

**ECOLE NATIONALE SUPÉRIEURE D’INFORMATIQUE ET D’ANALYSE DES SYSTEMES**

**MASTER RECHERCHE SCIENCES DE DONNÉES ET BIG DATA**

**MODULE : TECHNIQUES D’OPTIMISATION**

RAPPORT : PROJET DE FIN DE MODULE

Encadré par : Pr. A. ELAFIA

Réalisé par : Rashid Haffadi

**ANNÉE UNIVERSITAIRE 2018/2019**

# **Introduction et motivation**

L’optimisation par les méthodes du gradient est d’une importance pratique fondamentale dans de nombreux domaines de la science, de la technologie et de l’ingénierie.

De nombreux problèmes dans ces domaines peuvent être exprimés comme l'optimisation d'une fonction objective qui nécessite une maximisation ou une minimisation par rapport à ses paramètres.

Si la fonction est différentiable, la méthode du gradient est une optimisation relativement efficace, car le calcul des dérivées partielles du premier ordre ont la même complexité de calcule que l’évaluation de la fonction. Souvent, les fonctions objectives sont stochastiques.

Par exemple, de nombreuses fonctions objectives sont composées d’une somme de sous-fonctions évaluées à différents niveaux dans ce cas l'optimisation peut être rendue plus efficace en prenant des gradients des sous-fonctions individuelles, à savoir descente de gradient stochastique (SGD).

SGD a fait ses preuves comme une méthode d’optimisation efficace et centrale qui a joué un rôle central dans le succès de beaucoup d’apprentissage automatique.

Notre présentation ici va ce limité aux méthodes du premier ordre.

Sommaire

[**Introduction et motivation** 3](#_Toc1593514)

[I-Problème 8](#_Toc1593515)

[1-Gradient stochastique 8](#_Toc1593516)

[2- Les moyennes mobiles (Moving averages) 9](#_Toc1593517)

[3- Les moyennes mobiles à correction de biais 12](#_Toc1593518)

[II-Adam 15](#_Toc1593519)

[1-pseudo code 15](#_Toc1593520)

[2-Algorithme 15](#_Toc1593521)

[3-Analyse de convergence 16](#_Toc1593522)

[Lemme 3.1 : inégalité du gradient. 16](#_Toc1593523)

[Conjecture 3.2 : malheureusement ce n’est pas encore démontré. 16](#_Toc1593524)

[Définition 3.3 : Somme des erreurs 16](#_Toc1593525)

[Théorème 3.4: 17](#_Toc1593526)

[Corollaire 3.5 : 17](#_Toc1593527)

[III-AdaMax 19](#_Toc1593528)

[1-pseudo code 19](#_Toc1593529)

[2-algorithme 19](#_Toc1593530)

[3- Avantages de AdaMax : 20](#_Toc1593531)

[**a- pas de mise à jour des paramètres :** 20](#_Toc1593532)

[**b- Taille de mise à jour invariant:** 20](#_Toc1593533)

[**c- Les mêmes avantages de Adam :** 20](#_Toc1593534)

[**d- Le même taux de convergence qu’Adam** 20](#_Toc1593535)

[4- Problèmes de AdaMax : 21](#_Toc1593536)

[IV-Nesterov-accelerated Adaptive Moment Estimation (Nadam) 23](#_Toc1593537)

[1-pseudo code 23](#_Toc1593538)

[2-Algorithme 23](#_Toc1593539)

[V- recherche linéaire non monotone (F-rule) 26](#_Toc1593540)

[1-Définitions 26](#_Toc1593541)

[2-Algorithme 26](#_Toc1593542)

[3-Convergence globale 27](#_Toc1593543)

[Lemme 3.1 : 27](#_Toc1593544)

[Théorème 3.2 : 28](#_Toc1593545)

[Notation 3.3 : 28](#_Toc1593546)

[Lemme 3.4 : 28](#_Toc1593547)

[Théorème 3.5 : 28](#_Toc1593548)

[VI- Tests Numériques 29](#_Toc1593549)

[1 - Regression Lineaire 29](#_Toc1593550)

[2 - () Convexe 30](#_Toc1593551)

[3 - Beale Function 32](#_Toc1593552)

[VII-Conclusion 35](#_Toc1593553)

[Annex : Démonstrations 36](#_Toc1593554)

[Théorème 3.4 36](#_Toc1593555)

[Corollaire 3.5 : 44](#_Toc1593556)

[Lemme 3.1 : 46](#_Toc1593557)

[Théorème 3.2 : 47](#_Toc1593558)

[Théoréme 3.5 : 48](#_Toc1593559)

# I-Problème

## 1-Gradient stochastique

Soit P le problème de minimisation suivant : (1)

Où, sont des fonctions différentiables.

P le gradient de la fonction f dans le timestep i.

On note le gradient de la fonction objective f. (2)

Dans la méthode du gradient de descente nous avons utilisé le gradient de la fonction objective pour résoudre le problème (P), la méthode du gradient converge linéairement avec .

Quand est une fonction de perte basée sur l'instance de données d'apprentissage indexée par i. Il est important de souligner que le coût de calcul par itération en descente de gradient est linéaire avec la taille de l'ensemble de données d'apprentissage n (). Par conséquent, lorsque n est énorme, le coût de calcul par descente de la descente de gradient est très élevé.

De ce fait, la descente de gradient stochastique offre une solution plus légère. À chaque itération, plutôt que de calculer le gradient, la descente de gradient stochastique échantillonne de manière aléatoire (i) uniformément et calcule En effet la descente de gradient stochastique utilisecomme estimateur sans biais de f (x) puisque :

Dans la pratique c’est préférable d’utiliser un échantillon de taille N, à la place d’utiliser 1 seule,

(4)

On dit que est un estimateur sans biais de.

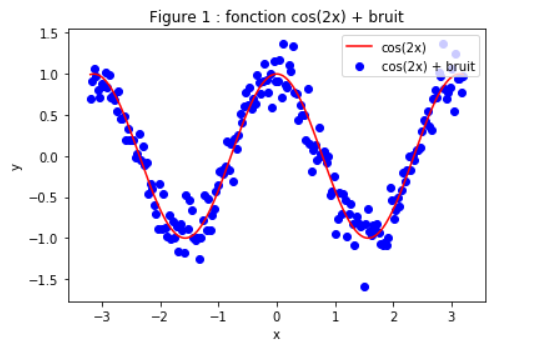
Plus N est grand plus l’estimation est plus précis.

Dans la suite on va introduire trois algorithmes qui utilisent le gradient stochastique pour la résolution de (P).

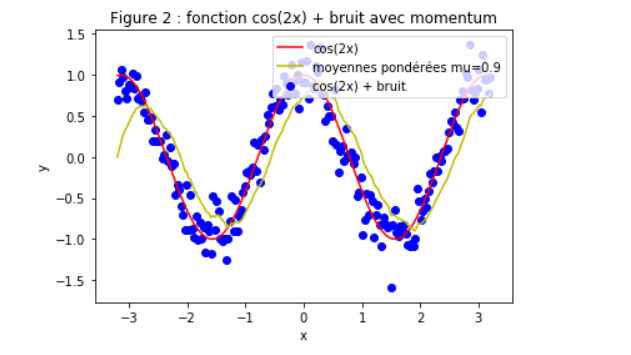
## 2- Les moyennes mobiles (Moving averages)

Supposons que nous ayons une séquence S de nombres qui soit bruyante. Pour cet exemple, on va tracer la fonction cosinus et ajouté du bruit. Cela ressemble à ceci :

S telle que b est un bruit aléatoire.



Ce que nous voulons faire avec ces données, c’est, au lieu de les utiliser, nous voulons une sorte de «moyenne pondérées» qui «nuirait» les données et les rapprocherait de la fonction d’origine. Les moyennes mobiles de manière exponentielle peuvent nous donner un résultat qui ressemble à ceci :



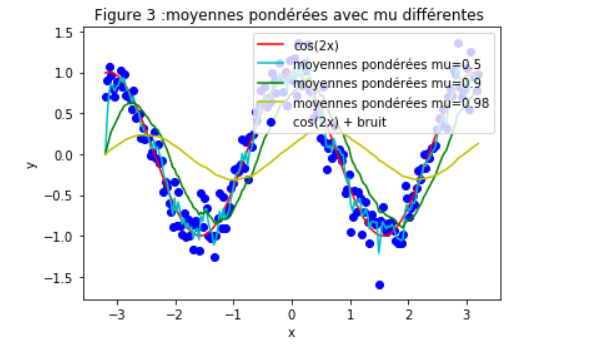
Comme vous pouvez constater, c’est un très bon résultat. Au lieu d'avoir des données avec beaucoup de bruit, nous avons une ligne beaucoup plus fluide, ce qui est plus proche de la fonction d'origine que les données dont nous disposions. Les moyennes mobiles de manière exponentielle définissent une nouvelle séquence V avec l’équation suivante :

Si alors :

(5)

Donc quand t devient important.

Cette séquence V est celle représentée en jaune ci-dessus. Le µ est un autre hyper-paramètre qui prend des valeurs entre 0 et 1. On a utilisé µ= 0.9 ci-dessus (Figure 2). Intuitivement, vous pouvez penser µ comme suit, nous faisons une moyenne approximative sur les derniers points de la séquence. Voyons comment le choix de µ affecte notre nouvelle séquence V.



Comme vous pouvez le constater, avec un nombre réduit de µ, la nouvelle séquence fluctue beaucoup, car nous calculons la moyenne sur un plus petit nombre d’exemples donc «plus proches» des données bruitées. Avec des valeurs de µ plus grandes, comme µ=0,98, nous obtenons une courbe beaucoup plus étouffée, mais elle est légèrement décalée vers la droite, car nous faisons la moyenne sur un plus grand nombre d’exemples (environ 50 pour µ = 0,98). µ = 0,9 fournit un bon équilibre entre ces deux extrêmes.

C’est pour cela qu’on va prendre µ = 0,9 pour tous nos tests dans la section des tests.

## 3- Les moyennes mobiles à correction de biais

La dernière chose à noter est que les deux premières itérations fourniront une très mauvaise moyenne car nous n’avons pas encore suffisamment de valeurs pour faire la moyenne. Au lieu d’utiliser V, la solution consiste à utiliser ce que l’on appelle la version de V à correction de biais.

Calculons En utilisant le résultat de (5).

On aura :

(6)

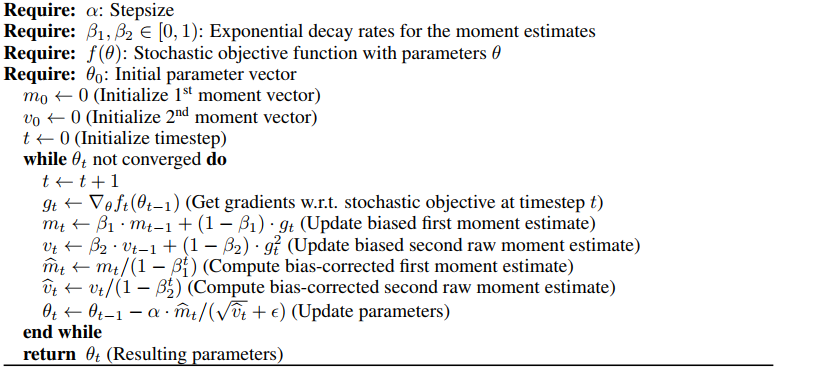
Nous devons maintenant corriger l’estimateur afin que la valeur attendue soit celle que nous voulons. Cette étape est généralement appelée correction de biais. La formule finale pour notre estimateur sera la suivante :

Avec de grandes valeurs de t, la puissance de t sera très proche de zéro, ne modifiant donc pas nos valeurs de V du tout. Mais pour de petites valeurs de t, cette équation produira des résultats un peu meilleurs.

Ce résultat est très important pour comprendre le fonctionnement des algorithmes qu’on va présenter après.

# II-Adam

## 1-pseudo code



## 2-Algorithme

Soit la fonction objective

Nous sommes intéressés à minimiser la fonction J respectivement à x et.

On note les réalisations de la fonction stochastique et.

L'algorithme met à jour les moyennes mobiles du gradient () et du gradient carré () où les paramètres β1, β2 ∈ [0, 1) contrôlent les vitesses de décroissance exponentielles de ces déplacements moyennes. Les moyennes mobiles elles-mêmes sont des estimations du premier ordre du moment (la moyenne) et du 2ème ordre du moment brut (la variance non centrée) du gradient. Cependant, ces moyennes mobiles sont initialisé avec 0 (), conduisant à des estimations de moment biaisées vers zéro, en particulier pendant les pas de temps initiaux, et en particulier lorsque les taux de décroissance sont faibles (c’est-à-dire que les β sont proches de 1).

Alors Adam peut être considéré comme une combinaison de RMSprop et de GSD avec moment. Il utilise les gradients au carré pour mettre à l'échelle le taux d'apprentissage comme RMSprop et la moyenne mobile du gradient au lieu du gradient lui-même comme GSD avec moment.

Parfois, la valeur de () pourrait être très proche de 0. La valeur absolue de nos poids pourrait devenir très grande, pour éviter cela nous incluons un paramètre epsilon () dans le dénominateur, qui prend valeur très faible.

## 3-Analyse de convergence

### Lemme 3.1 : inégalité du gradient.

Soit un ensemble convexe,

F est convexe sur ssi :

### Conjecture 3.2 : malheureusement ce n’est pas encore démontré.

Soit le ième élément de

Et Soit et alors :

### Définition 3.3 : Somme des erreurs

Soit

**:** Démonstrations

### Théorème 3.4:

Soit des fonctions différentiables et convexes pout tout t=1, 2, …, n

Soit

Supposons que :

### Corollaire 3.5 :

Soit des fonctions différentiables et convexes pout tout t=1, 2, …, n

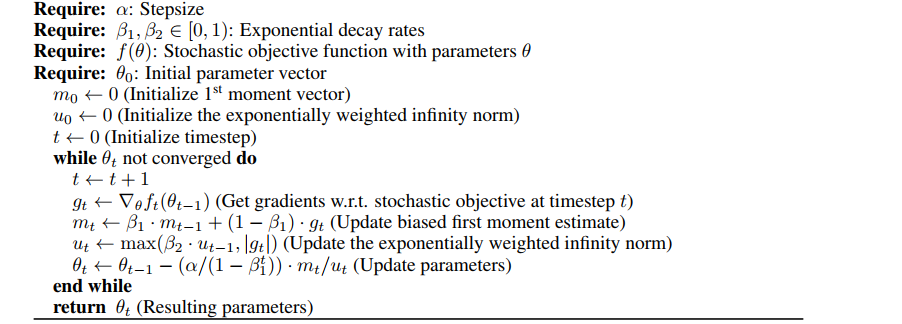
Soit

Supposons que :

Alors l’estimation de convergence suivant est vérifiée :

# III-AdaMax

## 1-pseudo code



## 2-algorithme

L'idée avec Adamax est de considérer la valeur comme la norme L2 des gradients (ou bien une estimation de). On peut généraliser le résultat pour les normes d’ordre p, mais cela devient assez instable pour les grandes valeurs de p. Mais si nous utilisons le cas particulier de la norme L(), il en résulte un algorithme étonnamment stable et performant.

Quand on aura :

Ce qui correspond à la formule récursive suivante :

## 3- Avantages de AdaMax :

### **a- pas de mise à jour des paramètres :**

La taille de pas réelle () prise par AdaMax à chaque itération est inférieur à l'hyper-paramètre α ().

### **b- Taille de mise à jour invariant:**

La taille de pas de la règle de mise à jour d'Adam est invariante par rapport à la magnitude du gradient, ce qui aide beaucoup lorsque on fait le parcoure des zones avec des gradients très petits.

### **c- Les mêmes avantages de Adam :**

Adam a été conçu pour combiner les avantages d'Adagrad, qui fonctionne bien avec des gradients clairsemés (sparse gradients), et de RMSprop, qui fonctionne bien dans l’optimisation en ligne. Avoir ces deux éléments nous permet d’utiliser Adam pour une gamme plus large de tâches. Adam peut également être considéré comme la combinaison de RMSprop et GSD avec moment.

### **d- Le même taux de convergence qu’Adam**

L’algorithme AdaMax converge avec le même taux de convergence qu’Adam

## 4- Problèmes de AdaMax :

Plusieurs articles ont mentionné des problèmes de convergence dans adam et adamax.

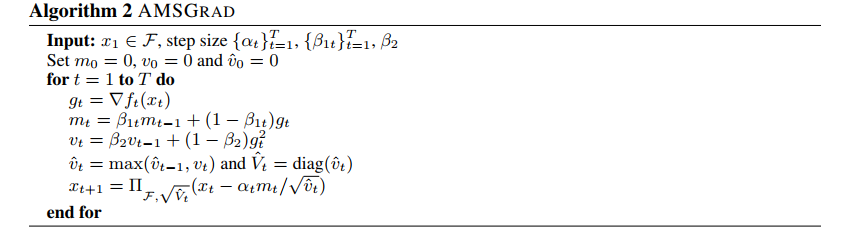
Par exemple dans l’article **[2]**  ont démontré qui a des fautes dans la démonstration de convergence dans l’article mère de Adamax et Adam **[1]** et ils ont donné une démonstration plus fiable.

Mais avec cela un autre article **[3]** récent publié en 2018 a démontré qu’il ne converge pas dans plusieurs cas par exemple Soit une fonction définit sur [-1,1] :

Note :% = modulo

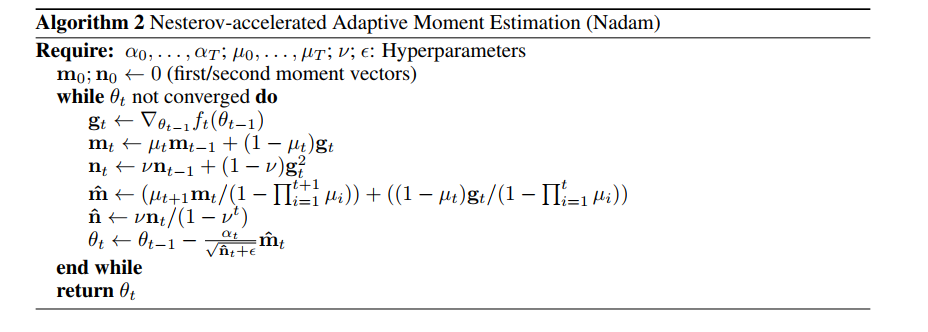
Quand C > 2 c’est évident que x=-1 donne le minimum de la somme des erreurs R(n), mais pour Adam diverge vers x=+1.

Un autre algorithme (AMSGrad) est proposé pour résoudre ce problème sans augmenter la complexité.



# IV-Nesterov-accelerated Adaptive Moment Estimation (Nadam)

## 1-pseudo code



## 2-Algorithme

Nous pouvons considérer les gradients accélérés de Nesterov comme facteur de correction pour les méthodes des moments. Prenons le cas où la vélocité ajoutée aux paramètres vous donne une perte élevée immédiate non désirée, par exemple le cas du gradient très grand. Dans ce cas, les méthodes des moments peuvent être très lentes car le chemin d'optimisation emprunté présente de fortes oscillations. Dans le cas du gradient accéléré de Nesterov, vous pouvez l'afficher comme si vous fouilliez à travers les paramètres intermédiaires lorsque la vélocité ajoutée mènerait les paramètres. Si la mise à jour de la vitesse conduit à une mauvaise perte, les gradients la dirigeront vers. Cela aide Nesterov Accelerated Gradient (NAG) à éviter les oscillations.

Nadam (Estimation du moment adaptatif accéléré par Nesterov) combine Adam et NAG.

Pour incorporer NAG dans Adam, nous devons modifier le terme du moment () :

On appliquant ce résultat sur Adam on aura :

Cela est faisable aussi avec AdaMax aussi.

# V- recherche linéaire non monotone (F-rule)

## 1-Définitions

**Notation 1 :**

On note

Supposons f(x) est borné dans L et f (x) est continue sur un ensemble ouvert qui contient L.

**Définition 1 :** La fonction est une fonction de forçage (fonction F) si pour toute séquence

**Définition 2 : Soit**

**une fonction définit par :**

## 2-Algorithme

Entrées :

Sortie :

Début

 := 1

si k=0 :

m(k) :=1

sinon :

m(k) := min(k+1, M)

Prendre

Tant que:

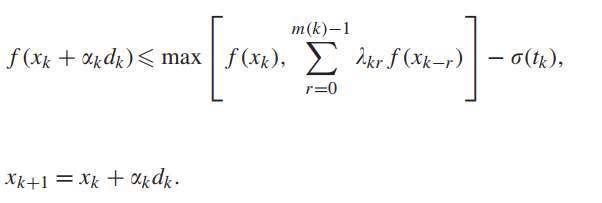
Fin tant que

Return

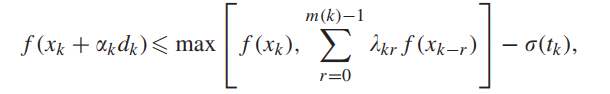
Fin.

## 3-Convergence globale

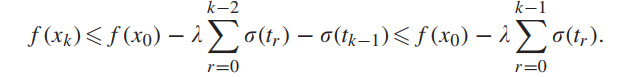
**Règle de mise à jour :**



### Lemme 3.1 :

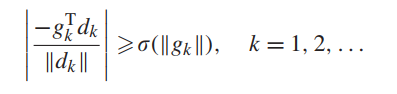
Si :

Et :

Alors 

### Théorème 3.2 :

Sous notation 1 si :



Alors  et

### Notation 3.3 :

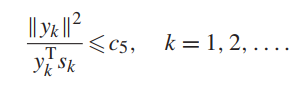
f(x) est deux fois différentiable alors : il existe c3, c4 telle que :





### Lemme 3.4 :

Si notation 2 est vérifié alors il existe un réel positive telle que :



### Théorème 3.5 :

Soit f(x) une fonction qui satisfait les notations 1 et 2,

suite infinie généré par Quasi-Newton avec la règle de mise à jour au-dessus alors :



# VI- Tests Numériques

## 1 - Regression Lineaire

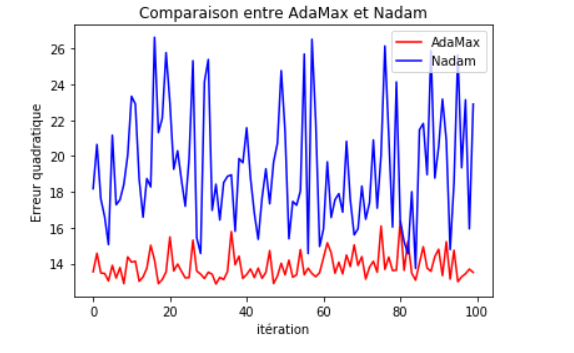
Soit un ensemble de données numérique.

Soit la fonction objective .

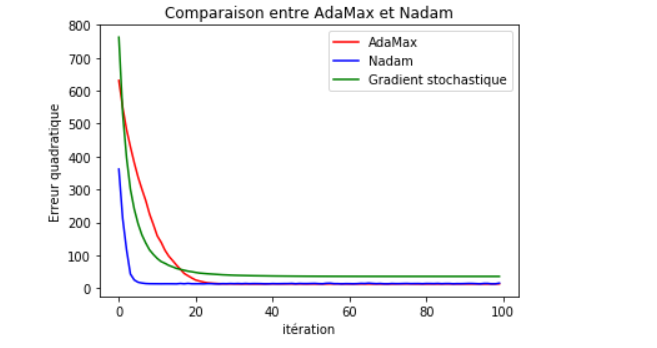
La fonction J est l’erreur quadratique de D réspectivement .

Soit

Le test est fait avec AdaMax et Nadam sur les paramétres : m = 10000 (taille des données), α=0.05 (pas d’apprentissage), β1=0.9, β2=0.999, d=10 (la dimension de données).

alors pour α=0.05 AdaMax et Nadam ne sont pas stable méme que AdaMax donne des résultat approché.

On vas tester avec un pas plus grand α=0.01 pour AdaMax et Nadam et α=2\*1e-5 pour l’algorithme du gadient stochastique.

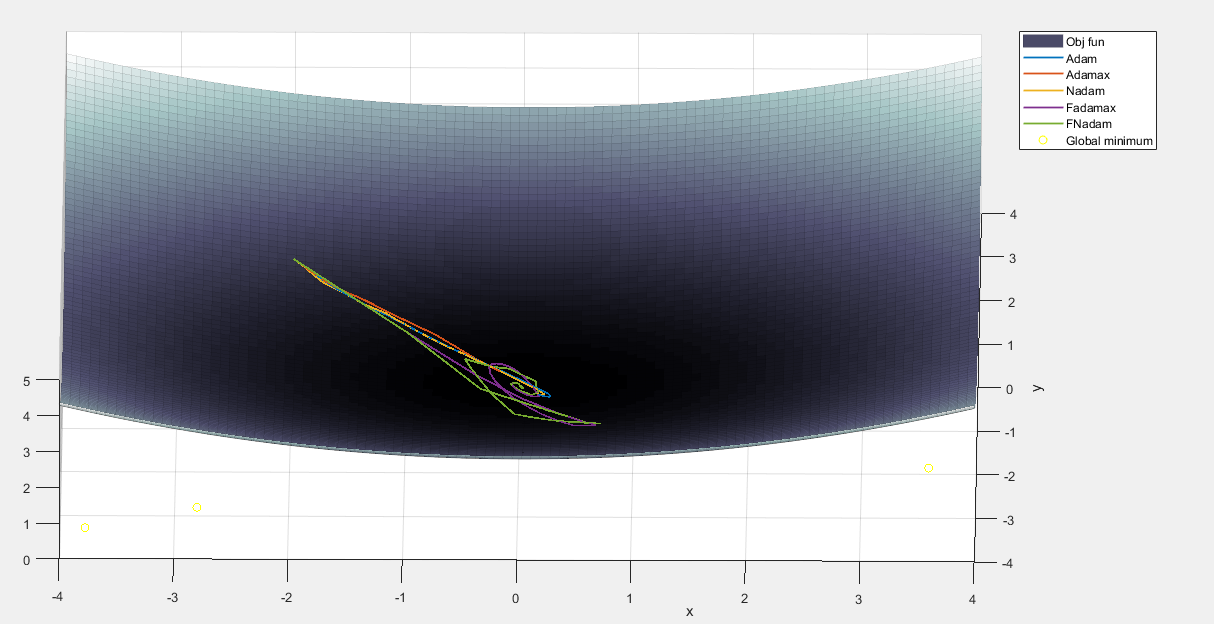


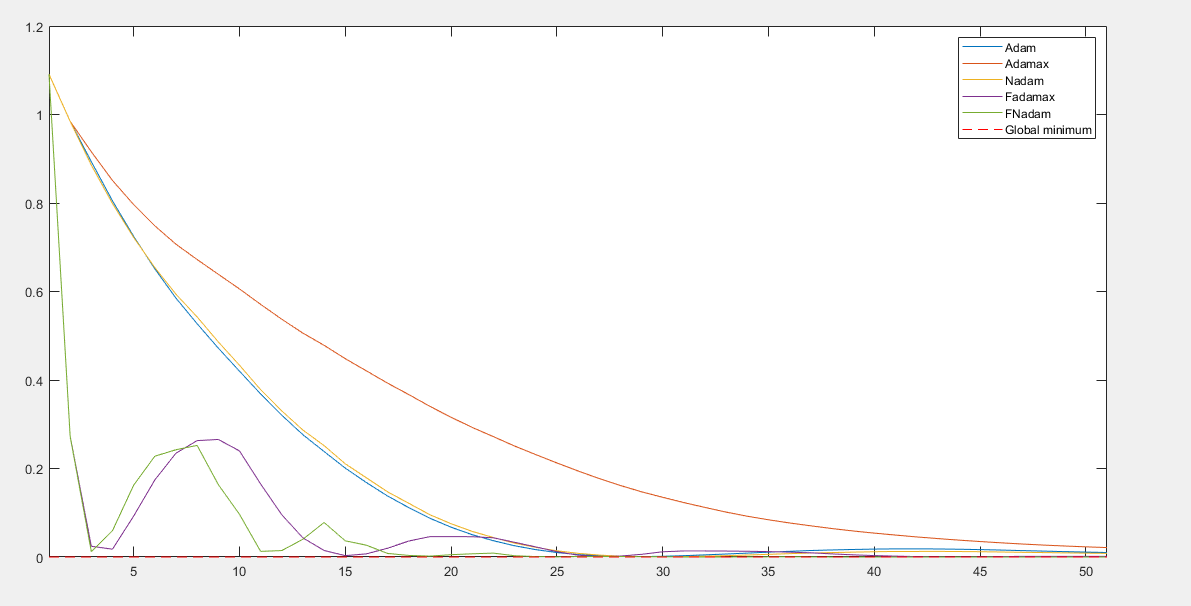
## 2 - () Convexe

Pout chaque itération t on choisie aléatoirement (méthode PEAR) telle que :

On veut minimiser la fonction J s.à x :

Pour .





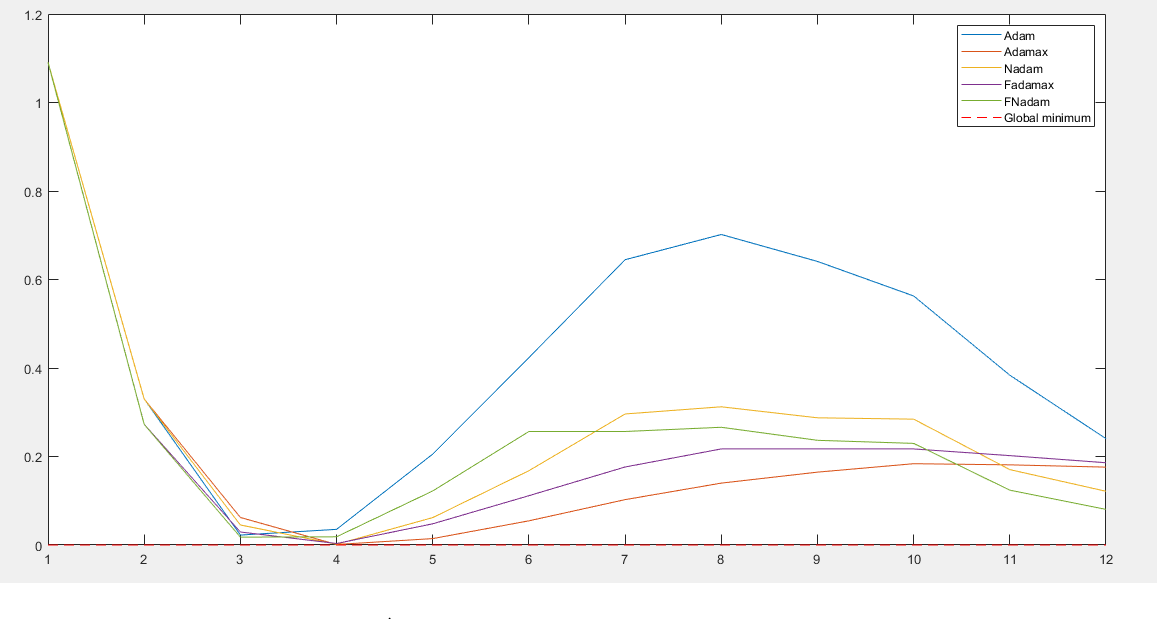
Fadamax : Adamax avec le pas approché.

FNadam : Nadam avec le pas approché.

Une premiére remarque qu’on peut faire sur les deux figure est que le pas approché a beucoup réduit le taux de convergence, pour Nadam par exemple il converge en 28 itérations mais avec le pas approché il converge en 3 itérations seulement .

Une deuxiéme remarque est que après l’algorithme converge une premiére fois, il diverge mais il converge encore une fois, c’est bien le fait du moment.

Avec α=0.9 pour Adam, AdaMax et Nadam on aura:



Tous les algorithms converge en 4 itérations, mais avec une supériorité de FNadam et Nadam en terme du précision.

## 3 - Beale Function

Soit la fonction objective

J est différentiable, multimodale et non-convex.

On vas étudier la fonction sur [-4.5 4.5]×[-4.5 4.5] :

Prenons  **, , .**

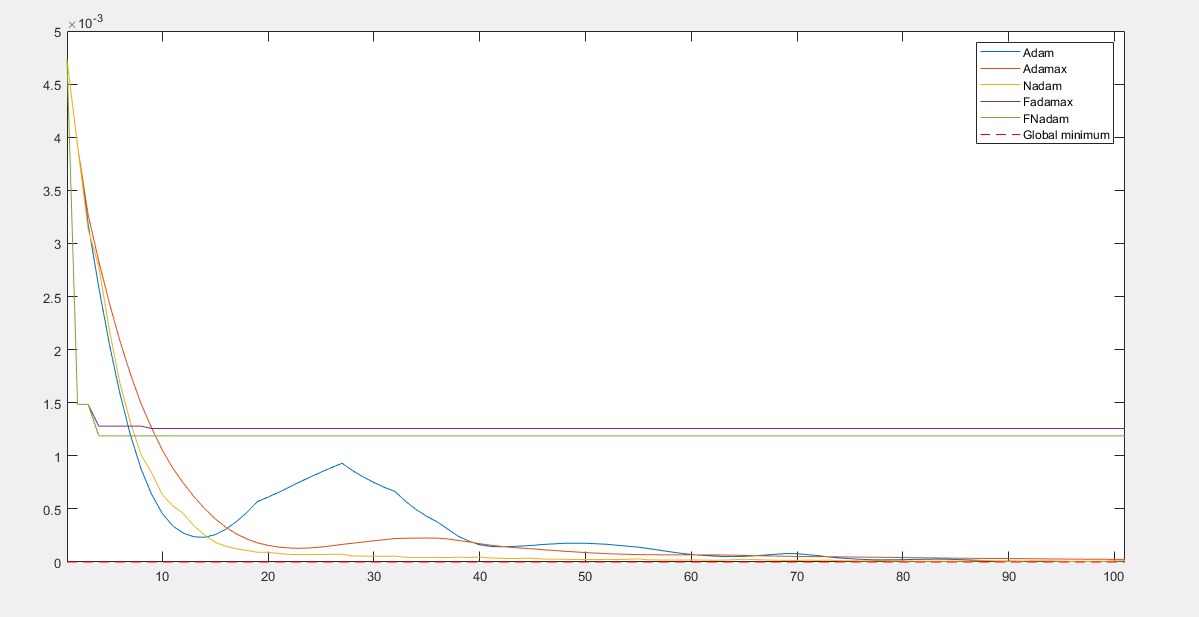
**Alors :**

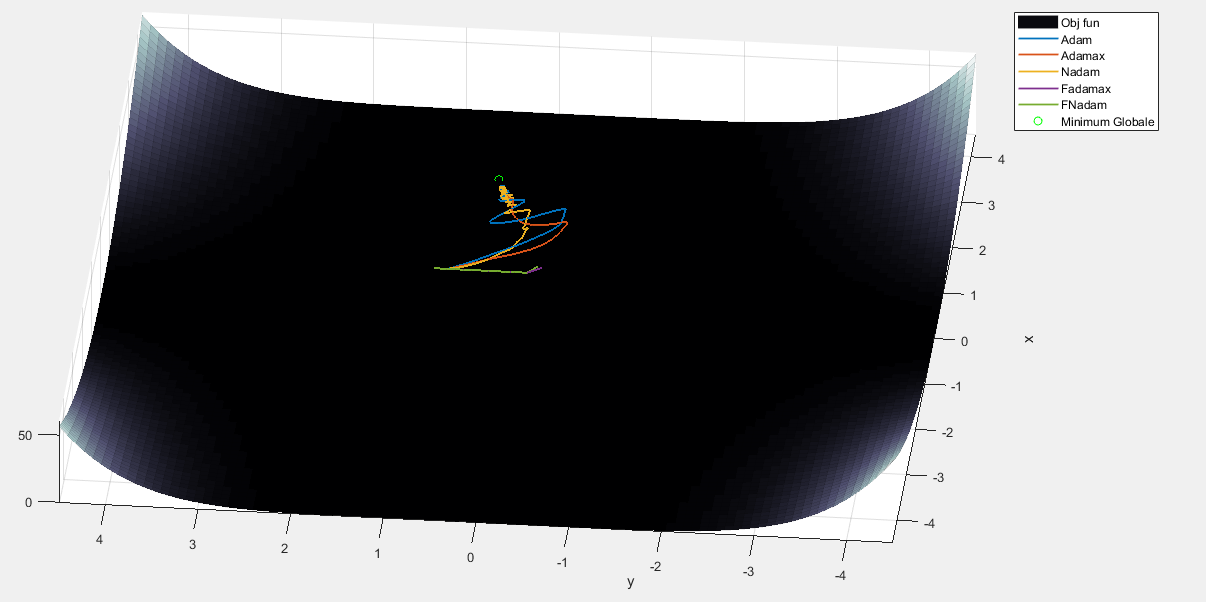
**,**

**,**

On vas prendre d’une maniére aleatoire avec remise pour chaque itération t :.

Les tests sont faites avec les paramétres : α= 0.1, x0 = [1 1], N=100, β1=0.9, β2=0.999.





**Ce qu’on peut tirer des deux figure ci-dessus est que les algorithmes Adam, Nadam et Adamax converge vers un minimum globale ou bien au moins vers un minimum locale.**

**Les deux algorithms avec le pas approché sont stocker dans un minimum locale.**

# VII-Conclusion

Dans ce rapport on a pu à étudier quatre algorithmes d’optimisation qui sont les plus utiliser dans les problèmes de machine/deep Learning avec des données de grands dimensions.

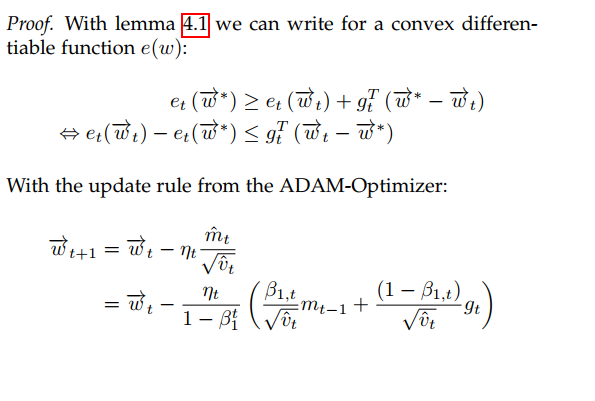
Pour récapituler, on a dit que Adam et AdaMax combine entre les avantages de deux méthodes populaires, AdaGrad pour les gradients ‘sparse’ et RMSProp pour les fonctions objectives non stationnaires.

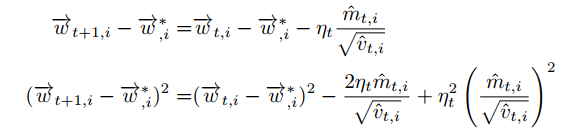
Nadam est une méthode plus robuste, car c’est une amélioration de Adam utilisant le moment de Nesterov.

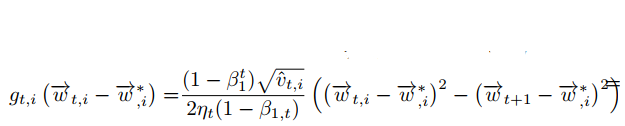
Comme ces méthodes sont très bonnes pour la résolution des problèmes convexes, ces algorithmes sont aussi bons pour la résolution de beaucoup de problèmes non convexe qu’on peut rencontrer en Machine/Deep Learning.

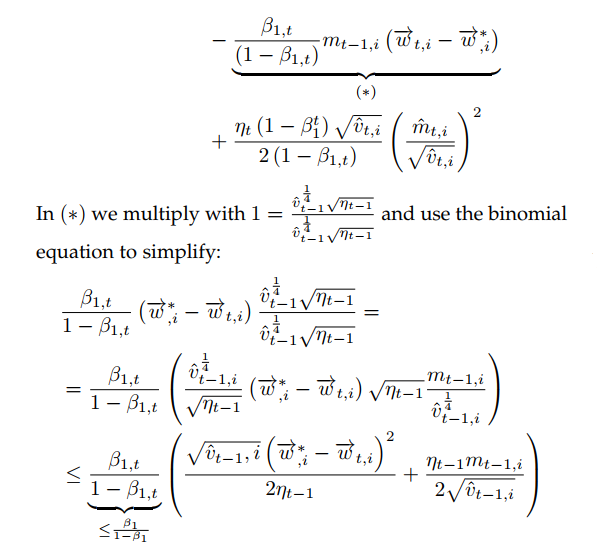
# Annex : Démonstrations

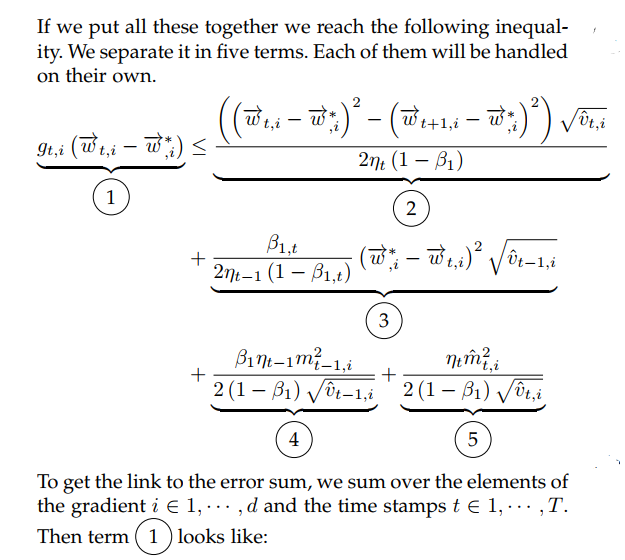
## Théorème 3.4

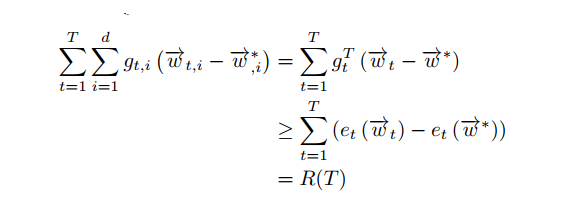




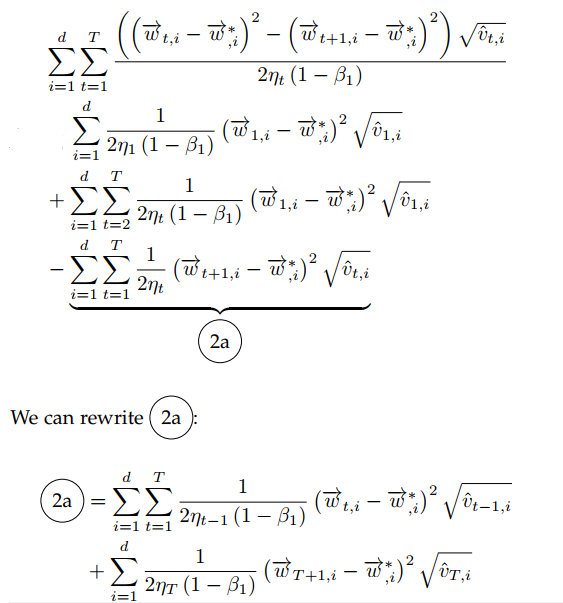


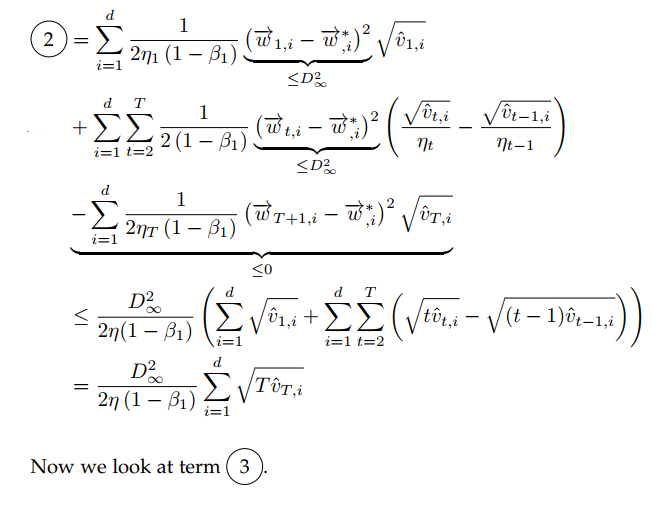


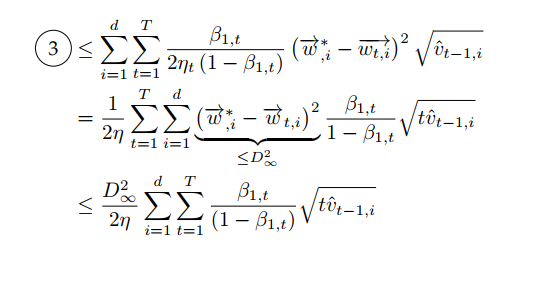


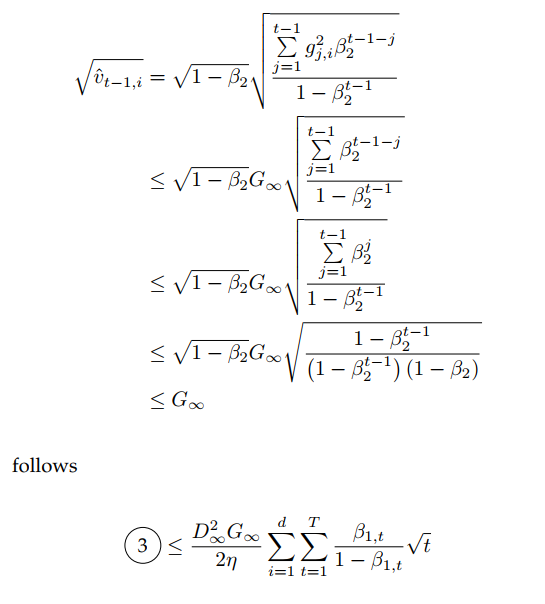


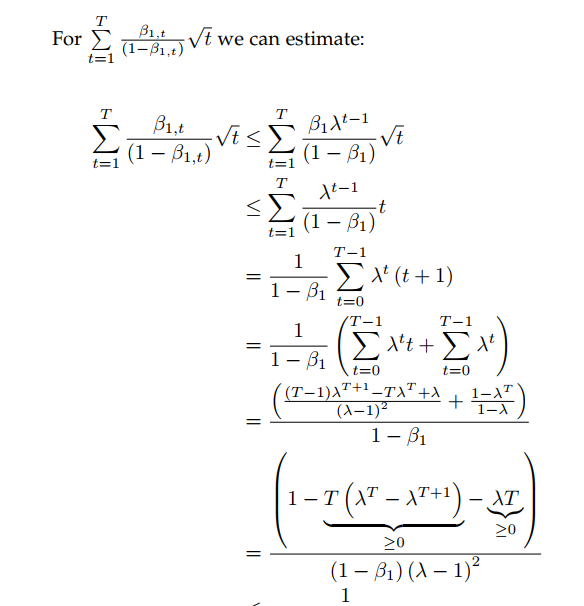


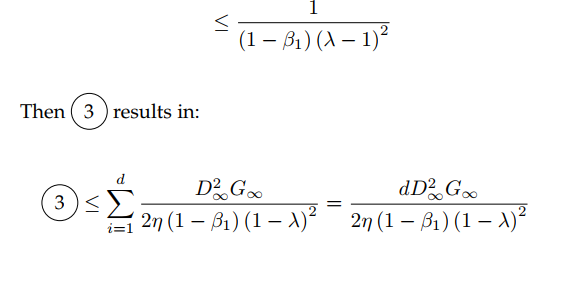


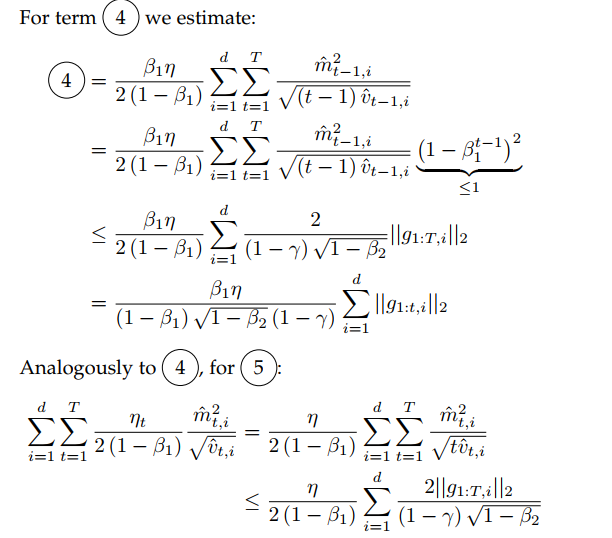


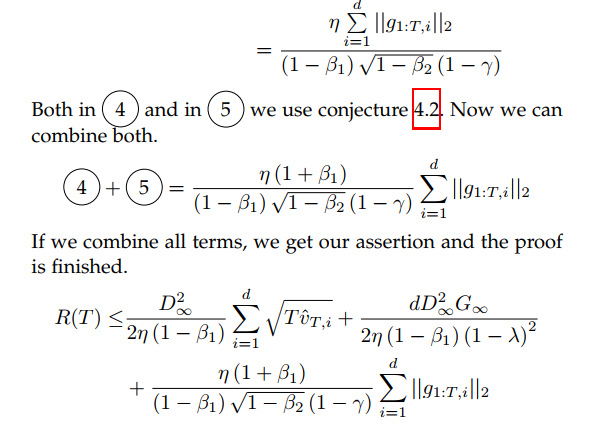




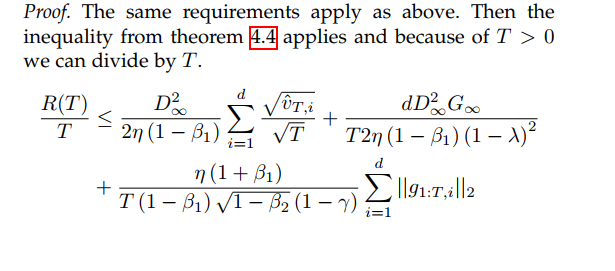


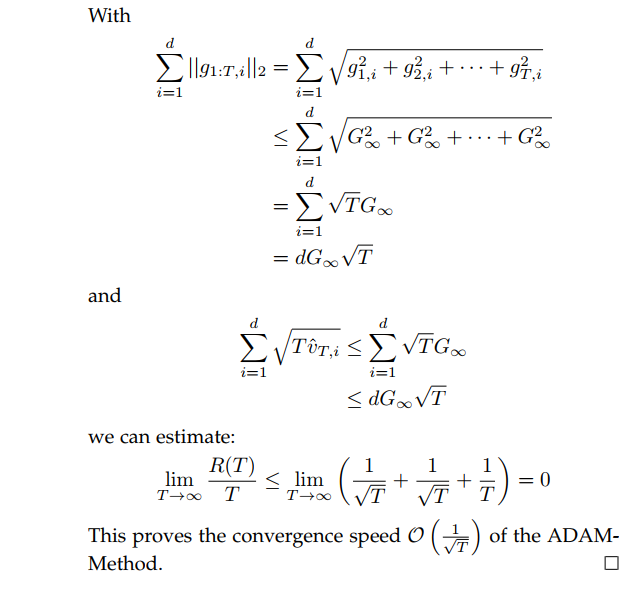




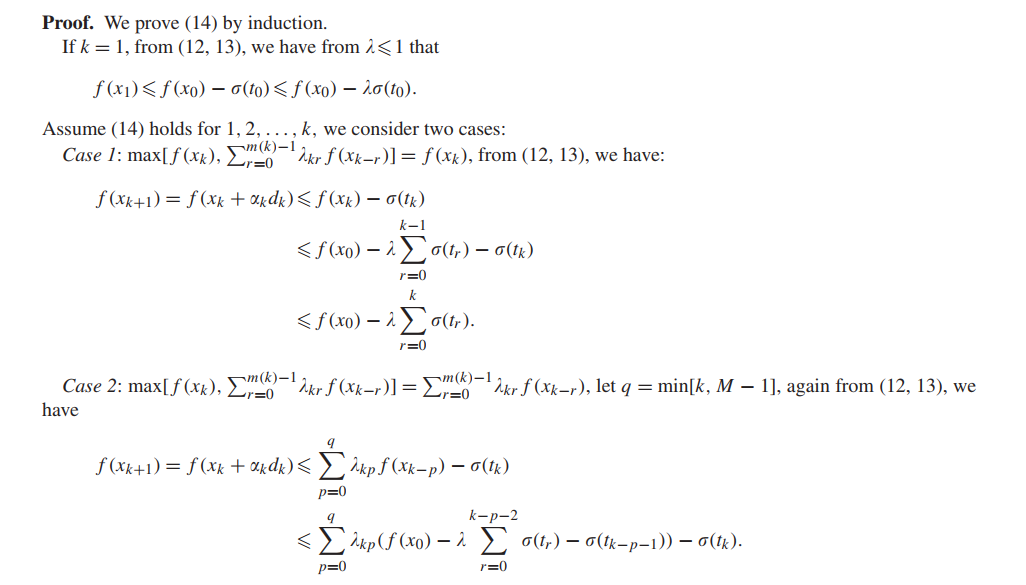


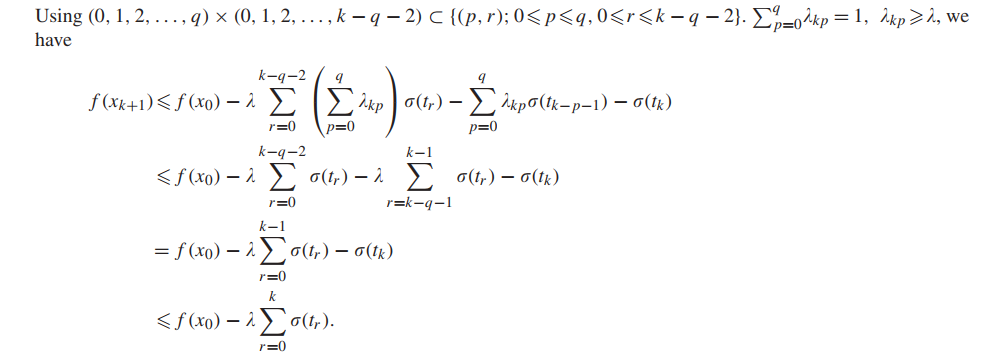
## Corollaire 3.5 :



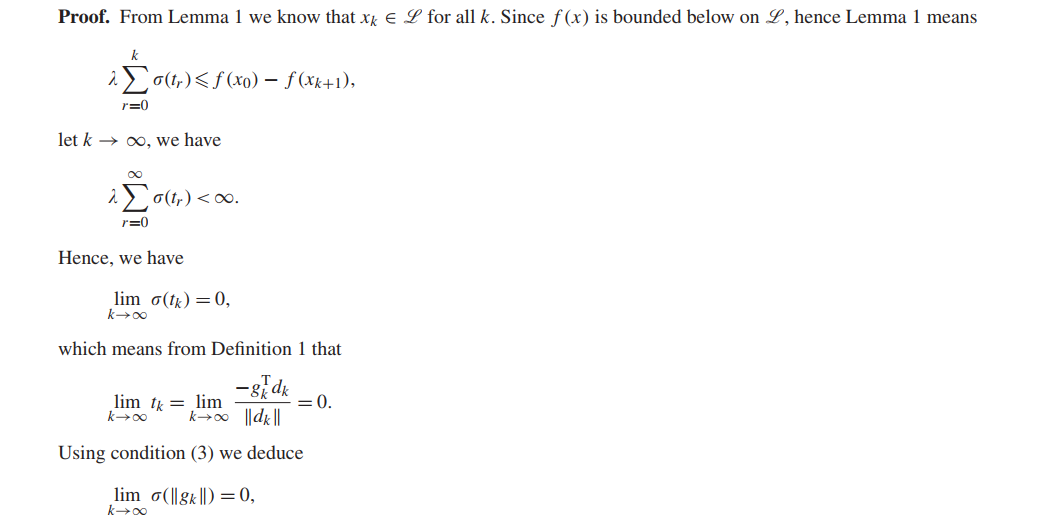


## Lemme 3.1 :

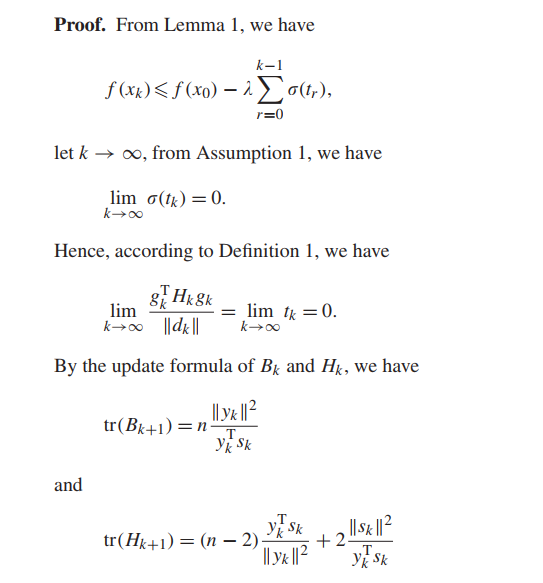


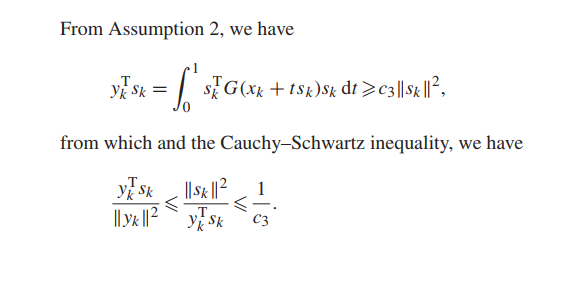


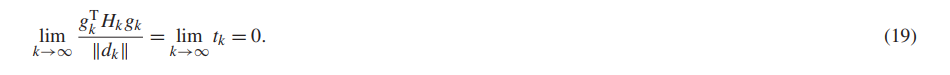
## Théorème 3.2 :



## Théoréme 3.5 :







**Alors :**

**.**

**Références**

**[1] -** Diederik P. Kingma and Jimmy Lei Ba. [Adam : A method for stochastic optimization](https://arxiv.org/abs/1412.6980). 2014. arXiv:1412.6980v9

**[2] -** [Sebastian Bock](https://arxiv.org/search/cs?searchtype=author&query=Bock%2C+S), [Josef Goppold](https://arxiv.org/search/cs?searchtype=author&query=Goppold%2C+J), [Martin Weiß](https://arxiv.org/search/cs?searchtype=author&query=Wei%C3%9F%2C+M). [An improvement of the convergence proof of the ADAM-Optimizer](https://arxiv.org/abs/1804.10587). 2018. arXiv:1804.10587v1

**[3] -** [Sashank J. Reddi](https://openreview.net/profile?email=sashank%40google.com), [Satyen Kale](https://openreview.net/profile?email=satyenkale%40google.com), [Sanjiv Kumar](https://openreview.net/profile?email=sanjivk%40google.com). [On the Convergence of Adam and Beyond](https://openreview.net/forum?id=ryQu7f-RZ). 2018.

**[4]** - Dozat, T. (2016). Incorporating Nesterov Momentum into Adam. ICLR Workshop, (1), 2013–2016.

**[5]** - [S. Ruder, “An overview of gradient descent optimization  
algorithms,” cite arxiv:1609.04747Comment: 12 pages, 6 figures.  
[Online].](http://arxiv.org/abs/1609.04747) Available: http://arxiv.org/abs/1609.04747