Deep Calibration of Interest Rates Model

Machine Learning for Finance

Alexandre ALOUADI

Université Paris Cité

Ensae Paris

Soutenance de projet - Mai 2024

Ce présent document est une version résumé du rapport complet que vous pouvez retrouver sur la page GitHub suivante : ici.

Table des matières

Table of Contents

Objectif de l'article

- Proposer une méthode de calibrations des cinq paramètres du modèle **G2++**

Objectif de l'article

- Proposer une méthode de calibrations des cinq paramètres du modèle **G2++**
- Méthode indirect: Covariances et Corrélation + FNN
- Méthode directe: Taux Zéro-Coupon + CNN

Principe de la Deep Calibration (DC)

Objectif: apprendre une map des données vers les paramètres à calibrer

- On dispose d'un modèle $M = M(\eta)_\eta$ avec les paramètres $\eta \in D \subset \mathbb{R}^d$ à calibrer et des prix observés sur le marché $\pi_{mkt}(\tau) = (\pi_{mkt}(\tau_1), \ldots, \pi_{mkt}(\tau_J) \in \mathbb{R}^J$

Principe de la Deep Calibration (DC)

Objectif: apprendre une map des données vers les paramètres à calibrer

- On dispose d'un modèle $M = M(\eta)_\eta$ avec les paramètres $\eta \in D \subset \mathbb{R}^d$ à calibrer et des prix observés sur le marché $\pi_{mkt}(\tau) = (\pi_{mkt}(\tau_1), \ldots, \pi_{mkt}(\tau_J) \in \mathbb{R}^J$
- On cherche η solution du problème de calibration suivant:

$$\eta^* = \arg\min_{\eta \in D} \delta\left(\pi_{\text{mod}}(\eta, \tau, e); \pi_{\text{mkt}}(\tau)\right),$$

où δ est un choix approprié de métrique, π_{mod} est le prix théorique du modèle et $e \in E$ est un facteur externe

Deep Learning pour résoudre un problème de DC

Utilisation des réseaux de neuronnes pour résoudre le problème de minimisation précédent.

- On approche la map par un réseau de neurones $\phi_\eta \in \mathrm{NN}_{J+\dim(E),d}$, qui prend en entrée un échantillion de prix et de facteurs externes, et renvoi en sortie les paramètres η
- Le réseau est entraîné en minimisant une fonction de perte, par exemple :

$$\inf_{\eta} \frac{1}{M} \sum_{m=1}^{M} \left(\phi_\theta(\pi^{(m)}, e^{(m)}) - \eta^{(m)} \right)^2,$$

via la descente de gradient stochastique.

Table of Contents

Modèle G2++

Le modèle G2++ modélise les taux d'interêts, où le taux court est donné par:

$$dr(t) = x(t) + y(t) + \phi(t)$$

avec

- $dx(t) = -\kappa_x x(t)dt + \sigma_x dW_x(t), x(0) = x_0$
- $dy(t) = -\kappa_y y(t)dt + \sigma_y dW_y(t), y(0) = y_0$

où $\kappa_x, \kappa_y, \sigma_x, \sigma_y > 0$, $d < W_x, W_y >_t = \rho dt$, et la fonction Φ déterminisme qui dépend des paramètres $\eta = (\kappa_x, \kappa_y, \sigma_x, \sigma_y, \rho)$ et du taux forward instantané en t.

Table of Contents

Création du dataset

On utilise ici la covariance et la corrélation des ZCs/FWDs dont on connaît leurs expressions analytiques, qui dépendent de η.

- Construction du dataset: génération de $N = 10000$ paramètres η dans une plage de paramètres

Création du dataset

On utilise ici la covariance et la corrélation des ZCs/FWDs dont on connaît leurs expressions analytiques, qui dépendent de η.

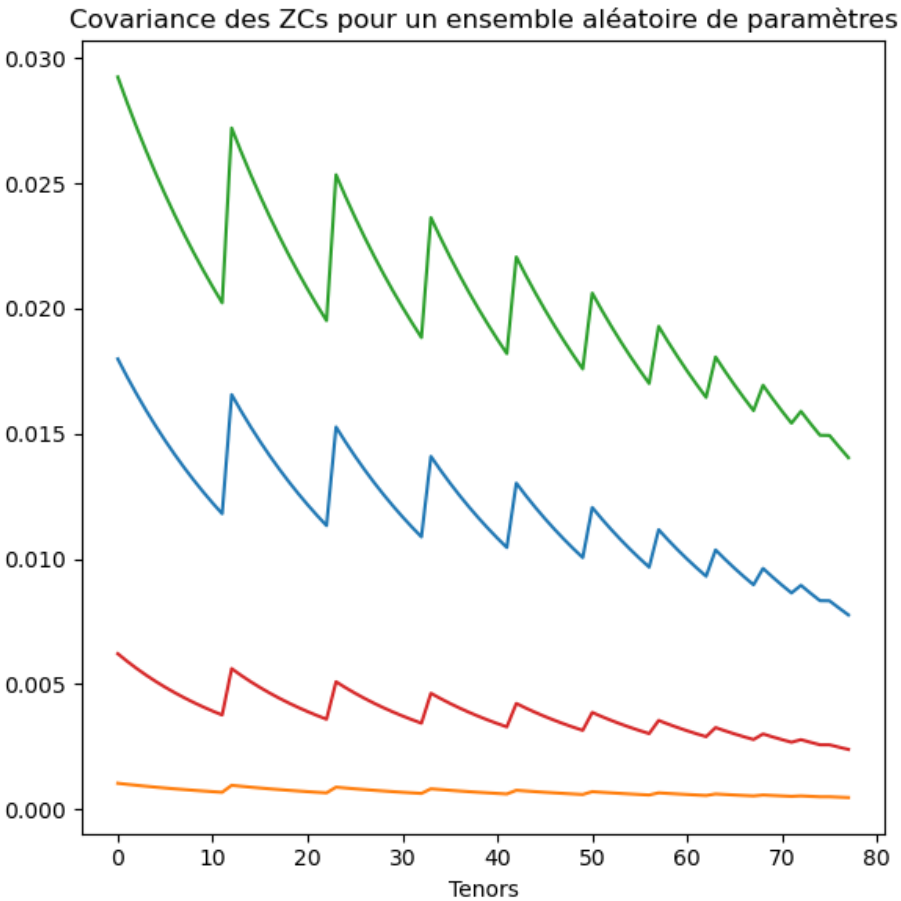
- Construction du dataset: génération de $N = 10000$ paramètres η dans une plage de paramètres
- Calcul des covariances et corrélation pour chaque η générés

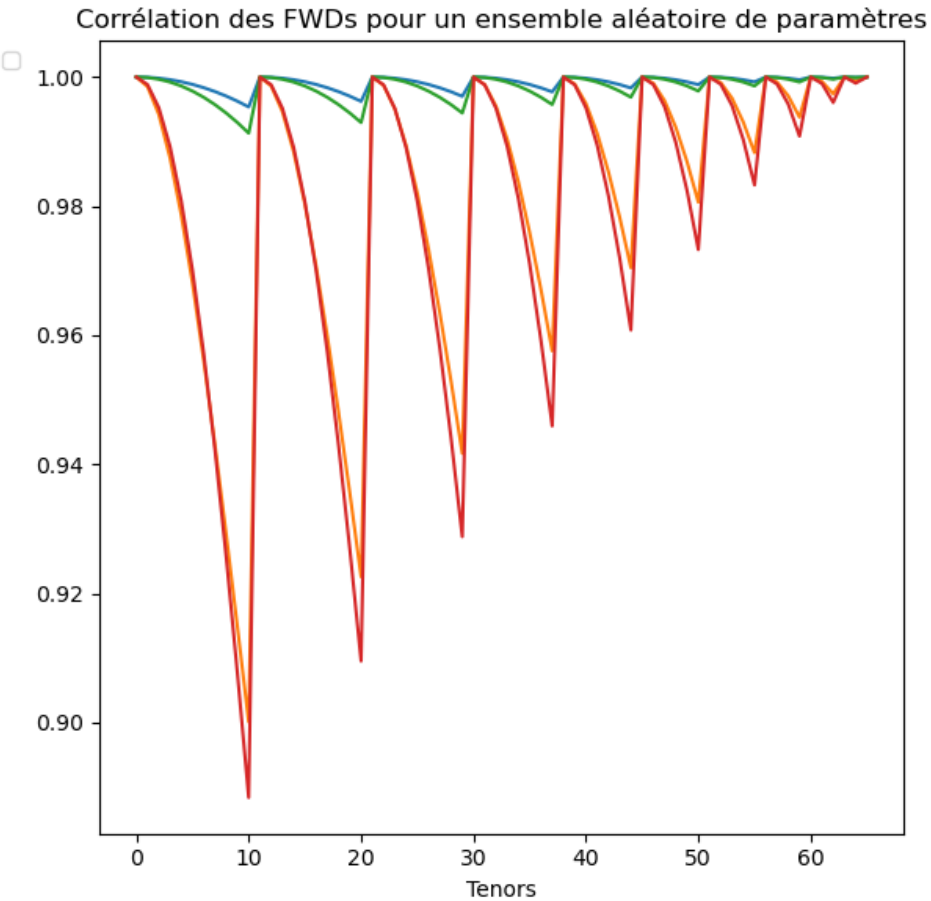
Création du dataset

On utilise ici la covariance et la corrélation des ZCs/FWDs dont on connaît leurs expressions analytiques, qui dépendent de η.

- Construction du dataset: génération de $N = 10000$ paramètres η dans une plage de paramètres
- Calcul des covariances et corrélation pour chaque η générés
- Division du dataset en Train/Test Set puis transformation min-max

Covariances et Corrélations des ZCs et FWDs pour des paramètres aléatoires





Entraînement du modèle

On utilise ici un FNN, qui a l'architecture suivante:

- Une couche d'entrée avec $nf = 78$ ou 66 features
- 3 *hidden layers* avec respectivement $1000, 1500$ et 1000 neuronnes, et la fonction d'activation ReLU
- On ajoute un *dropout* avec probabilité 0.25 pour éviter l'*overfitting*
- Une couche de sortie, sans activation, qui est la prédiction de nos paramètres à calibrer

Implémentation: optimiser **Adam** avec un taux d'apprentissage à 0.001, la perte quadratique **MSE**, 100 epochs et des mini-batches de taille 1000.

Résultat pour la Covariance: κ_x et κ_y

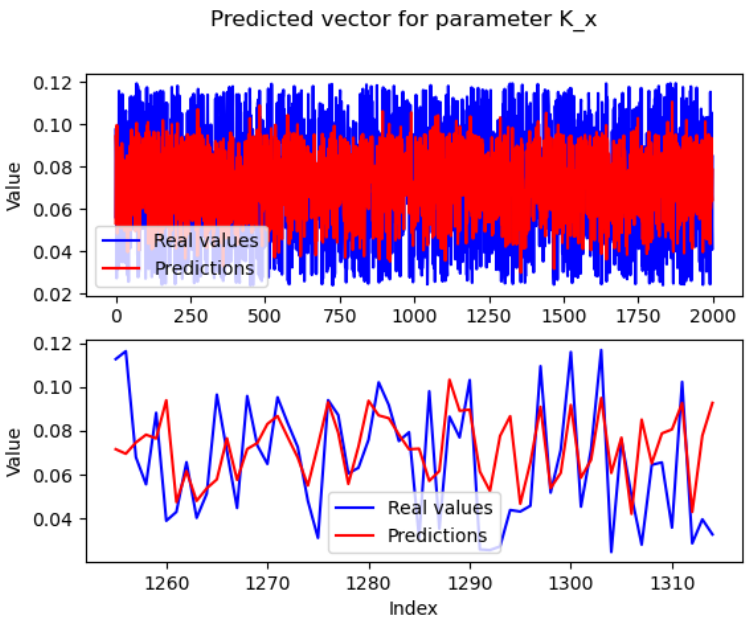


Figure: Calibration de κ_x

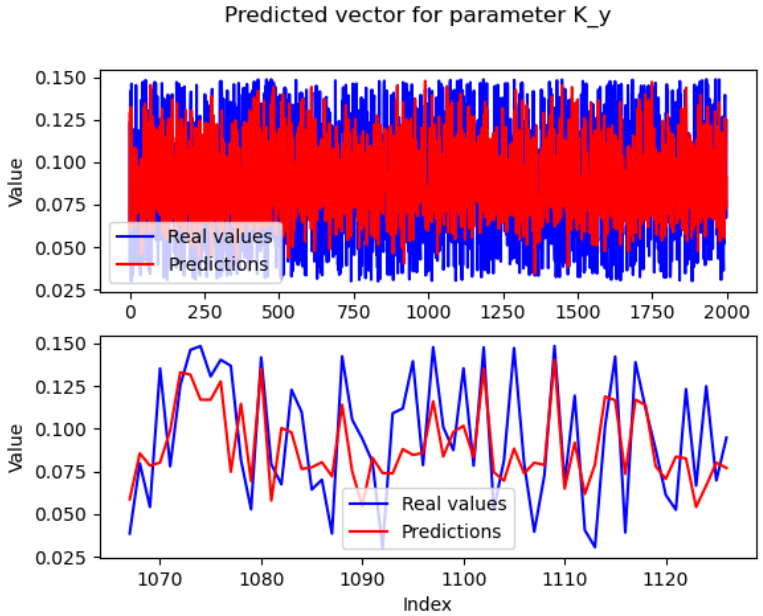


Figure: Calibration de κ_y

Résultat pour la Corrélation: σ_x et σ_y

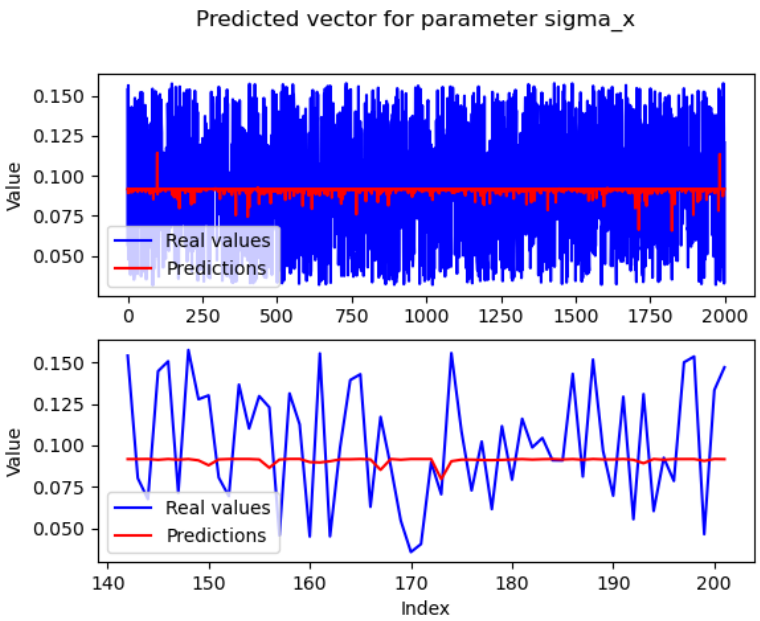


Figure: Calibration de σ_x

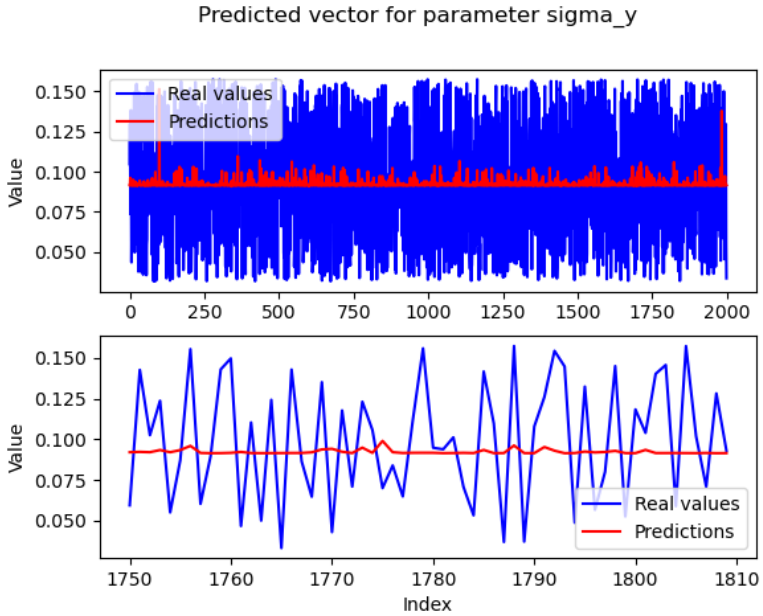


Figure: Calibration de σ_y

Table of Contents

Création du dataset

La méthode pour créer le dataset est la même que précédemment, mais où on utilise les courbes ZCs plutôt que la cov/corr. Dans le modèle G2++, on al'expression du taux ZC $Z(.,.)$ suivante:

$$Z(t,T) = -\frac{1}{T-t} ln(\frac{P^M(0,T)}{P^M(0,t)}) - \frac{1}{T-t} A(t,T)$$

Or, puisque $A(t,T)$ comporte une partie stochastique dans $x(t)$ et $y(t)$, nous prennons l'espérance des taux ZC calculés. En fixant $x_0 = y_0 = 0$, on obtient :

$$\mathbb{E}(Z(t,T)) = -\frac{1}{T} ln(\frac{P^M(0,T)}{P^M(0,t)}) - \frac{1}{2T}(V(t,T) + V(0,t) - V(0,T))$$

Exemple de courbe obtenue

On obtient alors, avec comme données de marché la courbe Euro telle qu'elle était le 27 février 2024 :

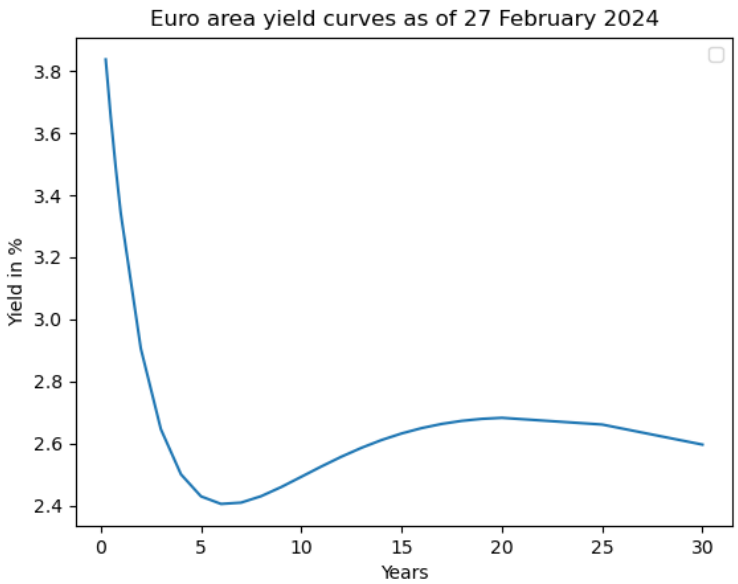


Figure: Données de marché initial

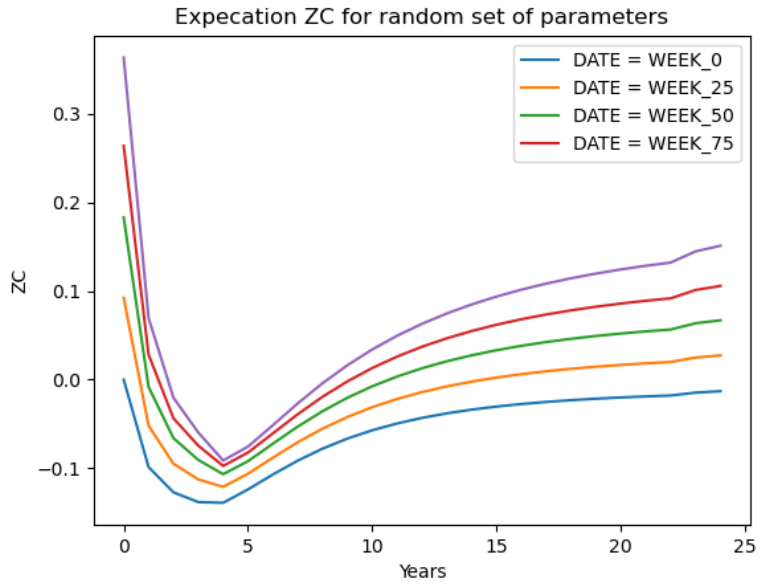


Figure: Exemple d'une courbe de ZC avec des paramètres aléatoire

Entraînement du modèle

On utilise ici un CNN, qui a l'architecture suivante:

- Une couche d'entrée avec $nf = nb_{tenors} = 25$ features
- Le CNN, avec un filtre et un noyeau de taille 7, une fonction d'activation ReLU et avec du *padding*
- Un pooling, ici le MaxPooling, de stride 2
- Une couche intermédiaire linéaire avec 100 neuronnes et une fonction d'activation ReLU, avec *dropout* à 0.25
- Une couche de sortie à 5 neuronnes qui contiendra les 5 paramètres.

Implémentation: optimiser **Adam** avec un taux d'apprentissage à 0.0002, la perte quadratique **MSE**, 200 epochs et des mini-batches de taille 50.

Résultat pour la calibration par CNN: κ_x et ρ

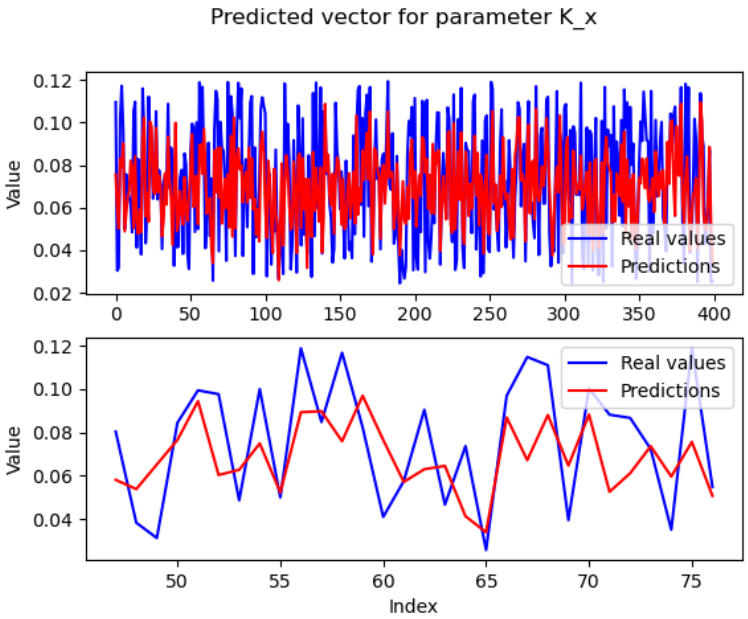


Figure: Calibration de κ_x

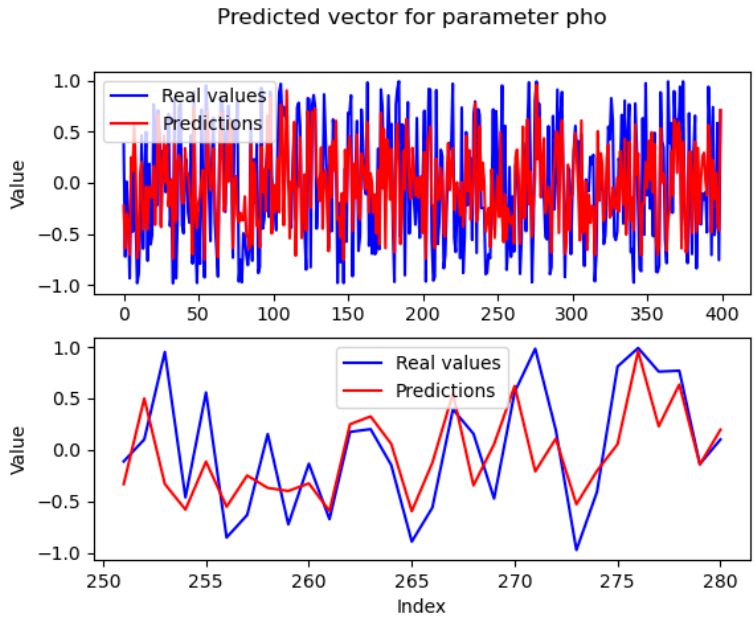


Figure: Calibration de ρ

Résultat fonction de perte des 2 modèles

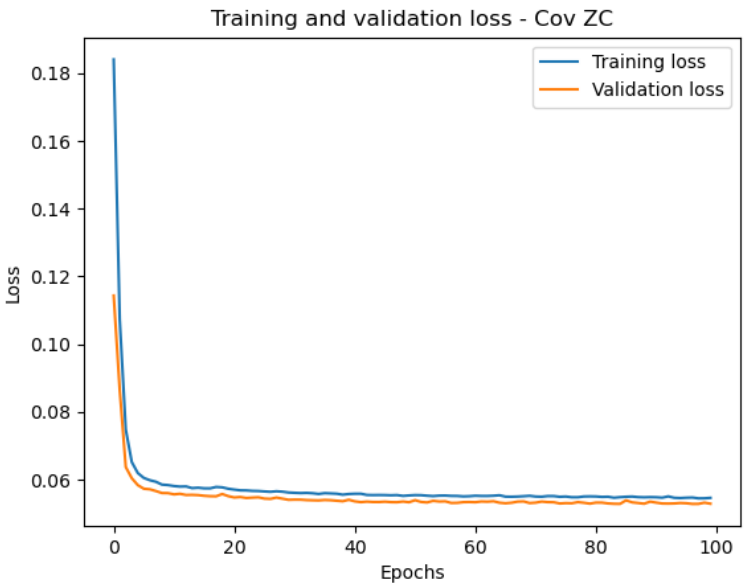


Figure: Fonction perte - Cov ZCs

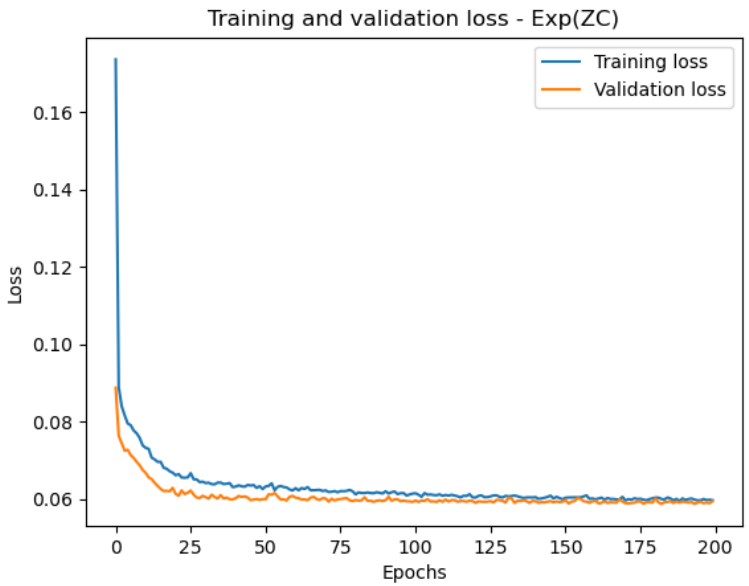


Figure: Fonction perte - CNN

Table of Contents

Conclusion sur les résultats obtenus

L'utilisation des covariances enrichit l'information des réseaux de neurones, tandis que la décomposition en composantes principales basée sur les corrélations engendre environ le double d'erreurs. Des ajustements comme la sélection des tenors et la révision de l'architecture des réseaux de neurones sont nécessaires pour optimiser les performances computationnelles et la précision.

A noter que nous obtenons une MSE correcte, de l'ordre de 0.05.