

Fits d'une gaussienne par réseaux de neurones

Clément Lotteau

May 2020

0.0.1 fit d'une gaussienne, variations du nombre d'outputs

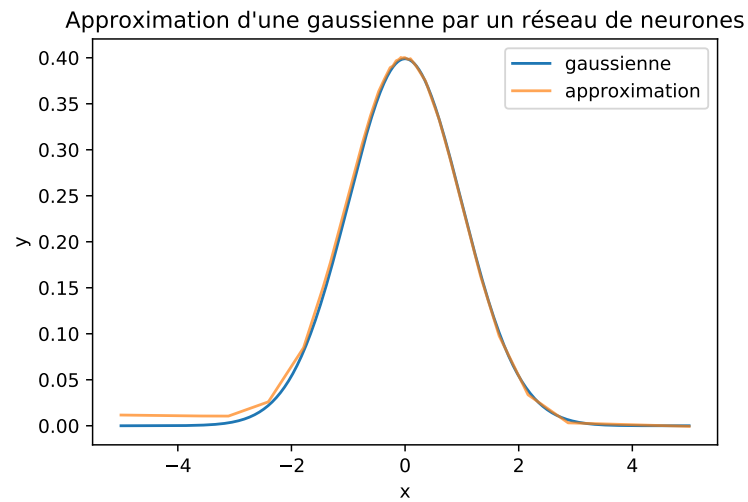


FIGURE 1 – Fit d'une gaussienne. 10001 points. outputs : 20, 20, 1. Total params : 481 Trainable params : 481. Epochs = 30, batch = 20

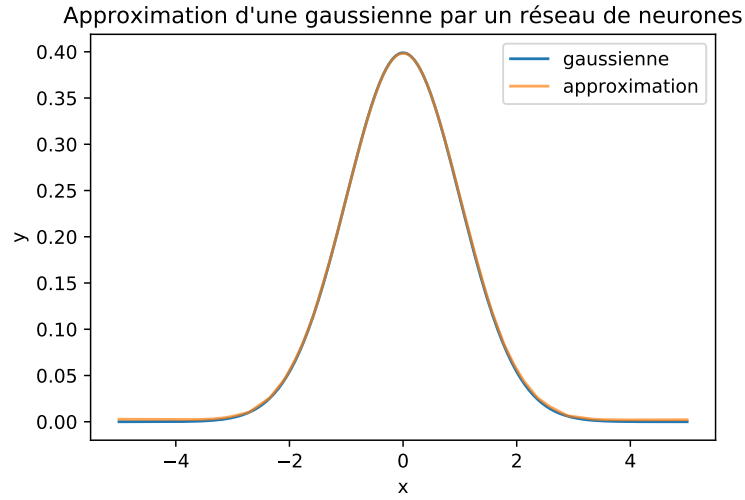


FIGURE 2 – Fit d'une gaussienne. 10001 points. outputs : 200, 200, 1. Total params : 40,801 Trainable params : 40,801. Epochs = 30, batch = 20

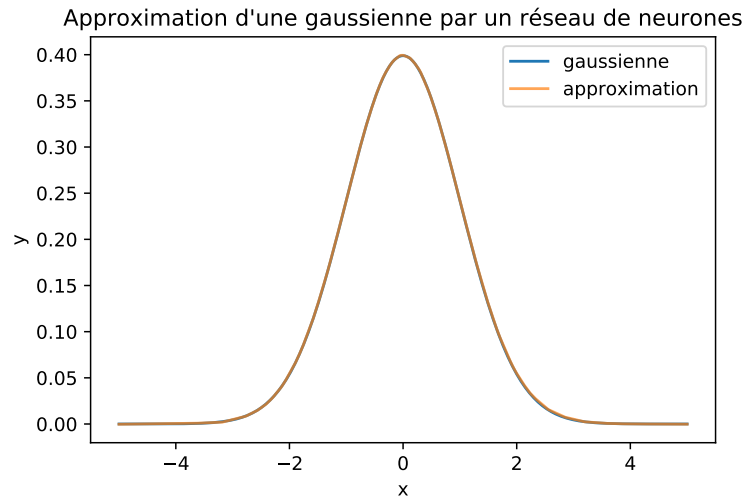


FIGURE 3 – Fit d'une gaussienne. 10001 points. outputs : 400, 400, 1. Total params : 161,601 Trainable params : 161,601. Epochs = 30, batch = 20

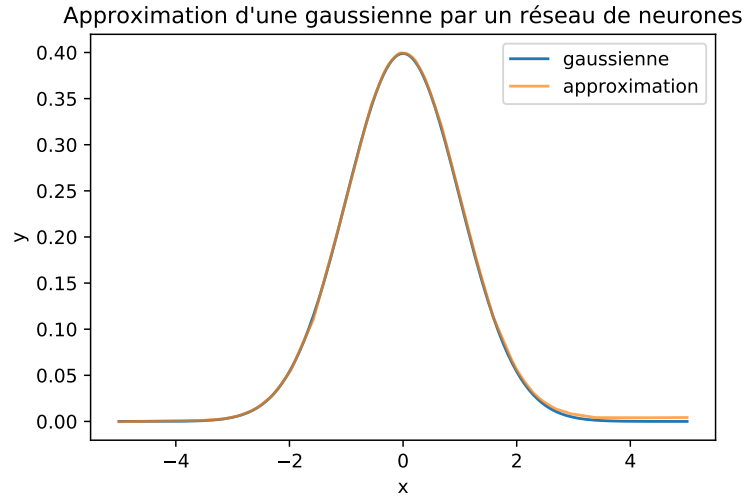


FIGURE 4 – Fit d'une gaussienne. 10001 points. outputs : 200, 20, 1. Total params : 4,441 Trainable params : 4,441. Epochs = 30, batch = 20

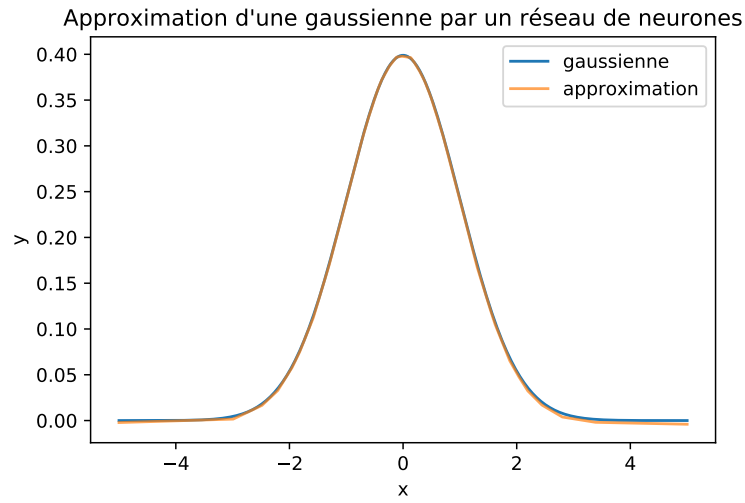


FIGURE 5 – Fit d'une gaussienne. 10001 points. outputs : 20, 200, 1. Total params : 4,441 Trainable params : 4,441. Epochs = 30, batch = 20

0.0.2 fit d'une gaussienne, variation du batch

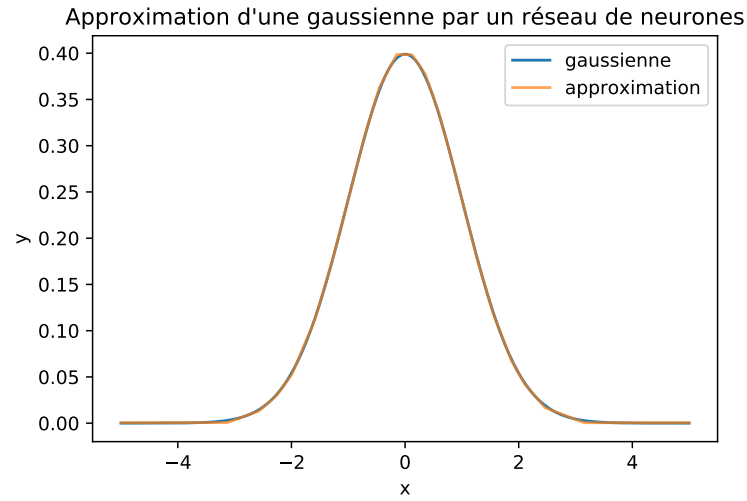


FIGURE 6 – Fit d'une gaussienne. 10001 points. outputs : 200, 200, 1. Total params : 40,801 Trainable params : 40,801. Epochs = 30, batch = 2

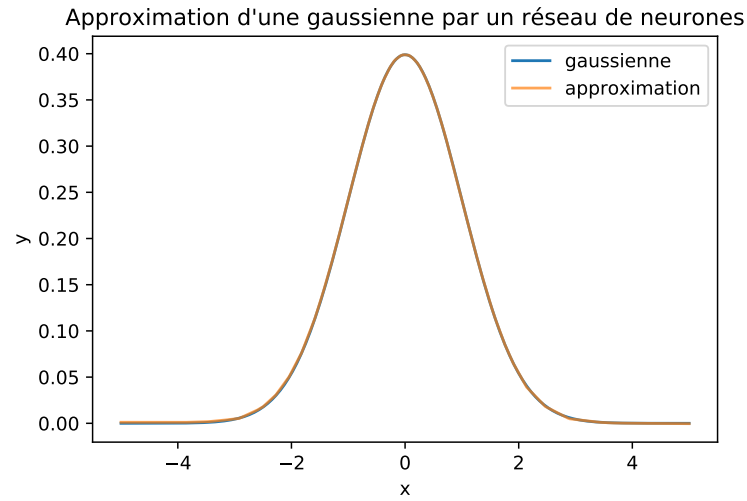


FIGURE 7 – Fit d'une gaussienne. 10001 points. outputs : 200, 200, 1. Total params : 40,801 Trainable params : 40,801. Epochs = 30, batch = 20

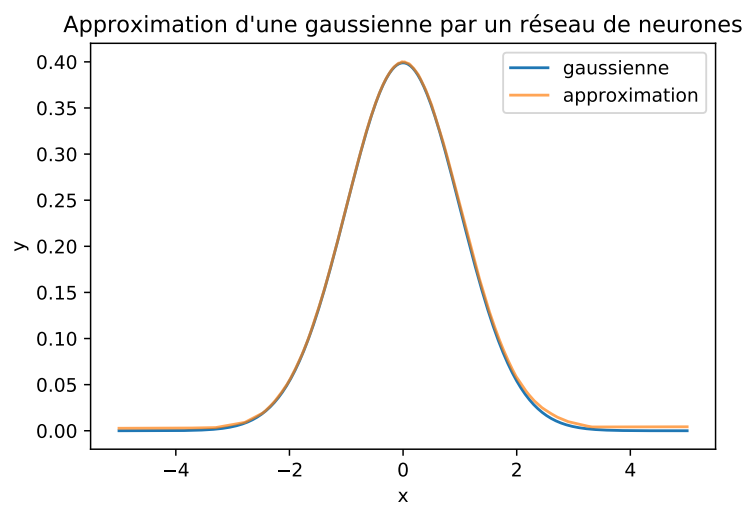


FIGURE 8 – Fit d'une gaussienne. 10001 points. outputs : 200, 200, 1. Total params : 40,801 Trainable params : 40,801. Epochs = 30, batch = 200

0.0.3 fit d'une gaussienne, variation des epochs

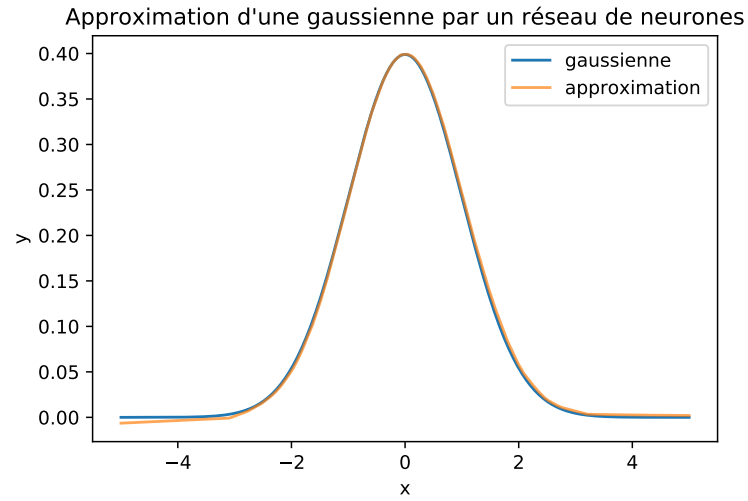


FIGURE 9 – Fit d'une gaussienne. 10001 points. outputs : 200, 200, 1. Total params : 40,801 Trainable params : 40,801. Epochs = 3, batch = 20

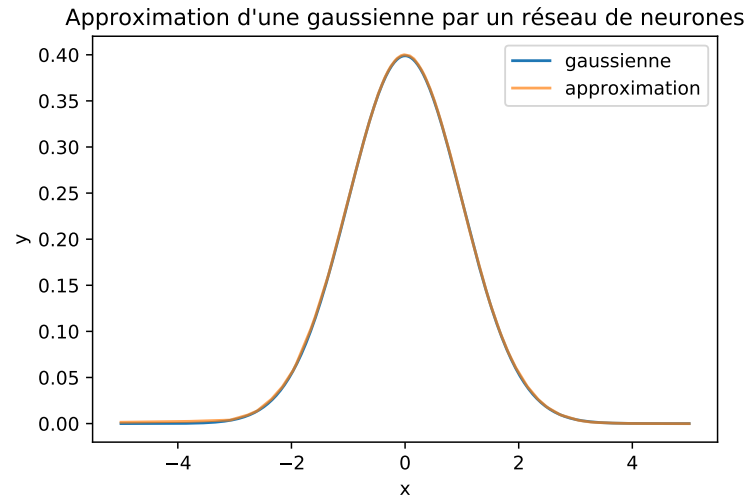


FIGURE 10 – Fit d'une gaussienne. 10001 points. outputs : 200, 200, 1. Total params : 40,801 Trainable params : 40,801. Epochs = 30, batch = 20

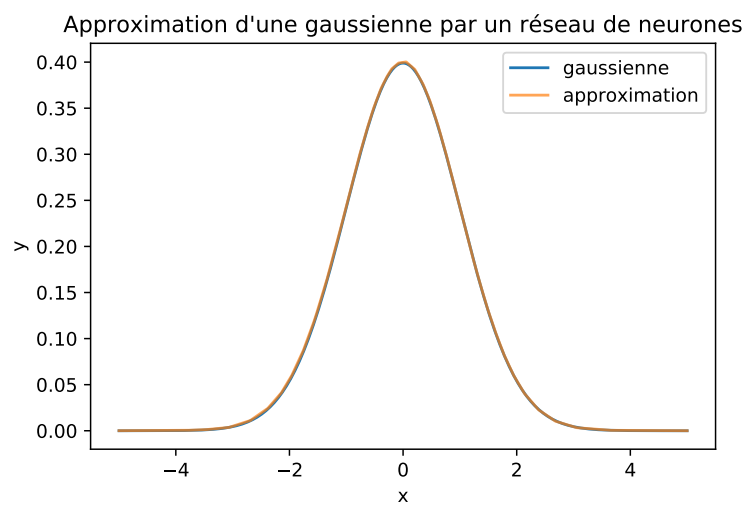


FIGURE 11 – Fit d'une gaussienne. 10001 points. outputs : 200, 200, 1. Total params : 40,801 Trainable params : 40,801. Epochs = 300, batch = 20

0.0.4 fit d'une gaussienne, variation batch vs epochs

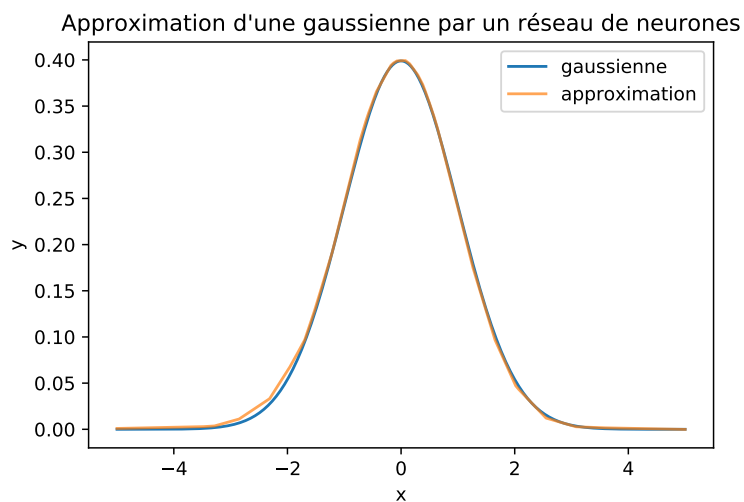


FIGURE 12 – Fit d'une gaussienne. 10001 points. outputs : 200, 200, 1. Total params : 40,801 Trainable params : 40,801. Epochs = 3, batch = 2. 77 secondes de calcul.

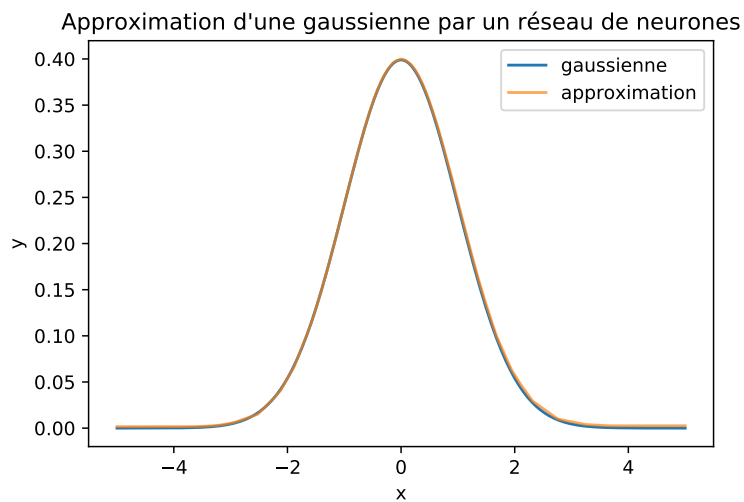


FIGURE 13 – Fit d'une gaussienne. 10001 points. outputs : 200, 200, 1. Total params : 40,801 Trainable params : 40,801. Epochs = 30, batch = 20. 83 secondes de calcul.

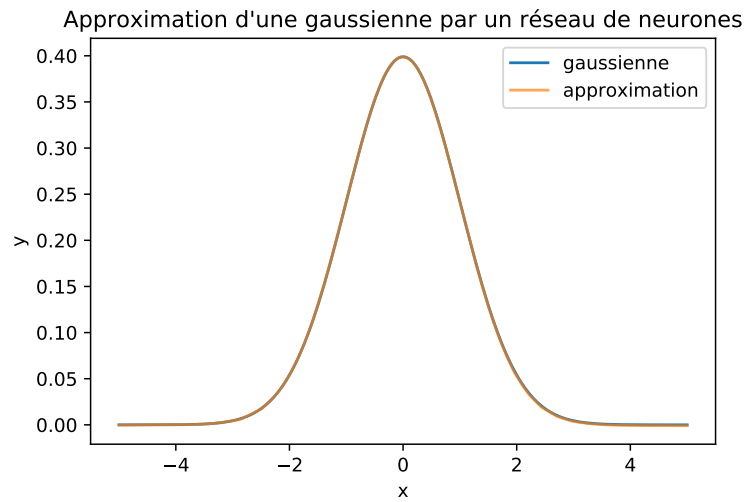


FIGURE 14 – Fit d'une gaussienne. 10001 points. outputs : 200, 200, 1. Total params : 40,801 Trainable params : 40,801. Epochs = 300, batch = 200. 96 secondes de calcul.

1 Différence gaussienne - fit

1.0.1 Écart gaussienne - fit. Calcul de la différence

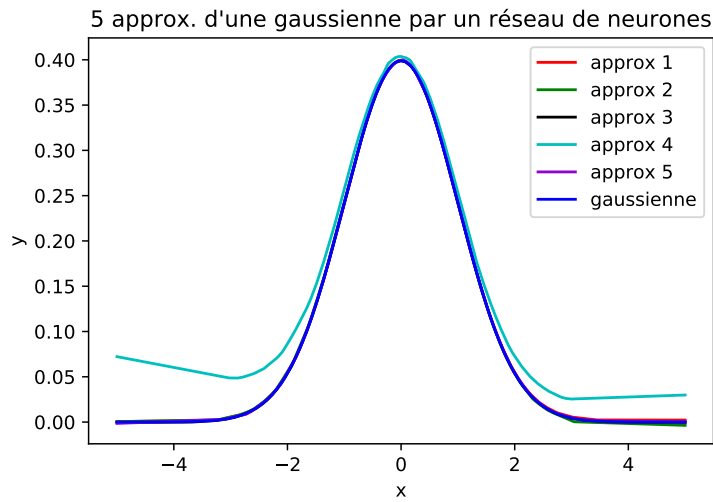


FIGURE 15 – 5 runs. 10001 points. outputs : 200, 200, 1. Epochs = 30, batch = 200.

On voit que certains apprentissages donnent des résultats très éloignés de la gaussienne. Il semble y avoir une part d'aléatoire.

Résultats sur 100 runs :

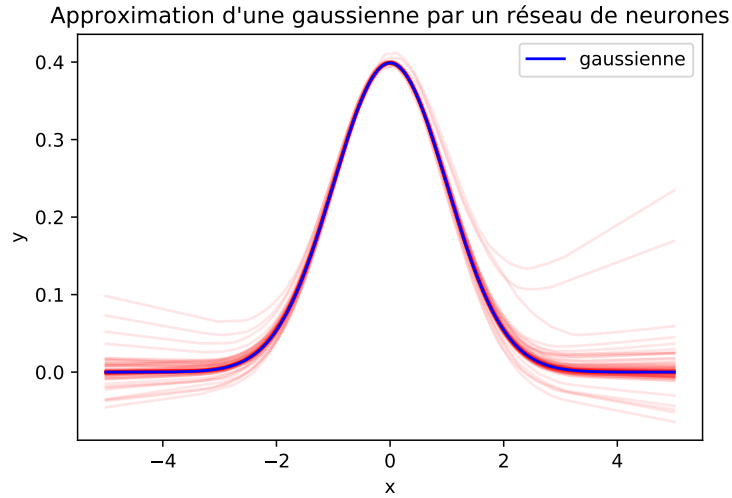


FIGURE 16 – 100 runs. 10001 points. outputs : 200, 200, 1. Epochs = 30, batch = 200.

On doit trouver les meilleurs paramètres pour minimiser la différence et le temps de calcul. On mesure la différence au carré en chaque point (10001) pour la gaussienne suivante :

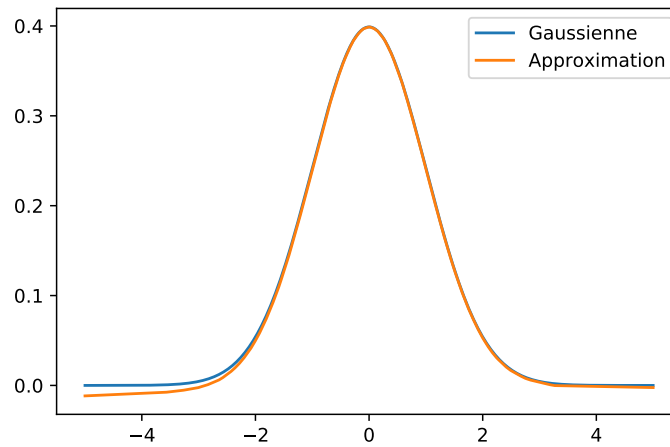


FIGURE 17 – 10001 points. outputs : 200, 200, 1. Epochs = 30, batch = 200.

Résultat :

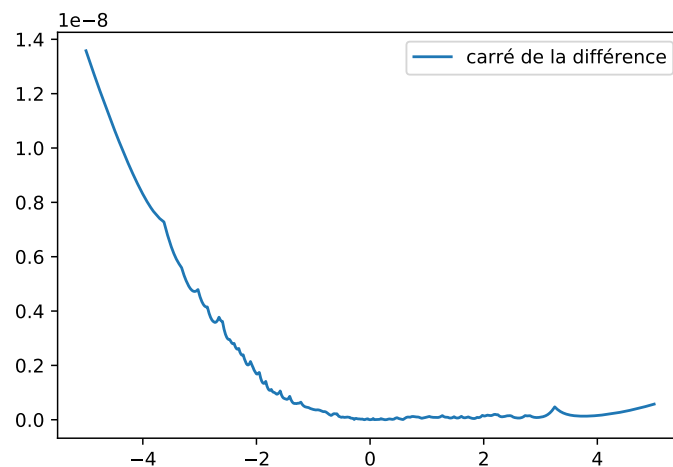


FIGURE 18 – Calcul du carré de la différence entre la gaussienne et le fit en chaque point de la figure précédente.

On somme ensuite toutes ces différences et on divise par le nombre de points pour obtenir la différence totale entre les courbes. On fait ensuite varier les paramètres suivants : nombre d'outputs, epochs, batch size.

1.0.2 Écart gaussienne - fit. Variation outputs

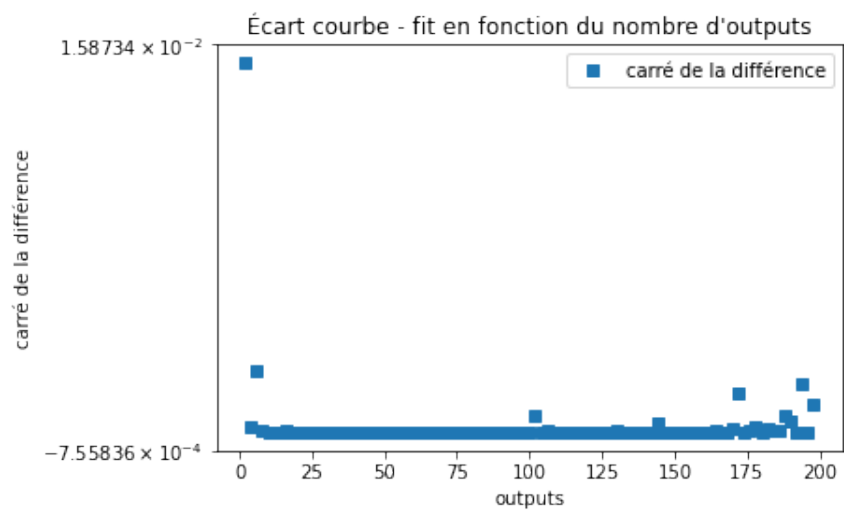


FIGURE 19 – 30 epochs, 200 batchsize.

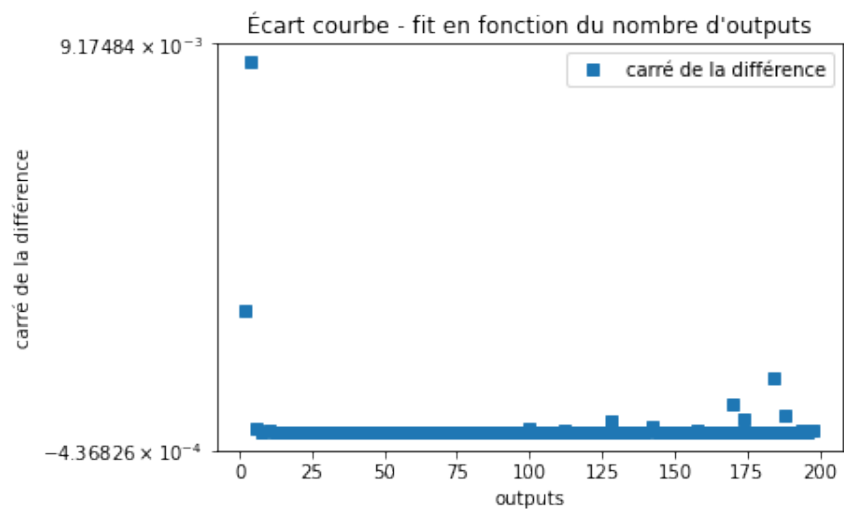


FIGURE 20 – 30 epochs, 200 batchsize.

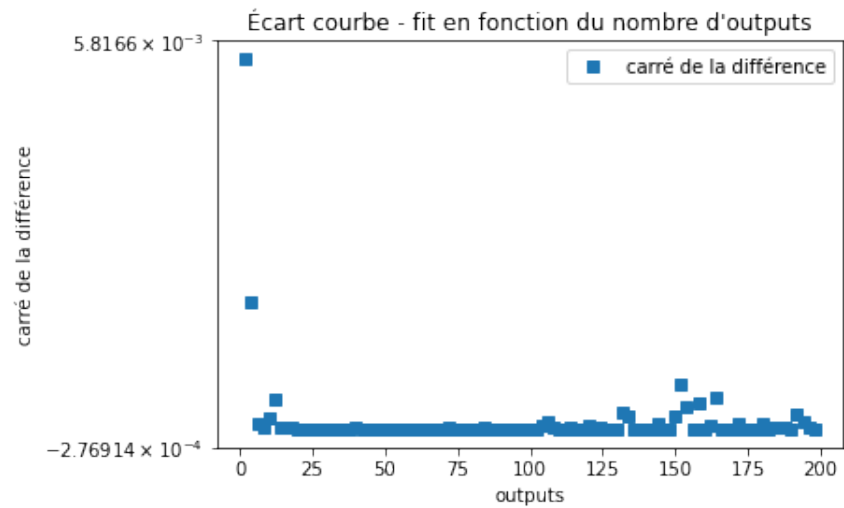


FIGURE 21 – 30 epochs, 200 batchsize.

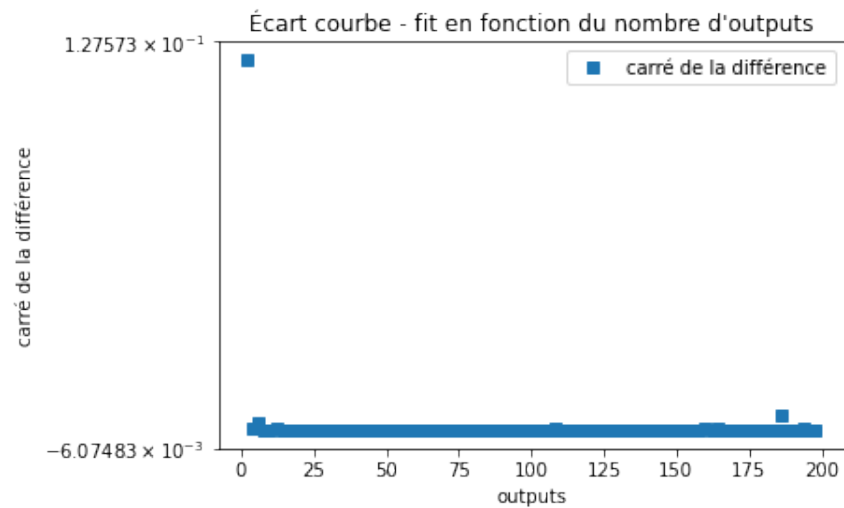


FIGURE 22 – 30 epochs, 200 batchsize.

On observe une part importante d'aléatoire quand le nombre d'ouputs est grand. Minimiser ce nombre nous permettrait ainsi de gagner en temps de calcul et en précision.

1.0.3 Écart gaussienne - fit. Variation epochs

Ici je fais varier le nombre d'apprentissage.

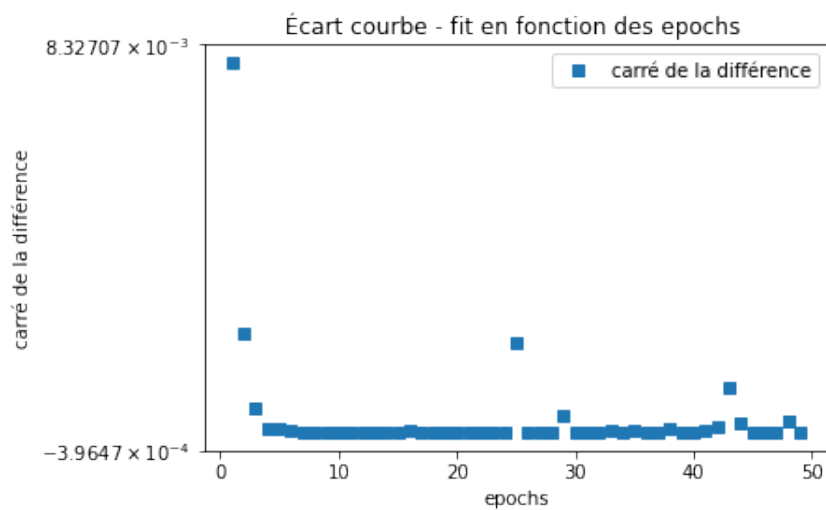


FIGURE 23 – 200 outputs, 200 batchsize.

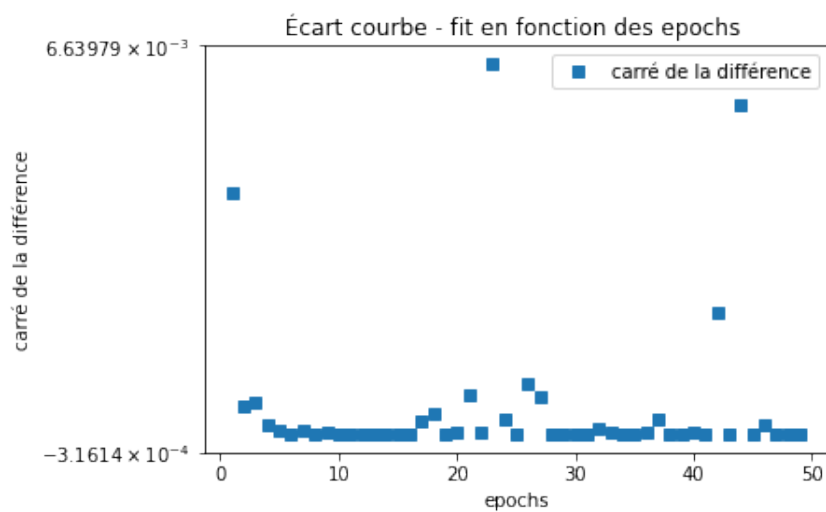


FIGURE 24 – 200 outputs, 200 batchsize.

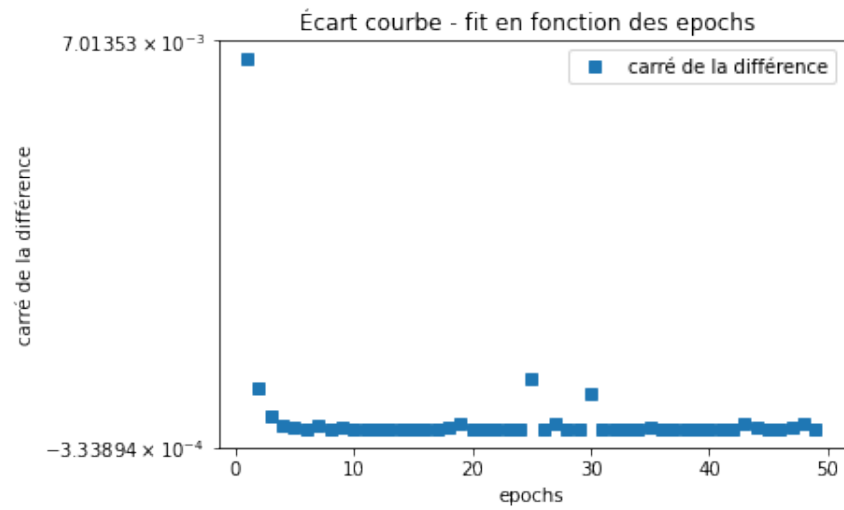


FIGURE 25 – 200 outputs, 200 batchsize.

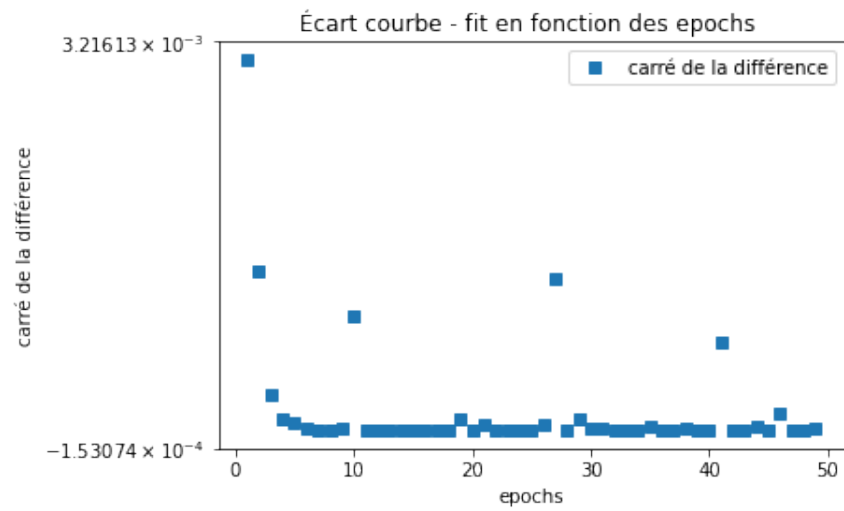


FIGURE 26 – 200 outputs, 200 batchsize.

Il semble se dessiner des zones dans lesquelles l'aléatoire est plus important. Il est néanmoins difficile de conclure avec seulement 4 figures.

1.0.4 Écart gaussienne - fit. Variation batchsize

Le batch size est un paramètre important car, plus il est grand, plus le temps de chaque apprentissage est court.

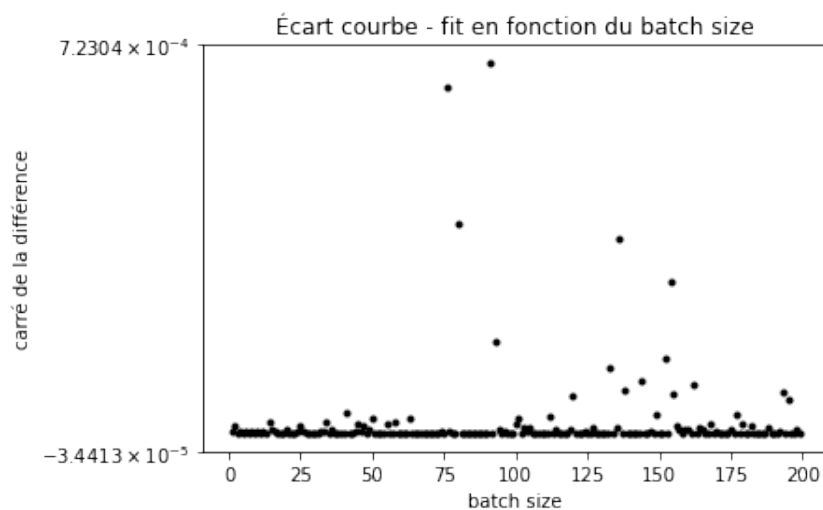


FIGURE 27 – 200 outputs, 30 epochs

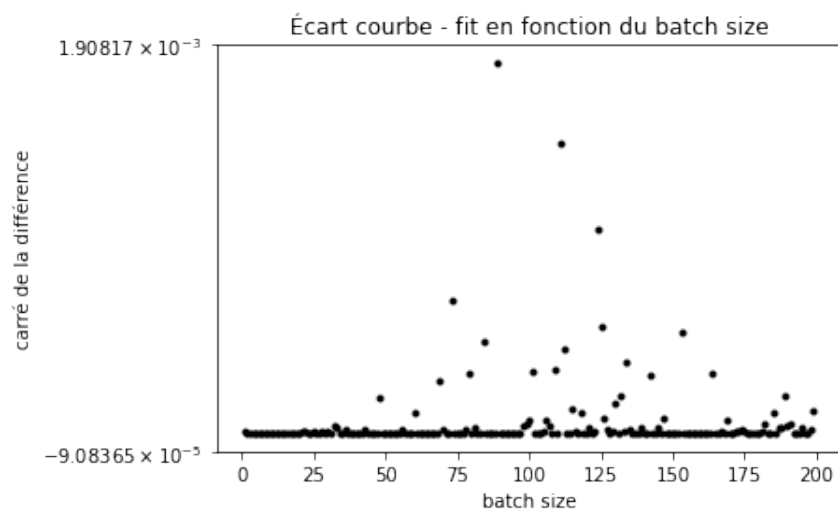


FIGURE 28 – 200 outputs, 30 epochs

2 Temps de calcul

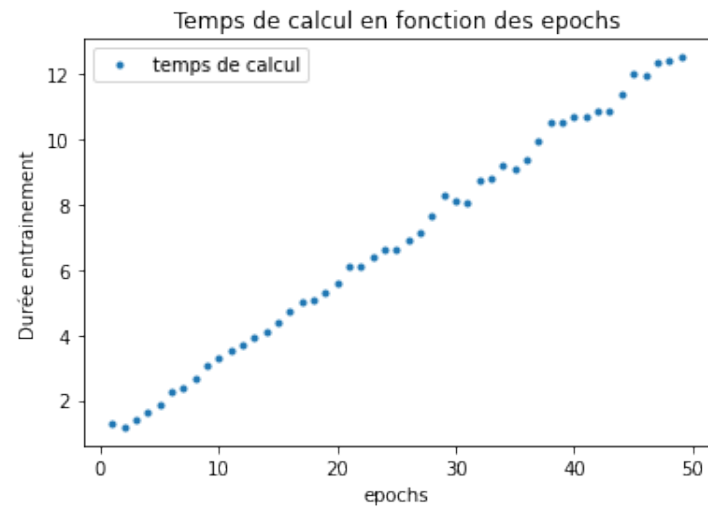


FIGURE 29 – 100 batch size, 200 outputs.

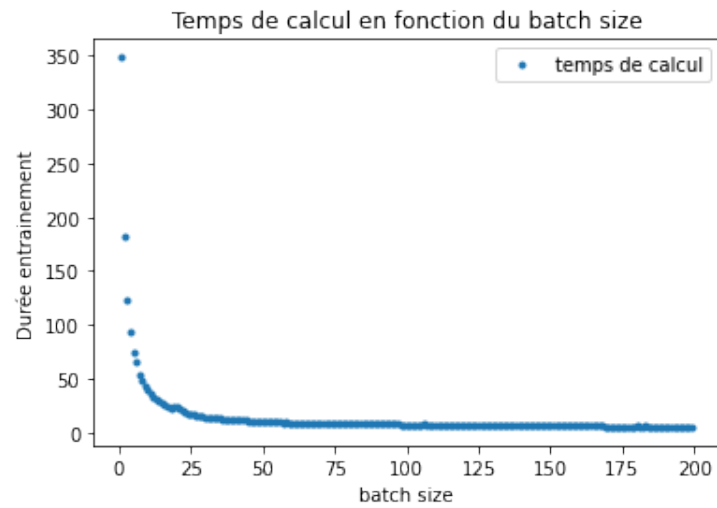


FIGURE 30 – Temps en secondes. 200 outputs, 30 epochs

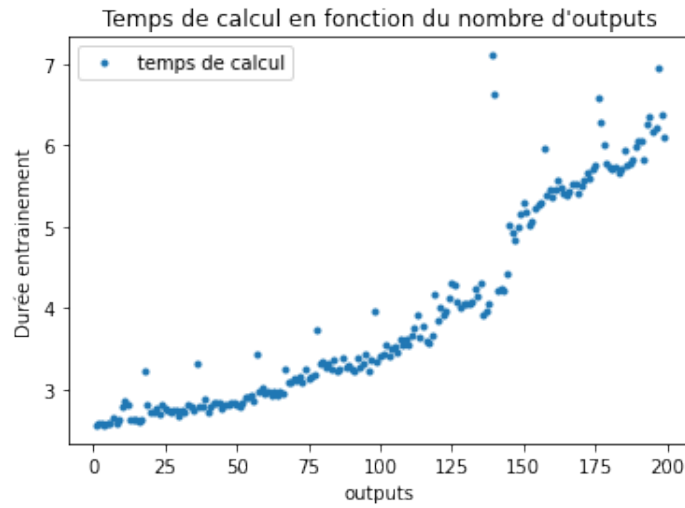


FIGURE 31 – Temps en secondes. 200 batch size, 30 epochs

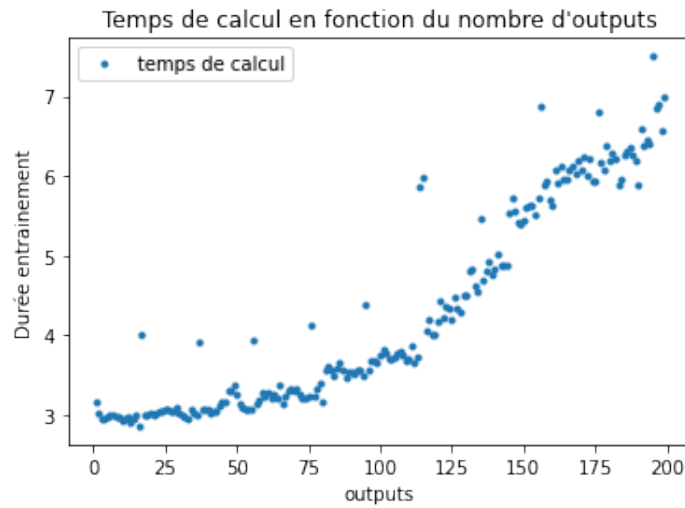


FIGURE 32 – Temps en secondes. 200 batch size, 30 epochs

Il semble y avoir une discontinuité dans le temps de calcul aux alentours de 150 outputs ainsi qu'un pattern régulier pour certaines valeurs entre 0 et 100. Je relance Le programme avec 3 runs par nombre d'outputs (r,g,b).

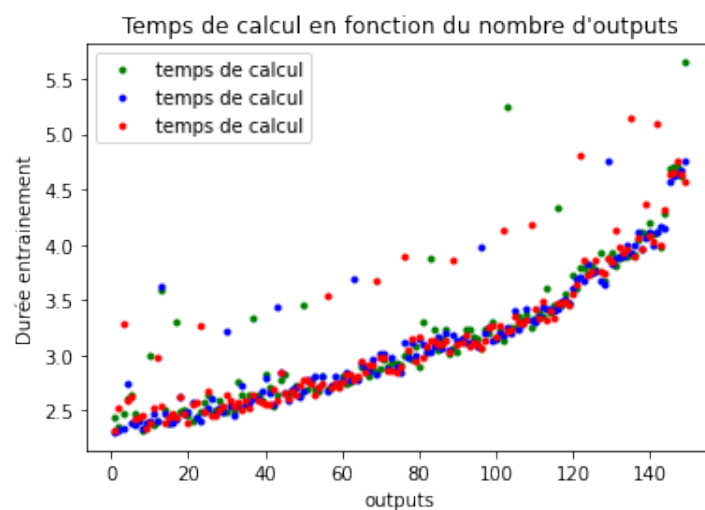


FIGURE 33 – Temps en secondes. 200 batch size, 30 epochs. Aucune différence entre les couleurs (3 runs par nombre d'outputs : r,g,b). On observe encore une discontinuité vers 150 epochs. Il apparait une probabilité d'un temps de calcul plus long pour tout nombre d'outputs.

3 Nouveau réglage

Je vais maintenant essayer avec 30 outputs, 100 de batch size et 15 epochs.
Le nombre de paramètre passe de 40801 à 1021.

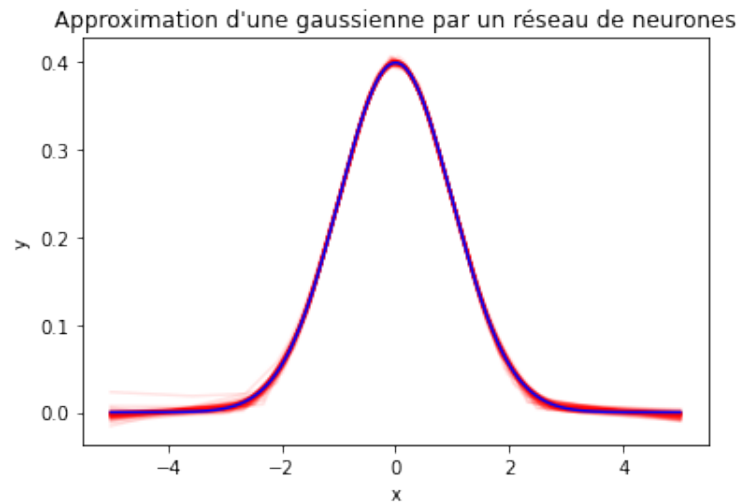


FIGURE 34 – 30 outputs, 15 epochs, 100 batchsize.

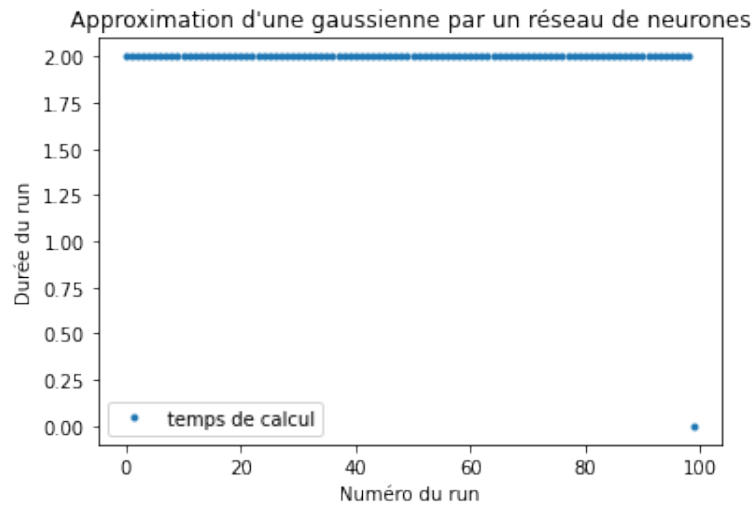


FIGURE 35 – Temps en secondes. 30 outputs, 15 epochs, 100 batchsize. Le dernier point ne correspond pas à un calcul et peut être éliminé.

On observe moins d'erreurs aléatoires de fit. Ces erreurs venaient peut-être du nombre trop élevé d'ouputs, générant un sur-entraînement. Je recommence les mesures de différence et de temps de calcul avec ces nouveaux réglages.

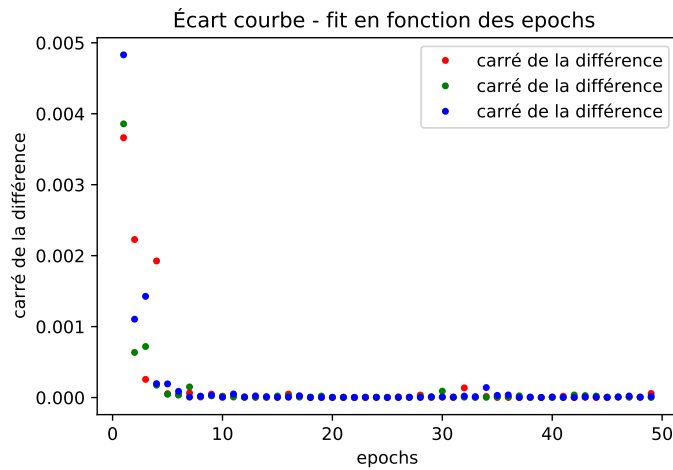


FIGURE 36 – 30 outputs, 100 batchsize. 15 epochs semble être un bon compromis pour minimiser la différence. Il ne semble pas y avoir d'amélioration de la précision pour un plus grand nombre d'epochs, et on reste assez loin de la zone entre 0 et 10 générant beaucoup d'erreurs.

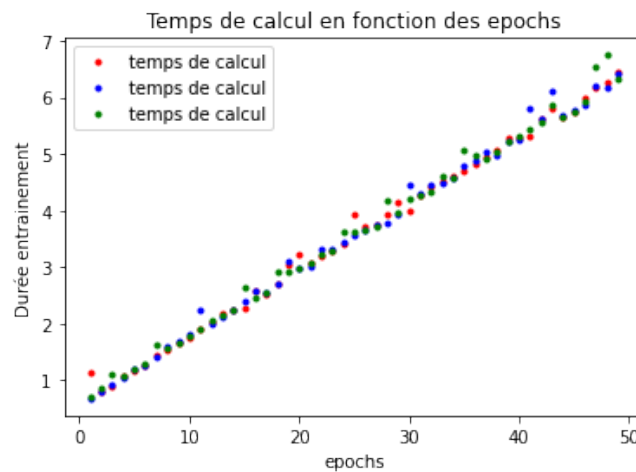


FIGURE 37 – Temps en secondes. 30 outputs, 100 batchsize. Le temps de calcul semble plus aléatoire lorsque le nombre d'epochs augmente.

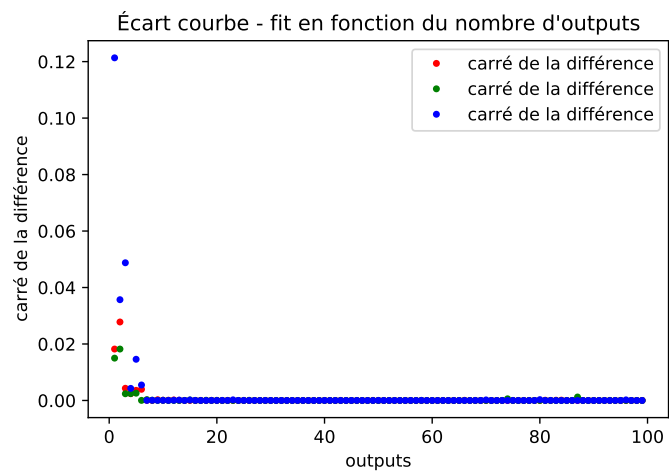


FIGURE 38 – 15 epochs, 100 batchsize. Il semble possible de diminuer encore le nombre d'outputs (20 par exemple). Je devrais faire une mesure plus précise entre 15 et 30.

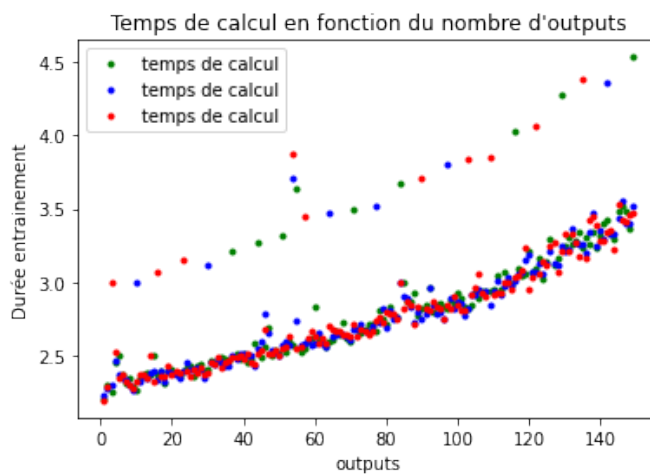


FIGURE 39 – Temps en secondes. 15 epochs, 100 batchsize.

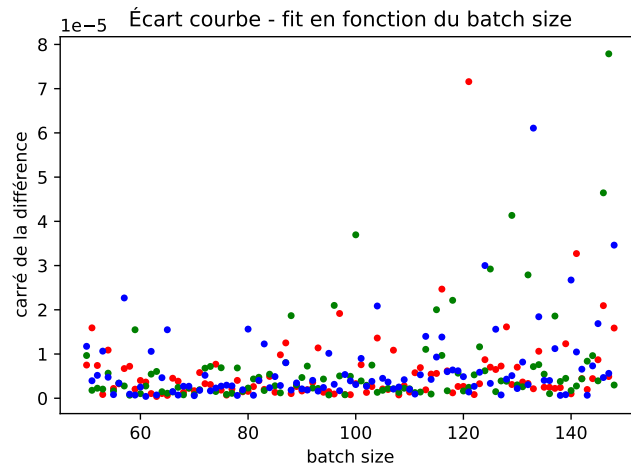


FIGURE 40 – 30 outputs, 15 epochs. Les erreurs aléatoires de calcul semblent plus probable lors que le batchsize augmente.

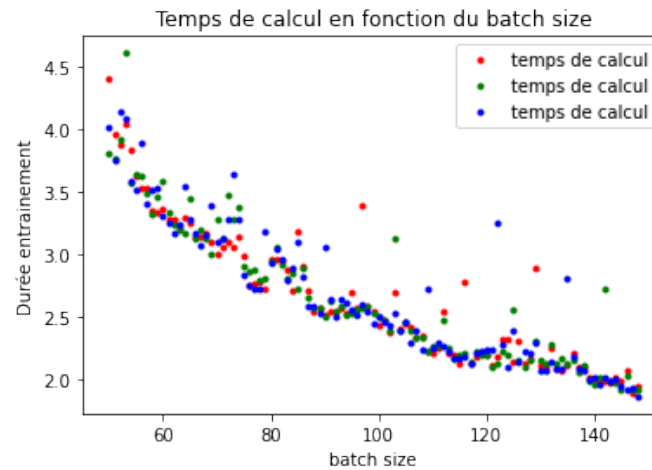


FIGURE 41 – Temps en secondes. 30 outputs, 15 epochs.

Idée : Maximiser le batchsize pour gagner du temps et faire une boucle infini qui exclue les apprentissages ratés.

Question : Est-ce qu'on gagne du temps en sortie ? Quitte à faire une boucle infinie, est-ce qu'on gagne PLUS de temps avec une boucle infinie et un batchsize plus faible (on entrerait moins fréquemment dans la boucle) ?

Question : Ces paramètres changent-ils pour d'autres gaussiennes ?

- Mesurer la différence et le temps d'entraînement en fonction du nombre de

points de la gaussienne

- Mesurer la différence et le temps d'entraînement en fonction de l'écart-type de la gaussienne

Question : Comment optimiser le fit proche de 0 ?

4 Dérivée 3 points

Dérivée première :

$$\frac{df}{dx} = \frac{f(x+h) - f(x)}{h} \quad (1)$$

Dérivée seconde :

$$\frac{d^2f}{dx^2} = \frac{f(x+h) + f(x-h) - 2f(x)}{h^2} \quad (2)$$

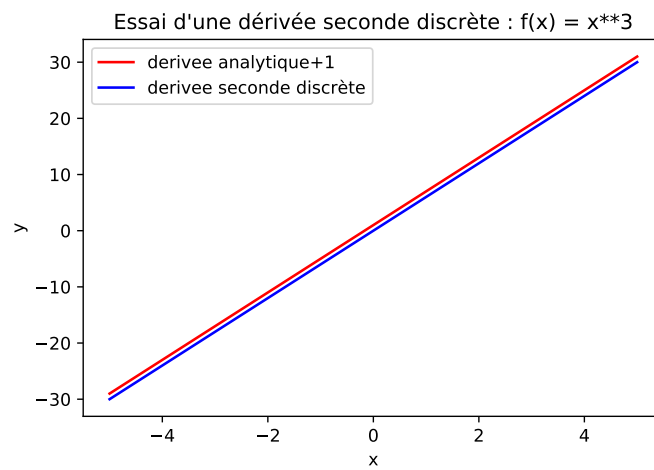


FIGURE 42 –

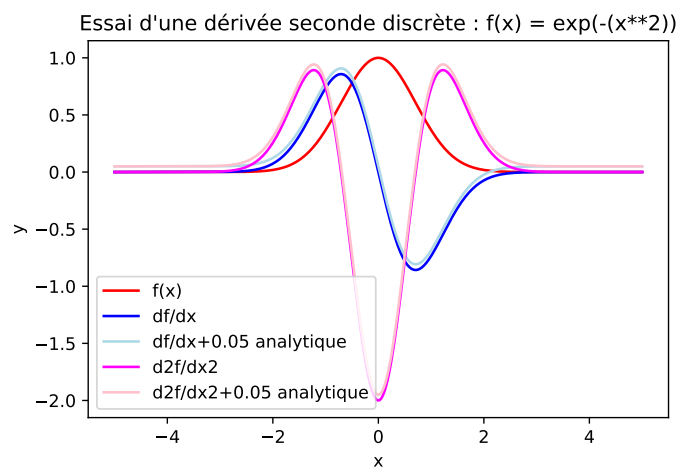


FIGURE 43 –

Maintenant je dérive la gaussienne de la partie précédente ainsi que son approximation par machine learning.

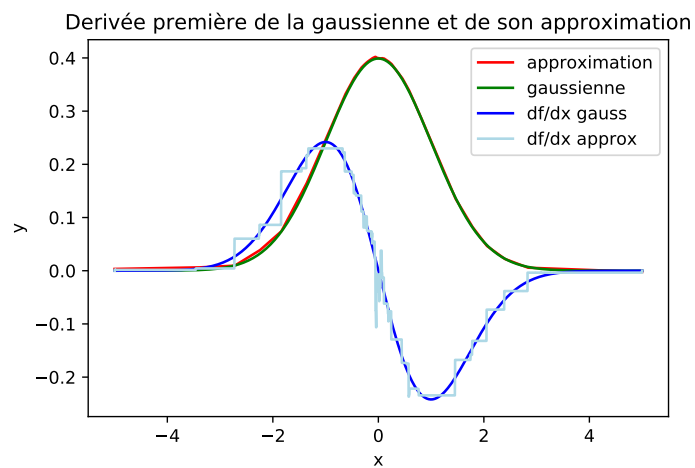


FIGURE 44 – 30 outputs, 15 epochs, 100 batchsize.

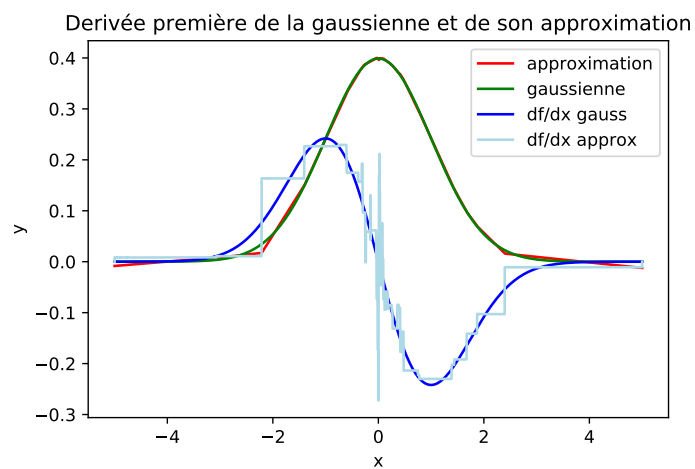


FIGURE 45 – 30 outputs, 15 epochs, 100 batchsize.

Et la dérivée seconde :

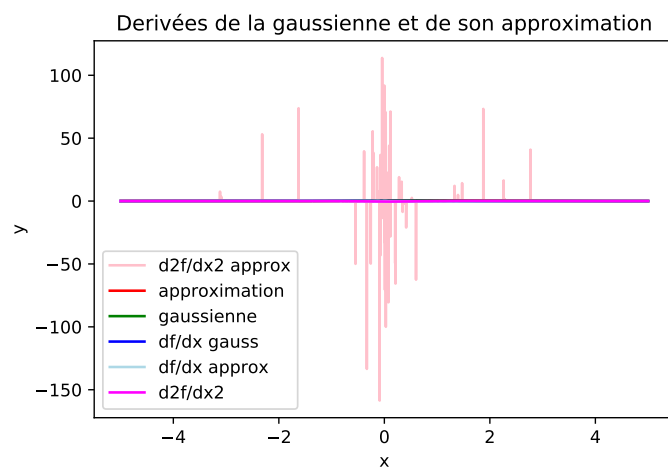


FIGURE 46 – 30 outputs, 15 epochs, 100 batchsize.

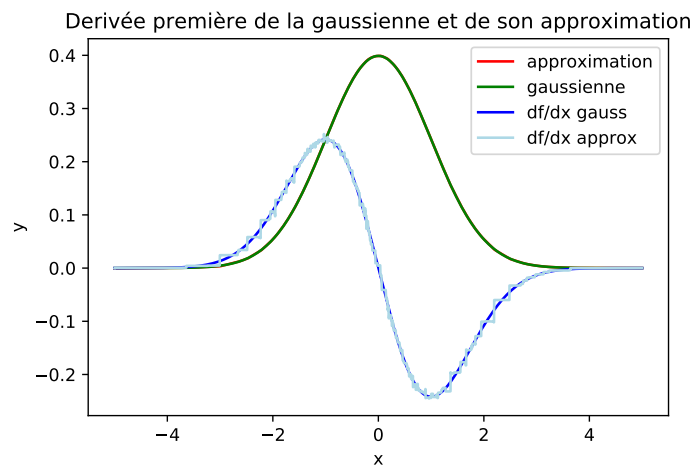


FIGURE 47 – 200 outputs, 100 epochs, 100 batchsize.

C'est mieux, essayons la dérivée seconde.

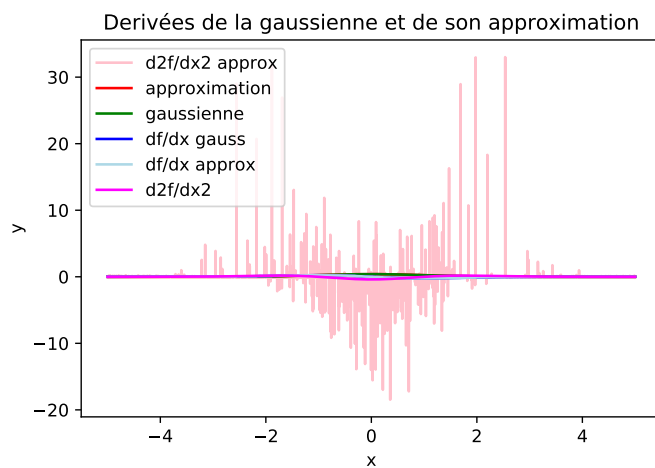


FIGURE 48 – 200 outputs, 100 epochs, 100 batchsize.

On reconnaît la forme attendue de la dérivée seconde (figure 43), mais le résultat reste inexploitable.

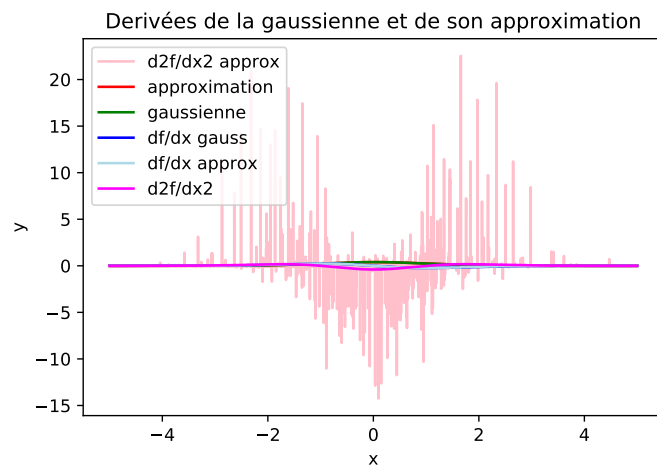


FIGURE 49 – 300 outputs, 100 epochs, 100 batchsize. 91.587 secondes de calcul. Légère amélioration, peut-être un coup de chance.

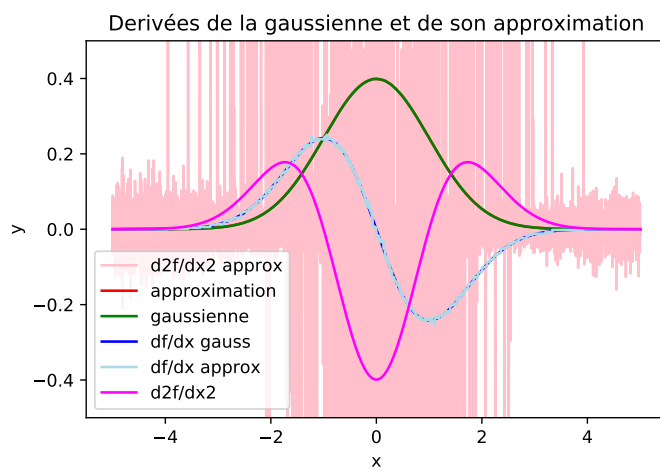


FIGURE 50 – 500 outputs, 20 epochs, 50 batchsize. 57.19 secondes de calcul.

4.0.1 Fit des polynômes d’hermite - variation du nombre de points et d’outputs (annexes)

On voit en annexes A et B que, pour un même réseau et un échantillon de 500 points, l’approximation d’une fonction d’onde semble plus précise que l’approximation du module carré. Cette différence, si elle existe, est plus difficile à détecter visuellement pour un échantillon à 10001 points (annexes C et D).

Mémo : [2] [8] [6] [3] [5] [4] [1] [7]

Bibliographie

- [1] Colin BERNET. *Handwritten Digit Recognition with scikit-learn*. URL : <https://thedatafrog.com/en/articles/handwritten-digit-recognition-scikit-learn/>.
- [2] Colin BERNET. *Le réseau à un neurone : régression logistique*. URL : <https://thedatafrog.com/fr/articles/logistic-regression/>.
- [3] Colin BERNET. *Le surentraînement*. URL : <https://thedatafrog.com/fr/articles/overfitting-illustrated/>.
- [4] Colin BERNET. *Matplotlib for Machine Learning*. URL : <https://thedatafrog.com/en/articles/matplotlib-machine-learning/>.
- [5] Colin BERNET. *Numpy Crash Course for Machine Learning*. URL : <https://thedatafrog.com/en/articles/numpy-crash-course-machine-learning/>.
- [6] Colin BERNET. *Premier réseau de neurones avec keras*. URL : <https://thedatafrog.com/fr/articles/first-neural-network-keras/>.
- [7] Colin BERNET. *Python Crash Course for Machine Learning*. URL : <https://thedatafrog.com/en/articles/python-crash-course-machine-learning/>.
- [8] Colin BERNET. *Régression Logistique vs Réseau de Neurones : Non Linéarités*. URL : <https://thedatafrog.com/fr/articles/logistic-regression-neural-network/>.

A Approximation des pol. d'Hermite par réseau de neurones (500 points)

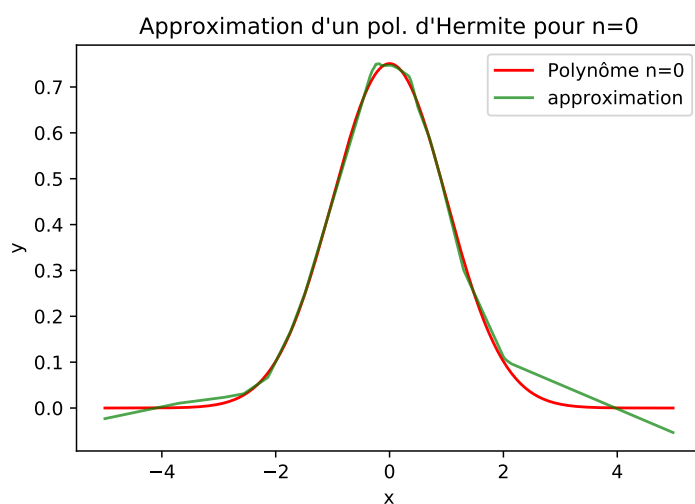


FIGURE 51 – $n=0$, 500 points. outputs : 20 20 1. Params : 481. Trainable : 481.

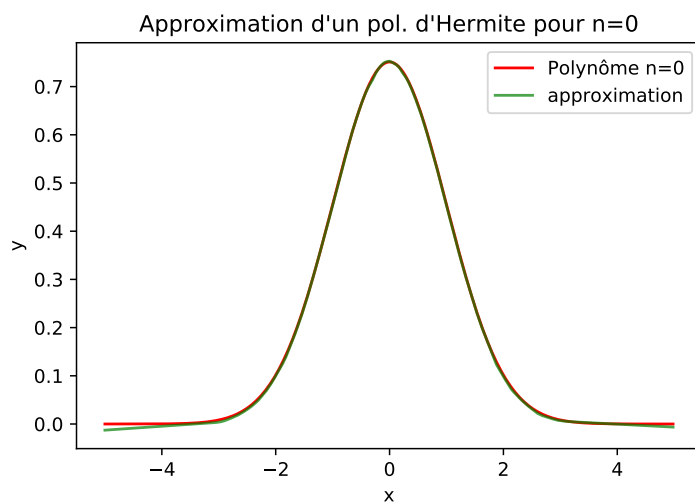


FIGURE 52 – $n=0$, 500 points. Réseau 200 200 1. Params : 40801. Trainable : 40801.

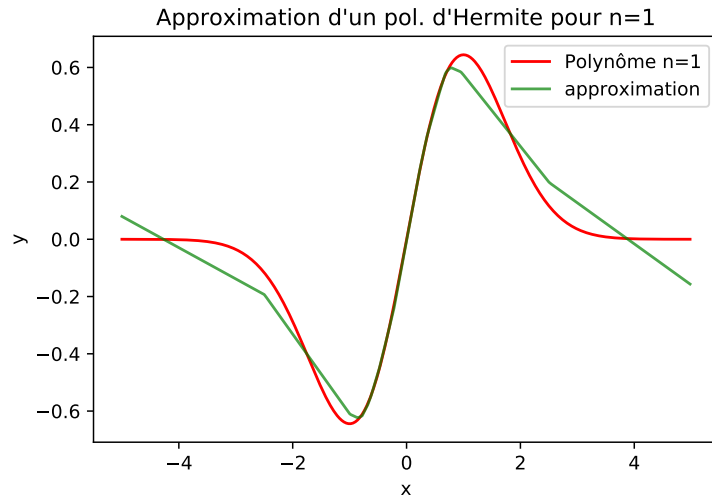


FIGURE 53 – $n=1$, 500 points. outputs : 20 20 1. Params : 481. Trainable : 481.

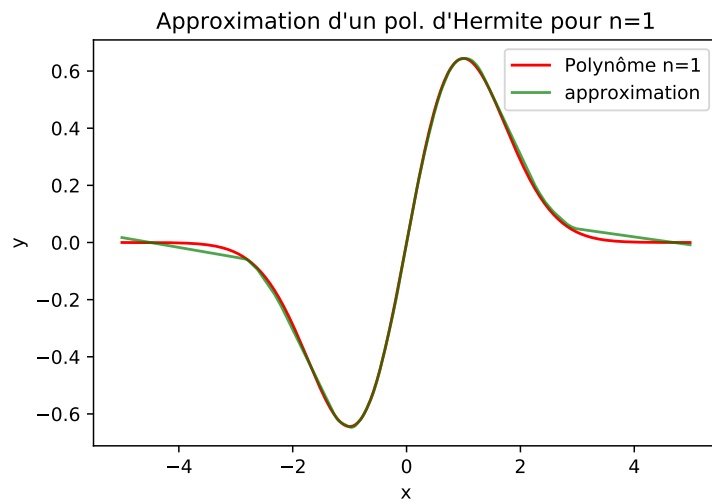


FIGURE 54 – $n=1$, 500 points. outputs : 200 200 1. Params : 40801. Trainable : 40801.

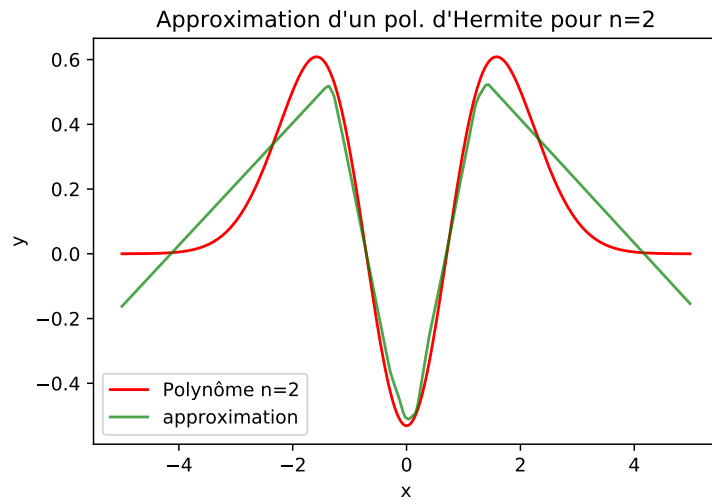


FIGURE 55 – $n=2$, 500 points. outputs : 20 20 1. Params : 481. Trainable : 481.

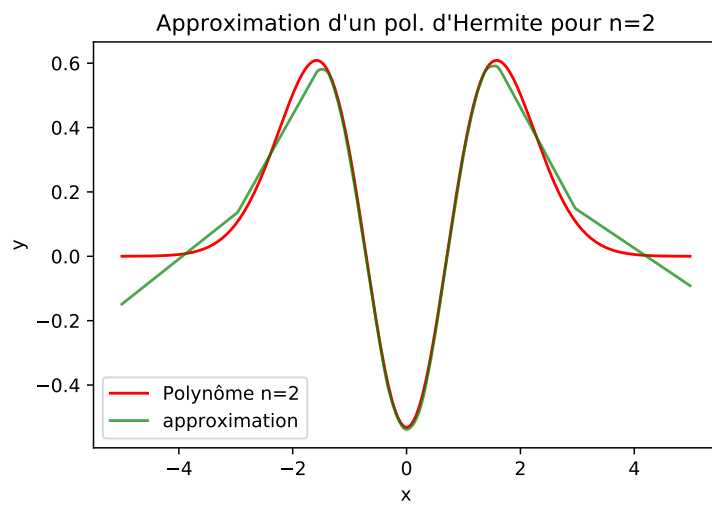


FIGURE 56 – $n=2$, 500 points. outputs : 200 200 1. Params : 40801. Trainable : 40801.

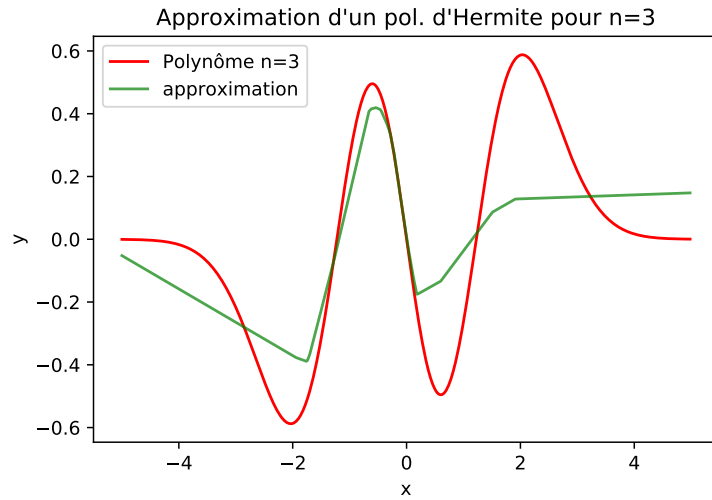


FIGURE 57 – $n=3$, 500 points. outputs : 20 20 1. Params : 481. Trainable : 481.

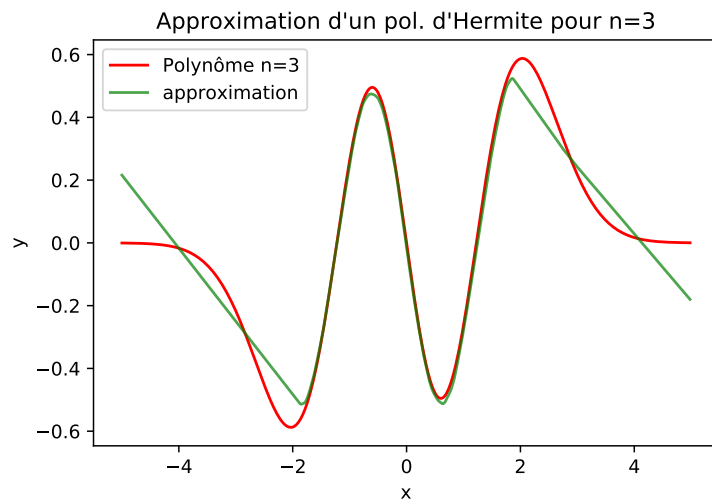


FIGURE 58 – $n=3$, 500 points. outputs : 200 200 1. Params : 40801. Trainable : 40801.

B Approximation des modules carrés des pol. d'Hermite par réseau de neurones (500 points)

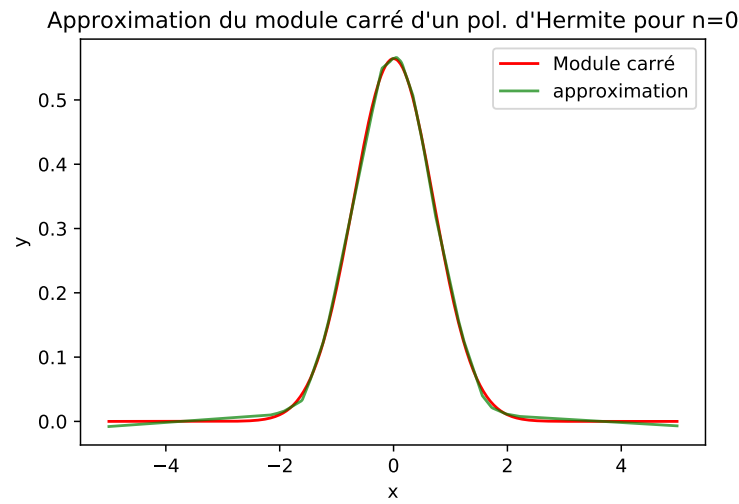


FIGURE 59 – $n=0$, 500 points. outputs : 20 20 1. Params : 481. Trainable : 481.

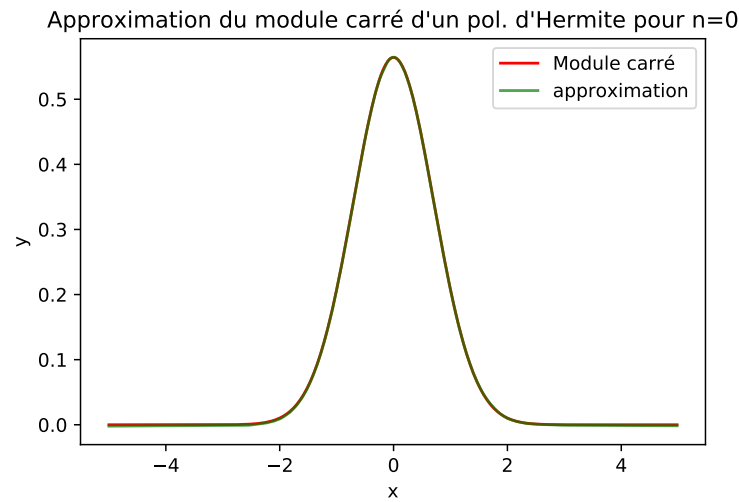


FIGURE 60 – $n=0$, 500 points. outputs : 200 200 1. Params : 40801. Trainable : 40801.

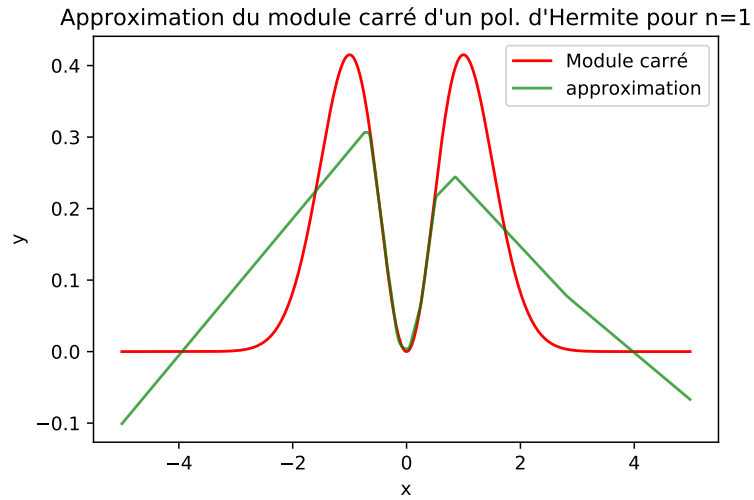


FIGURE 61 – $n=1$, 500 points. outputs : 20 20 1. Params : 481. Trainable : 481.

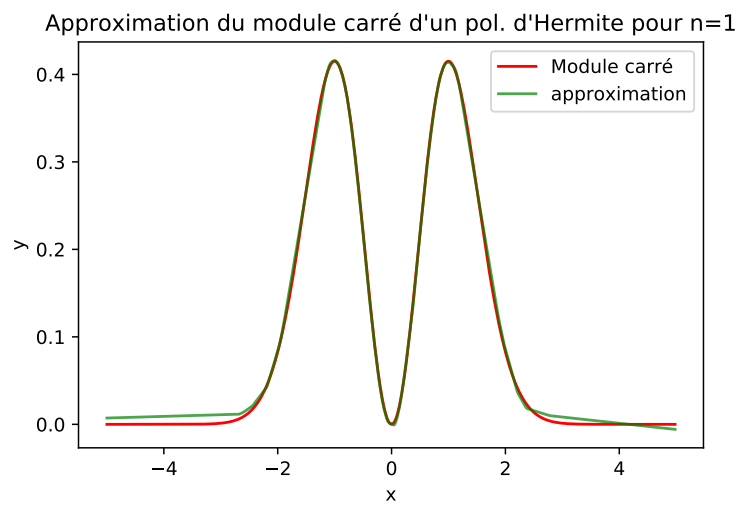


FIGURE 62 – $n=1$, 500 points. outputs : 200 200 1. Params : 40801. Trainable : 40801.

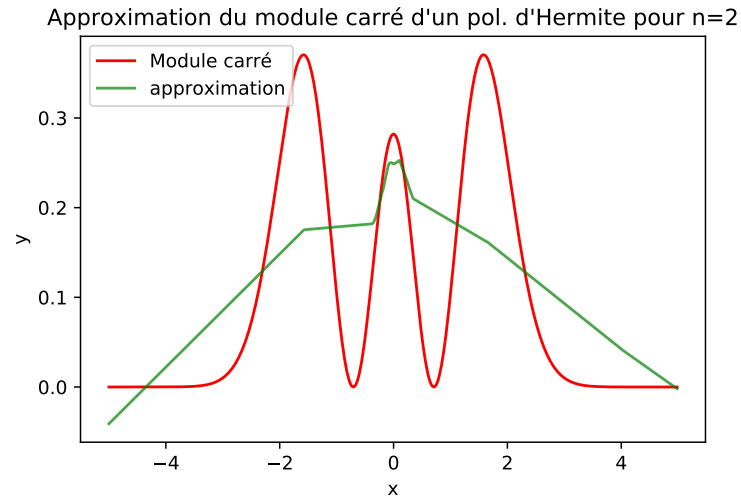


FIGURE 63 – $n=2$, 500 points. outputs : 20 20 1. Params : 481. Trainable : 481.

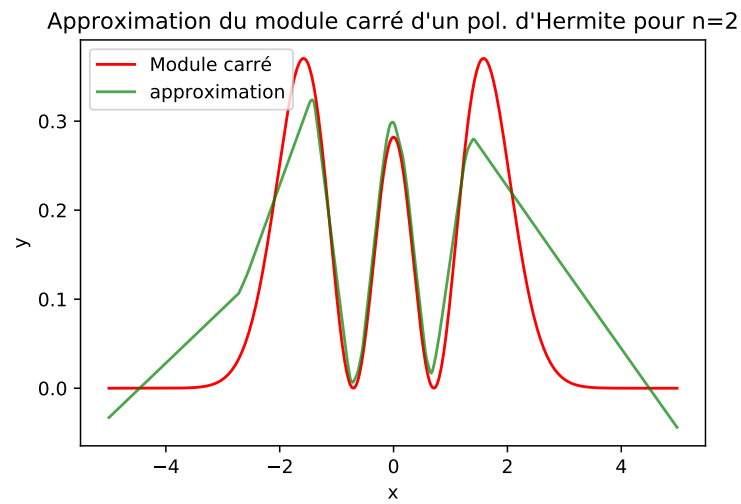


FIGURE 64 – $n=2$, 500 points. outputs : 200 200 1. Params : 40801. Trainable : 40801.

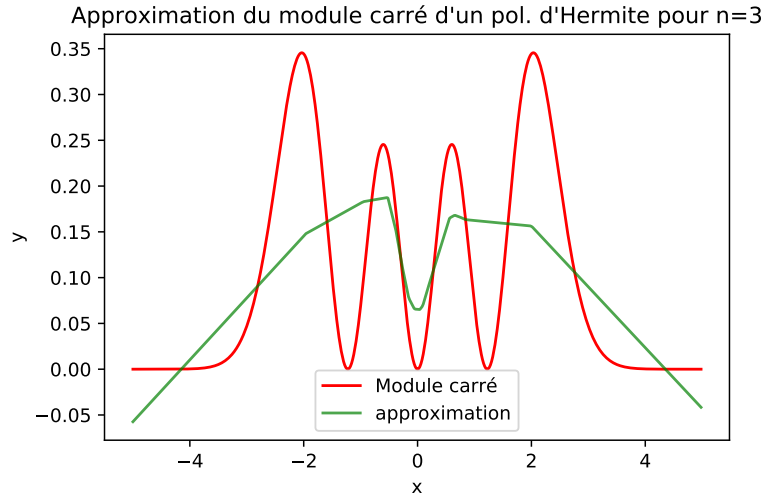


FIGURE 65 – $n=3$, 500 points. outputs : 20 20 1. Params : 481. Trainable : 481.

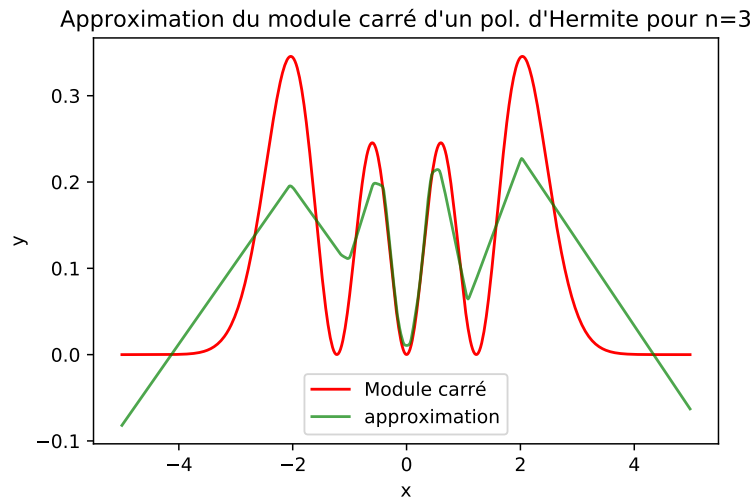


FIGURE 66 – $n=3$, 500 points. outputs : 200 200 1. Params : 40801. Trainable : 40801.

C Approximation des pol. d'Hermite par réseau de neurones (10001 pts)

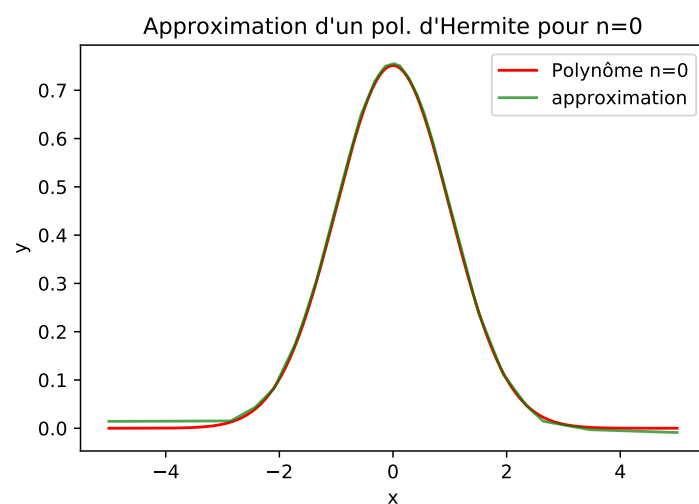


FIGURE 67 – $n=0$, 10001 pts. outputs : 20 20 1. Params : 481. Trainable : 481.

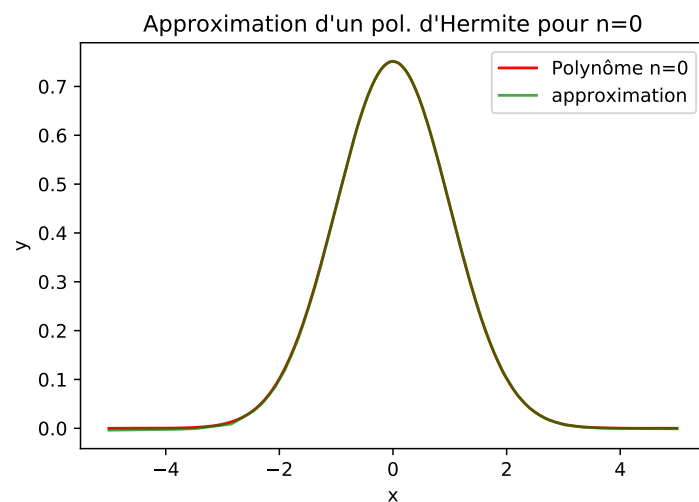


FIGURE 68 – $n=0$, 10001 pts. outputs : 200 200 1. Params : 40801. Trainable : 40801.

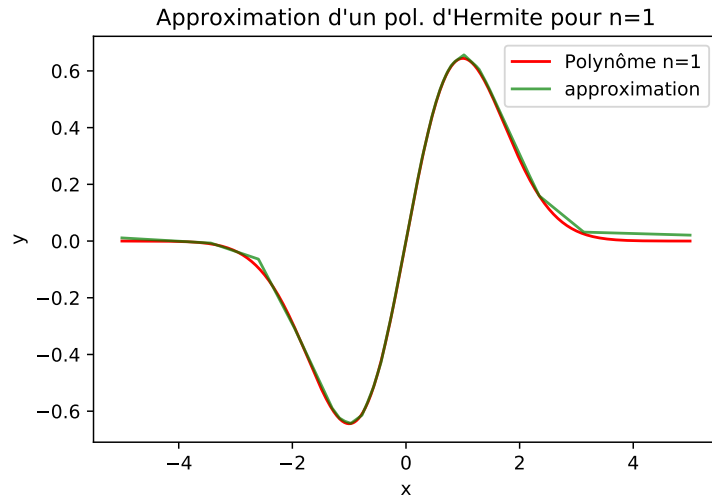


FIGURE 69 – $n=1$, 10001 pts. outputs : 20 20 1. Params : 481. Trainable : 481.

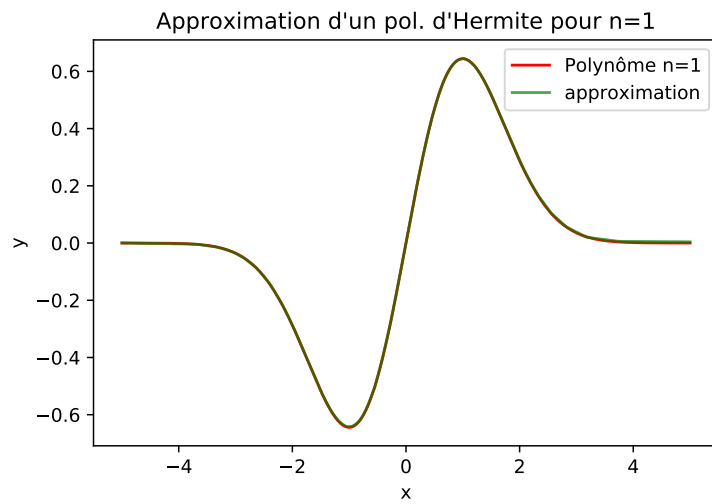


FIGURE 70 – $n=1$, 10001 pts. outputs : 200 200 1. Params : 40801. Trainable : 40801.

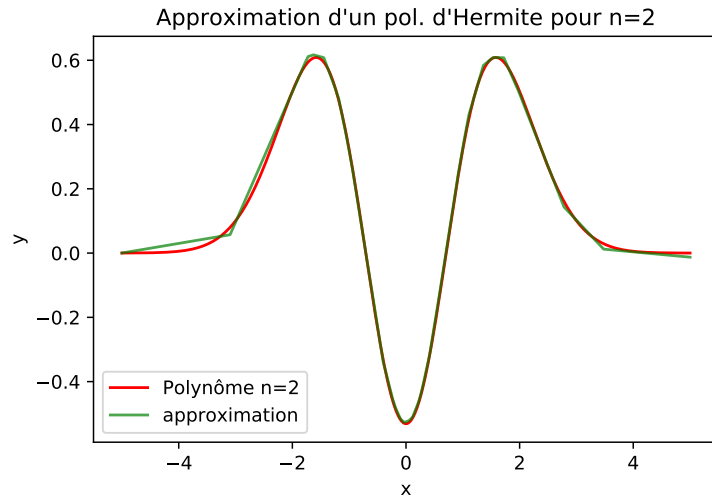


FIGURE 71 – $n=2$, 10001 pts. outputs : 20 20 1. Params : 481. Trainable : 481.

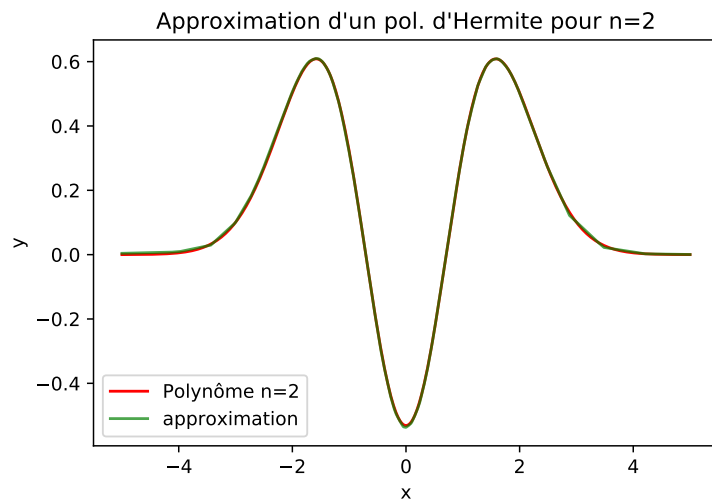


FIGURE 72 – $n=2$, 10001 pts. outputs : 200 200 1. Params : 40801. Trainable : 40801.

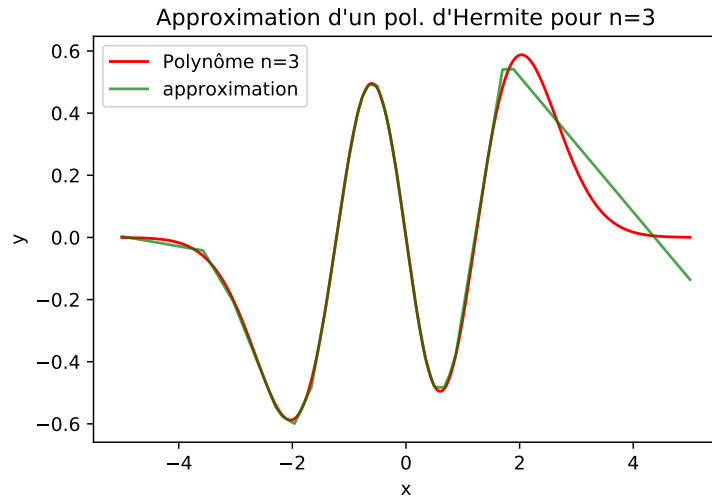


FIGURE 73 – $n=3$, 10001 pts. outputs : 20 20 1. Params : 481. Trainable : 481.

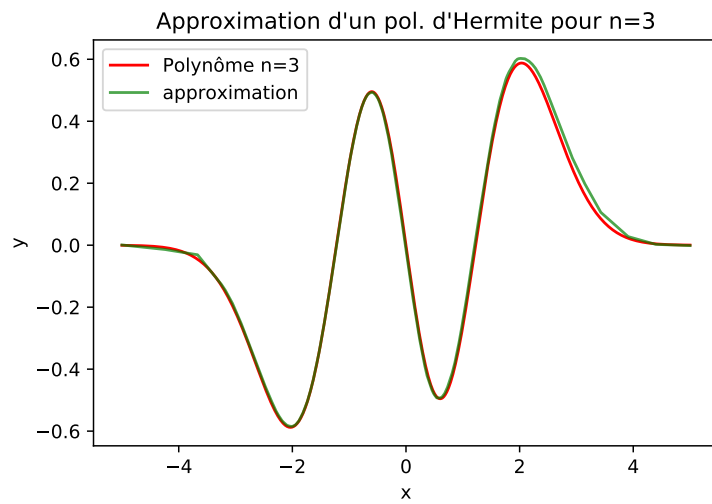


FIGURE 74 – $n=3$, 10001 pts. outputs : 200 200 1. Params : 40801. Trainable : 40801.

D Approximation des modules carrés des pol. d'Hermite par réseau de neurones (10001 pts)

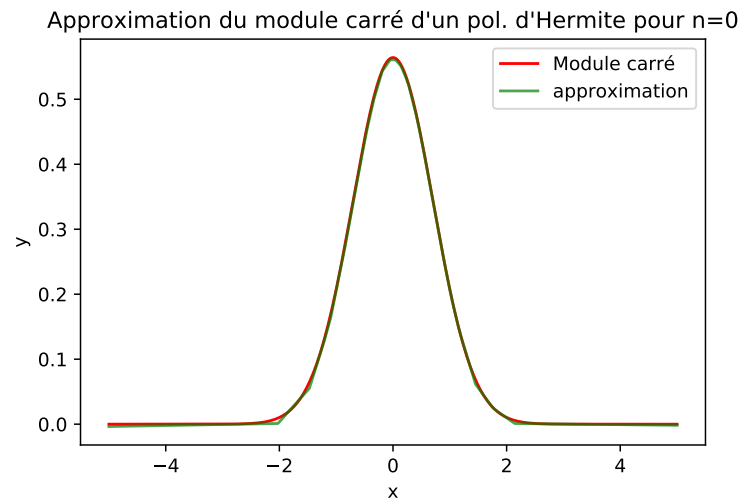


FIGURE 75 – $n=0$, 10001 pts. outputs : 20 20 1. Params : 481. Trainable : 481.

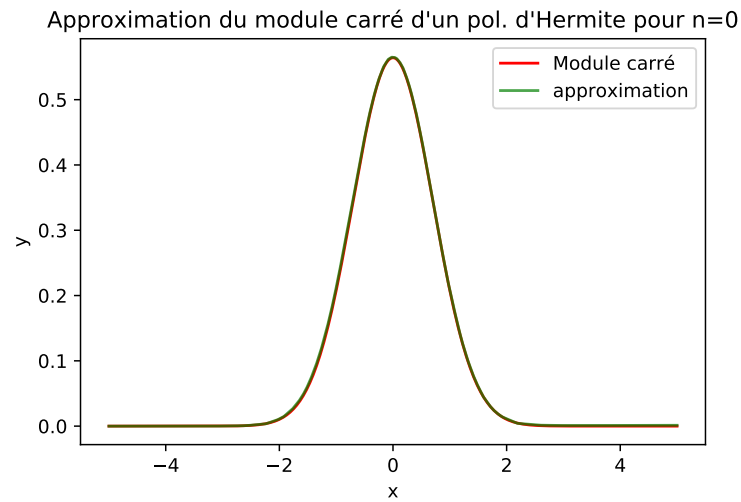


FIGURE 76 – $n=0$, 10001 pts. outputs : 200 200 1. Params : 40801. Trainable : 40801.

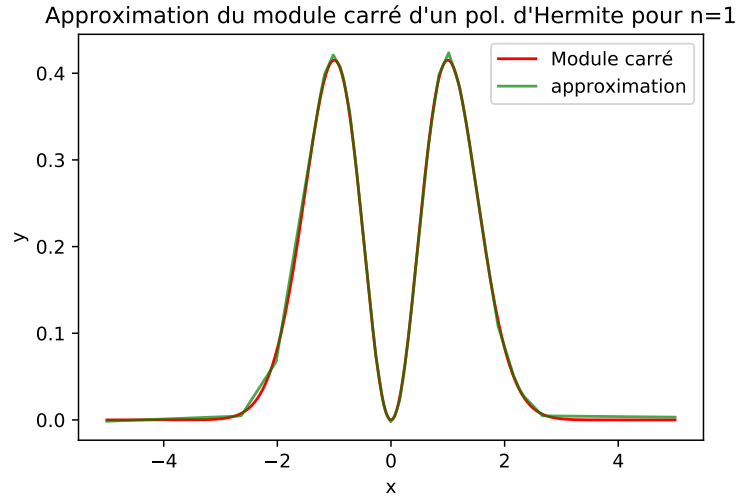


FIGURE 77 – $n=1$, 10001 pts. outputs : 20 20 1. Params : 481. Trainable : 481.

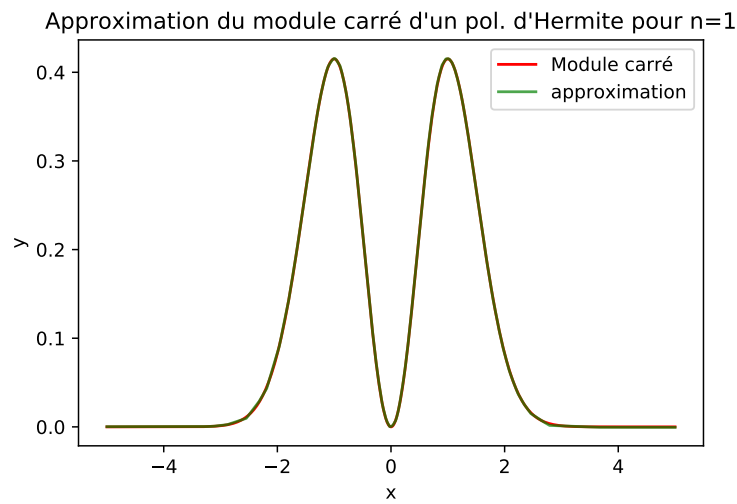


FIGURE 78 – $n=1$, 10001 pts. outputs : 200 200 1. Params : 40801. Trainable : 40801.

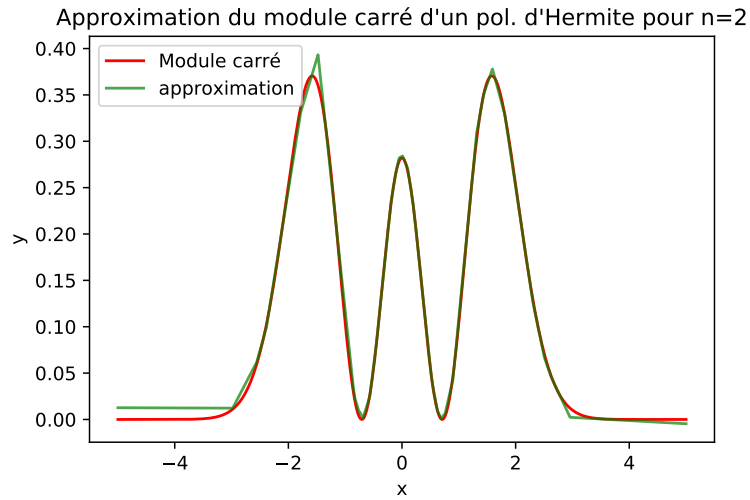


FIGURE 79 – $n=2$, 10001 pts. outputs : 20 20 1. Params : 481. Trainable : 481.

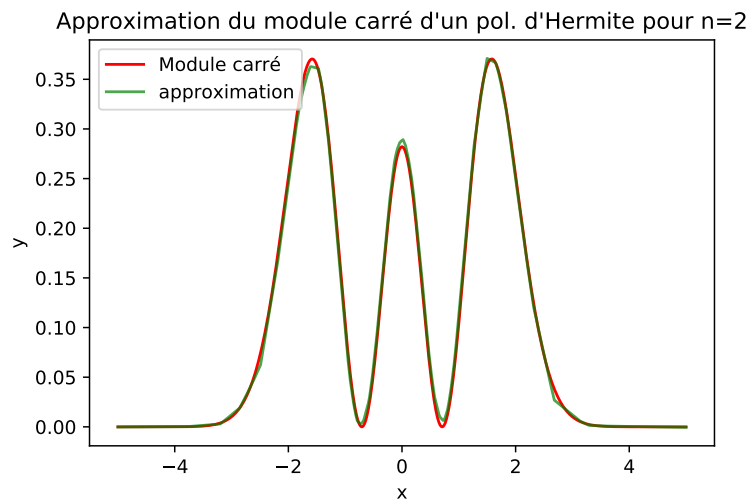


FIGURE 80 – $n=2$, 10001 pts. outputs : 200 200 1. Params : 40801. Trainable : 40801.

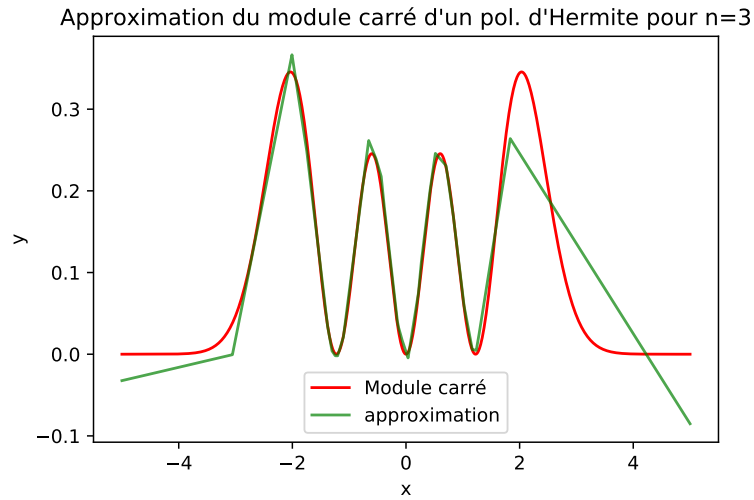


FIGURE 81 – $n=3$, 10001 pts. outputs : 20 20 1. Params : 481. Trainable : 481.

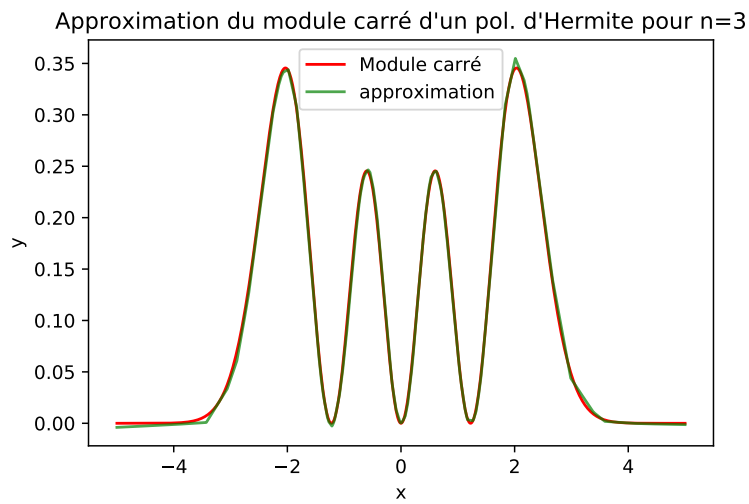


FIGURE 82 – $n=3$, 10001 pts. outputs : 200 200 1. Params : 40801. Trainable : 40801.