Adaptation de la matrice de covariance pour l'apprentissage par renforcement direct

Olivier Sigaud [1], **Freek Stulp** [2]

1. *Institut des Systèmes Intelligents et de Robotique*
Université Pierre et Marie Curie CNRS UMR 7222
4, place Jussieu 75252 Paris
olivier.sigaud@upmc.fr

2. *Cognitive Robotics*
École Nationale Supérieure de Techniques Avancées (ENSTA-ParisTech)
32, Boulevard Victor 75015 Paris
FLOWERS Research Team
INRIA Bordeaux Sud-Ouest
351, Cours de la Libération 33405 Talence
freek.stulp@ensta-paristech.fr

RÉSUMÉ. *La résolution de problèmes à états et actions continus par l'optimisation de politiques paramétriques est un sujet d'intérêt récent en apprentissage par renforcement. L'algorithme PI^2 est un exemple de cette approche qui bénéficie de fondements mathématiques solides tirés de la commande stochastique optimale et des outils de la théorie de l'estimation statistique. Dans cet article, nous considérons PI^2 en tant que membre de la famille plus vaste des méthodes qui optimisent une fonction de coût via une moyenne des valeurs des paramètres pondérée par les récompenses. Nous comparons PI^2 à d'autres membres de la même famille – la « méthode de l'entropie croisée » et* CMA-ES [1] *– au niveau conceptuel et en termes de performance. La comparaison débouche sur la dérivation d'un nouvel algorithme que nous appelons PI^2-CMA pour «* Path Integral Policy Improvement with Covariance Matrix Adaptation *». Le principal avantage de PI^2-CMA est qu'il détermine l'amplitude du bruit d'exploration automatiquement. Nous illustrons cet avantage sur un exemple non trivial de robotique simulée.*

ABSTRACT. *There has been a recent focus in reinforcement learning on addressing continuous state and action problems by optimizing parameterized policies. PI^2 is a recent example of this approach. It combines a derivation from first principles of stochastic optimal control with tools from statistical estimation theory. In this paper, we consider PI^2 as a member of the wider family of methods which share the concept of probability-weighted averaging to iteratively update parameters to optimize a cost function. We compare PI^2 to other members of the same family*

1. Covariance Matrix Adaptation – Evolutionary Strategies.

– the 'Cross-Entropy Method' and 'Covariance Matrix Adaptation - Evolutionary Strategy' – at the conceptual level and in terms of performance. The comparison suggests the derivation of a novel algorithm which we call PI^2-CMA for "Path Integral Policy Improvement with Covariance Matrix Adaptation". PI^2-CMA's main advantage is that it determines the magnitude of the exploration noise automatically. We illustrate this advantage on a non-trivial simulated robotics experiment.

MOTS-CLÉS : *adaptation de la matrice de covariance, entropie croisée, amélioration de politiques.*

KEYWORDS: *covariance matrix adaptation, cross-entropy, policy improvement.*

DOI:10.3166/RIA.27.243-263

1. Introduction

La mise au point de méthodes d'apprentissage par renforcement (A/R) passant à l'échelle sur des problèmes à états et actions continus tels que les tâches en robotique est au cœur de plusieurs travaux récents (Kober, Peters, 2011 ; Theodorou *et al.*, 2010 ; Tamosiumaite *et al.*, 2011). La plupart des progrès dans ce domaine proviennent des méthodes de recherche directe dans l'espace des paramètres associés à des contrôleurs [2]. Ces méthodes sont fondées sur des échantillonnages de trajectoires. Elles visent à déterminer les paramètres d'un contrôleur qui minimise une fonction de coût. L'algorithme récent « *Policy Improvement with Path Integrals* » (PI^2) est dérivé des principes mathématiques de la commande optimale stochastique et il surpasse de plus d'un ordre de grandeur les algorithmes d'A/R basés sur une descente de gradient tels que *REINFORCE* (Williams, 1992) et *Natural Actor-Critic* (Peters, Schaal, 2008), à la fois en termes de vitesse de convergence et de qualité de la solution trouvée (Theodorou *et al.*, 2010).

Ces méthodes de recherche directe ont souvent été utilisées combinées avec des « primitives dynamiques de mouvement » (DMP) [3], où le contrôleur paramétrique est représenté par un système dynamique doté d'un attracteur ponctuel g et un terme de forçage [4] qui consiste en un ensemble de fonctions de base multipliées par un vecteur de paramètres $\boldsymbol{\theta}$.

$$\frac{1}{\tau}\ddot{x}_t = \underbrace{\alpha(\beta(x_t - x^g) - \dot{x}_t)}_{\text{feedback controller}} + \underbrace{\Psi\boldsymbol{\theta}}_{\text{open loop controller}} \qquad (1)$$

2. Ces méthodes sont appelées improprement méthodes de recherche directe sur les politiques, mais elles ne produisent pas toujours des politiques : une politique détermine une action unique ou une distribution sur des actions pour tout état, alors qu'un contrôleur quelconque peut proposer des actions variables dans certains états et ne rien proposer dans d'autres.

3. Pour *Dynamic Movement Primitives*.

4. Pour *forcing term*.

En modifiant le vecteur $\boldsymbol{\theta}$, on modifie la forme du mouvement (Ijspeert *et al.*, 2002). Diverses tâches robotiques ont été apprises avec une recherche directe sur les paramètres des DMP, tel que le swing d'un lancer de balle de baseball (Peters, Schaal, 2008), le jet de fléchettes et le tennis de table (Kober, Peters, 2011), verser de l'eau (Tamosiumaite *et al.*, 2011) ou des tâches de manipulation variées (Stulp *et al.*, 2011).

Ce qui distingue PI^2 des autres méthodes d'A/R directes est son recours à la moyenne des valeurs de paramètres pondérées par leur valeur de récompense [5], pour mettre à jour des paramètres, au lieu de réaliser cette mise à jour sur la base de l'estimation d'un gradient. De façon intéressante, l'algorithme CMA-ES, qui est considéré comme l'état de l'art en optimisation « boîte noire », repose aussi sur une méthode RWA. La méthode de l'entropie croisée (CEM), très proche, est basée sur le même principe. Il est frappant que ces algorithmes, bien qu'ils dérivent de principes fondamentaux très différents, aient convergé sur des règles de mise à jour des paramètres quasiment identiques. Dans cet article, qui est une traduction de (Stulp, Sigaud, 2012) à laquelle nous avons ajouté des résultats tirés de (Stulp, 2012), nous explicitons la relation entre PI^2, CEM et CMA-ES. Par ailleurs, nous présentons une étude conceptuelle et empirique des différences et similarités entre PI^2, CEM et CMA-ES. Ces comparaisons débouchent sur un nouvel algorithme, PI^2-CMA, qui est structuré comme PI^2, mais utilise l'adaptation de la matrice de covariance de CEM et CMA-ES. Une contribution pratique de cet article est que nous montrons comment PI^2-CMA détermine automatiquement l'amplitude d'exploration appropriée, qui est le seul paramètre qui n'est pas facile à régler dans PI^2.

La suite de l'article est organisée de la façon suivante. Dans la section suivante, nous présentons les algorithmes CEM, CMA-ES et PI^2. Dans la section 3, nous comparons ces trois algorithmes sur leur façon d'engendrer un bruit d'exploration, leur définition de l'optimalité et leurs règles de mise à jour de la matrice de covariance. Chacune de ces comparaisons est appuyée par des évaluations empiriques sur un bras simulé à 10 degrés de liberté. Nous tirons de ces comparaisons un nouvel algorithme appelé PI^2-CMA dans la section 3.4. Nous évaluons PI^2-CMA sur une tâche robotique simulée dans la section 3.4, nous examinons ses capacités de réadaptation à une tâche changeante dans la section 4, nous discutons les travaux apparentés dans la section 5 avant de conclure dans la section 6.

2. Méthodes

Dans cette section, nous décrivons CEM, CMA-ES et PI^2. Ces trois algorithmes optimisent un vecteur de paramètres $\boldsymbol{\theta}$ par rapport à une fonction de coût J.

5. On parlera dans la suite de « méthode RWA », pour *Reward-Weighted Averaging*.

*2.1. Méthode de l'entropie croisée (*CEM*)*

Etant donnés un vecteur de paramètres $\boldsymbol{\theta}$ à n dimensions et une fonction de coût $J : \mathbb{R}^n \mapsto \mathbb{R}$, la méthode de l'entropie croisée pour l'optimisation cherche un minimum global par les étapes suivantes :

– **échantillonnage** – Tirer K échantillons $\boldsymbol{\theta}_{k=1\ldots K}$ selon une distribution ;

– **tri** – Trier les échantillons par ordre croissant de l'évaluation de leur fonction de coût $J(\boldsymbol{\theta}_k)$;

– **mise à jour** – Recalculer les paramètres de la distribution à partir des K_e meilleurs échantillons ;

– **itération** – Retourner à l'étape 1 avec la nouvelle distribution jusqu'à ce que le coût converge ou jusqu'à un certain nombre d'itérations.

On utilise couramment la distribution gaussienne multi-variée $\mathcal{N}(\boldsymbol{\theta}, \Sigma)$ avec les paramètres $\boldsymbol{\theta}$ (moyenne) et Σ (matrice de covariance), de telle façon que ces trois étapes soient implémentées comme dans (2)-(6) ci-dessous.

Méthode de l'entropie croisée (une itération)

$$\boldsymbol{\theta}_{k=1\ldots K} \sim \mathcal{N}(\boldsymbol{\theta}, \Sigma) \qquad \textbf{échantillonner} \tag{2}$$

$$J_{k=1:K} = J(\boldsymbol{\theta}_{k=1:K}) \qquad \textbf{évaluer} \tag{3}$$

$$\boldsymbol{\theta}_{k=1\ldots K} \leftarrow \texttt{trier}\ \boldsymbol{\theta}_{k=1\ldots K}\ \texttt{selon}\ J_{k=1\ldots K} \qquad \textbf{trier} \tag{4}$$

$$\boldsymbol{\theta}^{new} = \sum_{k=1}^{K_e} \frac{1}{K_e} \boldsymbol{\theta}_k \qquad \textbf{mettre à jour la moyenne} \tag{5}$$

$$\Sigma^{new} = \sum_{k=1}^{K_e} \frac{1}{K_e} (\boldsymbol{\theta}_k - \boldsymbol{\theta}^{new})(\boldsymbol{\theta}_k - \boldsymbol{\theta}^{new})^\intercal \qquad \textbf{mettre à jour la covariance} \tag{6}$$

Un exemple d'itération de CEM apparaît sur la figure 1, avec une distribution gaussienne multi-variée dans un espace de recherche à deux dimensions. Le graphe en haut à droite montre l'espace 2D des paramètres. Le coût d'un échantillon est sa distance à l'origine de l'espace cartésien. La distribution gaussienne multi-variée initiale $\mathcal{N}(\left[\begin{smallmatrix}8\\8\end{smallmatrix}\right], \left[\begin{smallmatrix}9&0\\0&9\end{smallmatrix}\right])$ est représentée par le cercle noir pointillé (intervalle de confiance à 68 %). $K = 10$ échantillons $\boldsymbol{\theta}_k$ sont tirés de cette distribution. Les $K_e = 5$ meilleurs échantillons sont utilisés pour calculer une nouvelle distribution gaussienne, qui est $\mathcal{N}(\left[\begin{smallmatrix}7.2\\6.4\end{smallmatrix}\right], \left[\begin{smallmatrix}0.6&-1.1\\-1.1&5.6\end{smallmatrix}\right])$ dans ce cas. Comme on peut le voir, la moyenne de la distribution s'est rapprochée de l'origine et l'axe principal de la matrice de covariance pointe davantage vers l'origine. La partie en bas à gauche du graphe montre la correspondance entre coût et probabilité, calculée avec (13). Quand $K_e = K$ (points hachurés), on estime simplement la distribution originale.

Dans (6), $\boldsymbol{\theta}$ est la moyenne de la distribution utilisée pour échantillonner dans (2), comme indiqué dans la section 3.1 de (Hansen, 2011). Par conséquent, une estimation non biaisée de la covariance est obtenue en multipliant par $\frac{1}{K_e}$, plutôt que par $\frac{1}{K_e-1}$, car nous savons que la vraie moyenne de la distribution est $\boldsymbol{\theta}$.

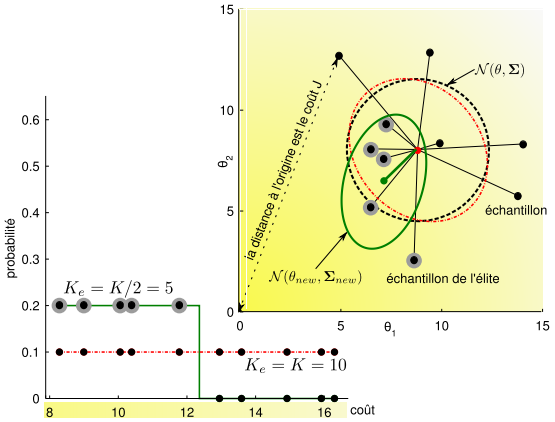

Figure 1. Visualisation d'une mise à jour de CEM

Dans cet article, nous considérons CEM en tant que méthode RWA, où les échantillons conservés se voient attribuer une probabilité $1/K_e$ et les autres une probabilité nulle. Avec ces valeurs de P_k, les équations (2) à (6) peuvent être réécrites conformément à l'algorithme de gauche du tableau **??**. Ici, $Q_{K_e/K}$ dénote le $K_e$ième quantile de la distribution $J_{k=1\ldots K}$. Cette notation est choisie pour sa compacité; elle signifie simplement que, dans le tableau croissant des J_k, P_k est $1/K_e$ si $K \leq K_e$ et 0 sinon, comme dans (5). Les mises à jour de paramètres résultantes sont équivalentes à celles de (5) et (6), mais cette représentation met en évidence la relation avec PI^2.

L'algorithme CEM étant très général, il peut être utilisé dans de nombreux contextes, notamment pour le contrôle de robots. Dans ce cadre, on optimise les paramètres d'un contrôleur en regard d'une fonction de coût liée à la tâche que doit réaliser le robot. Une telle utilisation de CEM pour optimiser une politique ou un contrôleur a par exemple été publiée dans (Mannor *et al.*, 2003 ; Busoniu *et al.*, 2011 ; Kobilarov, 2011 ; Marin *et al.*, 2011).

2.2. L'algorithme CMA-ES

L'algorithme CMA-ES (Hansen, Ostermeier, 2001) est très proche de CEM, mais il utilise une méthode plus sophistiquée pour mettre à jour la matrice de covariance, comme indiqué dans le tableau 1. Les règles utilisent la mémorisation (21) et (23). Le terme μ_P est la « *variance effective selection mass* », avec $\mu_P = 1/\sum_{k=1}^{K_e} P_k^2$. Ce paramètre ainsi que les paramètres utilisateurs c_σ, d_σ, c_Σ et c_1 sont expliqués en détail dans (Hansen, Ostermeier, 2001). L'algorithme CMA-ES complet est obtenu en

Tableau 1. Comparaison de CEM *et PI*^{2}*. Ce pseudo-code représente une itération de l'algorithme, avec une phase d'exploration et une phase de mise à jour des paramètres. Les deux algorithmes itèrent ces deux phases jusqu'à convergence des coûts, ou jusqu'à un certain nombre d'itérations.Les équations (18) et (20) sont utilisées seulement dans PI*^{2}*-CMA (voir section 3.4) et ne font pas partie de PI*^{2}*.*

Méthode de l'entropie croisée		Description	PI^2	
		Phase d'exploration		
	for $k = 1 \ldots K$ do	← boucle sur les essais →	for $k = 1 \ldots K$ do	
(7)	$\quad \boldsymbol{\theta}_k \sim \mathcal{N}(\boldsymbol{\theta}, \Sigma)$	← échantillonne →	$\quad \boldsymbol{\theta}_{k,i=1\ldots N} \sim \mathcal{N}(\boldsymbol{\theta}, \Sigma)$	(8)
		exécute la politique →	$\quad \tau_{k,i=1\ldots N} = \text{executepolicy}(\boldsymbol{\theta}_{k,i=1\ldots N})$	(9)
		Mise à jour des paramètres		
		boucle sur les pas de temps →	for $i = 1 \ldots N$ do	(10)
	for $k = 1 \ldots K$ do	← boucle sur les essais →	$\quad$ for $k = 1 \ldots K$ do	
(11)	$\quad J_k = J(\boldsymbol{\theta}_k)$	← évalue →	$\quad\quad S_{k,i} \equiv S(\tau_{k,i}) = \sum_{j=i}^{N} J(\tau_{j,k})$	(12)
(13)	$\quad P_k = \begin{cases} \frac{1}{K_e} & \text{if } J_k < Q_{K_e/K} \\ 0 & \text{if } J_k > Q_{K_e/K} \end{cases}$	← probabilité →	$\quad\quad P_{k,i} = \frac{e^{-\frac{1}{\lambda} S_{k,i}}}{\sum_{k=1}^{K} [e^{-\frac{1}{\lambda} S_{k,i}}]}$	(14)
(15)	$\boldsymbol{\theta}^{new} = \sum_{k=1}^{K} P_k \boldsymbol{\theta}_k$	← mise à jour des paramètres →	$\quad\quad \boldsymbol{\theta}_i^{new} = \sum_{k=1}^{K} P_{k,i} \boldsymbol{\theta}_k$	(16)
(17)	$\Sigma^{new} = \sum_{k=1}^{K} P_k (\boldsymbol{\theta}_k - \boldsymbol{\theta})(\boldsymbol{\theta}_k - \boldsymbol{\theta})^\intercal$	← adapt. matrice covar. →	$\quad\quad \Sigma_i^{new} = \sum_{k=1}^{K} P_k (\boldsymbol{\theta}_{k,i} - \boldsymbol{\theta})(\boldsymbol{\theta}_{k,i} - \boldsymbol{\theta})^\intercal$	(18)
		moyenne temporelle →	$\boldsymbol{\theta}^{new} = \frac{\sum_{i=0}^{N} (N-i) \boldsymbol{\theta}_i^{new}}{\sum_{i=0}^{N} (N-i)}$	(19)
		moyenne temporelle →	$\Sigma^{new} = \frac{\sum_{i=0}^{N} (N-i) \Sigma_i^{new}}{\sum_{i=0}^{N} (N-i)}$	(20)

remplaçant (17) de CEM dans le tableau **??** par ces quatre équations et en multipliant Σ par σ^2 dans (7).

Il y a quatre différences vis-à-vis de CEM :

• Dans CMA-ES, il n'est pas nécessaire que les probabilités vérifient $P_k = 1/K_e$ comme dans CEM, elles peuvent être choisies par l'utilisateur, pour peu que les contraintes $\sum_{k=1}^{K_e} P_k = 1$ et $P_1 \geq \cdots \geq P_{K_e}$ soient vérifiées. Ici, nous utilisons les probabilités par défaut suggérées par (Hansen, Ostermeier, 2001), soit

$$P_k = \ln(0,5(K+1)) - \ln(k).$$

• L'échantillonnage est effectué selon une distribution $\mathcal{N}(\boldsymbol{\theta}, \sigma^2\Sigma)$, donc la matrice de covariance de la distribution normale est multipliée par un pas de mise à jour. Ce pas de mise à jour détermine l'amplitude (σ) et la forme (Σ) de l'exploration.

• Une mémoire de l'évolution du pas de mise à jour scalaire et de la matrice de covariance est stockée (p_σ et p_Σ respectivement) via les mises à jour précédentes de $\boldsymbol{\theta}$. Cela améliore significativement la vitesse de convergence, parce que cela permet à l'algorithme d'exploiter des corrélations entre des pas de temps consécutifs. Pour une explication plus détaillée de l'algorithme, nous renvoyons à (Hansen, Ostermeier, 2001).

• CEM utilise $\boldsymbol{\theta}^{new}$ pour mettre à jour la matrice de covariance (6), tandis que CMA-ES utilise $\boldsymbol{\theta}$ (24). Les conséquences de cette différence sont discutées sur la figure 2 de (Hansen, 2006).

Tableau 2. Règles de mise à jour du pas (22) et de l'adaptation de la matrice de covariance (24) dans CMA-ES

Adaptation de la matrice de covariance de CMA-ES

$$p_\sigma \leftarrow (1-c_\sigma)\, p_\sigma + \sqrt{c_\sigma(2-c_\sigma)\mu_P \Sigma^{-1}}\, \frac{\boldsymbol{\theta}^{new} - \boldsymbol{\theta}}{\sigma} \tag{21}$$

$$\sigma_{new} = \sigma \times \exp\left(\frac{c_\sigma}{d_\sigma}\left(\frac{\|p_\sigma\|}{E\|\mathcal{N}(0,I)\|} - 1\right)\right) \tag{22}$$

$$p_\Sigma \leftarrow (1-c_\Sigma)\, p_\Sigma + h_\sigma \sqrt{c_\Sigma(2-c_\Sigma)\mu_P}\, \frac{\boldsymbol{\theta}^{new} - \boldsymbol{\theta}}{\sigma} \tag{23}$$

$$\Sigma^{new} = (1 - c_1 - c_\mu)\, \Sigma + c_1 (p_\Sigma p_\Sigma^T + \delta(h_\sigma)\Sigma) + c_\mu \sum_{k=1}^{K_e} P_k (\boldsymbol{\theta}_k - \boldsymbol{\theta})(\boldsymbol{\theta}_k - \boldsymbol{\theta})^\intercal \tag{24}$$

2.3. L'algorithme PI²

Une tendance récente en A/R consiste à utiliser des politiques paramétrées combinées avec le recours à la méthode RWA ; l'algorithme PI² est un exemple de cette approche. Utiliser des politiques paramétrées permet d'éviter la malédiction de la dimensionalité associée aux espaces de grande taille. En outre, utiliser la méthode RWA évite d'avoir à estimer un gradient, ce qui peut être difficile pour des fonctions de coûts bruitées ou discontinues. En fait, (Arnold *et al.*, 2011) montre que la mise à jour réalisée dans CMA-ES correspond à une descente du gradient naturel.

L'algorithme PI² est dérivé de fondements mathématiques issus de la commande stochastique optimale. Il tire son nom de l'application du lemme de Feynman-Kac pour transformer l'équation d'Hamilton-Jacobi-Bellman en intégrale sur un chemin [6], qui peut être approchée par des méthodes de Monte Carlo (Theodorou *et al.*, 2010). L'algorithme PI² est décrit dans la partie droite du tableau **??**. De même que dans CEM, K échantillons $\boldsymbol{\theta}_{k=1\ldots K}$ sont tirés d'une distribution gaussienne. Dans PI², le vecteur $\boldsymbol{\theta}$ représente les paramètres d'un contrôleur qui, quand il est exécuté, produit une trajectoire $\boldsymbol{\tau}_{i=1\ldots N}$ dans l'espace d'état du système en N pas de temps. Cette trajectoire multi-dimensionnelle peut représenter par exemple les angles articulaires d'un bras à n degrés de liberté ou bien la position cartésienne d'un effecteur terminal.

Jusqu'à présent, PI² a été appliqué principalement à des contrôleurs représentés sous la forme de DMP (Ijspeert *et al.*, 2002), où $\boldsymbol{\theta}$ détermine la forme du mouvement. Bien que PI² recherche dans l'espace des paramètres des politiques $\boldsymbol{\theta}$, les coûts sont définis sur les trajectoires $\boldsymbol{\tau}$ engendrées par une DMP au cours du temps. Le coût de la trajectoire est déterminé en évaluant J pour chaque trajectoire, où le coût à venir de la trajectoire au pas de temps i est défini comme la somme sur tous les coûts futurs $S(\boldsymbol{\tau}_{i,k}) = \sum_{j=i}^{N} J(\boldsymbol{\tau}_{j,k})$, comme dans (12) [7].

De même, la mise à jour des paramètres est appliquée à chaque pas de temps i par rapport au coût à venir $S(\boldsymbol{\tau}_i)$. Le poids associé à une trajectoire au pas de temps i est calculé comme l'exponentielle du coût, comme dans (14). Cela attribue un poids élevé aux essais de faible coût et vice versa. En pratique, $-\frac{1}{\lambda} S_{i,k}$ est implémenté avec le paramètre d'élitisme $\frac{-h(S_{i,k}-\min(S_{i,k}))}{\max(S_{i,k})-\min(S_{i,k})}$ cf. la section 3 et (Theodorou *et al.*, 2010).

Comme on peut le voir dans (16), une mise à jour différente des paramètres $\boldsymbol{\theta}_i^{new}$ est calculée pour chaque pas de temps i. Pour se ramener à la mise à jour unique du paramètre $\boldsymbol{\theta}^{new}$, la dernière étape consiste à appliquer la moyenne sur l'ensemble des pas de temps (19). Cette moyenne est pondérée de façon à ce que les mises à jour plus précoces contribuent plus que les mises à jour tardives, donc le poids au pas de temps i est $T_i = (N-i)/\sum_{j=1}^{N}(N-j)$. L'intuition est que les mises à jour plus précoces affectent un horizon plus important, donc elles influencent davantage le coût de la trajectoire.

---

6. *Path integral.*
7. Dans la suite, nous abrégeons $S(\boldsymbol{\tau}_{i,k})$ en $S_{i,k}$.

3. Comparaison de PI^2, CEM et CMA-ES

La comparaison des équations de CEM, CMA-ES et PI^2 montre des similarités et des différences intéressantes. Tous les échantillons tirés d'une gaussienne pour explorer l'espace des paramètres – (7) et (8) – sont identiques et reposent sur la méthode RWA pour mettre à jour les paramètres – (15) et (16). Il est frappant que ces algorithmes qui dérivent de cadres mathématiques très différents aient convergé vers les mêmes principes. Alors que la section 2 s'est plutôt focalisée sur les similitudes, cette section s'intéresse aux différences. Auparavant, nous présentons la tâche d'évaluation utilisée dans cet article.

3.1. Tâche d'évaluation

Nous utilisons une tâche d'évaluation basée sur un bras à 10 degrés de liberté. La tâche est décrite sur la figure 2. Elle est tirée de (Theodorou *et al.*, 2010), où elle est utilisée pour comparer PI^2 à POWER (Kober, Peters, 2011), NAC (Peters, Schaal, 2008), et REINFORCE (Williams, 1992).

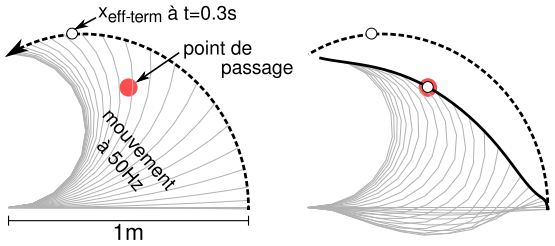

Figure 2. Tâche d'évaluation. La ligne grise représente un bras à 10 degrés de liberté de $1m$ de long, constitué de 10 segments de 0,$1m$. A $t = 0$, le bras est étiré horizontalement. Avant apprentissage (partie gauche), chaque segment fait un mouvement de type minimum-jerk *de 0,5s vers la position finale où le bras "touche" l'axe des ordonnées. La trajectoire (déterministe) de l'effecteur terminal (ligne noire épaisse) ainsi que des instantanés de la posture du bras à 50Hz (lignes grises fines) sont montrés. L'objectif de cette tâche est de faire en sorte que l'effecteur terminal traverse le point (0,5, 0,5) à $t = 0,3s$, tout en minimisant les accélérations. La partie droite montre un exemple de mouvement appris*

L'objectif de cette tâche est exprimé par la fonction de coût :

$$J(\boldsymbol{\tau}_t) = \delta(t - 0,3) \cdot ((x_t - 0,5)^2 + (y_t - 0,5)^2) + \frac{\sum_{d=1}^{D}(D + 1 - d)(\ddot{a}_t)^2}{\sum_{d=1}^{D}(D + 1 - d)} \qquad (25)$$

où δ est une fonction de Dirac, a représente les angles articulaires, x et y les coordonnées de l'effecteur terminal et $D = 10$ le nombre de degrés de liberté. Le poids $(D+1-d)$ pénalise les degrés de liberté plus proches de l'origine pour rendre compte

du fait que les mouvements éloignés de l'origine sont moins coûteux que ceux qui en sont proches, cf. (Theodorou *et al.*, 2010).

Les trajectoires articulaires des 10 segments sont engendrées par une DMP à 10 dimensions, où chaque dimension possède $B = 5$ fonctions de base. Les vecteurs de paramètres $\boldsymbol{\theta}$ (un vecteur 1×5 pour chacune des 10 dimensions) sont initialisés en entraînant la DMP avec un mouvement de type *minimum-jerk*. Le système est déterministe. Cependant, durant l'apprentissage, nous effectuons 10 essais par mise à jour ($K = 10$), où le premier de ces 10 essais est dépourvu de bruit pour des besoins d'évaluation. Pour PI^2, le paramètre d'élitisme est $h = 10$; pour CEM et CMA-ES c'est $K_e = K/2 = 5$. Pour les autres paramètres de CMA-ES, nous avons utilisé les valeurs génériques suggérées dans (Hansen, Ostermeier, 2001, Table 1). Le bruit d'exploration initial est fixé à $\Sigma = 10^4\mathbf{I}_{B=5}$ pour chaque dimension de la DMP.

3.2. Bruit d'exploration

Une première différence entre CEM/CMA-ES et PI^2 réside dans la façon dont le bruit d'exploration est engendré. Dans CEM et CMA-ES, le temps ne joue aucun rôle, donc un seul vecteur d'exploration $\boldsymbol{\theta}_k$ est généré à chaque essai. En commande optimale stochastique, dont PI^2 dérive, $\boldsymbol{\theta}_i$ représente une commande motrice au pas de temps i et la stochasticité $\boldsymbol{\theta}_i + \boldsymbol{\epsilon}_i$ est causée par l'exécution de cette commande dans l'environnement. Quand on applique PI^2 à des DMP, cette stochasticité représente plutôt un bruit contrôlé pour réaliser l'exploration, que l'algorithme échantillonne à partir de $\boldsymbol{\theta}_i \sim \mathcal{N}(\boldsymbol{\theta}, \Sigma)$. Nous qualifions cette exploration de « variant en fonction du temps ». Puisque ce bruit d'exploration est sous notre contrôle, il n'est pas nécessaire de le faire varier à chaque pas de temps. Dans (Theodorou *et al.*, 2010) par exemple, un seul vecteur d'exploration $\boldsymbol{\theta}_k$ est engendré au début d'un essai, et l'exploration n'est appliquée qu'à la fonction de base de la DMP qui a la plus forte activation. Nous appelons cette exploration « par fonction de base ». Dans la version la plus simple, appelée exploration « constante », nous échantillonnons $\boldsymbol{\theta}_{k,i=0}$ une fois au début pour $i = 0$ et nous la gardons inchangée durant l'exécution du mouvement, c'est-à-dire $\boldsymbol{\theta}_{k,i} = \boldsymbol{\theta}_{k,i=0}$.

Les courbes d'apprentissage de ces différentes variantes apparaissent sur la figure 3. On conclut que l'exploration variant en fonction du temps converge beaucoup plus lentement. Comme l'exploration constante produit la convergence la plus rapide, c'est celle que nous utilisons dans la suite de l'article.

3.3. Définition de l'élitisme

La correspondance entre les coûts et les probabilités est différente pour les trois algorithmes. CEM implémente un seuil pour l'élitisme : un échantillon est retenu ($P_k = 1/K_e$) ou pas ($P_k = 0$). PI^2 considère plutôt une valeur d'élitisme continue et inversement proportionnelle au coût d'une trajectoire. CMA-ES utilise une mesure hybride où les échantillons ont une probabilité nulle s'ils ne sont pas dans l'élite et

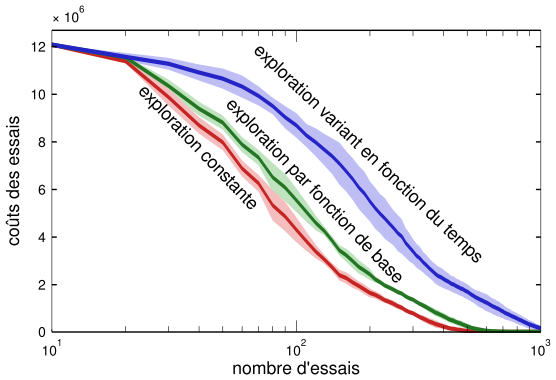

Figure 3. Courbes d'apprentissage pour une exploration variant en fonction du temps, par fonction de base et constante.

une valeur d'élitisme continue et inversement proportionnelle au coût d'une trajectoire s'ils en font partie. Comme indiqué précédemment, les probabilités dans CMA-ES n'ont pas besoin d'être $P_k = 1/K_e$ comme pour CEM, mais elles peuvent être choisies par l'utilisateur tant que les contraintes $\sum_{k=1}^{K_e} P_k = 1$ et $P_1 \geq \cdots \geq P_{K_e}$ sont vérifiées.

Ces différentes correspondances apparaissent sur la figure 4. Une similitude intéressante entre les algorithmes est qu'ils ont tous un paramètre – K_e dans CEM/CMA-ES et h dans PI^2 – qui détermine à quel point la correspondance est élitiste. Des valeurs classiques sont $h = 10$ et $K_e = K/2$ (voir figure 4).

Les courbes d'apprentissage pour les différents schémas de pondération avec leurs différentes approches de l'élitisme apparaissent sur la figure 5. Les courbes d'apprentissage moyennes de la figure 5 sont toutes très similaires sauf pour CEM avec $K_e = 5$ et $K_e = 7$. Cela confirme la conclusion de (Hansen, Ostermeier, 2001) selon laquelle choisir ces poids est « relativement secondaire et peut être effectué dans un large domaine sans perturber l'adaptation » et choisir les poids *optimaux* pour un problème spécifique « ne fait qu'accélérer d'un facteur inférieur à 2 » comparé à la pondération de type CEM où tous les poids sont $P_k = 1/K_e$. Puisque le choix des poids n'est pas critique, nous utilisons dans ce qui suit la pondération de PI^2 avec $h = 10$, la valeur par défaut suggérée par (Theodorou *et al.*, 2010).

3.4. Adaptation de la matrice de covariance

Nous étudions à présent la différence la plus intéressante et pertinente entre les trois algorithmes. Dans CEM/CMA-ES, la moyenne et la covariance de la distribution sont mises à jour, tandis que PI^2 ne met à jour que la moyenne. Cela résulte du fait que, dans PI^2, la forme de la matrice de covariance est contrainte par la relation $\Sigma = \lambda \mathbf{R}^{-1}$, où $\mathbf{R}$ est la matrice (fixée) de coût de contrôle et λ est un paramètre inversement proportionnel à h. Cette contrainte est nécessaire pour la dérivation mathé-

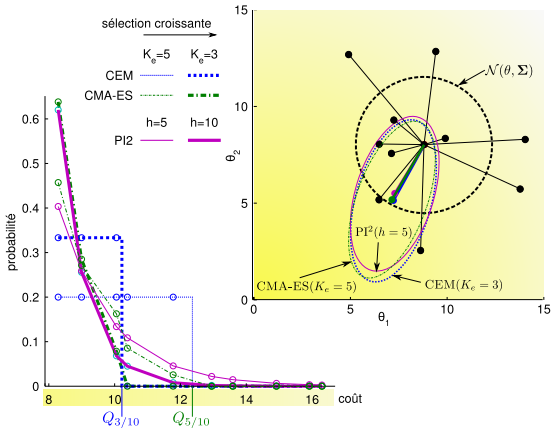

Figure 4. En bas à gauche : comparaison des correspondances entre coûts J_k et probabilités P_k pour PI^2 (avec $h = \{10, 5\}$) et CEM/CMA-ES *(avec $K_e = \{3, 5\}$). En haut à droite : les mises à jour des distributions sont très similaires entre* CEM *($K_e = 3$),* CMA-ES *($K_e = 5$) et PI^2 ($h = 5$). Nous montrons aussi la matrice de covariance utilisée par PI^2, même si PI^2 ne la met pas à jour en pratique, cf. Section 3.4*

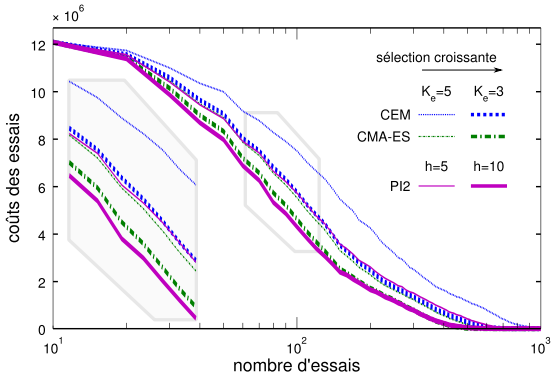

Figure 5. Courbes moyennes d'apprentissage pour les différents schémas de pondération, sur 3 sessions d'apprentissage. Les intervalles de confiance ont été supprimés pour la lisibilité, mais ils sont similaires à ceux de la figure 3. L'encart met en évidence la similitude entre CEM *($K_e = 3$),* CMA-ES *($K_e = 5$) et PI^2 ($h = 5$)*

matique de PI^2 (Theodorou *et al.*, 2010) ; l'intuition sous-jacente est qu'il devrait y avoir moins d'exploration dans les directions où le coût de contrôle est élevé.

Dans cet article, nous choisissons d'ignorer la contrainte $\Sigma = \lambda \mathbf{R}^{-1}$ et d'appliquer la mise à jour de la matrice de covariance dans PI^2 comme cela est réalisé dans CEM et CMA-ES, ce qui constitue la contribution centrale de cet article. Puisque cette mise à jour de la matrice de covariance est calculée à chaque pas de temps i (18), il est nécessaire de calculer une moyenne sur les pas de temps selon (20) comme pour la moyenne $\boldsymbol{\theta}$. Calculer la moyenne des matrices de covariance au cours du temps est possible parce que 1) toute matrice semi-définie positive et symétrique est une matrice de covariance et *vice versa* et 2) une moyenne pondérée de matrices semi-définies positives et symétrique est une matrice semi-définie positive et symétrique (Dattorro, 2011).

Donc, plutôt que d'avoir une matrice de covariance fixée, PI^2 adapte à présent Σ sur la base des coûts observés durant les essais. Ce nouvel algorithme, que nous appelons PI^2-CMA pour « *Path Integral Policy Improvement with Covariance Matrix Adaptation* », est décrit dans le tableau **??** (en faisant abstraction des indices $i = 1 \ldots N$ dans (8) et en incluant les équations (18) et (20)). Un deuxième algorithme, PI^2-CMAES, s'obtient directement en utilisant la mise à jour plus sophistiquée de la matrice de covariance de CMA-ES. L'évaluation qui suit met en évidence le principal avantage de ces algorithmes et compare leur performance.

Sur la figure 6, nous comparons PI^2 (où la matrice de covariance est constante [8]) avec PI^2-CMA (avec une mise à jour de la matrice de covariance de type CEM) et PI^2-CMAES (avec la mise à jour de la matrice de covariance de CMA-ES). Au départ, la matrice de covariance pour chacun des 10 degrés de liberté est fixée à $\Sigma_{init} = \lambda^{\text{init}} \mathbf{I}_5$, où 5 est le nombre de fonctions de base et $\lambda^{\text{init}} = \{10^2, 10^4, 10^6\}$ détermine l'amplitude d'exploration initiale. Toutes les expériences réalisent 200 mises à jour, avec $K = 20$ essais par mise à jour. Nous utilisons un K plus élevé car, à présent, nous ne calculons plus seulement la mise à jour de la moyenne des paramètres (un vecteur 1×5 pour chaque degré de liberté), mais aussi la matrice de covariance (une matrice 5×5), donc l'information requise à chaque essai pour une mise à jour robuste est plus importante (Hansen, Ostermeier, 2001). Après chaque mise à jour, un faible bruit d'exploration est ajouté à la matrice de covariance ($\Sigma_{new} \leftarrow \Sigma_{new} + 10^2 \mathbf{I}_5$) pour éviter une convergence prématurée, comme suggéré dans (Kobilarov, 2011 ; Marin *et al.*, 2011).

Quand les matrices de covariance ne sont pas mises à jour, l'amplitude d'exploration, définie comme la plus grande des valeurs propres de la matrice de covariance, reste constante durant l'apprentissage, c'est-à-dire $\lambda = \lambda^{\text{init}}$ (labels Ⓐ dans la figure 6), et le comportement de la convergence est différent pour différentes amplitudes d'exploration $\lambda^{\text{init}} = \{10^2, 10^4, 10^6\}$. Pour $\lambda^{\text{init}} = 10^4$ nous avons un bon comporte-

8. Attention à la différence entre 1) *exploration* constante comme dans la section 3.2, où le vecteur de paramètres échantillonnés $\boldsymbol{\theta}_k$ ne varie pas au cours du mouvement sur un essai; 2) *matrice de covariance* constante, où Σ n'est pas mise à jour durant toute une session d'apprentissage.

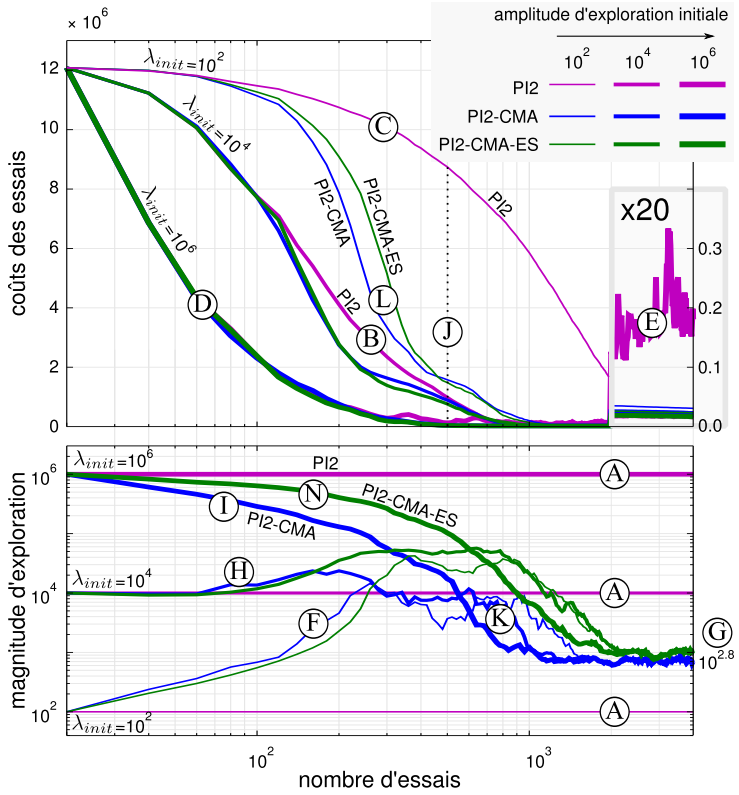

Figure 6. Haut : courbes moyennes d'apprentissage avec et sans mise à jour de la matrice de covariance pour différentes amplitudes de l'exploration initiale, sur 5 sessions d'apprentissage. Bas : évolution de l'amplitude d'exploration λ (définie comme la plus grande des valeurs propres de la matrice de covariance) au cours de l'apprentisssage. Au départ $\Sigma_{init} = \lambda^{init}\mathbf{I}_5$ pour chaque degré de liberté

ment de convergence (B), ce qui n'est pas une coïncidence – cette valeur a été réglée spécifiquement pour cette tâche, c'est la valeur par défaut utilisée jusqu'ici. Cependant, quand nous fixons l'amplitude d'exploration à un niveau très bas ($\lambda^{\text{init}} = 10^2$), la convergence est beaucoup plus lente (C). Quand l'amplitude d'exploration est fixée à un niveau très élevé ($\lambda^{\text{init}} = 10^6$), on obtient une convergence rapide (D). Mais, à cause de la stochasticité importante de l'échantillonnage, on a toujours une variabilité élevée dans le coût après convergence par comparaison avec un λ^{init} plus faible. Cela apparaît dans l'encart, où l'axe des ordonnées a été amplifié d'un facteur 20 (E).

Pour PI^2-CMA, donc *avec* mise à jour de la matrice de covariance, on voit que l'amplitude d'exploration λ change au cours du temps (courbe du bas), où λ est calculé à partir des valeurs propres de la matrice de covariance. Pour $\lambda^{\text{init}} = 10^2$, λ augmente rapidement (F) jusqu'à une valeur maximum, suite à quoi sa valeur diminue et converge vers $10^{2,8}$ (G). Cette valeur est proche de la valeur limite de 10^2 correspon-

dant au bruit ajouté. Cela vaut aussi pour $\lambda^{\text{init}} = 10^4$, mais l'augmentation initiale est moins rapide (H), même si le même maximum est atteint. Pour $\lambda^{\text{init}} = 10^6$, λ diminue seulement (I), mais converge vers $10^{2,8}$ comme les autres.

On peut tirer trois conclusions de ces résultats :

1. avec PI^2-CMA, la vitesse de convergence dépend beaucoup moins de l'amplitude d'exploration initiale λ^{init}: après 500 mises à jour, le coût $\mu \pm \sigma$ pour PI^2-CMA sur tous les λ^{init} est $10^5 \cdot (8 \pm 7)$, alors que pour PI^2 sans mise à jour de la matrice de covariance, il est de $10^5 \cdot (35 \pm 43)$ (J).

2. PI^2-CMA augmente automatiquement λ si davantage d'exploration peut conduire à une convergence plus rapide (F)(H).

3. PI^2-CMA diminue automatiquement λ une fois que la tâche a été apprise (G)(K).

Il faut noter que 2) et 3) sont des propriétés émergentes de la mise à jour de la matrice de covariance. En résumé, PI^2-CMA est capable de régler automatiquement le compromis exploration/exploitation, indépendamment de l'amplitude d'exploration initiale.

C'est une propriété importante, car régler l'amplitude d'exploration manuellement n'est pas facile, le bon réglage étant fortement dépendant de la tâche. Une contribution importante de cet article est d'avoir démontré comment le recours à la méthode RWA pour mettre à jour la matrice de covariance (tel que cela est réalisé dans CEM) permet à PI^2 de régler automatiquement l'amplitude d'exploration, supprimant ainsi la nécessité de régler manuellement ce paramètre. Les seuls paramètres restants dans PI^2 sont K (le nombre d'essais par mise à jour) et h (le paramètre d'élitisme), mais leur réglage n'est pas critique, comme l'ont indiqué indépendamment plusieurs auteurs (Theodorou *et al.*, 2010 ; Tamosiumaite *et al.*, 2011). Bien qu'un Σ initial doive être fourni, la figure 6 montre qu'avec une amplitude d'exploration initiale deux ordres de grandeur plus grande ou plus petite qu'une valeur finement réglée, PI^2-CMA converge toujours vers le même coût et la même amplitude d'exploration, avec seulement une variation mineure de la vitesse initiale de convergence.

En outre, la comparaison entre PI^2-CMA et PI^2-CMAES montre seulement une petite différence sur la vitesse de convergence quand l'exploration initiale est faible, $\lambda^{\text{init}} = 10^2$ (L). Cela provient du fait que la mise à jour de la covariance de CMA-ES est amortie, cf. (22) et (24) ; et les mises à jours sont plus conservatrices que dans CEM, cf. (I) and (N). Dans nos expériences, PI^2-CMAES utilise les paramètres par défaut suggérés par (Hansen, Ostermeier, 2001). Nous avons essayé des paramètres différents pour PI^2-CMAES, la conclusion étant que les meilleurs paramètres sont ceux qui ramènent CMA-ES à CEM, cf. section 2.2. En général, nous ne prétendons pas que PI^2-CMAES surpasse PI^2 et (Hansen, Ostermeier, 2001) conclut lui aussi qu'il y a des tâches pour lesquelles CMA-ES n'est pas meilleur que des algorithmes plus simples.

Enfin, notre comparaison des performances de PI^2-CMAES et PI^2-CMA ne permet pas de conclure. Une question intéressante est de savoir si des fonctions de coûts

typiques de tâches robotiques ont des propriétés qui permettent ou non à CMA-ES de bénéficier des avantages qu'il démontre sur les problèmes d'optimisation standards plus simples.

4. Ré-adaptation à des tâches changeantes

Dans cette section, après avoir étudié les propriétés de PI^2-CMA sur une tâche élémentaire, nous mettons à présent en évidence sa capacité de ré-adaptation sur une tâche dynamique plus complexe. Le problème d'adaptation posé ici consiste à déplacer de 5 cm une balle vis-à-vis de la position à laquelle le système avait appris à la frapper. Nous aurions pu aussi bien modifier la fonction de coût ou la dynamique du système, la capacité de ré-adaptation que nous mettons en évidence est générique.

4.1. Tâche d'évaluation

Le but de cette tâche inspirée de (Peters, Schaal, 2008) est de faire en sorte que le robot utilise une batte pour frapper une balle de telle façon qu'elle atteigne une cible fixée. Nous utilisons l'environnement de simulation SL (Schaal, 2007) pour simuler précisément le robot humanoïde CBi, comme cela apparaît sur la figure 7. Le torse du robot est fixé et on n'utilise que 7 degrés de liberté du bras droit. La batte est fixée à l'effecteur terminal. La fonction de coût de la tâche est :

$$J_{t_i} = 0,01 \sum_{d=1}^{7} \ddot{(a}_{t_i}^d)^2 / N + \phi \tag{26}$$

$$\phi = \begin{cases} 0 & \text{si la cible est atteinte} \\ \text{distance à la cible en } m & \text{sinon} \end{cases} \tag{27}$$

où nous pénalisons l'accélération $a_{t_i}^d$ de la $d^{\text{ième}}$ articulation pour éviter les mouvements à trop forte accélération. Nous divisons la performance par le nombre de pas de temps N afin d'être indépendant de la durée du mouvement. La cible se situe entre -1 et -1.5 m du robot selon l'axe des ordonnées, comme cela apparaît sur la figure 7.

La DMP est de dimension 7 pour contrôler les 7 angles articulaires du bras. Chaque dimension dispose de 3 fonctions de base. On les initialise par une trajectoire de *minimum-jerk* qui dure une seconde, cf. figure 7. Les paramètres de PI^2 sont $K = 10$, $h = 10$. Initialement, $\Sigma_{init} = \lambda^{\text{init}}\mathbf{I}_3$ avec λ^{init}=20 et λ^{min}=0,02.

4.2. Résultats et discussion

La courbe d'apprentissage et le degré d'exploration λ apparaissent au centre de la figure 8. La trajectoire de la balle après 1, 20, 21 et 40 mises à jour est montrée sur l'image du haut et les trajectoires du point terminal de la batte pour les 10 essais d'exploration après 1, 20, 27 et 40 mises à jour est montrée sur l'image du bas.

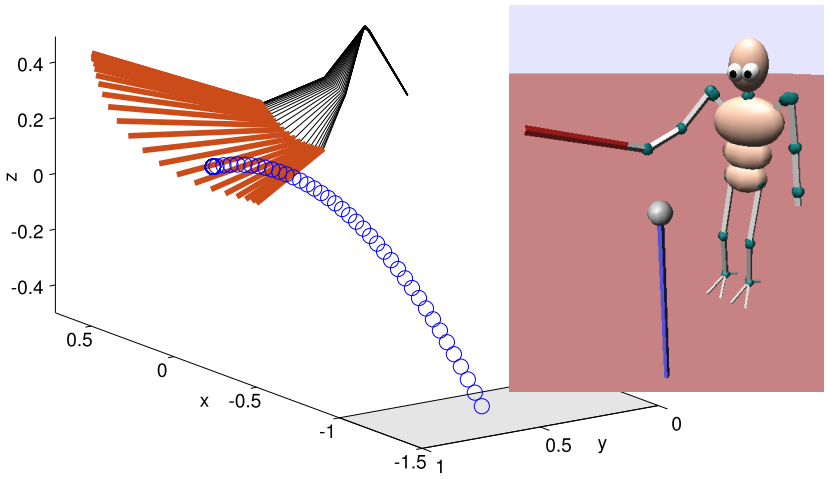

Figure 7. Tâche de T-ball *pour le robot CBi. Les trajectoires de la batte et la balle sont celles apprises après 20 mises à jour*

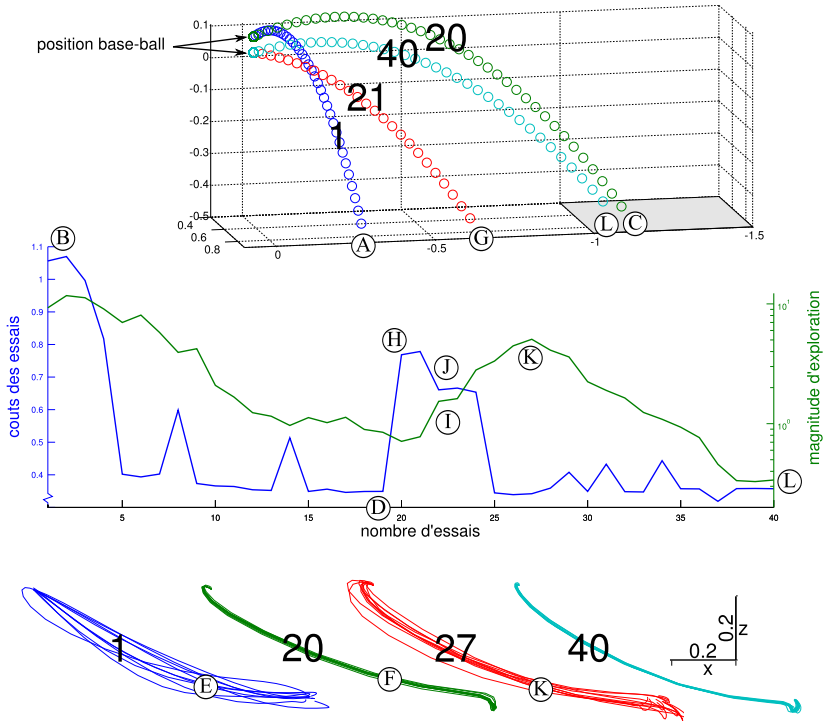

Figure 8. L'image centrale montre le coût d'évaluation et le degré d'exploration au fur et à mesure de l'apprentissage. Les trajectoires de la balle (image du haut) et du point terminal de la batte (image du bas) sont montrées après 1, 20, 21/27 et 40 mises à jour

Au début, la balle atterrit loin de sa cible Ⓐ, ce qui conduit à un coût élevé Ⓑ. Après 20 mises à jour, la balle atterrit dans la cible Ⓒ (c'est le cas après seulement 5 mises à jour), le coût est bien moindre Ⓓ, et l'exploration a considérablement diminué Ⓓ (noter l'échelle logarithmique sur l'axe des ordonnées pour λ). L'exploration moindre devient aussi évidente dans les images du bas, où la variance dans les mouvements de la batte est initialement beaucoup plus élevée Ⓔ qu'après 20 mises à jour Ⓕ.

Après la 20$^{\text{ième}}$ mise à jour, on place la balle 5cm plus bas, ce qui a plusieurs conséquences : la balle n'atterrit plus sur la cible Ⓖ donc le coût augmente immédiatement Ⓗ, après quoi l'exploration augmente à nouveau Ⓘ et les coûts se remettent à diminuer Ⓙ. Après 27 mises à jour, l'exploration atteint un maximum Ⓚ et recommence à baisser. Après 40 mises à jour, les coûts et l'exploration sont tous les deux très faibles Ⓛ. A cause de la pénalité élevée sur les hautes accélérations, qui est recommandée pour les mouvements ballistiques de cette tâche, les coûts ne convergent pas aussi près de 0 que pour l'autre tâche.

En résumé, PI^2-CMA est capable de commuter automatiquement entre des phases d'apprentissage et des phases d'exploitation de ce qu'il a appris.

5. Travaux apparentés

Puisque CEM est un algorithme très général, il est utilisé dans de nombreux contextes. CEM pour l'optimisation de politique a été introduit par (Mannor *et al.*, 2003). Bien qu'ils se soient surtout intéressés à la résolution de petits processus décisionnels de markov (PDM) discrets et finis, les auteurs proposent aussi d'utiliser CEM avec des politiques paramétrées pour résoudre des PDM à grands espaces d'état. De leur côté, (Szita, Lörincz, 2006) utilise CEM pour apprendre la fonction de valeur à Tetris et surpasse des algorithmes d'A/R de plus de deux ordres de grandeur.

(Busoniu *et al.*, 2011) étend ce travail et utilise CEM pour apprendre une politique d'actions discrètes sur des états continus, où les centres et largeurs des fonctions de base sont mis à jour automatiquement. La principale différence avec notre travail est que nous utilisons un espace d'actions continues de grande taille et nous comparons CEM à PI^2 et CMA-ES.

CEM a aussi été utilisé combiné avec de la planification de mouvement basée sur des échantillons (Kobilarov, 2011). Un aspect intéressant de ce travail est qu'il utilise un mélange de gaussiennes au lieu d'une simple distribution pour éviter la convergence prématurée vers un minimum local.

(Marin *et al.*, 2011 ; Marin, Sigaud, 2012b ; 2012a) utilisent CEM pour optimiser le contrôleur d'un bras à six muscles dans un espace à deux dimensions. Le contrôleur est d'abord appris par démonstration avec XCSF, un algorithme de régression qui combine de nombreux modèles linéaires (Butz, Herbort, 2008), ce qui implique l'optimisation par CEM d'un vecteur $\boldsymbol{\theta}$ très grand.

Enfin, (Heidrich-Meisner, Igel, 2008) utilise CMA-ES pour apprendre directement un contrôleur pour un double pendule inversé.

Une excellente discussion de la relation entre stratégies évolutionnistes et A/R est disponible dans (Rückstiess *et al.*, 2010), qui réalise des comparaisons empiriques systématiques entre diverses méthodes à base de gradient issues des deux domaines. Notamment, les auteurs utilisent *Natural Evolution Strategies* (NES), qui est proche de CMA-ES, pour optimiser des contrôleurs pour un pendule inversé, une station debout robuste et une saisie de balle. Les résultats sont comparés avec diverses méthodes de gradient telles que REINFORCE (Williams, 1992) et NAC (Peters, Schaal, 2008). Dans la limite de nos connaissances, notre travail décrit ici est le premier à comparer directement CMA-ES avec CEM et PI^2. De plus, nous utilisons les DMP comme représentation sous-jacente des contrôleurs que nous optimisons, ce qui nous permet de traiter des problèmes de plus grande taille, comme l'a montré (Theodorou *et al.*, 2010) et nous contraint à calculer des moyennes sur les pas de temps, cf. (19) et (20).

PoWeR est un autre algorithme d'amélioration de contrôleurs qui utilise la méthode RWA (Kober, Peters, 2011). Dans PoWeR, les coûts immédiats doivent se comporter comme une probabilité impropre, c'est-à-dire que leur somme doit être constante et toujours positive, mais pas nécessairement égale à 1. En pratique, cela complique la mise au point de la fonction de coût; par exemple, (25) ne peut pas être utilisée dans PoWeR. Dans PI^2, la fonction de coût n'est pas soumise à cette contrainte, elle peut être discontinue. Pour une fonction de coût compatible avec PoWeR et PI^2, les deux algorithmes fonctionnent de façon très similaire (Theodorou *et al.*, 2010).

6. Conclusion

Dans cet article, nous avons reconsidéré l'algorithme PI^2, un algorithme de l'état de l'art pour l'amélioration de contrôleurs, à partir de la perspective des approches qui ont recours à la méthode RWA.

Nous avons discuté les similitudes et les différences entre les algorithmes de cette famille, à savoir PI^2, CMA-ES et CEM. Notamment, nous avons démontré qu'utiliser la méthode RWA pour mettre à jour la matrice de covariance, comme cela est fait dans CEM et CMA-ES, permet à PI^2 de régler automatiquement l'amplitude d'exploration. L'algorithme résultant, PI^2-CMA, converge de façon plus robuste sous des conditions initiales variées et évite à l'utilisateur de régler l'amplitude d'exploration à la main.

Enfin, nous avons appliqué PI^2-CMA à des tâches plus difficiles sur un robot humanoïde, profitant de la capacité de PI^2 à apprendre des tâches complexes et de grande dimension sur des robots réels (Tamosiumaite *et al.*, 2011 ; Stulp *et al.*, 2011).

Remerciements

Ce travail est soutenu par le projet ANR MACSi (ANR 2010 BLAN 0216 01), voir http://macsi.isir.upmc.fr

Bibliographie

Arnold L., Auger A., Hansen N., Ollivier Y. (2011). *Information-geometric optimization algorithms: A unifying picture via invariance principles*. Rapport technique. INRIA Saclay.

Busoniu L., Ernst D., Schutter B. D., Babuska R. (2011). Cross-entropy optimization of control policies with adaptive basis functions. *IEEE Transactions on Systems, Man, and Cybernetics-Part B: Cybernetics*, vol. 41, n° 1, p. 196–209.

Butz M. V., Herbort O. (2008). Context-dependent predictions and cognitive arm control with XCSF. In *Proceedings of the 10th annual conference on genetic and evolutionary computation*, p. 1357-1364. ACM New York, NY, USA.

Dattorro J. (2011). *Convex optimization & euclidean distance geometry*. Meboo Publishing USA.

Hansen N. (2006). The CMA evolution strategy: a comparing review. In J. Lozano, P. Larranaga, I. Inza, E. Bengoetxea (Eds.), *Towards a new evolutionary computation. advances on estimation of distribution algorithms*, p. 75–102. Springer.

Hansen N. (2011, June). *The CMA evolution strategy: A tutorial.* (http://www.lri.fr/$\sim hansen/cmatutorial.pdf$)

Hansen N., Ostermeier A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, vol. 9, n° 2, p. 159-195.

Heidrich-Meisner V., Igel C. (2008). Evolution strategies for direct policy search. In *Proceedings of the 10th international conference on parallel problem solving from nature: Ppsn x*, p. 428–437. Berlin, Heidelberg, Springer-Verlag. Retrieved from http://dx.doi.org/10.1007/978-3-540-87700-4_43

Ijspeert A. J., Nakanishi J., Schaal S. (2002). Movement imitation with nonlinear dynamical systems in humanoid robots. In *Proceedings of the IEEE international conference on robotics and automation (icra).*

Kober J., Peters J. (2011). Policy search for motor primitives in robotics. *Machine Learning*, vol. 84, p. 171-203.

Kobilarov M. (2011, June). Cross-entropy randomized motion planning. In *Proceedings of robotics: Science and systems.* Los Angeles, CA, USA.

Mannor S., Rubinstein R. Y., Gat Y. (2003). The cross-entropy method for fast policy search. In *Proceedings of the 20th international conference on machine learning*, p. 512–519.

Marin D., Decock J., Rigoux L., Sigaud O. (2011). Learning cost-efficient control policies with XCSF: Generalization capabilities and further improvement. In *Proceedings of the 13th annual conference on genetic and evolutionary computation (GECCO'11)*, p. 1235–1242. ACM Press.

Marin D., Sigaud O. (2012a). Reaching optimally over the workspace: a machine learning approach. In *Proceedings IEEE BioRob*, p. 1-6.

Marin D., Sigaud O. (2012b). Towards fast and adaptive optimal control policies for robots: A direct policy search approach. In *Proceedings robotica*, p. 21-26. Guimaraes, Portugal.

Peters J., Schaal S. (2008). Natural actor-critic. *Neurocomputing*, vol. 71, n° 7-9, p. 1180-1190.

Rückstiess T., Sehnke F., Schaul T., Wierstra D., Sun Y., Schmidhuber J. (2010). Exploring parameter space in reinforcement learning. *Paladyn. Journal of Behavioral Robotics*, vol. 1, p. 14-24.

Schaal S. (2007). *The SL simulation and real-time control software package*. Rapport technique. University of Southern California.

Stulp F. (2012). Adaptive exploration for continual reinforcement learning. In *International conference on intelligent robots and systems (IROS)*, p. 1-8.

Stulp F., Sigaud O. (2012). Path integral policy improvement with covariance matrix adaptation. In *Proceedings of the 29th international conference on machine learning (ICML)*, p. 1-8.

Stulp F., Theodorou E., Buchli J., Schaal S. (2011). Learning to grasp under uncertainty. In *Proceedings of the IEEE international conference on robotics and automation (ICRA).*

Szita I., Lörincz A. (2006). Learning tetris using the noisy cross-entropy method. *Neural Comput.*, vol. 18, n° 12, p. 2936–2941. Retrieved from http://dx.doi.org/10.1162/neco.2006.18.12.2936

Tamosiumaite M., Nemec B., Ude A., Wörgötter F. (2011). Learning to pour with a robot arm combining goal and shape learning for dynamic movement primitives. *Robots and Autonomous Systems*, vol. 59, n° 11, p. 910-922.

Theodorou E., Buchli J., Schaal S. (2010). A generalized path integral control approach to reinforcement learning. *Journal of Machine Learning Research*, vol. 11, p. 3137-3181.

Williams R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, vol. 8, p. 229–256.