Titre: Title:	Programmation différentielle dynamique pour la commande optimale d'actionneurs biomimétiques
Auteur: Author:	Perle Geoffroy
Date:	2014
Type:	Mémoire ou thèse / Dissertation or Thesis
Référence: Citation:	Geoffroy, P. (2014). Programmation différentielle dynamique pour la commande optimale d'actionneurs biomimétiques [Mémoire de maîtrise, École Polytechnique de Montréal]. PolyPublie. https://publications.polymtl.ca/1370/

UNIVERSITÉ DE MONTRÉAL

PROGRAMMATION DIFFÉRENTIELLE DYNAMIQUE POUR LA COMMANDE OPTIMALE D'ACTIONNEURS BIOMIMÉTIQUES

PERLE GEOFFROY
INSTITUT DE GÉNIE BIOMÉDICAL
ÉCOLE POLYTECHNIQUE DE MONTRÉAL

MÉMOIRE PRÉSENTÉ EN VUE DE L'OBTENTION
DU DIPLÔME DE MAÎTRISE ÈS SCIENCES APPLIQUÉES
(GÉNIE BIOMÉDICAL)
AVRIL 2014

UNIVERSITÉ DE MONTRÉAL

ÉCOLE POLYTECHNIQUE DE MONTRÉAL

Ce mémoire intitulé :

PROGRAMMATION DIFFÉRENTIELLE DYNAMIQUE POUR LA COMMANDE OPTIMALE D'ACTIONNEURS BIOMIMÉTIQUES

présenté par : GEOFFROY Perle
en vue de l'obtention du diplôme de : Maîtrise ès sciences appliquées
a été dûment accepté par le jury d'examen constitué de :

Mme PÉRIÉ-CURNIER Delphine, Doctorat, présidente
M. RAISON Maxime, Doct., membre et directeur de recherche
M. ACHICHE Sofiane, Ph.D., membre et codirecteur de recherche
M. BARON Luc, Ph.D., membre

À toute chose malheur est bon,
Pour Dile, Richka et Socha

REMERCIEMENTS

Je remercie mon directeur de recherche, le professeur Maxime Raison, pour l'opportunité qu'il m'a offerte de poursuivre mes études à Polytechnique Montréal, et pour son soutien financier.

Je tiens à remercier Nicolas Mansard, mon tuteur de stage MEDITIS, sans qui cette maîtrise aurait pris un tout autre tour. Merci pour tout le temps que vous m'avez consacré, pour vos conseils si justes et de m'avoir encadrée tout au long du projet malgré la distance. Travailler avec vous, toujours si enthousiaste et attentif fut un réel plaisir et m'a rendu le goût de la recherche.

Je remercie également Sofiane Achiche, mon co-directeur, de m'avoir accordé sa confiance et de m'avoir permis de finir cette maîtrise. Merci pour sa gentillesse, ses recommandations et le temps qu'il m'a accordé.

Je remercie la bourse MEDITIS, grâce à laquelle j'ai notamment pu passer plusieurs mois au LAAS à Toulouse qui furent décisifs.

Je remercie la Fondation de l'École Polytechnique pour les jeunes professeurs, pour leur soutien financier.

Je remercie les chercheurs du groupe GEPETTO au LAAS et tous les étudiants de la Grande Salle, pour leur accueil si chaleureux au sein de leur équipe dynamique et soudée.

Je remercie mes amis, et tout particulièrement Pierre-Jean Geoffroy pour son soutien longue distance indéfectible et Florent Brach, pour son expertise mathématique et son entrain à relever des défis.

Et je souhaite remercier par dessus tout mes proches. Merci à ma mère Odile, et à mes grandes soeurs Iris et Sophie-Charlotte, qui ont toujours cru en moi, même lorsque je doutais et qui m'ont encouragée et soutenue dans chacun de mes projets. Merci à Loïc, d'être là, de m'ouvrir ses bras et de m'obliger à croire en moi. Et merci à mes chats pour leur compagnie et leur soutien moral, même aux heures avancées de la nuit.

RÉSUMÉ

Les prothèses sont des dispositifs permettant de remplacer la fonction d'un organe ou d'un membre. Elles doivent donc être le plus biofidèle possible pour rendre aux patients l'intégralité de leurs capacités perdues. Dans le cas d'un membre, il est donc nécessaire de reproduire le comportement des muscles et des os lors des mouvements. Or, ce dernier est bien plus complexe qu'il n'y paraît, et ce notamment à cause de la cocontraction des muscles agonistes et antagonistes qui assure stabilité et protection des articulations et qui permet d'en moduler la rigidité. Ce dernier point est très important, pour deux raisons principales : il permet d'optimiser la consommation d'énergie et d'interagir avec l'environnement en toute sécurité.

Afin de concevoir des prothèses biofidèles, il est donc pertinent de chercher à reproduire cette rigidité variable. Plusieurs approches ont été imaginées, celle que nous étudions ici consiste à employer des actionneurs à impédance variable. Ces actionneurs possèdent des ressorts dans leur mécanisme, que l'on peut régler de façon à changer la rigidité du système au cours du temps.

Le problème principal lié à ces actionneurs est leur commande. Il est possible de la découpler, i.e. de contrôler séparément la rigidité et la position de l'actionneur, mais on n'exploite pas dans ce cas le plein potentiel de la rigidité variable.

Notre problématique se résume donc ainsi : **comment calculer la commande optimale non découplée d'un actionneur à impédance variable ?**

Dans un premier temps, nous avons modélisé un actionneur à impédance variable et linéarisé sa fonction d'évolution. Puis nous avons testé plusieurs algorithmes (la méthode par collocation et le régulateur linéaire quadratique) parmi les plus classiques en contrôle optimal pour le contrôler à impédance fixe. Si la méthode de collocation exigeait un trop long temps de calcul, les résultats avec le régulateur linéaire quadratique étaient très satisfaisants, tant en convergence qu'en stabilité. Toutefois, cette méthode n'est pas envisageable à impédance variable. En effet, la linéarisation des équations n'est plus possible.

L'algorithme *Differential Dynamic Programming* (DDP) permet de résoudre le problème de contrôle optimal pour des systèmes non-linéaires. Son fonctionnement se rapproche de celui d'un descente de Newton dans laquelle on tiendrait compte des dérivées secondes, ce qui permet justement d'étudier des systèmes pour lesquels la linéarisation est impossible. Nous avons testé et validé cet algorithme en simulation, notamment sur un saut réalisé par un robot humanoïde équipé d'actionneurs à impédance variable. Néanmoins, la complexité de cet algorithme est très élevée et limite son utilisation en temps réel. Nous avons donc cherché à l'optimiser dans un second temps.

Afin de limiter le nombre d'opérations effectuées au cours d'un calcul par DDP, nous avons employé l'hyptohèse de Gauss-Newton pour approcher les dérivées secondes. Nous avons alors refor-

mulé l'algorithme en utilisant les racines carrées des matrices en jeu et diminué la complexité : de $11n^3$ opérations à chaque cycle à $3n^3$ opérations, où n est la taille de notre fenêtre de prévision.
En conclusion, on a montré qu'il est possible de contrôler de manière non découplée un actionneur à impédance variable, grâce à l'algorithme DDP, en simulation. La suite future de ce travail en serait la validation expérimentale.

ABSTRACT

Prostheses are artificial devices that replace organs or limbs. Therefore, they must be as much biofidelic as possible to provide back to patients close to their lost capacities. In the case of a limb, the complete behavior of the muscles and bones has to be reproduced during movement. However, it is much more complex than it seems, especially because of the cocontraction of agonist and antagonist muscles. This phenomenon ensures stability and protection of the joints, but also the modulation of the stiffness. The later is very important for two main reasons : the optimization of energy consumption, and safe interaction with the environment.

To design biofidelic prostheses, it is highly desirable to reproduce this variable stiffness. Several approaches have been proposed, the one we study here is to use of Variable Impedance Actuators (VIA). The mechanical system of these actuators have springs inside and so, the global stiffness can be changed over time.

The main problem with VIA is their control. We can decouple it, this means controlling separately position and stiffness, but in this way, the full potential of variable stiffness is not exploited.

Our research question can be summed up as follow : **How to perform the optimal non decoupled control of VIA ?**

At first, we modelized a VIA and linearized its evolution function. Then, we tested several algorithms (collocation and Linear Quadratic Regulator (LQR)) among the classical methods to control it with fixed stiffness. The collocation method required a too long computation time, and we hence prefered the LQR. The results using LQR were very satisfactory both in terms of convergence and in robustness. However this method is not efficient with variable stiffness, and we can't linearize the equations.

The algorithm *Differential Dyamic Programming* (DDP) solves optimal control problem for non linear systems. It is similar to a Newton descent, in which we take into account the second derivatives. This is precisely the key to non linearizable systems. We tested and validated with simulations this algorithm, using a jump performed by a humanoid robot with VIA. However, the complexity of DDP tends to be very high and limits its use in real time. We therefore sought to optimize it in a second time.

To limit the number of operations performed during a computation, we used the Gauss Newton hyptohesis to approximate the second derivatives. We then used a square-root formulation of the DDP algorithm. The complexity decreased from $11n^3$ operations at each cycle to $3n^3$.

To conclude, uncoupled control of VIA is possible using DDP in a simulation environment. The future step of this work would be the experimental validation of the results.

TABLE DES MATIÈRES

LISTE DES FIGURES

LISTE DES SIGLES ET ABRÉVIATIONS

AwAS	Actuator with Adjustable Stiffness
CPU	Central Processing Unit
DDP	Dynamic Differential Programming
DOF	Degree Of Freedom
iLQR	iterative Linear Quadratic Regulator
LQR	Linear Quadratic Regulator
MACCEPA	Mechanically Adjustable Compliance and Controllable Equilibrium Position Actuator
MPC	Model Predictive Control
SEA	Series Elastic Actuator
SQP	Sequential Quadratic Programming
VIA	Variable Impedance Actuator

CHAPITRE 1

INTRODUCTION

Pour plus d'efficacité, de sécurité et de confort, on cherche à concevoir des prothèses biofidèles, i.e. toujours plus proches du fonctionnement biomécanique des membres humains. Or, ces derniers peuvent moduler leur flexibilité, ce qui permet d'accroître leurs performances et de faire face aux collisions, chocs et impacts, et cette particularité n'est pas toujours présente dans les prothèses, même actives. Les actionneurs biomimétiques à impédance variable apparaissent alors comme de bons candidats pour pallier à ce problème.
Leur mise au point et leur étude, très active ces dernières années, par les roboticiens se justifient par leurs nombreux avantages par rapport aux systèmes rigides lorsque les robots effectuent des tâches en interaction avec leur environnement. Tout comme pour les systèmes biologiques, ils présentent une meilleure résistance aux chocs, mais sont également plus sécuritaires pour les êtres humains en cas de collisions et permettent de stocker de l'énergie. La rigidité de ces actionneurs peut être adaptée en fonction de la tâche souhaitée : de très rigide pour des tâches de précision à très flexible avant un impact ou un contact avec l'environnement.
Le contrôle de l'impédance de ces actionneurs constitue actuellement un véritable enjeu en robotique, mais c'est également un défi au vu de la complexité de ces systèmes. Il est possible de générer des commandes optimales à impédance fixe ou en suivant un profil préalablement déterminé d'impédance. Mais l'objectif à long terme serait de calculer des commandes qui utilisent activement cette impédance variable pour atteindre efficacement des objectifs en position ou en force et pour répondre de façon adéquate aux perturbations extérieures.

1.1 Contexte

Les prothèses sont des dispositifs artificiels, internes ou externes, remplaçant un membre ou un organe dans sa fonction. Nous nous intéresserons ici aux prothèses de membres : bras, mains, chevilles, genous, etc.
Si l'histoire des prothèses remontent à l'Egypte Ancienne, elles ont beaucoup évolué depuis grâce aux avancées technologiques notamment en robotique, gagnant toujours plus en confort et efficacité. Ainsi, des premières prothèses, fixes et plus esthétiques que pratiques (fig 1.1.a), nous sommes passés à des systèmes permettant de stocker de l'énergie et de la réinvestir pour faciliter les mouvements (fig 1.1.b) (Versluys *et al*. (2009)). Aujourd'hui c'est l'ère des prothèses bioniques (fig 1.1.c)

(Herr et Grabowski (2012)), c'est-à-dire imitant les systèmes biologiques, et même, pour certains projets, commandées par la pensée (Ha *et al*. (2011) ; Kuiken *et al*. (2009)), ou plus précisément par les influx nerveux envoyés du cerveau vers les nerfs des membres amputés. Le vértitable enjeu derrière toutes ces avancées et ces recherches est de pouvoir rendre aux personnes amputées l'intégralité de leurs capacités, en vue d'une parfaite mobilité, d'une totale indépendance au quotidien et d'une meilleure intégration dans la société.

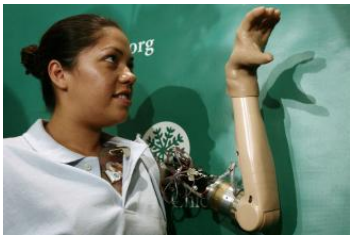
a)  b)  c) 

Figure 1.1 : **Evolution des prothèses :** a) prothèse esthétique [2], b) prothèse mécanique avec moteur (*MIT Media Lab*), c) prothèse bionique (*Rehabilitation Institute of chicago*).

Pour atteindre ces objectifs, il apparaît naturel de vouloir mettre au point des prothèses imitant au mieux le comportement biomécanique et sensoriel d'un membre sain (Bogue (2009)). Or cela est plus complexe qu'il n'y parait de prime abord : il faudrait des capteurs de température, de pression, de proprioception pour recouvrer les fonctions sensorielles, une interface avec le système nerveux central pour pouvoir contrôler la prothèse via le cerveau, et une structure mécanique permettant d'atteindre les mêmes performances en terme de précision, de vitesse, de préhension... Nous nous intéressons dans ce mémoire plus spécifiquement au comportement mécanique.

Lorsque l'on marche sur un terrain accidenté, que l'on saute par dessus un obstacle ou bien que l'on saisit un objet, sans même en avoir conscience, nos muscles vont entrer en action pour nous mouvoir, assurer notre équilibre, mais aussi amortir les chocs en dissipant de l'énergie et optimiser le métabolisme en stockant de l'énergie et en la réutilisant ultérieurement (Herr et Kornbluh (2004) ; Herr et Grabowski (2012)). Nous pouvons ainsi faire face à des perturbations et des chocs en toute sécurité. Les mécanismes biomécaniques grâce auxquels nous pouvons interagir avec notre environnement en toute quiétude ne sont pas encore entièrement compris. Cependant, il semble que la cocontraction des muscles agonistes et antagonistes joue un rôle déterminant (English et Russell (1999b)). De manière simplifiée, les muscles agonistes engendrent un mouvement donné, tandis que les muscles antagonistes s'y opposent (Fig 1.2). Lorsque les antagonistes se contractent à leur tour, c'est pour mieux contrôler le mouvement (amplitude, vitesse, précision) : ils régulent l'action des muscles agonistes. La cocontraction des muscles agonistes et antagonistes permet donc, entre

2. source : http ://tpe2013eiffel.wordpress.com/category/partie-i-presentation-chronologie

autres, de stabiliser les articulations et surtout d'en moduler l'impédance (Dal Maso *et al*. (2012) ; Stokes et Gardner-Morse (2003)).

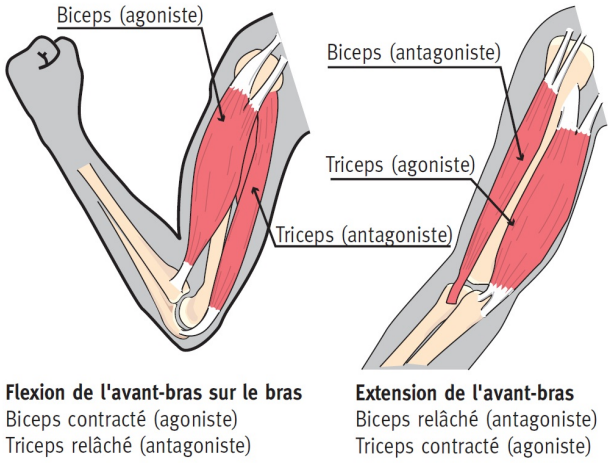


Figure 1.2 : **Cocontraction des muscles agonistes et antagonistes** [3]

En robotique, on travaille classiquement avec des systèmes rigides, relativement simples à contrôler (Ortega et Spong (1989)). Or de tels actionneurs ne permettent pas de reproduire un comportement biofidèle. Pire, ils pourraient s'avérer dangereux s'ils étaient utilisés directement sur des prothèses, sans systèmes d'amortissement appropriés. Les caractéristiques mécaniques des membres sont liées aux muscles et tendons. On a donc cherché à développer des muscles artificiels, notamment avec des systèmes pneumatiques et hydrauliques (Caldwell et Tsagarakis (2002) ; Herr et Kornbluh (2004)). Leurs performances sont en effet proches de celles de véritables muscles, mais la difficulté à les contrôler et les contraintes pratiques (entretien, fuites...) sont trop rédhibitoires. Des actionneurs à impédance variable en revanche pourraient être une solution viable (Martinez-Villalpando et Herr (2009)). Ce sont des actionneurs auxquels ont été ajoutés des éléments flexibles, généralement des ressorts. La liaison réalisée par de tels actionneurs possède alors une impédance ajustable à tout instant. La recherche sur ces systèmes est très active à l'heure actuelle (Serio *et al*. (2011) ; Shin *et al*. (2010) ; Vanderborght *et al*. (2009)). En effet, grâce à eux, on serait en mesure de développer des robots pouvant interagir avec des êtres humains de manière tout à fait sécuritaire (Tonietti *et al*. (2005)), mais également des robots humanoïdes plus performants dans des environnements inconnus (Wolf et Hirzinger (2008)) que ceux avec des actionneurs rigides. Leur emploi pour des prothèses de bras (English et Russell (1999b)), de chevilles (Blaya et Herr (2004)), ou d'épaules (Sensinger et Weir (2008)) a déjà été testé et les résultats sont encourgeants.
Toutefois, la commande de tels actionneurs est délicate et limite pour le moment leur utilisation.

3. source : http ://tpeprotheses.e-monsite.com/pages/le-bras-humain/le-bras-muscles-et-impulsions.html

Dans ce mémoire, nous nous sommes concentrés sur ce problème, et ce d'un point de vue purement robotique. Certes, les actionneurs à impédance variable sont des dispositifs inspirés des systèmes biologiques et qui trouvent une application biomédicale directe. Cependant, pour mener à bien notre projet, il a été primordial de connaître les bases du contrôle en robotique. La revue de littérature dans ce mémoire n'abordera donc que ces aspects.

1.2 Objectifs de recherche

Cette maîtrise a porté sur le contrôle optimal des actionneurs à impédance variable. Voici les objectifs qui l'ont étayée :

1. modéliser un actionneur à impédance variable :
   à partir d'un modèle relativement simple d'actionneur, on a déterminé les équations régissant son évolution et déterminé quels étaient les coûts pertinents à prendre en compte lors du contrôle optimal ;
2. réaliser la commande d'un actionneur à impédance variable, en position et couple, sous l'hypothèse d'une impédance fixe :
   cette phase permet notamment de tester plusieurs algorithmes, de les comparer et de sélectionner celui qui semble le plus adéquat pour la suite du projet de recherche ;
3. en réaliser la commande sans découpler position et impédance :
   il s'agit ici de contrôle optimal, il ne suffit donc pas de trouver une trajectoire menant d'un point A à un point B, mais de trouver celle minimisant les coûts retenus précédemment ;
4. valider le système de contrôle par une simulation de saut humanoïde ;
5. optimiser les temps de calcul de l'algorithme retenu en vue d'un contrôle en temps réel.

1.3 Organisation du mémoire

Ce mémoire est un mémoire par articles : les chapitres 4 et 5 sont les versions originales d'articles acceptés à des conférences scientifiques avec comité de lecture. Une unique bibliographie regroupe l'ensemble des références citées dans ce mémoire, les articles compris.
La structure de ce mémoire est la suivante :

1. **Chapitre 2 :** une revue de littérature portant sur le contrôle optimal en robotique et les actionneurs à impédance variable
2. **Chapitre 3 :** ce chapitre présente les travaux réalisés en amont des articles (modélisation d'un actionneur, contrôle sous l'hypothèse d'une impédance fixe et les limites des algorithmes utilisés pour cela) ainsi qu'une synthèse des deux articles.

3. **Chapitre 4 :** ce chapitre est le premier article, soumis et accepté à la conférence *European Control Conference 2014*. Il porte sur l'intérêt de l'algorithme DDP pour le contrôle optimal des actionneurs à impédance variable, ce dernier est par ailleurs validé par un test de saut humanoïde.
4. **Chapitre 5 :** ce chapitre correspond au second article, soumis et accepté à la conférence *Advances in Robot Kinematics 2014*. Il s'agit d'un apport technique au précédent : on présente une diminution de la complexité de l'algorithme DDP sous l'hyptohèse de Gauss.
5. **Chapitre 6 :** une discussion générale et une conclusion, sur l'ensemble des recherches menées au cours de la maîtrise et préalablement présentées, ainsi que sur les possibilités de travaux futurs.

CHAPITRE 2

REVUE DE LITTÉRATURE

Nous souhaitons présenter au lecteur quelques concepts de base de la robotique à travers cette revue de littérature. Toutefois, nous n'entrerons pas dans les détails ni ne dresserons de liste exhaustive de stratégies ou méthodes, car si quelques notions facilitent la lecture et la compréhension de ce mémoire, ce dernier n'a pas pour ambition une étude approfondie de ce domaine.

2.1 Robotique polyarticulée

2.1.1 Représentation de l'état d'un robot

Considérons un robot possédant n liaisons, et donc n degrés de liberté, l'état de chacune de ces liaisons peut être caractérisé par son angle q_i, où i renvoie à la i-ème liaison. On définit alors le vecteur $q = (q_1; \ldots; q_n)$, le vecteur des angles des articulations ou *joint angle vector* en anglais, qui caractérise entièrement l'état du robot à chaque instant. L'ensemble des vecteurs Q possibles est appelé l'ensemble des articulations ou *Joint Space* en anglais.
Notons que dans le cas d'un robot humanoïde, il faut également prendre en compte le fait que le robot peut se mouvoir entièrement dans la limite de son espace de travail. On définit alors un vecteur de base x_b donnant la position d'un point du robot dans l'espace ainsi que l'orientation du robot. Le vecteur $(x_b; q)$ est alors la représentation complète du robot dans son environnement.

2.1.2 Les différentes approches du contrôle

On présente ici les différentes approches possibles du problème de contrôle d'un robot, pour qu'il réalise une tâche spécifique, telle que attraper un objet ou aller à un endroit précis. Dans la majorité des cas, on sait quelle position on souhaite atteindre sans savoir à quel état du robot cela correspond, i. e. à quelles positions angulaires de chaque liaison, or nous avons besoin de connaître cet état pour guider le robot jusqu'à la position désirée. L'approche directe du problème, du vecteur d'état vers la position du robot, n'étant pas viable, on adopte le point de vue inverse du problème : depuis la position vers le vecteur d'état, pour résoudre ce problème.

Géométrie inverse

En géométrie inverse, on ne considère que la structure du robot et aucune force, vitesse, ni accélération. On cherche à atteindre une position avec le robot, qui ne dépend donc que de l'état des différentes liaisons. On peut alors définir une fonction de tâche $h(q)$, la position de la main du robot par exemple, dépendant des angles des liaisons q et h^* notre objectif. On cherche alors q^* tel que $h(q^*) = h^*$.
Si h est une fonction inversible, alors $q^* = h^{-1}(h^*)$. Mais c'est rarement le cas avec des robots polyarticulés (dont les humanoïdes). En effet, ils possèdent un nombre conséquent de liaisons à contrôler et on doit respecter un certain nombre de contraintes, telles que rester stable pour un robot humanoïde. Tout ceci mène à des fonctions de tâches h, très complexes et non inversibles. On privilégie alors une approche numérique et itérative pour résoudre le problème, telle que la descente de Gauss-Newton où l'on cherche à minimiser la norme $||h(q) - h^*||$ selon q.

Cinématique inverse

Tout comme dans l'approche géométrique, on ne tiendra pas compte ici des différentes forces s'appliquant sur le robot. En revanche, on ne cherchera pas simplement une configuration optimale du robot, mais toute une trajectoire permettant d'atteindre cette configuration. Pour résoudre ce problème, on introduit la jacobienne J définie par : $J = \frac{\delta h}{\delta q}$.
A partir d'un état initial h_0, on peut calculer par petits incréments successifs la trajectoire globale pour atteindre notre objectif h^*. En effet, pour obtenir une petite variation δh connue, il suffit de procéder à une petite variation des angles des liaisons : $\delta q = J^{-1}\delta h$.
Cependant, dans le cas général, J n'est pas inversible. Par ailleurs, pour limiter les temps de calculs lorsqu'on étudie un robot possédant un grand nombre de degrés de liberté, on emploiera des méthodes variationnelles pour trouver la trajectoire désirée. Ces méthodes sont nombreuses et regroupent entre autres la pseudo-inverse de la jacobienne et la méthode des moindres carrés.

Dynamique inverse

Comme l'appellation *dynamique* l'indique, cette approche consiste à établir le lien entre les forces et les couples s'appliquant au robot et son mouvement. La dynamique directe donnera le mouvement du robot en fonction des forces qui lui sont appliquées tandis la dynamique inverse cherche à déterminer les forces à appliquer pour obtenir une certaine trajectoire.
Soit $\tau = (\tau_1; ...; \tau_n)$ le couple résultant sur chaque liaison, le modèle de la dynamique inverse s'écrit alors :

$$A(q)\ddot{q} + b(q, \dot{q}) + g(q) = \tau \tag{2.1}$$

où $A(q)$ est la matrice d'inertie du robot, $b(q, \dot{q})$ la force centrifuge et la force de Coriolis, et $g(q)$ les forces de gravité.

2.1.3 Planification de mouvement

La planification de mouvement consiste à calculer des trajectoires pour un robot entre une position initiale et une position finale, tout en respectant un certain nombre de contraintes telles que l'absence de collision. Nous présentons ici plusieurs stratégies possibles pour que le lecteur entrevoie l'étendue et la complexité de ces problèmes.

Linéarisation instantanée

Considérons la fonction d'évolution de notre système :

$$\dot{x} = f(t, x, u) \tag{2.2}$$

où x est l'état du système et u le vecteur de contrôle. La planification de mouvement revient à chercher le contrôle u à chaque instant pour atteindre l'état x_{fin} désiré à l'instant final t_{fin}.
La solution brutale serait de résoudre directement cette équation sans envisager de simplification, mais les temps de calcul deviennent rapidement rédhibitoires, et ce d'autant plus que le robot considéré possède des liaisons (Canny (1988)). La solution classique est alors de linéariser la fonction d'évolution f à chaque instant. Si la fonction d'évolution ne tient compte que de la géométrie du système, nous ferons alors de la cinématique inverse. Si en revanche elle tient compte des forces et des couples, il s'agira de dynamique inverse. Si l'espace opérationnel est choisi correctement, alors les trajectoires seront généralement faciles à exprimer (Khatib (1987)).
Cette solution a néanmoins des limites. En effet, en robotique, on utlise des réducteurs pour augmenter le couple développé en sortie tout en réduisant la vitesse de rotation des moteurs, pour notamment limiter les coûts énergétiques. Mais du fait de la présence des réducteurs, les couples articulaires deviennent des fonctions très complexes et non linéaires de l'entrée des moteurs (Mansard (2013)). Toute linéarisation devient alors impossible car cela décrirait un comportement trop éloigné de la réalité.
Notons par ailleurs que dans le cadre de ce mémoire, nous étudions des actionneurs à impédance variable, qui contiennent des ressorts dans leur système. Or, linéariser le comportement d'un ressort serait dépourvu de sens : quelle que serait la force en entrée du ressort, on aurait un déplacement nul en sortie.

Echantillonnage aléatoire

La planification de mouvement par échantillonnage aléatoire est une méthode probabiliste. Elle ne repose pas sur une représentation exacte ni aprochée de l'espace des configurations avec calcul de solutions. Au contraire, il s'agit d'explorer l'espace des configuations admissibles jusqu'à obtenir une trajectoire pour aller de l'état initial à l'état final en passant par des états transitoires tirés aléatoirement (Kavraki *et al*. (1996)).
Dans un premier temps, l'algorithme va rechercher les composantes connexes de l'espace des configurations admissibles, cela revient à séparer cet espace en sous espaces à l'intérieur desquels toutes les configurations sont atteignables à partir de n'importe quelle autre. La figure (2.1.a) illustre un ensemble de configurations possèdant deux composantes connexes.
Pour chercher une trajectoire entre deux états donnés, l'algorithme va vérifier qu'ils appartiennent à la même composante connexe (Fig. 2.1 b), et donc qu'une telle trajectoire existe. Il va ensuite tirer aléatoirement un état dans cette composante connexe. Il teste s'il est possible de le relier directement à l'état initial et/ou final, par une trajectoire linéaire par exemple. Si c'est le cas il aura trouvé une trajectoire admissible. Sinon, il tire aléatoirement d'autres configurations jusqu'à pouvoir en relier depuis l'état initial jusqu'à l'état final (Fig. 2.1 c). Il est ensuite possible de lisser la trajectoire trouvée en tirant d'autres points aléatoirement autour de cette trajectoire.

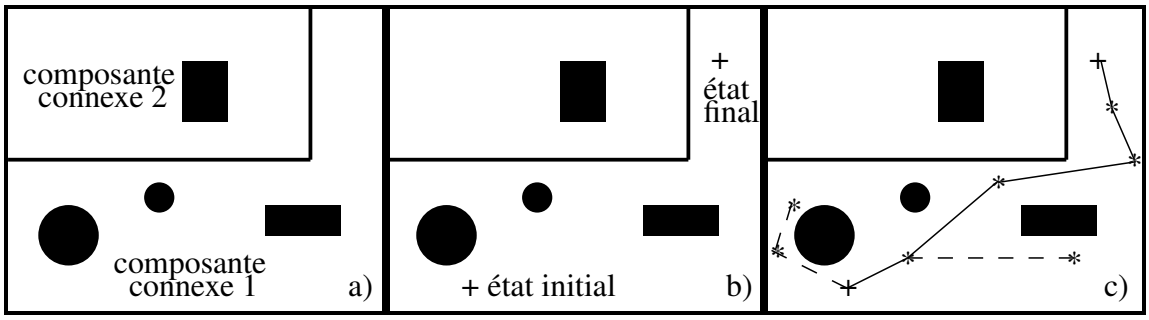


Figure 2.1 : **Planification de trajectoire par tir aléatoire :** a) Espace des configurations à deux composantes connexes. b) Etat initial d'un robot et état final à atteindre, dans la même composante connexe. c) Trajectoire par tir aléatoire : des états intermédiaires (*) sont tirés aléatoirement jusqu'à obtention d'une trajectoire complète entre l'état initial et l'état final (en traits pleins), les portions de trajectoire calculées au cours du processus et inutilisées pour la trajectoire finale sont représentées en pointillés.

2.2 Contrôle optimal en robotique

2.2.1 Présentation du contrôle optimal

Le contrôle optimal se différencie de la planification de mouvement en ce que la trajectoire recherchée ne doit pas simplement mener d'une configuraion initiale à une configuration finale, mais doit également minimiser un certain coût. Le coût, défini par le manipulateur, peut regrouper plusieurs critères, tels que la consommation d'énergie que l'on cherche à optimiser par exemple. Il représentera également la tâche à accomplir, reformulée comme un problème de minimisation.

Problème continu

Considérons un robot dont on note $x(t)$ l'état et $u(t)$ le contrôle, et de fonction d'évolution f :

$$\dot{x} = f(t, x(t), u(t)) \tag{2.3}$$

On souhaite que ce robot atteigne un état donné x^*, en un temps T. Par ailleurs, on désire minimiser certains coûts $c(t, x(t), u(t))$ à chaque instant, au cours de cette trajectoire, où c est une fonction positive ou nulle. Notons que ces coûts peuvent dépendre du contrôle employé $u(t)$ et de l'état du robot $x(t)$. Le problème, qui consiste à trouver un contrôle optimal U^* pour atteindre l'état x^* avec un coût minime (Sanchez et Ornelas-Tellez (2013)), peut alors s'écrire de la sorte :

$$\min_u \left(\int_{t=0}^{T} c(t, x(t), u(t)) dt \right), \qquad x(T) - x^* = 0 \tag{2.4}$$

Pour faciliter la compréhension, nous avons choisi de ne pas évoquer ici les éventuelles contraintes que devrait respecter le système, pour éviter les collisions par exemple, et qui prendraient la forme suivante :

$$h(t, x(t), u(t)) \leqslant 0 \qquad \forall t \in [0; T] \tag{2.5}$$

En effet, cela ne modifie en rien le fonctionnement des méthodes présentées ci-après de recherche de minimum.

Problème discret

Discrétisation complète du problème Il peut être intéressant de discrétiser le problème précédent pour le résoudre (Grieder *et al*. (2004)). Cela revient à considérer des pas de temps dt si petits que l'on puisse faire l'approximation suivante : $dx = dt\dot{x}$. Les trajectoires, contrôles et coûts ne

sont plus des fonctions, mais des vecteurs : $X = (x_i)_{1 \leqslant i \leqslant n}$, $U = (u_i)_{1 \leqslant i \leqslant n-1}$ et $C = (c_i)_{1 \leqslant i \leqslant n}$ tels que :

$$x_{i+1} = x_i + dtf(t, x_i, u_i) \tag{2.6}$$

$$c_i = c(t_i, x_i, u_i) \tag{2.7}$$

Ainsi, tout en respectant la condition (2.6), nous n'avons plus à minimiser une intégrale mais une somme finie :

$$\min_U (\sum_{i=1}^{n} c_i), \qquad x_T = x^* \tag{2.8}$$

Problème semi-discret Dans la formulation précédente, à la fois le contrôle et l'état du système ont été discrétisés. Il est cependant possible de n'en discrétiser que l'un des deux. Nous décrirons le cas où seul le contrôle est discrétisé. En effet, d'une part, nous l'emploierons dans ce qui suit et d'autre part, la méthode pour ne discrétiser que l'état est très similaire.
Nous considérons donc ici notre intervalle de temps $[0; T]$, que nous subdivisons en petits intervalles dt, pour obtenir comme un maillage du temps dont les noeuds seraient $t_i = i * dt$. Nous discrétisons le contrôle : celui-ci sera considéré comme constant sur chaque subdivision de notre intervalle de temps :

$$\forall t \in [t_i; t_{i+1}[\qquad u(t) = u_i \tag{2.9}$$

Le problème semidiscret est alors :

$$\min_U (\sum_{i=0}^{n-1} \int_{t_i}^{t_{i+1}} c_i(t) dt), \qquad x_T = x^* \tag{2.10}$$

$$\tag{2.11}$$

où $c_i(t) = c(t, x(t), u_i)$ et où U est la séquence de contrôle.

2.2.2 Méthodes de résolution

Plusieurs méthodes numériques ont été développées pour résoudre le problème optimal, cependant chacune a été mise au point pour résoudre des problèmes spécifiques (Zefran (1996)). Il n'existe donc pas de méthodes universelles pour résoudre tout problème de contrôle optimal. Les méthodes pour résoudre les problèmes de contrôle optimal peuvent être subdivisées en trois catégories (Diehl *et al*. (2006) ; Leboeuf *et al*. (2006)) :

1. les méthodes indirectes : l'objectif est de passer d'un problème aux limites à un système d'équations différentielles linéaires puis à le résoudre numériquement ;
2. les méthodes directes : elles consistent à discrétiser les équations, pour passer d'un problème de dimension infinie à un problème de dimension finie, puis à résoudre ce nouveau problème d'optimisation ;
3. les méthodes dynamiques : elles utilisent le principe de Bellman pour calculer rétrospectivement un contrôle U^* permettant d'approcher la trajectoire optimale.

Nous présentons ci-dessous plusieurs de ces algorithmes, les plus connus et les plus usités, et ceux employés au cours de nos recherches.

Méthodes directes

La méthode de la collocation et les méthodes de tir simple et de tir multiple appartiennent à la catégorie des méthodes directes. Toutes trois utilisent donc une version discrète du problème de contrôle optimal. Elles diffèrent par la façon dont la trajectoire X est calculée au cours de l'optimisation, autrement dit elles diffèrent par leur façon de résoudre approximativement l'équation différentielle régissant l'évolution du système.
Le problème étant discrétisé, on fait face à présent à un problème de dimension finie. On pourra donc utiliser un algorithme d'optimisation tel que le *Sequential Quadratic Programming* (SQP) (Nocedal et Wright (2006b)), pour déterminer les directions dans lesquelles déplacer une trajectoire initiale pour en obtenir une seconde, plus proche de l'optimum. Ces méthodes ont donc besoin d'une trajectoire initiale, non optimale, typiquement on choisit celle correspondant à une commande nulle. Puis on va itérer sur cette trajectoire jusqu'à approcher de manière acceptable la trajectoire optimale.

Collocation

La méthode de la collocation s'appuie sur la formulation complètement discrète du problème optimal. L'idée maîtresse est d'approximer, sur chaque intervalle de temps $[t_i; t_{i+1}]$, la solution de l'équation différentielle qui décrit l'évolution du système, par des fonctions de bases telles que des polynômes (Ascher *et al*. (1988)), on choisira d'ailleurs des polynômes pour le développement ci-après.
Disons que l'on souhaite interpoler nos trajectoires par des polynômes de degré k. On subdivise alors chacun de nos intervalles en k : $[t_i; t_{i,1}; ...; t_{i,j}; ...; t_{i,k}; t_{i+1}]$. Les points $t_{i,j}$ sont appelés les *points de collocation*, et on notera les états intermédiaires du système aux temps $t_{i,j}$: $x_{i,j}$, et le

contrôle : $u_{i,j}$. Le polynôme Π devra alors respecter les conditions suivantes :

$$\Pi(t_i) = x_i \tag{2.12}$$

$$\dot{\Pi}(t_{i,j}) = f(t_{i,j}, x_{i,j}, u_{i,j}) \qquad \forall \quad 1 \leqslant j \leqslant k \tag{2.13}$$

La collocation, bien qu'elle soit coûteuse en calcul, est notamment efficace pour étudier des systèmes instables (Diehl *et al*. (2006)). Elle est cependant difficile à mettre en place car elle est très sensible au choix des points de collocation. En effet, nous les avons pris réguliers ici pour plus de clarté, mais rien n'empêche de les choisir proches en début de trajectoire puis plus clairsemés par exemple, pour améliorer la convergence (Schulz (1996)).

Tir simple

La méthode du tir simple utilise la version semi-discrète du problème : seul le contrôle est discrétisé. Les états du système peuvent alors être notés $x(t, U)$ où U est la séquence de contrôle. Et on cherche U^*, un vecteur de dimension finie qui résoudrait :

$$\min_U \int_0^T c(t, x(t, U), U)dt \tag{2.14}$$

L'avantage de cette méthode est notamment de limiter les degrés de liberté pour l'optimisation, donc de la rendre plus facile et rapide (Diehl *et al*. (2006)). En revanche, cette méthode n'est pas toujours stable, ni robuste (Gill *et al*. (2000)).

Tir multiple

Cette méthode s'inspire des deux précédentes. En effet, on résout l'équation différentielle sur chaque intervalle $[t_i; t_{i+1}]$ de manière indépendante, avec comme valeur initiale de l'état s_i, une nouvelle variable, et avec un contrôle u_i constant. On obtient alors des sections de trajectoires pour chaque intervalle de temps, et pour chacune on calcule, via l'intégrale, le coût $c_i(s_i, u_i)$ correspondant.
On est ainsi ramené à un problème d'optimisation fini. Pour assurer la continuité de la trajectoire, on impose les contraintes suivantes à tous les noeuds :

$$s_{i+1} = x(t_{i+1}, s_i, u_i) \tag{2.15}$$

Les états s_i sont donc des degrés de liberté artificiels.
Cette méthode est plus facile à mettre en oeuvre que la collocation dans la mesure où l'on n'a pas à choisir les points de collocation, mais est également très coûteuse. Comparée à la méthode de tir

simple, elle est plus stable et plus robuste (Gill *et al*. (2000) ; Diehl *et al*. (2006)).

Régulteur linéaire quadratique ou Linear Quadratic Regulator

Le Linear Quadratic Regulator (LQR) est une méthode pour résoudre des problèmes de contrôle optimal dont l'équation différentielle est linéaire et la fonction de coût J quadratique :

$$\dot{x}(t) = A(t)x(t) + B(t)u(t) \tag{2.16}$$

$$J(x_0, u(t)) = \frac{1}{2}x(T)^T Q(T)x(T) + \frac{1}{2}\int_0^T (x(t)^T Q(t)x(t) + u(t)^T R(t)u(t)dt) \tag{2.17}$$

où les matrices $Q(t)$ et $R(t)$ sont symétriques et définies positives.
Cette méthode permet de réduire le problème d'optimisation à une résolution d'équation différentielle dans le cas d'un horizon fini, et d'une équation algrébrique en horizon infini (Blanchet (2007)). Si les matrices de poids R et Q sont en plus constantes au cours du temps, cette résolution est grandement simplifiée, et le calcul de la commande optimale devient très rapide et aisé. Néanmoins, on ne peut employer le LQR que dans des cas restreints : les problèmes linéaires, et les problèmes linéarisés, lorsque la linéarisation est possible et ne mène pas à l'instabilité du système. Nous présentons ci-dessous le LQR pour un problème continu, mais il est tout à fait applicable à un problème discret, suivant le même raisonnement. Par ailleurs, nous ne présentons que les grandes lignes de l'algorithme, sans démonstration, le lecteur pourra consulter l'ouvrage de Blanchet (Blanchet (2007)) pour plus de détails.
Le principe de la méthode LQR repose sur l'introduction d'un Hamiltonien H :

$$H = \frac{1}{2}(x(t)^T Q(t)x(t) + u(t)^T R(t)u(t)) + \lambda(A(t)x(t) + B(t)u(t)) \tag{2.18}$$

où λ est un multiplicateur de Lagrange. On va alors chercher la commande $u^*(t)$ qui minimise la fonction $J(t)$, soit à horizon fini T, soit à horizon infini. On notera $x^*(t)$ et $\lambda^*(t)$ la trajectoire et le multiplicateur de Lagrange correspondants.
A l'optimum, la dérivée de l'Hamiltonien par rapport à la commande sera nulle, nous aurons également les relations suivantes :

$$\frac{\delta H}{\delta u} = 0 \quad ; \quad \frac{\delta H}{\delta x} = -\dot{\lambda}^*(t) \quad ; \quad \frac{\delta H}{\delta \lambda} = \dot{x}(t) \tag{2.19}$$

Ceci mène à une nouvelle équation différentielle :

$$\begin{pmatrix} \dot{x}^*(t) \\ \dot{\lambda}^*(t) \end{pmatrix} = \begin{pmatrix} A(t) & -B(t)R^{-1}(t)B(t)^T \\ -Q(t) & -A(t)^T \end{pmatrix} \begin{pmatrix} x^*(t) \\ \lambda^*(t) \end{pmatrix} \tag{2.20}$$

et à une relation entre $u^*(t)$ et $\lambda^*(t)$:

$$u^*(t) = -R^{-1}(t)B(t)^T\lambda^*(t) \tag{2.21}$$

Horizon fini S'il existe une matrice $P(t)$ telle que : $\lambda^*(t) = P(t)x^*(t)$, alors l'équation différentielle (2.20) peut être réécrite sous la forme d'une équation différentielle matricielle, appelée *équation de Riccati* :

$$\dot{P}(t) = P(t)B(t)R^{-1}(t)B^T(t) - P(t)A(t) - Q(t) - A^T(t)P(t) \tag{2.22}$$

de condition finale : $P(T) = Q(T)$.
Il suffit alors de résoudre cette équation différentielle sur $P(t)$, et la commande à employer pour minimiser la fonction coût J sera alors :

$$u^*(t) = -K(t)x^*(t) \tag{2.23}$$

$$K(t) = R^{-1}(t)B^T(t)P(t) \tag{2.24}$$

La matrice $K(t)$ est appelée *matrice de gain de Kalman*.

Horizon infini On suppose ici que la matrice$Q(t)$ s'écrit à l'aide d'une matrice $G(t)$: $Q(t) = G^T(t)G(t)$ et que la matrice $R(t)$ est la matrice identité.
A horizon infini, nous n'avons pas de condition finale et l'équation (2.20) se simplifie pour donner l'*équation algébrique de Riccati* :

$$P(t)B(t)R^{-1}(t)B^T(t) - P(t)A(t) - Q(t) - A^T(t)P(t) = 0 \tag{2.25}$$

La commande à employer sera alors :

$$u^*(t) = -B^T(t)P(t)x^*(t) \tag{2.26}$$

$$\tag{2.27}$$

2.2.3 Programmation différentielle dynamique ou Differential Dynamic Programming

L'algorithme *Differential Dynamic Programming* est celui que nous avons retenu au cours de nos recherches pour contrôler un actionneur à impédance variable, après en avoir testés plusieurs (méthode LQR, collocation et DDP). Nous en expliquons ici brièvement le fonctionnement, les avantages et inconvénients, pour plus de détails, on pourra consulter le chapitre 4 de ce mémoire.

Principe

Le DDP (Tassa *et al*. (2012a)) est un algorithme itératif, proche de la descente de Newton et qui utilise la forme discrète du problème optimal, comme expliquée à la section (2.2.1). Le principe, comme dans les méthodes présentées précédemment, est d'approcher un optimum local en modifiant successivement une trajectoire, jusqu'à la stabilisation du coût associé. L'algorithme est initialisé avec une première trajectoire et la commande correspondante, généralement la commande nulle. Puis il calcule une nouvelle trajectoire, en deux étapes :

1. le calcul d'un modèle quadratique approché de la trajectoire courante, pour déterminer dans quelles directions la modifier pour diminuer la fonction coût ;
2. la modification de la trajectoire en calculant une nouvelle séquence de commande pour se rapprocher de l'optimum, cette nouvelle commande correspond en effet à l'annulation des dérivées du modèle quadratique.

Avantages et inconvénients

L'algorithme DDP prend en compte les dérivées secondes de la fonction coût, ce qui peut s'avérer décisif en terme de convergence et de stabilité lorsqu'on étudie un système non-linéaire complexe. Cependant, pour ces mêmes systèmes, l'optimum recherché peut être local, et la convergence de l'algorithme dépend donc énormément de la trajectoire initiale donnée en entrée (Tassa *et al*. (2012a)). De plus, la complexité de cet algorithme est en $O(T^3 n^3)$ à chaque itération, si l'on note n la taille du vecteur d'état x, ce qui augmente très rapidement lorsque n augmente.
On peut néanmoins accélerer le processus de deux manières :

1. généralement, on peut donc se limiter à une itération car la première itération permet d'approcher correctement l'optimum ;
2. l'hyptohèse de Gauss-Newton permet de négliger le second ordre des dérivées de la fonction d'évolution (Krejić *et al*. (2000)), et donc de diminuer le nombre de calculs. Notons toutefois que cette hypothèse n'est pas admissible pour tous les systèmes : pour les plus complexes, les dérivées primaires ne suffisent pas pour assurer la stabilité, on perd trop d'informations quant à l'évolution du système.

2.3 Systèmes à impédance variable

2.3.1 A l'origine des systèmes à impédance variable

Pour construire des robots, on privilégie les moteurs à courant continu (Mansard (2013)) pour des questions de simplicité. Cependant, ces moteurs donnent des vitesses de rotation en sortie très élevées, et de faibles couples. On utilise donc des réducteurs pour augmenter les couples et diminuer les vitesses. On peut ainsi réduire la taille des robots et les sources d'énergie. Toutefois, les réducteurs compliquent la commande du robot. En effet, beaucoup de frottements sont en jeu dans les réducteurs : on ne peut donc pas modéliser correctement la liaison. Le problème devient vraiment critique lorsqu'on souhaite que le robot interagisse avec son environnement : on utilise en effet préférentiellement une commande en force pour les interactions, or on ne peut pas savoir précisément le lien entre les couples d'entrée et de sortie du réducteur.
Dans l'optique de résoudre ce problème, on peut utiliser des actionneurs hydrauliques (Hunter *et al*. (1991)), le fluide sous pression permet de produire un mouvement de rotation ou de translation. Ces actionneurs permettent de produire un mouvement extrêmement rigide puisque l'huile est quasi incompressible, et de développer des forces et couples plus élevés que les actionneurs électriques. Mais en définitive, ils sont plus difficiles à contrôler car hautement non-linéaires, et d'un point de vue pratique, les risques de fuite d'huile ou d'usure limitent leur emploi en robotique humanoïde.
Une autre alternative réside dans les actionneurs pneumatiques (Tassa *et al*. (2013a)). Le principe est le même que précédemment, à ceci près qu'on n'utilise plus un fluide sous pression, mais de l'air comprimé. On perd alors la rigidité du mouvement, car l'air pouvant être plus ou moins comprimé sous l'effet de la charge extérieure, on ajoute de la flexibilité au système. Mais ceci est plutôt bénéfique pour des tâches incluant un contact : le contact sera plus stable et le système pneumatique peut même servir de capteur de force (Mansard (2013)). Ils sont néanmoins beaucoup plus lents que les actionneurs électriques et concrètement, les effets thermodynamiques ou les risques de résonnances restreignent leur utilisation (Robinson (2000)).

2.3.2 Actionneurs avec ressort en série

Pouvoir mesurer la charge extérieure permet d'avoir un feedback sur le système de contrôle et donc d'améliorer sa stabilité et sa convergence. Traditionnellement, sur les systèmes électriques et hydrauliques, on utilise des capteurs les plus rigides possibles : toute l'énergie est ainsi transmise sans stockage d'énergie au niveau du capteur (Robinson (2000)). Mais ceci nécessite de garder des gains de contrôle faibles et donc mène à de mauvaises performances, notamment en cas de chocs ou de frottements.
Pour pallier à ces problèmes, il est possible de placer un ressort en sortie d'un moteur électrique : on obtient un *Actionneur avec Ressort en Série* ou *Series Elastic Actuator* (SEA) en anglais. Via

l'élongation du ressort, il est alors possible d'évaluer la force au niveau de la liaison. La différence majeure par rapport aux systèmes rigides avec capteur est de pouvoir stocker de l'énergie dans le ressort, et l'un des avantages est de pouvoir mieux résister aux chocs (Pratt et Williamson (1995)). Pour plus de détails sur les gains de contrôle et la bande passante des SEA, on pourra se référer au travail de Robinson (Robinson (2000)).
Les limites de tels actionneurs sont inhérentes à leurs avantages : la flexibilité. En effet, si l'on souhaite réaliser un mouvement rapidement et avec précision, la flexibilité devient problématique et des actionneurs purement rigides sont bien plus efficaces. L'idéal serait donc de pouvoir changer la rigidité du système en fonction de la tâche réalisée par le robot.

2.3.3 Actionneurs à impédance variable

Présentation

Les actionneurs à impédance variable ou *Variable Impedance Actuators* (VIA) sont des actionneurs dont on peut moduler la rigidité. Comme attendu avec un système flexible, l'un des avantages majeurs est de pouvoir concevoir grâce à ces systèmes des robots évoluant dans des environnements et des situations inconnus : il peut y avoir choc sans dommage grâce à l'amortissement des ressorts. Toutefois, ces robots seraient tout aussi efficaces dans la réalisation de tâches minutieuses : il suffit de choisir une rigidité très élevée.
La figure (2.2) illustre une exploitation possible de l'impédance variable au cours d'une tâche (Tonietti *et al*. (2005)). La rigidité de l'actionneur diminue, le temps que l'actionneur atteigne sa vitesse en régime permanent : le mouvement est suffisamment lent pour ne pas être considéré comme dangereux. Puis la vitesse de l'actionneur est maximale et stable, et sa rigidité très faible : s'il y a collision lors de cette phase, la flexibilité de l'actionneur amortira le choc et il n'y aura pas ou peu de dommage. Enfin, la vitesse diminue et la rigidité augmente à nouveau : l'actionneur a atteint la position désirée et lors de cette étape, qui requiert une certaine précision, et donc une plus grande rigidité. Mais tout comme lors de la première phase, les risques sont minimes.

Ces actionneurs permettent également d'emmagasiner de l'énergie, puis de la réutiliser plus tard pour par exemple, augmenter la puissance en sortie. Il y a ainsi transfert d'une énergie cinétique en énergie potentielle, avant de la restituer sous forme d'énergie cinétique. La figure (2.3) illustre ce principe : une balle est lancée sur le robot et l'on souhaite qu'il la renvoie à son tour. Pour une telle tâche, un actionneur rigide arrêterait la balle lors de sa réception puis la renverrait comme il enverrait une balle au repos. L'énergie communiquée à la balle viendrait donc entièrement de l'actionneur. Au contraire, un actionneur à impédance variable sera capable de récupérer l'énergie cinétique de la balle avant réception, de la stocker grâce au ressort puis de la restituer

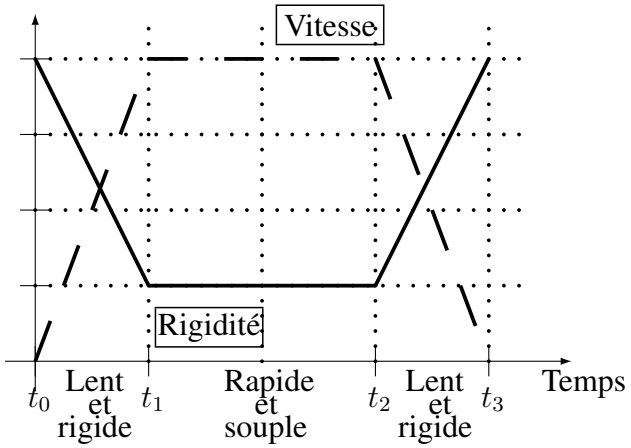


Figure 2.2 : **Stratégie d'utilisation de l'impédance variable :** Evolutions de l'impédance (trait plein) et de la vitesse (pointillés) d'un VIA lors d'une trajectoire d'un point précis à un autre

à la balle pour en augmenter sa vitesse lors du renvoi. Cette faculté permet donc d'améliorer les performances d'un robot lorsqu'il y a interaction, par exemple, un robot qui saute sur place fera des bonds plus élevés avec un VIA qu'avec un SEA (Vanderborght *et al*. (2009)).

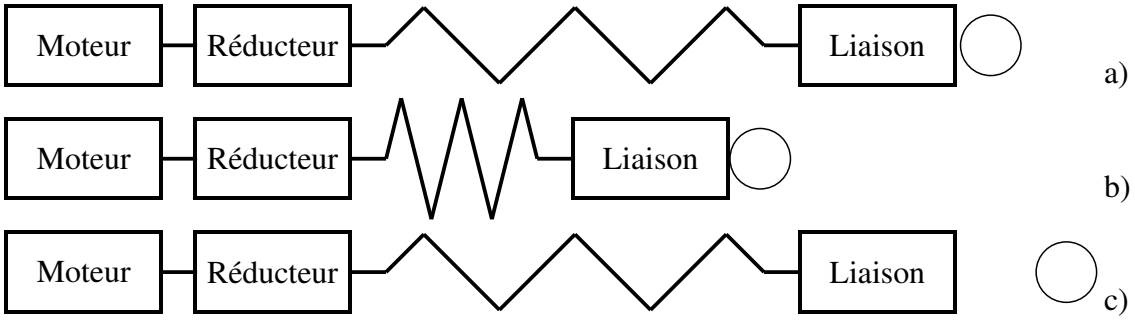


Figure 2.3 : **Exploitation de l'impédance variable lors d'un contact :** a) Une balle entre en contact avec la liaison b) Le ressort se comprime sous le choc et stocke de l'énergie c) L'impédance est utilisée activement pour renvoyer au loin la balle : le ressort s'étire et retransmet l'énergie emmagasinée.

En conclusion, tout comme pour les systèmes biologiques dont ils sont fortement inspirés, les VIA présentent une excellente résistance aux chocs, ils sont également plus sécuritaires pour les manipulateurs que des systèmes rigides. Enfin, ils permettent de stocker de l'énergie qui sera réinvestie par la suite (Ham *et al*. (2009)) et donc d'optimiser la consommation d'énergie.

Différents modèles de VIA

Il existe une grande variété d'actionneurs à impédance variable selon la stratégie adoptée pour ajouter l'impédance variable. Nous n'en présentons que quelques uns ici : les actionneurs biomimétiques, et les deux actionneurs, dont le fonctionnement est relativement simple et que nous avons étudiés au cours de nos travaux

Certains s'inspirent directement des systèmes biomécaniques de muscles antagonistes : les actionneurs antagonistes simples (Martinez-Villalpando et Herr (2009)). Deux ressorts sont placés de chaque côté du bras de sortie et leur élongation peut être ajustée aux extrémités (Fig 2.4.a). Ainsi pour augmenter la raideur, il suffit de comprimer les ressorts, et de les étirer pour la diminuer. Le contrôle des ressorts étant dissocié, on peut également varier la position du bras de sortie via ces deux ressorts.

D'autres actionneurs ne s'inspirent pas directement des systèmes biomécaniques. C'est le cas des deux que nous présentons ci-après. Le premier, *Actuator with Adjustable Stiffness* (AwAS) est sans doute le modèle le plus intuitif pour créer une impédance variable (Fig 2.4.b) (Jafari *et al*. (2010)). En effet, le bras de sortie est en rotation autour du bras d'entrée, un bras intermédiaire est ajouté : il est lié au bras de sortie par un ressort que l'on peut déplacer le long de ces deux bras. Le modèle AwAS possède des équations relativement simple, mais sa plage d'impédance est relativement limitée puisqu'elle ne dépend que de la course du ressort.

Un autre type de VIA est le *Mechanically Adjustable Compliance and Controllable Equilibrium Position Actuator* (MACCEPA) (Van Ham *et al*. (2007)). Cet actionneur ne comporte lui aussi qu'un seul ressort liant le bras de sortie à un bras intermédiaire, (Fig 2.4.c). La longueur de ce ressort peut être ajustée le long du bras intermédiaire afin de modifier l'impédance du système.

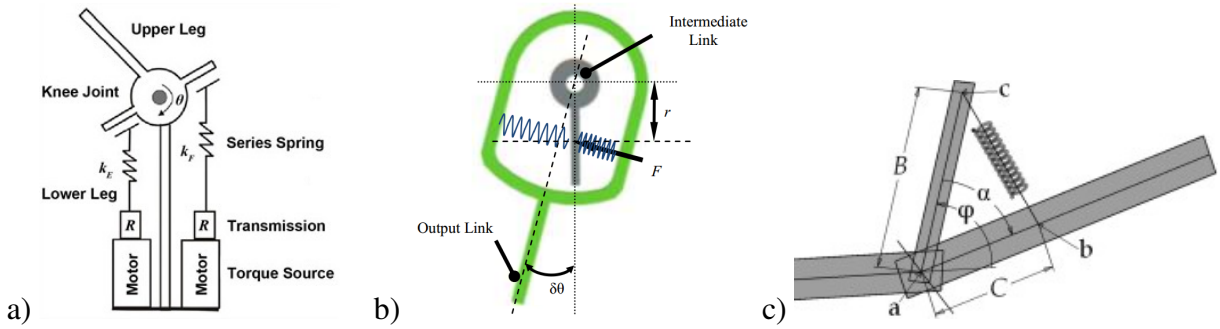


Figure 2.4 : **Modèles de VIA :** a) Actionneur antagoniste (Martinez-Villalpando et Herr (2009)), b) AwAS (Jafari *et al*. (2011)), c) MACCEPA (Van Ham *et al*. (2007))

Contrôle des VIA

L'enjeu actuel sur les VIA réside dans leur contrôle. En effet, l'impédance variable est très prometteuse en terme de performances comme nous l'avons vu précédemment. Mais elle ajoute également une variable à contrôler lors de la planification de trajectoire.
On a vu précédemment quelle stratégie pouvait être adoptée pour contrôler un robot possédant des VIA de façon à limiter les risques pour le robot et son entourage.Toutefois, l'impédance variable est délicate à prendre en compte dans le calcul de la commande, notamment parce qu'il est difficile de la mesurer avec certitude en temps réel lorsqu'elle varie (Serio *et al*. (2011)).
La problématique de recherche peut ainsi être reformulée : selon quelle commande contrôler l'impédance de façon à améliorer les performances de l'actionneur ?
Il est possible de découpler la commande de la position et de l'impédance, en ne contrôlant que l'impédance, on pourrait concrètement décider à quels moments de la trajectoire il est préférable d'être flexible ou rigide. Pour décider de cela, le plus simple est de suivre un profil donné d'impédance optimale. On peut obtenir un tel profil en copiant le mouvement humain correspondant à la tâche désiré, par exemple plier le bras (Howard *et al*. (2013)). Ceci semble logique dans la mesure où les VIA ont été développés dans l'optique de copier le fonctionnement biomécanique particulièrement efficace. Mais cette approche n'est pas toujours possible, notamment parce que cela limiterait les utilisations des VIA à des robots humanoïdes, et d'autre part car le système musculosquelettique ne fait pas face aux mêmes problématiques qu'un robot.
L'impédance peut être utilisée pour améliorer les performances des actionneurs, il serait donc judicieux de la prendre en compte dans le calcul global de la commande, c'est-à-dire sans découpler position et impédance. On ne suivrait pas alors de profil arbitraire d'impédance, mais on chercherait bien à chaque instant laquelle permet de minimiser la fonction coût du problème optimal. Ceci est le coeur même du travail de recherche présenté dans cette maîtrise : le contrôle non découplé des actionneurs à impédance variable.

CHAPITRE 3

Modélisation et contrôle d'un actionneur à impédance fixe

Nous présentons dans ce chapitre le travail réalisé en amont des deux articles qui suivent, ainsi que leur synthèse. L'objectif a été de modéliser un actionneur à impédance variable, puis de tester différents algorithmes de contrôle à impédance fixe, pour d'une part se familiariser avec le sujet et d'autre part déterminer sur quels critères nous devions en sélectionner un pour le contrôle à impédance variable.

3.1 Modélisation d'un actionneur à impédance variable

Nous avons choisi le modèle *Actuator with Adjustable Stiffness* (AwAS) pour les travaux préliminaires. En effet, cet actionneur est relativement simple du point de vue mécanique et son fonctionnement est assez intuitif. Nous rappelons que cet actionneur est composé de deux bras A et B en rotation, reliés par un ressort de position variable (Fig 3.1). Un moteur engendre la rotation du bras A, et ce dernier est retenu dans son mouvement par le bras B, via le ressort les liant.

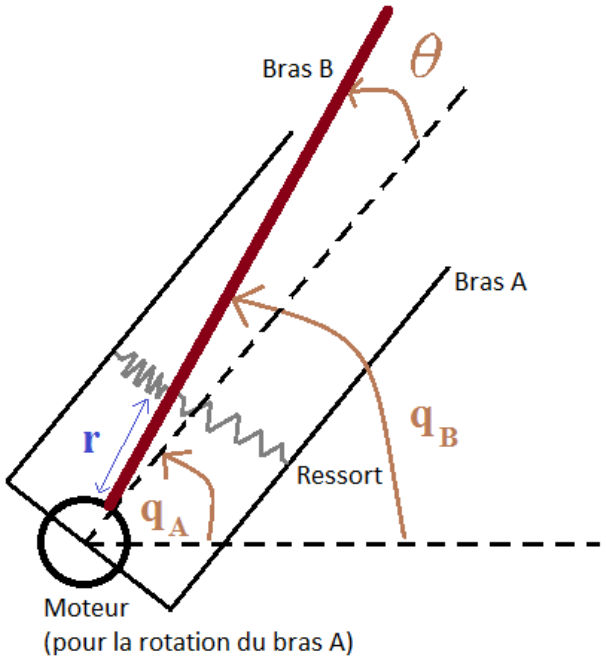


Figure 3.1 Schéma de l'actionneur AwAS utilisé

3.1.1 Notations

Variables du système	
q_A	position angulaire du bras A
q_B	position angulaire du bras B
$\theta = q_A - q_B$	angle formé par les deux bras A et B
r	position du ressort le long des bras
Paramètres du système	
$k = 10\,000\,Nm^{-1}$	raideur du ressort
$L = 0.2\,m$	longueur des bras
$S = 2.5 * 10^{-3}\,m^2$	section des bras
$\rho = 2\,700\,kg\,m^{-3}$	la masse volumique du matériau : aluminium
Commandes du système	
$\alpha = \ddot{q_A}$	accélération angulaire du bras A
$\beta = \dot{r}$	vitesse de translation du ressort

Figure 3.2 **Notations utilisées pour modéliser l'actionneur**

3.1.2 Equation d'évolution du système

On cherche l'équation différentielle régissant la dynamique du système. On suppose que la gravité est compensée, que les frottements sont négligeables et que les bras A et B sont rigides. On néglige également la masse du ressort et l'on considère que les bras A et B sont rigoureusement identiques en terme de longueur, section et matériau. On a alors, d'après le thérorème du moment cinétique :

$$\ddot{\theta}(t) + \frac{3kr(t)^2}{\rho SL^3} sin(\theta(t)) + \alpha(t) = 0 \tag{3.1}$$

Dans ce chapitre, nous supposons que les bras A et B de l'actionneur s'écartent très peu l'un de l'autre, on peut alors faire la simplification suivante : $sin(\theta) = \theta$.
D'autre part, nous supposons que le ressort est fixe dans un premier temps : $r(t) = r$. Ces hypothèses nous permettent de simplifier le problème de contrôle optimal et donc sa résolution, puisque l'équation différentielle régissant la dynamique devient alors une équation différentielle linéaire. Par ailleurs, nous pouvons ainsi nous familiariser progressivement avec le comportement de l'actionneur.
L'équation différentielle précédente peut alors être simplifiée :

$$\ddot{\theta}(t) + \omega\theta(t) + \alpha(t) = 0 \tag{3.2}$$

avec

$$\omega = \frac{3kr^2}{\rho S L^3}. \tag{3.3}$$

3.2 Problème de contrôle optimal

3.2.1 Formulation du problème

On souhaite déplacer le bras A à une position donnée q_{cible} en un temps T, ce qui se traduit sous la forme d'une contrainte :

$$q_A(T) = q_{cible} \quad ; \quad \dot{q_A}(T) = 0 \tag{3.4}$$

On définit alors les vecteurs d'état $x(t)$ et de contrôle $u(t)$:

$$x(t) = \begin{pmatrix} \theta(t) \\ \dot{\theta}(t) \\ q_A(t) - q_{cible} \\ \dot{q_A}(t) \end{pmatrix} \quad ; \quad u = \Big(\alpha(t)\Big) \tag{3.5}$$

L'évolution du système (3.2) peut alors être réécrite sous forme matricielle :

$$\dot{x}(t) = Ax(t) + Bu(t) \tag{3.6}$$

où :

$$A = \begin{pmatrix} 0 & 1 & 0 & 0 \\ -\omega & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \end{pmatrix} \quad ; \quad B = \begin{pmatrix} 0 \\ -1 \\ 0 \\ 1 \end{pmatrix} \tag{3.7}$$

On désire minimiser les coûts liés à la commande au cours de la trajectoire, pour limiter les dépenses énergétiques et éviter toute surchauffe des moteurs. On définit alors une fonction quadratique de coût, toujours positive $L_u(t)$:

$$L_u(t) = u(t)^T R(t) u(t) \tag{3.8}$$

où $R(t)$ est le poids de la fonction coût associé à la commande : c'est une matrice positive, carrée,

et ici de taille 1 (i.e. un scalaire), puisque le vecteur $u(t)$ est de taille 1.
Pour renforcer la contrainte finale (3.4), on va également définir une fonction de coût $L_x(t)$ liée à la position du levier A, qui vise à rapprocher rapidement le bras A de son objectif :

$$L_x(t) = x(t)^T Q(t) x(t) \tag{3.9}$$

avec $Q(t)$, la matrice de poids de cette fonction coût, diagonale et positive.

Le problème optimal peut alors être formulé comme la recherche de la commande $u(t)$ qui, en respectant la dynamique (3.2), minimise le coût suivant :

$$\min_{u(t)} \int_0^T L_x(t) + L_u(t) dt \tag{3.10}$$

3.2.2 Discrétisation du problème

Au vu de notre revue de littérature sur le controle optimal, nous souhaitons tester la méthode de la collocation pour résoudre ce problème d'optimalité. En effet, résoudre un problème semi-infini comme dans le cas de la méthode par tir unique et celle par tir multiple, semble délicat en première approche. Une discrétisation complète du problème est préférable dans un premier temps pour prendre en main les différents outils.

Nous discrétisons notre intervalle de temps $[0; T]$ en n intervalles de longueur dt : $t_i = i * dt$. La trajectoire et la commande ne sont plus des fonctions continues mais des vecteurs : $(x_i)_{0 \leqslant i \leqslant n}$ et $(u_i)_{0 \leqslant i \leqslant n-1}$ vérifiant :

$$x_{i+1} = \tilde{A} x_i + \tilde{B} u_i \tag{3.11}$$

$$\tilde{A} = Id + Adt \tag{3.12}$$

$$\tilde{B} = Bdt \tag{3.13}$$

$$\tag{3.14}$$

Le problème optimal discrétisé est alors :

$$\min_{u(t)} \sum_{i=1}^{n-1} (x_i^T Q_i x_i + u_i^T R_i u_i) + x_n^T Q_n x_n \tag{3.15}$$

3.2.3 Méthode de résolution

Pour calculer la commande optimale de cet actionneur, nous allons suivre le schéma représenté ci-dessous (Fig 3.3).

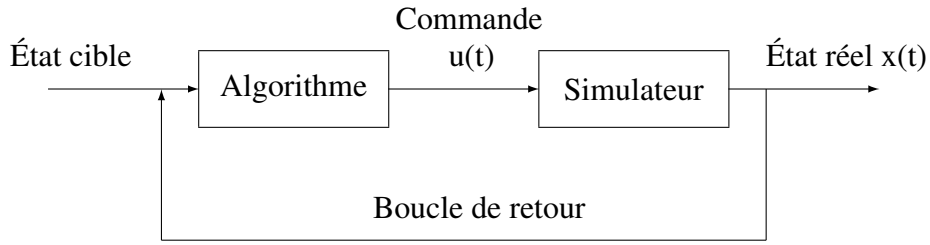


Figure 3.3 **Principe de résolution**

Un algorithme va calculer la commande optimale $u(t)$ à partir des équations précédentes, de l'état réel du système $x(t)$ et de l'état cible que l'on désire atteindre. Cette commande est ensuite envoyée à un simulateur, qui va calculer le nouvel état du système. Une boucle de retour permet de réinjecter cet état réel dans l'algorithme.
Un nouveau paramètre entre alors en jeu : la taille de la fenêtre de prévision, sur laquelle on calcule la commande que l'on fait entièrement jouer au simulateur avant de recalculer une nouvelle commande. Ce paramètre est très important pour la stabilité de la solution. En effet, l'algorithme utilise la version linéarisée de la dynamique du problème et non les équations réelles, à la différence du simulateur. Si l'on fait jouer une commande complète sans boucle de retour, nos trajectoires vont alors diverger, puisque la commande est calculée pour un système légèrement différent du système réel. Si on recalcule la commande à une grande fréquence (fenêtre de prévision petite), l'algorithme va utiliser l'état réel du système plus souvent pour calculer la commande, par ailleurs celle-ci sera calculée pour un court laps de temps durant lequel le système linéarisé est encore proche du système réel. Il n'y aura donc pas de divergence. Toutefois, plus on réduit la fenêtre de prévision, plus on augmente le temps de calcul, il faut donc trouver un juste milieu.

3.3 Choix des algorithmes

Nous présentons ici les deux algorithmes que nous avons testés pour calculer la commande optimale d'un actionneur à impédance variable, dans le cas d'une impédance fixe. Les simulations ont été réalisées avec le logiciel de calcul MATLAB ^{TM}.

3.3.1 Méthode de la collocation

Dans un premier temps, nous avons testé la méthode de la collocation, décrite dans la revue de littérature (Ascher *et al*. (1988) ; Diehl *et al*. (2006)). Pour cela, nous avons utilisé des polynômes de degré 1, c'est-à-dire des fonctions affines. Les résultats pour atteindre une position donnée de 0.3 radian avec le bras A et avec un pas de temps de 1ms, sont représentés à la figure 3.4.

Le bras A converge rapidement vers la position désirée, en 300 ms. Cependant le bras B oscille énormément au cours de ce mouvement, et ce, malgré nos efforts pour ajuster la valeur des poids dans la fonction coût. De plus, les temps de calcul sont prohibitifs, de l'ordre de l'heure. Il n'est donc pas envisageable de conserver cette méthode sans vérifier au préalable qu'aucune autre ne soit plus rapide.

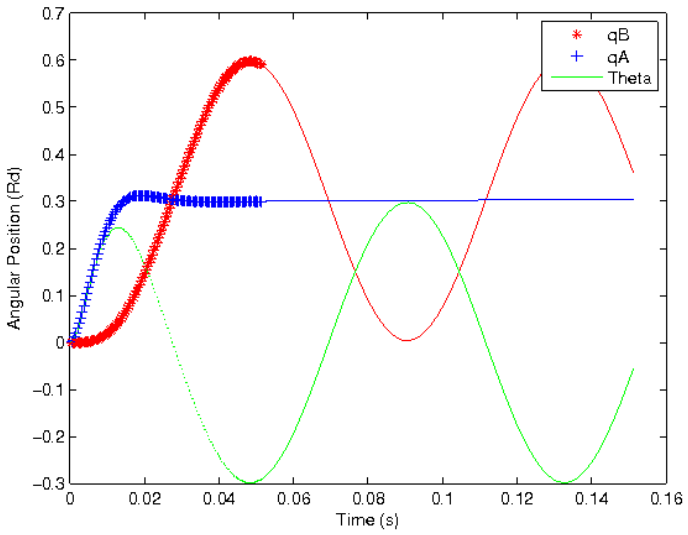
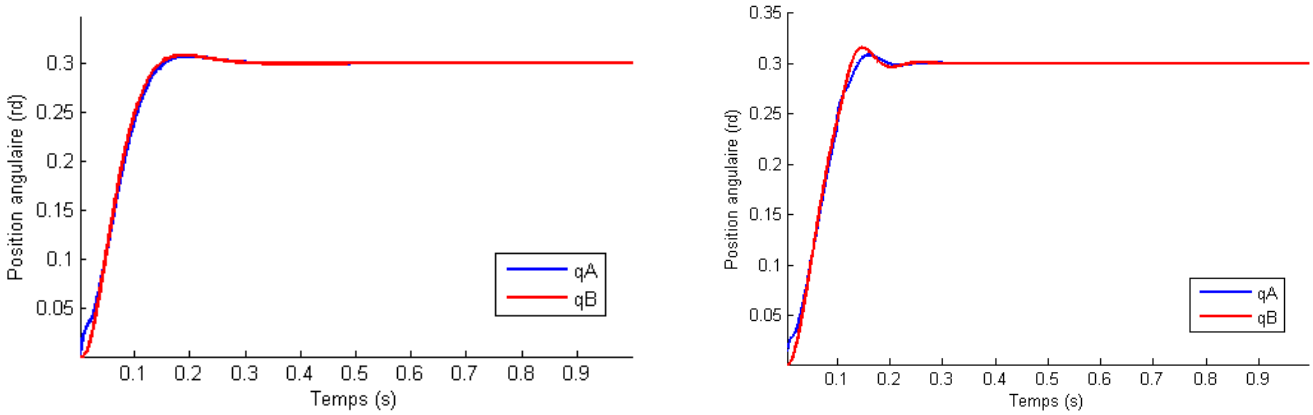


Figure 3.4 **Commande par collocation d'un actionneur AwAS :** Trajectoires des leviers A (bleu) et B (rouge) pour atteindre la position cible de 0.3 rd, en vert, l'évolution de θ au cours du mouvement

3.3.2 Méthode LQR

Dans un second temps, nous avons testé la méthode LQR (Blanchet (2007)), à horizons fini et infini, puisque l'évolution de notre système est une fonction linéaire en l'état et en la commande et que notre fonction de coût est une fonction quadratique.
Nous utilisons ici les mêmes fonctions de coût que prédécemment, un pas de temps toujours de 1 ms et la commande est recalculée à chaque instant.
A horizons fini (Fig 3.5.a) et infini (Fig 3.5.b), la position désirée est rapidement atteinte, également en 300 ms et avec un dépassement comparable. Notons néanmoins, que le dépassement est légèrement plus élevé dans la commande à horizon infini : 0.32 rd, contre 0.31 rd à horizon fini.

Les temps de calcul avec la méthode LQR sont bien plus intéressants : de l'ordre d'une dizaine de secondes et non plus d'une heure. C'est donc cette méthode que nous retenons pour notre étude pour le moment. De plus, nous préférerons travailler à horizon infini. En effet, les résultats sont



a) Horizon fini b) Horizon infini

Figure 3.5 **Commande LQR d'un actionneur AwAS :** Trajectoires des leviers A (rouge) et B (vert) pour atteindre la position cible de 0.3 rd

comparables à ceux obtenus en horizon fini, le dépassement n'étant pas significativement plus élevé, et l'algorithme est plus simple à mettre en oeuvre.

3.3.3 Stabilité et robustesse de LQR

Pour pouvoir recalculer efficacement la commande à chaque instant, un système réel a besoin des valeurs exactes des différentes positions et forces. On utilise pour cela des capteurs, or ils peuvent être sujets aux bruits. De plus, les résultats à l'issue d'une commande ne sont pas exactement ceux calculés en simulation. Il est donc important de tester la robustesse de notre algorithme en cas de tels incidents.

Nous avons ajouté un bruit blanc de l'ordre de 10% sur le vecteur d'état retourné à l'algorithme à chaque instant. La convergence était toujours assurée, avec de faibles fluctuations, inférieures à 1 %.

Un autre test a consisté à ajouter un bruit blanc de l'ordre de 50 % sur la valeur supposée de la rigidité du ressort : les résultats ne montrent aucune différence significative avec ou sans ce bruit. Nous étudions par ailleurs un actionneur à impédance variable. Or jusqu'à présent, celle-ci est restée fixe. Nous faisons alors varier la rigidité du ressort de 50 à 100 % de sa valeur nominale au cours de la trajectoire, en utilisant la vraie valeur de la rigidité à chaque instant dans nos calculs. L'algorithme n'a là encore aucune difficulté à converger aussi rapidement qu'avec une imépédance fixe.

Pour terminer nos essais, nous regroupons les trois phénomènes précédents (Fig 3.6). Les résultats sont satisfaisants vis à vis des grandes et nombreuses perturbations que nous simulons : on a bien convergence rapide vers la position désirée, avec des fluctuations inférieures à 1 %.

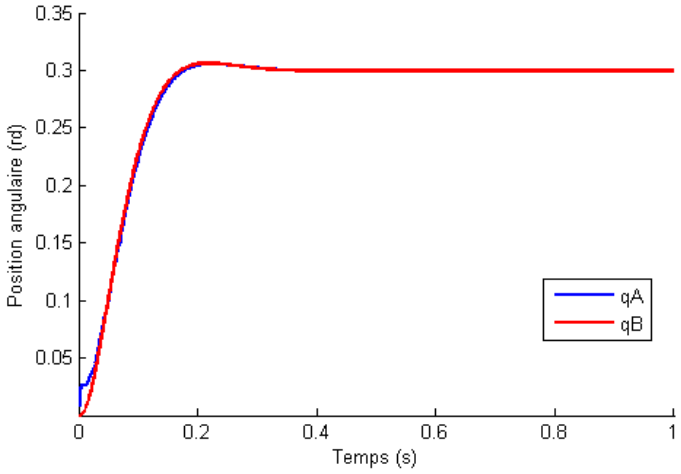


Figure 3.6 **Commande LQR à Horizon infini d'un actionneur AwAS en cas de perturbations**

En conclusion, l'algorithme LQR est stable et robuste, même en cas d'impédance variable. Toutefois, d'aussi bons résultats sont à rapprocher du fait que nous recalculions la commande à chaque instant : le système n'a pas réellement le temps de s'écarter de la trajectoire optimale.
Par ailleurs, notons que nous avons fait jouer ici un profil donné d'impédance et n'avons pas cherché à l'optimiser pour améliorer notre trajectoire. Néanmoins, on note une amélioration de la trajectoire par rapport à l'impédance fixe : le dépassement est diminué.

3.3.4 Limites de LQR

Les limites de cette méthode sont liées à son essence même : elle ne fonctionne que tant que le système évolue linéairement. D'une part, en cas d'impédance variable, notre équation différentielle n'est plus linéaire, au mieux peut elle être approchée par un polynôme d'ordre 3 en $r^2\theta$. D'autre part, nous n'avons ici effectué que de la commande en position. Or, en mécatronique, pour des raisons de sécurité et de précision, on est souvent amené à utiliser de la commande en force : au lieu de viser une position donnée q_{cible} on cherche à suivre un profil de couple en sortie du système. Ce couple s'exprime lui aussi comme un polynôme de degré 3 en $r^2\theta$. La fonction coût ne serait donc plus quadratique.
Remarquons qu'aucune linéarisation ne serait pertinente ici, car cela reviendrait à considérer que le ressort ne bouge pas et donc l'impédance ne varie pas, alors que c'est là que réside tout l'intérêt des VIA.
Après tout ce travail de prise en main, maintenant que nous avons pu voir évoluer le système et comprendre plus finement les points délicats que sa commande soulève, il est temps de chercher à contrôler l'actionneur pleinement, c'est-à-dire en contrôlant également son impédance. Pour cela,

il est nécessaire d'utiliser un algorithme qui va au-delà de la linéarisation, et qui tient donc au moins compte des dérivées secondes dans son approximation du système. D'après notre revue de littérature, nous nous sommes donc intéressés à l'algorithme DDP. L'ensemble de notre travail avec cet algorithme est présenté dans ce mémoire sous forme d'articles de conférence en anglais. Pour faciliter la lecture, nous proposons ci-dessous au lecteur une brève synthèse de ces articles en français.

3.4 Synthèse du premier article : *A two-stage suboptimal approximation for variable compliance and torque control*

Les actionneurs à impédance variable sont des dispositifs prometteurs en mécatroniques. En effet, ils présentent à la fois la précision des systèmes rigides évoluant dans des espaces libres, et les avantages des systèmes élastiques lorsqu'il y a interaction. La commande de ces systèmes demeure néanmoins un véritable défi. D'une part, ils sont non-linéaires et d'autre part, devoir contrôler l'impédance des actionneurs double les variables de commande à calculer.
Dans cet article, on propose une stratégie de contrôle originale pour calculer la commande d'un mouvement de corps complet d'un robot complexe, typiquement, un robot humanoïde, dont les actionneurs sont à impédance variable. Dans un premier temps, en s'appuyant sur l'espace de contrôle opérationnel, on propose de calculer les couples et les impédances des liaisons pour répondre aux exigences liées à la tâche considérée. Puis on présente un contrôleur non-linéaire de modèle prédictif qui permet de suivre à une fréquence élevée les couples et impédances de référence calculées précédemment, et ce pour chaque liaison séparément. Ce contrôleur, proche de la descente de Newton, utilise notamment jusqu'aux dérivées secondes du modèle pour chercher la trajectoire optimale et assure donc une plus grande stabilité malgré la non-linéarité des équations. On teste ensuite ce contrôleur avec deux modèles de VIA distincts : sa précision et son efficacité sont validées sur des trajectoires données de couple à suivre, et sa robustesse est validée en cas de perturbations survenant en régime permanent. Notre méthode a ensuite été validée pour un saut de robot humanoïde. En effet, les actionneurs à impédance variable sont particulièrement étudiés en robotique humanoïde pour pouvoir réaliser des tâches explosives.

3.5 Synthèse du second article : *From Inverse Kinematics to Optimal Control*

Le contrôle optimal numérique est un domaine ayant pour but de calculer une commande permettant d'approcher une trajectoire optimale, et ce grâce à des algorithmes numériques itératifs. Cette approche est particulièrement prometteuse pour contrôler des systèmes dynamiques complexes, pour lesquels la linéarisation instantanée n'est pas satisfaisante. Outre les problèmes liés aux coûts de calcul, ces méthodes soulèvent d'autres interrogations : sont-elles stables ? Robustes ?

Comment interpréter physiquement la solution obtenue ?
Dans cet article, on propose d'optimiser un de ces contrôleurs, le DDP, présenté dans l'article précédent : il s'agit donc d'un apport technique au travail déjà réalisé. Cette reformulation, et simplification, est réalisée sous l'hypothèse de Gauss-Newton, classique en robotique, et qui revient à négliger certains termes dans les dérivées secondes de la fonction coût. On exhibe ainsi un algorithme plus efficace et de complexité moindre que le DDP initial. Par ailleurs, on peut comparer cette réécriture avec la formulation classique de cinématique inverse. En particulier, la cinématique inverse peut être vue comme un cas particulier de ce nouvel algorithme lorsqu'on choisit une fenêtre de prévision nulle.

CHAPITRE 4

A two-stage suboptimal approximation for variable compliance and torque control

Article accepted to the *European Control Conference* in march 2014.

Auteurs :
Perle Geoffroy, École Polytechnique de Montréal, Montréal, Canada
Olivier Bordron, École Normale supérieure de Bretagne, Ker Lann, France
Nicolas Mansard, LAAS-CNRS, Univ. Toulouse, Toulouse, France
Olivier Stasse, LAAS-CNRS, Univ. Toulouse, Toulouse, France
Maxime Raison, Ecole Polytechnique de Montréal, Montréal, Canada
Tim Bretl, University of Illinois, Urbana-Champaign, USA

4.1 Introduction

This paper deals with the control of multiple body robot with compliant actuation. This paradigm of actuation originates from the series-elastic actuator (Pratt et Williamson (1995)). An example of complete series-elastic actuation is the Coman robot (Tsagarakis *et al*. (2011)). The compliance is a very positive property when the robot works in contact with its environment, but deeply complicates the control in particular when performing fast and accurate movements in free space. Variable-stiffness actuators are a nice trade-off, offering compliancy when in contact or before an impact but being stiff when accuracy is required (English et Russell (1999a)).

We are interested in the capability to generate and control complex movements, e.g. a humanoid robot grasping an object while keeping its balance (Mansard *et al*. (2007)) or several manipulator robots moving an object collaboratively. The brute-force solution is to model the movement to execute as an optimal motion problem, and use direct solution method to obtain an approximation of the optimum (Tassa *et al*. (2012b)). However, this solution is computationally expensive : it scales in the cube of the number of actuators. It is yet prohibitive for complex variable-stiffness robot that have two motors for each axis (Enoch *et al*. (2012)).

A classical solution to reduce the complexity is to consider only the instantaneous linearization of the system evolution (Whitney (1969)), resulting in inverse kinematics when only the geometry is considered (Siciliano et Slotine (1991)) or inverse dynamics when inertia and forces are considered. In particular, operational-space inverse dynamics, detailed below, proposes to compute the

output motor joint torques in function of references forces and acceleration in some specific operational spaces (Khatib (1987)). The interest is that, when the proper operational space is chosen, the reference motion is generally very easy to specify.

Two major problems limit the generalization of operation-space inverse dynamics as the classical motion generation and method. First, joint torque is not a direct input of the motors, and a complex non-linear output as soon as reductors are introduced between the motor and the joint. Reductors are indeed difficult to properly model, which prevents feedforward control, while feedback control requires both an expensive torque sensor and a very fast (and very expensive) electronics (Albu-Schaffer *et al*. (2007)). Second, there is no sense to instantaneously linearize a spring, since any force at the input of a spring would instantaneously produce zero motion at its output.

However, the spring actuation can be considered as an advantage to apply a reference torque on a real robot, by using the spring as a cheap force sensor. A first level compute the reference joint torques from the operational specification. The torques are then tracked by an optimal-motion solver for each joint, acting at higher frequency and taking care of the spring dynamics. Compared to the whole-body optimal solver described for example in (Enoch *et al*. (2012)), this solution decouples the space complexity (solved by the whole-body inverse dynamics) from the temporal complexity (solved by each actuator optimal controller). The decoupled solution can be seen as a suboptimal approximation to the whole-body optimal problem.

With respect to this strategy, the paper proposes three contributions. We propose a method to compute the reference joint compliance from operation compliance specification, based on the operational-space formulation (Section II). We then propose a generic method to efficiently compute the optimal control law at the joint level (Section III). This method can be directly applied to any possibly non-linear joint model and various cost function, for tracking the output acceleration, torque or stiffness as shown in Section IV. Finally, we propose an implementation of the optimal controller, coupled with the operational-space solver, to produce complex movements. In particular, we demonstrate a whole-body jump in simulation with a modified stiffness-variable humanoid in Section V.

4.2 From whole-body objectives to actuator references

We first recall in Section 4.2.1 the generic model of a free-floating robot in contact with a known, typically rigid, environment. From this model, it is classical to deduce a torque control law based on task specifications, which is also recalled in Section 4.2.2. Based on this model and this procedure, we propose a solution to deduce the needed whole-body compliance to meet some reference compliance expressed in given task spaces in Section 4.2.3.

4.2.1 Dynamic model

In general, the dynamics model of the robot whole body, considering all the bodies and the actuator to be stiff, is :

$$A(q)\ddot{q} + b(q,\dot{q}) = S^T\tau - \sum_{k=1}^{n_c} J_k(q)^T f_k \tag{4.1}$$

where A is the generalized inertia matrix, b is the dynamic drift (sum of Coriolis, centrifugal and gravity forces), τ are the joint torques (typically, the actuator output torques), S is the matrix that selects the actuated degrees of freedom (DOF), f_k are the n_c contact forces exerted by the robot on the environment at the contact points p_k and $J_k = \frac{\delta p_k}{\delta q}$ are the Jacobian of the contact points.

This equation represents the inverse dynamics : from a given joint acceleration given as a reference, it gives the forces that would lead to this acceleration. The three $\ddot{q}$, τ and $f_c = (f_1, ..., f_{n_c})$ are variable and evolve together. Very often in robotics, we want to impose a reference $\ddot{q}$ and search for a solution τ meeting this reference, while f_c is explicitly needed.

It is possible to get ride of f_c and then compute the needed joint torques if an interface model is given. For example, if assuming a rigid contact $J_k\ddot{q} + \dot{J}_k\dot{q} = 0$ [1], an acceleration-free relation can be obtained from(4.1) :

$$J_cA^{-1}\tau = J_cA^{-1}J_c^T f_c + b_c$$

where $J_c = (J_1, ..., J_{n_c})$ and $b_c = J_cA^{-1}b - \dot{J}_c\dot{q}$. When J_c is full row rank, f_c can be immediately deduced from τ. This leads to an equivalent dynamics in contact (Park et Khatib (2006)) :

$$A\ddot{q} + b_c = (SN_c)^T\tau \tag{4.2}$$

where $N_c = I - J_c^T(J_cA^{-1}J_c^T)^{-1}J_cA^{-1}$ is the projector in the null space of J alongside the direction A^{-1} and $b_c = N_c^Tb - (J_cA^{-1}J_c^T)^{-1}\dot{J}_c\dot{q}$. This equation still holds when J_c is not full row rank, even if slightly less intuitively obtained (Mansard (2012)).

In this last equation, the contact forces are considered as additional actuators to compensate the underactuation expressed by S^T. It is possible to separate the forces into support forces f_s, used as actuators, and the other forces, which would be typically used to accomplished a given force reference. In that case, the equation is written :

$$A\ddot{q} + b_S = (SN_S)^T\tau + \sum_{k=1}^{n_f} J_k f_k$$

1. The contact typically also imposes the forces to stay in the friction, which comes as an additional constraint expressed in f_c.

4.2.2 Operational-space inverse dynamics

For one task

The task-function (Samson *et al*. (1991)), or operational-space (Khatib (1987)), approach was proposed to enable the user to define the objectives of the motion to be executed in a dedicated task, or operational, space rather than directly in the configuration space. The task is defined by a vector function $e(q, \Omega) \in \mathbb{R}^m$ of the robot configuration q and the rest of the universe Ω, whose image space is called the task space and whose Jacobian with respect to the configuration is denoted $J = \frac{\partial e}{\partial q}$. In addition to the task function, the control in the task space is given as a reference vector field $\ddot{e}^*$ in the tangent to the task space. The task acceleration $\ddot{e}$ can be linked to the control input τ by multiplying (4.2) by JA^{-1} :

$$\ddot{e} = JA^{-1}(SN_S)^T\tau + \mu$$

where μ is the task drift that collects the non linear terms. τ is obtained by solving this equation in the least-square sense :

$$\tau = (JA^{-1}(SN_S)^T)^{\#}\left(\ddot{e}^* - \mu\right)$$

where $.^{\#}$ denotes any reflexive generalized inverse, typically the inverse weighted on the left by $SA^{-1}(SN_S)^T$ (Park (2006)).

For several tasks

This last form satisfies the task while minimizing the given norm, but the general solution is written :

$$\tau = (JA^{-1}(SN_S)^T)^{\#}\left(\ddot{e}^* - \mu\right) + P\tau_2$$

The second input τ_2 can be chosen arbitrarily ($\tau_2 = 0$ results in the least norm solution). This second input can be used to satisfy a secondary task, and recursively, any number of tasks set up in a hierarchy order. Denoting by e_1 ... e_p the hierarchy of tasks with J_i the task Jacobian, the general solution is :

$$\tau = \sum_i G^{\#}_{i|i-1}\left(\ddot{e_i}^* - \mu_{i|i-1}\right)$$

with $G_i = J_iA^{-1}(SN_S)^T$, $G_{i|i-1} = G_iP_{i-1}$, $P_0 = I$, $\forall i > 0, P_i = P_{i-1} - G^{\#}_{i|i-1}G_{i|i-1}$ and $\mu_{i|i-1}$ collects all the non linear terms. More details are given in (Saab *et al*. (2013)). In particular, the hierarchy can be implemented on an efficient manner and be coupled with an active-set search if some task references are given as bounds rather that equality.

4.2.3 Operational-space inverse compliance

Our objective in this section is to define a similar scheme to compute the configuration compliance, being given some reference compliance in dedicated task spaces. We first consider the fully actuated case, i.e. when SN_S is invertible (typically, a fixed manipulator whose only considered contact forces are at the interface with the ground).

Direct compliance model

We denote by Γ the compliance of the axes, supposed constant (linear springs). A variation of the torques then induces a variation of the axis position Δq given by

$$\Delta q = {}^{\Delta}\Gamma\tau$$

Γ can be seen both as a diagonal matrix (make explicit by the left exponent ${}^{\Delta}\Gamma$) or as a vector (denoted ${}^{\vee}\Gamma$).

Consider now a slight change of configuration Δq from the spring equilibrium. We search for the equivalent spring reflected at a given contact point. The change Δq leads both to a change of contact position and contact forces. The change of position is directly :

$$\Delta x = J\Delta q$$

The change of force is expressed from the steady-state force-to-torque equation $\tau = J^T f$. Plugging both equations, we get :

$$\Delta x = J\Delta q = J{}^{\Delta}\Gamma\tau = J{}^{\Delta}\Gamma J^T f$$

The apparent compliance can then be defined by :

$$\gamma = J{}^{\Delta}\Gamma J^T$$

where γ is a square symmetric matrix.

Quadratic solution

Now assume that we require a specific apparent compliance γ^*. The configuration compliance that best matches this reference can be expressed as the solution of the Frobenius problem :

$$\min_{{}^{\Delta}\Gamma\text{diagonal}} \|J{}^{\Delta}\Gamma J^T - \gamma^*\|_F$$

where $\|.\|_F$ denotes the Frobenius norm. This is a quadratic problem, since the square Frobenius norm is the sum of square of the matrix coefficients and since $J^{\Delta}\Gamma J^T$ is linear if Γ.

We denote by ${}^{\vee}(.)$ the matrix-to-vector operator stacking all the columns of the input matrix [2]. The Frobenius problem can be explicitly rewritten as a quadratic problem :

$$\min_{{}^{\vee}\Gamma} \| \mathscr{J} {}^{\vee}\Gamma - {}^{\vee}\gamma^* \|_2$$

where $\mathscr{J}$ is the unique matrix such that $\mathscr{J} {}^{\vee}\Gamma = {}^{\vee}(J^{\Delta}\Gamma J^T)$. This operator has the following properties :

$$\mathscr{J} {}^{\vee}\Gamma = J^{\Delta}\Gamma J^T \tag{4.3}$$

$$\mathscr{J}^T {}^{\vee}\gamma = {}^{\Delta}(J^T \gamma^* J) \tag{4.4}$$

where ${}^{\Delta}(.)$ denotes the matrix-to-vector operator that selects the diagonal elements of the input matrix. Finally, the square of $\mathscr{J}$ is the Hadamard square of the square of J :

$$\mathscr{J}^T \mathscr{J} = [\![J^T J]\!] \tag{4.5}$$

where $[\![A]\!] = A \circ A$ denotes the element-wise (Hadamard) square of A.

The solution to the unconstrained quadratic problem is given by the pseudo-inverse :

$${}^{\vee}\Gamma = \mathscr{J}^{+} {}^{\vee}\gamma^*$$

Since for any A, $A^{+} = (A^T A)^{+} A^T$ and using (4.5), this last form can be rewritten :

$${}^{\vee}\Gamma = [\![J^T J]\!]^{+} {}^{\Delta}(J^T \gamma^* J) \tag{4.6}$$

Hierarchical solution

Among all the solutions that fit at best in the least-norm-2 sense the reference γ^*, this last solution is the one of least norm-2 Γ. As usual for quadratic system, the redundancy can be made evident by expressing all the possible solutions using the projector on the null space of $\mathscr{J}$:

$${}^{\vee}\Gamma = [\![J^T J]\!]^{+} {}^{\Delta}(J^T \gamma^* J) + \mathscr{P} {}^{\vee}\Gamma_2$$

2. The operators ${}^{\vee}.$ is abusively applied to pass from the diagonal matrix ${}^{\Delta}\Gamma$ to the vector of the diagonal elements ${}^{\vee}\Gamma$, to keep the notations simple.

where $\mathscr{P} = I - [\![J^T J]\!]^+ [\![J^T J]\!] = ker[\![J^T J]\!]$ and ${}^{\vee}\Gamma_2$ is any configuration compliance that can be used to satisfy a second objective.

Assume now that several compliance objectives γ_1^*, ... , γ_p^* ordered in a hierarchy are given. Using the classical redundancy scheme detailed in (Siciliano et Slotine (1991)), the configuration compliance matching at best in the last-square sense the hierarchy of the γ^* is Γ_p obtained by the following iteration :

$$\begin{aligned} {}^{\vee}\Gamma_k = {}& {}^{\vee}\Gamma_{k-1} \\ & + (\mathscr{P}_{k-1}[\![J_k^T J_k]\!]\mathscr{P}_{k-1})^{+\,\Delta}\big(J_k^T(\gamma_k^* - J_k\,{}^{\Delta}\Gamma_{k-1}J_k^T)J_k\big) \end{aligned} \tag{4.7}$$

with $\mathscr{P}_k$ the projector onto the null space of the stacked k first Jacobian, that can be computed following the recurrence proposed in (Baerlocher (2001)).

Constrained system

All the development above holds for a fully-actuated unconstrained system, and by direct generalization to the case when $SN_S S^T$ is invertible. In the general case, the relation between a change of elastic torques $\tau = {}^{Delta}\Gamma\Delta q$ and contact force f is correlated with the support N_S :

$$(SN_S)^T\tau = J^T f$$

The displacement Δq cope with the rigid contact and thus is such that :

$$\Delta q = N_S \Delta q$$

since N_S is a projector onto the kernel of J_S. The compliance relation is then written :

$$\gamma = J^T(SN_S)^{T\,\Delta}\Gamma SN_S J$$

The same previous developments then still hold, with direct adaptations. The operational inverse compliance is then :

$$\begin{aligned} {}^{\vee}\Gamma_k = {}& {}^{\vee}\Gamma_{k-1} \\ & + (\mathscr{P}_{k-1|S}[\![J_{k|S}^T J_{k|S}]\!]\mathscr{P}_{k-1|S})^{+\,\Delta}\big(J_k^T(k_k^* - J_k\,{}^{\Delta}\Gamma_{k-1}J_k^T)J_k\big) \end{aligned}$$

with $J_{k|S} = SN_S J_k$ and $\mathscr{P}_{k|S}$ the projector onto the stacked $J_{k|S}$ Hadamard squares.

4.3 Model predictive control

In this section, we quickly recall the optimal control method that we used to drive the compliant actuators.

4.3.1 Principles and model

Consider a generic system, with state variable x and control variable u, defined by its discrete time evolution function :

$$x_{t+1} = f(x_t, u_t, t) \tag{4.8}$$

where f is the system evolution function and the time variable t is considered discrete. Optimal control aims at computing the control and state trajectory by minimizing a given cost function :

$$\min_{X,U} \sum_{t=0}^{T-1} l_t(x_t, u_t) + l_T(x_T)$$

where T is the preview interval length (fixed), $U = (u_0...u_{T-1})$, $X = (x_0, ..., x_T)$ are the control and state trajectories and l_t and l_T are the running and terminal costs functions. The optimal control problem is to be solved under the constraint that (4.8) is satisfied. In practice, the problem is solved for X only or U only, the other variable being deduced from the dynamic equation. The solution is said explicit when computing X and implicit when computing U. The optimal solution for a linear dynamics $x_{t+1} = F_x x_t + F_u u_t$ and a quadratic cost is given by Riccati equations as a linear-quadratic regulator (LQR).

Model predictive control (MPC) is an advance control technique to control a given system by optimizing its predicted evolution. It relies on the systematic evaluation of the optimal control of the system with respect to a reference cost function, while only the first few steps of the optimal trajectory are actually executed by the actuators before its complete re-evaluation. The main interest of MPC is the ability of dealing with non-linear systems whose instantaneous linearization is not meaningful.

4.3.2 Differential dynamic programming

The Differential Dynamic Programming (DDP) is an iterative algorithm to solve a non-linear continuous optimal control problem using implicit formulation (Tassa *et al*. (2012b)). It is nearly equivalent to the application of a Newton descent algorithm. As in the Newton descent, the main idea is to approach a local optimum by iteratively modifying a candidate solution until stabilization of the evaluation of the cost function. It starts with initial state and control trajectories, typically

obtained from the integration of the zero control if no other prior is available. The algorithm then iterates in two stages. It first computes an approximate quadratic model of the current candidate trajectory and compute the corresponding LQR (backward loop). Then the candidate is modified following the LQR (forward loop)

Bellman principle

We denote V_t the cost-to-go function, i.e. the cost function on the end of the preview window :

$$V_t(X_t, U_t) = \sum_{k=t}^{T-1} l_k(x_k, u_k) + l_T(x_T)$$

where $X_t = (x_t...x_T)$ and $U_t = (u_t...u_{T-1}$ are the trajectory tails. To simplify the presentation, we drop the t variable and denote the next quantity at $t+1$ by a prime : $V_{t+1} = V'$.

DDP relies on Bellman principle : the minimization on a global sequence can be separated in two minimization problems. It proceeds recursively from the back thanks to the following equation :

$$V^*(X, U) = \min_{x,u,X',U'} (l(x, u) + V'^*(X', U'))$$

Quadratic model

DDP recursively builds a quadratic model of V from the quadratic models of both l and V'^* :

$$\begin{aligned} V(x + \Delta x,u + \Delta u) = V(x, u) + V_x\Delta x + V_u\Delta u \\ + \frac{1}{2}\Delta x^T V_{xx}\Delta x + \Delta u^T V_{ux}\Delta x \\ + \frac{1}{2}\Delta u^T V_{uu}\Delta u + o(||\Delta x||^2 + ||\Delta u||^2) \end{aligned}$$

The quadratic model is defined by the quadratic coefficients :

$$V_x = l_x + V'_x\Delta x \tag{4.9}$$

$$V_u = l_u + V'_u\Delta u \tag{4.10}$$

$$V_{xx} = l_{xx} + \Delta x^T f_x^T V'_{xx} f_x\Delta x + \Delta x^T V'_x f_{xx}\Delta x \tag{4.11}$$

$$V_{ux} = l_{ux} + \Delta u^T f_u^T V'_{xx} f_x\Delta x + \Delta x^T V'_x f_{ux}\Delta u \tag{4.12}$$

$$V_{uu} = l_{uu} + \Delta u^T f_u^T V'_{xx} f_u\Delta u + \Delta u^T V'_x f_{uu}\Delta u \tag{4.13}$$

Backward pass

The optimum Δu can be computed for any Δx. It is obtained at the zero of the derivative of the quadratic model :

$$\Delta u^* = \lambda + \Lambda \Delta x \tag{4.14}$$

where $\lambda = V_{uu}^{-1} V_x$ and $\Lambda = V_{uu}^{-1} V_{ux}$ are the open-loop and close-loop gains (and can be linked to the LQR gains in the case where the quadratic model is exact).

From the optimal change Δu^*, the quadratic model of V^* can be computed :

$$V_x^* = V_x - \Lambda^T V_{uu} \lambda$$

$$V_{xx}^* = V_{xx} - \Lambda^T V_{uu} \Lambda$$

The backward pass started from the quadratic model of l_T and then recursively computes the optimal gains of all the control cycles from $T - 1$ down to 0.

Forward pass

The forward pass then computes the new candidate trajectory and control schedule. For each control cycle starting, a new control schedule $\tilde{u}$ could be established using (4.14). For each new $\tilde{u}$, the changes is x are obtained by integrating (4.8) from x_0. The changes in x are denoted by :

$$\Delta x' = x' - f(x, \tilde{u})$$

The new control is then :

$$\tilde{u} = u + \lambda + \Lambda \Delta x$$

Implementation details

The computation stops once there is no relevant improvement of the total cost. The first times of the computed control, the window preview, could be played by the system : the last are generally not admissible. Indeed, the approximation of the cost function is good closed to the start point, but is too far from reality progressively as we advance in time. The real state is then used to initialize a new DDP.

The DDP is a complex algorithm, see (Tassa *et al*. (2012b)) for more details. Two delicate points have to be mentioned :

– Hessian V_{uu} is not necessary invertible. The addition of the Marquardt term μI, where I is the identity matrix, can prevent this problem and ensures more robustness. The parameter μ

then varies over iteration following the Levenberg-Marquardt algorithm, and might involve several successive backward loops until a sufficient μ is chosen.

– The new trajectory may be too far from the previous because of the open-loop term, which likely would make the algorithm diverge. The contribution of this term then has to be reduced. This justifies the introduction of a second coefficient $0 < a < 1$ to control the step length. It is typically chosen by dichotomy.

4.3.3 Complexity and limits

The backward loop involves T inversion of a n_u matrix, where n_u is the control size. The cost is then $O(Tn_u^3)$, contrary to a full Newton step of the implicit problem whose complexity is $O(T^3n_u^3)$. An additional cost is the gradient l_x, l_u, f_x and f_u and the corresponding Hessian. This cost might be prevailing if the derivatives are computed numerically. The forward loop involves T integration of (4.8), but might be repeated several times when searching for a.

The DDP strongly relies on the quadratic model of the cost function, which can be very local for complex cost landscape and non linear dynamics. This might strongly reduce the convergence speed. It also implies to have a good initial guess.

For MPC, we generally can assume that the shifted control schedule at the previous control cycle (with zero or copied terminal control) is a good guess. In that case, two additional hypotheses can be taken to speed up the control process. First, we can assume that the optimal is reached in one iteration. Second, we can neglect the second order $f_{xx}\Delta x$, $f_{uu}\Delta u$ while assuming a quadratic model of l_t. This corresponds to the Gauss hypothesis of the Newton descent and is named iterative LQR (Todorov et Li (2005)).

4.4 Variable Stiffness Actuators

The previous section recalled a generic MPC framework. We now propose a solution to implement it to efficiently control the behavior of a variable-stiffness actuator.

4.4.1 Principles

The operational-space approach provides the reference joint behavior (torques and/or stiffness) from operational references set by the user. The computation is typically expensive, since many DOF are involved and the whole-body operations scale with the cube of the number of joints. We can expect them to be computed at a middle dynamics (e.g. 100Hz). For in contact movements, the actuators have to react much faster to track this reference behavior (typically, 1kHz is advised (Hogan (1985)), 3kHz is implemented on the LWR robot (Albu-Schaffer *et al*. (2007)) while 10kHz

are available on the iCub (Tsagarakis *et al*. (2011))). The joint references are then tracked by a DDP-based MPC whose cost function is described below.

4.4.2 Actuator generic model

We consider electrical-based variable stiffness actuators, where two motors in parallel are connected in series with some mechanical springs. This generic hypothesis is valid for a large class of actuators (Jafari *et al*. (2013); Vanderborght B. (2011)). See App. 4.7 for the description of two such actuators. While the general approach is still valid, this precise hypothesis cannot cover for example pneumatic actuators (Tondu et Lopez (2000); Tassa *et al*. (2013b)). The state variables are then reduced to the spring positions and velocities. Typically, two motors are involved :

$$x = (x_1, x_2, \dot{x}_1, \dot{x}_2)$$

If assuming linear (or at least static) models of the spring, the output torques can be deduced from the state.

We additionally make the hypothesis that the motor is capable of tracking acceleration references. While the electrical motors are typically driven in electrical power (current or voltage), a first close loop is typically configured on the feedback of a output position (angular) sensor. Moreover, a mechanical reductor is often added in series with the motor, which acts as a dynamic screen preventing the motor to feel the joint dynamics. These two facts make the acceleration an easy but realistic control input. The control is then simply :

$$u = (\ddot{x}_1, \ddot{x}_2)$$

4.4.3 Cost function

We consider the following terminal cost function :

$$l_T(x_T) = w_T(\tau(x_T) - \tau_T^*)^2$$

The current torque is a function of the spring states, while τ_T^* is given by the operation-space computations. The running cost function is :

$$l_t(x_t, u_t) = w_\tau(\tau(x_t) - \tau_t^*)^2 + ||u_t||_Q^2 + w_j j(x_t)$$

where τ_t^* is typically equals to τ_T^*, j is a barrier function enforcing the joint limits and w_f, w_τ, w_j and Q are arbitrary gains that encompass the unit differences and the relative importance of the terms. The first term enforces the MPC to track the reference. The second term penalizes strong

internal movements and regularizes the numerical solver behavior. The last term enforces any given constraints [3]. The barrier function is constructed to be positive, continuous and twice derivable, to value 0 on at ϵ from the limit and $+\infty$ on the limits.

An additional term $w_\Gamma(\Gamma(x_t) - \Gamma_t^*)^2$ can be added in both l_t and l_T to track a reference output stiffness, also function of the state. If no good reference stiffness is available, w_Γ is set to 0. The stiffness is then locally tuned by the optimal controller in answer to the actuator demanded dynamics. A comparison with and without reference stiffness is given in the next section.

4.4.4 Discussion

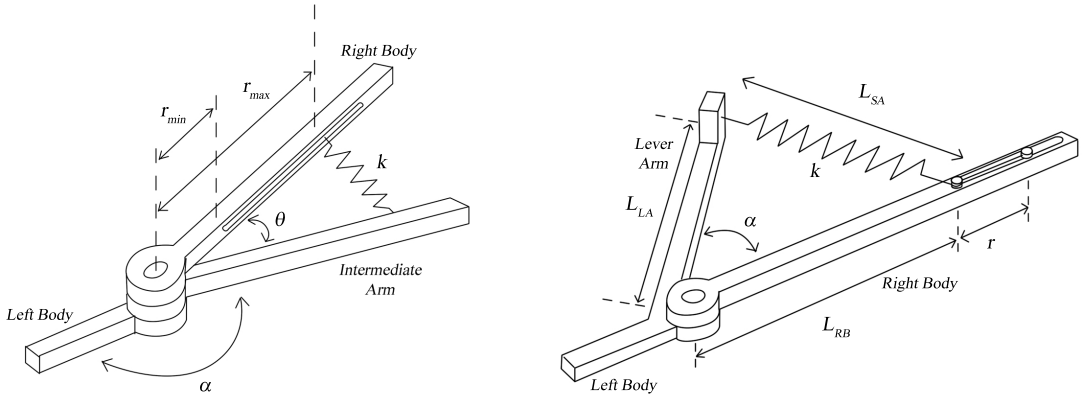
The obtained control scheme may be compared to the optimal controller described in (Sardellitti *et al*. (2013)). In this paper, the authors control a torque-driven actuator to enforce an output position on an AwAS actuator (Jafari *et al*. (2013)), see App. 4.7.1. They deployed a LQR by linearizing their actuator. This is possible since the torque-input position-output AwAS is linear for small spring deflection. For that mean, they have to separate the model between one main actuator driven the output motion and one stiffness actuator tuning the apparent stiffness. They finally proved the stability of their control scheme and proposed a theoretical study of its behavior that was used to automatically adapt the cost-function weights.

Our approach is more generic since no separation is needed and non-linear actuators may be considered, for example the MACCEPA, see App. 4.7.2. Moreover, if the stiffness is free, the controller will use it to reduce the oscillations of the output movement. Using the LQR formulation behind the DDP, the stability of our MPC can be derived using the same reasoning than in (Sardellitti *et al*. (2013)). It seems more difficult to adapt the pole study for tuning the weights in general, since the poles vary with the system non-linearities. However, the same study can be directly applied when the non-linearities can be neglected (e.g. for the AwAS at steady-state).

4.5 Results

We present in this section two set of simulations. In a first time, we checked the behavior of the MPC for a single actuator using the two actuator models described in App. 4.7. The AwAS has a nearly linear dynamics, which is interesting to validate the LQR behavior, while the MACCEPA is less linear, which validates the capabilities of the DDP to handle a more complex cost landscape. We then produce a complex whole-body movement for a humanoid robot whose joint references are tracked by the MPC.

3. The DDP is efficient partly because unconstrained. The barrier function is an efficient solution to enforce important terms to be treated as constraints.



AwAS MACCEPA

Figure 4.1 Schema of the two actuator models used in the experiments.

4.5.1 Single actuator

Set up

The simulations are made on the software Matlab. The model dimension are chosen to fit the requirements of a full size humanoid robot (for typical movements recorded on HRP-2). The control sampling frequency is 10kHz : the DDP is recomputed at 10kHz and the first sample is applied on the motor models. The preview windows lasts 2.5ms (that is to say 25 samples). The number of DDP iteration is bounded to 50 (while AwAS generally converges in less than 25). The MPC is used for tracking a smooth and a pulse-train reference without disturbance, and the behavior is then checked for external disturbances.

Torque reference tracking

DDP control has been tested for two scenarios : a pulse train and a polynomial. The response plot are given in Figures 4.2 and 4.3.

The first reference torque is a polynomial of degree three (Fig. 4.2). For both actuators, output torque is properly tracked. In the absence of discontinuity, DDP provides excellent control, no matter with the complexity of cost function.

The pulse-train amplitude is set to 10 Nm (calibrated on the knee torques during humanoid dynamics locomotion task). The AwAS is very reactive, reaching quickly 97% of the desired value and is stable. For the MACCEPA model, results are more mixed, with slower convergence time.

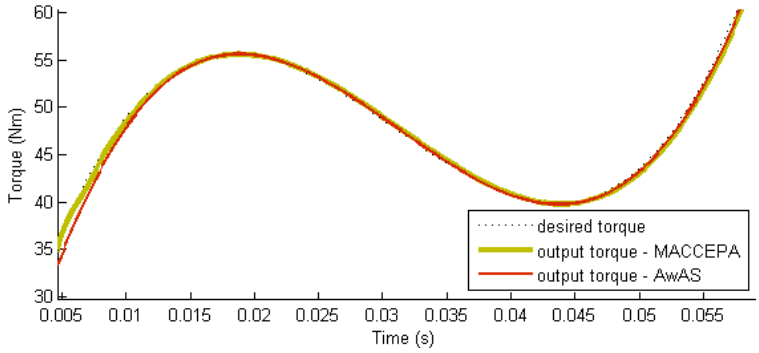


Figure 4.2 Output torques with AwAS and MACCEPA actuators for a polynomial control

These differences are explained by the different level of complexity of the two dynamics, excited by the train pulse that generates singularities. It is representative of a shock during an explosive task. The MACCEPA behavior might be improved by adjusting the cost weights, for example using a deeper analysis of the close-loop system.

Robustness to a perturbation

The robustness is tested for a constant reference. We let the system converge and then apply a static force of 5 Nm at the actuator output. The result is summarized in Fig. 4.4. The response time before new stability of the torque generated by the spring tension is of the same order as in the case of train pulse. The system naturally comes back to the reference position after a small adaptation time.

4.5.2 Whole-body movement

Set up

We consider a jumping movement with a model created from the HRP-2 robot (Kaneko *et al.* (2002)) : the body specifications have been kept as the actuator dimensions, but we have replaced in simulation the stiff actuation by a set of variable stiffness actuators. The movement is defined by a sequence of tasks using the jumping methodology proposed in (De Lasa *et al.* (2010)). The motion is composed of three phases : starting from a low squatting posture, the robot violently accelerates its center of mass (CoM) until the legs are stretched while keeping a controlled (typically zero) angular momentum. The robot then floats while its CoM decelerates due to the gravity. During the flight phase, only the orientations of the feet are controlled to anticipate the landing. The robot finally impacts the ground and gently decelerates its CoM to recover its rest posture. The joint references are computed at 200Hz. One MPC at each joint then tracks this reference at 10kHz

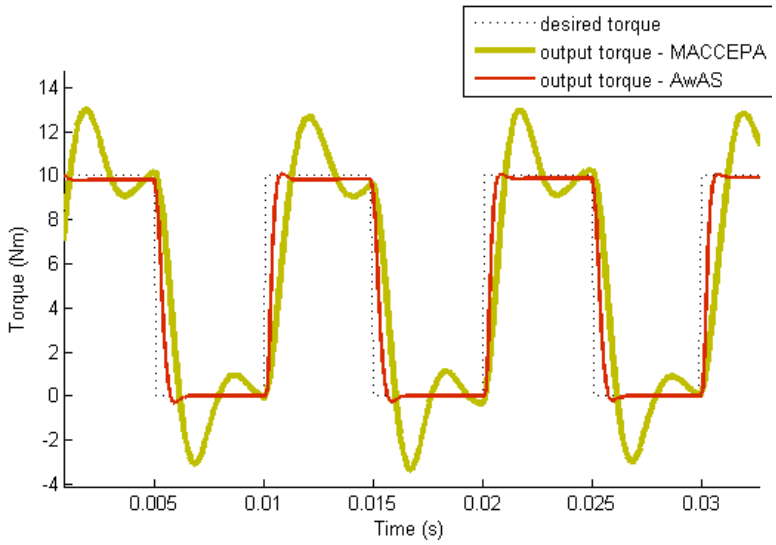


Figure 4.3 Output torques with AwAS and MACCEPA actuators for a pulse train control

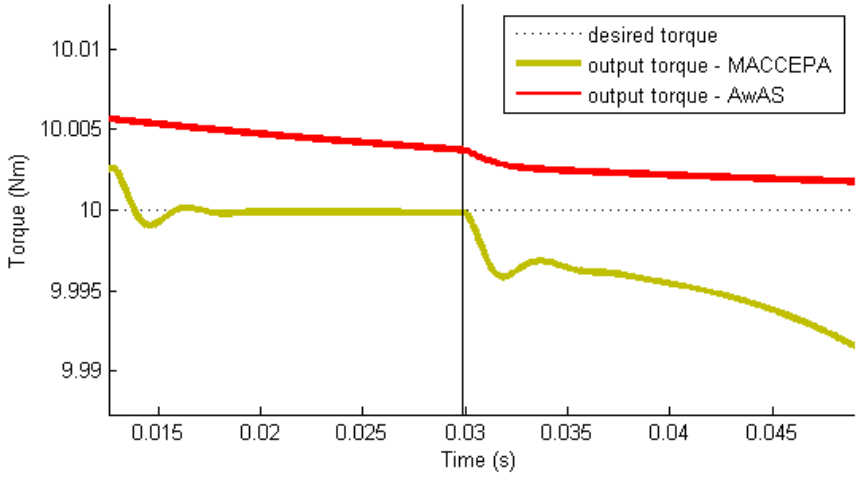


Figure 4.4 Effect of 5 Nm perturbation torque at time t = 30 ms on a constant control

following the same set-up as in the previous section.

Torque tracking

The motion is summarized in Figures 4.5 to 4.8. Snapshots of the movement are gathered in Fig. 4.5. The whole body movement is not detailed by lack of space. The joint torques are tracked by an AwAS on the hip and a MACCEPA on the knee (the choices are arbitrary). We spotted torques on joints of a humanoid robot during a simulation of jump on site. The tracking performances are displayed in Fig. 4.6. The explosive movement is properly tracked. The reference torque at the impact is 3 Nm for the hip, and the MPC produces 3.3 Nm (91%). At the knee, the reference torque is 9.2 Nm while the MPC produces 10 Nm (92%). The delay is 2 ms which corresponds to the spring dynamics. This result proves the interest of using the spring actuation with MPC for achieving complex whole-body dynamic movements.

Stiffness selection

The joint compliance is computed from the reference operational compliance. We set up a compliance of 2000 N/m on the z-axis of each foot, with a compliance of 40000 Nm on the two lateral rotations and stiff coupling between the translation and the rotations. The obtained joint stiffness of one leg is plotted in Fig. 4.7. It is perfectly tracked by the MPC. When the stiffness is let free, the MPC uses this additional DOF to optimize the output torque while minimizing the input. It is evident on the single-actuator pulse train displayed in Fig. 4.8. In particular, the stiffness changes at each new pulse and constantly decreases (the spring position increases) when the actuator position stabilizes.

4.6 Conclusion

In this paper, we have proposed an original approach to control the whole-body movement of complex variable-stiffness robot. The operational-space approach is first used to compute the joint torques and stiffness from operational force, acceleration and compliance references. An MPC controller is then used to control the movement of each elastic actuator to track the joint references. The global approach can be seen has an efficient suboptimal to the yet untractable whole-body optimal control problem. It is very appealing to apply torque references on electrical actuators while avoid an excessive mechatronic cost.

More particularly, we have proposed an original solution to compute the joint compliance from operational references. We have proposed an efficient MPC solution to track the joint references, both torques and stiffness, using DDP. This method can be seen as a generalization to a large class of actuator and to non-linear situations of the LQR proposed in (Sardellitti *et al*. (2013)). Finally, we

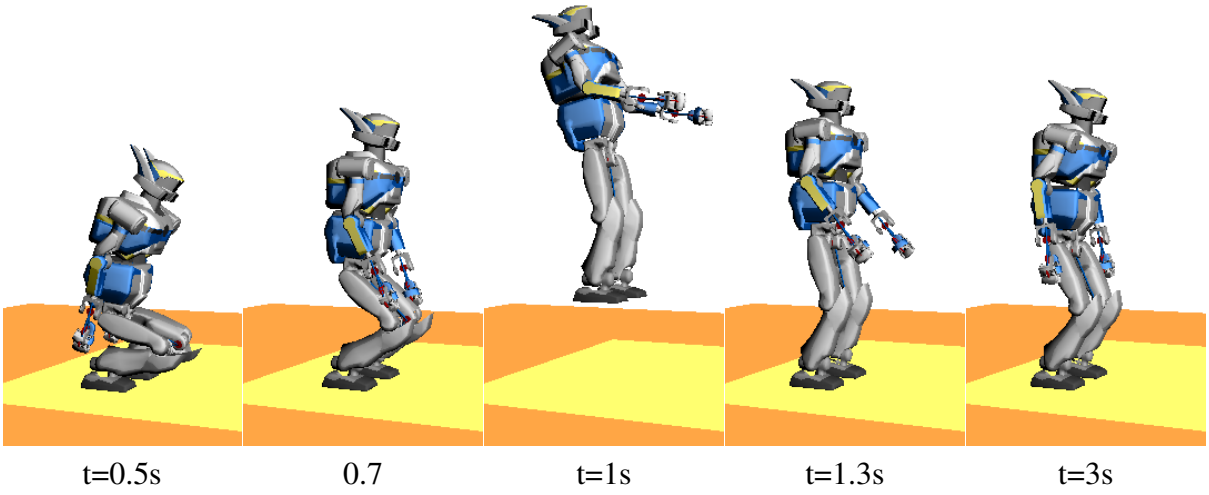


Figure 4.5 Snapshots of the jumping movement.

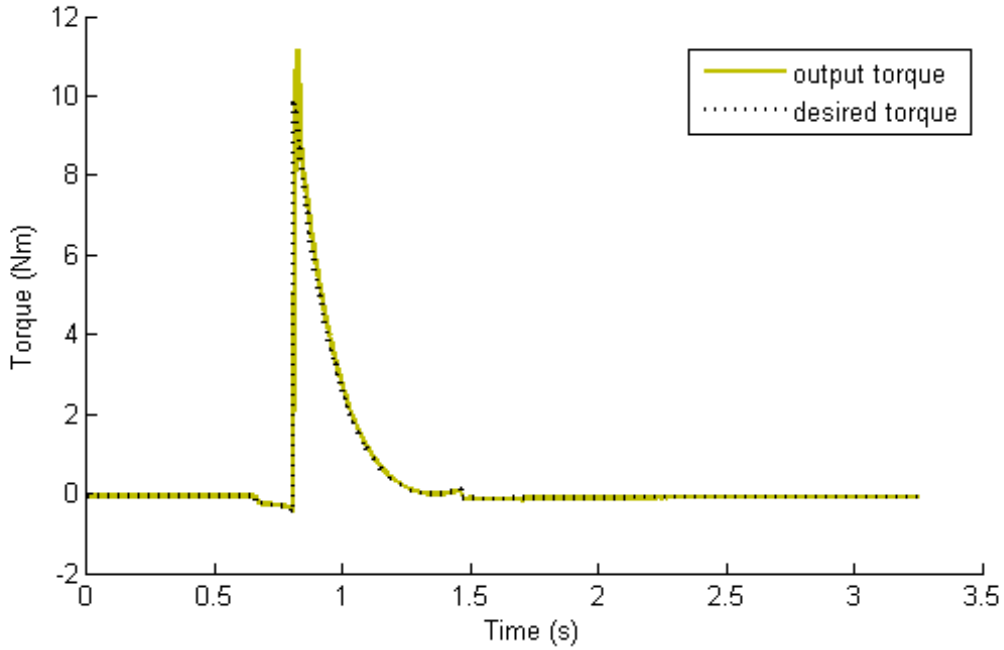


Figure 4.6 Output torque with AwAS for a jump on site, on the hip

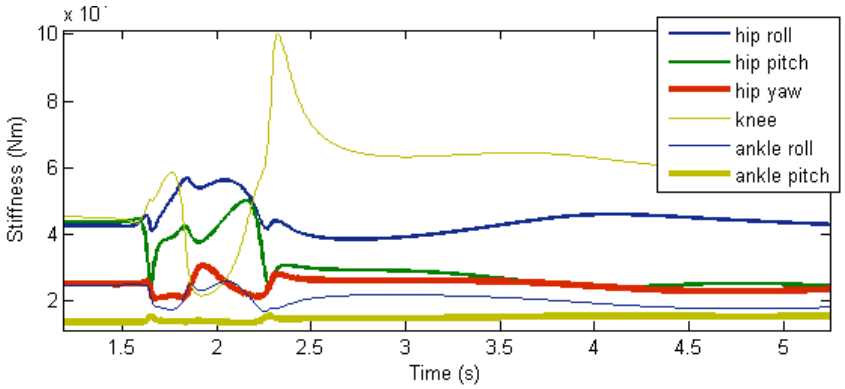


Figure 4.7 Reference stiffness of the right leg during the jump.

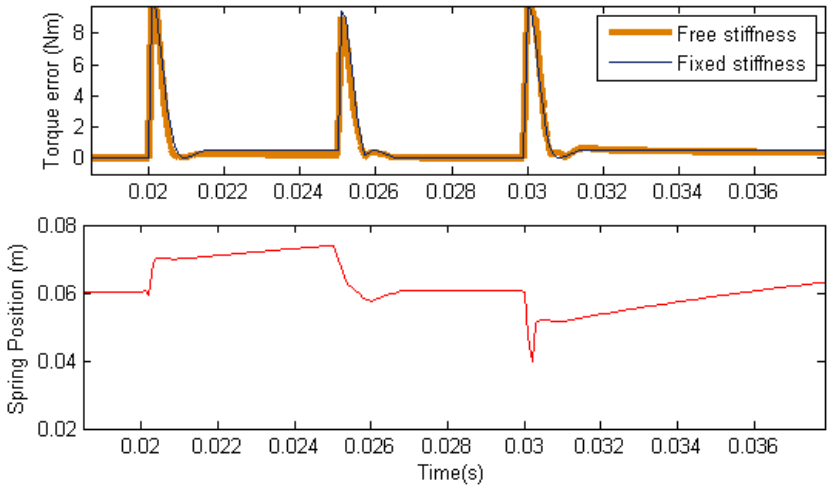


Figure 4.8 Advantage of the free stiffness (top) Output torque error with free and constant stiffness. (bottom) Spring motion when the stiffness is free.

have proposed a complete validation in simulation, by controlling the jump of a variable-stiffness full-size humanoid robot. Behind the application to a physical robot, the next step of our work is to exhibit the theoretical properties of our controller, in particular its stability.

4.7 Appendices : Actuator models

We quickly recall here the model of the two actuators used in the experiments and displayed in Figures 4.1.

4.7.1 Actuator with Adjustable Stiffness (AwAS)

This actuator is composed of two arms rotating about a common axis on the left body and linked by a spring (see Fig. 4.1, left). One arm is the right body whose position and velocity are the variables of interest. The other arm is an intermediate between the left and the right bodies : its rotation causes the rotation of the other arm thanks to the spring, whose position can be adjusted to change the apparent stiffness of the system.

The state of the system is $X = \begin{pmatrix} \Theta & \dot{\Theta} & r & \dot{r} \end{pmatrix}^T$, where Θ is the angle between the both arms. The equation of the system and the output torque function are so :

$$\ddot{\alpha} = \ddot{\Theta} + \frac{kr^2}{M} sin(\Theta), \quad \tau = kr^2 sin(\Theta)$$

where M is the moment of inertia of arms and k the stiffness of the spring.
The control $U = \begin{pmatrix} \ddot{\alpha} & \ddot{r} \end{pmatrix}^T$ includes two variables : the angular acceleration $\ddot{\alpha}$ of the intermediate arm and the acceleration $\ddot{r}$ of the spring along the arms.

4.7.2 Mechanically Adjustable and Controllable Compliance Equilibrium Position Actuator (MACCEPA)

The actuator MACCEPA is also in serial mode (see Fig. 4.1, right), but its output torque function is more complex. The right body rotates around the left body, and is linked to this one by a spring, whose extension can be adjusted. The control vector is quite the same as previously : $U = \begin{pmatrix} \ddot{\alpha} & r \end{pmatrix}^T$, where $\ddot{\alpha}$ is the angular acceleration on the right body, and r the extension of the spring along the right body. The output torque function is so :

$$\tau = k \frac{L_{RB} L_{LA} sin(\alpha)}{L_{SA}} (L_{SA} - |L_{RB} - L_{LA}| + r) \tag{4.15}$$

$$L_{SA} = \sqrt{L_{RB}^2 + L_{LA}^2 - 2 L_{RB} L_{LA} cos(\alpha)} \tag{4.16}$$

where L_{RB}, L_{LA} and L_{SA} are respectively the lengths of the right body, of the lever arm and of the arm with the spring. The equilibrium position is at $\alpha = 0$ and $r = 0$.

CHAPITRE 5

From Inverse Kinematics to Optimal Control

Article accepted to the *Advances in Robot Kinematics* conference in february 2014.

Auteurs :
Perle Geoffroy, École Polytechnique de Montréal, Montréal, Canada
Nicolas Mansard, LAAS-CNRS, Univ. Toulouse, Toulouse, France
Maxime Raison, École Polytechnique de Montréal, Montréal, Canada
Sofiane Achiche, École Polytechnique de Montréal, Montréal, Canada
Yuval, Tassa, Univ. Washington, Seattle, USA
Emo Todorov, Univ. Washington, Seattle, USA

5.1 Introduction

Both inverse geometry (McCarthy (1990)) and inverse kinematics [1] (Whitney (1969)) can be viewed as the resolution of an optimization problem : non-linear from the configuration space to the special Euclidean group $SE(3)$ for the first one (Das *et al*. (1988)), quadratic in the tangent space to the configuration space for the second (Escande *et al*. (2012)). This is only one view on the problem, but it helps to formulate efficient solvers and to understand their convergence properties, by using some powerful results of numerical optimization (Nocedal et Wright (2006a)). For controlling a robot, inverse kinematics is nowadays a standard technique, for its simplicity and the limited computation cost (e.g. 1ms is enough to invert the kinematics of a 40DOF humanoid robot (Escande *et al*. (2012))). Moreover, the structure of the problem is well understood and it is easy to diagnose any trouble during the execution.

On the other hand, model predictive control (MPC) is an advanced control technique to control a given system by optimizing its predicted evolution (Alamir (2006)). It relies on the systematic evaluation of the control of the system with respect to a reference cost function, while only the first few steps of the optimal trajectory are executed before its complete re-evaluation. The main interest of MPC is the ability of dealing with non-linear systems whose instantaneous linearization is not meaningful.

1. The problem we name *inverse geometry* is sometimes referred as *inverse kinematics*, the second being referred as *differential* (or closed-loop) *inverse kinematics*. We use geometry when only static postures are implied and keep the word kinematics when a motion is explicitly implied.

Like for inverse kinematics, MPC can be formulated as the resolution at each control cycle of a numerical optimization problem depending on the estimated state. However, the typical size of the problem generally makes it difficult to obtain real-time performances (Mombaur (2008)). Moreover, the understanding of this kind of formulation is much lower. It is typically difficult to quantify the robustness of such a controller (Alamir (2006)), or even to explain the reasons that have led to the chosen trajectory.

In this paper, we consider an optimal-control solver named Differential Dynamic Programming (Jacobson et Mayne (1970)). This numerical scheme provides a simple yet efficient solver of direct implicit (shooting) optimal-control problems, that makes it possible to control complex systems, like humanoid robots (Tassa *et al.* (2012a)), despite the inherent complexity of this class of problems. We propose a reformulation that provides numerical advantages and, more importantly, gives a better understanding of the structure of the optimal trajectory. In particular, when only the robot kinematics is considered, we show that every iteration of the algorithm amounts to a sequence of Jacobian pseudo-inversions along the trajectory. Classical pseudoinverse-based inverse kinematics is then equivalent to the optimization of a single-step trajectory. Consequently, once the ratio between the size of the system and the CPU load are sufficiently low, any inverse-kinematics should be considered with several steps ahead rather than with only a single one. The same observation seems valid for inverse dynamics (Khatib (1987)).

5.2 Model predictive control

5.2.1 Principles and model

Consider a generic system, with state x and control u, defined by its dynamics :

$$x_{t+1} = f(x_t, u_t, t) \tag{5.1}$$

where f is the evolution function and the time variable t is discrete. x is typically a finite sequence of derivatives of the configuration q, e.g. $x = (q, \dot{q})$. Optimal control computes the control and state trajectories that minimize a given cost function :

$$\min_{X,U} \sum_{t=0}^{T-1} l_t(x_t, u_t) + l_T(x_T) \tag{5.2}$$

subject to the constraint (5.1), where T is the preview-interval length (fixed here), $U = (u_0 ... u_{T-1})$ and $X = (x_0, ..., x_T)$ are the control and state trajectories and l_t and l_T are the running and terminal cost functions. Linear dynamics and quadratic cost lead to the linear-quadratic regulator, given by Riccati equations.

In practice, the information contained in X and U is somehow redundant. The problem is reformulated as a problem only on X or only U (the other variable being deduced from the dynamic equation). The formulation is said explicit when computing X (Mordatch *et al*. (2012)) (designated also by *collocation* (Schulman *et al*. (2013))) and implicit when computing U (Tassa *et al*. (2012b)) (designated also by *shooting* (Leineweber *et al*. (2003))). Both formulations have pros and cons (Biegler (2010)). We consider in the following the implicit formulation, cheaper to solve in practice, without the drawback that it might involve more local minima. For each formulation, the solution to the numerical problem is then approximated using any optimization solver, typically using Newton or quasi-Newton (Goldfarb (1970)) descent.

5.2.2 Differential dynamic programming

The Differential Dynamic Programming (DDP) is an iterative algorithm to solve a non-linear optimal control problem using implicit formulation (Tassa *et al*. (2012b)). It is nearly equivalent to the application of a Newton descent algorithm (Pantoja (1988)). As in the Newton descent, it approaches a local optimum by iteratively modifying a candidate solution.It starts with initial state and control trajectories (e.g. obtained by integration of the zero control) and then iterates in two stages. It first computes a quadratic model of the current candidate trajectory and computes the corresponding LQR (backward loop). The candidate is then modified following the LQR (forward loop).

Quadratic model : We denote v_t the cost-to-go function defined by :

$$v_t(X_t, U_t) = \sum_{k=t}^{T-1} l_k(x_k, u_k) + l_T(x_T) \tag{5.3}$$

where $X_t = (x_t..x_T)$ and $U_t = (u_t..u_{T-1})$ are the trajectory tails. To simplify, we drop the t variable and denote the next quantity at $t+1$ by a prime : $v_{t+1} = v'$.

DDP relies on Bellman principle : the minimization on a global sequence can be separated in two minimization problems. It proceeds recursively from the back thanks to the following equation :

$$v^*(X, U) = \min_{x,u,X',U'} (l(x, u) + v'^*(X', U')) \tag{5.4}$$

DDP recursively builds a quadratic model of v from the quadratic models of both l and v'^* :

$$\begin{aligned} v(x + \Delta x, u + \Delta u) = v(x, u) &+ v_x \Delta x + v_u \Delta u \\ &+ \frac{1}{2} \Delta x^T v_{xx} \Delta x + \Delta u^T v_{ux} \Delta x \\ &+ \frac{1}{2} \Delta u^T v_{uu} \Delta u + o(||\Delta x||^2 + ||\Delta u||^2) \end{aligned}$$

The quadratic model is defined by the quadratic coefficients v_x, v_u, v_{xx}, v_{ux} and v_{uu}, functions of the derivatives of l, f and v' (see (Tassa *et al*. (2012b)) for details).

Backward pass : The optimum Δu can be computed for any Δx. It is obtained at the zero of the derivative of the quadratic model :

$$\Delta u^* = \lambda + \Lambda \Delta x \tag{5.5}$$

where $\lambda = v_{uu}^{-1} v_x$ and $\Lambda = v_{uu}^{-1} v_{ux}$ are the open-loop and close-loop gains. From the optimal change Δu^*, the quadratic model of v^* can be computed :

$$v_x^* = v_x - \Lambda^T v_{uu} \lambda \tag{5.6}$$

$$v_{xx}^* = v_{xx} - \Lambda^T v_{uu} \Lambda \tag{5.7}$$

The backward pass starts from the quadratic model of l_T and then recursively computes the optimal gains of all the control cycles from $T-1$ down to 0.

Forward pass : The forward pass then computes the new candidate trajectory and control schedule. For each control cycle starting, a new control schedule $\tilde{u}$ could be established using 5.5. For each new $\tilde{u}$, the changes is x are obtained by integrating (5.1) from x_0 and then propagated through the closed-loop gains of the next time :

$$\Delta x' = x' - f(x, \tilde{u}) \tag{5.8}$$

$$\tilde{u} = u + \lambda + \Lambda \Delta x \tag{5.9}$$

Performance : The interest of DDP is that its simple formulation can be easily implemented in an efficient way, taking into account the inherent sparsity of a numerical optimal control problem. For example, in Tassa *et al*. (2012a), a dedicated solver was demonstrated to animate a humanoid virtual avatar in real-time in interaction with a user through a haptic device. It was used to control a simulated 25-DOF HRP2 robot in real-time (Tassa *et al*. (2014)). In that case, the preview horizon was 0.5s. The preview control was computed in 50ms and then interpolated using the underlying LQR at 5ms, enabling effective real-time control (see Fig. 5.1).

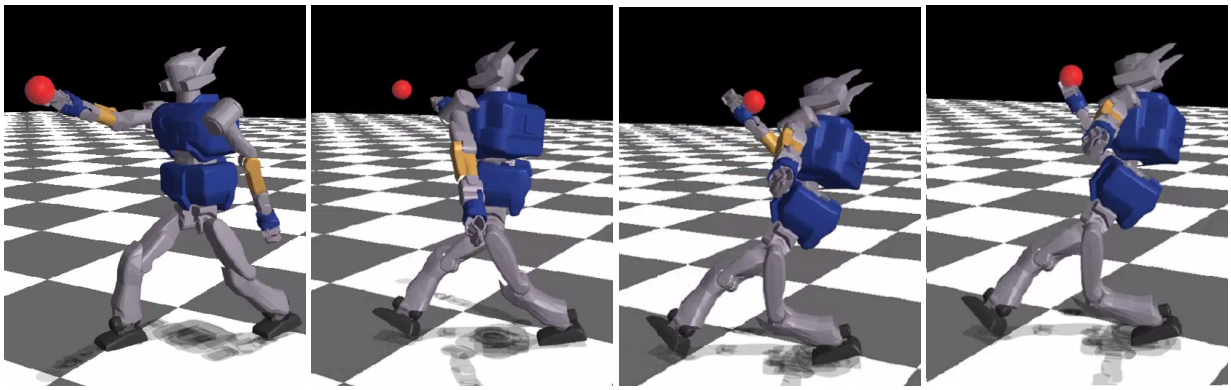


Figure 5.1 Snapshots of a whole-body grasping movement on a 25-DOF humanoid robot. The control is computed in real-time. Courtesy from Tassa *et al*. (2014).

5.3 Square-root differential dynamic programming

5.3.1 Algorithm reduction

The Gauss-Newton approximation : Very often in practice, both the running and terminal costs have sum-of-square form, with the residuals $r(x, u)$:

$$l(x, u) = r(x, u)^T r(x, u) \tag{5.10}$$

This specific shape is interesting in practice as it leads to a cheap approximation of the second-order derivatives of l in neglecting the second order derivative of r. This is referred as the Gauss-Newton approximation.

$$l_{xx} = r_x^T r_x \tag{5.11}$$
$$l_{ux} = r_u^T r_x \tag{5.12}$$
$$l_{uu} = r_u^T r_u \tag{5.13}$$

where r_x and r_u are respectively the derivatives of r by respect x and u. The approximation converges to the real Hessian when the residuals r converge to 0, which in general ensures a good convergence. On the other hand, the approximated Hessian is always positive, which prevent the algorithm to violently diverge like when the true Hessian becomes non-positive. Moreover, the particular shape of the approximated Hessian can be taken into account when inverting it, since we

have :

$$l_{xx}^{-1} l_x^T = \left(r_x^T r_x\right)^{-1} r_x^T = r_x^+ \tag{5.14}$$

where r_x^+ denotes the Moore-Penrose pseudoinverse of r_x and can be efficiently computed without explicitly computing the matrix product $r_x^T r_x$, using for example the SVD or other orthogonal decompositions (Golub et Van Loan (1996)).

In the literature, the Gauss-Newton approximation of the DDP algorithm is referred as the iterative LQR (iLQR) algorithm (Todorov et Li (2005)). In this section, we take advantage of the square shape of the cost and derivatives to propose a more efficient formulation of this algorithm. This shape will also be used to make some correlations with the classical inverse kinematics.

Square-root shape of v^* : In the DDP backward loop, we have to invert the derivatives of v. Being a sum of square, the cost-to-go v can be expressed as the square of some vector $v^* = {s^*}^T s^*$. However, the DDP does not compute explicitly the derivatives s_x but rather propagate directly the derivatives v_{xx}^* from $v_{xx}'^*$. In the following, we formulate the same propagation while keeping the square shape, by searching the vector $\hat{s}^*$ and matrix $\hat{s}_x^*$ such that

$$v_x^* = {\hat{s}_x^*}^T \hat{s}^* \tag{5.15}$$

$$v_{xx}^* = {\hat{s}_x^*}^T \hat{s}_x^* \tag{5.16}$$

At the beginning of the backward pass, the square shape is trivially given by $s(T) = r(T)$ and $s_x(T) = r_x(T)$. During the backward pass, the previous square-root shapes are written s' and s'_x. The derivative v_{xx}^* is given by the recurrence (5.6). The square shape of 5.7 is not trivial since it appears as a difference, that we can prove to be positive. We denote by s, s_x and s_u the square root of v, v_{xx} and v_{uu} :

$$s = \begin{bmatrix} r \\ s' \end{bmatrix} \tag{5.17}$$

$$s_x = \begin{bmatrix} s_x \\ {s_x^*}' f_x \end{bmatrix} \tag{5.18}$$

$$s_u = \begin{bmatrix} r_u \\ {s_x^*}' f_u \end{bmatrix} \tag{5.19}$$

It is easy to show that $v'_{xx} = {s'_x}^T s'_x$, $v'_{xu} = {s'_x}^T s'_u$ and $v'_{uu} = {s'_u}^T s'_u$. In that case, the gains are given

by the pseudo inverse of s_u :

$$\lambda = s_u^+ s \tag{5.20}$$

$$\Lambda = s_u^+ s_x \tag{5.21}$$

Thanks to the Moore-Penrose conditions, we can reduce $\hat{s}^*$ and $\hat{s}_x^*$ to :

$$\hat{s}^* = s \tag{5.22}$$

$$\hat{s}_x^* = (Id - s_u s_u^+) s_x \tag{5.23}$$

5.3.2 Advantages and discussion

Keeping the square shape of v_{xx} avoids some numerical trouble. In particular, decomposing s_x instead of v_{uu} offers a much better numerical behavior. Moreover, it avoids the complexity of a big matrix multiplication. This is formalized below.

Comparison of the costs : To evaluate the complexity of this algorithm, sizes of x, u and r are supposed all equal to n. The cost for one iteration of the loops is $8n^3$, against $11n^3$ for the classical DDP. Moreover, most operations are due to the QR decompositions and could be performed when computing the derivatives, that leads to a total cost of roughly $3n^3$.

Pseudo inverse and projection : The gains and propagation closed forms also provide a better understanding of the nature of the inversion. As in the developments, we consider only the current time of the backward loop. The Jacobian s_u is the derivative of the cost-to-go. The open-loop gain $s_u + s$ only tries to find the current control that minimizes the cost-to-go evolution. In most of cases, s_u has more rows than columns. The pseudoinverse will only provide the control that has the maximum efficiency in the least-square sense. It remains a part of the cost that can be nullified. This rest is given by the orthogonal to the image of s_u, *i.e.* the kernel of s_u^T, whose projector can be computed by $P_u = Id - s_u s_u^+$.

The backward loop then propagates backward the part of the cost that was not accomplished, and that is selected using the projector. The trajectory optimization then corresponds to a sequence of virtual configuration, each of them being moved to optimize its own cost r and to help the configurations ahead in the trajectory by optimizing their residual cost $s^{*'}$.

5.4 Kinematic simulation

Three-rotations planar (3R) Robot : Due to a lack of space, we only present some analytical results in simulation with a 3R kinematic model. The dynamic evolution function is reduced to a trivial integration scheme $f(x,u) = x + \Delta t u$, with $x = q$ and $u = \dot{q}$. The robot task is to reach a position p^{ref} with the robot end effector $p(q)$ while minimizing the velocities :

$$r_t = \begin{bmatrix} w_p(p(q) - p^{ref}) \\ w_u u \end{bmatrix} \tag{5.24}$$

with w_p and w_u the weight on the two components of the cost. In that case, the derivative r_x is the robot Jacobian J_q while $r_u = w_u Id$ is a regularization term. At the first step $T-1$ of the backward loop, the pseudo inverse is :

$$s_u(T-1)^+ = \begin{bmatrix} w_u Id \\ w_p \Delta t J_q \end{bmatrix}^+ = \frac{1}{w_p \Delta t} J_q^{\dagger\eta} \tag{5.25}$$

where $.^\dagger$ denotes the damped inverse (Deo et Walker (1992)) with damping $\eta = \frac{w_u}{w_p \Delta t}$. The last term of the trajectory indeed moves following an inverse-kinematics scheme. The same interpretation can be done on the other samples, with a similar regularization and a task that makes a trade-off between going to the target and helping the next sample in the trajectory to accomplish its residual.

Results : The Square Root algorithm on the simulated 3R Robot was implemented in language C++. The control sampling frequency is 1 kHz and the cycle of the robot lasted 0.1s (i.e. 100 points). We chose $w_u = 0.01$ and $w_p = 1$. The control reaches easily the target with a proper smoothing of the control, as expecting with such a simple system. With this setting, the robot needs $0.1s$ to reach the target *i.e.* 100 control cycles.

We mainly focus the discussion on the comparison with inverse kinematics. As explained above, inverse kinematics is obtained when the horizon T is reduced to 1. On the opposite, the optimum of the infinite-horizon problem is approximately obtained for a preview horizon of $0.1s$ (which is the time to the goal). We consider the performances in both the obtained cost and the computation load for T varying from 1 to 100. A summary of the results is given by Fig. 5.2.

On the left figure, we consider the total cost for the overall executed trajectory. This cost is computed a posteriori, after the execution by the robot. The cost is minimal when T is maximal and reciprocally. Most of the cost increase when T is small is due to the increase of the control term (with an artificial apparent minimum for some $T = 10$ that is due to the ratio over a changing total quantity). On the other hand, the computation load (right figure) increases linearly with the horizon

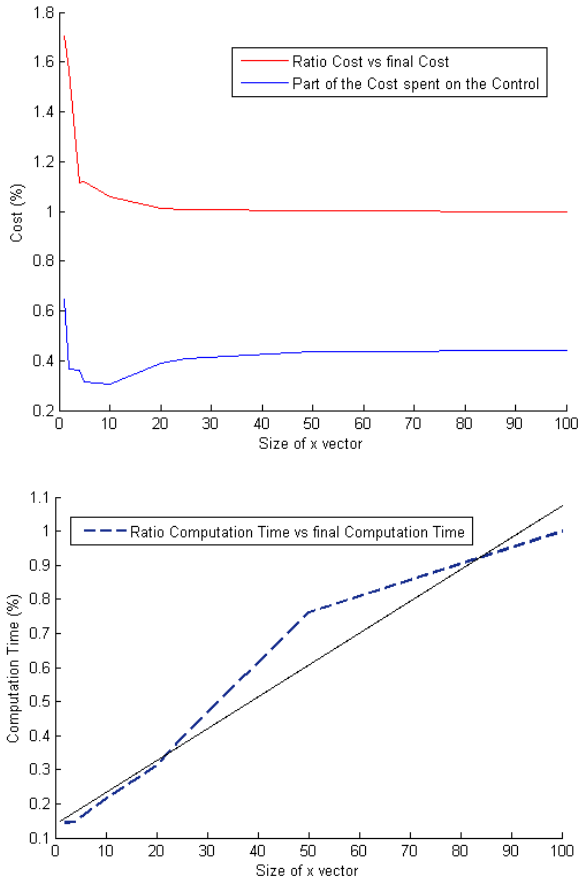


Figure 5.2 **Performance and computation ratio with respect to the preview length : (top)** Evolution of the cost when increasing the preview horizon : the total cost is plotted as a ratio with respect to the infinite-horizon optimum. Indicatively, the percentage of the control term of the cost (integral of the velocity norm) is also given. **(bottom)** Computation load, plotted as a ratio of the load needed to compute the trajectory with a single-step horizon (*i.e.* cost of an inverse kinematics). The cost increases linearly with the size of the horizon.

length (as expected). After a certain threshold on the horizon length, the obtained trajectories are the same, with same costs. As always with MPC, the horizon length has to be carefully adjusted : not to small, to find the best trajectory and not to big to limit computation times.

Inverse kinematics is obtained for $T = 1$. The cost is the lowest, but a poor resulting cost. For only a small expense, (e.g. $T = 4$), better trajectories are obtained.

5.5 Conclusion

In this paper, we have exhibited a square-root formulation of the DDP algorithm. The formulation is numerically more efficient, improving both the computation load and the numerical

conditioning. It also makes apparent the relation between MPC and other optimization-like robot algorithm. In particular, it reveals a sequence of pseudo-inverse of the cost Jacobians along the trajectory. In the particular case where the time evolution function is reduced to the robot kinematics, this sequence is equal to the pseudo-inverse of the cost Jacobian, with the first term of the trajectory following exactly an inverse-kinematics scheme to accomplish the final cost.

This study reveals that inverse kinematics is nothing but a MPC scheme with a singular horizon, and that the robot behavior might be very much improve if simply considering some few samples ahead of the current robot position when computing an inverse-kinematics scheme.

The same principle should apply with more complex time-evolution function. For example, when considering the robot dynamics (the state being the configuration and velocity, and the control to joint torques), the MPC should meet the operational-space inverse dynamics when the preview horizon collapses.

CHAPITRE 6

DISCUSSION GENERALE ET CONCLUSION

6.1 Discussion générale

Nous avons montré qu'il est possible de calculer une commande optimale pour les actionneurs à impédance variable, linéaires ou non, sans découpler la position et la rigidité, grâce à l'algorithme DDP. De ce point de vue, notre travail peut être vu comme une extension de celui de Sardellitti(Sardellitti *et al*. (2013)), qui utilise l'algorithme LQR. Nous avons validé complètement notre méthode en simulation, notamment au cours d'un saut réalisé par un robot humanoïde, ce qui est de bon augure pour un emploi futur des VIA sur des prothèses.
Nous avons également reformulé l'algorithme DDP avec les racines carrés des matrices en jeu, sous l'hypothèse de Gauss-Newton. Cette reformulation est plus efficace et de complexité moindre. De plus, elle permet de voir la méthode de la cinématique inverse comme un cas particulier de contrôle prédictif : lorsque la fenêtre de prévision est réduite à un unique point, et donc de réaliser combien la commande d'un robot peut être améliorée rien qu'en utilisant une fenêtre de prévision de quelques points.

6.2 Limitations

Nous avons certes atteint les objectifs visés à l'issue de cette maîtrise. Mais, d'autres questions sont apparues au fur et à mesure de notre étude, et nous n'avons pas toujours pu les traiter soit par manque de temps, soit par manque d'expertise.

- La stabilité théorique de l'algorithme DDP n'a pas été étudiée plus avant : elle est assurée avec les modèles de VIA que nous avons testés, mais nous n'avons pas cherché le cas général de stabilité en vue d'applications à d'autres problèmes.
- Nous avons implémenté la version optimisée de l'algorithme DDP, et vérifié son efficacité. Toutefois, nous n'avons pas été en mesure d'évaluer les temps de compilation. En effet, pour effectuer une comparaison exacte des deux formes de DDP, il est nécessaire d'optimiser tous nos codes. Or cela exige des compétences en informatique bien au-delà des nôtres.

6.3 Améliorations futures

Pour améliorer le travail déjà réalisé, il serait pertinent d'optimiser les codes actuels afin de pouvoir comparer les temps de compilation des deux formes de l'algorithme DDP. Il serait également intéressant de valider l'efficacité de l'algorithme DDP sur le calcul de commande d'actionneurs à impédance variable.

Le travail réalisé au cours de cette maîtrise s'inscrit non seulement dans la recherche en robotique mais également de la réhabilitation, avec de possibles applications sur des prothèses. A présent que nous disposons d'un algorithme capable de calculer des commandes optimales pour des VIA, validé en simulation, il faudrait le valider expérimentalement. Pour cela, on équiperait un robot humanoïde de VIA, par exemple au niveau des jambes, puis on testerait la réalisation de tâches de plus en plus complexes, telles que sauter sur place, puis sauter en avant. Passée cette étape, on pourrait alors construire un prototype de prothèse biofidèle de jambe utilisant des VIA.

RÉFÉRENCES

ALAMIR, M. (2006). *Stabilization of Nonlinear Systems Using Receding-Horizon Control Schemes*. Lecture Notes in Control and Information Sciences. Springer.

ALBU-SCHAFFER, A., HADDADIN, S., OTT, C., STEMMER, A., WIMBOCK, T. et HIRZINGER, G. (2007). The dlr lightweight robot : design and control concepts for robots in human environments. *Industrial Robot : An International Journal*, 34, 376–385.

ASCHER, U. U. M., MATTHEIJ, R. M. et RUSSELL, R. R. D. (1988). *Numerical solution of boundary value problems for ordinary differential equations*, vol. 13. Siam.

BAERLOCHER, P. (2001). *Inverse kinematics techniques for the interactive posture control of articulated figures*. Thèse de doctorat, EPFL.

BIEGLER, L. (2010). *Nonlinear programming : concepts, algorithms, and applications to chemical processes*. SIAM.

BLANCHET, G. (2007). Optimisation quadratique en automatique.

BLAYA, J. A. et HERR, H. (2004). Adaptive control of a variable-impedance ankle-foot orthosis to assist drop-foot gait. *Neural Systems and Rehabilitation Engineering, IEEE Transactions on*, 12, 24–31.

BOGUE, R. (2009). Exoskeletons and robotic prosthetics : a review of recent developments. *Industrial Robot : An International Journal*, 36, 421–427.

CALDWELL, D. et TSAGARAKIS, N. (2002). Biomimetic actuators in prosthetic and rehabilitation applications. *Technology and Health Care*, 10, 107–120.

CANNY, J. (1988). *The complexity of robot motion planning*. The MIT press.

DAL MASO, F., LONGCAMP, M. et AMARANTINI, D. (2012). Training-related decrease in antagonist muscles activation is associated with increased motor cortex activation : evidence of central mechanisms for control of antagonist muscles. *Experimental brain research*, 220, 287–295.

DAS, H., SLOTINE, J.-J. et SHERIDAN, T. (1988). Inverse kinematic algorithms for redundant systems. *IEEE Int. Conf. on Robotics and Automation (ICRA'88)*. Philadelphia, USA, 43–48.

DE LASA, M., MORDATCH, I. et HERTZMANN, A. (2010). Feature-based locomotion controllers. *ACM SIGGRAPH'10*.

DEO, A. et WALKER, I. (1992). Robot subtask performance with singularity robustness using optimal damped least squares. *IEEE ICRA*. Nice, France, 434–441.

DIEHL, M., BOCK, H. G., DIEDAM, H. et WIEBER, P.-B. (2006). Fast direct multiple shooting algorithms for optimal robot control. *Fast motions in biomechanics and robotics*, Springer. 65–93.

ENGLISH, C. et RUSSELL, D. (1999a). Implementation of variable joint stiffness through antagonistic actuation using rolamite springs. *Mechanism and Machine Theory*, 34, 27–40.

ENGLISH, C. et RUSSELL, D. (1999b). Mechanics and stiffness limitations of a variable stiffness actuator for use in prosthetic limbs. *Mechanism and Machine Theory*, 34, 7–25.

ENOCH, A. AMD SUTAS, A., NAKAOKA, S. et S., V. (2012). Blue : A bipedal robot with variable stiffness and damping. Osaka, Japan.

ESCANDE, A., MANSARD, N. et WIEBER, P.-B. (2012). Hierarchical quadratic programming. *Int. Journal of Robotics Research*. [in press].

GILL, P. E., JAY, L. O., LEONARD, M. W., PETZOLD, L. R. et SHARMA, V. (2000). An sqp method for the optimal control of large-scale dynamical systems. *Journal of computational and applied mathematics*, 120, 197–213.

GOLDFARB, D. (1970). A family of variable-metric methods derived by variational means. *Mathematics of computation*, 24, 23–26.

GOLUB, G. et VAN LOAN, C. (1996). *Matrix computations*. John Hopkins University Press, troisième édition.

GRIEDER, P., BORRELLI, F., TORRISI, F. et MORARI, M. (2004). Computation of the constrained infinite time linear quadratic regulator. *Automatica*, 40, 701–708.

HA, K. H., VAROL, H. A. et GOLDFARB, M. (2011). Volitional control of a prosthetic knee using surface electromyography. *Biomedical Engineering, IEEE Transactions on*, 58, 144–151.

HAM, R. V., SUGAR, T., VANDERBORGHT, B., HOLLANDER, K. et LEFEBER, D. (2009). Compliant actuator designs. *Robotics & Automation Magazine, IEEE*, 16, 81–94.

HERR, H. M. et GRABOWSKI, A. M. (2012). Bionic ankle–foot prosthesis normalizes walking gait for persons with leg amputation. *Proceedings of the Royal Society B : Biological Sciences*, 279, 457–464.

HERR, H. M. et KORNBLUH, R. D. (2004). New horizons for orthotic and prosthetic technology : artificial muscle for ambulation. *Smart structures and materials*. International Society for Optics and Photonics, 1–9.

HOGAN, N. (1985). Impedance control. *Transaction of the ASME, Journal of Dynamic Systems, Measurement, and Control*, 107, 1–24.

HOWARD, M., BRAUN, D. J. et VIJAYAKUMAR, S. (2013). Transferring human impedance behaviour to heterogeneous variable impedance actuators. *IEEE Transactions on Robotics*, 29.

HUNTER, I. W., HOLLERBACH, J. M. et BALLANTYNE, J. (1991). A comparative analysis of actuator technologies for robotics. *Robotics Review*, 2.

JACOBSON, D. H. et MAYNE, D. Q. (1970). *Differential Dynamic Programming*. Elsevier.

JAFARI, A., TSAGARAKIS, N. et CALDWELL, D. (2013). A novel intrinsically energy efficient development of a novel actuator with adjustable stiffness (AwAS). *IEEE Transactions on Mechatronics*, 18.

JAFARI, A., TSAGARAKIS, N. G. et CALDWELL, D. G. (2011). Exploiting natural dynamics for energy minimization using an actuator with adjustable stiffness (awas). *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 4632–4637.

JAFARI, A., TSAGARAKIS, N. G., VANDERBORGHT, B. et CALDWELL, D. G. (2010). A novel actuator with adjustable stiffness (awas). *Intelligent Robots and Systems (IROS), 2010 IEEE/RSJ International Conference on*. IEEE, 4201–4206.

KANEKO, K., KANEHIRO, F., KAJITA, S., YOKOYAMA, K., AKACHI, K., KAWASAKI, T., OTA, S. et ISOZUMI, T. (2002). Design of prototype humanoid robotics platform for hrp. *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'00)*. Lausanne, Switzerland.

KAVRAKI, L. E., SVESTKA, P., LATOMBE, J.-C. et OVERMARS, M. H. (1996). Probabilistic roadmaps for path planning in high-dimensional configuration spaces. *Robotics and Automation, IEEE Transactions on*, 12, 566–580.

KHATIB, O. (1987). A unified approach for motion and force control of robot manipulators : The operational space formulation. *International Journal of Robotics Research*, 3, 43–53.

KREJIĆ, N., MARTíNEZ, J. M., MELLO, M. et PILOTTA, E. A. (2000). Validation of an augmented lagrangian algorithm with a gauss-newton hessian approximation using a set of hard-spheres problems. *Computational Optimization and Applications*, 16, 247–263.

KUIKEN, T. A., LI, G., LOCK, B. A., LIPSCHUTZ, R. D., MILLER, L. A., STUBBLEFIELD, K. A. et ENGLEHART, K. B. (2009). Targeted muscle reinnervation for real-time myoelectric control of multifunction artificial arms. *JAMA : the journal of the American Medical Association*, 301, 619–628.

LEBOEUF, F., BESSONNET, G., SEGUIN, P. et LACOUTURE, P. (2006). Energetic versus sthenic optimality criteria for gymnastic movement synthesis. *Multibody System Dynamics*, 16, 213–236.

LEINEWEBER, D. B., SCHÄFER, A., BOCK, H. G. et SCHLÖDER, J. P. (2003). An efficient multiple shooting based reduced sqp strategy for large-scale dynamic process optimization : Part ii : Software aspects and applications. *Computers & chemical engineering*, 27, 167–174.

MANSARD, N. (2012). A dedicated solver for fast operational-space inverse dynamics. *IEEE ICRA'12*. St Paul, USA.

MANSARD, N. (2013). Habilitation a diriger des recherches : Semiotics of motion : Toward a robotics programing language.

MANSARD, N., STASSE, O., CHAUMETTE, F. et YOKOI, K. (2007). Visually-guided grasping while walking on a humanoid robot. *IEEE Int. Conf. on Robotics and Automation (ICRA'07)*. Roma, Italia.

MARTINEZ-VILLALPANDO, E. C. et HERR, H. (2009). Agonist-antagonist active knee prosthesis : a preliminary study in level-ground walking. *J Rehabil Res Dev*, 46, 361–73.

MCCARTHY, J. (1990). *Introduction to Theoretical Kinematics*. MIT Press.

MOMBAUR, K. (2008). Using optimization to create self-stable human-like running. *Robotica*, 27, 321.

MORDATCH, I., TODOROV, E. et POPOVIĆ, Z. (2012). Discovery of complex behaviors through contact-invariant optimization. *ACM SIGGRAPH'12*. Los Angeles, USA.

NOCEDAL, J. et WRIGHT, S. J. (2006a). *Numerical Optimization*. Springer, New York, seconde édition.

NOCEDAL, J. et WRIGHT, S. J. (2006b). Sequential quadratic programming. *Numerical Optimization*, 529–562.

ORTEGA, R. et SPONG, M. W. (1989). Adaptive motion control of rigid robots : A tutorial. *Automatica*, 25, 877–888.

PANTOJA, D. O. (1988). Differential dynamic programming and newton's method. *International Journal of Control*, 47, 1539–1553.

PARK, J. (2006). *Control Strategies for Robots in Contact*. Thèse de doctorat, Stanford University, USA.

PARK, J. et KHATIB, O. (2006). Contact consistent control framework for humanoid robots. *IEEE Int. Conf. on Robotics and Automation (ICRA'06)*. Orlando, USA.

PRATT, G. et WILLIAMSON, M. (1995). Series elastic actuators. *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems (IROS'95)*. 399–406.

ROBINSON, D. W. (2000). *Design and analysis of series elasticity in closed-loop actuator force control*. Thèse de doctorat, Massachusetts Institute of Technology.

SAAB, L., RAMOS, O., MANSARD, N., SOUÈRES, P. et FOURQUET, J.-Y. (2013). Dynamic whole-body motion generation under rigid contacts and other unilateral constraints. *IEEE Transaction on Robotics*, 346–362.

SAMSON, C., LE BORGNE, M. et ESPIAU, B. (1991). *Robot Control : the Task Function Approach*. Clarendon Press, Oxford, United Kingdom.

SANCHEZ, E. N. et ORNELAS-TELLEZ, F. (2013). *Discrete-time inverse optimal control for nonlinear systems*.

SARDELLITTI, I., MEDRANO-CERDA, G., TSAGARAKIS, N., JAFARI, A. et CALDWELL, D. (2013). Gain scheduling control for a class of variable stiffness actuators based on lever mechanisms. *IEEE Trans. on Robotics*, 29, 791–798.

SCHULMAN, J., LEE, A., AWWAL, I., BRADLOW, H. et ABBEEL, P. (2013). Finding locally optimal, collision-free trajectories with sequential convex optimization. *Robotics : Science and Systems*.

SCHULZ, V. (1996). *Reduced SQP methods for large scale optimal control problems in DAE with application to path planning problems for satellite mounted robots*. Thèse de doctorat, Citeseer.

SENSINGER, J. W. et WEIR, R. F. (2008). User-modulated impedance control of a prosthetic elbow in unconstrained, perturbed motion. *Biomedical Engineering, IEEE Transactions on*, 55, 1043–1055.

SERIO, A., GRIOLI, G., SARDELLITTI, I., TSAGARAKIS, N. G. et BICCHI, A. (2011). A decoupled impedance observer for a variable stiffness robot. *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 5548–5553.

SHIN, D., SARDELLITTI, I., PARK, Y.-L., KHATIB, O. et CUTKOSKY, M. (2010). Design and control of a bio-inspired human-friendly robot. *The International Journal of Robotics Research*, 29, 571–584.

SICILIANO, B. et SLOTINE, J.-J. (1991). A general framework for managing multiple tasks in highly redundant robotic systems. *IEEE Int. Conf. on Advanced Robotics (ICAR'91)*. Pisa, Italy.

STOKES, I. A. et GARDNER-MORSE, M. (2003). Spinal stiffness increases with axial load : another stabilizing consequence of muscle action. *Journal of electromyography and kinesiology*, 13, 397–402.

TASSA, Y., EREZ, T. et TODOROV, E. (2012a). Synthesis and stabilization of complex behaviors through online trajectory optimization. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS'12)*. 4906–4913.

TASSA, Y., EREZ, T. et TODOROV, E. (2012b). Synthesis and stabilization of complex behaviors through online trajectory optimization. *(IROS'12)*. Portugal.

TASSA, Y., MANSARD, N. et TODOROV, E. (2014). Control-limited differential dynamic programming. *Under Review*.

TASSA, Y., WU, T., MOVELLAN, J. et TODOROV, E. (2013a). Modeling and identification of pneumatic actuators.

TASSA, Y., WU, T., MOVELLAN, J. et TODOROV, E. (2013b). Modeling and identification of pneumatic actuators. *IEEE International Conference Mechatronics and Automation*.

TODOROV, E. et LI, W. (2005). A generalized iterative LQG method for locally-optimal feedback control of constrained nonlinear stochastic systems. *Proceedings of the American Control Conference (ACC'05)*. Portland, OR, USA, 300–306.

TONDU, B. et LOPEZ, P. (2000). Modeling and control of mckibben artificial muscle robot actuators. *IEEE Control Systems*, 20.

TONIETTI, G., SCHIAVI, R. et BICCHI, A. (2005). Design and control of a variable stiffness actuator for safe and fast physical human/robot interaction. *Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on*. IEEE, 526–531.

TSAGARAKIS, N., LI, Z., SAGLIA, J. et CALDWELL, D. (2011). The design of the lower body of the compliant humanoid robot iCub. *IEEE Int. Conf. on Robotics and Automation (ICRA'11)*. Shangai, China.

VAN HAM, R., VANDERBORGHT, B., VAN DAMME, M., VERRELST, B. et LEFEBER, D. (2007). Maccepa, the mechanically adjustable compliance and controllable equilibrium position actuator : Design and implementation in a biped robot. *Robotics and Autonomous Systems*, 55, 761–768.

VANDERBORGHT, B., TSAGARAKIS, N. G., SEMINI, C., VAN HAM, R. et CALDWELL, D. G. (2009). Maccepa 2.0 : Adjustable compliant actuator with stiffening characteristic for energy efficient hopping. *Robotics and Automation, 2009. ICRA'09. IEEE International Conference on*. IEEE, 544–549.

VANDERBORGHT B., TSAGARAKIS N., V. H. R. T. I. C. D. (2011). MACCEPA 2.0 : Compliant actuator used for energy efficient hopping robot chobino1d. *Autonomous Robots*, 31, 55–65.

VERSLUYS, R., BEYL, P., VAN DAMME, M., DESOMER, A., VAN HAM, R. et LEFEBER, D. (2009). Prosthetic feet : State-of-the-art review and the importance of mimicking human ankle-foot biomechanics. *Disability & Rehabilitation : Assistive Technology*, 4, 65–75.

WHITNEY, D. (1969). Resolved motion rate control of manipulators and human prostheses. *IEEE Transactions on Man-Machine Systems*, 10, 47–53.

WOLF, S. et HIRZINGER, G. (2008). A new variable stiffness design : Matching requirements of the next robot generation. *Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on*. IEEE, 1741–1746.

ZEFRAN, M. (1996). Continuous methods for motion planning. *IRCS Technical Reports Series*, 111.