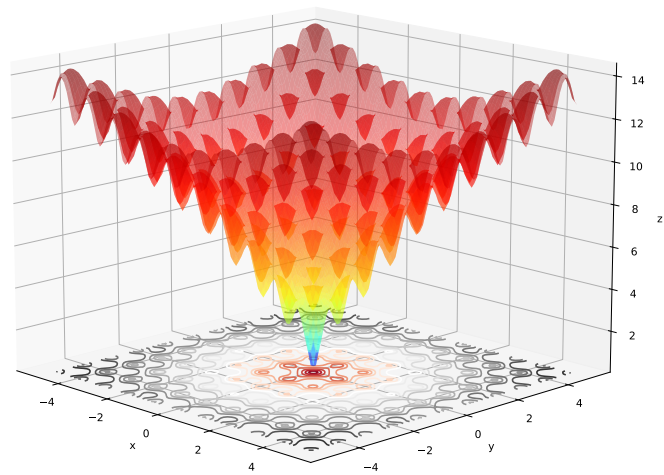


Descente de Gradient



ISC_423

Aliya Myaz

Omar Alkheja

Iliya Saroukhanian

17 juin 2024

Table des matières

1	Introduction	4
2	Expérimentation	5
2.1	Résumé des points abordés	5
2.2	Effet du taux d'apprentissage λ – descente simple	5
2.2.1	Conclusion intermédiaire	7
2.3	Effet du taux d'inertie γ – descente Momentum	7
2.3.1	Conclusion intermédiaire	8
2.4	Comparaison entre les méthodes Momentum et Nesterov	9
2.5	Méthode d'Adam	9
2.5.1	Variation du taux d'apprentissage λ – Adam	10
2.5.2	Comparaisons des descentes entre les quatre méthodes	11
2.5.3	Paramètres spécifiques	11
3	Pièges	13
3.1	Ravines	13
3.2	Plateaux	14
3.3	Minimums locaux	14
4	Descentes sur diverses fonctions	15
4.1	Le “puits”	15
4.1.1	Évolution de la valeur de la fonction de coût	16
4.2	Rosenbrock	16
4.3	Ackley	18

Liste des Figures

2.1	Variation de λ lors de la descente simple	6
2.2	Introduction du paramètre <i>momentum</i> γ	7
2.3	Descente Momentum, $\gamma = 0.9$	8
2.4	Trajectoires des descentes, Momentum vs Nesterov	9
2.5	Variation du taux d'apprentissage – Adam	10
2.6	Comparaisons des méthodes de descente de gradient	11
2.7	Influence des paramètres β_1 et β_2 sur la méthode d'Adam	12
3.1	Cas de divergence de la descente	13
3.2	Illustration du piège posé par un plateau	14
4.1	Descentes sur la fonction “puit”	15
4.2	Évolution de la valeur de la fonction de coût “puit”	16
4.3	Graphique des descentes sur la fonction de Rosenbrock	17
4.4	Descentes sur la fonction d'Ackley	18
4.5	Évolution de la valeur de la fonction d'Ackley à chaque itération	19

1 Introduction

Dans le cadre (subtilement élargi) de ce travail pratique, nous étions amenés à nous familiariser avec le concept de la descente de gradient dans le contexte de l'apprentissage machine et l'intelligence artificielle.

En effet, parmi d'autres utilités hautement majeures, la descente de gradient (le concept en lui-même ainsi que les divers algorithmes) est utilisée dans l'entraînement des réseaux neuronaux lors de la "rétropropagation" (*backpropagation*). En bref, la rétropropagation consiste à "remonter" dans le réseau depuis la couche de sortie en direction de la couche d'entrée. Cette remontée permet d'ajuster les poids des neurones ayant contribué à un résultat faussé de sorte à optimiser les performances du modèle. Cette correction consiste plus concrètement en l'optimisation des paramètres d'une fonction dite de "coût", qui représente l'écart entre les prédictions du réseau et les valeurs attendues. Il s'agit donc en fait d'un problème de minimisation d'erreur.

Or l'essence même de la descente de gradient réside dans sa capacité à orienter un processus d'optimisation vers un minimum (au moins local) en ajustant itérativement les paramètres d'un modèle ou variables d'une fonction. Cependant, ce voyage est jonché de multiples pièges que nous avons inévitablement rencontré dans notre exploration.

Car en effet, ayant compris l'importance de la descente de gradient, nous avons voulu à travers ce rapport (de notre propre volonté), explorer en détail différentes facettes de cet outil, de ses paramètres à ses implications pratiques, sur quelques fonctions dites de **test** qui permettent de mieux évaluer les diverses caractéristiques des algorithmes de descente. Perpendiculairement à la comparaison des fonctions, ce sont les méthodes de descente qui ont été confrontées, nous menant à méditer les subtilités de chacune d'entre elles.

Aussi, nous souhaitons-vous un bon voyage dans l'univers merveilleux de ce rapport immersif, et vous prions de vous laisser porter au-travers des dunes par les descentes de gradient, les plus rapides comme les plus erratiques et aventureuses.

2 Expérimentation

2.1 Résumé des points abordés

À présent, nous allons présenter les diverses expériences effectuées, notamment en ce qui concerne l'ajustement du *learning rate* (taux d'apprentissage) λ qui correspond donc à la taille du pas effectué à chaque itération des diverses méthodes de descente. Ceci sera illustré principalement à travers la descente **simple**.

Dans un second temps, nous présenterons l'impact du paramètre *momentum* (inertie) γ dans le cadre de la descente **Momentum**. À la suite de ceci, nous tenterons de faciliter la visualisation de la méthode **Nesterov** en la comparant immédiatement à la **Momentum**.

Par la suite, nous introduirons la méthode la plus élaborée des quatre présentées dans ce travail pratique, celle d'**Adam**. Initialement, nous allons tenter d'étudier l'effet du *learning rate* λ sur cet algorithme et les raisons pour lesquelles sa valeur devra diverger par rapport aux trois autres méthodes.

En suite, nous tenterons de jouer avec les deux paramètres spécifiques à **Adam**, β_1 et β_2 , afin d'essayer de visualiser la manière dont ces deux paramètres permettent de rendre le *learning rate* dynamique (c'est-à-dire l'ajuster en fonction des asymétries possiblement introduites par un gradient plus prononcé dans une des dimensions par rapport aux autres). Cette notion de taux d'apprentissage dynamique provient de l'algorithme **RMSPprop** (*Root Mean Square Propagation*) qui, en calculant la moyenne des carrés des composantes du gradient, permet d'établir un taux d'apprentissage variable par composante.

Les paramètres d'**Adam** permettent aussi d'émuler un comportement d'inertie présent dans les méthodes **Momentum** et **Nesterov** à l'aide d'un calcul des moyennes mobiles.

En somme, **Adam** est censé représenter le meilleur des deux mondes (taux d'apprentissage variable par composante et l'inertie qui permet de pouvoir passer au-delà de certains minima locaux dans le but d'en trouver un global).

2.2 Effet du taux d'apprentissage λ – descente simple

Pour pouvoir illustrer l'effet du taux d'apprentissage sur le comportement de la descente, nous avons pris trois valeurs de λ , séparées à chaque fois d'un ordre de grandeur pour rendre la visualisation plus claire. Les exemples ci-dessous ont été effectués sur la fonction f :

$$f(x, y) = x^2 + ky^2 \quad \forall k \in \mathbb{N} \quad (2.1)$$

n.b. : Le gradient ∇ de la fonction ci-dessous est accentué dans la direction de l'axe y à l'aide du facteur $k = 5$. Ceci aura son importance lors de l'explication de la problématique des ravines ainsi que de la visualisation de la méthode d'**Adam**.

Sur les figures ci-dessous nous pouvons voir l'effet du *learning rate* sur le nombre d'itérations de l'algorithme avant qu'on n'atteigne le minimum se situant en $f(0,0)$.

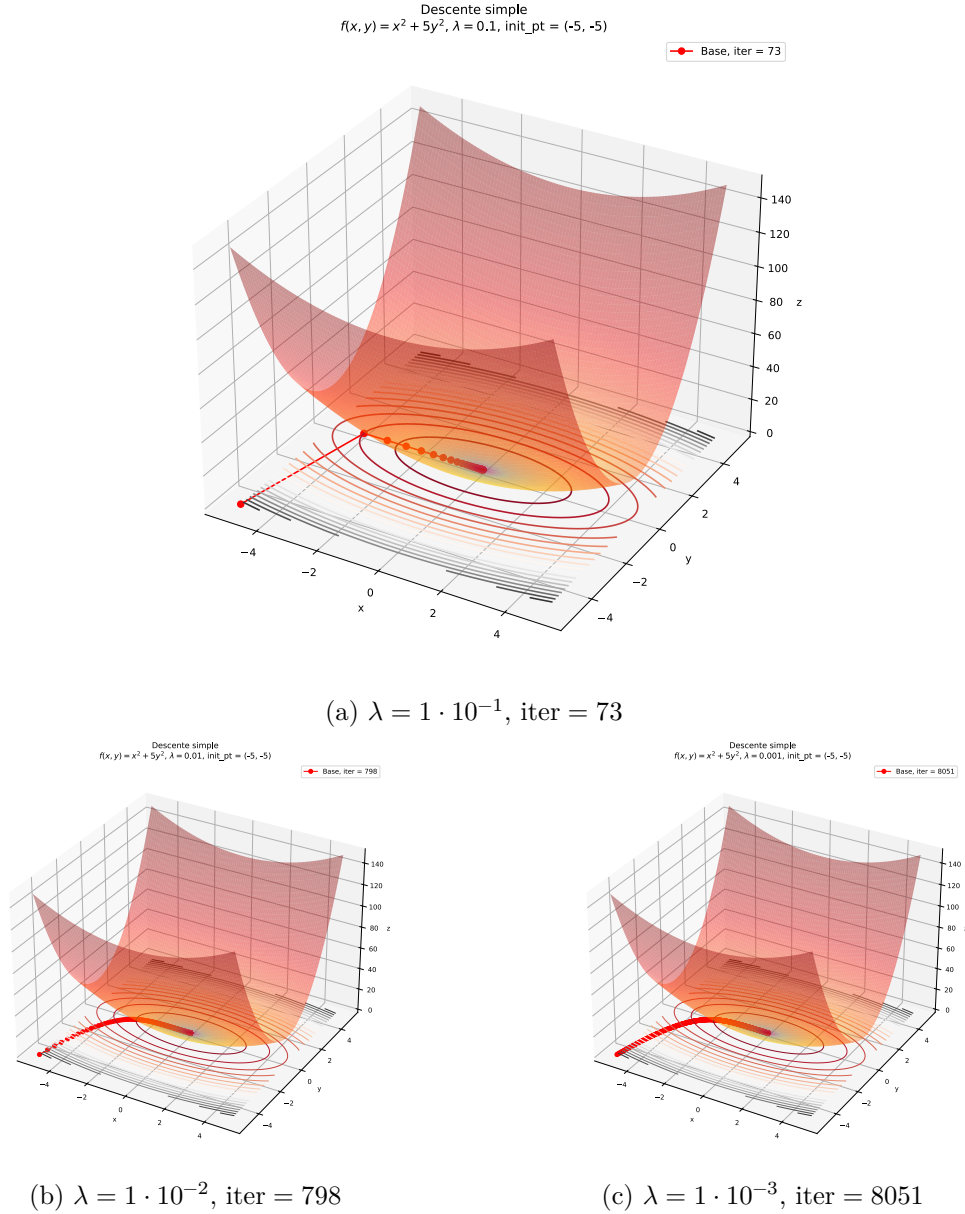


Figure 2.1: Variation de λ lors de la descente simple

Dans le cas de la Figure 2.1a, nous pouvons voir que suite aux grands pas effectués à chaque itération, la trajectoire est saccadée, même si le minimum est tout de même atteint. Le problème qui peut survenir suite à un λ si grand est le fait de potentiellement passer

au-delà d'un fossé qui puisse contenir le minimum recherché. Ce cas sera illustré plus tard.

2.2.1 Conclusion intermédiaire

A travers les exemples présentés sur la Figure 2.1, nous pouvons facilement se convaincre que le bon choix de la valeur du taux d'apprentissage, comme beaucoup de choses dans la vie, repose sur un compromis. La valeur de λ doit à la fois ne pas être trop petite (sinon le nombre d'itérations deviendra énorme), ni trop grande (risque de rater un minimum, en "sautant" par-dessus).

Dès lors, la valeur du taux d'apprentissage choisie pour les algorithmes de descente simple, Momentum et Nesterov sera de $\lambda = 1 \cdot 10^{-2}$.

2.3 Effet du taux d'inertie γ – descente Momentum

À présent, nous nous pencherons sur la première des trois déclinaisons des descentes "élaborées" en commençant par la méthode **Momentum**. Le but principal de celle-ci est d'introduire un second paramètre γ censé représenter la mémoire du "passé". Il est évident que la notion de "passé" est liée aux pas effectués précédemment. Ce paramètre permet donc principalement d'accentuer la taille du pas en fonction de la topologie dans son voisinage. De manière plus simple, cela signifie que si la pente est raide, on peut se permettre de faire un plus grand. Inversement, si le taux de variation est faible alors le pas devra être petit. Ceci permet de diminuer la quantité d'itérations nécessaires pour atteindre un minimum.

i Note

Une faible valeur de γ implique le fait qu'on porte peu d'importance aux pas précédents. En effet si $\gamma = 0$, alors on aboutit à une descente de gradient **simple**.

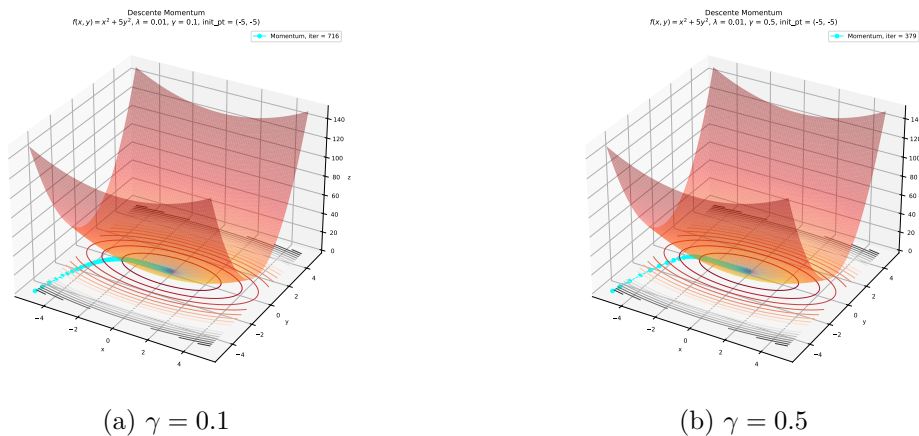


Figure 2.2: Introduction du paramètre *momentum* γ

Pour pouvoir mieux voir la différence avec la méthode simple, nous allons définir la valeur de $\gamma = 0.9$. Grâce à cette valeur nous pouvons à présent réellement apprécier la trajectoire totalement différente produite par la descente **Momentum**.

La trajectoire de la descente peut être assimilée à celle d’une bille qui roule le long d’un dénivelé. Lors d’une forte descente, la fréquence des “pas” est réduite car ceux-ci sont plus grands. La figure ci-dessous nous permet aussi de voir que suite aux grands pas effectués initialement, nous avons *overshoot* (en bon français) la zone en $y = 0$, d’où la “courbe” de correction tracée qui rappelle justement la trajectoire d’une bille.

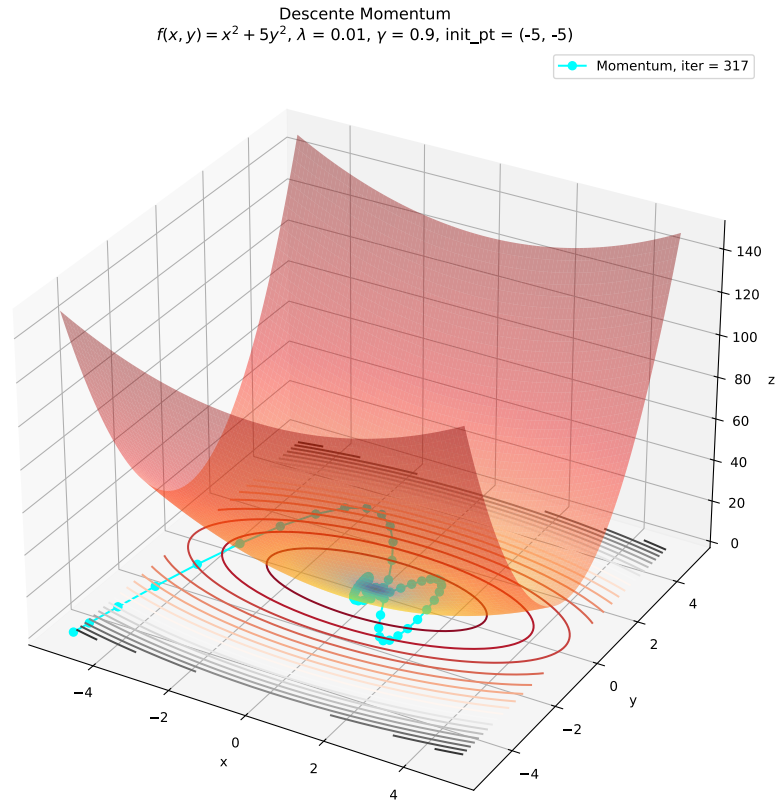


Figure 2.3: Descente Momentum, $\gamma = 0.9$

2.3.1 Conclusion intermédiaire

Dans cette partie nous avons pu voir la manière dont la méthode **Momentum** permet d’améliorer la descente de gradient. Grâce à l’inertie introduite à l’aide du paramètre γ , nous avons pu réduire d’un facteur de 2 (même un peu plus, ~ 2.517 pour être précis) le nombre d’itérations requises pour un même taux d’apprentissage de $\lambda = 0.01$. La méthode simple nécessitait 798 itérations contre les 317 de **Momentum**.

2.4 Comparaison entre les méthodes Momentum et Nesterov

Pour cette démonstration, nous avons dû réduire le canal α afin de diminuer l'opacité du graphique pour pouvoir mieux observer les deux trajectoires.

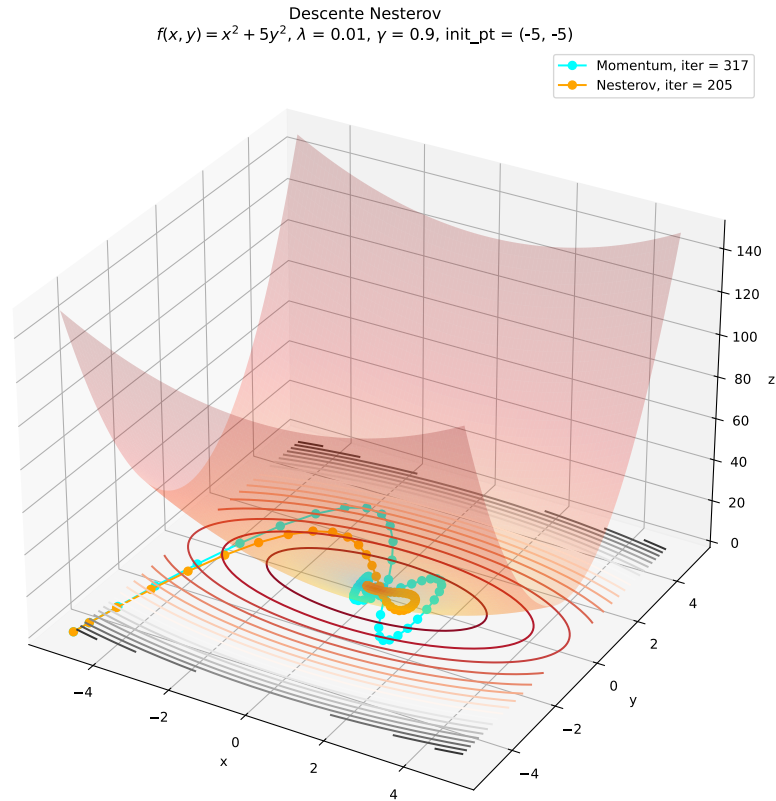


Figure 2.4: Trajectoires des descentes, Momentum vs Nesterov

La trajectoire empruntée par la descente **Nesterov** est *étrangement* similaire à celle de **Momentum** à un détail près, elle est **moins erratique**. Ceci est dû au fait qu'avant d'effectuer le pas (c'est-à-dire passer à la prochaine itération), **Nesterov** pré-calcule une approximation de $\nabla f(\vec{x}_{k+1})$ pour corriger le grand pas effectué par **Momentum** de sorte à pousser la trajectoire un peu plus tôt dans la bonne direction vers le minimum. La Figure 2.4 l'illustre très bien, les deux trajectoires ont les mêmes tendances sauf que l'orange (Nesterov) se "redresse" plus tôt et atteint le minimum plus rapidement que Momentum (205 contre 317 itérations).

2.5 Méthode d'Adam

Finalement, nous pouvons introduire la dernière des quatre méthodes nommée **Adam** (*Adaptive Momentum*). Les avantages principaux de cette méthode sont bien résumés dans

son nom. Essayons donc de le décortiquer :

- **Adaptive** → le principe “adaptatif” (ou dynamique) consiste à tenter d’adapter le taux d’apprentissage par rapport aux spécificités de chaque composantes / dimensions. En d’autres termes, le but est de diminuer le taux d’apprentissage pour les dimensions ayant un fort gradient et inversement, accentuer ce taux pour les dimensions ayant une faible variation. Ceci permet notamment de palier à certaines problématiques qui peuvent survenir lors d’une descente de gradient sur une fonction *non-isotropique* (possédant des asymétries au niveau des taux de variation de chaque dimension).
- **Momentum** → cet aspect, contrairement à celui énoncé précédemment, nous est à présent familier. En bref, il permet d’introduire un phénomène d’inertie en réduisant la fréquences des *pas* (en faisant des plus grands) lorsque le gradient présente une forte variation de sorte à accélérer la convergence vers un minimum.

2.5.1 Variation du taux d’apprentissage λ – Adam

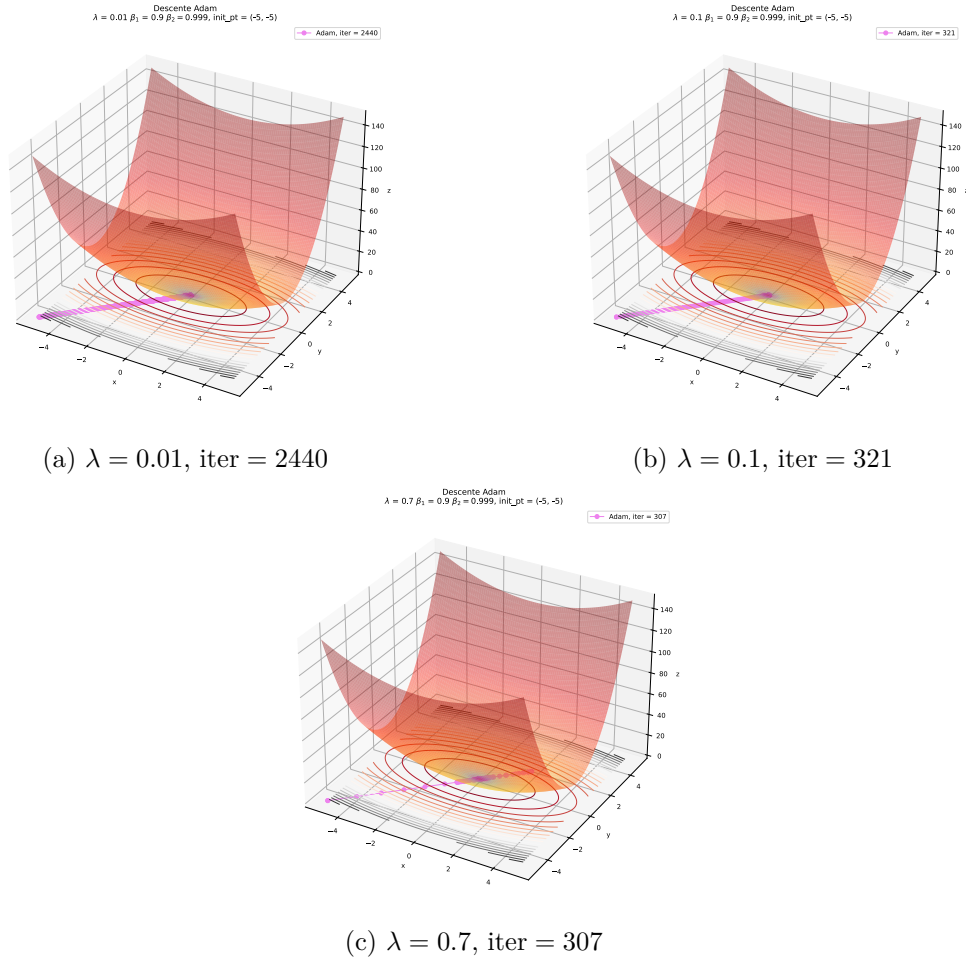


Figure 2.5: Variation du taux d’apprentissage – Adam

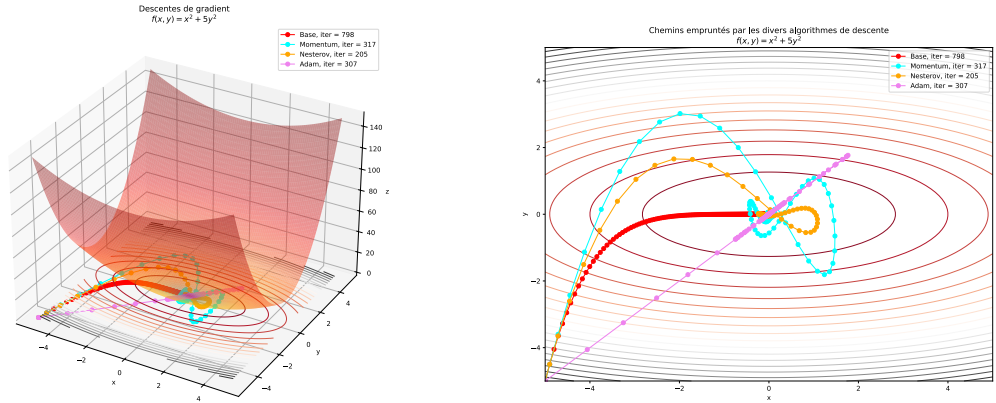
Grâce à la Figure 2.5, nous pouvons observer l'effet du taux d'apprentissage λ sur la vitesse de convergence de cette méthode. Ce à quoi il est nécessaire de prêter attention est le fait que la valeur de 0.01 utilisée pour ce paramètre par les méthodes précédentes s'avère être beaucoup trop faible dans le cas d'Adam. Le minimum n'est atteint qu'au bout de 2440 itérations ce qui est approximativement trois fois plus que pour la méthode simple.

Ceci nous a donc pousser à augmenter cette valeur jusqu'à $\lambda = 0.7$. Cette valeur pour le taux d'apprentissage s'est avéré être assez haute pour éviter des itérations inutiles mais en même temps pas trop grande non plus afin de ne pas rendre l'algorithme instable.

2.5.2 Comparaisons des descentes entre les quatre méthodes

Les deux graphiques ci-dessous permettent à présent de servir de résumé sur les diverses méthodes abordés à travers la visualisation des trajectoires différentes qu'elles tracent.

La méthode qui se distingue le plus reste quand même celle d'**Adam**, suite au fait qu'elle prend en compte l'aspect non-isotrope de la fonction. En l'occurrence, le gradient dans la direction de l'axe y est beaucoup plus prononcé que celui de l'axe x , or la trajectoire tracée par **Adam** ne priorise pas forcément la dimension ayant la plus grande variation du gradient. Cela lui permet donc de ne pas être "influencé" par une dimension en particulier. Ceci est important car dans le cas d'une vraie fonction de coût, il est possible que la dimension ayant en général une faible variation du gradient possède à un moment un minimum plus important que les autres directions.



(a) Graphes des descentes

(b) Chemins empruntés par les diverses méthodes

Figure 2.6: Comparaisons des méthodes de descente de gradient

2.5.3 Paramètres spécifiques

Adam introduit deux nouveaux paramètres, β_1 et β_2 , qui permettent de modéliser ces notions de taux d'apprentissage adaptatif et d'inertie. Nous allons à présent tenter de les

faire varier même si les valeurs conseillées dans le papier scientifique “*Adam: A Method for Stochastic Optimization*” sont respectivement 0.9 et 0.999.

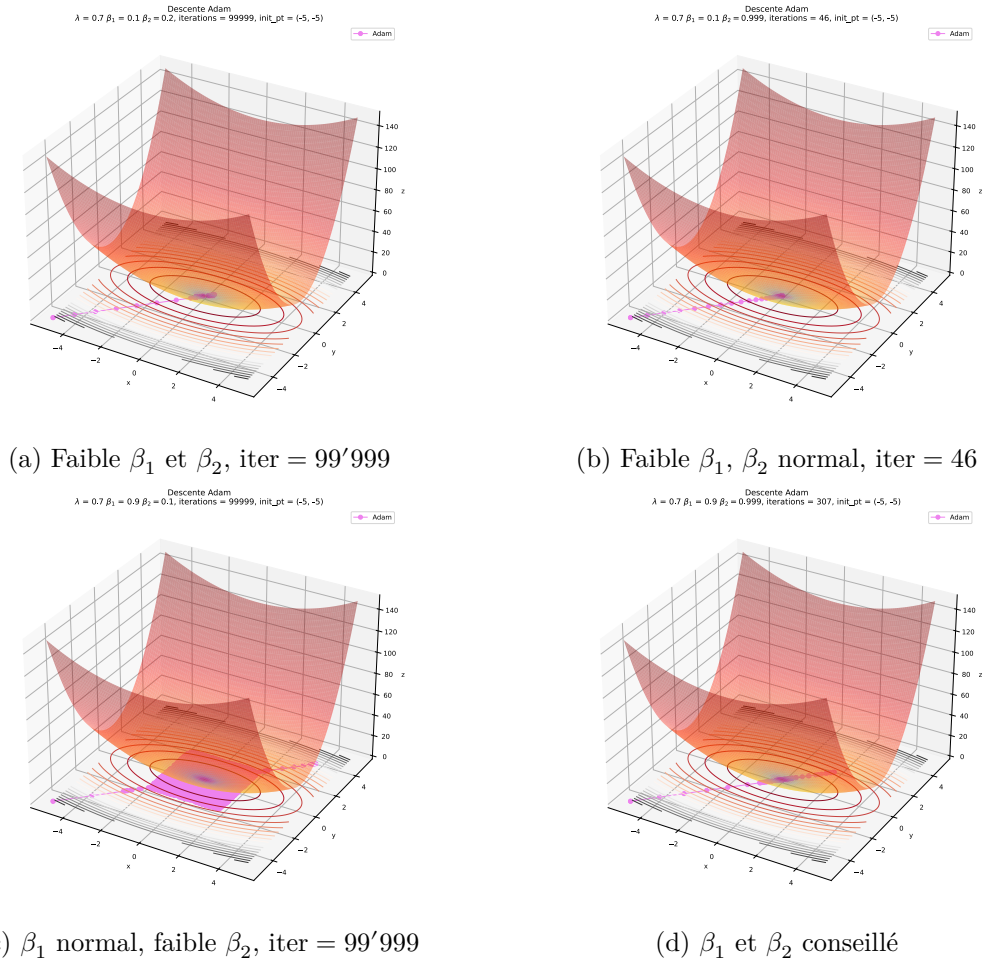


Figure 2.7: Influence des paramètres β_1 et β_2 sur la méthode d'Adam

3 Pièges

Après avoir passé en revue les particularités de chaque algorithme de descente, nous allons à présent parler de certains pièges auxquels ces méthodes peuvent faire face. Nous présenterons les problématiques des *ravines*, des *plateaux* ainsi que des *minimaux locaux*.

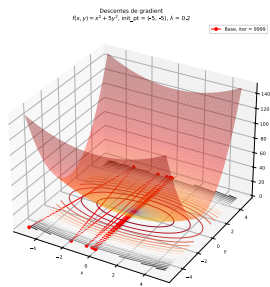
3.1 Ravines

Le problème des ravines auxquelles nous avons fait référence précédemment est illustré à l'aide la Figure 2.1a. Suite au fait que la fonction $f(x, y) = x^2 + 5y^2$ présente une non-isotropie (en d'autres termes, le gradient n'est pas uniforme dans toutes les dimensions), nous pouvons voir que l'algorithme de descente simple avec un taux d'apprentissage de $\lambda = 0.1$ arrive à descendre dans la "vallée" au bout d'**une unique itération** suite au fait que le gradient est très prononcé dans la direction de l'axe y . Cependant pour pouvoir par la suite converger vers le minimum, **72** itérations supplémentaires sont nécessaires car le gradient dans la direction de l'axe x est comparativement très faible.

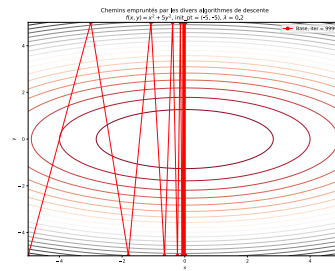
Une seconde problématique est aussi soulevée. Si la nature non-isotrope de la fonction est couplée à un taux d'apprentissage "trop" élevé, ceci peut potentiellement mener à une divergence de la descente comme l'illustre les graphiques ci-dessous.

! Important

Il est pertinent de noter le fait que le taux d'apprentissage λ ne fut augmenté que d'un dixième par rapport à la Figure 2.1a. Cet ajustement léger mène donc à une divergence.



(a) Graphique de la fonction



(b) Lignes de niveaux de la fonction

Figure 3.1: Cas de divergence de la descente

3.2 Plateaux

Concernant la problématique des plateaux, celle-ci est assez explicite. Si $\nabla f \approx \vec{0}$, alors les diverses méthodes auront beaucoup de peine à avancer / itérer. Ce phénomène peut être illustré à l'aide de la fonction $f(x, y) = 1 - \exp(-10x^2 - y^2)$ dont la particularité est qu'elle est extrêmement plate sauf dans le voisinage de $f(0, 0)$ où se situe son minimum global. Par la suite nous effectuerons tout de même une descente réussie sur cette fonction cependant le point de départ devra se situer pas loin de de l'extremum.

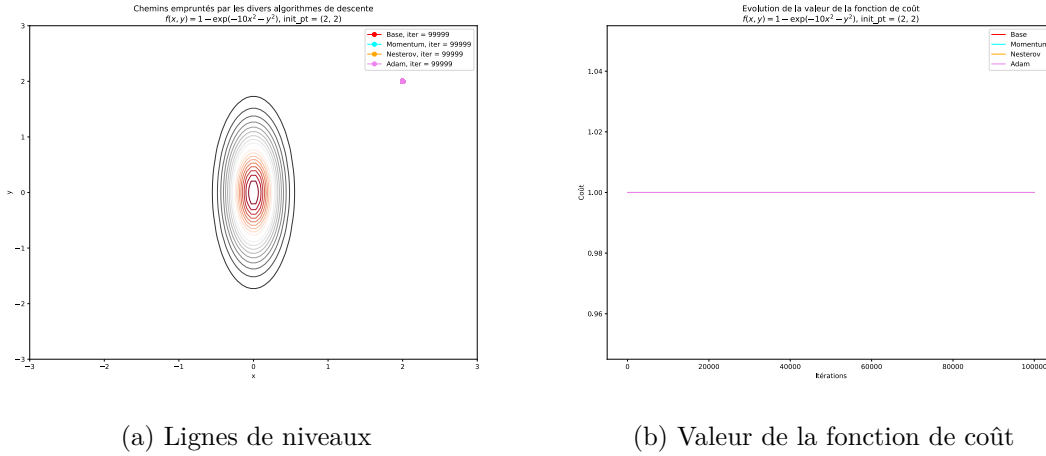


Figure 3.2: Illustration du piège posé par un plateau

Les graphiques ci-dessus illustrent bien le fait qu'en partant du point $(2, 2)$ de la fonction, même après $1 \cdot 10^5$ itérations, les quatre méthodes n'ont su avancer dans la direction du minimum suite au fait que le gradient présente peu de variation et par-dessus tout, le fait qu'il est presque égal au vecteur nul.

3.3 Minimums locaux

La problématique des minimums locaux est aussi assez explicite. Il se peut qu'une fonction en possède une multitude. Par conséquent, dépendamment du point de départ initial, il se peut qu'en effectuant une descente simple ou une descente Momentum qui n'a su accumuler assez d'inertie, on se retrouve coincé dans un minimum local. Ceci n'est bien évidemment pas idéal car à présent le but final d'atteindre un minimum global dépend aussi du point de départ choisi, or il s'avère que le choix de celui-ci est souvent aléatoire.

Cette problématique sera illustrée à l'aide de divers graphiques lorsque nous effectuerons les descentes de gradient sur la fonction d'**Ackley**. La particularité de cette fonction est le fait qu'elle possède justement un très grand nombre de minimum locaux et un minimum global qui se démarque en $(0, 0)$.

4 Descentes sur diverses fonctions

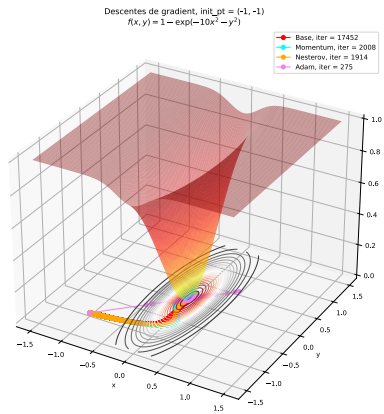
Après avoir pu discuter de manière détaillée des diverses méthodes de descente ainsi que des problématiques qui peuvent survenir suite aux particularités topologiques de la fonction à optimiser, nous pouvons à présent un peu nous amuser en essayant de trouver les minimums globaux de certaines fonctions non-triviales à l'aide des méthodes étudiées précédemment.

4.1 Le “puit”

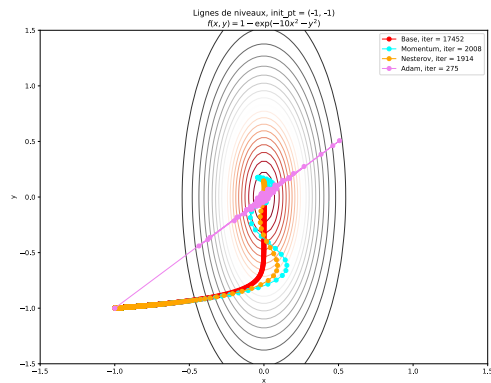
Ou plus communément connu sous le nom de :

$$f(x, y) = 1 - \exp(-10x^2 - y^2) \quad (4.1)$$

Nous avons déjà brièvement introduit cette fonction lors de la discussion de la problématique des plateaux. Les graphiques ci-dessous permettent de mieux visualiser sa particularité principale, celle d'un minimum global très prononcé en $(0, 0)$ mais d'une topologie extrêmement plate dans ses environs. La Figure 4.1b démontre cet aspect de topologie plate à travers le fait qu'aucune ligne de niveau ne s'affiche dans le voisinage proche de l'extremum.



(a) Graphique de la fonction “puit”



(b) Lignes de niveaux de descentes

Figure 4.1: Descentes sur la fonction “puit”

4.1.1 Évolution de la valeur de la fonction de coût

Le graphique ci-dessous permet d’illustrer clairement l’évolution de la valeur de la fonction de coût f par rapport au nombre d’itérations effectuées par chaque méthode. Nous pouvons voir qu’**Adam** a atteint le minimum en premier, au bout de seulement 275 itérations. **Momentum** et **Nesterov** sont tous deux très similaires, $\sim 2'000$ itérations. Puis finalement la méthode de base qui nécessite 17'452 itérations pour y parvenir.

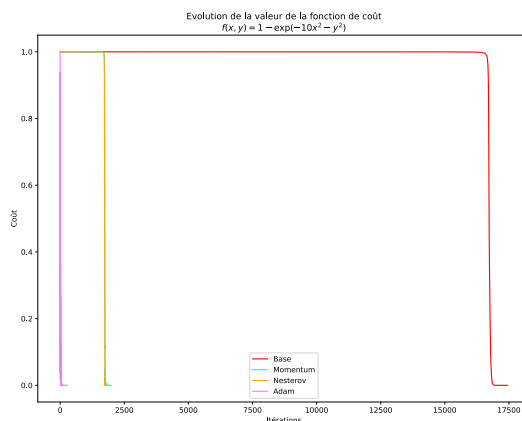


Figure 4.2: Évolution de la valeur de la fonction de coût “puit”

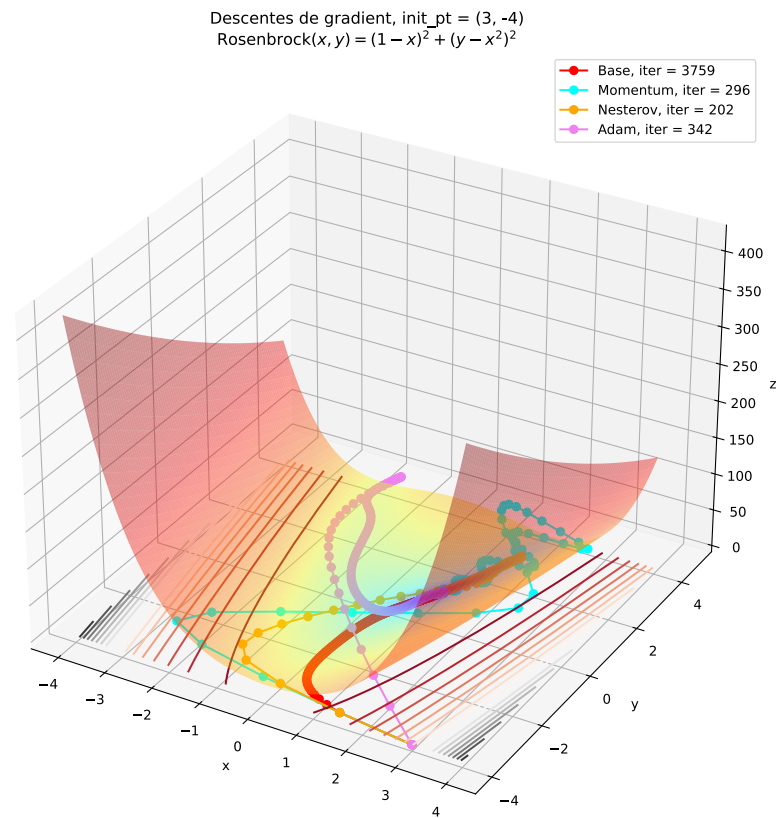
4.2 Rosenbrock

La fonction de Rosenbrock est une fonction du type $(a-x)^2 + b(y-x^2)^2$ souvent utilisée pour évaluer les performances d’algorithmes d’optimisation. Sa particularité est le fait que son minimum global se situe en $f(a, a^2)$ et la valeur à atteindre en ce point est 0. En l’occurrence, nous avons utilisé la variante avec $a = 1$ et $b = 1$ (sinon `sympy` explose) ce qui implique donc que le minimum global à atteindre se situe en $x = 1$ et $y = 1$

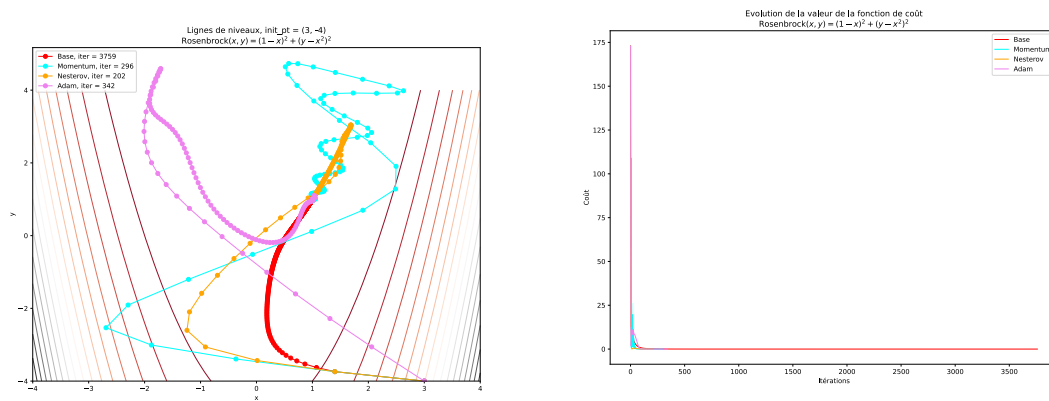
$$\text{Rosenbrock}(x, y) = (1 - x)^2 + (y - x^2)^2 \quad (4.2)$$

Les graphiques sur la page suivante illustrent les descentes de diverses méthodes depuis le point $(3, -4)$ jusqu’au minimum en $(1, 1)$. Le tableau ci-dessous résume le nombre d’itérations requises par chaque méthode avant d’atteindre le minimum.

Méthodes	Nombre d’itérations
Simple	3'759
Momentum	296
Nesterov	202
Adam	342



(a) Graphique de la fonction Rosenbrock



(b) Méthodes de descentes

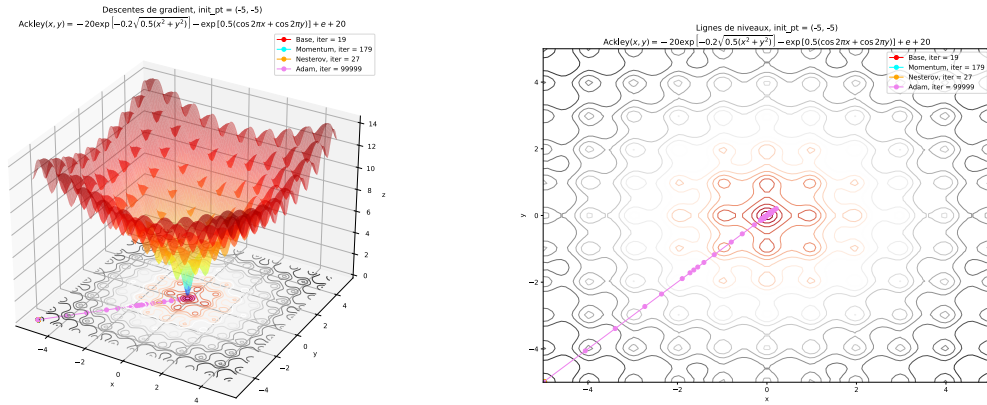
(c) Évolution de la minimisation de la valeur de la fonction Rosenbrock

Figure 4.3: Graphique des descentes sur la fonction de Rosenbrock

4.3 Ackley

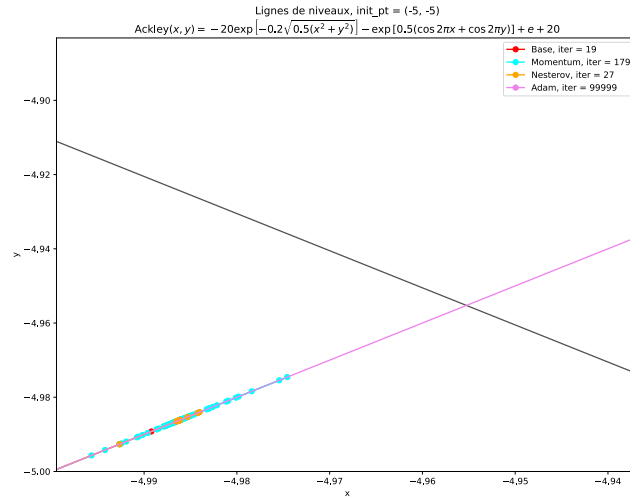
Comme cela fut promis dans la partie consacrée à la discussion des problématiques, nous allons à présent visualiser le problème des minimums locaux et leur influence sur la recherche d'un minimum global. Pour faire ceci, nous ferons appel à la fonction d'Ackley. La raison de ce choix est le fait que cette fonction se prête très bien à cet exercice suite au fait qu'elle possède une multitude de minimums locaux avec un minimum global en $f(0,0)$. Voici la fonction d'Ackley :

$$\text{Ackley}(x, y) = -20 \exp \left[-0.2 \sqrt{0.5(x^2 + y^2)} \right] - \exp [0.5(\cos 2\pi x + \cos 2\pi y)] + e + 20 \quad (4.3)$$



(a) Graphique de la fonction d'Ackley

(b) Vue macroscopique des trajectoires



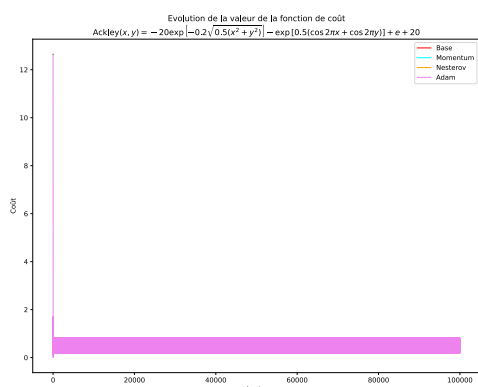
(c) Vue microscopique des trajectoires

Figure 4.4: Descentes sur la fonction d'Ackley

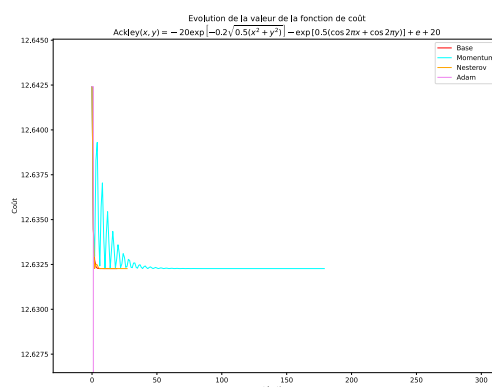
Premièrement, la Figure 4.4b illustre bien le fait qu'Adam a su se diriger vers le minimum global même s'il ne l'a pas totalement atteint suite au fait que nous imposons une limite de 100'000 itérations afin de pouvoir borner le temps d'exécution des descentes.

En ce qui concerne les trois autres méthodes, la Figure 4.4c permet de constater le fait qu'elles soient restées coincées très tôt dans un minimum local. En ce qui concerne la méthode simple, ce n'est pas surprenant vu son fonctionnement. Quant à Momentum et Nesterov qui possèdent une composante associée à l'inertie, il est légitime de se poser la question pourquoi eux aussi sont restés bloqués ? La raison est tout simplement car ils n'ont pas eu assez de temps pour gagner assez d'inertie pour pouvoir passer au-delà du premier minimum local. La Figure 4.4b permet de bien visualiser la topologie de cette fonction et en conclure que chaque minimum local est entouré par un "bol" qui compense l'inertie gagnée initialement lors d'une descente. Cela se voit sur la Figure 4.4c grâce au fait que les points de couleur cyan et orange ont l'air de d'abord passer par-dessus le minimum en $\sim f(-4.99, -4.99)$ grâce à l'inertie accumulée lors de la descente depuis $f(-5, -5)$ sauf que celle-ci n'a pas duré assez longtemps pour qu'ils puissent par la suite continuer à descendre. De ce fait, nous voyons donc que les points oscillent jusqu'à se stabiliser dans le minimum local.

Les graphiques ci-dessous de l'évolution de la valeur de la fonction à chaque itération, permettent aussi d'observer ces oscillations, notamment dans le cas de Momentum. Ceci indique qu'après être passé par un minimum, Momentum fait face à une montée qu'il n'arrive pas à surmonter (*no pun intended*). Concernant Adam, lui aussi affiche des tendances d'oscillations (long bloc violet sur la Figure 4.5a), cependant celles-ci se situent au niveau d'une valeur bien plus basse que dans le cas des trois autres méthodes.



(a) Vue macroscopique



(b) Vue microscopique

Figure 4.5: Évolution de la valeur de la fonction d'Ackley à chaque itération