Pricing Options and Computing Implied Volatilities Using Neural Networks

Qiyao Zhou, Yibo Wang

M2 IFMA, Sorbonne Uniersité

Janvier 2025

Abstract : Ce rapport présente la reproduction des résultats de l'article intitulé *Pricing Options and Computing Implied Volatilities Using Neural Networks*. L'étude originale l'utilisation d'un réseau de neurones artificiels pour valoriser plus rapidement des options financières et calculer des volatilités implicites, avec des méthodes numériques et des modèles financiers. Notre reproduction reprend en grande partie la méthodologie et le modèle décrits dans l'article, et nous avons comparé et analysé nos résultats avec ceux de la publication originale. Toutefois, nous n'avons pas adopté certaines méthodes numériques de l'article : nous avons choisi la méthode de Monte Carlo afin de mieux mettre en pratique et d'appliquer les connaissances acquises en cours.

---

1 Introduction

Un contrat d'option est un produit dérivé financier conférant à son détenteur le droit, sans toutefois l'obligation, d'acheter (call) ou de vendre (put) un actif sous-jacent, soit à une date d'expiration précise (option européenne), soit à tout moment avant cette échéance (option américaine). Plusieurs méthodes classiques sont couramment utilisées pour valoriser une option, parmi lesquelles le modèle binomial, les simulations de Monte-Carlo ou encore la formule de Black-Scholes. Le choix de la méthode mathématique et numérique la plus appropriée dépend généralement des caractéristiques spécifiques de l'option : dépendance au chemin suivi par l'actif, possibilité d'exercice anticipé et complexité dimensionnelle (faible ou élevée).

Cependant, ces méthodes d'évaluation reposent sur des hypothèses spécifiques concernant la dynamique sous-jacente, ce qui peut limiter leur capacité à refléter les observations empiriques. Par exemple, le modèle de Black-Scholes repose sur l'hypothèse d'une volatilité constante, ce qui l'empêche d'expliquer adéquatement la relation entre la volatilité implicite et la moneyness de l'option. Plus généralement, la complexité multidimensionnelle des problèmes réels de valorisation des dérivés rend souvent impossible l'obtention d'une formule fermée, nécessitant ainsi le recours à des approches numériques ou stochastiques plus flexibles.

Pour répondre à ces limitations, les algorithmes d'apprentissage profond (Deep Learning) ont gagné en popularité, offrant un compromis intéressant entre précision et efficacité numérique. En particulier, les réseaux de neurones artificiels (Artificial neural network, ANN) se distinguent par leur capacité à modéliser de manière approximative et efficace des relations de prix complexes et non linéaires, sans nécessiter d'hypothèses spécifiques sur la dynamique sous-jacente.

Ce projet a pour objectif de reproduire les travaux présentés dans ***Pricing Options and Computing Implied Volatilities Using Artificial Neural Network***[1]. Dans cet article, les auteurs proposent une approche reposant sur les réseaux de neurones artificiels pour évaluer les options d'achat européennes et calculer les volatilités implicites, qu'ils comparent ensuite à une méthode numérique classique, la méthode COS. Après avoir établi le cadre mathématique du problème (Section 2), nous évaluons l'approche ANN (Section 3) en la confrontant à des solutions analytiques. Plus précisément, nous utilisons l'équation de Black-Scholes, la méthode de Monte Carlo à la place de la méthode COS pour le modèle de volatilité stochastique de Heston, ainsi que la méthode itérative de Brent pour le calcul des volatilités implicites (Section 4).

2 Modèles d'Actifs et Évaluation des Options Vanilles

Dans cette section, nous présentons les outils mathématiques qui serviront de référence et qui seront utilisés pour générer des données de marché artificielles, notamment les modèles de Black-Scholes et de Heston. Nous décrivons également les méthodes numériques mises en œuvre pour l'évaluation des options d'achat dans le cadre du modèle de Heston (méthode de Monte Carlo) et pour le calcul de leur volatilité implicite correspondante (méthode de Brent).

2.1 Modèle de Black-Scholes

Un modèle bien connu pour le prix sous-jacent S_t est le modèle de Black-Scholes, dans lequel S_t suit un mouvement brownien géométrique qui satisfait l'EDS suivante :

$$dS_t = S_t \left(r\, dt + \sigma\, dW_t \right)$$

où r est le taux sans risque, σ la volatilité de l'actif sous-jacent, et le processus $(W_t)_{t\geq 0}$ est un mouvement brownien sous la mesure de probabilité neutre au risque Q. En utilisant une approche martingale ou un portefeuille auto-financé de réplication, le prix d'une option d'achat européenne doit satisfaire l'EDP suivante :

$$\frac{\partial V}{\partial t} + \frac{1}{2}\sigma^2 S^2 \frac{\partial^2 V}{\partial S^2} + rS\frac{\partial V}{\partial S} - rV = 0$$

avec la condition finale au temps T représentant le paiement de l'option d'achat :

$$V(T, S) = \max(S_T - K, 0)$$

La solution de l'équation ci-dessus est donnée par la formule de Feynman-Kac, qui affirme que la valeur à chaque instant t d'une option européenne peut être exprimée comme l'espérance conditionnelle des valeurs actualisées attendues de la fonction de paiement de l'option à l'échéance T, sous la mesure neutre au risque Q :

$$V(t, S) = e^{-r(T-t)} E_Q \left[\max(S_T - K, 0) \,|\, \mathcal{F}_t \right]$$

La résolution de l'EDS et le développement de la quantité ci-dessus conduisent à une formule explicite pour le prix de l'option d'achat V :

$$V(t,S) = SN(d_1) - Ke^{-r(T-t)}N(d_2)$$

où

$$d_1 = \frac{\ln(S/K) + \left(r - \frac{\sigma^2}{2}\right)(T-t)}{\sigma\sqrt{T-t}}, \quad d_2 = d_1 - \sigma\sqrt{T-t}.$$

Ce modèle est valide sous plusieurs hypothèses fondamentales : il n'existe aucune opportunité d'arbitrage, les coûts de transaction et la fourchette bid-ask sont négligeables, les échanges peuvent se faire de manière continue et en toutes quantités, y compris par la vente à découvert. De plus, le taux sans risque r et la volatilité σ sont supposés constants, le prix actualisé de l'actif sous-jacent suit un processus log-normal, et aucun dividende n'est distribué pendant la durée de vie de l'option.

2.2 Volatilité Implicite

Dans le modèle de Black et Scholes, le seul paramètre qui n'est pas directement observable est la volatilité σ. La fonction $\sigma \mapsto V(\sigma)$ vérifie les propriétés suivantes :

- $\lim_{\sigma\to 0} V(\sigma) = (S_t - Ke^{-r(T-t)})_+$,
- $\lim_{\sigma\to+\infty} V(\sigma) = S_t$,
- $\frac{\partial V(\sigma)}{\partial \sigma} = S_t n(d_1)\sqrt{T-t} > 0$,

Cette dernière quantité définit le Véga de l'option (avec n, la fonction densité de la loi normale centrée réduite). Ces trois relations impliquent que l'équation $V(\hat{\sigma}) = c$ admet une unique solution $\hat{\sigma}$ pour toute valeur de c dans l'intervalle (contraintes d'arbitrage) $(S_t - Ke^{-r(T-t)})_+ < c < S_t$. Cette solution $\hat{\sigma}$ est appelée la *volatilité implicite*, et peut aisément être estimée numériquement. Quand on réalise cet exercice, on constate que :

- La volatilité implicite est toujours supérieure à la volatilité du sous-jacent.
- Les volatilités implicites de différentes options sur le même sous-jacent dépendent de leur strike et maturité.
- Pour presque tous les strikes, la volatilité implicite décroît en fonction du strike (*skew*).
- Pour de très grands strikes, on observe parfois une légère remontée de la volatilité implicite (*smile*).
- Les phénomènes de *skew* et de *smile* sont le plus prononcés pour les options de courte maturité ; la courbe de volatilité implicite en fonction du strike s'aplatit pour les grandes maturités.

Ces différences entre la volatilité implicite et la volatilité historique du sous-jacent s'expliquent notamment par les imperfections de marché (coûts de transaction, incertitude sur la volatilité) qui rendent en pratique la couverture des options plus coûteuse que ne le prévoit la théorie.

Étant donné un prix d'option observé sur le marché V^{mkt}, la volatilité implicite de Black-Scholes σ^* peut être déterminée en résolvant l'équation :

$$BS(\sigma^*; S, K, \tau, r) = V^{\text{mkt}}.$$

La monotonie de l'équation de Black-Scholes par rapport à la volatilité garantit l'existence de $\sigma^* \in [0, +\infty]$. On peut exprimer la volatilité implicite sous la forme implicite suivante :

$$\sigma^*(K, T) = BS^{-1}(V^{\text{mkt}}; S, K, \tau, r), \tag{1}$$

où BS^{-1} désigne la fonction inverse de Black-Scholes. De plus, en adoptant le ratio de moneyness, $m = \frac{S_t}{K}$, et le temps jusqu'à l'échéance, $\tau = T - t$, on peut exprimer la volatilité implicite sous la forme $\sigma^*(m, \tau)$ [2].

Pour simplifier, nous notons ici $\sigma^*(m, \tau)$ par σ^*. Une solution analytique pour l'Équation (1) n'existe pas. La valeur de σ^* est déterminée à l'aide d'une méthode itérative numérique (en particulier la méthode de Brent dans notre cas), puisque l'Équation (1) peut être reformulée comme un problème de recherche de racine :

$$g(\sigma^*) = BS(S, \tau, K, r, \sigma^*) - V^{\text{mkt}}(S, \tau; K) = 0.$$

2.3 Méthode de Brent

La méthode de Brent[3] est une méthode numérique efficace utilisée pour résoudre des équations non linéaires $f(x) = 0$. Elle combine les avantages de la méthode de dichotomie, de la méthode de la sécante et de la méthode d'interpolation parabolique inverse, offrant une convergence rapide tout en maintenant une grande robustesse. Son principe de base consiste à utiliser d'abord des méthodes d'interpolation rapides (méthode de la sécante ou interpolation parabolique inverse) pour tenter de trouver la racine. En cas d'échec, elle revient à la méthode robuste de dichotomie pour garantir la convergence.

Formules de la méthode :

1. **Méthode de dichotomie** : Réduit l'intervalle de recherche $[a, b]$ en prenant son point milieu :

$$c = \frac{a + b}{2}$$

2. **Méthode de la sécante** : Trace une droite passant par les points $(a, f(a))$ et $(b, f(b))$ et utilise l'intersection de cette droite avec l'axe des x comme nouvelle approximation de la racine :

$$x = b - \frac{f(b) \cdot (b - a)}{f(b) - f(a)}$$

3. **Interpolation parabolique inverse** : Ajuste une parabole passant par trois points $(x_1, f(x_1)), (x_2, f(x_2)), (x_3, f(x_3))$ et calcule son intersection avec l'axe des x :

$$x = x_1 - \frac{(x_2 - x_1)^2 f(x_1)}{(x_2 - x_1)f(x_1) - (x_2 - x_3)f(x_3)}$$

La méthode de Brent sélectionne dynamiquement l'une de ces approches à chaque itération, permettant de se rapprocher rapidement de la solution tout en garantissant une convergence sûre même si les interpolations échouent.

2.4 Modèle de Heston

Le modèle de Heston est décrit par un processus stochastique bivarié pour le prix de l'action $(S_t)_{t\geq 0}$ et sa variance v_t, qui suit le modèle de Cox-Ingersoll-Ross (CIR) :

$$dS_t = rS_t\,dt + \sqrt{v_t}S_t\,dW_t^1$$
$$dv_t = \kappa(\theta - v_t)\,dt + \gamma\sqrt{v_t}\,dW_t^2$$

où :

- r : taux d'intérêt neutre au risque,
- θ : moyenne de long terme de la variance,
- κ : taux auquel la variance revient vers θ,
- γ : volatilité de la volatilité,
- ρ : corrélation entre les deux mouvements browniens, telle que $dW_t^1 dW_t^2 = \rho dt$.

La volatilité est toujours positive et ne peut pas être nulle ou négative si la condition de Feller $2\kappa\theta > \gamma^2$ est satisfaite. Le prix de l'option augmente si θ augmente. Les paramètres ρ et γ influencent respectivement l'asymétrie (skewness) et la kurtosis de la distribution des rendements.

Le modèle de Heston appartient à la famille des processus des diffusion dits affines. Cette famille de processus partage la propriété de posséder une fonction caractéristique explicite, ce qui permet notamment d'obtenir la méthode de COS qui utilise les développements en série de Fourier-cosinus pour approximer la fonction de densité de probabilité du prix de l'actif facilitant ainsi la valorisation des options européennes. Reconnue pour sa précision remarquable et son efficacité élevée, la méthode COS a été privilégiée et appliquée par les auteurs de l'article original[1] pour calculer les valeurs des options européennes dans le cadre du modèle de Heston. Cependant, pour de nombreux modèles modernes d'actifs, la fonction caractéristique n'est généralement pas disponible, ce qui signifie que la méthode COS ne présente pas une grande valeur d'application généralisée.

Ainsi, nous avons choisi d'utiliser la méthode de Monte-Carlo dans ce projet, plutôt que la méthode COS décrite dans l'article original. Cette approche est non seulement plus flexible, mais elle correspond également mieux au contenu de notre cours (Méthodes avancés en Probabilités numériques).

2.5 Méthode de Monte Carlo

La méthode de Monte Carlo est une approche de calcul numérique fondée sur l'échantillonnage aléatoire, largement employée dans la simulation, l'intégration et les problèmes d'optimisation. Son principe repose sur l'exploitation des propriétés statistiques d'un grand nombre d'échantillons aléatoires pour estimer des solutions à des problèmes complexes. Dans le domaine financier, les méthodes de Monte Carlo sont particulièrement utilisées pour la valorisation des options path-dépendantes à haute dimension, ainsi que pour le calcul de divers ajustements de valorisation en gestion des risques moderne. Cependant, ces méthodes souffrent généralement d'une convergence relativement lente, ce qui peut poser problème, notamment dans le cadre de la calibration des modèles.

Méthode de Monte Carlo classique:

- Échantillonnage aléatoire : Génération d'échantillons aléatoires dans un espace probabiliste donné.
- Estimation statistique : Utilisation des propriétés statistiques, telles que la moyenne ou la variance des échantillons, pour estimer la valeur espérée ou l'intégrale d'une fonction cible.

  - Pour calculer une intégrale $\int_a^b f(x)dx$, l'estimation avec N échantillons aléatoires $\{x_i\}_{i=1}^N$ est donnée par :

$$\int_a^b f(x)dx \approx \frac{b-a}{N}\sum_{i=1}^{N} f(x_i)$$

- Convergence : Selon la loi des grands nombres, lorsque $N \to \infty$, l'estimation converge vers la valeur réelle.

Méthode de Quasi-Monte Carlo :

La méthode de Quasi-Monte Carlo est une amélioration de la méthode de Monte Carlo qui utilise des séquences à faible discrépance (comme les séquences de Sobol ou de Halton) pour augmenter la vitesse de convergence lors de l'intégration ou des simulations.

- Séquences à faible discrépance : La méthode de Quasi-Monte Carlo remplace les échantillons aléatoires par des points répartis de manière plus uniforme.

  - Les séquences couramment utilisées incluent les séquences de Sobol et de Halton.

  - Ces séquences réduisent l'effet de regroupement des échantillons aléatoires et couvrent mieux l'espace des échantillons.

- Estimation d'intégrale : La formule d'estimation d'intégrale reste la même que pour la méthode de Monte Carlo :

$$\int_a^b f(x)dx \approx \frac{b-a}{N}\sum_{i=1}^{N} f(x_i)$$

La différence réside dans le fait que les x_i sont des points des séquences à faible discrépance au lieu de points aléatoires.

Dans notre projet, nous avons adopté une approche mixte combinant la méthode classique de Monte-Carlo (MC) et la méthode Quasi-Monte Carlo (QMC). Plus précisément, nous avons utilisé la méthode QMC basée sur le Latin Hypercube Sampling (LHS, section 3.1) pour générer les paramètres du modèle de Heston, puis appliqué la méthode classique de Monte-Carlo pour simuler les trajectoires du modèle de Heston. Enfin, cette approche nous a permis de produire un ensemble de données de prix d'options européennes basé sur le modèle de Heston.

3 Réseau de Neurones Artificiel (ANN)

Dans cette section, nous détaillons les architectures des réseaux de neurones utilisées pour deux objectifs principaux : l'évaluation des options d'achat (BS-ANN, Heston-ANN) et le calcul des volatilités implicites (IV-ANN). Une fois les données de marché des prix d'options générées, les réseaux sont calibrés en minimisant un critère d'erreur.

Pour garantir un entraînement précis du modèle, nous décrivons la méthodologie employée pour déterminer les hyperparamètres optimaux, tels que la fonction d'activation, le nombre de nœuds et de couches. Nous justifions également le choix du taux d'apprentissage optimal, essentiel pour une convergence rapide et efficace.

3.1 Génération de Données

La performance d'un réseau de neurones dépend étroitement de la qualité du jeu de données utilisé. Pour générer un jeu de données pertinent, les auteurs ont recours à la technique d'échantillonnage **Latin Hypercube Sampling (LHS)**. Cette méthode, une généralisation du carré latin, garantit qu'un seul échantillon est sélectionné dans chaque ligne et chaque colonne. Elle produit des échantillons aléatoires issus d'une distribution multidimensionnelle, assurant une représentation cohérente des données. L'échantillonnage de n points pour une variable aléatoire consiste à diviser son support en n intervalles et à tirer un échantillon dans chacun d'eux.

Dans ce contexte, la méthode LHS est utilisée pour échantillonner les intervalles des paramètres des modèles, qu'il s'agisse du modèle de Heston ou du modèle de Black-Scholes. Pour chaque échantillon généré, le prix correspondant est calculé soit à l'aide de la formule explicite, soit par la méthode de Monte Carlo. Les caractéristiques spécifiques de chaque jeu de données seront détaillées plus en profondeur dans le Chapitre 5.

Parameters	Options or Range
Activation	ReLU, tanh, sigmoid, elu
Dropout rate	[0.0, 0.2]
Neurons	[200, 600]
Initialization	uniform, glorot_uniform, he_uniform
Batch normalization	yes, no
Optimizer	SGD, RMSprop, Adam
Batch size	[256, 3000]

Table 1: La configuration de la recherche aléatoire sur des hyperparamètres

3.2 Structure du ANN

L'auteur de [4] nous indique que la méthode de recherche aléatoire (random search) est une approche très efficace pour l'optimisation des hyperparamètres. Les auteurs de l'article original [1] ont appliqué cette méthode aux différents types ou intervalles d'hyperparamètres présentés dans le Table 1. Cela leur a permis d'obtenir les résultats optimaux pour chaque hyperparamètre, récapitulés dans le Table 2. Dans notre projet, nous utilisons directement les résultats présentés dans le Table 2 comme hyperparamètres pour le réseau de neurones artificiels (ANN).

Parameters	Options
Hidden layers	4
Neurons (each layer)	400
Activation	ReLU
Dropout rate	0.0
Batch-normalization	No
Initialization	Glorot_uniform
Optimizer	Adam
Batch size	1024

Table 2: La modèle sélectionné après la recherche aléatoire

La structure du réseau de neurones artificiel (ANN) est illustrée à la Figure 1, qui montre quatre couches cachées, chacune contenant $n = 400$ neurones.

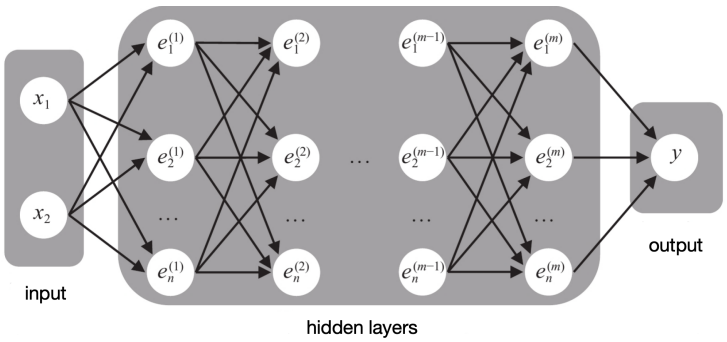

Figure 1: Un exemple d'ANN

Les valeurs des nœuds sont calculées comme suit :

$$e_j^{(l)} = \max\left(\sum_i w_{ij}^{(l)} e_i^{(l-1)} + b_j^{(l)}, 0\right), \quad \forall l \in \{1, 2, \ldots, L\},$$

où :

- $e_j^{(l)}$: valeur du j-ième nœud dans la l-ième couche,
- $e_i^{(l-1)}$: valeur de sortie du i-ième nœud dans la $(l-1)$-ième couche,
- $w_{ij}^{(l)}$: poids reliant le i-ième nœud de la $(l-1)$-ième couche au j-ième nœud de la l-ième couche,
- $b_j^{(l)}$: biais du j-ième nœud dans la l-ième couche.

Pour un vecteur d'entrée $x \in R^d$, la fonction de prédiction du réseau s'écrit :

$$y(x) = F(x, (W, b)).$$

Nous pouvons maintenant définir la fonction de perte de réseau de neurones artificiel :

$$L(\mathbf{W}, b) := D\big(f(x), F\big(x \mid \mathbf{W}, b\big)\big),$$

où f est une fonction objective. Le processus de l'entrâinement peut être formulé comme le problème d'optimisation suivant :

$$\arg\min_{(\mathbf{W},b)} \; L\big(\mathbf{W}, b \;\big|\; (x, y)\big).$$

3.3 Préparation de L'entraînement des ANNs

3.3.1 Déterminer le taux d'apprentissage optimal

Estimer un taux d'apprentissage optimal, c'est-à-dire la vitesse à laquelle les poids sont mis à jour pendant la phase d'entraînement, est une étape importante lors de la configuration d'un réseau de neurones. En effet, un taux d'apprentissage trop élevé peut entraîner une divergence, tandis qu'un taux trop faible peut ralentir la convergence.

Les auteurs de l'article original [1] ont utilisé la méthode proposée dans [5] pour déterminer l'intervalle optimal du taux d'apprentissage. Nous avons reproduit cette méthode afin de déterminer le taux d'apprentissage optimal adapté à notre projet. Cette méthode consiste à augmenter progressivement le taux d'apprentissage pendant l'entraînement et à observer son effet sur la perte moyenne. Comme illustré à la Figure 3, la perte se stabilise au début, commence à diminuer rapidement autour de 10^{-6}, puis oscille après 10^{-4}. Ainsi, le meilleur intervalle pour le taux d'apprentissage se situe entre 10^{-6} et 10^{-4}.

Figure 2: Average training loss selon learning rate

3.3.2 Comparaison des différents calendriers de taux d'apprentissage.

Comme l'illustre la figure 3, nous avons comparé l'historique des pertes lors de l'entraînement du réseau de neurones de modèle de Black-Scholes (BS ANN) en utilisant différents calendriers de taux d'apprentissage (DecayLR, CyclicalLR, ConstantLR). Il apparaît clairement que DecayLR offre les meilleures performances d'entraînement. Par conséquent, pour l'ensemble des entraînements de réseaux de neurones de notre projet, nous avons choisi d'appliquer DecayLR.

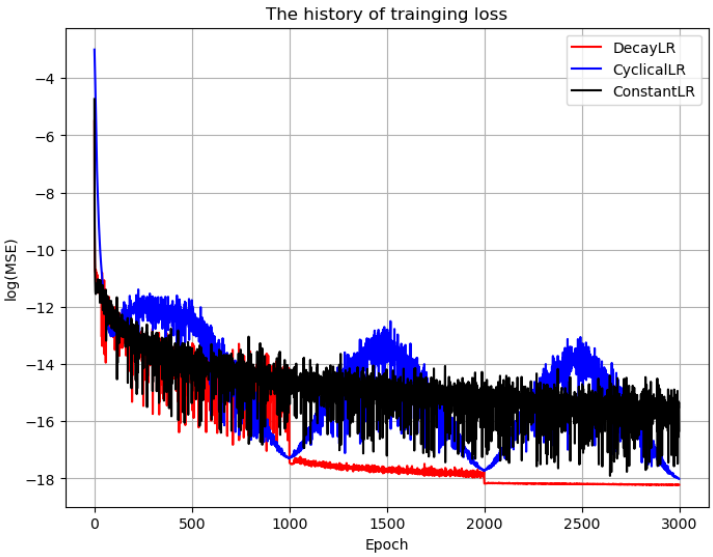

Figure 3: Différents calendriers de taux d'apprentissage pour l'entrainement de BS ANN

4 Méthodologie expérimentale

Les expériences menées par les auteurs visent à démontrer la capacité du réseau de neurones à évaluer les options d'achat vanilles et à calculer leurs volatilités implicites. Les expériences suivantes ont été réalisées :

1. Entraîner un réseau de neurones pour évaluer les options d'achat vanilles lorsque le sous-jacent suit le modèle de Black-Scholes. Les paramètres sont échantillonnés à l'aide de la méthode Latin Hypercube Sampling (LHS), et les prix des options sont calculés en utilisant la formule explicite (fermée) de Black-Scholes.
2. Entraîner un réseau de neurones pour calculer la volatilité implicite des options d'achat vanilles. Les paramètres sont échantillonnés à l'aide de la méthode LHS, et les données sont générées en inversant le prix de l'option et sa volatilité à partir du jeu de données précédemment généré (encore à l'aide du modèle de Black-Scholes).
3. Entraîner un réseau de neurones pour évaluer les options d'achat vanilles lorsque le sous-jacent suit le modèle de Heston. Les paramètres sont échantillonnés à l'aide de la méthode LHS, et les prix des options sont calculés en utilisant la méthode de Monte Carlo.
4. Comparer directement l'efficacité de calcul de la volatilité implicite entre l'IV ANN obtenu en 2. et la méthode de Brent. Ensuite, nous étendons cette comparaison en évaluant l'efficacité et la précision du calcul de la volatilité implicite en combinant le Heston ANN (obtenu en 3.) avec l'IV ANN, par rapport à une combinaison basée sur Monte Carlo et la méthode de Brent.

Afin d'évaluer la performance de chaque ANN pour les prix des options et le calcul des volatilités implicites, les métriques résumées dans le Table 3 sont utilisées.

Métrique d'erreur	**Formule**
Erreur Quadratique Moyenne (MSE)	$\frac{1}{N}\sum_{k=1}^{N}(y_k - \hat{y}_k)^2$
Racine de l'Erreur Quadratique Moyenne (RMSE)	$\sqrt{\text{MSE}} = \sqrt{\frac{1}{N}\sum_{k=1}^{N}(y_k - \hat{y}_k)^2}$
Erreur Absolue Moyenne (MAE)	$\frac{1}{N}\sum_{k=1}^{N}\lvert y_k - \hat{y}_k\rvert$
Erreur Absolue Moyenne en Pourcentage (MAPE)	$\frac{1}{N}\sum_{k=1}^{N}\frac{\lvert y_k - \hat{y}_k\rvert}{\lvert y_k\rvert}$
Coefficient de Détermination (R^2)	$1 - \frac{\sum_{k=1}^{N}(y_k - \hat{y}_k)^2}{\sum_{k=1}^{N}(y_k - \bar{y})^2}$

Table 3: Métriques d'erreur

La section suivante présente les résultats numériques ainsi qu'une analyse approfondie de ces expériences.

5 Résultats Numériques

Dans cette section, nous présentons les résultats numériques des différents réseaux de neurones en nous basant sur les métriques d'erreur introduites précédemment.

5.1 BS ANN

Les données d'entraînement pour le réseau de neurones basé sur le modèle de Black-Scholes (BS-ANN) sont détaillées dans le Table 4. Le BS-ANN prend en entrée les variables suivantes : $\{S_0/K, \tau, r, \sigma\}$. La sortie est le prix de l'option normalisé $\{V/K\}$, calculé à l'aide de la formule de Black-Scholes. Le modèle minimise la fonction de perte d'erreur quadratique moyenne (MSE). L'optimiseur Adam de PyTorch est utilisé pour mettre à jour les poids, avec un planificateur démarrant à 10^{-4} et diminuant tous les 1000 epochs jusqu'à atteindre 10^{-6}. L'entraînement est effectué sur 3000 epochs.

Les auteurs distinguent deux types de jeux de données : large plage et plage étroite. L'entraînement est réalisé séparément sur ces deux jeux de données afin d'observer si le réseau fonctionne mieux lorsque les données d'entrée ne se situent pas aux limites du domaine des entrées.

Paramètres	**Plage Large**	**Plage Étroite**	**Unité**
Prix de l'actif (S_0/K)	[0.4, 1.6]	[0.5, 1.5]	-
Temps à l'échéance (τ)	[0.2, 1.1]	[0.3, 0.95]	année
Taux sans risque (r)	[0.02, 0.1]	[0.03, 0.08]	-
Volatilité (σ)	[0.01, 1.0]	[0.02, 0.9]	-
Prix d'achat (V/K)	(0.0, 0.89)	(0.0, 0.73)	-

Table 4: Plages des paramètres pour Black-Scholes (large et étroite)

Les performances du BS-ANN sont présentées dans le Table 5. La deuxième ligne contient les résultats des auteurs, tandis que la troisième ligne présente nos résultats reproduits.

L'entraînement du réseau n'a été effectué qu'avec 10^5 échantillons en raison des restrictions des ressources. En revanche, les auteurs ont entraîné leur réseau avec 10^6 échantillons. En gardant cela à l'esprit, le Table 5 montre que les métriques d'erreur que nous avons obtenues ne diffèrent que d'un facteur de 10^{-1} au maximum.

Nous avons ainsi réussi à reproduire les résultats des auteurs. De plus, nous observons d'abord que le BS-ANN ne souffre d'aucun sur-apprentissage, car les métriques de test et d'entraînement sont globalement similaires. Ensuite, nous constatons que le réseau fonctionne légèrement mieux sur le jeu de données étroit, ce qui montre que l'entraînement sur un jeu de données plus large n'est pas une mauvaise pratique. Le RMSE en test est d'environ 1×10^{-4}, ce qui signifie que l'erreur moyenne de tarification est d'environ 0.01%. Dans l'article original, les auteurs n'ont pas présenté le R^2 des résultats expérimentaux, car ils considéraient qu'il n'y avait pas de différence significative. Nos résultats de reproduction confirment précisément cette observation.

BS-ANN	MSE	RMSE	MAE	MAPE	R^2
Article					
Entraînement-large	8.04×10^{-9}	8.97×10^{-5}	6.73×10^{-4}	3.75×10^{-4}	–
Test-large	8.21×10^{-9}	9.06×10^{-5}	6.79×10^{-4}	3.79×10^{-4}	–
Test-étroit	7.00×10^{-9}	8.37×10^{-5}	6.49×10^{-4}	3.75×10^{-4}	–
Projet					
Entraînement-large	1.20×10^{-8}	1.09×10^{-4}	8.13×10^{-5}	1.26×10^{-1}	0.9999998
Test-large	1.43×10^{-8}	1.20×10^{-4}	8.81×10^{-5}	1.07×10^{-1}	0.9999997
Test-étroit	9.28×10^{-9}	9.64×10^{-5}	7.16×10^{-5}	8.82×10^{-2}	0.9999997

Table 5: Performances du BS-ANN sur le jeu de données de test

La Figure 4 montre que les histogrammes des erreurs de prédiction pour les jeux de données larges et étroits suivent une distribution normale. L'erreur absolue maximale est respectivement de 8×10^{-4} et 7×10^{-4}.

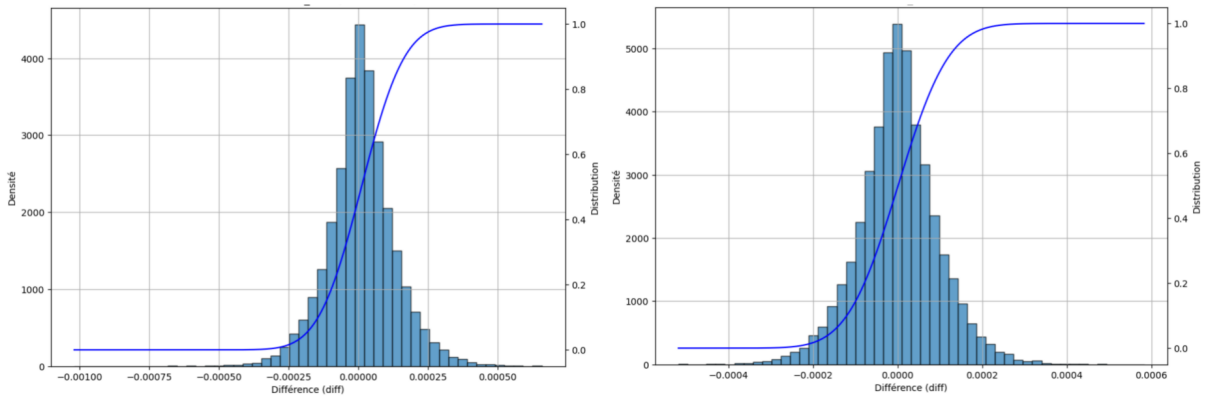

Figure 4: La distribution des erreurs. (Droite : Performance de ANN sur le jeu de donnée étroite ; Gauche : Performance de ANN sur le jeu de donnée large.)

5.2 IV ANN

Le réseau de volatilité implicite (IV-ANN) vise à apprendre la relation implicite entre le prix de l'option et sa volatilité implicite. Les entrées du réseau de neurones sont détaillées dans le Table 6. Plutôt que d'utiliser directement la valeur normalisée de l'option $\{V/K\}$, les auteurs alimentent le réseau de neurones avec le logarithme de la valeur temps de l'option $\{\log(V_t/K)\}$. La valeur temps est définie comme suit :

$$\tilde{V} = V_t - \max(S_t - Ke^{-r\tau}, 0),$$

où V_t est le prix de l'option et τ représente le temps à l'échéance. Cette transformation logarithmique est réalisée pour éviter le problème de gradient abrupt lors de l'entraînement de l'IV-ANN, qui pourrait conduire à une performance médiocre. De plus, les petites valeurs de $\tilde{V}$ (par exemple, $\tilde{V} < 10^{-7}$) sont exclues avant l'application du logarithme.

Le Table 6 résume les plages des entrées du réseau.

Parameter	Range	Unit
Prix de l'actif (S_0/K)	[0.5, 1.4]	-
Temps à l'échéance (τ)	[0.05, 1.0]	year
Taux sans risque (r)	[0.0, 0.1]	-
log(Valeur d'achat) ($\log(V_t/K)$)	$[-16.12, -0.94]$	-
Volatilité output (σ)	(0.05, 1.0)	-

Table 6: Plage des paramètres du jeu de données IV-ANN

La performance du réseau IV-ANN sur le jeu de données de test est détaillée dans le Table 7. La deuxième ligne présente les résultats des auteurs, tandis que la troisième ligne présente nos résultats. Il est montré que les métriques d'erreur que nous avons obtenues diffèrent uniquement d'un facteur de 10^{-2} au maximum. Nous observons que la transformation logarithmique améliore considérablement les performances du réseau de neurones.

La Figure 5 montre que les histogrammes des erreurs de prédiction pour le jeu de données de test suivent une distribution normale et que l'erreur absolue maximale est de 6×10^{-3}. Le réseau IV-ANN a réussi à capturer la relation entre les prix des options et la volatilité implicite.

IV-ANN	MSE	MAE	MAPE	R^2
Article				
Standard	6.36×10^{-4}	1.24×10^{-2}	7.67×10^{-2}	0.97510
Log transformé	1.55×10^{-8}	9.73×10^{-5}	2.11×10^{-3}	0.9999998
Projet				
Standard	6.06×10^{-4}	8.03×10^{-3}	3.87×10^{-2}	0.9919511
Log transformé	6.40×10^{-7}	5.87×10^{-4}	1.56×10^{-3}	0.9999908

Table 7: Performances du IV-ANN sur le jeu de données de test

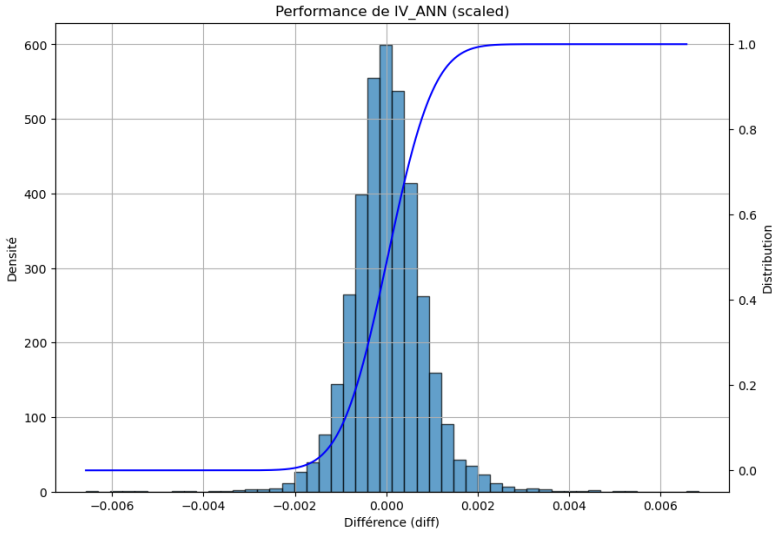

Figure 5: La distribution des erreurs. (Performance du IV-ANN sur les entrées avec le logarithme).

5.3 Heston ANN

Les données d'entraînement pour le réseau de neurones basé sur le modèle de Heston (Heston-ANN) sont détaillées dans le Table 8. Les entrées du réseau de neurones sont les paramètres de l'EDS de Heston et les caractéristiques des options d'achat : $(r, \rho, \kappa, \bar{\nu}, \gamma, \nu_0, \tau, S_0/K)$, et la sortie est le prix de l'option normalisé $\{V/K\}$. Les prix des options normalisés sont calculés à l'aide de la méthode de Monte Carlo.

L'entraînement du réseau est effectué sur 3000 epochs, en commençant par un taux d'apprentissage de 10^{-4} qui décroît jusqu'à 10^{-6} tous les 1000 epochs.

ANN	**Paramètres**	**Plage**	**Méthode**
NN Input	Monneyage, $m = S_0/K$	(0.6, 1.4)	LHS
	Temps à l'échéance, τ	(0.1, 1.4) (année)	LHS
	Taux sans risque, r	(0.0%, 10%)	LHS
	Corrélation, ρ	$(-0.95, 0.0)$	LHS
	Vitesse de réversion, κ	(0.0, 2.0)	LHS
	Variance moyenne long-terme, $\bar{\nu}$	(0.0, 0.5)	LHS
	Volatilité de la volatilité, γ	(0.0, 0.5)	LHS
	Variance initiale, ν_0	(0.05, 0.5)	LHS
NN Output	Prix d'achat européen, V	(0, 0.67)	MC

Table 8: Plage des paramètres de Heston pour l'entraînement du ANN

Les performances du Heston-ANN à l'aide de la méthode de Monte Carlo sont détaillées dans le Table 9. En réalité, les résultats obtenus diffèrent quelque peu de ceux présentés par les auteurs dans l'article original utilisant la méthode COS. En raison des contraintes de temps et de coût de calcul, nous avons utilisé la méthode de Monte Carlo avec seulement 10 000 trajectoires simulées. Par conséquent, la précision de nos résultats est inférieure à celle obtenue avec la méthode COS dans l'article original.

Cependant, nous avons constaté qu'à mesure que le nombre de trajectoires simulées par la méthode de Monte Carlo augmente, l'erreur diminue rapidement. Ainsi, nous croyons qu'avec un temps et des ressources de calcul suffisants, la méthode de Monte Carlo ne serait pas moins précise que la méthode COS.

La Figure 6 montre que les histogrammes des erreurs de prédiction pour le jeu de données de test suivent une distribution normale et que l'erreur absolue maximale est de 3×10^{-2}.

Heston-ANN	**MSE**	**MAE**	**MAPE**	R^2
Entraînement	8.98×10^{-6}	2.12×10^{-3}	4.05×10^{-2}	0.9995891
Test	1.82×10^{-5}	2.92×10^{-3}	4.03×10^{-2}	0.9991755

Table 9: Performances du Heston-ANN

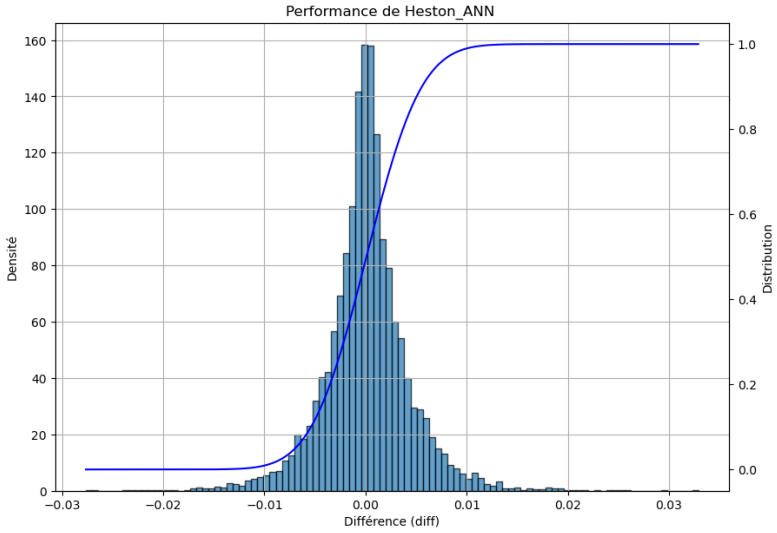

Figure 6: La distribution des erreurs. (Performance du Heston-ANN sur les entrées normalisées).

5.4 Approches ANNs vs Approches numérique

L'expérience finale réalisée par les auteurs démontre la capacité des réseaux de neurones entraînés à évaluer les options d'achat vanilles et à récupérer leur volatilité implicite. L'approche basée sur les réseaux de neurones est comparée à l'approche numérique classique, qui utilise les méthodes de Monte Carlo et de Brent.

5.4.1 Modèle de Black-Scholes et Volatilité implicite

Dans notre projet, nous avons tout d'abord réalisé une comparaison simple. En générant 10^6 échantillons basés directement sur le modèle de Black-Scholes , nous avons calculé la volatilité implicite en utilisant à la fois IV-ANN et la méthode de Brent. Ensuite, nous avons mesuré directement le temps de calcul des deux méthodes sur GPU et CPU, afin de comparer leur efficacité et d'observer la grande efficacité d'IV-ANN dans la Table 10.

Table 10: Performance comparison: CPU (Intel(R) Core(TM) i9-14900HX) and GPU (NVIDIA GeForce RTX 4070 Laptop GPU)

Method	**GPU (sec)**	**CPU (sec)**	**Robustness**
Brent	129.84	131.97	Yes
IV-ANN	0.45	1.77	Yes

5.4.2 Modèle de Heston et Volatilité implicite

Nous avons conçu deux expériences pour démontrer la capacité des réseaux de neurones artificiels (ANN) à calculer la volatilité implicite à partir des prix d'options générés selon

le modèle de Heston. Dans la première expérience, la volatilité implicite de référence est obtenue en deux étapes : à partir des paramètres du modèle de Heston, les prix d'options sont d'abord calculés à l'aide de la méthode de Monte Carlo, puis la méthode de Brent est utilisée pour déterminer la volatilité implicite.

L'approche basée sur l'apprentissage automatique suit également deux étapes. D'abord, le réseau Heston-ANN est utilisé pour prédire les prix des options, puis IV-ANN calcule les volatilités implicites correspondantes. Nous comparons ensuite ces deux approches selon les critères suivants.

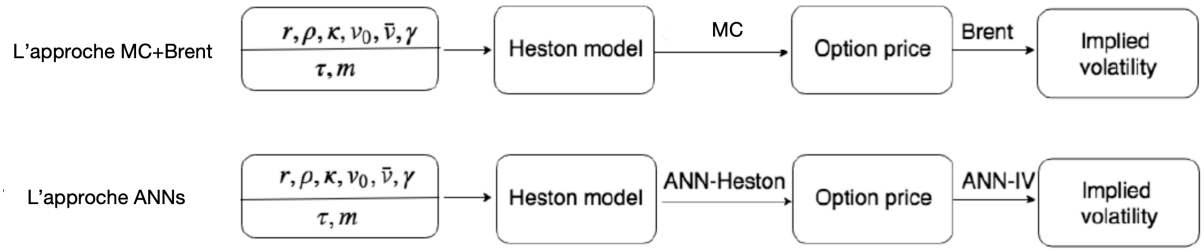

Figure 7: Deux approches pour le calcul de la volatilité implicite sous modèle de Heston

Le Table 11 présente les performances hors échantillon des réseaux Heston-ANN et IV-ANN pour notre cas étudié. Pour cela, nous avons appliqué directement le jeu de données généré dans la section 5.3 aux deux approches afin d'obtenir la volatilité implicite issue de la méthode numérique ainsi que celle calculée par les réseaux de neurones (ANNs). La volatilité implicite issue de la méthode numérique est considérée comme la valeur de référence théorique pour évaluer les performances de l'approche basée sur les ANNs.

Heston-ANN & IV-ANN	**MSE**	**MAE**	**MAPE**	$\mathbf{R^2}$
Résultats :	2.22×10^{-4}	9.13×10^{-3}	1.85×10^{-2}	0.9774944

Table 11: Performances hors échantillon des Heston-ANN et IV-ANN

Comme mentionné au début de ce rapport et dans la section 5.3, nous avons choisi d'utiliser la méthode de Monte-Carlo plutôt que la méthode COS pour résoudre le problème de tarification dans le modèle de Heston, en raison de certains avantages spécifiques de la méthode Monte-Carlo. Cependant, nous avons également pris en compte les limites mentionnées dans l'article original, telles que la lenteur de convergence, les exigences élevées en termes de ressources de calcul, ainsi que l'incapacité à réaliser une calibration parfaite.

Nos expériences ont confirmé ces observations. Lors de nos premiers essais, nous avons utilisé 1 000 trajectoires simulées avec la méthode MC, ce qui a abouti à un R^2 d'environ 0,86. En augmentant le nombre de trajectoires simulées à 10 000, nous avons finalement obtenu un R^2 d'environ 0,97, comme illustré dans la figure 8. Ces résultats nous permettent de conclure qu'avec suffisamment de temps et de ressources de calcul, il est tout à fait possible d'atteindre une précision proche de celle obtenue dans l'article original pour le calcul de la volatilité implicite.

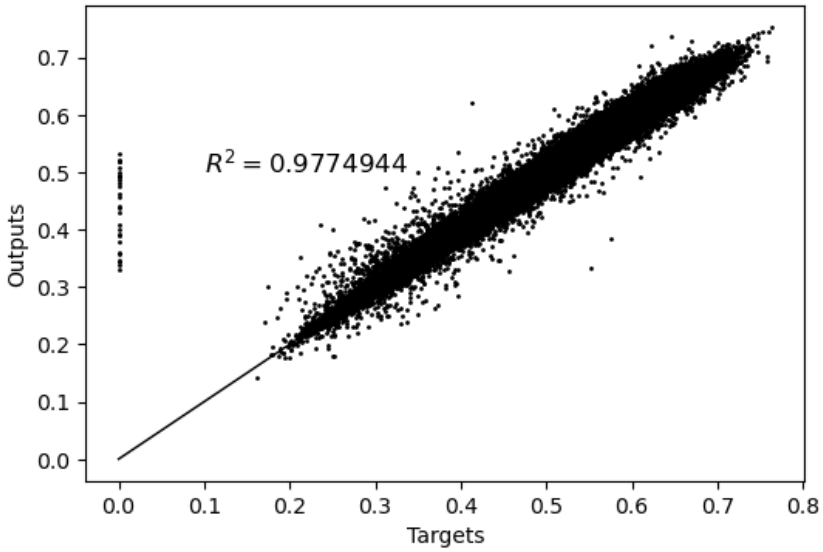

Figure 8: Comparaison de Volatilité implicite (l'approche MC+Brent VS l'approche ANNs)

Pour visualiser les capacités de l'IV-ANN à capturer la relation implicite entre les prix des options et leurs volatilités implicites, on a affiché la surface de volatilité en faisant varier $m \in [0.7, 1.3]$ et $\tau \in [0.5, 1.0]$, tout en fixant les paramètres de Heston à $\rho = -0.05$, $\kappa = 1.5$, $\gamma = 0.3$, $\bar{\nu} = 0.1$, $\nu_0 = 0.1$ et $r = 0.02$.

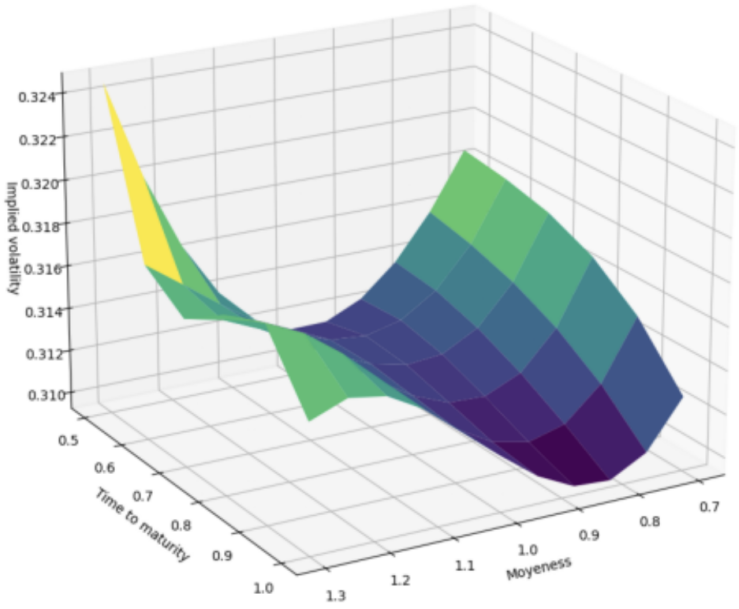

Figure 9: Surface de volatilité implicite de Heston générée par l'IV-ANN.

La Figure 9 montre la surface de volatilité obtenue par l'IV-ANN en utilisant une interpolation de surface par B-spline. Dans la pratique , il y a d'autre facons de calibration au delà de la simple interpolation[6]. Nous pouvons clairement observer que cette surface de

volatilité implicite possède des propriétés cohérentes avec celles de la surface de volatilité implicite basée sur le modèle de Heston, ce qui permet de démontrer la validité de nos expériences.

6 Conclusion

L'objectif de ce projet était de reproduire les expériences des auteurs visant à démontrer les capacités des réseaux de neurones à évaluer les options d'achat vanilles et à calculer leurs volatilités implicites. L'entraînement du réseau de neurones nécessitait une technique efficace de remplissage de l'espace. La méthode LHS a donc été utilisée pour générer un jeu de données d'entraînement viable. Les prix des options ont été calculés à l'aide de la formule explicite de Black-Scholes et de la méthode de Monte Carlo. Les volatilités implicites ont été calculées à l'aide de la méthode de Brent. Nous avons réussi à reproduire les expériences menées sur les réseaux BS-ANN, IV-ANN et Heston-ANN. Enfin, nous avons démontré qu'une approche basée uniquement sur les réseaux de neurones constitue une alternative viable à une approche numérique traditionnelle, tant pour la tarification que pour la génération de surfaces de volatilité. L'entraînement peut être réalisé sur une plage de paramètres plus large, et les ANNs offrent un moyen rapide d'exécuter des calculs numériques, bien que l'entraînement nécessite un temps plus long.

References

[1] Shuaiqiang Liu , Cornelis W. Oosterlee and Sander M.Bohte, *"Pricing options and computing implied volatilities using neural networks,"* 2018.

[2] Cont, R.; da Fonseca, J., *"Dynamics of implied volatility surfaces. ,"* 2002.

[3] P., B.R., *"Algorithms for Minimization without Derivatives,"* 1973.

[4] Bergstra, J.; Bengio, Y., *Random Search for Hyper-Parameter Optimization.* 2012.

[5] L. Smith, *"Cyclical learning rates for training neural networks,"* 2015.

[6] Olivier Bardou, *"Polycopié Marché incomplet,"* 2024.