



PROJET DE RECHERCHE LO3IN402

RAPPORT

---

Amélioration et évaluation de la *Cross Entropy Method*

---

*Étudiants :*

Anyes TAFOUGHALT  
Julien JOACHIM  
Robin SOARES

*Encadré par :*  
Olivier SIGAUD

## Résumé

Dans ce projet, nous comparons les performances relatives de la *Cross-Entropy Method* (CEM) et de la *Cross-Entropy Method* inverse (CEMi) lorsqu'elles sont utilisées pour optimiser un réseau neuronal contrôlant un agent d'apprentissage par renforcement. La CEMi diffère de la CEM par son utilisation originale de l'inverse de la matrice de covariance pour la génération de nouveaux poids, et semble mieux fonctionner que la CEM standard dans certains cas spécifiques. Nous réalisons donc une évaluation statistique approfondie des deux méthodes sur des références classiques d'apprentissage par renforcement, et proposons plusieurs variantes efficaces de la CEM et de la CEMi, présentant des améliorations nettes dans des environnements à deux dimensions.

**Mots-clés :** CEM, CEMi, méthode évolutionnaire, apprentissage par renforcement, optimisation de politique.

**Résultats :** Nous avons développé des variantes de la CEM efficaces en deux dimensions, mais avons été limités par des phénomènes d'instabilité numérique lors de leurs applications à des environnements de RL.

En utilisant les directives RLlib pour l'évaluation des performances des algorithmes sur des environnements à deux dimensions, la CEMi fonctionne mieux que la CEM avec tous les environnements testés. Nous avons amélioré la CEMi en redimensionnant les matrices de covariance inverses en fonction des valeurs propres de la matrice d'origine. Nous avons finalement développé une version finale combinant les meilleurs aspects de la CEM et de la CEMi redimensionnée, qui performe mieux dans nos benchmarks que la CEM initiale.

Sur les environnements de RL, la CEMi a de mauvaises performances.

# Table des matières

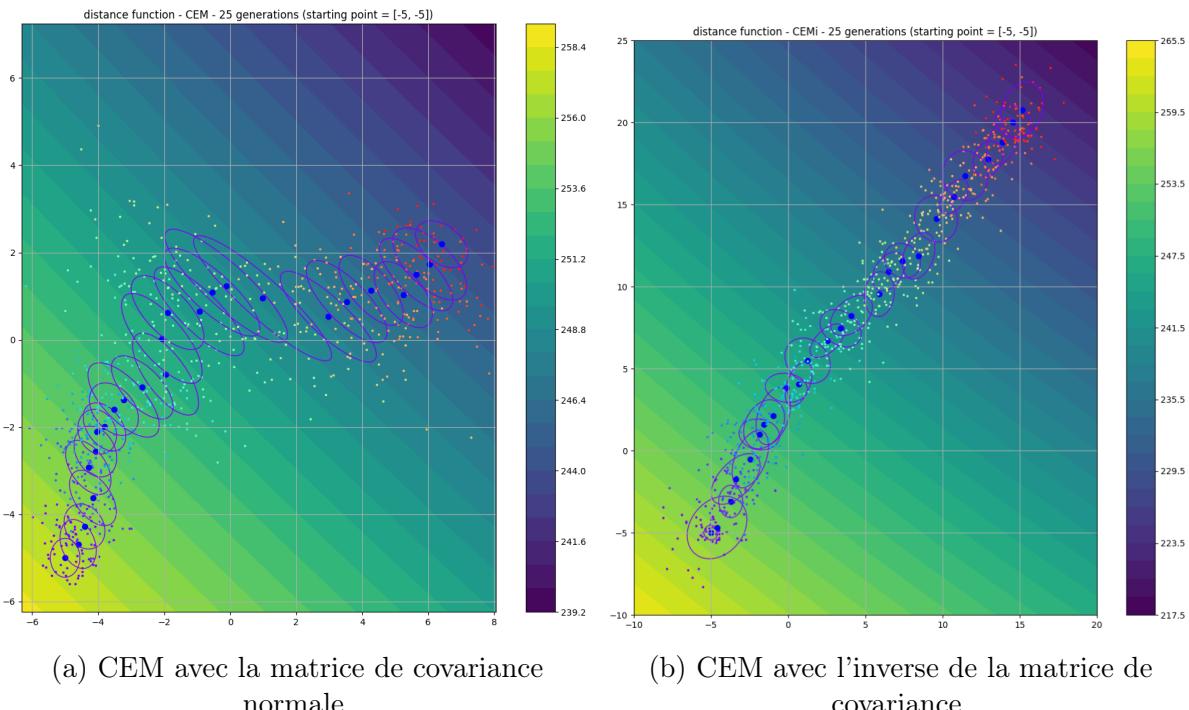
<b>1 Introduction</b>	<b>3</b>
<b>2 Méthodologie</b>	<b>4</b>
2.1 Méthodes du développement sur les environnements 2D . . . . .	4
2.2 Méthodes de comparaisons sur les environnements GYM . . . . .	6
2.3 Méthode d'évaluation statistique des performances . . . . .	6
2.4 Descriptions des algorithmes . . . . .	8
2.4.1 Cross Entropy Method (CEM) . . . . .	8
2.4.2 CEM-inverse (CEMi) . . . . .	8
2.4.3 CEMi - redimensionnée (CEMir) . . . . .	9
2.4.4 CEM+CEMi . . . . .	10
2.4.5 CEM+CEMir . . . . .	10
<b>3 Résultats Obtenus</b>	<b>10</b>
3.1 Étude sur le <i>benchmark</i> en 2D . . . . .	10
3.1.1 Courbe d'efficacité . . . . .	11
3.1.2 Probabilité d'amélioration . . . . .	11
3.1.3 Profil de performance . . . . .	12
3.1.4 Métriques agrégées . . . . .	12
3.1.5 Quelques évolutions de populations pour quelques fonctions . . . . .	13
3.2 Etude sur le benchmark des environnements de RL Gym . . . . .	14
3.2.1 Courbe d'efficacité . . . . .	14
3.2.2 Probabilité d'amélioration . . . . .	14
3.2.3 Profil de performance . . . . .	15
3.2.4 Métriques agrégées . . . . .	15
<b>4 Discussions</b>	<b>15</b>
4.1 Comportements en 2 dimensions . . . . .	16
4.2 Comportement sur les environnements GYM . . . . .	16
<b>5 Conclusions</b>	<b>17</b>
<b>A Problèmes rencontrés et résolutions</b>	<b>18</b>
A.1 Inversion de matrice instable . . . . .	18
A.2 Instabilité de la CEMir lorsqu'appliquée aux environnements Gym . . . . .	18
<b>B Récapitulatif des résultats sur les fonctions 2D</b>	<b>19</b>
<b>C Fonctions de tests d'optimisation utilisées</b>	<b>19</b>
<b>D Environnements et bibliothèques utilisées</b>	<b>20</b>

# 1 Introduction

Dans ce projet, nous explorons une stratégie d'optimisation de la politique d'un agent d'Apprentissage par Renforcement (RL) à partir d'un algorithme évolutionnaire : La *Cross-Entropy Method*.

La *Cross-Entropy Method* (CEM) est une méthode d'optimisation, de type *Monte-Carlo*. C'est une stratégie évolutionnaire avec adaptation de matrice de covariance [1] [2], faisant évoluer une population de points vers un optimum pour une fonction objectif donnée.

La matrice de covariance peut être représentée graphiquement par un ellipsoïde en dimension multiple. Les axes principaux correspondent aux directions propres de la matrice, et leurs longueurs aux valeurs propres associées. C'est ce qu'on a représenté dans la figure 1.



**FIGURE 1 – Exécution de la CEM (sans et avec inversion de la matrice de covariance)** dans le plan pour la fonction objectif égale à la distance au point (150,200). Chaque point représente un vecteur de deux paramètres, le gradient de couleur va du jaune pour un score faible au violet pour un score haut et les ellipses représentées sont associées aux matrices de covariance de chaque génération. Les points bleus représentent les centroïdes des ellipses.

On remarque que dans certains cas simples, la CEM converge en beaucoup moins d'itérations en inversant la matrice de covariance. C'est le cas par exemple des exécutions représentées dans la figure 1.

- Avec la matrice de covariance normale, les ellipses sont majoritairement orientées perpendiculairement à la direction de l'évolution de la population de points.
- Avec l'inverse de la matrice de covariance, les ellipses sont alignées avec la direction de l'évolution et par conséquent la convergence est atteinte plus rapidement.

C'est cette observation qui motive notre étude, où nous comparons la *Cross-Entropy Method* (CEM) et la *Cross-Entropy Method inverse* (CEMi) dans le cadre de l'apprentissage par renforcement.

La *Cross-Entropy Method inverse* est en tout point similaire à la CEM, à la seule différence qu'elle utilise l'inverse de la matrice de covariance pour générer de nouveaux points à la génération suivante.

Ainsi, notre projet a des objectifs multiples. On cherche dans un premier temps à développer un algorithme fonctionnel et efficace implémentant la CEMi, ainsi que plusieurs variantes qui pourraient amener des améliorations. Les performances de cette dernière seront ensuite comparées, dans un second temps, à celles de la CEM standard, au travers de l'évaluation de certains indicateurs de performance classiques de l'Apprentissage par Renforcement. En particulier, on s'attachera à fournir des mesures statistiques aussi exhaustives et rigoureuses que possible pour discuter les résultats que nous obtiendrons.

## 2 Méthodologie

La politique définissant le comportement d'un agent de RL peut être déterminée grâce à un réseau de neurones, dont les paramètres ont été ajustés à l'environnement. Ainsi, lors du processus de détermination des poids attribués à chaque neurone, on est amené à manipuler des vecteurs avec des centaines - voire des milliers de dimensions, pour les problèmes les plus complexes.

Pour cette raison, et afin de pouvoir mieux comprendre les comportements des algorithmes que nous développons, nous faisons d'abord le choix de mener notre études dans des environnements à 2 dimensions, avant de passer à l'échelle. Cette simplification du problème vient avec deux principaux avantages : d'abord, les problèmes à 2 dimensions peuvent être simplement représentés graphiquement, et montrent le comportement des algorithmes de manière claire, ce qui est un avantage évident. Ensuite - et surtout, la réduction de la dimensionnalité du problème réduit aussi le temps de calcul nécessaire à chaque exécution, ce qui accélère grandement le développement.

Ainsi, nous concentrons le principal de notre étude sur l'élaboration d'algorithmes améliorant la CEM standard sur des environnements à deux dimensions (cf. section 3.1), pour ensuite les appliquer à des problèmes concrets de RL - à savoir des environnements GYM (cf. section 3.2).

À chaque étape de notre développement, nous évaluons les performances de nos algorithmes selon les recommandations de l'article [3], grâce à la bibliothèque companion : RLiable. Elle fournit un ensemble d'outils d'évaluation statistique, permettant d'estimer l'incertitude relative aux performances mesurées au travers des différentes exécutions, la probabilité d'amélioration des performances d'un algorithme mis en vis-à-vis d'un autre, ainsi que différentes mesures agrégées représentatives des tendances mesurées : médiane, écart interquartile, écart d'optimalité, etc. (cf. section 3).

### 2.1 Méthodes du développement sur les environnements 2D

Afin d'explorer les pistes d'amélioration de la CEM, notamment à partir de la CEMi, nous avons sélectionné un ensemble de fonctions numériques classiques, sur lesquels nos

différents algorithmes doivent trouver un minimum global. Ces différentes fonctions possèdent en fait des propriétés spéciales qui les rendent particulièrement intéressantes pour les tests de méthodes d'optimisation. On pourrait par exemple citer la fonction de Hölder (Figure 2b) qui comporte une abondance d'optima locaux vers lesquels un algorithme pourrait converger prématurément, ou encore la fonction de Ackley (Figure 2a) qui offre des variations subites de gradient sur lesquelles certaines méthodes pourraient éprouver des difficultés (type méthode de Newton). Pour plus de détails, voir en Annexe C le tableau présentant l'entièreté des fonctions que nous avons choisi pour mener nos tests.

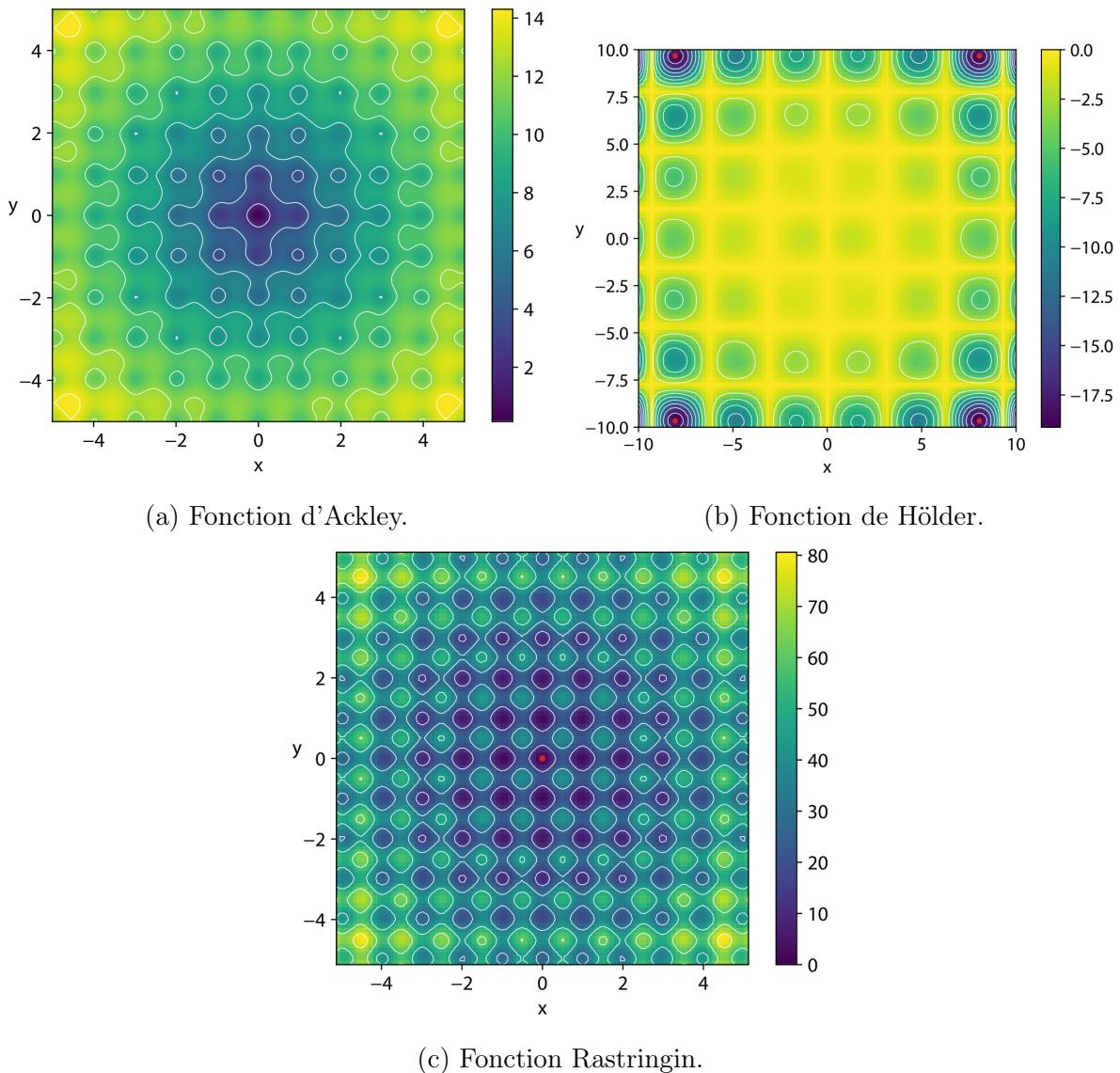


FIGURE 2 – Différentes fonctions de test d'optimisation.

En application, on demande simplement à nos différents algorithmes de faire évoluer une population de vecteurs vers les optimaux respectifs des fonctions en un minimum de générations.

## 2.2 Méthodes de comparaisons sur les environnements GYM

Une fois l'élaboration de nos différents algorithmes terminée, nous les appliquons à un ensemble d'environnements de RL concrets de la librairie GYM. Nous avons choisi de concentrer notre attention sur un nombre assez restreint d'environnement du fait de nos limitations matérielles en termes de puissance de calcul. Ainsi, nous utiliserons les environnements classiques : Acrobot-v0, Cartpole-v1 et Pendulum, dont les spécifications techniques sont listées en Annexe (cf. section D).

Dans la pratique, notre problème consiste à déterminer une politique d'action pour un agent RL évoluant dans les environnements cités ci-dessus. Cette politique est en fait une fonction utilisant un réseau de neurones, dont les poids sont déterminés par nos variantes de la CEM.

### Problème rencontré :

Nous n'avons malheureusement pas réussi à exécuter la *Cross-Entropy Method* inverse redimensionnée (CEMir) (cf. section 2.4.3), ainsi que l'ensemble des méthodes mixtes utilisant la CEMir, à cause de phénomènes d'instabilité numérique. Nous décrivons avec plus de précisions le problème dans la section A.2.

## 2.3 Méthode d'évaluation statistique des performances

Tant pour l'étape d'expérimentation en 2D que pour leurs applications sur les environnements de RL, nous avons évalué les performances de nos algorithmes avec la librairie RLiable en générant plusieurs analyses statistiques :

- *Performance profiles* : Elle permet d'évaluer la performance d'un algorithme en RL sur plusieurs tâches.

Lorsqu'un algorithme de RL est évalué sur plusieurs tâches différentes, il peut obtenir des résultats très différents selon la tâche. La "variabilité des performances entre les tâches" fait référence à cette différence dans la performance de l'algorithme d'une tâche à l'autre. Cette variabilité peut être due à des différences dans la complexité des tâches, la nature des données, la structure de l'environnement ou d'autres facteurs pertinents pour chaque tâche. Les *performance profiles* permettent de visualiser cette variabilité et d'évaluer la performance globale de l'algorithme sur l'ensemble des tâches et elles sont plus robustes que les estimations ponctuelles ou les intervalles de confiance. Les *score distributions* sont une forme de *performance profile* qui montre la fraction de runs avec un score supérieur à un certain seuil. Cette approche est plus adaptée pour les cas où il y a peu de runs. Les *average-score distributions* sont une autre forme de performance profile qui montre la fraction de tâches sur lesquelles un algorithme performe mieux qu'un certain score, mais elles sont biaisées et moins robustes que les *score distributions*, car elles ne prennent en compte que la moyenne des scores pour chaque tâches, et pas la distribution complète.

- Métriques agrégées : *Performance profiles* est un moyen de comparer rapidement différentes méthodes. Si une courbe est strictement au-dessus d'une autre, on dit que la meilleure méthode domine stochastiquement l'autre. Cependant, dans les tests d'apprentissage par renforcement avec un grand nombre de tâches, la domination stochastique est rarement observée et les courbes de performance peuvent

se croiser à plusieurs points. Ainsi, des métriques agrégées sont nécessaires pour des comparaisons quantitatives plus précises.

Des métriques agrégées peuvent être extraites des distributions de scores, telles que la médiane et la moyenne normalisée. Cependant, ces métriques sont insuffisantes pour fournir une indication précise de la performance globale d'un algorithme. Par exemple si un algorithme a un score très bas sur de nombreuses tâches, mais un score très élevé sur une seule tâche, sa médiane peut encore être relativement élevée et masquer sa faible performance globale. De même, la moyenne normalisée peut être influencée par des scores extrêmes, ce qui peut ne pas être représentatif de la performance globale de l'algorithme.

C'est pourquoi l'article recommande des métriques alternatives telles que l'Inter-Quartile Moyen (IQM) et l'*Optimality Gap* pour fournir une indication plus précise de la performance globale de l'algorithme. Ces métriques sont plus robustes et moins biaisées que la médiane et la moyenne normalisée.

- *IQM* : elle est calculée en excluant les 25% de scores les plus bas et les 25% de scores les plus élevés, puis en calculant la moyenne des scores restants. Elle est plus robuste que la médiane et plus efficace statistiquement.
- *Optimality Gap* : C'est une métrique qui mesure la différence entre la performance de l'algorithme et une performance idéale considérée comme optimale (elle mesure de combien l'algorithme ne parvient pas à atteindre un score de référence dit optimale). Par exemple, supposons qu'un algorithme obtienne un score moyen de 0,8 sur un ensemble de tâches, et que le score de référence soit de 1,0. Dans ce cas, l'écart d'optimalité est de 0,2 (1,0 - 0,8), ce qui signifie que l'algorithme est à 0,2 en dessous de la performance idéale. Alors, si un algorithme a un écart d'optimalité élevé, cela signifie qu'il ne parvient pas à atteindre des performances proches de l'optimal sur l'ensemble des tâches. Cette mesure est moins sensible aux valeurs aberrantes que la moyenne car elle ne prend en compte que la différence entre la solution trouvée et la solution de référence, plutôt que toutes les valeurs individuelles des solutions. Elle permet également de comparer facilement les performances de différents algorithmes en termes de proximité de l'objectif de référence, car elle normalise les écarts par rapport à l'optimum.

En somme, l'écart d'optimalité est une mesure qui permet de comparer les performances de différents algorithmes de manière équitable en prenant en compte la distance entre les solutions trouvées et les solutions optimales possibles, tout en étant plus robuste aux valeurs aberrantes que la moyenne.

- *Probability of improvement* : Si nous voulons savoir à quel point l'algorithme X est meilleur que Y, nous pouvons utiliser une troisième métrique appelée *probability of improvement*. Cette métrique mesure la probabilité que X soit meilleur que Y sur une tâche sélectionnée au hasard. Elle est calculée comme suit :

$$P(X > Y) = \frac{1}{M} \sum_{m=1}^M P(X_m > Y_m) \quad (1)$$

Où  $M$  est le nombre total de tâches,  $X_m$  est la performance de l'algorithme X sur la tâche  $m$ ,  $Y_m$  est la performance de l'algorithme Y sur la tâche  $m$ , et  $P(X_m > Y_m)$  est la probabilité que X soit meilleur que Y sur la tâche  $m$ .

Cette métrique ne tient pas compte de l'ampleur de l'amélioration. Cela signifie

qu'elle ne mesure que la probabilité que X soit meilleur que Y, mais pas la taille de l'amélioration. Cependant, cette métrique est utile pour comparer la performance relative de deux algorithmes et elle nécessite moins d'essais pour obtenir une estimation fiable de la performance que d'autres métriques.

## 2.4 Descriptions des algorithmes

### 2.4.1 Cross Entropy Method (CEM)

Ci-dessous, une description en pseudo-code de l'algorithme de la CEM.

---

#### **Algorithme 1 : Cross-Entropy Method**

---

**Entrée** : V : tableau de vecteurs de  $\mathbb{R}^n$  représentant une population initiale de points, K : le nombre de générations de l'algorithme,  $\mu$  : le nombre d'individus élites à sélectionner,  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  : une fonction objectif ;

**Sortie** : W : tableau de vecteurs de  $\mathbb{R}^n$  représentant des points qui maximisent  $f$  ;

**Début**

1	Population $\leftarrow$ V liste de tableaux de dimension n ; Centroid $\leftarrow$ tableau de dimension n non initialisé ; <b>pour</b> $i$ de 1 à $K$ <b>faire</b>
2	Trier Population en fonction du score de chaque point individuel ( $f(v)$ pour $v$ dans V) ;
3	Elites $\leftarrow$ Sélectionner les $\mu$ meilleurs individus de la population ;
4	Centroid $\leftarrow$ barycentre des Elites ;
5	CovMatrix $\leftarrow$ calculer la matrice de covariance des Elites ; Population $\leftarrow$ échantillonner une nouvelle population en utilisant une distribution normale multivariée centrée sur Centroid, et dispersée par CovMatrix ;

**return** Population ;

---

### 2.4.2 CEM-inverse (CEMi)

Comme présenté sommairement dans l'Introduction, on remarque en traçant les ellipsoïdes de dispersion que l'axe principal de génération est majoritairement placé perpendiculairement à la direction de progression vers l'optimum. On peut alors vouloir le faire basculer pour avoir des ellipsoïdes parallèles à la progression. Ceci est accompli en inversant la matrice de covariance :

On remplace la ligne 4 et 5 de l'algorithme de la CEM par :

---

**Algorithme 2 : CEMi**


---

CovMatrixInverse  $\leftarrow$  calculer l'inverse de la matrice de covariance des Elites à l'aide de la factorisation de Cholesky ;  
 Population  $\leftarrow$  échantillonner une nouvelle population en utilisant une distribution normale multivariée centrée sur Centroid, et dispersée par CovMatrixInverse ;

---

#### 2.4.3 CEMi - redimensionnée (CEMir)

Les expérimentations (cf. section 3.1) montrent qu'il y a des écarts de performances entre la CEM et la CEMi, qui peuvent motiver une nouvelle version de la CEM.

Il faut remarquer qu'inverser la matrice de covariance réduit considérablement la taille de l'ellipsoïde qui lui est associée. En effet, pour toute matrice  $M$ , on a

$$\forall \lambda \in \text{Sp}(M), \forall v \in E_\lambda, Mv = \lambda v \implies M^{-1}v = \frac{1}{\lambda}v$$

avec :

$\text{Sp}(M)$  : l'ensemble des valeurs propres de la matrice  $M$ .

$E_\lambda$  : le sous-espace propre associé à la valeur propre  $\lambda$ .

$M^{-1}$  : l'inverse de la matrice  $M$ .

Ainsi, si  $M$  est diagonalisable - ce qui est le cas pour une matrice de covariance,  $M^{-1}$  a exactement les mêmes vecteurs propres que  $M$ , mais avec des valeurs propres respectivement inversées.

Dans notre cas donc, les ellipsoïdes des matrices de covariances inverses sont en général bien plus petits (car la longueur et la direction de leurs axes principaux correspondent à ceux des vecteurs propres de la matrice associée).

Alors, bien que l'évolution de la population se fasse bien dans la direction de la dispersion, il se peut que les pas effectués vers l'optimum de la fonction de score soient trop petits pour que la convergence soit rapide.

La CEMi-redimensionnée (CEMir) mitige le rapetissement des ellipsoides inverses en les redimensionnant à la taille des ellipsoides de la matrice de covariance originale. Ci-dessous une description de l'algorithme :

On remplace la ligne 4 et 5 de l'algorithme de la CEM par :

---

**Algorithme 3 : CEMir**


---

CovMatrix  $\leftarrow$  calculer la matrice de covariance des Elites ;  
 CovMatrixInverse  $\leftarrow$  calculer l'inverse de la matrice de covariance des Elites à l'aide de la factorisation de Cholesky ;  

$$\lambda \leftarrow \frac{\max \text{Sp}(\text{CovMatrix})}{\max \text{Sp}(\text{CovMatrixInverse})}$$
  
 CovMatrixInverse  $\leftarrow$  CovMatrixInverse  $\times \lambda$   
 Population  $\leftarrow$  échantillonner une nouvelle population en utilisant une distribution normale multivariée centrée sur Centroid, et dispersée par CovMatrixInverse ;

---

#### 2.4.4 CEM+CEMi

La CEM et la CEMi étant des processus stochastiques, il se peut que la progression vers l'optimum recherché soit perturbée à cause d'une génération malheureuse d'individus de mauvaise qualité. On peut alors vouloir tirer parti de l'orthogonalité des ellipsoïdes de la CEM et de la CEMi, en espérant qu'une génération de mauvaise qualité selon une direction de l'espace soit rattrapée par une autre génération dans la direction orthogonale. C'est ce qui motive cette variante :

On remplace la ligne 4 et 5 de l'algorithme de la CEM par :

---

##### **Algorithme 4 : CEM + CEMi**

---

```
CovMatrixCEM ← calculer la matrice de covariance des Elites ;
CovMatrixCEMi ← calculer la matrice de covariance des Elites, puis
    l'inverser à l'aide de la méthode de Cholesky ;
PopulationCEM, PopulationCEMi ← échantillonner deux nouvelles
    populations en utilisant une distribution normale multivariée centrées
    sur Centroid, et dispersées respectivement par CovMatrixCEM et
    CovMatrixCEMi ;
Population ← fusionner PopulationCEM et PopulationCEMi ;
```

---

#### 2.4.5 CEM+CEMir

On construit la variante CEM+CEMir sur le même modèle que précédemment pour les mêmes justifications.

### 3 Résultats Obtenus

Ci après ; nous présentons nos résultats obtenus avec RLiable (cf. section 2.3), sur les environnements 2D ainsi que sur les environnements de RL Gym.

#### 3.1 Étude sur le *benchmark* en 2D

Nos résultats sur les environnements 2D sont basés sur des fonctions d'optimisations représentées dans le tableau en annexe (cf. section C).

### 3.1.1 Courbe d'efficacité

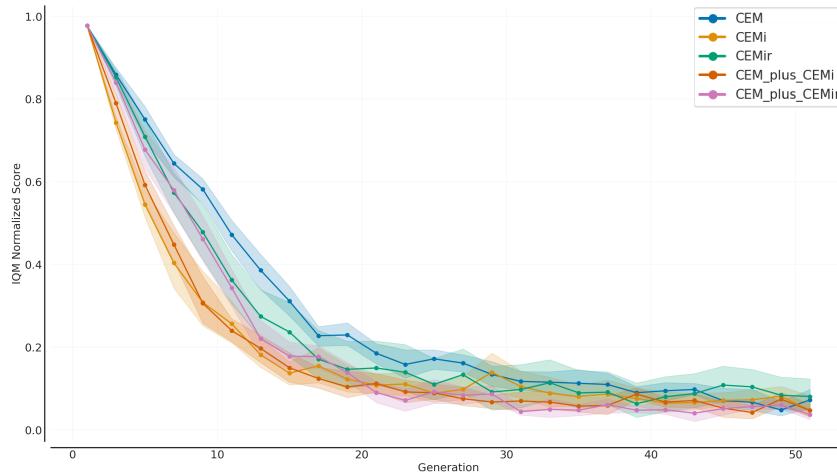


FIGURE 3 – Courbe d'efficacité - Environnements 2D.

La figure 3 représente la moyenne des écarts inter-quartiles des scores obtenus pour toutes les fonctions utilisées ; et ce à chaque génération. On a normalisé les scores spécifiques à chaque fonction afin de pouvoir avoir des données comparables.

Les rectangles de couleurs entourant les courbes représentent l'écart-type associé aux données. C'est un indicateur d'incertitude.

Au plus l'IQM est bas, au plus la performance de la méthode est bonne.

### 3.1.2 Probabilité d'amélioration

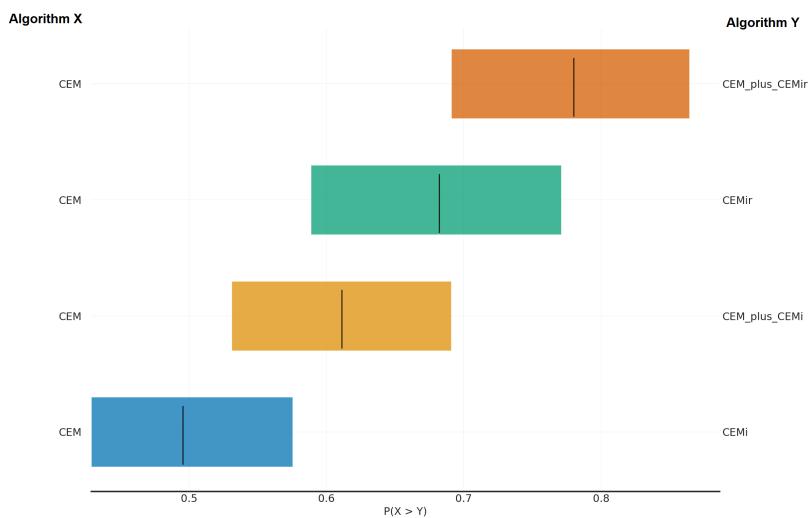


FIGURE 4 – Probabilité d'amélioration - Environnements 2D.

La figure 4 représente la probabilité, pour chaque paires d'algorithmes (X,Y), que les scores de X soient supérieurs à ceux de Y ; c'est-à-dire, la probabilité que les performances de Y soient supérieures à celles de X. Les rectangles de couleurs représentent l'incertitude sur les données à travers l'écart-type.

### 3.1.3 Profil de performance

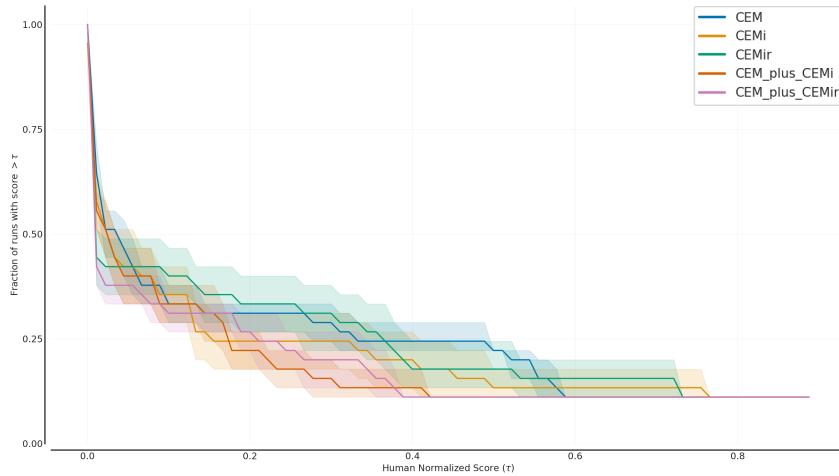


FIGURE 5 – Profil de performance - Environnements 2D.

La figure 5 représente le profil de performance des algorithmes respectifs à travers leurs évaluations. Plus précisément, on représente, sur l’axe des abscisses (les  $x$ ), une valeur score seuil  $\tau$ . En ordonnée, on représente la proportion des scores obtenus qui sont supérieurs au seuil  $\tau$ , parmi tous les scores réalisés. En faisant varier  $\tau$  de 0 à 1, on obtient les courbes représentées pour chaque algorithmes. Au plus la proportion des scores obtenus qui sont supérieurs au seuil  $\tau$  est basse, au plus la performance de la méthode est bonne.

### 3.1.4 Métriques agrégées

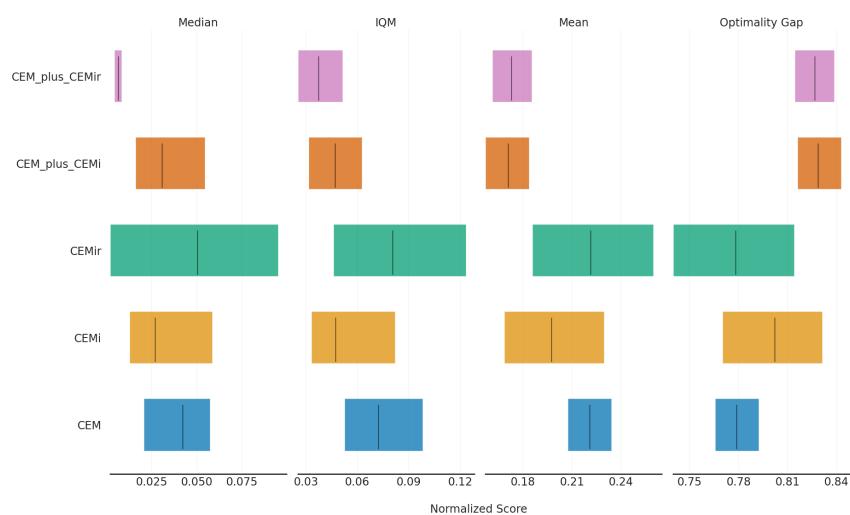


FIGURE 6 – Métriques agrégées - Environnements 2D.

La figure 6 représente respectivement les médianes, moyennes interquartiles, moyennes et écarts d’optimalité relatifs aux scores réalisés par les algorithmes sur toutes le benchmark des fonctions de test. Au plus la médiane, l’IQM ou la moyenne des scores normalisés

est basse, au plus la performance de la méthode est bonne. A l'inverse, au plus l'*optimality gap* est élevé, au plus la performance de la méthode est bonne.

### 3.1.5 Quelques évolutions de populations pour quelques fonctions

Ci-dessous ont été représentées quelques exécutions de certaines de nos méthodes sur les fonctions *Mishrabird*, *Hölder* et *Sphère* (Figure 7).

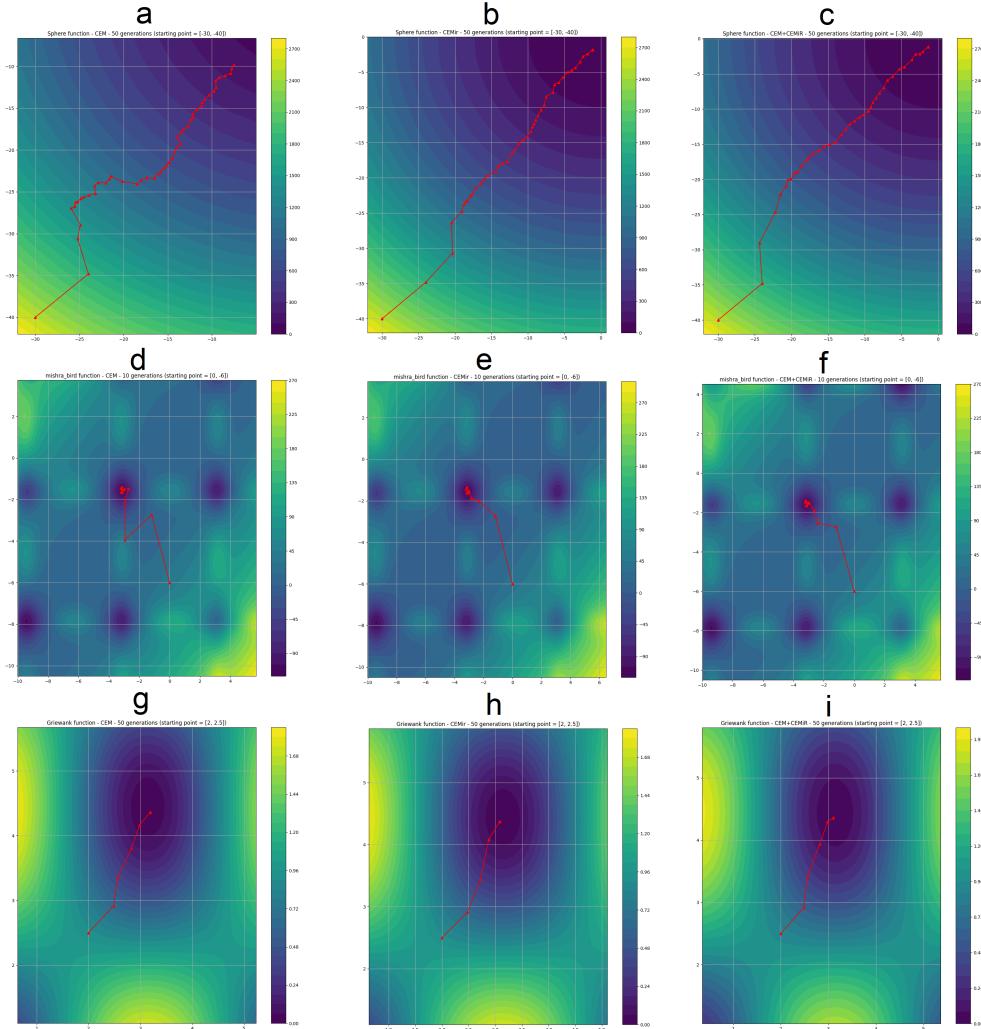


FIGURE 7 – (a) Exécution de la CEM sur la fonction objectif Sphère avec 50 générations.  
 (b) Exécution de la CEMir sur la fonction objectif Sphère avec 50 générations.  
 (c) Exécution de la CEM+CEMIR sur la fonction objectif Sphère avec 50 générations.  
 (d) Exécution de la CEM sur la fonction objectif Mishrabird avec 10 générations.  
 (e) Exécution de la CEMir sur la fonction objectif Mishrabird avec 10 générations.  
 (f) Exécution de la CEM+CEMIR sur la fonction objectif Mishrabird avec 10 générations.  
 (g) Exécution de la CEM sur la fonction objectif Griewank avec 50 générations.  
 (h) Exécution de la CEMir sur la fonction objectif Griewank avec 50 générations.  
 (i) Exécution de la CEM+CEMIR sur la fonction objectif Griewank avec 50 générations.

On a compilé, à l'Annexe B, l'ensemble des informations relatives à la convergence de nos méthodes sur le benchmark des fonctions d'optimisation en 2D.

## 3.2 Etude sur le benchmark des environnements de RL Gym

Les descriptions des figures sont identiques à celles de la sous-partie précédente.

### 3.2.1 Courbe d'efficacité

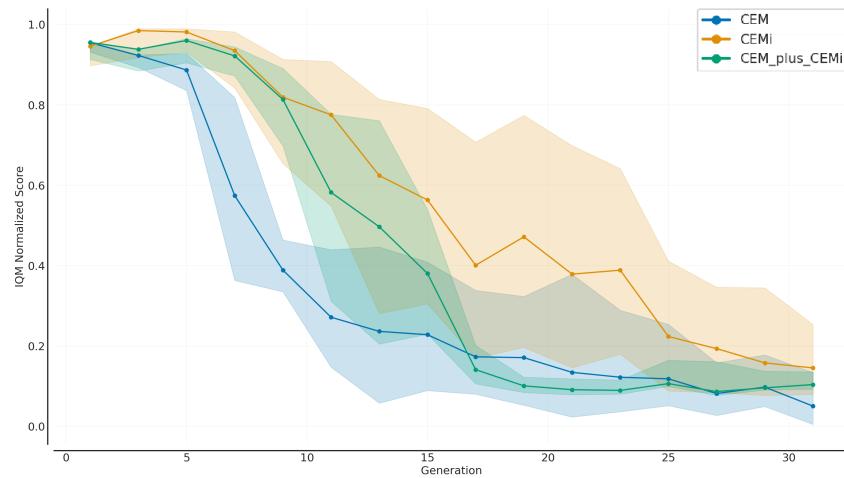


FIGURE 8 – Courbe d'efficacité - Environnements Gym.

### 3.2.2 Probabilité d'amélioration

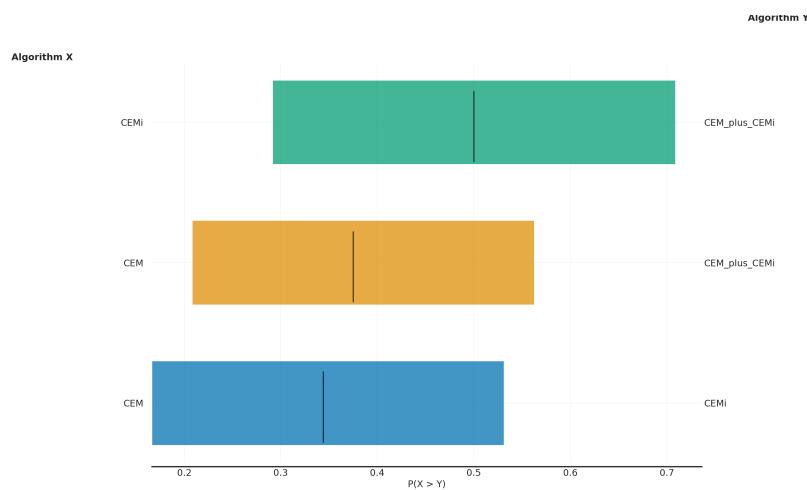


FIGURE 9 – Probabilité d'amélioration - Environnements Gym.

### 3.2.3 Profil de performance

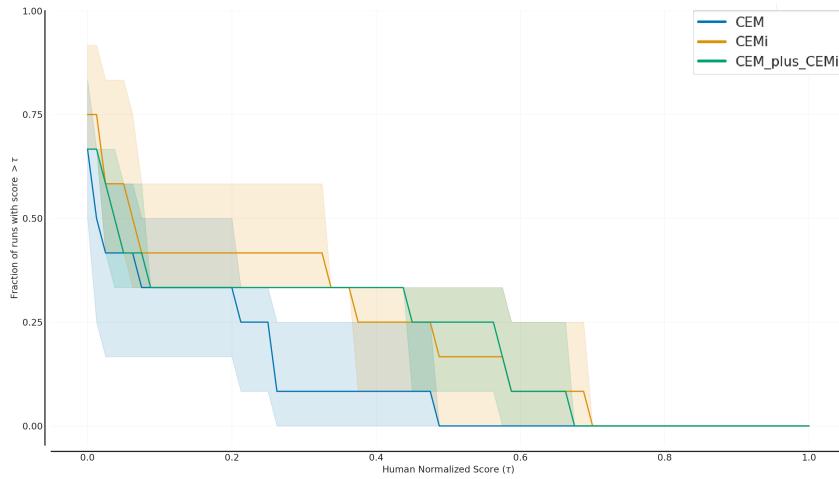


FIGURE 10 – Profil de performance - Environnements Gym.

### 3.2.4 Métriques agrégées

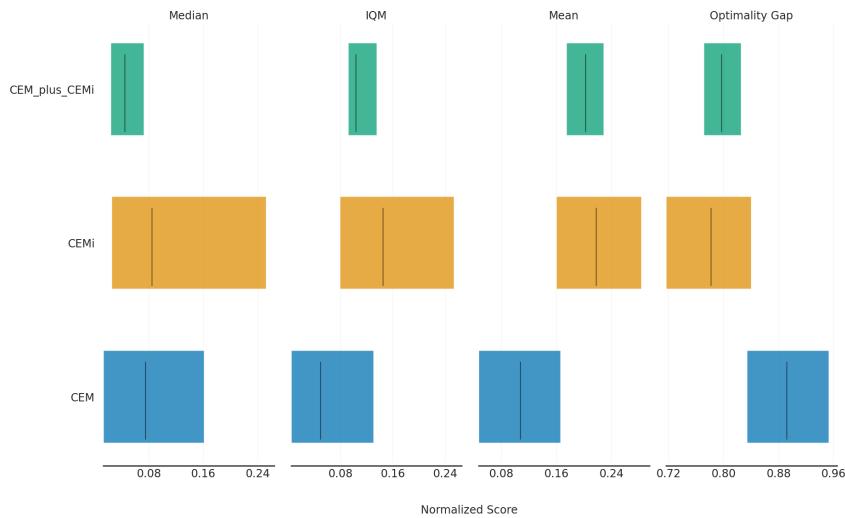


FIGURE 11 – Métriques agrégées - Environnements Gym.

## 4 Discussions

Notre problème était initialement de développer une version efficace de la CEMi, et de comparer ses performances à celle de la CEM, sur un benchmark d'environnements de RL GYM.

En plus d'avoir rempli cet objectif, nous avons développé plusieurs variantes de l'algorithme de départ plus performantes que la CEMi et que la CEM, en dimension 2 - à savoir la CEMir et la CEM+CEMir. Cependant, il est clair au vu des résultats obtenus, que ce gain de performance ne résiste pas à l'épreuve du passage à l'échelle.

## 4.1 Comportements en 2 dimensions

En se concentrant sur la Figure 4, on peut observer que toutes les variantes de la CEM sauf la CEMi, présentent un gain de performance notable vis-à-vis de l'algorithme initial. En effet, l'observation initiale selon laquelle inverser oriente la génération des individus dans la direction d'optimisation est confirmée par les meilleures performances de la CEMir par rapport à la CEM : en exacerbant les ellipsoïdes des inverses, on observe une amélioration des performances, ce qui était bien l'effet escompté.

En ce qui concerne la CEM+CEMir, l'algorithme le plus performant de ceux que nous avons développé, on constate en plus d'une amélioration des scores une réduction significative de la variance relative aux différentes exécutions (se reporter aux intervalles de variance dans l'Annexe B avec les intervalles de variance). Ce phénomène est particulièrement visible dans la comparaison des métriques agrégées (cf. figure 6), où la CEM+CEMir a les intervalles de variance les plus fins. Cependant, il faudrait conduire une étude mathématique plus poussée de la méthode pour expliquer les disparités sur l'incertitude.

On observe dans tous les cas une convergence suivant une loi de type  $1/x$ , comme le montre le graphe suivant : (plotter  $\log(Y)$  en fonction de  $\log(X)$  pour les valeurs normalisées, à faire avant le rendu final).

## 4.2 Comportement sur les environnements GYM

La méthode CEMir telle que nous l'avons développée n'est pas applicable aux environnements GYM, car les calculs matriciels dégénèrent au bout de quelques générations du fait de phénomènes d'instabilité numériques (cf. section A.2). Nous n'avons donc pas eu l'opportunité de vérifier si toutes les améliorations de performances que nous avons gagnées en 2 dimensions se traduisent bien dans des environnements de RL concrets, avec des centaines de dimensions.

Les courbes d'efficacités ainsi que les métriques agrégées montrent cependant un comportement surprenant : lorsqu'en 2 dimensions la CEMi et la CEM+CEMi avaient de meilleures performances que celle de la CEM, les deux variantes peinent à converger en dimensions supérieures.

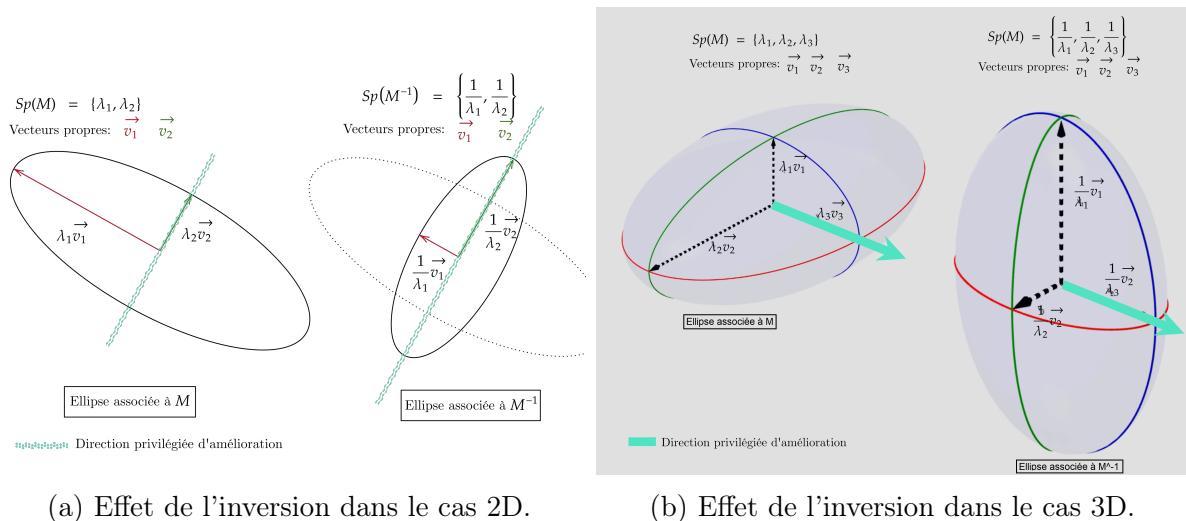
On pourrait évoquer quatre pistes pour expliquer cela :

- La première consisterait à dire que les valeurs sont tout simplement faussées par des imprécisions de calcul. Nous avons été confrontés de nombreuses fois à des problématiques d'instabilité numérique pendant ce projet, et il ne serait pas surprenant que malgré les précautions prises au moment de l'inversion avec la méthode de Cholesky (cf. section A.1), des erreurs de calcul apparaissent tout de même lors de la mise à jour des matrices de covariance.
- Ensuite, il se pourrait que les hyper-paramètres utilisés (tels que l'architechture du réseau de neurone, les paramètres de bruit, etc.) soient mal adaptés aux environnements. Ce sont des pistes que nous n'avons pas pu explorer à cause de limitations sur la puissance de calcul qui nous est disponible.
- Une autre piste consisterait à dire que la variance des individus générés par la méthode est beaucoup trop importante pour que la population évolue rapidement vers son objectif. En effet, si l'on s'attarde sur les écarts de variances entre les différentes méthodes dans le cas 2D (cf. les intervalles de variance en Annexe B), on constate que la CEMi est extrêmement instable, avec des amplitudes de variance de l'ordre de 3 pour certaines fonctions. Ce phénomène est manifestement amplifié

sur les environnements Gym, comme le montre les amplitudes d'écart-type pour la CEMi sur les mesures agrégées ainsi que sur les courbes de performance (cf. figure 3).

- Finalement, et c'est l'hypothèse la plus plausible, il se peut que l'inversion de la matrice ne tourne pas l'ellipsoïde associé dans la bonne direction de l'espace. En fait, comme le montre la figure 12, les observations faites en 2D sur l'alignement des ellipsoïdes inverses avec la progression vers l'optimum sont fausses à partir de la dimension 3.

Pour vérifier que la dernière hypothèse est celle qui traduit le mieux le comportement observé, il faudra mener plus d'expérimentations avec des matrices de covariances subissant des rotations dans les différentes dimensions de l'espace. Cependant, la recherche de l'optimum deviendrait d'autant plus coûteuse qu'il y a de nombre de dimensions à explorer, puisqu'il faudrait augmenter le nombre d'individus générés à chaque génération en conséquence.



(a) Effet de l'inversion dans le cas 2D.

(b) Effet de l'inversion dans le cas 3D.

FIGURE 12 – Schéma de comparaison entre une inversion en 2D et 3D. Comme expliqué à la section 2.4.3, les tailles des axes d'un ellipsoïde associé à une matrice de covariance sont égales à leurs valeurs propres. Inverser la matrice revient à inverser les longueurs des axes des ellipsoïdes, ce qui en 2D semble tourner l'ellipse de  $\frac{\pi}{2}$  radians. En dimensions multiples cependant, les ellipsoïdes sont simplement allongés dans la direction du plus petit axe, qui n'est pas forcément l'axe privilégié de la progression vers l'optimum.

## 5 Conclusions

Sur les environnements 2D, la CEM est moins performante que la CEMi, ainsi que toutes les variantes que nous avons développé. En particulier, la CEM+CEMir présente un gain significatif de performance sur l'ensemble du *benchmark* établi. (cf. section 3.1) Cependant, le gain de performance n'est pas préservé dans les environnements de RL (cf. section 3.2). La CEMi telle que développée n'a plus le comportement escompté en dimensions supérieures. Aussi, la CEMir se trouve être particulièrement instable en hautes dimensions, ce qui a rendu son application impossible aux environnements de RL, avec l'implémentation que nous avons réalisé.

Une extension naturelle du projet consisterait à transposer l'hypothèse originale de rotation des ellipsoïdes en dimension multiple. Plus précisément, on pourrait développer une variante de la CEM, où les nouveaux individus seraient générés à partir d'un ensemble de matrices de covariances ayant subies des rotations dans un certain nombre de dimensions de l'espace. Il deviendrait alors intéressant d'étudier le gain de performance obtenu, ainsi que l'impact sur les coûts de calculs additionnels.

## A Problèmes rencontrés et résolutions

### A.1 Inversion de matrice instable

L'inversion classique de la matrice de covariance dans la CEMi ne suffit pas lorsqu'il s'agit d'assurer une stabilité numérique en haute dimensions. En effet, après avoir effectué plusieurs inversions successives, les matrices de covariances nous avons obtenu une erreur indiquant que la matrice n'était pas définie positive.

Cette erreur est liée à l'instabilité et l'imprécision de l'inversion en nombres flottants. Au fur et à mesure des inversions successives, les erreurs d'arrondi s'accumulent, notamment pour les matrices de grande taille.

**Solution :**

Pour résoudre ce problème, nous utilisons l'inversion de Cholesky, une méthode d'inversion spécifique aux matrices symétriques définies positives. Cette méthode permet de mitiger les erreurs d'arrondi et de garantir que toutes les valeurs propres soient effectivement réelles et strictement positives, en exprimant la matrice à inverser sous une forme plus simple qui requiert moins d'opérations lors du calcul de son inverse.

L'inversion de Cholesky consiste à exprimer une matrice symétrique définie positive à inverser, comme un produit de la forme :  $A = L \cdot L^T$  où A est la matrice d'origine, L est une matrice triangulaire inférieure, et  $L^T$  la transposée de L.

De là, on a directement l'expression de  $A^{-1}$ , à savoir  $A^{-1} = (L^T)^{-1} \cdot L$ .

Le calcul de  $A^{-1}$  est moins prompt à générer des erreurs d'arrondi, car il n'implique qu'une inversion de matrice triangulaire (immédiate) et une multiplication, qui sont deux opérations stables numériquement.

On assure ainsi la cohérence de notre algorithme.

### A.2 Instabilité de la CEMir lorsqu'appliquée aux environnements Gym

L'inversion de matrice en nombre flottant est par nature instable, puisqu'elle implique le calcul de nombreuses divisions, surtout en haute dimensions. Dans notre algorithme de la CEMir (cf. section 2.4.3) nous redimensionnons la matrice *CovMatrixInverse* par le facteur  $\lambda \leftarrow \frac{\max \text{Sp}(\text{CovMatrix})}{\max \text{Sp}(\text{CovMatrixInverse})}$ , où le dénominateur  $\max \text{Sp}(\text{CovMatrixInverse})$  peut devenir très petit, ce qui accroît le risque de dégénérescence. Les expérimentations en 2D n'ont pas montré de signes d'instabilité numérique, mais la forte dimensionnalité des problèmes de RL sur les environnements GYM ont dû accumuler les erreurs au fil des générations, ce qui a résulté à des aberrations au niveau des résultats.

## B Récapitulatif des résultats sur les fonctions 2D

<b>CEM</b>				
Fonction	Centroid de départ	Génération de convergence	Intervalle de variance à la convergence	Intervalle de covariance à la convergence
Booth	[-10, -10]	30	0.1	0
Distance	[0, -6]	pas de convergence	-	-
Griewank	[2, 2.5]	5	[0.2, 0.3]	0
Holder	[0, -5]	31	0	0
Mishra Bird	[0, -6]	5	0	0
Rastrigin	[-5, -5]	30	[0.2, 1.55]	[-1.5, 0.5]
Rosenbrock	[-40, -40]	50	0.2	0
Sphere	[-30, -40]	60	0	0

(a) CEM.

<b>CEMi</b>				
Fonction	Centroid de départ	Génération de convergence	Intervalle de variance à la convergence	Intervalle de covariance à la convergence
Booth	[-10, -10]	12	[1, 3.5]	[0, 1.5]
Distance	[0, -6]	pas de convergence	-	-
Griewank	[2, 2.5]	21	[0.5, 3.5]	[-0.5, 1]
Holder	[0, -5]	pas de convergence	-	-
Mishra Bird	[0, -6]	10	[0.5, 4]	[-1, 1]
Rastrigin	[-5, -5]	18	[1.5, 3]	[-0.8, 0.9]
Rosenbrock	[-40, -40]	20	[0.5, 3.5]	[-1.5, 1]
Sphere	[-30, -40]	28	[0, 4]	0

(b) CEMi.

<b>CEMir</b>				
Fonction	Centroid de départ	Génération de convergence	Intervalle de variance à la convergence	Intervalle de covariance à la convergence
Booth	[-10, -10]	21	[0.2, 0.35]	[0, 0.1]
Distance	[0, -6]	pas de convergence	-	-
Griewank	[2, 2.5]	4	[0.2, 0.3]	0
Holder	[0, -5]	11	0	0
Mishra Bird	[0, -6]	4	0	0
Rastrigin	[-5, -5]	20	[0.4, 2.5]	[-0.8, 0.6]
Rosenbrock	[-40, -40]	40	[0.2, 0.5]	[-0.1, 0.1]
Sphere	[-30, -40]	36	0	0

(c) CEMir.

<b>CEM + CEMi</b>				
Fonction	Centroid de départ	Génération de convergence	Intervalle de variance à la convergence	Intervalle de covariance à la convergence
Booth	[-10, -10]	14	[0, 4]	[-0.5, 1.5]
Distance	[0, -6]	pas de convergence	-	-
Griewank	[2, 2.5]	6	[0.5, 3.5]	[-1.5, 1.5]
Holder	[0, -5]	pas de convergence	-	-
Mishra Bird	[0, -6]	5	[0, 5]	0
Rastrigin	[-5, -5]	12	[0.5, 2.6]	[-0.6, 1]
Rosenbrock	[-40, -40]	22	[0.2, 4.5]	[-1, 1.5]
Sphere	[-30, -40]	30	[0, 5]	0

(d) CEM plus CEMi.

<b>CEM + CEMir</b>				
Fonction	Centroid de départ	Génération de convergence	Intervalle de variance à la convergence	Intervalle de covariance à la convergence
Booth	[-10, -10]	21	[0.2, 0.4]	[-0.1, 0.1]
Distance	[0, -6]	pas de convergence	-	-
Griewank	[2, 2.5]	4	[0.2, 0.3]	0
Holder	[0, -5]	31	0	0
Mishra Bird	[0, -6]	4	0	0
Rastrigin	[-5, -5]	20	[0.2, 1.9]	[-0.6, 0.6]
Rosenbrock	[-40, -40]	40	[0.25, 0.7]	[-0.25, 0.25]
Sphere	[-30, -40]	32	0	0

(e) CEM plus CEMir.

FIGURE 13 – Récapitulatif des résultats observés sur le benchmark de fonctions 2D.

## C Fonctions de tests d'optimisation utilisées

Nom de la fonction	Formule	Minimum global	Domaine de recherche
Fonction de Booth	$f(\mathbf{x}, \mathbf{y}) = (x + 2y - 7)^2 + (2x + y - 5)^2$	$f(1, 3) = 0$	$-10 \leq x, y \leq 10$
Fonction de Griewank	$f(\mathbf{x}) = \frac{1}{4000} \sum_{i=1}^n x_i^2 - \prod_{i=1}^n \cos\left(\frac{x_i}{\sqrt{i}}\right) + 1$	$f(0, \dots, 0) = 0$	$-600 \leq x_i \leq 600$
Fonction de Holder	$f(x, y) = -\left \sin(x) \cos(y) \exp\left(1 - \frac{\sqrt{x^2+y^2}}{\pi}\right)\right $	$\text{textMin} = \begin{cases} f(8.05502, 9.66459) = -19.2085 \\ f(-8.05502, 9.66459) = -19.2085 \\ f(8.05502, -9.66459) = -19.2085 \\ f(-8.05502, -9.66459) = -19.2085 \end{cases}$	$-10 \leq x, y \leq 10$
Fonction de Mishra Bird	$f(x, y) = \sin(y)e^{[(1-\cos x)^2]} + \cos(x)e^{[(1-\sin y)^2]} + (x - y)^2$	$f(-3.1302468, -1.5821422) = -106.7645367$	$-10 \leq x \leq 0$ $-6.5 \leq y \leq 0$
Fonction de Rastrigin	$f(\mathbf{x}) = An + \sum_{i=1}^n [x_i^2 - A \cos(2\pi x_i)],$ où $A = 10$	$f(0, \dots, 0) = 0$	$-5.12 \leq x_i \leq 5.12$
Fonction de Rosenbrock	$f(\mathbf{x}) = \sum_{i=1}^{n-1} \left[ 100(x_{i+1} - x_i^2)^2 + (1 - x_i)^2 \right]$	$\text{Min} = \begin{cases} n = 2 \rightarrow f(1, 1) = 0, \\ n = 3 \rightarrow f(1, 1, 1) = 0, \\ n > 3 \rightarrow f(\underbrace{1, \dots, 1}_{n \text{ times}}) = 0 \end{cases}$	$-\infty \leq x_i \leq \infty,$ $1 \leq i \leq n$
Fonction de Sphère	$f(\mathbf{x}) = \sum_{i=1}^n x_i^2$	$f(x_1, \dots, x_n) = f(0, \dots, 0) = 0$	$-\infty \leq x_i \leq \infty,$ $1 \leq i \leq n$
Fonction distance au point (150,200)	$f(\mathbf{x}) = \sqrt{(x - 150)^2 + (y - 200)^2}$	$f(150, 200) = 0$	$\mathbb{R}^2$

TABLE 1 – Descriptions des fonctions de test pour l’optimisation.

## D Environnements et bibliothèques utilisées

Dans ce projet, nous utilisons les environnements d’apprentissage renforcé Cartpole, Pendulum, Lunar lander et Acrobot, sur lesquels nous avons pu entraîner des agents simples grâce aux bibliothèques Python suivantes, co-développées par Olivier Sigaud, notre encadrant : BBRL\_examples [4], BBRL [5] et BBRL Gym [6].

BBRL\_examples implémente à la base une version de la CEM, BBRL et BBRL Gym facilitent l’usage des environnements classiques de *Reinforcement Learning* fournis par la bibliothèque OpenAI Gym.

Description des librairies utilisées :

- OpenAI Gym :

C’est une bibliothèque *open source* qui fournit une collection d’environnements classiques d’apprentissage par renforcement. Nous utilisons ceux cités au début de la section.

- BBRL Gym : Cette bibliothèque contient des environnements *Gym* supplémentaires, comme CartPoleContinuous et MazeMDP.

- BBRL :

*BlackBoard Reinforcement Learning* est une bibliothèque dérivée de la bibliothèque *SaLinA*. Elle étend les fonctionnalités de *PyTorch* pour le développement d’algorithmes d’apprentissage par renforcement.

L’étude statistique des performances est réalisée en se basant sur les recommandations de l’article [3], qui propose différentes métriques pour l’évaluation et la comparaison d’algorithme de RL en rendant compte de la variabilité des résultats obtenus. Pour ce faire, on utilise la bibliothèque compagnon de l’article [7].

## Références

- [1] [https://en.wikipedia.org/wiki/Cross-entropy\\_method](https://en.wikipedia.org/wiki/Cross-entropy_method).
- [2] <https://en.wikipedia.org/wiki/CMA-ES>.
- [3] R. Agarwa, M. Schwarzer, P. S. Castro, A. Courville, and M. G. Bellemare, “Deep Reinforcement Learning at the Edge of the Statistical Precipice,” *arXiv preprint arXiv :2108.13264*, 2022.
- [4] [https://github.com/osigaud/bbrl\\_examples](https://github.com/osigaud/bbrl_examples).
- [5] <https://github.com/osigaud/bbrl>.
- [6] [https://github.com/osigaud/bbrl\\_gym](https://github.com/osigaud/bbrl_gym).
- [7] <https://github.com/google-research/rliable>.