

# MÃ©thode pour la Planification de Trajectoires Garanties

SÃ©bastien LENGAGNE<sup>1,2,3</sup>, Nacim RAMDANI<sup>2,3</sup>, Philippe FRAISSE<sup>1,3,4</sup>

<sup>1</sup> Ãquipe-projet DEMAR

<http://www-sop.inria.fr/demar/>

<sup>2</sup> Laboratoire d'Informatique de Robotique et de MicroÃ©lectronique de Montpellier : LIRMM-UMR 5506

161 rue Ada 34392 Montpellier Cedex 5 - France

<http://www.lirmm.fr>

<sup>3</sup> Institut National de Recherche en Informatique et en Automatique INRIA

<http://www.inria.fr>

<sup>4</sup> UniversitÃ© Montpellier 2 : UM-2

<http://www.univ-montp2.fr>

## RÃ©sumÃ© :

Dans cet article, nous prÃ©sentons une mÃ©thode gÃ©nÃ©rale de planification de trajectoires sÃ©curisÃ©es, appliquÃ©e Ã  l'optimisation d'un pas dans le plan sagittal pour le robot humanoÃ¯de HOAP-3. La planification de trajectoires sÃ©curisÃ©es utilise les mÃªmes algorithmes d'optimisation que les mÃ©thodes de planifications classiques, mais remplace la discrÃ©tisation par grille par une discrÃ©tisation garantie basÃ©e sur l'analyse par intervalles. Les rÃ©sultats expÃ©rimentaux obtenus avec un modÃ¨le 2-D Ã  six degrÃ©s de libertÃ© du robot humanoÃ¯de HOAP-3 prouve que la mÃ©thode classique peut engendrer la violation de contraintes alors que la mÃ©thode proposÃ©e garantit le respect des contraintes sur tout le mouvement assurant ainsi l'Ã©quilibre et l'intÃ©gritÃ© du robot.

**Mots-clÃ©s :** planification de trajectoires, contraintes garanties, optimisation, discrÃ©tisation, analyse par intervalle, robot humanoÃ¯de, sÃ©curitÃ©, boucle ouverte.

## 1 Introduction

Les travaux, prÃ©sentÃ©s dans cette communication, s'inscrivent dans le cadre des activitÃ©s du projet INRIA-DEMAR (DÃ©ambulation et Marche ARTificielle) (4) qui a pour but la restauration du mouvement, et notamment la dÃ©ambulation, de patients paraplÃ©giques. Dans le contexte de notre Ã©tude, contrairement Ã  la rÃ©alisation de mouvements pour des robots humanoÃ¯des, nous ne disposons pas des capteurs nÃ©cessaires au dÃ©veloppement d'une boucle de commande. Par consÃ©quent, l'exÃ©cution de mouvements sera rÃ©alisÃ©e en boucle ouverte. Il faut Ã©galement souligner que nous devons absolument garantir l'intÃ©gritÃ© des patients paraplÃ©giques. Il est donc nÃ©cessaire de dÃ©velopper une mÃ©thode gÃ©nÃ©rale de planification de trajectoires sÃ©curisÃ©es garantissant le respect des contraintes pour toute la durÃ©e du mouvement. Afin de valider notre mÃ©thode, nous l'expÃ©rimerons sur le robot humanoÃ¯de HOAP-3.

La planification de trajectoires a Ã©tÃ© le but de nombreuses recherches de la communautÃ© robotique humanoÃ¯de. Pour des systÃ©mes complexes tels que les robots humanoÃ¯des, la planification de trajectoires doit prendre en compte un nombre important de contraintes, ce qui augmente les temps de calcul. C'est pour cette raison que la planification de trajectoires se fait, dans le domaine de la robotique, gÃ©nÃ©ralement hors ligne dans le but de crÃ©er des bases de donnÃ©es de mouvements (1) qui seront exÃ©cutÃ©es en ligne afin de palier au problÃ¨me du temps de calcul. La planification de trajectoires est un domaine vaste qui rassemble le problÃ¨me de la locomotion d'acteurs virtuels dans des environnements encombrÃ©s (18), la gÃ©nÃ©ration de mouvement de coup de pied (10) ou le calcul de trajectoires pour des robots manipulateurs (13). Tous ces problÃ¨mes peuvent se ramener Ã  un problÃ¨me d'optimisation sous contraintes.

Dans la plupart des cas, les contraintes inhÃ©rentes au systÃ¨me doivent Ãªtre satisfaites tout au long de la trajectoire gÃ©nÃ©rÃ©e. En pratique, cependant, ces contraintes sont discrÃ©tisÃ©es en ne considÃ©rant leur valeur que pour des instants appartenant Ã  une grille temporelle. Par consÃ©quent l'algorithme d'optimisation produira un mouvement qui satisfera les contraintes aux instants considÃ©rÃ©s, sans garantir la non-violation entre

deux instants. Cette absence de garantie peut provoquer des violations de contraintes qui peuvent avoir des conséquences désastreuses pour l'intégrité du système.

Pour apporter une solution à ce problème, nous développons, dans cette communication, une méthode qui se base sur la discrétisation garantie des contraintes, déjà testé pour le calcul d'un pas pour un système à deux degrés de liberté (8). Nous étendons ces travaux à la planification de trajectoires sécurisées pour des systèmes plus complexes. Nous montrons comment générer un pas optimal dans le plan sagittal pour le robot humanoïde HOAP-3. Nous nous intéressons à la discrétisation temporelle, mais notre méthode peut être appliquée à une discrétisation à N-dimensions. Le principe de base est de considérer les contraintes sur un ensemble d'intervalles de temps au lieu de ne prendre en compte qu'un ensemble d'instantanés discrets. Pour cela, nous utilisons l'analyse par intervalle (11) qui a déjà été utilisée pour résoudre des problèmes de chemin garanti évitant les auto-collisions (2). Nous illustrons notre méthode de planification de trajectoires sécurisées avec un modèle 2-D de robot humanoïde, mais elle peut être généralisée à des modèles 3-D.

Ce document est structuré de la manière suivante : la section 2 introduit le modèle 2-D du robot humanoïde utilisé pour la génération de pas optimaux dans le plan sagittal. La section 3 présente le problème d'optimisation ainsi que la discrétisation par grille couramment utilisée. La section 4 introduit l'analyse par intervalle utilisée dans le cadre de la planification de trajectoires sécurisées présentées en section 5. La comparaison des résultats dans la section 6 précède la section 7 où les travaux futurs sont exposés.

## 2 Modélisation

Pour planifier un mouvement, il est nécessaire de connaître le système grâce à un modèle mathématique. La planification de trajectoires optimales doit tenir compte de plusieurs contraintes inhérentes au système qui sont obtenues grâce à un modèle. Dans notre cas, nous nous intéressons à la planification de mouvements pour le robot humanoïde HOAP-3 dans le sagittal et considérons que l'équilibre dans le plan frontal peut être assuré en utilisant, par exemple, la méthode présentée dans (3). Nous simplifions le robot humanoïde HOAP-3 (système 3D) en un système plan avec 6 degrés de liberté, en supposant que les membres supérieurs sont équivalents à une masse située au niveau du bassin ( cf. Fig 1).

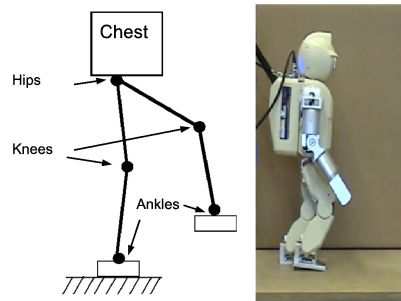


FIG. 1 – modèle plan du robot HOAP-3 pour la planification d'un pas dans le plan sagittal

Nous définissons le mouvement à travers un vecteur de paramètres  $\mathbf{X}$  qui contient des coefficients pondérateurs  $P_{j,k}$  ainsi que la durée du mouvement  $T_f$ .

$$\mathbf{X} = [P_{0,0}; P_{0,1}; \dots; T_f]^T \quad (1)$$

À partir de ce vecteur de paramètres nous allons calculer les positions, vitesses et accélérations articulaires. Puis, partant de ces valeurs, nous déterminerons les couples articulaires grâce au modèle dynamique inverse qui nous fournira également des informations pour calculer une grandeur représentative de l'équilibre du robot.

### 2.1 Variables articulaires

Chaque variable articulaire est calculée à partir de  $N_s$  coefficients pondérateurs appartenant au vecteur  $\mathbf{X}$ .

$$q_j(t) = \sum_{k=1}^{N_s} P_{j,k} \times b_k(t) \quad (2)$$

Les fonctions  $b_k(t)$  forment une base de fonctions représentée sur la Figure 2 Les vitesses et accélérations

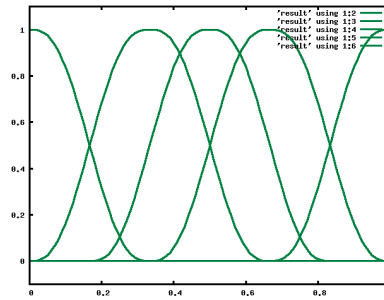


FIG. 2 – Représentation des fonctions de base  $b_k(t)$  pour  $N_s = 5$

articulaires sont obtenues par dérivation :

$$\dot{q}_j(t) = \sum_{k=1}^{N_s} P_{j,k} \times \dot{b}_k(t) \quad \ddot{q}_j(t) = \sum_{k=1}^{N_s} P_{j,k} \times \ddot{b}_k(t) \quad (3)$$

## 2.2 Modèle dynamique

Les équations du modèle dynamique permettent de calculer les couples articulaires  $\Gamma(t)$  à partir des positions, vitesses et accélérations articulaires et des efforts extérieurs  $f_{e(t)}$  (considérés nuls dans notre cas). Nous utilisons la méthode Newton-Euler (7) qui permet un calcul rapide des couples articulaires ainsi que des efforts au niveau du pied de support.

$$\begin{Bmatrix} \Gamma(t) \\ F_{pied} \end{Bmatrix} = NE(q(t), \dot{q}(t), \ddot{q}(t), f_{e(t)}) \quad (4)$$

## 2.3 Équilibre

Le calcul du Zero Moment Point (ZMP) nous informe sur l'équilibre des robots humanoïdes pendant le mouvement. (17) définit le ZMP comme le point sur la surface de contact où le moment est nul (cf. Fig 3). Si ce point reste dans la surface de support le robot conserve son équilibre, sinon il est dans un état de déséquilibre qui peut entraîner la chute.

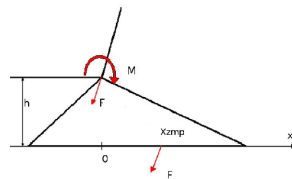


FIG. 3 – Représentation du ZMP

La valeur du ZMP dépend des positions, vitesses et accélérations articulaires, ainsi que des efforts extérieurs :

$$ZMP(t) = f(q(t), \dot{q}(t), \ddot{q}(t), f_{e(t)}) \quad (5)$$

## 2.4 Contraintes du mouvement

Dans les sections suivantes, nous considérerons qu'un mouvement optimal doit obligatoirement respecter les contraintes de butées articulaires ainsi que de vitesses articulaires limites.

$$\forall i, \forall t \quad q_{i,min} \leq q_i(t) \leq q_{i,max} \quad (6)$$

$$\forall i, \forall t \quad \dot{q}_{i,min} \leq \dot{q}_i(t) \leq \dot{q}_{i,max} \quad (7)$$

Afin d'obtenir un mouvement qui préserve l'équilibre, nous prendrons également en compte les limites du ZMP.

$$\forall t \quad ZMP_{min} \leq ZMP(t) \leq ZMP_{max} \quad (8)$$

### 3 Planification de trajectoires sous contraintes

Nous avons donc définis le modèle mathématique de notre système. Nous pouvons donc procéder à la planification des trajectoires articulaires qui définissent le mouvement (coup de pied, accomplissement d'un pas).

#### 3.1 Problème d'optimisation

Le problème de planification de trajectoires se résume souvent à un problème d'optimisation qui consiste à trouver le vecteur de paramètres optimal  $\mathbf{X}$  qui satisfait :

$$\min \int_0^T F(\mathbf{X}, t) dt \quad (9)$$

$$\text{subject to : } \begin{array}{ll} \forall i, \forall t \in [0, T] & g_i(\mathbf{X}, t) \leq 0 \\ \forall j & h_j(\mathbf{X}) = 0 \end{array} \quad (10)$$

où :

- $F$  est la fonction objectif à minimiser,
- $g_i$  est un ensemble de contraintes inégalités,
- $h_j$  est un ensemble de contraintes égalités.

Le problème posé (Eq. 9-10) est un problème de programmation semi-infinie (6) car il dispose d'un nombre fini de paramètres alors que le domaine des contraintes est infini. (14) décrit trois méthodes différentes de discrétisation des problèmes de programmation semi-infinie qui permettent d'approximer le problème semi-infini en un problème fini. L'idée principale de ces méthodes est de déterminer une grille temporelle afin de résoudre le problème d'optimisation aux intersections de la grille. Une fois le problème résolu, la grille est modifiée pour obtenir un résultat plus précis.

Cependant une grille de discrétisation n'assure pas le respect des contraintes pour des points non contenus dans cette grille. Le paragraphe suivant montre une discrétisation par grille et met en lumière la violation possible de contraintes.

#### 3.2 Discrétisation par grille

Dans la plupart des cas, discrétiser consiste à calculer les contraintes pour un ensemble discret de valeur du temps (16). Par conséquent, l'équation (10) devient :

$$\begin{array}{ll} \forall i, \forall t_k \in \mathbf{T} & g_i(\mathbf{X}, t_k) \leq 0 \\ \forall j & h_j(\mathbf{X}) = 0 \end{array} \quad (11)$$

avec  $\mathbf{T} = \{t_0 = 0, t_1, \dots, t_{N-1}, t_N = T\}$ .

Le problème semi-infini (9-10), défini comme  $\forall t \in [0, T]$ , est considéré comme un problème fini :  $\forall t_k \in \{0, t_1, \dots, t_{N-1}, T\}$  pour lequel les contraintes ne seront satisfaites uniquement aux instants pris en compte.

#### 3.3 Violation des contraintes

En utilisant une discrétisation par grille l'algorithme d'optimisation produira une solution qui satisfera les contraintes seulement pour les instants discrets  $\{t_0, t_1, \dots, t_{N-1}, t_N\}$ , mais aucune garantie n'est donnée concernant la validité des contraintes pour des instants non considérés. Afin de mettre en évidence ce phénomène nous avons utilisé cette méthode de discrétisation pour planifier le mouvement d'un pas pour le robot humanoïde HOAP-3. Nous considérons les contraintes des équations (6,7,8) et voulons minimiser la durée du mouvement.

La Figure 4 montre l'évolution du ZMP pour un pas optimisé en considérant 10 instants de discrétisation. Dans ce cas pour que le robot conserve son équilibre il faut que la valeur du ZMP reste dans l'intervalle

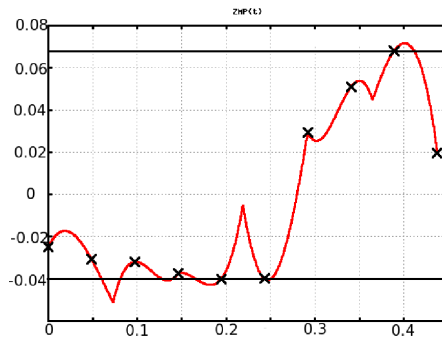


FIG. 4 – Représentation du ZMP, pour un pas optimisé en considérant 10 instants de discrétisation

$[-0,04;0,068]$ , qui correspond à la taille du pied. Nous pouvons constater que les contraintes sont bien respectées pour les instants discrets considérés par l’algorithme d’optimisation. Cependant, sur toute la durée du mouvement, la contrainte du ZMP est violée, par exemple entre  $t = 0.05s$  et  $t = 0.1s$ . Pour illustrer l’impact que cette violation de contrainte peut avoir, nous avons appliqué ce mouvement sur le robot humanoïde HOAP-3. Il s’avère donc que cette violation de contrainte entraîne la chute du robot (cf. Fig.5).

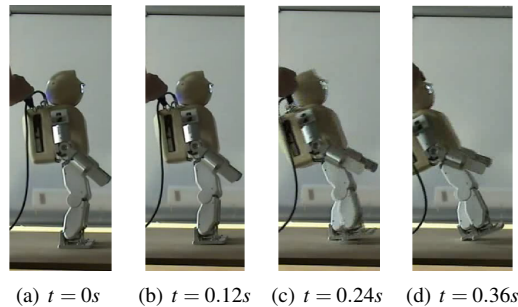


FIG. 5 – Mouvement optimisé en considérant 10 instants de discrétisation

Une solution pour résoudre ce problème serait d’augmenter le nombre  $k$  d’instants. Cependant cela ne permet pas d’éviter les possibles violations. De ce fait, nous présentons la contribution principale de cet article : la discrétisation garantie de contraintes.

## 4 Discrétisation garantie de contraintes

Le principe de la discrétisation garantie de contraintes est de calculer la valeur maximale de la fonction contrainte  $g_i(\mathbf{X}, t)$  pour un intervalle de temps  $[t] = [\underline{t}, \bar{t}]$  au lieu de la valeur calculée à un instant précis, afin de renvoyer à l’algorithme d’optimisation. L’analyse par intervalle va nous permettre d’obtenir une implémentation de ce calcul.

### 4.1 Analyse par Intervalle

Initialement l’analyse par intervalle fut développée pour quantifier les erreurs d’arrondi introduites par la représentation de nombres réels par des nombres flottants dans le domaine de l’informatique. Puis elle fût étendue à la validation numérique (15; 11; 12).

Un intervalle réel  $[a] = [\underline{a}, \bar{a}]$  est un sous ensemble de  $\mathbb{R}$ . Avec  $\underline{a} = \text{Inf}([a])$  et  $\bar{a} = \text{Sup}([a])$ . L’ensemble de tous les intervalles réels de  $\mathbb{R}$  est noté par  $\mathbb{IR}$ . Les opérations arithmétiques réelles sont étendues aux intervalles. Considérons un opérateur  $\circ \in \{+, -, *, \div\}$  et  $[a]$  et  $[b]$  deux intervalles. Alors :

$$[a] \circ [b] = [\text{inf}_{u \in [a], v \in [b]} u \circ v, \text{sup}_{u \in [a], v \in [b]} u \circ v] \quad (12)$$

Considérons  $\mathbf{m} : \mathbb{R}^n \mapsto \mathbb{R}^m$  ; le calcul de la fonction sur un vecteur intervalle  $[\mathbf{a}]$  est donné par :

$$\mathbf{m}([\mathbf{a}]) = \{\mathbf{m}(\mathbf{u}) \mid \mathbf{u} \in [\mathbf{a}]\} \tag{13}$$

La fonction intervalle  $[\mathbf{m}] : \mathbb{IR}^n \mapsto \mathbb{IR}^m$  est une fonction d'inclusion de  $\mathbf{m}$  si

$$\forall [\mathbf{a}] \in \mathbb{IR}^n, \mathbf{m}([\mathbf{a}]) \subseteq [\mathbf{m}]([\mathbf{a}]) \tag{14}$$

Une fonction d'inclusion de  $\mathbf{m}$  peut être obtenue en remplaçant chaque occurrence réelle par son correspondant intervalle et chaque fonction par son équivalent intervalle, dans ce cas elle est définie comme la fonction d'inclusion est appelé fonction naturelle d'inclusion. Les performances de cette fonction d'inclusion dépendent de l'expression formelle de  $\mathbf{m}$ .

Concrètement l'analyse par intervalle offre une implémentation pratique, tout en étant un moyen garanti, de calculer les valeurs minimale et maximale d'une fonction  $m(t)$  quand  $t$  est défini sur un intervalle  $[t]$ . Une borne supérieure pour la valeur maximale  $\max_{t \in [t]}(m(t))$  est donnée par  $Sup[m]([t])$  ainsi qu'une borne inférieure pour la valeur minimale  $\min_{t \in [t]}(m(t))$  par  $Inf[m]([t])$ .

En pratique, l'encadrement peut s'avérer trop large. Cependant il existe plusieurs techniques pour obtenir un encadrement plus étroit, telles que les expansions en série de Taylor ou d'autres techniques d'optimisation globale (5).

## 4.2 Application

Dans cette étude nous avons utilisé la méthode la plus simple pour affiner l'encadrement et obtenir une meilleure évaluation des fonctions contraintes sur un intervalle de temps. Nous avons choisi de séparer l'intervalle initial en un ensemble de  $N_b$  sous-intervalles sur lesquels le calcul sera relancé. Le nombre de sous-intervalles dépendra de la précision désirée.

Par exemple, considérons la fonction d'inclusion  $[ZMP]([t])$ . Dans l'équation (5) on remarque que la variable  $t$  apparaît plusieurs fois. De ce fait l'encadrement de  $[ZMP]([t])$  risque d'être très large. Grâce à la décomposition en sous intervalles, nous pouvons contrôler la précision du calcul des extremum.

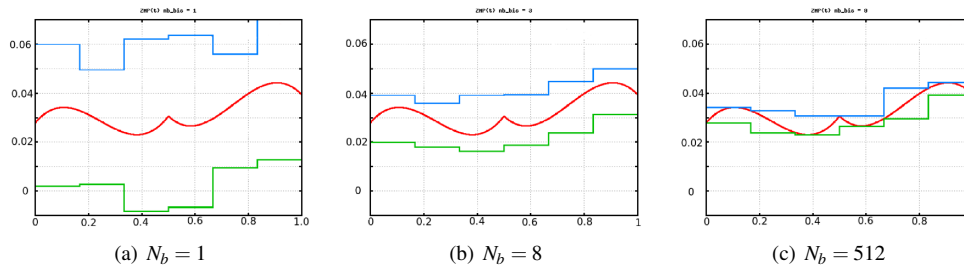


FIG. 6 – Encadrement du ZMP pour 6 intervalles de temps

La Figure 6(b), montre comment la décomposition en sous intervalles agit sur la largeur de l'encadrement.

Le tableau 1 montre la corrélation entre le temps de calcul est la précision. Le temps de calcul pour un instant discret est de 0.03ms. On s'aperçoit que la discrétisation par intervalle demande plus de temps de calcul. Pour une application hors ligne qui garantit la sécurité du mouvement nous pouvons considérer que cela reste convenable. Cependant cette méthode augmente le temps de calcul et l'utilisateur doit donc faire un compromis entre la précision et le temps de calcul.

La précision est calculée de la manière suivante :

$$\text{précision}_i(\%) = \frac{\varepsilon_i - \varepsilon_{2^{15}}}{\varepsilon_{2^{15}}} * 100 \tag{15}$$

avec :  $\varepsilon_i$  = Borne\_supérieure - Borne\_inférieure pour un calcul fait en considérant  $i$  sous intervalles.

Nous avons choisi comme référence l'encadrement obtenus pour  $2^{15}$  sous-intervalles.

$N_b$	1	2	4	8	16	32
précision(%)	72.9	48.9	19.4	8.2	3.7	1.7
temps de calcul	0.7ms	1.4ms	2.5ms	5.4ms	11ms	20ms

$N_b$	64	128	256	512	1024	$2^{15}$
précision(%)	0.8	0.34	0.17	0.08	0.04	0
temps de calcul	42ms	80ms	0.16s	0.32s	0.64s	20s

TAB. 1 – Comparaison : temps de calcul / précision

## 5 Planification de trajectoires garanties

### 5.1 Définition

La planification de trajectoires garanties se base sur les mêmes logiciels d'optimisation que les méthodes classiques de planification de trajectoires. Cependant, nous utilisons la discrétisation garantie des contraintes pour déterminer la valeur maximale des contraintes de type inégalité qui doivent être vérifiées sur tout un intervalle de temps. Par conséquent, l'équation (10) est remplacée par :

$$\forall i, \forall [t] \in \mathbf{IT} \quad \max_{\forall \tau \in [t]} g_i(\mathbf{X}, \tau) \leq 0 \quad (16)$$

avec  $\mathbf{IT} = \{[t_1], [t_2], \dots, [t_{k-1}], [t_k]\}$  où  $[t_n] = [t_{n-1}, t_n]$  représente un intervalle de temps

Dans ce cas, les bornes supérieures de l'équation (16)  $\max g_i$  peuvent être obtenues facilement en calculant la borne supérieure de la fonction d'inclusion  $[g_i]$  pour un intervalle de temps  $[t]$  :

$$\forall i, \forall [t] \in \mathbf{IT} \quad \text{Sup}[g]_i(\mathbf{X}, [t]) \leq 0 \quad (17)$$

### 5.2 Dérivée des contraintes

Nous utilisons l'algorithme d'optimisation C-FSQP (Lawrence *et al.*), qui a besoin du calcul de la dérivée des contraintes par rapport aux paramètres d'optimisation :  $\frac{\partial g(\mathbf{X}, t)}{\partial \mathbf{X}}$ . A l'heure actuelle cette dérivée est approchée par différence finie par le logiciel C-FSQP, mais des travaux en cours permettront de le calculer par différentiation automatique.

## 6 Résultats

Dans cette section, nous présentons les résultats de la planification de trajectoires appliquée au robot humanoïde HOAP-3 pour un pas de 7cm minimisant la durée du mouvement. Nous utilisons le logiciel C-FSQP (Lawrence *et al.*), sur un PC (Pentium-D, 3GHz).

Les résultats des processus d'optimisation sont présentés dans les tableaux 2, 3, en considérant les notations suivantes :

- $k$  : le nombre de points ou d'intervalles pour la discrétisation,
- $N_c$  : le nombre total de contraintes du problème d'optimisation,
- $T_c$  : le temps de calcul,
- $obj$  : la durée du mouvement obtenu (objectif)
- $It$  : le nombre d'itérations de l'algorithme.

### 6.1 Discrétisation par grille

Tout d'abord nous présentons les résultats des optimisations utilisant une discrétisation par grille (Cf. Section 3.2) :

Dans le Tableau 2, le nombre  $k$  d'instantanés discrets considérés influe sur le temps de calcul ainsi que sur la valeur finale de l'objectif. Nous pouvons supposer que le résultat optimal est obtenu pour  $k = 50$  et une durée de mouvement  $obj = 0.355s$ . Par conséquent, un mouvement plus rapide ( $obj < 0.355$ ), comme les mouvements obtenus pour  $k = 5, 10, 20$  provoquera la violation de certaines contraintes,

$k$	$N_c$	$T_c$ (s)	obj	Ite
5	137	103	0.320	467
10	267	137	0.344	336
20	527	1125	0.349	1510
50	1307	1587	0.355	742

TAB. 2 – Résultats des optimisations utilisant une discrétisation par grille.

Ces calculs soulignent la chose suivante : la méthode de discrétisation par grille peut amener à la violation de contraintes pour un mouvement prétendument optimal. Dans le cas d'un robot humanoïde, la violation de contrainte peut entraîner la chute ou la destruction du robot.

## 6.2 Discrétisation garantie

Le Tableau 3 présente les résultats d'un planification de trajectoires garanties :

$k$	$N_c$	$T_c$ (s)	obj	Ite
5	137	2959	0.61	688
6	163	1675	0.61	331
10	267	20676	0.48	2536
12	319	2417	0.48	254
18	475	3300	0.44	232

TAB. 3 – Résultats d'un planification de trajectoires garanties.

Le nombre d'intervalles a un effet positif sur la valeur de la fonction objectif. En effet une optimisation avec  $k = 6$  retourne un mouvement de 0,61s, alors que pour  $k = 18$  la durée du mouvement obtenu est de 0,44s. Les deux solutions produisent un mouvement qui ne violera aucune contrainte. On peut en déduire que l'optimisation faite avec  $k = 18$  prend en compte des solutions rejetées quand  $k = 6$ . Ces solutions sont rejetées à cause de l'encadrement trop large. Par conséquent, le choix du nombre d'intervalle  $k$  est important pour la précision de l'algorithme et doit être choisi avec soin.

Le tableau 3 montre que le fait d'avoir  $k$  un multiple de 6 permet de meilleur résultat. Pour plus d'information à ce sujet, nous invitons le lecteur à lire (9).

## 6.3 Expérimentation

Nous avons expérimenté le mouvement d'un pas de 7cm généré par la méthode de planification de trajectoires garanties en considérant  $k = 10$  sur le robot humanoïde HOAP-3. La Figure 8 montre le mouvement obtenu. On s'aperçoit que le robot conserve son équilibre contrairement au mouvement de la Figure 5. La Figure 7 montre l'évolution de la fonction ZMP. Il apparaît clairement que l'encadrement  $[ZMP]$  ainsi que la fonction continue  $ZMP(t)$  reste toujours dans les limites du pied :

$$-0,04 \leq ZMP(t) \leq 0,068 \quad (18)$$

La planification de trajectoires garanties assure la validité des contraintes sur toute la durée du mouvement.

## 7 Discussion

Cette communication présente une méthode de planification de trajectoires garanties appliquée au robot humanoïde HOAP-3. Cette méthode calcule les contraintes sur des intervalles de temps pour générer une solution optimale qui interdit toute violation de contraintes, alors que des méthodes de discrétisation classique ne donne aucune garantie, ce qui peut conduire à la chute ou à la destruction du robot.

Un inconvénient de cette méthode est le temps de calcul, trop long pour une application en ligne. Cependant, une solution serait de diminuer le nombre de sous-intervalles pour le calcul des contraintes. Mais cette

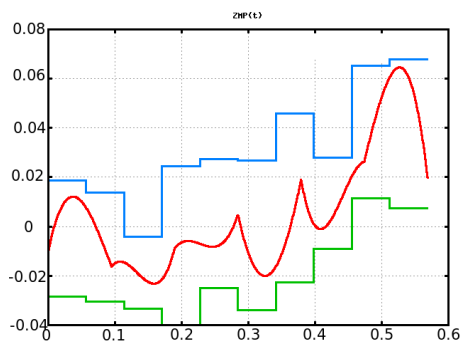


FIG. 7 – Représentation du ZMP, pour un mouvement obtenu grâce à la planification de trajectoires garanties avec  $k = 10$

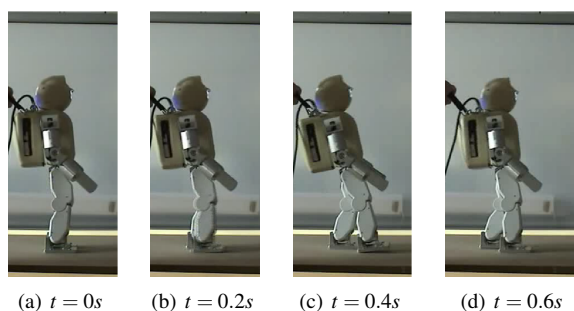


FIG. 8 – Mouvement obtenu grâce à la planification de trajectoires garanties avec  $k = 10$

solution réduirait la précision du calcul, ce qui produirait des solutions sans violation mais sous-optimales. Nous prévoyons de tester d'autres solutions (5), ce qui pourrait diminuer le temps de calcul pour une précision identique. Toutefois, il est important de noter que le résultat obtenu est garanti sans violation de contraintes.

Pour permettre une réutilisation facile de ce travail nous avons créé la Guaranteed Discretization Library (GDL) disponible sur <http://www.lirmm.fr/~lengagne/GDL>. Cette bibliothèque en C++ permet la discrétisation garantie en utilisant l'analyse par intervalle.

La méthode de discrétisation par intervalle peut être étendue à plusieurs dimensions. Dans notre cas nous n'avons considéré que la dimension temporelle. Mais une discrétisation 2-D de l'espace libre pour la planification de déplacement de robots mobiles peut facilement être envisagée. De même la discrétisation par intervalle peut s'appliquer à des espaces à N-dimensions, même si dans ce cas les temps de calculs augmenteraient considérablement.

## Conclusion

Dans ce papier nous avons présenté la planification de trajectoires garanties. Cette méthode utilise des algorithmes d'optimisation classiques, mais remplace la discrétisation par grille par une discrétisation garantie des contraintes. L'évaluation expérimentale sur le robot humanoïde HOAP-3 montre clairement la nécessité d'une garantie sur les contraintes pendant les mouvements en boucle ouverte. Les mouvements, dit optimaux, obtenus par des méthodes de discrétisation par grille peuvent provoquer une violation des contraintes avec des conséquences importantes sur le robot telles que sa chute ou sa destruction. Au contraire notre méthode de planification de trajectoires garanties génère des mouvements sûrs qui ne mettent pas en danger l'intégrité et l'équilibre du robot.

Le calcul de l'encadrement des contraintes et du gradient par rapport aux paramètres nécessitent d'avantage de recherches afin de palier au problème de temps de calcul

## Références

- [1] DENK J. & SCHMIDT G. (2001). Synthesis of a Walking Primitive Database for a Humanoid Robot using Optimal Control Techniques. In *Proceedings of IEEE-RAS International Conference on Humanoid Robots*, p. 319–326, Tokyo, Japan.
- [2] FANG H. & MERLET J. P. (2005). Dynamic interference avoidance of 2-DOF robot arms using interval analysis. In *Intelligent Robots and Systems, 2005. (IROS 2005). 2005 IEEE/RSJ International Conference on*, p. 3809–3814.
- [3] FRAISSE P., COTTON S., MURRAY A. & PIERROT F. (2007). Towards dynamic balance control of humanoid robots by using com and zmp. In *Workshop in the IEEE-RAS 7th International Conference on Humanoid Robots*.
- [4] GUIRAUD D., POIGNET P., WIEBER P., MAKKSOUH H. E., PIERROT F., DOMBRE E., DIVOUX J. & RABISCHONG P. (2003). Modelling of the human paralysed lower limb under fes. In *Robotics and Automation, 2003. Proceedings. ICRA apos ;03. IEEE International Conference on*, volume 2, p. 2218–2223.
- [5] HANSEN E. & WALSTER G. (2004). *Global optimization using interval analysis*. Marcel Dekker, 2nd edition.
- [6] HETTICH R. & KORTANEK K. O. (1993). Semi-infinite programming : theory, methods, and applications. *SIAM Rev.*, **35**(3), 380–429.
- [7] KHALIL W. & DOMBRE E. (2002). *Modeling, Identification & Control of Robots*. Hermes Sciences Europe, 3 edition.
- [Lawrence et al.] LAWRENCE C., ZHOU J. L. & TITS A. L. *User's Guide for CFSQP Version 2.5 : A C Code for Solving (Large Scale) Constrained Nonlinear (Minimax) Optimization Problems, Generating Iterates Satisfying All Inequality Constraints*. Electrical Engineering Department, Institute for Systems Research University of Maryland, College Park, MD 20742.
- [8] LENGAGNE S., RAMDANI N. & FRAISSE P. (2007). Guaranteed computation of constraints for safe path planning. In *IEEE-RAS 7th International Conference on Humanoid Robots*.
- [9] LENGAGNE S., RAMDANI N. & FRAISSE P. (2008.). Safe path planning for humanoid robots. In *submitted to Intelligent Robots and Systems, (IROS 2008)*.
- [10] MIOSSEC S., YOKOI K. & KHEDDAR A. (2006). Development of a software for motion optimization of robots - application to the kick motion of the hrp-2 robot. In *Proceedings of the 2006 IEEE International Conference on Robotics and Biomimetics*, p. 299–304.
- [11] MOORE R. E. & BIERBAUM F. (1979). *Methods and Applications of Interval Analysis (SIAM Studies in Applied and Numerical Mathematics) (Siam Studies in Applied Mathematics, 2.)*. Soc for Industrial & Applied Math.
- [12] NEUMAIER A. (1990). *Interval methods for systems of equations*. Cambridge : Cambridge university press.
- [13] PIAZZI A. & VISIOLI A. (2000). Global minimum-jerk trajectory planning of robot manipulators. In *IEEE Transactions on Industrial Electronics*, volume 47, p. 140–149.
- [14] REEMTSEN R. (1998). Semi-infinite programming : discretization methods.
- [15] SUNAGA T. (1958). Theory of interval algebra and its application to numerical analysis. *RAAG Memoirs, Ggijutsu Bunken Fukuy-kai*, **2**, 547–564.
- [16] VON STRYK O. (1993). Numerical solution of optimal control problems by direct collocation.
- [17] VUKOBRATOVIC M. & JURICIC D. (1969). Contribution to the synthesis of biped gait. *IEEE trans. Bio-Med Eng.*, **BME-16**, 1–6.
- [18] YOSHIDA E., BELOUSOV I., ESTEVES C. & LAUMOND J.-P. (2005). Humanoid motion planning for dynamic tasks. In *Humanoid Robots, 2005 5th IEEE-RAS International Conference on*, p. 1–6.