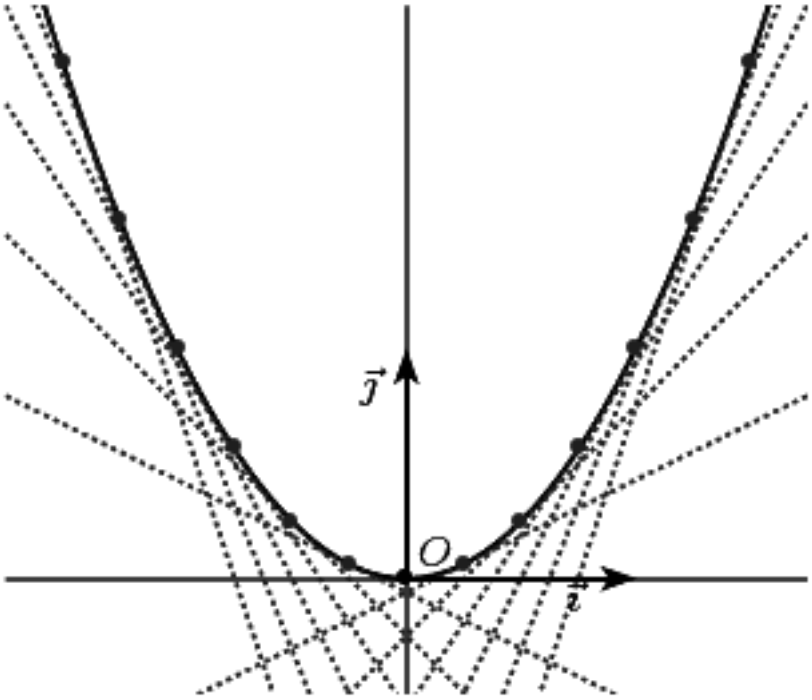


Rapport OCVX1 - TP1

Participants :
Adam ISMAILI
Noé TOPEZA
Scott TALLEC

Professseur :
Guillaume TOCHON

Table des matières

1 Introduction **2**
1.1 Répartition des tâches 2

2 Influence du pas sur le nombre d'itérations **3**

3 Impact de la méthode avec rebroussement (avec critère d'Armijo) **6**
3.1 Principe 6
3.2 Études 7
3.3 Impact du pas initial sur la descente 8

4 Impact de la méthode pour l'algorithme du gradient conjugué **9**
4.1 Objectif 9
4.2 Données et méthodologie 10
4.3 Applications et Résultats 14

5 Conclusion **17**

1 Introduction

La déscente de gradient est une technique d'optimisation largement utilisée, jouant un rôle crucial dans la résolution de problèmes d'optimisation sur des fonctions convexes et non convexes.

Il nous est alors demandé d'analyser les performances de différentes techniques de descente de gradient sur une variété de problèmes d'optimisation.

Dans notre étude, nous tenterons ainsi d'évaluer l'influence d'un hyperparamètre du critère d'Armijo dans le cas d'une descente de gradient pour une fonction convexe, l'influence du pas sur le nombre d'itérations dans le cas de la descente de gradient à pas constant, et l'influence du choix de la méthode (Fletcher-Reeves vs Polack-Ribière) pour l'algorithme du gradient conjugué pour une classe de fonction non-convexes.

Nous nous établierons à mesurer les performances itératives et temporelles de chacune de nos solutions par l'usage de benchmarks, permettant de décrire la solution la plus optimale rencontrée pour nos problèmes.

1.1 Répartition des tâches

— Scott : Étude sur l'influence du pas sur le nombre d'itérations ainsi que l'élaboration de graphiques pour étudier la convergence de différentes fonctions avec la méthode de descente de gradient.

— Noé : Impact de la méthode avec rebroussement (avec critère d'Armijo) + Pipeline des graphiques.

— Adam : Etude de l'influence du choix de la méthode pour l'algorithme du gradient conjugué sur une classe de fonctions non-convexes.

2 Influence du pas sur le nombre d'itérations

Dans la descente de gradient, il existe un compromis entre le pas et le nombre d'itérations. Un pas plus grand peut permettre à l'algorithme de faire de plus grands sauts dans l'espace des paramètres, ce qui peut potentiellement atteindre plus rapidement le minimum, mais il risque également de le dépasser. En revanche, un pas plus petit peut garantir que l'algorithme se rapproche plus précisément du minimum, mais cela peut nécessiter un plus grand nombre d'itérations pour y parvenir.

Cela a été démontré expérimentalement avec la fonction quadratique où

$$f(x) = x^2$$

, comme on peut le voir sur la figure 5.

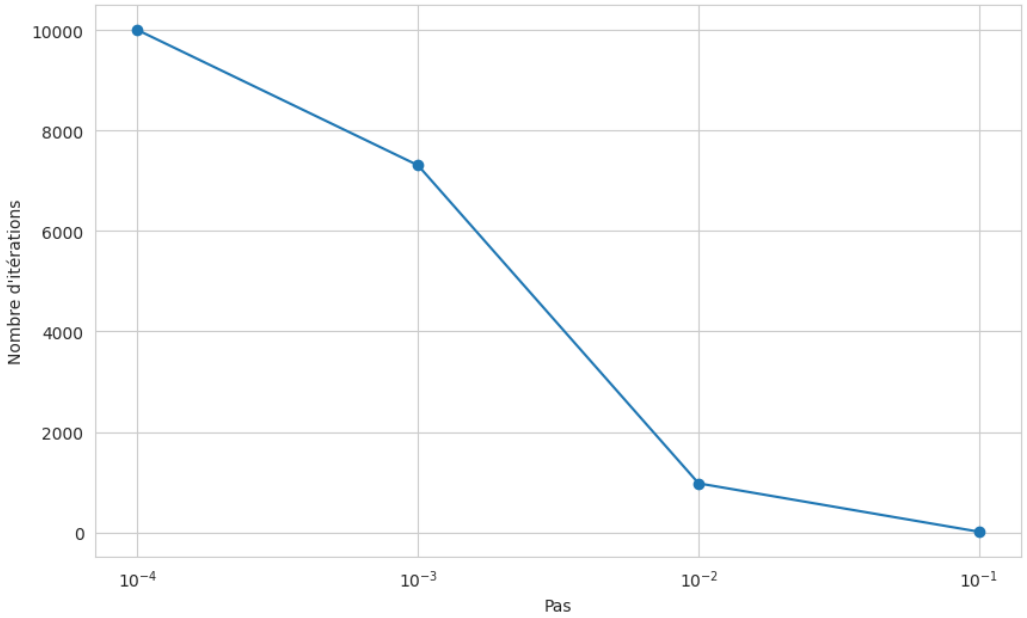
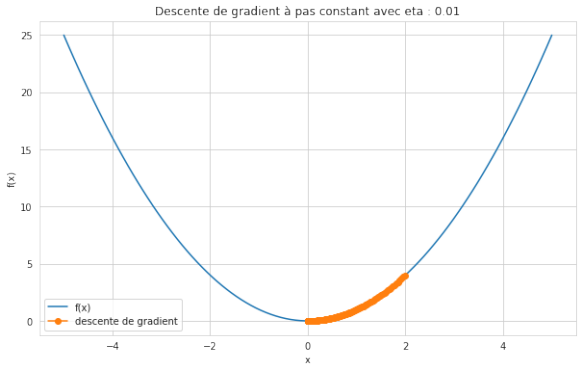


FIGURE 1 – Influence du pas sur le nombre d'itérations d'une fonction quadratique

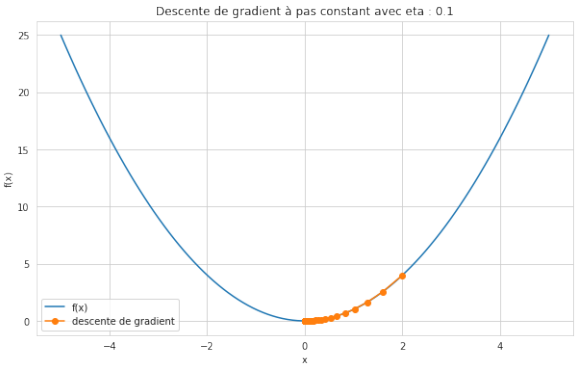
Alors, que se passe-t-il lorsque le pas est trop important ? Nous avons effectué des tests avec un pas large, ce qui peut être observé dans les résultats présentés dans les figures 2 et 3. L'algorithme de descente de gradient oscille autour du minimum et ne parvient pas à converger lorsque le pas est trop grand.

En effet, lorsque nous faisons un pas trop grand, nous supposons que la fonction se comporte de la même manière en un point éloigné comme elle le fait à notre position actuelle. Mais ce n'est pas nécessairement le cas, en particulier pour des fonctions complexes. Ainsi, avec une taille de pas importante, nous pourrions arriver à un point où la valeur de la fonction est bien plus élevée, ce qui peut entraîner une divergence. On voit que le calcul de x peut atteindre une valeur très élevée, comme illustré dans la figure 3b.

Ce phénomène n'est pas limité à un taux d'apprentissage de 1 ou plus, des taux d'apprentissage plus petits peuvent également provoquer une divergence, en particulier pour des fonctions

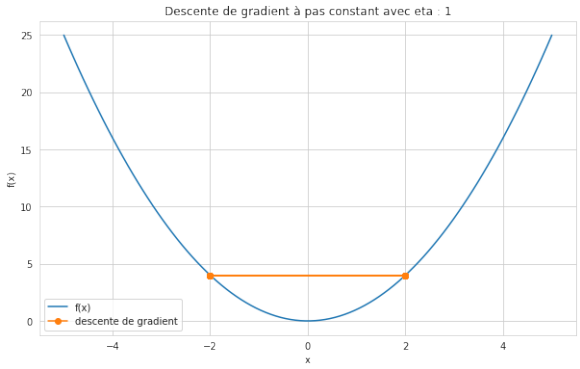


(a) Descente de gradiant a pas constant avec eta = 0.01, x calculé : 4.852279516386025e-05

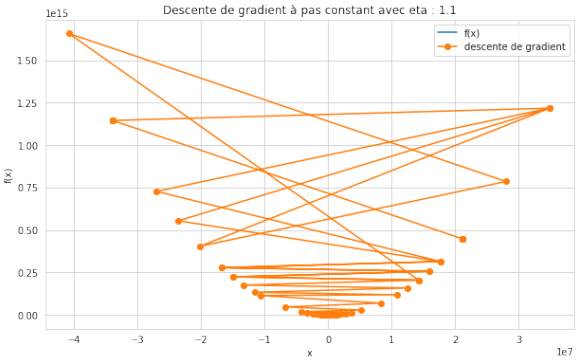


(b) Descente de gradiant a pas constant avec eta = 0.1, x calculé : 3.831238870732885e-06

FIGURE 2 – Traçage de la descente de gradient avec différents step, ici convergent



(a) Descente de gradiant a pas constant avec eta = 1, x calculé :1.9999753086399323



(b) Descente de gradiant a pas constant avec eta = 1.1, x calculé : 21153675.89480392

FIGURE 3 – Traçage de la descente de gradient avec différents page, ici divergent

qui ont des pentes abruptes. Par conséquent, le choix du taux d'apprentissage est un aspect clé de l'application réussie de la descente de gradient.

C'est la raison pour laquelle nous avons apporté une première amélioration : le choix d'un pas "optimal" à chaque étape grâce à la méthode du backtracking (ou rebroussement) avec le critère d'Armijo.

En utilisant cette approche, nous pouvons trouver un pas adapté à chaque itération de la descente de gradient, ce qui nous permet de bénéficier des avantages des pas plus grands en début de descente pour une convergence plus rapide, tout en garantissant une progression plus précise vers le minimum en réduisant le pas au fur et à mesure de l'optimisation.

Avec Rosenbrock on obtient, la convergence suivante :

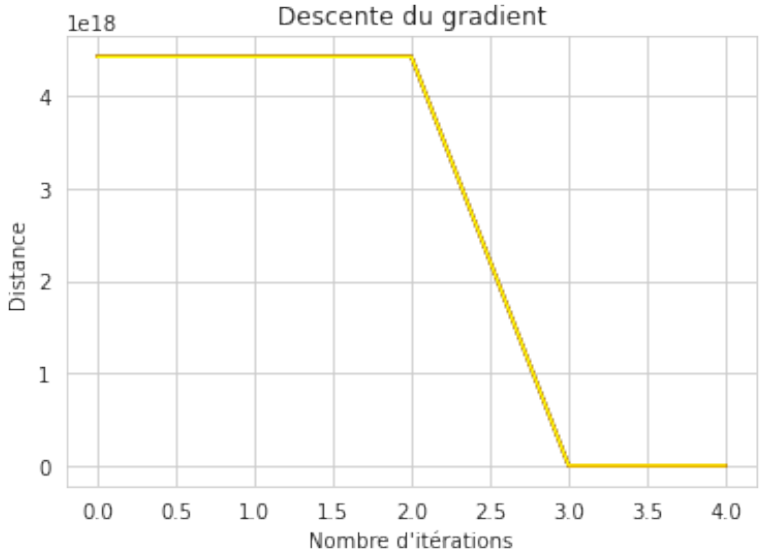


FIGURE 4 – Influence du pas sur le nombre d'itérations sur la fonction de Rosenbrock

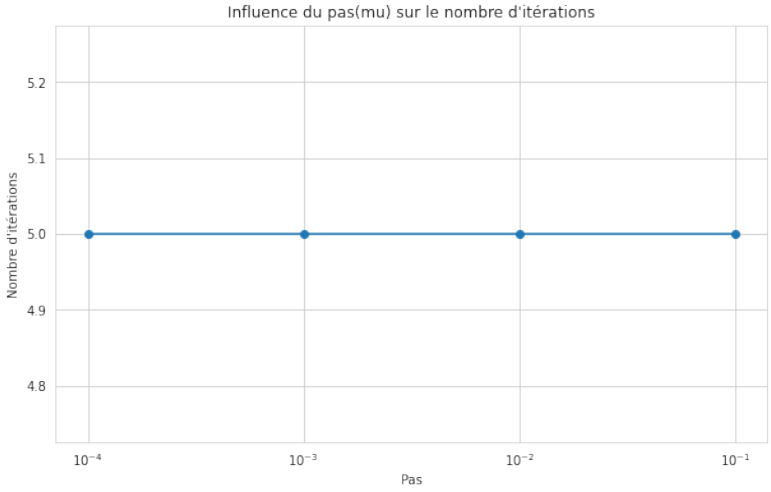


FIGURE 5 – Influence du pas(mu)

La fonction de **Rosenbrock** présente une vallée étroite et allongée avec une zone de pente plate. Cette zone plate peut poser des difficultés à une méthode de descente de gradient à pas constant, car la méthode peut rester piégée dans cette région sans converger vers le minimum global.

La méthode de descente de gradient à pas constant utilise une taille de pas fixe pour mettre à jour itérativement les valeurs des variables selon le gradient de la fonction. Cependant, dans le cas de la fonction de Rosenbrock, une taille de pas constante peut entraîner des oscillations ou des erreurs de convergence.

Pour surmonter ces problèmes, il est souvent recommandé d'utiliser des méthodes d'optimisation plus avancées, telles que la méthode de Newton, la méthode de quasi-Newton (comme la méthode BFGS), ou des variantes de la descente de gradient avec ajustement automatique du pas (par exemple, la méthode de descente de gradient avec recherche linéaire).

3 Impact de la méthode avec rebroussement (avec critère d'Armijo)

3.1 Principe

La méthode avec rebroussement (backtracking) en lien avec le critère d'Armijo est utilisée dans l'optimisation numérique, en particulier dans la recherche linéaire pour déterminer la taille d'un pas dans une direction donnée qui minimise une fonction objectif.

Il utilise le critère d'Armijo, utilisé pour choisir un pas de manière efficace dans l'algorithme de descente de gradient. Il aide à s'assurer que le pas choisi réduit suffisamment la fonction objectif.

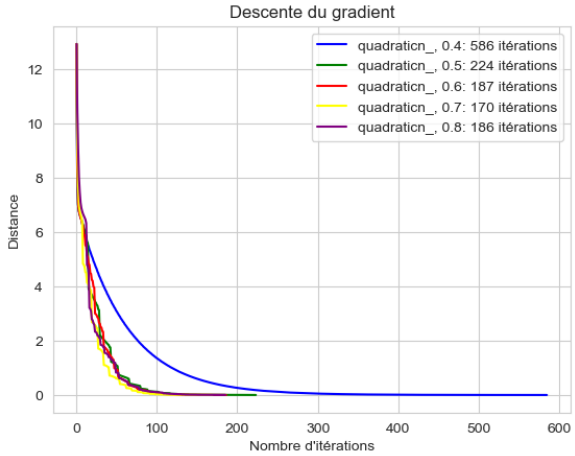
$$f(\mathbf{x}_k + \mu - \nabla f\,(x_k) < f(\mathbf{x}_k) + \alpha\,\mu\, - \nabla f\,(x_k)^T\ \nabla f(\mathbf{x}_k)$$

La méthode de backtracking avec le critère d'Armijo fonctionne en choisissant d'abord un pas de grande taille, puis en le réduisant jusqu'à ce que le critère d'Armijo soit satisfait. Cela peut être plus efficace que d'autres méthodes, car il peut éviter de faire de petits pas inefficaces lorsque de plus grands pas pourraient être plus efficaces.

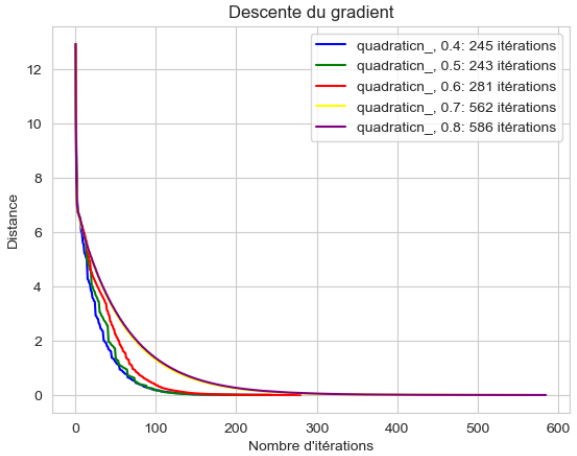
Le principal avantage de cette approche est qu'elle peut accélérer la convergence de l'algorithme d'optimisation en évitant à la fois les pas trop petits (qui peuvent ralentir la convergence) et les pas trop grands (qui peuvent dépasser le minimum de la fonction).

Dans cette étude, deux paramètres d'Armijo, alpha et beta, sont variés pour observer leur influence sur la vitesse de convergence de l'algorithme.

3.2 Études

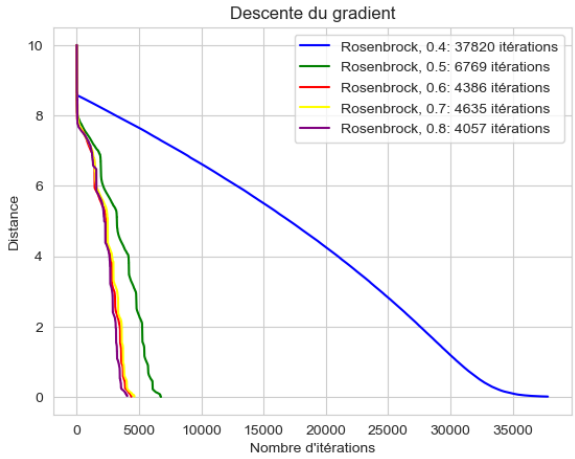


(a) Variation du facteur alpha (beta = 0.8)

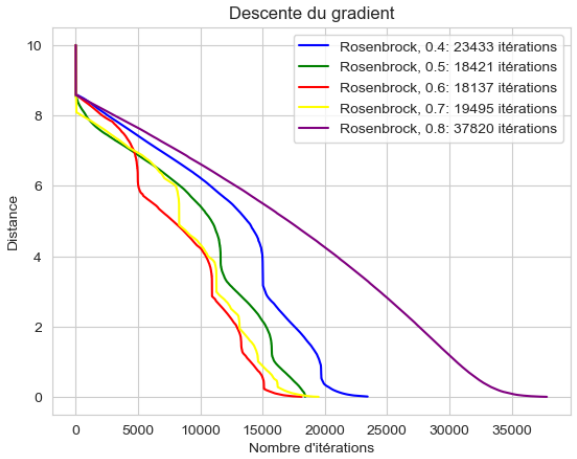


(b) Variation du facteur beta (alpha = 0.4)

FIGURE 6 – Descente de gradient sur fonction Quadratique



(a) Variation du facteur alpha (beta = 0.8)



(b) Variation du facteur beta (alpha = 0.4)

FIGURE 7 – Descente de gradient sur fonction de Rosenbrock

Les graphique ci-dessus illustre bien l'impact des critère d'Armijo sur la vitesse convergence de l'algorithme.

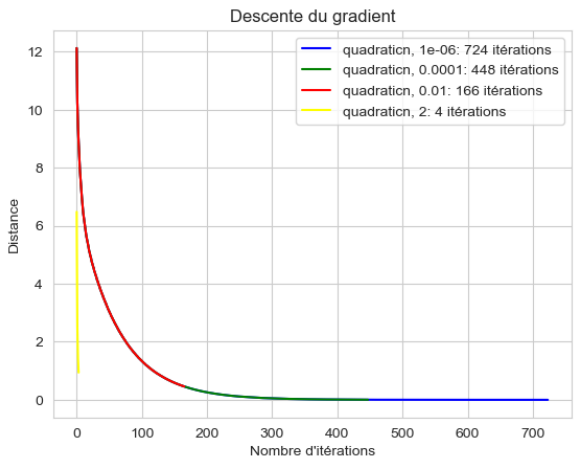
— Le premier affecte le choix de reduction du pas a chaque itération : C'est très visible dans le troisième graphique où l'algorithme a besoin d'un pas élevé pour atteindre plus rapidement le minimum de la fonction, ainsi la valeur d'alpha la plus élevé donne le meilleur résultat.

— Le second affecte la réduction du pas lorsqu'elle est appliqué : on voit ainsi que réduire trop peu le pas (avec un beta trop élevé) augmente le nombre d'itérations, de même pour un beta trop faible (les deux visible sur le dernier graphique).

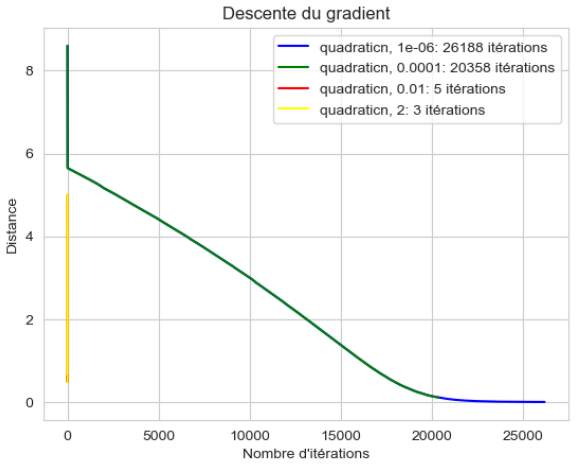
En somme, cette étude illustre l'importance des paramètres alpha et beta dans la méthode de backtracking avec le critère d'Armijo. Choisir des valeurs appropriées pour ces paramètres peut avoir un impact significatif sur l'efficacité de la descente de gradient.

3.3 Impact du pas initial sur la descente

Des tests on été menés pour évaluer l'impact du pas initial sur la vitesse de descente : On y



(a) Variation du pas (fonction Quadratique)



(b) Variation du pas (fonction Rosenberck)

FIGURE 8 – Descente de gradient sur fonction Quadratique

remarque que, excepté pour un pas absurde (> 1), le rythme de diminution de la distance reste stable peu importe le pas. avec un nombre d'itération inversement proportionnel au pas. Aussi, un pas élevé signifit une précision du résultat plus faible.

4 Impact de la méthode pour l'algorithme du gradient conjugué

4.1 Objectif

Un algorithme de gradient conjugué est une méthode d'optimisation utilisée pour résoudre des problèmes d'optimisation sans contraintes.
Avec l'usage du gradient conjugué, ce choix de descente est fait pour rendre deux directions de descentes orthogonales pour le produit scalaire qui vient de la Hessienne.
Il est particulièrement efficace pour les problèmes avec une fonction objectif quadratique, car le calcul est direct.
A contrario, il peut devenir compliqué quand la Hessienne n'est pas directement accessible.

Son usage est évident dans le cadre d'optimisation de fonctions convexes, mais l'est aussi pour l'optimisation de fonctions non-convexes, exploitant les propriétés de conjuguaison entre les directions de recherche pour converger plus efficacement vers la solution optimale.
En effet, au lieu de suivre simplement la direction du gradient négatif, comme dans l'algorithme du gradient classique, les directions de recherche conjuguées permettent de prendre en compte des informations passées pour guider la recherche vers les minima locaux et globaux.

Il est important de noter que la performance des algorithmes de gradient conjugué peut varier selon la nature spécifique de la fonction non-convexe et des conditions initiales.
Il est ainsi important d'expérimenter et d'évaluer différentes méthodes de gradient conjugué pour trouver celle qui convient le mieux à un problème d'optimisation non-convexe donné.

C'est donc ce que nous allons faire. Nous utiliserons des méthodes de descente de gradient conjugué sur une classe de fonctions non-convexes, et évaluerons les performances de chacune afin de choisir celle qui correspond le mieux à notre classe de fonctions.

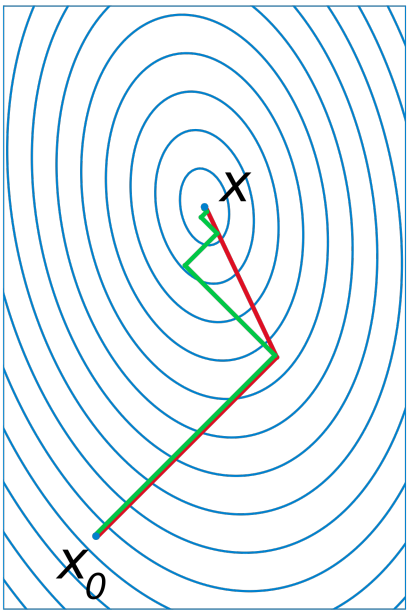


FIGURE 9 – Illustration de la méthode du gradient conjugué

4.2 Données et méthodologie

Nous disposons de deux méthodes de descente de gradient adaptées au problème :

— **Fletcher-Reeves**
Elle combine la méthode du gradient conjugué avec la règle de mise à jour de Fletcher-Reeves pour déterminer les directions de recherche et les pas de mise à jour optimaux.

La constante de mise à jour β_k dans la méthode de Fletcher-Reeves est calculée à partir des gradients successifs. Elle mesure la proportion de la nouvelle direction de recherche qui est conjuguée à la direction précédente.
La formule de mise à jour est la suivante :

$$\beta_k = \frac{g_{k+1}^T \cdot g_{k+1}}{g_k^T \cdot g_k}$$

où g_k est le gradient précédent et g_{k+1} est le gradient actuel de la fonction objectif.

Cette formule permet de déterminer la mesure de conjuguaison entre les gradients successifs et d'ajuster la nouvelle direction de recherche en conséquence.

La direction de recherche dans la méthode de Fetcher-Reeves est mise à jour en utilisant également le nouveau gradient et la constante de mise à jour β_k.
La mise à jour est effectuée selon la formule suivante :

$$d_{k+1} = -g_{k+1} + \beta_k \cdot d_k$$

où d_k est la direction de recherche précédente et d_{k+1} est la nouvelle direction de recherche.

Cette mise à jour garantit que la nouvelle direction de recherche est une combinaison de la direction opposée au gradient actuel et d'une composante conjuguée à la direction précédente, favorisant ainsi la recherche dans des directions conjuguées.

En somme, les règles de mise à jour de Fletcher-Reeves permettent de calculer une constante de mise à jour qui mesure la conjuguaison entre les gradients successifs, et d'utiliser cette constante pour mettre à jour la direction de recherche dans l'algorithme du gradient conjugué.

— **Polak-Ribière**
Elle combine la méthode du gradient conjugué avec la règle de mise à jour de Polak-Ribière pour déterminer les directions de recherche et les pas de mise à jour optimaux.

La constante de mise à jour β_k dans la méthode de Polak-Ribière est aussi calculée à partir des gradients successifs, en mesurant la proportion de la nouvelle direction de recherche qui est conjuguée à la direction précédente.
La formule de mise à jour est la suivante :

$$\beta_k = \frac{g_{k+1}^T \cdot (g_{k+1} - g_k)}{g_k^T \cdot g_k}$$

où g_k est le gradient précédent et g_{k+1} est le gradient actuel de la fonction objectif.

Cette formule permet également de déterminer la mesure de conjuguaison entre les gradients successifs et d'ajuster la nouvelle direction de recherche en conséquence.

La direction de recherche dans la méthode de Polak-Ribière est mise à jour en utilisant le nouveau gradient et la constante de mise à jour β_k.
La mise à jour est effectuée selon la formule suivante :

$$d_{k+1} = -g_{k+1} + \beta_k \cdot d_k$$

où d_k est la direction de recherche précédente et d_{k+1} est la nouvelle direction de recherche.

De même que pour la méthode précédente, cette mise à jour garantit que la nouvelle direction de recherche est une combinaison de la direction opposée au gradient actuel et d'une composante conjuguée à la direction précédente, favorisant ainsi la recherche dans des directions conjuguées.

En somme, les règles de mise à jour de Polak-Ribière permettent de calculer une constante de mise à jour qui mesure la conjuguaison entre les gradients successifs, et d'utiliser cette constante pour mettre à jour la direction de recherche dans l'algorithme du gradient conjugué.

```
Function GradientConjugueFR(function_objective, gradient_function,
starting_point, tolerance):
    Initialize x₀ as starting_point
    Initialize g₀ as gradient_function(x₀)
    Initialize d₀ as -g₀
    Initialize β₀ as 0

    For k ranging from 0 to a maximum number of iterations:
        Calculate αₖ by performing a linear search in the direction dₖ
        Update the point: xₖ₊₁ = xₖ + αₖ * dₖ
        Calculate the gradient: gₖ₊₁ = gradient_fonction(xₖ₊₁)

        Calculate βₖ using the Fletcher-Reeves method:
        βₖ = (gₖ₊₁ᵀ * gₖ₊₁) / (gₖᵀ * gₖ)

        Update search direction: dₖ₊₁ = -gₖ₊₁ + βₖ * dₖ

        Check the stopping condition: if ||gₖ₊₁|| ≤ tolerance,
        stop the algorithm

    Return xₖ₊₁
End of function
```

(a) Pseudo-code de la méthode Fletcher-Reeves (FR)

```
Function GradientConjugueFR(objective_function, gradient_function,
starting_point, tolerance):
    Initialize x₀ as starting_point
    Initialize g₀ as gradient_function(x₀)
    Initialize d₀ as -g₀
    Initialize β₀ as 0

    For k ranging from 0 to a maximum number of iterations:
        Calculate αₖ by performing a linear search in the direction dₖ
        Update the point: xₖ₊₁ = xₖ + αₖ * dₖ
        Calculate the gradient: gₖ₊₁ = gradient_fonction(xₖ₊₁)

        Calculate βₖ using the Polak-Ribière method:
        βₖ = max(0, (gₖ₊₁ᵀ * (gₖ₊₁ - gₖ)) / (gₖᵀ * gₖ))

        Update search direction: dₖ₊₁ = -gₖ₊₁ + βₖ * dₖ

        Check the stopping condition: if ||gₖ₊₁|| ≤ tolerance,
        stop the algorithm

    Return xₖ₊₁
End of function
```

(b) Pseudo-code de la méthode Polak-Ribière (PR)

FIGURE 10 – Pseudo-code des deux méthodes de descente de gradient conjugué étudiées

Notre problème étant l'optimisation de fonctions non-convexes, nous disposons également d'une classe de fonctions non-convexes, étant :

— **Rosenbrock**
Egalement connue sous le nom de "vallée de Rosenbrock" ou "fonction de la banane", l'allure de la courbe de la fonction ressemble à une vallée étroite et allongée avec une forme caractéristique en "banane".
Elle est définie comme suit :

$$f(x, y) = (1 - x)^2 + 100 \cdot (y - x^2)^2$$

Le minimum global de cette fonction se trouve à $(x, y) = (1, 1)$, où $f(x, y) = 0$.

— Circuit schematics highlight the circuit components and their connectivity.

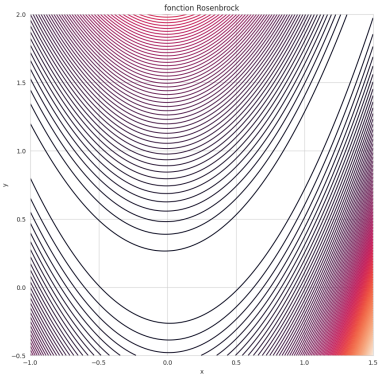


FIGURE 11 – Visualisation 2D de Rosenbrock

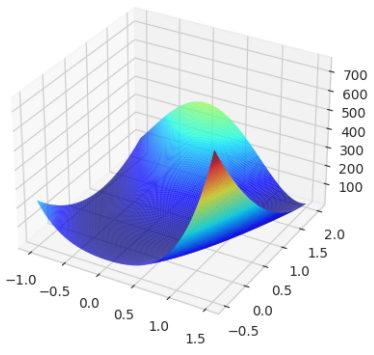


FIGURE 12 – Visualisation 3D de Rosenbrock

— **Rastrigin**
La courbe de cette fonction est caractérisée par des motifs de "bosses" et de "vallées" réguliers, disposant ainsi de plusieurs minima locaux et d'un minimum global connu.
Elle est définie comme suit :

$$f(x_1, x_2, ..., x_n) = A \cdot n + \Sigma(x_i^2 - A \cdot cos(2\pi \cdot x_i))$$

où A est une constante positive.
Son minimum global se trouve à $(x_1, x_2, ..., x_n) = (0, 0, ..., 0)$, où $f(x_1, x_2, ..., x_n) = 0$.

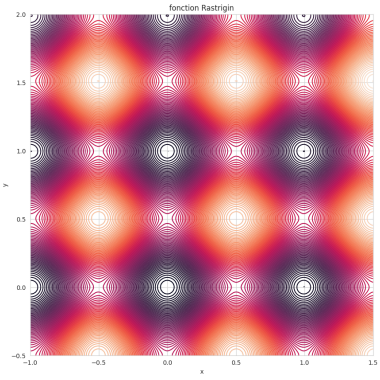


FIGURE 13 – Visualisation 2D de Rastrigin

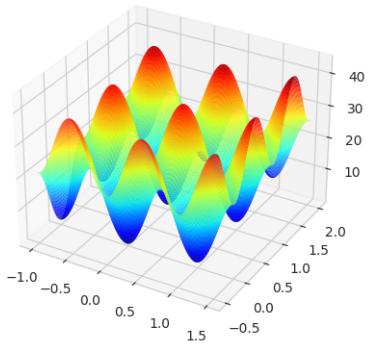


FIGURE 14 – Visualisation 3D de Rastrigin

— **Ackley**
La courbe de cette fonction présente généralement de nombreux petits creux et bosses répartis dans l'espace de recherche. Ces creux et bosses peuvent avoir différentes formes et profondeurs, créant une apparence complexe et irrégulière. Elle possède ainsi plusieurs minima locaux et un minimum global connu.
Elle se définie comme suit :

$$f(x_1, x_2, ..., x_n) = -a \cdot e^{-b \cdot \sqrt{\frac{1}{n} \cdot \Sigma x_i^2}} - e^{\frac{1}{n} \cdot \sqrt{cos(c \cdot x_i)}} + a + e$$

où A, B et C sont des constantes positives, et n est le nombre de dimensions.
Le minimum global de cette fonction se trouve à $(x, y) = (0, 0)$, où $f(x, y) = 0$.

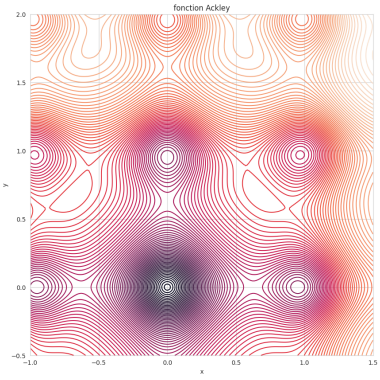


FIGURE 15 – Visualisation 2D d'Ackley

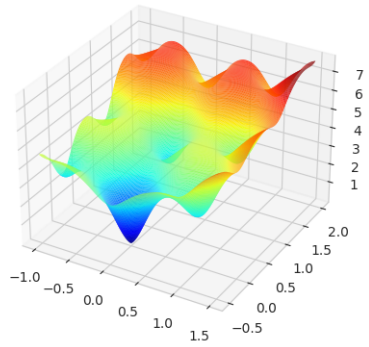


FIGURE 16 – Visualisation 3D d'Ackley

Ainsi, pour comparer les performances des deux méthodes étudiées, nous les appliquerons sur notre classe de fonctions et étudierons dans un premier temps le nombre d'itération minimal que chacune effectue, et dans un second temps le temps que chacune prend, pour converger.

L'étude de la convergence s'effectuera par l'évaluation de la distance entre les points calculés et le point optimal exact. La distance utilisée est la norme euclidienne, soit la norme L2, de la différence entre les points.

4.3 Applications et Résultats

Nous avons ainsi tout d'abord étudié les performances itératives de la méthode de Fletcher-Reeves (FR) et de Polak-Ribière (PR) sur les différentes fonctions non-convexes de notre classe de fonction.

Pour la fonction **Rosenbrock**, nous avons remarqué que la méthode FR converge plus vite que la méthode PR, ayant une distance proche de 0 à la 40ème itération, comparé à la méthode PR atteignant la même distance en 70 itérations.

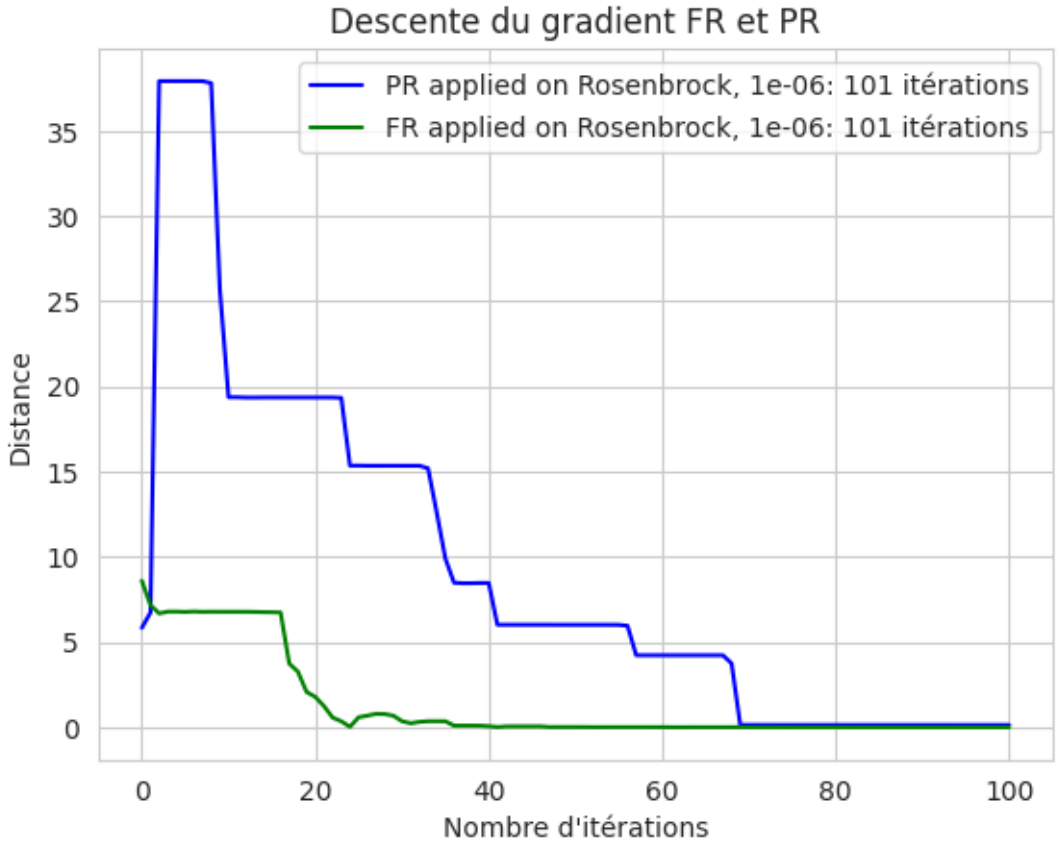


FIGURE 17 – Optimisation par les méthodes de Fletcher-Reevees et de Polak-Ribière de la fonction Rosenbrock (itérations)

Pour la fonction Rastrigin, nous avons remarqué que le choix de la méthode n'influençait pas significativement la vitesse de convergence en termes du nombre d'itérations. Les deux méthodes convergent rapidement et atteignent un minimum global en un nombre similaire d'itérations.

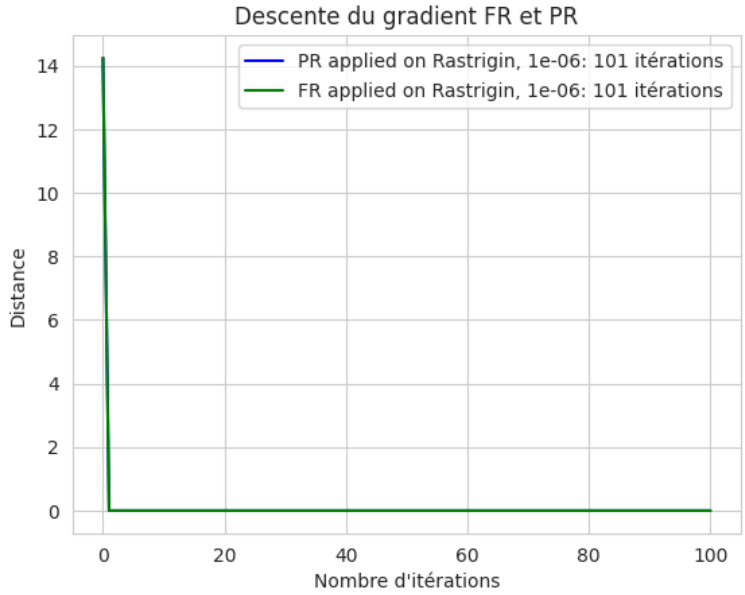


FIGURE 18 – Optimisation par les méthodes de Fletcher-Reevees et de Polak-Ribière de la fonction Rastrigin (itérations)

Egalement, pour la fonction Ackley, nous avons remarqué que le choix de la méthode n'influençait pas significativement la vitesse de convergence en termes de nombre d'itération, convergeant toutes les deux vers le point optimal en approximativement 8 itérations.

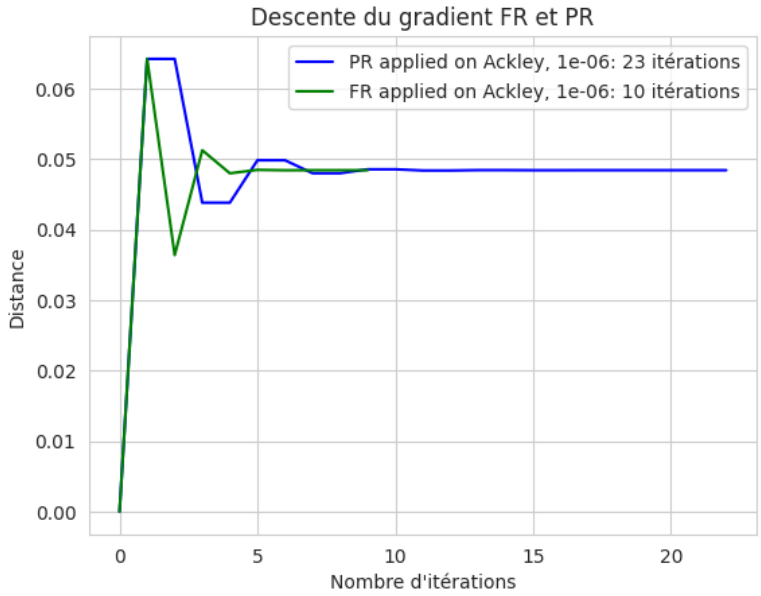


FIGURE 19 – Optimisation par les méthodes de Fletcher-Reevees et de Polak-Ribière de la fonction Ackley (itérations)

Nous avons donc remarqué que le choix de la méthode pouvait influencer la vitesse de convergence de nos fonctions, mais l'influence ne semble pas être importante étant donné que la majorité des fonctions de notre classe ne sont pas sensibles à cette condition.

Nous avons ensuite étudié les performances temporelles de la méthode de Fletcher-Reeves (FR) et de Polak-Ribière (PR) sur les différentes fonctions non-convexes de notre classe de fonction.

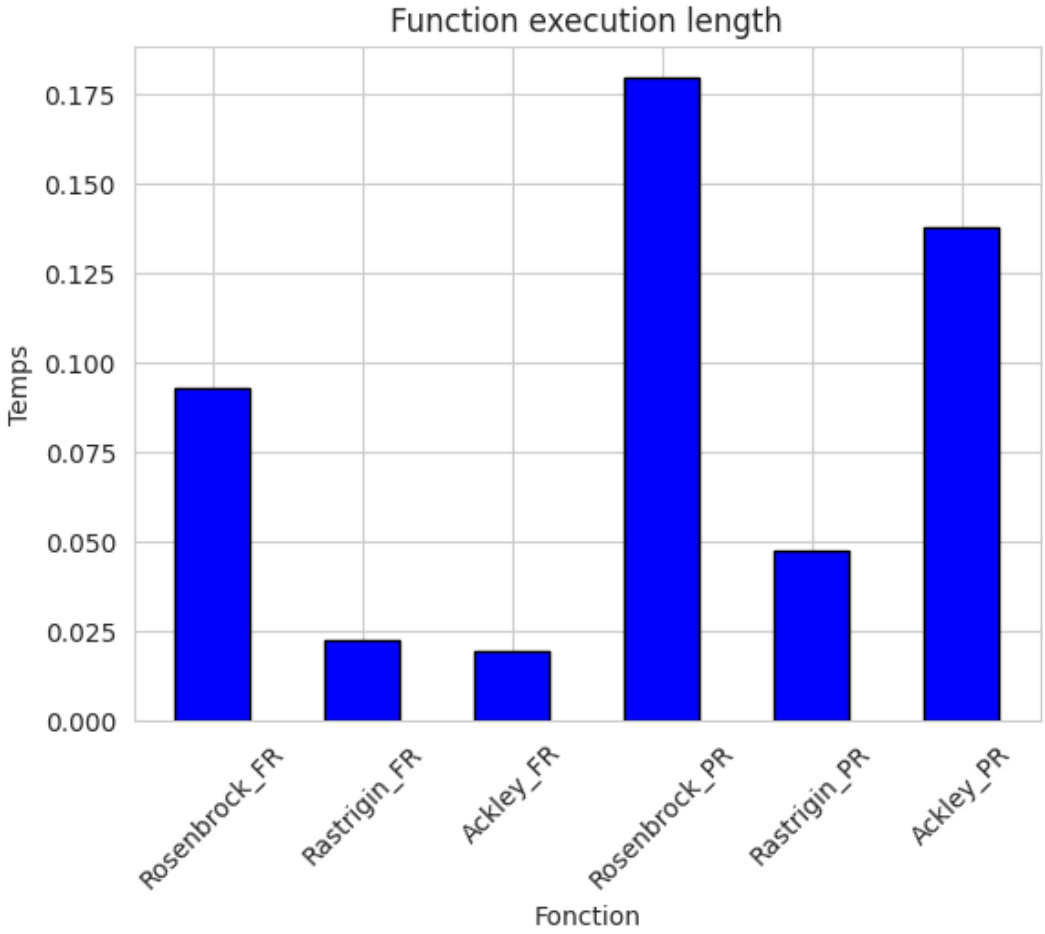


FIGURE 20 – Optimisation par les méthodes de Fletcher-Reevees et de Polak-Ribière de fonctions non-convexes (temps)

Nous avons enfin remarqué que la méthode FR semble tendre plus vite en moyenne que la méthode PR, en 0.05 seconde pour la méthode FR contre 0.12 seconde pour la méthode PR.

La méthode FR converge en effet plus vite que la méthode PR sur les 3 fonctions de notre classe, les différences temporelles étant souvent conséquentes, avec une différence de 0.12 secondes entre la méthode PR et la méthode FR appliquées à la fonction Ackley.

Nous constatons donc que le choix de la méthode peut avoir un impact significatif sur les performances de convergence, avec une preuve modérée par l'étude itérative et une preuve forte par l'étude temporelle.

5 Conclusion

En conclusion, le choix de la taille du pas dans la descente de gradient joue un rôle crucial dans la convergence de l'algorithme. Un pas plus grand permet d'avancer plus rapidement vers le minimum, mais il risque de le dépasser. En revanche, un pas plus petit garantit une approche plus précise du minimum, mais peut nécessiter un plus grand nombre d'itérations.

Des preuves expérimentales, comme celles démontrées avec la fonction quadratique, montrent l'influence de la taille du pas sur le nombre d'itérations nécessaires pour converger. Lorsque la taille du pas est trop grande, l'algorithme peut osciller autour du minimum sans converger.

Pour remédier à ce problème, il est possible d'utiliser une taille de pas optimale à chaque itération, comme la méthode du rebroussement avec le critère d'Armijo. Cette approche permet de faire de plus grands pas au début de la descente de gradient pour une convergence plus rapide, tout en réduisant progressivement la taille du pas pour garantir une progression plus précise vers le minimum.

Le choix de la méthode de descente de gradient dans l'algorithme de gradient conjugué pour l'optimisation non convexe est important. Les méthodes Fletcher-Reeves et Polak-Ribière utilisent des règles de mise à jour différentes. Leur efficacité varie en fonction de la fonction spécifique à optimiser. Dans nos tests, Fletcher-Reeves a été plus rapide pour la fonction Rosenbrock, tandis que pour les fonctions Rastrigin et Ackley, la méthode n'a pas eu d'impact significatif. Le choix dépend donc du problème spécifique.

En résumé, le choix de la taille du pas et de la méthode utilisée dans l'algorithme du gradient conjugué a un impact significatif sur la convergence et les performances des algorithmes d'optimisation. Une réflexion approfondie et des expérimentations sont nécessaires pour trouver la combinaison optimale pour un problème spécifique.