Méthodes avancées en probabilités numériques - S1-24

Article étudié : Automatic Control Variates for Option Pricing using Neural Networks

À l'intention de :

Monsieur Vincent Lemaire

Par:

Aymène Jamjama

Enzo Renard

Date:

9 Janvier 2025

Contents

1 Introduction

1.1 Sujet à l'étude et objectif

Dans le cadre de ce projet, nous avons entrepris d'analyser, d'expliciter et de reproduire les méthodes proposées dans l'article « Automatic Control Variates for Option Pricing using Neural Networks » de Jérôme Lelong, Zineb El Filali Ech-Chafiq et Adil Reghai. Cet article, publié dans la revue Monte Carlo Methods and Applications, propose deux approches visant à améliorer la convergence des estimateurs Monte Carlo, largement utilisés en finance pour la valorisation et la couverture d'options.

Les auteurs y développent des techniques de réduction de variance en exploitant les réseaux de neurones, offrant ainsi des gains significatifs en précision et en efficacité. Plus précisément, l'article se concentre sur deux approches principales : (1) la réduction de la dimension effective des problèmes financiers de grande dimension et (2) la construction de variables de contrôle automatiques, calculables de manière exacte ou numériquement précise.

L'objectif de ce projet est double : d'une part, reproduire le plus fidèlement possible les résultats numériques obtenus dans l'article en utilisant Python ; d'autre part, approfondir certaines démonstrations théoriques, lever les zones d'ombre et apporter des explications détaillées sur les mécanismes sous-jacents des méthodes présentées.

Vous pouvez consulter l'article ici : Automatic Control Variates for Option Pricing using Neural Networks

1.2 Problématique

Soient B_i des mouvements browniens corrélés, T > 0. Considérons l'équation différentielle stochastique suivante, qui régit la diffusion des actifs sous-jacents S_i :

$$dS_t^i = \mu_i(t, S_t^i)dt + \sigma_i(t, S_t^i)dB_t^i$$

et soit g une fonction de payoff dépendant des processus S_t^i sur l'intervalle de temps $[0, T]$. On sait que le calcul du prix de g revient au calcul d'une espérance de la forme $\mathbb{E}(f(X))$ et grâce à la loi forte des grands nombres, nous avons l'estimateur suivant :

$$S_N = \frac{1}{N}\sum_{i=1}^{N} f(X_i),$$

avec X_i sont des échantillons iid de X et N un nombre suffisamment grand d'échantillons. L'estimateur S_N est donc l'estimateur de Monte Carlo.

On a aussi :

$$\mathrm{Var}(S_N) = \frac{Var(f(X))}{N}$$

la variance de l'estimateur est necessairement proportionnelle à celle de la fonction intégrée $f(X)$. Il est donc toujours intéressant de trouver une variable Y telle que :

$$\mathbb{E}(Y) = \mathbb{E}(f(X)) \quad \text{et} \quad \mathrm{Var}(Y) < \mathrm{Var}(f(X))$$

On introduit alors une variable de controle $h(X)$ telle que $\mathbb{E}(h(X))$ est explicite ou calculable efficacement avec une méthode déterministe et telle que $Var(h(X))$ soit suffisamment faible.

Supposons que $h(X)$ est L^2 et posons $Y = f(X) - h(X) + \mathbb{E}(h(X))$.

$$\begin{aligned}\mathrm{Var}(Y) &= \mathrm{Var}(f(X) - h(X) + \mathbb{E}[h(X)]) \\ &= \mathrm{Var}(f(X) - h(X)) + \mathbb{E}[h(X)]) \\ &= \mathrm{Var}(f(X)) + \mathrm{Var}(h(X)) - 2\mathrm{Cov}(f(X), h(X)).\end{aligned}$$

Car $\mathbb{E}[h(X)]$ est une constante.

Donc $2\text{Cov}(f(X), h(X)) = \text{Var}(f(X)) + \text{Var}(h(X)) - \text{Var}(Y)$ Or, on veut que $\text{Var}(Y) < \text{Var}(f(X))$, donc :

$$2\text{Cov}(f(X), h(X)) > \text{Var}(f(X)) + \text{Var}(h(X)) - \text{Var}(f(X))$$

ainsi,

$$\text{Cov}(f(X), h(X)) > \frac{1}{2}\text{Var}(h(X)).$$

Par conséquent, $\text{Var}(Y) < \text{Var}(f(X))$.

L'article que nous étudions propose alors de mettre en oeuvre deux approches :

1. La première consiste à réduire la dimension effective des problèmes de grande dimension. En pratique, de nombreux problèmes financiers présentent une faible dimension effective, c'est-à-dire que les variations de la fonction de payoff dépendent d'un petit sous-ensemble des variables d'entrée. Nous exploitons cette propriété en utilisant un réseau de neurones pour identifier ces variables principales et simplifier le problème d'intégration. Une fois cette réduction effectuée, l'intégration peut être réalisée efficacement à l'aide de techniques numériques rapides, telles que les quadratures de Gauss. Ainsi la nouvelle variable de controle construite aura une variance inferieure à la vraiance initiale.

2. La seconde approche vise à construire directement une variable de contrôle en entraînant un réseau de neurones H à apprendre la fonction de payoff f. Ce réseau est conçu de manière à être fortement corrélé au payoff et à permettre un calcul précis de son espérance, soit de manière analytique, soit numériquement.

2 Réduction de la dimension du réseau

2.1 Réseau de neurones

Tout d'abord, commençons par rappeler un résultat fondamental:

Théorème 1. *Soit $\sigma : \mathbb{R} \to \mathbb{R}$ une fonction d'activation* ReLU*, c'est-à-dire une fonction continue telle que*

$$\mathrm{ReLU}(x) \;=\; \max(0, x).$$

Soit K un compact *de $\mathbb{R}^n$, et soit*

$$f : K \to \mathbb{R}$$

une fonction continue*.*

Alors, pour tout $\varepsilon > 0$, il existe *un entier N, des vecteurs de poids $W_1, W_2, \ldots, W_N \in \mathbb{R}^n$, des biais $b_1, b_2, \ldots, b_N \in \mathbb{R}$ et des coefficients de sortie $a_1, a_2, \ldots, a_N \in \mathbb{R}$, tels que la fonction*

$$F(x) \;=\; \sum_{i=1}^{N} a_i \, \sigma\big(W_i^\top x + b_i\big)$$

satisfasse

$$\max_{x \in K} \big|\, F(x) \;-\; f(x) \big| \;<\; \varepsilon.$$

En d'autres termes, toute *fonction continue sur un compact de $\mathbb{R}^n$ peut être approximée* aussi fidèlement que l'on veut *par un réseau de neurones* à une couche cachée *utilisant une fonction d'activation sigmoïdale ou ReLU, pourvu que l'on autorise* suffisamment de neurones *(c'est-à-dire un N assez grand).*

L'article propose de poser ψ une fonction inconnue que l'on souhaite approximer, afin de démontrer les capacités d'approximation d'un réseau de neurones. Nous considérons $\psi : \mathbb{R}^p \to \mathbb{R}^q$, où dans notre cas $p = N$ et $q = 1$. Le réseau de neurones construit est un réseau feedforward contenant L couches, défini comme suit :

$$\begin{aligned} y = a_L(x) &\in \mathbb{R}^q \\ a_L(x) &= W_L a_{L-1}(x) + b_L \\ a_k(x) &= \sigma_a(W_k a_{k-1}(x) + b_k), \quad \forall k \in \{1, \ldots, L-1\} \\ a_0(x) &= x \in \mathbb{R}^p \end{aligned}$$

La couche d'entrée a_0 contient simplement l'entrée x du réseau. Les couches $a_1, \ldots, a_{L-1}$ sont des couches cachées qui appliquent des transformations non linéaires à leurs entrées via une fonction d'activation σ_a. Enfin, a_L est la couche de sortie qui fournit la sortie finale du réseau. Chaque couche k est paramétrée par une matrice de poids W_k et un vecteur de biais b_k.

L'entraînement du réseau consiste à trouver les meilleurs paramètres $\theta = \{W_k, b_k \mid k = 1, \ldots, L\}$ permettant d'approximer la fonction ψ. Pour cela, on définit une fonction de perte $l_\theta(\psi(x), a_L(x))$ mesurant l'erreur d'approximation. Le problème d'optimisation s'écrit alors :

$$\min_\theta l_\theta(\psi(x), a_L(x))$$

Pour résoudre ce problème, on utilise des méthodes de premier ordre telles que la descente de gradient stochastique, Adam ou RMSprop. Ces méthodes nécessitent de calculer le gradient de la fonction objectif par rapport aux paramètres W_k et b_k de chaque couche.

Le calcul des gradients peut être coûteux, notamment dans les réseaux profonds comportant de nombreuses couches. C'est pourquoi on utilise l'algorithme de **rétropropagation** (backpropagation), qui permet de calculer les gradients efficacement en ne considérant que ceux de la dernière couche, puis en les propageant en arrière pour obtenir les gradients des couches précédentes. Comme fait dans l'article, nous utilisons la distance euclidienne comme fonction de perte, c'est-à-dire :

$$l(z, y) = \|y - z\|^2$$

et l'algorithme Adam pour la minimisation de cette fonction de perte.

Nous avons également discuté de la descente par gradient, vous devez donc savoir que lorsque les gens décrivent un réseau comme « apprentissage », ils veulent dire trouver les poids et les biais qui minimisent une certaine fonction de coût.

2.2 Adam algorithm et back propagation

Nous utiliserons Adam qui est deja implementé en PyTorch. Il n'est pas inutile de prendre connaissance de la documentation PyTorch concernant l'implementation d'Adam que vous trouverez ci dessous :

Algorithm 1 Optimisation Adam avec régularisation L2 et correction de biais

Require: γ (taux d'apprentissage), β_1, β_2 (coefficients de moment), θ_0 (paramètres : $\{W, b\}$), $f(\theta)$ (fonction objectif), λ (régularisation L2), `amsgrad`, `maximize`

Initialiser $m_0 \leftarrow 0$ (premier moment), $v_0 \leftarrow 0$ (second moment), $\hat{v}_0^{\max} \leftarrow 0$
for $t = 1$ **to** ... **do**
  if `maximize` **then**
    $g_t \leftarrow -\nabla_\theta f(\theta_{t-1})$ {Gradient opposé pour maximisation}
  else
    $g_t \leftarrow \nabla_\theta f(\theta_{t-1})$ {Gradient pour minimisation}
  end if
  if $\lambda \neq 0$ **then**
    $g_t \leftarrow g_t + \lambda\theta_{t-1}$ {Ajout de la régularisation L2}
  end if
  $m_t \leftarrow \beta_1 m_{t-1} + (1-\beta_1) g_t$ {Premier moment}
  $v_t \leftarrow \beta_2 v_{t-1} + (1-\beta_2) g_t^2$ {Second moment}
  $\hat{m}_t \leftarrow \frac{m_t}{1-\beta_1^t}$ {Correction de biais pour m_t}
  $\hat{v}_t \leftarrow \frac{v_t}{1-\beta_2^t}$ {Correction de biais pour v_t}
  if `amsgrad` **then**
    $\hat{v}_t^{\max} \leftarrow \max(\hat{v}_t^{\max}, \hat{v}_t)$
    $\theta_t \leftarrow \theta_{t-1} - \gamma \frac{\hat{m}_t}{\sqrt{\hat{v}_t^{\max}} + \epsilon}$ {Mise à jour des paramètres avec `amsgrad`}
  else
    $\theta_t \leftarrow \theta_{t-1} - \gamma \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$ {Mise à jour classique des paramètres}
  end if
end for
return θ_t

L'algorithme Adam est une méthode d'optimisation adaptative largement utilisée dans l'apprentissage des réseaux de neurones. Ce pseudo-code présente une version améliorée d'Adam incluant :

- une **régularisation L2** pour éviter le sur-apprentissage ;
- une option `Amsgrad` permettant une meilleure stabilisation des mises à jour ;
- une **correction de biais** des moments pour améliorer la convergence au début des itérations.

Étapes détaillées de l'algorithme

1. Initialisation

Avant de commencer les itérations, on initialise les moments et les variables nécessaires :

- $m_0 \leftarrow 0$: premier moment (moyenne exponentielle des gradients) ;
- $v_0 \leftarrow 0$: second moment (moyenne exponentielle des carrés des gradients) ;
- $\hat{v}_0^{\max} \leftarrow 0$: utilisé uniquement si l'option `Amsgrad` est activée.

2. Boucle d'optimisation principale

La boucle d'optimisation se déroule sur $t = 1, 2, \ldots$ jusqu'à un certain nombre d'itérations.

Étape 1 : Calcul du gradient

Si l'objectif est de maximiser la fonction $f(\theta)$, on calcule le gradient opposé :

$$g_t = -\nabla_\theta f(\theta_{t-1})$$

Sinon, on calcule le gradient classique :

$$g_t = \nabla_\theta f(\theta_{t-1})$$

Étape 2 : Ajout de la régularisation L2

Si la régularisation L2 est activée (*i.e.* $\lambda \neq 0$), on ajoute un terme de pénalisation au gradient :

$$g_t \leftarrow g_t + \lambda\theta_{t-1}$$

Cela permet de limiter la taille des paramètres et d'éviter le sur-apprentissage.

Étape 3 : Mise à jour des moments

L'algorithme Adam utilise deux moments pour adapter dynamiquement le taux d'apprentissage de chaque paramètre (poids et biais) :

1. **Premier moment (m_t)** : moyenne exponentielle des gradients.
2. **Second moment (v_t)** : moyenne exponentielle des carrés des gradients.

Ces moments permettent d'accélérer la convergence tout en stabilisant la mise à jour des paramètres.

—

Calcul du premier moment (m_t)

Le premier moment m_t est une moyenne pondérée exponentielle des gradients. Il est calculé à chaque itération selon la formule suivante :

$$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1)g_t$$

où :

- g_t est le gradient de la fonction de perte par rapport aux paramètres (poids ou biais) à l'itération t ;
- $\beta_1 \in [0, 1)$ est un hyperparamètre qui contrôle la contribution des gradients passés (typiquement, $\beta_1 = 0.9$).

Cette formule donne une sorte de **moyenne lissée** des gradients : les gradients récents ont une plus grande importance que les anciens, mais les anciens ne sont jamais complètement oubliés.

—

Calcul du second moment (v_t)

Le second moment v_t est une moyenne pondérée exponentielle des carrés des gradients. Il est calculé comme suit :

$$v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2)g_t^2$$

où :

- g_t^2 représente le carré du gradient (élément par élément si le gradient est un vecteur) ;
- $\beta_2 \in [0, 1)$ est un hyperparamètre qui contrôle la contribution des carrés des gradients passés (typiquement, $\beta_2 = 0.999$).

Cette formule capture la **variance des gradients**. Si les gradients varient beaucoup, cela signifie qu'il y a de grandes oscillations dans la descente, et il est alors préférable de réduire le taux d'apprentissage pour éviter des mises à jour trop brusques.

—

Pourquoi utiliser ces moments ?

1. **Le premier moment (m_t)** aide à accélérer la convergence en **lissant les gradients**. Au lieu d'utiliser directement le gradient brut, on utilise une version lissée qui représente mieux la tendance générale de la descente.
2. **Le second moment (v_t)** permet d'**adapter le taux d'apprentissage** en fonction de la variance des gradients :
   - Si les gradients varient beaucoup, cela signifie qu'il y a des oscillations importantes, donc on réduit le pas de mise à jour.
   - Si les gradients varient peu, cela signifie qu'on est proche d'un minimum, donc on peut utiliser un pas de mise à jour plus grand.

—

Poids et biais : même principe de mise à jour

Dans un réseau de neurones, chaque couche possède :

- des **poids** W utilisés pour calculer les combinaisons linéaires des entrées ;
- des **biais** b ajoutés à ces combinaisons pour ajuster les résultats.

Les gradients sont calculés pour les deux types de paramètres, et Adam applique la mise à jour suivante aux poids W et aux biais b :

$$W_t \leftarrow W_{t-1} - \gamma \frac{m_t}{\sqrt{v_t} + \epsilon} \quad \text{et} \quad b_t \leftarrow b_{t-1} - \gamma \frac{m_t}{\sqrt{v_t} + \epsilon}$$

où :

- γ est le taux d'apprentissage ;
- ϵ est une petite constante pour éviter la division par zéro ;
- m_t et v_t sont respectivement le premier et le second moment pour les poids ou les biais.

La régularisation L2 peut également être appliquée aux **poids**, mais en général elle n'est pas appliquée aux biais, car leur effet est moindre sur la complexité du modèle.

—

Étape 4 : Correction des biais des moments

Pour éviter les biais initiaux dus à l'initialisation des moments à zéro, on applique une correction de biais :

$$\hat{m}_t = \frac{m_t}{1 - \beta_1^t}, \quad \hat{v}_t = \frac{v_t}{1 - \beta_2^t}$$

Étape 5 : Mise à jour des paramètres

Deux cas sont possibles selon que l'option `Amsgrad` est activée ou non.

Avec Amsgrad : on utilise le maximum des moments passés pour la mise à jour :

$$\hat{v}_t^{\max} \leftarrow \max(\hat{v}_t^{\max}, \hat{v}_t)$$

$$\theta_t \leftarrow \theta_{t-1} - \gamma \frac{\hat{m}_t}{\sqrt{\hat{v}_t^{\max}} + \epsilon}$$

Sans Amsgrad : mise à jour classique d'Adam :

$$\theta_t \leftarrow \theta_{t-1} - \gamma \frac{\hat{m}_t}{\sqrt{\hat{v}_t} + \epsilon}$$

où :

- γ est le taux d'apprentissage (*learning rate*) ;
- ϵ est une petite constante ajoutée pour éviter la division par zéro.

3. Retour des paramètres optimisés

À la fin des itérations, l'algorithme retourne les paramètres optimisés θ_t, c'est-à-dire les poids W et les biais b du réseau de neurones :

Retourner θ_t

L'optimiseur **Adam** est une méthode d'optimisation adaptative qui utilise les gradients de la fonction de perte pour mettre à jour les paramètres du modèle. Il est important de noter que **Adam ne dépend pas directement de la fonction de perte utilisée**. Il fonctionne avec n'importe quelle fonction de perte, tant qu'il peut accéder aux gradients calculés par la rétropropagation.

Dans ce document, nous allons expliquer comment utiliser **Adam** avec la fonction de perte **Mean Squared Error (MSE)** dans une tâche de régression simple.

L'optimiseur Adam met à jour à la fois les poids et les biais d'un réseau de neurones en utilisant les mêmes principes :

- Il calcule les gradients des poids et des biais via la rétropropagation.
- Il met à jour ces paramètres à l'aide des moments m_t et v_t pour stabiliser et accélérer la convergence.

Les poids jouent un rôle majeur dans l'apprentissage des représentations, tandis que les biais permettent d'ajuster finement les sorties de chaque neurone. On met à jour les deux types de moments :

Premier moment : moyenne exponentielle des gradients

$$m_t \leftarrow \beta_1 m_{t-1} + (1 - \beta_1) g_t$$

où β_1 contrôle la contribution des gradients passés (typiquement, $\beta_1 = 0.9$).

Second moment : moyenne exponentielle des carrés des gradients

$$v_t \leftarrow \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$$

où β_2 contrôle la contribution des carrés des gradients passés (typiquement, $\beta_2 = 0.999$).

2.2.1 Exemple simple d'utilisation de Adam avec MSE

Voici un exemple en `Python` utilisant la bibliothèque `PyTorch` pour entraîner un modèle de régression linéaire avec l'optimiseur Adam et la fonction de perte MSE.

```python
import torch
import torch.nn as nn
import torch.optim as optim

# Definir un modele simple (regression lineaire)
model = nn.Linear(in_features=1, out_features=1)

# Definir la fonction de perte : Mean Squared Error (MSE)
criterion = nn.MSELoss()

# Definir l'optimiseur Adam
lr = 0.01  # Taux d'apprentissage
optimizer = optim.Adam(model.parameters(), lr=lr)

# Donnees d'entrainement
x = torch.tensor([[1.0], [2.0], [3.0], [4.0]])
y = torch.tensor([[2.0], [4.0], [6.0], [8.0]])

# Boucle d'entrainement
num_epochs = 1000
for epoch in range(num_epochs):
    optimizer.zero_grad()           # Reinitialiser les gradients
    y_pred = model(x)               # Prediction du modele
    loss = criterion(y_pred, y)     # Calcul de la perte
    loss.backward()                 # Calcul des gradients par retropropagation
    optimizer.step()                # Mise a jour des parametres avec Adam

    if (epoch + 1) % 100 == 0:
        print(f'Epoque [{epoch + 1}/{num_epochs}], Perte : {loss.item():.4f}')
```

1. **Définition du modèle** : `nn.Linear` crée un modèle de régression linéaire simple avec une entrée et une sortie.
2. **Définition de la fonction de perte** :

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

   La fonction de perte MSE (Mean Squared Error) mesure l'écart quadratique moyen entre les prédictions du modèle et les vraies valeurs.
3. **Définition de l'optimiseur Adam** : `optim.Adam` est initialisé avec les paramètres du modèle (`model.parameters()`) et un taux d'apprentissage `lr`.
4. **Boucle d'entraînement** :
   (a) On réinitialise les gradients avec `optimizer.zero_grad()`.
   (b) On effectue une prédiction avec le modèle : `y_pred = model(x)`.
   (c) On calcule la perte entre les prédictions et les vraies valeurs : `loss = criterion(y_pred, y)`.
   (d) On effectue la rétropropagation pour calculer les gradients : `loss.backward()`.
   (e) On met à jour les paramètres du modèle avec `optimizer.step()`.

Résumé

- L'algorithme Adam permet une mise à jour adaptative des paramètres en utilisant deux moments : la moyenne exponentielle des gradients et celle des carrés des gradients.
- La régularisation L2 (*weight decay*) aide à éviter le sur-apprentissage en limitant la taille des paramètres.

- L'option `Amsgrad` assure une meilleure stabilité en empêchant des mises à jour trop brusques lorsque les moments sont trop petits.
- La correction des biais des moments améliore la convergence, surtout au début de l'entraînement.

Cet algorithme est particulièrement adapté aux problèmes complexes comme le **pricing d'options** où la fonction objectif peut avoir une variance importante et des dimensions élevées.

Dans le cas d'un réseau de neurones, les paramètres du modèle, notés θ, comprennent à la fois :

- les **poids** W de chaque couche ;
- les **biais** b associés à chaque neurone de chaque couche.

L'algorithme Adam met à jour ces deux types de paramètres de manière indépendante mais selon le même principe, en utilisant les gradients calculés par la rétropropagation.

—

2.3 Explication : Network Dimension Reduction

1. Problème de départ : Approximation d'une fonction à partir de variables normales standard

On considère une fonction f qui dépend de N variables normales standard indépendantes notées $Z_1, Z_2, \ldots, Z_N$. L'objectif est de réduire la dimension du problème en ne conservant que $n \ll N$ directions principales qui expliquent la majeure partie de la variance de f.

On cherche donc n nouvelles variables normalisées $\tilde{Z}_i$ définies par :

$$\tilde{Z}_i = \sum_{k=1}^{N} u_{ik} Z_k, \quad \forall i = 1, \ldots, n$$

avec la contrainte suivante :

$$\sum_{k=1}^{N} u_{ik}^2 = 1, \quad \forall i = 1, \ldots, n$$

Cette contrainte garantit que les nouvelles variables $\tilde{Z}_i$ sont normalisées, c'est-à-dire que $\mathrm{Var}(\tilde{Z}_i) = 1$.

2. Construction du réseau neuronal pour la réduction de dimension

Le réseau neuronal est construit de manière à prédire la fonction de payoff f à partir des variables normales $Z_1, \ldots, Z_N$. Il est constitué de plusieurs couches cachées, divisées en deux parties :

1. **Cellule de réduction de dimension :**
   - Une couche cachée non activée avec exactement n neurones.
   - Cette couche applique une transformation linéaire sans biais de la forme WZ, où W est une matrice de taille $n \times N$ contenant les poids du réseau.
   - La sortie de cette couche correspond aux nouvelles variables $\tilde{Z}_i$ définies précédemment :
   $$\tilde{Z} = WZ$$
2. **Reconstruction du payoff :**
   - La seconde partie du réseau contient plusieurs couches cachées activées permettant de reconstruire la fonction de payoff f à partir des variables réduites $\tilde{Z}$.

3. Extraction des directions principales

Une fois le réseau neuronal entraîné, on obtient la matrice des poids W de la première couche, qui donne les coefficients u_{ik} des nouvelles variables $\tilde{Z}_i$ sous la forme :

$$\tilde{Z}_i = \sum_{k=1}^{N} u_{ik} Z_k, \quad \text{où } W = \begin{pmatrix} u_{11} & u_{12} & \ldots & u_{1N} \\ u_{21} & u_{22} & \ldots & u_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ u_{n1} & u_{n2} & \ldots & u_{nN} \end{pmatrix}$$

On extrait alors les poids W et on se débarrasse du reste du réseau.

4. Complétion de l'espace original

Pour compléter l'espace de dimension N, on cherche une matrice V telle que :

$$\mathrm{Im}(U) \oplus \mathrm{Im}(V) = \mathbb{R}^N$$

Autrement dit, V doit fournir une base orthogonale complémentaire à U. Cela implique la condition suivante :

$$UV^T = 0$$

Calcul de V :

1. La matrice U est de taille $n \times N$. On construit la matrice augmentée :

   $$(U \mid I_N)$$

   où I_N est la matrice identité de taille $N \times N$.

2. On applique une élimination de Gauss sur cette matrice pour obtenir une matrice de la forme :

   $$(A \mid B)$$

   Les colonnes non nulles de B correspondant aux colonnes nulles de A forment une base de $\ker(U)$.

3. Si cette base est de rang $N - n$, alors les colonnes de B constituent la matrice V.

Cependant, nous avons choisi de ne pas faire l'elimination de Gauss mais la décomposition en valeurs singulières (SVD)

Définition de la SVD (Singular Value Decomposition) Soit une matrice M de taille $m \times n$. La décomposition en valeurs singulières (SVD) de M s'écrit :

$$M \;=\; U\,\Sigma\,V^T,$$

où :

- $U \in \mathbb{R}^{m\times m}$ est une matrice orthonormée (i.e. $U^TU = I_m$).
- $\Sigma \in \mathbb{R}^{m\times n}$ est une matrice *diagonale par blocs* dont les éléments diagonaux (appelés *valeurs singulières*) sont réels, non-négatifs et ordonnés par taille décroissante.
- $V \in \mathbb{R}^{n\times n}$ est aussi une matrice orthonormée ($V^TV = I_n$).

En pratique, si $\min(m, n) = r$ est le rang de M, on peut écrire :

$$\Sigma \;=\; \begin{bmatrix} \sigma_1 & & 0 \\ & \ddots & \\ 0 & & \sigma_r \\ & & 0 \end{bmatrix},$$

où $\sigma_1 \geq \sigma_2 \geq \cdots \geq \sigma_r > 0$. Les colonnes de U (resp. de V) sont alors les vecteurs propres de MM^T (resp. de M^TM).

Construction de la matrice V orthogonale à U Soit $U \in \mathbb{R}^{n\times N}$. Nous souhaitons déterminer un ensemble de vecteurs orthonormés formant une base pour le sous-espace orthogonal à U. L'idée est de construire une *matrice augmentée* puis de lui appliquer la SVD.

Matrice augmentée En notation purement mathématique, définissons la matrice

$$A \;=\; \begin{bmatrix} U^T & I_N \end{bmatrix}^\top \;\in\; \mathbb{R}^{(n+N)\times N},$$

où $U^T \in \mathbb{R}^{N\times n}$ et $I_N \in \mathbb{R}^{N\times N}$. Concrètement :

$$\begin{bmatrix} U^T & I_N \end{bmatrix} \in \mathbb{R}^{N\times(n+N)}, \quad \text{puis on transpose pour obtenir} \quad A = \begin{bmatrix} U^T & I_N \end{bmatrix}^\top \in \mathbb{R}^{(n+N)\times N}.$$

Décomposition SVD On applique la SVD à la matrice A. On obtient

$$A \;=\; U_{\text{svd}}\,\Sigma\,V_{\text{svd}}^{T},$$

où $U_{\text{svd}} \in \mathbb{R}^{(n+N)\times N}$, $\Sigma \in \mathbb{R}^{N\times N}$ et $V_{\text{svd}} \in \mathbb{R}^{N\times N}$.

En partitionnant V_{svd}^{T} sous la forme

$$V_{\text{svd}}^{T} \;=\; \begin{bmatrix} V_1^T & V_2^T \end{bmatrix} \quad \text{où} \quad \begin{bmatrix} V_1^T & V_2^T \end{bmatrix} \in \mathbb{R}^{N\times N},$$

avec $\dim(V_1) = n$ et $\dim(V_2) = N - n$, on repère que les n premières colonnes de V_{svd}^{T} (donc V_1) sont associées à l'espace engendré par U, et les $(N-n)$ dernières colonnes (donc V_2) à l'orthogonal.

Extraction de la partie orthogonale La matrice retournée V dans notre algorithme est choisie comme

$$V \;=\; \begin{bmatrix} V_2^T \end{bmatrix}^{\top} \;=\; V_2 \;\in\; \mathbb{R}^{(N-n)\times N}.$$

On vérifie alors que chacune des $(N-n)$ lignes de V (*ou* les $(N-n)$ colonnes de V^T) constitue un vecteur orthonormé dans $\mathbb{R}^N$, et que ces vecteurs sont *tous orthogonaux* à l'espace engendré par les lignes ou les colonnes de U.

En d'autres termes, V forme une base orthonormale du sous-espace orthogonal à celui engendré par U.

Conclusion : cette procédure d'**extraction de composantes orthogonales** à U se résume ainsi :

1. Construire la matrice $A = \begin{bmatrix} U^T \mid I_N \end{bmatrix}^{\top}$.
2. Appliquer la décomposition SVD à A.
3. Séparer les premières colonnes (liées à l'espace de U) et les dernières (liées à l'orthogonal).
4. Renvoyer la matrice $V_2 \in \mathbb{R}^{(N-n)\times N}$ qui contient les vecteurs orthonormés du sous-espace orthogonal.

5. Simulation des variables réduites et complémentaires

Une fois U et V déterminés, on simule les nouvelles variables :

$$\tilde{Z} \sim \mathcal{N}_n(0, UU^T), \quad \tilde{Z}^{\perp} \sim \mathcal{N}_{N-n}(0, VV^T)$$

Le prix de l'option est donné par l'espérance $\mathbb{E}[f(Z)]$. On décompose cette espérance de la manière suivante :

$$\begin{aligned}
\mathbb{E}(f(Z)) &= \mathbb{E}\left(\mathbb{E}(f(Z) \mid \tilde{Z})\right) \\
&= \mathbb{E}\left(\mathbb{E}(f(Z) \mid \tilde{Z})\right) + f\left(\mathbb{E}(Z \mid \tilde{Z})\right) - f\left(\mathbb{E}(Z \mid \tilde{Z})\right) \\
&= \mathbb{E}\left(f\left(\mathbb{E}(Z \mid \tilde{Z})\right) + \left(f(Z) - f\left(\mathbb{E}(Z \mid \tilde{Z})\right)\right)\right) \\
&= \mathbb{E}\left(f\left(\mathbb{E}(Z \mid \tilde{Z})\right)\right) + \mathbb{E}\left(f(Z) - f\left(\mathbb{E}(Z \mid \tilde{Z})\right)\right)
\end{aligned}$$

On appliquera Monte Carlo pour la seconde esperance et on utilise la variable $f(\mathbb{E}(Z|\tilde{Z}))$ comme une **variables de contrôle**. Comme elle ne dépend que des nouvelles variables réduites $\tilde{Z}$ de petite dimension, on peut calculer l'intégrale numériquement de manière précise avec par exemple les **quadratures de Gauss**

6. Conclusion

Cette approche permet de réduire la dimension effective du problème en identifiant les directions principales qui concentrent la variance de la fonction de payoff. La réduction de dimension ainsi obtenue permet d'améliorer significativement l'efficacité des simulations Monte Carlo en utilisant des variables de contrôle basées sur des intégrales de faible dimension.

3 Variable de contrôle automatique

Contexte

Cette section présente deux méthodes d'intégration permettant de calculer l'espérance $\mathbb{E}(H(Z))$, où $H(Z)$ est une fonction de contrôle construite à partir d'un réseau de neurones.

Dans le cadre de la réduction de variance pour le pricing d'options via la méthode de Monte Carlo, l'idée est de construire une fonction de contrôle $H(Z)$ qui soit corrélée à la fonction de payoff $f(Z)$. Cette fonction de contrôle permet de réduire la variance de l'estimation Monte Carlo de $\mathbb{E}[f(Z)]$ en considérant une nouvelle variable définie par :

$$Y = f(Z) - H(Z) + \mathbb{E}[H(Z)]$$

Si $H(Z)$ est bien choisie, sa variance est faible et sa corrélation avec $f(Z)$ est élevée, ce qui garantit que la variance de Y est réduite par rapport à celle de $f(Z)$.

3.1 Explication de la méthode d'intégration numérique

Dans l'architecture du réseau de neurones, $\tilde{H}$ représente la partie du réseau située après la première couche cachée. Plus précisément, $\tilde{H}$ regroupe toutes les couches restantes du réseau de neurones, à partir de la sortie de la première couche $a_1(Z) = WZ$ jusqu'à la sortie finale du réseau $H(Z)$. Ainsi, on peut écrire :

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

où $a_1(Z) = WZ$ correspond à la transformation linéaire appliquée par la première couche cachée (couche de réduction de dimension), et $\tilde{H}$ est la fonction composée des couches suivantes.

Étapes du calcul de $\mathbb{E}(H(Z))$:

1. Architecture du réseau de neurones : Comme dit au dessus le réseau $H(Z)$ est représenté comme une composition de fonctions :

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

où $a_1(Z)$ est la première couche cachée du réseau définie par une transformation linéaire :

$$a_1(Z) = WZ$$

où W est une matrice de poids. La sortie de cette couche, notée $\tilde{Z} = WZ$, suit une loi normale $\mathcal{N}(0, WW^T)$ de faible dimension, car la matrice W effectue une réduction de dimension sur l'entrée Z.

2. Estimation numérique de l'espérance : Puisque la dimension de $\tilde{Z}$ est faible, l'espérance $\mathbb{E}[\tilde{H}(\tilde{Z})]$ peut être estimée numériquement à l'aide de méthodes d'intégration rapide et précise, telles que les polynômes de Gauss-Hermite. Cette approche est particulièrement efficace lorsque la dimension effective du problème est suffisamment petite, rendant l'intégration numérique plus performante que la simulation Monte Carlo classique.

Cette méthode repose sur l'idée que la première couche du réseau de neurones sert à réduire la dimension effective de l'entrée. Une fois cette réduction effectuée, l'espérance de la sortie du réseau peut être calculée avec une grande précision grâce à des techniques d'intégration numérique. Cela permet de construire une fonction de contrôle efficace pour réduire la variance de l'estimation Monte Carlo du prix de l'option.

3.2 Explication de la méthode d'intégration analytique

L'idée est de concevoir une architecture de réseau de neurones suffisamment simple pour que l'espérance du réseau puisse être calculée analytiquement, ce qui rend la méthode à la fois rapide et précise. L'architecture proposée comprend :

1. **Une couche linéaire suivie d'une activation ReLU** :
   Le réseau applique la fonction d'activation ReLU, définie par :

$$\mathrm{ReLU}(x) = \max(0, x) = x^+$$

2. **Formulation du réseau** :
   Le réseau H est exprimé sous la forme :

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

   où W_1 et W_2 sont des matrices de poids, et b_1 et b_2 sont des vecteurs de biais.

3. **Espérance analytique** :
   L'objectif est de calculer l'espérance $E(H(Z))$ de manière exacte. On note que :

$$W_1 Z + b_1 = (\sigma_i Y_i + \mu_i)_{i=1}^n$$

   où $Y_i \sim \mathcal{N}(0,1)$ sont des variables aléatoires normales standard, et σ_i et μ_i sont respectivement l'écart-type et la moyenne des composantes de $W_1 Z + b_1$.

   Ensuite, on utilise une formule fermée pour l'espérance d'une variable normale tronquée par zéro :

$$E\left((\sigma Y + \mu)^+\right) = \sigma\phi\left(-\frac{\mu}{\sigma}\right) + \mu\left(1 - \Phi\left(-\frac{\mu}{\sigma}\right)\right)$$

   où ϕ et Φ désignent respectivement la densité et la fonction de répartition de la loi normale standard.

4. **Formule finale** :
   En combinant les résultats pour toutes les composantes, on obtient une formule analytique exacte pour $E(H(Z))$:

$$E(H(Z)) = W_2\left(\frac{\sigma_i}{\sqrt{2\pi}}\exp\left(-\frac{\mu_i^2}{2\sigma_i^2}\right) + \mu_i\left(1 - \Phi\left(-\frac{\mu_i}{\sigma_i}\right)\right)\right) + b_2$$

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

En effet, Soit la fonction d'activation ReLU :

$$\mathrm{ReLU}(x) = \max(0, x) = x_+$$

On pose H le réseau, qui s'écrit alors :

$$H(Z) = Z_2 = W_2 Z_1 + b_2 = W_2(W_1 Z + b_1)^+ + b_2$$

Ainsi,

$$\mathbb{E}(H(Z)) = W_2\mathbb{E}\left((W_1 Z + b_1)^+\right) + b_2$$

Mais,

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

avec $\mu_i = (b_1)_i$ et $\sigma_i^2 = \sum_{j=1}^N (W_1)_{ij}^2$ et $Y_i \sim \mathcal{N}(0,1)$. Nous devons alors simplement calculer $\mathbb{E}\left((\sigma Y + \mu)^+\right)$. En effet,

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

$$= \frac{\sigma}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{\mu^2}{\sigma^2}\right) + \mu\left(1 - \mathcal{N}\left(-\frac{\mu}{\sigma}\right)\right),$$

où $\mathcal{N}$ est la fonction de répartition de la loi normale standard. Finalement,

$$\mathbb{E}(H(Z)) = W_2 \begin{pmatrix} \frac{\sigma_1}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{\mu_1^2}{\sigma_1^2}\right) + \mu_1\left(1 - \mathcal{N}\left(-\frac{\mu_1}{\sigma_1}\right)\right) \\ \vdots \\ \frac{\sigma_n}{\sqrt{2\pi}} \exp\left(-\frac{1}{2}\frac{\mu_n^2}{\sigma_n^2}\right) + \mu_n\left(1 - \mathcal{N}\left(-\frac{\mu_n}{\sigma_n}\right)\right) \end{pmatrix} + b_2.$$

4 Présentation du modèle de Black and Scholes pour plusieurs actifs

Avant de prèsenter les differents modèles utilisés nous fixons arbitrairement ces paramètres dans notre notebook:

- $S_0^i = 100.0$, $\forall i$: Le prix initial de l'actif i au temps 0.
- $K = 100.0$: Le prix d'exercice (strike) de l'option.
- $\omega_i = \frac{1}{\text{nbsj}}$, $\forall i$: La pondération de chaque actif dans le panier, où nbsj représente le nombre total d'actifs.
- $T = 1$ année : L'échéance de l'option (1 an).
- $d = 10$: Le nombre de dates d'observation pour l'option asiatique.
- $G = 100.0$: Le seuil utilisé pour l'option binaire (digit).
- $\sigma_i = 0.4$, $\forall i$: La volatilité constante de chaque actif i.
- $\rho_{ij} = 0.75$, $\forall i, j$: La corrélation entre les actifs i et j, avec $\rho_{ij} \in (-1, 1)$.
- nbsj $= 10$: Le nombre total d'actifs sous-jacents dans le panier.

4.1 Payoff

Voici les differents payoff que nous allons implementer :

- Une option call sur un panier : $\max(0.0, \sum_i \omega_i S_T^i - K)$
- Une option put sur le pire sous-jacent : $\max(0.0, K - \min_i(S_T^i))$
- Une option asiatique arithmétique : $\max(0.0, \frac{1}{d}\sum_i \sum_j \omega_j S_{t_i}^j - K)$
- Une option binaire (digit) sur le panier : $G1_{\sum_i \omega_i S_T^i \geq K}$

4.2 Équation du modèle

Le modèle de Black-Scholes peut être réécrit sous la forme :

$$dS_t^i = r\, S_t^i\, dt + \sigma_i\, S_t^i\, dB_t^i,$$

où B est un processus de Wiener tel que :

$$d\langle B^i, B^j \rangle_t = \rho_{ij}\, dt, \quad \text{avec } \rho_{ij} \in (-1, 1).$$

4.3 Matrice de corrélation

Le processus B prend ses valeurs dans $\mathbb{R}^N$ et est un processus de Wiener dont la matrice de corrélation Γ est donnée par :

$$\Gamma = \begin{pmatrix} 1 & \rho_{12} & \cdots & \rho_{1N} \\ \rho_{21} & 1 & \cdots & \rho_{2N} \\ \vdots & \vdots & \ddots & \vdots \\ \rho_{N1} & \rho_{N2} & \cdots & 1 \end{pmatrix}, \quad \text{avec } \rho_{ij} \in (-1, 1).$$

4.4 Simulation des trajectoires

1. Mise à jour des prix des actifs La dynamique des prix des actifs est donnée par :

$$S_{t+1}^i \;=\; S_t^i \;+\; r\, S_t^i\, \Delta t \;+\; \sigma_i\, \sqrt{\Delta t}\; S_t^i\, Z_t^i,$$

$$S_{t+1}^i = S_t^i \exp\left(\left(\mu - \frac{1}{2}\sigma^2\right)\Delta t + \sigma\sqrt{\Delta t} Z_t^i\right)$$

où Z_t^i sont des chocs gaussiens standardisés indépendants.

2. Application de la corrélation Les chocs browniens corrigés Z_t sont obtenus en appliquant la matrice de corrélation Γ (via sa décomposition de Cholesky) aux chocs standards dW :

$$Z_t \;=\; \Gamma_{\text{Cholesky}} \;\cdot\; dW,$$

où dW est un vecteur de variables normales standard (gaussiennes) représentant les chocs pour les prix des actifs.

4.4.1 Analyse de nos résultats

Nous utilisons les mêmes indicateurs évalués dans l'article :

- **Prix**: Estimation du prix de l'option par chaque méthode.
- **Var ratio**: Rapport des variances comparé à la méthode de Monte Carlo standard.
- **Cost**: Coût de calcul (lié à la variance et au temps de calcul).
- **Speed-up**: Amélioration relative de la vitesse de calcul.

Les résultats obtenus sont présentés dans le tableau ci-dessous.

Table 1: Résultats des simulations sous le modèle de Black-Scholes

Métrique	**Basket**	**Asian**	**A_digit**	**Worst_of**
Monte Carlo Prix	15.9517	8.4467	48.7090	21.0686
Méthode 1 Prix	15.9254	8.4411	48.7171	21.0442
Méthode 1 Var ratio	1.0256	1.0273	0.9939	1.0158
Méthode 1 Cost	12.6634	7.5234	42.2075	7.2348
Méthode 1 Speed-up	91.0944	39.8043	96.7163	79.0574
Méthode 2 Prix	15.9333	8.3990	48.6990	21.0943
Méthode 2 Var ratio	242.5370	332.8014	16.2567	20.5839
Méthode 2 Cost	1.7961	4.1422	3.5084	2.3694
Méthode 2 Speed-up	326.5299	36.7455	615.6546	127.5582
Variable de contrôle Prix	15.9452	8.4338	48.7211	21.0615
Variable de contrôle Var ratio	169.9899	128.8463	10.0625	15.0132
Variable de contrôle Cost	1.6414	0.7574	3.1690	0.6619
Variable de contrôle Speed-up	357.9295	201.9114	705.9095	464.4460

Analyse

Méthode 1 : Réduction de dimension : utilisation de Z

- **Prix**: Les prix obtenus sont très proches de ceux de Monte Carlo pour toutes les options, ce qui valide la précision de cette méthode.
- **Var ratio**: Les ratios de variance sont proches de 1, ce qui indique une légère réduction de la variance.
- **Cost**: Le coût est significativement réduit, notamment pour l'option Worst_of.

- **Speed-up**: La méthode offre un gain de vitesse important, particulièrement pour Basket et A_digit.

Méthode 2 : Réduction de dimension : utilisation d'une variable de contrôle

- **Prix**: Les prix restent cohérents avec Monte Carlo, avec de légères variations.
- **Var ratio**: Une réduction massive de la variance est observée, notamment pour Basket (242.5) et Asian (332.8).
- **Cost**: Le coût est significativement réduit, en particulier pour Basket et Worst_of.
- **Speed-up**: Cette méthode surpasse Méthode 1 en termes de vitesse, atteignant des facteurs de gain impressionnants (par exemple, 615.65 pour A_digit).

Variable de contrôle

- **Prix**: Les résultats sont très proches de Monte Carlo, avec une très faible erreur.
- **Var ratio**: Les ratios de variance sont élevés, ce qui confirme une réduction importante de la variance.
- **Cost**: Cette méthode minimise le coût, surtout pour Asian et Worst_of.
- **Speed-up**: Les gains de vitesse sont les plus élevés parmi les méthodes testées, en particulier pour A_digit (705.91) et Worst_of (464.45).

Conclusion

Parmi les méthodes testées :

- **Méthode 1** offre une bonne performance globale, avec un compromis entre coût et précision.
- **Méthode 2** est la meilleure en termes de réduction de variance et de vitesse, mais peut être moins précise pour certaines options.
- **Variable de contrôle** est idéale pour réduire la variance et le coût tout en maintenant une excellente précision.

4.4.2 Comparaison avec les résultats attendu

Table 2: Comparaison des résultats simulés et attendus sous le modèle de Black-Scholes

Métrique	**Basket**	**Asian**	**A_digit**	**Worst_of**
Monte Carlo Prix	15.9517	8.4467	48.7090	21.0686
Monte Carlo Attendu	15.95	9.75	46.00	26.48
Écart	**+0.0017**	**-1.3033**	**+2.7090**	**-5.4114**
Méthode 1 Prix	15.9254	8.4411	48.7171	21.0442
Méthode 1 Attendu	16.30	9.86	49.51	26.62
Écart	**-0.3746**	**-1.4189**	**-0.7929**	**-5.5758**
Méthode 1 Var ratio	1.0256	1.0273	0.9939	1.0158
Méthode 1 Var ratio Attendu	336.12	166.79	15.28	4.08
Écart	**-335.0944**	**-165.7627**	**-14.2861**	**-3.0642**
Méthode 1 Cost	12.6634	7.5234	42.2075	7.2348
Méthode 1 Cost Attendu	24.23	10.22	19.34	23.76
Écart	**-11.5666**	**-2.6966**	**+22.8675**	**-16.5252**
Méthode 1 Speed-up	91.0944	39.8043	96.7163	79.0574
Méthode 1 Speed-up Attendu	13.87	16.31	0.79	0.17
Écart	**+77.2244**	**+23.4943**	**+95.9263**	**+78.8874**
Méthode 2 Prix	15.9333	8.3990	48.6990	21.0943
Méthode 2 Attendu	16.19	9.86	47.50	26.55
Écart	**-0.2567**	**-1.4610**	**+1.1990**	**-5.4557**
Méthode 2 Var ratio	242.5370	332.8014	16.2567	20.5839
Méthode 2 Var ratio Attendu	266.93	379.32	35.98	11.44
Écart	**-24.3930**	**-46.5186**	**-19.7233**	**+9.1439**
Méthode 2 Cost	1.7961	4.1422	3.5084	2.3694
Méthode 2 Cost Attendu	14.32	3.34	14.97	14.23
Écart	**-12.5239**	**+0.8022**	**-11.4616**	**-11.8606**
Méthode 2 Speed-up	326.5299	36.7455	615.6546	127.5582
Méthode 2 Speed-up Attendu	18.64	113.46	2.43	0.80
Écart	**+307.8899**	**-76.7145**	**+613.2246**	**+126.7582**
Variable de contrôle Prix	15.9452	8.4338	48.7211	21.0615
Control variate (Attendu)	15.95	9.75	46.00	26.48
Écart	**-0.0048**	**-1.3162**	**+2.7211**	**-5.4185**
Variable de contrôle Var ratio	169.9899	128.8463	10.0625	15.0132
Control Var ratio (Attendu)	336.12	166.79	15.28	4.08
Écart	**-166.1301**	**-37.9437**	**-5.2175**	**+10.9332**
Variable de contrôle Cost	1.6414	0.7574	3.1690	0.6619
Control Cost (Attendu)	24.23	10.22	19.34	23.76
Écart	**-22.5886**	**-9.4626**	**-16.1710**	**-23.0981**
Variable de contrôle Speed-up	357.9295	201.9114	705.9095	464.4460
Control Speed-up (Attendu)	13.87	16.31	0.79	0.17
Écart	**+344.0595**	**+185.6014**	**+705.1195**	**+464.2760**

Les écarts observés entre les résultats obtenus et ceux attendus sous Black-Scholes peuvent s'expliquer par plusieurs facteurs. Premièrement, les hypothèses simplificatrices du modèle (volatilité constante, corrélations fixes, absence de skew/smile) diffèrent souvent des conditions de marché réelles. Deuxièmement, la précision de la simulation Monte Carlo dépend du nombre de trajectoires et de la qualité du sampling, ce qui peut engendrer des fluctuations notables, surtout pour des payoffs sensibles (Asian, worst-of). Enfin, toute approximation lors de la construction de variables de contrôle ou la réduction de dimension (via réseau de neurones) peut ajouter un biais ou influencer la variance.

En conclusion, notre modèle reste cohérent dans l'ensemble, même s'il présente d'importants écarts par rapport aux valeurs de référence (article). Cependant, on constate une réduction de variance très nette lorsque nous utilisons des variables de contrôle explicites (en particulier avec la Méthode 2 et la méthode avec la variable de contrôle.

5 Présentation du modèle à volatilité locale pour plusieurs actifs

5.1 Équations du modèle

Nous considérons un modèle de volatilité locale avec la fonction de volatilité paramétrique suivante :

$$\text{Pour } t \in [0,T], \quad \sigma(t,x) = 0.6\Big(1.2 - e^{-0.1\,t}\, e^{-0.001\left(xe^{r-t}-s\right)^2}\Big)\, e^{-0.05\,\sqrt{t}}.$$

où (s) correspond au prix spot de l'actif sous-jacent. Ce choix de (s) permet de positionner le minimum du sourire de volatilité au niveau du forward money, rendant la formule cohérente.

Contrairement au modèle de Black-Scholes, ce modèle n'admet pas de méthode de simulation exacte. Nous avons donc recours à un schéma de discrétisation d'Euler.

5.2 Dynamique des prix des actifs

Nous allons implementer une fonction qui permet de calculer les trajectoires sous le modèle à volatilité locale.

Le processus sous-jacent suit l'équation différentielle stochastique suivante :

$$\frac{dS_t}{S_t} = \left(r - \frac{1}{2}\sigma^2(t,S_t)\right)dt + \sigma(t,S_t)dW_t,$$

où dW_t est l'incrément d'un mouvement Brownien et r est le taux d'intérêt sans risque.

La discrétisation d'Euler donne l'évolution du prix entre deux instants t_{n-1} et t_n :

$$S_{t_n} = S_{t_{n-1}} \cdot \exp\left(\left(r - \frac{1}{2}\sigma^2(t_{n-1}, S_{t_{n-1}})\right)\Delta t + \sigma(t_{n-1}, S_{t_{n-1}})\sqrt{\Delta t} Z_n\right)$$

où $Z_n \sim \mathcal{N}(0,1)$ est une variable normale standard.

5.3 Matrice de corrélation

Pour plusieurs actifs (d actifs simulés), les chocs gaussiens $\mathbf{G}_n$ sont ajustés pour inclure les corrélations. On utilise une décomposition de Cholesky $\mathbf{C}$ de la matrice de corrélation. Les chocs corrigés sont donnés par :

$$\mathbf{dW} = \mathbf{G}_n \cdot \mathbf{C}^\top$$

où $\mathbf{C}^\top$ est la transposée de la matrice de Cholesky.

5.4 Simulation des trajectoirs

Pour chaque actif i, on met à jour le prix $S^i_{t_n}$ selon :

$$S^i_{t_n} = S^i_{t_{n-1}} \cdot \exp\left(\left(r - \frac{1}{2}\sigma^2(t_{n-1}, S^i_{t_{n-1}})\right)\Delta t + \sigma(t_{n-1}, S^i_{t_{n-1}})\sqrt{\Delta t} \cdot Z^i_n\right)$$

où Z^i_n sont les chocs gaussiens ajustés pour inclure la corrélation.

5.4.1 Analyse de nos résultats

Les résultats sont résumés dans le tableau suivant :

Table 3: Résultats des simulations sous le modèle de volatilité locale

Métrique	**Basket**	**Asian**	**A_digit**	**Worst_of**
Monte Carlo Prix	9.6621	4.6211	61.8383	7.1807
Méthode 1 Prix	9.6798	4.6428	61.8350	7.1755
Méthode 1 Var ratio	1.0015	1.0109	0.8182	1.1395
Méthode 1 Cost	6.0137	2.3041	50.5573	3.3769
Méthode 1 Speed-up	64.4982	27.5727	77.8597	43.0685
Méthode 2 Prix	9.6626	4.6172	61.8705	7.2007
Méthode 2 Var ratio	3.5843	1.5006	1.8214	1.1278
Méthode 2 Cost	2.3740	2.2611	23.1379	3.0502
Méthode 2 Speed-up	104.5600	23.5372	118.5897	47.9126
Variable de contrôle Prix	9.6822	4.6386	61.8382	7.1715
Variable de contrôle Var ratio	16.1072	32.2737	3.2896	6.1115
Variable de contrôle Cost	0.4427	0.5414	10.1724	0.5823
Variable de contrôle Speed-up	465.5864	60.8165	227.0723	154.7914

Analyse

Méthode 1 : Réduction de dimension : utilisation de Z

- **Prix**: Les prix sont très proches de ceux de Monte Carlo, indiquant une bonne précision.
- **Var ratio**: La réduction de variance est faible pour toutes les options. En particulier, le ratio est inférieur à 1 pour A_digit, ce qui peut augmenter l'incertitude.
- **Cost**: La méthode offre une réduction significative du coût, en particulier pour A_digit (50.56) et Worst_of (3.38).
- **Speed-up**: Les gains de vitesse sont modérés, variant de 27.57 pour Asian à 77.86 pour A_digit.

Méthode 2 : Réduction de dimension : utilisation d'une variable de contrôle

- **Prix**: Les résultats restent proches de Monte Carlo, avec une légère variation pour certaines options.
- **Var ratio**: Une réduction significative de la variance est observée, en particulier pour Basket (3.58) et A_digit (1.82).
- **Cost**: Cette méthode réduit davantage le coût, notamment pour A_digit (23.14) et Basket (2.37).
- **Speed-up**: Les gains de vitesse sont importants, atteignant 118.59 pour A_digit et 104.56 pour Basket.

Variable de contrôle

- **Prix**: Les prix sont très proches de Monte Carlo, avec une précision remarquable.
- **Var ratio**: La réduction de variance est spectaculaire, avec des ratios de 16.11 pour Basket et 32.27 pour Asian.
- **Cost**: Le coût est extrêmement réduit, atteignant des valeurs très faibles pour toutes les options.
- **Speed-up**: Cette méthode offre les meilleurs gains de vitesse, notamment pour A_digit (227.07) et Worst_of (154.79).

Conclusion

Parmi les méthodes testées :

- **Méthode 1** est une solution équilibrée offrant une précision élevée et un gain de coût modéré.
- **Méthode 2** excelle dans la réduction du coût et de la variance, tout en offrant des gains de vitesse significatifs.

- **Variable de contrôle** surpasse les autres en termes de réduction de la variance et d'accélération, tout en conservant une précision remarquable.

La méthode avec variable de contrôle semble être la meilleure option pour ces simulations.

5.4.2 Comparaison avec les résultats attendus

Table 4: Comparaison des résultats simulés et attendus sous le modèle de volatilité locale

Métrique	**Basket**	**Asian**	**A_digit**	**Worst_of**
Monte Carlo Prix	9.6621	4.6211	61.8383	7.1807
Monte Carlo Attendu	7.91	4.33	60.41	8.56
Écart	**+1.7521**	**+0.2911**	**+1.4283**	**-1.3793**
Méthode 1 Prix	9.6798	4.6428	61.8350	7.1755
Méthode 1 Attendu	7.92	4.37	61.75	8.56
Écart	**+1.7598**	**+0.2728**	**+0.0850**	**-1.3845**
Méthode 1 Var ratio	1.0015	1.0109	0.8182	1.1395
Méthode 1 Var ratio Attendu	36.58	76.48	2.81	2.68
Écart	**-35.5785**	**-75.4691**	**-1.9918**	**-1.5405**
Méthode 1 Cost	6.0137	2.3041	50.5573	3.3769
Méthode 1 Cost Attendu	8.83	3.02	4.89	4.71
Écart	**-2.8163**	**-0.7159**	**+45.6673**	**-1.3331**
Méthode 1 Speed-up	64.4982	27.5727	77.8597	43.0685
Méthode 1 Speed-up Attendu	4.14	25.32	0.57	0.17
Écart	**+60.3582**	**+2.2527**	**+77.2897**	**+42.8985**
Méthode 2 Prix	9.6626	4.6172	61.8705	7.2007
Méthode 2 Attendu	7.84	4.36	61.77	8.18
Écart	**+1.8226**	**+0.2572**	**+0.1005**	**-0.9793**
Méthode 2 Var ratio	3.5843	1.5006	1.8214	1.1278
Méthode 2 Var ratio Attendu	48.59	78.95	3.33	5.59
Écart	**-45.0057**	**-77.4494**	**-1.5086**	**-4.4622**
Méthode 2 Cost	2.3740	2.2611	23.1379	3.0502
Méthode 2 Cost Attendu	1.21	1.24	1.50	1.21
Écart	**+1.1640**	**+1.0211**	**+21.6379**	**+1.8402**
Méthode 2 Speed-up	104.5600	23.5372	118.5897	47.9126
Méthode 2 Speed-up Attendu	40.15	13.67	2.22	6.76
Écart	**+64.4100**	**+9.8672**	**+116.3697**	**+41.1526**
Variable de contrôle Prix	9.6822	4.6386	61.8382	7.1715
Control variate (Attendu)	7.85	4.36	60.38	8.49
Écart	**+1.8322**	**+0.2786**	**+1.4582**	**-1.3185**
Variable de contrôle Var ratio	16.1072	32.2737	3.2896	6.1115
Control Var ratio (Attendu)	8.29	9.77	2.62	5.59
Écart	**+7.8172**	**+22.5037**	**+0.6696**	**+0.5215**
Variable de contrôle Cost	0.4427	0.5414	10.1724	0.5823
Control Cost (Attendu)	1.85	1.75	0.78	1.85
Écart	**-1.4073**	**-1.2086**	**+9.3924**	**-1.2677**
Variable de contrôle Speed-up	465.5864	60.8165	227.0723	154.7914
Control Speed-up (Attendu)	4.48	5.58	0.78	3.02
Écart	**+461.1064**	**+55.2365**	**+226.2923**	**+151.7714**

Conclusion

Les résultats obtenus dans ce cadre de simulation sous le modèle de volatilité locale font clairement apparaître l'importance des techniques de réduction de variance pour améliorer à la fois la précision et l'efficacité numérique des estimations de prix d'options. Les trois approches testées (Méthode 1, Méthode 2 et Variable de contrôle) offrent chacune des avantages :

- **Méthode 1** s'avère équilibrée, avec une bonne précision et des gains de vitesse notables, sans toutefois proposer la meilleure réduction de variance.
- **Méthode 2** accentue la réduction de variance et propose d'importants gains de vitesse, se révélant plus performante que la première méthode pour la plupart des options.
- **Variable de contrôle** surpasse nettement les deux premières en termes de ratio de variance et de speed-up, tout en conservant une précision remarquable sur les prix simulés.

En comparant ces résultats aux valeurs attendues , on observe encore certains écarts, principalement dus :

- Aux hypothèses de modélisation et à la *calibration* de la surface de volatilité locale, qui peut s'avérer complexe.
- Au choix de paramètres et de méthodes numériques (discrétisation, pas de temps, etc.) susceptibles d'impacter les estimations finales.

Toutefois, l'étude démontre clairement que l'adoption de méthodes de réduction de variance, en particulier l'introduction d'une variable de contrôle adaptée, constitue un levier puissant pour accroître la fiabilité et réduire le coût computationnel de la simulation de ce modèle local. Dans une perspective industrielle, le choix de la méthode retenue dépendra des contraintes de performance (coût, temps de calcul) et des exigences de précision, mais il ressort de nos analyses que la **variable de contrôle** est la plus performante pour la plupart des instruments étudiés.

6 Présentation du modèle de Heston pour plusieurs actifs

6.1 Équations du modèle

1. Dynamique des prix des actifs

$$dS_t^i = r\, S_t^i\, dt + \sqrt{\sigma_t^i}\, S_t^i\, dB_t^i,$$

où :

- S_t^i est le prix de l'actif i au temps t,
- r est le taux d'intérêt sans risque,
- σ_t^i est le processus de variance stochastique associé à l'actif i,
- B_t^i est un mouvement Brownien.

2. Dynamique de la variance stochastique

$$d\sigma_t^i \;=\; \kappa\left(a - \sigma_t^i\right) dt \;+\; \nu\sqrt{\sigma_t^i}\left(\gamma\, dB_t^i \;+\; \sqrt{1-\gamma^2}\, d\widetilde{B}_t^i\right),$$

où :

- κ est le taux de retour à la moyenne de la variance,
- a est le niveau moyen de la variance,
- ν est la volatilité de la variance (vol de vol),
- γ est la corrélation entre le prix de l'actif et son processus de variance,
- $\widetilde{B}_t^i$ est un mouvement Brownien indépendant.

3. Matrice de corrélation Les corrélations entre les différents actifs et les processus de variance sont décrites par :

$$d\langle B\rangle_t = \Gamma\, dt, \quad d\langle \widetilde{B}\rangle_t = I_N\, dt,$$

où :

- Γ est la matrice de corrélation entre les différents actifs,
- I_N est la matrice identité de dimension $N \times N$.

6.2 Reformulation équivalente

Le modèle peut être réécrit sous la forme :

$$dS_t^i = r\, S_t^i\, dt + \sqrt{\sigma_t^i}\, S_t^i\, dB_t^i,$$

$$d\sigma_t^i = \kappa\,(a - \sigma_t^i)\, dt \;+\; \nu\, \sqrt{\sigma_t^i}\, d\widehat{B}_t^i,$$

où B et $\widehat{B}$ sont des processus de Wiener satisfaisant :

$$d\langle B\rangle_t = \Gamma\, dt, \quad d\langle B, \widehat{B}\rangle_t = \gamma\, \Gamma\, dt,$$

$$d\langle \widehat{B}\rangle_t = \gamma^2\, \Gamma + (1 - \gamma^2)\, I_N\, dt.$$

6.3 Matrice de covariance

Le processus $(B, \widehat{B})$ prend ses valeurs dans $\mathbb{R}^{2N}$ et est un processus de Wiener dont la matrice de covariance est donnée par :

$$\widetilde{\Gamma} \;=\; \begin{pmatrix} \Gamma & \gamma\, \Gamma \\ \gamma\, \Gamma & \gamma^2\, \Gamma \;+\; (1 - \gamma^2)\, I_N \end{pmatrix}.$$

6.4 Simulation des trajectoires

1. Mise à jour des volatilités La dynamique des volatilités stochastiques est donnée par :

$$v_{t+1}^i \;=\; \max\Big(\, v_t^i \;+\; \kappa\,(a - v_t^i)\, \Delta t \;+\; \nu\, \sqrt{v_t^i\, \Delta t}\; Z_t^i,\; 0\Big),$$

2. Mise à jour des prix des actifs La dynamique des prix est donnée par :

$$S_{t+1}^i \;=\; S_t^i \;+\; r\, S_t^i\, \Delta t \;+\; \sqrt{v_{t+1}^i\, \Delta t}\; S_t^i\, Z_t^i,$$

3. Application de la corrélation Les chocs browniens corrigés Z_t sont obtenus en appliquant la matrice de corrélation Γ (via sa décomposition de Cholesky) aux chocs standards dW :

$$Z_t \;=\; \Gamma_{\text{Cholesky}} \;\cdot\; \begin{pmatrix} dW_S \\ dW_v \end{pmatrix},$$

où dW_S et dW_v sont des variables normales standards (gaussiennes) représentant les chocs pour les prix et les volatilités, respectivement.

6.4.1 Analyse de nos résultats

Table 5: Résultats sous le modèle de volatilité stochastique (Heston)

Métrique	**Basket**	**Asian**	**A_digit**	**Worst_of**
Monte Carlo Prix	15.2409	8.9545	49.7810	22.7767
Méthode 1 Prix	15.2024	8.9477	49.7445	22.7428
Méthode 1 Var ratio	0.9646	0.9934	0.9407	1.1295
Méthode 1 Cost	30.0117	8.2450	87.3437	14.1459
Méthode 1 Speed-up	31.7450	37.4821	47.8597	49.9543
Méthode 2 Prix	15.2389	9.1095	49.7444	22.7420
Méthode 2 Var ratio	48.1635	215.1309	11.8753	13.1738
Méthode 2 Cost	1.6126	0.9549	5.2337	1.8611
Méthode 2 Speed-up	296.1008	162.0343	419.7495	216.6868
Variable de contrôle Prix	15.1945	8.9445	49.7685	22.7754
Variable de contrôle Var ratio	80.5011	114.3785	5.6253	11.2922
Variable de contrôle Cost	0.3950	0.2623	8.6391	1.4050
Variable de contrôle Speed-up	1198.8963	592.3260	276.2359	290.3988

Analyse

Méthode 1 (Réduction de dimension : utilisation de Z

- **Prix**: Les prix obtenus sont proches des estimations de Monte Carlo, indiquant une bonne précision.
- **Var ratio**: Les ratios de variance sont faibles, particulièrement pour A_digit (0.9407), ce qui montre une augmentation relative de la variance.
- **Cost**: Cette méthode offre un coût réduit par rapport à Monte Carlo, mais reste plus élevé que Méthode 2 et la variable de contrôle.
- **Speed-up**: Les gains de vitesse sont significatifs, variant de 31.75 (Basket) à 49.95 (Worst_of).

Méthode 2 (Réduction de dimension : utilisation d'une variable de contrôle)

- **Prix**: Les estimations restent précises et proches de Monte Carlo, avec des variations mineures.
- **Var ratio**: La réduction de variance est significative, particulièrement pour Asian (215.13) et Basket (48.16).
- **Cost**: Le coût est considérablement réduit, notamment pour Basket (1.61) et Worst_of (1.86).
- **Speed-up**: La méthode offre un gain de vitesse exceptionnel, atteignant 419.75 pour A_digit et 296.10 pour Basket.

Variable de contrôle

- **Prix**: Les prix sont très proches des estimations Monte Carlo, avec une très bonne précision.
- **Var ratio**: Cette méthode surpasse toutes les autres en réduisant la variance, avec des ratios atteignant 80.50 pour Basket et 114.38 pour Asian.
- **Cost**: Le coût est le plus bas parmi les méthodes, avec des valeurs aussi faibles que 0.26 pour Asian.
- **Speed-up**: La méthode offre des gains de vitesse spectaculaires, atteignant 1198.90 pour Basket et 592.33 pour Asian.

Conclusion Parmi les méthodes testées :

- **Méthode 1** fournit une solution équilibrée, mais son coût reste plus élevé que les autres méthodes pour certaines options.

- **Méthode 2** réduit la variance et le coût de manière significative tout en offrant des gains de vitesse impressionnants.
- **Variable de contrôle** est la méthode la plus performante en termes de réduction de variance, de coût et de vitesse, tout en maintenant une précision élevée.

6.4.2 Comparaison avec les résultats attendu

Table 6: Comparaison des résultats simulés et attendus sous le modèle Heston

Métrique	**Basket**	**Asian**	**A_digit**	**Worst_of**
Monte Carlo Prix	15.2409	8.9545	49.7810	22.7767
Monte Carlo Attendu	9.69	5.76	56.92	12.40
Écart	**+5.55**	**+3.19**	**-7.14**	**+10.38**
Méthode 1 Prix	15.2024	8.9477	49.7445	22.7428
Méthode 1 Attendu	9.49	5.58	56.37	12.43
Écart	**+5.71**	**+3.37**	**-6.63**	**+10.31**
Méthode 1 Var ratio	0.9646	0.9934	0.9407	1.1295
Méthode 1 Var ratio Attendu	20.91	32.83	1.00	2.54
Écart	**-19.95**	**-31.83**	**-0.06**	**-1.41**
Méthode 1 Cost	30.0117	8.2450	87.3437	14.1459
Méthode 1 Cost Attendu	13.85	13.54	11.70	13.82
Écart	**+16.16**	**-5.30**	**+75.64**	**+0.33**
Méthode 2 Prix	15.2389	9.1095	49.7444	22.7420
Méthode 2 Attendu	9.57	5.92	55.19	12.58
Écart	**+5.67**	**+3.19**	**-5.45**	**+10.16**
Méthode 2 Var ratio	48.1635	215.1309	11.8753	13.1738
Méthode 2 Var ratio Attendu	31.03	502.49	1.63	5.44
Écart	**+17.13**	**-287.36**	**+10.25**	**+7.73**
Variable de contrôle Prix	15.1945	8.9445	49.7685	22.7754
Control variate (Attendu)	9.57	5.71	56.24	12.53
Écart	**+5.62**	**+3.23**	**-6.47**	**+10.25**
Variable de contrôle Var ratio	80.5011	114.3785	5.6253	11.2922
Control Var ratio (Attendu)	4.02	4.40	1.73	3.98
Écart	**+76.48**	**+109.98**	**+3.90**	**+7.31**
Variable de contrôle Speed-up	1198.8963	592.3260	276.2359	290.3988
Control Speed-up (Attendu)	1.69	1.93	0.41	1.46
Écart	**+1197.21**	**+590.40**	**+275.83**	**+288.94**

Les écarts observés entre les résultats obtenus et ceux attendus peuvent être attribués à plusieurs facteurs. Premièrement, la calibration des paramètres du modèle de volatilité stochastique (Heston), tels que la volatilité initiale, la corrélation ou la vitesse de réversion, peut différer entre les simulations réalisées et celles utilisées comme référence. Deuxièmement, la précision des simulations Monte Carlo dépend fortement du nombre de trajectoires simulées, et une différence dans cette granularité peut impacter les résultats. Enfin, des variations dans la modélisation des payoffs complexes, comme les options Asian ou Worst_of, peuvent également introduire des divergences dues à la sensibilité de ces instruments aux spécificités du modèle et des hypothèses sous-jacentes.

En conclusion notre modèle est cohérent, mais diverge beaucoup des attentes du PDF. Cependant la reduction de variance est assez nette sur les 2 méthodes qui utilise une variable de controle explicite (Méthode 2 et méthode control variate).

7 Conclusion Générale des parties 4, 5 et 6

Les travaux présentés mettent en évidence l'apport substantiel des méthodes de réduction de variance dans l'estimation par Monte Carlo des prix d'options sous trois cadres de modélisation distincts (Black-Scholes, volatilité locale et Heston). Plus précisément :

7.1 Qualité globale et divergences observées

Les écarts constatés entre valeurs « simulées » et « de référence » proviennent en grande partie :

- Des simplifications structurelles inhérentes à chaque modèle (volatilité constante dans Black-Scholes, dépendance complète à une surface de volatilité dans le modèle local, dynamique stochastique dans Heston).
- De la calibration des paramètres, parfois délicate, surtout pour des payoffs exotiques (options asiatiques, worst-of, digitales).
- Du nombre de trajectoires et des techniques de discrétisation adoptées dans la simulation, pouvant engendrer une variabilité notable des résultats.

7.2 Comparaison des méthodes de réduction de variance

- **Méthode 1 :** Elle présente un compromis acceptable entre précision, rapidité d'exécution et facilité d'implémentation. Son impact sur la variance reste limité par rapport aux deux autres approches, mais elle confère déjà une amélioration tangible en vitesse de calcul.
- **Méthode 2 :** Généralement plus exigeante en termes de paramétrage, elle s'avère néanmoins plus performante que la Méthode 1 dans la majorité des cas, aussi bien en termes de réduction de variance que de gains de vitesse. Son efficacité se remarque particulièrement pour les options dont le payoff est plus complexe.
- **Variable de contrôle :** Les résultats montrent que cette technique demeure la plus efficace en matière de réduction de variance et de speed-up, tout en maintenant une très bonne exactitude des prix simulés. Elle requiert toutefois une construction adéquate de la variable de contrôle, ce qui suppose une certaine expertise et, parfois, un coût de mise en œuvre additionnel.

7.3 Analyse selon le type de modèle

- **Black-Scholes :** L'hypothèse de volatilité constante explique en partie les écarts plus marqués pour certains produits exotiques très sensibles à la dynamique sous-jacente. Les approches de réduction de variance permettent néanmoins de compenser en partie ces insuffisances en limitant la variabilité des estimateurs de prix.
- **Volatilité locale :** La complexité de la surface de volatilité (et son interpolation) rend la calibration sujette à de possibles erreurs ; les méthodes de réduction de variance se révèlent alors particulièrement pertinentes pour contenir la variabilité induite par ces incertitudes.
- **Heston :** La volatilité stochastique et la corrélation entre le sous-jacent et sa variance accroissent la difficulté de simulation. Ici encore, l'introduction de variables de contrôle et l'ajustement des trajectoires (Méthode 2) apportent des gains notables, en termes aussi bien de fiabilité que de coûts de calcul.

7.4 Perspectives et recommandations

Les résultats soulignent l'importance stratégique du choix de la méthode de réduction de variance au regard :

- Des exigences de précision (évaluation de produits complexes, calculs en temps réel, etc.).
- Des contraintes de performance (capacité de calcul disponible, rapidité de pricing souhaitée).
- Du niveau d'expertise et d'ingénierie requis pour la mise en œuvre (évaluation de la pertinence d'une variable de contrôle, paramétrage de la méthode, etc.).

En somme, la comparaison menée sous les trois modèles (Black-Scholes, volatilité locale, Heston) confirme l'efficacité des techniques de réduction de variance pour améliorer la qualité et la fiabilité des estimations de prix. Parmi les approches étudiées, la variable de contrôle apparaît comme la plus performante, combinant à la fois forte réduction de variance et speed-up élevé, moyennant une calibration soignée de la variable auxiliaire. Ces travaux ouvrent la voie à des extensions futures visant à raffiner la construction de ces variables de contrôle (par exemple via l'apprentissage automatique ou l'approximation semi-analytique) et à élargir le champ des produits dérivés exotiques pour lesquels ces stratégies de réduction de variance pourraient être appliquées de manière optimale.

8 Robustesse des variables de contrôle

Dans cette section, nous testons si les réseaux neuronaux présentés dans la Section 3 sont capables de résister aux variations des variables associées au payoff et au modèle. Étant donné que cette méthode apprend uniquement les directions les plus importantes pour le payoff et le modèle, et non leur combinaison exacte, nous espérons qu'elle sera résistante à certains changements de paramètres.

8.1 Sensibilité du prix des options asiatiques

Pour explorer cet aspect, nous considérons une option asiatique arithmétique définie par la formule suivante :

$$\left(\frac{1}{d}\sum_{i=1}^{d} S_{ti} - K\right)^{+}.$$

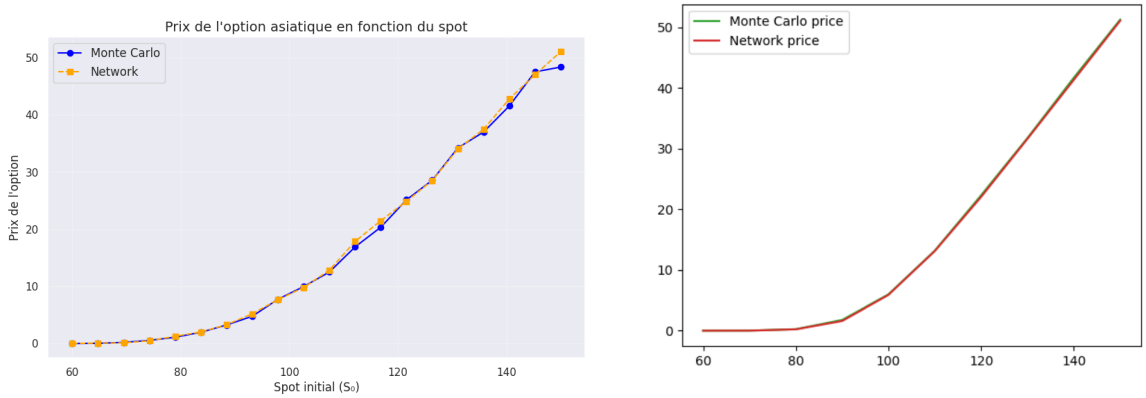
Le réseau est entraîné avec le jeu de paramètres suivant :

$$K = 100.0,\ \sigma = 0.2,\ S_0 = 100.0,\ T = 1 \text{ an},\ d = 20.$$

Une fois le réseau entraîné, il est testé pour une plage de paramètres plus large que celle utilisée pour l'entraînement. Les prix fournis par le réseau, illustrés dans les figures suivantes, sont les prix bruts, sans les utiliser comme variables de contrôle. Étant donné que cet exemple est relativement simple, ces prix sont assez précis sans nécessiter de correction du biais à l'aide de Monte Carlo.

Monte Carlo vs Réduction de dimension : utilisation de Z

8.1.1 Prix de l'option asiatique en fonction du spot



(a) Graphique réalisé

(b) Graphique du PDF

Figure 1: Comparaison des prix de l'option asiatique en fonction du spot

Analyse graphique

Le graphique ci-dessous présente le prix de l'option asiatique en fonction du spot initial (S_0). Deux méthodes sont comparées :

- Monte Carlo, représentée par une ligne bleue continue avec des points circulaires ;
- Network, représentée par une ligne orange pointillée avec des carrés (reduction de dimension, methode 1).

Observations

- Le prix de l'option asiatique augmente de manière monotone avec (S_0), ce qui correspond aux attentes théoriques.
- Les deux méthodes donnent des résultats cohérents, avec de légères différences pour certains points.

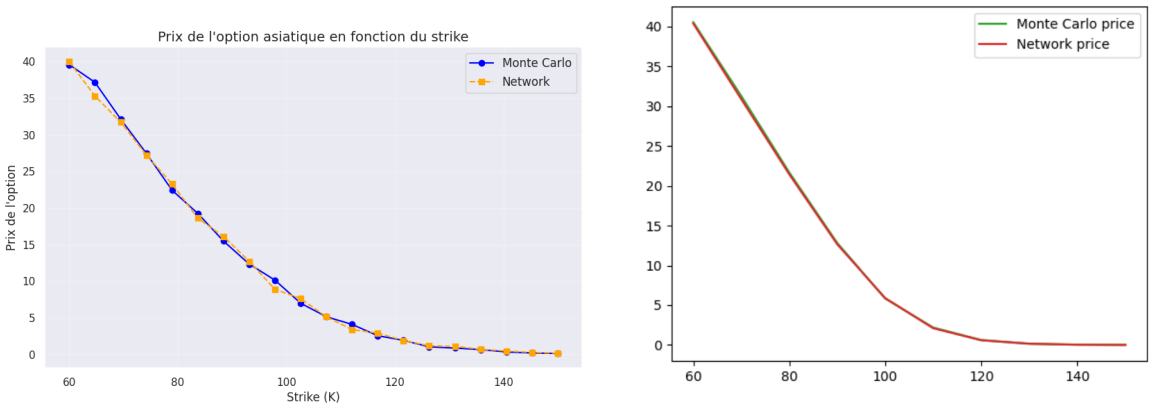
Conclusion

- L'approche Network fournit des résultats similaires à ceux de la méthodeMonte Carlo. Elle peut donc être considérée comme une alternative valide pour le calcul des prix des options asiatiques.

Comparaison

- Notre code semble reproduire exactement le graphique obtenue.

8.1.2 Prix de l'option asiatique en fonction du strike



(a) Graphique réalisé

(b) Graphique du PDF

Figure 2: Comparaison des prix de l'option asiatique en fonction du strike

Analyse graphique

Le graphique ci-dessous illustre l'évolution du prix d'une option asiatique en fonction du prix d'exercice (K) . Deux méthodes de calcul sont comparées :

- Monte Carlo, représentée par une ligne bleue continue avec des points circulaires ;
- Network, représentée par une ligne orange pointillée avec des carrés (reduction de dimension, methode 1).

Observations

- Le prix de l'option diminue de manière monotone à mesure que le strike (K) augmente.
- Les résultats des deux méthodes (Monte Carlo et Network) sont très similaires, avec seulement de faibles écarts ponctuels.

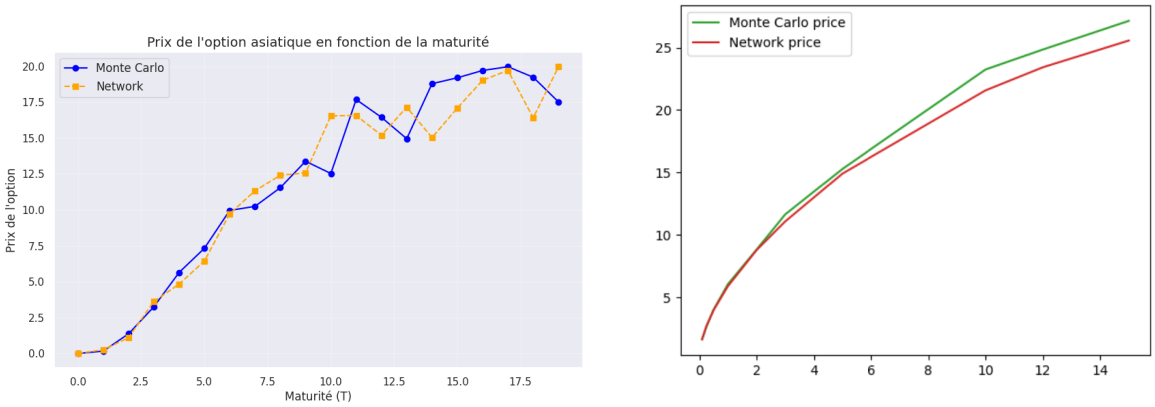
Conclusion

- L'approche Network reproduit fidèlement les résultats obtenus avec la méthodeMonte Carlo.

Comparaison

- Notre code semble reproduire exactement le graphique obtenue.

8.1.3 Prix de l'option asiatique en fonction de la maturité



(a) Graphique réalisé

(b) Graphique du PDF

Figure 3: Comparaison des prix de l'option asiatique en fonction de la maturité

Analyse graphique

Le graphique ci-dessous présente le prix de l'option asiatique en fonction du spot initial (T). Deux méthodes sont comparées :

- Monte Carlo, représentée par une ligne bleue continue avec des points circulaires ;
- Network, représentée par une ligne orange pointillée avec des carrés (reduction de dimension, methode 1).

Observations

- Le prix de l'option asiatique augmente globalement avec la maturité (T) , avec certaines fluctuations mineures.
- Les deux méthodes (Monte Carlo et Network) produisent des résultats cohérents, avec de faibles divergences ponctuelles.
- L'approche Network présente des variations plus douces par rapport à Monte Carlo.

Conclusion

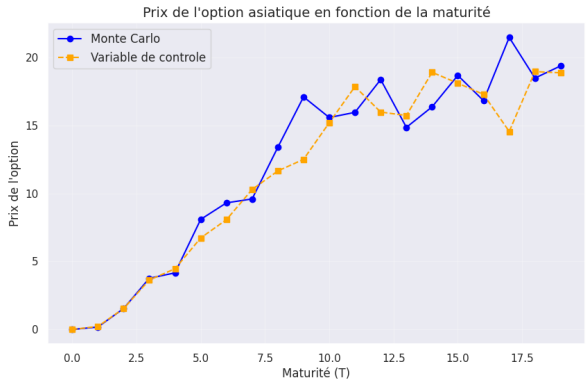
- L'approche Network offre des résultats similaires à ceux de la méthode Monte Carlo pour le calcul du prix d'une option asiatique en fonction de la maturité.

Comparaison

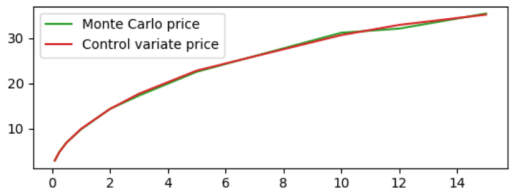
- Notre code semble reproduire assez similairement le graphique obtenue avec des fluctuation assez compliqué a expliquer.

Monte Carlo vs Réduction de dimension : Variable de contrôle

8.1.4 Prix de l'option asiatique en fonction de la maturité



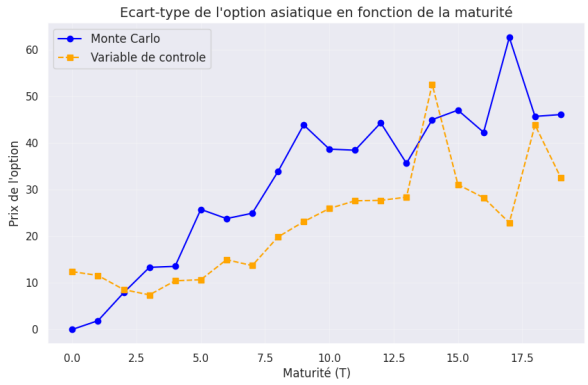
(a) Graphique réalisé



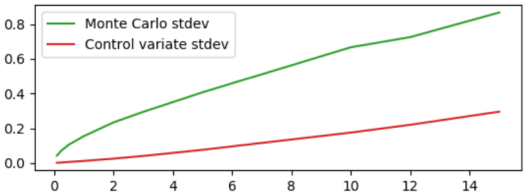
(b) Graphique du PDF

Figure 4: Comparaison des prix de l'option asiatique en fonction de la maturité

8.1.5 Ecart-type de l'option asiatique en fonction de la maturité



(a) Graphique réalisé



(b) Graphique du PDF

Figure 5: Comparaison de l'écart-type de l'option asiatique en fonction de la maturité

Analyse graphique

Deux graphiques sont présentés pour analyser le prix et l'écart-type d'une option asiatique en fonction de la maturité (T) , en comparant deux méthodes :

- Monte Carlo, représentée par une ligne bleue continue avec des points circulaires
- Variable de contrôle, représentée par une ligne orange pointillée avec des carrés (reduction de dimension, methode 2).

Graphique 1 : Prix moyen Observations

- Le prix moyen de l'option augmente globalement avec la maturité (T) .
- Les deux méthodes produisent des résultats très similaires, montrant une bonne concordance.
- La méthode utilisant une variable de contrôle présente des oscillations légèrement plus réduites.

Graphique 2 : Écart-type Observations

- L'écart-type des prix augmente avec la maturité pour la méthode Monte Carlo.

- La méthode utilisant une variable de contrôle réduit significativement la volatilité des estimations.

Conclusion

- Les deux graphiques montrent que la méthode Monte Carlo et celle utilisant une variable de contrôle produisent des prix moyens cohérents. Cependant, la méthode variable de contrôle offre une précision supérieure en réduisant significativement l'écart-type des estimations.

Comparaison

- Notre code ne semble pas reproduire le graphique 2 (Ecart-type). Mais pour le graphique 1, la resemblance est assez présente meme si les valeurs sont bien inferieur (environ 10)

8.2 Sensibilité du prix des AutoCall

Cette partie n'a pas été traité, malgres de nombreuse tentative nous n'avons pas réussi à implementer cette partie.

Profils de sensibilités (Greeks)

Étant donné que la méthode de réduction de dimension utilisant les variables de contrôle résiste à certains changements de paramètres, nous l'utilisons pour calculer certains *Greeks*, qui sont des quantités représentant la sensibilité du payoff à certains paramètres. Dans cette section, nous considérons l'option basket décrite précédemment.

Nous calculons, par différences finies, certains *Greeks* de l'option à l'aide du réseau. Il est important de noter que ces *Greeks* sont calculés à partir du réseau brut, sans l'utiliser comme variable de contrôle. Nous comparons ensuite ces valeurs avec celles obtenues par la méthode de Monte Carlo.

Les *Greeks* suivants sont définis :

- **Delta** : $\delta = \frac{\partial \Pi}{\partial S_i}$, où Π est le prix de l'option et S_i est le i-ème actif sous-jacent.
- **Vega** : $\nu = \frac{\partial \Pi}{\partial \sigma_i}$, où σ_i est la volatilité du i-ème actif sous-jacent.

Explication mathématique

1. Calcul du prix d'une option basket

Le prix d'une option basket est donné par la formule suivante :

$$\Pi = \mathbb{E}\left[e^{-rT}\,\text{Payoff}(S)\right],$$

où :

- $\mathbb{E}[\cdot]$ désigne l'espérance sous la mesure risque-neutre,
- e^{-rT} est le facteur d'actualisation avec r le taux d'intérêt sans risque et T la maturité,
- Payoff(S) représente le payoff de l'option, calculé à partir des valeurs simulées S des actifs sous-jacents.

2. Calcul des *Greeks*

Les *Greeks* mesurent la sensibilité du prix de l'option à certains paramètres du modèle. Dans ce code, deux *Greeks* principaux sont calculés :

a. Delta (δ)

Delta mesure la sensibilité du prix de l'option à une variation du prix initial de l'actif sous-jacent S_0 :

$$\delta_i = \frac{\partial \Pi}{\partial S_i},$$

où S_i est le i-ème actif sous-jacent.

Numériquement, cette dérivée est approchée par la méthode des différences finies centrées :

$$\delta_i \approx \frac{\Pi(S_i + h) - \Pi(S_i - h)}{2h},$$

où h est un petit incrément.

b. Vega (ν)

Vega mesure la sensibilité du prix de l'option à une variation de la volatilité initiale σ_i de l'actif sous-jacent :

$$\nu_i = \frac{\partial \Pi}{\partial \sigma_i}.$$

Numériquement, elle est calculée de manière similaire à Delta, en perturbant σ_i de $+h$ et $-h$, et en utilisant :

$$\nu_i \approx \frac{\Pi(\sigma_i + h) - \Pi(\sigma_i - h)}{2h}.$$

Cette partie n'a pas pu être traité en entierté, et ne peut donc pas etre analyser.

9 Conclusion

Ce projet nous a permis d'approfondir l'étude des méthodes avancées de réduction de variance pour la valorisation d'options via des simulations Monte Carlo, en nous appuyant sur l'article « Automatic Control Variates for Option Pricing using Neural Networks ». Trois méthodes principales ont été testées : la réduction de dimension, l'utilisation de variables de contrôle explicites et l'approche automatique via un réseau de neurones.

Les méthodes 2 et 3 ont montré des performances prometteuses. Nos résultats pour le modèle de Black-Scholes et le modèle de volatilité locale sont cohérents avec les attentes de l'article, avec des écarts de prix modérés. Cependant, dans le cadre du modèle de Heston, les prix obtenus diffèrent significativement de ceux attendus, ce qui peut être attribué à des différences de calibration ou à des approximations spécifiques dans les simulations. La méthode 1, bien qu'elle fournisse des prix précis, n'apporte qu'une très faible réduction de variance, contrairement aux résultats présentés dans l'article, ce qui limite son intérêt pratique.

En conclusion, les méthodes utilisant des variables de contrôle explicites (méthodes 2 et 3) semblent les plus efficaces pour améliorer la précision et réduire les coûts de calcul dans un cadre Monte Carlo. Elles constituent une piste intéressante pour de futures recherches et applications en finance quantitative, notamment pour des modèles complexes comme celui de Heston.

10 Bibliographie

Automatic Control Variates for Option Pricing using Neural Networks
Adam: a method for stochastic optimization
Documentation PyTorch
Cybenko, G. Approximation by superpositions of a sigmoidal function. Math. Control Signal Systems 2, 303–314 (1989)