OCVX

2024 - IMAGE

TP 1 : Benchmark des méthodes de descente

Étudiants :

Théo Bonzi
Marc Lagoin
Daniel Rosa

Enseignant :

Guillaume Tochon

14 juin 2023

Table des matières

1 Introduction

Dans le domaine de l'optimisation numérique, la méthode de descente de gradient est un algorithme essentiel. Plusieurs aspects de cet algorithme influencent son comportement et ses performances, et sont contrôlés par des paramètres spécifiques. Dans ce rapport, nous allons analyser l'influence de trois de ces paramètres : le pas, un hyperparamètre du critère d'Armijo, et le critère d'arrêt.

Dans la première section, nous allons explorer l'influence du pas sur le nombre d'itérations de la méthode de descente de gradient dans le cas d'un pas constant. Nous examinerons comment le choix du pas se reflète dans le comportement de l'algorithme et comment il s'aligne avec ce que prédit la théorie.

Dans la deuxième section, nous étudierons l'influence des hyperparamètres beta et alpha spécifique au critère d'Armijo, lors de l'application de la méthode de descente de gradient à une fonction convexe. Le critère d'Armijo est une règle utilisée pour choisir le pas à chaque itération de l'algorithme de manière à garantir une certaine diminution de la fonction à chaque étape. Les hyperparamètres que nous allons examiner contrôlent à quel point cette diminution doit être significative.

Enfin, dans la troisième section, nous nous pencherons sur l'influence du critère d'arrêt sur le nombre d'itérations de la méthode de descente. Le critère d'arrêt est un seuil définissant à quel point la solution doit être "proche" de l'optimum pour que l'algorithme s'arrête. Nous allons étudier comment le choix de ce critère influence le comportement de l'algorithme.

2 Découpage des taches

Influence du pas sur le nombre d'itérations dans le cas de la descente de gradient à pas constant et adéquation avec la théorie
- Réalisé par Marc Lagoin et Théo Bonzi

Influence des hyperparamètres du critère d'Armijo dans le cas d'une descente de gradient pour une fonction convexe
- Réalisé par Théo Bonzi et Daniel Rosa

Influence du critère d'arrêt sur le nombre d'itérations de la méthode de descente
- Réalisé par Daniel Rosa et Marc Lagoin

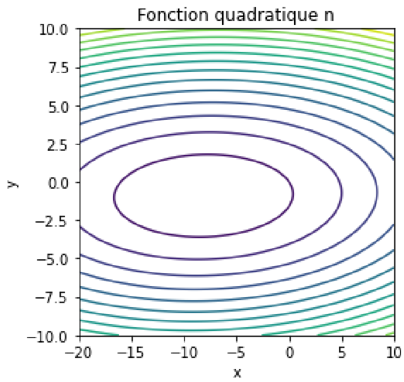
3 Benchmark 1 : Influence du pas sur le nombre d'itérations dans le cas de la descente de gradient à pas constant

3.1 La descente à pas constant

Cette méthode de descente est la plus simple des méthodes car elle ne nécessite aucun calcul du pas. L'itération est donnée par $x_{k+1} = x_k + \eta_k d_k$ avec η_k le pas constant.

3.2 Le nombre d'itérations pour les fonctions quadratiques

Pour l'analyse des fonctions convexes, nous avons choisi la fonction quadratique de différentes dimensions.

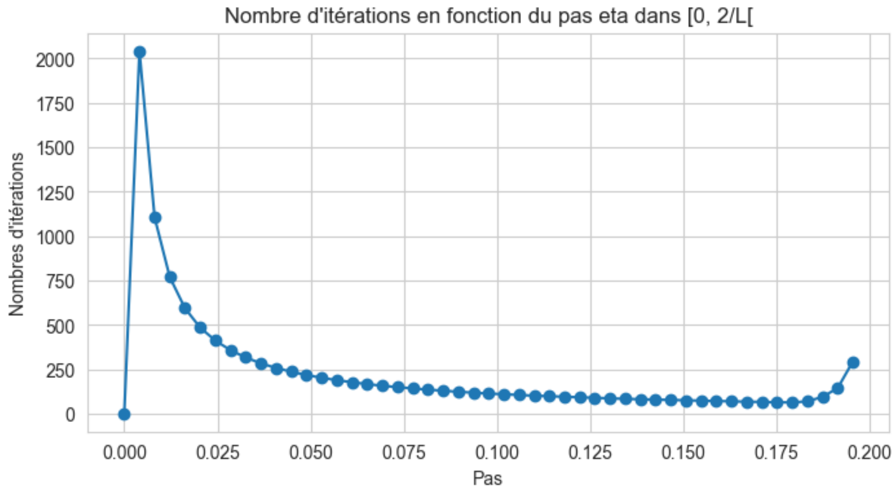


Pour une fonction quadratique de la forme $f(x) = \frac{1}{2}x^T Ax - b^T x$, où A est une matrice symétrique positive définie, la constante de Lipschitz L du gradient de la fonction est égale à la plus grande valeur propre de la matrice A.

Le gradient de la fonction f est $\nabla f(x) = Ax - b$, et sa constante de Lipschitz L est la plus grande valeur par laquelle la norme du gradient peut changer pour un petit changement dans x. Comme A est symétrique positive définie, elle a toutes ses valeurs propres réelles et positives. Ainsi, L peut être défini comme la plus grande valeur propre de A.

On sait théoriquement qu'un pas $\eta_k < 2/L$ garanti une convergence de la descente. Observons le nombres d'itérations en fonction du pas η_k dans l'intervalle [0, 2/L[.

Pour la fonction quadratique de dimension 2.



Les résultats obtenus sont très similaires quelque soit la dimension de la fonction quadratique On remarque que lorsque η_k est très petit, le nombre d'itérations nécessaire pour atteindre la convergence est extrêmement élevé (2042 itérations pour la deuxième valeur de η_k). Cela est attendu car avec un petit pas, l'algorithme avance très lentement vers le minimum.

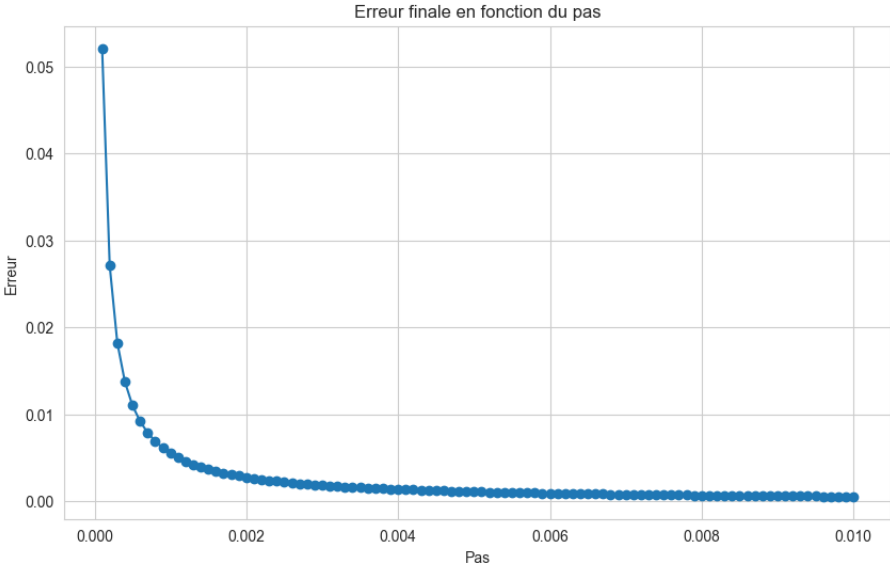
À mesure que η_k augmente, le nombre d'itérations nécessaire pour atteindre la convergence diminue rapidement, atteignant un minimum de 64 itérations. Cela suggère qu'il existe une valeur optimale de η_k dans cet intervalle qui minimise le nombre d'itérations nécessaire pour atteindre la convergence.

Cependant, après ce point, le nombre d'itérations commence à augmenter à nouveau, atteignant 295 itérations pour la dernière valeur de η_k. Cela suggère que lorsque η_k est trop grand, l'algorithme peut commencer à "rebondir" autour du minimum et à converger plus lentement.

Ce "rebond" correspond au résultat attendu car la valeur optimale théorique de η_k est 1/L. Ce qui explique que passé cette valeur le nombre d'itération sera croissant jusqu'à ce que η_k atteigne 2/L.

3.3 L'erreur finale en fonction du pas

Pour la fonction quadratique de dimension 2.



Lorsque le pas est très petit, l'erreur finale tend à être faible. Cela est dû au fait que l'algorithme avance lentement vers le minimum, permettant des itérations plus précise de l'espace de la fonction. Cependant, cela se fait au détriment d'un grand nombre d'itérations.

Quand le pas augmente, l'erreur finale commence à augmenter. Cela est dû au fait que l'algorithme fait de plus grands sauts dans l'espace de la fonction, ce qui peut entraîner un dépassement du minimum et une convergence vers une solution moins précise.

Cependant, il faut remarquer qu'il n'y a pas de corrélation parfaite entre le pas et l'erreur finale. En fonction de la nature spécifique de la fonction et du point de départ, un pas plus grand peut parfois conduire à une erreur finale plus faible. Cela suggère que le choix optimal du pas peut nécessiter une certaine exploration et ne peut pas être déterminé uniquement sur la base de considérations théoriques.

3.4 Conclusion sur le benchmark sur le pas constant η_k

L'étude de l'influence du pas sur le nombre d'itérations dans la descente de gradient à pas constant a révélé plusieurs points clés.

Premièrement, le choix du pas a un impact significatif sur la vitesse de convergence de l'algorithme. Un pas trop petit peut entraîner une convergence très lente. À l'inverse, un pas trop grand peut provoquer une divergence de l'algorithme.

Deuxièmement, il existe un intervalle optimal de valeurs de pas qui permettent à l'algorithme de converger le plus rapidement. Cet intervalle dépend de la fonction spécifique à minimiser et du point initial choisi.

Enfin, il est important de noter que même si un pas plus grand peut permettre une convergence plus rapide en termes de nombre d'itérations, cela ne garantit pas nécessairement une solution plus précise. En effet, un pas plus grand peut également entraîner une plus grande erreur finale, c'est-à-dire une plus grande différence entre la solution obtenue par l'algorithme et la solution optimale.

Ainsi, le choix du pas dans la descente de gradient à pas constant est un compromis entre la vitesse de convergence et la précision de la solution finale.

4 Benchmark 2 : Optimisation des hyperparamètres pour le critère d'Armijo

4.1 Critère d'Armijo

Le critère d'Armijo est un algorithme utilisée pour choisir, à chaque itération, un pas approprié. La règle assure que le pas choisi est suffisamment grand pour assurer un progrès significatif vers le minimum de la fonction, tout en étant suffisamment petit pour éviter un dépassement.

Voici un pseudo-code du critère d'Armijo :

Algorithm 1 Critère d'Armijo pour le choix du pas

```
procedure ARMIJO(x, d, f, ∇f, α, β)
    t ← 1
    while f(x + td) > f(x) + αt∇f(x)^T d do
        t ← βt
    end while
    return t
end procedure
```

Dans cet algorithme, x est le point courant, d est la direction de descente, f est la fonction objectif, ∇f est le gradient de la fonction objectif, et α et β sont des hyperparamètres donnés.

4.2 Les Hyperparamètres α et β

Le critère d'Armijo dépend de deux hyperparamètres, α et β. Ceux-ci ont des rôles spécifiques et leur choix peut avoir un impact sur la performance de l'optimisation.

4.2.1 α : Le Paramètre de Tolérance

Le paramètre $\alpha \in (0, 1)$ détermine le degré de réduction de la fonction objectif nécessaire pour qu'un pas soit considéré comme "suffisant". Un α faible permet un pas plus grand, ce qui peut accélérer la convergence mais aussi augmenter le risque de dépassement. Inversement, un α élevé demande une réduction plus importante de la fonction objectif, ce qui peut garantir une meilleure qualité d'optimisation mais aussi ralentir la convergence.

4.2.2 β : Le Paramètre de Réduction de Pas

Le paramètre $\beta \in (0, 1)$ est utilisé pour réduire le pas quand la condition d'Armijo n'est pas satisfaite. Un β faible réduit fortement le pas à chaque itération, ce qui peut aider à éviter les dépassements mais aussi ralentir la convergence. Un β élevé réduit moins le pas, ce qui peut accélérer la convergence mais aussi augmenter le risque de dépassement.

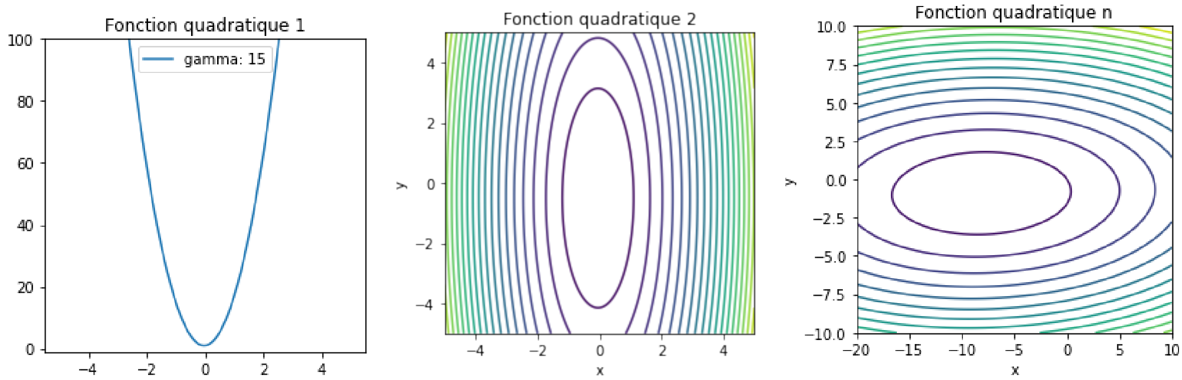
4.3 Métriques d'Évaluation

Pour évaluer l'influence des hyperparamètres α et β sur l'efficacité de l'optimisation, nous utilisons deux métriques principales :

1. Le nombre d'itérations nécessaires pour atteindre la convergence. Cette métrique donne une idée de la vitesse de convergence de l'algorithme.
2. Le temps de convergence, qui mesure le temps réel nécessaire pour atteindre la convergence. Cette métrique tient compte de la fois le nombre d'itérations et le temps de calcul de chaque itération.

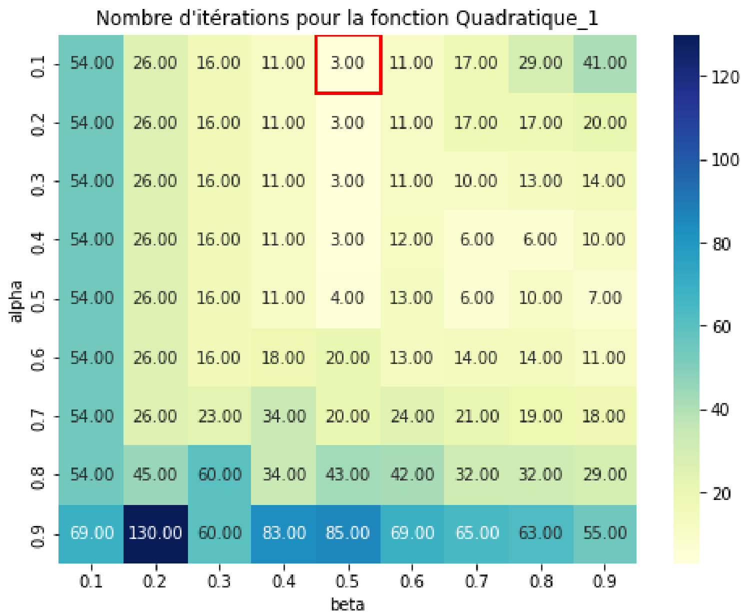
4.4 Analyse

Pour cette analyse, nous allons utiliser 3 fonctions convexes quadratiques ci-dessous :



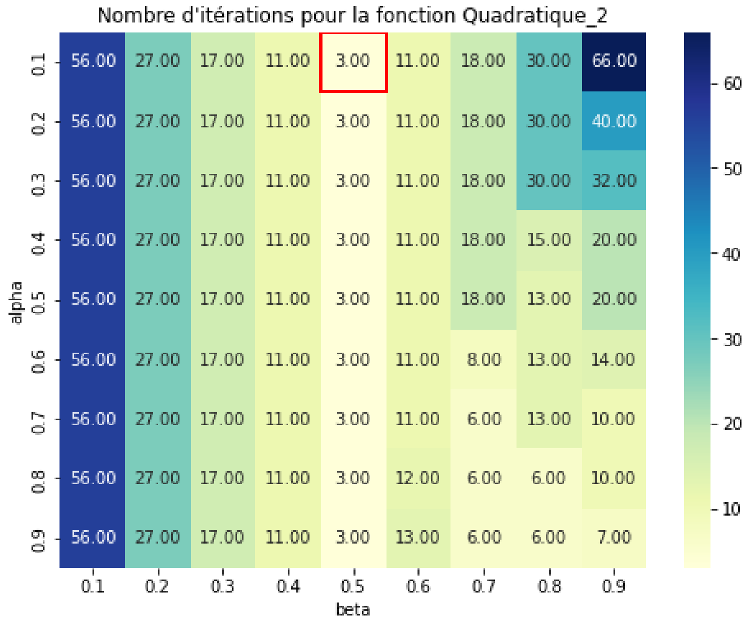
Pour chacune de ses fonctions, nous allons mesurer l'impact de alpha et beta sur le nombre d'itérations. Voici nos résultats.

Pour la fonction quadratique 1, nous pouvons remarquer que l'influence d'alpha est moins significative que celle de beta lorsque beta prend des valeurs plus élevées, notamment entre 0.7 et 0.9. Toutefois, pour réduire le nombre d'itérations, nous pouvons observer qu'un alpha plus faible conduit à une diminution du nombre d'itérations. En examinant attentivement le tableau, nous pouvons identifier la paire de valeurs optimale pour cette fonction quadratique 1, à savoir un alpha de 0.1 et un beta de 0.5. Ces valeurs sont mises en évidence en rouge dans le tableau des résultats.

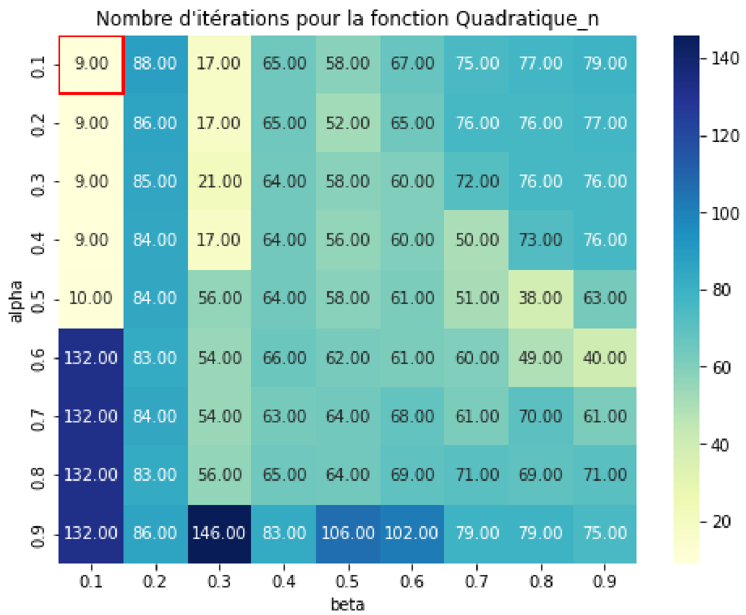


Pour la fonction quadratique 2, nous pouvons remarquer que l'alpha ne permet pas de réduire le nombre d'itérations, sauf lorsque beta atteint les valeurs 0.7, 0.8 ou 0.9.

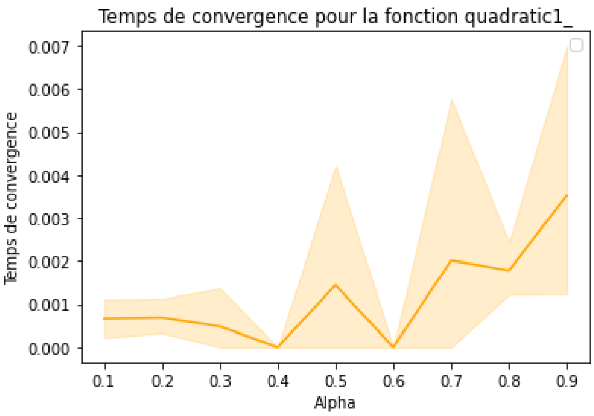
En examinant le tableau, nous pouvons également identifier la paire de valeurs optimale, indiquée en rouge, qui s'avère être un alpha de 0.1 et un beta de 0.5. Cependant, il est important de noter que pour cette fonction, la valeur de l'alpha n'a pas d'importance tant que le beta est inférieur à 0.7. Ainsi, nous aurions pu choisir n'importe quelle valeur d'alpha pour un beta inférieure à 0.7 et obtenir des résultats similaires. Néanmoins, dans notre cas, la paire de valeurs optimale que nous prenons est un alpha de 0.1 et un beta de 0.5.

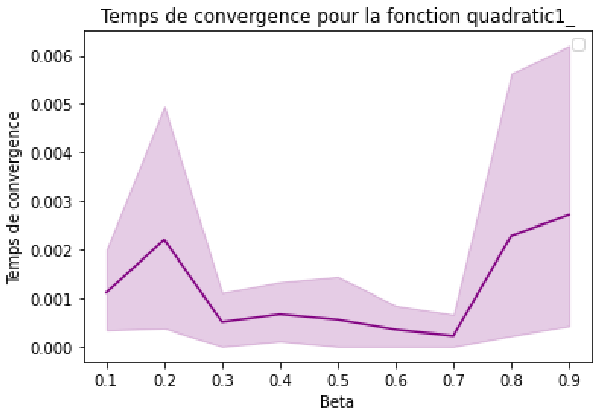


En ce qui concerne la fonction Quadratique n, nous remarquons un changement significatif du nombre d'itérations pour un alpha à 0.5. Pour un beta fixé à 0.1, le nombre d'itération passe de 132 pour alpha à 0.6 à 10 pour un alpha à 0.5. En analysant le tableau, nous pouvons repérer la paire de valeurs optimale, marquée en rouge. Dans ce cas, l'alpha optimal est de 0.1 et le beta optimal est de 0.1. Ces valeurs permettent d'obtenir le nombre d'itérations le plus faible pour cette fonction quadratique n.

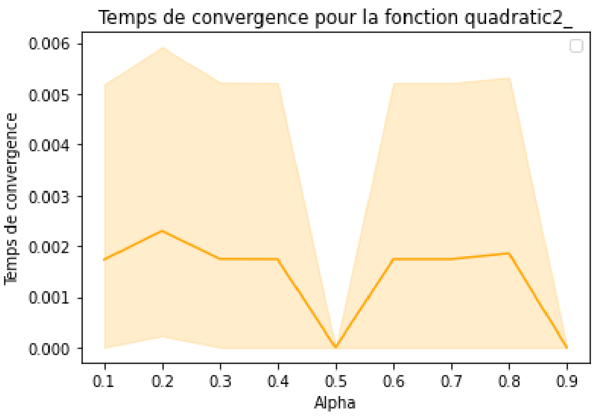


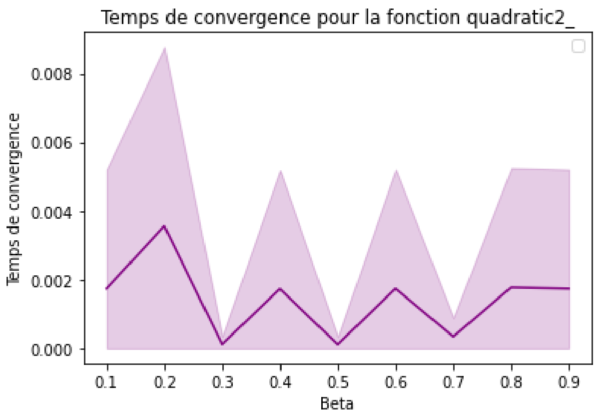
Nous allons maintenant analyser le temps de convergence qui prend en compte le nombre d'itérations et le temps d'execution pour converger.



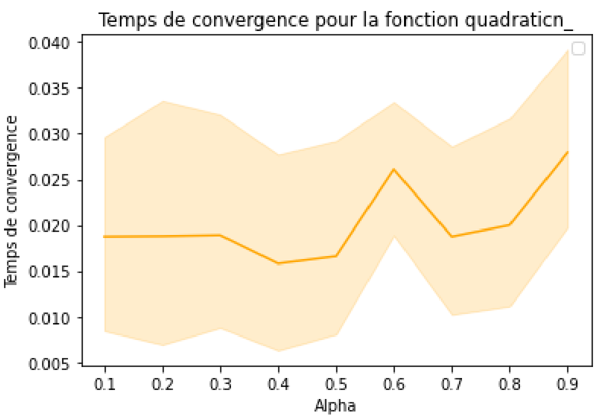


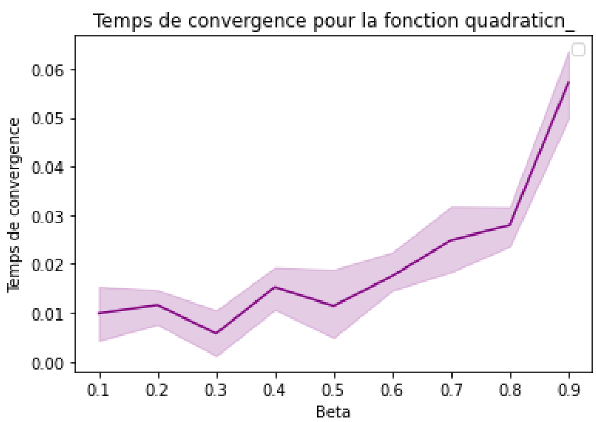
1. Alpha : Pour ce graphique, nous pouvons remarquer une tendence haussière pour un alpha qui augmente. En effet un alpha plus petit permet de réduire le pas de manière plus significative.
2. Beta : Nous pouvons remarquer sur le graphique du paramètre beta deux pices à 0.2 et 0.8 qui font augmenter le temps d'exectution et le nombre d'itérations.





1. Alpha : Pour ce graphique, nous remarquons 2 pics vers le bas aux valeurs 0.5 et 0.9.
2. Beta : Ici nous pouvons voir une tendance ou le temps de convergence baisse significativement aux valeurs impaires.





Ici pour alpha et beta nous pouvons remarquer une tendance haussière lorsque nous augmentons le paramètre alpha et beta indépendamment.

4.5 Conclusion du benchmark sur le critère d'Armijo

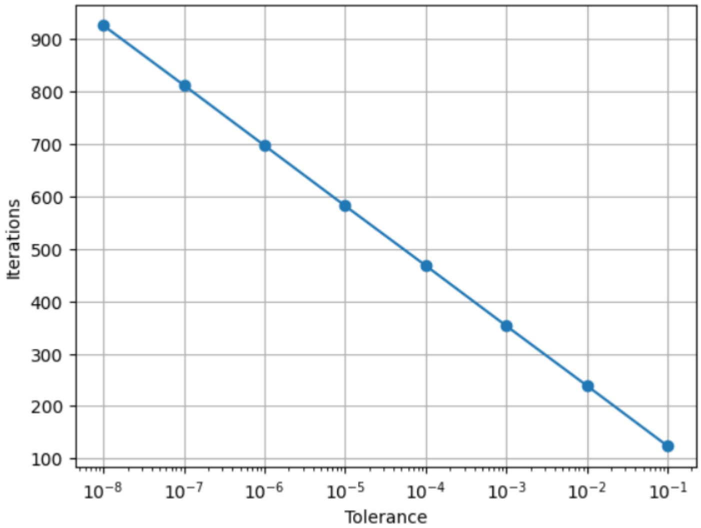
Pour conclure, nous pouvons remarquer que alpha et beta influent sur le nombre d'itérations et le temps de convergence. Plus nous prenons un alpha petit, plus le temps et le nombre d'itérations est réduit. Pour les fonctions quadratique 1 et 2, nous remarquons que la paire de valeurs (alpha, beta) est presque la même pour avoir les meilleurs résultats. En revanche pour la fonction qudratique n, nous remarquons que nous avons besoin d'un beta plus petit aux alentour de 0.1 pour avoir le meilleur résultats.

Function	**Best Alpha for Iterations**	**Best Beta for Iterations**	**Best Alpha for Convergence**	**Best Beta for Convergence**
quadratic1_	0.1	0.5	0.1	0.3
quadratic2_	0.1	0.5	0.1	0.5
quadraticn_	0.1	0.1	0.2	0.1

5 Benchmark 3 : Critère d'arret vs Nombre de itérations

5.1 Fonction quadratique dans R

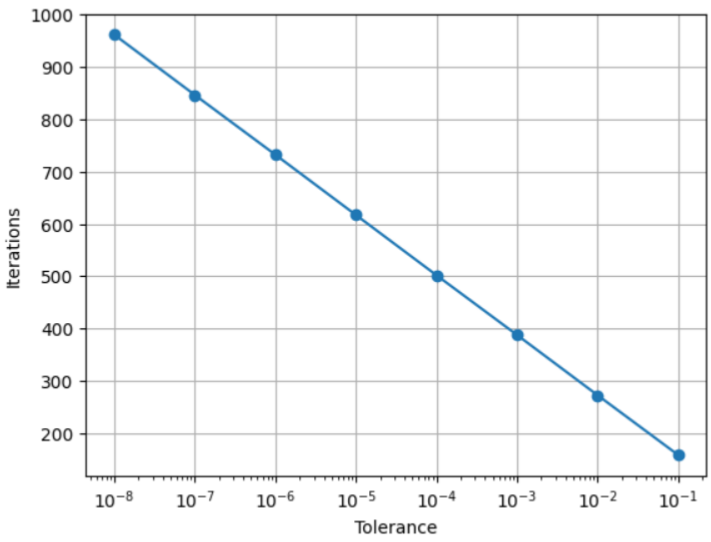
Formule : quadratic1(x, γ) = $\gamma(x^2)$ + x + 1
$\gamma = 0.5$



Ce graphique graphique représente la relation entre le nombre d'iterations et la tolérance pour la descente de gradiant de la fonction quadratique unidimensionnelle (R). Il illustre le nombre d'itérations nécessaires à la convergence de l'algorithme de descente de gradient pour différentes conditions d'arrêt, autrement dit : la tolérance. Le graphique montre que, à mesure que les critères d'arrêt deviennent plus stricts, le nombre d'itérations requis pour atteindre la convergence augmente. Ceci est attendu, car un critère d'arrêt plus strict exige une approximation plus proche du minimum, ce qui nécessite davantage d'itérations.

5.2 Fonction quadratique dans R^2

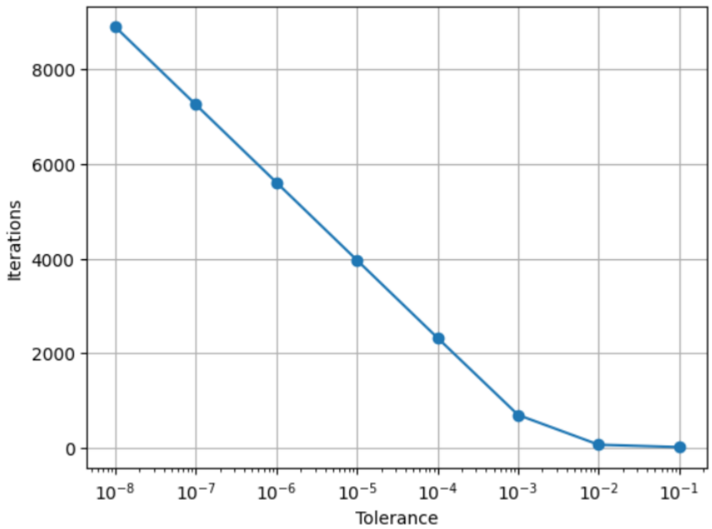
Formule : quadratic2(x, γ) = quadratic1(x_0, γ_0) + quadratic1(x_1, γ_1)
$\gamma = 0.5$



Ce deuxième graphique graphique représente la relation entre le nombre d'iterations et la tolérance pour la descente de gradiant de la fonction quadratique bidimensionnelle (R^2). Comme précédemment, on peut voir que à mesure que les critères d'arrêt deviennent plus stricts, le nombre d'itérations augmente. Cependant, le nombre d'itérations spécifique augmente légèrement par rapport au cas unidimensionnel en raison de la complexité supplémentaire de l'optimisation en deux dimensions. Malgré cette complexité supplémentaire, si la fonction est toujours séparable (ce qui est le cas ici), la fonction en 2D se comporte de manière similaire à la fonction en 1D, car elle optimise chaque dimension indépendamment.

5.3 Fonction quadratique dans R^n (n=10)

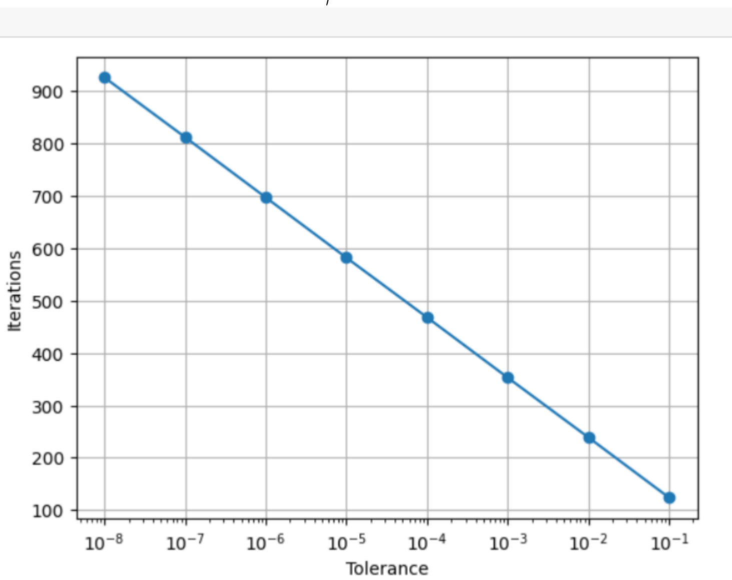
Formule : quadraticn(x) = $\frac{x^T A x}{2}$ - b^T x



Ce troisième graphique représente encore une fois la relation entre le nombre d'iterations et la tolérance pour la descente de gradiant de la fonction quadratique en dix dimensions (R^10). On observe que des critères d'arrêt plus stricts nécessitent un nombre supérieur d'itérations pour atteindre la même tolérance que pour la function en dimension 1 et 2. Cette augmentation du nombre d'itérations est due à la complexité accrue du problème dans un espace de dimension supérieure. Cependant, malgré cette complexité supplémentaire, le comportement de la fonction quadratique en 10 dimensions reste similaire à celui des cas 1D et 2D. Cela est le cas car la fonction quadratique est séparable lorsque la matrice A est symétrique, ce qui est le cas ici.

5.4 Fonction cubique dans R

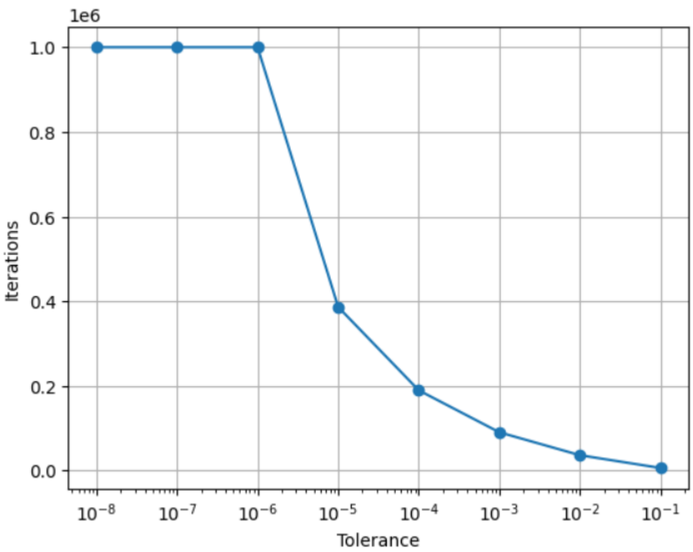
Formule : cubic1(x, γ) $= x^3 + \gamma x^2 + x + 1$
$\gamma = 0.5$



Ce graphique graphique représente la relation entre le nombre d'iterations et la tolérance pour la descente de gradiant de la fonction Cubic1. On observe encore une fois que lorsque nous diminuons la tolérance, le nombre d'itérations nécessaires pour que l'algorithme converge augmente linéairement. C'est le comportement attendu. L'algorithme de descente de gradient effectue davantage d'itérations pour trouver une solution qui est "plus proche" de la solution optimale, tout comme pour la fonction quadratique.

5.5 Fonction cubique dans R^2

Formule : cubic2(x, γ) $= x^3 + \gamma x^2 + x + 1$
$\gamma = 0.5$



Ce graphique graphique représente la relation entre le nombre d'iterations et la tolérance pour la descente de gradiant de la fonction Cubic2. Ici, la situation devient plus complexe. Pour des tolérances supérieures à 10^-6, le nombre d'itérations semble augmenter exponentiellement à mesure que la tolérance diminue, ce qui indique que l'optimisation devient plus difficile. Ceci pourrait être dû à la complexité de la fonction dans l'espace bidimensionnel. Lorsque la tolérance est de 10^-6 ou encore plus petite, les itérations requises dépassent les limites du graphique. Cela suggère que l'algorithme nécessite un nombre d'itérations nettement plus élevé pour converger. Il aurrait fallu lancer l'algoritme avec un nombre d'itérations maximales supérieur de 1000000 ou plus mais cela aurrrait pris bien trop de temps. Cependant, nous observons toujours le même comportement que auparavant, bien que de façon plus exagérée et exponentielle : plus on on cherche a avoir une tolerance faible, plus le nombre d'iterations va être élevé. Et plus la fonction va être complexe, plus le nombre d'itérations va être élevé pour la même tolerance.

5.6 Critique des résultats

Finalement, cette analyse a permis d'étudier l'influence du critère d'arrêt sur le nombre d'itérations nécessaires à la convergence de l'algorithme de descente de gradient. Il en ressort que plus le critère d'arrêt est strict (c'est-à-dire que la tolérance est faible), plus le nombre d'itérations requis pour atteindre la convergence augmente. Cette relation est observée pour les deux types de fonctions étudiées (quadratique ou cubique) et pour toutes les dimensions étudiées (unidimensionnelle ou multidimensionnelle).
En conclusion, le critère d'arrêt a une influence inversement proportionelle notable sur le nombre d'itérations nécéssaires à la méthode de descente, particulièrement pour les fonctions plus complexes et multidimensionnelles. Cependant, dans la pratique, il est important de choisir une tolérance qui équilibre efficacement la précision de la solution et le coût en temps de calcul. Une tolérance trop stricte peut entraîner un grand nombre d'itérations et donc une durée d'exécution plus longue, pour au final un gain de précision marginal.