Commande de chute pour robots humanoïdes par reconfiguration posturale et compliance adaptative

Vincent Samy

THÈSE

Pour obtenir le grade de

Docteur

Délivré par **l'Université de Montpellier**

Préparée au sein de l'école doctorale I2S – Information, Structure, Systèmes
Et de l'unité de recherche UMR 5506

Spécialité : **SYAM – Systèmes Automatiques et Microélectroniques**

Présentée par **Vincent SAMY**

Commande de chute pour robots humanoïdes par reconfiguration posturale et compliance adaptative

Soutenue le 13 Novembre 2017 devant le jury composé de

M. Vincent PADOIS	Maître de Conférences	ISIR	Rapporteur
M. Olivier STASSE	Directeur de Recherche	LAAS	Rapporteur
Mme Christine CHEVALLEREAU	Directeur de Recherche	IRCCyN	Examinateur
M. Shuuji KAJITA	Directeur de Recherche	AIST	Examinateur
M. Karim BOUYARMANE	Maître de Conférences	LORIA	Co-encadrant de thèse
M. Abderrahmane KHEDDAR	Directeur de Recherche	LIRMM	Directeur de thèse

Titre : Commande de chute pour robots humanoïdes par reconfiguration posturale et compliance adaptative

Résumé :

Cette thèse traite du problème de la chute de robots humanoïdes. L'étude consiste à découpler la stratégie de chute en une phase de pré-impact et une phase de post-impact. Dans la première, une solution géométrique permet au robot de choisir des points d'impact dans un environnement encombré. Le robot réadapte sa posture tout en évitant les singularités de chute et en préparant le seconde phase. La phase de post-impact utilise une commande par Programmation Quadratique (QP) qui permet d'adapter les gains Proportionnels-Dérivés (PD) des moteurs en ligne, afin d'atteindre une compliance dans les articulations et amortir activement le choc. L'approche consiste à incorporer les gains de raideur et d'amortissement dans le vecteur d'optimisation du QP avec les variables habituelles que sont l'accélération articulaire et les forces de contact. Enfin, pour compléter cette approche locale et en augmenter les performances, une commande de modèle prédictif sur un modèle simplifié du robot et son évolution sur une fenêtre glissante de temps est incorporée à la méthode. Plusieurs expériences et simulations sont présentées pour valider les différentes parties de cette thèse.

Mots-clefs : Robotique Humanoïde ; Chute ; QP Adaptatif ; Compliance Active.

Title: Humanoid fall control by postural reshaping and adaptive compliance

Abstract:

This thesis deals with the problem of humanoid falling with a decoupled strategy consisting of a pre-impact and a post-impact stages. In the pre-impact stage, geometrical reasoning allows the robot to choose appropriate impact points in the surrounding environment and to adopt a posture to reach them while avoiding impact singularities and preparing for the post-impact. The post-impact stage uses a quadratic program controller (QP) that adapts on-line the joint proportional-derivative (PD) gains to make the robot compliant, i.e. to absorb post-impact dynamics, which lowers possible damage risks. We propose a new approach incorporating the stiffness and damping gains directly as decision variables in the QP along with the usually-considered variables that are the joint accelerations and contact forces. Finally, to overcome the limitations of the local QP approach, we combine the method with a Model Predictive Controller (MPC) allowing for a preview on a time-horizon based on a reduced model. Several experiments on the real humanoid robot HRP-4 or in a full-dynamics simulator are presented and discussed to illustrate and validate each part of the thesis.

Keywords: Humanoid Robot ; Fall ; Adaptive QP; Active compliance.

Discipline : Systèmes Avancés et Microélectronique

Laboratoire d'Informatique, de Robotique et de Microélectronique de Montpellier
UMR 5506 CNRS/Université de Montpellier
Batiment 5 - 860 rue de St Priest

Résumé de la thèse

La robotique humanoïde est devenue un secteur de recherche prépondérant. Dans un monde en constante évolution, la robotique humanoïde est une exellente alternative à l'homme lorsque les situations sont dangereuses (intervention dans les centrales nucléaires), et elle est habilité à naviguer dans ces lieux sans avoir à les modifier. Ces environnements dans lesquels les robots sont de plus en plus amenés à évoluer ont été créés par l'homme pour l'homme, il apparaît ainsi naturel que le système le plus apte à se mouvoir dans cet univers, et interagir avec, soit de forme humanoïde. Cependant, la complexité de tels systèmes nécessite un effort important de recherche et de développement. Les difficultés techniques tournant autour de la robotique humanoïde sont multiples, les sujets à traiter sont donc nombreux, ils abordent des problèmatiques comme l'aptitude d'un robot à marcher, ouvrir une porte, monter des marches, monter une échelle, etc... C'est l'aboutissement de ces techniques qui permettront dans un futur proche d'avoir des robots avec une motricité permettant de réaliser des gestes identiques à l'homme.

En 2015, le DARPA Robotics Challenge (DRC) a permis de montrer les capacités de tels robots dans un milieu humain. On peut de plus noter que certains robots sont actuellement au contact d'humains (Nao, Asimo, ...). Cela montre que ces robots sont de plus en plus robustes et aptes à réaliser les tâches qu'on attend d'eux. C ependant, le DRC a également révélé que certains cas de figure n'ont pas fait l'objet d'études approfondies : comme par exemple la chute du robot qui était l'incident le plus récurrent . Il s'agissait d'un des rares cas d'application réelle où les robots étaient en dehors du laboratoire et n'étaient plus attachés à leur cordon de sécurité, ce qui a amplifié l'importance du phénomène. Lorsqu'un robot tombait, aucune stratégie de chute, aucun mouvement dédié à l'amortir ou l'éviter, n'était visible sur les robots, ce qui signifie que ce genre de situation n'a pas été pris en compte dans la commande du robot.

Le sujet de cette thèse s'inscrit suite à cette observation, *i.e.* comment agir lorsque la commande du robot n'est plus à même de satisfaire sa stabilité ou son équilibre, statique ou dynamique ? Une première approche en trois étapes se dégage naturellement:

(a) tenter de se ressaisir pendant une chute ;

(b) si la chute ne peut être évitée en (a), ou, si pour certaines raisons, le robot doit tomber volontairement, réduire les potentiels dégâts dûs à l'impact ;

(c) si aucune solution n'est possible, le robot devrait porter son propre système de sécurité (airbag, carapace compliante, etc....).

Cette thèse s'intéresse principalement à l'étape (b). Elle présente une nouvelle manière d'aborder le sujet de la chute et valide les théories développées par des simulations et/ou des expériences.

État de l'art et motivations

Le chapitre 1 se concentre sur la chute de robots humanoïdes dans la littérature scientifique et explore les solutions existantes. La recherche dans ce domaine étant à son commencement, le nombre d'articles le traitant est relativement réduit. Cependant, il a été possible de dégager une répartition des différentes solutions et leur classification en fonction des types de chute et de l'environnent.

Avant toute chose, pour contrôler un comportement lors d'une chute il faut définir ce qu'est une chute et comment la détecter. La définition même de la chute est une notion difficile à formaliser, une des plus proches définitions est celle donnée par Chang et al. [2013] : "It is clear that 'expectancy' is required to walk safely. There will be a motion perturbation if expectation and reality do not match" (Il est évident que 'l´anticipation' est nécessaire pour une marche sûre. Il y aura une perturbation dans le mouvement si ce qui est attendu ne coïncide pas avec la réalité). Elle est cependant mathématiquement complexe à formaliser et demande de pouvoir corréler correctement mesures et sorties de contrôleur.

Dans la littérature, les détections de chutes se font, par exemple, via des seuils angulaires à ne pas dépasser. Pour la marche, la projection du Centre de Masse sur le polygone de sustentation est un bon critère mais ne fonctionne que pour des marches quasi-statiques. Ogata et al. [2008] ont donc proposé une méthode basée sur la prédiction du Zero-Moment Point (ZMP). Celui-ci n'est fonctionnel que pour de la marche et l'idée doit être étendue pour des mouvements plus généraux (manipulation avec les bras, mouvements multi-contacts,...). Enfin il est notable qu'une méthode basée sur de l'apprentissage a été également proposée par Ogata et al. [2007].

Une fois la chute détectée, intervient alors le contrôleur. Deux types de contrôleurs se distinguent. Tout d'abord ceux où le robot cherche à éviter de tomber sur quelqu'un/-quelque chose. Ce sont notamment les travaux de Goswami et ses collègues. Goswami et al. [2014] utilisent des méthodes de reconfiguration de l'inertie de différentes manières (*cf.* Lee et Goswami [2013], Lee et Goswami [2007], Lee et Goswami [2013]). Le deuxième type de contrôleurs représente les études de minimisation des dégâts dûs aux chutes. Cette minimization peut être réalisée de manière passive (par exemple, Kajita et al. [2016] utilise des airbags), ou active (c'est le cas avec Lee et al. [2016a]). Il est possible de reconfigurer le robot durant la chute pour minimiser ses dégâts potentiels.

Cela s'effectue grâce à une technique bien connue dans des sports tels que le judo et l'aïkido : le Ukemi. Les algorithmes développés par Fujiware *et al.* permettent aux robots des chutes en avant et en arrière plus sûres. Enfin, Ha et Liu [2015] présentent un séquenceur de contacts pour des chutes en avant et Yun et Goswami [2014] proposent la méthode du trépied pour casser la chute le plus tôt possible.

La plupart des commandes en vue de réduire les dégâts de chute n'étudient que des chutes en avant et en arrière. Dans cette thèse, des environnements plus complexes et des chutes omnidirectionnelles seront considérés.

Chutes singulières

Nous commençons les contributions de cette thèse en prenant du recul par rapport à ce qui a été fait dans l'état de l'art, ce qui nous a permis de proposer plusieurs nouvelles idées. La première que nous proposons est celle des chutes dites singulières présentée dans le chapitre 2. Basée sur une observation de l'homme, une taxonomie des chutes les plus dangereuses a été extraite, nommées chutes singulières. Ce genre de chutes augmente le risque de destruction du squelette (humain ou robotique). Pour expliquer ici de manière simple, une chute singulière arrive lorsque, à l'impact, la force de réaction du sol (généralement normale au sol) est perpendiculaire à un axe de moteur. En multi-contact, il est également possible que ce type de chutes apparaisse suite à la fermeture d'une boucle cinématique. Une taxonomie de ces chutes est alors proposée.

Suite à cela une commande essentiellement basée sur la géométrie du robot et de l'environnement a permis de réaliser des simulations et expériences sur le robot HRP-4. Pour éviter de détruire les moteurs suite à l'impact de la chute, les gains Proportional-Derivative (PD) des moteurs ont sont volontairement diminués. Cela a non seulement permis de garder les moteurs indemnes après la chute, mais aussi de réaliser un comportement compliant du système. Les expériences sur la palteforme HRP-4 du LIRMM ont été effectuées pour des chutes en avant et en arrière sur un matelas standard.

QP-adaptatif et environnement encombré

L'étape suivante a été d'améliorer la modification des gains des moteurs pour aboutir à un comportement adaptatif (qui dépend donc de la chute et de son intensité) et ce, en ligne. Nous ne nous restreignons pas à des chutes en avant ou en arrière, en proposant des stratégies plus générales qui peuvent gérer des chutes dans n'importe quelle direction dans un environnement non vide (mais sans hommes). Le chapitre 3 présente les travaux effectués sur ces deux points.

Le phénomène problématique au cœur d'une chute est l'impact. Mathématiquement, un impact est un phénomène instantané, ce qui le rend incontrôlable directement. Il n'est donc possible d'agir qu'avant et après l'instant précis de l'impact. La chute a donc été séparée en deux phases. Une phase dite de pré-impact qui comprend toute la

partie entre la détection de la chute et l'impact et une phase dite de post-impact qui comprend la partie entre l'impact et l'arrêt total du robot à une position de repos (par arrêt, il est signifié ici que le robot atteint une vitesse nulle et non que celui-ci cesse de fonctionner).

La phase de pré-impact est cruciale quant à la réalisation de la phase de post-impact. En effet, elle va préparer le robot à l'impact et lui permettre d'annuler la vitesse accumulée pendant la chute. Dans un environnement 3D encombré, il est nécessaire de faire des choix pour le positionnement des mains du robot. La phase de pré-impact est elle-même subdivisée en trois étapes qui sont 1) la détection de chute ; 2) L'analyse de l'environnement et des surfaces potentielles sur lesquelles s'appuyer ; 3) le contrôle de la chute. Cette dernière étape combine trois actions :

(i) L'estimation de la direction de chute,

(ii) La recherche des points d'impact,

(iii) La mise à jour des tâches de chute.

La recherche des points d'impact est une partie délicate car le temps d'analyse et de décision doit être bref. En effet, la phase de pré-impact dure de 0.7 s à 1 s ce qui rend les algorithmes de planification courants quasi-inutilisables. La méthode que nous proposons répond à cette problématique en restant rapide et efficace. Elle peut au besoin être parallélisée (multi-thread) si l'environnement comporte un grand nombre de surfaces. La Fig. 3.2 illustre la méthode proposée. L'idée est de considérer le robot comme un objet non-actionné tombant. L'avantage d'un objet non-actionné est qu'il est rapide de calculer sa trajectoire (un arc de cercle dans notre cas). Il est alors possible de rechercher l'intersection des mains du robot (dans une configuration prédéfinie) avec tous les plans définis par les surfaces de contact potentielles. Cette intersection est appelée un Most Probable Impact Point (MPIP) (point d'impact le plus probable). La projection de ce point sur la surface réelle est appelée le Best Impact Point (BIP) (le meilleur point d'impact). Ensuite, il ne reste plus à l'algorithme qu'à choisir le meilleur des BIP parmi tous les points. Le meilleur étant le point le point le plus élevé (coordonnée verticale) et inclus dans l'espace de travail du bras du robot.

Ces étapes sont effectuées à chaque itération de la boucle de contrôle du robot (toutes les 5 ms sans le cas d'HRP-4) et doivent aussi prendre en compte les limites articulaires, de couple et de vitesse angulaire du robot ainsi que les singularités de chute précédemment identifiées.

La phase de post-impact, elle, permet la modification des gains en ligne. Une analyse sur un système à 1 degré de liberté à montré l'intérêt de rechercher des gains variables dans le temps (trajectoire dans le plan des gains), plutôt que d'en calculer une valeur unique optimale qui restera fixe. Ainsi, un Quadratic Programming (QP) adaptatif a été conçu pour rechercher les gains adéquats à modifier dans les moteurs pour atteindre un comportement compliant. Le QP est une reformulation des QP standards utilisés en robotique humanoïde. L'idée nouvelle ici est d'inclure les gains K et B en tant que

variable du vecteur d'optimisation (qui comportent traditionnellement l'accélération articulaire et les forces de contact).

La méthode a été validée dans le simulateur physique Gazebo et a permis de constater l'efficacité du QP-adaptatif dans le cas de chutes dans différents environnements et pouvant avoir de grandes vitesses à l'impact.

Commande prédictive basée polytope pour la réalisation de la compliance

Un des problèmes rencontrés avec le QP-adaptatif est l'impossibilité de savoir si les gains modifiés seront suffisants pour amener le robot au repos sur le long terme (quelques secondes après impact), le QP ne raisonnant que de manière locale sur une fenêtre de quelques millisecondes. La solution proposée dans la méthode précédente a consisté à augmenter les poids des tâches pour assurer l'arrêt du robot. Dans le chapitre 4, une méthode pour prédire une trajectoire du CoM a été conçue.

L'idée centrale et novatrice de cette méthode est la séparation de de chaque membres (bras et jambes) dans l'équation du mouvement (Eq. (4.3)). Il s'agit alors d'assigner de manière optimale à chacun de ces membres une partie de l'effort à fournir pour annuler la vitesse post-impact.

Pour chaque membre (bras, jambes), il est possible d'écrire la relation entre le couple articulaire et force que celui-ci exerce sur un point de contact situé au bout du membre (mains, pieds). Il est alors possible d'utiliser des polytopes représentant les couples minimum et maximum de chaque moteur pour représenter les forces minimum et maximum atteignables au point de contact. Ce polytope est alors intersecté avec le cône de friction pour recouvrir l'ensemble des forces applicables. Cette information va permettre de fournir des bornes en couple sur l'algorithme de prédiction.

Il faut maintenant assigner à chaque membre l'effort qu'il doit compenser. Pour ce faire, un nouveau QP appelé Force Distribution Quadratic Programming (FDQP) pour QP de distribution des efforts a été imaginé. Le principe est le suivant. le FDQP va répartir la force de gravité et la quantité de mouvement dans chaque membre en se basant sur les capacités de ceux-ci (fournies par les polytopes précédemment calculés).

Le résultat du FDQP permet de récupérer une quantité de mouvement à annuler et un effort à compenser. Un Linear Model Predictive Control (LMPC) est attribué à chaque membre et calcule la force à fournir pour atteindre une vitesse nulle sur un horizon de temps donné. Les solutions pour les différents membres sont ensuite recombinées en une seule équation au niveau du CoM, ce qui permet de calculer une trajectoire désirée du CoM.

La trajectoire est alors envoyée dans le QP-adaptatif formulé dans la méthode précédente, qui recherche les gains pour accomplir la trajectoire désirée. Les simulations dynamiques montrent l'efficacité de la méthode avec une chute sur un mur. Dans cette situation le robot a 4 points de contacts à l'impact (les deux mains et deux pieds).

Conclusion

Dans cette thèse, de nouvelles stratégies de chute ont été développées. Diverses simulations et expériences ont montré l'efficacité de ces méthodes. Cependant, tester de tels algorithmes en situations réelles reste dangereux et potentiellement coûteux pour le robot. Une amélioration des divers algorithmes de détection de chute, de détection de contact, etc.... est nécessaire, et constitue les travaux futurs à la suite de cette thèse.

Contents

List of Figures

Nomenclature

This nomenclature stands for all the thesis but Chapter 1 due to convention mismatches. **Bold symbols** are for vectors.

Acronyms

$\mathcal{H}$-representation	Intersection of finite halfspaces polytope representation
$\mathcal{V}$-representation	Convex hull of finite set of vertices polytope representation
BF	Back Fall
BIP	Best Impact Point
CoM	Center of Mass
CoP	Center of Pressure
CPU	Central Processing Unit
CRB	Composite Rigid Body
DoF	Degrees of Freedom
DRC	DARPA Robotics Challenge
FDQP	Force Distribution Quadratic Programming
FF	Front Fall
LIP	Linear Inverted Pendulum
LMPC	Linear Model Predictive Control
MPIP	Most Probable Impact Point
PD	Proportional-Derivative
QP	Quadratic Programming
SF	Side Fall
ZMP	Zero-Moment Point

List of symbols

B	Diagonal damping matrix
E	QP task

G	Matrix of friction cone generators
H	Inertia matrix
J	Jacobian matrix
K	Diagonal stiffness matrix
M	Total mass of the system
N_{dof}	Number of dof in the system
N_c	Number of contact points
N_g	Number of generators of the linearized friction cone
N_m	Number of motors
N_{cpu}	Number of CPUs
S_M	Selection matrix
Λ	Operational space inertia matrix
α	Positive coefficient of polytope $\mathcal{V}$-representation
$\dot{\boldsymbol{\vartheta}}$	Vector of task velocity error
$\boldsymbol{\lambda}$	Contact force magnitude vector
$\boldsymbol{\omega}\times$	Skew-symmetric matrix associated to $\boldsymbol{\omega}$
$\boldsymbol{\tau}$	Generalized force vector
$\boldsymbol{\vartheta}$	Vector of task error
$\dot{J}$	Time-derivative Jacobian matrix
$\dot{\hat{J}}$	Time-derivative task Jacobian
$\hat{J}$	Task Jacobian
$\mathbf{F}^{\text{ext}}$	External force applied to the system
$\mathbf{F}$	Net contact force
$\mathbf{X}$	QP decision vector
$\ddot{\mathbf{q}}$	Generalized acceleration vector
$\ddot{\mathbf{s}}$	Acceleration of the CoM
$\dot{\mathbf{e}}$	Joint velocity error vector
$\dot{\mathbf{q}}^{\text{ref}}$	Reference of the generalized velocity vector
$\dot{\mathbf{q}}$	Generalized velocity vector
$\dot{\mathbf{s}}$	Velocity of the CoM
$\mathbf{c}$	Coriolis and gravity vector
$\mathbf{d}^f$	Fall direction vector
$\mathbf{d}^t$	Torso yaw vector projected on the ground
$\mathbf{e}$	Joint position error vector
$\mathbf{g}$	Gravity vector
$\mathbf{p}$	Linear momentum at the the CoM
$\mathbf{q}^{\text{ref}}$	Reference of the generalized position vector
$\mathbf{q}$	Generalized position vector
$\mathbf{s}$	Position of the CoM
$\mathbf{v}$	Vertex of a polytope
ω	Task weight
NS	Subscript for matrix/vector of non-selected rows

S	Subscript for matrix/vector of selected rows
k_p	Task stiffness
k_v	Task damping gain
${}^{B}E_{A}$	Rotation matrix from A to B coordinates
${}^{B}X_{A}$	General transformation matrix from A to B coordinates
${}^{B}\mathbf{r}_{A}$	Translation vector from A to B in A coordinates

Introduction

Humanoid robots have come to a new era of development. There is a strong desire for making robots collaborate with humans in a close way. The robots have to navigate and move around in the human environment, interact with it, understand the needs of the humans, communicate with them, and interact with them. The surge in the development of humanoid robotics can be attributed to the simple following intuition: what is better than a human-like system to realize all these new human-centered challenges?

Recently, humanoid robots have been considered (i) for rescue and intervention in disaster situationsKakiuchi et al. [2017]; (ii) as home service companions to assist frail and aging people[1]; and (iii) as collaborative workers (i.e. as cobots termed "comanoids") in large-manufacturing assembly plants[2] where wheeled and rail-ported robots cannot be used (e.g. aircrafts and shipyards), among other applications. These three example applications have different social and economic impacts, different business models, but they also have different requirements in terms of hardware, perception capabilities, and dexterity. Humanoids are highly complex systems and stabilizing them is still a big challenge. Therefore, a rising number of works study the balance and stabilizing problem of humanoids, but few consider what should the robot do when their stabilizing control is undermined.

As for any other robotic systems, humanoid robots shall preserve first the human integrity, secondly their surrounding environment integrity, and finally their own integrity. This is a well-know behavior formulated in the iconic so-called Asimov's laws:

1. 'Law 1: A robot may not injure a human being or, through inaction, allow a human being to come to harm'

2. 'Law 2: A robot must obey the orders given it by human beings except where such orders would conflict with the First Law'

3. 'Law 3: A robot must protect its own existence as long as such protection does not conflict with the First or Second Law'

[1] www.projetromeo.com/en
[2] www.comanoid.eu

Law 1 could be extended with, 'A robot must avoid objects that are high-value objects' (for example a 3000 years old vase, with a high economic and cultural value). As for our concern in this thesis, these laws should also be enforced in the robot controller when falling.

In 2014, the DARPA Robotics Challenge (DRC)[3] showed how well humanoid robots are able to realize human-like manipulation and locomotion tasks, but also how much they struggled to this end. Different tasks were assigned to the robots, among them driving, opening a door and crossing the threshold, climbing stairs, navigating in a cluttered environment and drilling a hole. The challenge showed that robots with humanoid morphology were a good match for the problem, since some of them manage to realize all the tasks. Thanks to the humanoid technology, robots are getting closer to humans in terms of these capabilities. However, the challenge also revealed an overlooked behavior and a big hole, or missing component, in the controllers of humanoid robots: this overlooked behavior is the fall. Indeed, not a single robot had a strategy when loosing balance.

Humans are, from our scientific understanding and our technological advances perspective, one of the most sophisticated known systems, they are capable of planning their movement, walking and running in cluttered environment, etc... and they do it without even thinking about it. Nevertheless, they also still fall. Therefore, it is not irrelevant to consider that (i) even if the environment is well structured and even if we devote advanced strategies to walking, a humanoid robot will fall; and (ii) we are not able to list all the possible cases and situations where this will occur. This thesis tries to uncover what robot "can, should, and must" do when there is no other choice than falling.

A general common sense approach that accounts for the humanoid falling event would ideally operate as follows:

(a) devise strategies to avoid falling in the first place;

(b) if falling cannot be avoided in (a), or, for some reasons the robot must fall on purpose, then, if the robot is in servo-on, reduce as much as possible the damage resulting from the fall;

(c) when the two previous solutions are not applicable, i.e. if the robot is no more under control, it is better to simply resort to an extra shock absorbing system, such as an airbag, that can be triggered independently from the robot embedded control board.

Chapter 1 lays down some strategies that have been developed in the humanoid robotics research community to prevent damage from falling. As there are relatively few papers that deal with humanoid falls, all the strategies are reviewed in detail. Many of these strategies came up with the idea of *Ukemi* motion, which is the term

[3] https://www.darpa.mil/program/darpa-robotics-challenge

used by Judo and Aikido. *Ukemi* is a controled break-fall. In other words, it designates a technique which allows to control the fall to minimize or to cancel impact damages.

Humans have developed several techniques through the years to find the best reception techniques when falling. It is thus interesting to think here that humanoid robot falling strategies have been inspired by human's martial art sports. However other considerations have to be taken into account, e.g. avoiding humans when falling, which makes the fall riskier for the robot but safer for the human.

We decided to take a different perspective from the current state of the art in order to bring out new ideas on the subject. As it is difficult to precisely know what are the good postures to reach when falling, Chapter 2 considers the opposite problem, and seeks instead the postures that are dangerous for the robot. Knowing them, control tools can be designed to prevent bad fall landing.

Until now, most of the strategies focused and restricted on specific types of falls: Front Fall (FF) and Back Fall (BF). In this thesis however, we consider that a fall in any direction must be treated. Chapter 3 addresses falls in any direction. We have also decided to challenge another, never-treated-before, difficulty of the problem: the fall shall occur in a cluttered environment. To decrease the damages taken by the robot, the fall is split into two separate phases which are the phase before the impact and the phase after the impact. This separation allows us to prepare the robot for the impact and to cancel the remaining velocity after it. This cancellation is made through a Quadratic Programming (QP) and an on-line adaption of the motors' Proportional-Derivative (PD) gains.

As a QP is controlling the robot and due to its local nature, it can not foresee if, on the long term, the computed gains will be high enough to stop the robot motion completely before it reaches its limits. Chapter 4 thus proposes a way to predict what will happen on a longer horizon of time and to decide a trajectory for the Center of Mass (CoM) that stops the robot at the right time.

All the contributions of this thesis have been validating either by a simulation or by some experiments on the HRP-4 robot platform.

CHAPTER 1

State of the art and motivations

Introduction

Humanoid robots are an imitation of the humans. Even if human motor control is infinitely more complex and more advanced than whatever the state of the art in humanoid robotics can offer to this day, humans themselves still fall on occasion Lortie et Rizzo [1999]. Moreover, the DARPA Robotics Challenge (DRC), that took place between the years 2012 and 2015, has shown that falls do occur at a very high rate when the robots are not anymore attached to security cords Atkeson et al. [2015] Atkeson et al. [2016], even under the close supervision of humans and in partial autonomy conditions. The challenge also shed the light on the fact that no fall controls were used when the robots were falling. The motivation for developing such algorithms is, therefore, evident in this context.

When considering falls, an important question is "How the fall has occurred and in which conditions?". Indeed, existing control algorithms assume that the robot is pushed when it is in a vertical upright stationary configuration, and that the push is done at an upper point of the torso in the saggital or the coronal plane. For instance, Ogata et al. [2007] have considered this kind of push on a walking robot to validate their control law. However, in general, the robustness of a fall control law should be tested and demonstrated with different kinds of pushes, applied at different points on the robot. The robot may also slip on a surface or trip on an object. These two situations have been barely considered in the literature, in part due to the fact that the experimentation in this domain is risky, costly, and dangerous. Another major difficulty in these general cases is getting the information about the contact points that are lost. Closed-loop control with contact estimation is a strict necessity in these situations.

To satisfy Asimov-s laws, we first need to define what is a **fall**. Humans can 'sense'

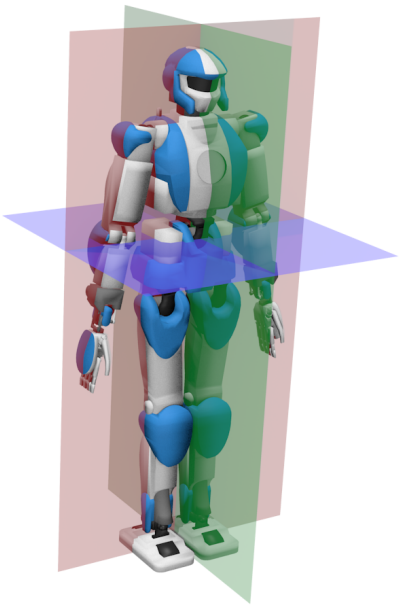


Figure 1.1 – Main planes of a humanoid robot. The green plane is called sagittal plane, the red one coronal plane and the blue one transverse plane

when they are just about to fall, then, and depending of their reaction speed, they move their body to either avoid someone or/and something during the fall, or if nothing is around try to break the fall. Thus, to activate the falling control algorithm, the robot needs to detect its fall. This part is considered in Section 1.3. Once the robot has detected its own fall, it has roughly about $0.7 - 1$ s to actuate its body to avoid to hurt and/or break something on it. Goswami et al. [2014] have presented change-of-direction algorithms during falls which are presented in Section 1.4. Breaking falls has been studied by Fujiwara *et al.* Fujiwara et al. [2007] using offline optimization tools, and Ogata *et al.* Ogata et al. [2008] who simplified the problem to obtain an online control algorithm.

Note that in this chapter, in order to not rewrite everything (because the authors do not have the same convention), the equation has been taken as is without being transformed to match the nomenclature.

Humanoid robot presentation

Humanoid robots are human-size and human-shaped robots. In this thesis, we consider that the robot has two distinct legs and two distinct arms attached to a torso and a waist respectively. We also recall the three main planes of humans (see Fig. 1.1). The platform used for the experiments is HRP-4 from Kawada. It is a relatively 'fragile' robot that has not been primarily designed to resist impacts and falls in general. At low level, it is a position-based controller that commands the motors.

We recall here the main equation of motion that described the robot

$$H\ddot{\mathbf{q}} + \mathbf{c} = J_{\text{concac}}^{\mathsf{T}} G\boldsymbol{\lambda} + \boldsymbol{\tau} \tag{1.1}$$

with H the inertia matrix, $\ddot{\mathbf{q}}$ the generalized acceleration vector, $\mathbf{c}$ the Coriolis and gravity forces, J_{concac} the concatenation of the Jacobian matrix of all contacts, G the matrix of friction cone generator, $\boldsymbol{\lambda}$ the contact force magnitude vector and $\boldsymbol{\tau}$ the generalized force vector.

We also recall the centroidal dynamics

$$\mathbf{F} = M(\ddot{\mathbf{s}} - \mathbf{g}) \tag{1.2}$$

with $\mathbf{F}$ the net force, M the total mass of the robot, $\ddot{\mathbf{s}}$ the acceleration of the Center of Mass (CoM) and $\mathbf{g}$ the gravity vector.

Algorithms developed in this thesis have been designed to deal with humanoid robots but most of them can be directly (or with some extensions) exploited on other types of legged robots.

Fall detection

Because a fall is a matter of only a few milliseconds, it is admitted that the sooner the fall is detected the more efficient the control law will be. Also, falling detection must happen only when the robot is in a situation in which it cannot avoid the fall. In another words, if it has been pushed, but can still maintain its stability by performing a fall-avoidance algorithm (Ogata et al. [2007] and Sugihara [2008] for example), then fall should not be detected and the fall-avoidance algorithm should be executed first. Therefore, the falling detection algorithm should not only be fast, but also robust, as it should not detect "false positives", nor trigger "false alarms".

Limit angle

As a first and very simple way to detect a fall, Lee et Goswami [2013] propose to consider the angle α between the *lean line* and the normal of the ground. The lean line is the line going through both the Center of Pressure (CoP) and the CoM as represented in Fig. 1.2.

The reported time delay before detection of the fall is about 30 ms with a push of 160 N. This kind of fall detection has the advantage of simplicity and quasi-instantaneous computation, but it also has a major limitation. That is, the robot could be stable even with $\alpha > 15°$. For example, the robots could be lying down on a couch or it could be leaning on one of its arms. Another possibility is a non planar ground, making the normal vector computation wrong. In the latter case, one could think of taking the angle between the gravity direction and the lean line, but in the same way, if the robot is standing on an inclined ground α would be greater than 15°. Finally, increasing the limit angle would make the robot react slower which is not advisable.

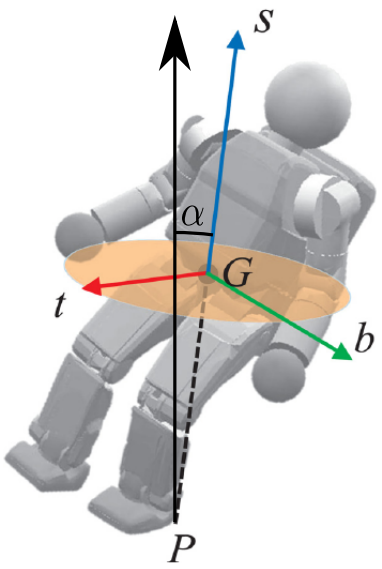


Figure 1.2 – Fall detection using the lean line. The lean line is the crossing of the CoP (point P) and the CoM (point G). The angle α is the angle between the ground normal and the lean line. Here, $\{\mathbf{t},\mathbf{b},\mathbf{s}\}$ is a frame attached to the CoM. Image edited from Lee et Goswami [2013].

Projection of the CoM

Another simple way of detecting a fall is the projection of the CoM on the ground. Fujiwara et al. [2004] suggests that idea. In a standing position, the CoM projection is inside the convex hull of the soles making the robot stable. When a push occurs on the torso, the robot bends moving the projection of the CoM outside the convex hull. Then, the robot is considered as falling and the falling control system is triggered. By looking closely, this solution amounts to the same as the limit angle fall detector (Section 1.3.1). Indeed, α and the CoM projection on the ground are related by

$$\cos(\alpha) = \frac{G_{proj}}{d} \tag{1.3}$$

where G_{proj} is the projection of the CoM on the ground and d the length of the vector $\mathbf{PG}$. This kind of detector is working for quasi-static behavior but may trigger false-positive when dynamics is considered.

These two detection methods have an important drawback: they do not take into account the possibility that the robot can try to stabilize itself.

Abnormality Detection Method

This method has been developed by Ogata et al. [2007]. It tries to make a difference between two states called *normal state* and *Unusual state*. The normal state is defined as "the state in which humanoid robots walk stably". The unusual state is defined as "the statFe resulting from a disturbance". To make the difference between those two

states, the former state has to be quantified. To do so, sensors data have been collected when the robot is performing a walk and the results have been put in the vector

$$\mathbf{s}^i = \left[s_1^i(k)\ s_2^i(k)\ ...\ s_{ns}^i(k)\right]^\mathsf{T} \tag{1.4}$$

where $\mathbf{s}$ is the sensor datas in a certain step i for one walking cycle, ns is the number of sensors, i represents the step time and k the number of samples. Then, the mean of the normal state becomes

$$\overline{\mathbf{s}}^i = \underset{k}{E}[\mathbf{s}^i(k)]. \tag{1.5}$$

which allows for the deviation to be calculated online. When the Euclidean distance between the current $\mathbf{s}^i$ and Eq. (1.5) is bigger than a threshold, the robot considers itself in an unusual mode.

This method also presents some drawbacks, since it is necessary to set a threshold, and it appears that sometimes the detection fall is triggered even if the robot is not about to fall. Additionally, it is very specific to walking fall detection, since it is based on reading sensors values when walking, which makes it impossible to use for fall detection in other 'regular' tasks. The authors were aware of that limitation and suggested another falling detection method based on learning algorithm.

Experiential Learning

Machine learning has been proposed as an approach to fall detection as in Ogata et al. [2007] Kalyanakrishnan et Goswami [2011] and more recently in André et al. [2016] Hofmann et al. [2016]. The idea proposed by Ogata et al. [2007] is to discriminate two states after a disturbance. The one in which the robot can recover its stability (using fall-avoidance control) and the one in which the robot falls (the actual fall detection). An experiment has been done with several learning data, those data were obtained by applying a push on the rear side of the robot meaning that the robot is not able to discriminate the two states with another kind of disturbance. Still, the results were positive in this experiment and the robot was able to perform both fall-avoidance control and fall control (here an UKEMI motion) depending of the magnitude of the disturbance. Kalyanakrishnan et Goswami [2011] *et al.* considered two data. The first is called the False Positive Rate (FPR) and the second is the lead time τ_{lead}. The former designates the fraction of trajectories in which falling is predicted for a balanced state and the latter is the average value of $t_{fallen} - t_{predict}$ over trajectories that terminate in fallen. t_{fallen} is the time at which fallen is reach and $t_{predict}$ the time at which falling is predicted. The training is done over several trajectories to find a low FPR with a high lead time. André et al. [2016] *et al.* used four data mining algorithms over sensor data to find classifier that identifies failures. It is based on the so-called Associative Skill Memories which tries to associate data together considering that stereotypical movement has sensor footprints. Hofmann et al. [2016] uses neural network with NAO's outputs (especially gyroscopes) as training data.

Predicted ZMP

The use of the Zero-Moment Point (ZMP)Kajita et al. [2003a] seems a straightforward solution for fall detection. By definition, a walking robot is stable as long as the ZMP is inside the supporting area. When the ZMP gets to an edge or a vertex of the polygon, it implies that the robot **might** fall (necessary but not sufficient condition). Thus, Ogata et al. [2008] suggest to predict the ZMP of the system as a falling detection criterion. To do so, they assuming that the stabilizing control is a implemented by a state space control which allows the computation of the an estimated ZMP.

From the reported experiments, the falling detector shows good performance and acts quickly as requested. However, using the ZMP for a fall is not recommended in certain situations since the ZMP is known to be erroneous when the feet slips, when the surface is uneven or when the robot has multi-contact points (for example, when the robot leans on one of its elbows). Yet, the first two cases are the principal reasons for human falls. Nevertheless, it can still be considered as an acceptable solution, particularly for robots which may work in a facility since, due to security, the ground is generally a non-slippery plane in this particular context.

Conclusion

Contrary to other fall detectors, the predicted ZMP algorithm and the experiential learning have the advantage of anticipation. Chang et al. [2013] states that "It is clear that 'expectancy' is required to walk safely. There will be a motion perturbation if expectation and reality do not match". This study is a survey on humans falls. A fall detector may be realized as a loss-of-control detector *e.g.* when the robot measurements diverge from the expected control output. In other terms, the robot falls when the robustness of the control law is exceeded. So, the fall detection should not be based on the current posture or configuration of the robot but it is a criterion that evaluates when the stabilizer can no more operate correctly. Such a detector needs then to be updated in all kind of controllers (walk control, run control, whole-body reaching control, etc) which makes it more complex. For example, let's imagine a robot pushing a wall. The equilibrium results from the contact on its feet and on its hands making the projection of the CoM undefined. Thus, all above fall detections become impractical. In particular, the fall dectectors using ZMP criteria should be adapted for multi-contact (see Caron et al. [2017]).

Avoiding human/high-value objects

Now that the fall can be detected, the robot needs to perform actions against it. In this section, the control laws ensure that the robot would not hurt anybody. As for general humanoid control, the difficulties come from the fact that the robot is under-actuated. When it is falling, the contact with the ground is reduced to only an edge or a vertex (fall with no contact has not been studied yet), meaning that 1 to 3 degrees of control

are lost. Goswami et al. [2014] proposes three different methods to perform a change of the fall direction. All of them come from two main observations, the first one deals with the support polygon of the humanoid. Indeed, while falling, the robot topples, at least, on a edge of the polygon meaning that if the geometry of the polygon is changed, the fall direction is changed. The second observation is that an overall change of inertia also changes the direction of the fall.

To start off, it is important to define the *reference point* and the *desired fall direction*. The former represents a point on the ground where the robot is estimated to fall. The latter is the most favorable direction of the fall which depends on the environment.

Direction of the fall

First, to avoid any objects, it is necessary to know the direction of the fall. This can be done knowing both the CoP and the supporting area at the moment of the push. As shown in Fig. 1.3a and 1.3b, the direction of the fall is perpendicular to the so-called *leading edge*. When the robot is about to fall, the CoP is getting closer to the limits of the support polygon. When the CoP reaches an edge, that edge is called the leading edge and it represents the joint around which the robot topples.

(a) Fall direction 1 (b) Fall direction 2

Figure 1.3 – Foot placement strategy to avoid falling on something/someone. Both figures represent the robot being pushed in the back. The fall direction is the perpendicular to the leading edge (green line), crossing the CoP (point P). Point T represents an object the humanoid has to avoid and point Q is the reference point. Image from Goswami et al. [2014]

Foot placement strategy

As it can be seen in Fig. 1.3b, even if the push is in the direction of a high-value object (point T), the robot falls besides it. Thus, considering the initial position as in Fig. 1.3a, one can change the position of a foot in order to change the direction of the fall.

Let's consider the legs velocities. We can write

$$\mathbf{V_L} - \mathbf{V_{body}} = J_L \dot{\boldsymbol{\theta}}_{\boldsymbol{L}} \tag{1.6}$$

$$\mathbf{V_R} - \mathbf{V_{body}} = J_R \dot{\boldsymbol{\theta}}_{\boldsymbol{R}} \tag{1.7}$$

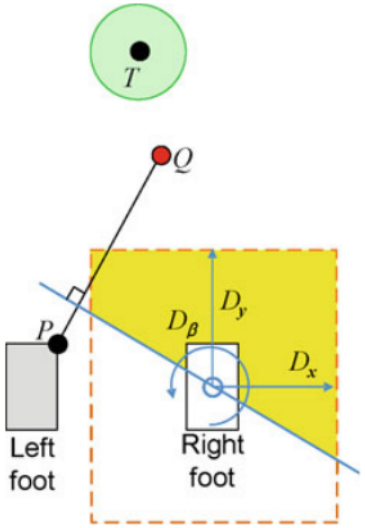


Figure 1.4 – Foot placement available region. The yellow area represents the allowable stepping zone. P is the CoP, Q is the reference point, T is the object to avoid to fall on, D_x and D_y are the maximum half lengths of the allowable stepping zone and D_β is the maximum amount of rotation of the swing foot. Image from Goswami et al. [2014]

where $\mathbf{V_L}$, $\mathbf{V_L}$ and $\mathbf{V_{body}}$ are respectively the linear and angular velocities of the left leg, right leg and body. $\dot{\boldsymbol{\theta}}_{\boldsymbol{L}}$ and $\dot{\boldsymbol{\theta}}_{\boldsymbol{R}}$ are the joints velocities of the left and right leg respectively. J_L and J_R represent the Jacobian matrices of the left and right leg respectively.

Subtracting Eq. (1.6) from Eq. (1.7)

$$\mathbf{V_R} - \mathbf{V_L} = [J_R, \ -J_L] \left[\dot{\boldsymbol{\theta}}_{\boldsymbol{R}}, \ \dot{\boldsymbol{\theta}}_{\boldsymbol{L}}\right]^{\mathsf{T}} \tag{1.8}$$

then, we can control the movement of a leg to change the direction of the fall. During the control time ΔT, and using Eq. (1.8), the reachable stepping area is deduced

$$D_k = \Delta T \sum_{i=1}^{12} |J_{R-L}(k,i)\dot{\theta}_i^{MAX}|. \tag{1.9}$$

Here i corresponds to one of the 12 joints of the two legs, k stands for x (the x-axis linear part), y (the y-axis linear part) and β (the z-axis anuglar part). In Fig. 1.4, the yellow area corresponds to the reachable stepping zone. Since the linear momentum of the robot is directing toward, it is intuitive that, because of a forward push, taking a step backward is meaningless. Once the reachable area is computed, the best position needs to be found. To do so, Goswami *et al.* propose to subdivide the reachable area in multiple cells, then for each cell, a range of angles is taken and the CoP is calculated. Finally, the reference point is estimated. From all the computations, the configuration that has the reference point closest to the desired fall direction is chosen.

The computation of this algorithm can be heavy. Moreover, the longer the legs of the robot are, the larger the reachable area is. Nevertheless, online calculation is possible by choosing large cells and few values of β. The main concern is then the control time approximation. Indeed, since it is impossible to know exactly the time of the fall, it has to be estimated. The simplest way is to represent the robot as a pendulum (2D or 3D) and to calculate the time it needs to fall. Unfortunately, this method is rather inaccurate. On a 0.4s time fall, the estimator exceeds an error of 20% Lee et Goswami [2013].

Inertia shaping

Inertia shaping consists of using the inertia of the robot in order to modify its fall direction. To do so, the robot is approximated as an inverted pendulum with rigid-body mass inertia.

The composite rigid body inertia matrix of the robot can be reduced to an ellipsoid called the reaction mass ellipsoid (RMP)Lee et Goswami [2007]. The inertia of the RMP is related to the robot angles by

$$\delta^s \hat{I} = J_I \delta \boldsymbol{\theta}. \tag{1.10}$$

Denoting I the Composite Rigid Body (CRB) inertia matrix, I can be strung out so that $I \in \mathbb{R}^{3\times3} \longrightarrow {}^s\hat{I} \in \mathbb{R}^{6\times1}$. The CRB inertia Jacobian J_I maps changes in the generalized coordinates into corresponding changes in the CRB inertia. The desired inertia is simply a transformation of the current inertia.

$$I_d = RIR^{-1} \tag{1.11}$$

and R is the rotation matrix

$$R = e^{[\boldsymbol{\omega}_d]} \tag{1.12}$$

where $[\boldsymbol{\omega}_d]$ is the skew-symmetric matrix corresponding to the vector $\boldsymbol{\omega}_d$. Thus, Eq. (1.10) is written

$$\dot{\boldsymbol{\theta}} = J_I^{\dagger}(I_d - I) \tag{1.13}$$

where $J_I^{\dagger}$ is the Moore-Penrose pseudo-inverse of J_I. The question is "How to choose the desired inertia ?". We can change the direction of the linear velocity at the CoM $\mathbf{V_G}$. It is linked with the angular velocity $\boldsymbol{\omega}_G^P$ at point P by

$$\mathbf{V_G} = \boldsymbol{\omega}_G^P \times \mathbf{PG}. \tag{1.14}$$

Then, let's set $\mathbf{e_{dfd}}$ = -$\mathbf{PT}$ the desired fall direction that opposes the fall direction. A simple way to write $\mathbf{V_G}$ would be $\mathbf{V_G} = c\mathbf{e_{dfd}}$ (Fig. 1.5) where c is the amplitude of the desired velocity. Let's set $\boldsymbol{\omega}_d = k\mathbf{e_z} \times \mathbf{e_{dfd}}$ with k the magnitude of the angular velocity. Substituting $\mathbf{V_G}$ and $\boldsymbol{\omega}_G^P$ in Eq. (1.14), k is calculated by

$$k = \frac{c\mathbf{e_{dfd}}}{(\mathbf{e_z} \times \mathbf{e_{dfd}}) \times \mathbf{PG}}. \tag{1.15}$$

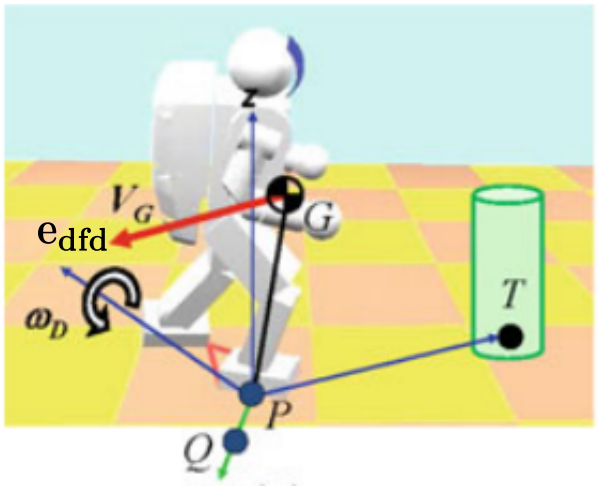


Figure 1.5 – Inertia reshaping strategy. If the robot falls over an object T, the desired linear velocity of the robot's CoM should be opposite to the fall direction as V_G. To do so, it can apply a torque ω_D on an axis passing by the CoP P and perpendicular to the fall direction. Image edited from Lee et Goswami [2013].

Thus, $\boldsymbol{\omega}_d$ is known and using Eq. (1.11), the desired inertia is found.

The work done on inertia shaping is one of the most important improvement of fall controllers because it enables the robot to operate during the fall to avoid any high-value objects. This still need to be enhanced. First, it has been made for a standing robot with a mild push represented by a force on the CoM. The robot decides where to fall only based on configuration of its environment but since a push could be on a shoulder (and thus could induce a rotation of the body), the robot may try to reach an impossible direction of fall. Therefore, the magnitude of the force and its application point should be considered for the choice of the new orientation of the fall. An extension to falls while operating a movement and while holding objects is also necessary.

Partial inertia shaping

The previous methods can be extended so that, both foot placement strategy and inertia are used simultaneously. As the foot placement strategy uses only the legs to perform its control, the idea is to manipulate the inertia of the upper part of the robot. Then, the angular velocity and the Jacobian are split such that $\dot{\boldsymbol{\theta}} = [\dot{\boldsymbol{\theta}}_{PIS}\ \dot{\boldsymbol{\theta}}_{FP}]$ and $J_I = [J_{PIS}\ J_{FP}]$. The subscript PIS stands for the Partial Inertia Shaping, so the upper part of the body, and FP represents the Foot Placement. To only shape the inertia of the upper body, the desired velocity is computed such that

$$\dot{\boldsymbol{\theta}}_{PIS} = J_I^{\dagger}(I_d - I - J_{FP}\dot{\boldsymbol{\theta}}_{FP}). \tag{1.16}$$

As the angular velocity is computed about the CoP, and for effectiveness purposes, it is preferable to perform the inertia shaping about the CoP.

Minimizing damages

A way of considering the 3^{rd} law of Isaac Asimov, i.e. "the robot must protect itself", two solutions have been considered. The first solution, and the most common one, is adding to the robot a soft carapace that can absorb shocks. The cover can be completely passive as in Kajita et al. [2016] or active Lee et al. [2016b]. Lee et Goswami [2013] suggest to only use a backpack as a cover and to change the configuration of the robot during the fall so that the robot falls on it. The second solution tries to move the robot in such a way it minimizes its shock. These two solutions have to be combined. Fujiwara *et al.* teach the robot martial art's techniques Fujiwara et al. [2002] called UKEMI to reduce damages and add the robot some shock absorbing cushion on robot strategic points. Ogata *et al.* propose two algorithms that can work online Ogata et al. [2007]Ogata et al. [2008]. Ha et Liu [2015] propose a planning algorithm to minimize the damages. Finally, Yun et Goswami [2014] suggests that a tripod fall should minimize damages considerably.

UKEMI technique

As said previously, shock absorbing components should be mounted on the robot. Fujiwara et al. [2002] suggests to give priority to knees and hips, then less effective shock absorbers are mounted on elbows, hands and back. The priority is given because of the following assumption: the robot should first fall on its knees or hips and then finishes its fall on elbows, hands or back. Since assumption comes from the fact that the formers are lower than the latters. When falling, the robot accumulates kinetic energy to the cost of potential energy. The more the robot has kinetic energy the stronger the impact is, thus breaking the fall as soon as possible would reduce damages. Thus, there is no need of covering all the robot which would increase the weight of the robot.

Backward falls

Fujiwara *et al.* have mostly concentrated their efforts on backward and forward falls. Kajita et al. [2003b] presents the backward fall. They modeled HRP-2P as a Linear Inverted Pendulum (LIP) from the CoP to the CoM. The control law consists in a succession of tasks:

1. **Squatting:** The impact force is mostly due to the potential energy. So, squatting allows the robot to reduce the altitude of the CoM minimizing the potential energy. It is triggered by the fall detection control.

2. **Extend 1:** The second factor that increases the impact force is the kinetic energy. Providing an extension of leg at a well-chosen moment allows the robot to reduce its angular velocity. It is triggered when the angle θ (angle between the ground and the line containing the heel and the hip) reaches a set angle θ_1 (experimental angle).

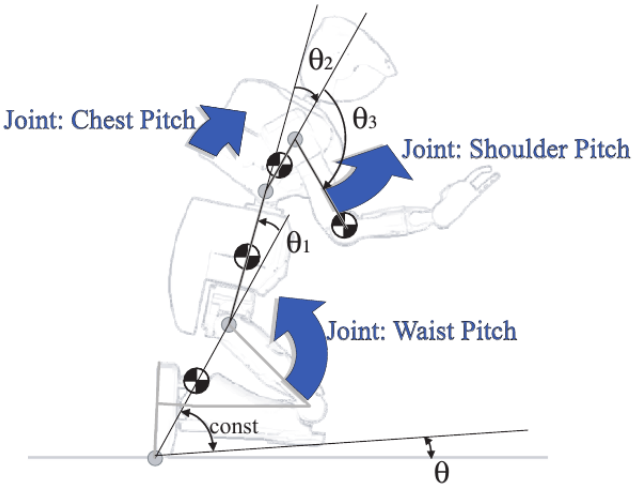


Figure 1.6 – Model of the quadruple inverted pendulum. Image from Fujiwara et al. [2004].

3. **Touchdown:** To not damage the motors, they are switched off just before the shock.

4. **Extend 2:** The motors are switched on to prevent the robot from rolling backward too far. It is triggered after an amount of time T_1, chosen experimentally.

5. **Finish:** Once the robot is stopped, it is prepared for getting up again.

The algorithm has been upgraded by using an offline non-linear optimization approach Fujiwara et al. [2006].

The main drawback is the time-consuming algorithm, that has to be done offline.

Forward falls

The forward fall is more complicated since a simple linear inverted pendulum can not be modeled to perform the UKEMI techniques. The chosen model is a quadruple inverted pendulum as shown in Fig. 1.6.

Like the previous control strategy, the minimization of the damages is partitioned in different steps:

1. **Knee Bending:** This step is equivalent to the **Squatting** strategy of the previous section. Bending knees allow the robot to decrease the height of the CoM, reducing the potential energy.

2. **Braking of the Landing Speed:** The hip pitch joint, waist pitch joint and shoulder pitch joint are actuated such that θ should be braked (Fig. 1.6).

3. **Landing:** The feedback gains are changed to make the robot more compliant to the impact of the landing.

The detail of the breaking of the landing speed step is given as follow. The angular momentum can be described as

$$\mathcal{L} = I_0(\dot{\theta} + \Delta\dot{\theta}) + I_1\dot{\theta}_1 + I_2\dot{\theta}_2 + I_3\dot{\theta}_3 \tag{1.17}$$

where θ is the angle between the knees and the ground (*cf.* Fig. 1.6), I_0 is the moment of inertia at the tip of the feet, I_i, $i = 1..3$ are the moment of inertia relative to the angular velocities $\dot{\theta}_i$, $i = 1..3$. $\mathcal{L}$ represents the angular momentum. This equation is rewritten to get the reference of the inputs $\dot{\theta}_j$, $j = 1..3$.

$$\begin{aligned}\begin{pmatrix} \dot{\theta}_1^{ref} \\ \dot{\theta}_2^{ref} \\ \dot{\theta}_3^{ref} \end{pmatrix} &= (I_1\ I_2\ I_3)^{\dagger} I_0(\dot{\theta}_0 - \dot{\theta}) \\ &= (K_1\dot{\theta}_0 + K_2) \begin{pmatrix} \dot{\theta}_1^{max} \\ \dot{\theta}_2^{max} \\ \dot{\theta}_3^{max} \end{pmatrix}\end{aligned} \tag{1.18}$$

with

$$\begin{cases} \begin{pmatrix} \dot{\theta}_1^{max} \\ \dot{\theta}_2^{max} \\ \dot{\theta}_3^{max} \end{pmatrix} = \frac{1}{\gamma}\sup((I_1\ I_2\ I_3)^{\dagger})I_0, \\ K_1 = \gamma, \\ K_2 = \text{-}\gamma\dot{\theta}. \end{cases} \tag{1.19}$$

γ is set so that $\dot{\theta}_i^{max}$ remains within actual speed limits. K_1 and K_2 are set experimentally, and, even if K_2 depends on $\dot{\theta}$, it is set to a constant.

This method is easy to implement but needs some gains tuning which are specific to each robot. It can minimize the fall quite efficiently but for stronger pushes, the upper part would fall too, thus, a control strategy of the upper part (or whole body) is also needed. To develop a better falling strategy, the robot HRP-2FX has been designed to facilitate the control law Fujiwara et al. [2006] and Fujiwara et al. [2007]. Then, a triple inverted pendulum has been considered. One of the most important quantities in falling control is the impact force. That impact is given at landing by

$$\frac{\partial\mathcal{T}^{+}}{\partial\dot{\boldsymbol{\xi}}^{+}} = \frac{\partial\mathcal{T}^{\text{-}}}{\partial\dot{\boldsymbol{\xi}}^{\text{-}}} + P \tag{1.20}$$

with $\mathcal{T}$ the kinetic energy, P the applied impact and $\dot{\boldsymbol{\xi}} = [\dot{\boldsymbol{\theta}},\ \dot{x}_j,\ \dot{z}_j]^{\mathsf{T}}$ where $\dot{x}_j$ and $\dot{z}_j$ are the horizontal and vertical velocities of the knees or hands. The signs $-$ and $+$ represent the variable just before and just after the shock happens. Assuming a perfect inelastic impact, $\dot{x}_j^{+}$ and $\dot{z}_j^{+}$ become zeros. Thus, we can calculate the change of velocity and the impact force. The angular momentum is given by

$$\mathcal{L}_j^{+} = M[\dot{x}_c^{+}\ \text{-}\dot{z}_c^{+}] \begin{bmatrix} x_g - x_j \\ z_g - z_j \end{bmatrix} + I\dot{\boldsymbol{\theta}}^{+} \tag{1.21}$$

where $\dot{x}_c$ and $\dot{z}_c$ are the coordinates of the CoM of the whole body. Then, an offline non-linear optimization is performed to minimize both the excessive angular momentum $\mathcal{L}_j^+$ and the impact force P. The experiments have proven the viability of the optimization, but it is necessary to seek for a solution offline because of the time-expensive method.

Another way to the optimal control is to decompose the optimal control problem into an NLP by the pseudospectral Legendre method Wang et al. [2012].

Contact planning

Ha et Liu [2015] present a contact planner to solve the frontward fall and backward fall. In this paper, the authors seek for a succession of contact points that minimize the maximum impulse. The contact sequence is predefined and based on common humanoid robot shape (see Fig. 1.7).

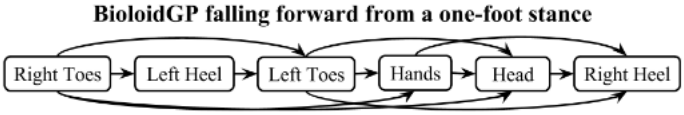


Figure 1.7 – Contact sequence for the BioloidGP robot. Image from Ha et Liu [2015].

The core component of the algorithm solves

$$v(\mathbf{x}) = \min_{\mathbf{a}} \max(g(\mathbf{x}, \mathbf{a}), v(f(\mathbf{x}, \mathbf{a}))) \tag{1.22}$$

with $\mathbf{x}$ the state of the system, $\mathbf{a}$ an action that depends on the contact, $g(\mathbf{x}, \mathbf{a})$ the computed vertical impulse local cost function, $f(\mathbf{x}, \mathbf{a})$ the transition function which outputs the state after taking $\mathbf{a}$ at $\mathbf{x}$ and $v(\mathbf{x})$ the cost-to-go function.

Once the solution is found, the results are mapped to a whole-body controller by solving an optimization problem that tries to match i) the current contact position of the robot with the model; ii) the CoM of the robot with the model; iii) the next contact position (the impact point) of the robot with the model.

Using this offline optimization tool, several simulation on different humanoid robot and an experiment shows the capability the robot of reducing falling damages by breaking a fall or by rolling when the push is strong.

Online methods

For forward methods, Ogata et al. [2007] suggest to perform the UKEMI motion using the 2D LIP equation of motion. They have not only been able to develop an algorithm considering the current location, but they have also made their controller running online. Taking into account the current location means that even if the robot is walking, the controller is still able to perform the UKEMI motion.

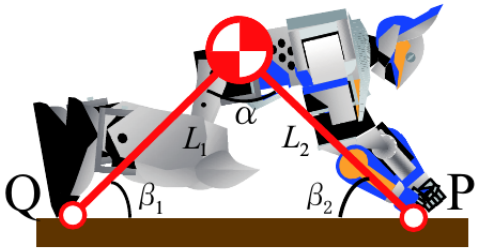


Figure 1.8 – Compass model of a falling robot. α is an angle between two support legs. β_1 and β_2 are an angle between each support leg and ground. L_1 and L_2 are each length of support leg. Image from Ogata et al. [2007].

The idea is to constrain the CoM on a line which enforces a contraction and expansion of the body length as in Fujiwara's work. The linear inverse pendulum trajectory at the CoM can be computed through an analytical solution

$$\begin{cases} x(t) = x(0)\cosh\omega t + \dfrac{\dot{x}(0)}{\omega}\sinh\omega t \\ z(t) = kx(0)\cosh\omega t + k\dfrac{\dot{x}(0)}{\omega}\sinh\omega t + z_c \end{cases}$$

with $x(0)$, $\dot{x}(0)$ the position and velocity at detection time, ω the angular frequency of the motion equation, k the slope of the line and z_c the z-intercept. Once the motion of the CoM is set, the position of the hand has to be set too. An elegant way to do it, is to consider the compass model as in Fig. 1.8. As previously, we can consider the impact force

$$\begin{cases} M(v_x^+ - v_x^-) &= P_x \\ M(v_z^+ - v_z^-) &= P_z \\ I_G(\omega^+ - \omega^-) &= -L_2 P_z \cos\beta_2 - L_2 P_x \cos\beta_2 \end{cases} \tag{1.23}$$

with v_x and v_z the linear velocities at point P on the x-axis and z-axis, ω the angular velocity, P_x and P_z are impulses, I_G the inertia around the CoM and L_2 and β_2 are defined as in Fig. 1.8. Considering a perfect inelastic collision with the ground, Ogata *et al.* show that the angle *alpha* (see Fig. 1.8) is constrained by

$$\cos\alpha \leq -\frac{I_G}{ML_1L_2} \tag{1.24}$$

with M the total mass and L_1 and L_2 defined as in Fig. 1.8.

Shock-reducing motion

Following the example of walking stabilizers, it is possible to use the equation of motion based on the ZMP Ogata et al. [2008].

$$p_x = x - \frac{z - p_z}{\ddot{z} + g}\ddot{x} \tag{1.25}$$

$$p_y = y - \frac{z - p_z}{\ddot{z} + q}\ddot{y} \tag{1.26}$$

Here $[p_x\ p_y\ p_z]$ is the position of the ZMP, g the gravity and $[x\ y\ z]$ the position of the CoM. Instead of constraining the CoM on a simple line, the authors propose to use a Bessel function that has the measured position and velocity at the initial state and has a zero velocity on the z-axis at the contact. The proposed solution is computed analytically and is of the form

$$z(t) = C_{z0}\tau I_1(\tau) + C_{z1}\tau K_1(\tau) + z_s(t) \tag{1.27}$$

where C_{z0} and C_{z1} are integration constants, $I_1(\tau)$ and $K_1(\tau)$ is the first and second modified Bessel function,

$$\tau = 2\sqrt{\frac{a_1 t + a_0}{a_1^2}} \tag{1.28}$$

and $z_s(t)$ is a particular solution of

$$\ddot{z}(t) - f(t)z(t) = -g,\ \ f(t) = \frac{1}{a_1 t + a_0} \tag{1.29}$$

with a_0 and a_1 arbitrary parameters.

Tripod fall

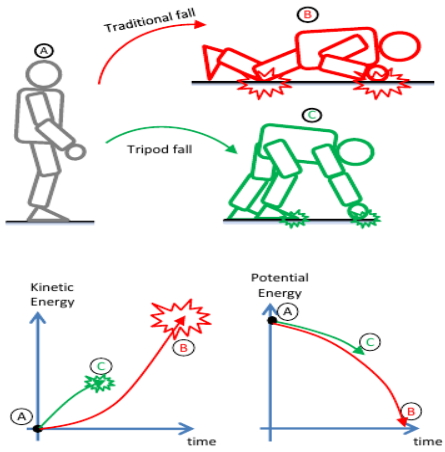
As said earlier, the damage of a fall is mostly due the potential energy. So an idea would be to stop the robot before it reaches the ground. This is the purpose of the tripod fall Yun et Goswami [2014]. By doing a step and bending such that the arms touch the ground, a tripod position comes out (Fig. 1.9a), the robot is then saving itself.

The algorithm is based on

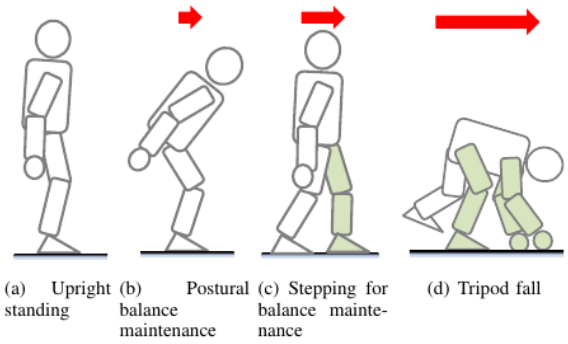
$$\dot{\mathbf{k}}_d = \Gamma_{11}(\mathbf{k}_d - \mathbf{k}) \tag{1.30}$$

$$\dot{\mathbf{l}}_d/m = \Gamma_{21}(\dot{\mathbf{r}}_G^d - \dot{\mathbf{r}}_G) + \Gamma_{22}(\mathbf{r}_{Gd} - \mathbf{r}_G) \tag{1.31}$$

with $\mathbf{k}$ and $\dot{\mathbf{k}}$ are centroidal angular momentum and its rate change, m is the total mass of the robot, $\mathbf{r}_G$ and $\dot{\mathbf{r}}_G$ are the locations and the velocity of the CoM. $\mathbf{k}_d$ and $\mathbf{l}_d$ are the desired rates of change of centroidal angular and linear momenta, respectively,



(a) Two ways of falling. The green robot is using a tripod fall thus stopping the robot as soon as possible. Compared to the red robot, the tripod results in an impact with low kinetic energy.



(b) Three ways of handling falls. (b) compensate the angular momentum to keep balance, (c) compensate the linear momentum by stepping forward and (d) uses the tripod fall that compromises both the linear and angular momentum.

Figure 1.9 – Tripod startegy. Image from Yun et Goswami [2014].

and $\mathbf{r}_G$ is the desired CoM position. Γ_{ij} represents a 3×3 diagonal matrix of feedback gain parameters.

It is then necessary to choose the gains. Tuning Γ_{11} to a small value and Γ_{21} and Γ_{22} to a large one induces a balance mode (Fig. 1.9b (b)). To obtain the stepping motion (Fig. 1.9b (c)), a large value of Γ_{11} is needed. In such case, a low-level stepping controller handles the stepping motion. Finally, to make the robot in a tripod solution, Γ_{ij} must be between Γ_{ij} of the balance mode and Γ_{ij} of the stepping motion. The values of the gains are produced with machine learning algorithms.

Compliant strategy in front of a wall

Most of the control laws don't consider the environment around the robot when falling. Hoffman et al. [2013] consider a robot that is pushed in front of wall. The main idea here is to make the robot use the wall as support zone. It has to fold its elbows for compliance.

The control law is based on the stiffness matrix which links the applied force and the position. An active compliant control is then realized. The arm is represented as a 2D planar manipulator which is modeled as two serial springs (one passive and one active). The originality of this work is to describe the arms as linear spring systems. This system is then represented by a length with a Cartesian stiffness. The main idea of Mingo *et al.* is to search for the Cartesian stiffness to reach contact stability while recovering from a disturbance. This means that the robot is capable of active

compliance by solving a non-linear optimization problem that minimizes errors between a desired Cartesian compliance matrix and the current Cartesian compliance matrix.

It is possible to map the Cartesian stiffness in the joint space with

$$K_q = J(\mathbf{q})^\intercal K_C J(\mathbf{q}) \tag{1.32}$$

where K_q and K_c are respectively the joint and cartesian stiffness and $J(\mathbf{q})$ is the jacobian from the root to the tip of the arm. This equation offers a mapping of the stiffness from the Cartesian space to the joint space and vice-versa. It means that finding a method that computes Cartesian space stiffness also computes joint space stiffness.

Discussion

In this chapter we have presented all the methods that deal with humanoid falls. At first glance, it is clear that this subject is still at its very beginning and research needs to keep going in order to have a more robust fall controller. Almost all the methods are only considering frontward fall and backward fall in an empty environment which enables to tackle the problem with some simplifications. First, it allows to reduce the problem to a 2D space and the model can be assimilated with an association of inverted pendulum. Secondly, there is no obstacle to the fall so the robot is not constrained by the environment.

We tried to classify the above research into a single graph (see Fig. 1.10) that can be seen as a global controller that selects a suited set of controllers depending on situations. The first task the robot is required to fulfill is to check whether or not humans are nearby. If the robot is holding something valuable or someone, there must be a specific controller for this case. Most of the controllers in the literature deal with falls where only foot contacts and only with the ground are involved. Free falls (no contact falls) and cluttered environment are yet to be considered.

As can be seen in Fig. 1.10 several problems are still unsolved. This thesis focuses on the free fall and falls in cluttered environment.

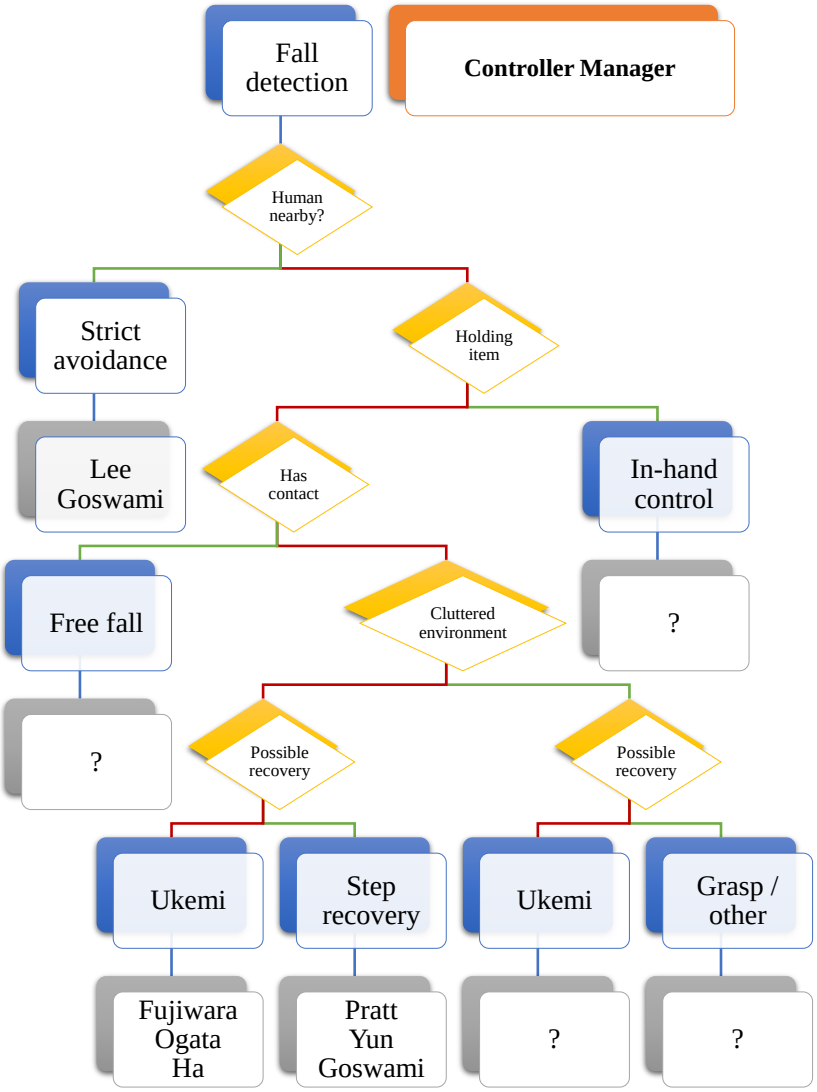


Figure 1.10 – Controller manager logic graph (our view). The green path means a 'yes', the red one, 'no'. Our contribution lies in the free fall box and in the ukemi box with or without the environment.

CHAPTER 2

Fall singularities

In the previous chapter, we reviewed several ways of handling falls for humanoid robots. From these methods we proposed a decision graph that selects one fall controller depending on the type of fall, the robot state and the surroundings. We showed that for some falling conditions, such as free falls and falls in a cluttered environment, no solution have been proposed yet in the state of the art. This chapter aims to give a solution for falls with no contacts (free falls) and to propose a way of decreasing the risks for the motors by tuning the Proportional-Derivative (PD)-gains at the impact.

Introduction

Our goal is to mitigate damage on the robot. We build our approach on the following hypotheses and choices:

- we consider that we reached the phase of total loss of attitude controllability: the robot is falling under gravity without any possible recovery other than reducing damage and the fall has been detected;
- we assume that no contact can be exploited or used to control the posture (free fall) during the falling motion;
- we assume that we can estimate reliably, using state estimators, the position of any part of the robot from embedded sensors (encoders and IMU-accelerometers);
- distances to contacts can be computed between the robot links and the environment: the environment is estimated using SLAM methodSalas-Moreno et al. [2014] which allows the computation of distances between robot links and the surroundings;

It is important to note that the robot is considered "fragile" in the sense that it has not been designed to resist impacts. Relative to existing work, we reshape the robot posture in closed-loop so that for all possible fall taxonomy cases, which we enumerate hereafter, we compute a posture that absorbs the shocks at each contact by making available *a priori* Degrees of Freedom (DoF) to comply with the impact force. Compliance, at the joints, is actively achieved by adjusting the PD gains right before the contact, the time of which can be continuously computed. We present proof-of-concepts of our approach in simulation and experimented it on the HRP-4 humanoid platform for two frequent falls: Front Fall (FF) and Back Fall (BF). The Side Fall (SF) is only considered theoretically and in simulation here, because of the high experimental risk of such falls for the robot.

Taxonomy of fall singularities

Humans intuitively try not to fall on knees first or elbows first, given that in such conditions, the damage on the skin, the bones and the body can lead to severe injuries. To prevent them, whenever possible we use our feet or hands to meet contacts and absorb impacts; some people do even roll if well trained. Hence, we start from this common sense observation to devise strategies of falling for humanoids. We term *fall singularities* the end-falling configurations that the robot should try to avoid at best. Fall singularities depend also on the surrounding environment and tell us that there are configurations which could potentially destroy the robot at the impact. First, we establish a taxonomy of fall-singularities that we illustrate in Fig. 2.1, and which cover such fall-singularities as they occur on a flat ground.

Definition 1 (Fall Singularity) *For all joints, a fall singularity exists if the line passing through the impacting body joint and its parent joint is aligned with the impact force direction.*

There are others fall singularities which are less obvious and happen when having simultaneous contacts at the end-fall. Indeed, when having two or more contacts, closed kinematics loops occur and a fall singularity can arise in the way illustrated in Fig. 2.2.

We use the following method to recognize such configurations. Considering only the kinematic loop, let $\mathbf{F}_i^{\text{ext}}$ be the impact force of the ground and J_i be the body Jacobian matrix for the impact point i. Let S_M be the motor selection vector (which have 1 for the considered motor and 0 otherwise). Thus, closed kinematic loop singularities happen when:

$$\left(\sum_i J_i^{\mathsf{T}} \mathbf{F}_i^{\text{ext}} \right) \cdot S_M = 0. \tag{2.1}$$

In such fall singularities, the impacts generate no torques on the motors, but it also means that it is impossible for these joints to comply. When a fall occurs, it is appro-

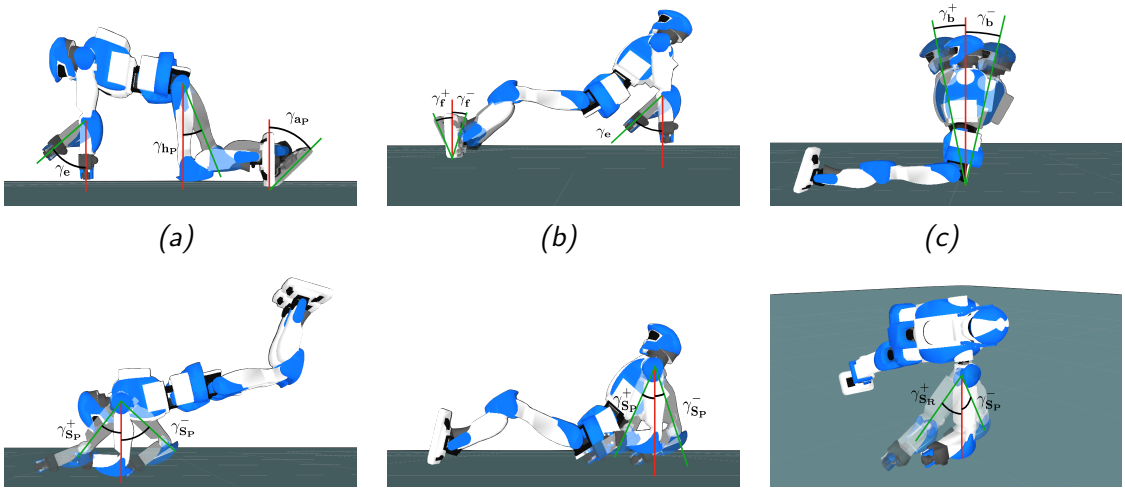


Figure 2.1 – Taxonomy of fall singularities. The red line is the impact force direction. The green lines are the possible unaligned axis the robot can choose to avoid the considered fall singularity. $\gamma_i, i \in h_p, a_p, f, b, S_p, S_R$ represents minimal angles to the fall singularity. (a) represents fall singularities due to the elbow pitch angle, the hip pitch angle and the foot pitch angle in FF. (b) shows the singularities for the ankle pitch angle and the elbow pitch angle in BF. (c) is a less common singularity of the chest pitch joint that happens when the robot falls on its buttock. (d) – (f) shows a singularity due to the shoulder pitch angle in respectively FF, BF and SF.

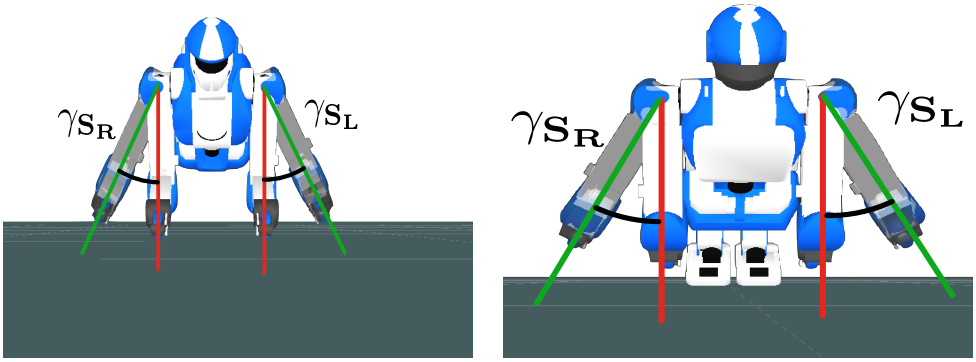


Figure 2.2 – Taxonomy of closed kinematic loop singularity. $\gamma_i, i \in S_L, S_R$ represents minimal angles to the fall singularity. The red line is the impact force direction. The green lines are the possible unaligned axis the robot can choose to avoid the considered fall singularity.

priate to use the motors as much as possible so that they can act as adjustable active spring-dampers and absorb impact shocks in the best possible way, resulting in the least possible damage.

To ensure that the robots avoid fall singularities minimal angles to their respective singularity can be considered. Those angles have been chosen so that the concerned motor can at least apply 15% of its maximal force in the impact direction (normal of the surface). The nice part of that definition is that it does not depends neither on the motor strength nor on the link length but only on the angle itself. We chose an angle of 10° which corresponds to around 17% of the motor maximal applicable force.

Singularity avoidance controller

We devised a set of tasks and their parameters embedded with a multi-objective two-priority Quadratic Programming (QP) controller to avoid fall singularities. The sensors/estimators data needed to be fed back to the controller are the distances between the robot links and the environment (the ground in this study), the robot postural configuration, and the robot attitude orientation obtained from the IMU. Here, we consider humanoid falls from an upright posture.

Fall Direction

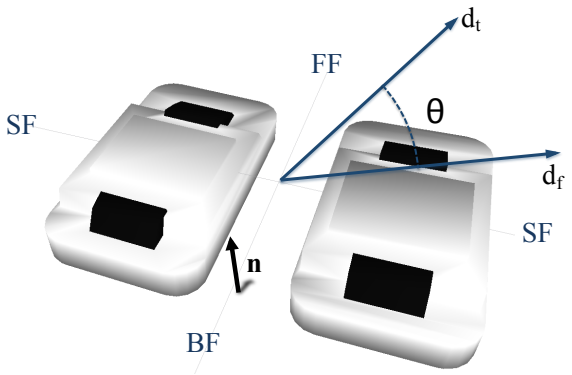


Figure 2.3 – Main directions of a fall. FF, SF and BF respectively stands for Front Fall, Side Fall and Back Fall. $\mathbf{d_t}$ *is projection of the torso yaw joint on the gravity orthogonal plan and* $\mathbf{d_f}$ *is the falling direction vector.* $\mathbf{n}$ *is the normal of the ground,* $\mathbf{d}^f$ *is the fall direction vector and* $\mathbf{d}^t$ *is the torso yaw projected vector.*

Having a body symmetry only in the sagittal plane, three mains directions, w.r.t the usual waist egocentric humanoid frame, are chosen: FF, BF and right/left SF. What differentiate SF from the other two directions is that only one arm and one leg will most likely impact the ground. For BF, the two legs, the arms, the elbows can possibly impact the ground. The FF adds possible impacts on the knees. In FF and

BF we can reshape the posture to have more impact points possibilities, and the shock can be absorbed by different parts of the robot.

Therefore, the use of BF and FF shall be preferred. A way to do it is to reshape the posture of the robot using the torso yaw joint and the humanoid arm endpoints (exploiting inertia). In the most general case, the humanoid hands/grippers are likely to be fragile, and the best option is to fall on the wrist-kind junction of the robot by keeping the hands away from impact (pulling them to the most up positions).

Once falling is detected, we compute the direction of fall. For a general environment configuration and robot posture, the direction can be obtained based on the IMU, projected in the gravity orthogonal plan. For our case study we assume a flat ground. The falling direction $\mathbf{d}^f$ is computed in a closed-loop way (i.e. during falling)

$$\mathbf{d}^f = \frac{{}^{\text{com}}\mathbf{r}_0^t - {}^{\text{com}}\mathbf{r}_0^{t_d}}{\|{}^{\text{com}}\mathbf{r}_0^t - {}^{\text{com}}\mathbf{r}_0^{t_d}\|}, \tag{2.2}$$

where t_d if the fall detection time, $t > t_d$ the current time and ${}^{\text{com}}\mathbf{r}_0$ the CoM projection point on the gravity orthogonal plan in the world frame.

Once $\mathbf{d}^f$ is obtained, the control strategy is always to try to position the humanoid's right and left arms from the right and left side of $\mathbf{d}^f$ respectively. In the case of SF, this behavior will favor landing as close as possible to a FF or BF falls. Let $\mathbf{d}^t$ be the projection of the vector defining the torso yaw joint on the gravity orthogonal plan. We design four main tasks:

- minimize $\theta = (\widehat{\mathbf{d}^t; \mathbf{d}^f})$ through posture reshaping;
- left/right wrist placing on left/right part of $\mathbf{d}^f$;
- use yaw joint to align the coronal plane with that of the ground;
- avoid fall-singular configurations.

Front fall

Fig. 2.1a gives a singularity linked to the elbow which is also a common Cartesian singularity. To get rid of it, we set an angle in the elbows. A way to reduce impact damages is to be compliant in the articulations. A compliant control can be realized but it needs the motors to be able to operate in any direction with maximum torque. We use the manipulability criterion measurement Yoshikawa [1985] for the 2-links arms.

In the following we reason in the sagittal plane. First, we compute the shoulders angle. Then, avoiding the singularity showed in Fig. 2.1d may be performed by positioning the hand relative to the shoulder. For a given elbow angle δ_e, we need to compute the angle δ represented in Fig. 2.4. P_h represents the closest point on the hand to the ground, P_g is the closest point on the ground to the shoulder and O the center of the shoulder joint. Given the vectors $\mathbf{OP_h}$ and $\mathbf{OP_g}$, δ is easily deduced. The sign of delta is calculated from $\mathbf{d}^f \cdot \mathbf{OP}_h$. The same reasoning applies to the transverse

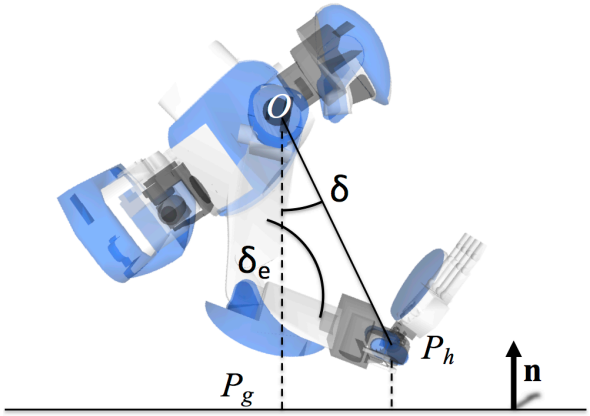


Figure 2.4 – Search of a shoulder sagittal angle in a FF. O is the center of rotation of the shoulder, P_g (resp. P_h) is the closest point on the ground (resp. the hand) to the shoulder (resp. the ground), $\mathbf{n}$ is the normal of the ground and δ_e is the current elbow angle. δ is the error angle.

plane, but we need to account for the closed kinematic fall singularity illustrated in Fig. 2.2a. To cancel it, we need to control the δ angle in Fig. 2.5 so that the line linking the origin of the elbow and the wrist point is not aligned. We have two possible options concerning the terminal point and hence δ: 1) if we know the friction coefficient, we can consider adjusting the δ angles of both the sagittal and transverse planes to lie within the friction cone of which the angle is determined from the friction coefficient; we may assume that it will be a good approximation of the reaction force direction; 2) consider that dynamic friction would allow dissipating impact energy as the wrists will slide and hence favor rather a wide δ in the wrist we want it to slide. Here, the δ angles clearances are set ad-hoc.

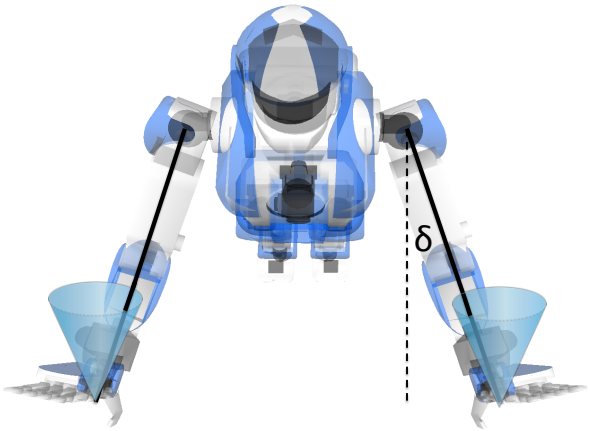


Figure 2.5 – Closed kinematic loop singularity avoidance in a FF. The blue cones are the friction cones. δ is the shoulder roll angle the robot should aim for.

The lower part of the robot embeds the strongest motors, thus, it can perform better compliance than the arms (and hence absorb higher impacts). It is preferable

that the two feet impact at the same moment, then the two knees at the same moment, in this order. This allows a better absorption of the impact and cancels out any torque generated by the collision force. To do so, hip joints are used while still paying attention to the singularity illustrated in Fig. 2.1d.

Back fall

For back falls, it is preferable that the arms touch the ground before the buttocks to have more compliance clearance. Here, the choice for δ_e must not be too small and can still be based on the manipulability criterion. This is due to the fact that humanoid robots have a shoulder joint limit more constraining backwards.

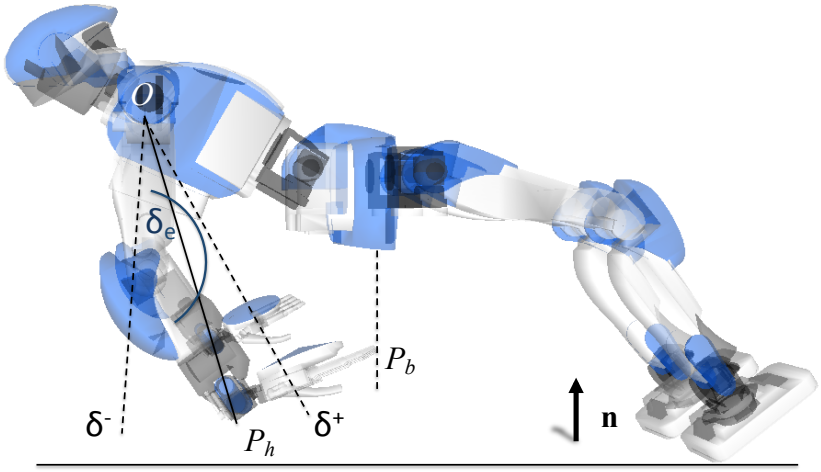


Figure 2.6 – Search of a shoulder angle in a BF. O is the shoulder center of rotation, $\mathbf{n}$ is the ground normal, P_b (resp. P_h) is the closest point on the ground (resp. the hand) to the buttock (resp. the ground) and δ_e is the current elbow angle. δ^- and δ^+ are the possible solution of Eq. (2.3).

Now that the elbow angle is set, a control similar to the FF is made. Let's find δ, the angle needed to set the shoulder pitch angle. First, we get the nearest buttock point P_b to the ground, the nearest hand point P_h to the ground and the shoulder point O. Here, an offset of the point P_b is made such that the targeting angle makes the hands below the buttock. An offset is set along the ground normal to get P_b^{offset}.

Now it is possible to find δ as shown in Fig. 2.6, such that the hands are in the plane defined by the vector $\mathbf{n}$ and the point P_b^{offset}. As the angle δ_e is known, we can just consider the equation of a circle of center O and radius $R = OP_h$. Then, the point P_n can be found by solving the system of equations:

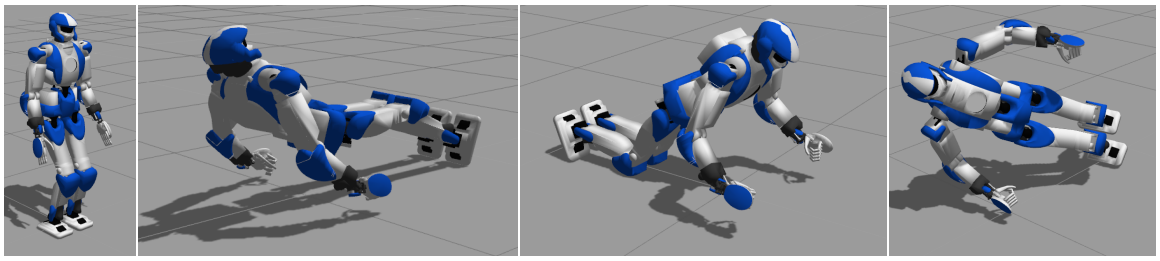
$$\begin{cases} \mathbf{P_n P_b^{\text{offset}}} \cdot \mathbf{n} = 0 \\ \mathbf{OP_n}^\top \mathbf{OP_n} = R^2 \end{cases} \tag{2.3}$$

This equation results in two possible values. The solution is the one that is in the range of motion of the shoulder joint and with the highest force in the direction of $\mathbf{n}$ (this can be found by looking at the manipulability ellipsoid of the arm).

In the transverse plane, the reasoning is similar to FF. As for FF, the feet should impact at the same time without being in the fall singularity shown in Fig. 2.1b.

Side fall

The controller tries to avoid SF at best. SF must avoid fall singularities but it also should ensure the continuity at the areas limits. Depending in the side, the left or right arm can be put in contact first (there is no other choice if the body attitude of the robot cannot be brought to a FF or BF schemes). The contact shall be made in a way to initiate a rotation of the body around the contact and feet, hopefully to reach a FF or BF with the other arm. We could not experiment real robot side falls because of the hardware high-risk failure.



(a) Standing *(b) Back fall* *(c) Front fall* *(d) Side fall*

Figure 2.7 – Different fall simulations. (a) is the static posture of the robot before the push. (b) – (d) are respectively a BF, a FF and a SF.

Compliance

In Kajita et al. [2003b], it is suggested to switch-off the motors at the impact so as not to damage mechanical parts such as gears. It is called the "TouchDown" state. We suggest another approach, that consists in reducing the gains of the motor PD-gains right before the contact occurs. We show its effectiveness despite an ad-hoc constant gain adjustment. When PD gains are reduced, the motor servos behaves like a spring-damper, and the gains can be adjusted on-line according to the falling case. In this experimentation, PD gains have been changed in an empiric way, however, we will show hereafter that it is possible to adapt the PD gains on-line considering falling speed estimation, possible posture reshaping, the environment obstacles, etc.

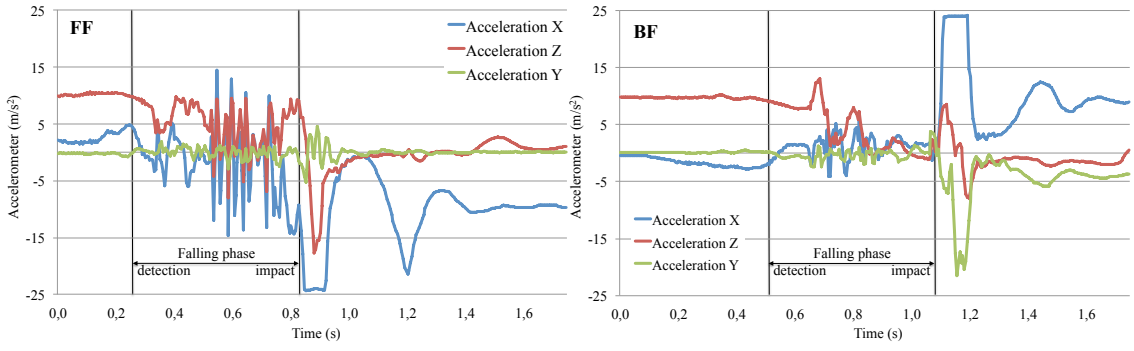
Simulation and experimentations

We use Gazebo as a dynamic simulator. The dynamics parameters and geometry are those of the HRP-4 humanoid robot. The high-level control loop of the robot runs

at 5ms. All simulations are made with the robot starting from a static stable half-sitting posture, see Fig. 2.7a. Then HRP-4 is pushed with a virtual force of 200N for a duration of 0.2s. The detection is triggered when the torso bending is bigger than a threshold of 15° (As the fall detection is complex problem, we decided to use the same arbitrary value than other paperLee et Goswami [2013]). Back fall and Front fall, Fig. 2.7c and 2.7b, behave as expected. The HRP-4 arms are used to do a more compliant control and the robot does avoid a fall singularity state. The side fall is more complex, but we considered the worst SF case where the robot does not have to reshape for a near FF of BF states, Fig. 2.7d. In this case, the controller succeeds in not falling in singularity, but the arm is not able to comply enough in such a posture, and it will be crushed by the trunk, which may result in damage. This shows the importance of favoring FF or BF falls in all situations.

Following successful trials in simulation, we decided to implement similar falling conditions on the HRP-4 humanoid robot. Because of the fragility of the robot's hardware and the lack of data from Kawada (concerning the maximum impact the gears and actuator can absorb), we decided to use a mattress, yet a relatively minimal stiff one ($>$ 3000N/m that increases with the deformation). We did not use any specific implements or shock absorbing material on the HRP-4 humanoid robot, which is used as commercially available.

The robot is placed on a stage at the same level of the mattress on which the robot is supposed to land. HRP-4 is first put in a half-sitting posture, in front of the mattress and then pushed frankly by the user (the exact value is unknown). The falling controller will trigger based on the IMU threshold (z-axis $>$ 15deg). The arms will then be servoed according to the FF or BF strategy: secure the hands, put the wrists in front and reduce the PD gains. We have performed several successful trials of front falls and back falls: the HRP-4 is still 'healthy', in the sense that it can still move and has not servo off the motors, after these trials.



(a) FF (b) BF

Figure 2.8 – HRP-4 Embedded accelerometer profile. The robot is falling toward (a) x (b) -x. At the impact the x accelerometer axis (blue line) is nearly colinear to the world z-axis.

The Fig. 2.8 illustrates the behavior of the accelerometers during the three phases and shows the acceleration due to impact at landing, see ≈ -2.6g on the x-axis which

will be nearly aligned with $-\vec{g}$ at the impact.[1] Notice an acceleration in the z-axis due to the mattress motion and a quasi-zero acceleration on the sagittal plane. On the back fall experiment, the acceleration measured on the y-axis clearly shows that the robot did not touch down on both hands at the same time. And even if the robot is symmetrical, the friction and the motors' behavior in the joints may vary (due to fatigue or damage taken before the experiment).

We do not analyze all of the joints, instead we focus on the arms. The Fig. 2.10 shows the control of the elbow to which we give more weight to handle the manipulability criteria. Whereas the shoulder joints is given more weight for positioning (δ angles in transverse and sagittal planes). Note that the positioning is made in closed-loop fashion and is correlated to the estimate of the closed-loop computed distance to impact. We can notice in both Fig. 2.10 and 2.9, that the impact occurs right after the arms reach their desired state. One can also notice that at the impact, the error between the desired state and the actual one increases as a result of shock absorption and low PD gains.

The PD gains are reduced just before the impact is detected/estimated: the P gain is divided by a factor of 1000 and the D gain by a factor of 100. On the Fig. 2.9, the roll angles are not solicited (they are of the same absolute value but opposite sign because of the rotation convention). The pitch angles behave as the elbow.

Conclusion and discussion

The fall-defined singularities combined with active compliance is a new way to consider falls control. They give a justification to the principle of self damage reduction algorithms. From that, new controllers can be developed to perform an avoidance singularity control. The controller developed here is a geometrical one. From simulation, it can be seen that it is not sufficient. Indeed, when rotation is induced to the robot (e.g. through pushing a shoulder), the rotation makes fall areas vary and the robot must re-adjust the positioning. The last part to be improved is the active compliant control. The aim is to find for each joints the gains P and D that maximize shock absorption at impact. We believe that compliant cover with eventually shock dissipative material must be combined with the controller to reduce damage in a more effective way Battaglia et al. [2009], Kajita et al. [2016].

Fall taxonomy can be exploited to perform the reverse of what is proposed here. From a design point of view the fall singularities show the worst impact cases the skeleton and the cover should resist. Thus, it is possible to consider these cases when designing the robot in order to make the robot able to passively absorb a part of the impacts.

[1] The acceleration along x-axis in Fig. 2.8a and Fig. 2.8b might be saturated just after the impact. It is inferred from the shape of the plots which keeps constant at their minimum/maximum. The true peak acceleration at the impact is assumed to be 10 to 20g from the similar experiments reported in Kajita et al. [2016]

This chapter is only considering humanoid robots, but all the work can be extended to different kinds of multi-limb robots. This work can be included in the graph Fig. 1.10 under the free fall box.

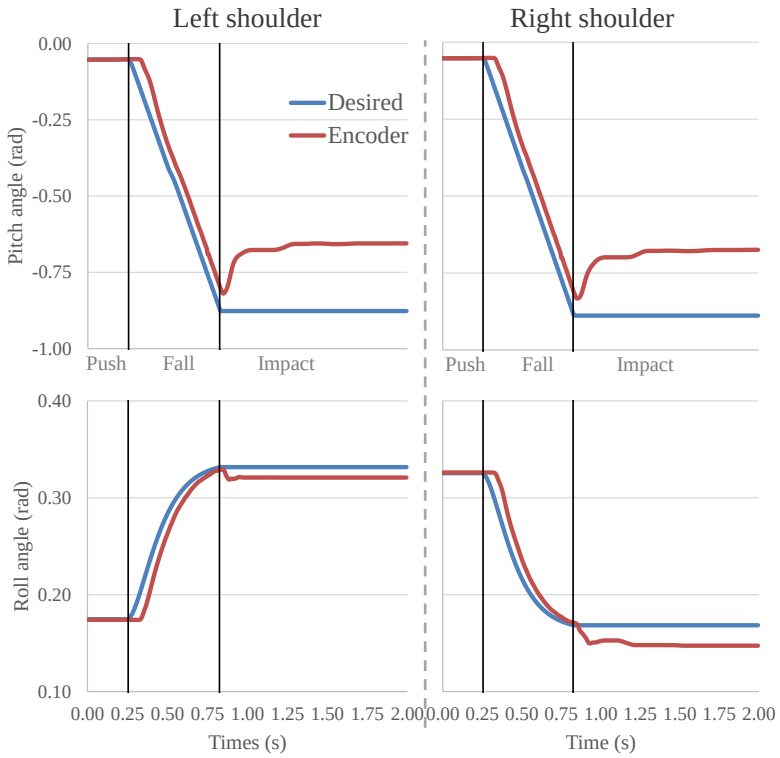


Figure 2.9 – HRP-4 left and right shoulders: desired and actual pitch and roll joint encoders. The first vertical black line is the fall detection time and the second one is the impact time. From the impact time, the PD gains are tuned down to allow a compliant behavior which leads to a bigger gap between the desired values and the encoder values.

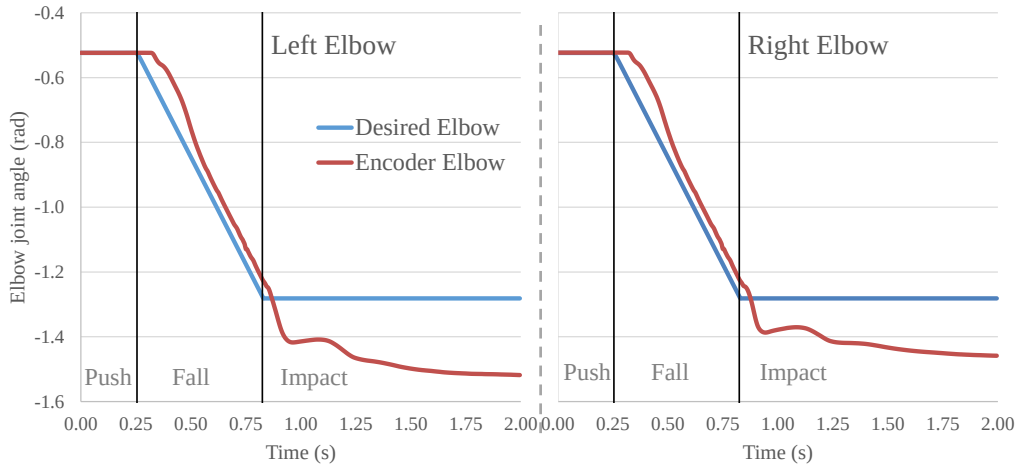
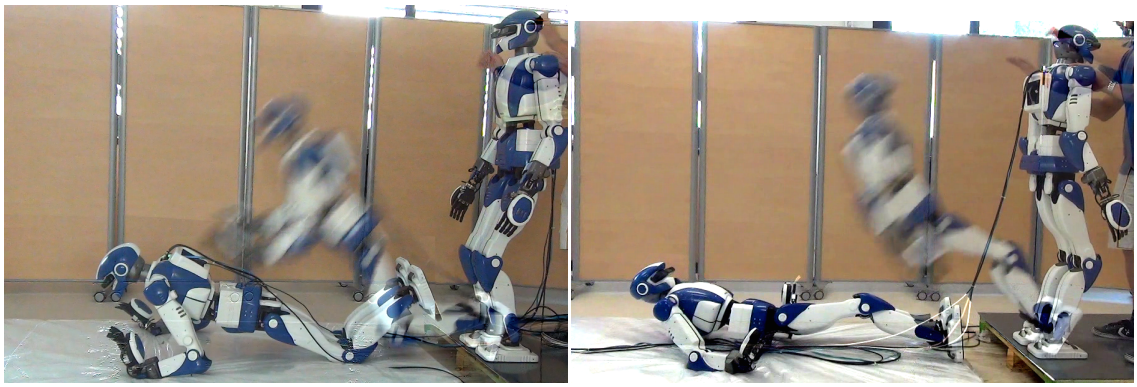


Figure 2.10 – HRP-4 left and right elbows: desired and actual joint encoders.



(a) Front fall *(b) Back fall*

Figure 2.11 – Fall experiments on HRP-4. (a) Front and (b) Back falls experiments with HRP-4: the user pushes frankly the HRP-4 humanoid. Falling is detected using a simple criterion, which results in fast posture reshaping for chosen contact landing and instant PD gain reductions to absorb the resulting impact.

CHAPTER 3

Cluttered environment and adaptive-QP

In the previous chapter we introduced the notion of fall singularity as a configuration in which the robot should not land at the impact time. We then designed a free fall controller that enables the avoidance of these singularities by the falling robot. The suggested solutions in Chapter 1 and Chapter 2 are mostly considering a fall in the sagittal plane (see Fig. 1.1) without any surroundings. In this chapter we will extend the notion of active compliance of the previous chapter while allowing fall in any directions in a cluttered environment.

Introduction

This chapter addresses the following two questions: 1) what to do in the case of a fall in a cluttered environment, and 2) how to extend the concept of gains tuning in a general setting. Both questions are an extensions of Chapter 2 and their answers lie under the *Ukemi box* in a cluttered environment in Fig. 1.10.

In our proposed method, we consider separating a fall into two phases: the pre-impact and the post-impact phase. The pre-impact phase should prepare the robot to receive the impact so that it can comply to it. Typically, at the end of this phase, the robot should be away from the singularity postures. The post-impact phase consists in using a method to tune PD gains automatically in an adaptive way. To do so, we have integrated the gain adaptation problem directly into the multi-objective QP formulation. This way, we can benefit from the on-line capabilities of the QP control approach –which has been widely adopted for controlling humanoid robots, and at the same time use the remaining DoF of the control for other tasks that can appear to be useful or necessary during falling, such as posture and CoM tasks as will be

demonstrated later.

Pre-impact phase

The pre-impact phase is an essential phase. Having a bad behavior in this phase could lead to the impossibility of performing a compliant control afterwards. It is in this phase that the robot can move/reshape its posture and prepare to the impact. We can divide it into three main parts: 1) Fall detection, 2) Environment analysis, and 3) Pre-impact strategy execution.

In **step 1)** a fall detection system must be constantly running in parallel to the performed tasks as a background process. The system should be able to stop the execution of the current tasks and switch to the pre-impact strategy execution whenever necessary. Note that this step might also include a fall recovery mode if possible.

In **step 2)** the robot performs an analysis of the situation (we exclude having humans or valuable items in the surroundings) in order to process useful information such as estimating the current pose of the robot and building a map of the surrounding's planar surfaces. Step 2) is out of this thesis' scope. We shall consider it as a black-box module and assume that the environment, the robot state, and the available environment falling surfaces are known. This is a plausible assumption considering the advances made recently in SLAM technology Salas-Moreno et al. [2014].

When the fall is detected, the controller goes through different states at each iteration loop in **step 3)**, as follows:

(i) estimate the fall direction,

(ii) search landing points,

(iii) update falling tasks,

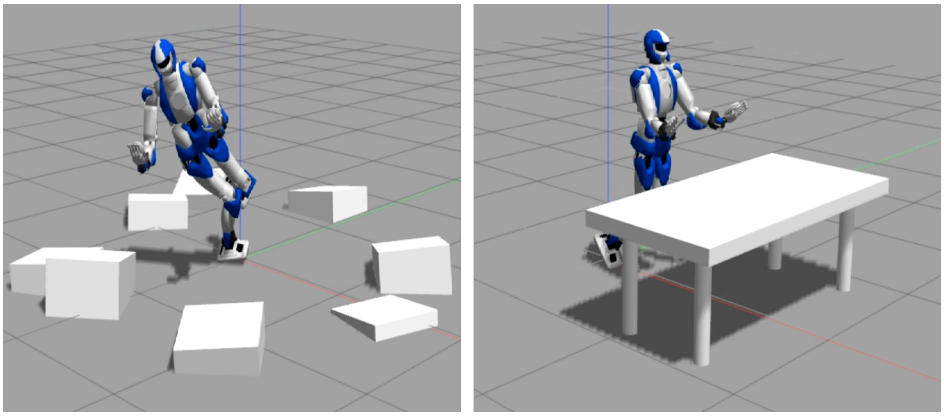


Figure 3.1 – Examples of falling in a cluttered environment. On the left, the robot is falling on randomly positioned multiple size boxes. On the right, the robot is falling toward a common table.

This step is detailed with Eq. (2.2) and in Section 3.2.1 and 3.2.2.

Since the robot is under multiple tasks and constraints, we rely on a multi-objective weighted QP controller Vaillant [2015]. The highest priority level is the QP constraints that must be satisfied without compromise:

- Joint limits, joint velocity limits, and torque limits
- Self-collision avoidance,
- The equation of motion and the physics model.

The second priority level (lowest level) are the tasks that are formulated as *set-point* tasks and combined in a weighted sum to form the objective function of the QP. A set-point task is defined as the minimization of the following quadratic cost:

$$E^{\text{sp}} = \frac{1}{2}\|S_M(k_p\boldsymbol{\vartheta} + k_v\dot{\boldsymbol{\vartheta}} + \hat{J}\ddot{\mathbf{q}} + \dot{\hat{J}}\dot{\mathbf{q}})\|^2\,, \tag{3.1}$$

where S_M is a selection matrix, k_p and k_v are proportional and damping task gains (not to be confused with the low-level joint PD gains that will be computed in Section 3.3, $\boldsymbol{\vartheta}$ is the task error and $\hat{J}$ is its Jacobian. More details on the controller can be found in Vaillant et al. [2016b].

In the following three subsections, we give further details on the three states described in **step 3)**.

Search of landing points

In order to choose the landing/impact points, we first need to know the potential impact surfaces. For a humanoid robot, the impact points are the hands, feet and knees (see Chapter 2). We also assume here that a SLAM routine coupled with appropriate segmentation algorithms can return the available planar surfaces in the environment, as in Salas-Moreno et al. [2014]. In simulation however this information is readily available.

To lower the damage risks resulting from the fall, we need to decide where to locate the impact points. These should be *reachable*, meaning that the robot can change its configuration in order to meet the desired landing spots during the falling time.

We approximate the robot as rigid stick that is falling, see Fig. 3.2 (green model on the figure). This stick lives in the plane defined by a contact point (if any, or a projection of the nearest point from feet), the fall direction vector and the gravity vector. The length of the stick is set to the distance between the latter point and the middle of the two shoulders. Both the plane of motion of the stick and its length are adjusted at each time step. The trajectory of this stick in the defined plane is a 2D circle. The shoulders' trajectories are directly computed from it and the desired whole-body posture of the robot is then generated aiming for the hands to be on their respective shoulders' trajectories.

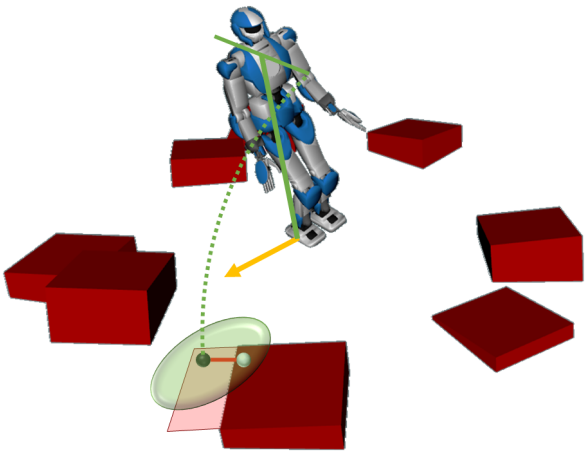


Figure 3.2 – Illustration of the search of possible impact points. The yellow arrow is the fall direction, the green lines represent a simplified stick model. The dotted arc is its trajectory. The transparent red plane is the plane where an impact surface exists. The transparent green ellipsoid is a gross representation of the polyhedron representing the arm's reachable workspace. Black and white points are the MPIP and BIP respectively. The red line represents the minimum distance between MPIP and BIP.

Finally, we compute all the intersections between shoulders' trajectories and the planes of the surfaces returned by **step 2)**. We call these points *Most Probable Impact Point (MPIP)* hereafter. Fig. 3.2 represents one such MPIP as a black point. These points may or may not be on the environment surfaces (in the example of Fig. 3.2, the MPIP does not belong to the environment), this is why we also need to compute for each MPIP its closest point belonging to its respective environment surface. These closest points on the environment surfaces are called *Best Impact Point (BIP)* (Fig. 3.2 represents the BIP corresponding to the MPIP as a white point). Note that the BIP are never on the edge of the surface, we add a safe distance (few centimeters) to the edge to avoid dangerous contact points.

We now need to make a choice between the different available BIP. First, the arms' workspaces define two polyhedra which are split in two by the coronal plane, leading to one polyhedron for reachable Front Fall points and one for reachable Back Fall points. We then compute the centroid of each polyhedron. Note that these are calculated off-line only once and are associated with the geometric model of the robot. Placing the centroid on the MPIP, the BIP is selected if the segment of line between the MPIP and the BIP is inside the given polyhedron.

In case more than one BIP satisfies the condition, then the highest BIP (highest vertical coordinate) is chosen because the impact happens sooner and less potential energy is converted into kinetic energy before the impact. In case none of the points are inside the polyhedron, the BIP having the minimum distance to its respective MPIP is chosen.

Reshaping tasks

To make the robot directly face the impact environment (Front Fall) or directly oppose it (Back Fall), since those two falls are the safest falls (see Chapter 2), we propose to use a vector orientation task which aims to align two vectors, one of which is linked to the torso and the other to the environment. A posture task is also included to help avoiding singular falls as defined in Chapter 2 and to bend the knees to lower the CoM. Finally, end-effector position tasks are used to reach the desired impact points. All of these tasks are run and their targets updated at each control loop. To implement a new set-point task (Eq. (3.1)), the task error $\boldsymbol{\vartheta}$, the task Jacobian $\hat{J}$ and the time-derivative of the task Jacobian $\dot{\hat{J}}$ are needed. We describe these derivations in the next subsections.

Vector orientation task

Let $\mathbf{u}^{\text{target}}$ be the desired goal unit vector and ${}^{r,b}\mathbf{u}$ a unit vector in the robot body coordinates. The task error is given by:

$$\begin{aligned} \boldsymbol{\vartheta}^{\text{vo}} &= \mathbf{u}^{\text{target}} - {}^{0}E_{r,b}\,{}^{r,b}\mathbf{u}\,, \\ &= \mathbf{u}^{\text{target}} - {}^{0}\mathbf{u}\,, \end{aligned} \tag{3.2}$$

where ${}^{0}E_{r,b}$ is the rotation from the body b to the world. As the target vector is considered as fixed in the world, only the time-derivative of the robot vector is considered:

$$\begin{aligned} {}^{0}\dot{\mathbf{u}} &= {}^{0}E_{r,b}[{}^{r,b}\boldsymbol{\omega} \times {}^{r,b}\mathbf{u}]\,, \\ &= -{}^{0}E_{r,b}({}^{r,b}\mathbf{u}\times)\,{}^{r,b}J^{\text{ang}}_{r,b}\dot{\mathbf{q}}\,, \\ &= \hat{J}^{\text{vo}}\dot{\mathbf{q}}\,. \end{aligned} \tag{3.3}$$

Here, ${}^{r,b}\omega$ is the angular velocity of the body in body coordinates and ${}^{r,b}J^{\text{ang}}_{r,b}$ is the angular part of the body Jacobian in body coordinates. Differentiating one more time, the time-derivative of the task Jacobian is then:

$$\dot{\hat{J}}^{\text{vo}} = {}^{0}E_{r,b}[{}^{r,b}\boldsymbol{\omega} \times ({}^{r,b}\boldsymbol{\omega} \times {}^{r,b}\mathbf{u}) + {}^{r,b}\mathbf{a}^{\text{vp,ang}} \times {}^{r,b}\mathbf{u}]\,, \tag{3.4}$$

where ${}^{r,b}\mathbf{a}^{\text{vp,ang}} = {}^{r,b}\dot{J}^{\text{ang}}_{r,b}\dot{\mathbf{q}}$ is the angular part of velocity-product of the acceleration in body coordinates Featherstone [2014].

The targeted vector is set so that $\mathbf{u}^{\text{target}} \in \mathcal{D}_0$ and $\mathbf{u}^{\text{target}} \cdot \mathbf{d}^{f} = 0$ (Fig. 3.3). ${}^{r,b}\mathbf{u}$ is chosen perpendicular to the torso sagittal plane in the torso coordinates. There are four possible solutions so both vectors must be chosen depending on whether a front fall or a back fall is desired.

Relative distance task

Ideally, we would like that multiple impacts occur all at the same time, but this is difficult to achieve in practice because it requires the estimation of the exact impact

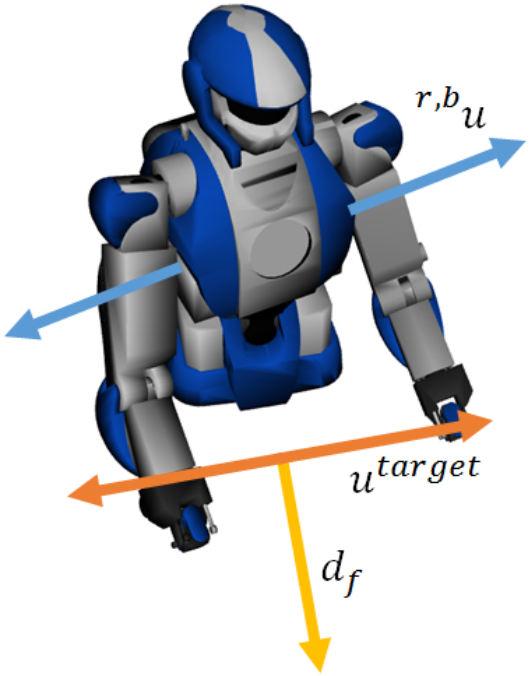


Figure 3.3 – The four possibilities for the vector orientation task. The blue vectors are the two possible body vectors ${}^{r,b}\mathbf{u}$. The yellow vector is the fall direction and the two orange vectors are the possible targets $\mathbf{u}^{target}$.

time. A solution is to control the distance between the desired environment impact surfaces and the robot impacting bodies so that the relative error of the distances between two pairs of surface-impacting bodies is zero. One of the advantages of this task is that it handles different heights of surfaces. We remind here that the considered surfaces are planar and the environment is static, so the time-derivatives of surfaces' normal are zero. Let ${}^{r,b_1,p_1}\mathbf{r}_0$ be the closest point of a body b_1 to a surface s_1 and ${}^{r,b_2,p_2}\mathbf{r}_0$ the closest point of a body b_2 to a surface s_2. Let ${}^{e,s_1,p_1}\mathbf{r}_0$ and ${}^{e,s_2,p_2}\mathbf{r}_0$ be points on s_1 and s_2 respectively. The distance of a pair of impact body and surface is:

$$d_i = \|{}^{r,b_i,p_i}\mathbf{r}_0 - {}^{e,s_i,p_i}\mathbf{r}_0\| = \|\mathbf{d}_i\|, \ i = 1, 2. \tag{3.5}$$

Here, we do not want to consider the minimal distance but rather a distance along an axis, which is more useful in our application. The task is designed to modify the distance between robot bodies and surface planes, so the distance along the normal of a plane is more relevant. This method controls only the motion along the normal of the plane, while the motion along the plane itself is left free and will be handled by a position task for reaching the desired impact points.

Let now $\mathbf{u}_1$ and $\mathbf{u}_2$ be unit vectors linked to s_1 and s_2 respectively. The task error is:

$$\vartheta^{\text{rd}} = \mathbf{d}_1 \cdot \mathbf{u}_1 - \mathbf{d}_2 \cdot \mathbf{u}_2 \,. \tag{3.6}$$

The surfaces s_1 and s_2 are considered fixed so the time-derivatives of $\mathbf{u}_i$ and ${}^{e,s_i,p_i}\mathbf{r}_0$ is zero ($i = 1, 2$). Note that if this assumption is false, then it means that the robot

would fall on a moving environment or on a curved surface. It is possible to adapt the tasks to handle such cases but we will not consider them here. The time-derivative of ϑ is given by:

$$\begin{aligned} \dot{\vartheta}^{\text{rd}} &= \mathbf{v}_1 \cdot \mathbf{u}_1 - \mathbf{v}_2 \cdot \mathbf{u}_2 \,, \\ &= (\mathbf{u}_1^{\mathsf{T}} J^{\text{lin}}_{r,b_1,p_1} - \mathbf{u}_2^{\mathsf{T}} J^{\text{lin}}_{r,b_2,p_2}) \dot{\mathbf{q}} \,, \\ &= \hat{J}^{\text{rd}} \dot{\mathbf{q}} \,, \end{aligned} \tag{3.7}$$

where $J^{\text{lin}}_{b_i,p_i}$ is the linear part of the body Jacobian of point p_i associated to body b_i $(i = 1, 2)$. The Jacobian time-derivative is:

$$\dot{\hat{J}}^{\text{rd}} = \mathbf{u}_1^{\mathsf{T}} \dot{J}^{\text{lin}}_{b_1,p_1} - \mathbf{u}_2^{\mathsf{T}} \dot{J}^{\text{lin}}_{b_2,p_2} \,. \tag{3.8}$$

End-effector position task

The end-effector position task is a common task Vaillant et al. [2016b]. The points on the hands to control are the closest points to their respective chosen BIP. We also mention that the task should be written in the surface frame so that only the x and y coordinates are controlled by the task. The z-coordinate (normal to the surface) is handled by the relative-distance task above.

Post-impact phase

The pre-impact process described in Section 3.2 shapes the robot into a "compliable" posture just before the impact. The impact is produced whenever the feet, knees or hands are about to touch down. From that instant, the controller behaves as an active compliance for lowering the damage, using a single QP whole-body controller.

In position-controlled humanoids, the low-level actuator controller consists in a PD, which leads to the simplified governing equation:

$$H\ddot{\mathbf{q}} + \mathbf{c} - J^{\mathsf{T}} G \boldsymbol{\lambda} = \boldsymbol{\tau} = K\mathbf{e} + B\dot{\mathbf{e}} \,, \tag{3.9}$$

where $H \in \mathbb{R}^{N_{\text{dof}} \times N_{\text{dof}}}$ is the robot inertia matrix, $\mathbf{c} \in \mathbb{R}^{N_{\text{dof}} \times 1}$ the gravity and Coriolis vector, $J \in \mathbb{R}^{6N_c \times N_{\text{dof}}}$ the contact Jacobian, $G \in \mathbb{R}^{6N_c \times N_c N_g}$ the matrix of friction cone generators, $\boldsymbol{\lambda}$ is the contact forces magnitude vector, and $\boldsymbol{\tau}$ the generalized forces (comprising the actuation torques for the actuated joints and zero entries for the non-actuated ones). The parameters $K \in \mathbb{R}^{N_{\text{dof}} \times N_{\text{dof}}}$ and $B \in \mathbb{R}^{N_{\text{dof}} \times N_{\text{dof}}}$ are the diagonal matrices of PD gains, $\mathbf{e} = \mathbf{q}^{\text{ref}} - \mathbf{q}$ and $\dot{\mathbf{e}} = \dot{\mathbf{q}}^{\text{ref}} - \dot{\mathbf{q}}$ are respectively the errors in joint position and velocity. For the free-flyer part, the torque is null so are K, B, $\mathbf{e}$ and $\dot{\mathbf{e}}$. $\mathbf{q}^{\text{ref}}$ is set to the current configuration just before the impact and $\dot{\mathbf{q}}^{\text{ref}}$ is set to zero. Note that in the case of joints without motors (e.g. the free-floating base) the corresponding entries in the diagonals of K and B are zeros. We denote $\mathbf{K}$ and $\mathbf{B}$ the vectors containing the diagonal entries of K and B respectively, i.e. $K = \text{diag}(\mathbf{K})$

and $B = \text{diag}(\mathbf{B})$. N_{dof}, N_c and N_g are respectively the number of degrees of freedom (dof) of the system, the number of contact points and the number of generators of the linearized friction cones. We also define N_m as the number of motors. K and B have constant values that encode the default high stiffness behavior of the motors. These values are generally very high to make the motors track the reference values as fast as possible accounting for perturbations, inertia –and more general dynamics, while avoiding overshooting. In order to comply, we need to relax and adapt these values.

1-dof analysis

Let's consider for the sake of analysis a mass-spring-damper system. This is a well known system that was thoroughly studied in the control literature. Unfortunately, there is not much existing work on the problem of gain-tuning of such a system at impact. In this section we aim to solve our problem analytically in order to have a good understanding of the problems at stake.

Analytical resolution

The analytical solution of our system comes from the resolution of the following differential equation:

$$\begin{aligned} m\ddot{z}(t) - Ke(t) - B\dot{e}(t) + mg = 0 \\ \Leftrightarrow m\ddot{z}(t) - K(z^{\text{ref}} - z(t)) - B(\dot{z}^{\text{ref}} - \dot{z}(t)) + mg = 0 \end{aligned} \tag{3.10}$$

where m is the mass of the system, $z(t)$ is the position, g the gravity and K, B the gains that need to be tuned. This equation can also be interpreted as a PD controller of a linear motor supporting a mass m under the gravity g. In such a case, the z^{ref} is the desired position which is set to the position just before impact $q^{-}(0)$ and $\dot{z}^{\text{ref}}$ is always 0.

When solving for $z(t)$, three distinct solutions are found. An under-damped solution occurs if $B^2 < 4mK$, in which case the system is oscillating at its natural frequency. The critically-damped solution occurs when $B^2 = 4mK$. It is the fastest way to reach the z^{des} position as a stable equilibrium point without overshooting. The last solution is the over-damped system when $B^2 > 4mK$. This leads to a long time reaching z^{des}. The derivations of the following equation are made with the symbolic computation software Maplesoft [2017].

Off-line non linear resolution

From Eq. (3.10), it is possible to find the gains K and B such that:

- The joint limits are respected
- The joint force limits are respected

- The robot reaches a zero velocity

Here, since we cannot afford overshooting and joint limit violation, we are only interested in over-damped solutions. It is thus a necessity to always include the constraint equation $B^2 > 4mK$ of the system. Note that we do not consider the joint velocity limit. This is due to two main consideration: i) During all the post-impact phase, the system is slowing down; ii) The joint velocity comes from the impact which is not directly controllable (input) so, the post-impact velocity might or might not cross the velocity limit.

Solving Eq. (3.10) with t, K and B as parameters leads to

$$z(t,K,B) = \frac{\mathrm{e}^{-\frac{1}{2m}(B-\sqrt{\gamma})t}(Bg + 2K\dot{z}(0) + \sqrt{\gamma}g)m}{2\sqrt{\gamma}K} - \frac{\mathrm{e}^{-\frac{1}{2m}(B+\sqrt{\gamma})t}(Bg - 2K\dot{z}(0) - \sqrt{\gamma}g)m}{2\sqrt{\gamma}K} + \frac{Kz(0) - mg}{K} \tag{3.11}$$

with $\gamma = B^2 - 4Km$.

To reduce the expressions and the number of variables, and without loose of genericity, the initial condition and system intrinsic parameters are numerically set so that

$$\begin{cases} m = 1 \\ g = 9.81 \\ z(0) = 0 \\ \dot{z}(0) = \text{-}5 \end{cases}. \tag{3.12}$$

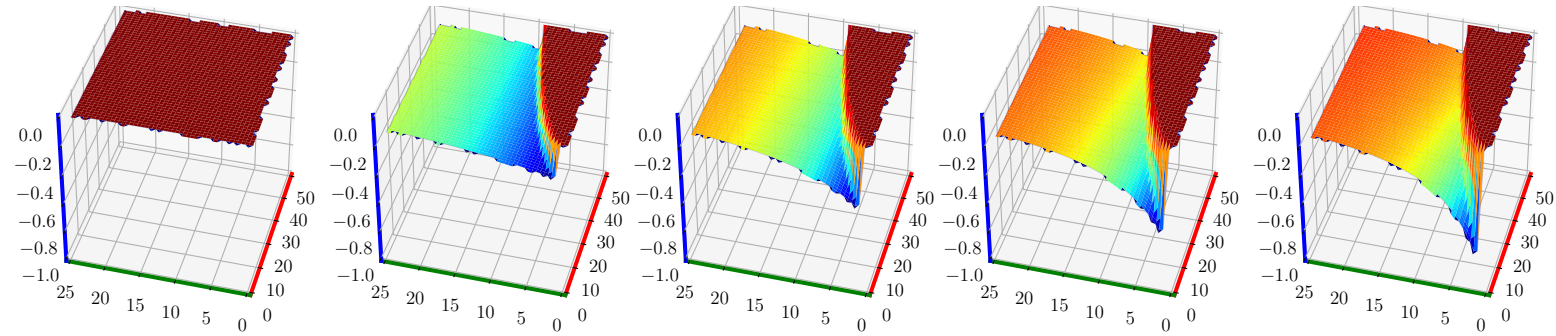
Initializing the velocity to a value different from zero simulates a system configuration just after an impact. Suppose the system is impacting with a net velocity $v^- = \text{-}0.5\text{ms}^{-1}$ and $\dot{z}^-(0) = 0\text{ms}^{-1}$ and that the impact is inelastic (no bounce), the remaining velocity in the joint is then $\dot{z}^+(0) = \text{-}5\text{ms}^{-1}$. $z(t,K,B)$ becomes

$$z(t,K,B) = \frac{\mathrm{e}^{-\frac{1}{2}(B-\sqrt{\gamma})t}(\text{-}10K + 9.81(B+\sqrt{\gamma})}{2\sqrt{\gamma}K} - \frac{\mathrm{e}^{-\frac{1}{2}(B+\sqrt{\gamma})t}(10K + 9.81(B-\sqrt{\gamma})}{2\sqrt{\gamma}K} - \frac{9.81}{K}. \tag{3.13}$$

As we seek the solution in the over-damped domain, $\gamma > 0$ is added as a constraint of the equations. A 3d representation (Fig. 3.4) of z over time as a function of K and B can thus be obtained.

Taking the time-derivative of $z(t,K,B)$ leads to the joint velocity

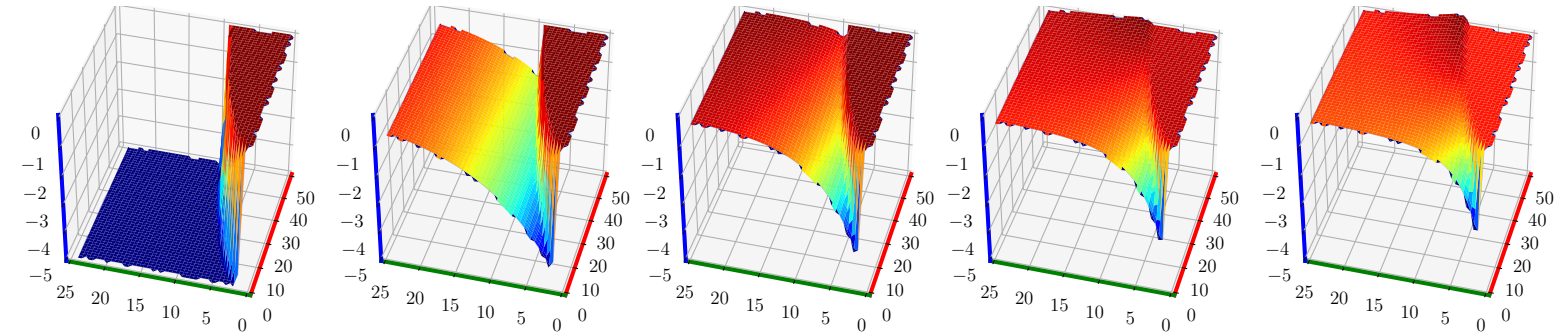
$$\dot{z}(t,K,B) = \frac{(\text{-}B+\sqrt{\gamma})\mathrm{e}^{-\frac{1}{2}(B-\sqrt{\gamma})t}(\text{-}10K + 9.81(B+\sqrt{\gamma})}{4\sqrt{\gamma}K} - \frac{(\text{-}B-\sqrt{\gamma})\mathrm{e}^{-\frac{1}{2}(B+\sqrt{\gamma})t}(10K + 9.81(B-\sqrt{\gamma})}{4\sqrt{\gamma}K}. \tag{3.14}$$



(a) $t = 0ms$ (b) $t = 50ms$ (c) $t = 100ms$ (d) $t = 150ms$ (e) $t = 200ms$

Figure 3.4 – Evolution of the position z for a given set ($K \in [0..50]$, $B \in [0..25]$). The red axis is the stiffness K in N.m^{-1}, the green axis is the damping B in N.s.m^{-1} and blue axis is position z in m.

The 3d representation of $\dot{z}$ over time as a function of K and B is also output (see Fig. 3.5).



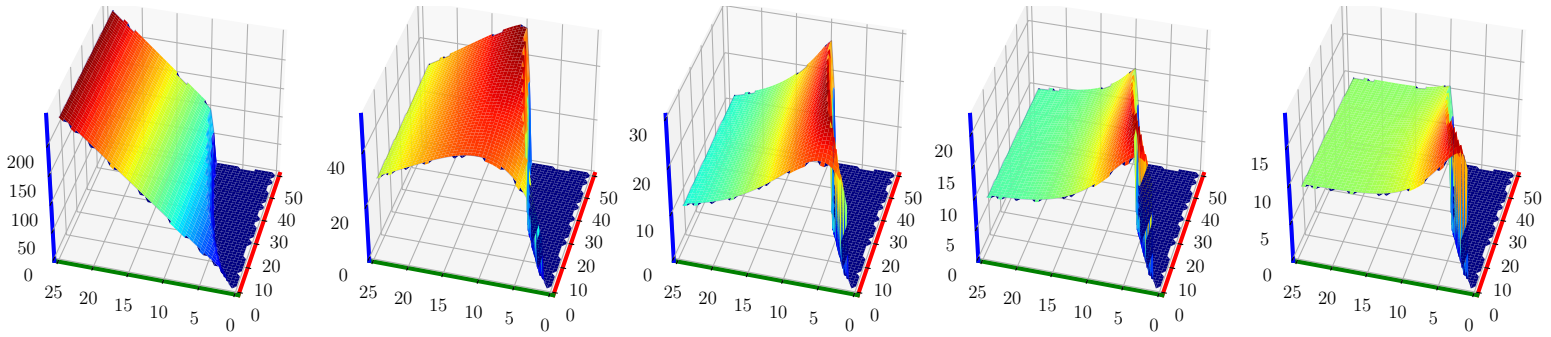
(a) $t = 0ms$ (b) $t = 50ms$ (c) $t = 100ms$ (d) $t = 150ms$ (e) $t = 200ms$

Figure 3.5 – Evolution of the velocity $\dot{z}$ for a given set ($K \in [0..50]$, $B \in [0..25]$). The red axis is the stiffness K in N.m^{-1}, the green axis is the damping B in N.s.m^{-1} and blue axis is velocity $\dot{z}$ in m.s^{-1}

Lastly, the force over time, K and B is computed (Eq. (3.15) and Fig. 3.6).

$$\begin{aligned} f(t,K,B) &= K(z^{\text{ref}} - z(t,K,B)) - B\dot{z}(t,K,B) \\ &= -\frac{(2.5B^2 - 5K - 4.905B + (4.905 - 2.5B)\sqrt{\gamma})\mathrm{e}^{-\frac{1}{2}(B-\sqrt{\gamma})t}}{\sqrt{\gamma}} \\ &\quad -\frac{(-2.5B^2 + 5K + 4.905B + (4.905 - 2.5B)\sqrt{\gamma})\mathrm{e}^{-\frac{1}{2}(B+\sqrt{\gamma})t}}{\sqrt{\gamma}} + 9.81 \end{aligned} \tag{3.15}$$

The Projection of the joint constraint and the torque constraint on the K-B plane through time is shown in Fig. 3.8. Having no overshoot (and thus no oscillation in the system) is the same as having no change of sign in the velocity. Thus, we can define the overshoot time t^{os} at which the velocity reaches 0. Solving Eq. (3.14) $= 0$ as a



(a) $t = 0ms$ (b) $t = 50ms$ (c) $t = 100ms$ (d) $t = 150ms$ (e) $t = 200ms$

Figure 3.6 – Evolution of the force f for a given set ($K \in [0..50]$, $B \in [0..25]$). The red axis is the stiffness K in N.m^{-1}, the green axis is the damping B in N.s.m^{-1} and blue axis is the force f in N

function of K and B, we get

$$t^{os} = \frac{\ln\left(-\frac{4905B-2500K-9623.61}{1250B^2-1250B\sqrt{\gamma}-2500K-4905B+4905\sqrt{\gamma}+9623.61}\right)}{\sqrt{\gamma}}. \tag{3.16}$$

This equation is stiffness-dependant and damping-dependant. The joint limit constraint is given by

$$\underline{z} \leq z(t = t^{os}, K, B) \leq \overline{z}. \tag{3.17}$$

Here, $\underline{z}$ and $\overline{z}$ are the lower and the upper bounds and z is given in Eq. (3.10).

The force limits can be reached in three possible ways: i) high damping when having high velocity; ii) high stiffness when having high error; iii) a mixed of the two. In this study, we considered impacts involving short post-impact distance between the initial position and the limit and high post-impact velocity. It means that the maximum force value comes from the damping of the velocity and not from the stiffness linked to the distance. Moreover, since we will only slow down the system, we can consider that the maximum force is reached at $t = 0$ thus giving the constraint:

$$\underline{f} \leq f(t = 0, K, B) \leq \overline{f}. \tag{3.18}$$

This assumption is confirmed in this example with Fig. 3.8b as the force limit is getting less an less restrictive with time. As only the over-damped system is taken into account, a third constraint is added.

$$4K - B^2 < 0. \tag{3.19}$$

Finally, to find the gains, we solve the nonlinear optimization problem

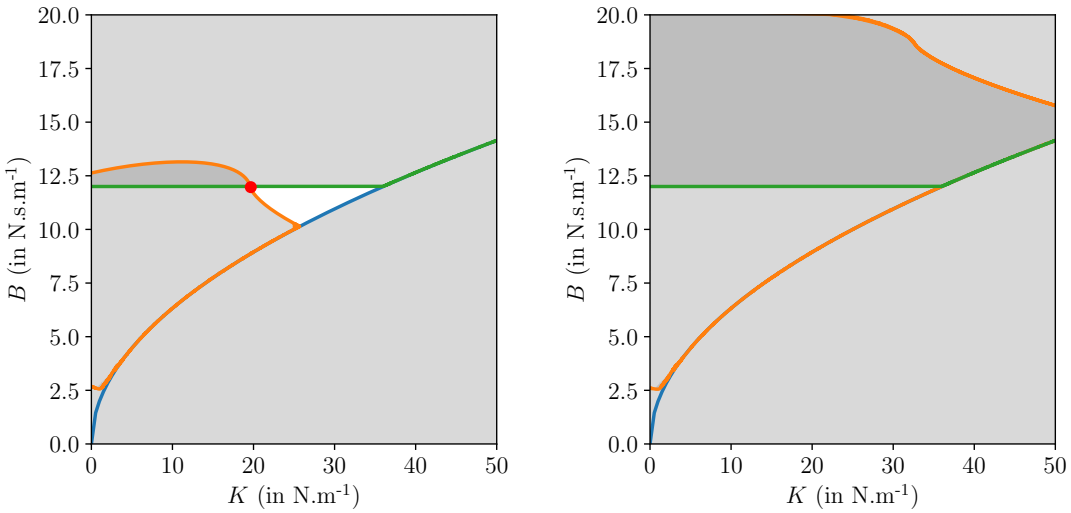
$$\begin{cases} \min\limits_{K,B} & K^2 + B^2 \\ s.t. & Eq.\ (3.17) - (3.19) \end{cases} \tag{3.20}$$

that finds the minimum solution for K and B that respects the constraints. The solution of this problem has been computed in a Matlab simulation MATLAB Software

[2016] and the result is shown in Fig. 3.7. In this example, the impulse happens at the time $t_0 = 0$ such that at t_0^- the velocity is zero and at t_0^+ the velocity is -5m.s^{-1}. This means that the impulse is

$$\iota = \int_{t_0^-}^{t_0^+} f(t)dt = m(\dot{z}(t_0^+) - \dot{z}(t_0^-)) = -5. \tag{3.21}$$

We also set $z(0) = 0$, $\underline{z}$ = -0.5, $\overline{f} = 60$.

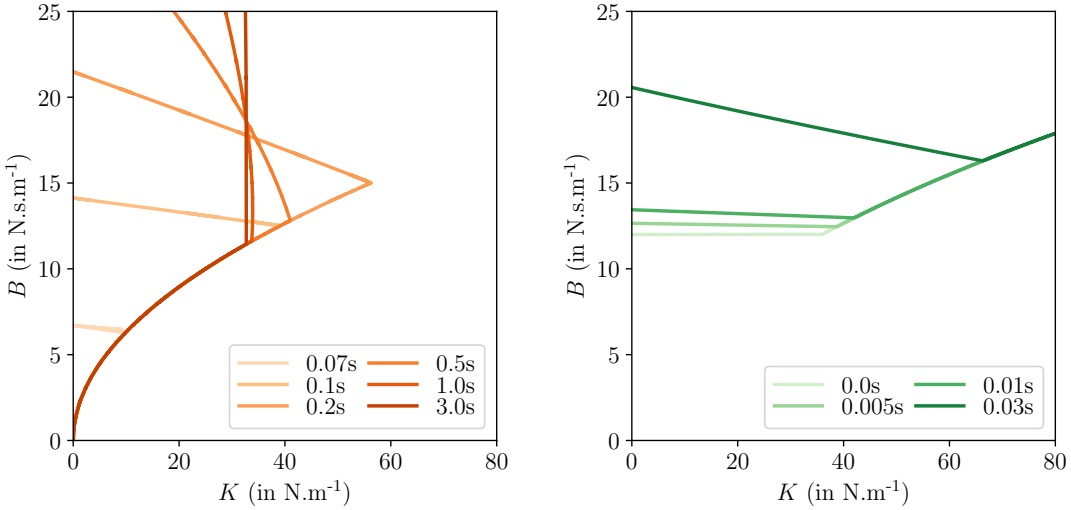


(a) Solution region ($\underline{z}$ = -0.5) *(b) No solution ($\underline{z}$ = -0.3)*

Figure 3.7 – Computation of the solution region depending on the problem constraints in the K-B domain. The blue curve represents Eq. (3.19), the green Eq. (3.18) and the orange Eq. (3.17). The possible solutions lie in the white region. The red dot is the solution of the QP Eq. (3.20).

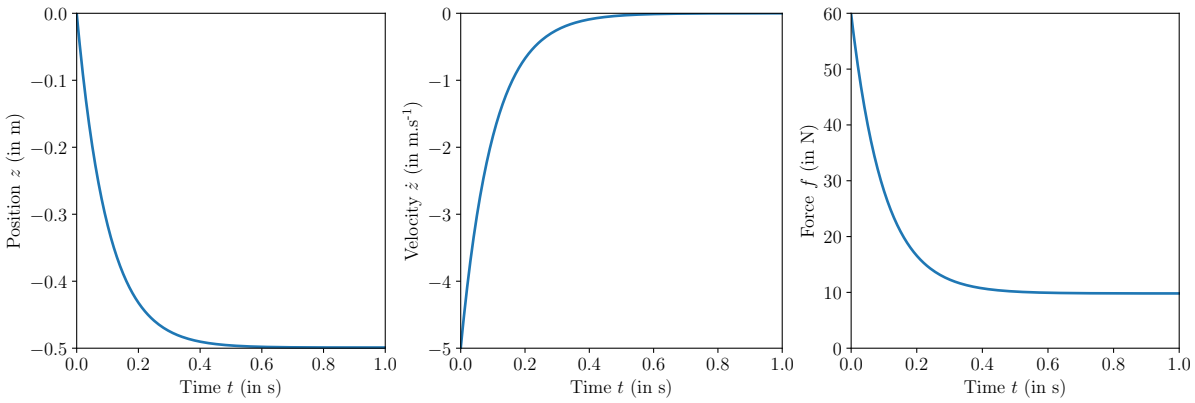
When the joint limit is not too close to the current position (Fig. 3.7a), we are able to find a set of solutions that respect all the constraints (Fig. 3.9). On the other hand, if the joint limit is set to 0.3 instead of 0.5 (much closer to the current position) as in Fig. 3.7b, no solution can be found. We can draw two conclusions from this study: i) the solutions to the problem are dependent on the system configuration, which means that the pre-impact phase is crucial; ii) the search for a single pair of (K,B) may not be the best choice. We have assumed here that the stiffness and damping coefficients stay constant during the post-impact phase. However, from Fig. 3.8a we can see that the joint limit constraint does not influence the choice of K and B before 0.07s. Moreover, Fig. 3.8b shows that after 0.03s K and B can be largely increased without reaching the torque limit.

These concerns lead to the consideration of finding a time-varying stiffness and damping coefficients during the post-impact phase. This is the goal of the following subsection below.



(a) Joint limit projection through time. *(b) Force limit projection through time.*

Figure 3.8 – Projection of the joint limit and the force limit on the K-B plane for different times with z = -0.3m.



(a) Evolution of $z(t)$. *(b) Evolution of $\dot{z}(t)$.* *(c) Evolution of $f(t)$.*

Figure 3.9 – Evolution of the different joint parameters through time. The system reaches the joint limit without crossing it and after around $0.5s$ it is completely stopped.

Multi-dof on-line solution

Our novel idea is to use a multi-objective QP formulation in the $\mathbf{X} = (\ddot{\mathbf{q}}, \boldsymbol{\lambda}, \mathbf{K}, \mathbf{B})$ decision vector.

First, to handle the impact/contact, a constraint is added so that at the contact points, the velocity is zero. This condition is realized with the following constraints Vaillant et al. [2016a]:

$$S_M \left(\frac{\underline{\mathbf{v}} - \mathbf{v}}{\Delta T} \leq J\ddot{\mathbf{q}} + \dot{J}\dot{\mathbf{q}} \leq \frac{\overline{\mathbf{v}} - \mathbf{v}}{\Delta T} \right) , \tag{3.22}$$

where S_M is a $n \times 6$ ($n \leq 6$) selection matrix, $\underline{\mathbf{v}}$ and $\overline{\mathbf{v}}$ are the minimal and maximal body velocity.

The primary objective of a compliant behavior is the motor's constraints. They are modeled as box torque constraints and added to the QP as follows

$$\underline{\boldsymbol{\tau}} \leq K\mathbf{e} - B\dot{\mathbf{e}} \leq \overline{\boldsymbol{\tau}} . \tag{3.23}$$

The other box constraints are the bounds over the parameters in $\mathbf{X}$:

$$\begin{cases} \underline{\ddot{\mathbf{q}}} \leq \ddot{\mathbf{q}} \leq \overline{\ddot{\mathbf{q}}} \\ \boldsymbol{\lambda} \geq \mathbf{0} \\ \mathbf{K} \geq \mathbf{0} \\ \mathbf{B} \geq \mathbf{0} \end{cases} . \tag{3.24}$$

Important note: in post-impact, constraints on joint limits and velocity limits are purposely not inserted as constraints in the QP. The reason for this is that we have no control over the impact behavior. Indeed, the impact is imposed to the robot in a very limited time. If it is large enough, the generated velocity would make the QP fail to find a feasible solution so the robot would remain stiff. Thus, the main advantage of not taking limits as constraints is that the robot will always comply until it has fully absorbed the post-impact dynamics or until it reaches a mechanical stop (joint limit). On the other hand, in case the impact is not large enough, nothing guarantees that the robot will not reach a joint limit. In order to ensure that the joints are kept inside their limits, a basic strategy would be to give high weight and stiffness to a posture task instead. This amounts to changing the 'priority' level, i.e. shifting the joint limit constraints from the constraint set to the cost function in the QP.

Finally, the QP writes as follow:

$$\begin{cases} \min\limits_{\ddot{\mathbf{q}},\boldsymbol{\lambda},\mathbf{K},\mathbf{B}} & \sum_k \omega_k E_k^{\text{sp}} + \omega_\lambda \|\boldsymbol{\lambda}\|^2 + \omega_G(\|\mathbf{K}\|^2 + \|\mathbf{B}\|^2) , \\ s.t. & Eq.\ (3.9), Eq.\ (3.22) - (3.24) \end{cases} \tag{3.25}$$

where k is an index over the tasks (posture, CoM). The number of parameters is equal to $\dim(\mathbf{X}) = N_X = N_{\text{dof}} + N_c N_g + 2N_m$, which is almost three times the number of variables of the more usual form of the QP used for general-purpose control (i.e. without $\mathbf{K}$ and $\mathbf{B}$). In order to improve the performance, we choose to restrain the

gain adaptation only to a selected set of joints directly involved in impact absorption. We propose to select all the motors in the kinematic chain between the end-effector contact points and the root of the kinematic chain of the robot. For example, if a contact is on the hand (actually the wrist on the HRP-4 robot), then the motors of the elbow and the shoulder are retained.

Once the joints are selected, we only need to extract their corresponding lines in the matrices H and $J^\mathsf{T}G$ and in the vectors $\mathbf{c}$ and $\boldsymbol{\tau}$. The other joints need also to be consistent with the dynamics and the torque limits, so a new constraint is added to the QP. The constraints (Eq. (3.9) and (3.23)) are removed and the following constraints are added to the QP Eq. (3.25):

$$\begin{cases} H_\mathrm{S}\ddot{\mathbf{q}} + \mathbf{c}_\mathrm{S} - (J^\mathsf{T}G)_\mathrm{S}\boldsymbol{\lambda} = K\mathbf{e}_\mathrm{S} - B\dot{\mathbf{e}}_\mathrm{S} \\ \underline{\boldsymbol{\tau}}_\mathrm{S} \leq K\mathbf{e}_\mathrm{S} - B\dot{\mathbf{e}}_\mathrm{S} \leq \overline{\boldsymbol{\tau}}_\mathrm{S} \\ \underline{\boldsymbol{\tau}}_\mathrm{NS} \leq H_\mathrm{NS}\ddot{\mathbf{q}} + \mathbf{c}_\mathrm{NS} - (J^\mathsf{T}G)_\mathrm{NS}\boldsymbol{\lambda} \leq \overline{\boldsymbol{\tau}}_\mathrm{NS} \end{cases} \tag{3.26}$$

with the subscript S (resp. NS) designating the matrix/vector of selected (resp. non-selected) rows.

Note that both the pre-impact and post-impact stages can be performed. As all bodies are not impacting at the same time (in a front fall the knees generally impact way before the hands), this QP form allows to perform both pre-impact and post-impact in parallel. (At knees' impact, the legs are set to the complying behavior whereas the upper part of the robot continues its pre-impact stage).

Simulations

A first simulation has been set up using Matlab software with a double inverted pendulum (see Fig. 3.10).

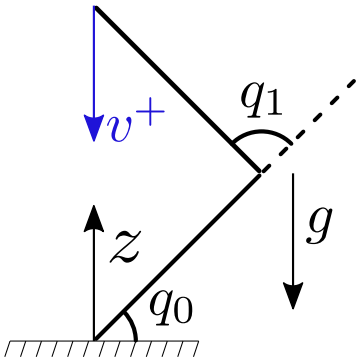
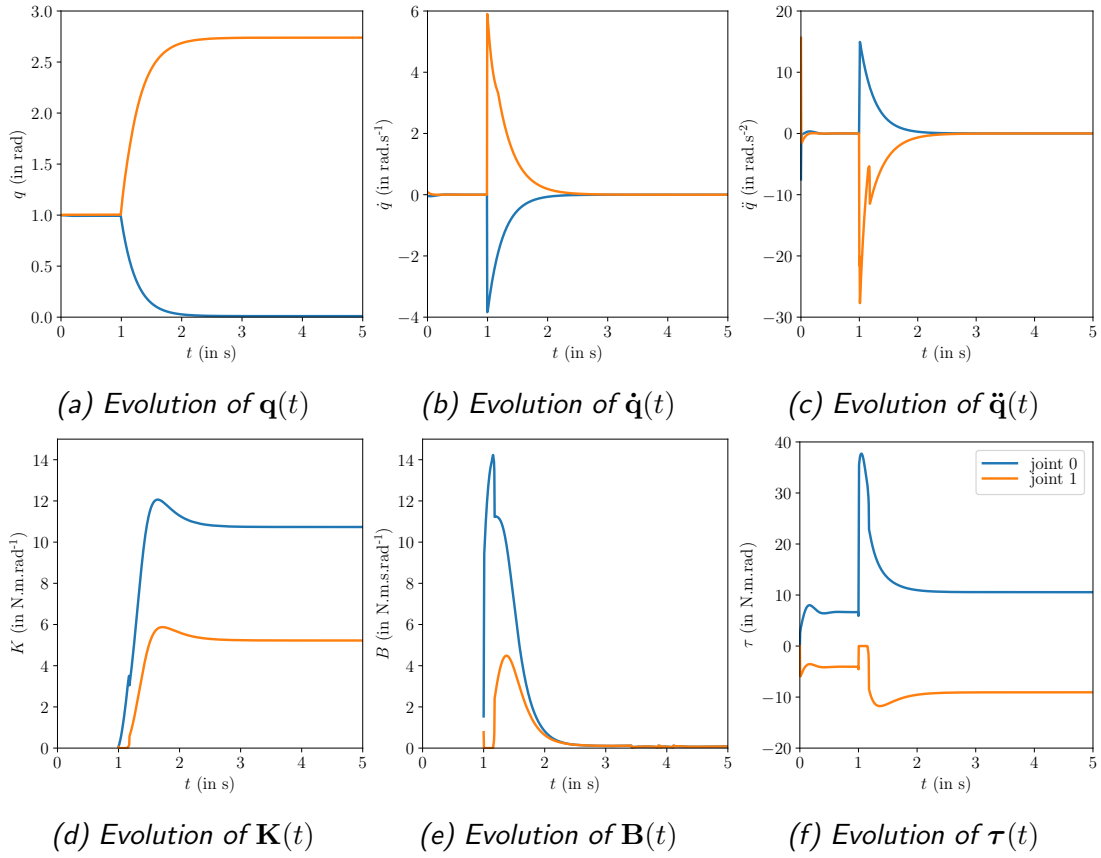


Figure 3.10 – 2-DoF system under the gravity $\mathbf{g}$*. An impact is simulated by injected a velocity* $\mathbf{v}^+$*.*

We have set the system parameters to be $\forall i \in [0,1]$: $m_i = 1\mathrm{Kg}$ the mass of the links, $l_i = 1\mathrm{m}$ the length of the link, $q_i(0) = 1\mathrm{rad}$ the joint angle at initialization, $\dot{q}_i(0) = 0\mathrm{rad.s^{-1}}$ the joint velocity at initialization, $\ddot{q}_i(0) = 0\mathrm{rad.s^{-2}}$ the joint acceleration

at initialization, $\underline{q}_i = 0$rad the lower joint limit, $\overline{q}_0 = \pi$rad and $\overline{q}_1 = \frac{7\pi}{8}$rad the upper joint limit. $\overline{\tau}_i = \text{-}\underline{\tau}_i = 60$N.m the torque limits and $g = -9.81\text{m.s}^{\text{-}2}$ the gravity in the z-axis. At $t = 1$s, the impact occurs and generates a velocity $v^+ = \text{-}5\text{m.s}^{\text{-}1}$ at the tip of the system in the z-axis. The reference values are then set so that $q_i^{\text{ref}} = q_i^-(1)$ and $q_i^{\text{ref}} = 0$.



(a) Evolution of $\mathbf{q}(t)$ *(b) Evolution of* $\dot{\mathbf{q}}(t)$ *(c) Evolution of* $\ddot{\mathbf{q}}(t)$

(d) Evolution of $\mathbf{K}(t)$ *(e) Evolution of* $\mathbf{B}(t)$ *(f) Evolution of* $\boldsymbol{\tau}(t)$

Figure 3.11 – Evolution of the 2-DoF system variables through time

The results are shown in Fig. 3.11. First of all, Fig. 3.11a and Fig. 3.11f show that the QP joint and torque limits constraints have been respected. During the post-impact phase (1s to 2.5s), the joints' velocity are smoothly decreasing and after around 2.5s (Fig. 3.11b and 3.11c), the system stops moving. As expected, the damping coefficient increases significantly just after the impact to decrease continuously (Fig. 3.11e) while the stiffness coefficient smoothly increases until reaching a constant value (Fig. 3.11d). As soon as the post-impact phase begins, the joint velocity reaches high values with an almost zero error between the reference and the current position. This means that with the damping coefficient, the QP is able to output more torque than with the stiffness which then leads to an abrupt change of B. On the contrary, when the system is about to stop, no velocity remains in the joints, so the system only needs to compensate its own weight, leaving constant values for K.

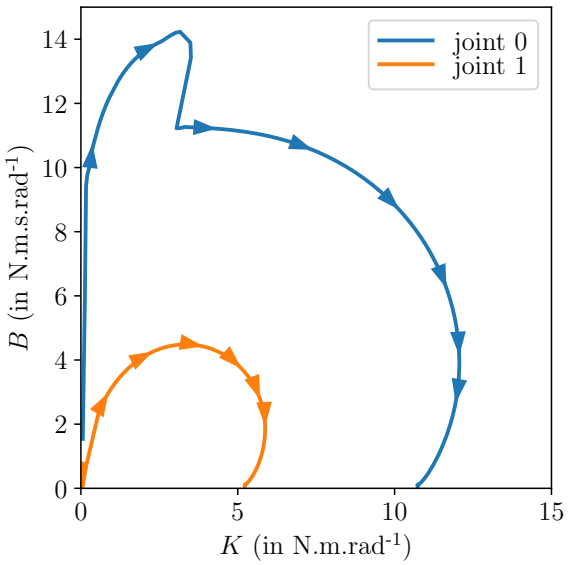


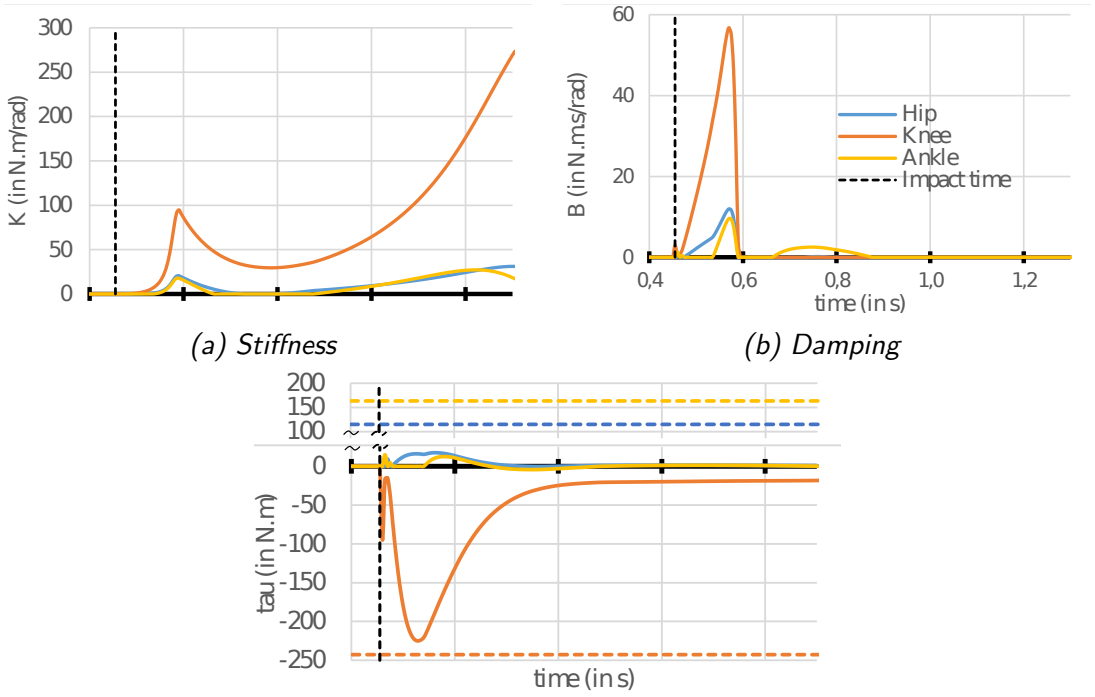
Figure 3.12 – K-B view of the 2-DoF system. The arrows show the direction of the curve through time.

To fully demonstrate the capabilities of the adaptive-QP, we performed several falling simulations of the HRP-4 robot in the Gazebo simulator.

We focus on the very first experiment consisting in dropping the robot from a given height (1m) and letting it land on its feet (at impact time $t \simeq 0.45$s with a velocity of 4.43m/s). Four methods are compared 1) keeping the robot's stiff initial gains, 2) using predefined static gains (as in Chapter 2), 3) using zero gains (shutting down the robot, in servo-off mode) and finally 4) adaptive gains (our proposed method). This experiment illustrates the post-impact gain adaptation strategy part of the paper. The pre-impact geometric reshaping part is illustrated along with the gain adaptation in all the other experiments of the video.

In order to back up our claim that the adaptive-QP complies with the post-impact dynamics and lowers the risk of damaging the robot, we chose to look at two qualifiers: (i) the IMU acceleration (in the waist) (Fig. 3.14b), and (ii) the joint positions (Fig. 3.14a). As the floating-base (waist) acceleration is proportional to the applied external forces, and as there are many contacts, we found the IMU acceleration to be a good indicator of how much total impact/contact force is applied on the structure. We use the IMU acceleration as damage quantification comparison quantity: the less acceleration there is, the better and the safer for the robot.

Let us first analyse the data from our proposed approach alone (Fig. 3.13). We can see that the damping coefficient increase very fast untill 0.6s. This is mostly due to the fact that right after the impact, the error is almost null whereas the velocity is high, hence the solver is mostly using the damping gains B (Fig. 3.13). To understand the high variation of the damping gain around 0.6s (from 60 to 0Nms/rad), we note in Fig. 3.13c that around 0.6s the torque's sign (minus) is unchanged. At this stage, the joint velocity is switching sign (from plus to minus). Considering the eq. $\tau = Ke + B\dot{e}$



(a) Stiffness *(b) Damping*

(c) Torque

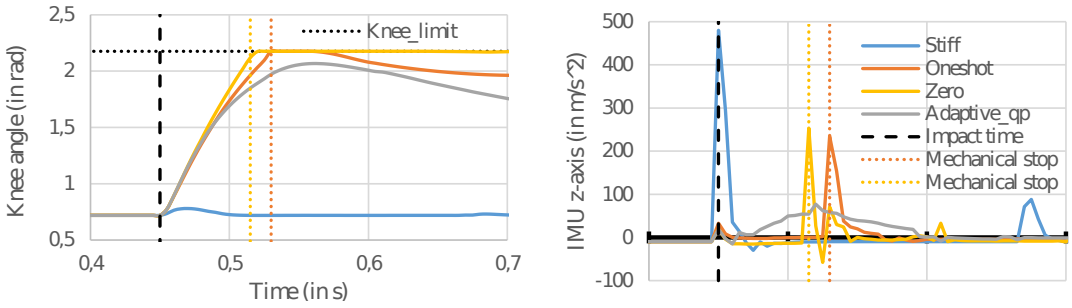
Figure 3.13 – Evolution of: (a) the stiffness, (b) the damping gains, and (c) the torque for the right leg pitch joints (hip, knee, ankle) resulting from our QP adaptive method. The dashed line is the impact time. In (c), the dashed horizontal lines are torque limits.

we can see that in order to have a negative torque with a positive velocity error ($\dot{e} = 0 - \dot{q}$) we need $B < 0$. But the constraint $B \geq 0$ enforces the non-negativity of the damping coefficient resulting in a zero value for it.

Fig. 3.14a shows that using the predefined static fixed PD gains or turning off the motors could be extremely risky since joint limits are reached fast. On the other hand, keeping the initial (stiff) gains does not make the robot reach the mechanical stops but leads to very high jerk and IMU acceleration (Fig. 3.14b) at the impact, which is a prediction of a high impact force. The proposed adaptive-QP approach avoids all these issues. It has a low jerk and a low acceleration profile while still staying under the joint limit and not reaching any mechanical stop at the joints. Fig. 3.13c shows that the adaptive-QP keeps the torques under their limits.

Discussion

We break down several potential problems relative to a real fall.



(a) Knee posture *(b) IMU*

Figure 3.14 – Comparison of (a) the knee position and (b) the z-axis of the IMU for different falling methods. Four methods are represented. Stiff: default gains, Oneshot: fixed gains as in Chapter 2, Zero: zero gains, Adaptive_qp: adaptive gains. The black dotted line is the joint limit of the knee and the colored dotted lines represent mechanical stops (joint limits).

Impact detection

Although the detection of the impact time is continuously processed, it does not need to be known perfectly in the pre-impact phase. Indeed, we set the gain to 0 prior to the near-impact (2 to 3 control loop time step) so that we are sure to be fully compliant at the impact. The estimation should return if in a near future (2 to 3 control loop time step) the impact occurs. If so, the gains are set to 0. Then an impact detection is needed in order to launch the adaptive-QP. The detection of the impact can be reliable enough by combining data from force sensors, IMU acceleration and/or joint velocities outputs. The process of the above principles is illustrated by the timeline in Fig. 3.15.

Gains stability

The system (Eq. (3.26)) assumes that the gains K and B are treated separately. Although in general K and B are coupled through system stability constraints (e.g. through the relation $B = 2\sqrt{K}$), we consider here that during the fall we are in a transition state where the main purpose is to generate enough torque to slow down the system, allowing temporary decoupled K and B. The system (Eq. (3.26)) states that the gains K and B are treated separately. Once the velocity is down (or close) to zero, the gains can be restored to a given control objectives. When all the gains have been reset to a stable state, the robot can get up and eventually pursue with its previous tasks.

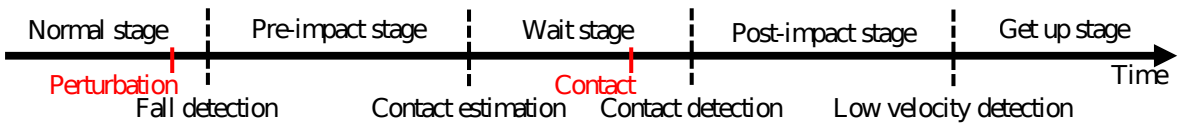


Figure 3.15 – Fall timeline in a real situation.

Torque-based controlled robots

Our approach applies also to torque-based humanoids. Torque control humanoids, e.g. Nori et al. [2015]Albu-Schäffer et al. [2007] implement a low-level torque controller in the following general form:

$$\mathbf{u} = f(\boldsymbol{\tau}_d, \dot{\boldsymbol{\tau}}_d, \ddot{\boldsymbol{\tau}}_d, \boldsymbol{\tau}, \dot{\boldsymbol{\tau}}, \boldsymbol{\kappa})\,. \tag{3.27}$$

The voltage u is computed using several parameters such as actuator identified constants, Coulomb friction and viscous friction coefficient compensation or any other compensation terms κ. There is also a PID control on the torque error $\tau_d - \tau$, with perturbation rejection (part of κ), etc. There are several implementations of the torque controller. But, the main purpose of all low-level torque controllers is to have a very good torque tracking, and changing the gains of the torque error parts would not influence the compliance of the robot: it would influence the backdrivability, but not the compliance. Notice that, above the torque controller bandwidth (like in impacts), the robot is as rigid as in position controlled humanoids. One possible way to have a adaptive compliance is to implement on top of the low-level control, the following law:

$$\boldsymbol{\tau}_d = K(\mathbf{q}_d - \mathbf{q}) + B(\dot{\mathbf{q}}_d - \dot{\mathbf{q}})\,, \tag{3.28}$$

which is exactly what we implement directly on the low-level position controlled robot, which adaptation can be handled straightforwardly by the QP. If for some reason one wants to use an adaptive-QP controller at the low-level torque control (Eq. (3.27)) it is possible only if there are no derivative terms of the torque unless they can explicitly be expressed in the time-discrete domain.

Actuator dynamics

We can easily include the actuator dynamics in order to use our method at the voltage level. Let $\boldsymbol{\tau}_m = K_c\mathbf{i}$ be the motor torque proportional to the current $\mathbf{i}$, $\mathbf{e} = K_e\dot{\mathbf{q}}_m$ the back electromotive force proportional to the motor velocity $\dot{\mathbf{q}}_m$, $\mathbf{u} = \mathbf{e} + L\frac{d\mathbf{i}}{dt} + R\mathbf{i}$ the electrical equation and $\boldsymbol{\tau}_m = J\ddot{\mathbf{q}}_m + f\dot{\mathbf{q}} + \boldsymbol{\Gamma}_{\text{load}}$ the mechanical equation. Neglecting the inductance term (for simplicity) and substituting the current and changing the load by the equation of motion $H\ddot{\mathbf{q}} + C\dot{\mathbf{q}} + \mathbf{g} = \boldsymbol{\Gamma}_{\text{load}}$, the following equation comes out:

$$\mathbf{u} = \frac{R}{K_c}\left(JN^2 + H\right)\ddot{\mathbf{q}} + \left(\frac{R}{K_c}\left(f + C\right) + K_eN\right)\dot{\mathbf{q}} + \mathbf{g}\,, \tag{3.29}$$

with R the actuator electrical resistance, J the motors inertia, N the gears ratio, f the motors viscous friction. The QP can be rewritten at the voltage level and we have:

$$\mathbf{u} = K\mathbf{e} + B\dot{\mathbf{e}}\,. \tag{3.30}$$

Of course, with the actuator model, the gains values will change but the methodology is the same. If one can access to the actuator data, they can be easily included to the QP.

Conclusion

We proposed an original way of addressing falls in a cluttered environment. First, an active reshaping fall strategy prepares the robot to the impact from the moment the fall is detected and up to just before the impact occurs. We showed with a simple 1-DoF system how important the pre-impact phase is. This study also demonstrated the importance of searching for a time-varying stiffness and a damping rather than fixed values. During the post-impact stage, a QP controller allows the robot to become compliant in order to absorb the impact energy while satisfying its structural constraints.

In order to implement this strategy on a real robot, two modules are necessary and were assumed as available black-boxes: robot state estimation and landing surfaces candidates computation, both can be provided by SLAM in future work.

For now, sliding contacts are not perfectly handled in the QP. A temporary solution we implemented was to release the tangent space dof of the contacts to allow sliding without any friction.

This part of the work can fill the parts under the *Ukemi box* of Fig. 1.10. It indeed proposes a way to handle falls in cluttered environment and aims to reduce the damages on the robot.

CHAPTER 4

Polytope-based model predictive control for compliance

In the previous chapter we introduced the notion of pre-impact and post-impact and we explained why pre-impact phase is important to accomplish a compliant behavior in the post-impact phase. We developed a strategy to fall in a cluttered environment and prepare the robot to the impact. After the impact, the robot is governed by an adaptive-QP that tunes PD gains on-line. The adaptive-QP is a local control and can only foresee 5ms in the future, therefore it cannot ensure that the robot will not reach the joint limits at a longer horizon. This chapter extends the adaptive-QP with additional tools that seek for solutions aver a longer horizon of time.

Reduced Dynamic Model

We approximate the whole-body dynamics at the CoM of the robot and search for contact forces that compensate for both gravity and post-impact linear momentum. The Newton equation of motion that governs the acceleration of the CoM is:

$$\mathbf{F} = M(\ddot{\mathbf{s}} - \mathbf{g}), \tag{4.1}$$

with M the total robot mass, $\ddot{\mathbf{s}}$ the acceleration of the CoM, $\mathbf{g}$ the gravity vector and $\mathbf{F}$ the net contact force (sum of all external forces) applied to the robot.

Under the assumption that the initial linear momentum $M\dot{\mathbf{s}}^0$ at impact is known, the goal of post-impact absorption is to find a trajectory $\ddot{\mathbf{s}}(t)$ such that, at time $t = t_f$, $\dot{\mathbf{s}}(t_f) = 0$, meaning that all the linear momentum accumulated while falling has been driven out of the system. Achieving this task requires proper control of the net contact force $\mathbf{F}$.

Actuation constraints

Let us consider again the full-body dynamics equation of motion of the system

$$H\ddot{\mathbf{q}} + \mathbf{c} = J^{\mathsf{T}}\mathbf{F} + S_M^{\mathsf{T}}\boldsymbol{\tau}. \tag{4.2}$$

$\mathbf{F} = G\boldsymbol{\lambda}$ is the stacked vector of contact forces and S_M is a selection matrix of actuated joints among all the degrees of freedom (DoFs) of the robot (hence the matrix that excludes the floating-base DoFs from the total DoFs Bouyarmane et Kheddar [2012].

A fundamental property of the humanoid kinematic tree topology is that it comprises at least 4 limbs (two legs and two arms), indexed here with the variable $c \in \{\mathrm{lh}, \mathrm{rh}, \mathrm{lf}, \mathrm{rf}\}$, that extend from the root of the kinematic tree to the end-effectors (respectively the two hands and two feet). The contact force applied at the end-effector c is denoted $\mathbf{F}_c$ (therefore we have $\mathbf{F} = (\mathbf{F}_c)_{c\in\{\mathrm{lh,rh,lf,rf}\}}$). The rest of the DoFs, including the floating base are indexed with 0. Eq. (4.2) can be rewritten as

$$\begin{bmatrix} H_0 \\ H_{\mathrm{lh}} \\ H_{\mathrm{rh}} \\ H_{\mathrm{lf}} \\ H_{\mathrm{rf}} \end{bmatrix} \ddot{\mathbf{q}} + \begin{bmatrix} \mathbf{c}_0 \\ \mathbf{c}_{\mathrm{lh}} \\ \mathbf{c}_{\mathrm{rh}} \\ \mathbf{c}_{\mathrm{lf}} \\ \mathbf{c}_{\mathrm{rf}} \end{bmatrix} = \begin{bmatrix} J_{0,\mathrm{lh}}^{\mathsf{T}} & J_{0,\mathrm{rh}}^{\mathsf{T}} & J_{0,\mathrm{lf}}^{\mathsf{T}} & J_{0,\mathrm{rf}}^{\mathsf{T}} \\ J_{\mathrm{lh}}^{\mathsf{T}} & 0 & 0 & 0 \\ 0 & J_{\mathrm{rh}}^{\mathsf{T}} & 0 & 0 \\ 0 & 0 & J_{\mathrm{lf}}^{\mathsf{T}} & 0 \\ 0 & 0 & 0 & J_{\mathrm{rf}}^{\mathsf{T}} \end{bmatrix} \begin{bmatrix} \mathbf{F}_{\mathrm{lh}} \\ \mathbf{F}_{\mathrm{rh}} \\ \mathbf{F}_{\mathrm{lf}} \\ \mathbf{F}_{\mathrm{rf}} \end{bmatrix} + \begin{bmatrix} \boldsymbol{\tau}_0 \\ \boldsymbol{\tau}_{\mathrm{lh}} \\ \boldsymbol{\tau}_{\mathrm{rh}} \\ \boldsymbol{\tau}_{\mathrm{lf}} \\ \boldsymbol{\tau}_{\mathrm{rf}} \end{bmatrix}, \tag{4.3}$$

where H_c, $\mathbf{c}_c$, $\boldsymbol{\tau}_c$ are the rows of respectively H, $\mathbf{c}$ and $\boldsymbol{\tau}$ and J_c is the *reduced* Jacobian of the contact points at the end-effector c to the root of the kinematic tree, with respect only to the set of joints of the considered limb (and not with respect to all the DoFs of the robot). The contact force at limb $c \in \{\mathrm{lh}, \mathrm{rh}, \mathrm{lf}, \mathrm{rf}\}$. $\mathbf{F}_c$, only affects the part of the dynamics equation in $\{c, 0\}$:

$$H_c\ddot{\mathbf{q}} + \mathbf{c}_c = J_c^{\mathsf{T}}\mathbf{F}_c + \boldsymbol{\tau}_c. \tag{4.4}$$

Using the reduced Jacobian, the kinematic contact constraint at the end-effector c is expressed in acceleration-form as:

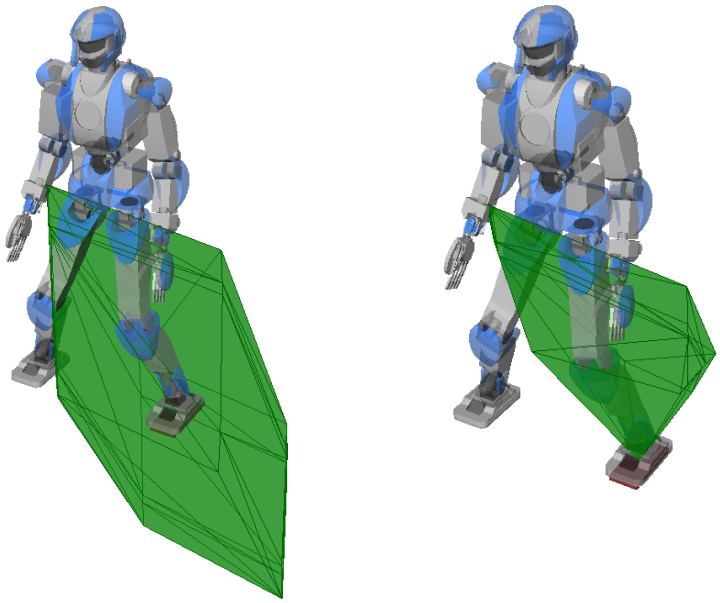
$$J_c\ddot{\mathbf{q}} + \dot{J}_c\dot{\mathbf{q}} = \mathbf{0}. \tag{4.5}$$

Substituting Eq. (4.4) in Eq. (4.5) yields

$$\begin{aligned} J_c H_c^{-1} J_c^{\mathsf{T}}\mathbf{F}_c &= J_c H_c^{-1}(\mathbf{c}_c - \boldsymbol{\tau}_c) - \dot{J}_c\dot{\mathbf{q}} \\ \Lambda_c^{-1}\mathbf{F}_c &= -J_c H_c^{-1}\boldsymbol{\tau}_c + J_c H_c^{-1}\mathbf{c} - \dot{J}_c\dot{\mathbf{q}} \\ \mathbf{F}_c &= -\Lambda_c A_c\boldsymbol{\tau}_c + \mathbf{d}_c \\ \mathbf{F}_c &= C_c\boldsymbol{\tau}_c + \mathbf{d}_c. \end{aligned} \tag{4.6}$$

The matrix Λ is the operational space inertia matrix at point c, C_c and $\mathbf{d}_c$ provide an affine mapping from torque limits to contact force limits and depends on $\mathbf{q}$ and $\dot{\mathbf{q}}$. The $\mathcal{H}$-representation of the torque-constraint polytope (see Appendix A) is:

$$\underline{\boldsymbol{\tau}}_c \;\leq\; \boldsymbol{\tau}_c \;\leq\; \overline{\boldsymbol{\tau}}_c \tag{4.7}$$



(a) *(b)*

Figure 4.1 – Torque-limited friction polytopes. Feasible contact forces at the left foot of HRP-4 under (a) leg torque limits and (b) both torque limits and friction constraints..

where $\boldsymbol{\tau}_c$ is the vector of selected joint torques for the contact c (corresponding to the limb at hand, *e.g.* left-leg joints for a left-foot contact), $\underline{\boldsymbol{\tau}}_c$ and $\overline{\boldsymbol{\tau}}_c$ are respectively the lower and upper torque limits of the actuators. Applying Eq. (4.6) yields the $\mathcal{V}$-representation of the torque-limited polytope of feasible end-effector forces at the contact point, depicted in Fig.4.1a. Using the double-description method, we convert it to $\mathcal{H}$-representation:

$$G_c^{\text{actuators}} \mathbf{F}_c \leq \mathbf{h}_c^{\text{actuators}}. \tag{4.8}$$

Distribution of gravity and linear momentum

Our primary idea is to make each limb $c \in \{\text{lh}, \text{rh}, \text{lf}, \text{rf}\}$ of the robot contribute, in post-impact, to a "share" of the pre-impact linear momentum absorption, summing up all contributions into the resultant equation of motion:

$$\sum_{c \in \{\text{lh,rh,lf,rf}\}} \mathbf{F}_c = M(\ddot{\mathbf{s}} - \mathbf{g}) \,. \tag{4.9}$$

This can be conceptually interpreted as "splitting" the CoM into four virtual mass points with state vectors $(\mathbf{s}_c, \dot{\mathbf{s}}_c)$, each going towards one of the limb extremities in contact, and each with a force $\mathbf{F}_c$ applied to it. Hence we are solving for the system of

four differential equations for the state variables $(\mathbf{s}_c, \dot{\mathbf{s}}_c)$:

$$\mathbf{F}_c = M(\ddot{\mathbf{s}}_c - \mathbf{g}_c)\,, \tag{4.10}$$

where $\mathbf{g}_c$ is a quantity to be defined below (gravity force distribution problem) and where the initial condition for each component is also to be defined below (initial impact momentum distribution problem). The force $\mathbf{F}_c$ is used to decelerate its respective CoM component until full initial momentum absorption, while being contrained to lie inside its respective actuation limit polytope (Section 4.1.1).

The momentum being an additive quantity, when virtually splitting the CoM into four virtual components, the initial momentum has to be distributed among the four components. Similarly, the total gravity force applied on the CoM has also to be split into four components, each applied to one of the four virtual CoM components. Therefore, in the following, we distribute the amount of gravity and initial linear momentum that each contact will support. This distribution must satisfy Eq. (4.9). Additionally, we want to make sure that the distribution is optimal with respect to the actuation constraint polytope at each contact.

Assuming that the linear momentum $\mathbf{p}$ after impact is known, we look for a corresponding force quantity $\mathbf{f}^a$ that would be physically consistent with the problem of distribution in the respective contact force polytopes. The force being a time-derivative of the linear momentum, we can write:

$$\mathbf{f} = \frac{d\mathbf{p}}{dt} = M\frac{d\dot{\mathbf{s}}}{dt}. \tag{4.11}$$

In the worst case scenario where the robot is completely rigid (non compliant) and the coefficient of restitution of the impact is zero, the impact force corresponding to the momentum $M\dot{\mathbf{s}}^0$ is:

$$\mathbf{f} = \lim_{h\to 0} M\frac{\dot{\mathbf{s}}^0}{h}. \tag{4.12}$$

Because this form is non-linear in h, we set $k = 1/h$ (hence k can be seen as a "gain" that maps a linear momentum quantity to a force quantity). The limit becomes

$$\mathbf{f} = \lim_{k\to\infty} kM\dot{\mathbf{s}}^0. \tag{4.13}$$

Then we define $\mathbf{f}^a$ as the exact applied force

$$\mathbf{f}^a \triangleq kM\dot{\mathbf{s}}^0 \tag{4.14}$$

where k is a parameter that approach infinity. $\mathbf{f}^a$ corresponds to the *average impact force* for impact time $1/k$.

Being already a force quantity, the gravity force:

$$\mathbf{f}^g = M\mathbf{g}\,, \tag{4.15}$$

can be readily distributed in the polytopes.

The idea now is to find which contact will better handle $\mathbf{f}^g$ and which will better handle $\mathbf{f}^a$. For example, in the case of Fig. 4.2, the feet of the robot are more suitable for compensating the gravity than the hands. From Eq. (A.1) of Appendix A we split α_i in two variables α_i^g and α_i^a corresponding respectively to the percentage of $\mathbf{f}^g$ and $\mathbf{f}^a$ to address to the contact.

$$\sum_i \alpha_i = \sum_i \alpha_i^g + \alpha_i^a. \tag{4.16}$$

This is done for each contact c. Using Eq. (A.1) and Eq. (4.14) – (4.16) we can write the following constraints

$$\sum_c \sum_i \alpha_{c,i}^g \mathbf{v}_{c,i} = -M\mathbf{g} \tag{4.17}$$
$$\sum_c \sum_i \alpha_{c,i}^a \mathbf{v}_{c,i} = -kM\dot{\mathbf{s}}^0 \tag{4.18}$$
$$\sum_i \alpha_{c,i}^g + \alpha_{c,i}^a \leq 1 \tag{4.19}$$
$$\alpha_{c,i}^g \geq 0 \tag{4.20}$$
$$\alpha_{c,i}^a \geq 0 \tag{4.21}$$
$$k \geq 0. \tag{4.22}$$

Constraints Eq. (4.17) and (4.18) distribute forces considering the polytope of each contact. Constraint Eq. (4.19) is a generalization of Eq. (A.1). Substituting the equality to an inequality does not change the definition of the polytope and relax the constraint. And finally, Eq. (4.20) – (4.22) make the variables consistent with the polytope definition Eq. (A.1) and the force generated at the CoM definition Eq. (4.14).

To have a good distribution over the feet and the hands (*i.e.* to avoid having the robot's weight supported by one foot), we design the following cost functions

$$\begin{cases} \| \sum_i \alpha_{\mathrm{rf},i}^g \mathbf{v}_{\mathrm{rf},i} - \sum_i \alpha_{\mathrm{lf},i}^g \mathbf{v}_{\mathrm{lf},i} \|^2 \\ \| \sum_i \alpha_{\mathrm{rh},i}^a \mathbf{v}_{\mathrm{rh},i} - \sum_i \alpha_{\mathrm{lh},i}^a \mathbf{v}_{\mathrm{lh},i} \|^2 \\ -k \end{cases} \tag{4.23}$$

where rf (resp. lf) stands for right foot (resp. left foot) and rh (resp. lh) for right hand (resp. left hand).

As we want the variable k to approach infinity, a cost function is set to maximize its value in the problem.

The overall system, with linear constraints Eq. (4.17) – (4.22) and quadratic cost functions Eq. (4.23) is a QP problem. Later on, we call it the Force Distribution Quadratic Programming (FDQP). Fig. 4.2 shows a possible repartition of the forces over two contact points. The FDQP looks for a solution that compensates the force from gravity and maximizes the force generated against linear momentum.

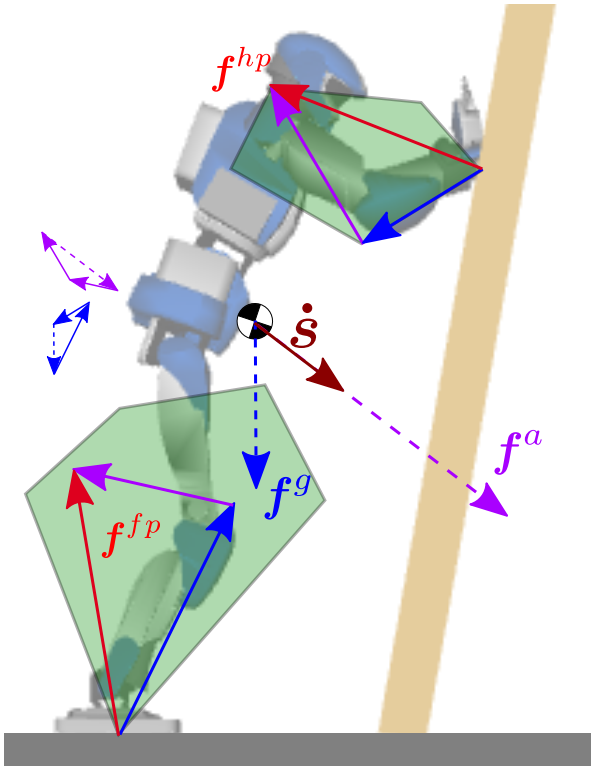


Figure 4.2 – Optimized contact force distribution. 2D illustration of a potential scenario. Dotted arrows $\mathbf{f}^g$ *and* $\mathbf{f}^a$ *are forces to be distributed. Plain arrows show their repartition inside force polytopes:* $\mathbf{f}^{fp}$ *and* $\mathbf{f}^{hp}$ *have to remain inside their respective polytope. The dotted vector* $\mathbf{f}^a$ *is taken so as to maximize the linear momentum coming from the CoM velocity* $\dot{\mathbf{s}}$.

Once the solution is found, α_c^g, α_c^a and k can be computed using the FDQP output:

$$-M\mathbf{g}_c = \sum_i \alpha_{c,i}^g \mathbf{v}_{c,i}\,, \tag{4.24}$$

$$-kM\dot{\mathbf{s}}_c^0 = \sum_i \alpha_{c,i}^a \mathbf{v}_{c,i}\,. \tag{4.25}$$

These two latter equations make Eq. (4.10) fully specified, with the specification of the constant vector $\mathbf{g}_c$ and the initial condition $\dot{\mathbf{s}}^0$.

CoM trajectory solution

We solve the system (Eq. (4.10)) using a Linear Model Predictive Control (LMPC). Let $\mathbf{x}_c = [\mathbf{s}_c^\mathsf{T}\ \dot{\mathbf{s}}_c^\mathsf{T}]^\mathsf{T}$ be the parameter vector. Eq. (4.10) can be put into matrix form:

$$\dot{\mathbf{x}} = \begin{bmatrix} 0_3 & I_3 \\ 0_3 & 0_3 \end{bmatrix} \mathbf{x} + \begin{bmatrix} 0_3 \\ 1_3 M^{-1} \end{bmatrix} \mathbf{u} + \begin{bmatrix} \mathbf{0}_3 \\ \mathbf{g}_c \end{bmatrix}$$
$$\dot{\mathbf{x}} = \mathcal{A}\mathbf{x} + \mathcal{B}\mathbf{u} + \mathcal{E}_c.$$

where $\dot{\mathbf{x}} = [\dot{\mathbf{s}}_c^\mathsf{T} \ \ddot{\mathbf{s}}_c^\mathsf{T}]^\mathsf{T}$. This is a continuous time system with $\mathbf{u} = \mathbf{F}_c$, the control vector, M the total mass of the system and $\mathbf{g}_c$ the contact distributed part of the gravity (computed with Eq. (4.24)). After discretization with a sampling time T, we get

$$\mathbf{x}_{k+1} = A\mathbf{x}_k + B\mathbf{u}_k + \mathbf{E}_c \tag{4.26}$$

and assuming the input is discretized with a zero-order block

$$\begin{cases} A = \begin{bmatrix} I_3 & I_3 T \\ 0_3 & I_3 \end{bmatrix} \\ B = M^{-1} \begin{bmatrix} I_3 \frac{T^2}{2} \\ I_3 T \end{bmatrix} \\ \mathbf{E}_c = \begin{bmatrix} \mathbf{g}_c \frac{T^2}{2} \\ \mathbf{g}_c T \end{bmatrix} \end{cases}. \tag{4.27}$$

The sampling time is not necessarily set to the robot control loop time. Indeed the greater the LMPC's sampling time the wider is the horizon of time. Then an interpolation is done to get the values at each control loop time. By recursion we can compute a prediction of the behavior of the simplified system Audren et al. [2014]. We only need to know the initialization parameter of the system.

By recursively calculating Eq. (4.26) we get (see Section B)

$$\mathbf{X}_N = \Phi\mathbf{x}_0 + \Psi\mathbf{U} + \boldsymbol{\zeta}_c. \tag{4.28}$$

Note that $\mathbf{x}_0 = [\mathbf{s}^{0\,T} \ \dot{\mathbf{s}}_c^{0\,T}]^\mathsf{T}$ where $\dot{\mathbf{s}}_c^0$ is the CoM distributed velocity (Eq. (4.25)) and $\mathbf{s}^0$ is the intial position of the CoM (which is the inital position of all four virtual CoMs).

The objective function of the LMPC is

$$E^d = (\mathbf{x}_N - \mathbf{x}^{\text{target}})\omega_x(\mathbf{x}_N - \mathbf{x}^{\text{target}}) + \mathbf{U}^\mathsf{T}\omega_U\mathbf{U} \tag{4.29}$$
$$= \mathbf{U}^\mathsf{T}Q\mathbf{U} + 2l_c^\mathsf{T}\mathbf{U}. \tag{4.30}$$

The target $\mathbf{x}^{\text{target}} = [\mathbf{s}^{\text{target}\,\mathsf{T}} \ \dot{\mathbf{s}}^{\text{target}\,\mathsf{T}}]^\mathsf{T}$ is such that $\mathbf{s}^{\text{target}} = \mathbf{s}^0$ and $\dot{\mathbf{s}}^{\text{target}} = \mathbf{0}$. The weights ω_x and ω_U are chosen so that the priority is to reach a zero velocity, then maintain the position and lastly minimize the effort. Briefly, this means that $\omega_x(3..6) \gg \omega_x(0..2) > \omega_U$. The force limit constraint (computed via Eq. (4.8)) is added to the system constraints. We then solve the LMPCs corresponding to each contact to get the vectors $\mathbf{U}$ and deduce the forces to apply at the contact points. As the adaptive-QP of the robot allows us to set a trajectory task for the CoM, the forces are converted into CoM acceleration.

$$\ddot{\mathbf{s}}(t) = \frac{1}{M}\sum_c \mathbf{F}_c(t) + \mathbf{g}. \tag{4.31}$$

Integrating forward returns $\dot{\mathbf{s}}(t)$ and $\mathbf{s}(t)$. Those three trajectories are sent to the adaptive-QP.

Finally, since the LMPCs are launched again and again and that it targets a zero velocity about $T \times N_{\text{steps}}$ seconds, the robot can never reach it completely so a stopping criterion is necessary. This criterion can simply be based on the current CoM velocity norm.

Discussion in Eq. (4.3), we have split the equation of motion into a coupling part (first row) and a decoupled, block-diagonal part (next rows). As shown in Wieber [2006], the coupling part corresponds to Newton and Euler equations. We enforce Newton's equation when combining LMPC results via Eq. (4.31), but chose to ignore Euler's equation (on the rate of change of the angular momentum) as a first approximation. This part will be further surveyed in future works. For now, we compute force polytopes by assuming that each limb is decoupled from each the others, and rely on the cost function of the FDQP to balance solutions and avoid adding unnecessary angular-momentum variations.

Simulations

The simulation sets up HRP-4 pushed against a wall, as depicted in Fig. 4.2. Here, it is assumed that the pre-impact phase (treated in Chapter 3) is already done and only the post-impact phase remains to be controlled. The control loop of HRP-4 is 5 ms, it is pushed on the back with a force of 300 N for 0.15 s. The wall is 10° inclined and we use an LMPC[1] with variables set as $N_{steps} = 15$ and $T = 15$ ms. The detection of the impact is done through the distance between the hands and the wall. As the algorithm computation time is higher than the control loop time, it launched is an estimated position of the hand touch the wall. This estimation is done with a simple Euler integration over 15 ms of the nearest point of a hand to the wall. To ensure that the adaptive-QP does not output high torque at the impact time, all tasks are removed but the minimization of the stiffness and damping coefficient. As soon as the LMPCs output the first results, a CoM trajectory task and posture task (with relatively low weight) is added. With an Intel(R) Core(TM) i7-4900MQ CPU at 2.8GHz (up to 3.8GHz) with 4 cores and 8 running threads, the time required to compute the polytopes, the distribution force QP and the LMPCs is up to 15 ms. To improve performance, the contact polytopes[2] and the LMPCs are computed in parallel.

The stopping criterion has been set so that it is triggered when the norm of CoM velocity reached 1 cm.s^{-1}. From there, the weight of the posture task is increased and the CoM trajectory is updated to a fixed set-point task at the current CoM position. We then stop the adaptive-QP and revert to the standard QP whole-body controller.

[1] https://github.com/vsamy/Copra

[2] The polyhedral library (`cddlib`) itself is not multi-threadable.

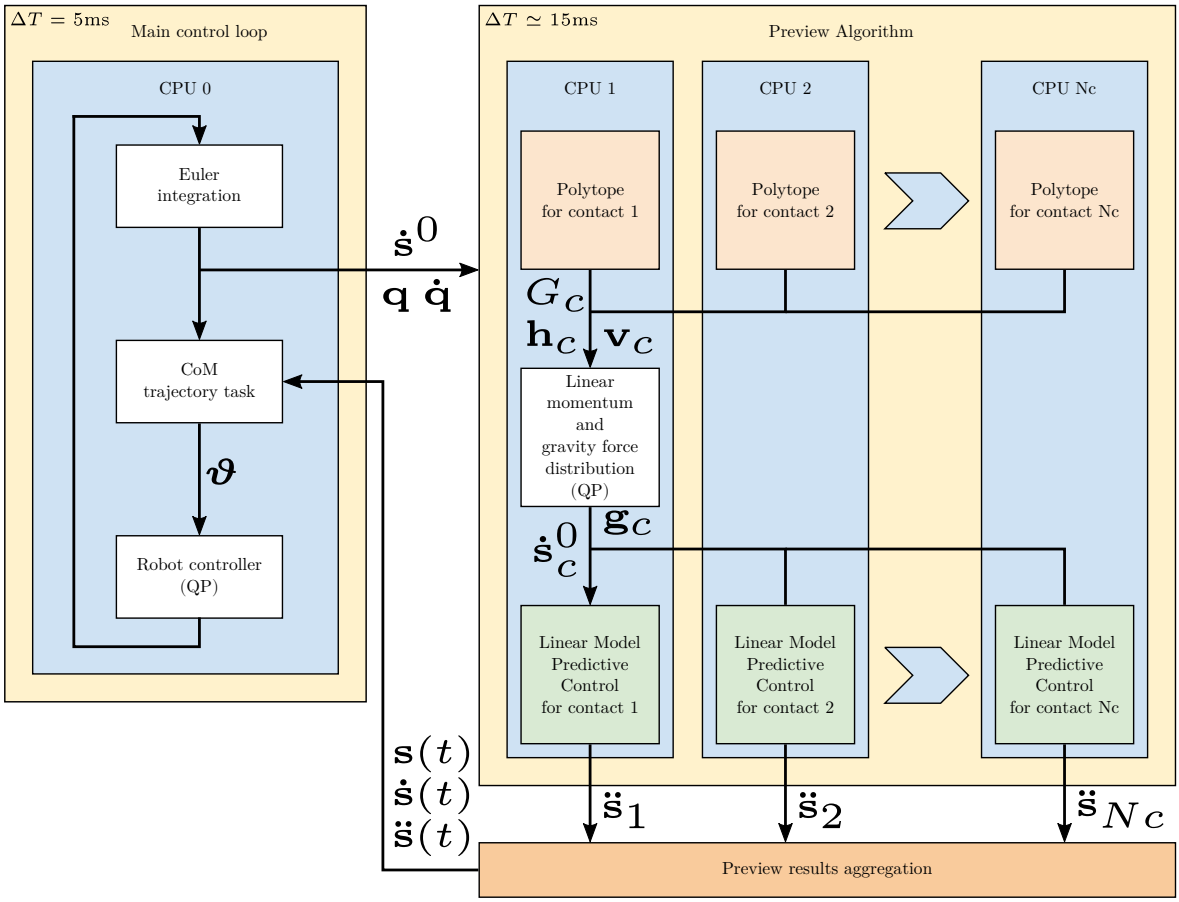


Figure 4.3 – Distribution of the preview algorithm's tasks over the Central Processing Unit (CPU)s for computing $\mathbf{s}(t)$, $\dot{\mathbf{s}}(t)$ *and* $\ddot{\mathbf{s}}(t)$. *One CPU is handling the adaptive-QP and performs the update of the robot's state. The preview algorithm is distributed over the other CPUs with one CPU per contact. Each CPU compute the Convex hull of finite set of vertices polytope representation ($\mathcal{V}$-representation) and Intersection of finite halfspaces polytope representation ($\mathcal{H}$-representation) of its contact polytope which are regrouped in one CPU for the FDQP. Then the results are reassigned to different CPUs to compute the LMPC for each contact. Finally, the desired CoM trajectory is computed and made available in shared memory for the main control loop.*

As the problem is symmetric the FDQP gives arms and legs the same amount of linear momentum and gravity hence Fig. 4.4 – 4.6b only display right side contact-s/joints. The linear momentum of the CoM (Fig. 4.4) smoothly decreases, on each axis, and reaches zero after roughtly 1.5 s. Notwithstanding the symmetrical properties of the problem, we can denote a deviation of the linear momentum on the y-axis. In Fig. 4.5, the FDQP choses to compensate the z-axis of the gravity with only the foot which is coherent since the maximal forces of the feet are on this axis. It is also interesting to note that the FDQP may give one or multiple contacts the ability to go along with the whole-body linear momentum, rather than against it. This is quite visible on Fig. 4.4 on the z-axis where after the impact time, the sum of the two feet is greater than the robot linear momentum. Thus the feet alone have to compensate

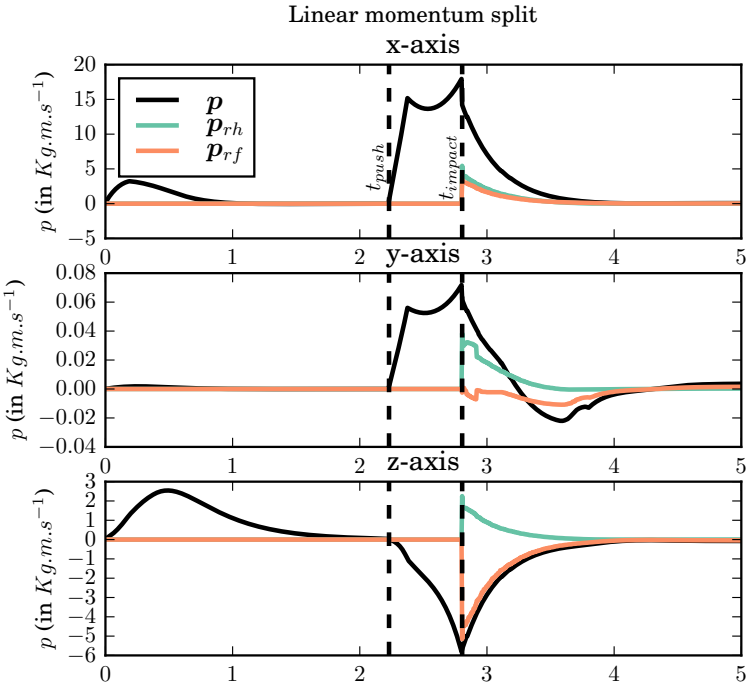


Figure 4.4 – Simulation results of the linear momentum and its split part. As the problem is symmetrical, $\mathbf{p}_{rh} = \mathbf{p}_{lh}$ and $\mathbf{p}_{rf} = \mathbf{p}_{lf}$. The black curve represents the total linear momentum at the CoM, the green curve is the split linear momentum for the hands and the orange is the split linear momentum for the feet. t_push is the time at which the robot is pushed and t_impact is the time at which the robot touches the wall.

more than needed. This is due to the fact that the FDQP gives to the hand linear momentum a sign opposite to the whole-body one. So, LMPCs attached to the hands are seeing the robot going upward while the feet sees the robot going downward with higher velocity.

The adaptive-QP is able to find the stiffness (Fig. 4.6a) and damping (Fig. 4.6b) coefficient for arms and legs. At impact time, the damping coefficient dominates the stiffness one since the error term is almost zero whereas velocity is high. It is very clear that the hip joint is making most of the effort to fulfill the tasks. From common sense, the elbow joint should have particapated more since it is the one that can directly damp the linear momentum but it is not. This can be explained by the fact that the hip motor has a torque limit 3.5 times superior to the elbow's one. Around $t = 3$ s, the knee joint takes over and is mainly set so that it maintains the height of the CoM and compensates gravity.

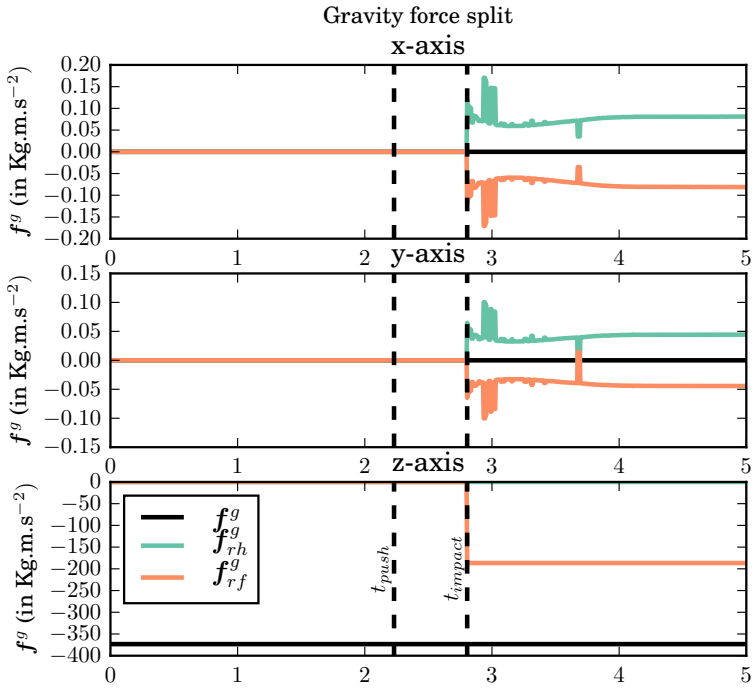
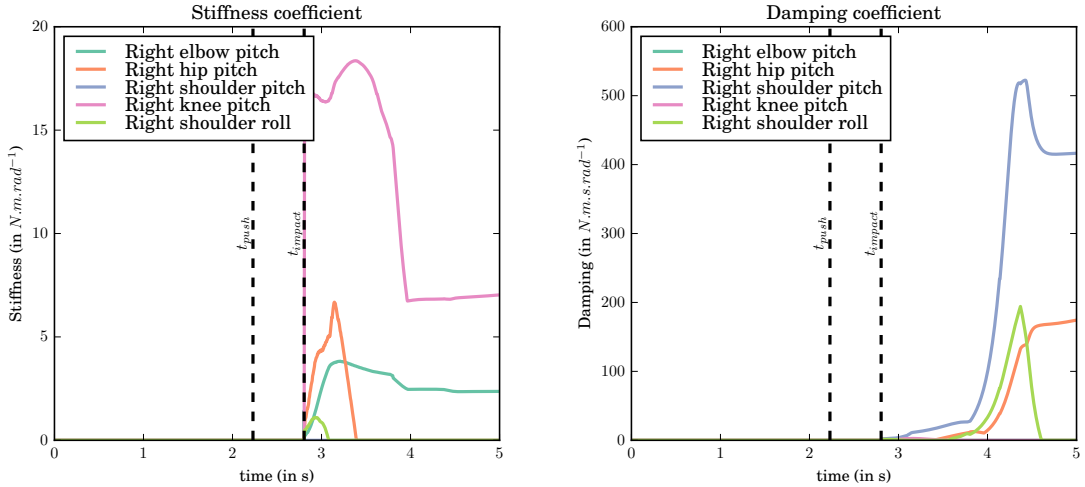


Figure 4.5 – Simulation results of the gravity force and its split part. As the problem is symmetrical, $\mathbf{F}^g_{rh} = \mathbf{F}^g_{lh}$ and $\mathbf{F}^g_{rf} = \mathbf{F}^g_{lf}$. The black curve represents the total gravity force at the CoM, the green curve is the split gravity force for the hands and the orange is the split gravity force for the feet. t_push is the time at which the robot is pushed and t_impact is the time at which the robot touches the wall.



(a) Stiffness coefficient *(b) Damping coefficient*

Figure 4.6 – Gains output of the adaptive-QP through time and for several joints. As the problem is symmetrical only the joints of the right leg and arm are shown.

Conclusion and discussion

We have presented an approach that allows a humanoid robot to actively control its compliance after falling and impacting its environment with its arms and/or legs. Parallel model predictive controllers are run at each of the four contacting limbs, the joint action of which is to absorb the undesired linear momentum accumulated during fall. These predictive controllers are instantiated by a FDQP that optimally distributes the initial momentum and gravity force across contacting limbs. Despite relying on reduced dynamic models, all predictive controllers are constrained by contact polytopes that encode both friction and joint torque limitations.

The approach was validated in full-body simulations with a model of the HRP-4 humanoid robot in falling scenarios. The simulation showed corroborating results making the linear momentum vanish after few seconds, in an active compliance way.

With the two previous chapters and the present one, we have now a complete framework for fall handling from pre-impact posture adoption, to impact-time motor PD-gains automatic adaptation, and with the present work, post-impact active compliance with LMPC to optimally absorb the impact and bring the robot to a safe rest.

Note that this method has been designed for falling humanoid robots. Even so, it can be easily extended other types of robot (especially multi-legs robots) and may also be adapted for other problems (walking, manipulation, *etc...*)

Although it is a very challenging experiment to be carried out on real hardware, we plan in a near future to test the method on HRP-4. Future work will also need to address sliding of contacts at impact points, reliable linear momentum estimation, visual estimation of the robot's surroundings for impact surfaces, and impact time detection are other open topics in fall control that are challenging and that we are actively researching.

Conclusion

In this thesis, we endowed humanoid robots with new strategies to handle falls and to prevent damaging themselves upon fall impact.

The complexity of the problem and the high experimental risks have made falling barely studied in the humanoid robotics community. The few existing papers on this subject in the literature mostly considered falls in simulation and/or with small robots. In the first part of this thesis, we reviewed all the strategies suggested by previous research. We then identified and classified what has been proposed in order to understand where we could make our contribution among the existing body of work.

Our first contribution was to set up a taxonomy of falls that characterizes the postures the robot should not be adopting when impacting the environment. With this knowledge at hand, we designed a geometrical solver that avoids fall singularities and prevents direct damage to the structure of the robot. We also experimentally tuned down the motors PD gains to comply with the impact. Simulations in Gazebo and experiments on the robot validated the concept introduced here.

We then decided to further investigate more general falls such as falls in any direction and/or in a cluttered environment. Because impacts are not directly controllable, we proposed to separate the falls into two different phases: the pre-impact and the post-impact phases. During the former, an algorithm prepares the robot to the impact *i.e.* it searches for impact points and reshapes the robot while trying to avoid fall singularities. Then we deigned an adaptive-QP to control the post-impact. This QP can tune the motors PD gains on-line and thus is able to comply to the contact forces generated after the impact.

One limitation of the adaptive-QP is that it can only solve the control problem for a very short amount of time which is generally the control loop time. Therefore, the tasks weights need to be tuned to high enough values to ensure a solution that will damp the system correctly after the impact. This motivated us to develop a strategy that to account for a time-horizon on a robot model reduced to the CoM. Using a novel polytopial representation of the torque limits in the limbs, we have been able to find reachable forces at the contact for each limb. We then proposed a FDQP that distributes the gravity force and the linear momentum among the different contacts. For each contact, a LMPC is used to find the force to apply at each contact to make the

robot comply. Finally, the LMPC solutions for each impacting limb were combined, returning as an output a trajectory for the CoM of the robot. This trajectory was finally sent as a task to the adaptive-QP.

The research in humanoid falls is only at its prelude and several problems need to be addressed before mastering this domain. First of all, the fall detection problem is still an open problem and deserves a more in-depth investigation. As mentioned in Chapter 1, fall detection algorithms should search for differences between expectation and realization so that it could be used for on type of motion controller (walking, whole-body reaching and manipulation, multi-contact locomotion, etc).

Most of the falls are considering a direct push of the robot, which is not so common in real situations. Falls are more often due to tripping or slipping. These two types of falls should be treated, and raise along the problem of contact characterization : contact position, contact reliability (whether or not the contact is safe to exploit), contact Degrees of Freedom (sliding direction if any), etc.

Other future research could also be made to extend the Fig. 1.10. These include: falls over a fence, falls that include poles (as in a subway), falls while the robot is holding something or someone. It is also possible to extend the Ukemi motion by performing a much higher dynamics reshaping (rolling on the ground).

The description and the algorithms given in Section 3.3.1 could also be better exploited for the post-impact phase. By representing the whole arm as a spring-damper system (see Fig. 4.7), estimating the velocity, and computing the effective mass at the link's contact points, it will be possible to compute the optimal coefficient K and B of the Cartesian virtual spring damper system. Once the Cartesian values are found, a mapping to the joint space could be also used.

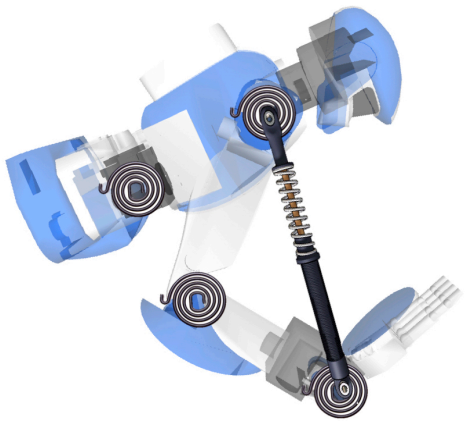


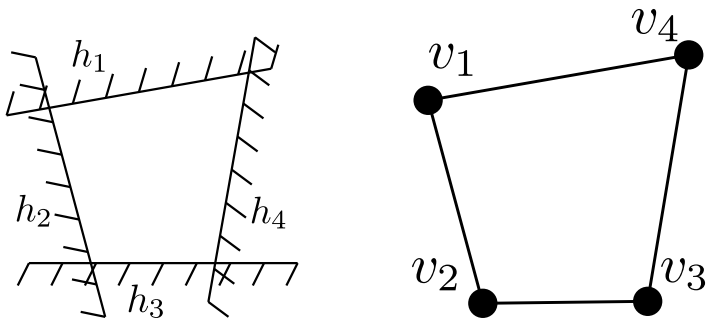
Figure 4.7 – Simulation of a spring-damper system

Although we have listed these many open questions and open problems left for future research, we still believe that our thesis contributed to identifying, formulating, and solving a number humanoid fall-related problems, and we hope that it will have contributed positively to the future of humanoid robotics with more robust humanoid robots, more autonomous robots, safer robots, and globally better humanoids.

APPENDIX A

Polytopes

Polyhedra are convex sets realized by the intersection of finitely-many halfspaces. They correspond to the inequality constraints found in linear and quadratic programs, as well as the dual constraints such as *friction cones* or *ZMP support areas* that arise in mechanics Caron et al. [2017]. A polytope is a bounded polyhedron. It can be represented equivalently (see Fig. A.1) as: (i) the intersection of finitely-many halfspaces, called the $\mathcal{H}$-representation, or (ii) the convex hull of a finite set of vertices, called the $\mathcal{V}$-representation.



(a) $\mathcal{H}$-representation (b) $\mathcal{V}$-representation

Figure A.1 – Example of the two representation of a polytope in a 2D space. (a) is the $\mathcal{H}$-representation where the polytope is defined by a set of halfspace. (b) is the $\mathcal{V}$-representation where the polytope is defined by the convex hull of its vertices.

The latter can be written mathematically as:

$$\mathcal{P} = \left\{ \mathbf{x} = \sum_{i=1}^{N} \alpha_i \mathbf{v}_i \;\middle|\; \forall i, \alpha_i \geq 0, \sum_{i=1}^{N} \alpha_i = 1 \right\} \tag{A.1}$$

where the positive coefficients α_i form a *convex combination* of the N vertices $\mathbf{v}_i$ of the

polytope. Meanwhile, the $\mathcal{H}$-representation can be concisely written as:

$$A\mathbf{x} \leq \mathbf{b} \tag{A.2}$$

where the matrix $A = [\mathbf{A}_0 \dots \mathbf{A}_N]^T$ stacks halfspace normals $\mathbf{A}_i$, so that $\mathbf{A}_i^T\mathbf{x} = b_i$ is the equation of the i^{th} supporting halfspace of the polytope. The conversion from $\mathcal{H}$-representation to $\mathcal{V}$-representation (resp. from $\mathcal{V}$-representation to $\mathcal{H}$-representation) representation of a polytope is known as the *vertex enumeration* problem. Both can be realized by the double-description algorithm Fukuda et Prodon [1996].

APPENDIX B

Linear Model Predictive Control

The LMPC, also called linear model preview control, is a common problem in robotics. It often found in walking problems such as in Kajita et al. [2003a] and allows to find a set of necessary force to reach a desired trajectory in a horizon of time. It can also be used reversely, to find the trajectory that force inputs will lead to. This appendix explains how it works and presents the Copra[1] library that implements it.

Let $\dot{\mathbf{x}} \in \mathbb{R}^n$ the state vector and $\mathbf{u} \in \mathbb{R}^m$ the control vector of the discrete system

$$\dot{x}_{k+1} = A\mathbf{x}_k + B\mathbf{u}_k + \mathbf{e}. \tag{B.1}$$

where $A \in \mathbb{R}^{n\times n}$ is the state matrix, $B \in \mathbb{R}^{n\times m}$ is the control matrix and $\mathbf{e} \in \mathbb{R}^n$ is a bias. By doing a simple recursion, we have:

$$\begin{aligned}
\mathbf{x}_1 &= A\mathbf{x}_0 + B\mathbf{u}_0 + \mathbf{e} && \text{(B.2)}\\
\mathbf{x}_2 &= A\mathbf{x}_1 + B\mathbf{u}_1 + \mathbf{e} && \text{(B.3)}\\
&= A(A\mathbf{x}_0 + B\mathbf{u}_0 + \mathbf{e}) + B\mathbf{u}_1 + \mathbf{e} && \text{(B.4)}\\
&= A^2\mathbf{x}_0 + [AB\ B][\mathbf{u}_0^T\ \mathbf{u}_1^T]^T + A\mathbf{e} + \mathbf{e} && \text{(B.5)}\\
\mathbf{x}_3 &= A\mathbf{x}_2 + B\mathbf{u}_2 + \mathbf{e} && \text{(B.6)}\\
&= A(A^2\mathbf{x}_0 + [AB\ B][\mathbf{u}_0^T\ \mathbf{u}_1^T]^T + A\mathbf{e} + \mathbf{e}) + B\mathbf{u}_1 + \mathbf{e} && \text{(B.7)}\\
&= A^3\mathbf{x}_0 + [A^2B\ AB\ B][\mathbf{u}_0^T\ \mathbf{u}_1^T\ \mathbf{u}_2^T]^T + A^2\mathbf{e} + A\mathbf{e} + \mathbf{e} && \text{(B.8)}
\end{aligned}$$

It is easy to show that after N step, we get:

$$\mathbf{X}_N = \Phi\mathbf{x}_0 + \Psi\mathbf{U} + \boldsymbol{\zeta} \tag{B.9}$$

with

$$\begin{aligned}
\mathbf{X}_N &= [\mathbf{x}_0^\mathsf{T}\ \mathbf{x}_1^\mathsf{T}\ \dots\ \mathbf{x}_N^\mathsf{T}]^\mathsf{T} && \text{(B.10)}\\
\mathbf{U} &= [\mathbf{u}_0^\mathsf{T}\ \mathbf{u}_1^\mathsf{T}\ \dots\ \mathbf{u}_{N-1}^\mathsf{T}]^\mathsf{T} && \text{(B.11)}
\end{aligned}$$

[1] https://github.com/vsamy/Copra

and where Φ, Ψ and $\boldsymbol{\zeta}$ are defined as

$$\Phi = \begin{bmatrix} A \\ A^2 \\ \vdots \\ A^N \end{bmatrix}, \ \Psi = \begin{bmatrix} B & 0 & \dots & 0 \\ AB & B & \ddots & \vdots \\ \vdots & \vdots & \ddots & 0 \\ A^{N-1}B & A^{N-2}B & \dots & B \end{bmatrix}, \ \boldsymbol{\zeta} = \begin{bmatrix} \mathbf{E} \\ A\mathbf{E} + \mathbf{E} \\ \vdots \\ \sum_{i=0}^{i=N-1} A^i \mathbf{E} \end{bmatrix} \tag{B.12}$$

Eq. (B.9) is the system to give to the library.

```
#include <memory>
#include <utility>
#include <Eigen/Core>
#include <copra/PreviewSystem.h>

// Create a system with:
// the state matrix A
// the control matrix B,
// the bias vector d
// an initial step x_0
// The number of step nStep.
auto ps = std::make_shared<copra::PreviewSystem>(A, B, d, x_0, nStep);
```

The task used in Chapter 4 can be decomposed into two subtask, a target cost and control cost. The former minimizes $\|M\mathbf{x}_N + \mathbf{p}\|$

```
#include <copra/costFunctions.h>

// Create a target cost with
auto targetCost = std::make_shared<copra::TargetCost>(M, p);
// add weight
controlCost->weight(w_x);
```

and the later minimizes $\|M\mathbf{U} + \mathbf{p}\|$.

```
// Create a control cost
auto controlCost = std::make_shared<copra::ControlCost>(N, p);
// add weight
controlCost->weight(w_u);
```

Then the $\mathcal{H}$-representation of the force is a constraint of the form $G\mathbf{U} \leq \mathbf{f}$

```
#include <copra/constraints.h>

// Create an inequality constraint with
auto controlConstr = std::make_shared<copra::ControlConstraint>(E, f);
```

Then, we just need to build the system and send it to the solver.

```
#include <copra/LMPC.h>

// Create the lmpc with the system
copra::LMPC controller(ps);
// Add the cost and constraint
controller.addConstraint(controlConstr);
controller.addCost(trajCost);
controller.addCost(controlCost);

// Build and solve the problem
controller.solve();
```

The results are directly accessible through the mpc

```
Eigen::VectorXd trajectory(controller.trajectory());
Eigen::VectorXd control(controller.control());
```

Bibliography

[Albu-Schäffer et al. 2007] Albu-Schäffer, Alin ; Ott, Christian ; Hirzinger, Gerd: A Unified Passivity-based Control Framework for Position, Torque and Impedance Control of Flexible Joint Robots. In: *The Int. J. of Robotics Research* 26 (2007), Nr. 1, S. 23–39. – URL http://ijr.sagepub.com/content/26/1/23.abstract

[André et al. 2016] André, João ; Faria, Brígida M. ; Santos, Cristina ; Reis, Luís P.: A Data Mining Approach to Predict Falls in Humanoid Robot Locomotion. In: Reis, Luís P. (Hrsg.) ; Moreira, António P. (Hrsg.) ; Lima, Pedro U. (Hrsg.) ; Montano, Luis (Hrsg.) ; Muñoz-Martinez, Victor (Hrsg.): *Robot 2015: Second Iberian Robotics Conference*. Cham : Springer International Publishing, 2016, S. 273–285. – ISBN 978-3-319-27149-1

[Atkeson et al. 2015] Atkeson, C. G. ; Babu, B. P. W. ; Banerjee, N. ; Berenson, D. ; Bove, C. P. ; Cui, X. ; DeDonato, M. ; Du, R. ; Feng, S. ; Franklin, P. ; Gennert, M. ; Graff, J. P. ; He, P. ; Jaeger, A. ; Kim, J. ; Knoedler, K. ; Li, L. ; Liu, C. ; Long, X. ; Padir, T. ; Polido, F. ; Tighe, G. G. ; Xinjilefu, X.: No falls, no resets: Reliable humanoid behavior in the DARPA robotics challenge. In: *IEEE-RAS Int. Conf. on Humanoids*, Nov 2015, S. 623–630

[Atkeson et al. 2016] Atkeson, C. G. ; Babu, B. P. W. ; Banerjee, N. ; Berenson, D. ; Bove, C. P. ; Cui, X. ; DeDonato, M. ; Du, R. ; Feng, S. ; Franklin, P. ; Gennert, M. ; Graff, J. P. ; He, P. ; Jaeger, A. ; Kim, J. ; Knoedler, K. ; Li, L. ; Liu, C. ; Long, X. ; Padir, T. ; Polido, F. ; Tighe, G. G. ; Xinjilefu, X.: *What Happened at the DARPA Robotics Challenge, and Why?* http://www.cs.cmu.edu/~cga/drc/jfr-what.pdf. 2016. – submitted to the DRC Finals Special Issue of the J. of Field Robotics

[Audren et al. 2014] Audren, H. ; Vaillant, J. ; Kheddar, A. ; Escande, A. ; Kaneko, K. ; Yoshida, E.: Model preview control in multi-contact motion-application to a humanoid robot. In: *2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2014, S. 4030–4035. – ISSN 2153-0858

[Battaglia et al. 2009] Battaglia, Martin ; Blanchet, Laurent ; Kheddar, Abderrahmane ; Kajita, Suuji ; Yokoi, Kazuhito: Combining haptic sensing with safe interaction. In: *IEEE/RSJ Int. Conf. on Intelligent Robotics and Systems*. Saint Louis, MO, USA : IEEE, 11-15 October 2009, S. 231–236

[Bouyarmane et Kheddar 2012] Bouyarmane, Karim ; Kheddar, Abderrahmane: On the dynamics modeling of free-floating-base articulated mechanisms and applications to humanoid whole-body dynamics and control. In: *Humanoid Robots (Humanoids), 2012 12th IEEE-RAS International Conference on* IEEE (Veranst.), 2012, S. 36–42

[Caron et al. 2017] Caron, Stéphane ; Pham, Quang-Cuong ; Nakamura, Yoshihiko: ZMP Support Areas for Multi-contact Mobility Under Frictional Constraints. In: *IEEE Transactions on Robotics* 33 (2017), Feb, Nr. 1, S. 67–80

[Chang et al. 2013] Chang, Wen-Ruey ; Leclercq, Sylvie ; Haslam, Roger ; Lockhart, Thurmon: The state of science on occupational slips, trips and falls on the same level. (2013)

[Featherstone 2014] Featherstone, Roy: *Rigid body dynamics algorithms*. Springer, 2014

[Fujiwara et al. 2006] Fujiwara, Kiyoshi ; Kajita, Shuuji ; Harada, Kensuke ; Kaneko, Kenji ; Morisawa, Mitsuharu ; Kanehiro, Fumio ; Nakaoka, Shinichiro ; Hirukawa, Hirohisa: Towards an optimal falling motion for a humanoid robot. In: *IEEE-RAS Int. Conf. on Humanoid Robots*, 2006, S. 524–529

[Fujiwara et al. 2007] Fujiwara, Kiyoshi ; Kajita, Shuuji ; Harada, Kensuke ; Kaneko, Kenji ; Morisawa, Mitsuharu ; Kanehiro, Fumio ; Nakaoka, Shinichiro ; Hirukawa, Hirohisa: An optimal planning of falling motions of a humanoid robot. In: *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2007, S. 456–462

[Fujiwara et al. 2004] Fujiwara, Kiyoshi ; Kanehiro, Fumio ; Kajita, Shuuji ; Hirukawa, Hirohisa: Safe knee landing of a human-size humanoid robot while falling forward. In: *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2004, S. 503–508

[Fujiwara et al. 2002] Fujiwara, Kiyoshi ; Kanehiro, Fumio ; Kajita, Shuuji ; Kaneko, Kenji ; Yokoi, Kazuhito ; Hirukawa, Hirohisa: UKEMI: falling motion control to minimize damage to biped humanoid robot. In: *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2002, S. 2521–2526

[Fukuda et Prodon 1996] Fukuda, Komei ; Prodon, Alain: *Double description method revisited*. S. 91–111. In: Deza, Michel (Hrsg.) ; Euler, Reinhardt (Hrsg.) ;

Manoussakis, Ioannis (Hrsg.): *Combinatorics and Computer Science: 8th Franco-Japanese and 4th Franco-Chinese Conference Brest, France, July 3–5, 1995 Selected Papers*. Berlin, Heidelberg : Springer Berlin Heidelberg, 1996. – ISBN 978-3-540-70627-4

[Goswami et al. 2014] Goswami, Ambarish ; Yun, Seung-kook ; Nagarajan, Umashankar ; Lee, Sung-Hee ; Yin, KangKang ; Kalyanakrishnan, Shivaram: Direction-changing fall control of humanoid robots: theory and experiments. In: *Autonomous Robots* 36 (2014), Nr. 3, S. 199–223

[Ha et Liu 2015] Ha, Sehoon ; Liu, C K.: Multiple contact planning for minimizing damage of humanoid falls. In: *IEEE/RSJ Int. Conf. on Intelligent Robotics and Systems* IEEE (Veranst.), 2015, S. 2761–2767

[Hoffman et al. 2013] Hoffman, E. M. ; Perrin, N. ; Tsagarakis, N. G. ; Caldwell, D. G.: Upper limb compliant strategy exploiting external physical constraints for humanoid fall avoidance. In: *2013 13th IEEE-RAS International Conference on Humanoid Robots (Humanoids)*, Oct 2013, S. 397–402. – ISSN 2164-0572

[Hofmann et al. 2016] Hofmann, Matthias ; Schwarz, Ingmar ; Urbann, Oliver ; Ziegler, Florian: A Fall Prediction System for Humanoid Robots Using a Multi-Layer Perceptron, 2016

[Kajita et al. 2016] Kajita, S. ; Cisneros, R. ; Benallegue, M. ; Sakaguchi, T. ; Nakaoka, S. ; Morisawa, M. ; Kaneko, K. ; Kanehiro, F.: Impact acceleration of falling humanoid robot with an airbag. In: *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, Nov 2016, S. 637–643

[Kajita et al. 2003a] Kajita, S. ; Kanehiro, F. ; Kaneko, K. ; Fujiwara, K. ; Harada, K. ; Yokoi, K. ; Hirukawa, H.: Biped walking pattern generation by using preview control of zero-moment point. In: *2003 IEEE International Conference on Robotics and Automation (Cat. No.03CH37422)* Bd. 2, Sept 2003, S. 1620–1626 vol.2. – ISSN 1050-4729

[Kajita et al. 2003b] Kajita, Suuji ; Fujiwara, Kiyoshi ; Kanehiro, Fumio ; Yokoi, Kazuhito ; Kaneko, Kenji ; Saito, Hajime ; Harada, Kensuke ; Hirukawa, Hirohisa: The first human-size humanoid that can fall over safely and stand-up again. In: *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*, 2003, S. 1920–1926

[Kakiuchi et al. 2017] Kakiuchi, Y. ; Kamon, M. ; Shimomura, N. ; Yukizaki, S. ; Takasugi, N. ; Nozawa, S. ; Okada, K. ; Inaba, M.: Development of Life-Size Humanoid Robot Platform with Robustness for Falling Down, Long Time Working and Error Occurrence. In: *2017 IEEE/RSJ International Conference on Intelligent Robots and Systems*, Sept 2017

[Kalyanakrishnan et Goswami 2011] Kalyanakrishnan, Shivaram ; Goswami, Ambarish: Learning to predict humanoid fall. In: *International Journal of Humanoid Robotics* 08 (2011), Nr. 02, S. 245–273. – URL http://www.worldscientific.com/doi/abs/10.1142/S0219843611002496

[Lee et al. 2016a] Lee, J. ; Choi, W. ; Kanoulas, D. ; Subburaman, R. ; Caldwell, D. G. ; Tsagarakis, N. G.: An active compliant impact protection system for humanoids: Application to WALK-MAN hands. In: *2016 IEEE-RAS 16th International Conference on Humanoid Robots (Humanoids)*, Nov 2016, S. 778–785

[Lee et al. 2016b] Lee, Jinoh ; Choi, Wooseok ; Kanoulas, Dimitrios ; Subburaman, Rajesh ; Caldwell, Darwin G. ; Tsagarakis, Nikos: An Active Compliant Impact Protection System for Humanoids: Application to WALK-MAN Hands. In: *IEEE-RAS Int. Conf. on Humanoid Robots*. Cancun, Mexico, 15-17 November 2016

[Lee et Goswami 2007] Lee, Sung-Hee ; Goswami, Ambarish: Reaction mass pendulum (RMP): An explicit model for centroidal angular momentum of humanoid robots. In: *IEEE Int. Conf. on Robotics and Automation*, 2007, S. 4667–4672

[Lee et Goswami 2013] Lee, Sung-Hee ; Goswami, Ambarish: Fall on Backpack: Damage Minimization of Humanoid Robots by Falling on Targeted Body Segments. In: *ASME J. of Computational and Nonlinear Dynamics* 8 (2013), Nr. 2, S. 1–10

[Lortie et Rizzo 1999] Lortie, Monique ; Rizzo, P: Reporting and classification of loss of balance accidents. In: *Safety science* 33 (1999), Nr. 1, S. 69–85

[Maplesoft 2017] Maplesoft: *Maple 2017, a division of Waterloo Maple Inc., Waterloo, Ontario*. 2017

[MATLAB Software 2016] *MATLAB Optimization Toolbox*. 2016. – The MathWorks, Natick, MA, USA

[Nori et al. 2015] Nori, Francesco ; Traversaro, Silvio ; Eljaik, Jorhabib ; Romano, Francesco ; Del Prete, Andrea ; Pucci, Daniele: iCub Whole-Body Control through Force Regulation on Rigid Non-Coplanar Contacts. In: *Frontiers in Robotics and AI* 2 (2015), S. 6. – URL http://journal.frontiersin.org/article/10.3389/frobt.2015.00006. – ISSN 2296-9144

[Ogata et al. 2007] Ogata, Kunihiro ; Terada, Koji ; Kuniyoshi, Yasuo: Falling motion control for humanoid robots while walking. In: *IEEE-RAS Int. Conf. on Humanoid Robots*, 2007, S. 306–311

[Ogata et al. 2008] Ogata, Kunihiro ; Terada, Koji ; Kuniyoshi, Yasuo: Real-time selection and generation of fall damage reduction actions for humanoid robots. In: *IEEE-RAS Int. Conf. on Humanoids*, 2008, S. 233–238

[Salas-Moreno et al. 2014] Salas-Moreno, R. F. ; Glocken, B. ; Kelly, P. H. J. ; Davison, A. J.: Dense planar SLAM. In: *2014 IEEE Int. Symposium on Mixed and Augmented Reality (ISMAR)*, Sept 2014, S. 157–164

[Samy et al. 2017] Samy, V. ; Bouyarmane, K. ; Kheddar, A.: QP-based adaptive-gains compliance control in humanoid falls. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, S. 4762–4767

[Samy et Kheddar 2015] Samy, V. ; Kheddar, A.: Falls control using posture reshaping and active compliance. In: *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, Nov 2015, S. 908–913

[Sugihara 2008] Sugihara, Tomomichi: Simulated regulator to synthesize ZMP manipulation and foot location for autonomous control of biped robots. In: *IEEE Int. Conf. on Robotics and Automation*, 2008, S. 1264–1269

[V. Samy et Kheddar 2017] V. Samy, K. B. ; Kheddar, A.: Post-Impact Adaptive Compliance for Humanoid Falls Using Predictive Control of a Reduced Model. In: *IEEE-RAS 17th International Conference on Humanoid Robots (Humanoids)*, Nov 2017. – Accepted

[Vaillant et al. 2016a] Vaillant, J. ; Bouyarmane, K. ; Kheddar, A.: Multi-Character Physical and Behavioral Interactions Controller. In: *IEEE Transactions on Visualization and Computer Graphics* PP (2016), Nr. 99, S. 1–1. – ISSN 1077-2626

[Vaillant 2015] Vaillant, Joris: *Programmation de mouvement de locomotions et manipulations en Multi-Contact pour Robots Humanoïdes et Expérimentations*, Université de Montpellier, Dissertation, 2015

[Vaillant et al. 2016b] Vaillant, Joris ; Kheddar, Abderrahmane ; Audren, Hervé ; Keith, François ; Brossette, Stanislas ; Escande, Adrien ; Bouyarmane, Karim ; Kaneko, Kenji ; Morisawa, Mitsuharu ; Gergondet, Pierre ; Yoshida, Eiichi ; Kajita, Suuji ; Kanehiro, Fumio: Multi-contact vertical ladder climbing with an HRP-2 humanoid. In: *Autonomous Robots* 40 (2016), Nr. 3, S. 561–580

[Wang et al. 2012] Wang, Jiuguang ; Whitman, Eric C. ; Stilman, Mike: Whole-body trajectory optimization for humanoid falling. In: *American Control Conf.*, 2012, S. 4837–4842

[Wieber 2006] Wieber, P.-B.: *Holonomy and Nonholonomy in the Dynamics of Articulated Motion*. S. 411–425. In: Diehl, Moritz (Hrsg.) ; Mombaur, Katja (Hrsg.): *Fast Motions in Biomechanics and Robotics: Optimization and Feedback Control*. Berlin, Heidelberg : Springer Berlin Heidelberg, 2006. – ISBN 978-3-540-36119-0

[Yoshikawa 1985] Yoshikawa, Tsuneo: Manipulability of robotic mechanisms. In: *The Int. J. of Robotics Research* 4 (1985), Nr. 2, S. 3–9

[Yun et Goswami 2014] Yun, Seung-kook ; Goswami, Ambarish: Tripod fall: Concept and experiments of a novel approach to humanoid robot fall damage reduction. In: *IEEE Int. Conf. on Robotics and Automation*, 2014, S. 2799–2805

Personal papers

Samy et Kheddar 2015 Samy, V. ; Kheddar, A.: Falls control using posture reshaping and active compliance. In: *2015 IEEE-RAS 15th International Conference on Humanoid Robots (Humanoids)*, Nov 2015, S. 908–913

Samy et al. 2017 Samy, V. ; Bouyarmane, K. ; Kheddar, A.: QP-based adaptive-gains compliance control in humanoid falls. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*, May 2017, S. 4762–4767

V. Samy et Kheddar 2017 V. Samy, K. B. ; Kheddar, A.: Post-Impact Adaptive Compliance for Humanoid Falls Using Predictive Control of a Reduced Model. In: *IEEE-RAS 17th International Conference on Humanoid Robots (Humanoids)*, Nov 2017. – Accepted