



UNIVERSITÉ  
LAVAL

Faculté des sciences et de génie  
Département de génie mécanique

# Optimisation de trajectoires pour augmenter les capacités d'un robot

Présenté au  
Professeur Claude-Guy Quimper

Par :  
André Gallant - 111 062 993  
Alexis Fortin-Côté - 908 191 694  
Vincent Babin - 910 227 872

dans le cadre du cours :  
**IFT-7020**  
**Optimisation Combinatoire**

25 avril 2014

# 1 Introduction

La robotique est omniprésente dans l'industrie manufacturière. Les robots sont utilisés pour déplacer des charges et faire des mouvements répétitifs. Un des défauts des robots utilisés présentement est que leur charge utile, c.-à-d. la masse maximale qu'ils peuvent manœuvrer, est faible par rapport à la masse du robot lui-même. Cela fait en sorte que les robots deviennent rapidement très gros et très cher dès que les charges à soulever augmentent. Présentement, la charge utile est déterminée comme étant la masse que le robot peut déplacer partout dans son espace de travail. Cet espace de travail inclut donc les situations où le robot a le plus de difficulté à soulever la charge, par exemple : un bras en pleine extension. Ce qui a comme effet d'imposer une charge utile faible. La solution envisagée pour pallier à ce problème est d'exploiter la dynamique du robot. En exploitant l'inertie de la masse, il est possible d'emmener de lourdes charges à des positions inatteignables dans un contexte normal (quasi-statique). Le défi est de trouver une stratégie pour générer des trajectoires exploitant ce phénomène. Ce travail présente trois méthodes innovantes pouvant potentiellement résoudre ce problème peu souvent abordé par la robotique. Les trois méthodes présentées reposent sur des principes très différents d'optimisation. La première repose sur l'optimisation de coefficients de polynômes générant des trajectoires admissibles. Le deuxième propose la discrétisation de l'espace d'état pour pouvoir appliquer la recherche dans un graphe. La dernière méthode repose sur l'utilisation d'une méthode d'optimisation par essais particuliers (PSO) pour trouver des trajectoires. L'objectif de chacun de ces méthodes est donc de pouvoir générer au moins une trajectoire qui permet au robot d'emmener une masse à un point inatteignable dans des conditions quasi-statiques.

# 2 Nomenclature

- $x_i$  : position de l'articulation  $i$
- $\dot{x}_i$  : vitesse de l'articulation  $i$
- $\ddot{x}_i$  : accélération de l'articulation  $i$
- $\mathbf{x}$  : vecteur position des articulations
- $\dot{\mathbf{x}}$  : vecteur vitesse des articulations
- $\ddot{\mathbf{x}}$  : vecteur accélération des articulations
- $u_i$  : effort de l'articulation  $i$
- $\mathbf{u}$  : vecteur des efforts articulaires

- $t$  : temps actuel
- $T$  : temps total de la trajectoire
- $ddl$  : degrés de liberté (nombre d'articulations)
- $\mathbf{y}$  : vecteur des variables d'optimisation

# 3 Manipulateurs étudiés

Dans ce projet, deux manipulateurs ont été étudiés. Le premier est tout simplement un pendule simple, c.-à-d. une masse ponctuelle fixée à l'extrémité d'une tige rigide, tel que montré dans la figure 1. Le mécanisme est actionné à sa seule articulation. L'angle de cette articulation est représenté par  $\theta$ . Puisque c'est la seule articulation,  $\mathbf{x} = x_1 = \theta$  (voir section 2).

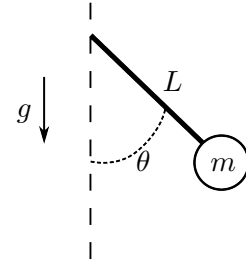


FIG. 1 – Manipulateur R (pendule)

Le deuxième manipulateur étudié est le pendule, mais avec l'articulation rotoïde montée sur une articulation prismatique. Ce manipulateur est montré à la figure 2. Puisque  $\rho$  est la première articulation et  $\theta$  est la seconde, le vecteur des positions articulaires est  $\mathbf{x} = [\rho, \theta]^T$ .

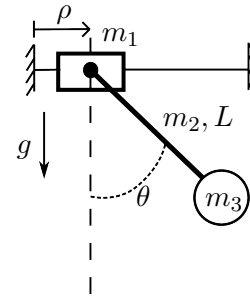


FIG. 2 – Manipulateur PR (chariot-pendule)

## 3.1 Équations dynamiques

Puisqu'il est désiré de déterminer des trajectoires pour effectuer des tâches qui excèdent les capacités

normales des manipulateurs, il est nécessaire de calculer les efforts articulaires nécessaires pour effectuer un mouvement. Pour le pendule simple l'équation du mouvement est la suivante :

$$\tau = mL^2\ddot{\theta} + \dots \quad (1)$$

où  $\tau$  est le couple nécessaire à l'articulation  $\theta$  pour faire le mouvement décrit par  $\theta$ ,  $\dot{\theta}$  et  $\ddot{\theta}$ . Puisque  $\theta$  est la seule articulation,  $\mathbf{u} = u_1 = \tau$ .

Le manipulateur de la figure 2 possède deux degrés de liberté (ddl). Un système de deux équations est alors requis pour calculer les efforts articulaires :

$$\mathbf{u} = A\ddot{\mathbf{x}} + \mathbf{b}(\mathbf{x}, \dot{\mathbf{x}}) + \mathbf{c}(\mathbf{x}) \quad (2)$$

où  $A$  est la matrice d'inertie du manipulateur,  $\mathbf{b}$  contient les termes de l'accélération centrifuge et de Coriolis et  $\mathbf{c}$  contient les termes de l'accélération gravitationnelle.

## 4 Méthode polynôme

La première méthode d'optimisation de trajectoire explorée dans ce projet est une méthode qui fait une approximation de la trajectoire de chaque articulation par un polynôme d'ordre  $n$ . Cette partie est alors divisée en trois sections. Tout d'abord, les modèles associés à cette méthode sont définis. Ensuite, les critères d'évaluation de la performance d'une trajectoire sont explicités. Enfin, les résultats montrant la performance des modèles et des fonctions objectifs sont présentés.

### 4.1 Modèles

Cette section présente deux modèles permettant de faire une approximation polynomiale de la trajectoire articulaire d'un robot. Avant de présenter les modèles, certaines propriétés de l'approximation polynomiale doivent être explicitées. Tout d'abord, l'équation d'un polynôme qui définit la position d'une articulation est

$$x_i(t) = a_0 + a_1t + a_2t^2 + \dots + a_nt^n. \quad (3)$$

Tel que défini précédemment, la tâche d'un robot dans le contexte de ce projet est bornée par un certain nombre de conditions initiales et finales. Ainsi, certains coefficients du polynôme peuvent être déterminés directement avec l'équation

$$x_i(0) = a_0, \quad \dot{x}_i(0) = a_1, \quad \ddot{x}_i(0) = a_2 \quad (4)$$

et certains paramètres imposent les contraintes :

$$\begin{aligned} x_i(T) &= a_0 + a_1T + a_2T^2 + \dots + a_nT^n \\ \dot{x}_i(T) &= a_1 + 2a_2T + \dots + na_nT^{n-1} \\ \ddot{x}_i(T) &= 2a_2 + \dots + n(n-1)a_nT^{n-2} \end{aligned} \quad (5)$$

Lorsque le temps  $T$  est connu, ces contraintes peuvent être écrites sous la forme d'un système linéaire. Par exemple, pour une tâche avec deux conditions initiales et deux conditions finales définie par un polynôme d'ordre 4, le système qui représente contraintes associées aux conditions limites est

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & T & T^2 & T^3 & T^4 \\ 0 & 1 & 2T & 3T^2 & 4T^3 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} x_i(0) \\ \dot{x}_i(0) \\ x_i(T) \\ \dot{x}_i(T) \end{bmatrix} \quad (6)$$

Ce système est sous-déterminé puisqu'il y a cinq coefficients à déterminer, mais seulement quatre contraintes linéaires. C'est justement sur ce fait que les modèles se basent. Par contre, les deux modèles diffèrent sur la façon de combler le manque d'équations. Le premier modèle est le plus simple, il fixe simplement le nombre manquant de coefficients et résout le reste avec les conditions limites. En reprenant le même exemples que l'équation (6), le système serait modifié ainsi :

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & T & T^2 & T^3 & T^4 \\ 0 & 1 & 2T & 3T^2 & 4T^3 \\ 0 & 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} x_i(0) \\ \dot{x}_i(0) \\ x_i(T) \\ \dot{x}_i(T) \\ y_{4,i} \end{bmatrix} \quad (7)$$

où  $y_{4,i}$  est le paramètre d'optimisation correspondant au coefficient  $a_4$  de l'articulation  $i$ . Les variables d'optimisation de chaque articulation ainsi que le temps total de la trajectoire peuvent donc être insérés dans un vecteur :

$$(M1) \quad \mathbf{y} = [y_{k1,1} \dots y_{n1,1} \quad y_{k2,2} \dots y_{n2,2} \dots T]^T \quad (8)$$

où  $ki$  est le nombre de conditions limites de la tâche pour l'articulation  $i$  et  $ni$  est le nombre de coefficients du polynôme de l'articulation  $i$ .

Le vecteur  $\mathbf{y}$  est alors le vecteur des variables d'optimisation dans lequel le temps final est toujours le dernier élément. Le modèle (M1) de l'équation (8) a un inconvénient majeur : les paramètres sont très couplés avec le temps total. En effet, si le paramètre

$T$  change, la signification des coefficients du polynôme changent également. Afin d'illustrer ce phénomène, la figure 3 montre un exemple d'un polynôme d'ordre 4 avec les mêmes conditions limites et le même coefficient  $a_4$  mais avec un différent temps total. De cette figure, il est apparent qu'avec ce modèle, le temps ne modifie non seulement la durée de la trajectoire mais également la forme du mouvement.

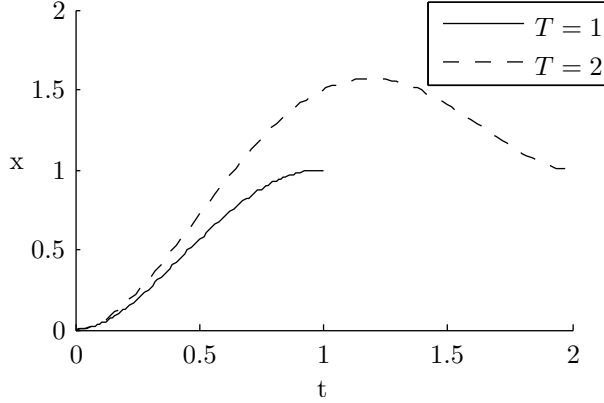


FIG. 3 – Polynômes d'ordre 4 avec  $a_4 = 1$

Afin d'éliminer ce phénomène, un second modèle est proposé. Toujours avec l'objectif de combler le manque d'équations dans le système de l'équation (6), il est possible d'imposer une position à un certain temps de la trajectoire et donc ajouter des contraintes au système. En outre, il est possible de normaliser ce temps avec le temps  $T$  afin que la forme du mouvement soit indépendante du temps total. Dans l'exemple du système de l'équation (6), ce modèle a la forme :

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 1 & T & T^2 & T^3 & T^4 \\ 0 & 1 & 2T & 3T^2 & 4T^3 \\ 1 & 0.5T & (0.5T)^2 & (0.5T)^3 & (0.5T)^4 \end{bmatrix} \begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{bmatrix} = \begin{bmatrix} x_i(0) \\ \dot{x}_i(0) \\ x_i(T) \\ \dot{x}_i(T) \\ y_{0.5,i} \end{bmatrix} \quad (9)$$

où  $y_{0.5,i} = x(0.5T)$  est la position de l'articulation à la moitié de la trajectoire. Le vecteur des variables d'optimisation du second modèle est alors :

$$(M2) \quad \mathbf{y} = [y_{c1,1} \quad y_{c2,1} \dots \quad y_{c1,2} \quad y_{c2,2} \dots \quad T]^T \quad (10)$$

où  $c_j$  est une valeur entre 0 et 1 et représente pourcentage de la trajectoire auquel le point est pris.

Il est à noter qu'il existe une différence de nomenclature entre les deux modèles. En effet, les indices

des paramètres correspondants au premier modèle sont des entiers strictement plus grand que 1 (par exemple :  $y_{4,i}$ ,  $y_{5,i}$ ), alors que les indices des paramètres correspondants au second modèle sont entre 0 et 1 (par exemple :  $y_{0.25,i}$ ,  $y_{0.5,i}$ ).

Le modèle (M2) a deux principaux avantages. Premièrement, tel que décrit si haut, le modèle (M2) ne souffre pas du problème où le temps  $T$  modifie la forme du mouvement. En effet, la figure 4 montre que la trajectoire ne fait que s'étaler horizontalement avec une augmentation du temps  $T$ . Deuxièmement, puisqu'il est plus facile de visualiser un point sur une trajectoire que de visualiser la représentation d'un coefficient d'un polynôme, faire un estimé initial pour le modèle (M2) est beaucoup plus intuitif que pour le modèle (M1).

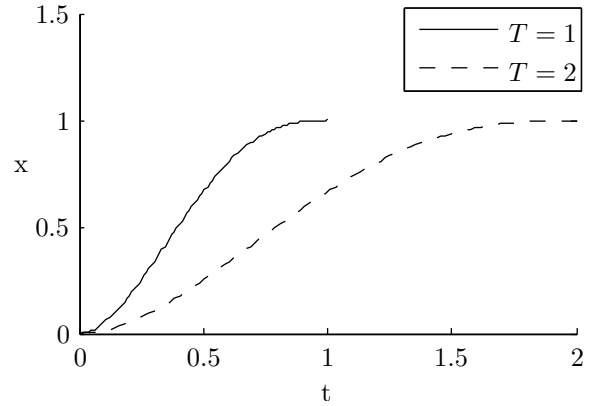


FIG. 4 – Polynômes d'ordre 4 avec  $x(0.25T) = 0.25$

Puisque la fonction polynomiale sous-jacente est la même pour les deux modèles, ceux-ci sont équivalents, c'est-à-dire toute trajectoire pouvant être produite par un modèle peut également être produite par l'autre. Cependant, la distribution de ces trajectoires dans l'espace des paramètres d'optimisation est différente et donc change potentiellement la performance de l'optimisation. Il a été conjecturé que le second modèle donnera des meilleurs résultats que le premier. L'effet du modèle sur la performance de la méthode a été étudié et les résultats sont présentés dans la section 4.3.

## 4.2 Fonctions objectifs

En plus des différents modèles, il est possible de formuler la fonction objectif de différentes façons. Dans cette section du projet, trois fonctions objectifs sont étudiées.

Premièrement, puisque l'objectif est de permettre aux robots d'effectuer des tâches qui excèdent leurs capacités, l'effort maximal et l'effort minimal peuvent être considérés comme des contraintes non-linéaires à être ajoutés au modèle. Dans ce cas, le problème d'optimisation est

$$(P1) \quad \begin{cases} \text{minimiser} & T \\ \text{subject à} & T > 0 \\ & u_i \geq umin_i \\ & u_i \leq umax_i \end{cases} \quad (11)$$

pour tout point sur la trajectoire où  $umax_i$  et  $umin_i$  sont l'effort maximal et l'effort minimal de l'articulation  $i$ , respectivement. Ici, on optimise le temps tout simplement pour empêcher l'optimisation de diverger en temps  $T$ .

La deuxième fonction étudiée est celle où le maximum de la valeur absolue de l'effort est à minimiser, c'est-à-dire :

$$(P2) \quad \begin{cases} \text{minimiser} & w_T T + \sum_{i=1}^{ddl} \max |u_i| w_i \\ \text{subject à} & T > 0 \end{cases} \quad (12)$$

où  $w_T$  et  $w_i$  sont des facteurs de poids qui permettent de donner plus d'importance à certains paramètres.

La dernière fonction objectif étudiée dans ce projet est celle où la valeur absolue de l'effort articulaire est à minimiser en tout instant, c'est-à-dire :

$$(P3) \quad \begin{cases} \text{minimiser} & w_T T + \sum_{i=1}^{ddl} w_i \int_0^T |u_i| dt \\ \text{subject à} & T > 0 \end{cases} \quad (13)$$

En résumé, (P1) impose une contrainte sur l'amplitude de l'effort de chaque articulation, (P2) minimise l'amplitude des efforts et (P3) minimise le travail effectué par chaque articulation.

### 4.3 Expérimentation

Cette section présente les résultats des optimisations avec les différents modèles et les différentes fonctions objectifs présentés dans les sections 4.1 et 4.2, respectivement. L'optimisation en question a été effectuée à l'aide de la fonction *fmincon* du logiciel MATLAB. Celle-ci est une fonction de minimisation avec contraintes non-linéaires et utilise un

algorithme de programmation quadratique séquentielle (SQP). L'algorithme SQP est une méthode très efficace en optimisation non-linéaire avec contraintes non-linéaires qui effectue séquentiellement une approximation quadratique du Lagrangien de la fonction objectif (multiplicateurs de Lagrange). Une explication détaillée de son implémentation par MATLAB est présentée dans [1].

La méthode d'approximation polynomiale de ce projet a été appliquée au manipulateur PR (chariot-pendule) de la figure 2. Les figures 5 et 6 montrent les trajectoires articulaires des six permutations des modèles de la section 4.1 et des fonctions objectifs de la section 4.2. La notation MiPj désigne le modèle  $i$  avec la fonction objectif  $j$ . Les figures 7 et 8 montrent les efforts articulaires correspondantes.

Le tableau suivant présente les poids utilisés dans chaque fonction objectif :

	P1	P2	P3
$w_T$	1	10e-5	1
$w_1$	n/a	1	0.1
$w_2$	n/a	2	0.2

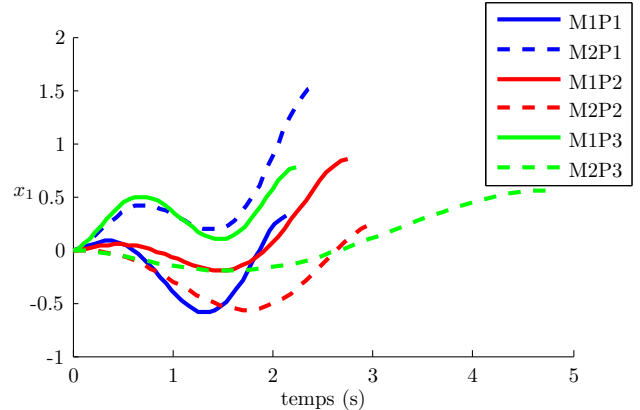


FIG. 5 – Trajectoires de l'articulation prismatique

Il peut être vu que dans tous les cas, le premier modèle donne des trajectoires avec un temps inférieur au deuxième alors que les efforts max sont très semblables. L'hypothèse de la section 4.1 que le deuxième modèle est meilleur est faux quand l'estimé initial est le même. Par contre, il est largement plus facile de trouver un estimé initial avec le deuxième modèle. Le deuxième modèle pourrait donc être utile pour trouver un estimé initial mais que l'optimisation serait effectuée avec le premier modèle. En outre, il peut être vu que la première

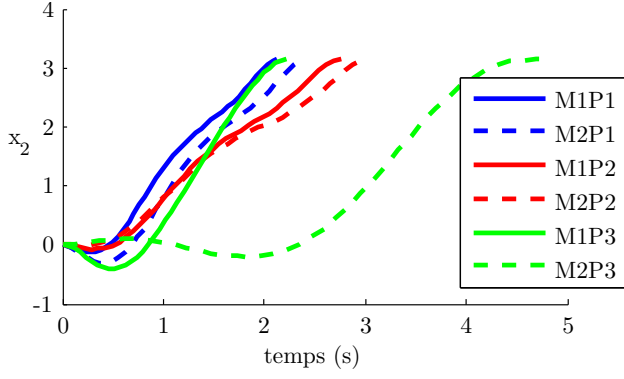


FIG. 6 – Trajectoires de l'articulation rotoïde

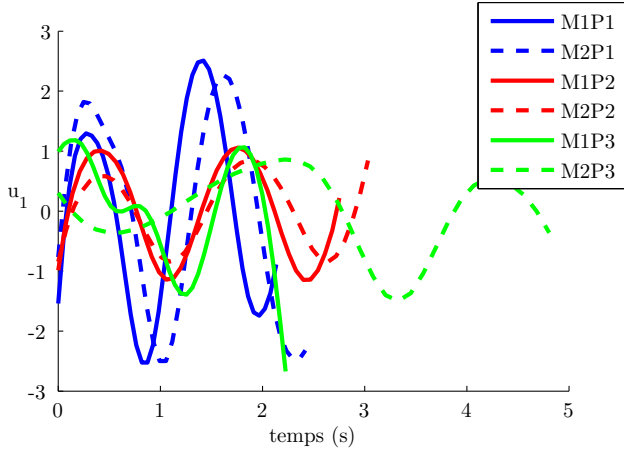


FIG. 7 – Efforts de l'articulation prismatique

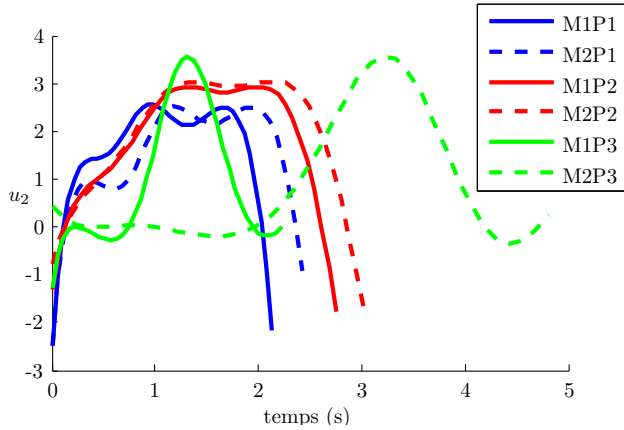


FIG. 8 – Efforts de l'articulation rotoïde

fonction objectif semble donner de meilleurs résultats que les autres.

Il serait, cependant, prématuré de généraliser ces résultats. En effet, ces résultats n'ont été appliqués qu'à une tâche à être effectuée par un manipulateur particulier. Les résultats montrent par contre, qu'il est possible de réduire les efforts articulaires

et donc d'augmenter la charge utile d'un robot en employant cette méthode. Il y a alors plusieurs avenues de recherche intéressantes à explorer dans des travaux futurs.

## 5 Recherche dans un graphe

Une des avenues explorées pour générer des trajectoires permettant l'exploitation de la dynamique d'un robot pour augmenter ces capacités est basée sur la recherche dans un graphe. Cette méthode se base sur la discrétisation de l'espace d'état et son exploration. L'état d'un mécanisme est défini par un vecteur de ses coordonnées articulaires  $\mathbf{x}$  ainsi qu'un vecteur de ses vitesses articulaires  $\dot{\mathbf{x}}$ . Par exemple, pour un pendule simple, l'état du mécanisme est représenté par l'angle par rapport à son axe de référence et par la dérivé en fonction du temps de cet angle, comme présenté à la figure 1. On remarque ici que le vecteur des coordonnées articulaires est  $\mathbf{x} = [\theta]$  et le vecteur des vitesses articulaires est  $\dot{\mathbf{x}} = [\dot{\theta}]$ .

### 5.1 Problème simple

Le problème sur lequel la méthode sera appliquée initialement est celui du pendule simple présenté à la figure 1. L'objectif est de trouver la trajectoire permettant au pendule de passer de l'état initial, vitesse nulle et position basse, à l'état final, vitesse nulle et position haute, tout en respectant les contraintes sur le couple maximum pouvant être appliqué à l'articulation. Pour que les solutions à ce problème représentent une augmentation des capacités du robot, le couple maximum devra être inférieur au couple nécessaire au maintien du pendule à l'horizontale. Ce couple maximum devra donc être

$$\tau_{max} \leq mgL \quad (14)$$

où  $m$  est la masse du pendule,  $L$  est la longueur de la tige et  $g$  est l'accélération gravitationnelle.

### 5.2 Discrétisation

Pour pouvoir discrétiser l'espace d'état, le temps doit lui aussi être discrétisé. Un choix est effectué de le discrétiser selon un pas de temps constant ( $\delta t$ ). Ce choix de paramètre est d'une grande importance, car les équations dynamiques seront linéarisées sur cette

plage. Un pas de temps trop grand permet à des solutions infaisables d'être considéré, car la linéarisation peut éliminer les sections violant les contraintes. De l'autre côté, la taille du graphe augmente proportionnellement avec  $(\frac{1}{\delta t})^n$ , où  $n$  est la dimension de l'espace d'état. Un compromis doit donc être fait.

Pour ce qui est de la discrétisation de l'espace d'état, un compromis similaire s'impose. Le choix de discrétisation est largement dépendant de la discrétisation temporelle. La première étape est de borner l'espace d'état. Dans le problème du pendule simple, le domaine de la coordonnée articulaire est simple, il est  $[-\pi, \pi]$ . Pour borner la vitesse articulaire, on doit poser une vitesse maximale et minimale. Une borne maximale pour le problème est de trouver la vitesse au point bas lorsque le pendule tombe à partir du point haut, ce qui nous assure d'englober la solution. L'équation de la vitesse maximale obtenue est donc

$$\dot{\theta}_{max} = \sqrt{\frac{4g}{L}} \quad (15)$$

Il en découle que le domaine de la vitesse articulaire est  $[-\dot{\theta}_{max}, \dot{\theta}_{max}]$ .

La deuxième étape est de déterminer les pas de discrétisation des variables d'états. Pour ce qui est du pas de discrétisation de la coordonnée articulaire, il est possible de trouver une borne supérieure en déterminant la distance que peut parcourir la coordonnée articulaire partant de la position basse à vitesse nulle en lui appliquant le couple max ( $\tau_{max}$ ) pendant le pas de temps ( $\delta t$ ). Il en découle l'équation suivante

$$\delta\theta \leq \frac{\tau_{max}}{2mL^2} (\delta t)^2. \quad (16)$$

Cette équation représente le déplacement maximal que le pendule peut effectuer depuis sa position basse en un pas de temps. Le pas de discrétisation  $\delta\theta$  ne peut donc pas être supérieur à cette valeur, car le premier déplacement serait infaisable. Une hypothèse suppose que ce premier mouvement sera un des plus lents de la trajectoire. Une discrétisation correspondant à une fraction de ce pas peut donc être utilisée.

De façon similaire, pour déterminer le pas de discrétisation de la vitesse articulaire, on utilise la situation position base et vitesse nulle pour trouver la vitesse atteinte en un pas de temps  $\delta t$  au couple maximal. On obtient ainsi l'équation

$$\delta\dot{\theta} \leq \frac{\tau_{max}}{mL^2} \delta t. \quad (17)$$

Avec l'hypothèse que ce mouvement est l'un des plus lents, il est donc aussi possible d'utiliser une fraction de ce pas pour discrétiser.

Il faut être conscient qu'une discrétisation trop grossière de l'espace d'état élimine des solutions possibles et souvent optimales. Par exemple, une solution au problème du pendule simple peut inclure un passage à basse vitesse à  $\theta = 90^\circ$  et est une solution optimale qui minimise le travail effectué. Cette solution demande alors une discrétisation très fine de  $\dot{\theta}$  à cet endroit. Si ce critère n'est pas atteint, cette solution sera alors ignorée. Il serait possible d'affiner la discrétisation aux points les plus critiques, mais ce sujet ne sera pas abordé dans ce travail.

### 5.3 Arbre de recherche

Suite à la discrétisation, il est possible de générer l'arbre de recherche. Les nœuds sont représentés par une paire de variables d'états  $(\theta_i, \dot{\theta}_j)$ . Les arêtes représentent la possibilité de passer d'un nœud à l'autre et doivent donc respecter la dynamique du robot. Il y a donc une arête entre le nœud  $(\theta_i, \dot{\theta}_j)$  et le nœud  $(\theta_k, \dot{\theta}_l)$  si la dynamique du système permet de passer de l'un à l'autre pendant le temps  $\delta t$ . Pour qu'il soit possible d'atteindre une position le trajet on doit respecter en tout point l'équation suivante

$$\tau_{max} \geq mL^2\ddot{\theta} + mgL\sin(\theta). \quad (18)$$

De plus, l'équation

$$\theta_k = \dot{\theta}t + \theta_i \quad (19)$$

doit aussi être respecté. Il est possible de linéariser entre deux points en sachant que  $\delta t$  est petit avec

$$\ddot{\theta} = \frac{\dot{\theta}_l - \dot{\theta}_j}{\delta t} \quad (20)$$

$$\theta = \frac{\theta_k + \theta_i}{2}. \quad (21)$$

On suppose donc le couple constant durant le temps  $\delta t$ . Il est alors possible de déterminer s'il est possible d'atteindre le nœud  $(\theta_k, \dot{\theta}_l)$  du nœud  $(\theta_i, \dot{\theta}_j)$  avec l'équation

$$\tau_{max} \geq mL^2 \frac{\dot{\theta}_l - \dot{\theta}_j}{\delta t} + mgL\sin\left(\frac{\theta_k + \theta_i}{2}\right) \quad (22)$$

et

$$\theta_k \approx \frac{\dot{\theta}_l + \dot{\theta}_j}{2} \delta t + \theta_i. \quad (23)$$

L'équation (23) doit être une approximation, car la discrétisation de l'espace d'état ne permet pas l'égalité. Des bornes trop permissives pourraient créer une situation où l'atteinte du point violerait les contraintes de couple même si l'équation (22) est respectée.

Avec ces contraintes, il est facile de déterminer les arêtes entre chaque nœud. Par contre, générer le graphe en entier avant la fouille peut créer une structure gigantesque et n'est donc pas faisable. De plus, la plupart des nœuds resteront inexplorés, il est donc inefficace de chercher leurs arêtes. La méthode employée repose sur la recherche des nœuds adjacents qu'au moment de l'exploration d'un nœud. Aussi, déterminer les arêtes d'un nœud est un calcul couteux si on doit vérifier l'existence d'une arête avec tous les autres nœuds du graphe. Pour pallier à ce problème, l'exploration du graphe ne se fera que sur deux arêtes pour chaque nœud visité. Ces arêtes correspondent à l'utilisation du couple maximum et du couple minimum. Cette méthode retire des solutions possibles, mais réduit considérablement la complexité du graphe. De plus, si une solution existe, la solution optimale dans le temps sera constituée d'une trajectoire nécessitant le couple maximum à ces articulations sur toute la plage (solution du Bang-Bang). On s'assure donc, en ne choisissant que les arêtes correspondant au couple maximum, que la solution optimale est conservée.

Pour trouver les nœuds adjacents aux nœuds en visites on doit d'abord trouver l'accélération maximale que l'on peut imposer grâce à l'équation

$$\ddot{\theta} = \frac{\tau_{max}}{mL^2} - \frac{g}{L \sin(\theta_i)}. \quad (24)$$

On approxime considérera alors cette accélération comme constante sur le temps  $\delta t$ . On peut ensuite déterminer la vitesse maximale et minimale que peut atteindre le pendule après le temps  $\delta t$  avec

$$\dot{\theta}_{\{l+,l-\}} = \dot{\theta}_j \pm \ddot{\theta} \delta t. \quad (25)$$

Il est ensuite nécessaire d'arrondir les deux valeurs de vitesse à des vitesses correspondantes à une valeur discrète de notre espace d'état. Pour se faire, on arrondit de façon conservatrice en choisissant la valeur la plus proche qui requière une accélération

plus faible à atteindre. Ces vitesses sont ensuite utilisées pour calculer les deux positions articulaires avec l'équation

$$\theta_{\{k+,k-\}} = \theta_i + \frac{\dot{\theta}_j + \dot{\theta}_{\{l+,l-\}}}{2} \delta t \quad (26)$$

En arrondissant de la même façon qu'avec les vitesses articulaires, on obtient les deux nouveaux nœuds :  $(\theta_{k+}, \dot{\theta}_{l+})$  et  $(\theta_{k-}, \dot{\theta}_{l-})$ . La fouille est donc poursuivie sur chacun de ces deux nœuds.

## 5.4 Fouille

La fouille est basée sur un algorithme de fouille en profondeur modifié. La fouille s'effectue du nœud initial représentant l'état initial du mécanisme et branche sur chacun des deux nœuds adjacents trouvés plus haut. Une solution est trouvée lorsque l'algorithme tombe sur le nœud correspondant à l'état final. Étant donnée la méthode utilisée plus haut, qui ne suit que les arêtes correspondant au couple maximal, on doit imposer un critère de solution plus souple que de tomber directement sur le nœud final, car ceci crée des artéfacts en fin de trajectoire, étant donné que le nœud final peut être adjacent au nœud en exploration, sans être relié par une des deux arêtes qui sera suivi dans la fouille. On doit donc rendre tous les nœuds adjacents au nœud final des nœuds pouvant être acceptés comme solution. On peut générer cette liste de nœuds grâce aux équations 22 et 23.

Lors de l'exploration d'un nœud, un choix doit être fait entre suivre l'arête correspondante à couple maximum ou à celle du couple minimum en premier. Une heuristique est donc appelée pour faire ce choix. Cette heuristique a une importance majeure dans le temps de recherche d'une solution. Différentes approches peuvent être utilisées comme la norme 1,2 ou inf entre le prochain nœud et la solution finale ou bien privilégiée toujours le couple maximum. Ces différentes heuristiques sont intéressantes, mais ne seront pas abordées en détail dans ce travail.

Pour que la fouille soit efficace, on doit pouvoir générer des retours arrière. Ces retours arrière sont générés lorsque les deux arêtes suivies mènent à des nœuds invalides. Un nœud peut être invalide, pour plusieurs raisons. La première raison est que ce nœud a déjà été visité, c.-à-d. qu'il a déjà causé un retour arrière. Une deuxième raison est que le nœud fait partie de la liste de nœud considéré "en visite", c.-à-d. que le choix qui a été fait sur ce nœud a mené



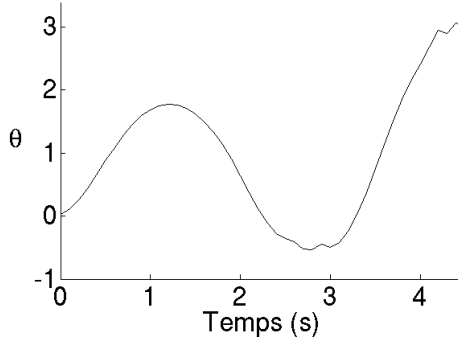


FIG. 9 – Solution fouille dans le graphe

au nœud présentement visité. Une condition spéciale correspond à l’atteinte d’un nœud qui fait partie de la meilleure solution trouvée précédemment. Si ce nœud est atteint plus rapidement que le temps où il a été atteint lors de la meilleure solution, une nouvelle meilleure solution est donc trouvée. Si ce nœud est atteint plus lentement que lors de la solution optimale, il est considéré invalide. Cette méthode permet de poursuivre la recherche et permet de trouver des solutions de meilleurs en meilleurs qui optimisent le temps du trajet.

## 5.5 Expérimentation

L’algorithme de fouille a été mis à l’épreuve à l’aide du logiciel Matlab sur le problème du pendule simple avec comme condition initial la position basse et comme objectif la position haute avec les paramètres suivant :

- $m = 1kg$
- $L = 1m$
- $g = 9.81 \frac{m}{s^2}$

Les paramètres de discrétisation trouvés selon la méthode présentée à la section 5.2 sont les suivants :

- $\delta t = 0.1s$
- $\delta \theta = 0.097rad$
- $\delta \dot{\theta} = 0.18 \frac{rad}{s}$
- Domaine de  $\theta = [-\pi, \pi]rad$
- Domaine de  $\dot{\theta} = [-4.5, 4.5] \frac{rad}{s}$

Un exemple de solution trouvé est présenté à la figure 9. Cette solution a été trouvée avec 14013 retours arrière et a pris 4.5 secondes. On remarque que le nombre de retour arrière est élevé pour ce problème simple. L’heuristique priorisant le couple maximum a donné des résultats le plus rapidement. Il a aussi été possible d’utiliser une discrétisation plus grossière et d’arriver à une solution, par contre des tests

à discrétisation plus fine se sont avérés beaucoup plus long et plus difficile à obtenir. La méthode serait donc inefficace sur des problèmes avec un nombre de degré de liberté plus élevé.

## 6 Optimisation par solution en essaim (Particle Swarm Optimisation (PSO))

L’algorithme d’optimisation par essaims particuliers (PSO) est un algorithme itératif qui imite le comportement des essaims d’oiseaux. Cet algorithme tente de minimiser une fonction objectif non convexe lorsque des algorithmes de descente de gradient ou de méthode de Newton échouent. Une solution candidate se déplace donc dans l’espace de recherche parmi une collection de solution candidates en utilisant l’information individuelle et commune à l’essaim pour ainsi tenter de converger vers un minimum global.

### 6.1 Algorithme

Lors de l’optimisation, les solutions convergent avec l’aide des équations (27) et (28) où  $i$  est la  $i^{eme}$  particule et  $k$  la présente itération nommée génération.

$$\mathbf{v}_i^k = \omega_1 \mathbf{v}_i^{k-1} + \omega_2 \mathbf{R}_1^k (\mathbf{P}_i^k - \mathbf{y}_i^k) + \omega_3 \mathbf{R}_2^k (\mathbf{P}_g^k - \mathbf{y}_i^k) \quad (27)$$

$$\mathbf{y}_i^k = \mathbf{y}_i^{k-1} + \mathbf{v}_i^k. \quad (28)$$

Le nombre de particules compte *population* particules de  $n$  dimensions,  $P_i$  la solution associée à la meilleure valeur objective trouvée par la particule  $x_i$ ,  $fP_i$  la valeur objective pour  $P_i$ ,  $P_g$  la meilleure solution trouvée par toutes les particules à l’instant  $k$ ,  $R_1$  et  $R_2$  deux vecteurs de nombres aléatoires à distribution uniforme entre 1 et 0, et finalement  $\omega_1$ ,  $\omega_2$  et  $\omega_3$  des poids associés aux différentes composantes de la vitesse d’une particule. La vitesse d’une particule est composée de trois termes calculés à chaque génération. La première composante identifiée comme l’inertie de la particule est une tendance de cette dernière à garder la même vitesse. La deuxième composante est une attraction vers la meilleure solution trouvée par cette même particule communément appelée mémoire de la particule. La troisième compo-

sante de la vitesse souvent nommée coopération est l'attraction de la particule  $i$  vers  $P_g$ .

### 6.1.1 redémarrages

Cet algorithme est conçu pour vaincre les problèmes de fonction objectifs non convexes cependant, pour exploiter une solution potentielle les particules devront converger et par conséquent risquer de rester coincé dans un minimum local. De plus, comme on peut le voir à la Figure 10 pour le modèle du pendule simple, la fonction objectif est composée en grande partie de minimums locaux ayant peu de surface dans le plan  $x$ - $y$  d'où la difficulté de trouver la solution optimale. Les redémarrage vont donc augmenter la probabilité d'explorer tout l'espace  $\mathbb{R}^n$  en injectant en quelque sorte de l'énergie dans l'essaim après chaque multiple d'itération nommée *Redemarrages*. Pour un redémarrage les par-

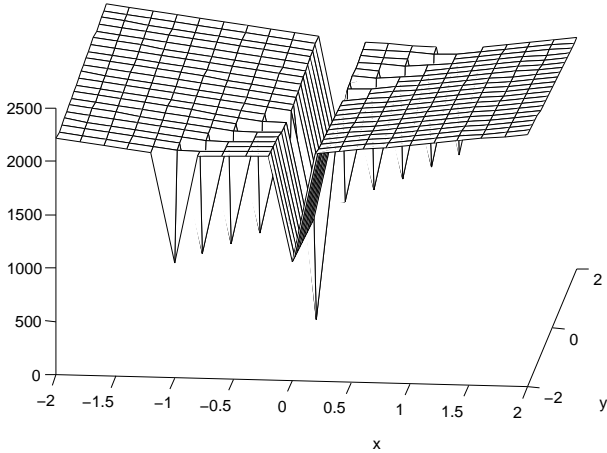


FIG. 10 – Fonction objective pour deux variables

ticules sont réinitialisée aléatoirement,  $\mathbf{P}_i = \mathbf{y}_i^k$  et  $f\mathbf{P}_i$  recalculé en conséquence. Le fait d'oublier en quelque sorte les meilleurs valeurs objectives individuelle empêche de converger vers le même minimum local a nouveau. On note que  $\mathbf{P}_g$  et  $f\mathbf{P}_i$  restent inchangées.

### 6.1.2 Réduction de l'espace de recherche

Lors de l'évolution de l'algorithme de base, aucune information n'est stockée à propos de l'apprentissage de la fonction objectif. Nous allons toutefois apporter une modification qui viendra tirer avantage de l'exploration de l'espace. À chaque itération, on divise donc le domaine de chaque variable en 4 parties et on compte le nombre de particules dans

chaque partie. On ajoute cette valeur a un compteur et lors des redémarrages un pourcentage de l'intervalle entre la valeur minimum et maximum du domaine d'une variable soustraite au domaine de cette dernière. Cette quantité  $\delta$  est exprimée en pourcentage et l'intervalle est  $\Delta_j = (max_j - min_j) \forall j \in n$  où  $min_j$  et  $max_j$  sont les extremums du domaine  $j$ . Dans le but de raffiner une solution l'équation (29) exprime le nombre de génération nécessaire pour que la plus grande valeur de  $\Delta_j \forall j \in n$  soit plus petit ou égal que la valeur *tolerance*.

$$k = \max(\frac{\log(tolerance/\Delta_j)}{\log(1 - \delta)}), \forall i \in n \quad (29)$$

On note cependant que l'algorithme converge généralement avant que cette valeur soit atteinte.

### 6.1.3 Accélération de la convergence

Afin d'accélérer la convergence de l'algorithme la connaissance des état initiaux est utilisée. En effet en supposant que le robot ou le pendule est initialement à l'équilibre, assumer que le robot de bouge pas pour toute la durée de la simulation est une solution faisable qui n'est pas optimale selon la fonction (31), mais cependant ne viole pas la contrainte d'inégalité  $g(\tau_i)$ . Une seule des particules est initialisée avec  $\mathbf{y}_1^0 = \mathbf{0}$ ,  $\mathbf{0}$  étant le vecteur nul de dimension  $n$ . Une deuxième stratégie est d'initialiser un compteur et d'incrémenter ce dernier à chaque itération si la valeur de  $\mathbf{P}_g$  est inchangée et remis a zéro lorsque cette dernière est différente. On utilise ce compteur de manière à arrêter l'algorithme lorsqu'on atteint une valeur *Stall*. Cette stratégie prévient généralement d'effectuer des itération lorsque l'algorithme n'améliore plus la meilleure solution trouvée depuis *Stall* itération.

## 6.2 Modèle

Chaque particule représente une trajectoire divisée en  $n$  intervalles égaux. On obtient la vitesse et l'accélération de chaque trajectoire en utilisant respectivement la dérivée numérique à un et deux pas de temps précédent. L'équation du couple nécessaire à chaque instant  $idt$ ,  $dt$  étant la période entre chaque point de la trajectoire, est :

$$\tau_i = \frac{d^2 \mathbf{y}_i}{dt^2} \quad (30)$$

La fonction objective est donc :

$$F_{objective} = f(\mathbf{y}_i) + \sum_{i=1}^n g(\tau_i) \quad (31)$$

avec

$$f(\mathbf{y}_i) = (\mathbf{y}_i - \boldsymbol{\pi})^T (\mathbf{y}_i - \boldsymbol{\pi})$$

$$g(\tau_i) = \begin{cases} 0 & \text{si } \tau_i \leq 0.7g \\ \sum_{j=1}^n (|\tau_{i,j}| - 0.7g + 1000) & \text{si } \tau_i > 0.7g \end{cases}$$

où  $\boldsymbol{\pi}$  un vecteur de même dimension que  $\mathbf{x}$  de valeurs  $\pi$ .

### 6.3 Expérimentation

On lance l'algorithme avec les paramètres  $Generations = 2e4$   $Stall = 5e3$ ,  $\omega_1 = 0.65$ ,  $\omega_2 = 2$ ,  $\omega_3 = 2.2$ ,  $Redemarages = 2.5e2$ ,  $VitesseMax = 1e-2$  On obtient la solution :

$$\begin{aligned} x = & \begin{bmatrix} 0.155 & 0.464 & 0.927 & 1.545 & 2.179 \\ 2.659 & 2.984 & 3.155 & 3.172 & 3.141 \\ 3.142 & 3.141 & 3.142 & 3.142 & 3.142 \\ 3.141 & 3.141 & 3.141 & 3.142 & 3.142 \end{bmatrix} \end{aligned} \quad (32)$$

La gravité a été négligée lors de cette optimisa-

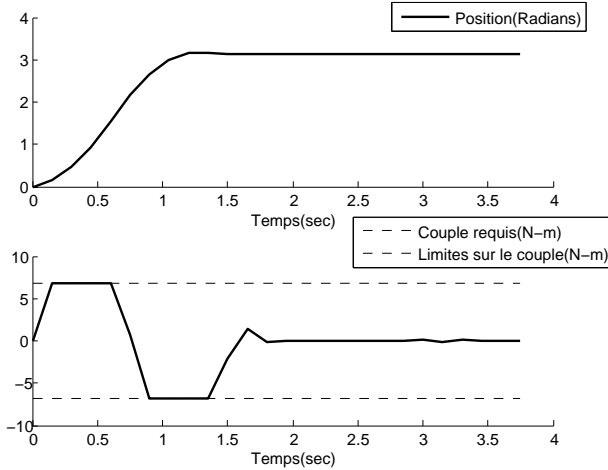


FIG. 11 – Position et couple en fonction du temps suite à l'optimisation.

tion car cette dernière ne convergeait pas. En effet suite aux expérimentations nous concluons que la non linéarité de l'équation dynamique apporte une grande difficulté supplémentaire au problème. Une contrainte d'inégalité étant directement sur les variables serait beaucoup plus facile à gérer que le présent modèle où les variables sont extrêmement couplées dû à  $g(\tau_i)$  faisant intervenir la deuxième dérivée des variables indirectement.

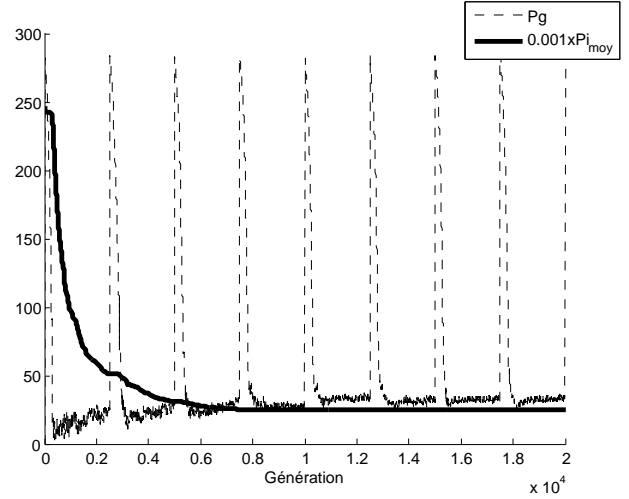


FIG. 12 – Meilleure valeur objective ( $f\mathbf{P}_g$ ) et moyenne des meilleures valeurs objectives individuelles ( $f\mathbf{P}_i$ ) en fonction du temps

## 7 Conclusion

Trois approches très différentes ont été utilisées pour essayer d'améliorer les capacités d'un robot. Chaque approche a ces avantages, mais ont encore des défauts limitant leurs utilisations à des problèmes spécifiques. Le défi de trouver une stratégie générale pour trouver des trajectoires augmentant les capacités d'un robot est encore un problème ouvert. Notre travail présente des avenues de solution intéressantes. Des travaux futurs pourraient explorer la possibilité de combiner ces approches et de créer une stratégie plus robuste permettant de générer des trajectoires pour un plus grand nombre de problèmes.

## Références

- [1] MathWorks, *Constrained Nonlinear Optimization Algorithms*, 2014.