Contrôle optimal inverse appliqué au problème de la marche

Rapport de stage de licence

Émile Enguehard

1 Motivations

L'équipe Gepetto[1], créée en 2006, se spécialise dans l'étude du mouvement anthropomorphique, et notamment du problème de la génération de mouvements pour des robots humanoïdes (les expériences de l'équipe sont réalisées sur le robot japonais HRP-2).

Le mouvement le plus essentiel d'un robot humanoïde est la marche, et dans un premier temps la marche sur un sol plan. Un robot humanoïde tel qu'HRP-2 est animé par une trentaine de moteurs indépendants. Pour le contrôler, il est donc nécessaire de calculer des trajectoires dans des espaces de grande dimension, et soumises à des contraintes nombreuses ; chaque moteur et chaque jointure évoluent dans des intervalles restreints, et à cela s'ajoutent des contraintes liées à la structure du robot (par exemple que les membres ne se cognent pas entre eux), et aux forces de contact (soumises à la loi de Coulomb). Pour cette raison, il est fréquent de séparer ce le problème en plusieurs sous-tâches interagissant à des degrés divers, dont notamment le calcul de la trajectoire du centre de masse du robot.

Pour ce faire, on utilise diverses techniques d'optimisation numérique et notamment de *contrôle optimal*. Pour obtenir des résultats acceptables de manière efficace, il est nécessaire de simplifier les modèles à l'extrême, par exemple en constraignant la dimension z. Le travail de l'équipe Gepetto vise notamment à se débarrasser de ces contraintes et simplifications. Ainsi, l'objectif idéal de ce stage est de déterminer comment générer des trajectoires où le mouvement suivant l'axe vertical est autorisé, à partir de données enregistrées sur des sujets humains. Il s'agit d'un problème de *contrôle optimal inverse* ; le stage a été principalement employé à l'expérimentation d'une technique nouvelle de contrôle optimal inverse, conçue par des membres de l'équipe MAC (*Méthodes et Algorithmes de Contrôle*) du LAAS.

Tous les concepts introduits dans le paragraphe précédent sont expliqués par la suite. Au sujet du problème de la marche, on pourra se reporter notamment à [1].

2 Le problème de la marche

2.1 Dynamique du centre de masse

Le centre de masse d'un robot humanoïde peut être représenté par un vecteur à trois dimensions c. Sa dynamique s'exprime ainsi :

$$\begin{cases} m(\ddot{c} - g) = \sum F_i \\ mc \wedge (\ddot{c} - g) + \dot{L} = \sum p_i \wedge F_i \end{cases} \tag{1}$$

où :

– m est la masse du robot ;

1. Dont le site internet est accessible à http://projects.laas.fr/gepetto/

– les F_i sont les forces de contact, et les p_i les points où elles s'appliquent ;
– g est le champ gravitationnel ;
– L est le moment cinétique du robot.

2.2 Simplifications

Ce modèle peut être simplifié en supposant que le robot avance à peu près droit ($\dot{L}$ est négligeable) est que les contacts ont tous lieu sur un plan horizontal. Dans ce cas, en notant F la somme des forces, et en introduisant le *centre de pression* p (aussi appelé *point de moment nul* ou ZMP, *Zero Momentum Point*), le point où le moment des forces de contact s'annule, on obtient :

$$\begin{cases} m(\ddot{c} - g) = F \\ mc \wedge (\ddot{c} - g) = p \wedge F \end{cases} \tag{2}$$

Cette équation peut-être simplifiée à nouveau en identifiant les deux occurrences de F et en tenant compte du fait que p se trouve dans le plan du sol. On obtient :

$$\begin{cases} p_x = c_x - \dfrac{c_z}{\ddot{c_z} + g}\ddot{c_x} \\ p_y = c_y - \dfrac{c_y}{\ddot{c_z} + g}\ddot{c_y} \end{cases} \tag{3}$$

où g représente désormais l'amplitude du champ gravitationnel.

Une dernière simplification s'obtient lorsqu'on néglige le mouvement vertical du centre de masse du robot. La dynamique qui en résulte est linéaire, et les deux dimensions sont indépendantes :

$$\begin{cases} p_x = c_x - \dfrac{c_z}{g}\ddot{c_x} \\ p_y = c_y - \dfrac{c_z}{g}\ddot{c_y} \end{cases} \tag{4}$$

2.3 Problème

Le problème de la marche consiste à trouver une trajectoire $c(t)$ obéissant à une contrainte linéaire de la forme $A(t)p(t) \leq b(t)$. La contrainte exprime que le centre de pression doit rester dans un polygône dit *polygône de pression* autour de la trajectoire désirée ; on peut la voir comme une contrainte de la forme $||p(t) - p^*(t)|| \leq \eta$, où $||\cdot||$ est une certaine norme et p^* est une trajectoire de référence. Cela permet à la fois de garantir que le robot réalise le mouvement désiré et que la trajectoire soit faisable mécaniquement. La trajectoire doit par ailleurs minimiser un coût donné ; on choisit souvent le *minimal jerk* ($\int_0^T ||\dddot{c}(t)||^2 \mathrm{d}t$) qui produit des trajectoires lisses et facilement réalisables. En pratique, on choisit p^* constant par morceaux, chaque valeur correspondant à un pas.

Tel quel, le problème est difficile à résoudre numériquement. De plus, il faut soit choisir les contraintes de manière conservatrice, soit prendre le risque que le mouvement reste longtemps à la limite du réalisable, ce qui peut être source d'échec. Pour cette raison, on peut choisir de résoudre une version sans contraintes du problème. Il s'agit de minimiser un nouveau coût : $\int_0^T ||\dddot{c}(t)||^2 + K||p(t) - p^*(t)||^2 \mathrm{d}t$, où K est grand devant un (typiquement 10^5).

3 Le contrôle optimal

Le problème de la marche peut être considéré comme un problème de *contrôle optimal*. Étant donné un espace d'états $X \subset \mathbb{R}^p$, un espace de contrôle $U \subset \mathbb{R}^{p'}$, un espace final $X_T \subset X$, un temps fixé $T > 0$, une dynamique $f : X \times U \to \mathbb{R}^p$, un coût $l : [0, T] \times X \times U \to \mathbb{R}$ et un état initial $x_0 \in X$, un problème de contrôle optimal direct (OCP) consiste à trouver un

contrôle $u(t) \in U$ tel que la trajectoire $x(t)$ définie par $x(0) = x_0$ et $\dot{x}(t) = f(x(t), u(t))$ minimise $\int_0^T l(t, x(t), u(t))\mathrm{d}t$ et que $x(T) \in X_T$ [2]. L'histoire et la théorie du contrôle optimal sont décrites dans [2].

Il est utile d'introduire la fonction suivante, dite *fonction valeur* :

$$\begin{aligned} v(s,y) = \inf_u \int_s^T & l(t, x(t), u(t))\mathrm{d}t \\ & \forall t \in [0,T],\ \dot{x}(t) = f(x(t), u(t)) \\ & x(s) = y \\ & x(T) \in X_T \end{aligned} \tag{5}$$

3.1 Application au problème de la marche

Une manière possible d'exprimer le problème de la marche comme un OCP, dans le cadre de la dynamique linéaire simplifiée, est de choisir $x = (c, \dot{c}, \ddot{c})$ et $u = \dddot{c}$. On a : $X = \mathbb{R}^6$ (si c a deux dimensions), $U = \mathbb{R}^2$, et :

$$f(x_1, x_2, x_3, u) = (x_2, x_3, u) \tag{6}$$

$$l(t, x_1, x_2, x_3, u) = ||u||^2 + K||x_1 - \frac{z}{g}x_3 - p^*(t)||^2 \tag{7}$$

où z et g sont des constantes (par exemple $z = 1\mathrm{m}$ et $g = 10\mathrm{N}$).

Ce problème peut être résolu en discrétisant le temps : on obtient un problème d'optimisation quadratique à contraintes linéaires d'égalité, ce qui se ramène à l'inversion d'une matrice éparse (voir par exemple [3]).

3.2 Équations d'Hamilton-Jacobi-Bellman

On dispose de conditions suffisantes pour qu'une trajectoire soit optimale dans le cadre d'un OCP où le coût l ne dépend pas du temps [3]. En reprenant les notations précédentes, on définit un opérateur agissant sur les fonctions de coût et les fonctions valeurs :

$$\mathcal{L}(l, v) = l + \frac{\partial v}{\partial t} + {}^t(\nabla v) f \tag{8}$$

On a alors les conditions suffisantes suivantes pour qu'une trajectoire (x, u) qui respecte la dynamique et les contraintes soit optimale pour l : il existe une fonction $v \in \mathcal{C}^1([0,T] \times X)$ telle que :

$$\forall (t,x,u) \in [0,T] \times X \times U, \quad \mathcal{L}(l,v)(t,x,u) \geq 0 \tag{9}$$

$$\forall x \in X_T, \quad v(T,x) = 0 \tag{10}$$

$$\forall t \in [0,T], \quad \mathcal{L}(l,v)(t, x(t), u(t)) = 0 \tag{11}$$

La fonction v minore la fonction valeur associée au coût l.

Si on remplace la condition (11) par la condition plus faible suivante :

$$\int_0^T \mathcal{L}(l,v)(t, x(t), u(t))\mathrm{d}t \leq \varepsilon \tag{12}$$

2. Cette définition n'est pas universelle. Elle pose notamment problème car u n'existe pas forcément ; on définit correctement le problème comme celui du calcul de v.

3. Cette limitation n'est pas imposée par la théorie ; on l'introduit pour des raisons exposées plus loin.

on a l'assurance que la trajectoire est ε-optimale :

$$\int_0^T l(x(t), u(t))\mathrm{d}t \leq v^*(0, x(0)) + \varepsilon \tag{13}$$

où v^* est la fonction valeur associée à l.

Ces résultats sont prouvés dans [4].

3.3 Problème inverse

Le problème inverse (IOCP) est le suivant : étant donné un ensembles de trajectoires ou de points de trajectoires, pour quel coût ces trajectoires sont-elles optimales ? Exprimé ainsi, le problème est assez mal posé ; il n'y aucune assurance qu'il existe une solution unique ou un petit nombre de solutions, et le coût nul est systématiquement solution. Cela n'empêche pas d'obtenir des résultats en pratique, au moyen de diverses heuristiques que nous allons détailler plus loin.

La résolution numérique de problèmes inverses est difficile. Plusieurs méthodes ont été proposées ; la plus simple consiste à résoudre à chaque itération un problème direct et affiner le coût ou le contrôle par un algorithme d'optimisation numérique. Cela nécéssite de choisir une base finie de fonctions pour le coût ou le contrôle. D'autres méthodes existent, comme celle présentée par Dvijotham et Todorov dans [5] (où l'on trouve également un résumé de l'état de la recherche en contrôle optimal inverse).

La méthode employée ici, introduite par Pauwels, Henrion et Lasserre, du l'équipe MAC du LAAS, dans [4], utilise les équations HJB. On cherche l et v sous la forme de polynômes d'un degré fixé, et on minimise ε tel que les trajectoires soient ε-optimales. Le problème est posé ainsi, étant donné un ensemble de points $(t_i, x_i, u_i)_{i\in I}$ provenant de trajectoires supposée optimales :

$$\begin{aligned}
&\inf_{l,v} \frac{1}{|I|}\sum_{i\in I}\mathcal{L}(l,v)(t_i,x_i,u_i)\\
&\quad \mathcal{L}(l,v)(t,x,u) \geq 0 \quad \forall (t,x,u) \in [0,T]\times X\times U\\
&\quad l \in \mathbb{R}_d[x,u]\\
&\quad v \in \mathbb{R}_{d'}[t,x]\\
&\quad v(T,x) = 0 \quad \forall x \in X_T
\end{aligned} \tag{14}$$

où d et d' sont des constantes prédéterminées. Ce problème d'optimisation numérique est de dimension finie et se ramène à un SDP lors X, U et X_T sont des ensembles *semi-algébriques*, c'est-à-dire définis par des inégalités polynomiales, comme décrit dans [4] et [6].

Notons que l'objectif minimisé est une version discrétisée de la contrainte 11, tandis que les autres contraintes sont gardées telles quelles.

On utilisera par la suite la notation $\varepsilon = \sum_I \mathcal{L}(l,v)(t_i,x_i,u_i)$. Les trajectoires données sont en effet ε-optimales pour le coût calculé l.

3.4 Résolution du problème inverse

Des méthodes heuristiques permettent d'obtenir plus certainement des résultats intéressants avec cette technique. Tout d'abort, il faut éviter les cas triviaux où $\mathcal{L}$ est nul sur $[0,T]\times X\times U$. Pour cela, on rajoute une condition de normalisation :

$$\mathcal{A}(\mathcal{L}(l,v)) = 1 \tag{15}$$

où $\mathcal{A}$ est un opérateur linéaire. On peut par exemple prendre l'intégration sur $[0,T]\times X\times U$ si X et U sont compacts et « simples » (par exemple des pavés) ; on peut autrement intégrer sur un pavé inclus dans $[0,T]\times X\times U$.

De plus, l'expérience montre que pour obtenir des polynômes suffisamment simples pour qu'on puisse les interpréter, il peut être utile de minimiser également la norme L^1 de l, définie par : $||l||_1 = \sum |l_i|$. Comme cette norme n'est pas différentiable aux points où certains coefficients sont nuls, ces points ont tendance à correspondre à des minima locaux. On remplace donc l'objectif dans (14) par :

$$\inf_{l,v} \frac{1}{|I|} \sum_{i \in I} \mathcal{L}(l,v)(t_i, x_i, u_i) + \lambda ||l||_1$$

où λ est un paramètre à fixer. Dans le cas des essais réalisés, les polynômes désirés étaient relativement éparses, ce qui est une raison supplémentaire d'utiliser cette technique.

L'objective de minimisation de la norme L^1 peut s'exprimer avec des contraintes linéaires. Minimiser $\sum |l_i|$ est en effet équivalent à minimiser $\sum l'_i$ soumis aux contraintes $l'_i \geq l_i$ et $l'_i \geq -l_i$.

Le problème final est donc :

$$\begin{aligned}
&\inf_{l,v} \frac{1}{|I|} \sum_{i \in I} \mathcal{L}(l,v)(t_i, x_i, u_i) + \lambda ||l||_1 \\
&\quad \mathcal{L}(l,v)(t,x,u) \geq 0 \quad \forall (t,x,u) \in [0,T] \times X \times U \\
&\quad l \in \mathbb{R}_d[x,u] \\
&\quad v \in \mathbb{R}_{d'}[t,x] \\
&\quad v(T,x) = 0 \quad \forall x \in X_T \\
&\quad \mathcal{A}(\mathcal{L}(l,v)) = 1
\end{aligned} \tag{16}$$

4 Résultats expérimentaux

4.1 Détails d'implémentation

L'essentiel du travail réalisé lors de ce stage s'est fait à l'aide du système Matlab.

Pour résoudre les problèmes directs, j'ai utilisé la méthode décrite dans [3]. Le problème de la marche considéré a une dynamique linéaire et un coût quadratique. Si l'on discrétise la trajectoire, on obtient un problème d'optimisation quadratique à contraintes d'égalité linéaires, dont la dimension est huit fois le nombre de points de la trajectoires (le vecteur position c a deux dimensions, et on calcule c, $\dot{c}$, $\ddot{c}$, et $\dddot{c}$). Ce problème est équivalent à la résolution d'un système d'équations linéaires (conditions dites KKT, pour Karush-Tuhn-Tucker), formé à partir de la matrice du coût et de celles des contraintes. Comme chaque point n'est lié qu'à ses voisins immédiats par la dynamique, ce système est très épars et on peut obtenir des résultats en peu de temps (moins d'une seconde) pour des trajectoires avec beaucoup de points (j'ai considéré des trajectoires de 1 à 10 secondes avec des points espacés de 10 millisecondes).

Une autre manière, plus précise de résoudre le problème direct est de passer par une équation de Ricatti. On peut en effet démontrer (voir par exemple [7]) que la solution de cette équation différentielle matricielle (que l'on peut calculer assez précisément avec Matlab) fournit le contrôle optimal en fonction de la position. On gagne ainsi en précision par rapport à la discrétisation un peu naïve envisagée dans le paragraphe précédent. Cependant, j'ai abandonné cette méthode moins flexible après avoir constaté qu'elle n'apportait pas d'amélioration en pratique.

Pour traduire les problèmes de contrôle optimal inverse sous une forme compréhensible par un solveur, j'ai utilisé YALMIP [4], interfacé avec le solveur MOSEK [5]. Ce logiciel permet d'exprimer des problèmes d'optimisation dans un langage de haut niveau, et les transmet à des solveurs spécialisés. La contrainte de positivité de $\mathcal{L}(l,v)$ sur $[0,T] \times X \times U$, par exemple, peut s'exprimer dans YALMIP par une série de contraintes *sum-of-squares* (cela repose sur le

4. http://users.isy.liu.se/johanl/yalmip/
5. http://www.mosek.com/products/mosek

théorème de Putinar, exposé entre autres dans [6]). À degré fixé, les contraintes *sum-of-squares* sont en fait plus fortes que la simple positivité. La nullité de la fonction $v(t,x)$ en $t = T$ peut s'exprimer par une égalité polynômiale (le polynôme v est un multiple de $T - t$).

J'ai également écrit un script en Python pour visualiser les trajectoires calculées en 3D.

4.2 Travaux initiaux

J'ai commencé par travailler sur un problème simple unidimensionnel, pour tester la méthode de contrôle optimal inverse utilisée. Le problème est le suivant : on dispose d'un capital $x(t)$, et à chaque instant on peut investir une proportion $0 \leq \alpha(t) \leq 1$ pour obtenir des retours futurs, et économiser le reste ; x respecte la dynamique $\dot{x}(t) = \alpha(t)x(t)$. Sur une durée $T \geq 1$, comment choisir $\alpha(t)$ de manière à maximiser les économies finales, à savoir $G = \int_0^T x(t)(1-\alpha(t))\mathrm{d}t$?

On démontre que le contrôle optimal $\alpha^*(t)$ est donné par $\alpha^*(t) = 1$ si $t \leq T-1$ et $\alpha^*(t) = 0$ sinon.

L'expérience réalisée consiste à générer des trajectoires optimales et à essayer de retrouver $l(x,\alpha) = x(1-\alpha)$ en faisant varier les valeurs de λ, d et d' et en fournissant des jeux de données plus ou moins fournis.

Les calculs effectués donnent l, v et ε. Pour rappel, ε est la valeur de l'objectif à minimiser au terme du calcul. Les trajectoires données sont ε-optimales pour l.

Les résultats ne sont pas ceux attendus. On observe tout d'abord que, par contraste avec les expériences décrites dans [4], la valeur de λ a peu d'importance. Pour rappel, les auteurs n'avaient pas de résultat satisfaisant en deçà d'une certaine valeur de λ, après quoi la qualité du résultat s'améliorait subitement puis se dégradait. Ici, en deçà d'une grande valeur de λ, on obtient toujours le même résultat, y compris pour $\lambda = 0$. Au-delà de cette grande valeur, le solveur termine sur une erreur.

Cependant, les résultats obtenus pour λ faibles sont intéressants. On ne retrouve pas le polynôme $x(1-\alpha)$ désiré ; les polynômes obtenus (si on ne considère que les termes dominants), qui dépendent principalement des données, sont des multiples de $\alpha(1-\alpha)$; on a notamment souvent $x^2\alpha(1-\alpha)$ et $\alpha(1-\alpha^3)$. Ces polynômes apparaissent assez distinctement ; le rapport entre les deux coefficients dominaux et les autres est toujours supérieur à 100 et j'ai pu obtenir pour certains jeux de données le polynôme $\alpha(1-\alpha^3)$ presque exactement.

Notons qu'il est apparu essentiel de limiter d (le degré du coût) pour obtenir des résultats intéressants.

Les trajectoires générées sont effectivement optimales pour les polynômes obtenus, vu que $\alpha(t)$ évolue dans $\{0,1\}$. En soi, cela démontre que l'approche adoptée conduit à des résultats corrects et non-triviaux. Cela dit, cela fait également ressortir ses limites ; on obtient un coût pour lequel les trajectoires sont optimales, mais qui n'explique pas forcément le choix du contrôle. Ici, la valeur de $\alpha(t)$ dépendait de t d'une manière très visible (pour l'œil humain ou peut-être pour une autre méthode), ce qui n'a pas de conséquence sur le résultat avec la méthode utilisée. Il serait donc impossible de regénérer des trajectoires avec les coûts obtenus. Il apparaît en tout cas que la méthode, si elle est intéressante, est peu adaptée à ce type de problèmes ; les faits que α ne dépende pas de x et qu'il évolue dans un ensemble fini semblent avoir porté préjudice.

4.3 Problème de la marche : premiers essais

On s'intéresse maintenant au problème de la marche tel que décrit en 3.1. Comme les deux dimensions sont indépendantes, on peut considérer uniquement le mouvement suivant l'axe x ce qui réduit la dimension de l'espace d'états de 6 à 3, et celle du contrôle de 2 à 1.

L'une des principales limitations de la méthode employée, telle que donnée par (16), est qu'elle ne permet la dépendance directe de l par rapport à t. Nous verrons plus loin comment on peut contourner cet obstacle. En théorie, on pourrait étendre le raisonnement au cas où l

dépend de t, mais cela rend le problème encore plus mal posé, et augmente les chances d'obtenir des résultats triviaux ou inintéressants ; je m'en suis donc tenu au problème posé en (16). Une raison supplémentaire de faire ce choix est que la dépendance en temps de l dans (7) provient du terme $p^*(t)$, qui n'est pas continu et serait donc difficile à approcher par un polynôme.

Dans un premier temps, on pose $p^* = 0$. Les trajectoires considérées sont donc des retours à l'équilibre en 0 depuis des positions initiales choisies aléatoirement, et le coût s'écrit :

$$l_0(x,u) = l_0(c,\dot{c},\ddot{c},\dddot{c}) = ||\dddot{c}||^2 + K||c - \frac{z}{g}\ddot{c}||^2 = \dddot{c}^2 + Kc^2 - 2K\frac{z}{g}c\ddot{c} + K\left(\frac{z}{g}\right)^2\ddot{c}^2 \qquad (17)$$

Cette fois-ci, sans doute parce que le problème est plus « lisse, » on retrouve les résultats de [4]. Pour de petites valeurs de λ, le polynôme l obtenu ne présente pas de caractéristique notable, et ε est extrêmement faible ; on a obtenu un résultat trivial tel que $\mathcal{L}(l,v)$ est quasi-nul sur $[0,T] \times X \times U$. Pour de très grandes valeurs de λ, ε prend des valeurs grandes devant 1, ce qui signifie que les trajectoires sont loin d'être optimales pour les coûts calculés. Il existe cependant un palier intermédiaire où l a des coefficients dominants discernables.

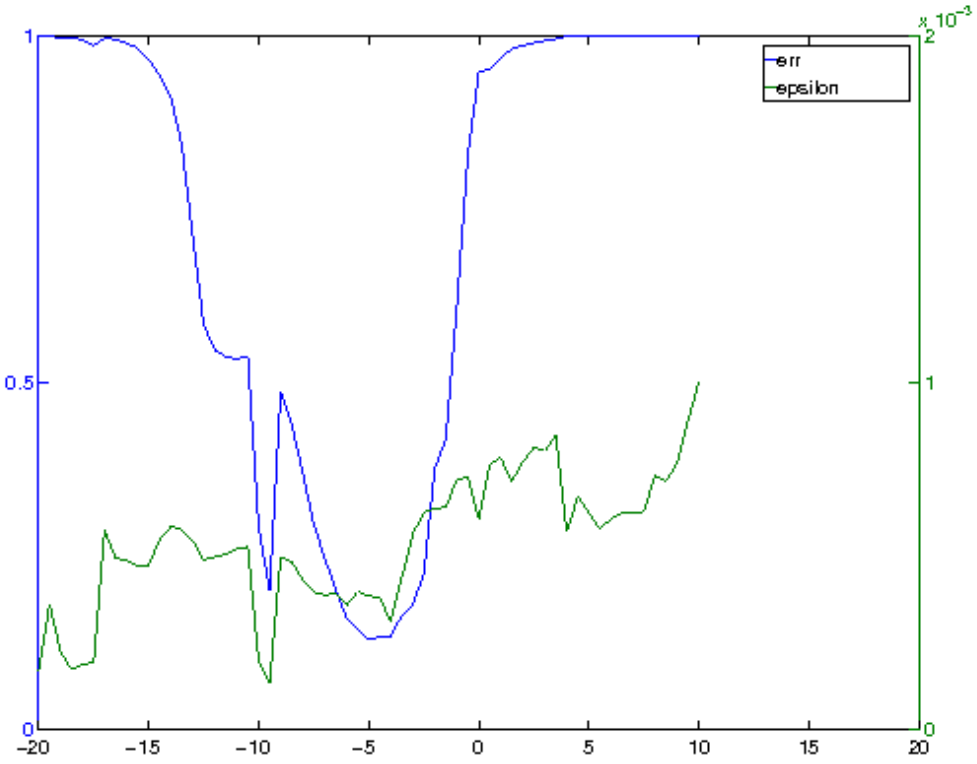


FIGURE 1 – Évolution de l'erreur et de ε en fonction de $\log_2(\lambda)$

Ces résultats sont indiqués sur le graphique 1. La courbe verte représente ε, et la courbe bleue représente l'erreur entre l calculé et l_0, définie par :

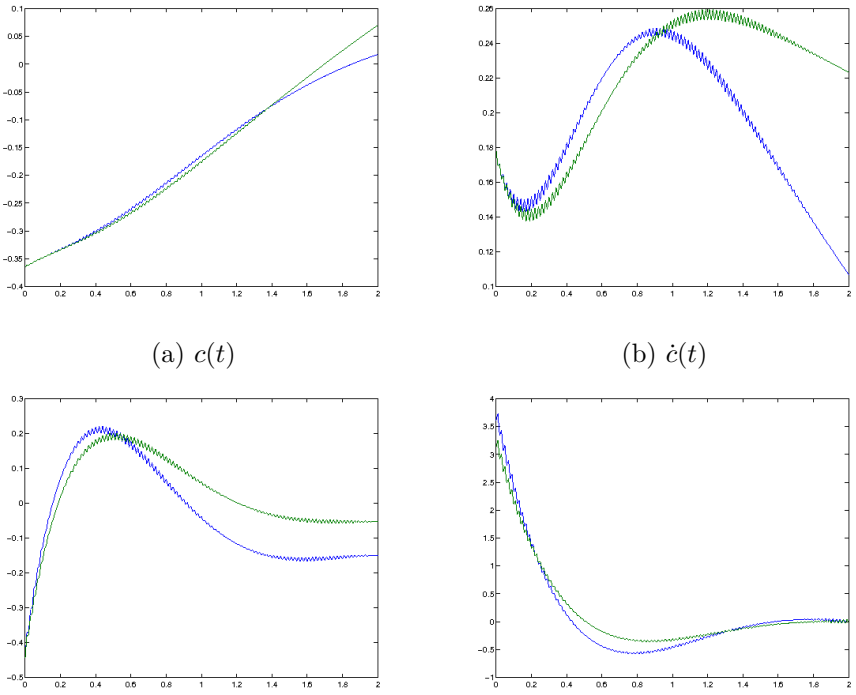
$$e(l,l_0) = \left(1 - \frac{\langle l, l_0\rangle^2}{||l||^2||l_0||^2}\right)^{\frac{1}{2}}$$

où toutes les opérations sont dans $\mathbb{R}^{d+1}$ (on considère les polynômes comme des vecteurs).

Les valeurs de l obtenues ont principalement des termes d'ordre 2, comme le coût l_0 initial. Malheureusement, ces termes ne correspondent pas exactement à ceux de l_0 ; on retrouve bien

le terme dominant en c^2, et souvent les termes plus faibles $\ddot{c}^2$ et $\dddot{c}^2$, mais le terme négatif en $c\ddot{c}$ n'apparaît pas clairement dans le « bruit » des autres termes d'ordre 2, où, en revanche, le coefficient de $\dot{c}^2$ est souvent significatif.

En fait, les trajectoires générées ont tendance à tendre rapidement vers 0 à tous les ordres, ce qui favorise sans doute les termes carrés au dépens des doubles produits. En utilisant les coûts calculés, on peut regénérer des trajectoires très proches des trajectoires originales, et les faibles valeurs de ε (de l'ordre de 10^{-5}) indiquent que les trajectoires originales sont optimales pour les coûts calculés. Les résultats sont donc plutôt satisfaisants. On peut notamment regénérer des trajectoires, comme le montre la figure 2.



(a) $c(t)$ (b) $\dot{c}(t)$

(c) $\ddot{c}(t)$ (d) $\dddot{c}(t)$

FIGURE 2 – En bleu, une trajectoire générée directement ; en vert, une trajectoire générée avec un coût calculé.

À nouveau, on observe qu'il est nécessaire de limiter le degré d pour obtenir des résultats significatifs. Le degré d' est limité en pratique à 12 environ par la puissance de calcul et la mémoire (la taille des polynômes augmente rapidement avec d').

4.4 Extension à plusieurs pas

On désire désormais faire prendre deux ou trois valeurs successives à p^*. Le problème, comme on l'a vu, est que la méthode employée pour la résolution inverse ne permet pas de retrouver des coûts l dépendant directement du temps.

Une première méthode suggérée consiste à considérer les intervalles de temps où p^* est constant comme des dimensions supplémentaires. Ainsi, dans le cas de deux pas on pose un nouvel état :

$$x'(t) = \begin{pmatrix} x_1(t) \\ x_2(t) \end{pmatrix} = \begin{pmatrix} x(t) \\ x(t + \frac{T}{2}) \end{pmatrix}$$

où t varie dans $[0, \frac{T}{2}]$.

Cependant, la condition de continuité $x_1(\frac{T}{2}) = x_2(0)$ s'est avérée difficile à exprimer dans le cadre dans lequel nous travaillons, et nous avons renoncé à cette approche.

Une seconde approche plus simple consiste à dédoubler le coût l, puisque le coût recherché ne dépend pas du temps à l'intérieur des intervalles $[0, \frac{T}{2}]$ et $[\frac{T}{2}, T]$. On écrit ainsi :

$$l(t, x, u) = \begin{cases} l_1(x, u) & \text{si } t < \dfrac{T}{2} \\ l_2(x, u) & \text{sinon} \end{cases}$$

Pour affiner le résultat, j'ai introduit après coup plusieurs changements à la méthode. Ainsi, on peut également dédoubler λ : au lieu de minimiser $\lambda(||l_1||_1 + ||l_2||_1)$, on minimise $\lambda_1||l_1||_1 + \lambda_2||l_2||_1$. v peut aussi être dédoublée : on la représente par un polynôme v_1 sur $[0, \frac{T}{2}]$, et un autre, v_2, sur $[\frac{T}{2}, T]$ (en imposant l'égalité de v_1 et v_2 en $t = \frac{T}{2}$). Le principal inconvénient du dédoublement de λ est que la recherche de la valeur optimale de (λ_1, λ_2) prend beaucoup plus de temps que lorsqu'il n'y a qu'une seule dimension.

Ces fins réglages ont cependant peu changé les résultats obtenus pour les meilleures valeurs de λ ou (λ_1, λ_2). La qualité du résultat en fonction de λ évolue comme précédemment ; on observe une nette amélioration au-delà d'une certaine valeur, puis la qualité se dégrade. Le graphique 3 donne un exemple de cette tendance. La courbe verte représente ε, qui est très faible tant que λ est raisonnablement petit. Les autres courbes représentent l'erreur des polynômes l_1 et l_2 calculés par rapport aux polynômes originels l_1^* et l_2^*, définie comme :

$$e(l, l^*) = \left(1 - \frac{\langle l, l^* \rangle^2}{||l||^2 ||l^*||^2}\right)^{\frac{1}{2}}$$

où l et l^* où toutes les opérations sont dans $\mathbb{R}^{d+1}$ (l et l^* sont considérés comme des vecteurs). La courbe rouge représente l'erreur de l_1 ; nous reviendrons sur la différences entre les deux courbes restantes.

On voit que l_1 et l_2 se calculent le mieux pour des valeurs différentes de λ. Malheureusement, ce résultat demeure vrai lorsqu'on dédouble λ, c'est-à-dire qu'il n'existe pas de couple (λ_1, λ_2) avec lequel on puisse calculer à la fois l_1 et l_2 de manière satisfaisante.

4.5 Résultats

Comme on peut le voir sur la figure 3, l'erreur pour l_1 a un minimum assez élevé d'environ $0,2$. En pratique, on obtient certes des coefficients dominants en c^2 et $\ddot{c}^2$, ainsi qu'un terme négatif en $c\ddot{c}$, mais les autres termes d'ordre deux – notamment $\dot{c}^2$ – apparaissent tous avec des coefficients non négligeables, voire plus importants que les termes désirés, et tout cela varie beaucoup en fonction de λ. Il est donc assez difficile d'interpréter les résultats de manière concluante.

Cependant, le résultat le plus décevant concerne l_2. Pour rappel, comme p^* est non nul :

$$l_2^*(x, u) = l_2^*(c, \dot{c}, \ddot{c}, \dddot{c}) = \dddot{c}^2 + Kc^2 + Kp^{*2} + K\left(\frac{z}{g}\right)^2 \ddot{c}^2 - 2K\frac{z}{g}c\ddot{c} - 2Kp^*c + 2k\frac{z}{g}p^*\ddot{c}$$

Or, les valeurs optimales de l_2 obtenues sont très proches de celles de l_1. La courbe bleue de la figure 3 représente ainsi l'écart de l_2 à l_1^*, avec un minimum à $0,2$, comme précédemment. L'écart de l_2 à l_2^*, représenté en violet, a un minimum bien plus élevé à $0,4$. Cela s'explique par le fait que les termes de degré 1 de l_2 comme de l_1 sont systématiquement négligeables devant les coefficients les plus gros. Cela est très dommageable, car en l'absence de ces termes rien n'entraîne le mouvement vers l'avant des trajectoires, et si on utilise les coûts calculés pour regénérer des trajectoires, celles-ci se contentent de tendre vers 0 et non vers p^*.

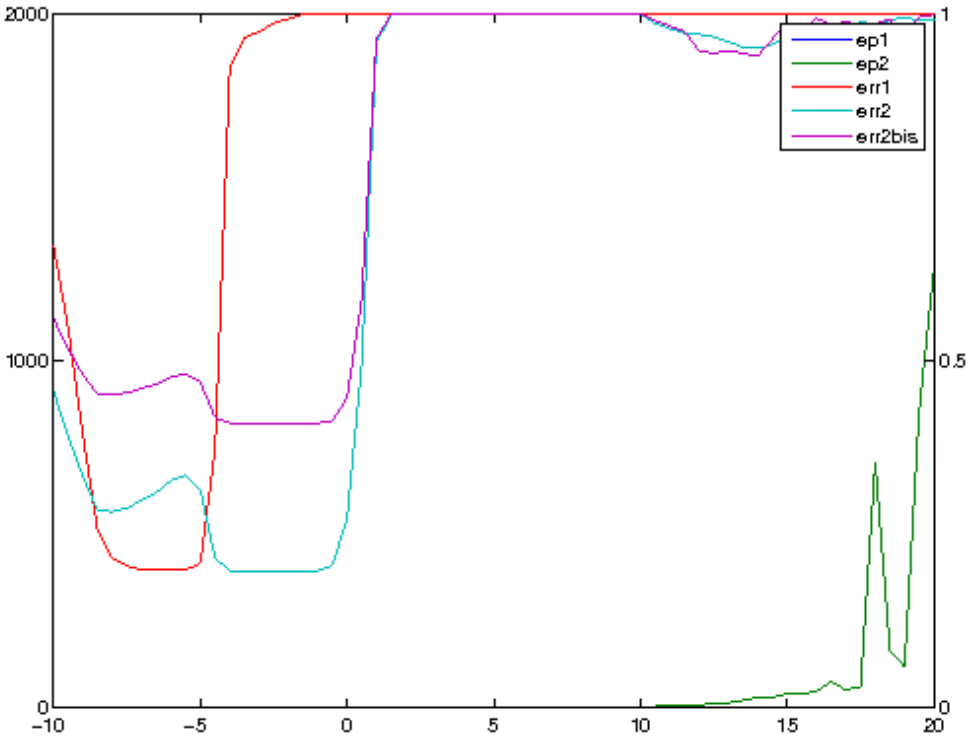


FIGURE 3 – Évolution de l'erreur en fonction du logarithme binaire de λ.

5 Conclusion et perspectives de recherche

Le résultat obtenu semble étrange ; en théorie, si ε est petit, les trajectoires générées au départ devraient être quasi-optimales pour les coûts calculés. On a pourtant du mal à accepter que des trajectoires qui tendent à s'éloigner de l'origine avec p^* soient optimales pour des coûts sans termes linéaires.

En fait, on observe que ε est bien plus faible (de l'ordre de 10^{-5} voire 10^{-6}) lorsque les positions initiales choisies s'écartent beaucoup de zéro, que si elles en sont toutes très proches (ε est alors plutôt de l'ordre de 10^{-2}). Dans le premier cas, il semble que le mouvement de retour vers le voisinage de l'origine (et de $p^* = 0,2$) l'emporte sur la poursuite de p^*. Si on regénère des trajectoires avec les polynômes obtenus, celles-ci suivent les trajectoires originelles au moins pour une moitié du mouvement. Dans le second cas, le solveur semble incapable de calculer un coût et une fonction valeur pour lesquels les trajectoires seraient optimales.

Quoiqu'il en soit, la méthode n'est manifestement pas en état d'être appliquée à des données réelles. En plus des problèmes déjà évoqués, les valeurs de λ pour lesquelles on obtient les meilleurs résultats ne semblent pas avoir de qualité intrinsèque qui permettrait de les reconnaître si l'on ne savait pas ce que l'on cherchait. Cela dit, cela n'empêcherait pas de mener des expériences en regénérant des trajectoires avec les coûts calculés pour évaluer la qualité des solutions.

Une source d'erreur probable est le grand nombre de termes (de l_2^* notamment) et les écarts entre les différents coefficients (rappelons que $K = 10^5$ et $\frac{z}{g} = \frac{1}{10}$). L'écart important entre les différents ordres dans les trajectoires considérées cause sans doute également des imprécisions : alors que c et $\dot{c}$ restent de l'ordre de 1, $\dddot{c}$ a des variations très importantes avec un pic de l'ordre de -100 au moment du saut de p^*, qui correspond à une quasi-discontinuité de $\ddot{c}$.

Cela a pu entraîner des problèmes numériques, d'où certains résultats étranges, notamment lorsqu'on considère des positions initiales proches de 0. Il conviendrait sans doute d'expérimenter le dédoublement du coût sur un problème plus simple, afin de s'assurer de l'efficacité de la méthode.

Certains aspects du problème méritent sans doute d'être approfondis, notamment l'influence du choix des trajectoires données, ou des degrés. Notons que dans les deux cas, les limitations de ressources (mémoire et capacité de calcul) obligent à se restreindre. Dans le cas du degré il semble difficile d'éviter cela vu que la complexité en degré est importante (la taille du problème d'optimisation, notamment, est en d'^4).

Dans la visée d'appliquer l'expérience obtenue à des données réelles, il n'est pas vraiment utile d'utiliser des données vraiment invraisemblables. Cependant, il serait sans doute intéressant de faire des expériences avec des données moins uniformes que celles utilisées (qui étaient générées avec des positions initiales naïvement aléatoires).

Au-delà des difficultés pratiques, la méthode proposée par [4] s'est avérée capable de donner des résultats intéressants, et mérite d'être développée plus avant. Elle présente les avantages de ne pas nécessiter de résoudre de problèmes directs, d'être simple dans sa formulation, et de fournir la fonction valeur ce qui permet de faire certaines vérifications (autre piste de poursuite possible de cette recherche, soit dit en passant).

De mon point de vue, ce stage a présenté un caractère expérimental auquel je n'étais pas vraiment habitué, ainsi que l'expérience d'un centre de recherche plus technique. Malgré les conclusions plutôt décevantes de mon travail, j'ai pu m'initier à l'optimisation numérique (y compris sur certains points qui n'étaient pas directement liés à mon travail) et avoir un aperçu de la robotique.

Références

[1] Pierre-Brice Wieber : On the stability of walking systems. *In Proceedings of the International Workshop on Humanoid and Human Friendly Robotics*, Tsukuba, Japan, 2002.

[2] Héctor J. Sussmann et Jan C. Willems : 300 years of optimal control : From the brachystochrone to the maximum principle, 1997.

[3] Jorge Nocedal et Stephen Wright : *Numerical Optimization*, chapitre 16. Springer, 2006.

[4] Edouard Pauwels, Didier Henrion et Jean-Bernard Lasserre : Inverse optimal control with polynomial optimization, 2014. arXiv:1403.5180.

[5] Krishnamurthy Dvijotham et Emanuel Todorov : Inverse optimal control with linearly-solvable mdps. *In International Conference on Machine Learning*, 2010.

[6] Didier Henrion : Optimization on linear matrix inequalities for polynomial systems control. http://homepages.laas.fr/henrion/Papers/henrion-grenoble14.pdf, 2014.

[7] Article « Linear Quadratic Regulator » de Wikipedia. https://en.wikipedia.org/wiki/Linear-quadratic_regulator.