UE STAGE DE RECHERCHE ACADEMIQUE

- RAPPORT DE STAGE -

Sujet : Méthodes d'apprentissage profond pour des problèmes de contrôle optimal avec un terme de switching

CELIK Baran

ESPCI PARIS | PSL

LPSM LABORATOIRE DE PROBABILITÉS STATISTIQUE & MODÉLISATION

Référent académique : A. ALLAUZEN
Tuteur : M. TALBI @LPSM

2024/2025

Table des matières

1 Introduction

Les problèmes de *contrôle optimal stochastique avec switching* modélisent des situations dans lesquelles un agent rationnel peut modifier, à des instants choisis, le régime de fonctionnement d'un système soumis à de l'aléa. Chaque changement de régime entraîne un coût, et la stratégie optimale doit équilibrer les gains liés à l'adaptation du système avec les pénalités de commutation. Ces modèles apparaissent dans de nombreux contextes appliqués, tels que la gestion de production, la valorisation d'options réelles, ou le pilotage de réseaux énergétiques sous incertitude.

Sur le plan mathématique, ces problèmes s'expriment à l'aide d'équations différentielles stochastiques (EDS) contrôlées, dont les coefficients dépendent du régime actif. Le contrôle prend la forme d'un processus à valeurs dans un ensemble fini de régimes, constant par morceaux, qui permet à l'agent de modifier le mode de fonctionnement du système au cours du temps. La fonctionnelle de coût, à maximiser, combine un gain courant, un gain terminal, et une pénalité liée au nombre de changements de régime.

La résolution numérique de ces problèmes passe classiquement par une discrétisation temporelle du système, notamment via le schéma d'Euler–Maruyama. Le problème est alors ramené à un processus de décision markovien en temps discret, dont la stratégie optimale peut être caractérisée par une équation de Bellman. Sous des hypothèses usuelles de régularité, il est possible de démontrer que la valeur optimale discrétisée converge vers la valeur continue, avec un taux d'erreur en $\mathcal{O}(\sqrt{h})$ (voir par exemple [1]).

Cependant, ces méthodes traditionnelles deviennent inopérantes en grande dimension, en raison de la *malédiction de la dimension* : le nombre d'états à considérer croît exponentiellement avec la dimension de l'espace d'état, rendant les méthodes par maillage inapplicables au-delà de quelques dimensions. Pour surmonter cette difficulté, des approches récentes fondées sur l'*apprentissage profond* utilisent des réseaux de neurones pour approximer les fonctions valeur et les politiques optimales à partir de trajectoires simulées.

La problématique étudiée dans ce travail est la suivante : *comment concevoir et implémenter une méthode d'apprentissage automatique, fondée sur des réseaux de neurones, permettant d'approximer efficacement la solution d'un problème de contrôle optimal avec switching à horizon fini ?*

Ce rapport propose une étude du problème de switching optimal à horizon fini, à la fois sur le plan théorique et numérique. Il s'articule en deux parties :

— une **partie théorique**, où le problème est modélisé rigoureusement, puis discrétisé à l'aide du schéma d'Euler–Maruyama, et pour laquelle une borne d'erreur de convergence est établie entre la valeur discrète et la valeur continue ;

— une **partie numérique**, consacrée à l'implémentation d'un algorithme d'apprentissage profond exploitant deux réseaux de neurones : l'un pour estimer les espérances conditionnelles intervenant dans l'équation de Bellman, l'autre pour approximer la politique optimale.

L'efficacité de la méthode proposée est évaluée sur deux exemples en dimension un : un cas simple à solution explicite, permettant de valider les prédictions de l'algorithme, et un exemple plus complexe, sans solution analytique, comparé à une méthode par différences finies. Une extension en dimension deux est également étudiée, afin d'évaluer la robustesse de l'algorithme dans un espace d'état plus complexe. Cette étude met en évidence les avantages et limites de l'approche par apprentissage profond dans la résolution de problèmes de contrôle stochastique à switching, notamment en présence de bruit Monte Carlo et de forte dimension.

2 Cadre théorique du switching optimal

Les problèmes de switching optimal consistent à piloter un système dynamique évoluant dans le temps, en changeant à certains instants le *régime* ou *mode* de fonctionnement, de façon à maximiser une fonction de récompense. Chaque changement de régime entraîne un coût, ce qui oblige le contrôleur à arbitrer entre rester dans le régime actuel ou passer à un autre plus favorable. Ces problèmes apparaissent notamment dans les modèles de production industrielle, les options réelles en finance, ou encore les systèmes énergétiques sous incertitude.

2.1 Cadre probabiliste

On se place sur un espace probabilisé complet $(\Omega, \mathscr{F}, \mathbb{P})$ muni d'une filtration $(\mathscr{F}_t)_{t\geq 0}$ satisfaisant les conditions usuelles. On considère un mouvement brownien $(W_t)_{t\geq 0}$ de dimension d, et on fixe un horizon temporel $T > 0$. L'état du système est modélisé par un processus $(X_t)_{t\in[0,T]}$ à valeurs dans $\mathbb{R}^d$.
Le contrôle s'exerce par choix d'un processus à valeurs dans un ensemble compact $A \subset \mathbb{R}^m$, représentant les différents régimes de fonctionnement disponibles.

2.2 Contrôles admissibles

Définition 2.1. Un *contrôle admissible* est un processus càdlàg $\alpha : [0, T] \to A$, constant par morceaux, et ayant un nombre fini de sauts :

$$N(\alpha) = \sum_{0<t\leq T} \mathbf{1}_{\{\alpha_{t^-} \neq \alpha_t\}} < \infty.$$

On note $\mathcal{A}$ l'ensemble de ces contrôles.

Les instants de commutation sont interprétés comme des temps d'arrêt, et les valeurs prises par α entre deux sauts correspondent au régime actif.

2.3 Équation différentielle stochastique contrôlée

Pour un contrôle admissible $\alpha \in \mathcal{A}$, on considère l'équation différentielle stochastique (EDS) suivante :

$$dX_t^\alpha = b(X_t^\alpha, \alpha_t)\, dt + \sigma(X_t^\alpha, \alpha_t)\, dW_t, \qquad X_0 = x_0 \in \mathbb{R}^d. \tag{1}$$

On fait les hypothèses suivantes sur les coefficients :

$\mathbf{H_{Lip}}$ (Lipschitz + croissance linéaire) : il existe $L, K > 0$ tels que

$$\begin{aligned} &\|b(x,a) - b(y,a)\| + \|\sigma(x,a) - \sigma(y,a)\| \leq L\|x - y\|, \\ &\|b(x,a)\| + \|\sigma(x,a)\| \leq K(1 + \|x\|), \qquad \forall x, y \in \mathbb{R}^d,\ a \in A. \end{aligned}$$

Sous cette hypothèse standard, l'EDS (1) admet une solution forte unique, au sens d'Itô.

2.4 Fonctionnelle de coût

On suppose donnés une fonction de coût courant $f : \mathbb{R}^d \times A \to \mathbb{R}$, une fonction de coût terminal $g : \mathbb{R}^d \to \mathbb{R}$, et un coût de switching constant $c > 0$.

$\mathbf{H_L}$ (Lipschitz des coûts) : f et g sont Lipschitz de constante $L > 0$.

Pour un contrôle $\alpha \in \mathcal{A}$, on définit la fonctionnelle de coût associée :

$$J(\alpha) = \mathbb{E}\left[\int_0^T f(X_t^\alpha, \alpha_t)\, dt + g(X_T^\alpha) - c\, N(\alpha)\right],$$

et on définit la *valeur optimale* comme :

$$V_0 := \sup_{\alpha \in \mathcal{A}} J(\alpha).$$

Ce problème est un exemple de *problème de contrôle impulsionnel*, avec un coût discret lié aux changements de stratégie.

Dans la suite, nous étudierons la discrétisation de ce problème par un schéma d'Euler–Maruyama, et nous établirons la convergence de la valeur approchée vers V_0 lorsque le pas de discrétisation tend vers zéro.

3 Discrétisation par le schéma d'Euler–Maruyama

Dans cette section, nous introduisons une discrétisation temporelle du système stochastique (1) à l'aide du schéma d'Euler–Maruyama, puis nous définissons une version discrète du problème de contrôle. Cela permettra, dans la section suivante, d'analyser rigoureusement l'erreur induite par cette approximation.

3.1 Grille temporelle

On considère une grille uniforme $\pi_h = \{t_k = kh \mid k = 0, \dots, N\}$ de pas $h > 0$, avec $T = Nh$. Les contrôles admissibles dans le cadre discret sont définis comme étant *constants par morceaux* sur chaque intervalle de la grille.

Définition 3.1. On note $\mathcal{A}_h$ l'ensemble des contrôles admissibles discrets, définis par :

$$\alpha_t = \alpha_{t_k}, \quad \text{pour } t \in [t_k, t_{k+1}).$$

Ces contrôles possèdent un nombre fini de sauts, et prennent des valeurs constantes sur les mailles de la grille.

3.2 Schéma d'Euler–Maruyama

Pour un contrôle $\alpha \in \mathcal{A}_h$, on approxime la trajectoire de l'état $(X_t^\alpha)_{t\in[0,T]}$ par une suite $(X_{t_k}^{\alpha,h})_{k=0,\dots,N}$ définie par récurrence :

$$X_{t_{k+1}}^{\alpha,h} = X_{t_k}^{\alpha,h} + b(X_{t_k}^{\alpha,h}, \alpha_{t_k})h + \sigma(X_{t_k}^{\alpha,h}, \alpha_{t_k})\sqrt{h}\, \xi_{t_k}, \tag{2}$$

où $(\xi_{t_k})_{k=0,\dots,N-1}$ est une suite de variables aléatoires indépendantes, de loi $\mathcal{N}(0, I_d)$, définies sur un espace probabilisé suffisamment riche. On suppose que, pour chaque k, la variable ξ_{t_k} est indépendante du passé jusqu'au temps t_k, conditionnellement à l'état $X_{t_k}^{\alpha,h}$.
Ce schéma d'Euler–Maruyama est une méthode classique d'approximation des EDS en temps discret, préservant les propriétés statistiques des trajectoires simulées.

3.3 Valeur approchée et coût discretisé

On définit la version discrète de la fonctionnelle de coût :

$$J^h(\alpha) = \mathbb{E}\left[h\sum_{k=0}^{N-1} f(X_{t_k}^{\alpha,h}, \alpha_{t_k}) + g(X_{t_N}^{\alpha,h}) - c\sum_{k=0}^{N-1} \mathbf{1}_{\{\alpha_{t_k} \neq \alpha_{t_{k+1}}\}}\right],$$

et la valeur optimale discrète associée :

$$V_0^h := \sup_{\alpha \in \mathcal{A}_h} J^h(\alpha).$$

Cette valeur représente le gain maximal espéré pour une stratégie discrète, lorsque la dynamique de l'état est approximée par le schéma (2). Dans la section suivante, nous montrerons que cette valeur discrète converge vers la valeur continue V_0 et déterminerons la vitesse de convergence.

4 Analyse de l'erreur de discrétisation

Nous étudions ici la précision de l'approximation numérique induite par le schéma d'Euler–Maruyama. En particulier, nous établissons une borne sur l'erreur entre la valeur optimale du problème continu V_0 et celle du problème discret V_0^h.

4.1 Erreur forte sur les trajectoires

La convergence du schéma d'Euler–Maruyama est classique pour les EDS à coefficients Lipschitziens. Le résultat suivant garantit une erreur forte d'ordre 1/2 pour l'approximation des trajectoires, uniformément sur les contrôles admissibles.

Lemme 4.1 (Erreur forte d'ordre **1/2**)**.** *Sous l'hypothèse* $\mathbf{H_{Lip}}$*, il existe une constante* $C > 0$*, indépendante de* h *et du contrôle* α*, telle que :*

$$\mathbb{E}\left[\sup_{0\leq t\leq T} \left\| X_t^\alpha - X_{\lfloor t/h \rfloor}^{\alpha,h} \right\|\right] \leq C\sqrt{h}.$$

4.2 Comparaison des valeurs optimales

Nous comparons à présent la valeur optimale du problème continu, V_0, avec celle du problème discret, V_0^h. L'idée est de transporter des stratégies presque optimales d'un problème à l'autre, en contrôlant la perte sur chaque terme de la fonctionnelle.

4.2.1 Borne supérieure : $V_0 \leq V_0^h + C\sqrt{h}$

Soit $\hat{\alpha} \in \mathcal{A}$ ε-optimal pour le problème continu, et soit α^h son image sur la grille :

$$\alpha_t^h := \hat{\alpha}_{t_k}, \quad \text{pour } t \in [t_k, t_{k+1}).$$

Alors $\alpha^h \in \mathcal{A}_h$, et $N(\alpha^h) \leq N(\hat{\alpha})$.
On compare les coûts dans $J(\hat{\alpha})$ et $J^h(\alpha^h)$:

— **Terme terminal** : Lipschitzité de g et Lemme 4.1 $\Rightarrow$

$$|\mathbb{E}[g(X_T^{\hat{\alpha}})] - \mathbb{E}[g(X_{t_N}^{\alpha^h,h})]| \leq LC\sqrt{h}.$$

— **Terme courant** : approximation de l'intégrale par une somme de Riemann, Lipschitzité de f et régularité de $X_t^{\hat{\alpha}} \Rightarrow$

$$\left| \mathbb{E}\left[\int_0^T f(X_t^{\hat{\alpha}}, \hat{\alpha}_t) dt \right] - h \sum_k \mathbb{E}[f(X_{t_k}^{\alpha^h,h}, \alpha_{t_k})] \right| \leq LCT\sqrt{h}.$$

— **Coût de switching** : pas de nouveau saut introduit $\Rightarrow N(\alpha^h) \leq N(\hat{\alpha})$.

On obtient ainsi :

$$J^h(\alpha^h) \geq J(\hat{\alpha}) - C_1\sqrt{h},$$
$$J^h(\alpha^h) \geq V_0 - C_1\sqrt{h} - \varepsilon,$$

puis en laissant $\varepsilon \to 0$:

$$V_0 \leq V_0^h + C_1\sqrt{h}.$$

4.2.2 Borne inférieure : $V_0^h \leq V_0 + C\sqrt{h}$

L'existence d'un contrôle optimal pour le problème discret s'obtient par compacité. En effet, l'ensemble $\mathcal{A}_h$ des contrôles admissibles discrets, à valeurs dans un ensemble compact A, et constant par morceaux sur une grille finie, est lui-même fini ou compact selon la topologie de la convergence simple. Par continuité de la fonctionnelle J^h par rapport aux trajectoires (induite par la continuité de f, g, et la stabilité du schéma d'Euler–Maruyama sous les hypothèses de Lipschitz), le supremum dans la définition de V_0^h est atteint.

Il existe donc $\hat{\alpha}^h \in \mathcal{A}_h$ tel que $V_0^h = J^h(\hat{\alpha}^h)$. Le contrôle $\hat{\alpha}^h$ peut être vu comme un processus constant par morceaux sur la grille temporelle, et il est alors identifié à un contrôle admissible en temps continu noté $\tilde{\alpha}$, défini par :

$$\tilde{\alpha}_t := \hat{\alpha}_{t_k}^h, \qquad \text{pour } t \in [t_k, t_{k+1}).$$

Il en résulte que $N(\tilde{\alpha}) = N(\hat{\alpha}^h)$.

L'argument utilisé pour comparer les fonctionnelles de coût J^h et J est symétrique à celui présenté dans la borne supérieure, en inversant les rôles des problèmes discret et continu :

— **Terme terminal** : la continuité de g et le Lemme 4.1 impliquent que l'erreur sur le terme final est de l'ordre $\sqrt{h}$;

— **Terme courant** : l'approximation de l'intégrale par une somme de Riemann, combinée à la régularité de $X_t^{\tilde{\alpha}}$ et de f, induit une erreur également contrôlée en $\sqrt{h}$;

— **Coût de switching** : la contribution est identique dans les deux modèles, en raison de l'égalité $N(\tilde{\alpha}) = N(\hat{\alpha}^h)$.

Il est ainsi établi que :

$$J(\tilde{\alpha}) \geq J^h(\hat{\alpha}^h) - C_2\sqrt{h} = V_0^h - C_2\sqrt{h},$$

ce qui implique, après réarrangement :

$$V_0^h \leq V_0 + C_2\sqrt{h}.$$

4.3 Résultat de convergence

En combinant les deux inégalités précédentes, on obtient :

$$\boxed{|V_0^h - V_0| \leq C\sqrt{h}} \tag{3}$$

où $C = \max(C_1, C_2)$ est indépendant de h. Le schéma d'Euler–Maruyama permet donc une approximation de la valeur optimale avec un ordre de convergence fort $1/2$.

5 Équation de Bellman discrète et principe de programmation dynamique

Le schéma de discrétisation introduit dans les sections précédentes permet de ramener le problème de switching optimal à un processus de décision markovien en temps discret. On peut alors caractériser la valeur optimale via une équation récursive, appelée *équation de Bellman*, qui repose sur le principe de la programmation dynamique.

5.1 Principe de programmation dynamique

Fixons une grille $\pi_h = \{t_k = kh\}_{k=0}^N$ et un état $x_k \in \mathbb{R}^d$ au temps t_k. À chaque instant t_k, l'agent doit choisir une action (régime) $a_k \in A$. Le système évolue selon la dynamique stochastique donnée par le schéma d'Euler :

$$X_{t_{k+1}} = x_{k+1} = x_k + b(x_k, a_k)h + \sigma(x_k, a_k)\sqrt{h}\,\xi_k,$$

où $\xi_k \sim \mathcal{N}(0, I_d)$, indépendants.

On note $v_k(x, i)$ la valeur optimale restante au temps t_k lorsque le système est en position x et dans le régime i. Le principe de Bellman affirme que :

$$v_k(x,i) = \max_{j \in A} \mathbb{E}\left[f(x,j)h - c\,\mathbf{1}_{j\neq i} + v_{k+1}(X_{k+1}, j) \,\middle|\, X_k = x\right], \tag{4}$$

avec condition terminale :

$$v_N(x,i) = g(x).$$

Cette équation exprime le fait que, au temps t_k, l'agent peut :

— rester dans le même régime $j = i$ sans coût ;

— ou passer à un autre régime $j \neq i$, ce qui induit un coût c.

Il s'agit donc d'un problème de *contrôle par switching en temps discret*, dans lequel la fonction valeur v_k est obtenue par backward induction sur la grille temporelle.

5.2 Lien avec les équations variationnelles

Lorsque $h \to 0$, cette équation de Bellman discrète converge formellement vers un système d'EDP variationnelles de la forme :

$$\min\left\{\beta v_i(x) - \mathcal{L}_i v_i(x) - f_i(x), v_i(x) - \max_{j\neq i}\left(v_j(x) - g_{ij}\right)\right\} = 0,$$

où $\mathcal{L}_i$ est le générateur infinitésimal associé au régime i :

$$\mathcal{L}_i\varphi(x) = b(x,i)\cdot\nabla\varphi(x) + \frac{1}{2}\mathrm{Tr}\left(\sigma(x,i)\sigma(x,i)^\top\nabla^2\varphi(x)\right).$$

Cette équation admet une formulation en solutions de viscosité, comme cela est établi au chapitre 5.3 de [2].

6 Approximation de la solution par apprentissage profond : méthode à deux réseaux

Nous décrivons ici une méthode d'approximation numérique de la solution d'un problème de switching optimal, fondée sur des techniques d'apprentissage automatique. Cette approche repose sur une discrétisation du temps et une résolution approximative de l'équation de Bellman par backward induction. L'originalité de la méthode réside dans l'utilisation de deux réseaux de neurones distincts, chargés d'estimer respectivement une espérance conditionnelle et la politique de commutation optimale. Ce type d'algorithme, inspiré des méthodes hybrides proposées dans [3], permet de traiter efficacement des problèmes en dimension élevée, où les méthodes classiques deviennent inopérantes.

6.1 Principe général

Après discrétisation temporelle, la fonction valeur $v_k(x, i)$ satisfait une équation de Bellman du type :

$$v_k(x,i) = \max_{j \in A} \left\{ f(x,j)\, h - c \cdot \mathbf{1}_{j \neq i} + \mathbb{E}\left[v_{k+1}(X_{k+1}, j) \,\middle|\, X_k = x \right] \right\},$$

où X_{k+1} suit la dynamique induite par le schéma d'Euler–Maruyama :

$$X_{k+1} = x + b(x,j)\, h + \sigma(x,j)\, \sqrt{h}\, \xi.$$

Deux difficultés majeures apparaissent dans cette expression :

— l'évaluation de l'espérance conditionnelle en grande dimension ;

— la maximisation sur l'ensemble des actions possibles A.

Pour surmonter ces obstacles, nous introduisons deux réseaux de neurones :

1. un réseau ϕ_θ chargé d'approximer l'espérance conditionnelle ;
2. un réseau p_ψ chargé d'estimer la politique optimale sous forme d'une distribution de probabilité sur les régimes disponibles.

6.2 Estimation de l'espérance conditionnelle

Le premier réseau $\phi_\theta : \mathbb{R}^d \times A \to \mathbb{R}$ vise à approximer la quantité :

$$\phi_\theta(x,j) \approx \mathbb{E}\left[v_{k+1}(X_{k+1}, j) \mid X_k = x \right],$$

qui correspond à la valeur future attendue en choisissant le régime j au temps t_k, à partir de l'état x.
Cette estimation repose sur une interprétation variationnelle de l'espérance conditionnelle comme projection orthogonale dans L^2 :

$$\mathbb{E}\left[v_{k+1}(X_{k+1}, j) \mid X_k = x \right] = \arg \min_{\phi \in L^2(\sigma(X_k))} \mathbb{E}\left[\left(v_{k+1}(X_{k+1}, j) - \phi(X_k) \right)^2 \right].$$

Dans la pratique, cette projection est approximée en restreignant l'espace des fonctions candidates à une famille paramétrée par un réseau de neurones, et en minimisant l'erreur quadratique moyenne :

$$\min_{\theta \in \mathbb{R}^p} \mathbb{E}\left[\left(v_{k+1}(X_{k+1}, j) - \phi_\theta(x,j) \right)^2 \right],$$

à partir de données simulées (x, X_{k+1}) générées selon la dynamique contrôlée.

6.3 Approximation de la politique optimale

Dans le cadre discret considéré, l'ensemble des régimes A est fini. L'approximation de la politique optimale revient donc à estimer une fonction à valeurs discrètes, ce qui complique l'utilisation des méthodes de descente de gradient classiques, fondées sur la différentiabilité de la fonction de perte. Pour remédier à cette difficulté, une approche inspirée de l'algorithme ClassifPI introduit dans [3] est employée, reposant sur une représentation probabiliste des politiques.

L'idée consiste à associer à chaque état $x \in \mathbb{R}^d$ une distribution de probabilité sur A, via un réseau de neurones $p_\psi : \mathbb{R}^d \to \Delta^{|A|-1}$, où $\Delta^{|A|-1}$ désigne le simplexe des probabilités sur A. Pour tout $x \in \mathbb{R}^d$, le vecteur $p_\psi(x) = (p_j(x))_{j \in A}$ définit une politique stochastique, c'est-à-dire :

$$\mathbb{P}[\alpha_k = j \mid X_k = x] = p_j(x).$$

Cette relaxation continue permet d'évaluer une version différentiable de la fonction valeur, sous forme d'une espérance pondérée par la politique stochastique :

$$v_k(x, i) \approx \sum_{j \in A} p_j(x) \left(f(x, j)\, h - c\, \mathbf{1}_{j \neq i} + \phi_\theta(x, j) \right),$$

où ϕ_θ désigne le réseau de neurones approximant l'espérance conditionnelle $\mathbb{E}[v_{k+1}(X_{k+1}, j) \mid X_k = x]$.

Les paramètres ψ du réseau p_ψ sont optimisés en minimisant une fonction de perte différentiable :

$$\mathcal{L}(\psi) = -\mathbb{E}_{x \sim \mu_k} \left[\sum_{j \in A} p_j(x) \left(f(x, j)\, h - c\, \mathbf{1}_{j \neq i} + \phi_\theta(x, j) \right) \right],$$

où μ_k désigne une mesure de probabilité sur les états au temps t_k (par exemple une mesure empirique obtenue par simulation).

La contrainte de positivité et de sommation à un des coefficients $p_j(x)$ est assurée par l'utilisation d'une couche `softmax` en sortie du réseau p_ψ. Une fois ce dernier entraîné, une politique déterministe peut être obtenue par sélection du maximum de probabilité :

$$\pi(x) := \arg\max_{j \in A} p_j(x).$$

7 Implémentation numérique de l'algorithme

Cette section présente l'implémentation complète de la méthode d'approximation par apprentissage profond décrite précédemment. Le code est structuré de manière modulaire et disponible dans un notebook Jupyter associé, permettant de reproduire toutes les expériences numériques, visualiser les courbes de pertes et tester différentes configurations de paramètres.

7.1 Structure du code

L'implémentation repose sur une organisation modulaire des différentes composantes de l'algorithme. Les principaux blocs fonctionnels sont :

— la classe `FiniteDifferenceSolver` pour la résolution de l'équation de Bellman par différences finies ;

— les classes `ExpectationNetwork`, `PolicyNetwork` et `PolicyNetwork2` représentant les trois architectures de réseaux de neurones utilisées ;

— des fonctions de génération de données simulées selon la dynamique contrôlée ;

— des fonctions d'apprentissage pour chaque type de réseau ;

— une fonction maître `solve_neural_networks()` orchestrant l'induction backward sur l'horizon temporel.

7.2 Génération des données

À chaque pas de temps k, un ensemble de données est généré de manière simulée, en construisant un dataset de taille M. Pour chaque état $x \in \mathbb{R}^d$ échantillonné uniformément dans un intervalle pertinent (centré autour de l'origine), et pour chaque action $a \in A$, on estime la récompense cumulée attendue en simulant L trajectoires indépendantes de bruit brownien. Cette procédure permet d'approximer par une moyenne de Monte Carlo la valeur espérée de la continuation sous l'action a.
Le processus est implémenté dans la fonction `generate_training_data`, qui renvoie un ensemble de triplets (x, a, y) où y est une approximation empirique de la cible :

$$y(x,a) \approx f(x,a)\,h + \frac{1}{L}\sum_{\ell=1}^{L} v_{k+1}(X_{k+1}^{(\ell)}, a).$$

Le paramètre M contrôle le nombre d'états x simulés, c'est-à-dire la taille du dataset à chaque pas k. Tandis que L gouverne la précision de l'estimation de l'espérance conditionnelle. Un compromis doit être trouvé entre variance et coût de calcul : augmenter L réduit le bruit sur les cibles y, tandis qu'augmenter M enrichit la diversité des états simulés.

7.3 Architecture et apprentissage des réseaux de neurones

Les deux réseaux de neurones utilisés dans l'algorithme jouent des rôles distincts mais complémentaires : ϕ_θ approxime l'espérance conditionnelle dans l'équation de Bellman, tandis que p_ψ approxime la politique de commutation optimale. Tous deux sont implémentés à l'aide de réseaux feedforward profonds, entraînés par descente de gradient à l'aide de l'optimiseur `Adam`, avec régularisation L^2 (poids de $5 \cdot 10^{-5}$) et scheduler de type `CosineAnnealingLR`.

Réseau ϕ_θ : approximation d'espérance. Le réseau $\phi_\theta : \mathbb{R}^d \times A \to \mathbb{R}$ reçoit en entrée l'état x et l'action a, concaténés dans un vecteur de dimension $d+1$. L'architecture utilisée comprend trois couches cachées de taille 128, avec activations `ReLU` et normalisation `LayerNorm` sur les deux premières couches. L'entraînement s'effectue sur $E_\phi = 300$ époques, avec un taux d'apprentissage initial fixé à $\alpha_\phi = 10^{-3}$. L'objectif est de minimiser l'erreur quadratique moyenne entre la sortie du réseau et une estimation Monte Carlo de l'espérance conditionnelle :

$$\min_\theta \ \mathbb{E}\left[(y - \phi_\theta(x,a))^2\right],$$

où y est obtenu en simulant L trajectoires browniennes.

Réseaux p_ψ : approximation de politique. Deux variantes de politique sont explorées :

— **Politique déterministe** (`PolicyNetwork`) : le réseau retourne des logits pour chaque action possible, à partir de l'état $x \in \mathbb{R}^d$. Il est composé de deux couches cachées de taille 128, avec activations `SiLU` et normalisation `LayerNorm`. L'apprentissage repose sur une perte de classification croisée, en prenant comme cible l'action optimale $j^*(x)$ qui maximise le gain estimé par ϕ_θ :

$$j^*(x) := \arg\max_{j \in A} \{f(x,j)\,h - c\,\mathbf{1}_{j \neq i} + \phi_\theta(x,j)\} \text{ avec } i \text{ le régime courant},$$

$$\min_{\psi} \ \mathbb{E}_{x\sim\mu_k}\left[\texttt{CrossEntropy}\left(p_\psi(x), j^*(x)\right)\right].$$

— **Politique stochastique** (`PolicyNetwork2`) : le réseau retourne directement une distribution de probabilité $(p_j(x))_{j\in A}$ via une couche `softmax`. L'objectif est de maximiser la valeur espérée du gain pondéré par cette politique :

$$\mathcal{L}(\psi) = -\mathbb{E}_x\left[\sum_{j\in A} p_j(x)\ \left(f(x,j)\,h - c\,\mathbf{1}_{j\neq i} + \phi_\theta(x,j)\right)\right].$$

L'architecture du réseau est analogue à celle de `PolicyNetwork`, avec des activations `ReLU`.

L'ensemble des réseaux est entraîné sur un échantillon d'états x tirés uniformément dans un intervalle centré autour de 0, avec taille de batch 64. Les courbes de perte associées à chaque réseau sont tracées pour chaque pas de temps k, et permettent d'évaluer la stabilité de l'apprentissage et l'impact du bruit de Monte Carlo.

7.4 Fonction `solve_neural_networks()`

La fonction `solve_neural_networks()` constitue le cœur de la résolution numérique par apprentissage profond. Elle implémente une procédure d'induction arrière (backward induction) sur les temps discrets $k = N-1, \ldots, 0$, en construisant les approximations successives de la fonction valeur par les deux réseaux de neurones.

À chaque pas de temps k, les étapes suivantes sont exécutées :

1. Construction du réseau ϕ_k et des deux réseaux de politique π_k^{classif} et π_k^{stoch}.
2. Génération d'un dataset (x, a, y) via la fonction `generate_training_data`, en utilisant la fonction valeur estimée v_{k+1} (si $k < N-1$) pour construire les cibles y.
3. Entraînement du réseau ϕ_k sur les données générées, par régression sur la cible y.
4. Entraînement de π_k^{classif} par classification supervisée (choix optimal d'action), et de π_k^{stoch} par maximisation de l'espérance du gain futur sous politique stochastique.
5. Stockage des réseaux entraînés dans des listes pour les réutiliser à l'étape précédente $(k-1)$.

Cette approche repose sur une propagation arrière de l'information à partir du coût terminal, en mettant à jour itérativement les approximations des fonctions valeur et politique.

8 Application numérique

Cette section présente des expériences numériques illustrant l'efficacité de l'algorithme proposé. L'objectif est double :

— dans un premier temps, tester l'algorithme sur un exemple simple en dimension un, dont la solution optimale peut être déterminée explicitement ;

— dans un second temps, étudier un cas plus complexe, également en dimension un, sans solution analytique, et comparer les résultats obtenus avec ceux d'une méthode par différences finies.

8.1 Exemple simple à solution explicite

Avant de tester l'algorithme sur des exemples complexes, il est pertinent de le valider sur un cas simple pour lequel la solution optimale est connue analytiquement. On considère ici un problème à une dimension, avec une dynamique indépendante du contrôle et des fonctions de coût particulièrement simples.

Données du problème

On se place dans le cadre suivant :

- ensemble des régimes : $A = \{0, 1\}$;
- dynamique : $dX_t = \sigma\, dW_t$, avec $X_0 = x_0 \in \mathbb{R}$;
- coût courant : $f(x, a) = 0$ pour tout (x, a) ;
- gain terminal : $g(x) = x$ pour tout x ;
- coût de switching : $c > 0$ constant.

Dans ce cadre, la trajectoire de X_t ne dépend pas du contrôle, et suit une loi gaussienne :

$$X_T \sim \mathcal{N}(x_0, \sigma^2 T).$$

Le seul terme influencé par la politique de switching est le nombre de changements de régime $N(\alpha)$, qui engendre une pénalité $-c\,N(\alpha)$. Le problème revient alors à maximiser :

$$J(\alpha) = \mathbb{E}[X_T] - c\,\mathbb{E}[N(\alpha)].$$

Solution optimale

L'espérance $\mathbb{E}[X_T]$ étant égale à x_0 pour toute politique, le contrôle optimal consiste à minimiser le nombre de commutations. Ainsi, la politique optimale est celle qui ne commute jamais, et reste dans le régime initial sur tout l'intervalle $[0, T]$. On en déduit :

$$V_0 = \sup_{\alpha \in \mathcal{A}} \left(\mathbb{E}[X_T] - c\,\mathbb{E}[N(\alpha)]\right) = x_0.$$

Autrement dit, toute tentative de changer de régime réduit strictement la récompense espérée. Ce problème simple constitue donc un bon cas test : l'algorithme doit apprendre à ne jamais commuter et à estimer correctement la fonction valeur constante $v_k(x, i) = x$ pour tout k et tout i.

Résultats numériques

L'algorithme a été appliqué au problème simple décrit ci-dessus, avec les paramètres suivants :

- horizon temporel : $T = 1$;
- nombre de pas de temps : $N = 10$;
- volatilité : $\sigma = 0.2$;
- coût de switching : $c = 0.05$;
- nombre d'états simulés par pas de temps : $M = 1000$;
- nombre de trajectoires Monte Carlo par état : $L = 10000$;
- nombre d'époques : $E_\phi = 100$ pour ϕ_θ, $E_\pi = 300$ pour p_ψ ;
- taux d'apprentissage : $\alpha_\phi = 10^{-3}$ et $\alpha_\pi = 5 \times 10^{-3}$.

Les courbes de pertes obtenues lors de l'apprentissage sont illustrées en figure 1. Elles montrent une bonne convergence de l'ensemble des réseaux à chaque pas de temps.

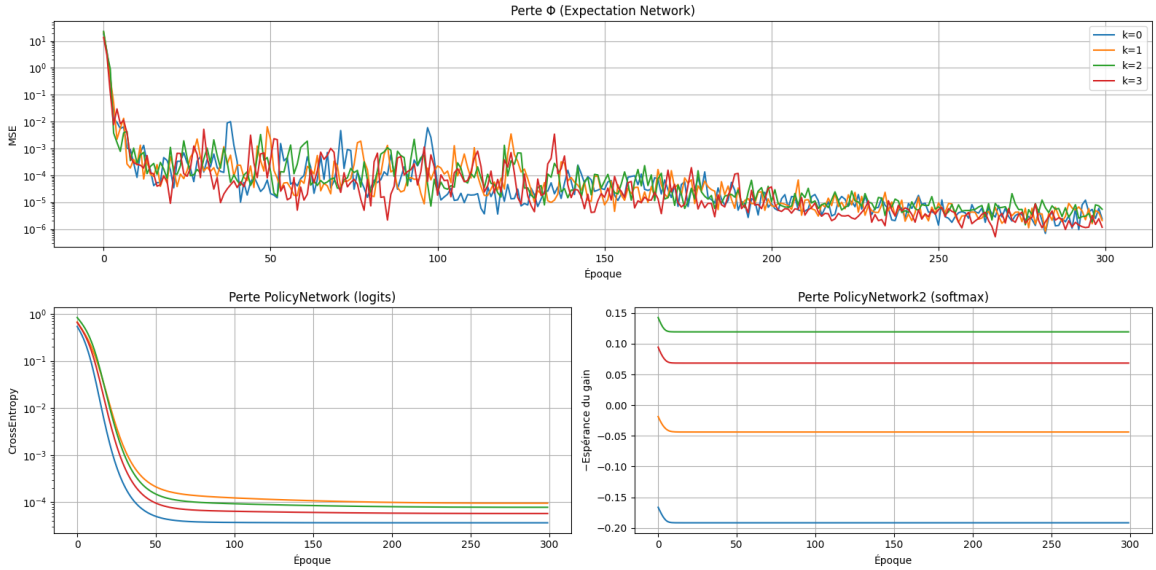

FIGURE 1 – Évolution des pertes pendant l'entraînement des réseaux de neurones. **Haut** : perte quadratique moyenne (MSE) du réseau ϕ_θ pour chaque instant k. **Bas gauche** : entropie croisée pour `PolicyNetwork`. **Bas droite** : perte négative de l'espérance du gain pour `PolicyNetwork2`.

Après entraînement, la fonction valeur $v_0(x, i)$ et la politique optimale $\pi(x)$ sont évaluées sur une grille de points $x \in [-40, 40]$. Les résultats sont représentés à la figure 2, pour les deux architectures testées : `PolicyNetwork` (logits) et `PolicyNetwork2` (softmax stochastique).

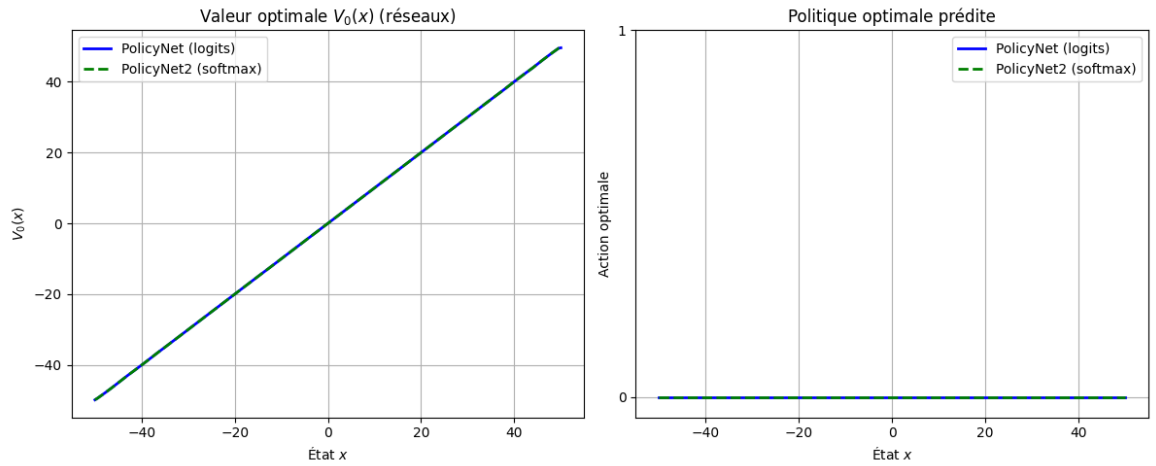

FIGURE 2 – Résultats de l'algorithme sur l'exemple simple. **Gauche** : estimation de la fonction valeur $v_0(x, i)$ par les deux variantes de politique. **Droite** : politique optimale apprise $\pi(x)$, constante égale à l'action initiale (absence de switching).

Les résultats numériques sont conformes aux attentes :

— la fonction valeur estimée $v_0(x, i)$ coïncide avec la fonction identité $x \mapsto x$;

— la politique optimale apprise est constante sur tout l'intervalle, indiquant l'absence de switching, ce qui est effectivement la stratégie optimale dans ce cas.

Ce premier test valide donc la capacité de l'algorithme à retrouver une solution exacte dans un cadre simple, en l'absence de structure complexe dans les coûts ou la dynamique.

8.2 Exemple non trivial : dynamique contrôlée et coûts non linéaires

On considère ici un problème plus complexe, dans lequel la dynamique du processus X_t dépend du contrôle, et les fonctions de coût ne sont plus linéaires. Ce cas ne possède pas de solution explicite, ce qui motive l'usage de méthodes numériques. On souhaite comparer les performances de l'algorithme proposé à celles d'une méthode par différences finies, en résolvant numériquement l'équation de Bellman discrète (4) introduite en section 5.

Données du problème

Les paramètres utilisés sont les suivants :

— horizon temporel : $T = 1$;

— nombre de pas de temps : $N = 4$;

— ensemble des actions : $A = \{0, 1\}$;

— volatilité : $\sigma = 0.1$;

— coût de switching : $c = 0.05$;

— fonctions de coût :

$$f(x, a) = -\frac{1}{2}x^2(1 + 0.2\,a), \quad g(x) = \begin{cases} x^2 & \text{si } x \geq 0, \\ -\frac{1}{2}x^2 & \text{sinon,} \end{cases}$$

— dynamique contrôlée :

$$dX_t = a_t X_t dt + \sigma(1 + 0.1 a_t X_t^2) dW_t.$$

Méthodes de résolution

Trois approches numériques sont comparées pour estimer la fonction valeur et la politique optimale. Les deux premières reposent sur l'utilisation de réseaux de neurones : l'une avec la politique déterministe (`PolicyNetwork`), l'autre avec la version stochastique (`PolicyNetwork2`). Ces méthodes exploitent une régression backward fondée sur l'équation de Bellman discrète. En parallèle, **une méthode de différences finies** est implémentée comme référence. Elle consiste à résoudre numériquement l'équation de Bellman (4) sur une grille d'états régulière, avec un pas de temps $\Delta t = T/N$. La discrétisation est centrée et implicite, tant pour les dérivées premières que secondes, et des conditions de Neumann homogènes sont imposées sur les bords. La politique optimale est ensuite obtenue par maximisation du gain attendu à chaque pas de temps.

Résultats numériques

La figure 3 illustre l'évolution des fonctions de perte au cours de l'apprentissage pour les différents réseaux, avec un batch size fixé à 128. Le réseau ϕ_θ parvient à atteindre une erreur quadratique moyenne de l'ordre de 10^{-4} pour les instants $k = 1, 2, 3$, tandis que les performances sont légèrement moins stables pour $k = 0$, ce qui s'explique par l'accumulation d'erreurs dans la régression backward et la propagation de bruit Monte Carlo depuis l'instant terminal. Les deux réseaux de politique montrent également une bonne convergence : la perte d'entropie croisée du `PolicyNetwork` décroît régulièrement pour tous les instants, tandis que le `PolicyNetwork2`, fondé sur une politique stochastique softmax, parvient à stabiliser l'espérance de gain dès les premières itérations. Ces observations confirment la robustesse de l'apprentissage, malgré la complexité du problème non linéaire considéré.

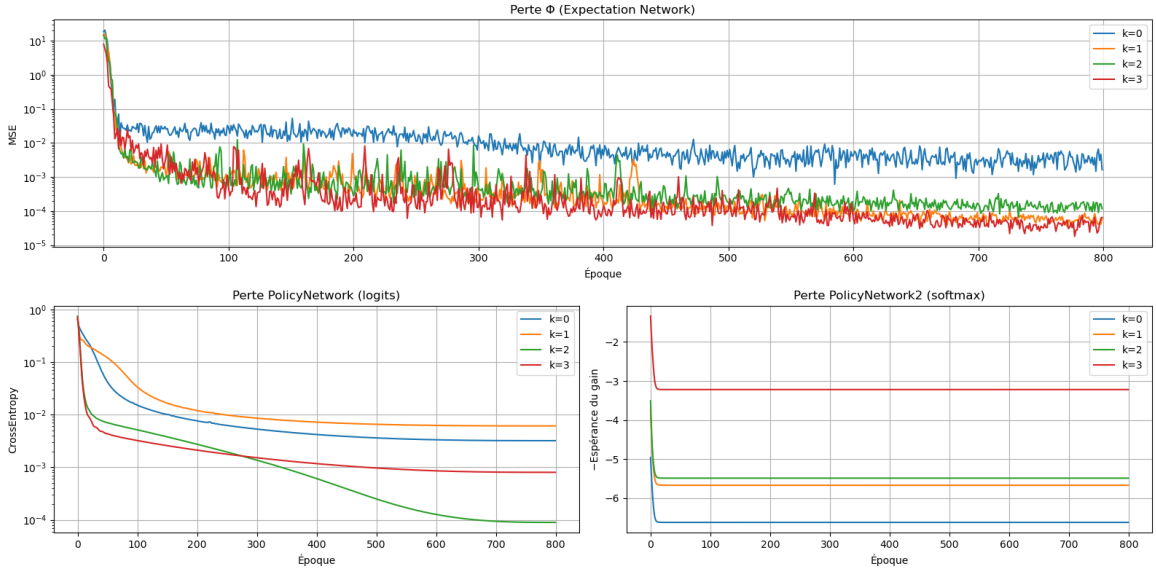

FIGURE 3 – Évolution des pertes pendant l'entraînement pour l'exemple non trivial. **Haut** : perte quadratique moyenne (MSE) du réseau ϕ_θ pour l'estimation des espérances conditionnelles, à chaque instant k. **Bas gauche** : entropie croisée pour le réseau `PolicyNetwork` (classification logits). **Bas droite** : perte négative de l'espérance de gain pour le réseau `PolicyNetwork2` (politique softmax).

La fonction valeur estimée $v_0(x, i)$ et la politique optimale $\pi(x)$ ont été comparées à la solution obtenue par différences finies. Les résultats sont illustrés dans les figures 4 et 5.

La figure 4 présente une comparaison entre les valeurs optimales initiales $v_0(x)$ estimées par les réseaux de neurones et celle obtenue par la méthode de différences finies. On observe une excellente concordance sur la partie gauche du domaine, pour $x \in [-3, 0]$, où l'approximation neuronale reproduit fidèlement la fonction valeur. En revanche, des écarts plus marqués apparaissent pour $x > 0$.

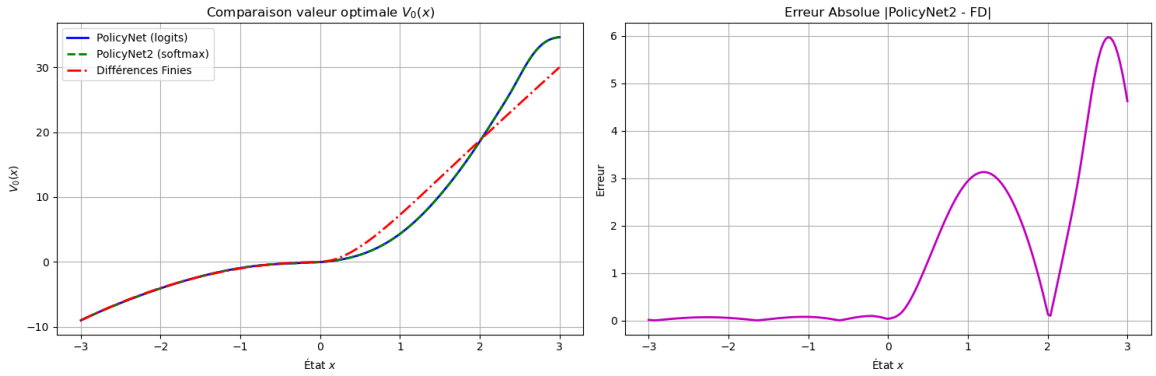

FIGURE 4 – **Gauche** : comparaison de la fonction valeur initiale $v_0(x)$ obtenue par `PolicyNetwork`, `PolicyNetwork2`, et la méthode de différences finies. **Droite** : erreur absolue $|v_0^{\text{PolicyNet2}}(x) - v_0^{\text{FD}}(x)|$ entre la méthode neuronale stochastique et les différences finies.

Remarque. Si les conditions de Neumann homogènes ($\frac{\partial v}{\partial n} = 0$) imposées dans la méthode par différences finies étaient responsables de l'erreur, celle-ci devrait se manifester de manière symétrique aux deux extrémités du domaine. Or, l'absence d'écart significatif pour $x < 0$ suggère que la source principale de l'erreur réside ailleurs. L'asymétrie du gain terminal, combinée à une dynamique contrôlée potentiellement plus instable pour les états positifs, pourrait induire une variabilité accrue dans l'estimation Monte Carlo, expliquant ainsi la dégradation locale de la précision pour $x > 0$.

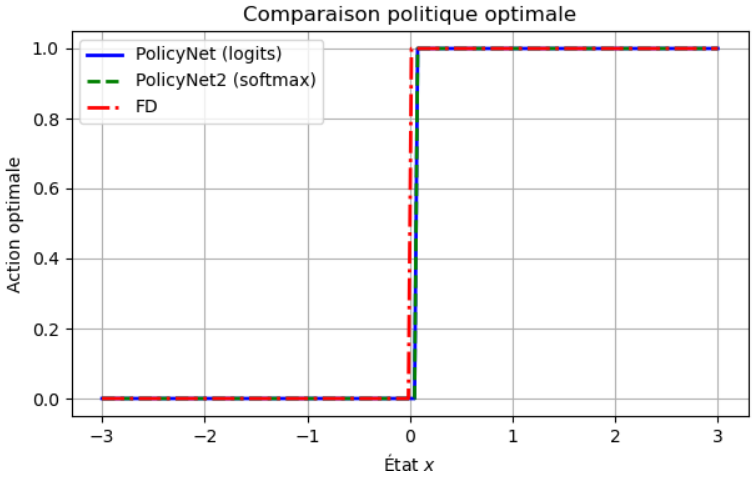

FIGURE 5 – Comparaison de la politique optimale $\pi(x)$ au temps initial $t = 0$. Les deux architectures de politique (logits et softmax) sont confrontées à la méthode de différences finies.

La figure 5 confirme que la politique optimale apprise par les deux réseaux est presque parfaitement alignée avec celle issue de la méthode de différences finies. La zone de switching autour de $x = 0$ est détectée avec une précision remarquable par les deux architectures, malgré la discrétisation implicite du problème et le bruit introduit par la simulation.

9 Extension en dimension deux : cas $d = 2$

Cette section propose une extension de l'algorithme au cas bidimensionnel, où l'état du système est un vecteur $X_t \in \mathbb{R}^2$. L'objectif est d'évaluer la robustesse de la méthode dans un espace d'état plus complexe, et de comparer les résultats obtenus avec ceux d'une méthode par différences finies.

9.1 Spécification du problème

La dynamique contrôlée est donnée par l'équation stochastique :

$$dX_t = b(X_t, a_t)\, dt + \sigma(X_t, a_t)\, dW_t,$$

où $a_t \in \{0, 1\}$ est le régime actif et W_t un mouvement brownien bidimensionnel. La dérive oriente le système vers le point $(-1, -1)$ ou $(1, 1)$ selon le régime choisi :

$$b(x, a) = \tfrac{1}{2}(2a - 1) \cdot \mathbf{1}_2, \quad \sigma(x, a) = 0.1 \cdot I_2.$$

Le coût courant est défini par :

$$f(x, a) = -\left[(x_1 - (2a - 1))^2 + (x_2 - (2a - 1))^2\right],$$

et le coût terminal par :

$$g(x) = -(x_1^2 + x_2^2).$$

Les paramètres numériques utilisés sont :

- horizon temporel $T = 1$ avec $N = 4$ pas de discrétisation ;
- ensemble des régimes $A = \{0, 1\}$ et coût de switching $c = 0.05$;
- taille d'échantillon $M = 3000$ et nombre de trajectoires Monte Carlo $L = 1000$;
- nombre d'époques d'apprentissage $E_\phi = E_\pi = 800$.

Méthode par différences finies en 2D

Une méthode de différences finies implicite est utilisée comme référence. L'équation de Bellman discrète est résolue sur une grille cartésienne de taille 31×31 sur le domaine $[-3, 3]^2$. Les dérivées spatiales sont approximées par des schémas centrés, et une maximisation est effectuée à chaque pas de temps pour prendre en compte le switching.

Résolution par apprentissage profond

Dans le cas bidimensionnel, l'algorithme neuronal repose sur les mêmes principes qu'en dimension un, mais avec un état $x = (x_1, x_2) \in \mathbb{R}^2$ et une dynamique indépendante sur chaque composante. Les réseaux de neurones utilisés sont adaptés à une entrée de dimension 2, et les hyperparamètres d'entraînement ont été ajustés (augmentation de la taille d'échantillon M, des époques E_ϕ, E_π, et du batch size).

La figure 6 montre l'évolution des fonctions de perte pour le réseau de régression Φ (gauche) et pour le classifieur de politique π (droite), sur les quatre derniers instants temporels $k = 0, 1, 2, 3$.

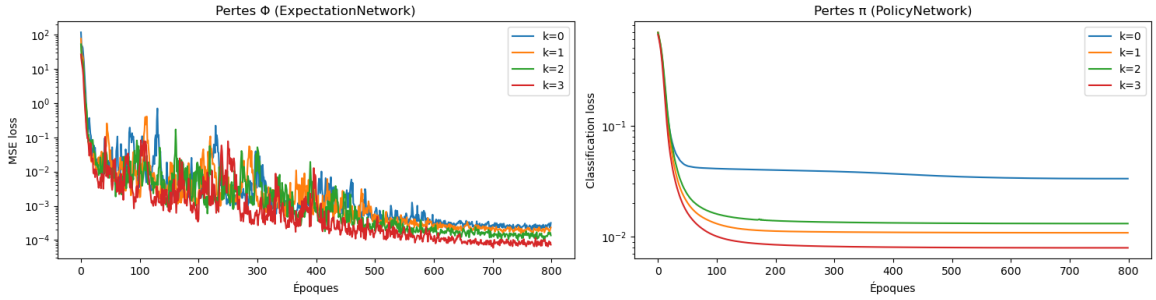

FIGURE 6 – Pertes d'apprentissage des réseaux en dimension $d = 2$. **Gauche** : fonction de coût quadratique pour Φ, montrant une convergence progressive. **Droite** : perte de classification pour π, décroissante et stabilisée à des valeurs faibles.

On observe que les pertes Φ diminuent rapidement dans les premières époques, puis atteignent une stabilité en dessous de 10^{-3}, malgré une variabilité due au bruit Monte Carlo. Les pertes π sont quant à elles significativement plus stables et décroissent rapidement pour tous les instants k, ce qui indique que la classification de l'action optimale est bien apprise. Ces observations suggèrent que l'algorithme par réseaux de neurones reste efficace malgré l'augmentation de la dimension de l'espace d'état.

9.2 Comparaison des résultats

La figure 7 présente une comparaison entre la fonction valeur initiale $v_0(x)$ obtenue par l'algorithme à réseaux de neurones et celle calculée par la méthode de différences finies, ainsi qu'une carte de l'erreur absolue entre les deux estimations.

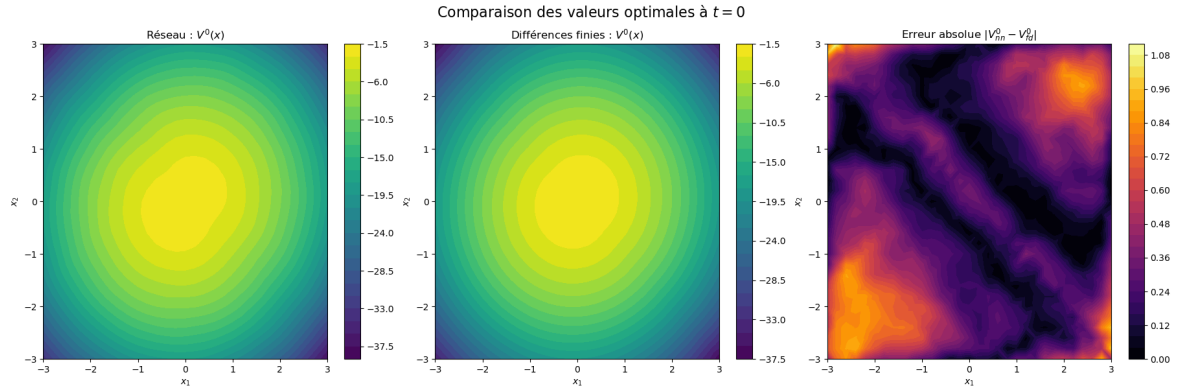

FIGURE 7 – Comparaison des fonctions valeur à $t = 0$ dans le cas $d = 2$. **Gauche** : valeur estimée par le réseau de neurones. **Centre** : valeur obtenue par différences finies. **Droite** : erreur absolue $|V^0_{\text{nn}} - V^0_{\text{fd}}|$.

Les deux surfaces présentent une structure similaire : v_0 est maximal (le moins négatif) au centre du domaine et décroît vers les bords, avec des lignes de niveau proches et une légère anisotropie. La carte d'erreur révèle une large zone centrale où l'écart est faible, tandis que des poches plus marquées apparaissent près de certains coins et le long de crêtes obliques, avec un maximum d'environ 1.1. Ces motifs sont cohérents avec la présence d'une frontière de switching orientée diagonalement et avec des effets de bord liés aux discrétisations.

La figure 8 compare ensuite les politiques optimales apprises par le réseau de neurones et par la méthode de différences finies, ainsi que les zones de désaccord entre les deux.

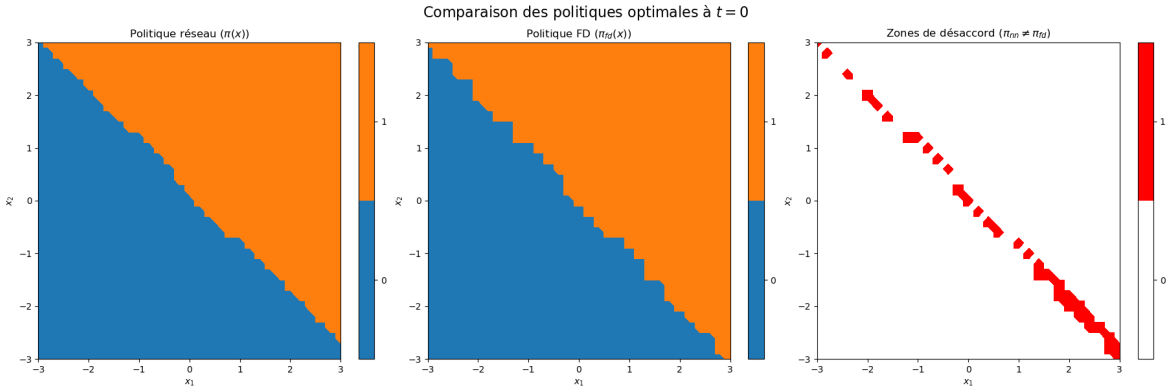

FIGURE 8 – Comparaison des politiques optimales à $t = 0$. **Gauche** : politique apprise par le réseau. **Centre** : politique obtenue par différences finies. **Droite** : carte binaire des désaccords, où la valeur 1 (rouge) indique un désaccord ($\pi_{\text{nn}} \neq \pi_{\text{fd}}$) et la valeur 0 (blanc) indique une concordance ($\pi_{\text{nn}} = \pi_{\text{fd}}$).

La frontière de switching $\pi_0(x)$ séparant les deux régimes apparaît clairement sur les deux méthodes, suivant approximativement la diagonale ($x_1 + x_2 = 0$). La politique apprise par le réseau est lisse et respecte globalement cette symétrie, tandis que celle calculée par différences finies présente une frontière plus anguleuse, reflétant la discrétisation en grille. Les zones de désaccord sont très localisées autour de cette frontière et traduisent une incertitude sur le régime optimal lorsque les valeurs associées aux deux actions sont proches. Globalement, la cohérence entre les deux politiques confirme que l'approche neuronale capture efficacement la structure optimale du problème, même en dimension deux.

9.3 Analyse des résultats en dimension deux

Cet exemple confirme que l'approche neuronale peut être généralisée à des espaces d'état de dimension deux, sans modification fondamentale de l'algorithme. Les résultats obtenus sont satisfaisants tant en termes de fonction valeur que de politique optimale. Néanmoins, le coût de calcul augmente significativement, et l'apprentissage peut devenir plus sensible aux fluctuations du bruit Monte Carlo.

Ces observations suggèrent que, pour des dimensions plus élevées, il pourrait être utile d'explorer des variantes plus stables de l'algorithme, intégrant par exemple des techniques de réduction de variance ou des architectures convolutives exploitant la structure spatiale.

10 Discussion

Nous discutons ici trois aspects clés mis en évidence par les expériences : le coût de calcul, l'effet du bruit Monte Carlo sur la stabilité de l'apprentissage, et l'écart au taux de convergence attendu. Ces éléments éclairent les performances observées pour v_0 et pour la politique optimale.

Temps d'exécution de l'algorithme

À chaque pas de temps, la procédure nécessite la simulation de M états et, pour chaque action, L trajectoires de Monte Carlo, soit une complexité par pas de l'ordre de $\mathcal{O}(ML|A|)$. Sur l'horizon entier, la complexité globale est donc en $\mathcal{O}(NML|A|)$. Ce coût, supérieur à celui d'une méthode par différences finies, reflète le compromis entre flexibilité en dimension et ressources de calcul : l'approche neuronale est plus adaptée aux problèmes de grande dimension, mais nécessite davantage de ressources.

Bruit Monte Carlo et instabilité de l'apprentissage

L'estimation Monte Carlo des espérances conditionnelles introduit un bruit non négligeable dans les cibles d'entraînement. Pour chaque état x et chaque action a, la quantité $\mathbb{E}[v_{k+1}(X_{k+1}, a) \mid X_k = x]$ est approchée par une moyenne empirique sur un nombre fini de trajectoires, ce qui engendre une variance significative lorsque L est modéré. Ce bruit affecte directement l'apprentissage de la fonction ϕ_k et se propage ensuite à l'apprentissage de la politique.
Cette instabilité est particulièrement visible dans les régions où la différence entre les gains associés aux différentes actions est faible : de petites fluctuations dans les estimations peuvent entraîner des décisions erronées. Elle est également amplifiée à mesure que l'on remonte dans le temps, puisque les erreurs sur les pas futurs s'accumulent dans la régression arrière.
Pour atténuer ce bruit, plusieurs techniques standard peuvent être intégrées : (i) *antithetic variates* et *common random numbers* (CRN) entre actions pour réduire la variance différentielle, (ii) *control variates*/baselines pour centrer les cibles, (iii) suites quasi-Monte Carlo (Sobol) pour une meilleure couverture, (iv) régularisation de la politique (entropie ou label smoothing) et *early stopping* guidé par une métrique de validation.

Écart entre le taux de convergence théorique et les résultats numériques

L'analyse théorique suggère que l'erreur induite par la discrétisation temporelle est en $\mathcal{O}(\sqrt{h})$, conformément à la convergence forte du schéma d'Euler–Maruyama. Toutefois, les résultats numériques n'ont pas permis de vérifier ce taux de manière claire. Lorsque $h = T/N$ diminue, la valeur estimée ne converge pas régulièrement vers la référence, et l'erreur ne décroît pas selon le comportement attendu.
Le taux $\mathcal{O}(\sqrt{h})$ concerne l'erreur de discrétisation sous un contrôle optimal exact. En pratique, l'erreur totale se décompose en

$$\underbrace{|V_0 - V_0^h|}_{\text{discrétisation}} + \underbrace{\varepsilon_{\text{stat}}}_{\text{Monte Carlo}} + \underbrace{\varepsilon_{\text{approx}}}_{\text{capacité réseau}} + \underbrace{\varepsilon_{\text{opt}}}_{\text{optimisation}},$$

ce qui peut masquer la décroissance en $\sqrt{h}$.

11 Conclusion

Ce travail a présenté et mis en œuvre une méthode d'apprentissage profond pour les problèmes de contrôle optimal stochastique avec switching à horizon fini. L'approche repose sur une discrétisation d'Euler–Maruyama et une induction arrière couplant deux réseaux : un estimateur d'espérance conditionnelle ϕ_k et une politique π_k. Les expériences en dimension un (cas simple à solution explicite puis exemple non linéaire) et l'extension en dimension deux montrent que la méthode retrouve des politiques proches de l'optimum et des fonctions valeur en bon accord avec une référence par différences finies, avec des écarts essentiellement localisés près des frontières de switching et en périphérie du domaine.

Cette étude met également en lumière plusieurs limites pratiques : un coût de calcul global en $\mathcal{O}(NML|A|)$ lié aux simulations Monte Carlo et à l'induction backward, une sensibilité au choix d'hyperparamètres, et une variabilité due au bruit statistique. Par ailleurs, le taux de convergence théorique $\mathcal{O}(\sqrt{h})$ n'a pas été observé de manière nette, ce qui s'explique par la superposition d'erreurs de discrétisation, statistiques (Monte Carlo), d'approximation (capacité des réseaux) et d'optimisation.

Dans la continuité de ce travail, nous envisageons d'étendre l'approche au *cadre à horizon infini* en formulant le problème en coût escompté (taux d'actualisation $\beta > 0$) comme un point fixe de l'opérateur de Bellman stationnaire. Cette reformulation permet d'apprendre des réseaux $\phi(x, a)$ et $\pi(x)$ *stationnaires* (indépendants du temps) et de s'appuyer sur les propriétés de contraction et les résultats d'existence/unicité de la fonction valeur associés à l'équation de Hamilton–Jacobi–Bellman stationnaire, tels que présentés dans [4]. Une validation empirique sur des cas tests à solution explicite et une analyse de stabilité (ergodicité, comportement aux bords) peuvent constituer les prochaines étapes.

Références

[1] P. E. Kloeden and E. Platen. *Numerical Solution of Stochastic Differential Equations.* Springer, Stochastic Modelling and Applied Probability, Vol. 23, 1992.

[2] B. Bouchard, I. Elie and N. Touzi. *Continuous-Time Stochastic Control and Optimization with Financial Applications.* Springer, Probability Theory and Stochastic Modelling, Vol. 94, 2021.

[3] A. Bachouch, C. Huré, A. Langrené and H. Pham. *Deep neural networks algorithms for stochastic control problems on finite horizon : numerical applications. Journal of Scientific Computing*, 87(2) :1–41, 2021.

[4] H. Pham, *Continuous-time Stochastic Control and Optimization with Financial Applications*, Springer, 2009.