École Polytechnique / Sorbonne Université

Algorithmes de Langevin pour les réseaux de neurones markoviens et le contrôle stochastique profond

M2 Probabilité et Finance (Ex DEA El Karoui)

Realisé par : Ameur Nadir, El Moubaraki Anass

Résumé : *Dans ce rapport, nous analysons la robustesse de la méthode de Langevin dans le cadre du contrôle optimal profond, où la politique de contrôle est modélisée par un réseau de neurones. Nous commençons par un état de l'art synthétique des méthodes de descente de gradient appliquées à l'apprentissage supervisé, puis nous introduisons le formalisme du contrôle optimal profond, ainsi que les préconditionneurs classiques (RMSProp, Adadelta et Adam) et leurs variantes intégrant un bruit de Langevin. Enfin, nous évaluons ces algorithmes sur deux études de cas : le problème des quotas de pêche et celui de la couverture des actifs financiers.*

Mots clés : *Méthode de Langevin, Optimisation stochastique, Apprentissage profond, Contrôle optimal, Préconditionnement*.

Table des matières

1 Introduction

De nombreuses situations en finance nécessitent la résolution de problèmes de contrôle stochastique, ce qui implique d'optimiser des politiques dépendant d'états aléatoires évoluant dans le temps. Lorsque le contrôle est paramétré par un réseau de neurones, cette optimisation revient à entraîner un réseau très profond. Dans ce contexte, les méthodes classiques de descente de gradient stochastique peuvent converger lentement, voire se bloquer sur des points selles, en particulier lorsque la dimension du problème et sa non-convexité augmentent.

Pour remédier à ces difficultés d'optimisation, ce rapport explore l'injection d'un terme de bruit exogène de type Langevin dans les algorithmes de descente de gradient stochastique. Après un état de l'art des méthodes d'optimisation stochastique, nous formalisons le cadre du contrôle optimal profond et présentons la méthode de Langevin ainsi que sa version préconditionnée. Afin d'évaluer concrètement les apports de ces approches, nous étudions deux cas d'application : la régulation des quotas de pêche et la couverture d'un portefeuille soumis à des frictions de marché.

2 Apprentissage profond et optimisation par descente de gradient

Dans cette section, nous présentons un état de l'art synthétique des méthodes d'optimisation par descente de gradient utilisées en apprentissage profond. L'accent sera mis sur la modélisation mathématique, ainsi que sur la comparaison des différentes méthodes en termes de robustesse et de vitesse de convergence lorsque l'espace des états est de grande dimension.

2.1 Problème à résoudre

Nous nous plaçons dans le cadre de l'apprentissage supervisé, où l'on considère une base de données composée de réalisations $(X_i, Y_i)_{i=1:M}$ indépendantes et identiquement distribuées (i.i.d.) d'une variable aléatoire $(X, Y) \sim p_{X,Y}$. Typiquement, X_i représente la donnée d'entrée et Y_i le label associé. Pour simplifier, nous supposons que X prend ses valeurs dans $\mathbb{R}^d$ et Y dans $\mathbb{R}$. Nous faisons également l'hypothèse que $(X, Y) \in \mathbb{L}_2(\mathbb{P})$ et que la mesure de probabilité jointe est absolument continue par rapport à la mesure de Lebesgue définie sur $(\mathbb{R}^d \times \mathbb{R}, \mathcal{B}(\mathbb{R}^d \times \mathbb{R}))$, c'est-à-dire $p_{X,Y} = f_{X,Y}\, dx\, dy$. Enfin, sauf mention contraire, nous utiliserons la notation (x, y) pour désigner une valeur numérique, et (X, Y) lorsqu'il s'agit de variables aléatoires.

Afin de construire une règle de décision permettant de prédire le label à partir d'une donnée d'entrée, nous cherchons à résoudre le problème (1), où $f : \mathbb{R}^p \times \mathbb{R}^d \to \mathbb{R}$ et $\mathcal{L}_{emp} : \mathbb{R}^p \to \mathbb{R}^+$.

$$\theta^* \in \underset{\theta \in \mathbb{R}^p}{\operatorname{Argmin}} \ \mathcal{L}_{emp}(\theta) := \frac{1}{M} \sum_{i=1}^{M} (y_i - f(\theta, x_i))^2 \tag{1}$$

Dans la suite, nous supposons que $\mathcal{L}_{emp}$ est de classe C^1, coercive, et que son gradient $\nabla_\theta \mathcal{L}_{emp}$ est lipschitzien avec une constante de Lipschitz égale à κ. Nous supposons également que $f(\theta, X) \in \mathbb{L}_2(\mathbb{P})$, condition vérifiée notamment dans le cas d'une régression linéaire classique ou pour un réseau à une seule couche dont la fonction d'activation satisfait une hypothèse de croissance linéaire.

2.2 Descente de gradient déterministe

La méthode de descente de gradient déterministe consiste à approcher récursivement θ^*. Il s'agit d'une méthode de point fixe de type Picard, reposant sur l'itération (2). Sous les hypothèses de

régularité sur $\mathcal{L}_{emp}$ et pour un pas de descente $\gamma \in [0, \frac{2}{\kappa}]$, la procédure (2) est convergente. Il existe également d'autres variantes de la descente de gradient, telles que la descente à pas variable (3), ou encore la méthode de Newton (4), qui fait intervenir l'inverse de la hessienne de $\mathcal{L}_{emp}$ lorsque celle-ci existe et est inversible. Nous nous abstenons ici de fournir un choix optimal du pas variable, celui-ci s'inscrivant dans un cadre plus général que nous développerons dans la partie stochastique.

$$\theta_{n+1} = \theta_n - \gamma \nabla_\theta \mathcal{L}_{emp}(\theta_n) \tag{2}$$

$$\theta_{n+1} = \theta_n - \gamma_{n+1} \nabla_\theta \mathcal{L}_{emp}(\theta_n) \tag{3}$$

$$\theta_{n+1} = \theta_n - (\nabla^2_\theta \mathcal{L}_{emp}(\theta_n))^{-1} \nabla_\theta \mathcal{L}_{emp}(\theta_n) \tag{4}$$

La méthode de descente de gradient déterministe fonctionne très bien dans le cadre de l'optimisation convexe en faible dimension. Par exemple, pour la résolution du problème d'optimisation de Rockafellar-Uryasev [11], la méthode de Newton converge rapidement, à condition de choisir une initialisation judicieuse.

Cependant, dans le contexte de l'apprentissage profond, le nombre de poids p à estimer ainsi que le nombre d'échantillons M sont très élevés. De plus, la fonction de perte est rarement convexe, ce qui peut entraîner la convergence de la procédure déterministe vers des points selles, qui sont souvent des minimas locaux. Une alternative, largement explorée par les praticiens, consiste à perturber la descente déterministe en y ajoutant un bruit endogène. Cette approche permet une meilleure exploration de l'espace des états et évite plus facilement les pièges des minimas locaux.

2.3 Descente de gradient stochastique

Avant d'introduire la procédure de descente stochastique, rappelons que la loi forte des grands nombres implique la convergence décrite en (5). L'erreur quadratique moyenne que nous cherchons à estimer constitue donc un estimateur sans biais de l'erreur quadratique théorique. Le théorème central limite fournit, quant à lui, la vitesse de convergence en $O(\frac{1}{\sqrt{M}})$, ainsi qu'une borne sur l'erreur d'estimation à un niveau de confiance donné.

$$\frac{1}{M} \sum_{i=1}^{M} (y_i - f(\theta, x_i))^2 \xrightarrow{p.s} \mathcal{L}(\theta) := \mathbb{E}\big((Y - f(\theta, X))^2\big) \tag{5}$$

Pour alléger les notations, nous introduisons la fonction $H : \mathbb{R}^p \times \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}$ définie par (6). Nous supposons que H est de classe C^1 par rapport à θ et que son gradient $\nabla_\theta H$ est borné par une fonction $g : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}$ telle que $g \in \mathbb{L}_1(p_{X,Y})$. Sous ces hypothèses, le théorème de dérivation des intégrales à paramètres garantit l'existence de $\nabla_\theta \mathcal{L}$, et nous avons alors (7) :

$$H(\theta, x, y) = (y - f(\theta, x))^2 \tag{6}$$

$$\nabla_\theta \mathcal{L} = \mathbb{E}\left[\nabla_\theta H(\theta, X, Y)\right] \tag{7}$$

Supposons maintenant que les conditions du théorème de Robbins-Siegmund [10] (dans sa version applicable au SGD) sont satisfaites, à savoir :

— $\mathcal{L}$ est coercive, de classe C^1, et son gradient $\nabla_\theta \mathcal{L}$ est lipschitzien ;

— $\theta_0 \in \mathbb{R}^p$;

— $||\nabla_\theta \mathcal{L}(\theta)||^2 \leq C(1 + \mathcal{L}(\theta))$;

— $||\nabla_\theta H(\theta, X, Y)||_{L_2} \leq C\sqrt{1 + \mathcal{L}(\theta)}$;

— La suite de pas $(\gamma_n)_n$ appartient à $(\mathbb{R}^+)^{\mathbb{N}}$ et vérifie $\sum_{n\geq 1} \gamma_n = +\infty$ et $\sum_{n\geq 1} \gamma_n^2 < +\infty$.

Alors, en considérant des réalisations i.i.d. $(U_n)_{n\geq 1}$ d'une loi uniforme $\sim \mathcal{U}(\{1, 2, \ldots, M\})$, la procédure (8) converge presque sûrement vers θ^* si $\{\theta^*\} = \{\nabla_\theta \mathcal{L} = 0\}$. Si, typiquement dans le cas d'une fonction coût non strictement convexe, cette condition n'est pas satisfaite, alors le théorème multicible garantit la convergence vers une composante connexe θ_∞ de l'ensemble $\{\nabla_\theta \mathcal{L} = 0\} \cap \{\mathcal{L} = l\}$ pour un certain niveau l. Dans ce cas, la procédure peut se retrouver piégée dans un point selle qui n'est pas un minimum global. Toutefois, un théorème dû à Duflo et al [1] stipule que la procédure (8) converge vers un point selle indésirable avec probabilité nulle. Théoriquement, cette propriété fonde la **robustesse de l'approche stochastique** comparée à l'approche déterministe, en particulier dans les contextes non convexes.

$$\theta_{n+1} = \theta_n - \gamma_{n+1} \nabla_\theta H(\theta_n, X_{U_{n+1}}, Y_{U_{n+1}}) \tag{8}$$

Il existe d'autres variantes de la descente de gradient stochastique, notamment dans le cadre de l'apprentissage profond où l'on considère des sous-ensembles de données (mini-batchs) de taille $m_{\text{batch}} \ll M$. La procédure (9) constitue un compromis entre, d'une part, la réduction du coût de calcul du gradient par rapport à une approche déterministe, où il faudrait l'évaluer sur l'ensemble du jeu de données, et, d'autre part, la diminution de la variance de son estimation par rapport à une descente de gradient stochastique pure. La convergence de la méthode mini-batch a été démontrée pour les algorithmes Adam [7], AdaDelta [13] et RMSProp [12].

$$\theta_{n+1} = \theta_n - \gamma_{n+1} \frac{1}{n_{batch}} \sum_{i=1}^{n_{batch}} \nabla_\theta H(\theta_n, X_{U_i^{batch}}, Y_{U_i^{batch}}) \tag{9}$$

L'approche stochastique a été adoptée par les pionniers de l'apprentissage profond, notamment dans le développement d'algorithmes de type descente de gradient stochastique avec momentum (ADAM, RMSprop...). Ce succès témoigne de l'efficacité de la perturbation des algorithmes déterministes par un bruit endogène. Dans cette perspective, les praticiens se sont intéressés au "boosting" des algorithmes classiques par l'ajout d'un bruit exogène de type brownien, ce qui a motivé la naissance de la descente de gradient par la méthode de Langevin.

2.4 Descente de gradient par méthode de Langevin

Nous supposons dans toute la suite que l'hypothèse (10) est vérifiée, où λ_p désigne la mesure de Lebesgue sur $\mathbb{R}^p$. En plus des hypothèses précédemment formulées sur la fonction coût, nous supposons que $\mathcal{L}$ est α-convexe pour un certain $\alpha > 0$. Cette condition impose une forte régularité à la fonction $\mathcal{L}$. Toutefois, il existe des relaxations de cette hypothèse d'α-convexité, en particulier lorsqu'on restreint $\mathcal{L}$ à vérifier les conditions de type Poincaré, à savoir : $\mathcal{L}$ et $|\nabla_\theta \mathcal{L}|$ sont coercives, et le laplacien de $\mathcal{L}$ satisfait $\Delta\mathcal{L} \leq C(1 + |\nabla_\theta \mathcal{L}|^2)$.

$$(\exists \sigma_0 > 0) : \quad e^{-\frac{\mathcal{L}}{\sigma_0^2}} \in \mathbb{L}_1(\lambda_p) \tag{10}$$

La méthode de Langevin appliquée à la descente de gradient stochastique s'écrit (11) :

$$\theta_{n+1} = \theta_n - \gamma_{n+1} \nabla_\theta H(\theta_n, X_{U_{n+1}}, Y_{U_{n+1}}) + \sigma\sqrt{\gamma_{n+1}} Z_{n+1} \tag{11}$$

Elle est principalement utilisée pour l'entraînement des réseaux de neurones génératifs, en raison de sa capacité à éviter les points selles indésirables, mais aussi à identifier les minimas globaux par le biais d'une distribution cible (12) qui se concentre autour de ces derniers. L'équation de

Fokker-Planck permet de démontrer que cette distribution est une mesure ergodique de l'équation différentielle stochastique de Langevin (13). Cette dernière admet une solution forte dès lors que $\nabla_\theta \mathcal{L}$ est lipschitzienne. Le terme de volatilité σ est supposé borné par σ_0, et C_σ représente une constante de normalisation assurant que la distribution est bien une densité de probabilité.

$$\nu_\sigma(d\theta) = C_\sigma \, e^{-\frac{(\mathcal{L}(\theta)-\mathcal{L}(\theta^*))}{\sigma^2}} \tag{12}$$

$$dX_t = -\nabla_\theta \mathcal{L}(X_t)dt + \sigma dW_t \tag{13}$$

Sous les hypothèses précédentes, et en considérant des réalisations i.i.d. $(Z_n)_n$ issues d'une loi gaussienne $\sim \mathcal{N}(0, I_p)$, la discrétisation de l'équation (13) par le schéma d'Euler-Maruyama (14), ainsi que la méthode de Langevin appliquée à la descente de gradient stochastique (11), convergent en loi vers la distribution ν_σ.

$$\bar{X}_{n+1} = \bar{X}_n - \gamma_{n+1}\nabla_\theta \mathcal{L}(\bar{X}_n) + \sigma\sqrt{\gamma_{n+1}}Z_{n+1} \tag{14}$$

2.5 Descente de gradient par méthode de Langevin pré-conditionnée

La méthode de Langevin avec volatilité constante permet de rendre l'optimisation plus robuste en augmentant la capacité d'exploration de l'espace des états et en évitant les points selles indésirables. Nous présentons ci-dessous une version préconditionnée de la procédure de Langevin, dans laquelle nous considérons une volatilité locale satisfaisant $(\forall \theta \in \mathbb{R}^p) : \vartheta(\theta) \in \mathcal{M}_p(\mathbb{R})$. Cette volatilité doit également satisfaire une condition d'ellipticité, garantissant ainsi l'existence d'une solution forte pour l'équation différentielle stochastique (15). Sous ces hypothèses, nous revisitons le schéma d'Euler-Maruyama (16), ainsi que la mise à jour des poids dans le cadre de la méthode de Langevin stochastique (17). Contrairement à la méthode de Langevin classique, cette approche introduit une correction supplémentaire du drift (18), donnée par $\Lambda(\theta) = \left(\sum_{j=1}^p \partial_{\theta^j}(\vartheta\vartheta^T)_{ij}\right)_{i=1:p}$, et permet de moduler le bruit exogène ajouté en fonction de l'état actuel, offrant ainsi un contrôle plus fin du processus d'optimisation.

$$dX_t = b(X_t)dt + \sigma\vartheta(X_t)dW_t \tag{15}$$

$$\bar{X}_{n+1} = \bar{X}_n - \gamma_{n+1}b(\bar{X}_n) + \sigma\sqrt{\gamma_{n+1}}\vartheta(\bar{X}_n)Z_{n+1} \tag{16}$$

$$\theta_{n+1} = \theta_n - \gamma_{n+1}\vartheta\vartheta^T(\theta_n)\nabla_\theta H(\theta_n, X_{U_{n+1}}, Y_{U_{n+1}}) + \sigma^2\gamma_{n+1}\Lambda(\theta_n) + \sigma\sqrt{\gamma_{n+1}}\vartheta(\bar{X}_n)Z_{n+1} \tag{17}$$

$$b = \sigma^2\Lambda - (\vartheta\vartheta^T)\nabla_\theta \mathcal{L} \tag{18}$$

La procédure (17) converge vers θ^*. Le choix optimal consiste à adopter une volatilité de la forme $\sigma_n = \frac{c}{\sqrt{\log(n)}}$, décroissante par paliers. Cette stratégie permet une atténuation progressive du bruit exogène (phénomène de recuit), ce qui conduit à une convergence en probabilité vers la valeur cible. Ce résultat est bien plus fort que dans le cas d'une volatilité constante, où l'on obtient seulement une convergence en loi vers la mesure ergodique du processus de Langevin. Enfin, d'autres variantes de la méthode de Langevin ont été proposées dans [2], où les auteurs considèrent un terme de volatilité σ dépendant du temps, ainsi que des processus évoluant par plateaux. Dans les deux cas, la convergence de la procédure de Langevin est établie en distance de Wasserstein $\mathcal{W}_1$, à l'aide d'une méthode dite de stratégie domino.

2.6 Vitesses de convergence et choix pratiques

Pour les algorithmes de descente de gradient stochastique ainsi que pour la descente de gradient déterministe à pas variable, et sous les hypothèses assurant la convergence de la procédure, la vitesse de convergence forte en norme $\mathbb{L}_2$ est de l'ordre $O(\sqrt{\gamma_n})$ lorsque le pas est de la forme $\gamma_n = \frac{\gamma_1}{n^r}$ avec $0 < r < 1$, et que la fonction coût $\mathcal{L}$ vérifie l'hypothèse d'α-convexité. Cette hypothèse peut toutefois être relâchée afin d'adapter notre analyse au cadre plus complexe de l'apprentissage profond. Par ailleurs, le choix du paramètre r doit être fait avec soin, dans le but de trouver un compromis adéquat entre robustesse et vitesse de convergence. Le principe de moyennisation peut également s'avérer pertinent. En effet, sous des hypothèses de régularité en norme $\mathbb{L}_p$, de croissance linéaire du gradient $\nabla_\theta H$, et en supposant que $\nabla_\theta \mathcal{L}$ vérifie une hypothèse de différentiation rapide, le théorème central limite de Ruppert et Polyak [9] assure la convergence en loi de la moyenne empirique centrée $\bar{\theta}_n = \frac{1}{n}\sum_{i=1}^n \theta_i - \theta^*$ vers une variable gaussienne centrée, dont la variance dépend à la fois de la fonction coût et de θ^*, avec une vitesse d'ordre $O(\frac{1}{\sqrt{n}})$. Si la fonction $\mathcal{L}$ est plus régulière, des vitesses de convergence faibles de l'ordre $O(\gamma_n)$ peuvent être atteintes. En pratique, on retiendra que la vitesse de convergence forte de la procédure de Langevin est généralement de l'ordre $O(\sqrt{\gamma_n})$.

Dans la section suivante, nous abordons une problématique d'apprentissage profond qui s'inscrit dans le cadre du contrôle optimal stochastique, où la stratégie de contrôle est définie via un réseau de neurones markovien.

3 Contrôle stochastique optimal et méthode de Langevin

Nous présentons ici la formulation générale d'un problème de contrôle stochastique, dans lequel la stratégie d'optimalité est définie à l'aide d'un ou plusieurs réseaux de neurones. Les résultats évoqués s'appuient sur les travaux de Pierre Bras et Gilles Pagès dans [3].

3.1 Cadre théorique

3.1.1 Problème à résoudre

Nous nous intéressons à la résolution du problème (19), où $b : \mathbb{R}^{d_1} \times \mathbb{R}^{d_3} \to \mathbb{R}^{d_1}$, $\sigma : \mathbb{R}^{d_1} \times \mathbb{R}^{d_3} \to \mathcal{M}_{d_1,d_2}(\mathbb{R})$, W est un mouvement brownien à valeurs dans $\mathbb{R}^{d_2}$, et u un processus continu à valeurs dans $\mathbb{R}^{d_3}$, adapté à la filtration naturelle $(\mathcal{F}_t)_{t\geq 0}$. On suppose $T > 0$, $G : [0,T] \times \mathbb{R}^{d_1} \to \mathbb{R}$ et $F : \mathbb{R}^{d_1} \to \mathbb{R}$. Pour approximer l'équation différentielle stochastique (20), nous utilisons un schéma d'Euler-Maruyama (22) construit à partir d'une subdivision régulière (21). Cette même subdivision est utilisée dans (23) pour évaluer la fonction coût J le long d'une trajectoire donnée. Nous étudions ensuite deux configurations pour la modélisation du contrôle. Dans la première, le contrôle est représenté par un unique réseau de neurones dépendant du temps, paramétré par $\theta \in \mathbb{R}^d$, et le contrôle u est discrétisé selon la procédure décrite en (24). Dans la seconde configuration, nous considérons N réseaux de neurones, chacun associé à un pas de discrétisation $(t_k)_{0\leq k\leq N-1}$, et paramétré par un vecteur $\theta = (\theta_0, \theta_1, \ldots, \theta_{N-1}) \in (\mathbb{R}^d)^N$. La discrétisation du contrôle est alors définie par (25). Sans perte de généralité, nous noterons respectivement $\bar{J}$, $\bar{X}$ et $\bar{u}$ les discrétisations des grandeurs J, X et u.

$$\min_u J(u) := \mathbb{E}\left[\int_0^T G(t, X_t)dt + F(X_T)\right] \tag{19}$$

$$dX_t = b(X_t, u_t)dt + \sigma(X_t, u_t)dW_t \tag{20}$$

$$(t_k)_{0\leq k\leq N} = (\frac{kT}{N})_{0\leq k\leq N} \tag{21}$$

$$\bar{X}^{\theta}_{t_{k+1}} = \bar{X}^{\theta}_{t_k} + (t_{k+1}-t_k)b(\bar{X}^{\theta}_{t_k}, \bar{u}_{k,\theta}(\bar{X}^{\theta}_{t_k})) + \sqrt{t_{k+1}-t_k}\sigma(\bar{X}^{\theta}_{t_k}, \bar{u}_{k,\theta}(\bar{X}^{\theta}_{t_k}))Z_{k+1} \ , \quad \left[(Z_i)_{1\leq i\leq N} \ iid \sim \mathcal{N}(0, I_{d_2})\right] \tag{22}$$

$$\min_{\theta} \bar{J}(\bar{u}_\theta) := \sum_{k=0}^{N-1}(t_{k+1}-t_k)G(t_{k+1}, \bar{X}^{\theta}_{t_{k+1}}) + F(\bar{X}^{\theta}_T) \tag{23}$$

$$u_{t_k} = \bar{u}_\theta(t_k, X_{t_k}) \ \ \& \ \ \bar{u}_{k,\theta} = \bar{u}(t_k, .) \tag{24}$$

$$u_{t_k} = \bar{u}_{\theta^k}(X_{t_k}) \ \ \& \ \ \bar{u}_{k,\theta} = \bar{u}_{\theta^k} \tag{25}$$

3.1.2 Lien avec l'apprentissage profond et l'optimisation stochastique

Afin de faire le lien avec le problème d'apprentissage supervisé étudié dans la section précédente, nous considérons M réalisations $(X^i_0)_{i=1:M}$ de l'état initial du processus $(X_t)_t$. Le modèle présenté dans [3] consiste à utiliser un réseau de neurones qui comporte soit une seule couche dépendante du temps, comme illustré dans la figure 1, soit N couches, chacune associée à un pas de temps, comme le montre le schéma 2. À partir des réalisations initiales $(X^i_0)_{i=1:M}$ et pour une valeur fixée du vecteur de poids θ_n, on obtient alors les réalisations de la fonction coût $(J^i_{\theta_n})_{i=1:M}$. Un mini-batch de taille n_{batch} est ensuite sélectionné de manière aléatoire, et l'on applique l'une des mises à jour définies dans (26) ou (27). Ces dernières font référence à la section précédente, dans laquelle nous avons présenté la descente de gradient stochastique par méthode de Langevin préconditionnée. La suite de préconditionneurs utilisée, notée $(P_{i+1})_{i\geq 0}$, correspond quant à elle aux algorithmes Adam [7], Adadelta [13] et RMSprop [12].

$$\theta_{n+1} = \theta_n - \gamma_{n+1}P_{n+1}\frac{1}{n_{batch}}\sum_{i=1}^{n_{batch}}\nabla_\theta \bar{J}(\bar{u}_{\theta_n}, (Z^{i,n+1}_k)_{1\leq k\leq N}) \tag{26}$$

$$\theta_{n+1} = \theta_n - \gamma_{n+1}P_{n+1}\frac{1}{n_{batch}}\sum_{i=1}^{n_{batch}}\nabla_\theta \bar{J}(\bar{u}_{\theta_n}, (Z^{i,n+1}_k)_{1\leq k\leq N}) + \sigma_{n+1}\sqrt{\gamma_{n+1}}\mathcal{N}(0, P_{n+1}) \tag{27}$$

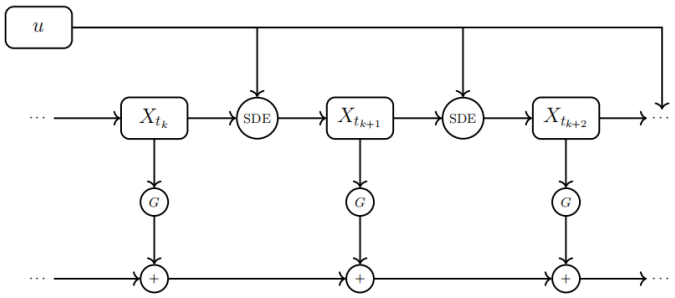

FIGURE 1 – Configuration où le contrôle est défini par un seul réseau de neurones markovien dépendant du temps.

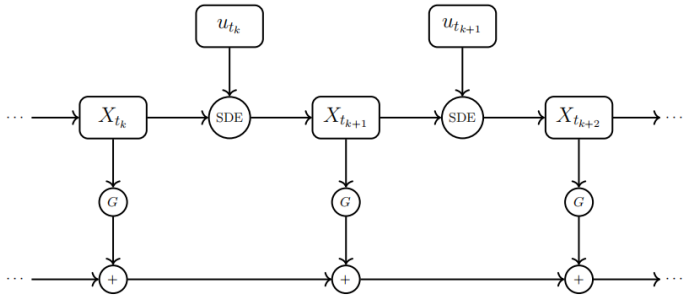

FIGURE 2 – Configuration où le contrôle est défini par plusieurs réseaux de neurones markoviens chacun caractérisant un pas de discrétisation.

3.1.3 Différentiation automatique pour le calcul du gradient

Dans cette partie, nous discutons de la méthode de différentiation automatique forward utilisée pour calculer le gradient de la fonction coût, par rapport aux composantes du vecteur poids, le long d'une trajectoire simulée. Nous nous baserons sur les travaux de Glasserman et Giles dans [6]. Pour adopter un formalisme qui correspond a notre problème de contrôle stochastique optimal, nous fixons $\theta \in \mathbb{R}^d$, $k \in \{0, ..N-1\}$ et enfin $j \in \{1, ..d\}$. Pour simplifier les notations, nous posons $\delta_{ij}(k) = \frac{\partial \bar{X}^\theta_{t_k,i}}{\partial \theta_j}$ la dérivée partielle de la i-ème composante du vecteur $\bar{X}^\theta_{t_k}$ par rapport a la j-ème composante du vecteur θ. En partant de (22) et en appliquant la règle de la chaîne, nous obtenons la relation (28) qui permet de calculer les composantes de $\nabla_\theta \bar{J}(\bar{u}_\theta)$ de manière incrémentale. En effet, $\frac{\partial \bar{J}}{\partial \theta_j}(\bar{u}_\theta)$ vérifie (29). Les coefficients des matrices $\tilde{b}(k)$ (30) et $\tilde{\sigma}(k)$ (31) dépendent uniquement de l'expression du drift b, des dérivées partielles de la fonction réseau de neurone par rapport aux entrées et des composantes d'un vecteur gaussien $Z_{k+1} \sim \mathcal{N}(0, I_{d_2})$.

$$\delta_{ij}(k+1) = \delta_{ij}(k) + (t_{k+1} - t_k)\left[\sum_{l=1}^{d_1} \tilde{b}_{il}(k)\delta_{lj}(k)\right] + \sqrt{t_{k+1} - t_k}\left[\sum_{l=1}^{d_1} \tilde{\sigma}_{il}(k)\delta_{lj}(k)\right] \tag{28}$$

$$\frac{\partial \bar{J}}{\partial \theta_j}(\bar{u}_\theta) = \sum_{l=1}^{d_1}\left[\delta_{lj}(N)\frac{\partial F}{\partial x_l}(\bar{X}^\theta_{t_N}) + \sum_{k=0}^{N-1}(t_{k+1} - t_k)\delta_{lj}(k)\frac{\partial G}{\partial x_l}(t_k, \bar{X}^\theta_{t_k})\right] \tag{29}$$

$$\tilde{b}_{il}(k) = \frac{\partial b_i}{\partial x_l}(\bar{X}^\theta_{t_k}, \bar{u}_{k,\theta}(\bar{X}^\theta_{t_k})) + \sum_{r=1}^{d_3}\frac{\partial b_i}{\partial u_r}(\bar{X}^\theta_{t_k}, \bar{u}_{k,\theta}(\bar{X}^\theta_{t_k}))\frac{\partial \bar{u}_{k,\theta}}{\partial x_l}(\bar{X}^\theta_{t_k}) \tag{30}$$

$$\tilde{\sigma}_{il}(k) = \sum_{m=1}^{d_2}\left[\frac{\partial \sigma_{im}}{\partial x_l}(\bar{X}^\theta_{t_k}, \bar{u}_{k,\theta}(\bar{X}^\theta_{t_k})) + \sum_{r=1}^{d_3}\frac{\partial \sigma_{im}}{\partial u_r}(\bar{X}^\theta_{t_k}, \bar{u}_{k,\theta}(\bar{X}^\theta_{t_k}))\frac{\partial \bar{u}_{k,\theta}}{\partial x_l}(\bar{X}^\theta_{t_k})\right]Z_{k+1}(m) \tag{31}$$

3.2 Optimiseurs

L'objectif de cette section est de présenter les optimiseurs Adam, Adadelta et RMSProp, puis de comparer chacun d'eux à sa version Langevin.

3.2.1 RMSProp

Dans une descente de gradient classique, un unique taux d'apprentissage s'applique à toutes les composantes, ce qui peut freiner l'optimisation. AdaGrad [5] corrige cette limitation en ajustant le pas

individuellement : pour chaque paramètre, il cumule les carrés de ses gradients et, plus cette somme est grande, plus le pas associé diminue. Les composantes rarement sollicitées conservent donc un pas important, tandis que celles fréquemment mises à jour voient leur pas se réduire.

Cependant, comme l'accumulateur croît sans borne, le taux d'apprentissage dans un réseau profond finit par devenir infime et l'optimisation se bloque. Pour éviter cette décroissance irréversible, RMSProp [12] remplace la somme cumulative par une moyenne exponentielle glissante des carrés des gradients. Ainsi, le dénominateur reste borné et le pas ne s'annule pas. Avec les notations d'Hadamard ($\odot$ pour le produit terme à terme et $\oslash$ pour la division terme à terme), l'algorithme s'écrit 3 :

Parameters: $\alpha, \lambda > 0$

$\mathrm{MS}_{n+1} = \alpha\,\mathrm{MS}_n + (1-\alpha) g_{n+1} \odot g_{n+1}$

$P_{n+1} = \mathrm{diag}\left(\mathbb{1} \oslash \left(\lambda \mathbb{1} + \sqrt{\mathrm{MS}_{n+1}}\right)\right)$

$\theta_{n+1} = \theta_n - \gamma_{n+1} P_{n+1} \cdot g_{n+1}$

FIGURE 3 – Algorithme RMSProp

3.2.2 Adadelta

Adadelta [13] remédie quant à lui à la décroissance irréversible du pas en remplaçant la somme cumulative des carrés des gradients par deux moyennes exponentielles glissantes : l'une pour les carrés des gradients, l'autre pour les carrés des mises à jour passées. À chaque itération, la mise à jour est obtenue en multipliant le gradient par le rapport des racines carrées de ces deux estimations 4. Ce procédé ajuste automatiquement l'échelle des pas (auto-normalisé) et maintient une optimisation efficace, même en profondeur. Cependant, le suivi simultané de deux accumulateurs rend l'algorithme plus coûteux en calcul et en mémoire qu'AdaGrad ou RMSProp.

Require: Decay rate ρ, Constant ϵ
Require: Initial parameter x_1
1: Initialize accumulation variables $E[g^2]_0 = 0$, $E[\Delta x^2]_0 = 0$
2: **for** $t = 1 : T$ **do** %% Loop over # of updates
3: Compute Gradient: g_t
4: Accumulate Gradient: $E[g^2]_t = \rho E[g^2]_{t-1} + (1-\rho) g_t^2$
5: Compute Update: $\Delta x_t = -\frac{\mathrm{RMS}[\Delta x]_{t-1}}{\mathrm{RMS}[g]_t} g_t$
6: Accumulate Updates: $E[\Delta x^2]_t = \rho E[\Delta x^2]_{t-1} + (1-\rho)\Delta x_t^2$
7: Apply Update: $x_{t+1} = x_t + \Delta x_t$
8: **end for**

FIGURE 4 – Algorithme Adadelta

3.2.3 Adam

Adam [7] est aujourd'hui l'optimiseur le plus utilisé en apprentissage profond. Il calcule, pour chaque paramètre, une moyenne exponentielle glissante des gradients (momentum) et une moyenne exponentielle glissante de leurs carrés (comme RMSProp et Adadelta), tous deux corrigés du biais initial (sous-estimation systématique de la moyenne et de la variance du gradient au début de l'entraînement induite par l'initialisation des moyennes exponentielles à zéro). Le pas effectif s'obtient alors en divisant le premier moment par la racine carrée du second 5. Cette mise à jour a l'avantage d'être facile à implémenter, de nécessiter peu de mémoire et d'être très efficace.

Parameters: $\beta_1, \beta_2, \lambda > 0$

$M_{n+1} = \beta_1 M_n + (1-\beta_1) g_{n+1}$

$\mathrm{MS}_{n+1} = \beta_2 \mathrm{MS}_n + (1-\beta_2) g_{n+1} \odot g_{n+1}$

$\widehat{M}_{n+1} = M_{n+1}/(1-\beta_1^{n+1})$

$\widehat{\mathrm{MS}}_{n+1} = \mathrm{MS}_{n+1}/(1-\beta_2^{n+1})$

$P_{n+1} = \mathrm{diag}\left(\mathbb{1} \oslash \left(\lambda \mathbb{1} + \sqrt{\widehat{\mathrm{MS}}_{n+1}}\right)\right)$

$\theta_{n+1} = \theta_n - \gamma_{n+1} P_{n+1} \cdot \widehat{M}_{n+1}.$

FIGURE 5 – Algorithme Adam

3.3 Cadre expérimental

Dans la section suivante, nous comparons les algorithmes RMSProp, Adadelta et Adam à leurs variantes de Langevin à travers différents problèmes. Tout d'abord, nous considérons le cas où le contrôle est modélisé par un unique réseau de neurones dépendant du temps, avant d'étudier le cas où le contrôle est donné à chaque instant par un réseau de neurones distinct. Dans ce dernier, comme les paramètres des réseaux de contrôle successifs sont généralement proches, nous commençons par entraîner un nombre réduit de réseaux (c'est à cette étape que l'on utilise Langevin), avant d'initialiser les poids des réseaux restants avec ceux des réseaux déjà entraînés, puis de les fine-tuner cette fois de manière classique.

Nous allons également tester l'algorithme Layer Langevin qui repose sur l'idée d'ajouter du bruit exogène uniquement à certaines couches du réseau de neurones, plutôt qu'à l'ensemble des paramètres. L'intuition derrière cette méthode est que, dans les réseaux profonds, les non-linéarités les plus complexes se concentrent souvent dans les couches les plus profondes. En ciblant spécifiquement ces couches, le bruit injecté permet de mieux explorer l'espace des paramètres sans perturber l'ensemble du réseau. On parle alors de LL-name p% lorsque seules les p premières couches du réseau reçoivent ce bruit de Langevin.

Pour tout ce qui concerne l'implémentation, nous nous appuierons sur le dépôt GitHub mis à disposition par Pierre Bras [1], tel que présenté dans [3]. Nous réutiliserons également les mêmes valeurs de paramètres que celles employées dans cet article.

4 Méthode de Langevin et contrôle optimal des quotas de pêche

Dans cette section, nous étudions la régulation de la pêche par l'imposition de quotas. Une telle mesure vise à prévenir l'épuisement de la biomasse halieutique, qui pourrait priver les pêcheurs de leurs moyens de subsistance et, dans certaines régions, conduire à une famine.

Le modèle que nous considérons cherche à maintenir la biomasse au voisinage d'un état cible, noté $\mathcal{X}_t \in \mathbb{R}^{d1}$ à l'instant t, tout en pénalisant les variations temporelles du quota. Le cadre mathématique adopté s'inspire des travaux de Gilles Pagès *et al.* [3, 8].

4.1 Cadre théorique

Nous désignons par $X_t \in \mathbb{R}^{d_1}$, la biomasse de poissons pour chaque espèce présente dans le milieu étudié a l'instant t. La valeur de X au cours du temps dépend de différents facteurs :

1. https://github.com/Bras-P/langevin-for-stochastic-control

— $Q_i(t) \in \mathbb{R}^+ \equiv$ poids maximal de poissons de l'espèce $i \in \{1, \dots, d_1\}$ qu'un pêcheur est autorisé à pêcher à l'instant t.

— $qX_i(t) \in \mathbb{R}^+ \equiv$ capacité mécanique d'un bateau à pêcher l'espèce i, proportionnelle à la biomasse disponible. Nous supposons que la valeur de q est la même pour toutes les espèces par souci de simplicité.

— $E(t) \in \mathbb{R}^+ \equiv$ nombre total de bateaux en activité à l'instant t, reflétant l'intensité de la pression de pêche.

— $r \in \mathbb{R}^{d_1} \equiv$ croissance ou décroissance naturelle de la biomasse de chaque espèce, en absence de pêche.

— $\kappa \in \mathcal{M}_{d_1}(\mathbb{R}) \equiv$ matrice modélisant les interactions entre espèces (prédation, compétition).

En introduisant le contrôle $u_t = \left(E(t)\frac{Q_i(t)}{X_i(t)}\right)_{1\le i\le d_1}$, nous obtenons l'équation différentielle stochastique (32), dont le drift reproduit le mécanisme naturel de l'évolution de la biomasse. À ce dernier, nous ajoutons un bruit exogène de type brownien, avec une volatilité constante $\eta \in \mathcal{M}_{d_1,d_2}(\mathbb{R})$. Le contrôle u est supposé borné, appartenant à l'ensemble $[u_m, u_M]^{d_1}$, ce qui permet de limiter l'excès de pêche tout en garantissant un quota minimal pour chaque espèce. Nous introduisons également le produit de Schur $*$, correspondant à la multiplication composante par composante dans $\mathbb{R}^{d_1}$. L'EDS (32) admet une unique solution forte : en effet, le terme de diffusion est linéaire en X_t, donc lipschitzien à croissance linéaire, tandis que le drift est une fonction quadratique en X_t, dépendant des paramètres r, κ et du contrôle u_t ; ce dernier étant borné, le drift est localement lipschitzien uniformément en u. Par ailleurs, la solution X_t ne peut pas "exploser" en un temps fini, c'est-à-dire que le processus reste bien défini pour tout $t \ge 0$. On précise que le processus $(X_t)_{t\ge 0}$ est continu et satisfait l'expression (33) par la formule d'Itô. Enfin, si l'on suppose $X_0 \in \mathbb{L}_2$ et $X_0 > 0$ p.s., alors $\mathbb{E}\left(\int_0^T ||X_t||^2 dt\right) < \infty$, ce qui permet de bien définir la fonction coût. En pratique, les auteurs de [3] considèrent une loi normale tronquée sur un compact préalablement défini pour générer X_0, ce qui permet à la fois de garantir que $X_0 \in \mathbb{L}_2$ et de borner le processus, assurant ainsi une définition rigoureuse de la fonction coût discrétisée.

$$dX_t = X_t * ((r - \kappa X_t - u_t)dt + \eta dW_t) \tag{32}$$

$$\forall t \in [0,T],\ X_t = X_0 * \left(e^{(r_i - \frac{||\eta_{i,.}||^2}{2})t - \int_0^t (u_s^i + \langle \kappa_{i,.}, X_s\rangle)ds + \langle \eta_{i,.}, W_t\rangle}\right)_{1\le i\le d_1} \tag{33}$$

Nous présentons dans (34) le problème de contrôle optimal stochastique à résoudre. Ce dernier se scinde en trois parties : un terme de conformité à l'état idéal $\mathcal{X}$, un terme de pénalisation β sur la variation quadratique du contrôle permettant d'éviter des variations journalières brusques de la biomasse dues à la méthode de pêche et un terme avec pénalisation α pour éviter des quotas de pêche excessivement petits(35) . Nous noterons que β est commun a toutes les espèces de poissons dans notre modèle tandis que $\alpha \in \mathbb{R}^{d_1}$ caractérise chaque espèce.

$$\min_u J(u) := \mathbb{E}\left[\int_0^T (||X_t - \mathcal{X}_t||^2 - \langle \alpha, u_t\rangle)dt + \beta[u]^{0,T}\right] \tag{34}$$

$$[u]^{0,T} = \lim_{\delta t\to 0} [u_N]^{0,T} := \sum_{k=0}^{N-1} ||u_{t_{k+1}} - u_{t_k}||^2 \tag{35}$$

Pour résoudre (34) nous utiliserons le cadre théorique détaillé dans la section 3. Le schéma d'Euler associé au processus de biomass vérifie(36) et la discrétisation d'une réalisation de la fonction coût est donnée par (37).

$$\bar{X}^{\theta}_{t_{k+1}} = \bar{X}^{\theta}_{t_k} + \bar{X}^{\theta}_{t_k} * \left((t_{k+1} - t_k)(r - \kappa \bar{X}^{\theta}_{t_k} - u_{k,\theta}(\bar{X}^{\theta}_{t_k})) + \eta \sqrt{t_{k+1} - t_k} Z_{k+1} \right) \tag{36}$$

$$\bar{J}(\bar{u}_\theta) = \sum_{k=0}^{N-1} (t_{k+1} - t_k) \left[||\bar{X}^{\theta}_{t_k} - \mathcal{X}_{t_k}||^2 - \langle \alpha, \bar{u}_{k,\theta}(\bar{X}^{\theta}_{t_k}) \rangle \right] + \beta ||\bar{u}_{k+1,\theta}(\bar{X}^{\theta}_{t_{k+1}}) - \bar{u}_{k,\theta}(\bar{X}^{\theta}_{t_k})||^2 \tag{37}$$

Une heuristique sur l'existence d'un minimiseur :

Nous discutons dans cette partie l'existence d'un minimiseur lorsque la fonction coût J est définie aussi bien dans un cadre discret que continu. Pour simplifier l'analyse, nous nous plaçons dans le cas $d_1 = 2$. Le contrôle u, en tant que processus continu, vérifie $u \in (C([0,T] \times \Omega, [u_m, u_M]^2), ||\cdot||_\infty)$, c'est-à-dire que nous minimisons J sur un fermé borné de l'espace de Banach des processus continus et bornés, à valeurs dans $\mathbb{R}^2$, muni de la norme sup. Dans ce contexte, cette norme est définie par $||u||_\infty = \sup_{w \in \Omega} \sup_{t \in [0,T]} \max(|u^1_t(w)|, |u^2_t(w)|)$. La fonction J, considérée comme fonctionnelle de u, est continue. Heuristiquement, pour démontrer ce résultat, on considère une suite de contrôles $(u^n_t)_{0 \le t \le T}$ qui converge uniformément vers $(u_t)_{0 \le t \le T}$, et l'on montre que le processus $(X^n_t)_{0 \le t \le T}$ induit par cette suite vérifie (38). Cette propriété est valide lorsque les termes de drift et de diffusion sont à croissance linéaire et lipschitziens par rapport au contrôle $(u_t)_{0 \le t \le T}$ et au processus $(X_t)_{0 \le t \le T}$. La démonstration repose sur une combinaison des inégalités de Young, de Doob, de Cauchy-Schwarz ainsi que du lemme de Grönwall. Dans notre cadre, le drift est seulement localement lipschitzien ; néanmoins, si le processus $(X_t)_{0 \le t \le T}$ est borné au sens de la norme infinie par exemple, en définissant une biomasse maximale et en tronquant le processus sur un compact alors l'équation (38) reste valable. Cette hypothèse peut être encore relâchée en supposant que X_0 est tiré selon une loi à support compact, auquel cas le processus discrétisé issu du schéma d'Euler est automatiquement borné. Enfin, pour établir la convergence de la variation quadratique discrète $[u^n]^{0,T}$ vers $[u]^{0,T}$, on part de la définition (35) : pour un N fixé, on choisit $\epsilon > 0$ et n tel que $||u^n - u||_\infty < \frac{\epsilon}{4N}$, ce qui implique $|[u_N]^{0,T} - [u^n_N]^{0,T}| < 4\epsilon u_M$. Cela assure la convergence de la variation quadratique discrète. Concernant le terme de pénalisation en α, le raisonnement est similaire : il suffit d'appliquer l'inégalité de Cauchy-Schwarz, puis de majorer la norme euclidienne de $u^n - u$ par une constante multipliée par sa norme infinie. En combinant ces trois résultats, et en tenant compte des hypothèses susmentionnées sur $(X_t)_{0 \le t \le T}$, nous obtenons l'existence d'un minimiseur dans le cas discret. Pour passer au cas continu, on remplace la variation quadratique par sa forme continue, puis on utilise l'inégalité (39) en choisissant $N = \max(N_1, N_2, N_3)$ et un n tel que $|[u_{N_1}]^{0,T} - [u]^{0,T}| < \frac{\epsilon}{3}$, $|[u_{N_2}]^{0,T} - [u^n_{N_2}]^{0,T}| < \frac{\epsilon}{3}$ et $|[u^n_{N_3}]^{0,T} - [u^n]^{0,T}| < \frac{\epsilon}{3}$.

$$\int_0^T \mathbb{E}||X_t - X^n_t||^2 dt \xrightarrow{n \to \infty} 0 \tag{38}$$

$$|[u]^{0,T} - [u^n]^{0,T}| < |[u]^{0,T} - [u_N]^{0,T}| + |[u_N]^{0,T} - [u^n_N]^{0,T}| + |[u^n_N]^{0,T} - [u^n]^{0,T}| \tag{39}$$

4.2 Résultats et discussion

Les figures 6 et 7 comparent les algorithmes Adam, RMSProp et Adadelta avec leurs variantes de Langevin dans le cas d'un unique réseau de contrôle. La figure 8 présente quant à elle les performances de ces algorithmes dans un cadre multi-réseaux.

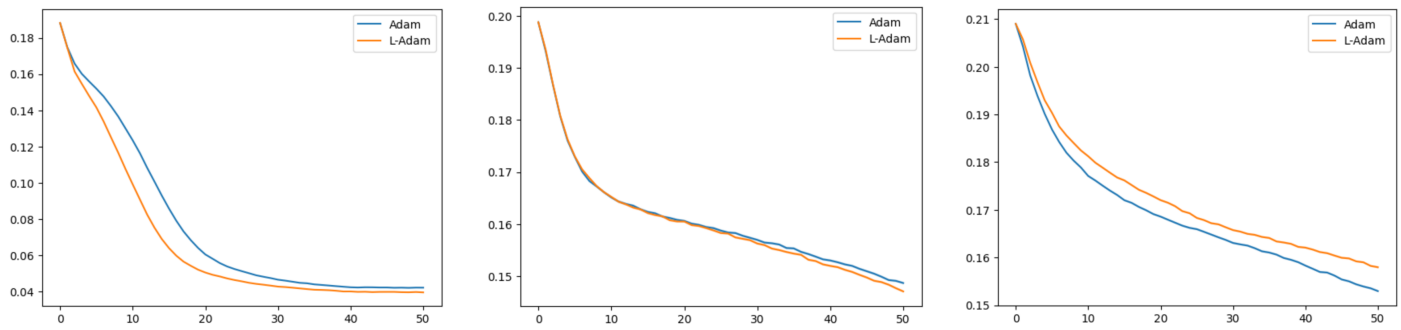

FIGURE 6 – Comparaison des algorithmes Adam et L-Adam lors de l'entraînement pour le problème des quotas de pêche pour un nombre de pas $N = 20, 50, 100$ respectivement et un unique réseau de contrôle.

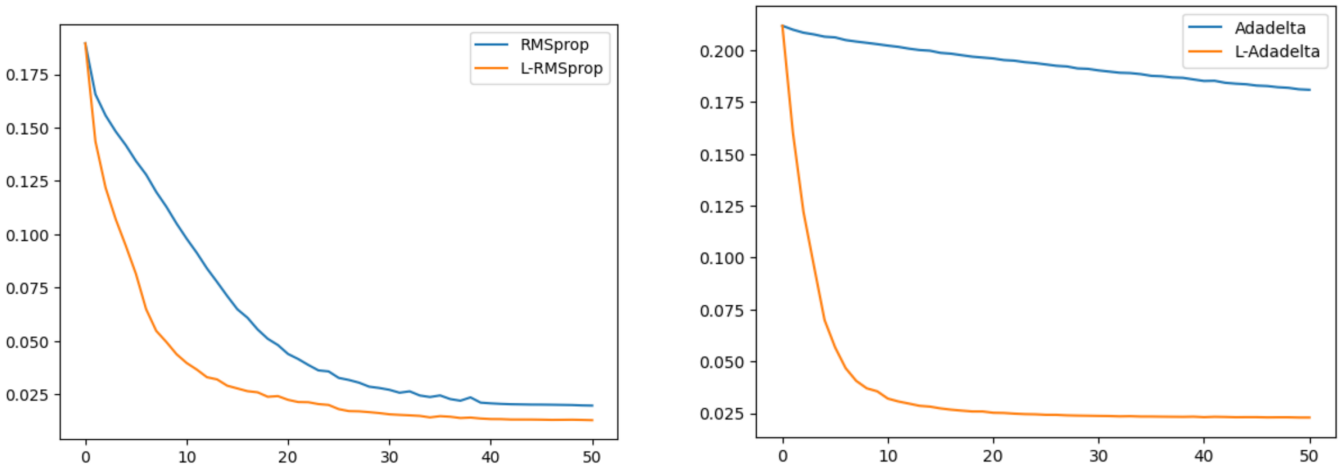

FIGURE 7 – Comparaison des algorithmes RMSProp et L-RMSProp, puis Adadelta et L-Adadelta lors de l'entraînement pour le problème des quotas de pêche avec un nombre de pas $N = 50$ et un unique réseau de contrôle.

Nous observons que les algorithmes de Langevin permettent, dans la majorité de nos expériences, une convergence plus rapide vers une valeur de loss plus faible. Cependant, il est difficile de dire dans quelle mesure le gain est important ou encore quel algorithme en bénéficie le plus. En effet, et comme c'est souvent le cas en optimisation stochastique, ces simulations restent très sensibles aux conditions initiales et donnent des résultats radicalement différents d'un test à l'autre pour peu que l'on retire un paramètre selon une loi normale tronquée. De plus, nous ne constatons pas d'amélioration importante lorsque le nombre de pas de discrétisation N croît.

D'autre part, dans le cas où l'on utilise plusieurs contrôles, on observe dans la plupart des expériences que les algorithmes Layer Langevin offrent de meilleures performances que la méthode de Langevin classique ou les algorithmes standards pour de petites valeurs de N.

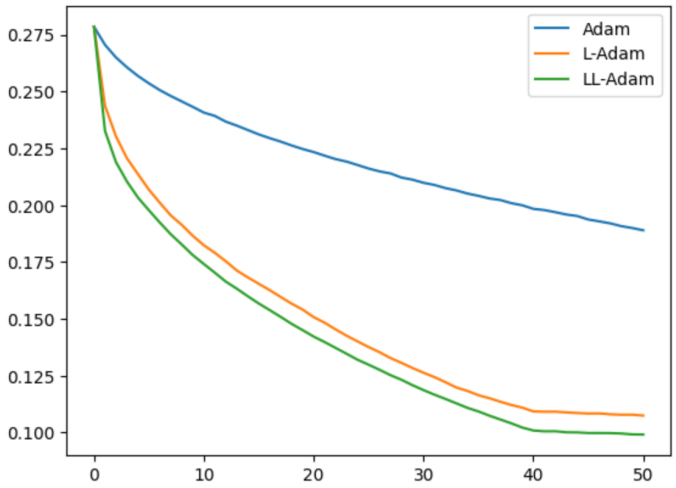

FIGURE 8 – Comparaison des algorithmes Adam, L-Adam et LL-Adam lors de l'entraînement pour le problème des quotas de pêche avec un nombre de pas $N = 10$ et plusieurs réseaux de contrôle.

5 Méthode de Langevin et stratégie optimale de couverture financière

Dans cette section, nous nous intéressons à la recherche d'une stratégie de couverture financière pour un portefeuille d'actifs, en tenant compte des frictions de marché telles que les coûts de transaction, le market impact ainsi que les contraintes de liquidité. Les résultats présentés s'inspirent fortement des travaux de [3] et [4].

5.1 Cadre théorique

Stratégie de trading :

Nous considérons un portefeuille composé de produits dérivés sur au plus d_1 actifs, notés $S_t \in \mathbb{R}^{d_1}$. Soit $\mathcal{H}^\nu$ l'ensemble des stratégies de couverture non contraintes. En pratique, chaque composante u_{t_k} de la stratégie est soumise à des contraintes supplémentaires, liées notamment à la liquidité du marché, à la disponibilité des actifs ou à des restrictions réglementaires. Ces contraintes peuvent également refléter des limitations spécifiques, comme l'interdiction de négocier une option avant sa date d'éligibilité. Par exemple, dans le cas d'une option listée dans trois mois, l'ensemble des positions admissibles avant cette date est réduit à $\{0\}$. Afin de modéliser ces restrictions, nous supposons que, pour chaque instant k, la variable de contrôle u_{t_k} appartient à un sous-ensemble $\mathcal{H}_k$ de $\mathbb{R}^{d_1}$, défini comme l'image d'une application continue $H_k : \mathbb{R}^{d_1(k+1)} \to \mathbb{R}^{d_1}$, $\mathcal{F}_k$-mesurable, avec $\mathcal{H}_k := H_k(\mathbb{R}^{d_1(k+1)})$ et la condition $H_k(0) = 0$.

Étant donnée une stratégie non contrainte $u^\nu \in \mathcal{H}^\nu$, nous définissons par récurrence sa projection contrainte $(H \circ u^\nu)_k$ dans $\mathcal{H}_k$ selon (40). L'ensemble des stratégies admissibles respectant les contraintes est alors donné par $\mathcal{H} := H \circ \mathcal{H}^\nu \subset \mathcal{H}^\nu$, et constitue un sous-ensemble non vide de $\mathcal{H}^\nu$.

$$(H \circ u^\nu)_{t_k} := H_k\left((H \circ u^\nu)_{t_0}, \dots, (H \circ u^\nu)_{t_{k-1}}, u^\nu_{t_k}\right) \tag{40}$$

Stratégie de hedging :

Dans un marché sans coûts de transaction, la richesse de l'agent à l'horizon T est donnée par (41). On rappelle que Z représente la valeur du passif à échéance (par exemple, le payoff d'un produit

dérivé), u_{t_k} la position sur les actifs sous-jacents à l'instant t_k, et S_{t_k} les prix correspondants. En pratique, les coûts de transaction ne peuvent être négligés. En supposant qu'ils sont proportionnels aux quantités échangées, le payoff à maturité s'écrit alors comme dans (42).

$$\mathrm{PL}_T(Z, u, S) = -Z + \sum_{k=0}^{N-1} u_{t_k} \cdot (S_{t_{k+1}} - S_{t_k}) \tag{41}$$

$$\mathrm{PL}_T(Z, u, S, c) := -Z + \sum_{k=0}^{N-1} u_{t_k} \cdot (S_{t_{k+1}} - S_{t_k}) - \sum_{k=0}^{N} c_{t_k}.S_{t_k} * |u_{t_k} - u_{t_{k-1}}| \tag{42}$$

Nous adoptons les conventions $u_{t_{-1}} = u_{t_N} := 0$, ce qui impose une liquidation complète à l'échéance.

Mesure de risque et fonction objectif :

Nous cherchons le montant minimal que l'agent doit facturer afin que sa position terminale soit acceptable, à condition qu'elle soit couverte de manière optimale. Autrement dit, il s'agit de minimiser le risque associé au payoff à maturité. Pour cela, nous cherchons à minimiser (43), où ρ désigne une mesure de risque convexe définie sur $\mathbb{L}_1(\Omega)$. La fonction π est bien une mesure de risque convexe dès lors que l'ensemble $\mathcal{H}$ est lui-même convexe. En pratique, nous faisons le choix d'une mesure de risque définie à partir d'une fonction de perte l selon (44). Cette fonction π est bien convexe lorsque l est continue, convexe et croissante. Par ailleurs, la propriété suivante est satisfaite : si $\pi(0) > -\infty$, alors $\forall X,\ \pi(X) > -\infty$, ce qui est garanti si l'espace de probabilité initial est discret ou si l'on se restreint à $(\mathbb{L}_\infty, ||\cdot||_\infty)$. Cette propriété assure que l'infimum défini dans (43) est bien fini. Afin de satisfaire ces conditions, nous choisissons la mesure de risque comme étant la CVaR (Conditional Value at Risk) au niveau de confiance α, telle que définie en (45). La fonction coût s'écrit alors sous la forme (46). Enfin, nous pouvons réécrire (46) de manière à effectuer la minimisation sur l'ensemble non contraint $\mathcal{H}^v$, ce qui nous conduit à l'équation (47).

$$\pi(-Z) := \inf_{u\in\mathcal{H}} J(u) := \rho\left(-Z + \sum_{k=0}^{N-1} u_{t_k} \cdot (S_{t_{k+1}} - S_{t_k})) - \sum_{k=0}^{N} c_{t_k}.S_{t_k} * |u_{t_k} - u_{t_{k-1}}| \right) \tag{43}$$

$$\forall X \in \mathbb{L}_1(\Omega)\ : \rho(X) = \inf_{w\in\mathbb{R}} (w + \mathbb{E}(l(-X - w))) \tag{44}$$

$$\forall x \in \mathbb{R}\ : l(x) = \frac{1}{1-\alpha} x^+ \tag{45}$$

$$\inf_{u\in\mathcal{H}} J(u) = \inf_{u\in\mathcal{H}} \inf_{w\in\mathbb{R}} (w + \frac{1}{1-\alpha}\mathbb{E}((Z - \sum_{k=0}^{N-1} u_{t_k} \cdot (S_{t_{k+1}} - S_{t_k})) + \sum_{k=0}^{N} c_{t_k}.S_{t_k} * |u_{t_k} - u_{t_{k-1}}| - w)^+) \tag{46}$$

$$\inf_{u\in\mathcal{H}^v} J(u) = \inf_{u\in\mathcal{H}^v} \inf_{w\in\mathbb{R}} (w + \frac{1}{1-\alpha}\mathbb{E}((Z - \sum_{k=0}^{N-1} (H\circ u)_{t_k} \cdot (S_{t_{k+1}} - S_{t_k}) + \sum_{k=0}^{N} c_{t_k}.S_{t_k} * |(H\circ u)_{t_k} - (H\circ u)_{t_{k-1}}| - w)^+) \tag{47}$$

Une heuristique sur l'existence d'un minimiseur :

Nous discutons dans cette partie, de façon succincte, l'existence d'un minimiseur pour le problème (43), ou de manière équivalente pour (46). Pour simplifier l'analyse, nous supposons que

$Z \in (\mathbb{L}_\infty, ||\cdot||_\infty)$, et nous considérons $\mathcal{H}^\nu$ comme un sous-ensemble compact de l'espace vectoriel des processus discrets bornés à $N+1$ éléments, que nous notons $\mathcal{E}$, muni de sa norme infinie $||\cdot||_{\infty,\mathcal{E}}$. Cette hypothèse revient à supposer que, dans un marché idéalisé et en l'absence de contraintes explicites, le nombre de parts détenues dans chaque actif par l'agent ne peut dépasser une certaine limite. Il en découle que $\mathcal{H}$ est également un compact, la fonction de projection H étant continue. L'application h définie par (48) est continue puisqu'elle correspond à la Conditional Value at Risk. L'application g donnée par (49) est également bien définie : en effet, la suite $(u_{t_k})_{k=0:N}$ étant bornée, et sous une initialisation bornée du vecteur de prix S_0 et de la volatilité V_0, le processus discrétisé issu du schéma d'Euler est lui aussi borné. De plus, g étant construite à partir de sommes, différences et normes appliquées aux composantes du processus, elle est continue. Il en résulte que $J = f \circ g$ est continue. Le problème revient donc à minimiser une fonction continue sur un compact, ce qui garantit l'existence d'un minimiseur. L'extrapolation au cas continu se fait naturellement en remplaçant les sommes par des intégrales, l'espace $\mathcal{E}$ par l'espace des processus continus bornés, et en considérant des processus de prix spot $(S_t)_{0\le t\le T}$ et de volatilité $(V_t)_{0\le t\le T}$ bornés, ce que l'on peut assurer par une troncature appropriée.

$$\begin{array}{rcl} h : (\mathbb{L}_\infty, ||.||_\infty) & \longrightarrow & (\mathbb{R}, |.|) \\ X & \longmapsto & \inf\limits_{w\in\mathbb{R}} \left(w + \frac{1}{1-\alpha}\mathbb{E}(X-w)^+\right) \end{array} \tag{48}$$

$$\begin{array}{rcl} g : (\mathcal{E}, ||.||_{\infty,\mathcal{E}}) & \longrightarrow & (\mathbb{L}_\infty, ||.||_\infty) \\ (u_{t_k})_{k=0:N} & \longmapsto & Z - \sum_{k=0}^{N-1} u_{t_k}\cdot(S_{t_{k+1}} - S_{t_k})) + \sum_{k=0}^{N} c_{t_k}.S_{t_k} * |u_{t_k} - u_{t_{k-1}}| \end{array} \tag{49}$$

Dynamique des actifs et pricing :

Nous considérons un modèle de Heston multidimensionnel pour modéliser un vecteur de prix de dimension $d_1' = \frac{d_1}{2}$ ainsi que la volatilité associée, comme indiqué dans (50). Ce modèle est paramétré par une volatilité de volatilité η, un vecteur de prix initial $s_0 \in (\mathbb{R}^+)^{d_1'}$, une moyenne de long terme pour le processus de volatilité notée $b \in (\mathbb{R}^+)^{d_1'}$, et une force de rappel vers cette moyenne, notée $a \in (\mathbb{R}^+)^{d_1'}$. Les couples $(S_t^{1,i}, S_t^{2,i})_{1\le i\le d_1'}$ représentant respectivement les prix spot et les swaps de variance sur la volatilité constituent le portefeuille utilisé pour couvrir le passif Z, ce dernier étant choisi comme une somme de calls sur les spots, conformément à (52). En appliquant le lemme d'Itô, nous obtenons également l'expression du prix du swap de variance, donnée par (51). Enfin, le contrôle u est modélisé par un réseau de neurones feedforward et vérifie la structure définie en (53).

$$\begin{cases} dS_t^{1,i} = \sqrt{V_t^i} S_t^{1,i} dB_t^i, \quad S_0^{1,i} = s_0^i \\ dV_t^i = a^i(b^i - V_t^i)dt + \eta^i\sqrt{V_t^i}dW_t^i, \quad V_0^i = v_0^i \end{cases} \tag{50}$$

$$S_t^{2,i} := \mathbb{E}\left[\int_0^T V_s^i ds \,\middle|\, \mathcal{F}_t\right] = \int_0^t V_s^i ds + L^i(t, V_t^i) \quad \left[L^i(t,v) := \frac{v-b^i}{a^i}\left(1 - e^{-a^i(T-t)}\right) + b^i(T-t)\right] \tag{51}$$

$$Z = \sum_{i=1}^{d_1'} \left(S_T^{1,i} - K^i\right)_+ \tag{52}$$

$$u_{t_k}^\theta = u_{k,\theta}(S_{t_k}, V_{t_k}, u_{t_{k-1}}^\theta) \tag{53}$$

Nous remarquons en particulier que le choix de modéliser u comme un processus markovien dépendant du prix spot et de la volatilité est justifié en pratique. En effet, dans un marché idéalisé,

c'est-à-dire sans coûts de transaction, sans contraintes de liquidité ni autres imperfections, il est possible de répliquer parfaitement le payoff d'une option européenne en utilisant uniquement le prix spot et un swap de variance sur la volatilité. Pour illustrer cette idée, considérons un actif $(S_t)_t$ à valeurs dans $\mathbb{R}$ suivant un modèle de Heston avec une volatilité stochastique $(V_t)_t$, et $g(S_T)$ un payoff européen de maturité T. Notons $(\tilde{S}_t)_t$ le prix du swap de variance sur la volatilité V, et $L(t, v)$ la fonction définie dans (51). En exploitant la propriété de Markov du couple (S, V) et en valorisant sous la probabilité risque-neutre, la relation (54) est vérifiée. De plus, en appliquant la formule d'Itô, on obtient (55). Cette démarche permet de construire une stratégie de réplication dite delta-vega hedgée, dans laquelle les contrôles optimaux correspondent exactement aux sensibilités delta et vega, comme exprimé dans (56). Cette observation suggère que, dans un marché imparfait, la stratégie optimale u doit nécessairement dépendre à la fois du prix spot et de la volatilité, ce qui justifie l'utilisation d'un modèle riche de type Heston, capable de capturer les dynamiques conjointes de ces deux processus.

$$\mathbb{E}_{\mathbb{Q}}[g(S_T)|\mathcal{F}_t] = f(t, S_t, V_t) \tag{54}$$

$$g(S_T) = E_{\mathbb{Q}}[g(S_T)] + \int_0^T \frac{\partial f}{\partial S}(t, S_t, V_t) dS_t + \int_0^T \frac{\frac{\partial f}{\partial v}(t, S_t, V_t)}{\frac{\partial L}{\partial v}(t, V_t)} d\tilde{S}_t \tag{55}$$

$$u_t = \frac{\partial f}{\partial S}(t, S_t, V_t) \quad \& \quad \tilde{u}_t = \frac{\frac{\partial f}{\partial v}(t, S_t, V_t)}{\frac{\partial L}{\partial v}(t, V_t)} \tag{56}$$

5.2 Résultats et discussion

La figure 9 compare les performances des algorithmes Adam et L-Adam pour le problème de deep hedging dans le cas d'un unique réseau de contrôle, pour deux niveaux de discrétisation ($N = 30$ et $N = 50$). La figure 10 étend cette comparaison au cas où un réseau de neurones distinct est utilisé à chaque instant de contrôle.

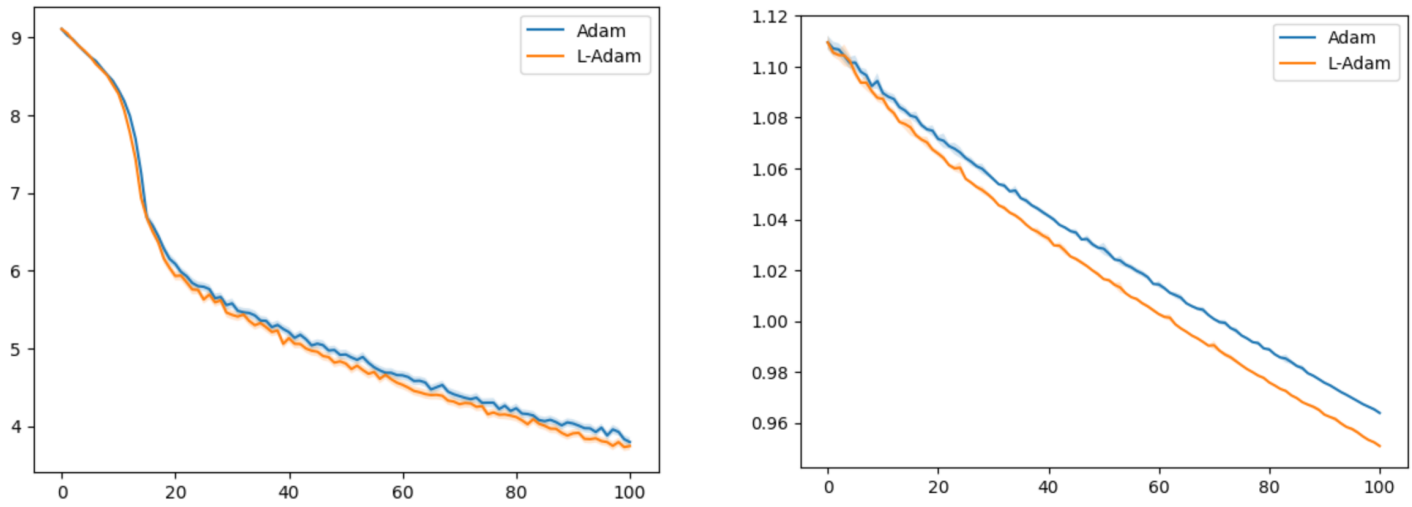

FIGURE 9 – Comparaison des algorithmes Adam et L-Adam lors de l'entraînement pour le problème de deep hedging pour un nombre de pas $N = 30$ et $N = 50$ respectivement avec un unique réseau de contrôle.

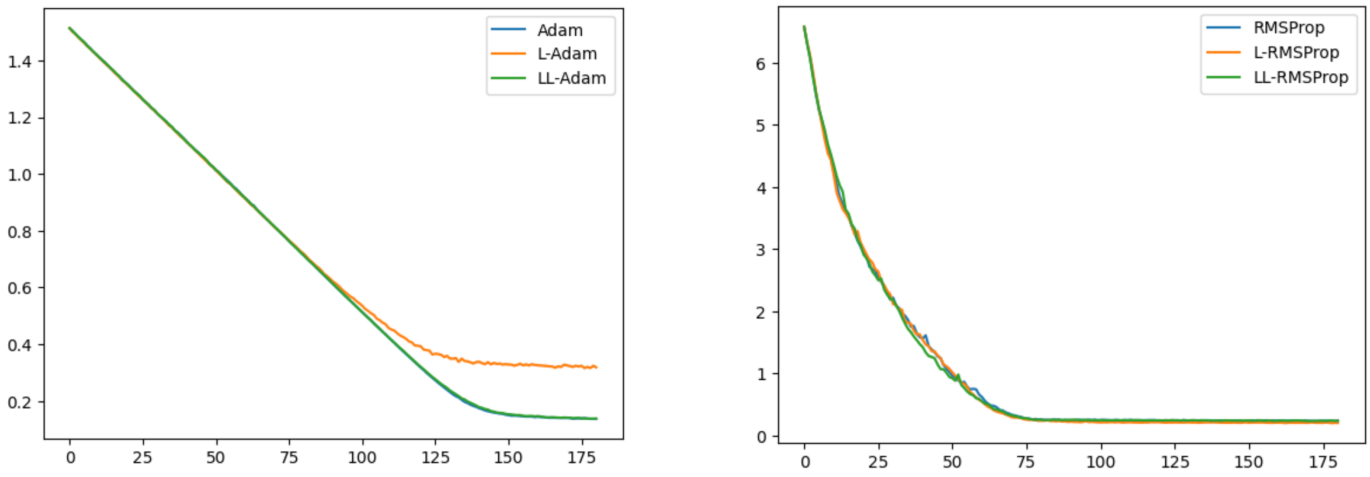

FIGURE 10 – Comparaison des algorithmes Adam, L-Adam et LL-Adam puis RMSProp, L-RMSProp et LL-RMSProp lors de l'entraînement pour le problème de deep heging pour un nombre de pas $N = 10$ et plusieurs réseaux de contrôle.

Comparé au problème précédent, le gain apporté par la méthode de Langevin est ici nettement moins visible. Dans de nombreuses simulations, les versions Langevin n'apportent pas d'amélioration significative par rapport aux méthodes classiques, et peuvent même parfois s'avérer légèrement moins performantes. Dans le cas avec plusieurs contrôles, les différents algorithmes testés produisent également des résultats très proches, avec une forte variabilité d'une exécution à l'autre, reflétant une grande sensibilité aux conditions initiales.

6 Conclusion

Pour conclure, nous avons présenté la méthode de Langevin en exposant l'intérêt théorique qu'il y a derrière, notamment à travers son lien avec les équations différentielles stochastiques et la mesure ergodique associée. Nous avons ensuite évalué empiriquement ses performances sur deux problèmes de contrôle optimal : la régulation des quotas de pêche et le choix des stratégies de couverture en finance. L'accent a été mis tout au long du rapport sur le volet mathématiques, tant dans l'analyse de la méthode de Langevin que dans la modélisation rigoureuse des problèmes de contrôle optimal étudiés.

Nos résultats empiriques ne permettent cependant pas de conclure de façon tranchée à la supériorité systématique de la méthode de Langevin. Si celle-ci peut apporter des gains notables dans certains contextes, ses performances varient fortement selon le problème étudié, le choix du préconditionneur, ainsi que les conditions initiales. Ce constat nous amène à souligner que l'utilisation de la méthode de Langevin doit être envisagée au cas par cas.

Bibliographie

[1] Odile Brandière and Marie Duflo. Les algorithmes stochastiques contournent-ils les pieges ? In *Annales de l'IHP Probabilités et statistiques*, volume 32, pages 395–427, 1996.

[2] Pierre Bras and Gilles Pagès. Convergence of langevin-simulated annealing algorithms with multiplicative noise ii : Total variation. *Monte Carlo Methods and Applications*, 29(3) :203–219, 2023.

[3] Pierre Bras and Gilles Pagès. Langevin algorithms for markovian neural networks and deep stochastic control. In *2023 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2023.

[4] Hans Buehler, Lukas Gonon, Josef Teichmann, and Ben Wood. Deep hedging. *Quantitative Finance*, 19(8) :1271–1291, 2019.

[5] John Duchi, Elad Hazan, and Yoram Singer. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12 :2121–2159, 2011.

[6] Mike Giles and Paul Glasserman. Smoking adjoints : Fast monte carlo greeks. *Risk*, 19(1) :88–92, 2006.

[7] Diederik P Kingma and Jimmy Ba. Adam : A method for stochastic optimization. *arXiv preprint arXiv :1412.6980*, 2014.

[8] Mathieu Laurière, Olivier Pironneau, et al. Performance of a markovian neural network versus dynamic programming on a fishing control problem. *arXiv preprint arXiv :2109.06856*, 2021.

[9] Boris T Polyak and Anatoli B Juditsky. Acceleration of stochastic approximation by averaging. *SIAM journal on control and optimization*, 30(4) :838–855, 1992.

[10] Herbert Robbins and David Siegmund. A convergence theorem for non negative almost supermartingales and some applications. In *Optimizing methods in statistics*, pages 233–257. Elsevier, 1971.

[11] R Tyrrell Rockafellar, Stanislav Uryasev, et al. Optimization of conditional value-at-risk. *Journal of risk*, 2 :21–42, 2000.

[12] Tijmen Tieleman. Lecture 6.5-rmsprop : Divide the gradient by a running average of its recent magnitude. *COURSERA : Neural networks for machine learning*, 4(2) :26, 2012.

[13] Matthew D Zeiler. Adadelta : an adaptive learning rate method. *arXiv preprint arXiv :1212.5701*, 2012.