Pricing Options and Computing Implied Volatilities Using Neural Networks

Qiyao Zhou, Yibo Wang

M2 IFMA, Sorbonne Université

10 Janvier 2025

Introduction

Objectif du projet:

- Reproduire les résultats de l'article "Pricing Options and Computing Implied Volatilities Using Neural Networks".
- Comparer les méthodes classiques aux approches basées sur les réseaux de neurones artificiels (ANN).

Méthodologie:

- Générer des données avec les modèles de Black-Scholes et Heston.
- Entraîner et appliquer des ANNs pour évaluer les options et calculer les volatilités implicites.
- Comparaison des approches : 'MonteCarlo + Brent' VS 'Heston-ANN + IV-ANN'.

Modèle de Black-Scholes

Équation différentielle stochastique:

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

Prix de l'option d'achat européenne:

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

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

Paramètres d'entrés: r: Taux sans risque, σ: Volatilité, τ: Temps à l'échéance, $\frac{S_0}{K}$: Moneyness.
Paramètre de sortie: $\frac{V}{K}$: Prix de l'option.

Modèle de Heston

Processus stochastiques:

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

Paramètres d'entrés:

- κ: Vitesse de retour à la moyenne
- θ: Variance moyenne de long terme
- γ: Volatilité de la volatilité
- ρ: Corrélation entre les mouvements brownien W^1 et W^2
- $\frac{S_0}{K}$: Moneyness
- r: Taux sans risque
- τ: Temps à l'échéance
- v_0: Variance initiale

Paramètres de sortie: V : Prix de l'option

Volatilité Implicite

Propriétés:

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

Ces propriétés définissent le **Véga** de l'option et assurent l'existence et l'unicité de la volatilité implicite σ^*.

Formule de Black-Scholes:

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

Définition de la volatilité implicite:

- $\sigma^*(K, T) = BS^{-1}(V^{\text{mkt}}; S, K, \tau, r)$
- En adoptant le ratio de moneyness $m = \frac{S_t}{K}$, et $\tau = T - t$, on exprime σ^* sous la forme $\sigma^*(m, \tau)$.

Résolution:

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

Méthods numériques et Génération de données

Utilisation de la méthode de Brent pour estimer σ^*.

Méthode Monte Carlo:

- Utilisée pour simuler les trajectoires des modèles.
- Approche flexible mais convergence lente.

Méthode de Brent:

- Recherche de racines pour calculer les volatilités implicites.
- Combine dichotomie, sécante, et interpolation parabolique.

Génération de données:

- La technique d'échantillonnage Latin Hypercube Sampling (LHS).
- qmc.LatinHypercube(d = 4) (par exemple)

Objectifs des ANNs:

- Évaluer les options d'achat.
- Calculer les volatilités implicites.

Structure:

- 4 couches cachées, 400 neurones par couche.
- Fonction d'activation: ReLU.
- Optimiseur: Adam, taille de batch: 1024.

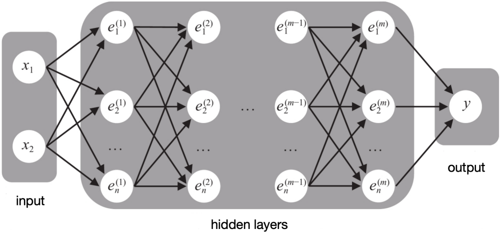


Figure: Un exemple d'ANN

Optimisation des hyperparamètres :

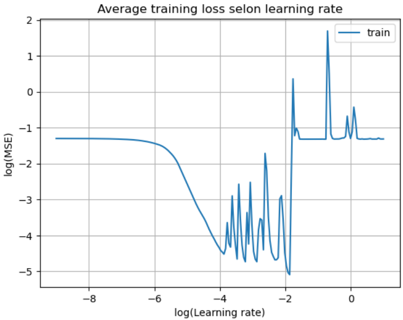
Parameters	**Options**
Hidden layers	4
Neurons (each layer)	400
Activation	ReLU
Dropout rate	0.0
Batch-normalization	No
Initialization	Glorot_uniform
Optimizer	Adam
Batch size	1024

Table: Le modèle sélectionné après recherche aléatoire

Préparation de l'entraînement des ANNs

Déterminer le taux d'apprentissage optimal

- Estimer un taux d'apprentissage optimal consiste à ajuster la vitesse à laquelle les poids sont mis à jour pendant la phase d'entraînement.
- Un taux trop élevé peut entraîner une divergence, tandis qu'un taux trop faible ralentit la convergence.
- L'intervalle optimal du taux d'apprentissage se situe entre 10^{-6} et 10^{-4}, comme illustré ci-dessous.



Comparaison des calendriers de taux d'apprentissage

- Nous avons comparé l'historique des pertes lors de l'entraînement du réseau de neurones de modèle Black-Scholes (BS ANN) avec différents calendriers : DecayLR, CyclicalLR et ConstantLR.
- DecayLR a offert les meilleures performances d'entraînement.
- Pour cette raison, DecayLR a été appliqué à l'ensemble des réseaux de neurones dans notre projet.

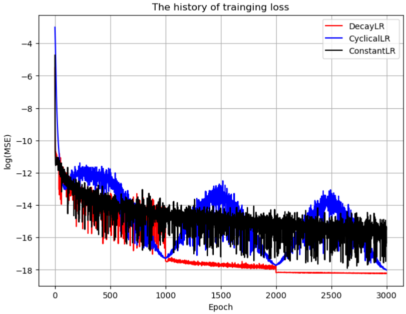
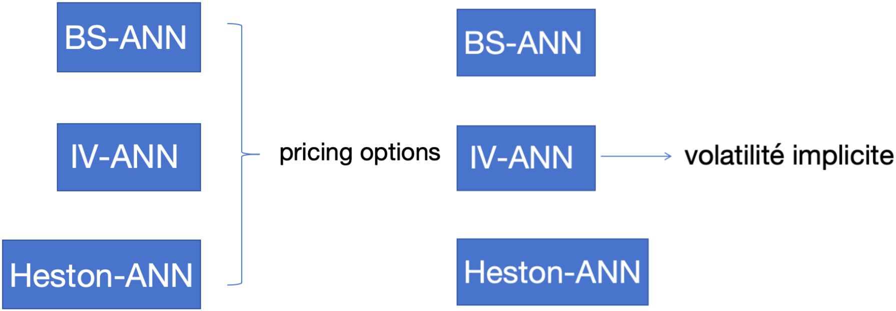


Figure: Figure: Average training loss selon learning rate

Résultats Numériques



BS-ANN

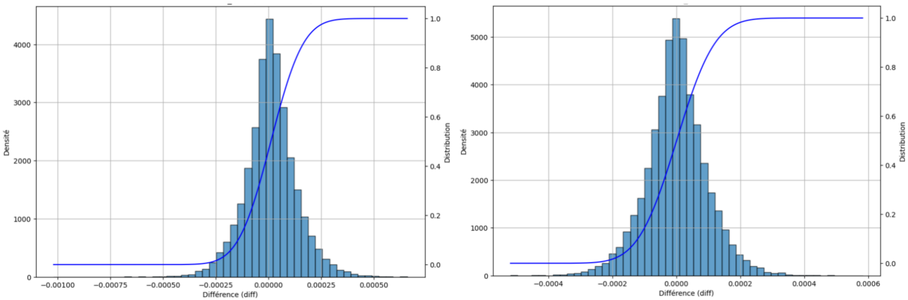
- Fonction de perte : MSE
- Optimisateur : Adam (Pytorch)
- Calendrier : DecayLR
- Entraînement : 3000 époques

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)	-

BS-ANN

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

BS-ANN



L'erreur absolue maximale : $8 \times 10^{-4} > 7 \times 10^{-4}$

IV-ANN (BS)

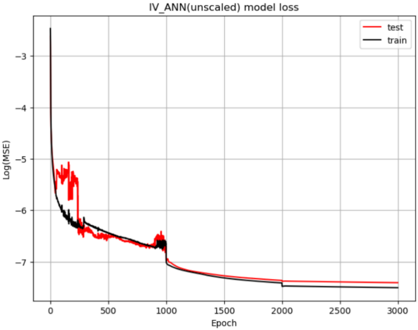
$$V/K --> log(V_t/K)$$
$$\tilde{V} = V_t - \max(S_t - Ke^{-r\tau}, 0)$$

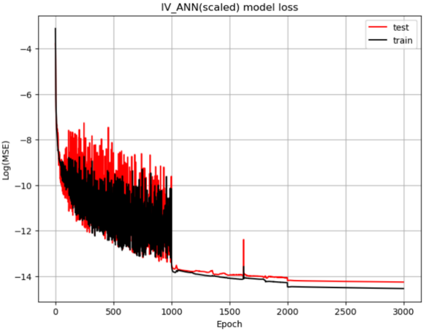
Parameter	**Range**	**Unit**
Prix de l'actif (S_0/K)	[0.5, 1.4]	-
Temps à l'échéance (τ)	[0.05, 1.0]	year
Taux sans risque (r)	[0.0, 0.1]	-
log(Valeur d'achat) ($\log(\tilde{V}/K)$)	$[-16.12, -0.94]$	-
Volatilité output (σ)	(0.05, 1.0)	-

IV-ANN

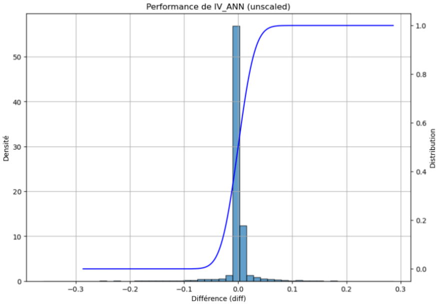
IV-ANN	**MSE**	**MAE**	**MAPE**	R^2
Article				
Standard	6.36×10^{-4}	1.24×10^{-2}	7.67×10^{-2}	0.97510
Log transformé	1.55×10^{-8}	9.73×10^{-5}	2.11×10^{-3}	0.9999998
Projet				
Standard	6.06×10^{-4}	8.03×10^{-3}	3.87×10^{-2}	0.9919511
Log transformé	6.40×10^{-7}	5.87×10^{-4}	1.56×10^{-3}	0.9999908

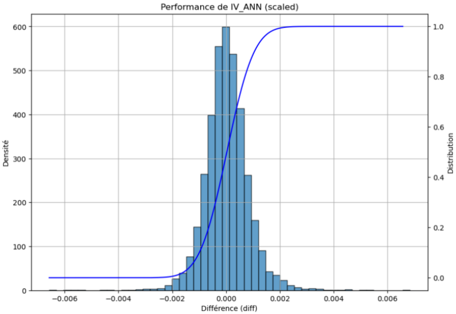
IV-ANN


IV_ANN(unscaled) model loss
test
train
Log(MSE)
Epoch
−3
−4
−5
−6
−7
0
500
1000
1500
2000
2500
3000


IV_ANN(scaled) model loss
test
train
Log(MSE)
Epoch
−4
−6
−8
−10
−12
−14
0
500
1000
1500
2000
2500
3000

IV-ANN





L'erreur absolue maximale : 6×10^{-3}

Heston-ANN

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

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

Heston-ANN

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

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

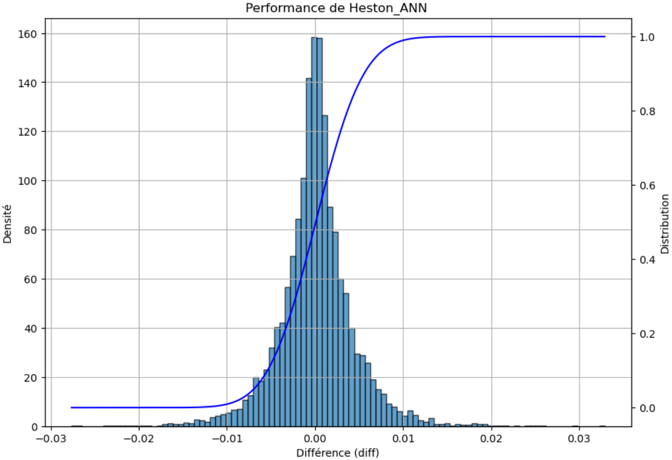
Heston-ANN

- Dans l'article : Méthode de COS
- Dans notre projet : Méthode de MC

	Trajectoires (M)	Prix estimé (mean)	Variance
0	100	0.263399	1.031622e-04
1	1000	0.250541	4.040018e-05
2	10000	0.249881	3.415849e-06
3	100000	0.249806	6.007312e-07
4	1000000	0.249592	3.325770e-08

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

Heston-ANN



L'erreur absolue maximale : 3×10^{-2}

Approches ANNs vs Méthodes Numériques

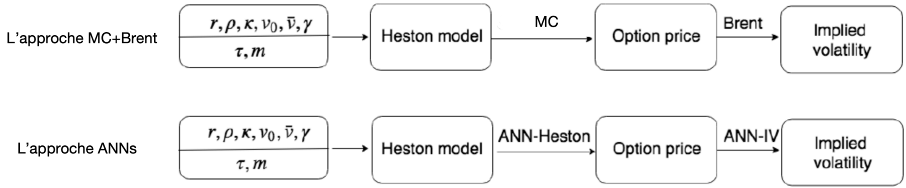
- Volatilité implicite par rapport à Black-Scholes
- Volatilité implicite par rapport à Heston

Volatilité implicite par rapport à Black-Scholes

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

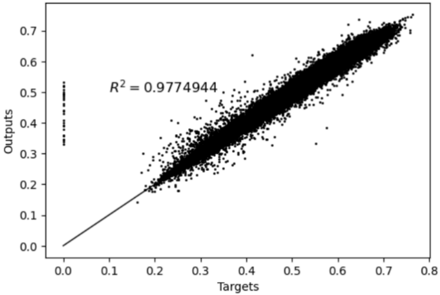
Conclusion : IV-ANN sur GPU **meilleur** !

Volatilité implicite par rapport à Heston



Volatilité implicite par rapport à Heston

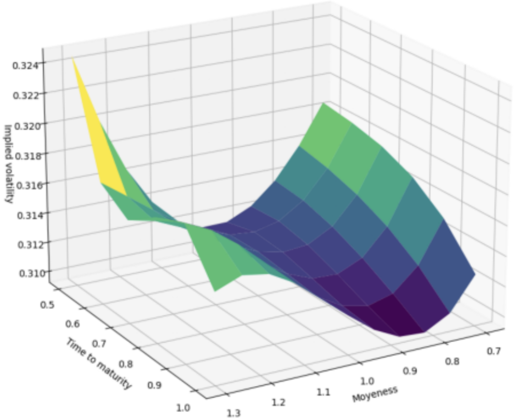
Heston-ANN & IV-ANN	MSE	MAE	MAPE	R^2
Résultats :	2.22×10^{-4}	9.13×10^{-3}	1.85×10^{-2}	0.9774944



Surface de volatilité implicite (Heston)

$m \in [0.7, 1.3]$, $\tau \in [0.5, 1.0]$

$\rho = -0.05$, $\kappa = 1.5$, $\gamma = 0.3$, $\bar{\nu} = 0.1$, $\nu_0 = 0.1$, $r = 0.02$



Merci beaucoup pour votre attention !

Yibo WANG & Qiyao ZHOU
10 Janvier 2025