Réseaux de neurones pour le pricing d'options bermudiennes

"Neural network regression for Bermudan option pricing", B. Lapeyre & J. Lelong (2021)

Céline Nguyen-Tu, M2 IFMA Ingénerie Financière et Modèles Aléatoires

Janvier 2025

Créées pour répondre aux besoins spécifiques des investisseurs, les options bermudiennes sont particulièrement populaires dans le cadre des produits structurés, des swaps de taux d'intérêt, ainsi que des instruments de crédit et de devises. Leur importance réside dans leur capacité à offrir des opportunités d'exercice discrètes en cours de vie, tout en conservant une structure de valorisation mathématiquement plus abordable que celle des options américaines. Ces instruments permettent aux investisseurs sur le marché de gré à gré *(over-the-counter)* de gérer leurs risques avec une granularité accrue. Leur popularité croissantes ont suscité un intérêt significatif dans la recherche académique, afin de développer des méthodes de valorisation efficaces et précises, adaptées à des marchés de plus en plus complexes.

L'article "Neural network regression for Bermudan option pricing" de B. Lapeyre et J. Lelong, publié en 2021, s'inscrit dans les avancées récentes en finance computationnelle. La valorisation des options bermudiennes repose historiquement sur des méthodes comme l'algorithme de Longstaff-Schwartz (2001), qui utilise des régressions linéaires pour estimer la valeur de continuation dans le cadre de la programmation dynamique. Bien que cette approche ait marqué une avancée majeure, elle présente des limites dans des scénarios complexes ou en haute dimension, où une relation linéaire ne suffit plus à capturer les dépendances entre variables. Pour surmonter ces limites, les auteurs proposent une méthode qui remplace les régressions linéaires par des réseaux de neurones *feed-forward* (FFNN), capables d'approcher des relations complexes et non linéaires entre les variables d'état, même dans des espaces de grande dimension.

Ce projet explore la valorisation des options bermudiennes à l'aide de la méthode proposée par l'article. Après une présentation du fonctionnement de ces instruments et des bases théoriques de leur valorisation, nous analyserons l'algorithme de Longstaff-Schwartz en mettant l'accent sur l'utilisation des réseaux de neurones pour approximer les valeurs de continuation. Enfin, des exemples numériques seront présentés pour comparer les performances des régressions linéaires et des FFNN dans des cas pratiques, soulignant les gains de précision et les implications computationnelles de cette nouvelle approche.

1 Pricing des options bermudiennes

Une option bermudienne est un produit dérivé qui donne à son détenteur le droit, mais non l'obligation d'acheter *(call)* ou de vendre *(put)* un actif sous-jacent (actions, indices, matières premières, etc.) à un prix d'exercice prédéfini *(strike)* à certaines dates spécifiques avant l'échéance. Elle combine les caractéristiques des options américaines (exerçables à tout moment) et européennes (exerçables uniquement à l'échéance) en proposant une flexibilité intermédiaire.

L'option peut être exercée à certaines dates discrètes $T_0, T_1, \ldots, T_N$, où T_0 est la date actuelle et T_N est l'échéance. À chaque date T_i, le détenteur choisit entre :

— Exercer l'option et recevoir son payoff immédiat $\max(S_{T_i} - K, 0)$ (pour une option call)

— Attendre et conserver l'option pour potentiellement obtenir un meilleur payoff à une date future.

L'objectif est de déterminer la valeur de l'option à T_0 en tenant compte de la stratégie optimale (politique d'arrêt) d'exercice. Pour comprendre l'approche de valorisation, intéressons nous au cas simple d'un modèle binomial avec seulement deux dates d'exercice.

1.1 Premier exemple simple avec deux dates d'exercice

On considère un actif sous-jacent X suivant un modèle binomial à deux périodes et une option bermudienne à deux dates d'exercice. Nous cherchons à valoriser l'option pour l'exemple suivant.

Soit $T_0 = 0$ la date actuelle et $N = 2$ dates d'exercice (T_1, T_2). On fixe les paramètres :

— Prix d'exercice $K = 100$.

— $X_0 = 100$ (prix initial de l'actif sous-jacent).

— Facteurs de montée et descente : $u = 1.1$, $d = 0.9$.
— Taux sans risque : $r = 0.05$.
— Probabilité neutre au risque : $p = \frac{e^{r\Delta t} - d}{u - d} \approx 0.756$.

Le prix de l'actif sous-jacent évolue selon :

$$X_{T_1} = \begin{cases} X_0 \cdot u & \text{avec proba } p \\ X_0 \cdot d & \text{avec proba } 1-p \end{cases} \qquad X_{T_2} = \begin{cases} X_{T_1} \cdot u & \text{avec proba } p \\ X_{T_1} \cdot d & \text{avec proba } 1-p \end{cases}$$

Le payoff du call au temps T_n est défini par $\tilde{Z}_{T_n} = \max(S_{T_n} - K, 0)$. On notera $Z_{T_n} = e^{-rT_n}\tilde{Z}_{T_n}$ le payoff actualisé, Δt l'écart de temps entre deux périodes, $\mathbb{P}$ la probabilité risque neutre et $\mathbb{E}$ l'espérance sous cette mesure.

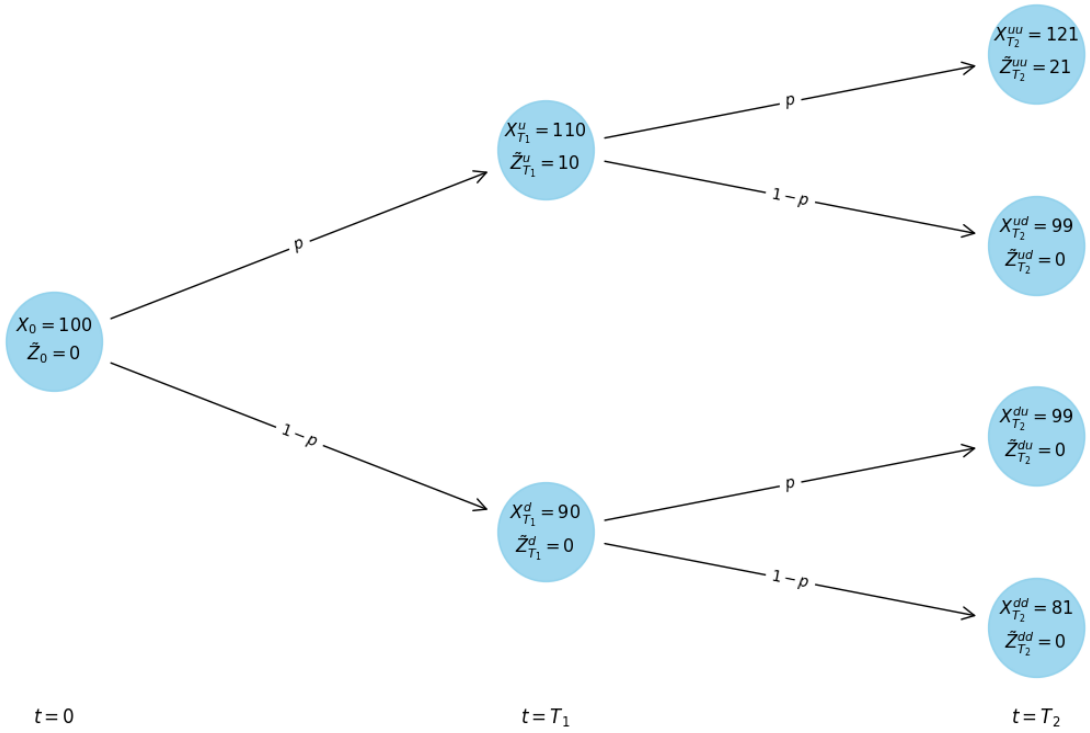

FIGURE 1 – Arbre binomial des prix de l'actif et des payoffs du call bermudien à 2 dates d'exercice

À T_2, la valeur de l'option V_{T_2} est simplement le payoff immédiat $\tilde{Z}$ car il s'agit de la date d'échéance :

$$V_{T_2} = \tilde{Z}_{T_n}$$

$$V^{uu}_{T_2} = 21 \qquad V^{ud}_{T_2} = 0 \qquad V^{du}_{T_2} = 0 \qquad V^{dd}_{T_2} = 0$$

À T_1, le prix de l'option est fixé non seulement par la nécessité de se couvrir pour la future valeur estimée de l'option au temps T_2, $\mathbb{E}[V_{T_2} \mid \mathcal{F}_1]$, mais aussi par le payoff obtenu si l'action est exercée en T_1, c'est-à-dire $\tilde{Z}_{T_1}$. Cette approche permet d'anticiper la stratégie d'exercice optimale du détenteur de l'option. Pour chaque noeud, on compare le payoff immédiat avec l'espérance actualisée des valeurs futures à T_2 :

$$V_{T_1} = \max\left(\tilde{Z}_{T_1}, e^{-r\Delta t}\mathbb{E}[V_{T_2} \mid X_{T_1}]\right)$$

$$V^u_{T_1} = \max\left(\tilde{Z}^u_{T_1}, e^{-r\Delta t}\mathbb{E}[V_{T_2} \mid X^u_{T_1}]\right) = \max\left(10, e^{-r\Delta t}(p \cdot V^{uu}_{T_2} + (1-p) \cdot V^{ud}_{T_2})\right) \approx 15.089$$

$$V^d_{T_1} = \max\left(\tilde{Z}^d_{T_1}, e^{-r\Delta t}\mathbb{E}[V_{T_2} \mid X^d_{T_1}]\right) = \max\left(0, e^{-r\Delta t}(p \cdot V^{du}_{T_2} + (1-p) \cdot V^{dd}_{T_2})\right) = 0$$

Enfin, à T_0, nous devons faire le même calcul : comparer le payoff immédiat avec l'espérance actualisée des valeurs à T_1.

$$V_{T_0} = \max\left(\tilde{Z}_{T_0}, e^{-r\Delta t}\mathbb{E}[V_{T_1} \mid X_{T_0}]\right) = \max\left(0, e^{-r\Delta t}(p \cdot V^u_{T_1} + (1-p) \cdot V^d_{T_1})\right) \approx 10.834$$

Ainsi, on observe une relation de récurrence :

$$\begin{cases} V_{T_N} = \tilde{Z}_{T_N} & n = N \\ V_{T_n} = \max\left(\tilde{Z}_{T_n}, e^{-r\Delta t}\,\mathbb{E}[V_{T_{n+1}} \mid \mathcal{F}_{T_n}]\right) & 0 \le n < N \end{cases}$$

En posant $U_{T_n} = e^{-rT_n} V_{T_n}$, on a la dynamique des valeurs actualisées au temps présent :

$$\begin{cases} U_{T_N} = e^{-rT_N} V_{T_N} = e^{-rT_N} \tilde{Z}_{T_N} = Z_{T_N} & n = N \\ U_{T_n} = e^{-rT_n} V_{T_n} = \max\left(e^{-rT_n} \tilde{Z}_{T_n}, e^{-rT_n} e^{-r\Delta t}\,\mathbb{E}[V_{T_{n+1}} \mid \mathcal{F}_{T_n}]\right) & \\ \quad = \max\left(Z_{T_n}, e^{-rT_{n+1}}\,\mathbb{E}[V_{T_{n+1}} \mid \mathcal{F}_{T_n}]\right) = \max\left(Z_{T_n}, \mathbb{E}[U_{T_{n+1}} \mid \mathcal{F}_{T_n}]\right) & 0 \le n < N \end{cases}$$

Le prix de l'option à la date actuelle T_0 est donc $U_0 = \max(Z_0, \mathbb{E}[U_{T_1} \mid \mathcal{F}_0])$.

1.2 Généralisation

En général, l'évolution du prix de l'actif sous-jacent ne suit plus un modèle discret, mais plutôt un processus stochastique markovien continu. Cela inclut notamment le modèle de Black-Scholes qui sera utilisé pour nos résultats numériques.

Pour une option européenne, il n'y a pas de choix sur le moment d'exercice : on exerce forcément à T. Par conséquent, le prix actualisé de l'option au temps T_0 est simplement l'espérance conditionnelle du payoff à la maturité.

$$U_{T_0} = \mathbb{E}[Z_T \mid \mathcal{F}_{T_0}]$$

Cela correspond à une situation où l'ensemble des temps d'arrêt admissibles $\mathcal{T}_{T_0,T}$ ne contient qu'un seul élément, $\tau = T$.

Pour une option bermudienne au temps T_n, on considère l'ensemble de toutes les dates d'exercice futures admissibles $\mathcal{T}_{T_n,T_N}$, et la valeur de l'option est déterminée par le meilleur payoff espéré, en maximisant sur tous les scénarios futurs (dates d'exercice) possibles.

$$U_{T_n} = \sup_{\tau \in \mathcal{T}_{T_n,T_N}} \mathbb{E}\left[Z_\tau \mid \mathcal{F}_{T_n}\right] \tag{1}$$

Dans l'exemple précédent, on a utilisé une intuition dite d'arrêt optimal *(optimal stopping)* afin de calculer le prix de l'option. Pour le détenteur de l'option, l'optimal stopping est une stratégie qui permet de décider au moment le plus avantageux quand exercer une option, en équilibrant le gain immédiat et la valeur future attendue. Cette stratégie optimale consiste à exercer à la première date T_i où le payoff immédiat dépasse la valeur d'attente future :

$$\tau^* = \inf\left\{T_i \in \{T_0, T_1, \ldots, T_N\} : \tilde{Z}_{T_i} \ge \mathbb{E}\left[U_{T_{i+1}} \mid X_{T_i}\right]\right\}.$$

Lorsqu'on cherche à calculer la valeur de l'option à chaque instant, il nous faut prendre en compte la possibilité d'exercer ou d'attendre en suivant cette stratégie optimale. C'est pourquoi à chaque point dans le temps, nous choisissons de maximiser la valeur de l'option en comparant l'exercice immédiat et l'attente jusqu'à la prochaine date d'exercice. Le principe demeure le même dans un cadre à temps continu et étendu à multiples dates $T_1, T_2, \cdots, T_N$.

On pourra alors évaluer le prix de l'option bermudienne à l'instant T_0 avec le même algorithme de programmation dynamique appelé enveloppe de Snell :

$$\begin{cases} U_{T_N} = Z_{T_N}, \\ U_{T_n} = \max\left(Z_{T_n}, \mathbb{E}\left[U_{T_{n+1}} \mid \mathcal{F}_{T_n}\right]\right), \quad 0 \le n \le N-1. \end{cases} \tag{2}$$

Cette équation peut être réécrite en terme de temps d'arrêt optimaux, c'est à dire pour déterminer le plus petit temps d'arrêt $\tau_n \ge T_n$ qui atteint le supremum des payoffs espérés dans l'équation (1) :

$$\begin{cases} \tau_N = T_N \\ \tau_n = T_n \mathbb{1}_{\left\{Z_{T_n} \ge \mathbb{E}\left[Z_{\tau_{n+1}} \mid \mathcal{F}_{T_n}\right]\right\}} + \tau_{n+1} \mathbb{1}_{\left\{Z_{T_n} < \mathbb{E}\left[Z_{\tau_{n+1}} \mid \mathcal{F}_{T_n}\right]\right\}}, \quad 1 \le n \le N-1 \end{cases} \tag{3}$$

Cette formulation permet de décomposer le problème complexe de valorisation des options bermudiennes en une suite de décisions locales.

Une fois le temps d'arrêt optimal $\tau_1 \in \mathcal{T}_{T_1,T_N}$ identifié à l'aide de cet algorithme, on peut estimer le prix de l'option au temps actuel T_0. En effet, d'après (1), on a $U_{T_1} = \mathbb{E}[Z_{\tau_1} \mid \mathcal{F}_{T_1}]$, et en utilisant l'expression de U_{T_0} dans (2), on a :

$$\begin{aligned} U_{T_0} &= \max\left(Z_{T_0}, \mathbb{E}\left[U_{T_1} \mid \mathcal{F}_{T_0}\right]\right) = \max\left(Z_{T_0}, \mathbb{E}\left[\mathbb{E}[Z_{\tau_1} \mid \mathcal{F}_{T_1}] \mid \mathcal{F}_{T_0}\right]\right) \\ &= \max\left(Z_{T_0}, \mathbb{E}[Z_{\tau_1} \mid \mathcal{F}_{T_0}]\right) = \max\left(Z_{T_0}, \mathbb{E}[Z_{\tau_1}]\right) \end{aligned}$$

Puisque l'on a choisit de considérer le modèle de Black-Scholes, on a identifié le processus markovien X et on peut réécrire les espérances conditionelles comme :

$$\mathbb{E}\left[Z_{\tau_{n+1}} \mid \mathcal{F}_{T_n}\right] = \mathbb{E}\left[Z_{\tau_{n+1}} \mid X_{T_n}\right] = \psi_n\left(X_{T_n}\right)$$

Cette équation correspond à un problème de régression qui se résout en minimisant le critère des moindres carrés, ainsi ψ_n satisfait :

$$\inf_{\psi \in L^2(\mathcal{L}(X_{T_n}))} \mathbb{E}\left[\left|Z_{\tau_{n+1}} - \psi\left(X_{T_n}\right)\right|^2\right],$$

où $L^2\left(\mathcal{L}\left(X_{T_n}\right)\right)$ est l'ensemble des fonctions mesurables f telles que $\mathbb{E}[f(X_{T_n})^2] < +\infty$.

Cette méthode de valorisation constitue l'algorithme de Longstaff-Schwartz où l'espérance conditionnelle à chaque itération, aussi appelée fonction de continuation ψ, est traditionnellement approchée par une régression polynomiale. Cette dernière est l'étape de l'algorithme qui sera remplacée par un réseau de neurones. La nouvelle approche permet de bénéficier de la non-linéarité des réseaux de neurones, ce qui est un atout par rapport aux méthodes de régression classiques qui se limitent souvent à des approximations linéaires, offrant ainsi une meilleure performance dans des contextes de haute dimension.

2 Feed-Forward Neural Network (FFNN)

Un feed-forward neural network (FFNN) est un type de réseau de neurones artificiels où les informations circulent uniquement dans une direction : des entrées vers les sorties, en passant par une ou plusieurs couches cachées. Il n'y a pas de cycles ni de rétroactions dans ce type de réseau. C'est une architecture fondamentale souvent utilisée pour approcher des fonctions complexes, ce qui en fait un choix pertinent pour approcher les espérances conditionnelles dans l'algorithme de Longstaff-Schwartz.

Dans notre contexte, il s'agit d'approcher une espérance conditionnelle que l'on notera pour généraliser dans cette section :

$$\mathbb{E}[Y \mid X = x] = \mathbb{E}[Y \mid x] \approx \Phi_\theta(x) = \Phi(X;\theta)$$

où $Y \in \mathbb{R}$ correspond au payoff et $X \in \mathbb{R}^r$ est la valeur de l'actif sous-jacent.

On utilise un échantillon représentatif de taille m caractérisé par les couples de variables (Y_1, X_1), $(Y_2, X_2), \ldots, (Y_m, X_m)$. À partir des observations (y_i, x_i), qui sont des réalisations des couples (Y_i, X_i) pour chaque trajectoires simulées $i = \{1, \ldots, m\}$, on peut estimer les paramètres θ qui caractérisent la fonction Φ_θ c'est-à-dire notre modèle. Pour se faire, on note $\hat{y}_i = \Phi_\theta(x_i)$ l'estimation de $\mathbb{E}[Y_i|x_i]$ par la fonction Φ_θ. L'objectif est de minimiser la distance entre les observations y_i et les prédictions du modèle, représentée par l'erreur quadratique moyenne :

$$\inf_{\theta \in \Theta} \frac{1}{m} \sum_{i=1}^{m} \left(y_i - \hat{y}_i\right)^2$$

La structure d'un FFNN correspond à la forme de la fonction non linéaire Φ_θ, plus ou moins profond selon le nombre de couches $L \geq 2$ et le nombre de neurones d_ℓ sur chacune des couches $\ell = 1, \cdots, L$. Le traitement des données de l'échantillon pour ajuster le modèle correspond à la phase d'apprentissage du modèle et se déroule comme suit :

1. **Entrée *(input)*** : Les variables $x^1, x^2, \ldots, x^r$, qui constituent les caractéristiques *(features)* du problème sont dirigées vers chaque neurone de la première couche.
2. **Couches cachées** :
   — Sur la première couche, chaque neurone $d = 1, \cdots, d_1$ effectue un calcul basé sur une combinaison linéaire des entrées
   $$z_d^{[1]} = \sum_{j=1}^{r} \omega_{d,j}^{[1]}\, x^j + \beta_d^{[1]}$$

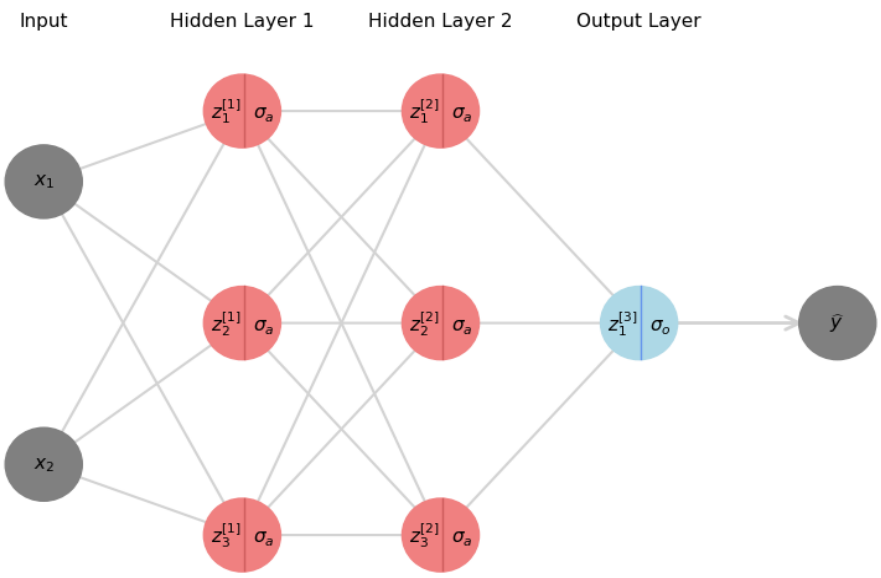

FIGURE 2 – Réseau de neurones FFNN à 2 couches cachées de 3 neurones avec entrées à 2 dimensions

où les $\omega_{d,j}^{[1]}$ sont les poids et les $\beta_d^{[1]}$ sont les biais de la couche $\ell = 1$. Puis son résultat est passé à une fonction dite d'activation σ_a qui permet d'introduire de la non-linéarité au modèle

$$\sigma_d^{[1]} = \sigma_a(z_d^{[1]})$$

— Sur les prochaines couches cachées, les mêmes calculs sont effectués mais cette fois à partir des données obtenus à l'issue de la couche précédente et non plus des données d'entrées. Ainsi, sur la couche ℓ, pour chaque neurone $d = 1, \cdots, d_\ell$, la somme pondérée est obtenue par :

$$z_d^{[\ell]} = \sum_{j=1}^{d_{\ell-1}} \omega_{d,j}^{[\ell]}\, \sigma_j^{[\ell-1]} + \beta_d^{[\ell]}$$

Cette étape correspond à une transformation par la fonction affine sur le vecteur des activations de la couche précédente via :

$$A_\ell : \left\{ \begin{array}{rcl} \mathbb{R}^{d_{\ell-1}} & \to & \mathbb{R}^{d_\ell} \\ x & \mapsto & A_\ell(x) = W^{[\ell]} \cdot x + \beta^{[\ell]} \end{array} \right.$$

où $W^{[\ell]} \in \mathbb{R}^{d_\ell \times d_{\ell-1}}$ est la matrice de poids et $\beta^{[\ell]} \in \mathbb{R}^{d_\ell}$ est le vecteur de biais qui caractérisent A_ℓ. Avant l'apprentissage, les poids sont initialisés aléatoirement et les biais à zéro (plusieurs méthodes d'initialisation existent).
La même fonction d'activation est ensuite appliquée au résultat de la transformation :

$$\sigma_d^{[\ell]} = \sigma_a(z_d^{[\ell]})$$

Les activations les plus utilisées sont ReLU, sigmoid et tanh. Dans notre cas, la fonction d'activation retenue est Leaky ReLU définie par :

$$\sigma_a(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0.3 \times x & \text{if } x < 0 \end{cases}$$

3. **Sortie** : La couche de sortie possède le même nombre de neurones que la dimension de la variable à prédire $Y \in \mathbb{R}$, donc ici $d_L = 1$. Elle applique une dernière fois une transformation A_L. Sa fonction d'activation est choisie selon la tâche du modèle (classification binaire, multi-classes, etc.). Pour une régression sans contrainte sur la plage des sorties, on n'en utilise pas ($\sigma_o(x) = x$). Cette couche retourne une valeur $\widehat{y}$, représentant la réponse du réseau, dans notre cas, une estimation de l'espérance conditionnelle.
4. **Rétropropagation *(Back-propagation)*** : À ce stade, une fois toutes les données d'un mini-lot *(batch)* de l'échantillon ayant traversé le réseau, on a prédit l'espérance conditionnelle avec une approximation paramétrique Φ_θ arbitraire. Pour finaliser l'entraînement il faut maintenant ajuster

le modèle, autrement-dit son paramètre $\theta = (W^{[\ell]}, \beta^{[\ell]})_{\ell=1,\dots,L} \in \mathbb{R}^{N_d}$, où $N_d = \sum_{\ell=1}^{L} d_\ell(1+d_{\ell-1})$, de sorte à minimiser la fonction de coût aussi dite de perte définie par :

$$\mathcal{L}(y, \widehat{y}) = \frac{1}{m} \sum_{i=1}^{m} (y_i - \hat{y}_i)^2$$

Cette fonction permet d'évaluer les erreurs du modèles par rapport aux variables cibles y (valeurs réelles) et sa forme est spécifique à un problème de régression.

Le réseau de neurones traite les données par batch et ajuste les paramètres pour chaque batch. Pour mettre à jour les paramètres, la back-propogation consiste à calculer les gradients ou les dérivées partielles de la fonction de perte $\mathcal{L}$ par rapport aux poids $W^{[\ell]}$ et biais $\beta^{[\ell]}$ pour chaque couche $\ell = 1, \cdots, L$ du réseau en partant de la couche de sortie et en remontant couche par couche vers l'entrée en utilisant la règle de la chaîne. Elles permettent de comprendre comment la fonction coût évolue par rapport aux différents paramètres W et β. Une fois les gradients calculés, les poids et biais sont mis à jour à l'aide d'un algorithme d'optimisation, le plus simple et connu étant la descente de gradient avec un taux d'apprentissage *(learning rate)* η :

$$W^{[\ell]} \leftarrow W^{[\ell]} - \eta \cdot \frac{\partial \mathcal{L}}{\partial W^{[\ell]}} \qquad \beta^{[\ell]} \leftarrow \beta^{[\ell]} - \eta \cdot \frac{\partial \mathcal{L}}{\partial \beta^{[\ell]}}$$

Pour nos simulations, on utilisera l'optimisateur Adam *(Adaptive Moment Estimation)* qui combine l'utilisation des moyennes carrées des gradients et un taux d'apprentissage adapté dynamiquement. Si on note les gradients de la fonction de perte :

$$g_{W^{[\ell]}} = \frac{\partial \mathcal{L}}{\partial W^{[\ell]}} \qquad g_{\beta^{[\ell]}} = \frac{\partial \mathcal{L}}{\partial \beta^{[\ell]}}$$

On définit la moyenne exponentielle des gradients (premier moment) où t est le nombre d'itération de l'optimisation depuis le début de l'entraînement :

$$m^{(t)}_{W^{[\ell]}} = \beta_1 \cdot m^{(t-1)}_{W^{[\ell]}} + (1-\beta_1) \cdot g^{(t)}_{W^{[\ell]}} \qquad m^{(t)}_{\beta^{[\ell]}} = \beta_1 \cdot m^{(t-1)}_{\beta^{[\ell]}} + (1-\beta_1) \cdot g^{(t)}_{\beta^{[\ell]}}$$

et la moyenne exponentielle des carrés des gradients (second moment) :

$$v^{(t)}_{W^{[\ell]}} = \beta_2 \cdot v^{(t-1)}_{W^{[\ell]}} + (1-\beta_2) \cdot \left(g^{(t)}_{W^{[\ell]}}\right)^2 \qquad v^{(t)}_{\beta^{[\ell]}} = \beta_2 \cdot v^{(t-1)}_{\beta^{[\ell]}} + (1-\beta_2) \cdot \left(g^{(t)}_{\beta^{[\ell]}}\right)^2$$

Pour les premières itérations, les estimations des moments sont biaisées (proches de 0). Adam applique une correction pour compenser ce biais :

$$\hat{m}^{(t)}_{W^{[\ell]}} = \frac{m^{(t)}_{W^{[\ell]}}}{1-\beta_1^t} \qquad \hat{m}^{(t)}_{\beta^{[\ell]}} = \frac{m^{(t)}_{\beta^{[\ell]}}}{1-\beta_1^t} \qquad \hat{v}^{(t)}_{W^{[\ell]}} = \frac{v^{(t)}_{W^{[\ell]}}}{1-\beta_2^t} \qquad \hat{v}^{(t)}_{\beta^{[\ell]}} = \frac{v^{(t)}_{\beta^{[\ell]}}}{1-\beta_2^t}$$

Les paramètres sont mis à jour avec le taux d'apprentissage η et les moments corrigés :

$$W^{[\ell]} \leftarrow W^{[\ell]} - \eta \cdot \frac{\hat{m}^{(t)}_{W^{[\ell]}}}{\sqrt{\hat{v}^{(t)}_{W^{[\ell]}}} + \epsilon} \qquad \beta^{[\ell]} \leftarrow \beta^{[\ell]} - \eta \cdot \frac{\hat{m}^{(t)}_{\beta^{[\ell]}}}{\sqrt{\hat{v}^{(t)}_{\beta^{[\ell]}}} + \epsilon}$$

où ϵ est une petite constante pour éviter la division par zéro. Par défaut, les hyperparamètres de l'algorithme d'Adam sont un taux d'apprentissage $\eta = 0.001$, des coefficients pour les moments $\beta_1 = 0.9$ (contrôle la contribution des gradients précédents) et $\beta_2 = 0.999$ (contrôle l'amortissement), et $\epsilon = 10^{-8}$. Adam est une version plus avancée et adaptative de la descente de gradient, particulièrement utile pour les problèmes complexes ou dans les premiers stades de l'entraînement.

Avec ce réseau de neurones, on aura approché la fonction de continuation avec la fonction Φ_θ exprimée comme une composition de fonction :

$$\Phi_\theta = A_L \circ \sigma_a \circ A_{L-1} \circ \cdots \circ \sigma_a \circ A_1$$

L'entraînement du modèle à l'aide de l'échantillon de données permet d'ajuster le paramètre θ qui la caractérise. Il est possible d'entraîner cet échantillon plusieurs fois en augmentant le nombre d'époques *(epochs)* afin d'améliorer la convergence des paramètres et la performance du modèle.

3 Convergence de la méthode

On note $\tau_n^p \in \mathcal{T}_{T_n,T_N}$ l'estimation du temps d'arrêt optimal au temps T_n obtenu à l'aide de l'algorithme de programmation dynamique (3) avec un réseau de neurone profond à p neurones sur chacune de ses couches cachées. L'approximation de la valeur de l'option bermudienne en T_n est alors exprimée comme :

$$U_{T_n}^p = \mathbb{E}\left[Z_{\tau_n^p} \mid \mathcal{F}_{T_n}\right]$$

Dans le contexte de la valorisation d'options, il est essentiel d'assurer une convergence fiable des approximations utilisées. C'est dans ce cadre que la proposition suivante se révèle fondamentale :

Proposition 3.1. *Si les payoffs sont intégrables dans L^2, i.e. $\mathbb{E}\left[\max_{0\leq n\leq N} \mid Z_{T_n} \mid^2\right] < \infty$. Alors,*

$$\lim_{p\to\infty} \mathbb{E}[Z_{\tau_n^p} \mid \mathcal{F}_{T_n}] = \mathbb{E}[Z_{\tau_n} \mid \mathcal{F}_{T_n}] \qquad \text{dans } L^2(\Omega) \quad \forall\, 1 \leq n \leq N$$

Ce résultat démontre la convergence de l'algorithme proposé pour le prix des options bermudiennes en utilisant des réseaux de neurones. Cette convergence est en particulier vérifiée pour $n = 1$. Ainsi, la valeur estimée de l'option $U_{T_0}^p$ au temps actuel $T_0 = 0$ avec l'approche par réseau de neurones est :

$$U_0^p = \max\left(Z_{T_0}, \mathbb{E}[U_{T_1}^p \mid \mathcal{F}_{T_0}]\right) \underset{p\to+\infty}{\longrightarrow} \max\left(Z_{T_0}, \mathbb{E}[U_{T_1} \mid \mathcal{F}_{T_0}]\right) = U_0$$

La valeur estimée de l'option converge vers la valeur réelle à mesure que la complexité du réseau de neurones (représentée par le paramètre p) augmente.

De plus, on peut fournir une évaluation du taux de convergence de cette approximation, à partir de la convergence du réseau de neurone vers l'espérance conditionnelle qu'il approche à chaque étape du principe de programmation dynamique. Soit $\Phi(X_{T_n};\theta_n)$ le réseau de neurone approchant la fonction de continuation $\mathbb{E}[Z_{\tau_{n+1}} \mid \mathcal{F}_{T_n}]$, son paramètre θ_n satisfait le problème de régression

$$\inf_{\theta\in\Theta_p} \mathbb{E}\left[|\Phi(X_{T_n},\theta) - Z_{\tau_{n+1}}|^2\right]$$

Or, comme l'espérance conditionnelle est une projection orthogonale, on a

$$\mathbb{E}\left[|\Phi\left(X_{T_n};\theta\right) - Z_{\tau_{n+1}}|^2\right] = \mathbb{E}\left[|\Phi\left(X_{T_n};\theta\right) - \mathbb{E}\left[Z_{\tau_{n+1}} \mid \mathcal{F}_{T_n}\right]|^2\right] + \mathbb{E}\left[|Z_{\tau_{n+1}} - \mathbb{E}\left[Z_{\tau_{n+1}}|\mathcal{F}_{T_n}\right]|^2\right]$$

Cela revient donc au problème de minimisation

$$\inf_{\theta\in\Theta_p} \mathbb{E}\left[|\Phi\left(X_{T_n};\theta\right) - \mathbb{E}\left[Z_{\tau_{n+1}} \mid \mathcal{F}_{T_n}\right]|^2\right] \tag{4}$$

Proposition 3.2. *Supposons que, pour tout $0 \leq n \leq N-1$, il existe une suite $(\delta_n^p)_p$ de réels positifs telle que*

$$\inf_{\theta\in\Theta_p} \mathbb{E}\left[|\Phi\left(X_{T_n};\theta\right) - \mathbb{E}\left[Z_{\tau_{n+1}} \mid \mathcal{F}_{T_n}\right]|^2\right] = \mathcal{O}(\delta_n^p) \quad \text{lorsque } p \to \infty$$

Alors,

$$\mathbb{E}\left[|\mathbb{E}\left[Z_{\tau_n^p} \mid \mathcal{F}_{T_n}\right] - \mathbb{E}\left[Z_{\tau_n} \mid \mathcal{F}_{T_n}\right]|^2\right] = \mathcal{O}\left(\sum_{i=n}^{N-1} \delta_i^p\right) \quad \text{lorsque } p \to \infty$$

Ce résultat met en évidence que la précision des approximations des espérances conditionnelles influence la convergence des prix des options bermudiennes. Plus précisément, il établit que si l'erreur d'approximation des espérances conditionnelles par les réseaux de neurones converge lorsque p tend vers l'infini, alors l'approximation du prix de l'option converge vers son prix réel dans $L^2(\Omega)$ à un taux de convergence proportionnel aux taux de convergence des réseaux de neurones.

On peut également démontrer la convergence de la méthode lorsque la taille de l'échantillon utilisé par l'algorithme augmente. En effet, le calcul du prix de l'option repose sur la relation suivante

$$U_0 = \max(Z_{T_0}, \mathbb{E}[Z_{\tau_1}])$$

où l'espérance est approchée par des simulations de Monte-Carlo sur un ensemble de M trajectoires. Pour démontrer la loi forte des grands nombres pour les estimateurs de l'algorithme de Longstaff-Schwartz par réseaux de neurones, on pose les hypothèses suivantes :

($\mathcal{H}1$) Pour tout $p \in \mathbb{N}, p > 1$, il existe $q \geq 1$ et $\kappa_p > 0$ tels que

$$|\Phi(x,\theta)| \leq \kappa_p \left(1 + |x|^q\right) \quad \text{pour tout } x \in \mathbb{R}^r \text{ et tout } \theta \in \Theta_p.$$

De plus, pour tout $1 \leq n \leq N-1$, les fonctions $\theta \in \Theta_p \mapsto \Phi\left(X_{T_n}, \theta\right)$ sont continues presque-sûrement.
($\mathcal{H}2$) Pour ce même q, on a $\mathbb{E}\left[\left|X_{T_n}\right|^{2q}\right] < \infty \quad$ pour tout $0 \leq n \leq N$.
($\mathcal{H}3$) Pour tout $p \in \mathbb{N}, p > 1$, et tout $1 \leq n \leq N-1$, on a $\mathbb{P}\left(Z_{T_n} = \Phi\left(X_{T_n}; \theta_n^p\right)\right) = 0$.
($\mathcal{H}4$) Pour tout $p \in \mathbb{N}, p > 1$, et $1 \leq n \leq N$, pour tous $\theta^1, \theta^2 \in \mathcal{S}_n^p = \arg\inf_{\theta \in \Theta_p} \mathbb{E}\left[\left|\Phi\left(X_{T_n}; \theta\right) - Z_{\tau_{n+1}^p}\right|^2\right]$,

$$\Phi\left(x; \theta^1\right) = \Phi\left(x; \theta^2\right) \quad \text{pour tout } x \in \mathbb{R}^r.$$

Dès lors, à p la profondeur des réseaux de neurones fixée, on peut montrer la convergence des réseaux vers les espérances conditionnelles puis du prix estimé U_0^M vers le prix réel de l'option lorsque $M \to +\infty$:

Proposition 3.3. *Soit $\theta_n^p \in \mathcal{S}_n^p$ qui minimise le problème de régression (4) et $\widehat{\theta}_n^M$ qui minimise le problème empirique. Alors, sous les hypothèses $(\mathcal{H}1) - (\mathcal{H}4)$, pour tout $n = 1, \ldots, N$ et $m = 1, \ldots, M$,*

$$\Phi\left(X_{T_n}^{(m)}; \widehat{\theta}_n^M\right) \underset{M \to +\infty}{\longrightarrow} \Phi\left(X_{T_n}^{(m)}; \theta_n^p\right) \quad p.s.$$

Théorème 3.4. *Sous les mêmes hypothèses, pour $\alpha = 1, 2$ et pour tout $n = 1, \ldots, N$,*

$$\lim_{M \to +\infty} \frac{1}{M} \sum_{m=1}^{M} \left(Z_{\hat{\tau}_n^{(m)}}^{(m)}\right)^{\alpha} = \mathbb{E}\left[\left(Z_{\tau_n}\right)^{\alpha}\right] \quad p.s.$$

En particulier, pour $\alpha = 1$, on la convergence de l'algorithme vers la valeur réelle du prix de l'option :

$$U_0^M = \max\left(Z_0, \mathbb{E}[Z_{\hat{\tau}_1}^M]\right) = \max\left(Z_0, \frac{1}{M} \sum_{m=1}^{M} Z_{\hat{\tau}_1^{(m)}}^{(m)}\right) \underset{M \to +\infty}{\longrightarrow} \max\left(Z_0, \mathbb{E}[Z_{\tau_1}]\right) = U_0 \quad \text{p.s.}$$

L'article de référence fournit une analyse théorique de la convergence de l'algorithme proposé, montrant que sous certaines conditions, l'utilisation de réseaux de neurones permet d'obtenir des estimations de prix des options qui s'améliorent avec la complexité du modèle p et la taille de l'échantillon M.

4 Mise en application

4.1 Algorithme de Longstaff-Schwartz

L'algorithme de Longstaff-Schwartz repose sur une approche de programmation dynamique pour déterminer le temps d'arrêt optimal à chaque étape. L'idée est d'estimer la valeur de l'option en simulant les trajectoires du sous-jacent et en régressant les payoffs futurs sur ces trajectoires. L'algorithme se structure de la façon suivante pour un nombre de neurones p fixé :

1. **Simulation des trajectoires du sous-jacent et des payoffs** :
   — On génère un grand nombre de trajectoires du prix de l'actif sous-jacent en utilisant, par exemple, le modèle de Black-Scholes : $X^{(m)} = \{X_{T_0}^{(m)}, X_{T_1}^{(m)}, \cdots, X_{T_N}^{(m)}\}$ avec $m = 1, \ldots, M$.
   — À partir des trajectoires du sous-jacent et de la fonction de payoff, on calcule les trajectoires des payoffs actualisés de l'option : $Z^{(m)} = \{Z_{T_0}^{(m)}, Z_{T_1}^{(m)}, \cdots, Z_{T_N}^{(m)}\}$ de l'option pour $m = 1, \cdots, M$.
   — On génère un deuxième ensemble de trajectoires indépendantes $(\widehat{X}^{(m)}, \widehat{Z}^{(m)})_{1 \leq m \leq M}$ de même taille que le précédent qui servira à l'entraînement des réseaux de neurones seulement. Celà permet d'éviter un biais dû au sur-ajustement de la fonction de continuation lorsqu'on utilise les mêmes données pour l'entraînement et l'estimation.
2. **Détermination des temps d'arrêt optimaux** :
   — Pour chaque trajectoire $m = 1, \ldots, M$, on détermine le temps d'arrêt optimal $\widehat{\tau}_1^{(m)}$ par la stratégie itérative rétrograde en comparant à chaque pas de temps n le payoff actualisé $Z_{T_n}^{(m)}$ et l'espérance conditionnelle $\mathbb{E}\left[Z_{\tau_{n+1}}^{(m)} \mid X_{T_n}^{(m)}\right] = \Phi(X_{T_n}^{(m)}; \widehat{\theta}_n^M)$ qui dépend de $X_{T_n}^{(m)}$.

$$\begin{cases} \hat{\tau}_N^{(m)} = T_N, \\ \hat{\tau}_n^{(m)} = T_n \mathbb{1}_{\left\{Z_{T_n}^{(m)} \geq \Phi\left(X_{T_n}^{(m)}; \widehat{\theta}_n^M\right)\right\}} + \hat{\tau}_{n+1}^{(m)} \mathbb{1}_{\left\{Z_{T_n}^{(m)} < \Phi\left(X_{T_n}^{(m)}; \widehat{\theta}_n^M\right)\right\}} \quad 1 \leq n \leq N-1 \end{cases}$$

— Pour faire celà, à chaque pas de temps $n = 1, \cdots, N$, on approche la fonction de continuation associée $\Phi(\cdot\,; \widehat{\theta}_n^M)$ par un réseau de neurones FFNN avec les entrées $(\widehat{X}_{T_n}^{(m)})_m$ et les cibles $(\widehat{Z}_{T_n}^{(m)})_m$. Le réseau est entraîné seulement avec les trajectoires *in-the-money* (c'est-à-dire $\widehat{Z}_{T_n}^{(m)} > 0$) car celles qui sont *out-of-the-money* n'ont pas de valeur à cet instant et ne sont donc pas pertinentes pour la décision d'exercice. Cette sélection de données améliore l'efficacité numérique. Le problème de régression revient donc à trouver le paramètre $\widehat{\theta}_n^M$ qui minimise la quantité :

$$\inf_{\theta \in \Theta_p} \mathbb{E}\left[\left| \Phi\left(\widehat{X}_{T_n}; \theta\right) - \widehat{Z}_{\tau_{n+1}} \right|^2 \mathbb{1}_{\{\widehat{Z}_{T_n} > 0\}} \right]$$

L'entraînement permet d'ajuster $\widehat{\theta}_n^M$ pour approximer $\mathbb{E}\left[\widehat{Z}_{\tau_{n+1}}^{(m)} \mid \widehat{X}_{T_n}^{(m)}\right] = \Phi(\widehat{X}_{T_n}^{(m)}; \widehat{\theta}_n^M)$. Ce paramètre définit la politique d'exercice optimale approximée et est conservé pour calculer le temps d'arrêt optimal dans l'algorithme avec les données $(X_{T_n}^{(m)}, Z_{T_n}^{(m)})_{1 \le m \le M}$, puisque les trajectoires de l'ensemble d'entraînement et l'ensemble d'application sont identiquement distribuées.

— Les temps d'arrêt optimaux des trajectoires *out-of-the-money* ne sont pas mis à jour ($\hat{\tau}_n^{(m)} = \hat{\tau}_{n+1}^{(m)}$) car un payoff nul ne justifie jamais l'exercice de l'option et ceux des trajectoires *in-the-money* sont mis à jour ($\hat{\tau}_n^{(m)} = T_n$) si le payoff $Z_{T_n}^{(m)}$ est supérieur ou égal à la fonction de continuation Φ évaluée en $X_{T_n}^{(m)}$.

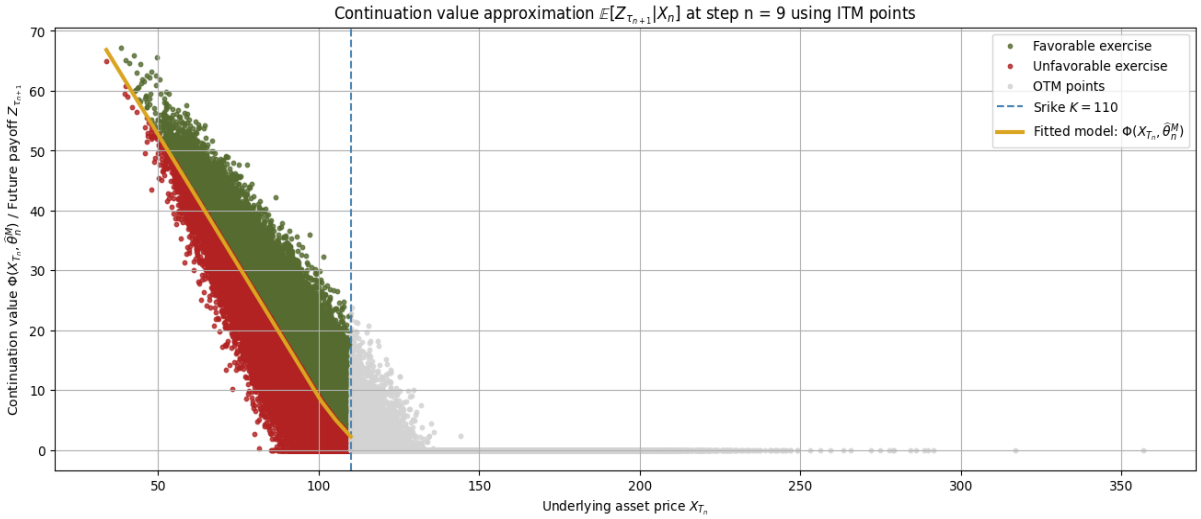

FIGURE 3 – Approximation de l'espérance conditionnelle par FFNN pour l'option Put en 1 dimension (voir section Exemples numériques)

3. **Monte Carlo** :
   — À partir des temps d'arrêt optimaux $\hat{\tau}_1^{(m)}$ obtenus pour toutes les trajectoires $m = 1, \cdots, M$, on peut désormais approcher $\mathbb{E}[Z_{\tau_1}]$ par son estimateur de Monte-Carlo :

$$\mathbb{E}[Z_{\tau_1}] \approx \frac{1}{M} \sum_{m=1}^{M} Z_{\hat{\tau}_1^{(m)}}^{(m)}$$

   — On obtient ainsi une approximation du prix de l'option bermudienne au temps $T_0 = 0$:

$$U_0^M = \max\left(Z_0, \frac{1}{M} \sum_{m=1}^{M} Z_{\hat{\tau}_1^{(m)}}^{(m)}\right)$$

4.2 Optimisation computationnelle

Il existe plusieurs manières d'intégrer ces réseaux dans l'algorithme de programmation dynamique.

La première consiste à construire et entraîner un nouveau modèle à chaque itération $1 \le n \le N-1$ de l'algorithme. Cette méthode est intuitive, car l'espérance conditionnelle à approcher diffère selon le moment de l'évaluation n. Cependant, son coût computationnel est très élevé car elle nécessite de

réentraîner un modèle FFNN à chaque étape, avec des poids initialisés à chaque fois, impliquant une convergence plus longue.

La seconde approche consiste à entraîner un modèle unique pour tous les pas de temps, en ajoutant une caractéristique temporelle aux entrées, soit $(x^1, x^2, \cdots, x^d, x^{d+1}) = (\widehat{X}^1_{T_n}, \widehat{X}^2_{T_n}, \cdots, \widehat{X}^d_{T_n}, T_n)$, où d est la dimension du problème. Cela permet aux approximations des valeurs de continuation d'être adaptées pour chaque étape de l'algorithme. L'entraînement peut alors commencer avec les poids optimaux de l'itération précédente $n+1$ et le modèle doit apprendre à s'adapter aux conditions spécifiques du moment n. Cette dernière permet un apprentissage déjà bien plus rapide.

Ces deux stratégies ont été utilisées lors des expérimentations numériques et ont donné de bons résultats, malgré le coût important de la première. Cependant, étant donné que l'espérance conditionnelle varie peu au fil du temps et qu'il n'est pas nécessaire de capturer les variations ou tendances entre les différentes étapes, on peut finalement opter pour un modèle unique à travers les itérations, sans ajouter la caractéristique temporelle explicite $x^{d+1} = T_n$, ce qui ajouterait un coût supplémentaire léger. La dynamique temporelle, bien que non explicitement modélisée, est déjà prise en compte grâce à la stratégie d'entraînement où les paramètres optimaux précédents $\widehat{\theta}^M_{n+1}$ sont réutilisés comme point de départ pour l'étape n. Cela permet au modèle d'apprendre les relations temporelles de manière implicite, en ajustant les poids et biais au fur et à mesure des itérations. Cette approche permet une convergence plus rapide de l'entraînement, car les poids et biais sont déjà proches de l'optimum pour chaque étape. Ainsi, après un bon ajustement aux données de l'étape initiale $N+1$ définit par le nombre d'époques, un seul passage sur les données pour les étapes ultérieures $1 \leq n < N+1$ est suffisant. Cette configuration réduit considérablement le temps de calcul tout en maintenant une bonne performance. C'est celle qui sera retenue pour obtenir les résultats de la section suivante.

5 Exemples numériques

Cette section a pour objectif de reproduire les exemples présentés dans l'article de référence afin d'évaluer les performances de la méthode proposée en les comparant à celles de la méthode traditionnelle basée sur la régression polynomiale. Elle vise également à étudier l'impact de la complexité du réseau (nombre de couches et de neurones) ainsi que du nombre d'époques d'entraînement sur la précision de l'approximation du prix des options. Enfin, l'application à différents types d'options bermudiennes mettra en évidence les performances de cette approche pour des problèmes en grande dimension. Dans la mesure du possible, nous utiliserons les mêmes configurations et paramètres que ceux de l'article afin de garantir une comparaison équitable et d'effectuer un benchmarking rigoureux de nos résultats.

Pour cela, nous simulons les trajectoires des actifs sous-jacents à l'aide de la méthode de Cholesky, puis nous implémentons l'algorithme de Longstaff-Schwartz avec deux approches distinctes : l'une basée sur des régressions polynomiales et l'autre sur des réseaux de neurones FFNN. Les modèles FFNN sont construits et entraînés avec la bibliothèque `tensorflow`, exploitant ses capacités de calcul parallèle pour optimiser le processus. Les régressions linéaires, quant à elles, sont réalisées à l'aide de la bibliothèque `scikit-learn`.

On se place dans un cadre neutre au risque et on considère le modèle de Black-Scholes multidimensionnel où le prix de l'actif sous-jacent $\mathbf{S}_t = (S_t^{(1)}, S_t^{(2)}, \ldots, S_t^{(d)})$ suit un mouvement brownien géométrique de dimension d :

$$dS_t^{(i)} = (r - \delta_i) S_t^{(i)}\, dt + \sigma_i S_t^{(i)}\, dW_t^{(i)} \qquad 0 \leq t \leq T$$

où $S_t^{(i)}$ est le prix du sous-jacent i à l'instant t, r est le taux d'intérêt sans risque, δ_i est le taux de dividende de l'actif i, σ_i est la volatilité de l'actif i et $\mathbf{W}_t = (W_t^{(1)}, W_t^{(2)}, \ldots, W_t^{(d)})$ un processus de Wiener multidimensionnel, avec $dW_t^i dW_t^j = \rho\, dt$ pour $i \neq j$, où $\rho \in]-\frac{1}{d-1}, 1]$ est le coefficient de corrélation commun entre tous les actifs. La matrice de corrélation associée $\mathbf{\Gamma}$ est donc définie par :

$$\mathbf{\Gamma} = \begin{pmatrix} 1 & \rho & \cdots & \rho \\ \rho & 1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \rho \\ \rho & \cdots & \rho & 1 \end{pmatrix}$$

Pour chacun de nos cas pratiques et pour chaque combinaison d'hyperparamètres, on simule 100 prix de l'option bermudienne avec l'algorithme de Longstaff-Schwartz (avec régression polynomiale et avec FFNN) pour en calculer la moyenne et l'intervalle de confiance à 95%. On pourra comparer nos résultats aux prix de références mentionnés par l'article.

5.1 Option Put à une dimension

L'option put est une option dérivée dont le sous-jacent est unidimensionnel et son payoff en t est :

$$(K - S_t)_+$$

où K est le prix d'exercice *(strike)* de l'option et S_t est le prix de l'actif sous-jacent à l'instant t.

Longstaff-Schwartz avec FFNN

L	d_ℓ	époques		
		1	5	10
2	32	11.969 (± 0.0064)	11.971 (± 0.0062)	11.973 (± 0.0063)
2	128	11.960 (± 0.0066)	11.967 (± 0.0066)	11.968 (± 0.0066)
2	512	11.942 (± 0.0082)	11.946 (± 0.0077)	11.948 (± 0.0074)
4	32	11.920 (± 0.0099)	11.929 (± 0.0100)	11.934 (± 0.0095)
4	128	11.885 (± 0.0122)	11.904 (± 0.0112)	11.916 (± 0.0103)
4	512	11.847 (± 0.0162)	11.877 (± 0.0146)	11.892 (± 0.0132)
8	32	11.886 (± 0.0129)	11.909 (± 0.0115)	11.917 (± 0.0109)
8	128	11.868 (± 0.0141)	11.896 (± 0.0119)	11.908 (± 0.0103)
8	512	11.643 (± 0.0641)	11.664 (± 0.0558)	11.676 (± 0.0628)

Longstaff-Schwartz avec régression polynomiale

ordre		
1	3	6
11.860 (± 0.0064)	11.984 (± 0.0063)	11.986 (± 0.0063)

FIGURE 4 – Option put : $d = 1$, $r = 0.1$, $\sigma = 0.25$, $\delta = 0$, $T = 1$, $K = 110$, $S_0 = 100$, $N = 10$, $M = 100000$ (Prix de référence = 11.987)

Ce premier exemple simple montre qu'avec des réseaux de neurones peu profonds, comme à 2 couches de 32 neurones, on peut déjà obtenir de bons résultats proches du prix réel de l'option, en particulier en augmentant le nombre d'entraînements pour corriger le biais d'approximation des espérances conditionnelles. Même si un reseau plus profond converge également mieux en augmentant le nombre d'epochs, un grand nombre de neurones entraine néanmoins une augmentation de la variance de l'algorithme.

5.2 Option Put Basket géométrique

Une option basket géométrique est une option dérivée dont le sous-jacent est une moyenne géométrique des prix de plusieurs actifs sous-jacents. Contrairement à une option classique dont le sous-jacent est un seul actif, une option basket prend en compte plusieurs actifs, ce qui permet de diversifier le risque tout en offrant un profil de payoff plus complexe. Ce type d'option est souvent utilisé dans les portefeuilles diversifiés, notamment pour couvrir des indices ou des paniers d'actions.

Le payoff d'une option put basket géométrique au temps t est défini par la formule suivante :

$$\left(K - \left(\prod_{j=1}^{d} S_t^j \right)^{1/d} \right)_+$$

où K est le prix d'exercice *(strike)* de l'option, S_t^i est le prix du i-ème actif sous-jacent à l'instant t et d est le nombre d'actifs sous-jacents dans le panier.

L'option put basket géométrique est un cas particulier d'option à haute dimension qui possède un équivalent en dimension 1, et dont le prix peut être calculé en utilisant les paramètres de ce dernier. Cette option constitue donc un bon exemple d'application pour évaluer la méthode et scalabilité.

Longstaff-Schwartz avec FFNN

L	d_ℓ	époques		
		1	5	10
2	32	4.547 (± 0.0040)	4.555 (± 0.0040)	4.556 (± 0.0038)
2	128	4.542 (± 0.0041)	4.545 (± 0.0042)	4.547 (± 0.0041)
2	512	4.534 (± 0.0043)	4.538 (± 0.0042)	4.538 (± 0.0043)
4	32	4.526 (± 0.0043)	4.528 (± 0.0044)	4.530 (± 0.0043)
4	128	4.508 (± 0.0050)	4.515 (± 0.0049)	4.519 (± 0.0047)
4	512	4.482 (± 0.0055)	4.501 (± 0.0051)	4.509 (± 0.0056)
8	32	4.508 (± 0.0050)	4.518 (± 0.0049)	4.523 (± 0.0046)
8	128	4.499 (± 0.0054)	4.510 (± 0.0048)	4.519 (± 0.0048)
8	512	4.427 (± 0.0217)	4.418 (± 0.0289)	4.453 (± 0.0153)

Longstaff-Schwartz avec régression polynomiale

ordre		
1	3	6
4.464 (± 0.0038)	4.570 (± 0.0037)	4.574 (± 0.0038)

FIGURE 5 – Option put Basket géométrique : $d = 2$, $r = 0.05$, $\sigma^i = 0.2$, $\delta^i = 0.2$, $\rho = 0$, $T = 1$, $K = 100$, $S_0^i = 100$, $N = 10$, $M = 100000$ (Prix de référence = 4.57)

Longstaff-Schwartz avec FFNN

L	d_ℓ	époques		
		1	5	10
2	32	2.716 (± 0.0027)	2.727 (± 0.0027)	2.728 (± 0.0027)
2	128	2.728 (± 0.0027)	2.727 (± 0.0027)	2.727 (± 0.0027)
2	512	2.717 (± 0.0027)	2.718 (± 0.0028)	2.719 (± 0.0027)
4	32	2.717 (± 0.0026)	2.722 (± 0.0028)	2.723 (± 0.0028)
4	128	2.708 (± 0.0027)	2.715 (± 0.0027)	2.718 (± 0.0026)
4	512	2.680 (± 0.0037)	2.692 (± 0.0034)	2.697 (± 0.0032)
8	32	2.711 (± 0.0028)	2.716 (± 0.0026)	2.718 (± 0.0026)
8	128	2.705 (± 0.0030)	2.713 (± 0.0027)	2.715 (± 0.0026)
8	512	2.666 (± 0.0052)	2.674 (± 0.0065)	2.675 (± 0.0074)

Longstaff-Schwartz avec régression polynomiale

ordre		
1	2	3
2.693 (± 0.0026)	2.752 (± 0.0027)	2.782 (± 0.0026)

FIGURE 6 – Option put Basket géométrique : $d = 10$, $r = 0.05$, $\sigma^i = 0.2$, $\delta^i = 0$, $\rho = 0.2$, $T = 1$, $K = 100$, $S_0^i = 100$, $N = 10$, $M = 100000$ (Prix de référence = 2.97)

Longstaff-Schwartz avec FFNN

L	d_ℓ	époques		
		1	5	10
2	128	2.300 (± 0.0020)	2.309 (± 0.0020)	2.310 (± 0.0019)
2	256	2.295 (± 0.0020)	2.303 (± 0.0021)	2.304 (± 0.0020)
2	512	2.285 (± 0.0020)	2.294 (± 0.0021)	2.296 (± 0.0021)
4	128	2.293 (± 0.0022)	2.301 (± 0.0021)	2.299 (± 0.0022)
4	256	2.287 (± 0.0021)	2.296 (± 0.0022)	2.293 (± 0.0021)
4	512	2.277 (± 0.0024)	2.284 (± 0.0023)	2.283 (± 0.0024)
8	128	2.300 (± 0.0025)	2.304 (± 0.0025)	2.301 (± 0.0025)
8	256	2.295 (± 0.0027)	2.299 (± 0.0028)	2.299 (± 0.0026)
8	512	2.277 (± 0.0039)	2.282 (± 0.0047)	2.278 (± 0.0071)

Longstaff-Schwartz avec régression polynomiale

ordre	
1	2
2.321 (± 0.0020)	2.428 (± 0.0019)

FIGURE 7 – Option put Basket géométrique : $d = 40$, $r = 0.5$, $\sigma^i = 0.2$, $\delta^i = 0$, $\rho = 0.2$, $T = 1$, $K = 100$, $S_0^i = 100$, $N = 10$, $M = 100000$ (Prix de référence = 2.52)

5.3 Option Put Basket

Une option basket est un produit dérivé qui offre à son détenteur le droit d'acheter ou vendre un panier d'actifs sous-jacents (souvent des actions). Contrairement à une option classique portant sur un seul actif, cette option dépend de la valeur agrégée pondérée des actifs constituant le panier.

Le payoff de cette option put au moment de l'exercice t est donné par :

$$\left(K - \sum_{i=1}^{d} w_i S_t^i\right)_+$$

où K est le prix d'exercice, S_t^i représente le prix de l'actif i dans le panier au temps t, w_i est le poids de l'actif i, et d est le nombre total d'actifs.

Longstaff-Schwartz avec FFNN

L	d_ℓ	époques		
		1	5	10
2	32	3.065 (± 0.0025)	3.084 (± 0.0026)	3.085 (± 0.0025)
2	128	3.080 (± 0.0027)	3.081 (± 0.0027)	3.082 (± 0.0028)
2	512	3.071 (± 0.0029)	3.073 (± 0.0030)	3.074 (± 0.0030)
4	32	3.068 (± 0.0030)	3.072 (± 0.0030)	3.075 (± 0.0031)
4	128	3.060 (± 0.0032)	3.066 (± 0.0032)	3.071 (± 0.0031)
4	512	3.035 (± 0.0033)	3.046 (± 0.0037)	3.052 (± 0.0035)
8	32	3.057 (± 0.0031)	3.064 (± 0.0033)	3.068 (± 0.0030)
8	128	3.053 (± 0.0031)	3.060 (± 0.0031)	3.067 (± 0.0029)
8	512	3.019 (± 0.0088)	3.033 (± 0.0051)	3.030 (± 0.0098)

Longstaff-Schwartz avec régression polynomiale

ordre		
1	2	3
3.063 (± 0.0026)	3.098 (± 0.0025)	3.108 (± 0.0024)

FIGURE 8 – Option put Basket : $d = 5$, $r = 0.05$, $\sigma^i = 0.2$, $\delta^i = 0$, $w_i = \frac{1}{d}$, $\rho = 0.2$, $T = 1$, $K = 100$, $S_0^i = 100$, $N = 10$, $M = 100000$

En dimension 5, les résultats obtenus avec réseaux de neurones sont semblables de ceux obtenus par régression polynomiale, les plus petits réseaux se rapprochant le plus du prix calculé par la régression polynomiale d'ordre 3.

Longstaff-Schwartz avec FFNN

L	d_ℓ	époques		
		1	5	10
2	32	2.132 (± 0.0019)	2.143 (± 0.0018)	2.147 (± 0.0019)
2	128	2.128 (± 0.0019)	2.138 (± 0.0019)	2.139 (± 0.0019)
2	512	2.117 (± 0.0021)	2.125 (± 0.0020)	2.127 (± 0.0021)
4	32	2.134 (± 0.0021)	2.142 (± 0.0022)	2.143 (± 0.0021)
4	128	2.125 (± 0.0022)	2.131 (± 0.0022)	2.128 (± 0.0022)
4	512	2.112 (± 0.0025)	2.120 (± 0.0028)	2.119 (± 0.0027)
8	32	2.139 (± 0.0023)	2.145 (± 0.0022)	2.144 (± 0.0023)
8	128	2.133 (± 0.0027)	2.138 (± 0.0025)	2.136 (± 0.0025)
8	512	2.110 (± 0.0038)	2.119 (± 0.0051)	2.118 (± 0.0051)

Longstaff-Schwartz avec régression polynomiale

ordre	
1	2
2.151 (± 0.0018)	2.244 (± 0.0019)

FIGURE 9 – Option put Basket : $d = 40$, $r = 0.05$, $\sigma^i = 0.2$, $\delta^i = 0$, $w_i = \frac{1}{d}$, $\rho = 0.2$, $T = 1$, $K = 100$, $S_0^i = 100$, $N = 10$, $M = 100000$ (Prix de référence $\in [2.15, 2.22]$)

Pour le problème en grande dimension, les prix obtenus sont proches de l'intervalle de référence. Les plus petits réseaux tendent encore à être plus précis que les réseaux profonds.

5.4 Option Call sur maximum

Une option sur le maximum d'un panier est un produit dérivé qui offre à son détenteur un droit basé sur la valeur maximale atteinte par un ensemble d'actifs sous-jacents (par exemple, des actions). Ce type d'option est particulièrement intéressant pour les investisseurs cherchant à tirer parti des meilleures performances individuelles d'un panier d'actifs.

Le payoff d'un call de cette option au temps t est donné par :

$$\left(\max_{i=1,\dots,d} S_t^i - K\right)_+$$

où K est le prix d'exercice, S_t^i est le prix de l'actif i au temps t et d est le nombre d'actifs dans le panier.

Longstaff-Schwartz avec FFNN

L	d_ℓ	époques		
		1	5	10
2	32	25.965 (± 0.0129)	25.854 (± 0.0130)	25.866 (± 0.0119)
2	128	25.917 (± 0.0126)	25.868 (± 0.0119)	25.898 (± 0.0121)
2	512	25.807 (± 0.0125)	25.838 (± 0.0119)	25.876 (± 0.0133)
4	32	25.794 (± 0.0141)	25.838 (± 0.0153)	25.852 (± 0.0149)
4	128	25.729 (± 0.0176)	25.754 (± 0.0166)	25.799 (± 0.0152)
4	512	25.628 (± 0.0196)	25.677 (± 0.0186)	25.722 (± 0.0177)
8	32	25.685 (± 0.0194)	25.726 (± 0.0187)	25.765 (± 0.0181)
8	128	25.596 (± 0.0238)	25.672 (± 0.0210)	25.726 (± 0.0194)
8	512	25.436 (± 0.0695)	25.545 (± 0.0437)	25.620 (± 0.0365)

Longstaff-Schwartz avec régression polynomiale

ordre		
1	3	6
25.312 (± 0.0125)	25.937 (± 0.0123)	26.026 (± 0.0122)

FIGURE 10 – Option call sur maximum : $d = 5$, $r = 0.05$, $\sigma^i = 0.2$, $\delta^i = 0.1$, $\rho = 0$, $T = 3$, $K = 100$, $S_0^i = 100$, $N = 9$, $M = 100000$ (Prix de référence $\in [26.14, 26.17]$)

Longstaff-Schwartz avec FFNN

L	d_ℓ	époques		
		1	5	10
2	128	68.823 (± 0.0226)	68.876 (± 0.0221)	68.769 (± 0.0242)
2	256	68.930 (± 0.0236)	68.835 (± 0.0247)	68.783 (± 0.0253)
2	512	68.935 (± 0.0262)	68.807 (± 0.0267)	68.788 (± 0.0267)
4	128	68.599 (± 0.0734)	68.634 (± 0.0636)	68.684 (± 0.0561)
4	256	68.616 (± 0.0809)	68.647 (± 0.0729)	68.672 (± 0.0672)
4	512	68.618 (± 0.0890)	68.629 (± 0.0827)	68.651 (± 0.0761)
8	128	68.484 (± 0.1090)	68.522 (± 0.0974)	68.516 (± 0.0833)
8	256	68.554 (± 0.0982)	68.574 (± 0.0925)	68.594 (± 0.0805)
8	512	68.363 (± 0.1979)	68.539 (± 0.0916)	68.563 (± 0.0936)

Longstaff-Schwartz avec régression polynomiale

ordre	
1	2
67.959 (± 0.0130)	69.198 (± 0.0141)

FIGURE 11 – Option call sur maximum : $d = 50$, $r = 0.05$, $\sigma^i = 0.2$, $\delta^i = 0.1$, $\rho = 0$, $T = 3$, $K = 100$, $S_0^i = 100$, $N = 9$, $M = 100000$ (Prix de référence $\in [69.56, 69.95]$)

Les prix obtenus en dimension 5 avec un petit réseau de neurones sont les plus proches du prix de référence, avec une meilleure précision pour les réseaux peu entraînés. On observe clairement dans cet exemple qu'augmenter le nombre d'époques apporte une amélioration que pour les réseaux très profonds. Il en est de même pour le problème en dimension 50. Comme pour les options précédentes, les régressions polynomiales sont plus limitées à mesure que la dimension augmente, rendant la méthode par FFNN plus scalable.

6 Conclusion

L'article de Lapeyre et Lelong représente une avancée notable dans le domaine de la valorisation des options bermudiennes, en intégrant des outils modernes d'apprentissage automatique dans un problème classique de finance quantitative. Les enjeux dans le calcul du prix des options bermudiennes sont principalement liés à l'estimation de la valeur de continuité, qui est réalisée à l'aide de réseaux de neurones à propagation avant (FFNN) dans la nouvelle approche. Cette approche est applicable à divers modèles financiers standards, incluant ceux à volatilité locale et stochastique, ce qui en fait un outil prometteur pour la valorisation des options bermudiennes.

Les résultats de nos diverses applications numériques montrent que les prix calculés sont comparables à ceux obtenus par l'algorithme Longstaff-Schwartz classique. En effet, les réseaux de neurones, même de petite taille, fournissent des estimations précises et nécessitent moins de passages sur les données, optimisant ainsi le temps de calcul. Néanmoins, le principal avantage de cette méthode réside dans la capacité des réseaux de neurones à modéliser des relations complexes et non linéaires, ce qui leur permet de mieux capturer les valeurs de continuation dans des contextes à haute dimension. Cet atout la différencie de la méthode à régression polynomiale, très limitée en raison de la malédiction de la dimensionnalité.

L'optimisation computationnelle choisie suggère que les réseaux de neurones peuvent être suffisamment flexibles pour modéliser les variations temporelles, tout en restant efficaces en termes de calcul. Cela dit, le choix entre un modèle dédié pour chaque date n ou un modèle unique avec ou sans feature temporelle supplémentaire dépendra des spécificités de la situation, telles que la variabilité des données à travers le temps et les différentiels entre les dates d'exercice.

Rappelons que le vrai prix d'une option bermudienne est donné par la valeur maximale que l'on peut obtenir en suivant une politique d'exercice optimale τ^*. Or, l'algorithme de Longstaff-Schwartz repose sur une approximation $\tau^{p,M}$ de cette politique et donc n'est pas garantie d'être exactement optimale.

$$U_0^{p,M} = \mathbb{E}\left[Z_{\tau^{p,M}}\right] \leq \mathbb{E}\left[Z_{\tau^*}\right] = U_0$$

Ainsi les résultats obtenus avec cette méthode constituent une borne inférieure du prix réel de l'option. En pratique, il sera judicieux de la combiner avec des méthodes de dualité comme Andersen-Broadie pour calculer une borne supérieure.

7 Bibliographie

B. Lapeyre & J. Lelong, "Neural network regression for Bermudan option pricing", *Monte Carlo Methods Applications 27(3) : 227-247*, De Gruyter, 2021.

J. Hull, "Options, futures et autres actifs dérivés", *10^{th} Edition*, Pearson, 2018.

F. A.Longstaff & R. S.Schwartz, "Valuing American options by simulation : A simple least-square approach", *Rev.Financial Stud. 14 ,113-147*, 2001.

E. Clément, D. Lamberton & P. Protter, "An analysis of the Longstaff-Schwartz algorithm for American option pricing", 2001.

L. Andersen & M. Broadie, "Primal-Dual Simulation Algorithm for Pricing Multidimensional American Options", 2001