

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

Freek Stulp^{1,2}, Olivier Sigaud³

¹ Cognitive Robotics
École Nationale Supérieure de Techniques Avancées (ENSTA-ParisTech)
32, Boulevard Victor
75015 Paris

² FLOWERS Research Team
INRIA Bordeaux Sud-Ouest
351, Cours de la Libération
33405 Talence

³ Institut des Systèmes Intelligents et de Robotique
Université Pierre Marie Curie CNRS UMR 7222
4, place Jussieu
75252 Paris

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 partagent le concept de moyenne pondérée par les probabilités pour mettre à jour itérativement des paramètres afin d'optimiser une fonction de coût. Nous comparons PI^2 à d'autres membres de la même famille – la « méthode d'entropie croisée » et CMA-ES¹ – 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.

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; Tamosiunaite *et al.*, 2011). La plupart des progrès dans ce domaine proviennent des méthodes de recherche directe sur les politiques, fondées sur des échantillonnages de trajectoires. Ces méthodes visent à déterminer les paramètres d'une politique 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).

Les méthodes de recherche directe sur les politiques ont souvent été utilisées combinées avec des « primitives dynamiques de mouvement » (DMP)², où la politique est représentée par un système dynamique doté d'un point attracteur g et un terme de forçage qui consiste en un ensemble de fonctions de base multipliées par un vecteur de paramètres θ qui détermine la forme du mouvement. Diverses tâches robotiques ont été apprises avec une recherche directe sur les politiques et des DMP, tel que le swing du baseball (Peters &

1. Covariance Matrix Adaptation – Evolutionary Strategies

2. pour *Dynamic Movement Primitives*

Schaal, 2008), le jet de fléchettes et le tennis de table (Kober & Peters, 2011), verser de l'eau (Tamosiunaite *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 pondérée par les probabilités 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 la moyenne pondérée par les probabilités. La méthode d'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. Pour autant que nous sachions, cet article est le premier à expliciter la relation entre ces trois algorithmes. Notre contribution repose sur 1) une ré-interprétation de CEM en tant que méthode reposant sur la moyenne pondérée par les probabilités ; et 2) la démonstration que CEM est un cas particulier de CMA-ES, obtenu en fixant certains paramètres de CMA-ES à des valeurs extrêmes.

Une contribution supplémentaire de cet article consiste en une étude conceptuelle et empirique des différences et similarités entre PI^2 , CEM et CMA-ES. Ces comparaisons débouchent sur un nouvel algorithme, $\text{PI}^2\text{-CMA}$, 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 $\text{PI}^2\text{-CMA}$ détermine automatiquement l'amplitude d'exploration appropriée, alors qu'il s'agit du 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 prochaine section, nous présentons les algorithmes CEM, CMA-ES et PI^2 . Dans la section 3, nous comparons ces trois algorithmes en fonction de leur façon d'engendrer un bruit d'exploration, de leur définition de l'optimalité et de 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 $\text{PI}^2\text{-CMA}$ dans la section 3.4. Nous discutons les travaux apparentés dans la section 4 et nous concluons dans la section 5.

2 Méthodes

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

2.1 Méthode d'entropie croisée (CEM)

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

- **échantillonne** – Prend K échantillons $\theta_{k=1\dots K}$ d'une distribution ;
- **trie** – Trie les échantillons par ordre croissant de l'évaluation de leur fonction de coût $J(\theta_k)$;
- **met à jour** – Recalcule les paramètres de la distribution à partir des K_e meilleurs échantillons.
- **itère** – Retourne à l'étape 1 avec la nouvelle distribution jusqu'à ce que le coût converge ou jusqu'à un certain nombre d'itérations.

Méthode d'entropie croisée (une itération)

$$\theta_{k=1\dots K} \sim \mathcal{N}(\theta, \Sigma) \quad \text{échantillonne} \quad (1)$$

$$\forall k J_k = J(\theta_k) \quad \text{évalue} \quad (2)$$

$$\theta_{k=1\dots K} \leftarrow \text{sort } \theta_{k=1\dots K} \text{ w.r.t } J_{k=1\dots K} \quad \text{trie} \quad (3)$$

$$\theta^{new} = \sum_{k=1}^{K_e} \frac{1}{K_e} \theta_k \quad \text{met à jour la moyenne} \quad (4)$$

$$\Sigma^{new} = \sum_{k=1}^{K_e} \frac{1}{K_e} (\theta_k - \theta)(\theta_k - \theta)^\top \quad \text{met à jour la covariance} \quad (5)$$

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 (1)-(5). 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. Dans (5), $\boldsymbol{\theta}$ est la moyenne de la distribution utilisée pour échantillonner dans (1), 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 $\boldsymbol{\theta}$.

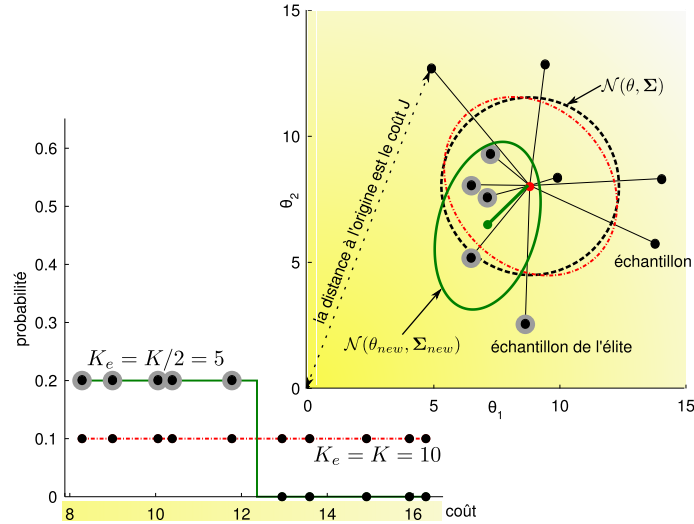


FIGURE 1 – Visualisation d'une mise à jour de CEM. Le graphe en haut à droite montre l'espace 2D des paramètres. Le coût d'un échantillon est sa distance à l'origine de l'espace cartésien. La distribution gaussienne multi-variée initiale est représentée par le cercle noir pointillé (intervalle de confiance à 68%). $K = 10$ échantillons θ_k sont tirés de cette distribution. Les $K_e = 5$ meilleurs échantillons sont utilisés pour calculer une nouvelle distribution gaussienne, qui dans ce cas est $\mathcal{N}(\begin{bmatrix} 7.2 \\ 6.4 \end{bmatrix}, \begin{bmatrix} 3.2 & 1.4 \\ 1.4 & 8.0 \end{bmatrix})$. 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 (12). Quand $K_e = K$ (points rouges achurés), on estime simplement la distribution originale.

Dans cet article, nous considérons CEM en tant que méthode basée sur la moyenne pondérée par les probabilités, où les échantillons conservés se voient attribuer une probabilité $1/K_e$ et les autres une probabilité nulle. Avec ces valeurs de P_k , (1)-(5) peuvent être réécrites conformément à l'algorithme de gauche de la table 1. Ici, $Q_{K_e/K}$ dénote le K_e^{th} quantile de la distribution $J_{k=1\dots K}$. 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 (4). Les mises à jour de paramètres résultantes sont équivalentes à celles de (4) et (5), mais cette représentation met en évidence la relation avec PI².

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 la table 2. Il y a trois 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 $P_1 \geq \dots \geq P_{K_e}$ 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 scalaire σ . Ce pas de mise à jour détermine l'amplitude (σ) et la forme (Σ) de l'exploration.

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

TABLE 1 – Comparaison de CEM et PI². 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 en vert – (17) et (19) – sont utilisées seulement dans PI²-CMA (voir section 3.4) et ne font pas partie de PI².

- 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 θ . 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).

Adaptation de la matrice de covariance de CMA-ES

$$p_\sigma \leftarrow (1 - c_\sigma) p_\sigma + \sqrt{c_\sigma(2 - c_\sigma)\mu_P \Sigma^{-1}} \frac{\theta^{new} - \theta}{\sigma} \quad (20)$$

$$\sigma_{new} = \sigma \times \exp\left(\frac{c_\sigma}{d_\sigma} \left(\frac{\|p_\sigma\|}{E\|\mathcal{N}(0, I)\|} - 1\right)\right) \quad (21)$$

$$p_\Sigma \leftarrow (1 - c_\Sigma) p_\Sigma + h_\sigma \sqrt{c_\Sigma(2 - c_\Sigma)\mu_P} \frac{\theta^{new} - \theta}{\sigma} \quad (22)$$

$$\Sigma^{new} = (1 - c_1 - c_\mu) \Sigma + c_1 (p_\Sigma p_\Sigma^T + \delta(h_\sigma) \Sigma) + c_\mu \sum_{k=1}^{K_e} P_k (\theta_k - \theta)(\theta_k - \theta)^\top \quad (23)$$

TABLE 2 – Règles de mise à jour du pas de mise à jour (21) et de l'adaptation de la matrice de covariance (23) dans CMA-ES, utilisant la mémorisation (20) et (22). μ_P est la « variance effective selection mass », avec $\mu_P = 1 / \sum_{k=1}^{K_e} P_k^2$. 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 remplaçant (16) de CEM dans la table 1 par ces quatre équations et en multipliant Σ par σ^2 dans (6).

De CMA-ES à CEM.

On passe de CMA-ES à CEM en fixant certains paramètres à des valeurs extrêmes : 1) fixer l'horizon temporel $c_\sigma = 0$. Cela réduit (21) à $\sigma_{new} = \sigma \times \exp(0)$, ce qui implique que le pas de mise à jour reste constant au cours du temps. Fixer le pas de mise à jour initial à $\sigma_{init} = 1$ signifie que σ vaut toujours 1, et

qu'il n'a donc aucun effet durant l'échantillonnage. 2) Pour la mise à jour de la matrice de covariance, on fixe $c_1 = 0$ et $c_\mu = 1$. Les deux premiers termes de (23) disparaissent, il reste $\sum_{k=1}^{K_e} P_k(\boldsymbol{\theta}_k - \boldsymbol{\theta})(\boldsymbol{\theta}_k - \boldsymbol{\theta})^\top$, ce qui est équivalent à (16) dans CEM, si P_k est choisi comme dans (12).

2.3 L'algorithme PI²

Une tendance récente en A/R consiste à utiliser des politiques paramétrées combinées avec le recours à la moyenne pondérée par les probabilités ; l'algorithme PI² est un exemple de cette approche. Utiliser des politiques paramétrées permet d'éviter la malédiction de la dimensionalité associée aux espaces d'état-action discrets, et utiliser la moyenne pondérée par les probabilités évite d'avoir à estimer un gradient, ce qui peut être difficile pour des fonctions de coûts bruitées ou discontinues.

L'algorithme PI² est dérivé de fondements mathématiques solides 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 une intégrale sur un chemin³, qui peut être approchée par des méthodes de Monte Carlo (Theodorou *et al.*, 2010). L'algorithme PI² est décrit dans la partie droite de la table 1. De même que CEM, K échantillons $\boldsymbol{\theta}_{k=1\dots K}$ sont tirés d'une distribution gaussienne. Dans PI², le vecteur $\boldsymbol{\theta}$ représente les paramètres d'une politique qui, quand elle est exécutée, produit une trajectoire $\boldsymbol{\tau}_{i=1\dots N}$ en N pas de temps. Cette trajectoire multi-dimensionnelle peut représenter 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 politiques représentées sous la forme de primitives dynamiques de mouvement (Ijspeert *et al.*, 2002), où $\boldsymbol{\theta}$ détermine la forme du mouvement. Bien que PI² recherche dans l'espace des $\boldsymbol{\theta}$, les coûts sont définis en termes de trajectoires $\boldsymbol{\tau}$ engendrées par une DMP quand elle est intégrée 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 (11)⁴.

De même, la mise à jour des paramètres est appliquée à chaque pas de temps i par rapport au coût à venir $S(\boldsymbol{\tau}_i)$. La probabilité d'une trajectoire à i est calculée comme l'exponentielle du coût, comme dans (13). Cela attribue une forte probabilité aux essais de faible coût et vice versa. En pratique, $-\frac{1}{\lambda} S_{i,k}$ est implémenté avec $\frac{-h(S_{i,k} - \min(S_{i,k}))}{\max(S_{i,k}) - \min(S_{i,k})}$ cf. (Theodorou *et al.*, 2010).

Comme on peut le voir dans (15), 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 (18). Cette moyenne est pondérée de façon à ce que les mises à jour plus précoces contribuent plus que les mises à jours 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.

3 Comparaison de PI², CEM et CMA-ES

La comparaison des équations de CEM, CMA-ES et PI² montre des similarités et des différences intéressantes. Tous les échantillons tirés d'une gaussienne pour explorer l'espace des paramètre – (6) et (7) sont identiques – et reposent sur la moyenne pondérée par les probabilités pour mettre à jour les paramètres – (14) et (15). 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 PI² à POWER (Kober & Peters, 2011), NAC (Peters & Schaal, 2008), et REINFORCE (Williams, 1992).

3. path integral

4. Nous abrégeons $S(\boldsymbol{\tau}_{i,k})$ en $S_{i,k}$.

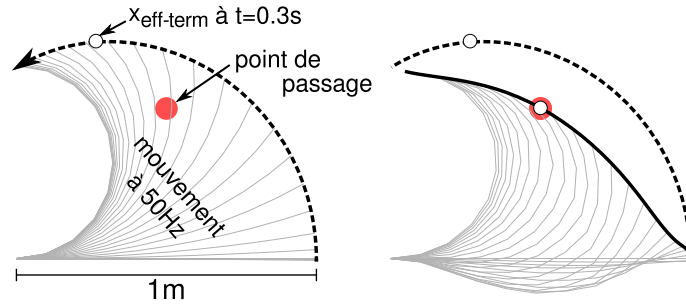


FIGURE 2 – Tâche d'évaluation. La ligne grise représente un bras à 10 degrés de liberté de 1m de long, constitué de 10 segments de 0.1m. A $t = 0$, le bras est étiré horizontalement. Avant apprentissage (partie gauche), chaque segment fait un mouvement de type *minimum-jerk* de 0.5s vers la position finale où le bras "touche" l'axe des y . La trajectoire de l'effecteur terminal (ligne noire épaisse) ainsi que des *snapshots* 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 dans (24), où 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).

$$J(\tau_t) = \delta(t - 0.3) \cdot ((x_t - 0.5)^2 + (y_t - 0.5)^2) + \frac{\sum_{d=1}^D (D + 1 - d)(\ddot{a}_t)^2}{\sum_{d=1}^D (D + 1 - d)} \quad (24)$$

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 θ (un vecteur 1×5 pour chacune des 10 dimensions) sont initialisés en entraînant la DMP avec un mouvement de type *minimum-jerk*. 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'élistisme 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 θ_k est généré à chaque essai. En commande optimale stochastique, dont PI^2 dérive, θ_i représente une commande motrice au pas de temps i et la stochasticité $\theta_i + \epsilon_i$ 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 $\theta_i \sim \mathcal{N}(\theta, \Sigma)$. 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 θ_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 $\theta_{k,i=0}$ une fois au début pour $i = 0$ et nous la gardons inchangée durant l'exécution du mouvement, i.e. $\theta_{k,i} = \theta_{k,i=0}$.

Les courbes d'apprentissage de ces différentes variantes apparaissent sur la FIGURE 3. On conclut que l'exploration variant en fonction du temps convergence 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.

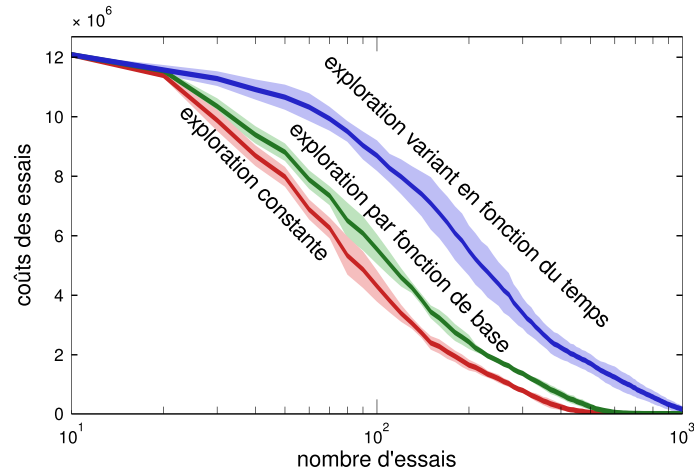


FIGURE 3 – Courbes d'apprentissage pour une exploration variant en fonction du temps, par fonction de base et constante.

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 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 $P_1 \geq \dots \geq P_{K_e}$ 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).

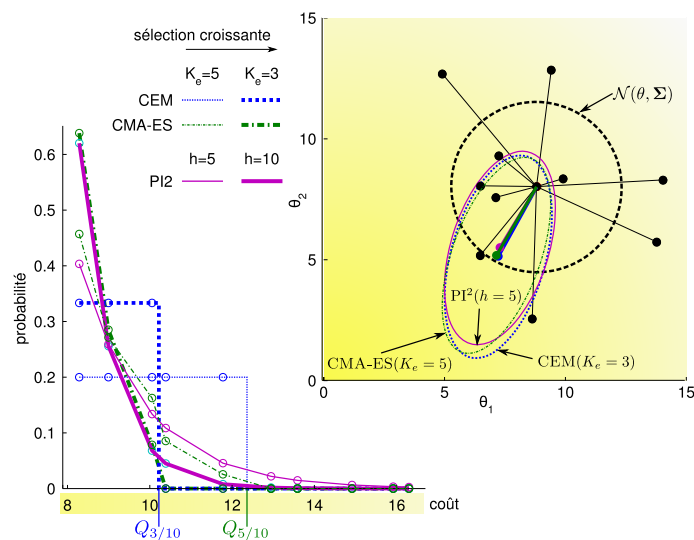


FIGURE 4 – En bas à gauche : comparaison des correspondances entre coûts J_k et probabilités P_k pour PI^2 (avec $h = \{10, 5\}$) et CEM/CMA-ES (avec $K_e = \{3, 5\}$). En haut à droite : les mises à jour des distributions sont très similaires entre CEM ($K_e = 3$), CMA-ES ($K_e = 5$) et PI^2 ($h = 5$).

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).

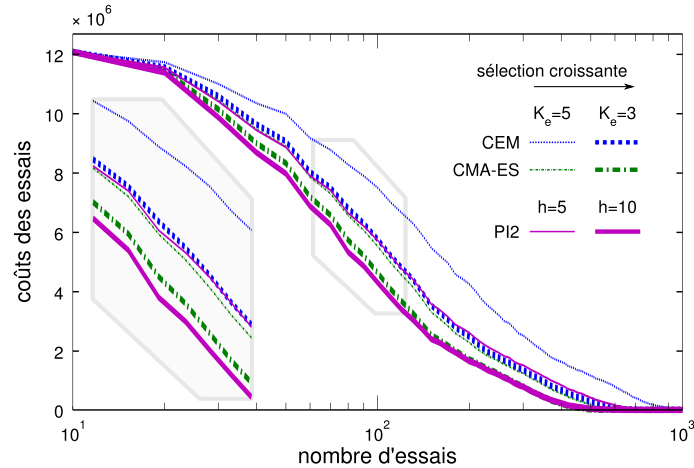


FIGURE 5 – Courbes moyennes d’apprentissage pour les différents schémas de pondération, sur 3 sessions d’apprentissage. Les intervalles de confiance ont été supprimés pour la lisibilité, mais ils sont similaires à ceux de la FIGURE 3. L’encart met en évidence la similitude entre CEM ($K_e = 3$), CMA-ES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

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 mathématique 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 . Puisque cette mise à jour de la matrice de covariance est calculée à chaque pas de temps i (17), il est nécessaire de calculer une moyenne sur les pas de temps (19) exactement comme pour la moyenne θ . Calculer la moyenne des matrices de covariance au cours du temps est possible parce que 1) toute matrice définie semi-positive est une matrice de covariance et vice versa et 2) une moyenne pondérée de matrices définies semi-positives est une matrice définie semi-positive (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, comme il apparaît sur la FIGURE 4. Ce nouvel algorithme, que nous appelons PI^2 -CMA pour « Path Integral Policy Improvement with Matrice De Covariance Adaptation », est décrit dans la table 1 (en faisant abstraction des indices $i = 1 \dots N$ en rouge dans (7) et en incluant les équations (17) et (19) en vert). 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 constante⁵) 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

5. Attention à la différence entre 1) *exploration* constante comme dans la section 3.2, où le vecteur de paramètres échantillonnés θ_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.

est fixé à $\Sigma_{init} = \lambda_{init} \mathbf{I}_5$, où 5 est le nombre de fonctions de base et $\lambda_{init} = \{10^2, 10^4, 10^6\}$ 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).

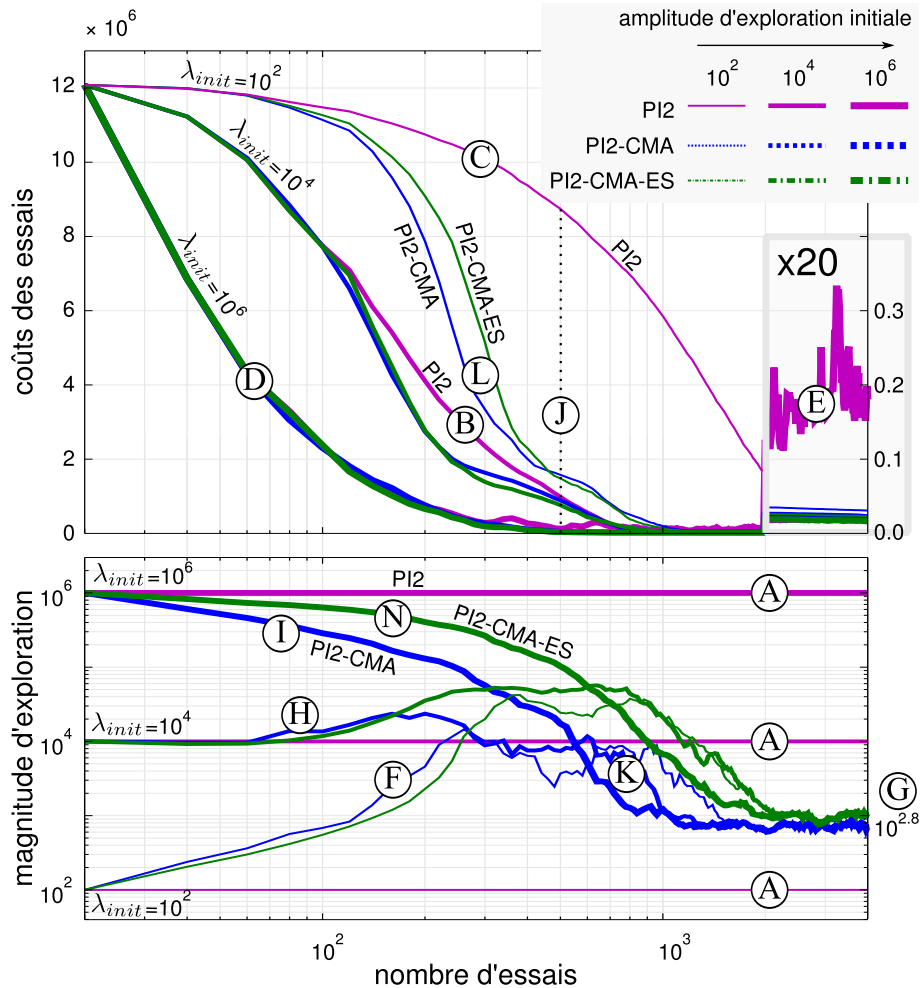


FIGURE 6 – Haut : courbes moyennes d'apprentissage avec et sans mise à jour de la matrice de covariance pour différentes amplitudes de l'exploration initiale, sur 5 sessions d'apprentissage. Bas : évolution de l'amplitude d'exploration λ au cours de l'apprentissage. Au départ $\Sigma_{init} = \lambda_{init} \mathbf{I}_5$ pour chaque degré de liberté.

Quand les matrices de covariance ne sont pas mises à jour, l'amplitude d'exploration reste constante durant l'apprentissage, i.e. $\lambda = \lambda_{init}$ (labels (A) dans FIGURE 6), et le comportement de la convergence est différent pour différentes amplitudes d'exploration $\lambda_{init} = \{10^2, 10^4, 10^6\}$. Pour $\lambda_{init} = 10^4$ nous avons un bon comportement 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 ($\lambda_{init} = 10^2$), la convergence est beaucoup plus lente (C). Quand l'amplitude d'exploration est fixée à un niveau très élevé ($\lambda_{init} = 10^6$), 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 y a été amplifié d'un facteur 20 (E).

Pour PI²-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 $\lambda_{init} = 10^2$, λ augmente rapidement (F) jusqu'à une valeur maximum, suite à quoi il diminue et converge vers $10^{2.8}$ (G). Cela vaut aussi pour $\lambda_{init} = 10^4$, mais l'augmentation initiale est moins rapide (H). Pour $\lambda_{init} = 10^6$, λ diminue seulement (I), mais converge vers $10^{2.8}$ comme les autres.

On peut tirer trois conclusions de ces résultats :

1. avec PI²-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 PI²-CMA sur tous les λ_{init} est $10^5 \cdot (8 \pm 7)$, alors que pour PI² sans mise à jour de la matrice de covariance, il est de $10^5 \cdot (35 \pm 43)$ (J).
2. PI²-CMA augmente automatiquement λ si davantage d'exploration peut conduire à une convergence plus rapide (F)(H).
3. PI²-CMA augmente 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, PI²-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 principale de cet article est que nous démontrons comment le recours à la moyenne pondérée par les probabilités pour mettre à jour la matrice de covariance (tel que cela est réalisé dans CEM) permet à PI² de régler automatiquement l'amplitude d'exploration, supprimant ainsi la nécessité de régler manuellement ce paramètre. Les seuls paramètres restants dans PI² 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 groupes de recherche (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, PI²-CMA converge toujours vers le même coût et la même amplitude d'exploration, avec seulement des différences mineures dans la vitesse initiale de convergence.

La comparaison entre PI²-CMA et PI²-CMAES montre seulement une petite différence sur la vitesse de convergence quand l'exploration initiale est faible $\lambda_{init} = 10^2$ (L). Cela provient du fait que la mise à jour de la covariance de CMA-ES est amortie, cf. (21) et (23); et les mises à jours sont plus conservatrices que dans CEM, cf. (I) and (N). Dans nos expériences, PI²-CMAES utilise les paramètres par défaut suggérés par Hansen & Ostermeier (2001). Nous avons essayé des paramètres différents pour PI²-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 PI²-CMAES surpasse PI² 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.

Par ailleurs, notre comparaison de PI²-CMAES et PI²-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 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 MDP discrets et finis, ils proposent aussi d'utiliser CEM avec des politiques paramétrées pour résoudre des MDP à grands espaces d'état.

Szita & Lőrincz (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 mises 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 PI² et CMA-ES.

CEM a aussi été utilisé combiné avec de la planification de mouvement basées 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 (2012) 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 θ très grand.

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. Rückstieß *et al.* (2010) utilise *Natural Evolution Strategies* (NES), qui est proche de CMA-ES, pour optimiser des politiques 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 article est le premier à comparer directement CMA-ES avec CEM et PI^2 . De plus, nous utilisons les DMP comme représentation sous-jacente des politiques que nous optimisons, ce qui 1) nous permet de traiter des problèmes de plus grande taille, comme l'a montré (Theodorou *et al.*, 2010); 2) nous contraint à calculer des moyennes sur les pas de temps, cf. (18) et (19).

POWER est un autre algorithme récent d'amélioration de politique qui utilise la moyenne pondérée par les probabilités (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, (24) 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).

5 Conclusion

Dans cet article, nous avons reconsidéré l'algorithme PI^2 , un algorithme de l'état de l'art pour l'amélioration de politiques, à partir de la perspective spécifique de la famille des méthodes qui ont recours à la moyenne pondérée par les probabilités.

Nous avons discuté les similitudes et les différences entre les algorithmes de cette famille, à savoir PI^2 , CMA-ES et CEM. En particulier, nous avons démontré qu'utiliser la moyenne pondérée par les probabilités 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.

Nos prochains travaux vont consister à appliquer PI^2 -CMA à des tâches difficiles sur un robot humanoïde, ce qui devrait réussir étant donné la capacité de PI^2 à apprendre des tâches complexes et de grande dimension sur des robots réels (Tamosiunaite *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>

Références

- ADETOLA V. & GUAY M. (2006). Excitation signal design for parameter convergence in adaptive control of linearizable systems. In *Proceedings of the 45th IEEE Conference on Decision and Control*.
- BUSONI 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*, **41**(1), 196–209.
- M. V. BUTZ AND O. HERBORT. (2008). Context-dependent predictions and cognitive arm control with XCSF. In *Proceedings of the 10th annual conference on Genetic and evolutionary computation*, pages 1357–1364. ACM New York, NY, USA,

- CAO C., HOVAKIMYAN N. & WANG J. (2007). Intelligent excitation for adaptive control with unknown parameters in reference input. *IEEE Transactions on Automatic Control*, **52**(8), 1525–1532.
- DATTORRO J. (2011). *Convex Optimization & Euclidean Distance Geometry*. Meboo Publishing USA.
- HANSEN N. (2011). The CMA evolution strategy : A tutorial. <http://www.lri.fr/hansen/cmatutorial.pdf>.
- HANSEN N. & OSTERMEIER A. (2001). Completely derandomized self-adaptation in evolution strategies. *Evolutionary Computation*, **9**(2), 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.
- 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*, **84**, 171–203.
- KOBILAROV M. (2011). 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. (2012). Towards fast and adaptive optimal control policies for robots : A direct policy search approach. In *Proceedings Robotica*, p. 21–26.
- PETERS J. & SCHAAL S. (2008). Natural actor-critic. *Neurocomputing*, **71**(7-9), 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*, **1**, 14–24.
- 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.*, **18**(12), 2936–2941.
- 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*, **59**(11), 910–922.
- THEODOROU E., BUCHLI J. & SCHAAL S. (2010). A generalized path integral control approach to reinforcement learning. *Journal of Machine Learning Research*, **11**, 3137–3181.
- WILLIAMS R. J. (1992). Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning*, **8**, 229–256.
- WITTENMARK B. (2008). Adaptive dual control.