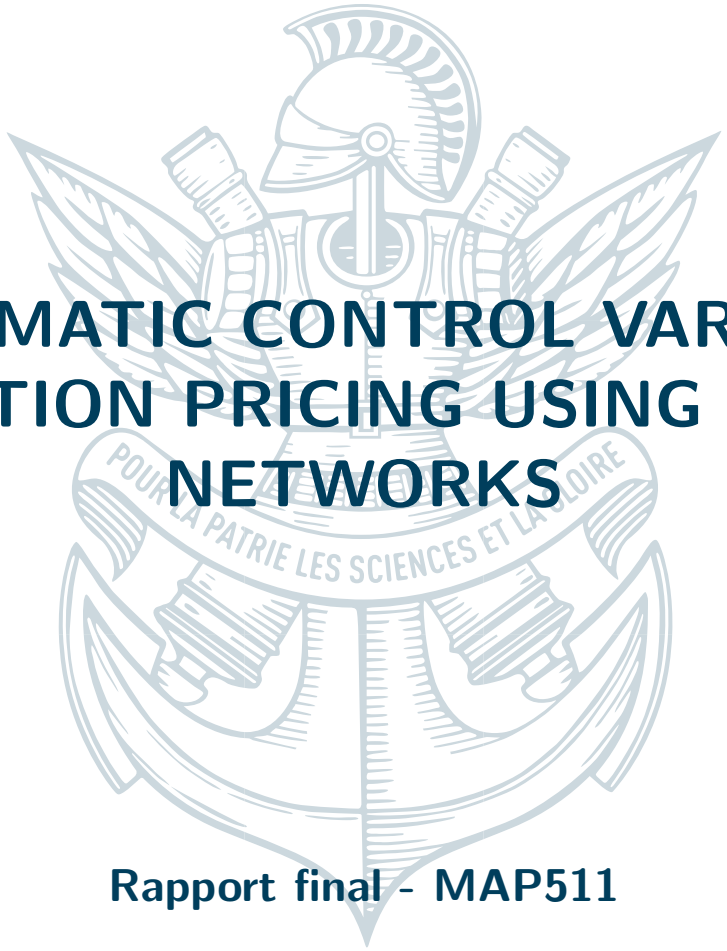


AUTOMATIC CONTROL VARIATES FOR OPTION PRICING USING NEURAL NETWORKS

Rapport final - MAP511

30 août 2024

Alain Kuissu, Khadija Slim

TABLE DES MATIÈRES

INTRODUCTION

Dans ce rapport, nous allons explorer le fascinant domaine de tarification, un sujet d'autant plus intéressant en raison de la diversité des produits financiers que nous pouvons évaluer. Les méthodes de tarification, avec les différentes techniques que nous pouvons développer pour minimiser la variance, offrent un terrain d'études permettant de comprendre comment, à partir des mêmes données, nous pouvons prédire de manière plus optimisée le prix d'un produit financier dans un contexte d'improbabilité intensément disputée.

Afin de restreindre notre domaine d'étude et de faciliter la comparaison des diverses méthodes que nous implémenterons par la suite, nous avons décidé de nous concentrer sur l'analyse des résultats liés au prix d'un panier composé de 10 sous-jacents, ainsi que sur l'option de vente d'un portefeuille comprenant le pire rendement parmi 10 sous-jacents également. Nous aborderons ces problématiques dans le cadre du modèle Black-Scholes, où chaque volatilité sous-jacente demeure constante puis des modèles plus complexes comme le modèle de Heston ayant une volatilité stochastique.

Ces problèmes de tarification se résument souvent au calcul d'une intégrale de grande dimension, qui est généralement estimée à l'aide de la méthode de Monte Carlo. En fait, la précision d'un estimateur de Monte Carlo avec M simulations est donnée par $\frac{\sigma}{\sqrt{M}}$. Cela signifie que sa convergence est insensible à la dimension du problème. Cependant, cette convergence peut être relativement lente en fonction de la variance σ de la fonction à intégrer. Pour résoudre un tel problème, on effectuerait des techniques de réduction de variance telles que l'échantillonnage d'importance, la stratification, ou les variables de contrôle. Dans ce rapport, nous concentrerons nos efforts sur la méthode des variables de contrôle.

Tout au long de notre recherche, nous ferons appel à plusieurs architectures de réseaux de neurones afin de construire des variables de contrôle. En effet, les réseaux de neurones, de par leur propriété d'approximation universelle, apparaissent comme de bons choix pour simuler des variables de contrôle. Dans un premier temps, nous les utiliserons pour réduire la dimension des entrées pour mieux calculer l'espérance de la variable de contrôle par intégration numérique, espérance ayant un rôle important dans la construction de la nouvelle variable sur laquelle nous appliquerons la méthode de Monte Carlo. Ensuite, nous les déploierons pour calculer directement ladite variable de contrôle et retrouver analytiquement une expression de cette espérance.

Pour finaliser notre travail, nous nous efforcerons d'appliquer les modèles que nous avons développés dans des situations plus complexes et réalistes, capturant ainsi les variations de la volatilité des actifs financiers de manière plus précise au fil du temps. Nous nous concentrerons notamment sur le modèle de volatilité locale et le modèle de volatilité stochastique afin d'appréhender les nuances de manière réaliste.

1
PRÉSENTATION DU PROBLÈME

Les méthodes classiques de tarification et de couverture peuvent parfois être très lentes. Avec le nouveau contexte réglementaire et alors que nous recherchons des performances de haut niveau, la demande de méthodes de tarification et de couverture efficaces devient très pressante.

Nous remarquons que la methode de Monte Carlo est le plus couramment utilisé. En fait, elle offre une grande flexibilité car elle peut être utilisé sous n'importe quel modèle de diffusion et avec autant d'actifs sous-jacents que nécessaire. De plus, sa variance est insensible à la dimension du problème, ce qui la rend très adapté aux problèmes d'intégration de grande dimension qui sont très courants en finance. Alors, comment fonctionne la méthode de Monte Carlo et comment l'utilisons-nous dans les problèmes financiers ?

Nous considérons l'équation differentielle stochastique suivante qui régit la diffusion des actifs sous-jacents S_i :

$$dS_{i_t} = \mu_i(t, S_{i_t})dt + \sigma_i(t, S_{i_t})dB_{i_t}. \tag{1}$$

où les B_i sont des mouvements browniens corrélés et soit g le payoff dépendant des processus $(S_{i_t})_{0\leq t\leq T}$. Le calcul du prix de g se limite souvent au calcul d'une espérance de la forme $E(f(X))$ qui, en utilisant la loi forte des grands nombres, peut être approchée avec l'estimateur $S_M = \frac{1}{M}\sum_{i=1}^{M} f(X_i)$, où les X_i sont des échantillons aléatoires de X et M un nombre suffisamment grand. S_M est l'estimateur de Monte Carlo. Nous posons $\sigma^2 = Var(f(X))$, la variance de S_M est donnée par

$$Var(S_M) = Var\left(\frac{1}{M}\sum_{i=1}^{M} f(X_i)\right) = \frac{\sigma^2}{M}. \tag{1}$$

Cependant, cette convergence peut être relativement lente en fonction de la variance σ de la fonction à intégrer. Pour résoudre un tel problème, on effectuerait des techniques de réduction de variance.

C'est dans ce cadre que s'inscrit notre projet, réduire la variance de notre estimateur de Monte Carlo par une méthode dite de la variable de contrôle.

L'innovation de notre approche sera d'exploiter le principe d'approximation universelle des réseaux de neurones pour construire des variables de contrôle automatiques et adéquates pour réduire la variance.

2
PRINCIPE DE RÉSOLUTION PAR LA MÉTHODE DE VARIABLE DE CONTRÔLE :

Nous pouvons notez que la variance de l'estimateur est proportionnelle à la variance de l'intégrande. Par conséquent, il serait intéressant de trouver une variable Y telle que :

$$\begin{cases} E(Y) = E(f(X)), \\ Var(Y) < Var(f(X)). \end{cases}$$

Il existe plusieurs techniques pour trouver une telle variable Y. Ici, nous nous concentrerons sur **les variables de contrôle**.

Soit $Y = f(X) - h(X) + E(h(X))$, où nous supposons que nous pouvons calculer $E(h(X))$ exactement.La variable $h(X)$ est notre variable de controle.

En écrivant que :

$$\begin{cases} Var(Y) = Var(f(X)) + Var(h(X)) - 2Cov(f(X), h(X)), \\ Cov(f(X), h(X)) > \frac{1}{2} Var(h(X)) \end{cases}$$

Nous déduisons que $Var(Y) < Var(f(X))$.

Nous souhaitons créer des variables de contrôle sophistiquées qui s'adaptent automatiquement à la fonction f. Pour ce faire, il faut trouver une fonction h, fortement corrélée à f, et pour laquelle nous avons une méthode exacte pour calculer $E(h(X))$ ou au moins une méthode numérique plus précise que Monte Carlo.

Nous suggérons deux méthodes principales pour créer des variables de contrôle automatiques.

La première concerne les problèmes de haute dimension avec de faibles dimensions effectives. Pour ce cas, nous proposons un rééseau de neurone H adéquat visant à réduire la dimension des entrées tout en apprenant le payoff pour pouvoir effectuer une integration numerique en faible dimension permettant d'obtenir $E(H(X))$.

Dans la deuxième méthode, nous suggérons d'utiliser un réseau neuronal H comme variable de contrôle. Le réseau apprend le payoff $f(X)$ de sorte qu'il est naturellement corrélé à lui (nous apprenons en fait le modèle de diffusion, la discrétisation et le payoff en une seule fois). L'architecture du réseau est choisie judicieusement, de sorte que $E(H(X))$ peut être calculé facilement.

3
FONCTIONNEMENT D'UN RESEAU DE NEURONE :

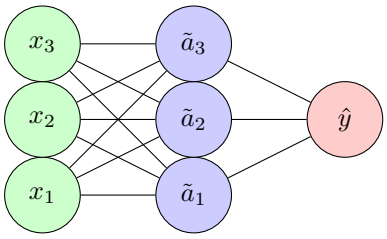
Un réseau de neurones est un modèle mathématique inspiré du fonctionnement du cerveau humain. Il est composé de neurones organisés en couches, chaque neurone étant connecté à tous les neurones de la couche suivante. Un réseau typique comporte trois types de couches : la couche d'entrée, une ou plusieurs couches cachées, et la couche de sortie.

Structure d'un Neurone

Un neurone dans une couche cachée prend en entrée un vecteur $\mathbf{x}$ de valeurs provenant de la couche précédente. Chaque entrée est associée à un poids w_i, et le neurone applique une fonction d'activation g à la somme pondérée des entrées et des poids pour introduire de la non-linearité(sinon ce serait juste une combinaison lineaire), additionnée à un biais b :

$$\tilde{x} = \sum_{i=1}^{N} w_i \cdot x_i + b$$

$$a = g(\tilde{x})$$



Où N est le nombre d'entrées dans la couche d'entrée du reseau(en vert sur le graphe), $\tilde{x}$ est le potentiel d'activation, g est la fonction d'activation, et a est la sortie de la couche cachée (en bleu).

Propagation Avant (Forward Propagation)

La propagation Avant consiste à calculer les sorties du réseau pour un ensemble d'entrées. Pour une entrée $\mathbf{X}$, la sortie de la couche cachée est calculée comme suit :

$$\tilde{\mathbf{X}} = \mathbf{W} \cdot \mathbf{X} + \mathbf{b}$$

$$\mathbf{A} = g(\tilde{\mathbf{X}})$$

Où $\mathbf{W}$ est la matrice des poids, $\mathbf{b}$ est le vecteur des biais, g est la fonction d'activation appliquée élément par élément, et $\mathbf{A}$ est la sortie de la couche cachée.

De même, la sortie de la couche de sortie est calculée de manière similaire.

Entraînement Supervisé - Régression

Pour un problème de régression, nous disposons d'un ensemble d'entraînement $\{(\mathbf{X}_1, y_1), (\mathbf{X}_2, y_2), \ldots, (\mathbf{X}_N, y_N)\}$ où $\mathbf{X}_i$ est l'entrée et y_i est la sortie attendue. L'objectif est de minimiser la différence entre les sorties prédites $\hat{y}_i$ et les sorties réelles y_i.

• Fonction de Coût

La fonction de coût mesure la différence entre les sorties prédites et les sorties réelles. Une fonction de coût couramment utilisée pour la régression est l'erreur quadratique moyenne (MSE) :

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_i - y_i)^2$$

• Rétropropagation (Backpropagation)

La rétropropagation est utilisée pour ajuster les poids et les biais du réseau afin de minimiser la fonction de coût. Elle consiste à calculer les gradients de la fonction de coût par rapport aux poids et aux biais, puis à mettre à jour ces paramètres en utilisant une technique d'optimisation telle que la descente de gradient.

• Mise à Jour des Paramètres

Les poids et les biais sont mis à jour en utilisant la règle de mise à jour de la descente de gradient :

$$W_{ij} = W_{ij} - \alpha\frac{\partial \text{MSE}}{\partial W_{ij}}$$

$$b_i = b_i - \alpha\frac{\partial \text{MSE}}{\partial b_i}$$

Où α est le taux d'apprentissage.

Approximation Universelle

Un résultat clé dans la théorie des réseaux de neurones est le théorème d'approximation universelle. Ce théorème énonce que, sous certaines conditions, un réseau de neurones à une couche cachée peut approximer n'importe quelle fonction continue sur un sous-ensemble borné de $\mathbb{R}^n$ avec une précision arbitraire.

Plus formellement, soit $f : \mathbb{R}^n \to \mathbb{R}$ une fonction continue. Alors, pour tout $\epsilon > 0$ et tout sous-ensemble borné $D \subset \mathbb{R}^n$, il existe un réseau de neurones avec une seule couche cachée tel que, pour tout $\mathbf{x} \in D$:

$$|f(\mathbf{x}) - \hat{f}(\mathbf{x})| < \epsilon$$

Où $\hat{f}(\mathbf{x})$ est la sortie du réseau de neurones pour l'entrée $\mathbf{x}$.

Ce résultat montre la puissance expressive des réseaux de neurones, ce qui signifie qu'ils sont capables d'approximer une large classe de fonctions, faisant d'eux des outils polyvalents dans le domaine de l'apprentissage automatique.

Ceci justifie egalement son choix pour trouver de facon automatique des variables de controle par approximation de la fonction de payoff pour une option donnée.

4
DIFFÉRENTES MÉTHODES DE DÉTERMINATION DE LA VARIABLE DE CONTRÔLE PAR DES RÉSEAUX DE NEURONES :

4.1 Variables de contrôle avec intégration numérique :

Considerons $H : \mathbb{R}^N \to \mathbb{R}$ un réseau neuronal qui approxime la payoff f étant donnée la variable normale $Z = (Z_1, \ldots, Z_N)$. Nous écrivons

$$Y = f(Z) - H(Z) + E(H(Z))$$

Si le réseau est bien entraîné, $H(Z)$ est fortement corrélé à $f(Z)$ et nous avons juste besoin de savoir comment calculer $E(H(Z))$ analytiquement pour l'utiliser comme variable de contrôle.

Pour cette partie, nous nous en tenons à l'architecture du réseau donnée par la figure 1 avec un seul neurone dans la couche intermediaire :

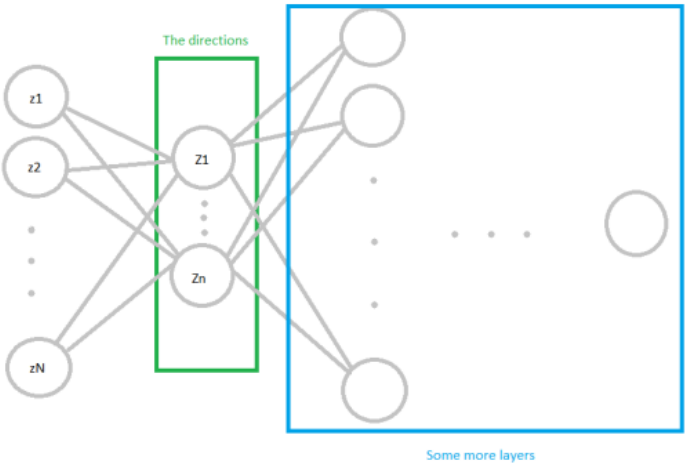


Figure 1 – Architecture de réseau pour la réduction de dimension

Nous pouvons écrire

$$H(Z) = (\tilde{H} \circ a_1)(Z)$$

où $a_1(Z) = WZ$ est la première couche cachée du réseau (le bloc vert à la Figure 1) et $\tilde{H}$ représente le reste des couches (le bloc bleu à la Figure 1).

$$E(H(Z)) = E((\tilde{H} \circ a_1)(Z)) = E(\tilde{H}(\tilde{Z}))$$

où $\tilde{Z} = a_1(Z) = WZ \sim \mathcal{N}(0, WW^T)$.

Comme $\tilde{Z}$ a une petite dimension, l'intégrale peut être estimée numériquement en utilisant la methode de quadrature de notre choix. Bien sûr, cela ne devrait être utilisé que lorsque la dimension effective est suffisamment petite. C'est pour cela que nous avons choisi de prendre $n == 1$ dans ce cas pour la premiere couche.

Dès lors on peut poser $WW^T = v^2$ avec v dans $\mathbb{R}$ car $\tilde{Z}$ devient alors une variable gaussienne de dimension 1.

$$E(H(Z)) = E(\tilde{H}(\tilde{Z})) = \int_{-\infty}^{\infty} \tilde{H}(z) \frac{1}{\sqrt{2\pi v^2}} e^{-\frac{z^2}{2v^2}} dz$$

où $\frac{1}{\sqrt{2\pi v^2}} e^{-\frac{z^2}{2v^2}}$ est la densité de probabilité de la loi normale $N(0, v^2)$.

Nous utilisons enfin une methode de quadrature telle que la methode des trapezes pour calculer cette intégrale.

$$Y = f(Z) - H(Z) + \int_{-\infty}^{\infty} \tilde{H}(z) \frac{1}{\sqrt{2\pi v^2}} e^{-\frac{z^2}{2v^2}} dz$$

La **méthode des trapèzes** est une technique numérique utilisée pour approximer l'intégrale d'une fonction continue sur un intervalle. Elle consiste à diviser l'intervalle en plusieurs segments et à approximer l'aire sous la courbe de la fonction sur chaque segment par un trapèze.

Le calcul de l'aire d'un trapèze formé par deux points adjacents $(x_i, h(x_i))$ et $(x_{i+1}, h(x_{i+1}))$ est donné par la formule :

$$\text{Aire} = \frac{1}{2} \cdot (x_{i+1} - x_i) \cdot (h(x_i) + h(x_{i+1}))$$

Où h est une fontion continue le segment $[x_i, x_{i+1}]$.
En appliquant cette formule à chaque segment de l'intervalle, la méthode des trapèzes approxime l'intégrale totale de la fonction.

Maintenant, en remplaçant l'intégrale impropre dans l'équation donnée par son approximation par la méthode des trapèzes, nous obtenons :

$$\begin{aligned} Y &= f(Z) - H(Z) + \sum_{i=1}^{N} \frac{1}{2} \cdot \Delta z \cdot (\tilde{H}(z_i) + \tilde{H}(z_{i+1})) \\ &= f(Z) - H(Z) + \frac{1}{2} \cdot \Delta z \cdot \sum_{i=1}^{N} (\tilde{H}(z_i) + \tilde{H}(z_{i+1})) \end{aligned}$$

Où N est le nombre de segments utilisés pour diviser l'intervalle de l'intégrale (destiné à tendre vers l'infini), Δz est la largeur de chaque segment, et z_i et z_{i+1} sont les points de division successifs le long de l'axe z.

4.2 Variables de contrôle avec intégration analytique :

Nous considérons ici une nouvelle architecture de réseau de neurones plus simple, nous permettant de retrouver analytiquement l'expression de l'espérance de la variable de contrôle évitant ainsi une intégration numérique. Elle est constituée de :

— Une couche d'entrée
— Une couche intermédiaire avec une fonction d'activation ReLu (Rectified Linear Unit)
— Une couche de sortie renvoyant le payoff

Cette architecture est visible sur la figure 2 .

En considérant $H(Z)$ la sortie du neurone correspondant à la variable de contrôle après entrainement, on a :

$$\begin{aligned} H(Z) &= Z_2 \\ &= W_2 Z_1 + b_2 \\ &= W_2 (W_1 Z + b_1)^+ + b_2 \end{aligned}$$

Ainsi,

$$E(H(Z)) = W_2 E((W_1 Z + b_1)^+) + b_2$$

Mais

$$\begin{aligned} W_1 Z + b_1 &= \begin{pmatrix} \sum_{j=1}^N ((W_1))_{1j} Z_j + (b_1)_1 \\ \vdots \\ \sum_{j=1}^N ((W_1))_{nj} Z_j + (b_1)_n \end{pmatrix} \\ &= \begin{pmatrix} \sigma_1 Y_1 + \mu_1 \\ \vdots \\ \sigma_n Y_n + \mu_n \end{pmatrix} \end{aligned}$$

Avec $\mu_i = (b_1)_i$ et $\sigma_i^2 = \sum_{j=1}^N ((W_1)_{ij})^2$ et $Y_i \sim N(0,1)$. On peut alors calculer $E((\sigma Y + \mu)^+)$.
En effet,

$$\begin{aligned} E((\sigma Y + \mu)^+) &= \sigma \int_{-\mu/\sigma}^{\infty} y f_{N(0,1)}(y) dy + \mu P\left(Y \geq -\frac{\mu}{\sigma}\right) \\ &= \sigma \int_{-\mu/\sigma}^{\infty} \frac{1}{\sqrt{2\pi}} y \exp\left(-\frac{1}{2} y^2\right) dy + \mu \left(1 - N\left(-\frac{\mu}{\sigma}\right)\right) \\ &= \sigma \sqrt{\frac{2}{\pi}} \left[-\exp\left(-\frac{1}{2} y^2\right)\right]_{-\mu/\sigma}^{\infty} + \mu \left(1 - N\left(-\frac{\mu}{\sigma}\right)\right) \\ &= \frac{\sigma}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{\mu}{\sigma}\right)^2\right) + \mu \left(1 - N\left(-\frac{\mu}{\sigma}\right)\right), \end{aligned}$$

Avec N la fonction de répartition d'une loi normale centrée réduite. On a alors,

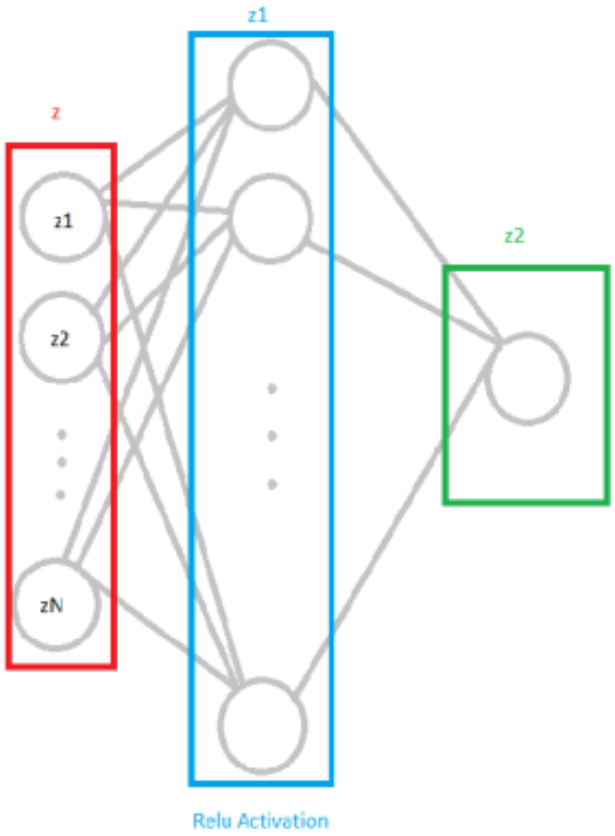


Figure 2 – Architecture de réseau pour intégration analytique

$$E(H(Z)) = W_2 \begin{pmatrix} \vdots \\ \frac{\sigma_i}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\left(\frac{\mu_i}{\sigma_i}\right)^2\right) + \mu_i\left(1 - N\left(-\frac{\mu_i}{\sigma_i}\right)\right) \\ \vdots \end{pmatrix} + b_2. \tag{1}$$

Avec $\mu_i = (b_1)_i$ and $\sigma_i^2 = \sum_{j=1}^{N}((W_1)_{ij})^2$.

5
APPLICATIONS À DIFFÉRENTS MODÈLES ET RÉSULTATS :

Nous avons appliqué ces méthodes à différents modèles en dimension 1, puis dans des cas de grandes dimensions. Nous sommes passés d'un modèle simple tel que le modèle de Black-Scholes à des modèles de plus grande complexité, comme celui de Heston.

5.1 Cas Black-Scholes en dimension 1 :

Le modèle de Black-Scholes est un modèle mathématique largement utilisé pour estimer le prix des options financières, en particulier les options européennes. Le modèle prend en compte les éléments suivants :

— $S_0 = S(0)$: le prix initial de l'actif sous-jacent (par exemple, une action).
— K : le prix d'exercice de l'option.
— T : la durée de vie restante de l'option (exprimée en années).
— r : le taux d'intérêt sans risque.
— σ : la volatilité de l'actif sous-jacent.

Le modèle de Black-Scholes unidimensionnel décrit le cours d'une action $S(t)$ en utilisant un mouvement brownien géométrique (GBM). L'équation différentielle stochastique pour $S(t)$ est donnée par :

$$dS(t) = rS(t)dt + \sigma S(t)dW(t)$$

où :
— $S(t)$ est le cours de l'action au temps t,
— $W(t)$ est un mouvement brownien standard.

La solution de cette équation est un mouvement brownien géométrique, donné par l'équation :

$$S(t) = S(0)\exp\left((r - \frac{\sigma^2}{2})t + \sigma W(t)\right)$$

où $S(0) = S_0$ est le cours initial de l'action. Le prix d'une option d'achat (call) européenne selon le modèle de Black-Scholes est donné par la formule :

$$C = S_0 \cdot N(d_1) - K \cdot e^{-rT} \cdot N(d_2) \tag{2}$$

Où :

$$d_1 = \frac{\ln(S_0/K) + (r + (\sigma^2/2))T}{\sigma\sqrt{T}}$$
$$d_2 = d_1 - \sigma\sqrt{T}$$
$$N(x) = \text{la fonction de distribution cumulative de la loi normale standard.}$$

L'estimation de Monte Carlo peut être utilisée pour estimer le prix d'une option en simulant un grand nombre de trajectoires du prix de l'actif sous-jacent et en prenant la moyenne des payoffs actualisés.

Le prix d'un call selon l'estimation de Monte Carlo est donné par la formule :

$$C \approx e^{-rT} \cdot \frac{1}{M}\sum_{i=1}^{M} \max(S_T^{(i)} - K, 0) \tag{3}$$

Où $S_T^{(i)}$ est le prix de l'actif sous-jacent à l'instant T dans la i-ème trajectoire simulée et M le nombre totale de simulation.

Nous appliquons nos techniques de réduction de variance dans ce cas pour le pricing d'un call avec les paramètres suivants :

— Nombre de simulations M : 1000
— Prix initial S_0 : 100
— Temps total T : 2.0 années
— Taux d'intérêt constant r : 0.05
— Volatilité σ : 0.3
— Prix d'exercice K : 105

Methode	**Prix du Call**	$\pm 1.96 \cdot$ **Std/M**	**Variance de l'estimateur**
Monte Carlo Naif	19.321743	2.394494	**1492.503586**
Integration numerique	18.990173	0.125136	**4.076201**
Integration analytique	18.987725	0.125544	**4.102779**
Methode	**Ratio des Variances**		
Naive Monte Carlo	1.000000		
Integration numerique	**366.150613**		
Integration analytique	**363.778652**		

Table 1 – Resultats du pricing d'un call avec reduction de variance

Nous commençons par simuler un échantillon de M gaussiennes i.i.d que nous utilisons pour calculer un vecteur de payoff pour chaque trajectoire simulée. Ce vecteur, de dimension M, sera la cible/vecteur cible de notre réseau de neurones dans le cadre d'un apprentissage supervisé.

Suite à l'application de nos différentes méthodes, et compte tenu des paramètres choisis pour lesquels le prix du call dans le modèle de Black-Scholes est de 18.99, nous obtenons le tableau 1. Ce tableau présente le prix du call fourni par chaque estimateur de Monte Carlo, la demi-largeur de l'intervalle de confiance, la variance des estimateurs, ainsi que le ratio entre la variance de l'estimateur de chaque méthode et l'estimateur naïf.

Nous pouvons constater l'apport significatif de nos deux méthodes par rapport à l'approche naïve. Le tableau 1 démontre que la variance a été réduite de 366.15 dans le cas de l'intégration numérique et de 363.77 dans le cas de l'intégration analytique. Nous proposons ensuite une explication détaillée de notre approche pour l'entraînement des différents réseaux de neurones.

5.1.1 • Intégration numérique :

Après avoir testé plusieurs architectures différentes, nous avons choisi une structure simple avec un nombre raisonnable de couches cachées afin de pouvoir effectuer plusieurs entraînements et de limiter le temps de calcul.

Il s'agit d'une structure illustrée dans la figure 1. Pour le bloc en bleu, nous avons opté pour 3 couches cachées de dimensions respectives 100, 200 et 10, en plus de la couche de sortie.

En ce qui concerne l'entraînement de notre réseau de neurones, nous utilisons les hyperparamètres suivants :

Table 2 – Hyperparametre pour le reseau servant à l'integration numerique

Hyperparametre	Valeurs
batch_size	64
learning_rate ou pas d'apprentissage ou taux d'apprentissage	0.001
epochs	200
Optimizer	Adam
Loss function	MSE(Mean Square Error)

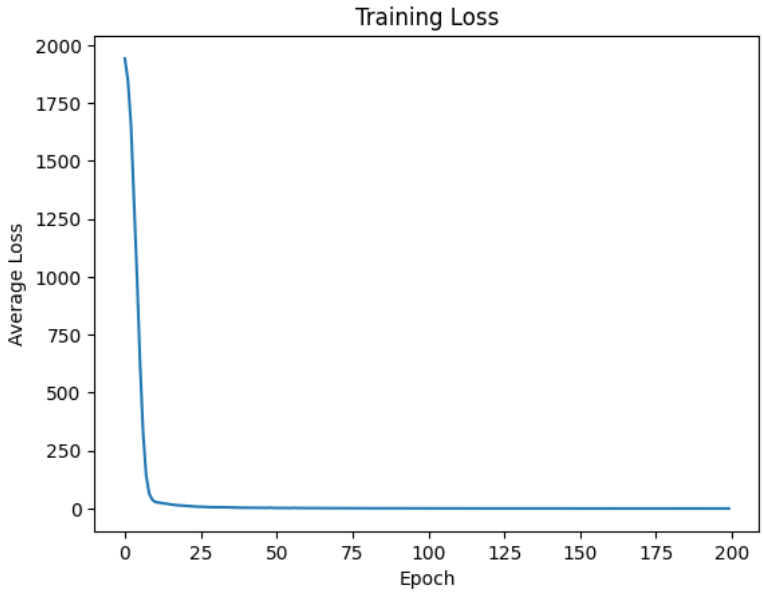


Figure 3 – Courbe d'apprentissage lors de l'entrainement du réseau de neurone

1. **Batch Size (`batch_size`) :** Le `batch_size` est le nombre d'échantillons d'entraînement utilisés dans une itération pour mettre à jour les poids du réseau. Lorsque le `batch_size` est égal à 64, cela signifie que 64 exemples sont traités simultanément avant la mise à jour des poids. Une plus grande valeur de `batch_size` peut accélérer l'entraînement, mais nécessite plus de mémoire. Un `batch_size` plus petit peut améliorer la généralisation du modèle.
2. **Learning Rate (`learning_rate`) :** Le taux d'apprentissage contrôle la taille des pas que prend l'optimiseur lors de la mise à jour des poids du réseau. Un taux d'apprentissage plus élevé permet des mises à jour plus rapides, mais peut conduire à une instabilité. Un taux d'apprentissage trop bas peut ralentir l'entraînement et le modèle peut rester bloqué dans un minimum local.
3. **Epochs (`epochs`) :** Le nombre d'epochs représente le nombre de fois que l'ensemble complet des données d'entraînement est passé à travers le réseau. Un nombre trop faible d'epochs peut entraîner un sous-apprentissage, tandis qu'un nombre trop élevé peut entraîner un surapprentissage ou overfitting.
4. **Optimizer :** L'optimiseur détermine comment les poids du réseau sont mis à jour pour minimiser la fonction de perte. Adam est un optimiseur populaire qui adapte automatiquement le taux d'apprentissage pendant l'entraînement tel une descente de gradient à pas optimal.
5. **Loss Function (`MSE`) :** La fonction de perte mesure la différence entre les valeurs prédites par le modèle et les valeurs réelles. La MSE (Mean Square Error) est très utilisée pour les tâches de régression. Elle calcule la moyenne des carrés des écarts entre les prédictions et les valeurs réelles.

Ces hyperparamètres ne sont pas choisis au hasard. Ils résultent d'un processus de tuning qui nous a permis de comprendre que **plus le réseau overfit, meilleure est l'approximation**. En effet, plus H est fortement lié aux données Z, plus il existe une corrélation significative conduisant à une meilleure variable de contrôle. **L'overfitting devient donc un objectif pour nous, justifiant ainsi le choix d'un grand nombre d'epochs et de batch_size**. Nous pouvons l'observer à travers la courbe d'apprentissage présentée dans la figure 3.

Nous prenons aussi un taux d'apprentissage assez haut pour accélérer la convergence.

Une fois la variable de contrôle obtenue après l'entraînement, nous pouvons passer à une intégration numérique avec la méthode de quadrature de notre choix. Nous sommes restés sur une approche simple avec une méthode des trapèzes, comme décrite en 4.2, avec 10 000 subdivisions.

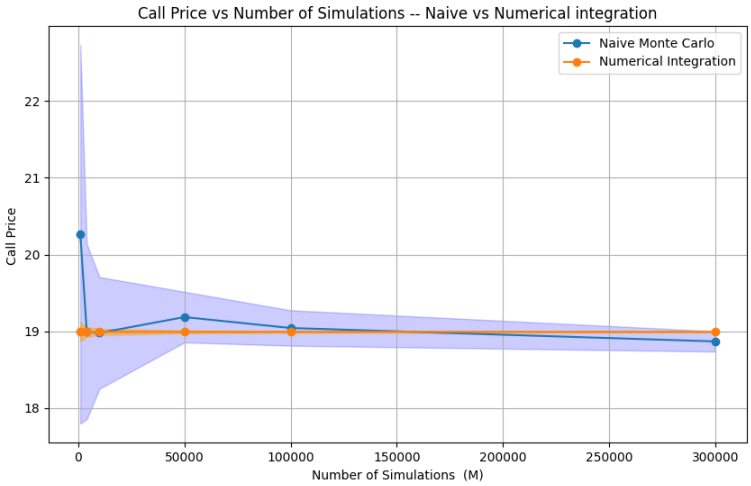


Figure 4 – Convergence et intervalle de confiance avant et après réduction par intégration numérique

Cette méthode réduit la variance de l'estimateur de Monte Carlo naif de **99,73%**. Cette réduction est énorme et montre l'efficience d'un réseau ayant overfitté sur les données.

Pour d'autres valeurs de M, on obtient le graphe de la Figure 4 présentant la convergence du prix du call avec un intervalle de confiance à 95%.

5.1.2 • Integration analytique :

Nous avons décidé de prendre 3000 neurones dans notre unique couche cachée pour ce cas ci.

Nous utilisons les hyperparamètres suivants :

Table 3 – Hyperparamètres pour le réseau servant à l'intégration analytique

Hyperparamètres	Valeurs
batch_size	64
learning_rate ou pas d'apprentissage ou taux d'apprentissage	0.001
epochs	300
Optimizer	Adam
Loss function	MSE(Mean Square Error)

Le choix des hyperparamètres est motivé par les mêmes raisons que pour le réseau précédent : nous cherchons l'overfitting/surapprentissage pour notre modèle.

Une fois le réseau entrainé nous calculons analytiquement l'espérance par la formule donnée en 4.3.

Cette méthode réduit la variance de **99,72%**. On observe donc la même efficacité que la méthode précédente. L'entraînement est efficace grâce aux paramètres permettant au modèle d'être en surapprentissage, et l'expression analytique fournit une valeur exacte de l'espérance de la variable de contrôle

Pour d'autres valeurs de M, on obtient le graphe de la Figure 5 présentant la convergence du prix du call avec un intervalle de confiance à 95%.

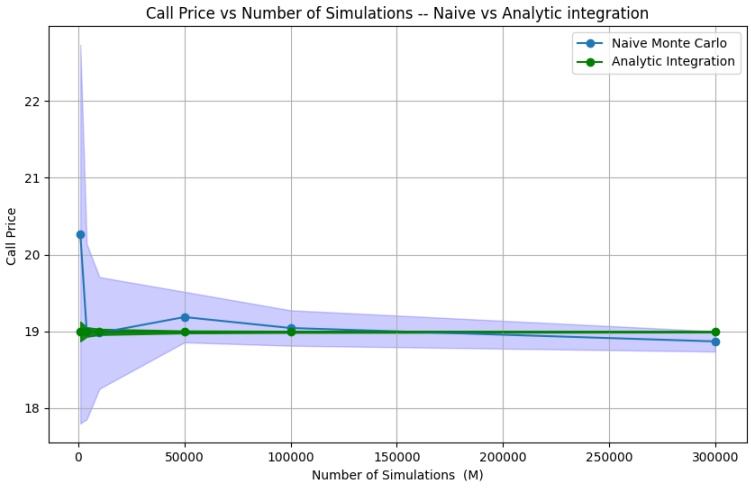


Figure 5 – Convergeance et intervalle de confiance avant et apres reduction par integration analytique

5.2 Cas Black-Scholes multidimensionnel :

Le modèle de Black-Scholes multidimensionnel est une extension du modèle de Black-Scholes original pour des actifs multiples. Dans ce modèle, chaque actif suit un mouvement brownien géométrique avec une corrélation possible entre les différents actifs. Le modèle est défini par l'équation différentielle stochastique suivante :

$$dS_i(t) = rS_i(t)dt + \sigma_i S_i(t)dB_i(t),$$

$$d(B_i(t)B_j(t)) = \rho_{ij}dt$$

où :

— $S_i(t)$ est le prix de l'actif i au temps t,
— r est le taux d'intérêt constant,
— σ_i est la volatilité de l'actif i,
— $dB_i(t)$ est l'incrément d'un mouvement brownien associé à l'actif i,
— ρ_{ij} est la corrélation entre les mouvements browniens associés aux actifs i et j.

L'équation différentielle stochastique ci-dessus modélise l'évolution du prix des actifs dans le temps. Le premier terme $rS_i(t)dt$ représente la croissance déterministe due au taux d'intérêt. Le deuxième terme $\sigma_i S_i(t)dB_i(t)$ représente la composante stochastique due à la volatilité de l'actif i.

Ce modèle est utilisé pour évaluer des options et d'autres produits dérivés dans un contexte multidimensionnel. Il repose sur les mêmes hypothèses que le modèle unidimensionnel comme l'absence de frais de transaction, la possibilité de vendre à découvert et la volatilité et le taux d'intérêt constants.

Nous utiliserons dans ce contexte nos méthodes pour pricer 2 types d'options exotiques :

1. **Call sur un panier d'actifs** :

$$\max\left(0, \sum_i \omega_i S_i(T) - K\right)$$

où :

— $S_i(T)$ est le prix de l'actif i à l'échéance T,

— ω_i est le poids de l'actif i dans le panier,
— K est le prix d'exercice (strike).

Ce payoff représente la différence entre la somme pondérée des prix des actifs à l'échéance et le prix d'exercice, prenant la valeur maximale entre zéro et cette différence.

2. **Put sur le Pire des Actif** : Le payoff d'un put sur le pire des actifs est donné par l'expression :

$$\max\left(0, K - \min_i S_i(T)\right)$$

où :
— $S_i(T)$ est le prix de l'actif i à l'échéance T,
— K est le prix d'exercice (strike).

Ce payoff représente la différence entre le prix d'exercice et le prix minimal parmi les actifs à l'échéance, prenant la valeur maximale entre zéro et cette différence.

Nous utilisons les paramètres suivants :

$$S_i(0) = 100.0 \quad \forall i, \quad K = 100.0, \quad \omega_i = \frac{1}{\text{nbsj}} \quad \forall i, \quad T = 1 \text{ an}$$

$$\sigma_i = 0.4 \quad \forall i, \quad \rho_{ij} = 0.75 \quad \forall i, j, \quad \text{nbsj} = 10., M = 1000$$

Ici, les paramètres sont définis comme suit :
— $S_i(0)$: Prix initial de l'actif i,
— K : Prix d'exercice (strike),
— ω_i : Poids de l'actif i dans le panier, défini comme $\frac{1}{\text{nbsj}}$,
— T : Durée de vie de l'option (1 an),
— σ_i : Volatilité de l'actif i,
— ρ_{ij} : Corrélation entre les browniens des actifs i et j,
— nbsj : Nombre d'actifs à considerer,
— M : Nombre de simulations
Une premiere difficulté notable est de pouvoir simuler les browniens corrélés.

La méthode de Cholesky est souvent utilisée pour simuler des variables aléatoires multivariées avec une structure de corrélation spécifiée. Pour simuler des browniens corrélés, le processus peut être décomposé en utilisant la factorisation de Cholesky de la matrice de corrélation.

Soit $Z = (Z_1, Z_2, \ldots, Z_n)$ un vecteur de variables aléatoires indépendantes et identiquement distribuées suivant une loi normale standard. La matrice de corrélation R est définie comme $R = \begin{bmatrix} 1 & \rho_{12} & \cdots & \rho_{1n} \\ \rho_{21} & 1 & \cdots & \rho_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{n1} & \rho_{n2} & \cdots & 1 \end{bmatrix}$, où ρ_{ij} est la corrélation entre Z_i et Z_j.

La factorisation de Cholesky de la matrice de corrélation R est donnée par $R = LL^T$, où L est une matrice triangulaire inférieure. La simulation des browniens corrélés est alors réalisée en multipliant la matrice L par le vecteur Z.

L'algorithme de simulation est le suivant :

1. Générer un vecteur Z de variables aléatoires normales indépendantes.
2. Calculer la factorisation de Cholesky $R = LL^T$ de la matrice de corrélation R.
3. Calculer $X = LZ$, où L est la matrice de Cholesky.

Le vecteur X ainsi obtenu constitue une simulation de browniens corrélés.

Une fois les browniens obtenus, nous pouvons simuler les trajectoires en utilisant le fait que le cours de chaque actif suit un mouvement brownien géometrique. On calcule les payoffs et on applique nos méthodes pour obtenir les résultats suivants :

Intégration numérique

L'architecture du réseau reste la même que pour le cas à une dimension et les hyperparamètres présentés dans le tableau 2 sont les mêmes également. La méthode réduit la variance de l'estimateur du prix du call

Méthode	Call sur panier	±1.96*Std/M Call	Variance Call	Put sur le pire	±1.96*Std/M Put
MC naif	17.85	1.92	964.23	25.73	1.25
Intégration numérique	16.21	0.08	1.49	26.46	0.37
Méthode	**Variance Put**	**Ratio Variance Call**	**Ratio Variance Put**		
MC naif	408.61	1.00	1.00		
Intégration numérique	36.26	**647.41**	**11.27**		

Table 4 – Résultats pour la méthode Naïve + Intégration Numérique

sur panier de **99,845%** et du put sur le pire des actifs de **91,126%**, comme en témoigne le tableau 4. Ce résultat atteste de l'efficacité de l'architecture choisie et de la combinaison entre la réduction de dimension avec le réseau de neurones suivi de l'intégration numérique en faible dimension (1 dans ce cas précis) pour le modèle de Black-Scholes multidimensionnel.

Integration Analytique

L'architecture du réseau reste la même que pour le cas à une dimension et les hyperparamètres presentés dans le tableau 3 sont les mêmes également.

La méthode réduit la variance de l'estimateur du prix du call sur panier de **99,9253%** et du put sur le pire des actifs de **99,28%** comme le montre le tableau 5 .

L'intégration analytique donne de meilleurs résultats dans les deux cas. Cela peut être dû aux limites de l'approximation de l'espérance de la variable de contrôle par intégration numérique, là où l'intégration analytique est plus précise.

Méthode	Call sur panier	±1.96*Std/M Call	Variance Call	Put sur le pire	±1.96*Std/M Put
MC naif	17.85	1.92	964.23	25.73	1.25
Intégration analytique	16.25	0.05	0.72	26.65	0.11
Méthode	**Variance Put**	**Ratio Variance Call**	**Ratio Variance Put**		
MC naif	408.61	1.00	1.00		
Intégration analytique	2.93	**1341.45**	**139.29**		

Table 5 – Résultats pour la méthode Naïve + Intégration Analytique

5.3 Volatilité Locale :

Nous considérons un modèle de volatilité locale où la fonction de volatilité prend la forme suivante :

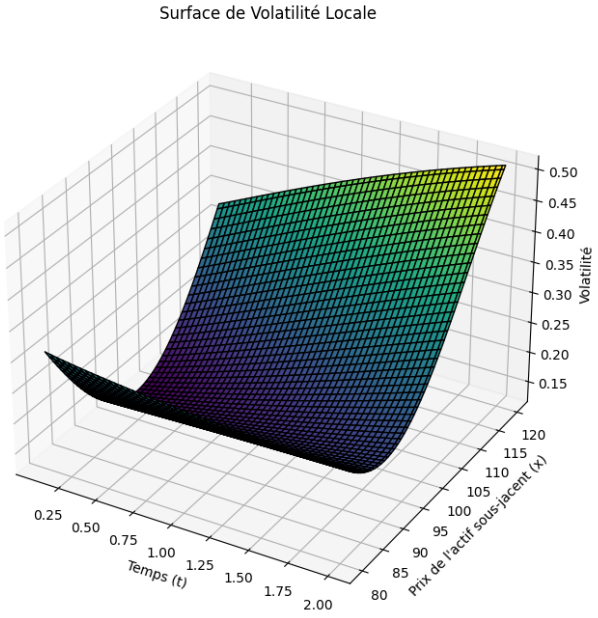


Figure 6 – Surface de Volatilité Locale

$$\sigma(t,x) = 0.6 \cdot (1.2 - e^{-0.1t} e^{-0.001(xe^{rt}-s)^2}) e^{-0.05\sqrt{t}}$$

Ce modèle de volatilité locale est paramétrique. Les auteurs de l' article [1] recommandent de prendre s égal au prix initial de l'actif sous-jacent afin que la formule ait un sens. Cela permettra de localiser le bas du "smile" comme vu sur la figure 6.

Ce modèle ne dispose pas d'une méthode de simulation exacte comme dans le modèle de Black-Scholes. Par conséquent, une méthode de discrétisation est nécessaire. Nous utilisons une discrétisation d'Euler avec 100 pas par an.

Integration Numerique

Les hyperparamètres et l'architecture du réseau sont identiques aux cas précédents.

On a une bonne réduction de la variance (99.24%) comme vu dans le tableau 6 mais la valeur est légérement en dehors de l'intervalle de confiance de l'estimateur naif.

Intégration Analytique

Les hyperparamètres et l'architecture du réseau sont identiques aux cas précédents.

Méthode	Call sur panier	±1.96*Std/M Call	Variance Call	Put sur le pire	±1.96*Std/M Put
MC naif	8.51	0.61	383.44	10.67	0.39
Intégration numérique	9.19	0.05	2.89	10.93	0.09
Méthode	Variance Put	Ratio Variance Call	Ratio Variance Put		
MC naif	156.40	1.00	1.00		
Intégration numérique	9.06	**132.65**	**17.27**		

Table 6 – Résultats Intégration Numérique Volatilité locale

Méthode	Call sur panier	±1.96*Std/M Call	Variance Call	Put sur le pire	±1.96*Std/M Put
MC naif	8.51	0.61	383.44	10.67	0.39
Intégration analytique	9.47	0.02	0.31	10.60	0.02
Méthode	Variance Put	Ratio Variance Call	Ratio Variance Put		
MC naif	156.40	1.00	1.00		
Intégration analytique	0.34	**1229.57**	**461.61**		

Table 7 – Résultats Intégration Analytique Volatilité locale

On a une bonne réduction de la variance (99.91%) comme vu dans le tableau 7 mais la valeur est en dehors de l'intervalle de confiance de l'estimateur naif.

Interprétation des résultats

On rajoute une dimension en plus à notre probleme à cause de la discrétisation du pas de temps.

On remarque que certains prix ne se situent pas dans l'intervalle de confiance à 95% du Monte Carlo naïf. Cela peut s'expliquer par plusieurs raisons :

Il peut y avoir un biais d'estimation introduit par les méthodes de réduction de variance, qui ne sont pas nécessairement sans biais. Par exemple, la méthode d'intégration numérique repose sur une approximation de l'espérance de la variable de contrôle, laquelle peut être imprécise si le nombre de points d'intégration est trop faible ou si la fonction à intégrer est irrégulière. La méthode d'intégration analytique dépend, quant à elle, de la qualité de l'approximation du payoff par le réseau de neurones, ce qui peut être imparfait si le réseau n'est pas bien entraîné ou si l'architecture n'est pas adaptée à la dimension qui a augmenté.

L'intervalle de confiance du Monte Carlo naïf n'est pas non plus une garantie absolue ; il existe toujours une probabilité non nulle que la vraie valeur soit en dehors de cet intervalle.

Enfin, ce décalage peut aussi être lié au modèle de volatilité locale choisi, qui résulte d'une calibration dont la qualité nous est inconnue.

5.4 Volatilité Stochastique :

Nous considérons le modèle de Heston pour la diffusion des actifs :

$$\begin{aligned} dS_i(t) &= rS_i(t)dt + \sqrt{\sigma_i(t)}S_i(t)dB_i(t) \\ d\sigma_i(t) &= \kappa_i(\bar{\sigma}_i - \sigma_i(t))dt + \nu_i\sqrt{\sigma_i(t)}(\gamma_i dB_i(t) + \sqrt{1-\gamma_i^2}d\tilde{B}_i(t)) \\ d < B >_t &= \Gamma dt, \quad d < \tilde{B} >_t = I_N dt \end{aligned}$$

Ici, le modèle inclut les composantes suivantes :
— $S_i(t)$: Le prix de l'actif i au temps t.
— r : Taux d'intérêt constant.
— $\sigma_i(t)$: Volatilité de l'actif i au temps t.
— $B_i(t)$ et $\tilde{B}_i(t)$: Processus de Wiener.
— κ_i : Taux de réversion du processus de volatilité.
— $\bar{\sigma}_i$: Niveau moyen du processus de volatilité.
— ν_i : Volatilité du processus de volatilité.
— γ_i : Corrélation constante entre l'actif i et son processus de volatilité.
— Γ : Matrice de corrélation entre les différents actifs.
— I_N : Matrice identité de taille $N \times N$.
— N : le nombre d'actifs.
Le modèle peut également être écrit de manière équivalente :

$$\begin{aligned} dS_i(t) &= rS_i(t)dt + \sqrt{\sigma_i(t)}S_i(t)dB_i(t) \\ d\sigma_i(t) &= \kappa_i(\bar{\sigma}_i - \sigma_i(t))dt + \nu_i\sqrt{\sigma_i(t)}d\widehat{B}_i(t) \end{aligned}$$

Avec B and $\widehat{B}$ des processus de Wiener satisfaisant :

$$\begin{aligned} d < B >_t &= \Gamma dt \\ d < B, \widehat{B} >_t &= \gamma\Gamma dt \\ d < \widehat{B} >_t &= (\gamma^2\Gamma + (1-\gamma^2)I_N)dt \end{aligned}$$

Le processus $(B, \widehat{B})$ avec des valeurs dans $\mathbb{R}^{2N}$ est un processus de Wiener avec une covariance

$$\widetilde{\Gamma} = \begin{bmatrix} \Gamma & \gamma\Gamma \\ \gamma\Gamma & \gamma^2\Gamma + (1-\gamma^2)I_N \end{bmatrix}$$

Ainsi, la paire de processus $(B, \widehat{B})$ peut être facilement simulée en appliquant la factorisation de Cholesky de $\widetilde{\Gamma}$ à un mouvement Brownien standard avec des valeurs dans $\mathbb{R}^{2N}$. Nous utilisons les paramètres de modèle suivants

$$\forall i \quad \kappa_i = 2, \quad \sigma_i(0) = 0.04, \quad a_i = 0.04, \quad \nu_i = 0.01, \quad \gamma_i = -0.2$$

- **Intégration numérique**

Les hyperparamètres et l'architecture du réseau sont identiques aux cas précédents.

On a une bonne réduction de la variance (99.60%) comme vu dans le tableau 8 et la valeur est bien dans l'intervalle de confiance donnée par l'estimateur naif. Cette méthode donne de bons résultats malgré la complexité du modèle et la dimension des entrées qui a encore augmenté avec le couple de brownien à simuler. Le réseau apprend bien le payoff et on a une courbe d'apprentissage proche de celle de la figure 3.

Méthode	Call sur panier	±1.96*Std/M Call	Variance Call	Put sur le pire	±1.96*Std/M Put
MC naif	9.309893	0.850798	188.425825	12.836192	0.732351
Intégration numérique	9.009555	0.053281	0.738985	12.579633	0.037379
Méthode	Variance Put	Ratio Variance Call	Ratio Variance Put		
MC naif	139.613163	1.00	1.00		
Intégration numérique	0.363706	254.979153	383.862694		

Table 8 – Résultats Intégration numérique Volatilité stochastique

- Intégration analytique

Les hyperparamètres et l'architecture du réseau sont identiques aux cas précédents.

La valeur est en dehors de l'intervalle de confiance comme vu dans le tableau 9. Le réseau à une couche a du mal à apprendre le payoff avec la dimension qui a drastiquement augmenté. En augmentant le nombre de neurones dans la couche unique, on constate une évolution mais le trade-off avec le temps de calcul n'est pas forcément intéressant.

Méthode	Call sur panier	±1.96*Std/M Call	Variance Call	Put sur le pire	±1.96*Std/M Put
MC naif	9.309893	0.850798	188.425825	12.836192	0.732351
Intégration analytique	8.477737	0.026954	0.189121	10.431773	0.033167
Méthode	Variance Put	Ratio Variance Call	Ratio Variance Put		
MC naif	139.613163	1.00	1.00		
Intégration analytique	0.286349	996.324609	487.562649		

Table 9 – Résultats Intégration analytique Volatilité stochastique

6
CONCLUSION :

En résumé, notre étude a introduit et évalué deux nouvelles techniques décrites dans l'article intitulé "Automatic Control Variates for Option Pricing using Neural Networks". Ces méthodes exploitent des réseaux neuronaux pour créer des variables de contrôle, et nos résultats démontrent leur efficacité dans le contexte de l'évaluation d'options financières.

À travers plusieurs tests sur différents modèles, nous avons illustré la capacité de ces variables de contrôle à réduire de manière significative la variance des estimations, tout en maintenant une précision élevée dans les résultats.

Cependant, dans des contextes plus complexes où de nouvelles dimensions sont introduites, nos observations soulignent que les prédictions générées par les deux méthodes peuvent être en dehors de l'intervalle de confiance à 95% établi par l'estimateur naïf de Monte Carlo. D'où la nécessité de prudence dans de telles situations. Dans l'ensemble, la méthode d'intégration numérique a montré plus de résilience dans de grandes dimensions, sans doute grâce à son nombre de couches beaucoup plus élevé et l'efficacité de la réduction de dimensions. Pour la méthode analytique, nous devrions sans doute augmenter le nombre de neurones présents pour nos prochains tests afin de permettre au réseau de mieux apprendre le payoff sachant qu'il n'a qu'une couche cachée unique.

En somme, les méthodes de tarification utilisant les variables de contrôle se révèlent prometteuses pour améliorer la précision des estimations tout en offrant des réductions significatives de la variance, ouvrant ainsi de nouvelles perspectives dans le domaine de la finance quantitative.

-

RÉFÉRENCES

[1] Automatic Control Variates for Option Pricing using Neural Networks