuera un mouvement en translation, entraînant le robot à effectuer aussi un mouvement en translation. D'un point de vue énergétique, il s'agit d'un mouvement très coûteux à cause de la gravité. Par conséquent, plutôt que d'effectuer un mouvement en translation, nous préférerons un mouvement en rotation de type pendule inversé.

Nous expliquons maintenant plus en détail les différentes étapes représentées en figure 5.15. Pour plus de compréhension, nous inversons l'ordre des étapes.

5.2.1.2 Pendule inversé

Une modélisation équivalente du système robot-objet est le pendule inversé de la figure 5.16. Le centre de masse du pendule A et la masse sont ceux du système robot-objet. Ce système tourne autour du centre du pied. Nous supposerons que la configuration de la partie supérieure du robot est fixée après que le robot a soulevé l'objet et que la position du centre de masse du système est contrôlée par les jambes pour pouvoir suivre la trajectoire d'un pendule inversé. Nous imposerons enfin que la position finale du centre de masse à atteindre sur l'axe horizontal est sur la ligne verticale passant par le centre du pied. La position finale de la main sur l'axe horizontal C_e est donnée par l'intersection de l'arc de cercle centré sur le pied et passant par la position initiale de la main, et l'altitude maximale de levée de l'objet z_{lev} (voir figure 5.16).

Pour imposer la position du centre de gravité, il est nécessaire de connaître le moment angulaire L_B autour du centre du pied après l'impact qui permet au robot de garder la stabilité. De fait, la position finale peut être vue comme le point d'équilibre du pendule.

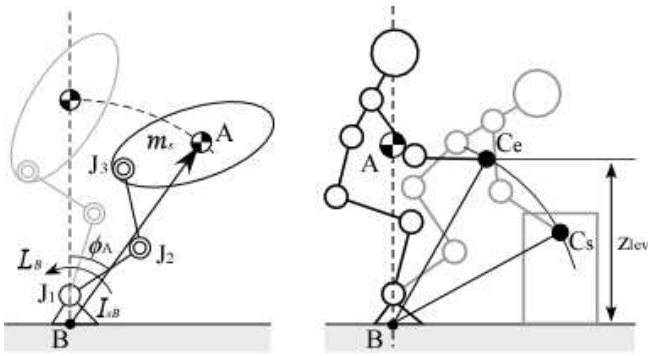


figure 5.16: *Modèle du pendule inversé équivalent au système robot-objet.*

5.2.1.3 Impact

L'impact permet au robot de soulever l'objet, cependant il peut entraîner une perte de stabilité du robot car la force de réaction due à l'impact peut être suffisamment élevée pour que la vitesse et l'accélération de chaque membre du robot changent instantanément. Il faut s'assurer en particulier que le pied ne bouge pas pendant l'impact.

Nous introduisons pour cela le centre de percussion. Le centre de percussion est le point où l'impact produit une force en translation et une en rotation qui s'annulent parfaitement, autrement dit, les vitesses de translation et de rotation créées par l'impact s'annulent parfaitement [3]. Comme nous ne voulons pas que le pied bouge pendant l'impact, nous choisissons de placer le centre de percussion au centre du polygone de sustentation du robot, plus précisément au centre du pied.

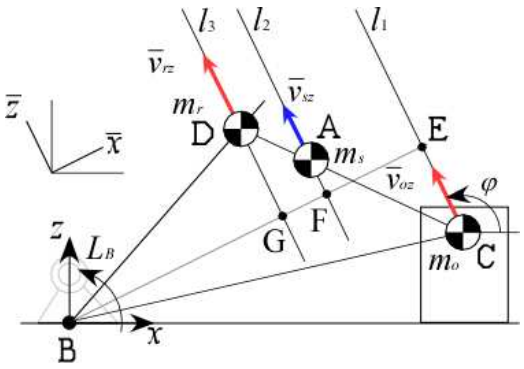


figure 5.17: *Position et vitesse des centres de masse au moment de l'impact. l_1 est la direction du mouvement de levée, l_2 et l_3 sont les directions parallèles à l_1 passant par les autres centres de masse, φ est l'angle entre l_1 et l'horizontale, le repère $O\bar{x}\bar{z}$ est tel que l'axe $\bar{z}$ est parallèle à l_1, le segment BE passant par G et F est perpendiculaire aux lignes l_1, l_2 et l_3, m_s est la masse du système global robot-objet, I_s est l'inertie du système global et ω_s est sa vitesse angulaire.*

En supposant qu'il n'y a pas de couple extérieur qui s'applique sur le système pendant l'impact, avec les notations de la figure 5.17 nous pouvons écrire la loi de conservation du moment angulaire :

$$I_r\bar{\omega}_r + \bar{\mathbf{r}}_{AD} \wedge m_r\bar{\mathbf{v}}_r^- + \bar{\mathbf{r}}_{AC} \wedge m_o\bar{\mathbf{v}}_o^- = I_s\bar{\omega}_s^+ + \bar{\mathbf{r}}_{AD} \wedge m_r\bar{\mathbf{v}}_r^+ + \bar{\mathbf{r}}_{AC} \wedge m_o\bar{\mathbf{v}}_o^+ \tag{5.9}$$

3. `http://en.wikipedia.org/wiki/Center_of_percussion`

où $\bar{\mathbf{v}}^-$ et $\bar{\mathbf{v}}^+$ sont les vitesses avant et après impact dans le repère $O\bar{x}\bar{z}$ respectivement, $\bar{\mathbf{r}}$ est le vecteur entre les deux points indiqués en indice dans le même repère $O\bar{x}\bar{z}$.

Notre but est de réduire les effets de l'impact sur la stabilité du robot. Nous rappelons que l'impact se fait dans la direction du mouvement de levée. Il suffit pour cela de trouver une relation liant la vitesse du robot selon la direction du mouvement $\bar{v}_{rz}$ et sa vitesse angulaire $\bar{\omega}_r$. Nous supposons que :

- l'objet est statique avant l'impact ;
- le robot ne bouge pas dans la direction perpendiculaire à celle du mouvement de levée ;
- après l'impact, le robot et l'objet sont considérés comme un seul corps, de fait toutes les vitesses après l'impact sont égales ;
- les vitesses du robot et celle de l'objet après impact sont égales à celles avant impact dans la direction perpendiculaire à celle du mouvement de levée car l'impact ne se fait que dans la direction du mouvement de levée.

En observant que $m_r\bar{\mathbf{r}}_{ADx} + m_o\bar{r}_{ACx} = 0$ (voir figure 5.17), l'équation 5.9 devient alors :

$$I_r\bar{\omega}_r = I_s\bar{\omega}_s^+ + \bar{r}_{ACx}m_o\bar{v}_{rz}^- \tag{5.10}$$

Par ailleurs, en considérant que le centre de percussion est le point B, la vitesse du système en ce point est constante avant et après impact, elle est même nulle :

$$\bar{v}_{sz_B}^+ = \bar{v}_{sz}^+ - |\bar{\mathbf{r}}_{BF}|\bar{\omega}_s^+ = 0 \tag{5.11}$$

Enfin, le moment angulaire autour du point B après l'impact est donné par :

$$\begin{aligned} L_B &= |\bar{\mathbf{r}}_{BF}|m_s\bar{v}_s^+ + I_s\bar{\omega}_s^+ && (5.12)\\ &= \left(|\bar{\mathbf{r}}_{BF}|^2 m_s + I_s\right)\bar{\omega}_s^+ && (5.13)\end{aligned}$$

En utilisant la dernière équation, l'équation 5.10 s'écrit :

$$I_r\bar{\omega}_r = I_s\frac{L_B}{|\bar{\mathbf{r}}_{BF}|^2 m_s + I_s} + \bar{r}_{ACx}m_o\bar{v}_{rz}^- \tag{5.14}$$

5.2.1.4 Mouvement préliminaire

Maintenant que nous avons déterminé la relation entre les vitesses linéaire et angulaire du robot permettant de limiter les effets de l'impact, nous nous intéressons au mouvement préliminaire. Nous émettrons alors plusieurs hypothèses pour toute la durée de ce mouvement :

- la puissance des actionneurs placés dans les bras est faible. Ainsi nous veillerons à ce que le bras soit presque tendu pour éviter que l'impact ne crée de force dans la direction du bras.
- les positions articulaires du torse et du poignet sont fixes.
- les positions de la cheville et du poignet sont fixes.

Lorsque la configuration du corps de base du robot est donnée, avec les conditions énoncées juste avant, les différentes positions articulaires du robot peuvent être calculées par la cinématique inverse. Il suffit

alors de déterminer la trajectoire du corps principal. Pour cela, nous allons définir la trajectoire du centre de masse du robot. Pour simplifier, nous choisissons une trajectoire parallèle à la direction du mouvement de levée (voir figure 5.18). Il nous reste alors à donner une position initiale au centre de masse du robot en tenant compte de la géométrie du robot. La position finale, qui est la position à laquelle l'impact va être appliqué, est quant à elle donnée en considérant le mouvement de pendule inversé présenté précédemment.

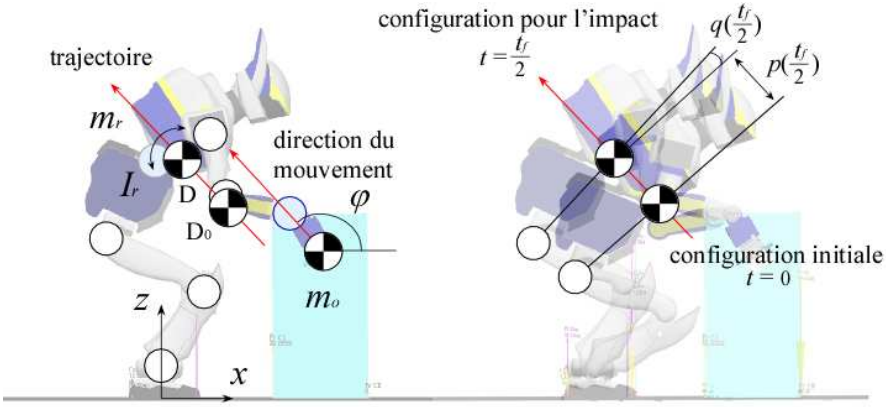


figure 5.18: *Trajectoire du centre de masse du robot (à gauche) et configurations initiale et finale du robot (à droite). La position finale du centre de masse pour le mouvement préliminaire est celle à laquelle l'impact sera appliqué.*

Pour passer de la position initiale à celle finale du centre de masse du robot, nous prenons le profil utilisé pour décrire le mouvement d'une came [4], c'est-à-dire telle que la vitesse est constante pendant la majeure partie du mouvement (voir figure 5.19). Considérant un point dans l'espace, P, V et A sont respectivement les position, vitesse et accélération de ce point. Le changement de 0 à 1 de la position se fait de manière lisse et en grande partie de manière linéaire. Dans notre cas, nous ne considèrerons que la moitié de la courbe ($0 < T < 0,5$) car nous prendrons $T = 0,5$ comme instant où se produit l'impact.

Nous pouvons maintenant définir la longueur du chemin à suivre $p(t)$ et l'angle du corps de base du robot $q(t)$ par :

$$p(t) = \alpha P\left(\frac{t}{t_f}\right),\ q(t) = \beta P\left(\frac{t}{t_f}\right) \tag{5.15}$$

où t_f est le temps final sur la courbe de la figure 5.19 (sur cette figure $t_f = 1$), α et β sont les déplacements maximaux. En dérivant ces deux grandeurs, nous obtenons :

$$v(t) = \frac{\alpha}{t_f} V\left(\frac{t}{t_f}\right),\ \omega(t) = \frac{\beta}{t_f} V\left(\frac{t}{t_f}\right) \tag{5.16}$$

4. `http://fr.wikipedia.org/wiki/Came_(mecanique)`

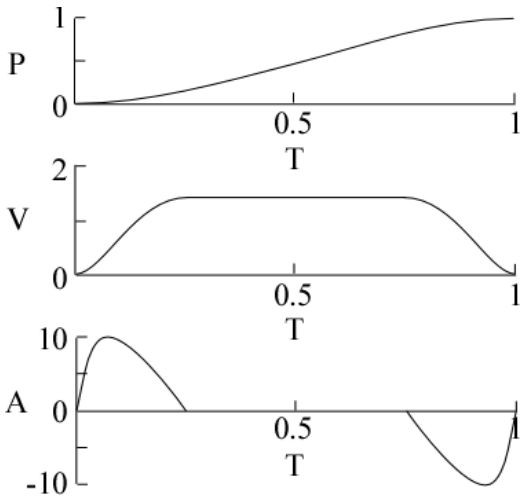


figure 5.19: *Mouvement de came. La courbe de position est ici normalisée. La position P évolue linéairement sur une majeure partie du temps.*

Si $T = 0,5$ est l'instant où l'impact est appliqué, autrement dit $t = \frac{t_f}{2}$, les vitesses du robot sont données par :

$$v\left(\frac{t_f}{2}\right) = \bar{v}_{rz}^{-},\ \omega\left(\frac{t_f}{2}\right) = \bar{\omega}_r^{-} \tag{5.17}$$

Nous pouvons alors en déduire α et β :

$$\alpha = \frac{p\left(\frac{t_f}{2}\right)}{P\left(\frac{t_f}{2}\right)},\ \beta = \omega\left(\frac{t_f}{2}\right)\frac{t_f}{V\left(\frac{t_f}{2}\right)} \tag{5.18}$$

en sachant que $p\left(\frac{t_f}{2}\right)$ est la distance entre les positions initiale et finale du centre de masse et $P\left(\frac{t_f}{2}\right)$ est la position finale du centre de masse. En définissant la position de l'objet, celle du pied, la configuration initiale du robot et la direction du mouvement de levée, nous déterminons complètement la trajectoire du corps de référence du robot.

5.2.1.5 Contraintes

En maintenant l'objet au sol, le robot étend son domaine de stabilité. De fait le robot peut accélérer ou décélérer son mouvement de levée tout en gardant l'objet au sol, ce qui cependant ne garantit pas à l'objet de rester immobile. Afin d'éviter que l'objet ne glisse ou ne bascule, il est nécessaire d'imposer diverses contraintes sur les forces à appliquer au niveau de la main du robot :

$$f_{hx} < \mu_o(m_o g - f_{hz}) \tag{5.19}$$

$$\tau_{e1} = \tau_h - m_o g x_o + f_{hz} x_h - f_{hx} z_h < 0 \tag{5.20}$$

$$\tau_{e2} = \tau_h + m_o g(d_o - x_o) - f_{hz}(d_o - x_h) - f_{hx} z_h > 0 \tag{5.21}$$

$$m_o g - f_{hz} > 0 \tag{5.22}$$

où μ_o est le coefficient de frottement statique entre l'objet et le sol, τ_{e1} et τ_{e2} sont les moments autour de chaque arête de l'objet en contact avec le sol. Ces quatre équations indiquent que l'objet ne doit pas glisser, basculer en arrière, en avant, et quitter le sol respectivement.

De même nous devons imposer des contraintes sur le robot pour qu'il ne tombe pas ou que ses pieds quittent le sol lorsqu'il tient l'objet :

$$f_{px} < -\mu_p f_{pz} \tag{5.23}$$

$$f_{pz} < 0 \tag{5.24}$$

La première équation traduit la loi de frottement de Coulomb en deux dimensions. La seconde équation permet au robot de ne pas décoller du sol.

Enfin le ZMP doit se trouver à l'intérieur du polygone de sustentation formé par les extrêmités des pieds du robot et de l'objet.

5.2.1.6 Simulation

Pour réaliser la simulation, nous avons pris une boîte de $4,5$kg dont le centre de gravité se trouve à la même altitude que le point de saisie. La figure 5.20 montre des captures d'écran de la simulation. La transition entre l'état initial et l'état final par un état instable se fait sans perte de stabilité du robot.

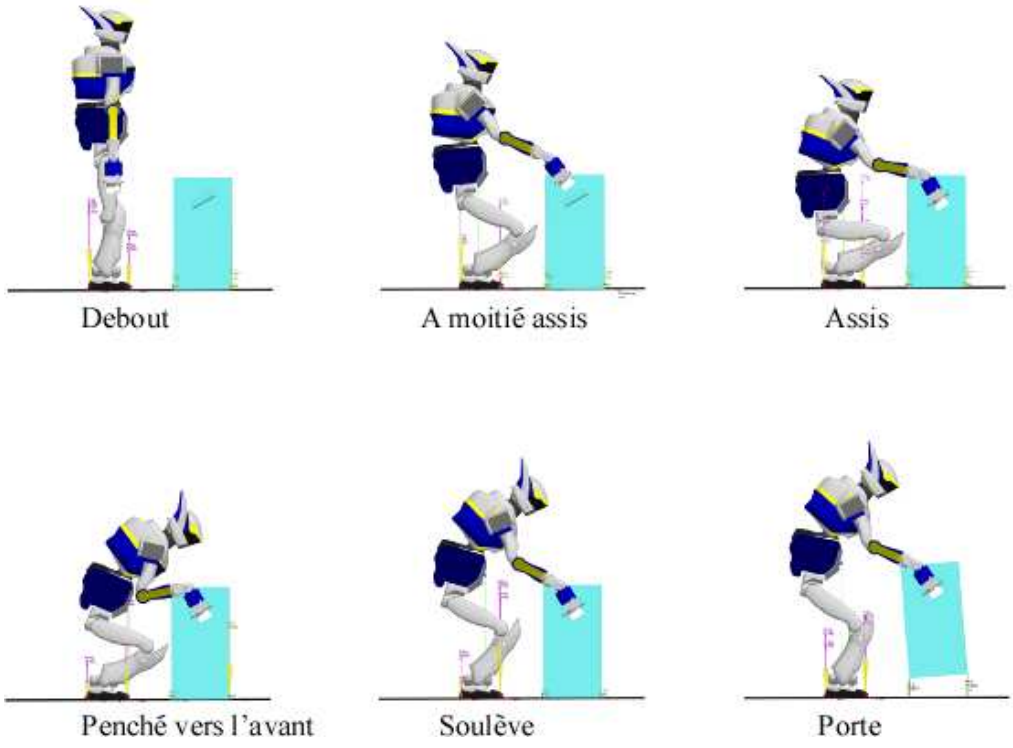


figure 5.20: *Simulation du robot HRP-2 soulevant une boîte.*

Dans un premier temps, nous mesurons la force appliquée au niveau de la main à la boîte $\mathbf{f}_h$ pendant le mouvement préliminaire. Cette force permet d'éviter à la boîte tout glissement ou basculement (voir figure 5.21). La zone rouge représente le domaine dans lequel la force $\mathbf{f}_h$ satisfait les contraintes

d'inégalité énoncées ci-dessus. Ici nous avons pris $z_h = 0,495\text{m}$, $z_o = 0,15\text{m}$, $d_o = 0,3\text{m}$, $\mu_o = 0,4$. Nous voyons bien que la force reste toujours comprise à l'intérieur de cette zone rouge et que donc la boîte ne bouge pas durant le mouvement préliminaire.

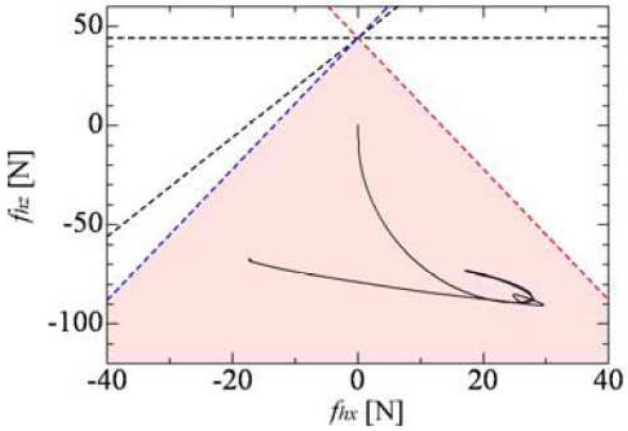


figure 5.21: *Evolution de la force au niveau de la main sur la boîte dans le plan horizontal. La zone rouge représente le domaine pour lequel la force respecte les conditions 5.22. La droite en pointillés noirs horizontale est la dernière contrainte, la seconde droite en pointillés noirs est la première contrainte et les deux autres droites bleue et rouge sont les deux autres contraintes.*

Dans un second temps, nous mesurons l'évolution du ZMP selon l'axe x avec le temps (voir figure 5.22). L'instant où se produit l'impact est représenté par la ligne discontinue verticale, les limites du polygone de sustentation étant représentées par les lignes horizontales en pointillés. Nous rappelons que l'instant d'impact est l'instant à partir duquel le robot commence à porter l'objet. Nous remarquons que le ZMP reste bien compris à l'intérieur du polygone de sustentation, y compris après l'impact. Par ailleurs, la variation du ZMP est faible au moment de l'impact, ce qui montre que le robot peut soulever l'objet de manière stable par notre méthode.

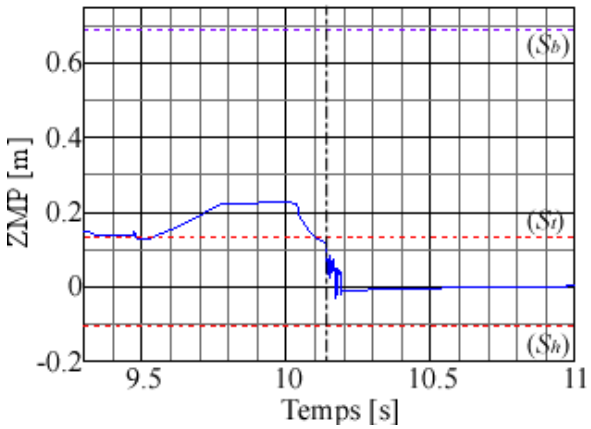


figure 5.22: *Evolution de la position du ZMP selon l'axe x en simulation. La ligne verticale en pointillés est le temps à partir duquel le robot soulève l'objet. Les lignes horizontales en pointillés sont les limites de la zone de sustentation, en particulier les lignes rouges représentent les limites pour le pied seul et les lignes bleues pour l'ensemble pied-boîte.*

5.2.1.7 Expérimentation

Nous avons réalisé des expériences sur le robot réel HRP-2 afin de valider notre méthode. Les captures d'écran de l'expérience sont reproduites sur la figure 5.23. Nous voyons que le robot peut soulever l'objet de manière dynamique sans perdre son équilibre comme prévu dans la simulation. On a observé également la présence d'oscillations qui sont dues aux flexibilités au niveau des chevilles du robot. Nous n'avons pas utilisé le stabilisateur car pour le mouvement présent ce stabilisateur aurait un effet négatif car l'amplitude des efforts exercés sur le robot est telle qu'au final le robot finirait par tomber si ses articulations étaient plus rigides.

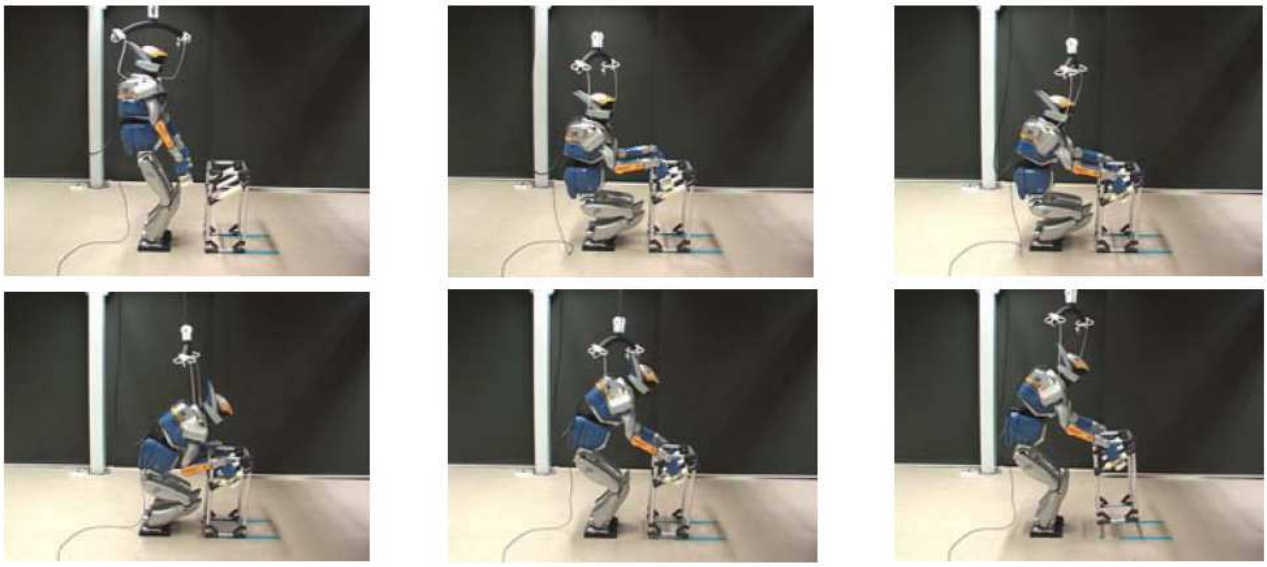


figure 5.23: *Expérience avec le robot HRP-2.*

5.2.2 Haltérophilie

Une extension du mouvement de levée dynamique présenté précédemment est de porter des objets lourds jusqu'à une hauteur supérieure ou égale à la taille du sujet, par exemple pour mettre des objets dans des étagères qui sont plus hautes que soi (voir figure 5.24). Encore une fois, il s'agit d'accroître la mobilité et les capacités des robots humanoïdes. Dans l'application précédente, nous n'avons pas considéré les caractéristiques des moteurs si bien que les actionneurs peuvent arriver à saturation, en particulier créant au cours du mouvement une zone d'instabilité dans laquelle le robot ne peut pas porter un objet lourd de manière statique. Ici nous nous proposons de prendre en compte ces caractéristiques et de calculer un mouvement optimal des bras permettant à un robot de porter un objet lourd sans que les actionneurs saturent : c'est une tâche extrême et complexe, aucune loi de commande ne permet de réaliser cette tâche en spécifiant uniquement les conditions initiales et finales. Il faut donc générer une trajectoire, et aucun planificateur ne pourra venir à bout de ce problème, c'est le cas typique où une optimisation de trajectoire est la seule solution qui nous semble possible dans ce cas d'exemple. Nous ferons encore usage de la dynamique de tout le corps pour permettre la réalisation

de cette tâche.

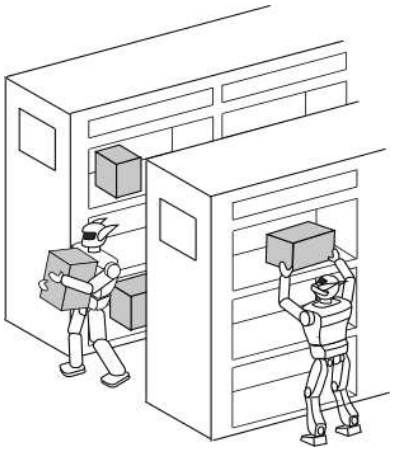


figure 5.24: *Exemple de situation où le robot pose un objet sur une étagère plus haute que lui.*

Plusieurs études ont été réalisées sur des athlètes afin d'analyser le mouvement effectué et les paramètres physiques entrant en jeu lors de la manipulation d'haltères. Ici on ne cherche évidemment pas à entraîner le robot à des compétitions d'haltérophilie mais à s'inspirer des athlètes sur la façon de soulever des objets qui sont plus lourds que ce qu'une personne normale peut porter. Rarement ces travaux ont été appliqués sur des robots car leur structure mécanique et leur performance sont très différentes et surtout généralement plus faibles que celles de l'être humain et c'est précisément parce que le robot n'a pas une grande capacité pour porter des objets lourds que nous nous inspirons de ce domaine là. On peut noter cependant les travaux menés sur le robot Puma dans lesquels un mouvement optimal est calculé et la position des articulations, la vitesse et les limites de couples articulaires sont considérées mais où le modèle des actionneurs n'est pas suffisamment précis, en particulier la prise en compte de frottement dans les articulations est absente [Wang *et al.*, 2001]. Par ailleurs, la stabilité est à prendre en compte dans notre cas.

Ici aussi nous proposons de réaliser le mouvement de manière séquentielle. Nous expliquerons d'abord la stratégie adoptée pour réaliser notre mouvement, la création du mouvement optimal et nous montrerons les résultats associés.

5.2.2.1 Principe

Le mouvement à effectuer est encore une fois symétrique par rapport au plan médian du robot, de même que le robot et l'objet à porter. Par conséquent nous travaillerons en deux dimensions. Le robot est donc modélisé par sept articulations (voir figure 5.25). Ensuite on définit les configurations du bras pour lesquelles il n'est pas nécessaire d'exercer un couple élevé et pour lesquelles les actionneurs ne seront pas saturés. Si l'objet se trouve loin du robot, même dans le cas d'objets légers, les actionneurs peuvent arriver rapidement à saturation, autrement dit plus l'objet sera proche du robot, plus il sera facile de soulever cet objet. Par conséquent, l'espace accessible statique de la main du robot défini dans l'application précédente est limité par la taille du système robot-objet, par la projection de centre de masse global du système qui doit être comprise dans le polygone de support du système, par les limites

articulaires et par les limites de couples articulaires. Il est à noter que comme le robot possède des capteurs d'efforts dans la main, nous devons également tenir compte de la charge maximale applicable sur ces capteurs. Pour les limites de couples articulaires, nous devons considérer que les actionneurs utilisés sur le robot sont à courant continu et que par conséquent il n'est pas possible d'envoyer une commande dépassant la sortie nominale maximale sur une durée supérieure à un temps t_m défini par le constructeur, au risque d'endommager les moteurs. Les limites des actionneurs sont généralement données par le couple maximal $\mathbf{\Gamma}$ et par la vitesse angulaire maximale $\dot{\mathbf{q}}$. Ces deux grandeurs sont liées par :

$$\left| \frac{\dot{q}_i}{\dot{q}_i^M} + \frac{\Gamma_i}{\Gamma_i^M} \right| < 1 \tag{5.25}$$

avec

$$|\Gamma_i| < \Gamma_i^M \tag{5.26}$$

et $\dot{q}_i^M$ et Γ_i^M la vitesse angulaire maximale et le couple maximal respectivement pour l'articulation i. Il est à noter que Γ_i^M n'est pas la sortie nominale maximale qui est plus petite que Γ_i^M. Enfin dans le cas statique où des forces extérieures $\mathbf{f}$ s'exercent sur le robot, nous pouvons écrire :

$$\mathbf{\Gamma} = \mathbf{g} - \mathbf{J}^T\mathbf{f} \tag{5.27}$$

où $\mathbf{g}$ est le couple dû à la gravité. Ce couple est ici limité par la sortie nominale car il est appliqué sur une longue durée. Comme le couple maximal applicable à la main du robot est assez faible, pour éviter une saturation des actionneurs les bras doivent être le plus longtemps possible dans une configuration singulière, ce qui est également observé chez l'être humain : il est plus facile de tenir un objet lourd les bras étendus dans une position verticale que pliés. On considère alors deux configurations singulières verticales, une configuration singulière horizontale n'ayant pas de sens pour les raisons évoquées juste avant. Sur la figure 5.26, les configurations (A) et (C) ne sont pas parfaitement singulières, pour des raisons évidentes de collision entre l'objet et le robot. Néanmoins, des configurations proches de celles singulières doivent permettre également de porter l'objet de manière statique. La partie la plus difficile ici est de définir une trajectoire permettant de passer de la configuration (A) à celle (C), en passant par une zone instable (B).

Nous pouvons énoncer alors deux méthodes pour réaliser ce mouvement. La première méthode est illustrée sur la figure 5.27. Le robot part de la première configuration singulière des bras en étant un peu accroupi puis se relève un peu vers l'arrière. Comme les couples maximaux applicables au niveau des jambes sont élevés, ce mouvement est faisable. L'inertie de l'objet créera une accélération vers le haut permettant d'appliquer des couples plus faibles aux actionneurs du bras et donc de soulever l'objet plus facilement pour atteindre la seconde configuration singulière. L'inconvénient d'utiliser cette méthode est que si l'objet à soulever est très lourd, les actionneurs d'autres membres du robot peuvent saturer, empêchant alors d'atteindre la seconde configuration singulière. La seconde méthode est illustrée sur la figure 5.28. Ici le robot doit au contraire s'accroupir et passer en dessous de l'objet pour atteindre la seconde configuration singulière. L'inconvénient de cette méthode est qu'il est alors nécessaire de

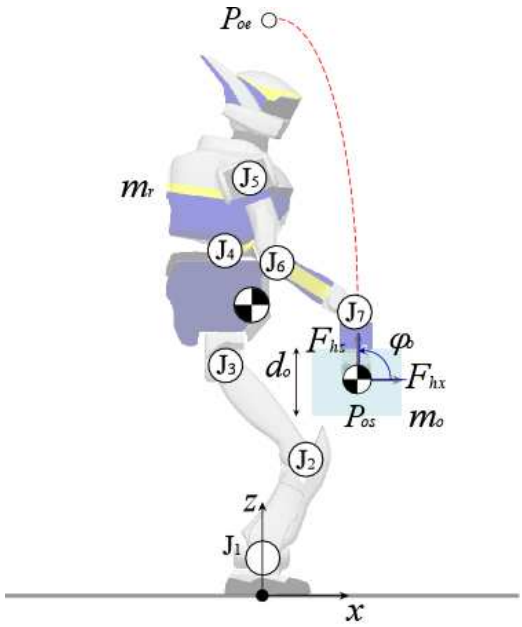


figure 5.25: *Le modèle utilisé. Le robot, l'objet et le mouvement à réaliser étant symétriques, nous pouvons considérer le système en deux dimensions.* P_{os} *et* P_{oe} *sont respectivement les points de départ et de fin.*

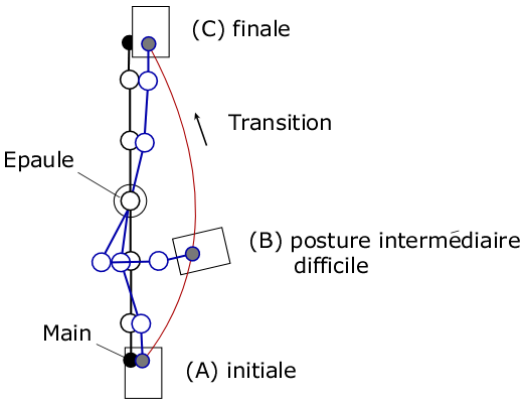


figure 5.26: *Configurations du bras pour lesquelles le robot peut porter un objet lourd de manière statique sans saturer les actionneurs.*

planifier et contrôler un mouvement très rapide pour éviter que l'objet ne tombe à cause du changement de configurations.

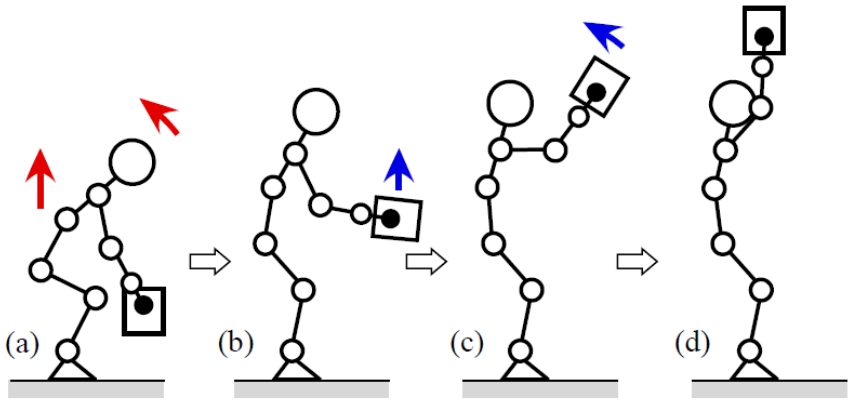


figure 5.27: *Première méthode pour soulever l'objet.*

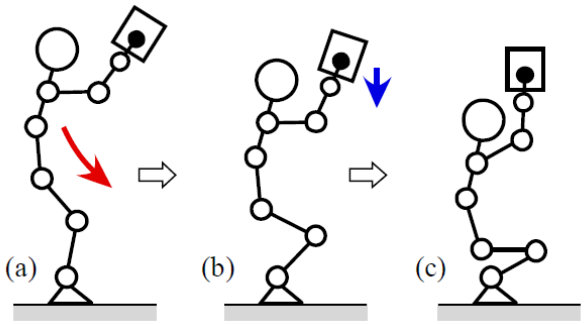


figure 5.28: *Seconde méthode pour soulever l'objet.*

Ces deux méthodes ont toutes les deux un inconvénient majeur qui empêche le robot d'atteindre la configuration singulière finale. Pourtant, en combinant ces deux méthodes, ces deux inconvénients peuvent disparaître. En effet, si le robot part de la configuration initiale de la première méthode, il n'est pas nécessaire de planifier un mouvement très rapide pour passer sous l'objet avec la seconde méthode. D'autre part, il n'est pas nécessaire de soulever l'objet de beaucoup pour arriver à la seconde configuration singulière. En résumé, en utilisant d'une part l'inertie de l'objet, d'autre part en ne modifiant pas sensiblement la hauteur de l'objet une fois celui-ci soulevé, on assure au robot de mener à bien la tâche sans saturer les actionneurs et en réduisant la charge exercée sur le bras. La combinaison de ces deux méthode est illustrée sur la figure 5.29. Nous allons alors concevoir une trajectoire qui suit cette stratégie.

5.2.2.2 Trajectoire

Pour concevoir la trajectoire désirée nous nous servons d'une méthode de contrôle optimal basée sur les travaux de [Miossec *et al.*, 2006]. Dans cette méthode, les trajectoires des positions angulaires sont vues comme des B-Splines et des contraintes discrétisées. Le problème est résolu en utilisant un programme d'optimisation spécifique.

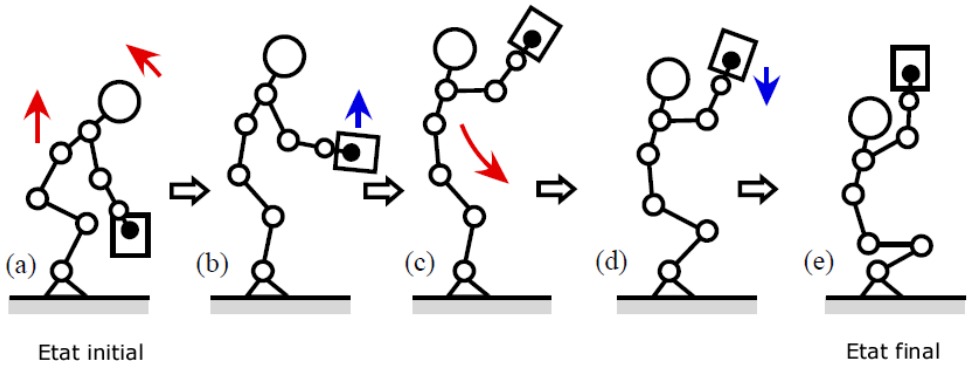


figure 5.29: *La stratégie finale pour soulever l'objet.*

Pour calculer les couples à appliquer, le modèle dynamique inverse, indiqué dans le premier chapitre, est utilisé. Comme on désire prendre en compte les caractéristiques des actionneurs et des articulations précisément, nous intégrons les frottements visqueux et sec dans les articulations. Ces frottements sont ceux utilisés dans la partie 5.1.2. Enfin pour modéliser l'objet à soulever, nous rajoutons artificiellement à la main la masse et l'inertie de l'objet.

Pour passer d'une configuration singulière initiale à une autre finale, il existe une infinité de configurations possibles. Les positions articulaires sont alors paramétrisées à l'aide de B-Splines comme indiqué plus haut et s'écrivent :

$$q_i(p,t) = \sum_{j=0}^{n_b} p_{ij} B_j(t) \tag{5.28}$$

où p sont les paramètres des B-Splines, B est la fonction de base des B-Splines, n_b est le nombre de ces fonctions.

Les contraintes à appliquer sont d'une part que les articulations ne participant pas au mouvement sont contraintes à zéro, d'autre part que les limites de l'espace accessible doivent être respectées. A cela, s'ajoutent la position du ZMP du système qui doit être comprise dans la zone de support, le temps de transition d'une configuration singulière à une autre qui doit être inférieur au temps maximal autorisé t_m pour envoyer des couples sans endommager les actionneurs, et enfin les pieds qui ne doivent ni glisser, ni décoller du sol.

Enfin, le mouvement optimal est obtenu en minimisant l'énergie consommée par les actionneurs pour en accroître la performance. Cette minimisation de l'énergie permet de minimiser les pertes par effet Joule qui sont généralement un facteur limitant dans la puissance des moteurs et qui se traduisent par des limites de couples plus contraignantes. Cette énergie vaut :

$$E_r = \int_0^{t_f} \sum_j c_j \Gamma_j^2 + \dot{q}_j \Gamma_j dt \tag{5.29}$$

où c_j est un paramètre dépendant de l'actionneur, Γ_j est toujours le couple articulaire et t_f est le temps final, c'est-à-dire celui où le bras atteint la configuration singulière finale. Avec toutes les contraintes

formulées plus haut, le problème à résoudre est alors :

$$\min_{p,t_f} E_r(p,t_f),\ \text{avec}\ g_e(p,t_f)=0,\ g_i(p,t_f)\leq 0 \tag{5.30}$$

Ce problème est résolu par le programme d'optimisation de [Wachter et Biegler, 2006] qui est une méthode du point intérieur. Pour accélérer la convergence et le calcul, nous avons calculé le gradient du critère et des contraintes par rapport à p et t_f.

5.2.2.3 Simulation

Nous calculons d'abord l'espace accessible statique des mains pour des objets de diverses masses. Pour calculer cet espace nous avons juste besoin des considérations géométriques du robot et des contraintes de couple articulaire et de vitesse angulaire énoncées plus haut. Nous considérons ici les orientations initiale et finale de l'objet ($\varphi_o = 90\text{deg}$ et $\varphi_o = -90\text{deg}$ respectivement) qui permettent aux actionneurs de la main de ne pas saturer. Les limites de couple à la main sont très faibles en raison du capteur d'effort qui ne supporte que 5N.m autour de l'axe y de la main. La limite de forces des capteurs utilisés est de 200N dans la direction de la main et de 100N dans la direction perpendiculaire. Les données constructeur des moteurs nous indiquent que les sorties nominales maximales sont inférieures à 20% des couples maximaux applicables. Nous obtenons alors les espaces atteignables de la figure 5.30. D'après cette figure, nous pouvons remarquer que plus l'objet est lourd, plus la configuration du bras doit être proche d'une configuration singulière. Par ailleurs, pour $\varphi_o = 0\text{deg}$, autrement dit lorsque l'objet est tenu horizontalement, l'espace accessible n'existe pas en raison des limites articulaires et de couples, ce qui confirme les propos énoncés en partie 5.2.2.1. Comme l'objet doit cependant passer par l'état $\varphi_o = 0\text{deg}$, la transition d'un espace accessible initial à celui final doit se faire dynamiquement par la stratégie présentée en partie 5.2.2.1 et en utilisant la trajectoire conçue dans la partie précédente.

A l'aide de la figure 5.30, et en considérant une répartition uniforme des couples ainsi que la singularité des bras, nous choisissons les postures initiale et finale du robot comme suit :

- Etat initial : pour l'objet $(x_o, y_o, \varphi_o) = (0,26\text{m}; 0,65\text{m}; 90\text{deg})$ et pour les articulations 3 et 5 $(x_{r3}, z_{r3}) = (-0,1\text{m}; 0,63\text{m})$ et $(x_{r5}, z_{r5}) = (-0,039\text{m}; 1,156\text{m})$;
- Etat final : pour l'objet $(x_o, y_o, \varphi_o) = (0,2\text{m}; 1,3\text{m}; -90\text{deg})$ et pour les articulations 3 et 5 $(x_{r3}, z_{r3}) = (-0,12\text{m}; 0,3\text{m})$ et $(x_{r5}, z_{r5}) = (-0,043\text{m}; 0,824\text{m})$.

Avec ces valeurs nous obtenons les positions articulaires de tout le robot. Dans ce qui suit, nous prenons un objet de masse $m_o = 8,4\text{kg}$ et qui a pour taille $0,15\text{m}\times 0,3\text{m}$ en largeur et hauteur respectivement.

La trajectoire calculée avec la méthode présentée dans la partie précédente nous permet d'obtenir toutes les positions articulaires au cours du temps. Pour accélérer les temps de calcul, nous avons de plus fourni au programme une posture intermédiaire. On obtient alors un temps de passage entre les configurations initiale et finale de $1,68\text{s}$, ce qui est très satisfaisant car le temps maximal t_m autorisé est de 3s. La trajectoire est ensuite jouée dynamiquement dans notre simulateur. Les captures d'écran de la simulation sont représentées sur la figure 5.31.

Le robot soulève l'objet sans tomber comme prévu. A la fin du mouvement, la configuration du

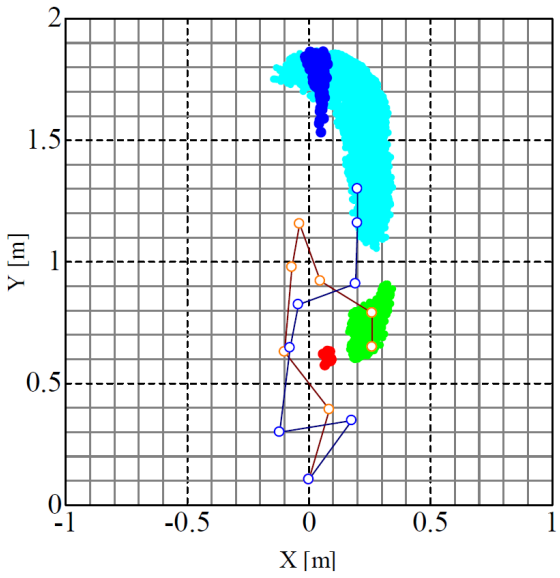


figure 5.30: *Espace accessible de la main en fonction d'objets de diverses masses. Les zones verte et bleu clair correspondent à* $(m_o, d_o) = (8, 4kg; 0, 3m)$ *et à* $\varphi_o = 90deg$*, les zones rouge et bleu foncé à* $(m_o, d_o) = (23, 4kg; 0, 1m)$ *et à* $\varphi_o = -90deg$*.*

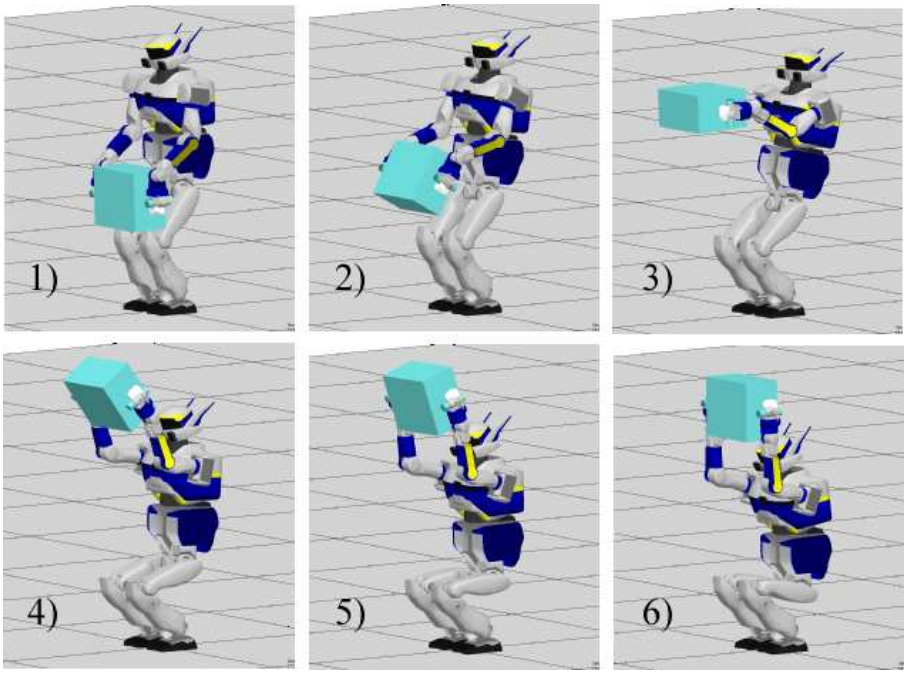


figure 5.31: *Captures d'écran de la simulation.*

bras est proche d'une configuration singulière et est stable.

Pour montrer l'intérêt d'utiliser un mouvement dynamique, nous avons également réalisé une simulation d'un mouvement statique pour lequel toutes les articulations sont contrôlées en suivant la même trajectoire de référence du mouvement dynamique. Cette simulation s'est trouvée être un échec. D'une part, le mouvement est beaucoup plus lent (environ 5 fois plus long, autrement dit le mouvement dure environ 10s) provoquant ainsi un dépassement du temps maximal t_m autorisé, d'autre part, et

en conséquence du premier point, les couples appliqués dépassent largement les valeurs maximales qui sont ici les sorties nominales maximales, autrement dit les actionneurs sont saturés.

Comme nous l'avons mentionné plus haut, cette tâche a en réalité été réalisée grâce à l'optimisateur de [Miossec *et al.*, 2006]. Notre simulateur a permis de vérifier en simulation que la trajectoire calculée par l'optimisateur était réalisable sur le robot réel pour permettre une expérimentation : il s'agit typiquement d'un exemple où l'on peut vérifier que les modèles physiques implémentés dans notre simulateur sont corrects.

5.2.2.4 Expérimentation

Le robot HRP-2 est ici encore utilisé pour nos essais. Nous avons mesuré la relation couples appliqués aux articulations et vitesses angulaires, les résultats sont reportés sur la figure 5.33. Sur cette figure, sont également reportés les couples calculés en simulation et en utilisant un mouvement statique. Les parallélogrammes noirs représentent les limites maximales de couple et de vitesse angulaire pour des mouvements de durée inférieure à t_m. Les parallélogrammes verts sont les sorties nominales maximales. Nous voyons que les courbes en simulation et expérimentales sont toujours très inférieures aux limites, alors que dans le cas d'un mouvement statique, les courbes dépassent les limites.

Nous avons également mesuré les forces exercées sur la main dans les deux directions x et z ainsi que le moment. Les résultats sont reportés sur la figure 5.34. Les traits verticaux en pointillés représentent le début et la fin du mouvement de levée de l'objet. Les forces augmentent juste après le début du mouvement car l'objet est tiré dans la direction inverse des axes x et z. A partir de $t = 3,6$s, les efforts augmentent brutalement, en particulier dans la direction de l'axe z, ce qui signifie que la charge appliquée à la main diminue brutalement à cause de l'accélération créée par l'inertie de l'objet. Nous pouvons observer que les efforts sont similaires en simulation et expérimentalement. Enfin, les valeurs des efforts ne dépassent pas les limites maximales du capteur d'effort.

Enfin, le ZMP selon l'axe x est mesuré. Les résultats sont reportés sur la figure 5.35. Les lignes horizontales en pointillés sont les limites du polygone de support, les lignes verticales en pointillés sont ici aussi le début et la fin du mouvement de levée. Les valeurs sont comprises dans les limites, signifiant que le robot ne tombe pas. On peut noter un pic à la fin du mouvement qui est sûrement dû à un impact qui se produit à ce moment là. Ici non plus nous n'utilisons pas le stabilisateur car la trajectoire a été conçue pour obtenir une grande stabilité du robot.

Nous avons également réalisé des essais avec un objet de masse $m_o = 23,4$kg. Les résultats montrent que les trajectoires sont différentes mais le comportement global du robot est similaire au cas du lever de masse de $m_o = 8,4$kg.

5.2.3 Génération de postures

Notre simulateur a été utilisé dans la génération de postures, plus spécifiquement pour valider les différentes postures générées par un planificateur de points d'appuis [Escande *et al.*, 2006]. Ici nous montrons deux exemples de succession de postures générées pour le robot HRP-2. Ces exemples, à cause

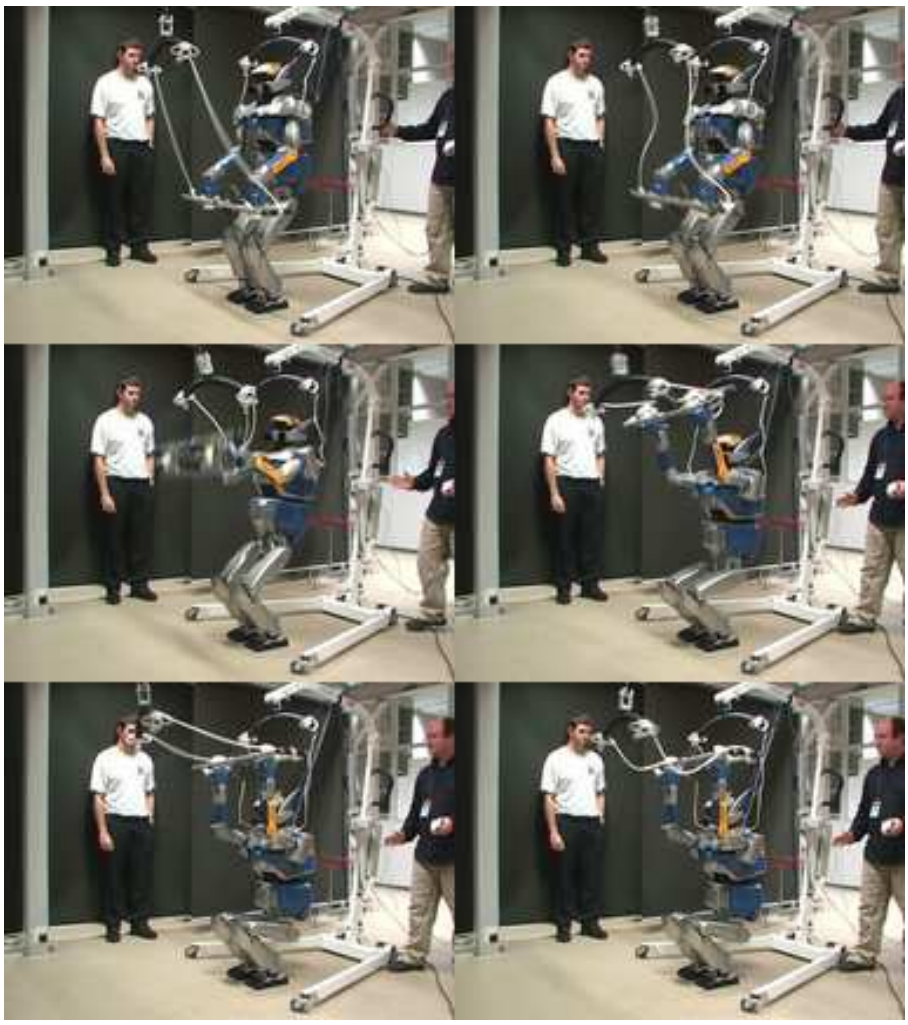


figure 5.32: *Captures d'écran de l'expérience.*

de leurs complexités, n'ont pas été testés sur le robot réel. Les trajectoires données au robot HRP-2 sont créées en utilisant le générateur de posture développé par Adrien Escande [Escande *et al.*, 2006] pour des configurations statiquement stables et le logiciel de Sylvain Garsault [Garsault, 2008] pour générer une trajectoire du centre de masse du robot dynamiquement stable. Toutes les postures du robot sont générées à des pas discrets de 5ms (qui correspond à la boucle de commande du robot HRP-2). Dans la génération de trajectoire dynamique, le frottement est pris en compte et les trajectoires sont conçues pour que les forces de contact restent toujours à l'intérieur du cône de frottement, si possible le plus loin possible du bord du cône. Dans [Garsault, 2008], les cônes sont discrétisés et dans les scénarios suivants, nous avons considéré des cônes à huit facettes. Pour réaliser une trajectoire d'un point initial à un point final, des splines sont utilisées. Celles-ci sont calculées par le générateur de trajectoire mais il est possible de les modifier manuellement, le générateur de posture calcule ensuite les configurations du robot statiquement stables, c'est-à-dire avec le centre de masse qui se situe dans la zone de support. La modification des splines se fait grâce au logiciel 3DSMax (voir figure 5.36).

Le premier scénario que nous proposons est de faire faire au robot quelques pas sur un sol plat. Ici nous n'utiliserons pas le générateur de marche conçu spécifiquement pour le robot HRP-

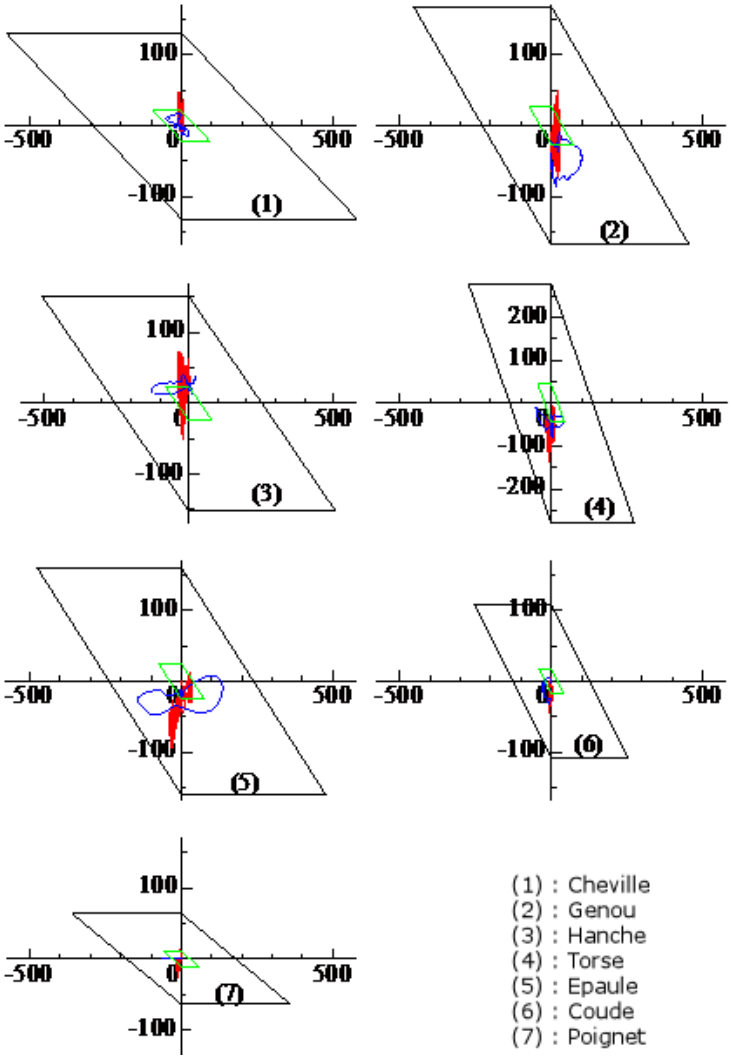


figure 5.33: *Relations couple-vitesse angulaire. Les courbes bleues sont données par la simulation, les courbes vertes sont données par l'expérience et les courbes rouges sont données dans le cas d'un mouvement statique.*

2 [Stasse *et al.*, 2008], nous préférerons utiliser une trajectoire générée par [Garsault, 2008] qui prend en compte un critère de stabilité plus général que le ZMP classique. En effet, nous souhaitions générer des mouvements acycliques en multi-contacts non coplanaires. Mais comme première étape de validation, il faut que l'approche générale marche aussi bien que l'approche classique lorsque le robot marche sur un sol plan. Dans ce cas, les seuls contacts que le robot peut avoir se font entre les pieds et le sol. Le nombre de pas que nous demandons de faire est de 4 (voir figure 5.37) et le mouvement s'effectue sur une durée de $10,8$s. Par ailleurs ce mouvement est composé de 13 sous-trajectoires optimisées toutes les 600ms (voir figure 5.38). Le temps de calcul pour générer ce mouvement est d'environ 90s. L'évolution du centre de masse et du ZMP dans l'espace est représentée sur la figure 5.39. Une fois la trajectoire obtenue, celle-ci est jouée sur notre simulateur. On a alors les résultats de la figure 5.40.

On peut noter que le centre de gravité donné en simulation est proche de celui que l'on désire.

Le second scénario est plus complexe car il fait intervenir plusieurs contacts avec plusieurs objets

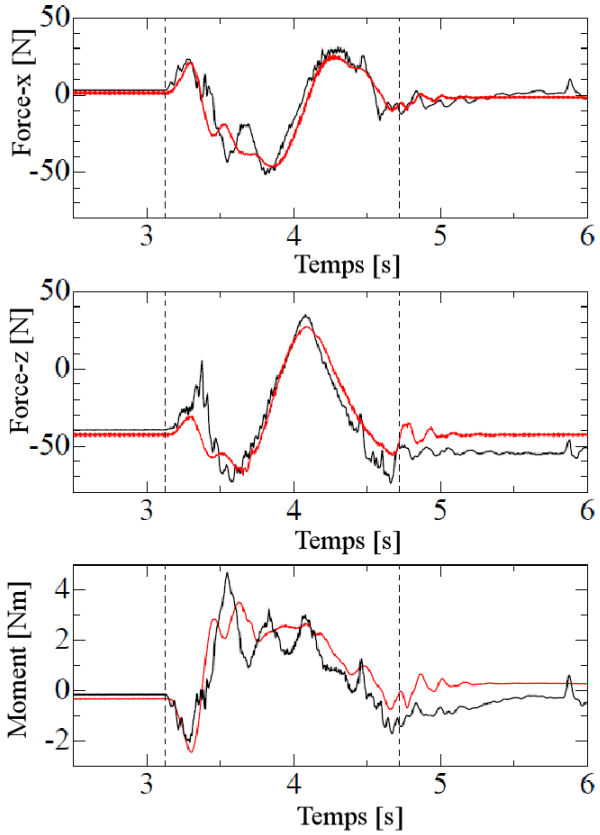


figure 5.34: *Forces selon les axes x et z et moment exercés sur la main lors du mouvement de levée. Les courbes rouges représentent les valeurs calculées par la simulation, les courbes noires représentent les valeurs lues expérimentalement.*

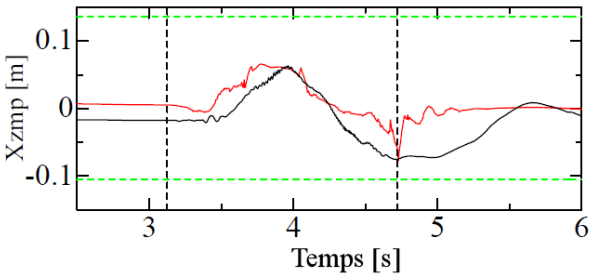


figure 5.35: *Evolution du ZMP selon l'axe x.*

dans l'environnement, en particulier nous plaçons une estrade inclinée et une poutre à proximité du robot. Celui-ci a au départ un pied posé à terre et l'autre sur l'estrade. Nous demandons alors au robot de déplacer le pied qui est posé sur l'estrade à un endroit situé après l'estrade, puis de déplacer l'autre pied pour le positionner à coté du premier pied. Le mouvement dure 6s et est composé de 3 sous-trajectoires mises à jour toutes les 600ms. Le temps de calcul pris est d'environ une minute. Nous avons testé différentes trajectoires du centre de masse (voir figure 5.41). La simulation donne le résultat suivant (voir figure 5.42). Malheureusement la trajectoire finale n'est pas très probante et le robot exécute le mouvement tant bien que mal. Cela est dû à une mauvaise convergence du générateur de postures. Même en changeant les splines, nous obtenons toujours un mouvement très peu naturel.

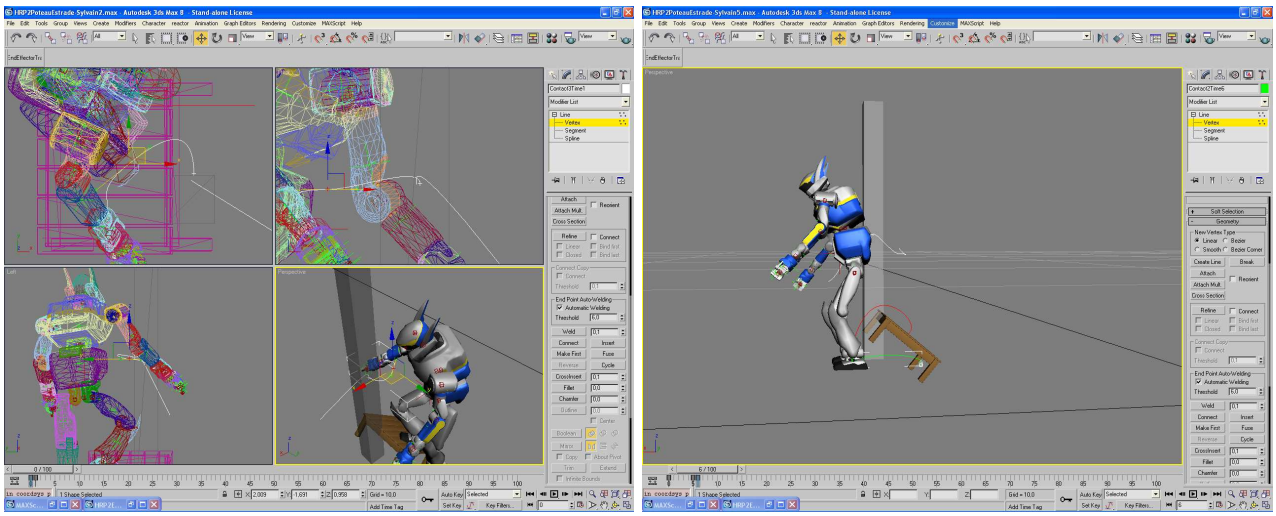


figure 5.36: *Modification de splines avec le logiciel 3DSMax. Les lignes blanche derrière le bassin, rouge au-dessus de l'estrade et verte derrière le pied sont les splines précalculées par le générateur de trajectoire et qui sont à modifier si besoin est.*

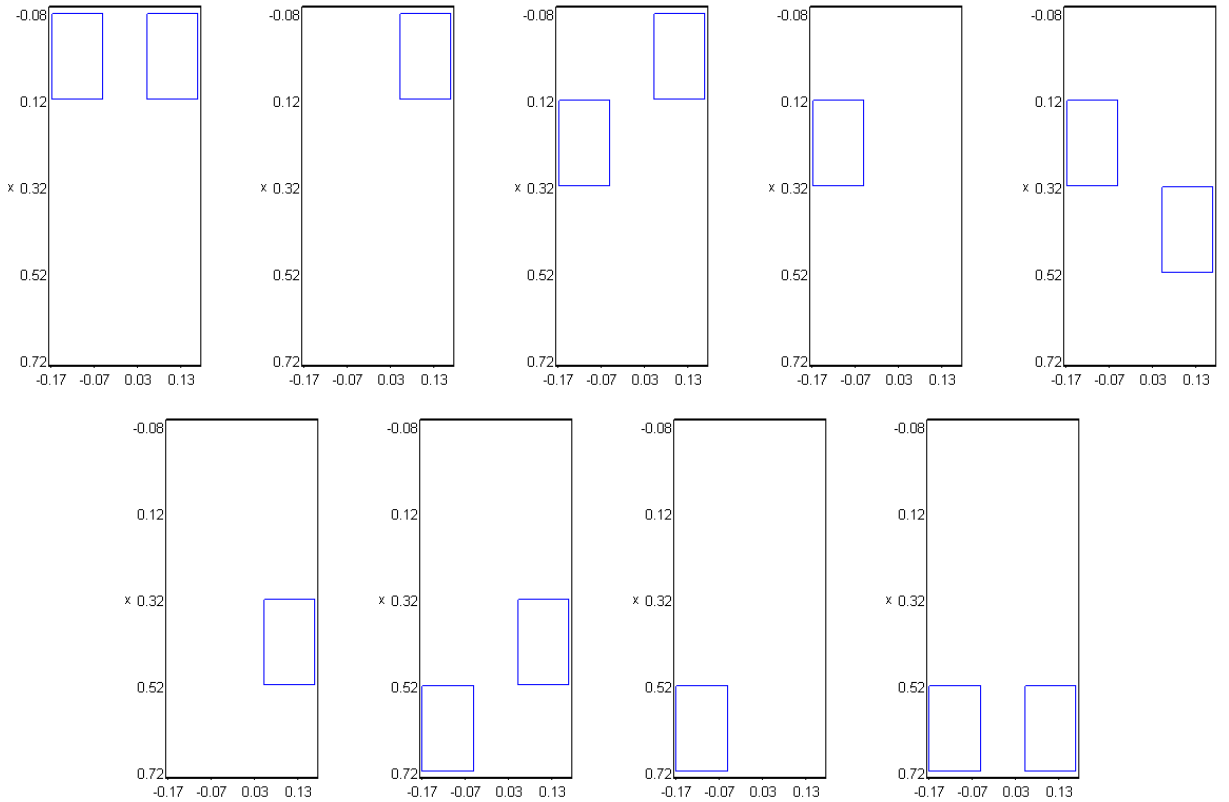


figure 5.37: *Séquence de pas pour la marche.*

5.3 Interfaçage avec SICONOS

Au cours d'un séjour à l'INRIA Rhône-Alpes, nous avons tenté d'interfacer notre simulateur AME-LIF avec la librairie dédiée à la simulation de systèmes non réguliers nommée SICONOS et rapidement présentée dans le premier chapitre. SICONOS [Acary et Perignon, 2007] est issue d'un projet européen

figure 5.38: *Sous-trajectoires optimisées.*

et possède principalement deux parties : un noyau qui fournit tous les algorithmes haut niveau de calcul de la dynamique de systèmes non-réguliers appelé Kernel, et une partie contenant tous les algorithmes bas niveau de résolution de problèmes divers et nommée Numerics.

La partie Kernel contient la modélisation et la simulation. La partie modélisation s'occupe de formuler la dynamique des systèmes qui peuvent être du premier ordre linéaires ou non, ou bien des systèmes lagrangiens linéaires ou non. Les différentes lois physiques d'interaction sont ensuite ajoutées.

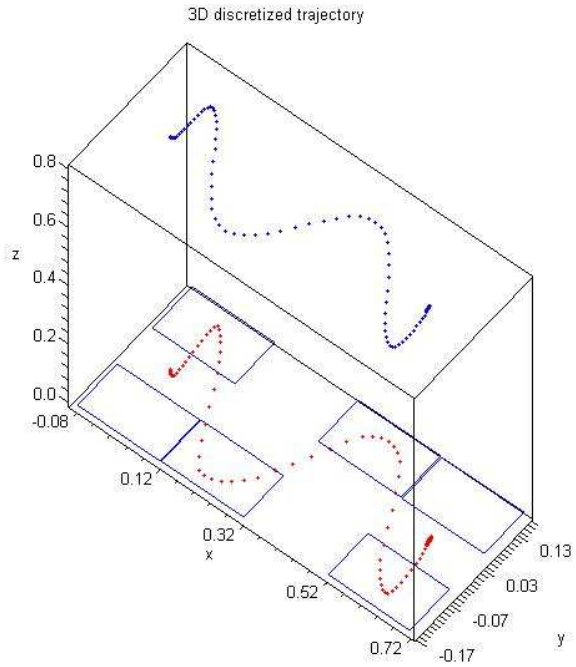


figure 5.39: *Evolution du centre de masse et du ZMP.*

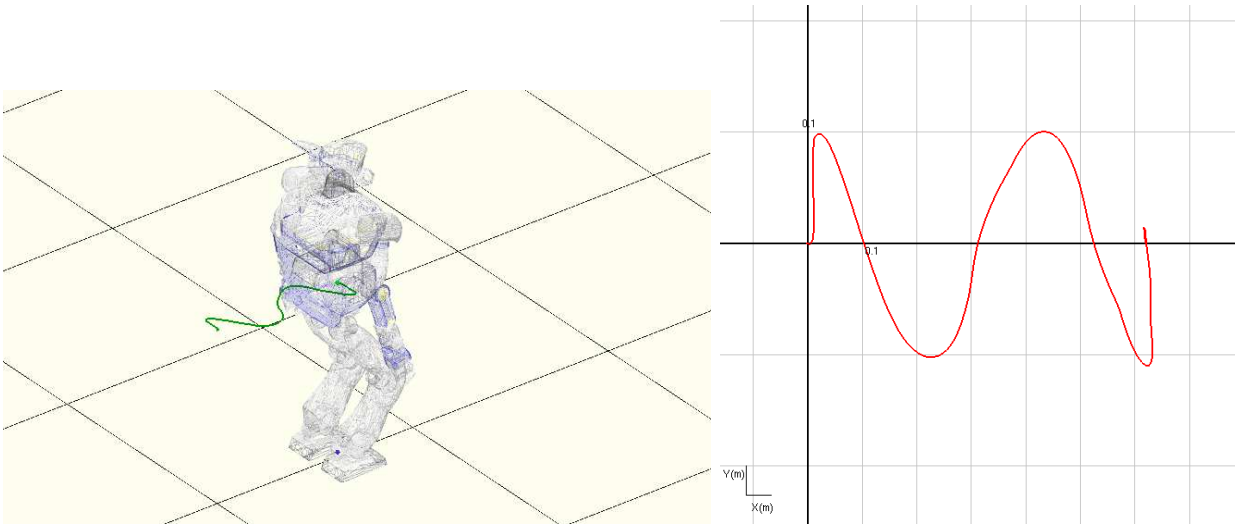


figure 5.40: *Simulation de la marche et évolution du centre de gravité en simulation.*

Ces lois peuvent être des conditions de complémentarité, des lois d'impact de Newton, le traitement du contact frottant ou bien la description des interactions entre les systèmes. Pour la simulation il est possible de choisir entre un schéma d'intégration de type *Event-driven* et un schéma de type *Time-stepping* tous les deux présentés dans le premier chapitre. Kernel utilise la partie Numerics pour résoudre les lois physiques tels que le contact frottant.

La partie Numerics propose une vaste palette de méthodes de résolution des problèmes formulés dans la partie Kernel. Cette partie est indépendante de Kernel et peut être utilisée pour d'autres librairies. Les problèmes peuvent être des équations différentielles, des problèmes de complémentarité linéaire (LCP), des problèmes de complémentarité non-linéaire (NCP), des problèmes de complémen-

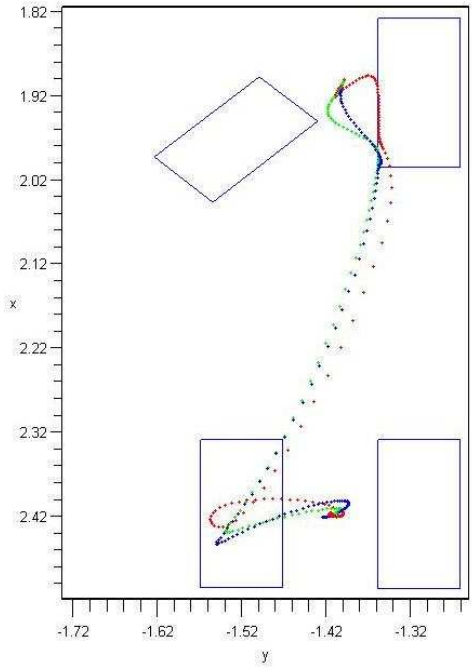


figure 5.41: *Différentes trajectoires du centre de masse pour le scénario complexe. Ces trajectoires ont été obtenues en modifiant les splines.*

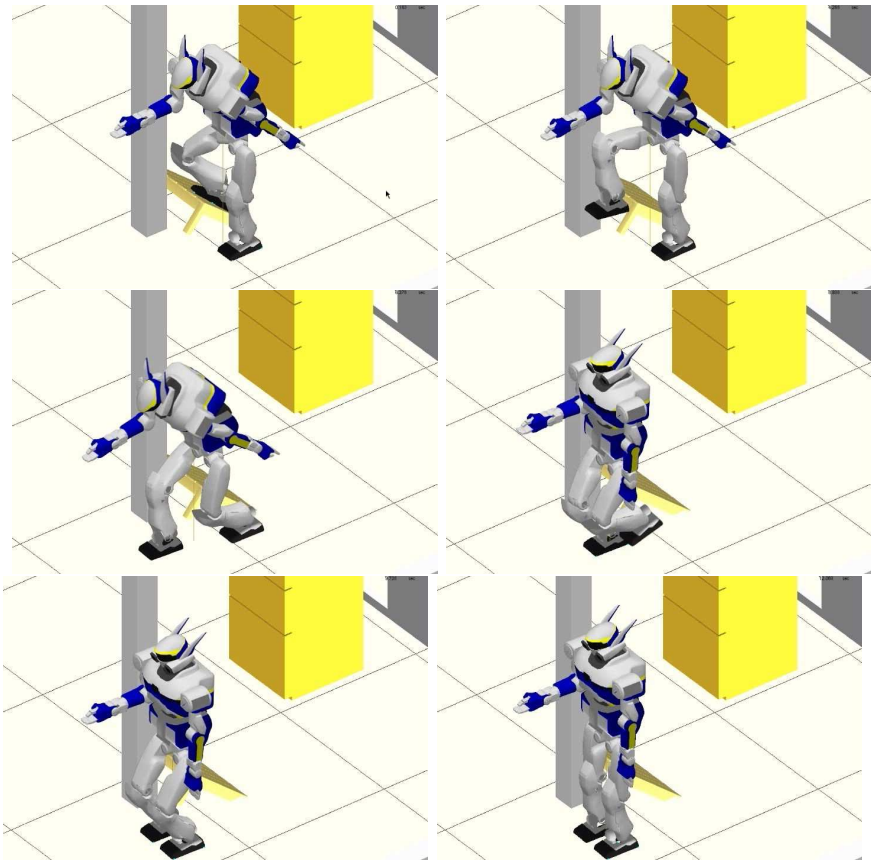


figure 5.42: *Simulation du scénario complexe.*

tarité linéaire mixte (MLCP), des programmes quadratiques (QP) ou bien des problèmes de contact frottant (méthodes itératives, projections, etc.).

Nous nous sommes principalement intéressés à l'exploitation dans le cadre d'AMELIF, des méthodes de résolution proposées par Numerics. Numerics étant développé sous Linux, pour interfacer Numerics avec AMELIF qui est principalement développé sous Microsoft Visual C++, nous avons d'abord créé un projet sous Microsoft Visual C++. La compilation de Numerics sous Windows donne une librairie qui est ajoutée à AMELIF comme extension. L'utilisateur d'AMELIF peut choisir alors entre les méthodes de résolution déjà proposées dans AMELIF et les méthodes de SICONOS. Pour ce faire il doit d'abord spécifier s'il veut résoudre le problème sans discrétisation du cône de frottement (appelée `NSGS`) ou bien par un formalisme LCP (appelé `LCP`), donc discrétisant les cônes de frottement. Puis il spécifie la méthode :

- pour les méthodes sans discrétisation du cône de frottement :
  - `AMELIFGS` : l'algorithme de Gauss-Seidel implémenté par défaut dans AMELIF et présenté dans le premier chapitre ;
  - `AMELIFGSNEWTON` : l'algorithme de Gauss-Seidel couplé avec une méthode de Newton implémenté également par défaut dans AMELIF et détaillé dans le deuxième chapitre ;
  - `AC` : le même algorithme que le précédent mais implémenté dans Numerics ;
  - `FIXEDPOINT` : une méthode du point fixe implémentée dans Numerics ;
  - `PROJECTION` : une méthode de projection implémentée dans Numerics et basée sur les travaux de De Saxcé et Feng [de Saxcé et Feng, 1998] ;
- pour le formalisme LCP :
  - `AMELIFLCP` : l'algorithme de Lemke implémenté par défaut dans AMELIF et qui est une méthode de pivot ;
  - `LEMKE` : le même algorithme mais implémenté dans Numerics ;
  - `QP` : le problème est formulé de manière équivalente par un problème QP.

Ces deux spécifications sont données dans un fichier lu à l'initialisation de la simulation.

Nous avons effectué des tests sur le robot HRP-2 en simulation dans un cas très simple où le robot se tient debout. Il y a 16 points de contact. On obtient les temps de calcul de la table 5.1, exprimés en ms.

Pour les méthodes de résolution de problèmes LCP, nous avons considéré le cas sans frottements. On peut observer des résultats inégaux, en particulier la méthode `LEMKE` est $2,6$ fois plus longue que la méthode `AMELIFLCP`. Nous pensons que ces différences viennent essentiellement de l'implémentation de ces méthodes.

La partie Kernel est quant à elle plus difficile à interfacer avec AMELIF car les structures de données de SICONOS sont très différentes de celles d'AMELIF, amenant ainsi à des adaptations de code assez conséquentes, ce que nous n'avons pas eu le temps de terminer.

Nous avons donc pu préparer les mécanismes qui permettent d'interfacer AMELIF et SICONOS mais il aurait fallu dédier plus de temps de programmation car les problèmes sont plus d'ordre technique que fondamental.

TABLE 5.1 – Temps de calcul pour les différentes méthodes.

Sans discrétisation du cône de frottement	
AMELIFGS	$0,057611$
AMELIFGSNEWTON	$0,347008$
AC	$14,6426^{*}$
FIXEDPOINT	$14,92765$
PROJECTION	$0,181587$
Formalisme LCP	
AMELIFLCP	$0,071642$
LEMKE	$0,188781$
QP	$0,078202$

* Ce temps est anormalement élevé. En fait nous avons remarqué des problèmes d'implémentation dans l'algorithme qui n'ont pu être corrigés.

5.4 Conclusion

Dans ce chapitre nous avons montré plusieurs scénarios de validation de notre simulateur. Nous avons pour cela identifié les paramètres des actionneurs et les coefficients de frottement entre le robot et les objets de l'environnement, et modélisé les capteurs présents dans le robot. Nous avons obtenu des résultats en simulation proches de ceux obtenus par l'expérimentation sur le robot réel. Encore une fois l'intérêt de procéder de la sorte pour valider notre simulateur n'est pas tant dans l'identification des paramètres réels mais plutôt dans une modélisation correcte des phénomènes physiques de telle sorte qu'on obtienne des comportements globaux du robot qui soient à peu près les mêmes en simulation et dans la réalité. Au travers d'applications ayant fait usage de notre simulateur, aussi bien pour des tâches de manipulation que pour la génération de postures, nous avons montré une manière de valider notre simulateur et prouvé l'efficacité des méthodes de calcul mises en place.

Nous avons également tenté d'interfacer AMELIF et SICONOS afin d'utiliser les différentes méthodes de résolution de problèmes de contacts frottants proposées par SICONOS. Même si cet interfaçage a posé quelques problèmes techniques, en particulier dûs à des problèmes d'implémentation de ces méthodes, nous avons montré qu'il était possible d'interfacer des librairies externes et AMELIF.

Chapitre 6

Conclusion

Ce travail présente un simulateur dynamique interactif pour robots humanoïdes avec prise en compte des contacts frottants. L'interactivité implique des temps de calcul très petits avec un retour réaliste pour l'utilisateur. Pour obtenir un rendu de qualité, il est important de ne pas trop simplifier les lois physiques issues de la mécanique générale et de bien choisir les méthodes pour le calcul de la dynamique.

Nous avons vu dans le premier chapitre que le domaine était largement étudié. Nous avons exploré les différentes méthodes pour résoudre les problèmes de contact frottant, en particulier nous nous sommes intéressés aux méthodes à base de pénalités et aux méthodes à base de contraintes. Les méthodes à base de pénalités présentent l'avantage d'être très rapides et donc idéales pour des simulations interactives temps-réel. Cependant elles sont basées sur des distances de pénétration et nécessitent un long travail de réglage des paramètres qui peuvent aboutir à des instabilités numériques. Les méthodes à base de contraintes intègrent les contraintes de non-pénétration dans les équations de la dynamique qui peuvent s'écrire sous la forme d'un problème de complémentarité linéaire (LCP). Ces méthodes sont plus précises que les méthodes à base de pénalités mais elles requièrent le calcul de l'opérateur de Delassus qui, dans le cas du contact, est la matrice d'inertie dans l'espace des contacts, et qui est coûteux. Par ailleurs, l'ajout de frottements secs rend le problème non-linéaire : le problème ne peut plus être écrit sous la forme d'un LCP à moins de linéariser la loi de frottement, autrement dit de discrétiser le cône de frottements en autant de facettes que désiré. La conséquence est une augmentation conséquente de la taille du problème à résoudre avec une perte de précision, ce qui n'est pas adapté pour des simulations interactives temps-réel. Une résolution itérative permet de conserver le cône exact et donc d'obtenir des temps de calculs performants par rapport à une approche LCP.

Avec toutes les méthodes présentées dans le premier chapitre, nous avons conçu dans le deuxième chapitre un logiciel modulaire nommé AMELIF et destiné au prototypage virtuel. Nous avons mis en

avant la modularité et la facilité avec laquelle il est possible de créer des environnements virtuels de simulation. Nous nous sommes intéressés plus particulièrement au module de la dynamique. Nous avons utilisé les méthodes à base de contraintes avec une résolution itérative pour résoudre les problèmes de contact frottant. Nous avons obtenu des simulations réalistes rapides. Nous avons aussi pu montrer que ces méthodes étaient plus adaptées à notre problème que celles utilisées dans OpenHRP2 qui sont à base de pénalités. Nous avons observé que les parties les plus coûteuses en temps de calcul étaient d'une part la détection de collision, et d'autre part, comme attendu, le calcul de l'opérateur de Delassus. Nous avons alors proposé plusieurs améliorations du calcul de la dynamique ; en particulier nous avons proposé un nouvel algorithme de calcul de l'opérateur de Delassus ainsi que de travailler par groupes de collision, ce qui nous a permis d'accélérer les temps de calcul.

Nous avons également montré une extension de notre simulateur à des corps poly-articulés à articulations sphériques tels que les avatars humains. Plusieurs approches existent pour calculer la dynamique de ces corps, parmi lesquelles les matrices de rotation, les quaternions et le vecteur de rotation. Nous avons choisi d'utiliser les quaternions car les matrices de rotation présentent des singularités qui nécessitent un traitement particulier, et le vecteur de rotation s'est révélé plus long en temps de calcul.

Grâce aux améliorations du calcul de la dynamique et à la modularité d'AMELIF, nous avons pu intégrer facilement un module d'interaction et nous avons interfacé un dispositif à retour d'effort. Cette fonctionnalité nous permet la réalisation de tâches collaboratives dans des environnements collaboratifs ou l'étude du comportement d'un système face à une perturbation. Nous avons montré les deux principales possibilités d'interaction avec l'environnement virtuel. L'une permettant la manipulation d'objets consiste à attacher un ressort-amortisseur entre l'objet et le pointeur haptique ; le paramétrage peut être cependant source d'instabilités numériques.

Nous nous sommes intéressés ensuite à la possibilité de simuler des corps déformables. Comme première application, nous avons choisi d'intégrer une semelle déformable sous les pieds du robot HRP-2 et d'en étudier les propriétés d'absorption des chocs lors de la marche du robot, par rapport à la compliance passive interne déjà présente dans les chevilles de celui-ci. Nous avons utilisé une méthode par éléments finis pour simuler les semelles. Cette méthode présente l'avantage d'être précise mais nécessite un temps de calcul élevé ; dans ce chapitre nous ne nous sommes pas intéressés à obtenir des simulations temps-réel car le but à court terme d'une telle intégration est d'étudier une solution permettant au robot de marcher sur un sol quelconque tout en gardant sa stabilité et en atténuant les chocs engendrés par l'impact des pieds sur le sol.

Notre simulateur a enfin été validé dans le dernier chapitre. Nous avons présenté une manière de valider notre simulateur en identifiant les différents paramètres physiques comme le coefficient de frottement entre les différents membres du robot et l'environnement, les paramètres des actionneurs du robot et les modèles des capteurs présents dans celui-ci, et en comparant les valeurs données par les capteurs réels avec celles calculées par notre simulateur. Nous avons obtenu des résultats similaires. Cependant, ce qui est important n'est pas l'exacte similitude des résultats fournis par les capteurs réels et virtuels mais d'obtenir des comportements similaires ; plutôt que d'insister sur cette démarche de comparaison, nous avons préféré montrer l'intérêt et la validité de notre simulateur en présentant

certaines applications de notre simulateur, en particulier pour la réalisation de tâches extrêmes ou la génération de postures. En faisant usage de notre simulateur, nous avons permis de valider les modèles et les lois de commande développés dans le cadre des travaux de recherche en optimisation de tâches extrêmes, ou la planification de points d'appuis.

Les bases d'un simulateur de qualité sont maintenant bien définies. Il existe de nombreuses perspectives pour notre travail. D'abord, pour l'interactivité, nous pouvons encore améliorer les méthodes de calcul dynamique. En effet, l'interaction temps-réel fonctionne bien pour des situations simples où nous n'avons beaucoup de points de contact. Cependant, lorsque le nombre de contact augmente sensiblement, le calcul des forces de contact est ralenti par celui de la matrice d'inertie dans l'espace des contacts. Cela est en particulier dû au fait que nous travaillons avec des forces ponctuelles et non avec des forces linéiques ou surfaciques selon le cas, ou alors avec des résultantes de forces. Travailler avec des forces linéiques ou surfaciques, ou bien avec des résultantes de forces nous permettra de réduire considérablement la taille du système et donc de résoudre le problème en des temps très courts, autorisant ainsi une interactivité accrue. On aurait dans ce cas une complexité du calcul qui ne serait plus en nombre de points de contact, mais simplement en nombre de corps en contact. Le souci d'appliquer une telle méthode serait pour les corps déformables où, en travaillant avec des éléments finis, nous sommes obligés de travailler avec des forces ponctuelles. La recherche de nouvelles méthodes et de nouveaux algorithmes toujours plus rapides est motivée par les limites de puissance des ordinateurs. Aussi l'intérêt d'une telle recherche peut être occulté par les progrès rapides dans le domaine du matériel informatique. Ce qui est en revanche important d'explorer est la recherche de nouveaux modèles physiques qui prennent en compte les phénomènes physiques de manière réaliste. Par exemple, nous avons toujours considéré un frottement sec en translation mais nous devons prendre en compte aussi les frottements en rotation et les frottements visqueux.

Nous avons vu également que la détection de collision était un élément important dans les temps de calcul. Nous pensons remplacer la détection actuelle par des méthodes récemment mises au point, comme les méthodes de calcul de volumes englobants de type STP-BV [Benallegue *et al.*, 2009].

Dans le traitement des corps déformables, nous nous sommes contentés de placer des semelles déformables sous les pieds du robot HRP-2 mais il sera important d'étendre ce traitement à tout le robot, pour permettre de simuler des avatars flexibles. Sur le long terme, nous pourrons concevoir des mouvements de marche permettant à un avatar de se mouvoir dans des environnements mous ou présentant des aspérités. Il est donc nécessaire de poursuivre l'étude sur les matériaux et les formes des semelles.

Lorsque nous aurons intégré pleinement toutes ces fonctionnalités, nous pourrons envisager de nouvelles applications futuristes. Une des applications de plus en plus courantes est le jeu vidéo. Nous pourrions concevoir un jeu vidéo où l'utilisateur serait totalement immergé dans le monde virtuel, et avec lequel il pourrait interagir pour accroître les sensations de jeu. Nous pourrions aussi imaginer qu'un robot, pour apprendre à réaliser une tâche, "imagine lui-même" ce qui se passerait s'il la réalisait, en effectuant une simulation rapide.

Références bibliographiques

[Acary et Brogliato, 2008] V. Acary et B. Brogliato. *Numerical Methods for Nonsmooth Dynamical Systems*. Springer, 2008.

[Acary et Perignon, 2007] V. Acary et F. Perignon. An introduction to siconos. Rapport technique TR-0340, INRIA, `http://hal.inria.fr/inria-00162911/en/`, 2007.

[Alart, 1993] P. Alart. Critères d'injectivité et de surjectivité pour certaines applications de $\mathbb{R}^n$ dans lui-même : application à la mécanique du contact. *Mathematical modeling and numerical analysis*, 27(2): 203–222, 1993.

[Alart et Curnier, 1991] P. Alart et A. Curnier. A mixed formulation for frictional contact problems prone to newton like solution methods. *Computer Methods in Applied Mechanics and Engineering*, 92(3):353–375, 1991.

[Allard *et al.*, 2007] J. Allard, S. Cotin, F. Faure, P.-J. Bensoussan, F. Poyer, C. Duriez, H. Delingette et L. Grisoni. Sofa – an open source framework for medical simulation. *In Medecine Meets Virtual Reality*, pages 13–18, 2007.

[Altomonte *et al.*, 2008] M. Altomonte, D. Zerbato, D. Botturi et P. Fiorini. Simulation of deformable environment with haptic feedback on gpu. *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3959–3964, Nice, France, 2008.

[Anderson et Duan, 2000] K. Anderson et S. Duan. Highly parallelizable low-order dynamics simulation algorithm for multi-rigid-body systems. *AIAA Journal on Guidance, Control and Dynamics*, 23(2):355–364, 2000.

[Anitescu et Hart, 2004] M. Anitescu et G. Hart. A constraint-stabilized time-stepping approach for rigid multibody dynamics with joints, contact and

friction. *International Journal for Numerical Methods in Engineering*, 60(14):2335–2371, 2004.

[Anitescu et Potra, 1997] M. Anitescu et F. A. Potra. Formulating dynamic multi-rigid-body contact problems with friction as solvable linear complementarity problems. *Nonlinear Dynamics*, 14(3):231–247, 1997.

[Anitescu et Potra, 2002] M. Anitescu et F. A. Potra. A time-stepping method for stiff multibody dynamics with contact and friction. *International Journal for Numerical Methods in Engineering*, 55(7):753–784, 2002.

[Arisumi *et al.*, 2007] H. Arisumi, J.-R. Chardonnet, A. Kheddar et K. Yokoi. Dynamic lifting motion of humanoid robots. *In IEEE International Conference on Robotics and Automation*, pages 2661–2667, Rome, Italy, 2007.

[Arisumi *et al.*, 2008] H. Arisumi, S. Miossec, J.-R. Chardonnet et K. Yokoi. Dynamic lifting by whole body motion of humanoid robots. *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 668–675, Nice, France, 2008.

[Baraff, 1994] D. Baraff. Fast contact force computation for nonpenetrating rigid bodies. *In SIGGRAPH*, pages 23–34, Orlando, FL, USA, 1994.

[Barclay *et al.*, 2000] G. J. Barclay, D. F. Griffiths et D. J. Higham. Theta method dynamics. *LMS J. Comput. Math.*, 3:27–43, 2000.

[Benallegue *et al.*, 2009] M. Benallegue, A. Escande, S. Miossec et A. Kheddar. Fast $\mathcal{C}^1$ proximity queries using support mapping of sphere-torus-patches bounding volumes. *In IEEE International Conference on Robotics and Automation*, Kobe, Japan, 2009.

[Bro-Nielsen et Cotin, 1996] M. Bro-Nielsen et S. Cotin. Realtime volumetric deformable models for surgery simulation using finite elements and condensation. *In Eurographics Conference, Computer Graphics Forum*, pages 57–66, Poitiers, France, 1996.

[Brogliato *et al.*, 2002] B. Brogliato, A. ten Dam, L. Paoli, F. Génot et M. Abadie. Numerical simulation of finite dimensional multibody nonsmooth mechanical systems. *Applied Mechanics*, 55(2):107–150, 2002.

[Chakraborty *et al.*, 2007] N. Chakraborty, S. Berard, S. Akella et J. Trinkle. An

implicit time-stepping method for multibody systems with intermittent contact. *In Robotics : Science and Systems*, Atlanta, GA, USA, 2007.

[Chang et Khatib, 1999] K.-S. Chang et O. Khatib. Efficient algorithm for extended operational space inertia matrix. *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 350–355, Kyongju, Korea, 1999.

[Chang et Khatib, 2000] K.-S. Chang et O. Khatib. Operational space dynamics : Efficient algorithms for modeling and control of branching mechanisms. *In IEEE International Conference on Robotics and Automation*, pages 850–856, San Francisco, CA, USA, 2000.

[Chardonnet *et al.*, 2008a] J.-R. Chardonnet, A. David, A. Kheddar et K. Yokoi. Interactive dynamic simulator for humanoid robots with deformable soles. *In RSJ Conference*, Kobe, Japan, 2008a.

[Chardonnet *et al.*, 2008b] J.-R. Chardonnet, F. Keith, A. Kheddar, K. Yokoi et F. Pierrot. Interactive dynamic simulator for humanoid with haptic feedback. *In 17th CISM-IFToMM Symposium on Robot Design, Dynamics, and Control*, Tokyo, Japan, 2008b.

[Chardonnet *et al.*, 2008c] J.-R. Chardonnet, F. Keith, S. Miossec, A. Kheddar, F. Pierrot et K. Yokoi. Simulation dynamique interactive pour les robots humanoïdes. *In 3e Journées Nationales de la Robotique Humanoïde*, Paris, France, 2008c.

[Chardonnet *et al.*, 2006] J.-R. Chardonnet, S. Miossec, A. Kheddar, H. Arisumi, H. Hirukawa, F. Pierrot et K. Yokoi. Dynamic simulator for humanoids using constraint-based method with static friction. *In IEEE International Conference on Robotics and Biomimetics*, pages 1366–1371, Kunming, China, 2006.

[Clauser *et al.*, 1969] C. E. Clauser, J. T. McConville et J. W. Young. Weight, volume, and center of mass of segments of the human body. Rapport technique, Air Force Systems Command, 1969.

[David *et al.*, 2008] A. David, J.-R. Chardonnet, A. Kheddar, K. Kaneko et K. Yokoi. Study of an external passive shock-absorbing mechanism for walking robots. *In IEEE/RAS International Conference on Humanoid Robots*, pages 435–440, Daejeon, Korea, 2008.

[de Saxcé et Feng, 1998] G. de Saxcé et Z.-Q. Feng. The bipotential method : a

constructive approach to design the complete contact law with friction and improved numerical algorithms. *Int. J. Math. and Comp. Modeling*, 28(4–8):225–245, 1998.

[Debunne *et al.*, 2001] G. Debunne, M. Desbrun, M.-P. Cani et A. H. Barr. Dynamic real-time deformations using space and time adaptive sampling. *In SIGGRAPH*, pages 31–36, Los Angeles, CA, USA, 2001.

[Delassus, 1923] E. Delassus. Sur les lois du frottement de glissement. *Bulletin de la S. M. F.*, 51:22–33, 1923.

[Duriez, 2004] C. Duriez. *Contact frottant entre objets déformables dans des simulations temps-réel avec retour haptique.* Thèse de doctorat, Université d'Evry-Val-d'Essonne, 2004.

[Duriez *et al.*, 2006] C. Duriez, F. Dubois, A. Kheddar et C. Andriot. Realistic haptic rendering of interacting deformable objects in virtual environments. *IEEE Transactions on Vizualization and Computer Graphics*, 12(1):36–47, 2006.

[Eberly, 2003] D. H. Eberly. *Game Physics.* Elsevier Science Inc., New York, NY, USA, 2003.

[Ericson, 2005] C. Ericson. *Real-Time Collision Detection.* Morgan Kaufmann, 2005.

[Escande, 2008] A. Escande. *Planification de points d'appuis pour la génération de mouvements acycliques : application aux humanoïdes.* Thèse de doctorat, Université d'Evry-Val-d'Essonne, 2008.

[Escande *et al.*, 2006] A. Escande, A. Kheddar et S. Miossec. Planning support contact-points for humanoid robots and experiments on hrp-2. *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2974–2979, Beijing, China, 2006.

[Evrard *et al.*, 2008] P. Evrard, F. Keith, J.-R. Chardonnet et A. Kheddar. Framework for haptic interaction with virtual avatars. *In 17th IEEE International Symposium on Robot and Human Interactive Communication*, pages 15–20, München, Germany, 2008.

[Featherstone, 1987] R. Featherstone. *Robot dynamics algorithms.* Kluwer Academic Publishers, 1987.

[Featherstone, 1999] R. Featherstone. A divide-and-conquer articulated-body al-

gorithm for parallel $\mathcal{O}(\log(n))$ calculation of rigid-body dynamics. *International Journal on Robotics Research*, 18(9):867–892, 1999.

[Featherstone, 2007] R. Featherstone. *Rigid Body Dynamics Algorithms*. Springer, New-York, NY, USA, 2007.

[Featherstone et Orin, 2000] R. Featherstone et D. Orin. Robot dynamics : Equations and algorithms. *In IEEE International Conference of Robotics and Automation*, pages 826–834, San Francisco, CA, USA, 2000.

[Fijany *et al.*, 1995] A. Fijany, I. Sharf et G. D'Eleuterio. Parallel $\mathcal{O}(\log(n))$ algorithms for computation of manipulator forward dynamics. *IEEE Transactions on Robotics and Automation*, 11(3):389–400, 1995.

[Förg *et al.*, 2005] M. Förg, F. Pfeiffer et H. Ulbrich. Simulation of unilateral constrained systems with many bodies. *Multibody System Dynamics*, 14(2):137–154, 2005.

[Fujiwara *et al.*, 2002] K. Fujiwara, F. Kanehiro, S. Kajita, K. Kaneko, K. Yokoi et H. Hirukawa. Ukemi : Falling motion control to minimize damage to biped humanoid robot. *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2521–2526, Lausanne, Switzerland, 2002.

[Galoppo *et al.*, 2006] N. Galoppo, M. A. Otaduy, P. Mecklenburg, M. Gross et M. C. Lin. Fast simulation of deformable models in contact using dynamic deformation textures. *In Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, pages 73–82, Vienna, Austria, 2006.

[Garsault, 2008] S. Garsault. Non-gaited dynamic motion with multi-contact transition. Mémoire de D.E.A., Ecole Centrale de Paris, 2008.

[Gavrea *et al.*, 2008] B. Gavrea, M. Anitescu et F. Potra. Convergence of a class of semi-implicit time-stepping schemes for nonsmooth rigid multibody dynamics. *SIAM J. Optim*, 19(2):969–1001, 2008.

[Gibson et Mirtich, 1997] S. F. F. Gibson et B. Mirtich. A survey of deformable modeling in computer graphics. Rapport technique TR-97-19, Mitsubishi Electric Research Laboratory, 1997.

[Grassia, 1998] F. S. Grassia. Practical parameterization of rotations using the exponential map. *Journal of Graphics Tools*, 3(3):29–48, 1998.

[Gravez *et al.*, 2005] F. Gravez, O. Bruneau et F. B. Ouezdou. Analytical and automatic modeling of digital humanoids. *International Journal of Humanoid Robotics*, 2(3):337–359, 2005.

[Grinspun *et al.*, 2002] E. Grinspun, P. Krysl et P. Schröder. Charms : A simple framework for adaptive simulation. *In SIGGRAPH*, pages 281–290, San Antonio, TX, USA, 2002.

[Hale, 2004] J. G. Hale. Forward and inverse dynamics for articulated bodies and the dojinii simulator. Rapport technique, ATR Computational Neuroscience Laboratories, 2004.

[Hale *et al.*, 2008] J. G. Hale, B. Hohl, S.-H. Hyon, T. Matsubara, E. M. Moraud et G. Cheng. Highly precise dynamic simulation environment for humanoid robots. *Advanced Robotics*, 22(10):1075–1105, 2008.

[Hanavan, 1964] E. P. Hanavan. A mathematical model of the human body. Rapport technique, Air Force Systems Command, 1964.

[Harada *et al.*, 2003] K. Harada, S. Kajita, K. Kaneko et H. Hirukawa. Pushing manipulation by humanoid considering two-kinds of zmps. *In IEEE International Conference on Robotics and Automation*, pages 1627–1632, Taipei, Taiwan, 2003.

[Harada *et al.*, 2005] K. Harada, S. Kajita, H. Saito, M. Morizawa, F. Kanehiro, F. Fujiwara, K. Kaneko et H. Hirukawa. A humanoid robot carrying a heavy object. *In IEEE International Conference on Robotics and Automation*, pages 1712–1717, Barcelona, Spain, 2005.

[Hasegawa et Sato, 2004] S. Hasegawa et M. Sato. Real-time rigid body simulation for haptic interactions based on contact volume of polygonal objects. *Computer Graphics Forum*, 23(3):529–538, 2004.

[Hauth et Strasser, 2004] M. Hauth et W. Strasser. Corotational simulation of deformable solids. *Journal of WSCG*, 12(1–3):137–145, 2004.

[Hodgins et Wooten, 1998] J. K. Hodgins et W. L. Wooten. Animating human athletes. *In International Symposium on Robotics Research*, pages 356–367, 1998.

[Hwang *et al.*, 2003] Y. Hwang, E. Inohira, A. Konno et M. Uchiyama. An order n dynamic simulator for a humanoid robot with a virtual spring-damper contact model. *In IEEE International Conference on*

Robotics and Automation, pages 31–36, Taipei, Taiwan, 2003.

[Imbert, 1995] J.-F. Imbert. *Analyse des Structures par Eléments Finis.* Cepadues-Editions, 1995.

[Isozumi *et al.*, 2004] T. Isozumi, K. Akachi, M. Hirata, K. Kaneko, S. Kajita et H. Hirukawa. Development of humanoid robot hrp-2. *Journal of the Robotics Society of Japan*, 22(8):52–60, 2004.

[Jean, 1993] M. Jean. Numerical methods for three dimensional dynamical problems. *In Conference Contact Mechanics*, page 71, Southampton, United Kingdom, 1993.

[Jean, 1999] M. Jean. The non-smooth contact dynamics method. *Computer Methods in Applied Mechanics and Engineering*, 177(3):235–257, 1999.

[Jourdan *et al.*, 1998] F. Jourdan, P. Alart et M. Jean. A gauss-seidel like algorithm to solve frictional contact problems. *Computer Methods in Applied Mechanics and Engineering*, 155(1):31–47, 1998.

[Kajita *et al.*, 2005] S. Kajita, K. Yokoi, H. Hirukawa et K. Harada. *Humanoid robot (in Japanese).* Ohmsha, Japan, 2005.

[Kanehiro *et al.*, 2004] F. Kanehiro, H. Hirukawa, et S. Kajita. Openhrp : Open architecture humanoid robotics platform. *International Journal on Robotics Research*, 23(2):155–165, 2004.

[Kaufman *et al.*, 2005] D. M. Kaufman, T. Edmunds et D. K. Pai. Fast frictional dynamics for rigid bodies. *In SIGGRAPH*, pages 946–956, Los Angeles, CA, USA, 2005.

[Kaufman *et al.*, 2008] D. M. Kaufman, S. Sueda, D. L. James et D. K. Pai. Staggered projection for frictional contact in multibody systems. *ACM Transactions on Graphics*, 27(5):164–1–164–11, 2008.

[Keith, 2007] F. Keith. Algorithmes temps-réels de calcul du modèle dynamique d'avatars virtuels en interaction haptique avec un opérateur via un objet manipulé en commun. Mémoire de D.E.A., Ecole Nationale Supérieure d'Informatique pour l'Industrie et l'Entreprise, 2007.

[Keith *et al.*, 2008] F. Keith, P. Evrard, J.-R. Chardonnet, S. Miossec et A. Kheddar. Haptic interaction with virtual avatars. *In Eurohaptics*, pages 630–639, Madrid, Spain, 2008.

[Khatib, 1987] O. Khatib. A unified approach for motion and force control of robot manipulators : The operational space formulation. *IEEE Journal of Robotics and Automation*, 3(1):43–53, 1987.

[Kokkevis, 2004] E. Kokkevis. Practical physics for articulated characters. *In Game Developers Conference*, San Jose, CA, USA, 2004.

[Larsen *et al.*, 1999] E. Larsen, S. Gottschalk, M. C. Lin et D. Manocha. A proximity query package. `http://www.cs.unc.edu/~geom/SSV/`, 1999.

[Liu et Wang, 2005] T. Liu et M. Y. Wang. Computation of three-dimensional rigid-body dynamics with multiple unilateral contacts using time-stepping and gauss-seidel methods. *IEEE Transactions on Automation Science and Engineering*, 2(2):19–31, 2005.

[Lloyd, 2005] J. E. Lloyd. Fast implementation of lemke's algorithm for rigid body contact simulation. *In IEEE International Conference on Robotics and Automation*, pages 4549–4554, Barcelona, Spain, 2005.

[Miller et Christensen, 2003] A. T. Miller et H. I. Christensen. Implementation of multi-rigid-body dynamics within a robotic grasping simulator. *In IEEE International Conference on Robotics and Automation*, pages 2262–2268, Taipei, Taiwan, 2003.

[Miossec *et al.*, 2006] S. Miossec, K. Yokoi et A. Kheddar. Development of a software for motion optimization of robots - application to the kick motion of the hrp-2 robot. *In IEEE International Conference on Robotics and Biomimetics*, pages 299–304, Kunming, China, 2006.

[Mirtich, 1996] B. Mirtich. *Impulse-based Dynamic Simulation of Rigid Body Systems*. Thèse de doctorat, University of California at Berkeley, 1996.

[Moissenet, 2007] F. Moissenet. Conception de semelles flexibles pour le déplacement sur sol incertain d'un robot humanoïde. Mémoire de D.E.A., Institut Francais de Mécanique Avancée, 2007.

[Möller, 1997] T. Möller. A fast triangle-triangle intersection test. *Journal of Graphics Tools*, 2(2):25–30, 1997.

[Moreau, 1988] J.-J. Moreau. Unilateral contact and dry friction in finite freedom dynamics. *Nonsmooth Mechanics and Applications*, 302:1–

82, 1988.

[Moreau, 2003] J.-J. Moreau. Modélisation et simulation de matériaux granulaires. *In Actes du Canum*, 2003.

[Murray *et al.*, 1994] R. M. Murray, Z. Li et S. S. Sastry. *A Mathematical Introduction to Robotic Manipulation*. CRC Press, Inc, 1994.

[Murty et Yu, 1999] K. G. Murty et F.-T. Yu. *Linear Complementarity, Linear and Nonlinear Programming*. Internet Edition, `http://ioe.engin.umich.edu/people/fac/books/murty/linear_complementarity_webbook/` édition, 1999.

[Nagasaka *et al.*, 2008] K. Nagasaka, A. Miyamoto, M. Nagano, H. Shirado, T. Fukushima et M. Fujita. Motion control of a virtual humanoid that can perform real physical interactions with a human. *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 2303–2310, Nice, France, 2008.

[Nakaoka *et al.*, 2007] S. Nakaoka, S. Hattori, F. Kanehiro, S. Kajita et H. Hirukawa. Constraint-based dynamics simulator for humanoid robots with shock absorbing mechanisms. *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 3641–3647, San Diego, CA, USA, 2007.

[Olsson *et al.*, 1998] H. Olsson, K. J. Aström, C. C. de Wit, M. Gäfvert et P. Lischinsky. Friction models and friction compensation. *European Journal of Control*, 4(3):176–195, 1998.

[Paoli et Schaztman, 2002] L. Paoli et M. Schaztman. A numerical scheme for impact problems. *SIAM Journal on Numerical Analysis*, 40(2):702–768, 2002.

[Park *et al.*, 2007] J.-J. Park, B.-S. Kim, J.-B. Song et H.-S. Kim. Safe link mechanism based on passive compliance for safe human-robot collision. *In IEEE International Conference on Robotics and Automation*, pages 1152–1157, Rome, Italy, 2007.

[Pfeiffer et Glocker, 1996] F. Pfeiffer et C. Glocker. *Multibody Dynamics with Unilateral Contacts*. John Wiley and Sons, New-York, NY, USA, 1996.

[Potra *et al.*, 2006] F. A. Potra, M. Anitescu, B. Gavrea et J. Trinkle. A linearly implicit trapezoidal method for integrating stiff multibody dynamics with contact, joints, and friction. *International*

Journal for Numerical Methods in Engineering, 66(7):1079–1124, 2006.

[Redon *et al.*, 2002] S. Redon, A. Kheddar et S. Coquillart. Fast continuous collision detection between rigid bodies. *In Eurographics Conference, Computer Graphics Forum*, 2002.

[Renouf et Acary, 2006] M. Renouf et V. Acary. Comparison and coupling of algorithms for collisions, contact and friction in rigid multi-body simulations. *In European Conference on Computational Mechanics*, Lisbon, Portugal, 2006.

[Renouf *et al.*, 2005] M. Renouf, V. Acary et G. Dumont. 3d frictional contact and impact multibody dynamics. a comparison of algorithms suitable for real-time applications. *In ECCOMAS Thematic Conference*, Madrid, Spain, 2005.

[Ruspini et Khatib, 1998] D. Ruspini et O. Khatib. Dynamic models for haptic rendering systems. *In Advances in Robot Kinematics*, pages 523–532, Strobl/Salzburg, Austria, 1998.

[Ruspini et Khatib, 2000] D. Ruspini et O. Khatib. A framework for multi-contact multibody dynamic simulation and haptic display. *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1322–1327, Takamatsu, Japan, 2000.

[Ruspini et Khatib, 1999] D. C. Ruspini et O. Khatib. Collision/contact models for dynamics simulation and haptic interaction. *In International Symposium on Robotics Research*, pages 185–195, Snowbird, UT, USA, 1999.

[Sauer et Schömer, 1998] J. Sauer et E. Schömer. A constraint-based approach to rigid body dynamics for virtual reality applications. *In ACM Symposium on Virtual Reality Software and Technology*, pages 153–162, Taipei, Taiwan, 1998.

[Schwab, 2002] A. L. Schwab. Quaternions, finite rotation and euler parameters. `http://audiophile.tam.cornell.edu/~als93/quaternion.pdf`, 2002.

[Shapiro *et al.*, 2007] A. Shapiro, D. Chu, B. Allen et P. Faloutsos. The dynamic controller toolkit. *In 2nd Annual ACM SIGGRAPH Sandbox Symposium on Videogames*, pages 15–20, San Diego, CA, USA, 2007.

[Siciliano et Khatib, 2008] B. Siciliano et O. Khatib. *Handbook of Robotics*. Springer-Verlag New-York Inc., 2008.

[Sifakis *et al.*, 2007] E. Sifakis, T. Shinar, G. Irving et R. Fedkiw. Hybrid simulation of deformable solids. *In Eurographics/ACM SIGGRAPH Symposium on Computer Animation*, pages 81–90, San Diego, CA, USA, 2007.

[Son *et al.*, 2004] W. Son, K. Kim et N. M. Amato. A generalized framework for interactive dynamic simulation for multirigid bodies. *IEEE Transactions on Systems, Man, and Cybernetics*, 34(2):912–924, 2004.

[Song *et al.*, 2004] P. Song, J.-S. Pang et R. V. Kumar. A semi-implicit time-stepping model for frictional compliant contact problems. *International Journal for Numerical Methods in Engineering*, 60(13): 2231–2261, 2004.

[Stasse *et al.*, 2008] O. Stasse, B. Verrelst, P.-B. Wieber, B. Vanderborghtand, P. Evrard, A. Kheddar et K. Yokoi. Modular architecture for humanoid walking pattern prototyping and experiments. *Advanced Robotics, Special Issue on Middleware for Robotics–Software and Hardware Module in Robotics System*, 22(6):589–611, 2008.

[Stewart et Trinkle, 1996] D. Stewart et J. C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with inelastic collisions and coulomb friction. *International Journal for Numerical Methods in Engineering*, 39(15):2673–2691, 1996.

[Stewart et Trinkle, 2000] D. Stewart et J. C. Trinkle. An implicit time-stepping scheme for rigid body dynamics with coulomb friction. *In IEEE International Conference on Robotics and Automation*, pages 162–169, San Francisco, CA, USA, 2000.

[Studer, 2008] C. W. Studer. *Augmented time-stepping integration of non-smooth dynamical systems*. Thèse de doctorat, ETH Zürich, 2008.

[Terzopoulos et Fleischer, 1988] D. Terzopoulos et K. Fleischer. Deformable models. *The Visual Computer*, 4(6):306–331, 1988.

[Terzopoulos et Witkin, 1988] D. Terzopoulos et A. Witkin. Physically based models with rigid and deformable components. *IEEE Computer Graphics and*

Applications, 8(6):41–51, 1988.

[Trinkle *et al.*, 1997] J. C. Trinkle, J. S. Pang, S. Sudarsky et G. Lo. On dynamic multi-rigid-body contact problems with coulomb friction. *Zeitschrift für Angewandte Mathematik und Mechanik*, 77(4):267–279, 1997.

[Turk et Wyeth, 2006] D. Turk et G. Wyeth. Semi-analytic method of contact modelling. *In IEEE International Conference on Robotics and Automation*, pages 1957–1962, Orlando, FL, USA, 2006.

[van den Bergen, 2003] G. van den Bergen. *Collision Detection in Interactive 3D Environments*. Morgan Kaufmann, 2003.

[Wachter et Biegler, 2006] A. Wachter et L. T. Biegler. On the implementation of an interior point filter line search algorithm for large-scale nonlinear programming. *Mathematical Programming*, 106(1):25–57, 2006.

[Walker et Orin, 1982] M. W. Walker et D. E. Orin. Efficient dynamic computer simulation of robotic mechanisms. *Transaction ASME, Journal of Dynamic Systems, Measurement and Control*, 104:205–211, 1982.

[Wang *et al.*, 2001] C. Wang, W. Timoszyk et J. Bobrow. Payload maximization for open chained manipulators : Finding weightlifting motions for a puma 762 robot. *IEEE Transactions on Robotics and Automation*, 17(2):218–224, 2001.

[Weinstein *et al.*, 2006] R. Weinstein, J. Teran et R. Fedkiw. Dynamic simulation of articulated rigid bodies with contact and collision. *IEEE Transactions on Vizualization and Computer Graphics*, 12(3):365–374, 2006.

[Yamaguchi *et al.*, 1995] J. Yamaguchi, A. Takanishi et I. Kato. Experimental development of a foot mechanism with shock absorbing material for acquisition of landing surface position information and stabilization of dynamic biped walking. *In IEEE International Conference on Robotics and Automation*, pages 2892–2899, Nagoya, Japan, 1995.

[Yamane et Nakamura, 2002] K. Yamane et Y. Nakamura. Efficient parallel dynamics computation of human figures. *In IEEE International Conference on Robotics and Automation*, pages 530–537, Washington, DC, USA, 2002.

[Yamane et Nakamura, 2006] K. Yamane et Y. Nakamura. Stable penalty-based model of

frictional contacts. *In IEEE International Conference on Robotics and Automation*, pages 1904–1909, Orlando, FL, USA, 2006.

[Yamane et Nakamura, 2007] K. Yamane et Y. Nakamura. Automatic scheduling for parallel forward dynamics computation of open kinematic chains. *In Robotics : Science and Systems*, pages 25–31, Atlanta, GA, USA, 2007.

[Zilles et Salisbury, 1995] C. B. Zilles et J. K. Salisbury. A constraint-based god-object method for haptic display. *In IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 146–151, Pittsburgh, PA, USA, 1995.

Titre : Modèle dynamique temps-réel pour l'animation d'objets poly-articulés dans des environnements contraints, prise en compte des contacts frottants et des déformations locales : application en robotique humanoïde et aux avatars virtuels

Résumé : Ce travail de thèse présente un simulateur dynamique interactif pour corps poly-articulés utilisant des méthodes par contraintes pour calculer les efforts d'interaction avec frottements. Ce simulateur est une partie intégrante d'un logiciel de prototypage nommé AMELIF. Nous nous intéressons à optimiser le calcul de la dynamique pour obtenir des simulations en temps-réel et qui nous permettent de réaliser des tâches collaboratives interactives. Nous intégrons également des modèles de déformations pour pouvoir simuler, d'une part les flexibilités internes présentes sur les robots actuels, et d'autre part les futurs robots munis d'une peau flexible. Notre simulateur a été validé par différents scénarios de manipulation et de génération de postures.

Mots-clés : Simulation dynamique, méthodes par contraintes, interaction haptique, contact avec frottements, corps déformables, manipulation, robots humanoïdes

Title : Real-time dynamic model for animation of poly-articulated objects in constrained environments with contact with friction and local deformations : application to humanoids and virtual avatars

Abstract : This thesis proposes an interactive dynamic simulation for multibody systems using constraint-based methods for computing interaction forces with friction. This simulator is a part of a general framework for prototyping called AMELIF. We focus on optimizing the computation of the dynamics to obtain real-time simulations that allow to realize interactive collaborative tasks. We also integrate deformation models in order to, first simulate internal flexibilities of actual robots, and second simulate future robots that will have a deformable skin. Our simulator has been validated through different scenarios for manipulation and posture generation.

Keywords : Dynamic simulation, constraint-based methods, haptic interaction, contact with friction, deformable bodies, manipulation, humanoid robots

Discipline :
Génie Informatique, Automatique et Traitement du Signal

Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier
UMR 5506 CNRS/Université de Montpellier 2
161 rue Ada - 34392 Montpellier Cedex 5 - FRANCE