

# Projet de Machine learning pour la finance : Deep Calibration of Interest Rates Model

Alexandre Alouadi

Mars 2024

# 1 Introduction

Pour toute institution financière, il est impératif de pouvoir appréhender le comportement des taux d'intérêt. Malgré l'utilisation croissante du *deep learning*, pour diverses raisons telles que l'expertise et la facilité d'utilisation, les modèles classiques de taux tels que CIR ou la famille gaussienne sont encore largement utilisés.

L'objectif de l'article "*DEEP CALIBRATION OF INTEREST RATES MODEL*" (pour consulter l'article, cliquez [ici](#)) est de proposer une méthode de calibrations des cinq paramètres du modèle **G2++** en utilisant des réseaux neuronaux. Pour ce faire, nous allons construire notre propre dataset synthétiques de paramètres tirés uniformément à partir d'un ensemble de référence de paramètres calibrés sur le marché. À partir de ces paramètres, nous calculerons les taux zéro-coupon et les taux forward ainsi que leurs covariances et corrélations. Pour ce faire, nous commencerons par utiliser un réseau de neuronne entièrement connecté (FNN) qui utilisera uniquement les covariances et les corrélations. Nous montrons que les covariances sont mieux adaptées au problème que les corrélations. Nous utiliserons ensuite un deuxième modèle de réseau de neurones, convolutif (CNN), n'utilisant que les taux zéro-coupon sans transformation.

## 1.1 Principe de la Deep Calibration

La calibration de modèles est une tâche centrale effectuée quotidiennement par les institutions financières, impliquant la sélection de modèles pour expliquer un ensemble donné de prix du marché. Ce problème inverse consiste à apprendre une **map** entre les données et les paramètres du modèle. Traditionnellement, ce choix est basé sur la facilité de calcul plutôt que sur la capacité à saisir les caractéristiques du marché. Cependant, l'intégration croissante du *machine learning* (ML) cherche à changer cette approche.

Pour ce faire, nous devons disposer d'un modèle financier  $M$  que nous voulons calibrer, tel que  $M = M(\eta)$  avec les paramètres  $\eta \in D \subset \mathbb{R}^d$ , ainsi que les prix de marché  $\pi_{mkt}(\tau) = (\pi_{mkt}(\tau_1), \dots, \pi_{mkt}(\tau_J)) \in \mathbb{R}^J$  avec  $\tau = (\tau_1, \dots, \tau_J)$  qui peuvent être des maturités, des strikes, etc. De plus, nous disposons également d'une map qui à chaque combinaison de paramètres  $(\eta, \tau, e)$  associe une valeur théorique de prix de modèle, *i.e* :

$$(\eta, \tau, e) \mapsto \pi_{\text{mod}}(\eta, \tau, e) = (\pi_{\text{mod}}(\eta, \tau_1, e), \dots, \pi_{\text{mod}}(\eta, \tau_J, e)) \in \mathbb{R}^J$$

où  $\pi_{\text{mod}}$  est le prix théorique du modèle et  $e \in E$  est un facteur externe, tel qu'une courbe de taux, sur lequel dépend le prix.

Le but est alors de trouver les paramètres  $\eta$  solution du problème de calibration suivant :

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

où  $\delta$  est un choix approprié de métrique, par exemple,  $\delta(\pi, \tilde{\pi}) = \sum_{j=1}^J w_j (\pi_j - \tilde{\pi}_j)^2$  pour certains poids  $w_j$ .

Cela définit une map de calibration  $h : \mathbb{R}^J \times E \rightarrow D$  à partir des (prix du marché, facteurs externes)  $\mapsto$  paramètres du modèle :

$$\eta^* = h(\pi_{mkt}(\tau), e)$$

Nous pouvons utiliser les réseaux de neurones pour réaliser cette tâche. On approche alors  $h$  par  $\phi_\eta(\cdot)$  où  $\phi_\eta$  est un réseau de neurones avec des paramètres  $\eta$ , une dimension d'entrée  $J + \dim(E)$ , et une dimension de sortie  $d$ , noté  $\phi_\eta \in \text{NN}_{J+\dim(E),d}$ . Le problème de calibration est résolu en apprenant les paramètres  $\eta$ . Pour ce faire, on commence par créer des données d'entrée et de sortie de taille  $M$  pour la map à apprendre. Ici :

- **Entrée** : un échantillon de prix  $(\pi^{(1)}, \dots, \pi^{(M)})$  et de facteurs externes  $(e^{(1)}, \dots, e^{(M)})$ , où chaque  $\pi^{(m)} \in \mathbb{R}^J$ , et  $e^{(m)} \in E$ .
- **Sortie** : un échantillon de paramètres  $(\eta^{(1)}, \dots, \eta^{(M)})$ , où chaque  $\eta^{(m)} \in \mathbb{R}^d$ .

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.

## 2 Calibration du modèle G2++

Il est nécessaire pour les acteurs financiers de pouvoir comprendre, prévoir et analyser les moteurs de valorisation des actif. Parmi ces moteurs se trouvent les taux d'intérêt. Les modèles de taux d'intérêt sont alors largement utilisés pour expliquer le comportement de ces derniers. Il existe de nombreux modèles, comme le modèle de Cox Ingersoll Ross (CIR) qui est assez simple à utiliser et à calibrer, mais est unifactoriel, ce qui limite son utilisation. Dans le modèle gaussien G2++, le processus de taux court est donné par la somme des deux facteurs distribués par la loi normale et par une fonction déterministe.

### 2.1 Modèle G2++

On suppose ici que sous la mesure risque-neutre  $Q$ , le taux court s'écrit :

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

où les deux facteurs  $x(t)$  et  $y(t)$  sont deux processus stochastique tel que :

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

avec  $\kappa_x, \kappa_y, \sigma_x, \sigma_y > 0$ ,  $d < W_x, W_y >_t = \rho dt$ , et la fonction  $\Phi$ , déterminisme, permettant de calibrer le modèle gaussien à deux facteurs sur la courbe  $PM(0, T)$ ,  $T > 0$ , de prix d'obligation observé sur le marché. On a alors grâce aux équations des prix forward en  $t$  d'une obligation zéro coupon d'échéance  $T$ , noté  $P(t, T)$  :

$$\phi(t) = f(0, t) + \frac{\sigma_x^2}{2\kappa_x^2} \cdot (1 - e^{-2\kappa_x t})^2 + \frac{\sigma_y^2}{2\kappa_y^2} \cdot (1 - e^{-2\kappa_y t})^2 + \frac{\rho\sigma_x\sigma_y}{\kappa_x\kappa_y} \cdot (1 - e^{-\kappa_x t}) \cdot (1 - e^{-\kappa_y t})$$

avec  $f(0, t)$  le taux forward instantané en  $t$ . Les paramètres du modèle G2++ à calibrer sont alors  $\eta = (\kappa_x, \kappa_y, \sigma_x, \sigma_y, \rho)$ .

## 2.2 Calibration par FNN

Pour cette première approche, nous remarquons que les taux zéro-coupon (ZCs) notés  $Z(.,.)$  et les taux forward (FWDs) notés  $f(.,.)$  sont facilement observables sur le marché, et des expressions analytiques peuvent être déduites de la plupart, voire de tous les modèles de taux d'intérêt. Cependant, pour maintenir une architecture simple du FNN, nous choisissons d'utiliser des quantités intermédiaires, comme les matrices de corrélation et de covariance des variations des ZCs et des FWDs, pour permettre une calibration dynamique efficace. Les expressions des corrélations et des covariances des ZCs et des FWDs sont données par :

$$\text{cov}(dG(T_i), dG(T_j)) = [X(T_i)X(T_j) + Y(T_i)Y(T_j) + \rho(X(T_i)Y(T_j) + X(T_j)Y(T_i))] \quad (1)$$

$$\text{cor}(dG(T_i), dG(T_j)) = \frac{\text{cov}(dG(T_i), dG(T_j))}{\sqrt{(X^2(T_i) + Y^2(T_i) + 2\rho X(T_i)Y(T_i))(X^2(T_j) + Y^2(T_j) + 2\rho X(T_j)Y(T_j))}} \quad (2)$$

où

$$X(T) = \begin{cases} \sigma_x \frac{1-e^{-\kappa_x T}}{\kappa_x T} & \text{si } G(T) = Z(.,T) \\ \sigma_x e^{-\kappa_x T} & \text{si } G(T) = f(.,T) \end{cases}$$

De plus,  $Y(T)$  est obtenue par symétrie avec  $X$ .

### Construction du dataset

Afin de pouvoir entraîner notre modèle, nous avons besoin de générer un dataset, c'est à dire de disposer d'un grand nombre de paramètres  $\eta$ , que nous utiliserons pour calculer les covariances des ZCs et des FWDs. Pour ce faire, nous commençons par une calibration classique de minimisation de l'erreur à partir des taux zéro-coupon Euro du marché, couvrant la période de juin 2019 à novembre 2020. Les paramètres obtenus sont les suivants :

$\kappa_x$	$\kappa_y$	$\sigma_x$	$\sigma_y$	$\rho$
0.07173132	0.08930784	0.09465584	0.094675523	-0.999318

TABLE 1 – Paramètres de référence

Nous allons ensuite générer  $N = 10000$  quintuplets de paramètres  $\eta$ , que nous tirerons uniformément dans les intervalles suivants :

$\kappa_x$	$\kappa_y$	$\sigma_x$	$\sigma_y$	$\rho$
MIN	0.02391044	0.02976928	0.03155195	0.03155851
MAX	0.1195522	0.1488464	0.15775973	0.15779254

TABLE 2 – Plage de paramètres

Nous obtenons alors les distributions suivantes :

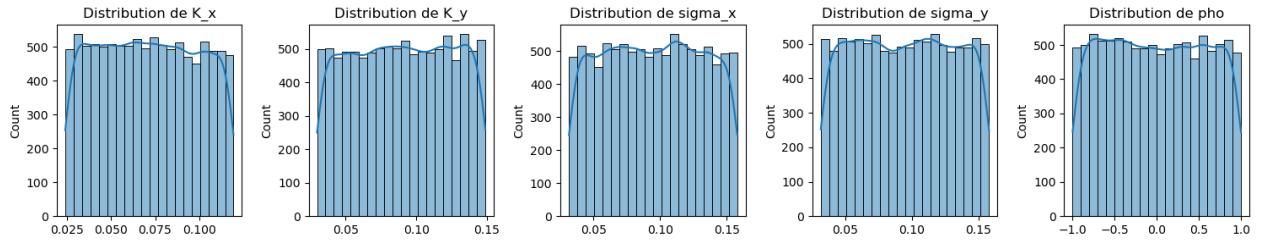


FIGURE 1 – Distributions des paramètres pour  $N = 10000$  tirages

Nous pouvons alors calculer les covariances et les corrélations pour les ZCs et les FWDs selon les formules (1) et (2). Pour cela, nous avons besoin de définir au préalable plusieurs maturités. Comme nous souhaitons limiter le nombre de caractéristiques, nous décidons de ne pas prendre d'échéances à long terme, au-delà de 12 ans. Nous prenons enfin des échéances de 1 à 12 ans avec un pas d'un an. On a donc  $nf = 78$  features pour la covariance, et  $nf = 66$  pour la corrélation. La figure 2 ci-dessous montre la covariance des ZCs et la corrélation des FWDs pour 4 sets de paramètres aléatoires(voir figures [23](#), [24](#) et pour les résultats de l'article) :

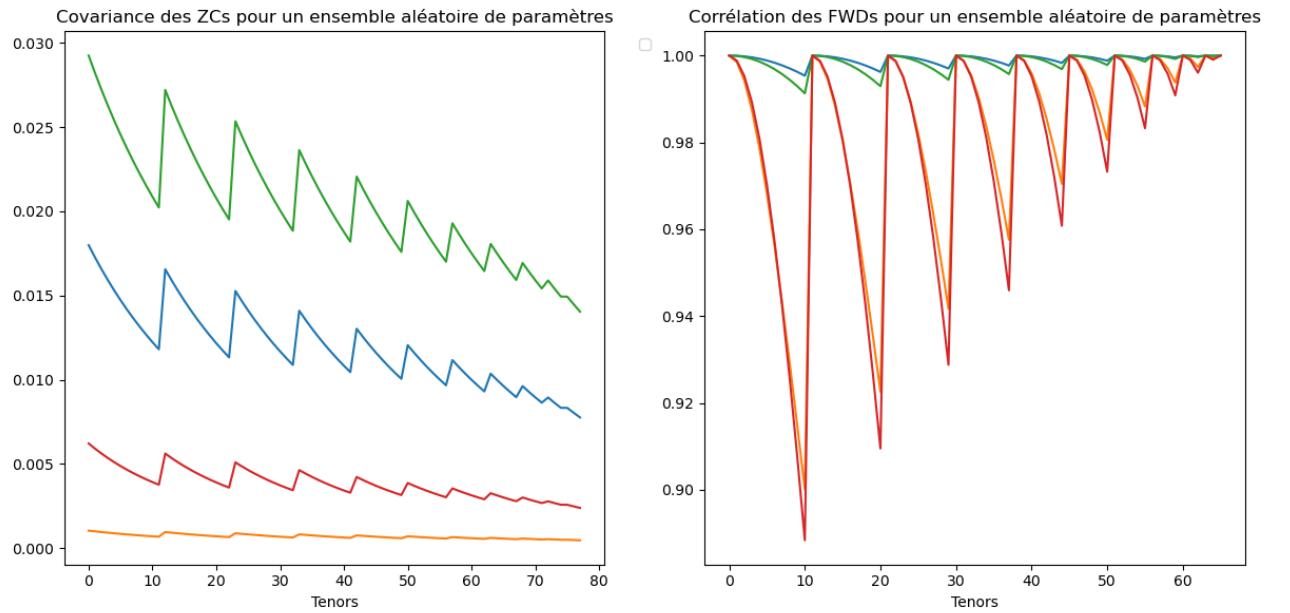


FIGURE 2 – Covariances et Corrélations des ZCs et FWDs pour des paramètres aléatoires

Pour conclure cette étape, nous divisons alors le dataset en un ensemble *Train Set* de taille 8000, et un ensemble *Test Set* de taille 2000, puis nous leur appliquons une transformation **min-max**.

## Entraînement du modèle

Le modèle qui sera utilisé sur notre Train Set est un FNN. L'architecture de notre NN que nous mettons en place est la suivante :

- Une couche d'entrée avec  $nf$  features

- 3 *hidden layers* avec respectivement 1000, 1500 et 1000 neurones, 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 prédition de nos paramètres à calibrer

Dans l'implémentation, on utilise l'optimiser **Adam** avec un taux d'apprentissage à 0.001, la perte quadratique **MSE**, 100 epochs et des mini-batches de taille 1000.

Les figures suivantes montrent les résultats dans le cas de la **covariance** obtenus pour chaque paramètre, avec un zoom effectué à un endroit aléatoire, ainsi que l'évolution de la fonction de perte du Train et du Test Set (voir figure 25 pour les résultats de l'article) :

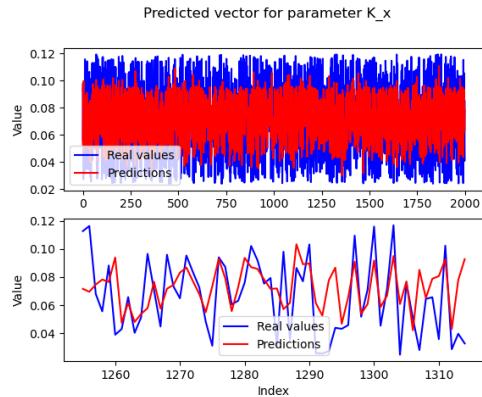


FIGURE 3 – Calibration de  $\kappa_x$

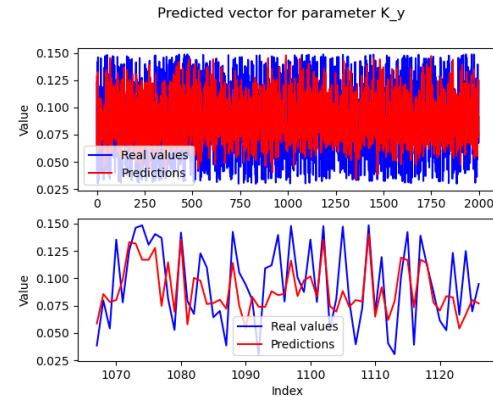


FIGURE 4 – Calibration de  $\kappa_y$

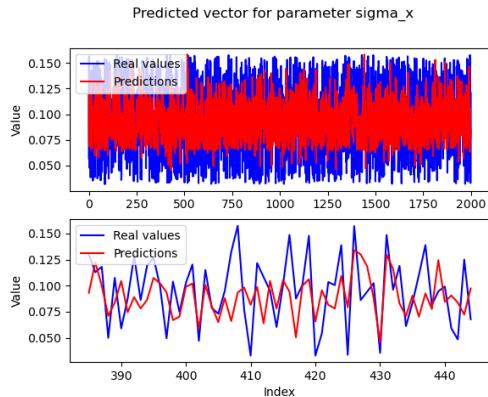


FIGURE 5 – Calibration de  $\sigma_x$

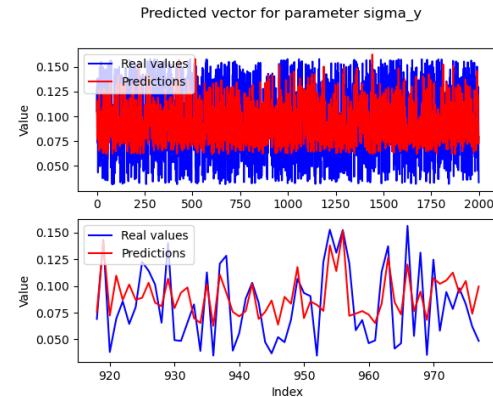


FIGURE 6 – Calibration de  $\sigma_y$

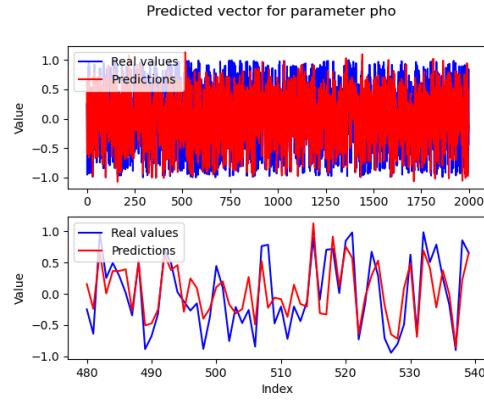


FIGURE 7 – Calibration de  $\rho$

On montre de même les résultats obtenus dans le cas de la **corrélation** (voir figure 26 pour les résultats de l'article) :

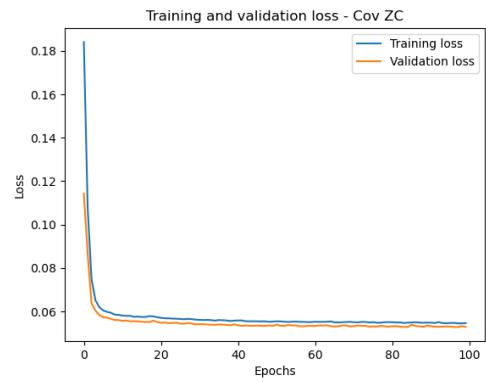


FIGURE 8 – Evolution de la fonction perte

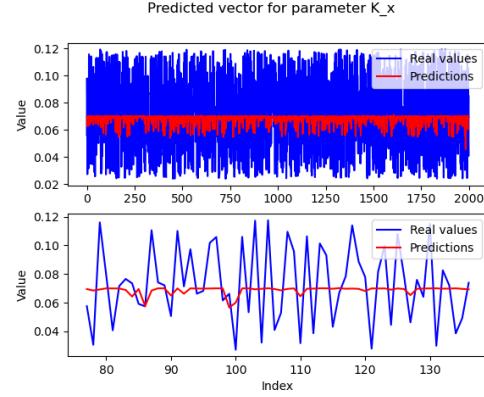


FIGURE 9 – Calibration de  $\kappa_x$

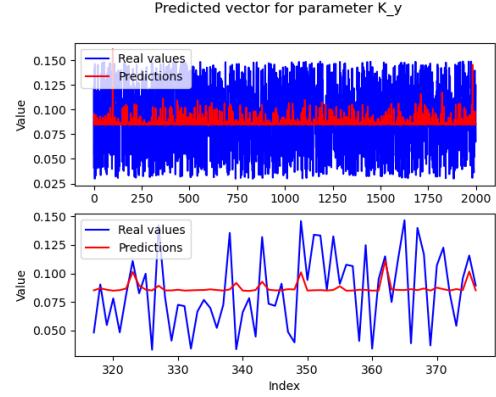


FIGURE 10 – Calibration de  $\kappa_y$

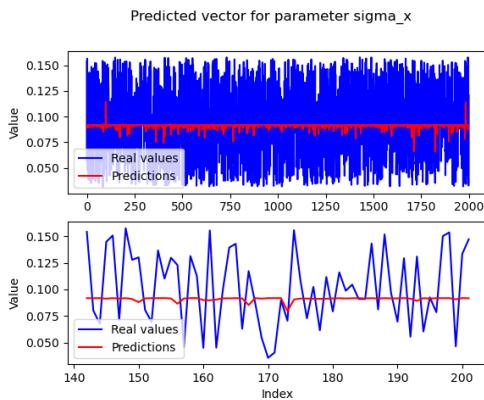


FIGURE 11 – Calibration de  $\sigma_x$

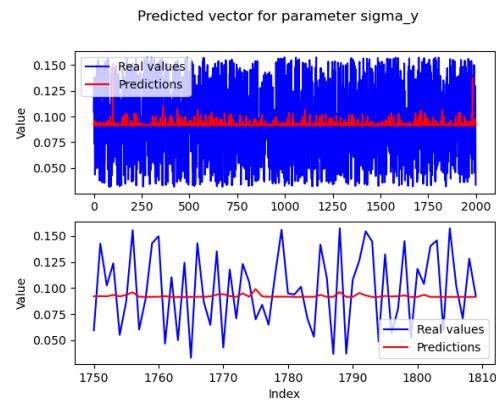


FIGURE 12 – Calibration de  $\sigma_y$

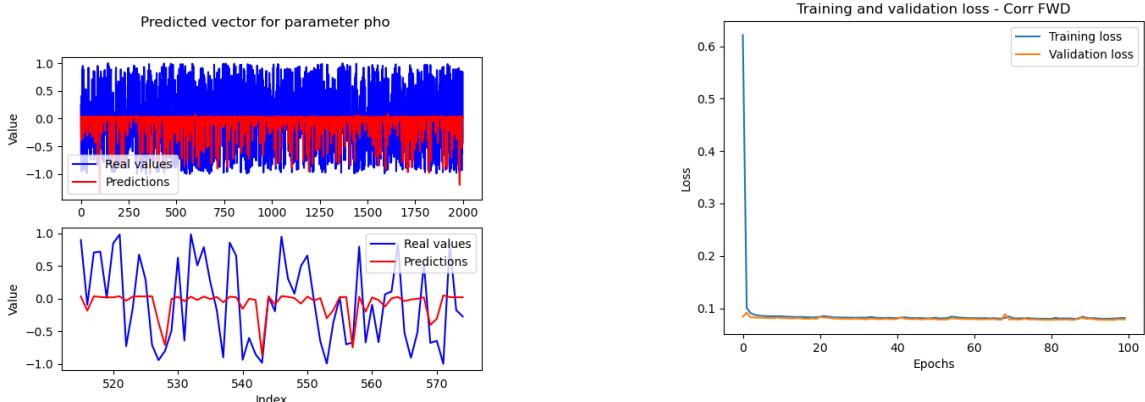


FIGURE 13 – Calibration de  $\rho$

FIGURE 14 – Evolution de la fonction perte

On observe que les covariances fournissent plus d’informations aux NN. La DC indirecte utilisant les corrélations a entraîné environ le double d’erreurs qu’en utilisant les covariances. Il pourrait être alors utile d’inclure davantage de maturités et de combiner le modèle avec un autre pour la réduction de dimension. On note tout de même que les *test loss* sont correctes, 0.052 pour dans le cas de la covariance et 0.078 pour la corrélation.

### 2.3 Calibration par CNN

Nous voulons maintenant envisager la calibration directe en utilisant les courbes ZCs, qui sont des observations directes du marché contrairement aux corrélations et covariances.

#### Construction du dataset

La méthode de construction du dataset est la même que dans le cas précédent, à l’exception que nous devons calculer les taux zéros coupons.

Pour ce faire, on utilise que dans le modèle G2++, le prix de l’obligation ZC au temps  $t$  de maturité  $T$  est :

$$P(t, T) = \frac{P^M(0, T)}{P^M(0, t)} e^{A(t, T)}$$

avec :

- $A(t, T) = \frac{1}{2}(V(t, T) - V(0, T) + V(0, t)) - \frac{1-e^{-\kappa_x(T-t)}}{\kappa_x}x(t) - \frac{1-e^{-\kappa_y(T-t)}}{\kappa_y}y(t)$
- $V(t, T) = \frac{\sigma_x^2}{\kappa_x^2}(T-t + \frac{2}{\kappa_x}e^{-\kappa_x(T-t)} - \frac{1}{2\kappa_x}e^{-2\kappa_x(T-t)} - \frac{3}{2\kappa_x}) + \frac{\sigma_y^2}{\kappa_y^2}(T-t + \frac{2}{\kappa_y}e^{-\kappa_y(T-t)} - \frac{1}{2\kappa_y}e^{-2\kappa_y(T-t)} - \frac{3}{2\kappa_y}) + \frac{2\rho\sigma_x\sigma_y}{\kappa_x\kappa_y}(T-t + \frac{e^{-\kappa_x(T-t)}-1}{\kappa_x} + \frac{e^{-\kappa_y(T-t)}-1}{\kappa_y} - \frac{e^{-(\kappa_x+\kappa_y)(T-t)}-1}{\kappa_x+\kappa_y})$
- $P^M(0, T)$  la courbe de taux ZC initiale sur le marché de maturité  $T$

L’expression du taux ZC  $Z(., .)$  est donc donnée par :

$$Z(t, T) = -\frac{1}{T-t} \ln\left(\frac{P^M(0, T)}{P^M(0, t)}\right) - \frac{1}{T-t} A(t, T) \quad (3)$$

Or, l’expression de  $A(t, T)$  comporte une partie stochastique dans  $x(t)$  et  $y(t)$ , ce qui signifie qu’il n’y aura pas qu’une seule valeur pour  $Z(t, T)$ . Pour contourner ce problème, nous prenons l’espérance des taux ZC calculés. En fixant  $x_0 = y_0 = 0$ , on obtient :

$$\mathbb{E}(Z(t, T)) = -\frac{1}{T} \ln\left(\frac{P^M(0, T)}{P^M(0, t)}\right) - \frac{1}{2T}(V(t, T) + V(0, t) - V(0, T)) \quad (4)$$

Pour la courbe  $P^M(0, t)$ , nous prenons la courbe Euro telle qu'elle était le 27 février 2024. On utilise les maturités suivantes :  $[3M, 6M, 9M, 1Y, 2Y, \dots, 19Y, 20Y, 25Y, 30Y]$ , et pour la propagation des taux, on prend des pas de 1 semaine et nous faisons 106 pas (environ 2 ans). Nos données d'entrées seront donc dans  $\mathbb{R}^{nb_{step} \times nb_{tenors}}$ , avec  $nb_{step} = 106$  et  $nb_{tenors} = 25$ .

On obtient alors (voir figures 27, 28 pour les résultats de l'article) :

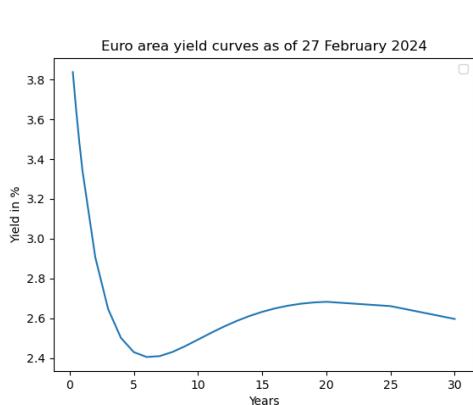


FIGURE 15 – Données de marché initial

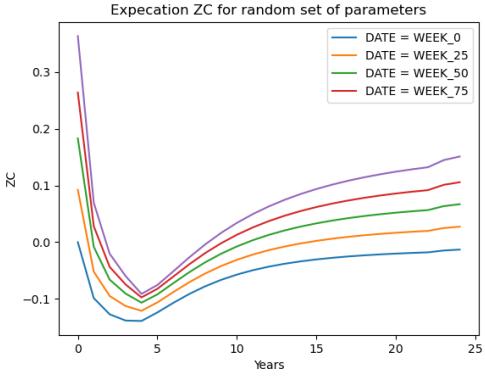


FIGURE 16 – Exemple d'une courbe de ZC avec des paramètres aléatoire

## Entraînement du modèle

Le modèle qui sera utilisé sur notre Train Set est un CNN. Ici, on prend un Train Set de taille 2000, et un Test Set de taille 400 afin de réduire le temps d'exécution de l'entraînement. L'architecture de notre NN que nous mettons en place est la suivante :

- Une couche d'entrée avec  $nf = nb_{tenors} = 25$  features
- Le CNN, avec un filtre et un noyau 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 neurones et une fonction d'activation ReLU, avec *dropout* à 0.25
- Une couche de sortie à 5 neurones qui contiendra les 5 paramètres.

Comme précédemment, nous utilisons l'optimiseur Adam, appliqué à l'erreur quadratique moyenne et de taux d'apprentissage de 0.0002. On fixe le nombre d'epochs à 200 et des mini-batches de taille 50 afin de simplifier la phase d'entraînement.

Les figures suivantes montrent les résultats obtenus pour chaque paramètres, avec un zoom fait à un endroit aléatoire, ainsi que l'évolution de la fonction perte du Train et du Test Set (voir figure 29 pour les résultats de l'article) :

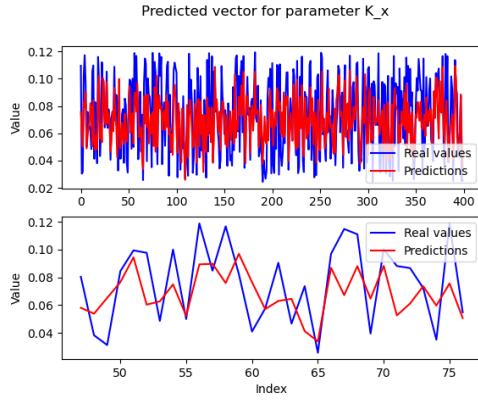


FIGURE 17 – Calibration de  $\kappa_x$

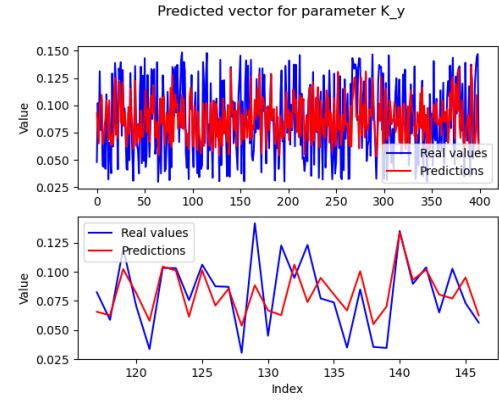


FIGURE 18 – Calibration de  $\kappa_y$

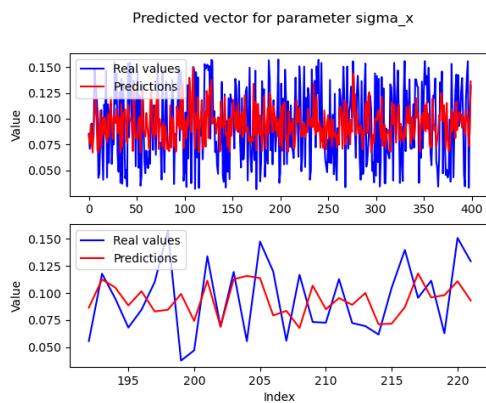


FIGURE 19 – Calibration de  $\sigma_x$

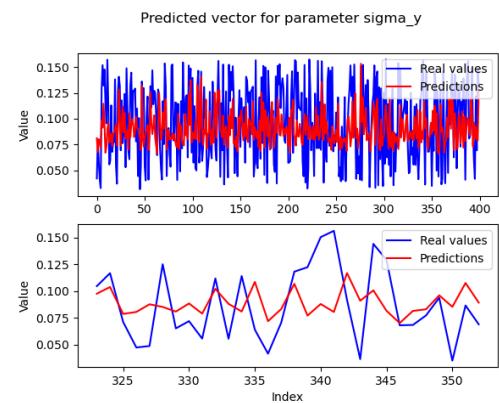


FIGURE 20 – Calibration de  $\sigma_y$

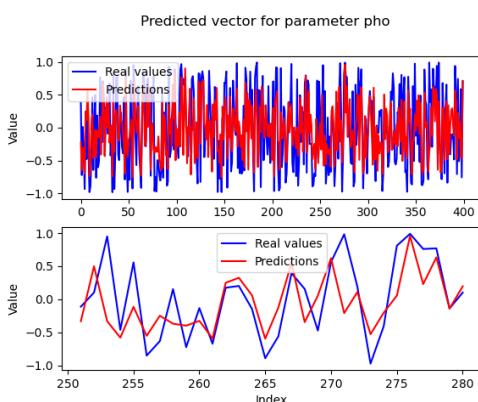


FIGURE 21 – Calibration de  $\rho$

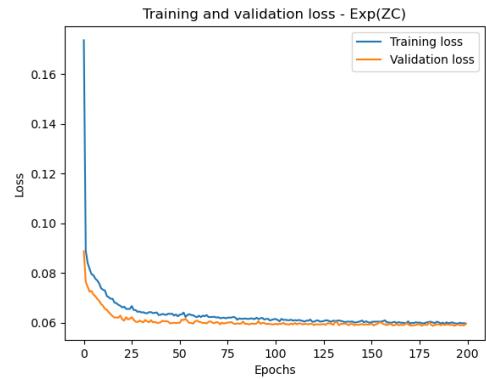


FIGURE 22 – Evolution de la fonction perte

La méthode directe de DC conduit à des calibrations légèrement moins précises par rapport à la méthode indirecte avec les covariances des taux ZCs. Bien que les différences ne soient pas importantes, elles sont justifiées car la méthode indirecte bénéficie d'informations déjà extraites de la courbe des taux ZCs. Améliorer cette méthode serait très utile, notamment en sélectionnant judicieusement les tenors pour obtenir de meilleurs résultats et des performances

computationnelles optimisées, et en revisitant l'architecture du CNN pour prendre en compte la complexité des données d'entrée et améliorer la précision. La encore, la perte du test set obtenue est bonne : 0.059.

Ci-dessous, un résumé de l'erreur quadratique moyenne obtenue sur le test set pour chaque paramètre (voir tableau 4 pour les résultats de l'article) :

MSE	$K_x$	$K_y$	$\sigma_x$	$\sigma_y$	$\rho$
COV - ZC	$4.8 \times 10^{-4}$	$7.2 \times 10^{-4}$	$9.7 \times 10^{-4}$	$9.8 \times 10^{-4}$	$1.5 \times 10^{-1}$
CORR - FWD	$7.4 \times 10^{-4}$	$1.1 \times 10^{-3}$	$1.3 \times 10^{-3}$	$1.3 \times 10^{-3}$	$2.7 \times 10^{-1}$
ZC	$7.4 \times 10^{-4}$	$1.1 \times 10^{-3}$	$1.3 \times 10^{-3}$	$1.3 \times 10^{-3}$	$2.7 \times 10^{-1}$

TABLE 3 – Erreurs de calibration sur le test set

### 3 Conclusion

L'article "*DEEP CALIBRATION OF INTEREST RATES MODEL*" a montré que l'on peut utiliser les réseaux de neurones pour calibrer un modèle, ici le modèle de taux G2++.

Pour cela, nous avons dans un premier temps créer notre dataset sur la base d'échantillons tirés aléatoirement autour de nos 5 paramètres de références. Nous avons ensuite utilisés 2 méthodes. La première, pour le calibrage indirect qui consiste en des corrélations et des covariances de ZCs FWDS, et la seconde pour le calibrage direct qui utilise des courbes de taux ZCs brutes.

Nous avons constaté que les covariances fournissaient plus d'informations aux NN, ce qui se traduisait par une meilleure précision pour le calibrage indirect par rapport aux corrélations. Le calibrage direct avec les taux ZCs a également montré une bonne précision. Nous avons également identifié des pistes d'amélioration, notamment l'ajout de plus de maturités pour le premier modèle et l'amélioration de l'architecture et de l'optimisation des hyperparamètres pour le deuxième modèle. Les performances obtenues permettent de conclure que l'utilisation du DL permet un calibrage très rapide avec de faibles erreurs pour le modèle G2++, et par extension pour tout autre modèle répondant à nos exigences pour les données d'observation du marché et possédant une expression analytique.

## 4 Annexes

Nous présentons ici les résultats de l'article "*DEEP CALIBRATION OF INTEREST RATES MODEL*" afin de pouvoir les comparer avec les résultats présentés dans les sections précédentes.

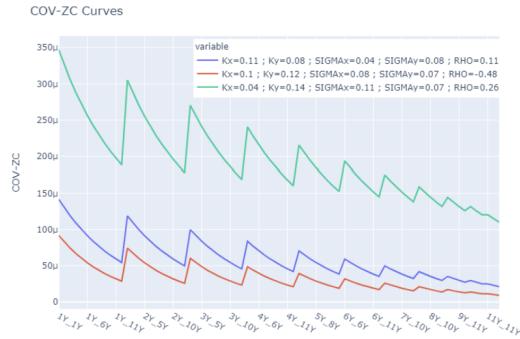


FIGURE 23 – Covariances des ZCs avec paramètres aléatoires

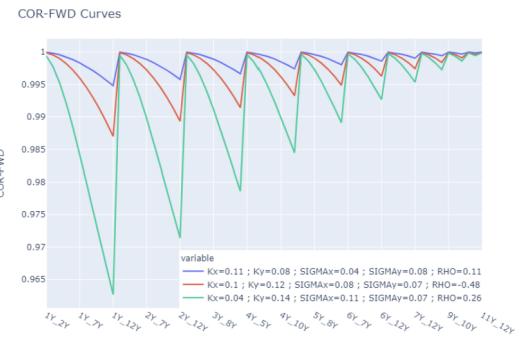


FIGURE 24 – Corrélation des FWDs avec paramètres aléatoires

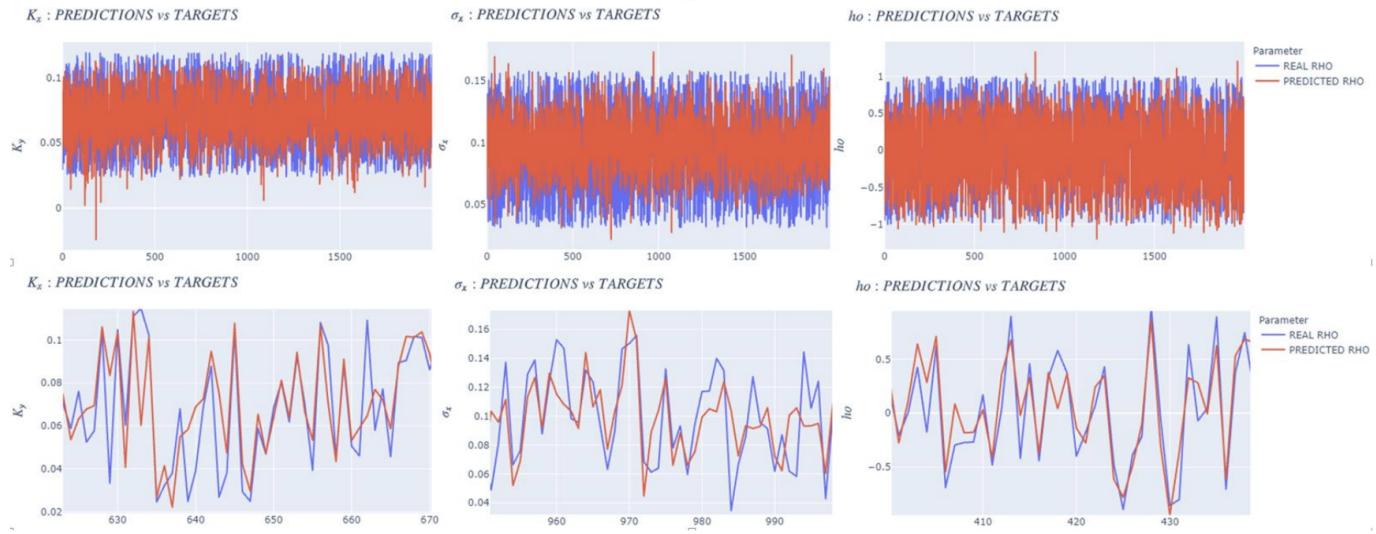


FIGURE 25 – Calibration de  $\kappa_x$ ,  $\sigma_x$  et  $\rho$  dans le cas de la Cov sur ZCs



FIGURE 26 – Calibration de  $\kappa_y$ ,  $\sigma_y$  et  $\rho$  dans le cas de la Corr sur FWDs

Market Data as of 2020/11/04

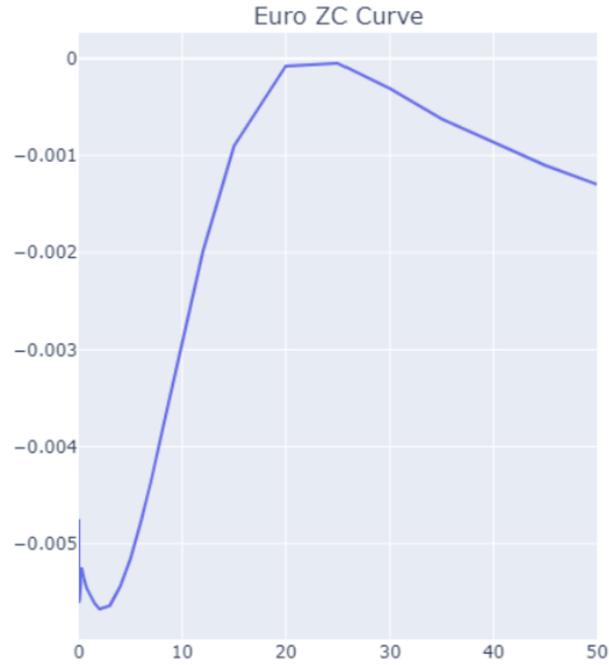


FIGURE 27 – Données de marché initial

Expectation ZC ||  $K_x = 0.07173$  |  $K_y = 0.08931$  |  $SIGMAx = 0.09466$  |  $SIGMAY = 0.09468$  |  $RHO = -0.99932$

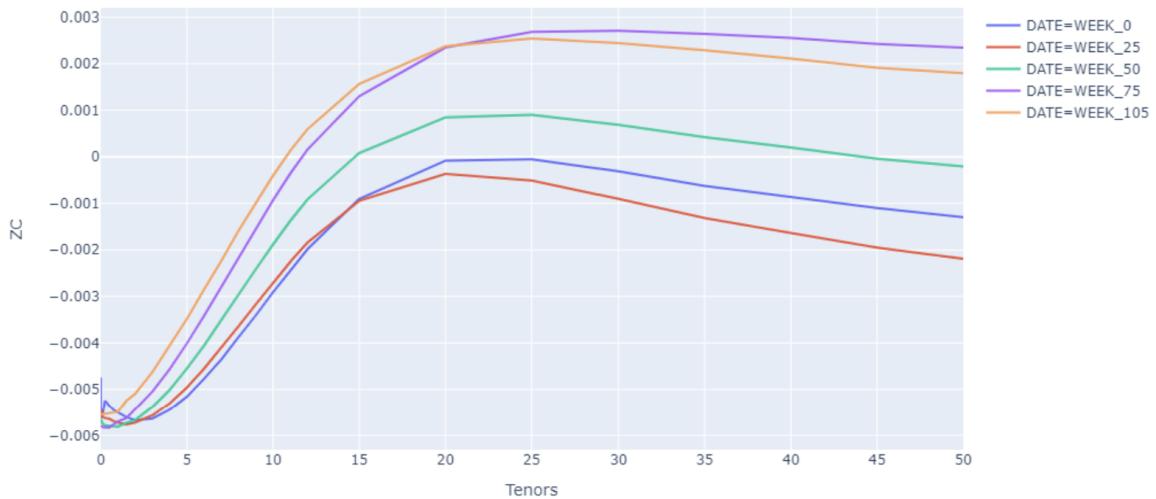


FIGURE 28 – Exemple d'une courbe de ZC avec des paramètres aléatoire

Ici, les courbes ne sont pas les mêmes. Cela s'explique car l'article utilise les données de marché du 4 novembre 2020, alors que nous utilisons celles du 27 février 2024. Cela n'affecte pas les résultats.

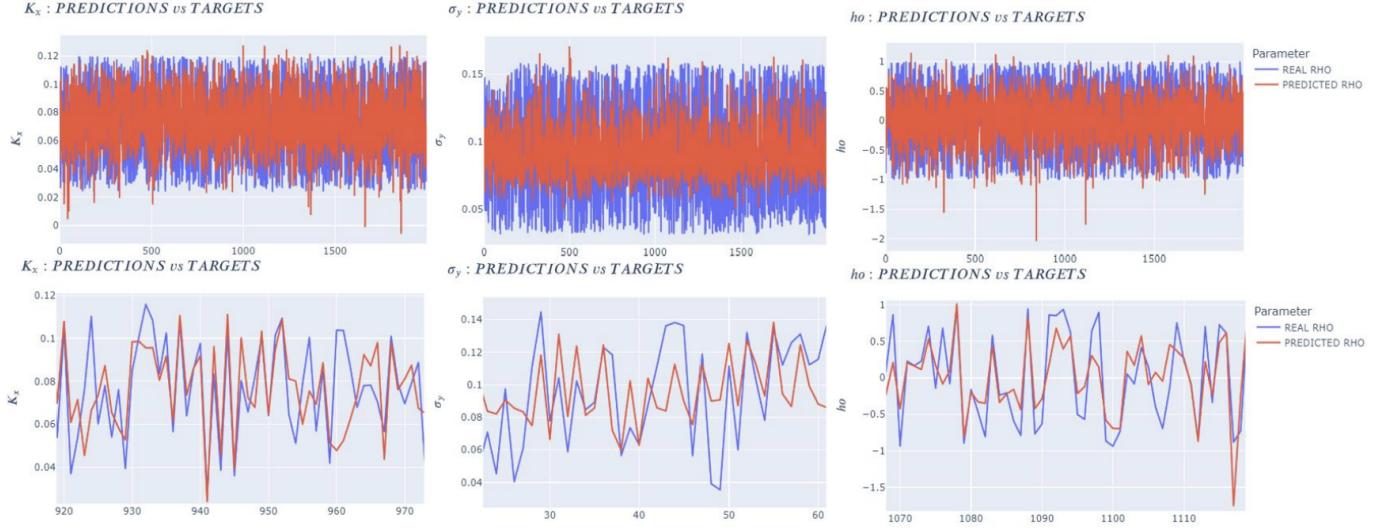


FIGURE 29 – Calibration de  $\kappa_y$ ,  $\sigma_y$  et  $\rho$  dans le cas direct

MSE	$K_x$	$K_y$	$\sigma_x$	$\sigma_y$	$\rho$
COV - ZC	$3.5 \times 10^{-4}$	$5.6 \times 10^{-4}$	$7.7 \times 10^{-4}$	$8.3 \times 10^{-4}$	$8.2 \times 10^{-2}$
COV - FWD	$3.5 \times 10^{-4}$	$5.5 \times 10^{-4}$	$7.6 \times 10^{-4}$	$8.3 \times 10^{-4}$	$8.2 \times 10^{-2}$
COR - ZC	$7.5 \times 10^{-4}$	$1.1 \times 10^{-3}$	$1.3 \times 10^{-3}$	$1.4 \times 10^{-3}$	$2.0 \times 10^{-1}$
COR - FWD	$7.4 \times 10^{-4}$	$1.1 \times 10^{-3}$	$1.3 \times 10^{-2}$	$1.4 \times 10^{-2}$	$2.4 \times 10^{-2}$
ZCs	$4.0 \times 10^{-4}$	$6.3 \times 10^{-4}$	$9.8 \times 10^{-4}$	$1.0 \times 10^{-4}$	$15.2 \times 10^{-2}$

TABLE 4 – Erreurs de calibration sur le test set

Finalement, nous retrouvons bien les résultats de l'article.