ENSIIE - M2QF

Rapport Projet Cutting Edge

CVA Pricing

Ouassim Sebbar
Ghada Ben Said
Houssem Fendi
Mame Diarra Toure
Issame Sarroukhe
Gabriel Moran

Professeurs :

31/03/2021

17 mai 2021

Table des matières

1 Introduction

Il y a eu au cours de l'histoire plusieurs exemples de défaut d'entreprises ou de souverains qui ont eu des répercussions négatives importantes sur les marchés financiers. On peut pour illustrer ce propos citer pour les institutions financières les faillites du « Long Term Capital Management » (1998) et de « Lehman Brothers » (2008). Même si le risque de crédit et de contrepartie est mesuré depuis longtemps par les institutions financières, sa gestion consistait souvent à imposer des limites d'exposition par contrepartie (lignes de crédit) et à diversifier les contreparties pour réduire la concentration du risque. La valorisation des produits dérivés de gré à gré se faisait en environnement risque-neutre, ne tenant pas compte du risque de défaut propre à chaque contrepartie. Les grandes institutions financières qui prennent part aux marchés de gré à gré estimaient qu'elles étaient de taille importante « too big to fail » et donc que le risque de défaut de la contrepartie était négligeable.
Lors de la crise récente des subprime, suite à l'assèchement du marché interbancaire et à une hausse soudaine du prix de la liquidité due à une hausse des primes de risques exigées par les acteurs de ce marché, les grandes institutions financières virent la valeur de marché de leurs portefeuilles de négociation fondre, surtout ceux contenant des dérivés de gré à gré. Une grande défiance s'installa sur les marchés financiers et la faillite de « Lehman Brothers » en 2008, ainsi que les multiples plans de sauvetages qui ont suivi notamment le sauvetage de « American International Group (AIG) », confirmèrent que même les institutions les plus prestigieuses étaient soumises au risque de défaut.

Depuis la crise, un certain nombre d'ajustements sont nécessaires dans la gestion des risques des produits dérivés de gré-à-gré afin de prendre en compte le risque de contrepartie et les coûts de financement, en particulier l'ajustement de valorisation du crédit (Credit Valuation Adjustment, CVA), de la dette (Debt Valuation Adjustment, DVA), et du financement (Funding Valuation Adjustment, FVA).
Dans le cadre de ce projet nous allons nous intéresser principalement à l'étude et au calcul de la CVA.
La modélisation du risque de contrepartie est un défi de calcul car elle nécessite l'évaluation simultanée de toutes les transactions avec chaque contrepartie à la fois sous les risques de marché et de crédit. L'article étudié présente une méthodologie utilisant un processus de régression multi-gaussien pour estimer le risque de portefeuille, et en particulier pour le calcul de CVA. En utilisant une approche de modélisation spatio-temporelle la méthode permet d'eviter la simulation imbriquée de Monte Carlo(nested MC) en apprenant ce qu'on appelle communement un "kernel pricing layer" [1]

2 CVA - Credit Value Adjustment

2.1 Definitions

Notion de risque de contrepartie : Le risque de contrepartie peut être défini comme étant le risque de perte lié à un éventuel manquement d'une contrepartie à honorer ses obligations contractuelles en raison d'un défaut de paiement.

Notion de CVA : *La CVA (Credit Value Adjustment)* est la valeur de marché du risque de défaut d'une contrepartie. On la mesure par la différence entre la valeur sans risque d'un portefeuille et la valeur de celui-ci en tenant compte du défaut potentiel des contreparties. Elle permet de déterminer la « fair value» de certains produits dérivés et de mettre en place des réserves pour se prémunir du défaut d'une contrepartie. En pratique la CVA est calculée indépendamment pour chaque contrepartie puis agrégée pour obtenir une CVA globale par portefeuille.

2.2 CVA unilatérale et CVA bilatérale

Le risque de crédit supporté par les contreparties dans un contrat de produits dérivés peut être de nature bilatérale dans la mesure où le contrat peut prendre une valeur de marché positive pour une des parties, ce qui implique que le même contrat aura une valeur de marché négative pour l'autre partie. Cette valeur de marché peut, en cours de vie du contrat, changer de signe pour l'une (respectivement l'autre) contrepartie, ce qui signifie que le risque de crédit est présent de part et d'autre. C'est le cas par exemple d'un swap de taux.

Cependant, certains contrats ont une valeur de marché qui ne peut pas changer de signe. C'est le cas des obligations. L'investisseur (acheteur de l'obligation) est toujours celui qui supporte le risque de contrepartie, en l'occurrence le risque que l'émetteur fasse défaut.

Par analogie, nous allons dans le cadre de la mesure de la CVA considérer deux principaux cas :
1 - Le cas ou uniquement la qualité de crédit de la contrepartie est prise en compte, on parlera de CVA unilatérale.
2 - Le cas où la qualité de crédit de la contrepartie est prise en compte (CVA), mais également la qualité de crédit propre de l'institution évaluatrice du risque (DVA). Dans ce cas on parle de CVA bilatérale.

La DVA (Debit Valuation Adjustment) représente le risque de contrepartie vu de la perspective de la contrepartie. Elle est de signe opposé à la CVA unilatérale.

3 Calcul de la CVA

Avant de définir mathématiquement la CVA, il convient d'introduire la notion d'exposition de la banque vis à vis de sa contrepartie. Cette exposition désigne la valeur de l'ensemble des contrats de la banque qu'elle a en commun avec une contrepartie, qu'elle perdra en cas de défaut de cette dernière avant la maturité. Toutefois, en cas de défaut, la banque peut récupérer un collatéral, qui désigne l'ensemble des actifs, titres ou liquidités, garanties par la contrepartie auprès de la banque en cas de défaillance. On a alors :

$$E(t) := \max(V(t) - C(t); 0)$$

avec :

— $E(t)$: la valeur de l'exposition de la banque à la date t

— $V(t)$: la valeur du portefeuille de la banque avec la contrepartie à la date t.

— $C(t)$: la valeur de collatéral à la date t.

A partir de l'expression de l'exposition, nous pouvons déterminer celle de la perte (loss) de la banque en cas de défaut de la contrepartie. En effet, cette dernière correspond à une fraction (déterminée par le taux de recouvrement) de l'exposition de la banque actualisée, conditionnée au fait que la contrepartie fasse défaut avant la maturité du contrat. D'où :

$$L := 1_{\{\tau \leq T\}}(1 - R)E(\tau)D(\tau)$$

avec :

— τ : la date de défaut de la contrepartie

— L : la perte de la banque liée au défaut de la contrepartie

— T : la maturité maximale des contrats

— R : le taux de recouvrement (fraction de l'exposition que la banque reçoit en cas de défaut de la contrepartie).

— $E(\tau)$: l'exposition de la banque au moment du défaut de la contrepartie

— $D(\tau)$: coefficient d'actualisation

On peut alors en déduire l'expression de la CVA, qui correspond à l'espérance des pertes de la banque dues au défaut de la contrepartie :

$$CVA = \mathbb{E}[L] = (1 - R) \int_0^T EE(t) dP_\tau(t)$$

avec :

— $EE(t) := \mathbb{E}[E(t)D(t)]$ l'exposition espérée actualisée de la banque

— $P_\tau(t) := \mathbb{P}(\tau \leq t)$: la probabilité que la contrepartie fasse défaut avant l'instant t. $dP_\tau(t)$ représente alors la probabilité que la contrepartie fasse défaut autour de la date t.

Maintenant que nous avons une expression mathématique de la CVA, nous allons étudier la méthode benchmark de calcul qui est la méthode de Monte-Carlo.

3.1 Méthode de Monte carlo pour le calcul de La CVA

La principale difficulté liée au calcul de la CVA est l'estimation de l'exposition espérée. En effet, ce paramètre n'admet pas, en général, de formules explicites. La méthode de Monte-Carlo est l'approche la plus standard pour calculer ce paramètre et en déduire l'estimation de la CVA par intégration numérique. Nous détaillons les étapes de cette méthode :

— Générer M scénarios de marché sur une grille de temps $0 = t_1, \ldots, t_N = T$.

— Pour chaque scénario j et pour chaque date t_k on calcule la valeur du portfeuille $V^{(j)}(t_k)$

— On en déduit l'exposition de la banque vis à vis de la contrepartie pour le scenario j à la date t_k :

$$E^{(j)}(t_k) = \max\left\{V^{(j)}(t_k) - C; 0\right\}$$

— Après avoir effectué les M scénarios, on peut calculer l'exposition espérée de la banque pour chaque date t_k :

$$EE\left(t_k\right) = \frac{1}{M}\sum_{i=1}^{M} D^{(j)}\left(t_k\right) E^{(j)}\left(t_k\right)$$

— On peut finalement calculer la CVA :

$$CVA = (1-R)\sum_{k=1}^{N} EE\left(t_k\right)\left[P\left(t_k\right) - P\left(t_{k-1}\right)\right]$$

4 Couverture de l'exposition en CVA

L'exposition au risque CVA peut être réduite en mettant en place une couverture. Pour ce faire il faudrait couvrir chaque composante de ce risque, notamment : le spread de crédit, les facteurs de risque de marché qui déterminent l'exposition en risque de contrepartie ainsi que les facteurs de corrélation entre le spread de crédit et les facteurs de risque de marché.

La couverture de la composante risque de crédit consiste à mettre en place un portefeuille de CDS (credit default swap) de maturités différentes (structure par terme de l'exposition). Dans le cas où la liquidité serait manquante sur les CDS « single name », une couverture peut être mise en place à partir de CDS sur Indices pour un portefeuille de produits dérivés avec plusieurs contreparties.

La couverture de la composante risque de marché de la CVA consiste à prendre des positions en produits dérivés sur les facteurs de risques sous-jacents (couverture en delta, gamma, volatilité).

5 Article : Multivariate Gaussian Process Regression

Dans cette partie, nous allons présenter plus particulièrement l'algorithme de régression par processus gaussien. Nous rappellerons dans un premier temps quelques définitions et propriétés de cette classe de processus et leur application sur des problématiques de machine learning. Puis nous détaillerons la phase d'entraînement de ce modèle et la détermination des hyperparamètres. Enfin, nous expliciterons la phase de prédiction via ce modèle. Quelques définitions et propriétés Un processus gaussien f^* est un processus stochastique tel que chaque collection finie

$$\left\{f^*\left(x_1\right), \ldots, f^*\left(x_n\right); 1 \leq i \leq n\right\}$$

suit une loi normale multivariée. Nous pouvons voir le processus gaussien comme la généralisation de la loi normale multivariée avec une infinité de dimension. A ce titre, un processus gaussien $f^* : \mathbb{R}^p \mapsto \mathbb{R}$ est caractérisé par sa fonction moyenne μ et sa fonction de covariance k (aussi

appelée la fonction noyau) : $f^* \sim \mathcal{GP}(\mu, k)$

$$\begin{aligned} &\text{avec } \forall x, x' \in \mathbb{R}^p : \\ &\qquad \mu(x) = \mathbb{E}\left[f^*(x)\right] \\ &\qquad k\left(x, x'\right) = \operatorname{cov}\left(f^*(x), f^*\left(x'\right)\right) = \mathbb{E}\left[\left(f(x) - \mu(x)\right)\left(f\left(x'\right) - \mu\left(x'\right)\right)\right] \end{aligned}$$

A partir de ces deux fonctions, nous pouvons déterminer le vecteur moyenne m et la matrice de covariance K du vecteur aléatoire $\left\{f^*\left(x_1\right), \ldots, f^*\left(x_n\right); 1 \leq i \leq n\right\}$ comme suit :

$$\begin{aligned} m &= \left(\mathbb{E}\left[f^*\left(x_1\right)\right], \ldots, \mathbb{E}\left[f^*\left(x_n\right)\right]\right) \\ K &= \begin{bmatrix} k\left(x_1, x_1\right) & \ldots & k\left(x_1, x_n\right) \\ \vdots & \ddots & \vdots \\ k\left(x_n, x_1\right) & \ldots & k\left(x_n, x_n\right) \end{bmatrix} \end{aligned}$$

Le choix de la fonction noyau k est déterminant dans l'analyse des processus gaussien. Cette fonction, au même titre que la matrice de covariance d'une loi normale multivariée, doit être symétrique, définie et positive. Un choix commun, en particulier pour l'utilisation des processus gaussiens en machine learning, est la fonction de base radiale (RBF) définie par $\forall x, x' \in \mathbb{R}^p$:

$$k\left(x, x'\right) = \sigma_f^2 \exp\left(\frac{-\left\|x - x'\right\|^2}{2l^2}\right)$$

avec $\sigma_f, l \in \mathbb{R}$ des hyperparamètres. Nous allons voir à présent, comment utiliser cette classe de processus dans des problématiques de régression en machine learning. Utilisation des processus gaussiens en machine learning

En machine learning, la régression par processus gaussien est un algorithme de régression supervisée. Considérons une base de données d'entraînement $\{\{(x_i, y_i) \mid i = 1, 2, \ldots, n\}\}$ où chaque x_i est un vecteur d'input de dimension d et y_i est l'output associé. L'objectif est de trouver la relation entre les inputs et les outputs, pouvant être formulée de la manière suivante :

$$y_i = f\left(x_i\right) + \epsilon_i \quad 1 \leq i \leq n$$

où les ϵ_i sont des lois normales centrées réduites indépendantes et identiquement distribuées représentant le bruit des données.

L'idée derrière la régression par processus gaussien est de définir une distribution a priori pour la fonction f (cette distribution étant alors un processus gaussien), et de mettre à jour cette distribution et le bruit compte-tenu des observations de la base d'entraînement : $\mathbb{P}(f \mid X, y)$. C'est le principe de l'inférence bayésienne. La mise à jour de cette distribution consiste à trouver les hyperparamètres optimaux de la fonction kernel du processus gaussien et du bruit des données. Nous allons détailler cette étape, que l'on appelle la phase d'entraînement du modèle de régression par processus gaussien, dans la prochaine partie. Phase d'entraînement du modèle de régression par processus gaussien

Comme évoqué dans la partie précédente, le choix de la fonction kernel est déterminant et va influencer grandement la performance de notre modèle. En effet, cette phase consiste à calibrer les

hyperparamètres Θ de cette fonction ainsi que le paramètre de bruit σ_n (ex : $\Theta = \{\sigma_f, \sigma_n, l\}$ pour un RBF). L'approche la plus classique pour les déterminer est de maximiser le logarithme de la fonction de vraisemblance marginale : $l(\Theta) = \ln p(y \mid X, \Theta)$. Cette fonction traduit la probabilité d'observer nos outputs, sachant les inputs et les hyperparamètres de la fonction kernel. Nous souhaitons alors trouver les hyperparamètres maximisant cette probabilité : $\hat{\theta} = \arg\max_\theta l(\theta)$ Voici l'expression de la fonction de vraisemblance :

$$l(\theta) := \log p(y \mid x, \theta) = \underbrace{-\frac{1}{2} y^T \left(K + \sigma_n^2 \times I\right)^{-1} y}_{\text{data fit}} \underbrace{-\frac{1}{2} \log \left|K + \sigma_n^2 \times I\right|}_{\text{pénalité}} \underbrace{-\frac{n}{2} \log 2\pi}_{\text{normalisation}}$$

Nous pouvons exprimer alors les dérivées partielles de l en fonction des hyperparamètres et calculer ceux optimaux :

$$\begin{aligned} \frac{\partial l(\theta)}{\theta_i} &= \frac{1}{2} y^T K^{-1} \frac{\partial K}{\partial \theta_i} K^{-1} y - \frac{1}{2} \operatorname{tr}\left(K^{-1} \frac{\partial K}{\partial \theta_i}\right) \\ &= \frac{1}{2}\left(\omega\omega^T - K^{-1} \frac{\partial K}{\partial \theta_i}\right) \end{aligned}$$

avec $\omega = K^{-1} y$. Nous en déduisions ainsi une distribution dite "a posteriori" de notre processus gaussien (ie : la distribution ayant les hyperparamètres calibrés via la méthode précédente).[2]

5.0.1 Prédictions par un modèle de régression par processus gaussien

Sachant une nouvelle combinaison X^* de n^* inputs, nous souhaitons prédire la valeur de l'output associé $y^* = f(X^*)$ grâce à notre modèle de régression par processus gaussien entraîné (et donc via sa distribution a posteriori).

En notant X l'ensemble des n inputs $(x_1, x_2, \ldots, x_n)$ de la base d'entraet y les outputs associés, nous rappelons la relation suivante (1) entre les inputs et les outputs :

$$y = f(X) + \sigma_n \epsilon$$

avec $f \sim \mathcal{GP}(0, k(x, x'))$ et $\epsilon \sim \mathcal{N}(0, 1)$. Il est donc clair que $y \sim \mathcal{N}(0, K(X, X) + \sigma_n^2 \times I)$ avec I la matrice identité de dimension n (le nombre d'observations). De la même manière, la prédiction $y^* = f(X^*) \sim \mathcal{GP}(0, K(X^*, X^*))$ avec $K(X^*, X^*) = k(x^*, x^*)$ De ces deux résultats, nous déduisons la distribution jointe des outputs y d'entrainement et de la prédiction f^* :

$$\begin{pmatrix} y \\ y^* \end{pmatrix} \sim \mathcal{N}\left(\begin{bmatrix} 0 \\ 0 \end{bmatrix}, \begin{bmatrix} K(X, X) + \sigma_n \times I & K(X, X^*) \\ K(X^*, X) & K(X^*, X^*) \end{bmatrix}\right)$$
$$\text{avec } K(X^*, X) = \left[k(x^*, x_1), k(x^*, x_2), \ldots, k(x^*, x_n)\right].$$

Notons que $y = f(x)$ lorsque que nous supposons notre base d'entrainement sans bruit.

Nous sommes alors intéressé au calcul de la probabilité conditionnelle suivante $p(y^* \mid X, y, f, X^*)$: sachant les observations et la distribution a posteriori que l'on a, qu'elle est la probabilité d'observer la prédiction y^*. D'après le fameux résultat de la distribution conditionnelle d'une loi normale bivariée (preuve en Annexe A), on a :

$$\begin{aligned} p(y^* \mid y) &\sim \mathcal{N}\left(K^* \times K^{-1} y; K^{**} - K^* \times K^{-1} \times K^{*T}\right) \\ &\text{avec } K^* = K(X^*, X) \text{ et } K^{**} = K(X^*, X^*) \end{aligned}$$

Nous obtenons alors une distribution pour la prédiction, à partir de laquelle nous pouvons déterminer le meilleur estimateur de y^* et l'erreur commise. Le meilleur estimateur correspond à la moyenne a posteriori de la distribution précédente :

$$\bar{y}^* = \mathbb{E}\left[y^* \mid y\right] = K^* \times K^{-1} y = K^* \times K^{-1} f(X)$$

Et l'erreur commise correspond à la variance a posteriori :

$$\text{error}\left(y^*\right) = \text{Var}\left[y^* \mid y\right] = K^{**} - K^* \times K^{-1} \times K^{*T}$$

Nous pouvons interpréter l'estimateur de y^* (la moyenne a posteriori) de la manière suivante : il peut être vu comme une combinaison linéaire des valeurs de la fonctions kernel. En effet :

$$\bar{y}^* = \sum_{i=1}^{n} \omega_i k\left(x^*, x_i\right)$$

avec $\omega_i = K^{-1} f(X)$ (ce qui permet de mieux comprendre l'importance du choix de la fonction kernel). Avantages :

- Erreur de prédiction très minimale.
- Interprétation bayésienne probabiliste / incertitude quantification (UQ)
- Utilisation de la probabilité marginale pour définir les hyperparamètres du processus gaussien.
- Les processus gaussiens peuvent gérer des données bruyantes
- Peut utiliser un arbitraire, éventuellement non structuré (par exemple, de manière stochastique simulé) «grille» d'observations

Inconvénients :

- Le problème de calcul avec les processus gaussiens, c'est que le temps d'entrainement pour chaque processus gaussien est en $O(n^3)$ donc demande plus de temps que celui des méthodes simples.
  Avec n est le nombre d'observations dans le cas d'un seul facteur de risque. Dans le cas où on a plusieurs facteurs de risque, n sera $n = \prod_i n_i$ où les n_i : le nombre d'observations du i^{me} facteur de risque.

5.1 Multivariate Gaussian Process Regression for CVA computation

5.1.1 Entrainement du modèle

L'entrainement se fait en maximisant la fonction évidence qui repésente la probabilité de l'output du modèle sachant les input et les hyperparamètres :

$$\log p(\mathbf{y} \mid \mathbf{x}, \lambda) = -\left[\mathbf{y}^\top \left(K + \sigma^2 I\right)^{-1} \mathbf{y} + \log\det\left(K + \sigma^2 I\right)\right] - \frac{n}{2}\log 2\pi$$

$\lambda^* = \arg\max_{\lambda} \log p(Y \mid X, \lambda)$

5.1.2 Calcul de la CVA

$\text{CVA}_0 = (1-R)\int_0^T \mathbb{E}\left[\pi_u^+ N_u^{-1}\delta(u,\tau)du\right]$
avec :

- N_t : numéraire au temps t
- π_t : la valeur du portefeuille au temps t
- τ : temps de défaut

En supposant que le temps de défaut τ est indépendant de la valeur du portefeuille et du numéraire :

$$\text{CVA}_0 = (1-R)\int_0^T \mathbb{E}\left[\pi_u^+ N_u^{-1}\right] p(u)du$$

.

Pour évaluer l'exposure $\mathbb{E}\left[\pi_u^+ N_u^{-1}\right]$ on considère l'ensemble des dates suivant : $t_1, ..., t_n = T$ où π ne dépend du temps qu'à travers les facteurs de risque X.

La probabilité de défaut sera modélisée par une densité exponentielle :

$$P\left(t_i \leq \tau < t_{i+1}\right) = \exp\left\{-\int_{s=0}^{t_i}\lambda(s)ds\right\} - \exp\left\{-\int_{s=0}^{t_{i+1}}\lambda(s)ds\right\}$$

$$\text{CVA}_M = \frac{(1-R)}{M}\sum_{j=1}^{M}\sum_{i=1}^{n}\pi\left(X_{t_i}^{(j)}\right)^+\left(N_{t_i}^{(j)}\right)^{-1}\Delta p_i$$

On va remplacer $\pi\left(X_{t_i}^{(j)}\right)$ par $\left|\mathbb{E}\left[\pi_* \mid X, Y, \mathbf{x}_* = \mathbf{X}_{t_i}^{(j)}\right]\right|$

$$\widehat{\text{CVA}}_M = \frac{(1-R)}{M}\sum_{j=1}^{M}\sum_{i=1}^{n}\left|\mathbb{E}\left[\pi_* \mid X, Y, \mathbf{x}_* = \mathbf{X}_{t_i}^{(j)}\right]\right|^+\left(N_{t_i}^{(j)}\right)^{-1}\Delta p_i$$

Le but est toujours de modéliser l'exposure. Dans notre cas, On va utiliser un modèle "Multi Gaussian Process Regression".
L'erreur à minimiser sera notée ϵ_M :

$$\epsilon_M = \frac{(1-R)}{M}\sum_{j=1}^{M}\sum_{i=1}^{n}1_{(\mathbb{E}[\pi_*|\cdots]>0)}\,\text{cov}\left(\pi_* \mid X, Y, \mathbf{x}_* = \mathbf{X}_{t_i}^{(j)}\right)\left(N_{t_i}^{(j)}\right)^{-1}\Delta p_i$$

[3]

5.1.3 Résultats numériques

Pour calculer la valeur de la CVA, on doit déterminer la valeur du portefeuille à chaque instant et pour chaque simulation du sous-jacent.On a entrainé notre modèle dans deux cas. Un premier cas consiste à utiliser un seul facteur de risque (un seul sous-jacent). Un deuxième cas consiste à

utiliser plusieurs facteurs de risque (plusieurs sous-jacents). Notre portefeuille est construit à partir de l'achat de deux call et la vente d'un put avec la configuration suivante : $KC = 110, KP = 90, r = 0.05, \sigma = 0.25, T = 2.0, S0 = 100$ où

KC : le strike associé au call, KP : strike associé au put

r : taux sans risque, σ : la volatilité, T : maturité . 1^{er} cas : Calcul de la CVA pour un portefeuille avec un seul facteur de risque(un seul sous-jacent) Dans cette partie nous souhaitons déterminer la CVA d'un portefeuille composé de l'achat de deux call de strike KC et d'un put de strike KP sur une meme actif sous jacent ayant comme dynamique celle du modéle de Black & Scholes.

On commence d'abord par évaluer la valeur du portefeuille à chaque instant en utilisant la formule fermée obtenue dans le modéle de Black & Scholes comme benchmark pour déterminer la valeur des options à chaque instant. On entraine ensuite un modéle de regression gaussienne sur les données obtenues avec l'approche benchmark.
Ce modéle entrainé est enfin utiliser pour predire la valeur du portefeuile en de nouveaux points. Pour évaluer notre modele nous comparons les nouvelles valeurs prédites avec la valeur exacte calculée avec la fonction de BlackScholes en utilisant l'erreur quadratique, RMSE.

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{n}}$$

avec y_i la valeur réelle, $\hat{y}_i$ et n le nombre d'observations

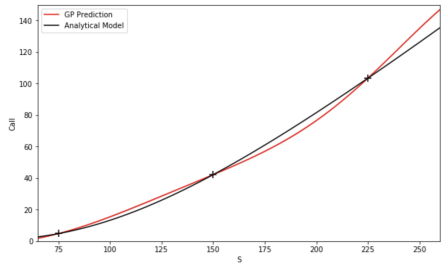


FIGURE 1 – Call price

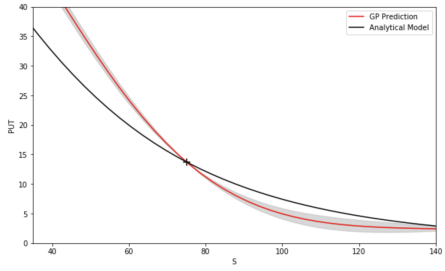


FIGURE 2 – Put price

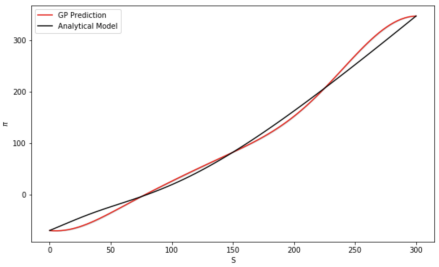


FIGURE 3 – portfolio value

Nous remarquons que les prix obtenus par formule fermée sont trés proche de ceux obteue par regression gaussienne meme si le decalage est plus conséquent pour le put. La valeur du portfeuille resultant est quant à elle trés proche de la vraie valeur(celle obtneue par formule fermée).

```
In [16]:
from sklearn.metrics import mean_squared_error
import math

MSECALL = mean_squared_error(np.array(portfolio['call']['y_tests']), np.array(portfolio['call']['preds']))
MSEPUT= mean_squared_error(np.array(portfolio['put']['y_tests']), np.array(portfolio['put']['preds']))

In [17]:
print('RMSE call',math.sqrt(MSECALL))
print('RMSE put',math.sqrt(MSEPUT))

RMSE call 0.10128318695344801
RMSE put 0.0864691399549336
```

FIGURE 4 – RMSE du modéle de regression gaussienne

Nous voyons que le RMSE est trés faible , de l'ordre de 0.1 pour le call et le put.
Une fois que la valeur du portefeuille est calculée par la méthode de regression gaussienne on passe au calcul de la CVA.
Pour cette partie l'approche benchmark utilisée est celle décrite dans la partie 3.1, c'est a dire la méthode de monte carlo. Cette methode nous permet donc d'obtenir la valuer exacte de la CVA du portefeuille à un intant t donné ainsi que l'intervalle de confinace de cette derniére. Nous pouvons donc ainsi la comparer à celle obtenue par regression gaussienne.

```
In [12]: 1 CVA_0

Out[12]: {'tilde': array([0.17208618], dtype=float32),
          'exact': array([0.1720862], dtype=float32),
          'tilde_up': array([0.18688278], dtype=float32),
          'tilde_down': array([0.15728956], dtype=float32)}
```

FIGURE 5 – Valeur la CVA a t=0 (exact= methode de monte carlo/ tilde=methode de regression gaussienne) ainsi que l'intervalle de confiance

Nous voyons que la valeur obtenue par la méthode de regression gaussienne est trés proche de la valuer obtenue par l'approche benchmark qu'est la methode de Monte Carlo

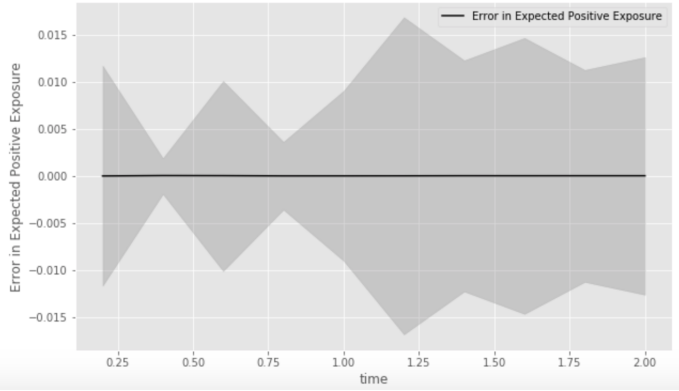


FIGURE 6 – erreur et intervalle de confiance de la regression gaussienne en fonction du temps

5.1.4 2^{eme} cas : Calcul de la CVA pour un portefeuille avec plusieurs facteurs de risque(plusieurs sous-jacents correlés)

Dans cette partie , pour tester les limites de l'approche, nous avons décidé de modifier la composition du portefeuille en remplacant le call et le put par un call basket et un put basket.

5.1.5 Définition et pricing des options basket

L'option,appelée « panier » ou « basket », se classe dans la famille des options sur plusieurs actifs sous-jacents. Cette option ne prend pas en compte la somme des performances de chacun des actifs sous-jacent du panier, pris de façon indépendante, mais elle a les mêmes caractéristiques de remboursement à l'échéance que l'option standard, mais l'actif sous-jacent servant de référence

représente, en fait, un panier de plusieurs actifs équipondérants ou non. Le détenteur de ce panier peut ainsi voir la baisse d'un actif compenser, en tout ou partie, la hausse d'un autre. Dans le cadre de ce projet nous avons choisi d'étudier les options basket suivantes portant sur 10 sous jacents correlés :

$$dS_t^i = S_t^i(\mu dt + \sigma dW_t^i)$$

avec $< W_t^i, W_t^j >= \rho t$

— Call basket de strike K, de maturité T et de payoff

$$\max(\frac{1}{10}\sum_{i=1}^{10} S_T^i - K, 0)$$

— put basket de strike K, de maturité T et de payoff

$$\max(K - \frac{1}{10}\sum_{i=1}^{10} S_T^i, 0)$$

Pour pricer ces options baskets,en l'absence de formule fermée nous avons utiliser la méthode de monte carlo qui est donc utlisée ici comme référence. Pour ce faire il a fallu simuler des mouvements browniens géometriques correlés. Nous avons ensuite, comme dans la partie précédente, entrainé un modéle de régression gaussienne afin d'évaluer la valeur de notre portefeuille et nous l'avons comparé à l'approche benchmark.

```
from sklearn.metrics import mean_squared_error
import math

MSECALL = mean_squared_error(np.array(portfolio['call']['y_tests']), np.array(portfolio['call']['preds']))
MSEPUT= mean_squared_error(np.array(portfolio['put']['y_tests']), np.array(portfolio['put']['preds']))
print('RMSE call  basket',math.sqrt(MSECALL))
print('RMSE put basket',math.sqrt(MSEPUT))
```

```
RMSE call  basket 2.1502485416430663
RMSE put basket 0.24406392721840284
```

FIGURE 7 – RMSE de la regression gaussienne pour pricing d'options baskets

Nous remarquons que l'erreur obtenue pour le prix des call est un peu élevé comparé à celui du put. Il est également important de noter que contrairement au 1er cas, l'entrainement du noyau gaussien est trés time consuming dans le cas des options baskets. Pour une base de données de 150 observations nous avons un temps d'entrainement aux alentours de 400 secondes. Cependant une fois le modéle entrainer la prediction de nouvelle valeur est quant à elle quasi immediate ce qui represente donc un avantage de la methodes.

Une fois que nous avons évalué la valeur de notre protefeuille à l'aide de la régression gaussienne nous pouvons maintenannt passer au calcul de la CVA avec, comme dans le 1er cas la méthode de monte carlo comme méthode benchmark.

Ici également l'entrainement du modéle de régression gaussienne dure assez longtemps ($\simeq$ 500s)

Le resultat obtenu est beaucoup moins precis que celui obtenu dans le cas d'un seul facteur de risque.

```
In [34]: 1 CVA_0
Out[34]: {'tilde': array([0.20623054], dtype=float32),
          'exact': array([0.15888596], dtype=float32),
          'tilde_up': array([0.2454259], dtype=float32),
          'tilde_down': array([0.16703515], dtype=float32)}
```

FIGURE 8 – Valeur la CVA a t=0 (exact= methode de monte carlo/ tilde=methode de regressiongaussienne) ainsi que l'intervalle de confianc

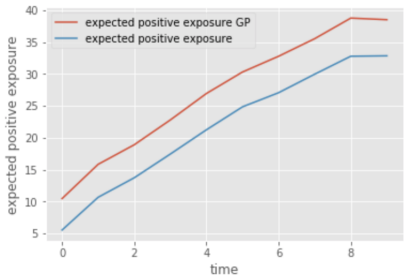


FIGURE 9 – Expected positive exposure

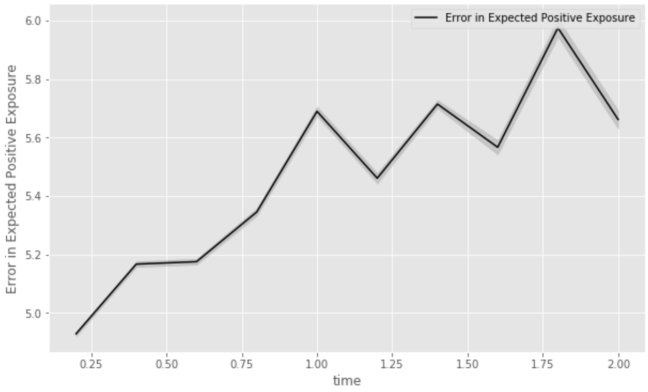


FIGURE 10 – Erreur et intervalle de confiance de la regression gaussienne en fonction du temps

5.2 Conclusion de la méthode des Processus Gaussiens

Nous avons étudié l'utilisation des méthodes de regression gaussienne afin de quantifier le risque de contrepartie notamment à travers le calcul de la CVA et de l'EPE. L'approche a été implémentée dans le cadre d'un portfeuille simple composé d'options européennes sous un seul sous jacent et nous avons vu que la méthode etait trés efficace et une fois le noyaux de regression appris, la predicition de nouvelles valeurs est assez rapide ce qui en fait une bonne alternative aux methodes

tel que le Monte Carlo imbriqué. En étandant l'approche en haute dimension, c'est à dire avec un portefeuille d'options dependants de plusieurs sous-jacents correlées, on observe que l'entrainement du modéle dure beaucoup plus longtemps et les resultats obtenus sont beaucoup moins précis que dans un cas avec un seul facteur de risque.

6 Article Neural network regression for Bermudan option pricing : Bernard Lapeyre & Jérôme Lelong

Cette section vise à implémenter l'idée sous-jacente de l'article **Neural network regression for Bermudan option pricing** qui présente une application des réseaux de neurones pour la valorisation des options bermudiennes. Le principe est de remplacer la régression polynomiale dans le cadre de la méthode LongStaff Schwartz utilisé pour le pricing des produits callables par une approximation à base de réseaux de neurones.

Idée générale :

Une option bermudienne est une option intermédiaire entre les options américaines et européennes. En effet, alors qu'une option américaine peut être exercée n'importe quand jusqu'à la date d'échéance et une option européenne seulement à la date d'échéance, l'option bermudienne peut être exercé à plusieurs dates entre son émission et son échéance.

Si on fixe un horizon de temps donné T et on introduit un espace de probabilité filtré $(\Omega, \mathcal{F}, (\mathcal{F}_t)_{0\leq t\leq T}, \mathbb{P})$ modélisant un marché financier tel que $\mathcal{F}_0$ désigne la filtration triviale. On considère une option bermudienne de dates d'exercices $0 = T_0 \leq T_1 \leq ... \leq T_N = T$ et de payoff actualisé $\tilde{Z}_{T_n}$ si exercé au moment T_n. Dans un marché complet, si on note $\mathbb{E}$ l'espérance sous la mesure risque neutre, la valeur de l'option bermudienne aux instants $(T_n)_{0\leq n\leq N}$ est :

$$V_{T_n} = \sup_{\tau\in\{T_n,T_{n+1},...,T_N\}} \mathbb{E}[Z_\tau/\mathcal{F}_{T_n}] \tag{1}$$

La formule (1) est équivalente à l'équation de programmation dynamique suivante :

$$\begin{cases} V_{T_N} = Z_{T_N} \\ V_{T_k} = max[Z_{T_k}, \mathbb{E}(V_{T_{k+1}}/\mathcal{F}_{T_k})] \quad \forall 0 \leq k \leq N-1 \end{cases} \tag{2}$$

L'espérance conditionnelle $\mathbb{E}(V_{T_{k+1}}/\mathcal{F}_{T_k})$ est dite continuation value. L'approche LongStaff Schwartz est la méthode classique généralement déployée pour calculer cette espérance en utilisant des techniques de régression qui cherche un meilleur ajustement au sens des moindres carrés pour estimer sa valeur.

Cependant l'article propose d'utiliser les réseaux de neurones à la place de la régression pour approcher cette espérance conditionnelle.

Donc dans le cas si $Z_{T_n} = \phi_n(X_{T_n})$, où X définit le facteur de risque sous-jacent considéré un processus markovien, on a :

— LongStaff Schwartz classique $\Longrightarrow$ E($U_{T_{k+1}}/\mathcal{F}_{T_k}$) = Polynôme($X_{T_k}$)
— LongStaff Schwartz NN $\Longrightarrow$ E($U_{T_{k+1}}/\mathcal{F}_{T_k}$) = NN($X_{T_k}$)

Dans la suite, on focalisera notre étude sur un cas plus précis celui d'un swaption bermudien. Il s'agit d'une option bermudienne qui donne à son détenteur le droit d'entrer dans un swap à plusieurs dates durant la durée de vie du contrat.

Si on considère un swaption bermudien de dates d'exercices $\{T_1, ..., T_m\}$ indexé sur un swap de prix $V_{swap}(t)$, où la fonction payoff est $h(t) = \max(V_{swap}(t), 0)$, le prix à chaque date d'exercice sera :

$$\begin{cases} V_{T_m} = h(T_m) \\ V_{T_k} = max[h(T_k), \mathbb{E}(e^{-\int_{T_k}^{T_{k+1}} rsds} V_{T_{k+1}}/r_{T_k})] \quad \forall 0 \leq k \leq m-1 \end{cases} \tag{3}$$

Où $(r_t)_{t\geq 0}$ définit le processus modélisant le taux court.

L'idée de l'article, comme évoqué avant, va consister à approcher l'espérance $\mathbb{E}(e^{-\int_{T_k}^{T_{k+1}} rsds} V_{T_{k+1}}/r_{T_k})$ par un réseau de neurones qui prend en entrée les simulations de taux court (r_{T_k}) et qui admet comme sortie pour son entraînement la quantité $e^{-\int_{T_k}^{T_{k+1}} rsds} V_{T_{k+1}}$ dite la discounted value.

Similairement, pour calculer le prix à une date d'exposition t autres que des dates d'exercices tel que $T_k < t < T_{k+1}$:

$$V_t = \mathbb{E}(e^{-\int_t^{T_{k+1}} rsds} V_{T_{k+1}}/r_t),$$

on utilisera un réseau de neurones $NN(r_t)$ qui prend en entrée les scénarios de r_t et estime la valeur de V_t.

Pour pouvoir générer des scénarios de taux, on aura besoin d'un modèle de diffusion. Dans la section suivante, on présente le modèle Hull&White à 1 facteur, modèle dans lequel on modélisera notre étude de l'article.

Modèle sous-jacent :

On considère le modèle Hull&White qui nous permet d'avoir une dynamique de taux court (r_t) consistante avec la courbe de taux à la date courante :

$$dr_t = \kappa(\theta(t) - r_t)dt + \sigma dW_t$$

Avec :

$$\theta(t) = \frac{1}{\kappa}\frac{\partial}{\partial T} f^M(0,t) + f^M(0,t) + \frac{\sigma^2}{2\kappa^2}(1 - e^{-2\kappa t})$$

Où $f^M(0,t) = -\frac{\partial ln\, P^M(0,t)}{\partial T}$ le taux forward instantané marché en 0.

Pour pouvoir calibrer la fonction $\theta(t)$, on aura besoin de la courbe de taux de la date courante. Dans la suite, pour faire l'exercice, on considère une courbe de taux zéro-coupon d'une soixantaine de maturités :

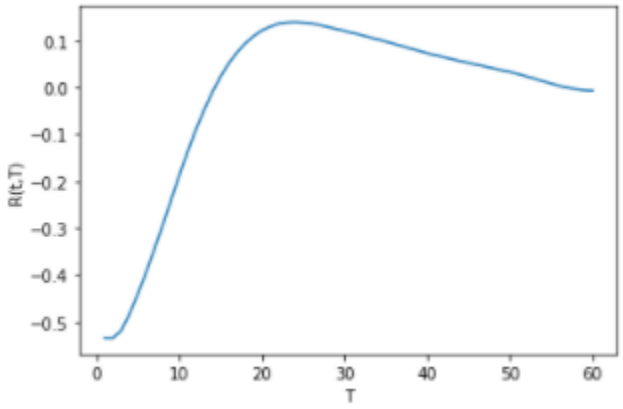


FIGURE 11 – Courbe de taux zéro coupon

Cette courbe est ensuite interpolée par des splines cubiques pour pouvoir calculer le taux forward instantané marché de la date actuelle :

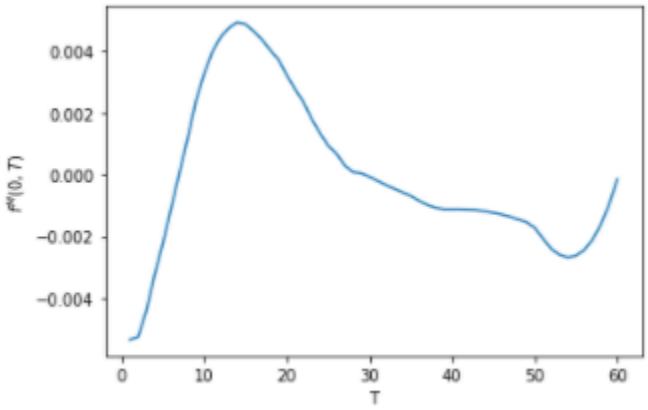


FIGURE 12 – Taux forward instantané marché

Dans le cadre du modèle Hull&White, le prix à l'intant t d'un swap de nominal N, de maturité T_0 et de ténor $T_n - T_0$, payeur de taux fixe k est donnée par la formule fermée suivante :

$$V_{swap}(t) = N[P(t,T_0) - P(t,T_n) - k\sum_{i=1}^{n}\delta_i P(t,T_i)]$$

Où $\delta_i = T_i - T_{i-1},\ \forall 1 \leq i \leq n$ représente la fréquence d'échange des flux et $P(t,T)$ est le prix à t d'une obligation zéro coupon de maturité T donné par :

$$P(t,T) = A(t,T)e^{-B(t,T)r_t}$$

Où :

$$B(t,T) = \frac{1 - e^{-\kappa(T-t)}}{\kappa}$$
$$A(t,T) = \frac{P^M(0,T)}{P^M(0,t)} \exp\big[B(t,T) f^M(0,t) - \frac{\sigma^2}{4\kappa}(1 - e^{-2\kappa t}) B(t,T)^2\big]$$

Exemple numérique :

On considère un swaption bermudien indéxé sur un swap payeur de caractéristisques suivantes :

— Maturité 5Y
— Tenor 5Y
— Fréquence 6M : $\{\tau_k = 5 + 0.5k\}_{1 \leq k \leq 10}$
— Nominal 1
— Taux fixe 0.3%

Les dates d'exercices du swaption bermudien :

$$T_j = 0.25 + 0.5j \quad \forall 0 \leq j \leq 18$$

Les dates d'expositions sont mensuelles :

$$t_i = \frac{i}{12} \quad \forall 0 \leq i \leq 119$$

C'est dans le cadre de cet exemple que se situe notre étude qui suit.

Calibration de H&W :

Avant d'utiliser une diffusion par le modèle Hull&White pour valoriser le swaption bermudien précédent, il faut calibrer le modèle à la base des swaptions co-terminaux qui correspondent à chaque date d'exercice bermudien.

Le modèle Hull&White propose une formule fermée pour calculer le prix à l'instant t d'un swaption européen payeur d'un taux fixe k et de nominal N, de maturité T_α et de ténor $T_\beta - T_\alpha$:

$$\text{Swaption}(t, T_\alpha, T_{\alpha+1}, ..., T_\beta, N, k) = N \sum_{i=\alpha+1}^{\beta} (1_{\{i=\beta\}} + k(T_i - T_{i-1})) ZBP(t, T_\alpha, T_i, K_i)$$

où :

$$
\begin{aligned}
&ZBP(t,T,S,K) = K\ P(t,T)\phi(-d+\sigma_p) - P(t,S)\phi(-d)\\
&\sigma_p = \sigma\sqrt{\frac{1-e^{-2\kappa(T-t)}}{2\kappa}}B(T,S);\ d = \frac{1}{\sigma_p}log(\frac{P(t,S)}{P(t,T)K}) + \frac{\sigma_p}{2}\\
&K_i = A(T_\alpha,T_i)\exp(-\frac{1-e^{-\kappa(T_i-T_\alpha)}}{\kappa}r^*)
\end{aligned}
$$

tel que r^* est la solution de :

$$\sum_{i=\alpha+1}^{\beta}(1_{\{i=\beta\}} + k(T_i - T_{i-1}))A(T_\alpha,T_i)\exp(-\frac{1-e^{-\kappa(T_i-T_\alpha)}}{\kappa}r^*) = 1$$

Cette formule de pricing fermée va nous permettre de définir une fonction de coût qu'on cherchera à optimiser pour calibrer notre modèle et estimer ses deux paramètres (κ,σ). Cette fonction de coût sera la somme des écarts quadratiques entre les prix modèle des swaptions et les prix marché. Ainsi, le problème de calibration s'écrit :

$$(\hat{\kappa},\hat{\sigma}) = \underset{(\kappa,\sigma)}{argmin} \sum_{i\in Maturite}\sum_{j\in Tenor}(Swaption_{ij}^{H\&W}(\kappa,\sigma) - Swaption_{ij}^{M})^2$$

$Swaption_{ij}^{M}$ représente le prix de marché d'un swaption de maturité i et de tenor j. Vu qu'on ne dispose pas de prix de marché, on a calculé à l'aide de la formule (6) les prix Hull&White des swaptions co-terminaux qui constituent le swaption bermudien en fixant $\kappa = 0.02$ et $\sigma = 0.01$, auquels on a ajouté un bruit pour pouvoir tester l'approche de calibrage.

Les figures suivantes montrent les swaptions co-terminaux composants du swaption bermudien, leurs caractéristiques : maturité et tenor, ainsi que leux prix Hull&White bruités.

	Maturity	Tenor	Price
0	4.75	5	0.038842
1	4.25	5	0.0368447
2	3.75	5	0.0332653
3	3.25	5	0.0283669
4	2.75	5	0.0271713
5	2.25	5	0.0205188
6	1.75	5	0.0184138
7	1.25	5	0.0138681
8	0.75	5	0.0103067
9	0.25	5	0.0030294

	Maturity	Tenor	Price
9	0.25	5	0.0030294
10	0.25	4.5	0.00220207
11	0.25	4	0.00317679
12	0.25	3.5	0.0030563
13	0.25	3	0.00282767
14	0.25	2.5	0.00332892
15	0.25	2	0.00148771
16	0.25	1.5	0.00164005
17	0.25	1	0.00270214
18	0.25	0.5	0.00368225

FIGURE 13 – Swaptions co-terminaux pour le calibrage du modèle Hull&White, La colonne Price est un prix bruité de Hull&White pour $\kappa = 0.02$ et $\sigma = 0.01$.

Après optimisation, on trouve : $\hat{\kappa} = 0.01981959, \quad \hat{\sigma} = 0.00999683$
La fonction de coût vaut : 2.1507964099542434e-05

Visuellement,

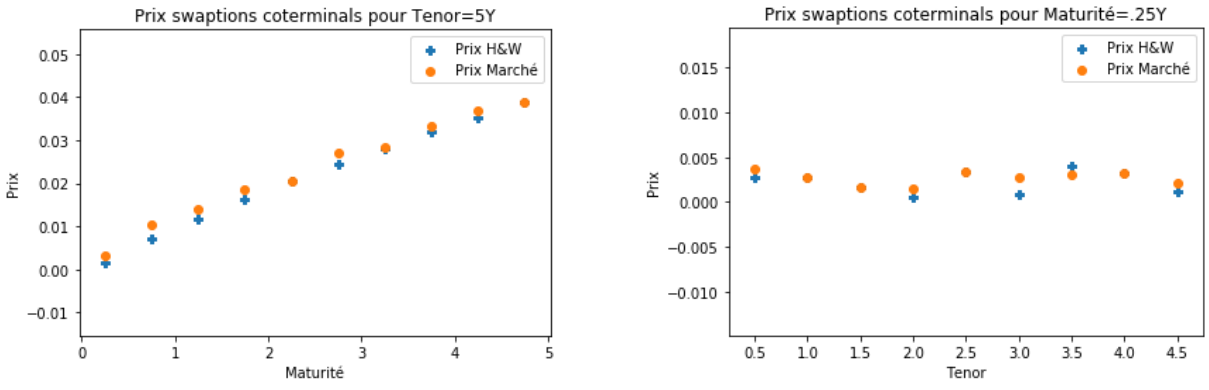


FIGURE 14 – Comparaison entre le prix modèle après optimisation et le prix bruité

Une fois notre modèle calibré, une étape primoridale de l'étude est d'implémenter notre benchmark qui sera effectué dans la section suivante.

Benchmark de l'étude :

Dans cette partie, on va implémenter l'edp de taux du modèle Hull&White pour valoriser un swaption bermudien.

Si on considère un swaption bermudien de dates d'exercices $\{T_1, ..., T_m\}$ indexé sur un swap de prix $V_{swap}(t)$, où on note la fonction payoff $h(t) = max(V_{swap}(t), 0)$. La valorisation du swaption bermudien peut se faire via une approche EDP classique qui s'écrit dans le cadre du modèle Hull&White comme suit :

$$\begin{cases} \text{Si } t \in \{T_1, ..., T_m\} : \\ Max\big[\frac{\partial}{\partial t}V(t,r) + \kappa(\theta(t) - r)\frac{\partial}{\partial r}V(t,r) + \frac{\sigma^2}{2}\frac{\partial^2}{\partial r^2}V(t,r) - rV(t,r), V(t,r) - h(t)\big] = 0 \\ \frac{\partial}{\partial t}V(t,r) + \kappa(\theta(t) - r)\frac{\partial}{\partial r}V(t,r) + \frac{\sigma^2}{2}\frac{\partial^2}{\partial r^2}V(t,r) = rV(t,r) \quad \text{Si } t \notin \{T_1, ..., T_m\} \\ V(T_m, r) = h(T_m) \end{cases} \tag{4}$$

Pour la résolution de (4), on applique un schéma de résolution d'Euler explicite en adoptant une grille (t, r_t) où le temps correspond aux dates d'expositions mensuelles et le taux court varie de -0.1 à 0.25. On obtient ainsi une surface de prix pour la grille considérée représentée dans la figure suivante :

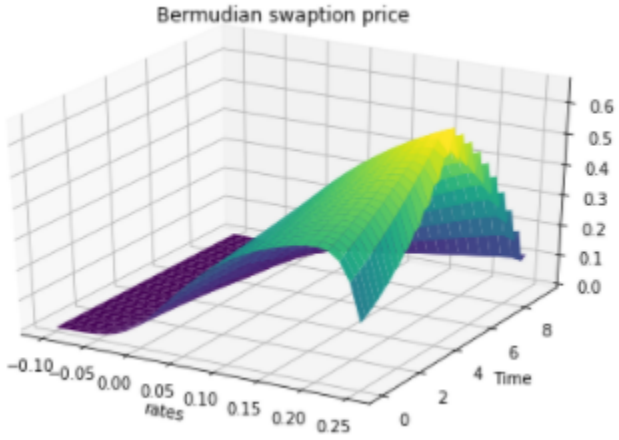


FIGURE 15 – Surface de prix du swaption bermudien par résolution de l'EDP 4

La forme des escaliers descendants visible au bord de la surface de prix ne sont pas dû à des effets numériques, ce sont des chocs naturels qui se propage sur toute la surface en raison de l'expiration des flux du swap sous-jacent au fur et à mesure qu'on avance dans le temps.

Cette surface est ensuite interpolée par des splines cubiques pour pouvoir calculer le prix du swaption bermudien à chaque date et à chaque réalisation du taux court et ainsi obtenir l'EPE du produit en moyennant à tout moment d'exposition les scénarios de prix correspondants aux scénarios de taux issus du modèle Hull&White.

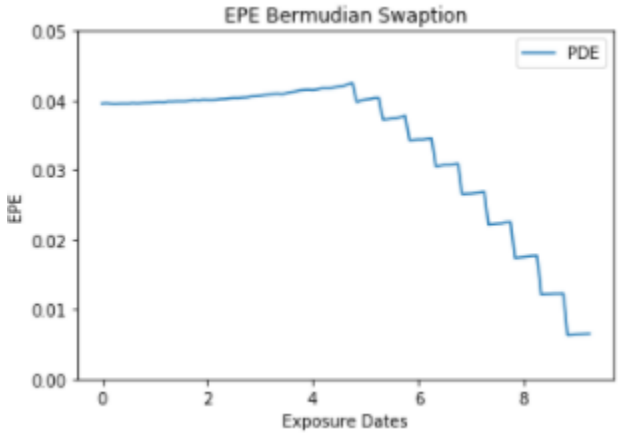


FIGURE 16 – EPE du swaption bermudien par l'approche EDP

Résultats numériques :

Dans cette section, on présente les résultats obtenus dans le cadre de l'implémentation de l'idée de l'article.

Après avoir essayé différents paramétrisations, afin d'assurer la stabilité, la convergence et une erreur minime, nous avons choisi d'utiliser un réseau de neurones avec :

— 2 couches cachées.

— 32 unités par couches.

— la fonction d'activation elu.

— Adam optimizer avec un taux d'apprentissage de $\lambda = 0.001$.

Dans la suite, on montre les résultats de l'implémentation de la méthode LongStaff Schwartz avec réseau de neurones pour estimer l'EPE du swaption bermudien à la base des scénarios de taux du modèle Hull&White, en faisant varier le nombre d'epochs du réseau qui correpond au nombre de passage sur les données pour apprentissage à chaque pas de temps.

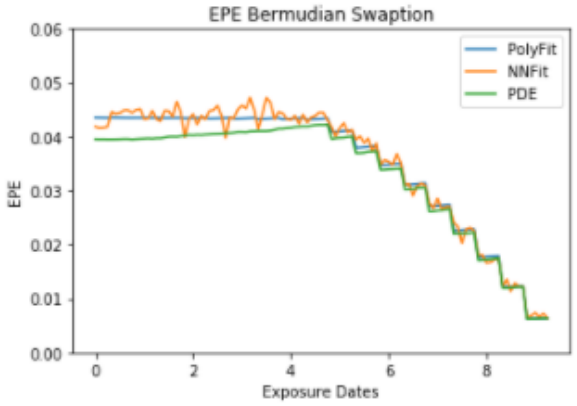


FIGURE 17 – epochs = 30, τ = 276.05328249931335 sec

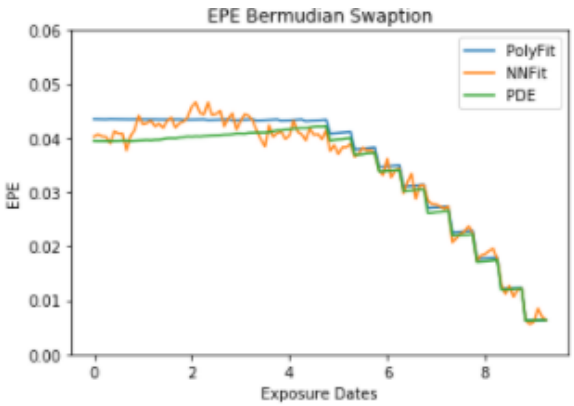


FIGURE 18 – epochs = 20, τ = 186.56576776504517 sec

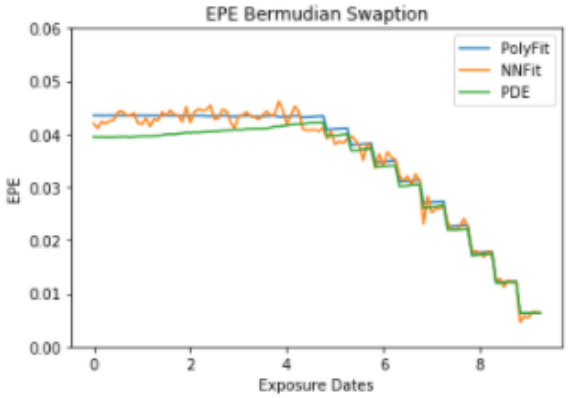


FIGURE 19 – epochs = 10, τ = 89.57721257209778 sec

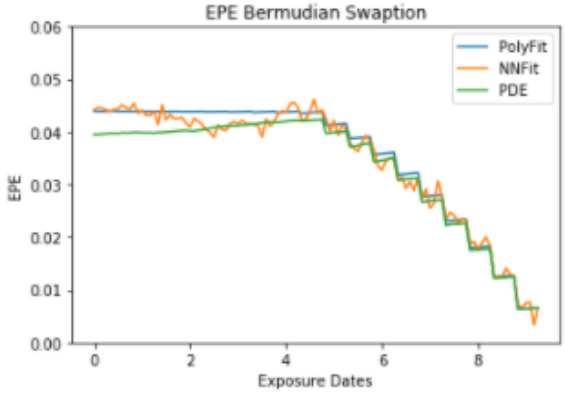


FIGURE 20 – epochs = 5, τ = 52.598469972610474 sec

Sur les figures, on a représenté l'EPE relative au swaption bermudien par les 3 méthodes : LongStaff Schwartz avec régression polynomiale, LongStaff Schwartz avec réseau de neurones pour différents nombres d'epochs, et le résultat de l'approche EDP.

La méthode LongStaff Schwartz avec régression (degré=3) demande un temps de calcul extrêmement petit (τ = 0.5018110275268555 sec) relativement à sa variante à base de réseau de neurones. Cette dernière, en plus de son coût de calcul long par rapport aux méthodes classiques, présente des problèmes de stabilité et convergence de l'EPE. Cependant, il est à noter que contrairement à la regression polynomiale où un polynôme est estimé à chaque pas de temps, un réseau de neurones initialisé au départ de la boucle rétrogade est suffisant pour capturer les continuation values au lieu de générer un réseau à chaque date d'exposition.

Le réseau de neurones nécessite généralement plusieurs passages à travers l'ensemble de données. Pourtant, dans nos exemples cela semblait à peu près inutile principalement parce que la représentation fonctionnelle des espérances conditionnelles (continuation values) ne devrait pas varier beaucoup dans le temps. Ainsi d'après les figures, on remarque qu'un petit nombre d'epochs

est suffisant pour reproduire la même profil de l'EPE ce qui ouvre une éventualité de réduire le temps de calcul en diminuant le nombre d'epochs. La figure ci-dessous montre le profil obtenu dans le cas de 5 epochs :

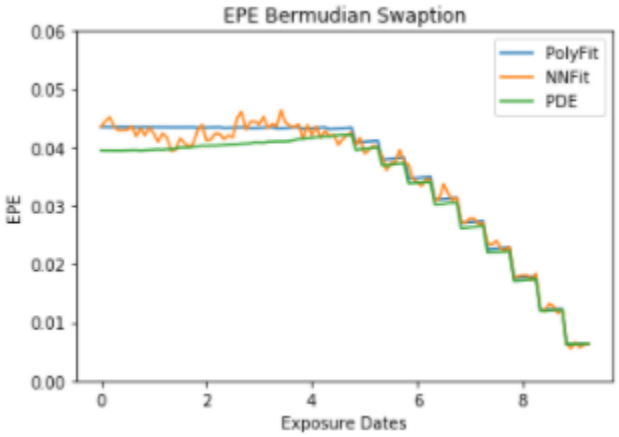


FIGURE 21 – epochs = 5, $\tau = 52.598469972610474$ sec

En fait, une fois que le réseau de neurones a été bien formé, un petit nombre de passage sur les données suffit pour adapter le réseau à une nouvelle date, cela économisera beaucoup de temps de calcul.

D'après ce qu'on voit, l'utilisation des réseaux de neurones n'est pas trop utile pour les problèmes de petite dimension mais peut potentiellement s'adapter beaucoup mieux aux problèmes de grande dimension contrairement à la régression polynomiale qui nécessite un ordre relativement élevé pour fournir des prix précis. Ceci n'étant pas pratique en hautes dimensions car demandera un temps de calcul significatif tandis que le réseau de neurones, comme observé avant, avec un nombre d'epochs très petit est capable de fournir une bonne estimation des continuation values vu que ces dernières ne diffèrent pas beaucoup entre deux dates consécutives.

Dans la figure ci-dessous, on présente à différentes dates d'expositions sur le même graphique les valeurs actualisées (discounted values) et leurs espérances conditionnelles (continuation values), ceci pour la régression polynomiale et le réseau de neurones :

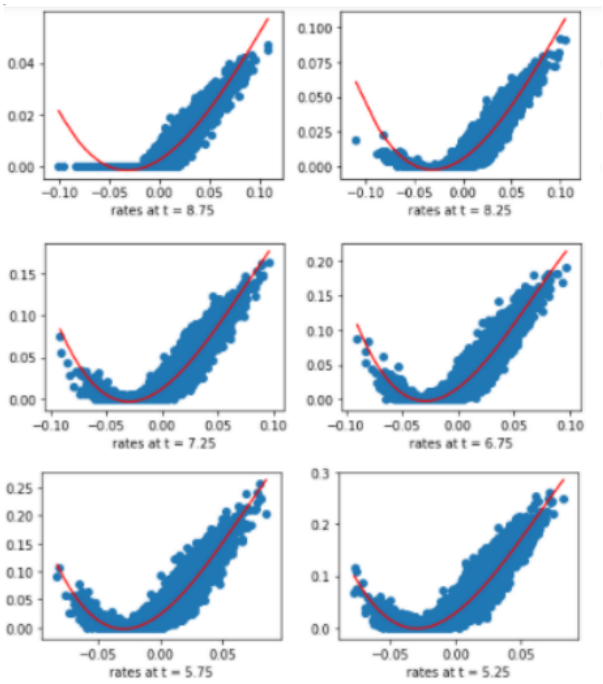


FIGURE 22 – Régression polynomiale deg=3

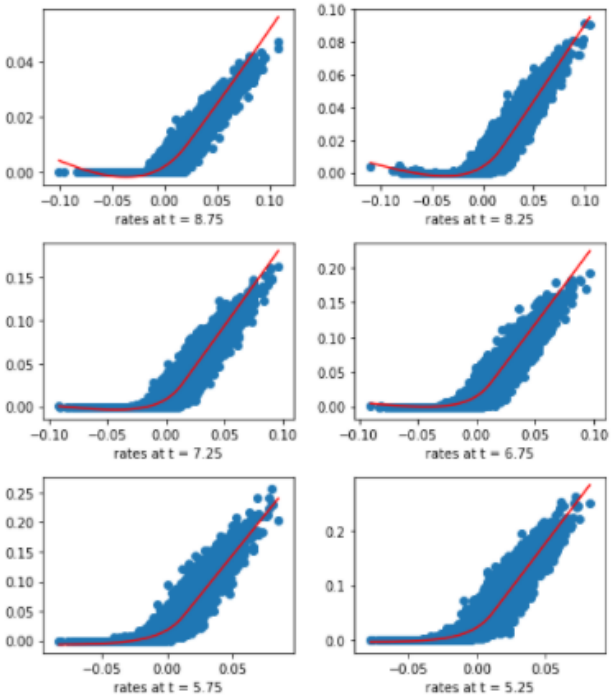


FIGURE 23 – Réseau de neurones

Visuellement, on remarque que le réseau de neurones capture mieux le nuage de points représentative des valeurs actualisées relativement la régression polynomiale.

Conclusion :

Points faibles :

— Temps relativement long par rapport aux méthodes classiques.

— Problèmes de convergence et stabilité de EPE.

— Pas trop utile dans des problèmes de faible dimension.

Points forts :

— Possibilité de réduire le temps de calcul.

— Plus adapté aux grandes dimensions relativement à l'EDP ou la régression polynomiale.

Perspectives : Tester l'approche dans le cadre d'un modèle de taux multi-facteur (Modèle LMM par exemple).

7 NN for CVA

La méthode NN for CVA permet le calcul des prix des options sans devoir à passer par des données d'entraînement. La méthode consiste à la résolution de l'équation différentielle stochastique

du prix de l'option par discrétisation en utilisant du Deep Learning, cela se fait en approchant le δ à chaque instant t présent dans l'équation différentielle par un réseau de neurones. La détermination de la valeur du portefeuille actualisé s'appuie sur l'équation différentielle suivante dans sa forme Forward :

$$\tilde{V}(T_{n+1}) = \tilde{V}(T_n) + \sum_{ij} f_i^{(n)}\left(T_n, \mathbf{X}_n\right) \sigma_{ij}\left(T_n, \mathbf{X}_n\right) \left(W_j(T_{n+1}) - W_j(T_n)\right)$$

où $\tilde{V}$ est le prix actualisé du portefeuille

$f_i^{(n)}\left(T_n, \mathbf{X}_n\right)$ est la fonction δ approché par un réseau de neurone associé au facteurs de risque i qui seront l'entrée du modèle.

L'équation dans sa forme Backward est :

$$\tilde{V}(T_n) = \tilde{V}(T_{n+1}) - \sum_{ij} f_i^{(n)}\left(T_n, \mathbf{X}_n\right) \sigma_{ij}\left(T_n, \mathbf{X}_n\right) \left(W_j(T_{n+1}) - W_j(T_n)\right)$$

La complexité de l'implémentation provient de la nature du réseau de neurones, qui contrairement aux modèles simples, elle ne se fait pas d'une manière directe, mais en passant par le codage de plusieurs classes d'apprentissage. Dans les applications simples du réseaux de neurones, il y une entrée et une sortie du modèle, avec une fonction qui relie les deux. Dans cette application, le réseau est l'équation différentielle, l'entrée c'est les facteurs de risque simulés à chaque instant et la sortie c'est le payoff dans le cas de l'utilisation de l'approche Forward, et le prix dans le cas Backward. Les facteurs de risque sont simulés sur tout le trajectoire à chaque étape d'entraînement. Cette méthode ne nécessite aucune connaissance préalable du prix de l'option. Le prix va découler de l'optimisation réalisée.

Le réseau de neurones calcule les prix en se basant sur l'équation différentielle. Il est constitué de plusieurs sous-réseaux, chaque sous-réseau comporte plusieurs couches : la couche d'entrée qui est les simulations des facteurs de risque à l'instant t, des couches successives : Une couche neuronal, suivi par du BatchNormalisation suivi ensuite par la fonction d'activation (elu dans le cas de notre application) puis la couche neuronal de sortie modélisant la valeur du δ à l'instant t. Une nouvelle classe d'apprentissage est créé à partir de ces différents sous-réseaux et de l'équation différentielle donnant les prix à chaque instant. L'équation différentielle avec la matrice des trajectoires des simulations de facteurs de risque simulés à chaque étape de la descente du gradient avec les simulations des mouvements Browniens associés sont les entrées du modèle.

Lorsque le réseau de neurones s'appuie sur l'équation dans sa forme Forward, la valeur du prix et du δ à l'instant 0 sont pris comme paramètre du modèle, car il ne sont pas connus au départ, il faut par contre spécifier un intervalle pour V_0 pour qu'une première valeur soit initialiser en tirant d"une manière uniforme dans l'intervalle spécifié. La fonction loss à optimiser est $\frac{1}{n}\sum(\tilde{V}_{T,i}^{app} - \tilde{V}_{T,i})$, puisque la valeur du portefeuille à l'instant T correspondant au payoff est connue, les simulation doivent s'approcher au mieux du payoff à l'instant T. Dans le cas de l'utilisation de l'équation dans

sa forme Backward, Le prix est calculé depuis le payoff, il ne faut spécifier ni un intervalle pour V_0 ni δ_0, la fonction loss à optimiser est $\frac{1}{n}\sum(\tilde{V}_{0,i}^{app} - \frac{1}{n}\sum \tilde{V}_{0,j}^{app})$. La variance est ce qui est minimiser dans le cas Backward, car la valeur du portefeuille $\tilde{V}_0$ doit être une constante.

Lors de l'entraînement, dans le cas Forward, le réseau est initialisé, et V_0 est tiré d'une manière uniforme dans l'intervalle spécifié. La spécification de l'intervalle permet seulement d'accélérer l'entraînement, δ_0 est aussi choisi aléatoirement dans l'intervalle $[-1, 1]$. Les paramètres de chaque sous réseau sont initialisées. Les facteurs de risque et le mouvement Brownien sont simulés. Ensuite, le prix est calculé à chaque instant t en commençant par V_0 et δ_0 jusqu'à obtenir V_T et δ_T selon l'équation différentielle stochastique, l'obtention de la valeur de δ_t est donnée par le sous réseau avec les paramètres initialisés du sous réseau à l'instant t, il faut imaginer le sous réseau comme une fonction $f_t(\Theta(t), X_t)$ avec $\Theta(t)$ les paramètres initialisés du sous réseau et X_t les simulations de facteurs de risque à l'instant t. Après obtention du prix V_T le prix est comparé au prix donné par le payoff pour chaque simulation, la descente des gradient va optimiser tous les paramètres et le prix à l'instant 0 pour que le prix à l'instant T soit proche au mieux dans l'étape suivante, les facteurs de risques sont simulés de nouveau, et le calcule de V_T à partir de V_0 est relancé, jusqu'à ce que l'erreur à l'instant T soit assez petite. Dans le cas Backward, les mêmes étapes sont suivis mais dans le sens inverse, V_0 n'est plus comme paramètre du modèles, et la variance est minimisé de V_0.

Lorsque les paramètres du modèle sont optimisés, le réseau peut être utiliser ensuite pour donner les prix de l'option à chaque instant en spécifiant seulement les simulations de facteurs de risque dans le cas Backward, et aussi un intervalle pour V_0 le prix de l'option à l'instant 0 dans le cas Forward.

Dans le cas du Call, on teste les performances des deux approches pour calculer EPE (expected positive exposure) et valider les performances de l'approximation des prix des options en comparant avec la formule fermée du $\tilde{V}_t$ pour plusieurs instants.

Nous n'allons pas spécifier les paramètres et les différentes optimisations car cela sort du scope de ce rapport, on ne va spécifier que les paramètres associés au calcul du prix du Call.

Nous allons aussi examiner le calcule du δ du Call, cela peut faire l'objet d'un article scientifique, car cette étude est absente des papiers sur les approches par Deep Learning. Nous allons cherché à calculer la fonction δ_t de l'option Call, sachant que cette fonction est comprise entre 0 et 1, nous allons ajouter la fonction d'activation sigmoïde en sortie de chaque sous-réseau. Notons quelques éléments pour la comparaison avec la formule fermée.

Le payoff actualisé est :

$$\tilde{V}_T = e^{-rT}(S_T - K)^+$$

Le prix actualisé à l'instant t est :

$$\hat{V}_t = \hat{S}_t\mathcal{N}(d_1) - \hat{K}e^{-rT}\mathcal{N}(d_2)$$

Avec :

$$d_1 = \frac{1}{\sigma\sqrt{T-t}}[\ln(\frac{S_t}{K}) + (r + \frac{1}{2}\sigma^2)(T-t)], \quad d_2 = d_1 - \sigma\sqrt{T-t}$$

La valeur du EPE à l'instant t est égale à :

$$\text{EPE}(t) = \mathbb{E}[max(V_t, 0)] = \mathbb{E}[V_t^+]$$

Dans le cas du Call V_t est positive donc :

$$\text{EPE}(t) = \mathbb{E}[max(V_t, 0)] = exp(rt)\mathbb{E}[\hat{V}_t] = exp(rt)V_0$$

Nous allons effectuer l'exercice de comparaison pour un Call de maturité T=1, de taux sans risque 0.1, de strike K=100, de S_0=100 et de σ=0.25

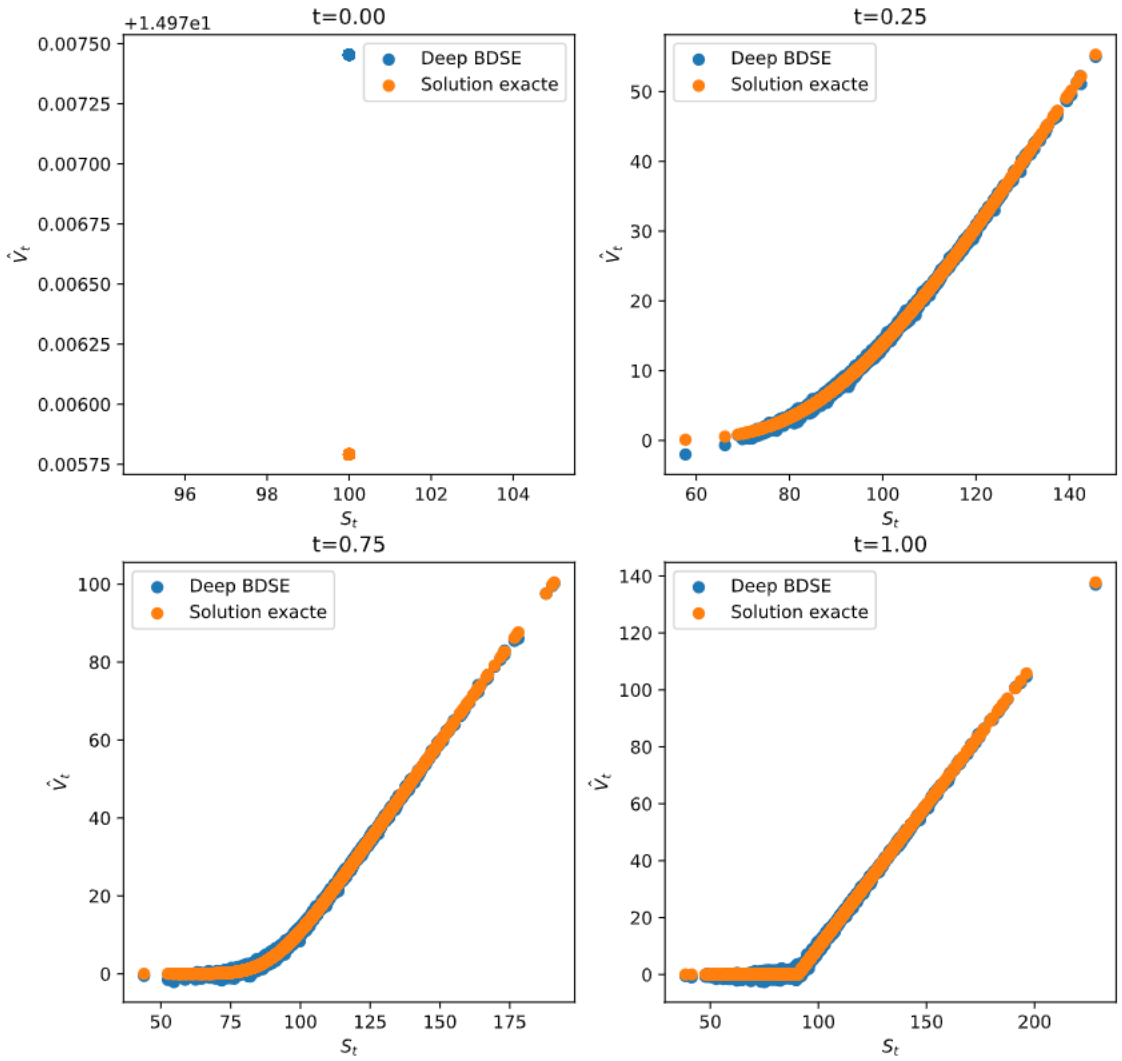


FIGURE 24 – Comparaison en le prix du Call exacte et le prix approximé par Deep BDSE dans sa version Forward.

Le test est effectué sur des nouvelles simulations des facteurs de risque. Les prix par approche BDSE aux instants $t = 0$, $t = 0.25$, $t = 0.75$ et $t = 1$ sont très proches des prix obtenus par formule fermée. Ce réseau de neurones entraîné permet de donner le prix pour n'importe quel valeur du sous-jacent et à un instant t de la discrétisation. Cette approche nécessite par contre de bien spécifier un intervalle du prix de l'option à l'instant 0.

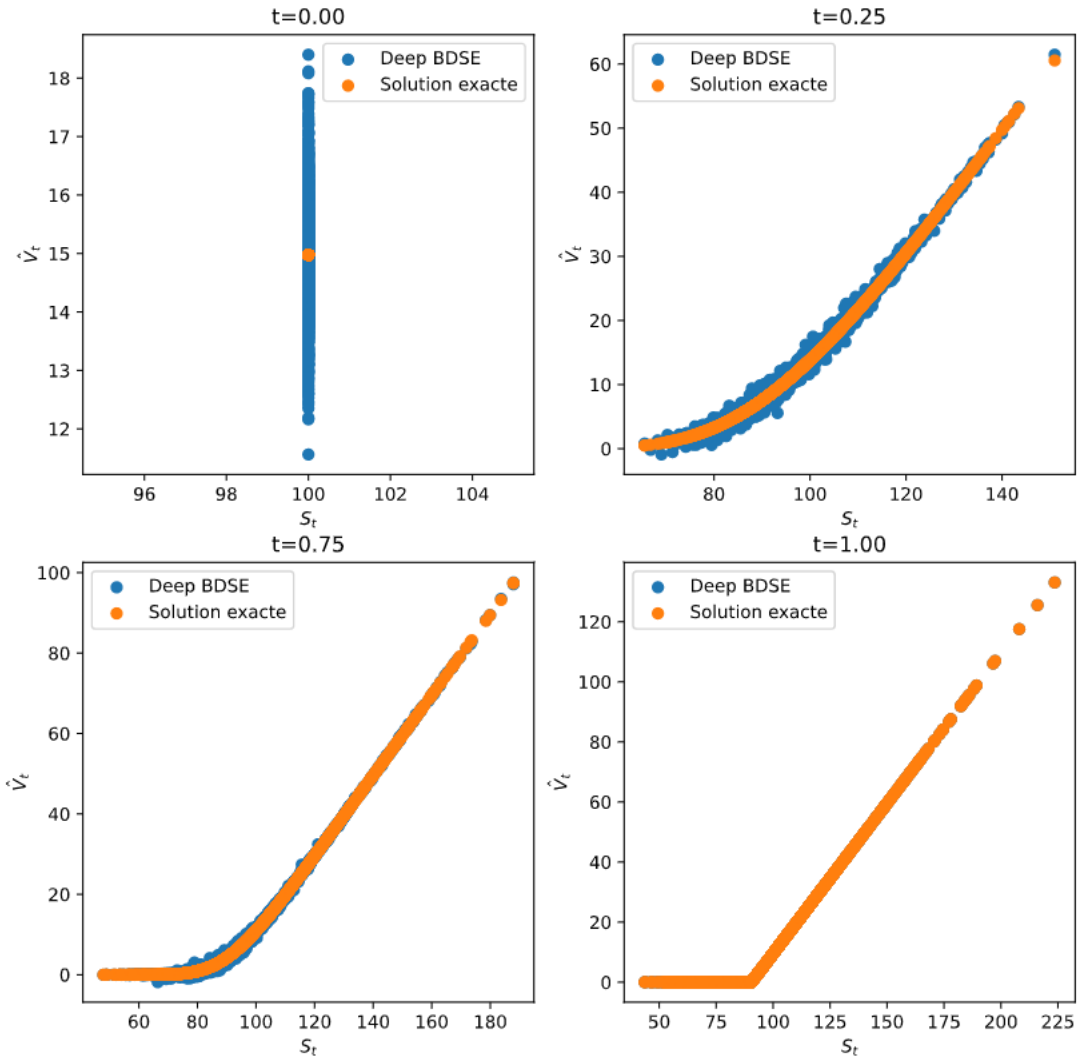


FIGURE 25 – Comparaison en le prix du Call exacte et le prix approximé par Deep BDSE dans sa version Backward.

L'approche Backward donne des résultats moins satisfaisantes qu'une résolution par une utilisation de l'équation différentielle dans sa forme Forward. Elle présente l'avantage de ne pas nécessiter de spécifier un intervalle pour le prix à l'instant 0, en plus, cette méthode pourra être employée dans le pricing des options comme le Call et le Put Américain, le swaption bermudien, qui nécessitent de calculer le maximum entre le prix donné par la résolution de l'équation et du payoff à plusieurs instants.

L'obtention d'une bonne approximation du prix à l'instant 0 dans le cas Backward se fait simplement en moyennant les prix obtenus pour plusieurs simulations, cela permet de minimiser l'erreur dans le pricing, puisqu'on s'attend à ce que le prix à l'instant 0 soit constant.

Nous allons commencer par estimer l'EPE (Expected positive exposure) en testant les deux méthodologies.

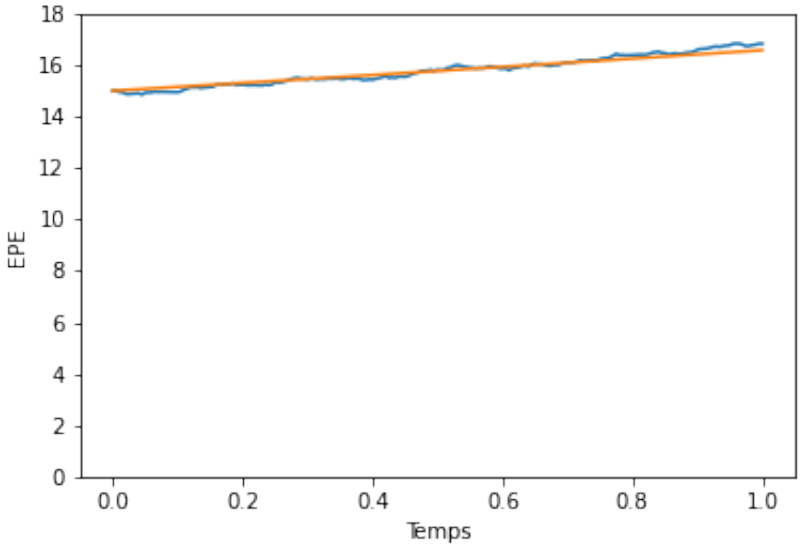


FIGURE 26 – Calcul de la valeur du EPE avec l'approche Forward (Solution exacte en orange, solution approchée en bleu)

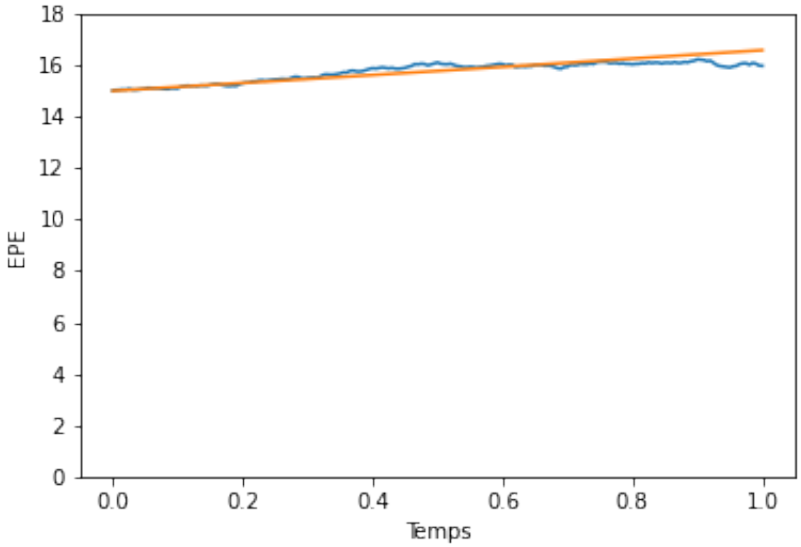


FIGURE 27 – Calcul de la valeur du EPE avec l'approche Backward (Solution exacte en orange, solution approchée en bleu)

L'approche Backward est légèrement moins bonne que l'approche Forward, avec une erreur plus marquée vu l'incertitude présente lors du pricing par l'approche Backward plus on s'approche de l'instant 0, non remarqué lors de l'utilisation de l'approche Forward.

Nous allons aussi examiner le calcul du δ. L'approximation de l'équation différentielle stochastique par Deep Learning avec les approches Forward et Backward permet de sortir aussi le δ à chaque instant de l'option. Sous les mêmes conditions que dans les calculs du Call dans les cas précédente, nous allons examiner le δ dans le cas de l'utilisation de l'approximation Backward.

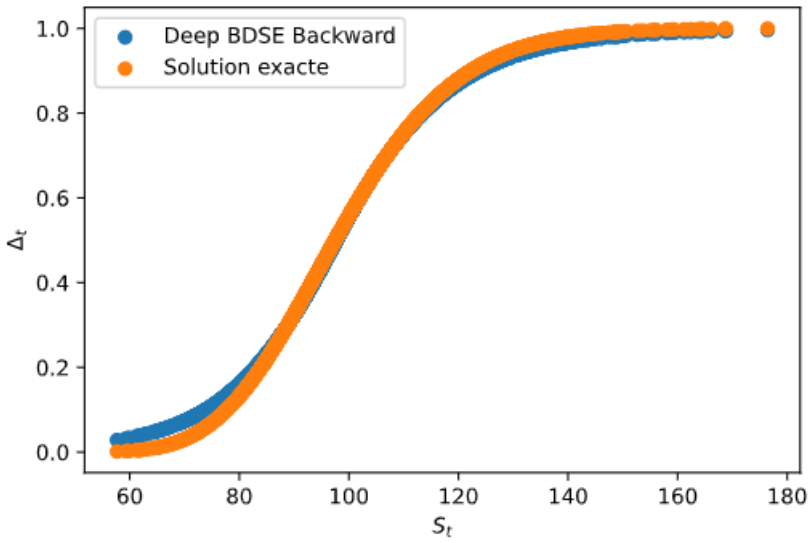


FIGURE 28 – Calcul de la valeur du δ avec l'approche Backward (Solution exacte en orange, solution approchée en bleu) pour $t = T/2$ avec $T = 1$

L'approche a été codé aussi dans le cas du Basket Call de facteurs de risques de volatilités différentes, on a aussi codé cette approche dans le cas de la swaption Bermudiane et de l'option américaine, avec des résultats satisfaisantes pour le calcul de l'EPE. Cette approche peut être adapté pour pricer les options exotiques avec un minimum de changement dans le code. Il suffit de bien choisir les paramètres du modèle et avoir une compréhension globale du code.

Nous allons donner l'exemple d'un Basket Call de trois facteurs de risque de strike $K = 100$, de volatilités $\sigma_1 = 0.2$, $\sigma_2 = 0.25$ et $\sigma_3 = 0.4$, $S_0 = 33$ pour les trois sous-jacents, $r = 0.1$, la maturité $T = 1$. La valeur du portefeuille donné par l'approche Forward est comparée avec celle donnée par la méthode Long Staff Shwartez, en utilisant l'approximation par des polynômes de degré 5, la convariance entre les simulations du mouvement Brownien est $d < W_t^i, W_t^j >= \rho dt = 0.5dt$ constante pour $i \neq j$. Le résultat suivant donne une comparaisons de l'EPE dans les deux cas de figures :

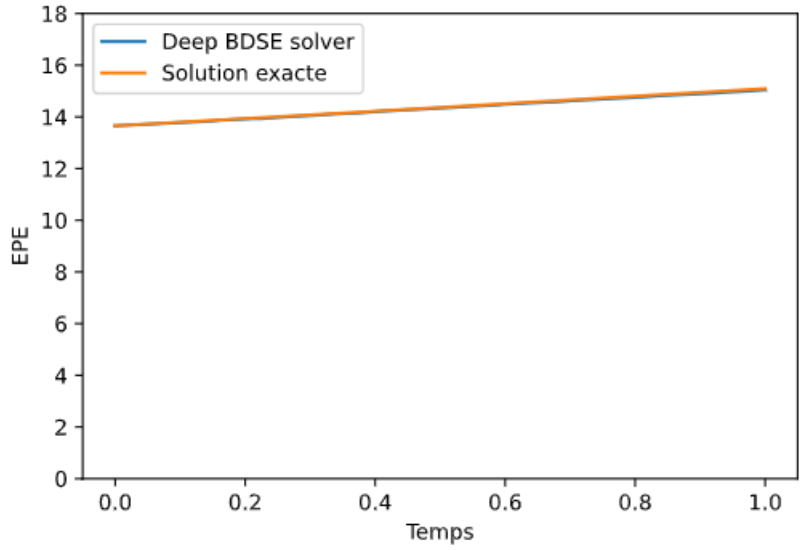


FIGURE 29 – Calcul de l'EPE du Basket avec l'approche Forward (Solution exacte en orange, solution approchée en bleu)

L'approximation est presque parfaite du prix, les résultats sont satisfaisants et confirme l'utilité de cette approche.

Cette approche diffère de l'approche Gaussienne par le fait qu'elle ne nécessite pas des données d'entraînement, elle est adaptée aux grandes dimensions, lorsque le modèle est entraîné, il est rapide de donner les prix à chaque instant d'une option, même dans le cas des grandes dimensions, pour n'importe quel trajectoire de facteurs de risque. L'entraînement est assez rapide avec un GPU. Le modèle présente quelques points faibles, particulièrement, le fait que le modèle ne donne pas un intervalle de confiance des prix approximés contrairement à des approches bayésienne comme l'approche Gaussienne. Le choix des paramétres du modèle nécessite une bonne compréhension du Deep Learning. Le maintien du code et l'implémentation pour des nouvelles options nécessitent une connaissance approfondie des méthodes du Deep Learning. Le modèle se base sur la résolution de l'EDP de l'option qui reste une approximation.

Les simulations dans cette partie ont été réalisées sur une machine Cloud Google EC2 avec 8 coeurs 32 Go de la RAM.

8 Deep Primal Dual Algorithm for BSDES : Applications of machine learning to CVA and IM, Pierre Henry-Labordère

Comme expliqué dans le livre de Green[0](section 3.2), il existe deux approches menant à la formule CVA (espérance de l'*expected positive exposure* sous un certain modèle de crédit pour la probabilité de défaut) : l'une est une approche de calcul par espérance, l'autre se base sur un argument de stratégie de réplication. Plus précisément, concernant cette dernière, on construit

de manière similaire à la réplication Black-Scholes un portefeuille constitué du sous-jacent, d'une obligation risque-neutre et d'une obligation de la contrepartie (avec risque de crédit) de sorte que ce portefeuille soit considéré sans risque. L'application du lemme d'Itô conduit alors à une EDP vérifiée par la valeur de ce portefeuille, dont peut être déduit une EDP vérifiée par la CVA. On voit alors que les approches aboutissent à la même solution, la première étant la représentation Feynman-Kac de la solution à l'EDP.
Si la plupart des méthodes d'approximation numérique de la CVA s'appuient sur cette formule, la méthode développée dans cette article s'intéresse plutôt à l'EDP vérifiée par la CVA. Or, cette EDP est d'une part non-linéaire (non-linéarité donné par la fonction $u \mapsto u^+$ représentant la partie positive de la position mark-to-market) mais aussi à haute dimension (une banque possède généralement de nombreuses transactions avec chaque contrepartie qui doivent toute être prises en comptent dans la CVA). Le fléau de la dimensionnalité empêche l'utilisation des méthodes déterministes comme les différences finies. Il faut alors avoir recours aux méthodes probabilistes Monte-Carlo qui, d'après le théorème central limite, ne souffrent pas d'une augmentation du temps de calcul avec la dimension. Concrétement, pour une discrétisation en temps donnée, l'espérance conditionnelle des valeurs futures du portefeuille, conditionnée par les facteurs de risque observés sur le marché, est approximée par une régression polynomiale sur ces facteurs de risque. Outre la non-optimalité du temps de défaut donné par la discrétisation en temps, le principal défaut de cette méthode est son temps de calcul.
Récemment, une nouvelle approche, nommée *Deep BSDE Solver* [0], a été développé dans la littérature scientifique qui ne requiert pas le calcul d'espérance conditionnelle. Cela consiste à approximer les deltas du portefeuille par un réseaux de neurones afin de minimiser la variance du P&L au temps terminal. L'auteur explique alors que lorsque cette erreur vaut 0, on peut en déduire que l'estimateur a convergé vers la solution exacte. Or, on ne peut rien déduire dans le cas commun d'une erreur non-nul. La valeur ajouté de cet article consiste à utilisé le lient entre les EDP non-linéaires et le contrôle stochastique pour déduire des bornes inférieures et supérieures robustes et rapides à calculer.
Nous présentons donc dans un premier temps la méthode *Deep BSDE Solver* puis détaillons l'obtention des bornes inférieures et supérieures. Enfin, nous explicitions nos résultats numériques et comparaisons avec d'autres méthodes numériques.
Comme expliqué précédemment, la CVA u vérifie l'EDP non-linéaire suivante :

$$\partial_t u + \mathcal{L}u + f(u) = 0, u(T,x) = g(x), x \in \mathbb{R}^d$$

où f est une fonction lipschitzienne et $\mathcal{L} := b\nabla + \frac{1}{2}\sigma\sigma^T\nabla^2$ est le générateur infinitésimal d'un processus d'Itô $X \in \mathbb{R}^d$:

$$dX_t = b(t, X_t)dt + \sigma(t, X_t)dW_t$$

L'EDP précédente peut être représentée probabilistiquement par une BSDE sur deux processus (Y_t, Z_t) adaptés :

$$dY_t = -f(Y_t)dt + Z_t\sigma(t, X_t)dW_t \qquad Y_T = g(X_T)$$

Le lien entre l'EDP et la BSDE est donné par les relations $Y_t = u(t, X_t), Z_t = \nabla_x u(t, X_t)$, ce qui peut être vérifié par application du lemme d'Itô.

De par ces relations, il y a une équivalence entre la résolution de l'EDP pour u et la résolution de la BSDE pour Y. Pour ce dernier problème, il manque la donnée du processus Z, qui peut être interprété comme le delta d'une option de payoff $g(X_T)$ à la maturité, pour pouvoir vérifier la contrainte $Y_T = g(X_T)$. Ainsi, Y_0 est la solution du problème de minimisation :

$$Y_0^* := \arg\min_{\hat{Y_0}} \min_{z_t} \mathbb{E}[g(X_T) - Y_T^z]$$

où

$$dY_t^z = -f(Y_t^z)dt + z_t\sigma(t, X_t)dW_t$$

Ce problème peut ensuite être discrétisé par schéma d'Euler :

$$Y_0^{\Delta,*} := \arg\min_{\hat{Y_0}} \min_{(z_i)_{0\leq i\leq n-1}} \mathbb{E}[g(X_{t_n}^\Delta) - Y_{t_n}^\Delta]$$

avec

$$Y_{t_{i+1}}^\Delta - Y_{t_i}^\Delta = -f(Y_{t_i}^\Delta)\Delta + z_i(X_{t_i}^\Delta)\sigma(t_i, X_{t_i}^\Delta)\Delta W_i$$

C'est alors qu'intervient le machine learning. On approxime la fonction $x \mapsto \nabla u(x,t)$ à chaque temps de discrétisation $t = t_n$ par un réseau de neurones *feedforward* :

$$Z_{t_n} = \nabla u(t_n, X_{t_n}) \approx \nabla u(t_n, X_{t_n}|\theta_n)$$

où θ_n sont les paramètres du réseau de neurones.
On a alors comme entrée du réseau de neurones les simulations $(X_{t_n})_{0\leq n\leq N}$ et $(W_{t_n})_{0\leq n\leq N}$ puis en sorties les $\hat{u}(t_n, X_{t_n})$ comme approximation des $u(t_n, X_{t_n})$. L'algorithme de la descente du gradient stochastique minimse ajuste alors les paramètres du réseau de neurones afin de minimiser la fonction coût, $J(\theta) = \mathbb{E}[g(X_{t_n}^\Delta) - Y_{t_n}^\Delta]$, qui donne après entraînement l'estimation de Y_0^*.
Valeur ajoutée de l'article :
Cette méthode de résolution d'EDP *Deep BSDE Solver* avait déjà été développé auparavant. La valeur ajouté de l'article d'Henry-Labordère et la formulation d'un problème de contrôle stochastique pour le calcul de bornes inférieures et supérieures à $\hat{Y_0}$ et son application au calcul de la CVA.

La solution à l'EDP précédente peut être représentée par un problème de contrôle stochastique (les EDP non-linéaires découlent de l'équation HJB qui elle-même se déduit de principe de programmation dynamique) :

$$u(t,x) = \sup_{a\in A} \mathbb{E}_{t,x}[e^{\int_t^T a_s ds} g(X_T) - \int_t^T e^{\int_t^s a_u du} f^*(a_s)ds]$$

Le choix d'un contrôle arbitraire $\tilde{\alpha} \in A$ donne donc une borne inférieure. Celle-ci est optimale lorsque $\alpha_t^* := \{aY_t - f^*(a)\}$, avec Y_t solution de la BSDE. Le calcul de l'espérance ci-dessus avec $\alpha := \alpha^*$ donne une borne inférieure à $u(t,x)$, Y_0^{lower}.

L'auteur donne ensuite la borne supérieure d'un problème de contrôle stochastique général : $u(t,x) = \inf_Z E[U_{t,T}]$ avec

$$U_{t,T} = \sup_{a\in A} e^{-\int_t^T a_s ds} g(X_T) + \int_t^T e^{-\int_t^s a_u du} f^*(a_s)ds - \int_t^T e^{-\int_t^s a_u du} Z(s, X_s)\sigma(s, X_s)dW_s$$

L'obtention d'une BSDE backward sur U conduit à l'obtention d'une borne supérieure : on peut alors étendre l'algorithme pour incorporer le calcul de Y_0^{Upper}.

Les applications numériques sont faites avec $\Delta t_{Euler} = \frac{1}{100}$, $\Delta t_Z = \frac{1}{20}$, $M = 10000$ (contre $M = 2^{17}$ dans l'article), $\beta = 0.03$,$\sigma = 0.2$,$T = 1$. On a pour $(X_t^i)_{1\leq i\leq d}$ un modèle Black-Scholes non-corrélé : $\frac{dX_t^i}{X_t^i} = \sigma dW_t^i, d < dW^i, dW^j >_t = \delta_{ij}dt, X_0^i := 1$ et comme fonction terminale $g(x_1, .., x_d) = \sum_{i=1}^d (1 - 21_{x_i\geq 1})$.
Par ailleurs, pour réduire la dimensionnalité du problème, l'auteur préconise d'utiliser un nombre de réseau de neurones réduit, en considérant les valeur de Z_t constantes sur plusieurs temps de discrétisation. C'est donc ce que nous avons fait avec $\Delta Z = 5$.
Ci-dessous l'estimation de Y_0 obtenue pour différentes maturités, avec $d = 1$ (résultats de l'article à gauche, puis nos résultats à droite) :

T (years)	Lower bound	Upper bound	Exact(PDE)	BS($\beta = 0$)
2	12.40	12.46	12.40	11.24
4	17.89	18.05	17.89	15.85
6	22.12	22.28	22.12	19.35

TABLE 1. CVA: $\beta = 0.03$, $\sigma = 0.2$ and $d = 1$. $g(x) = 1 - 21_{x>1}$.

	T (years)	Lower Bound	Upper Bound	Y_0 Estimate
0	2.0	12.3810	13.6606	11.1435
1	4.0	18.0174	20.1948	17.6792
2	6.0	22.4119	24.4257	21.1269

Ci-dessous l'estimation de Y_0 obtenue pour différentes dimensions, avec $T = 1$ (résultats de l'article à gauche, puis nos résultats à droite) :

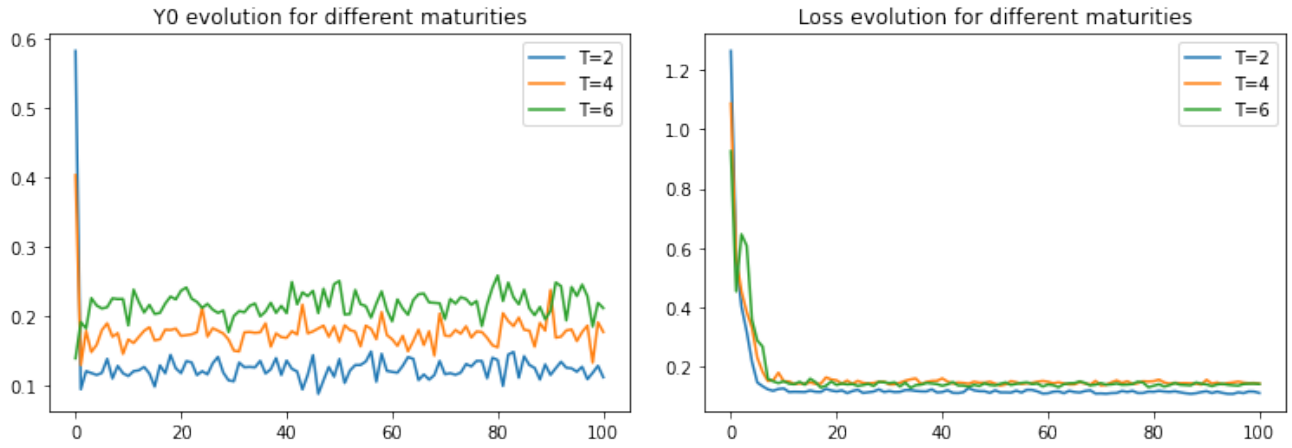
d (number of assets)	Lower bound	Upper bound	BS($\beta = 0$)
2	16.65	16.64	15.91
3	24.75	24.97	23.87
4	32.79	32.81	31.82
5	40.83	40.96	39.78
6	48.80	48.83	47.73

TABLE 2. CVA: $\beta = 0.03$, $\sigma = 0.2$. $g(x_1, \cdots, x_d) = \sum_{i=1}^d (1 - 21_{x_i>1})$. $T = 1$ year.

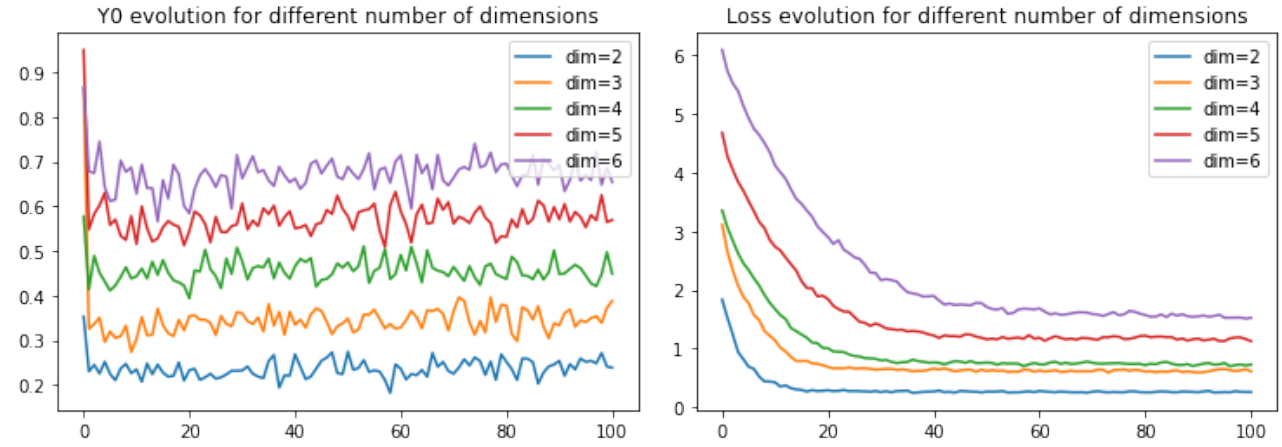
	d (assets_nber)	Lower Bound	Upper Bound	Y_0 Estimate
0	2.0	23.7851	24.0169	23.8353
1	3.0	35.9113	36.7426	38.7889
2	4.0	44.6608	44.6510	44.8838
3	5.0	62.7121	61.2642	56.9545
4	6.0	72.9906	72.2568	65.4087

Si nos résultats ne correspondent pas exactement aux résultats de l'article, nous obtenons tout de mêmes des bornes assez fines avec des estimations relativement proches de celles de l'article. Une explication possible des différences serait l'écart entre le nombre de simulations Monte Carlo utilisées : 10000 contre $2^{17}(\approx 130000)$ dans l'article, dû au temps de calcul trop important. Cependant, certains résultats interpellent. D'une part certaines de nos valeurs tombent en-dehors des bornes, mais aussi à quelques reprises la borne inférieure et et supérieure à la borne inférieure. On remarque que ce phénomène a même lieu pour dans les résultats de l'article pour $d = 2$. Il est possible que cette méthode soit trop dépendante sur les simulations et nécessite un nombre plus important d'itérations.
En terme de temps de calcul, l'algorithme est nettement plus coûteut que celui de Longschtaff-Schwarz, chaque estimation nécessitant environ 5 minutes avec les paramètres utilisés (10000 simulations), contre quelques secondes pour ce dernier. Le temps de calcul est cependant linéaire avec le nombre d'itérations Monte-Carlo (5 minutes pour $M = 10000$ et 1 minute pour $M = 2000$). Par ailleurs, si l'utilisation des réseaux de neurones permet d'apprendre des fonctions fortement non-linéaires, il y a une certaine perte d'interprétabilité du fait du fonctionnement "boîte noire" du réseau de neurones.
Nous avons tracé l'évolution de l'estimation de Y_0 avec le nombre d'itérations de l'algorithme, chaque itération correspondant à une nouvelle simulation de (X_{t_n}, W_{t_n}) calcul de (Z_{t_n}) et minimisation de la fonction coût (on a en abscisse 100 itérations car on feedforward un échantillon de

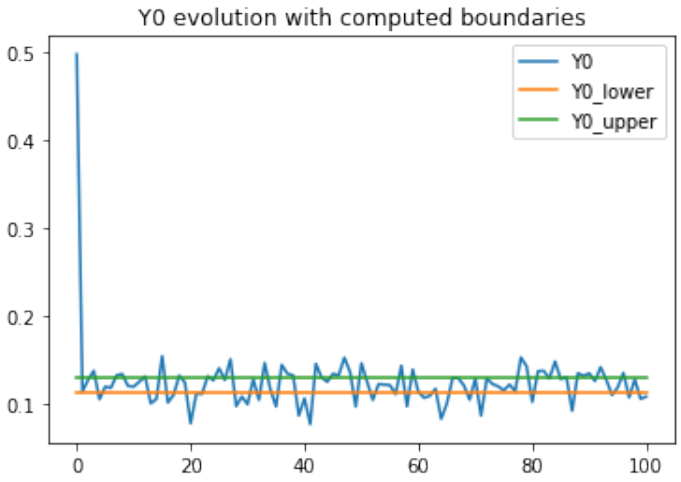
l'ensemble d'apprentissage 99/100 et de validation 1/100). Ci-dessous le cas avec les différentes maturités :



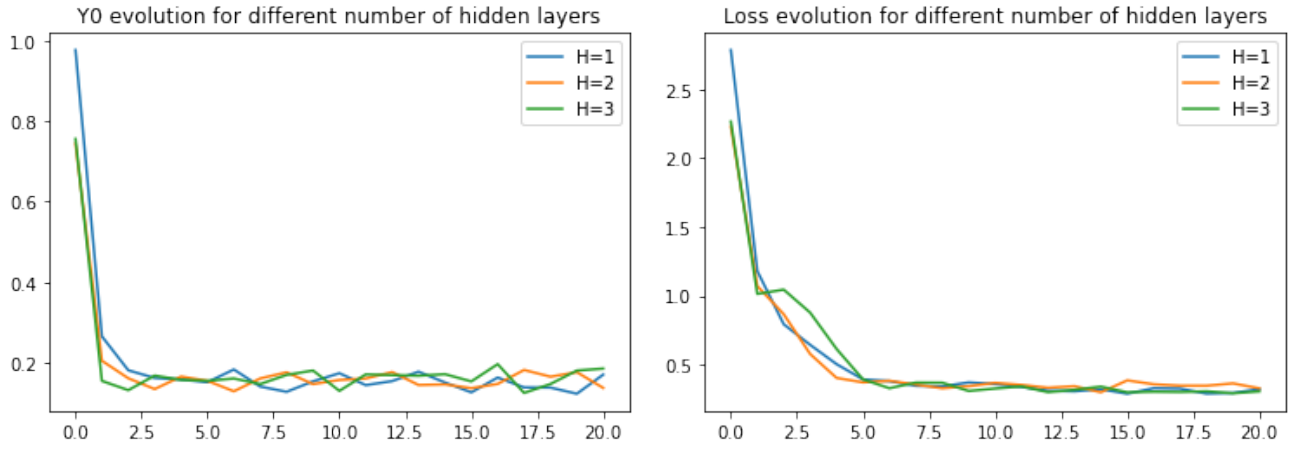
Ci-dessous le cas avec les différentes dimensions :



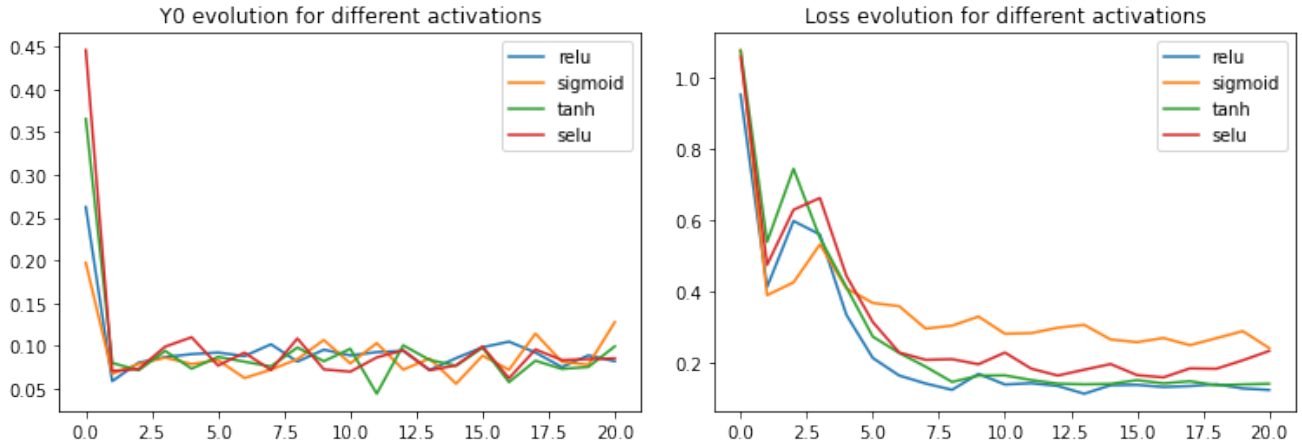
Si l'on observe comme attendu une augmentation de la CVA avec la maturité et la dimension, on retient surtout la caractéristique de non-convergence de l'estimateur énoncée par l'auteur. Etant donné la non-convexité du problème de minimisation, la convergence de la fonction coût n'implique pas une convergence de l'estimateur $\hat{Y}_0$. Cela permet de mieux appréhender la valeur ajoutée des bornes inférieures et supérieures.
Ci-dessous, on a tracé l'évolution de l'estimateur avec les bornes finalement calculées. On observe que non-seulement l'estimateur sort en-dehors des bornes à de nombreuses reprises, mais aussi que la variance de celui-ci ne semble pas diminuer.



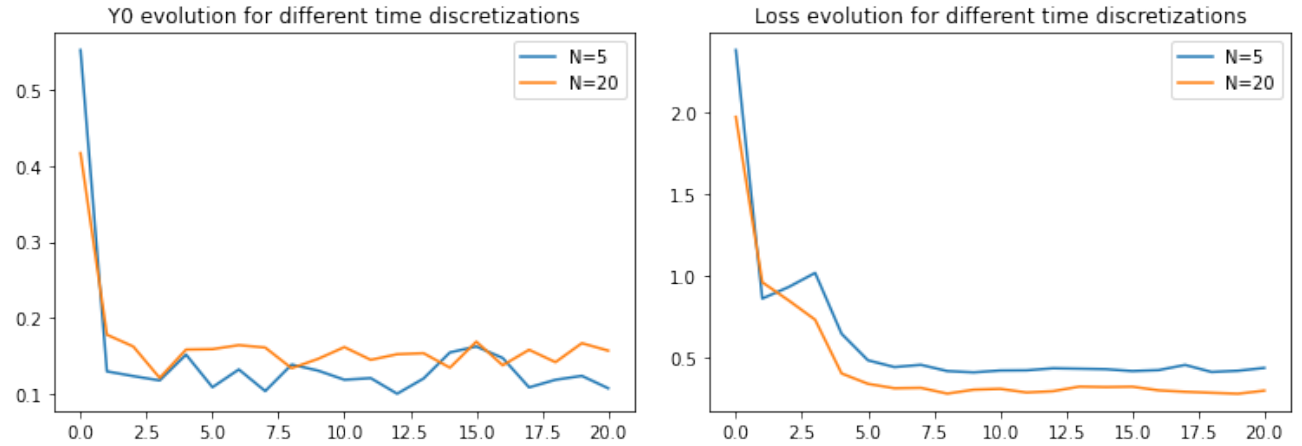
Nous nous sommes ensuite intéressés à l'architecture du réseau de neurones et plus particulièrement à son influence sur la vitesse de convergence de la fonction coût, le temps de calcul et la convergence de l'estimateur.
Ci-dessous nous avons tracé les mêmes graphiques que précédemment pour différent nombres de hidden layers à chaque temps de discrétisation :



Nous observons qu'il n'y a pas d'amélioration notable avec le nombre de hidden layers (en dépit d'une augmentation du temps de calcul, quasiment doublé entre $H = 1$ et $H = 2$ puis entre $H = 2$ et $H = 3$).
Ensuite, le même tracé pour différentes fonction d'activation des neurones confirme le choix de l'activation ReLu faite par l'auteur.



Enfin, si l'augmentation du nombre de temps de discrétisation améliore la précision de l'estimateur, avec une convergence plus rapide de la fonction coût avec le nombre d'itérations, le temps de calcul augmente fortement (temps de calcul multiplié par 5 entre $N = 5$ et $N = 20$) et il doit y avoir un compromis.



La méthode *Deep BSDE Solver* permet de surmonter le problème de la dimensionalité auquel sont confrontés les méthodes déterministes tels que les différences finies dans la résolution d'EDP. Or, lorsque la fonction coût à l'instant terminal est non-convexe avec les paramètres du réseau de neurones, la minimisation de cette dernière ne garantit pas la convergence de l'estimateur. Seule une valeur nulle permet d'inférer l'exactitude de la solution, ce qui n'est pas le cas générale. L'article propose alors, en s'appuyant sur l'équivalence du problème avec un problème de contrôle stochastique, une extension de la méthode pour le calcul de bornes inférieures et supérieures. Si, contrairement à la méthode des processus gaussiens, nous n'avons pas d'intervalle de confiance sur ces bornes, ces dernières sont rapidement calculées une fois le réseau de neurone entraîné. Cette phase d'entraînement est relativement plus coûteuse en temps que les méthodes traditionnelles de calcul CVA (Longstaff-Schwartz) mais ce temps de calcul est linéaire avec la dimension. Comme pour toutes les méthodes d'approximation numérique de la CVA vu ici, une erreur de discrétisation est induite par la discrétisation en temps. Pour cette méthode, le temps de calcul croît exponentiellement avec la finesse de cette discrétisation. L'auteur propose alors de considérer les deltas du

portefeuille, approximés par les réseaux de neurones, constant sur intervalles de temps restreints, ce qui permet de réduire la dimensionnalité du problème mais induit une nouvelle erreur de discrétisation. Enfin, on peut reprocher à l'utilisation des réseaux de neurones une perte d'interprétabilité, du fait du fonctionnement "boîte noire" de ces derniers, mais on retient leur adaptabilité à l'apprentissage de fonctions non-linéaires, telle que la partie positive $u \mapsto u^+$ présente dans le calcul de la CVA.

9 Bibliography

Références

[1] S. Crépey, "Xva : About cva, dva, fva and other market adjustments preprint of opinion and debates num. 5, june 2014," 2014.

[2] J. De Spiegeleer, D. B. Madan, S. Reyners, and W. Schoutens, "Machine learning for quantitative finance : fast derivative pricing, hedging and fitting," *Quantitative Finance*, vol. 18, no. 10, pp. 1635–1643, 2018.

[3] S. Crépey and M. Dixon, "Gaussian process regression for derivative portfolio modeling and application to cva computations," *arXiv preprint arXiv :1901.11081*, 2019.

XVA : Credit, Funding and Capital Valuation Adjustments Andrew Green

Deep learning-based numerical methods for high-dimensional parabolic partial differential equations and backward stochastic differential equations E, W., Jiequn, J., Jentzen, A