Minimisation de l'énergie

Clement Lotteau

June 2020

Table des matières

1 Minimisation sans fonction de coût

1.1 Premiers essais

Le but ici est de se servir des erreurs de fit pour sélectionner une prédiction comme nouvel objectif de fit si l'énergie de cette dernière est inférieure à la fonction à fitter. Le réseau "se fit lui-même". On note que le réseau est reconstruit à chaque fois qu'une nouvelle cible est choisie.

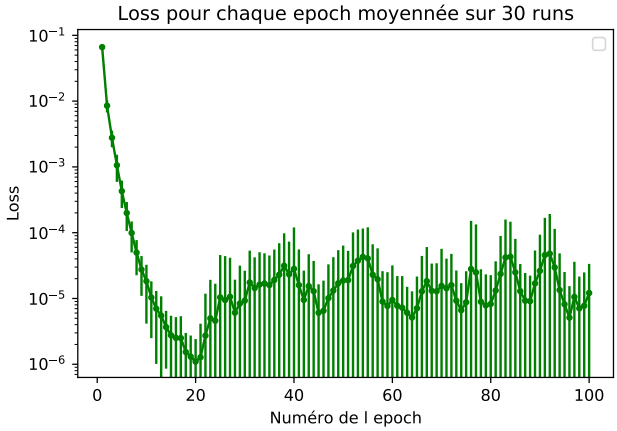

FIGURE 1 – Loss pour un réseau 200x200x200x200 (batch 50).

Je me sers des oscillations autour du minimum de la fonction de coût en 'croisant les doigts' pour que le réseau donne une prédiction ayant une énergie inférieure à celle de sa cible. Je remplace ensuite la cible par la prédiction. J'ai donc besoin d'un réseau qui minimise rapidement la fonction de coût. La précision de la minimisation de l'énergie dépend (à priori) de l'amplitude des oscillations autour du minimum. De grandes oscillations sont (à priori) bonnes pour une première approximation, et de petites oscillations permetteraient peut-être d'affiner la minimisation.

```
#INITIALISATION DU MODEL
model = models.Sequential([
  layers.Dense(200, input_shape=(1,), activation='relu'),
  layers.Dense(200, input_shape=(1,), activation='relu'),
  layers.Dense(1), # no activation -> linear function of the input
])
model.summary()
opt = optimizers.Adam(learning_rate=0.001)
model.compile(loss=min_loss,optimizer=opt)


for i in range(0,1001):

  #fit de l'onde
  model.fit(linx,onde,epochs=1,batch_size=50)
  predictions = model.predict(linx)
  preds = predictions.reshape(-1)


  #sélection de l'onde avec l'énergie la plus faible
  if (calcul_energie(linx,preds) < calcul_energie(linx,onde)):
    print('Energie onde = ',calcul_energie(linx,onde))
    print('Energie preds = ',calcul_energie(linx,preds))
    onde = preds

    #clear et recréation du modèle
    keras.backend.clear_session()
    model = models.Sequential([
      layers.Dense(200, input_shape=(1,), activation='relu'),
      layers.Dense(200, input_shape=(1,), activation='relu'),
      layers.Dense(1), # no activation -> linear function of the input
    ])
    model.summary()
    opt = optimizers.Adam(learning_rate=0.001)
    model.compile(loss=min_loss,optimizer=opt)
```

FIGURE 2 – premier code de la minimisation sans passer par la fonction de coût.

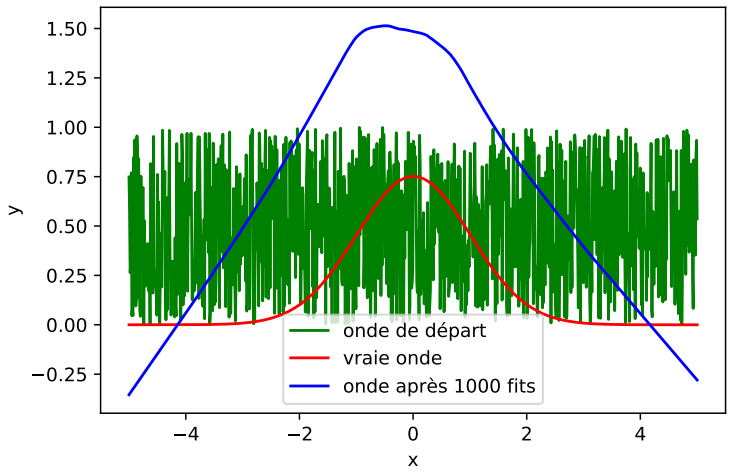

FIGURE 3 – Réseau 200x200.

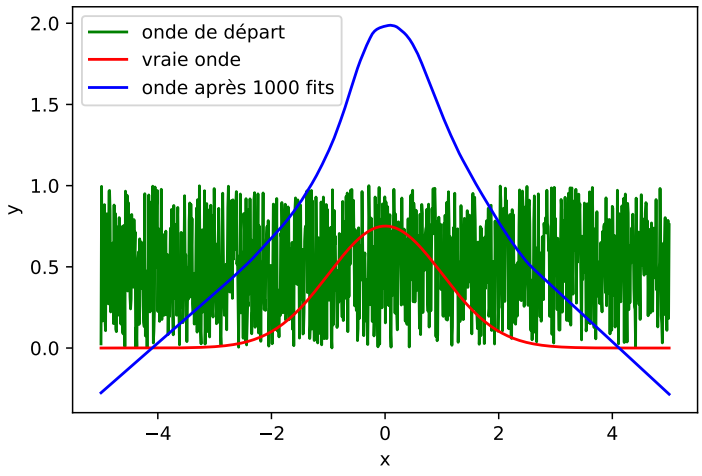

FIGURE 4 – Réseau 200x200x200.

Je dois imposer à la fonction d'onde d'être symétrique, toujours positive et normée à 1.

1.2 Prédiction et target du N.N. symétriques, positives et normées à 1

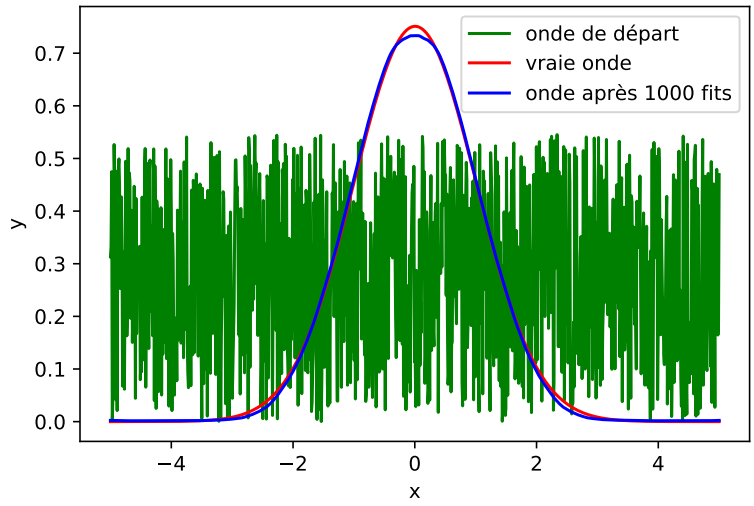

FIGURE 5 – 7 couches, 30671 paramètres. 71x71x71x71x71x70x70. Energie = 0.50258 (0.50001 à trouver).

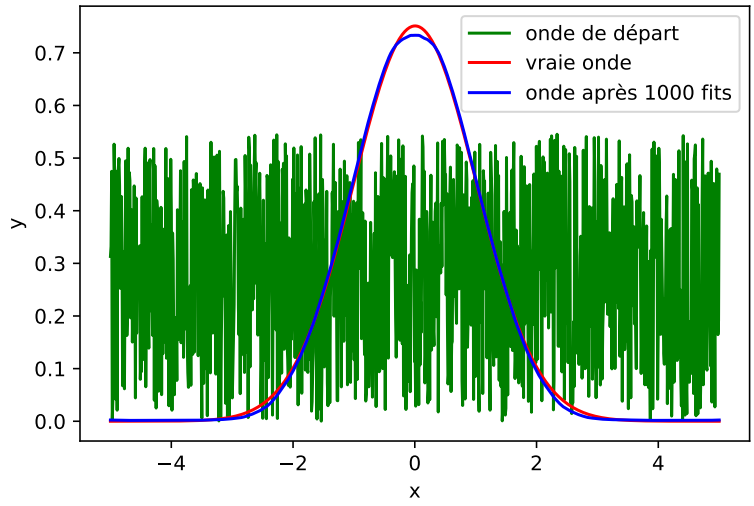

FIGURE 6 – 6 couches, 30574 paramètres. 78x78x78x77x77x76. Energie = 0.50200 (0.50001 à trouver)

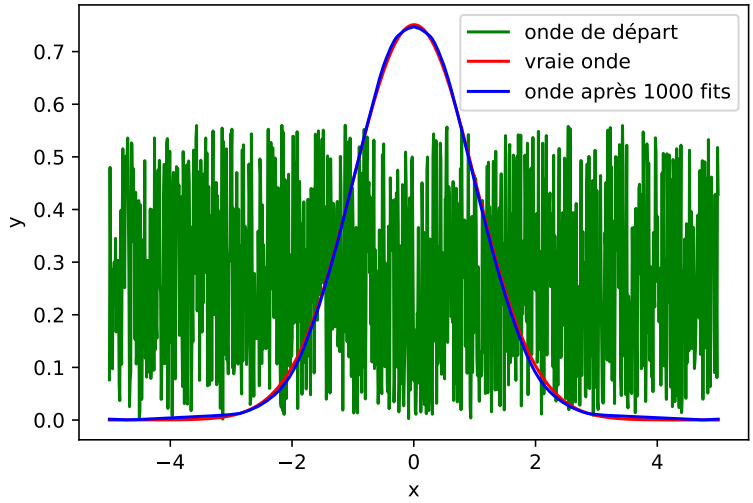

FIGURE 7 – 5 couches, 30623 paramètres. 87x87x87x86x86. Energie = 0.50215 (0.50001 à trouver)

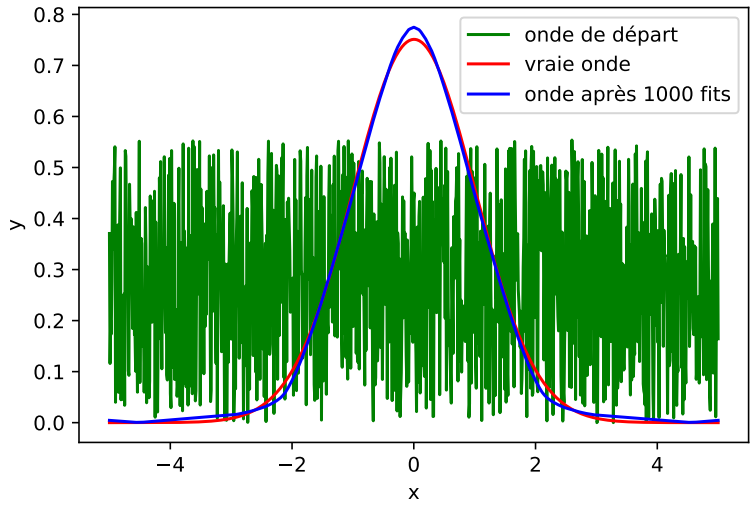

FIGURE 8 – 4 couches, 30601 paramètres. 100x100x100x100. Energie = 0.50572 (0.50001 à trouver)

1.3 Sans reconstruction au delà d'un seuil de l'énergie

La reconstruction du réseau sert à oublier les apprentissages précédents et à accélérer le passage d'une onde "plate" à une gaussienne. Beaucoup de temps est perdu à cause de cette reconstruction lorsque la fonction a déjà la forme d'une gaussienne. J'ai donc placé un seuil à une énergie de 0.6 en dessous de laquelle le réseau cesse de se reconstruire et se contente de fitter en boucle 1000 fois. Un seuil de 1 donnait des résultats moints concluants (la gaussienne était aplatie en haut).

J'ai aussi symétrisé et normalisé la fonction d'onde cible, ainsi que recalculé son énergie. Pareil pour la fonction d'onde aléatoire.

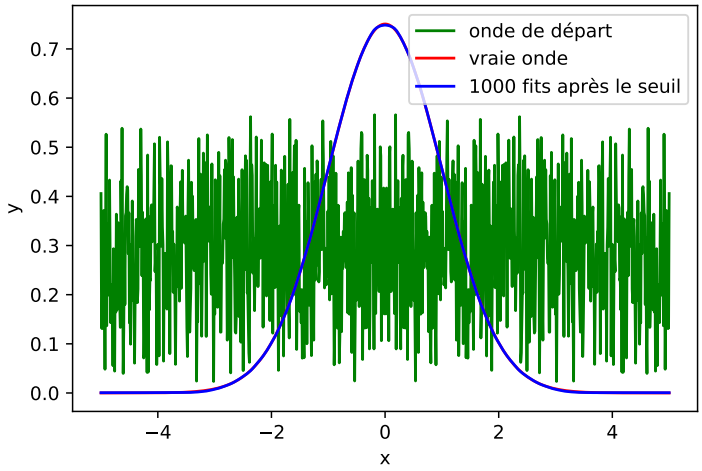

FIGURE 9 – 4 couches, 30601 paramètres. 100x100x100x100 (avant et après le seuil). Seuil à 0.6

Energie = 0.50033 (0.500001 à trouver). Je n'arrive pas à descendre vraiment plus en dessous de cette énergie.

1.4 100 points, sans reconstruction, sans seuil

J'ai abandonné la fonction d'onde aléatoire au profit d'une fonction d'onde constante et normalisée.

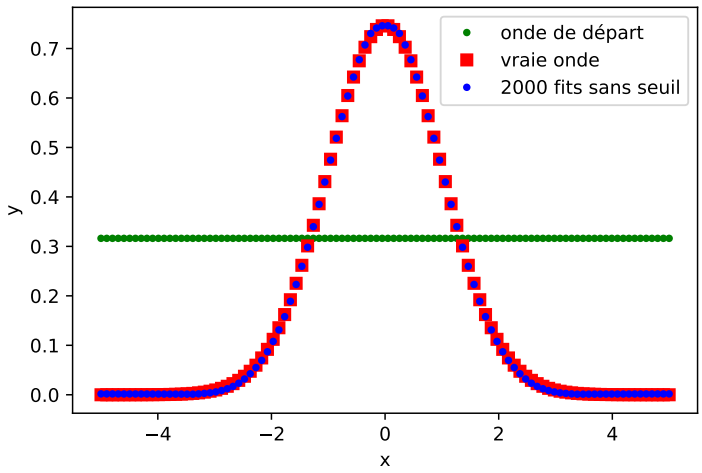

FIGURE 10 – 4 couches, 30601 paramètres. 100x100x100x100. 2000 fits.

Energie = 0.5005 (0.5001 à trouver). Temps de calcul : 18.3 secondes

Il serait intéressant de regarder l'écart en fonction de x. Il serait peut-être possible de minimiser cet écart avec des biais dans le réseau aux endroits où le fit est le moins bon (à vue d'oeil : souvent où la courbure est forte).

2 Energie en fonction du temps (10.000 fits)

2.1 Échelle linéaire

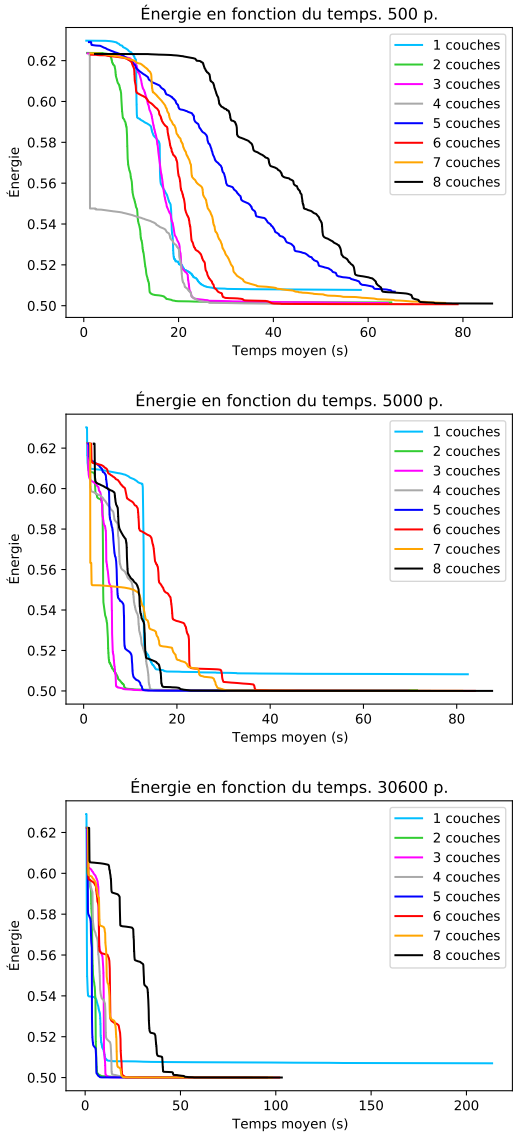

FIGURE 11 – Les réseaux à 1 couche ne convergent pas même après 10000 fits, ils sont donc à exclure. Le choix de la géométrie et du nombre de paramètres apparaissent important mais il est difficile de tirer une conclusion sans une étude plus fine.

2.2 Échelle log

Toutes les courbes ne commencent pas au même endroit car le temps de construction du réseau par l'ordinateur est pris en compte. Chaque courbe commence à la fin du premier fit et se termine à la fin du 10.000^e.

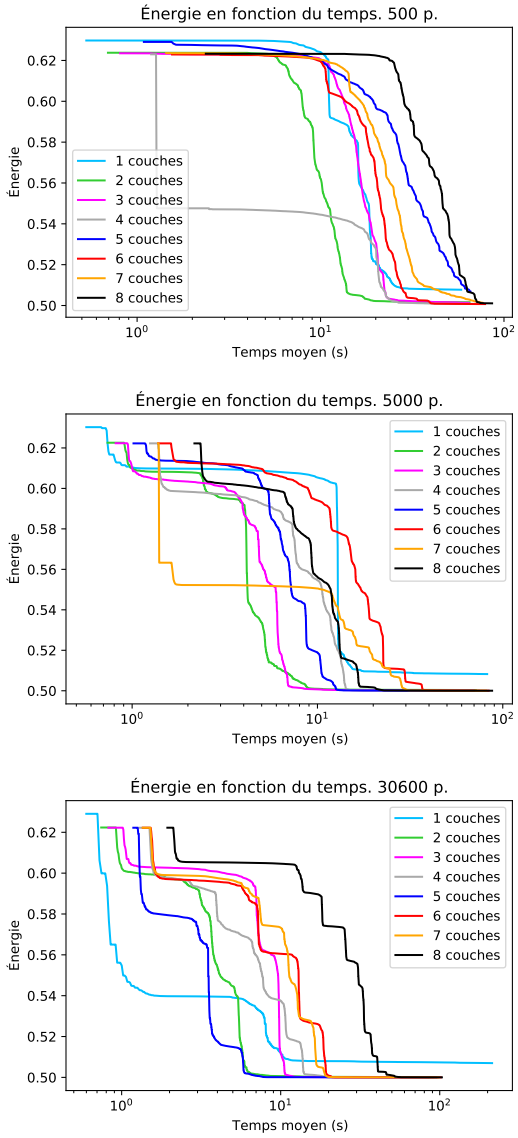

FIGURE 12 – Avec la figure précédente, on observe que la descente en paliers s'accentue lorsqu'on augmente le nombre de paramètres

3 Energie en fonction du temps (10.000 fits). Sans les réseaux à 1 couche.

3.1 Échelle linéaire

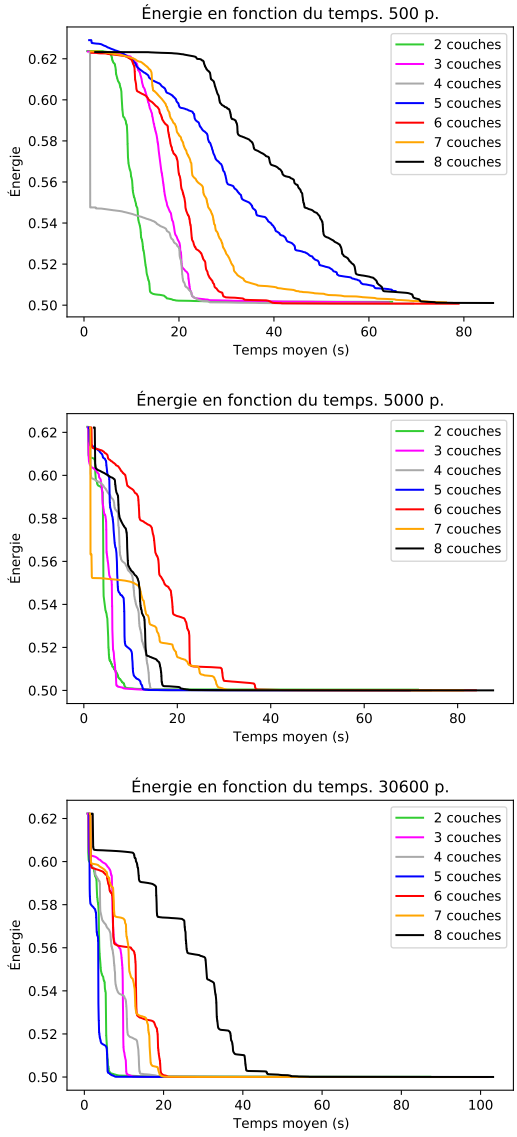

FIGURE 13 – On observe que le comportement de réseaux à géométries équivalentes varie en fonction du nombre de paramètres. Tous ne convergent pas dans le même ordre à première vue.

3.2 Échelle log

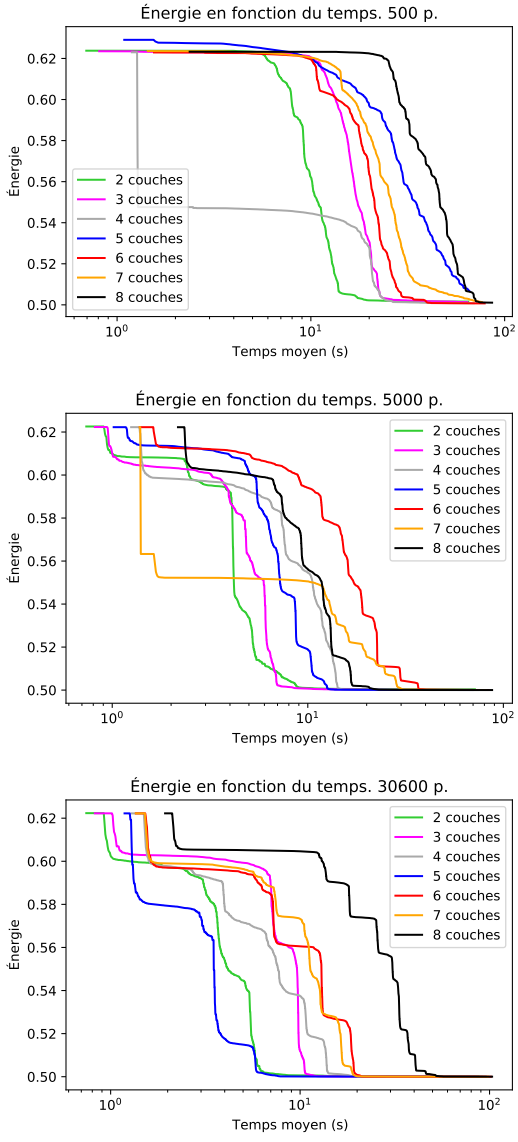

FIGURE 14 – Il m'est difficile de tirer une conclusion sur la meilleure géométrie ainsi que sur le meilleur nombre de paramètres sans une étude plus fine de la convergence.

4 Précision en fonction du temps de calcul

4.1 500 paramètres

Ici je fais tourner les réseaux sur 10000 fits et je mesure leur énergie ainsi que le temps de calcul à chaque fit. Les courbes s'arrêtent lorsque les réseaux ont fini leurs 10000 fits. Je moyenne sur 30 runs.

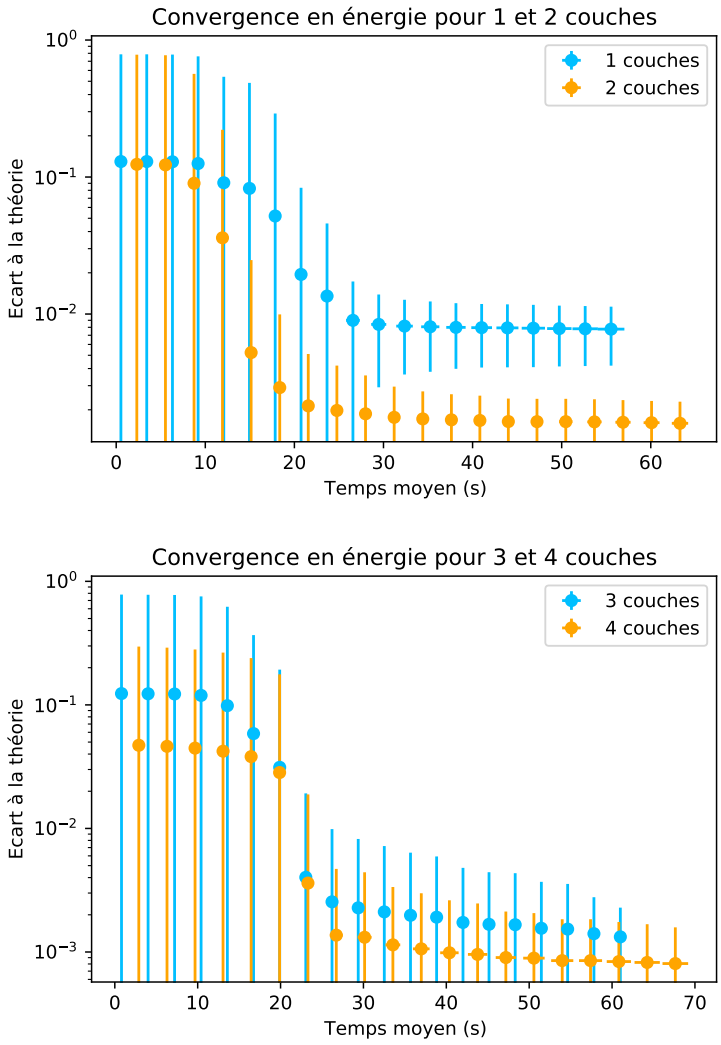

FIGURE 15 – Écart entre l'énergie du N.N. et la valeur théorique (0.5). Moyenne sur 30 runs. Environ 500 paramètres.

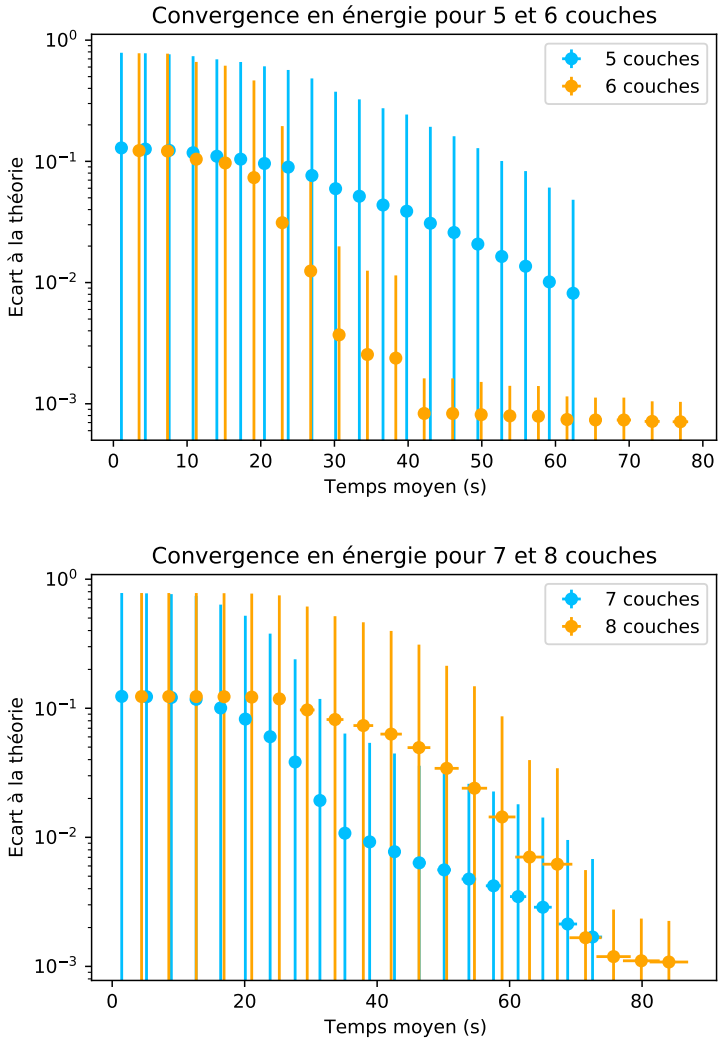

FIGURE 16 – Écart entre l'énergie du N.N. et la valeur théorique (0.5). Moyenne sur 30 runs. Environ 500 paramètres.

4.2 5000 paramètres

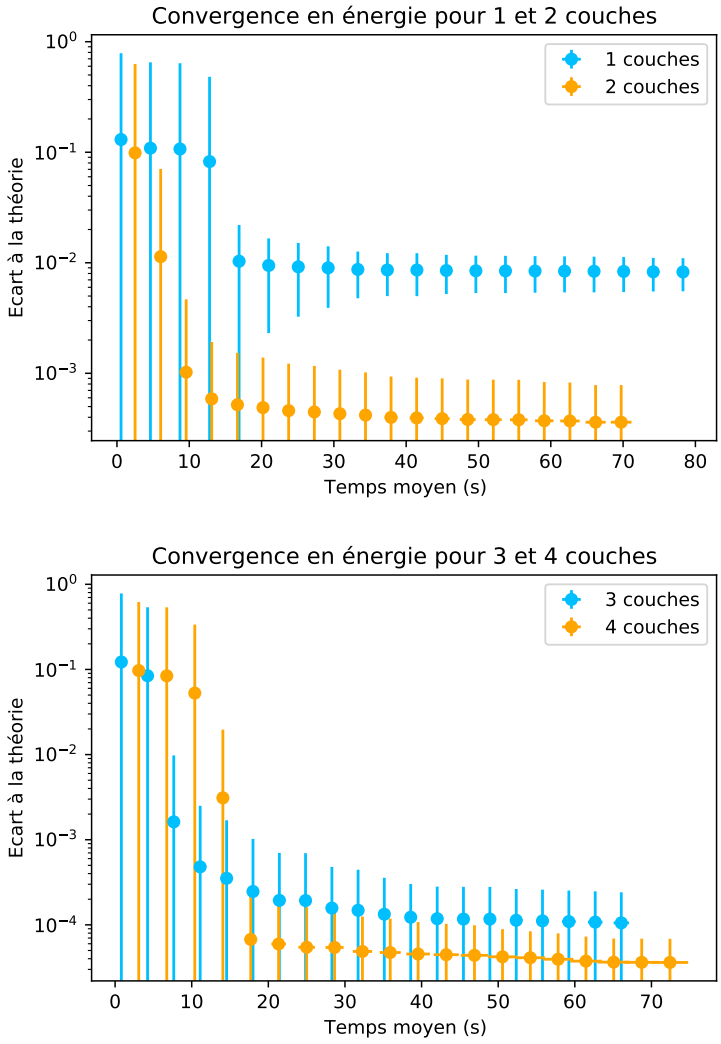

FIGURE 17 – Écart entre l'énergie du N.N. et la valeur théorique (0.5). Moyenne sur 30 runs. Environ 5000 paramètres.

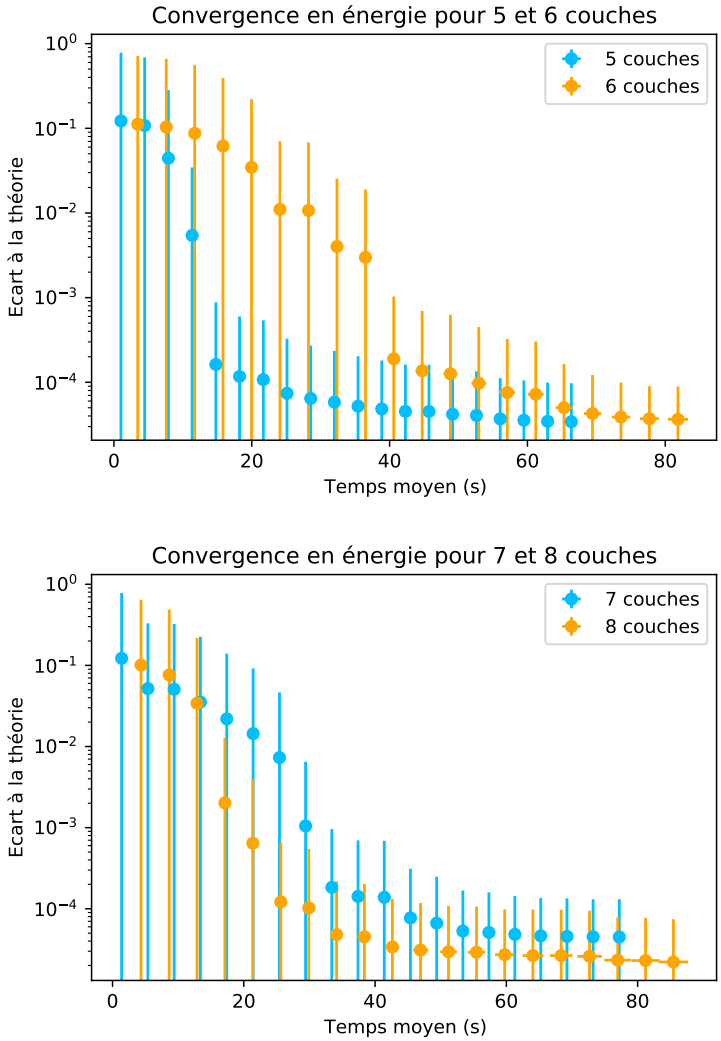

FIGURE 18 – Écart entre l'énergie du N.N. et la valeur théorique (0.5). Moyenne sur 30 runs. Environ 5000 paramètres.

4.3 30600 paramètres

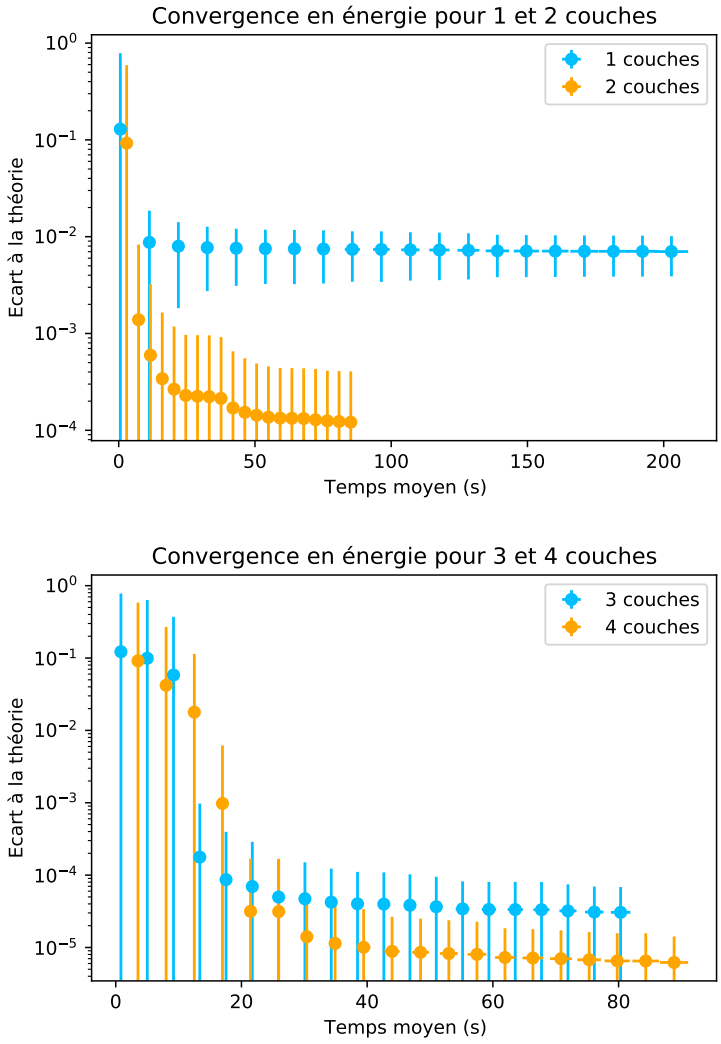

FIGURE 19 – Écart entre l'énergie du N.N. et la valeur théorique (0.5). Moyenne sur 30 runs. Environ 30600 paramètres.

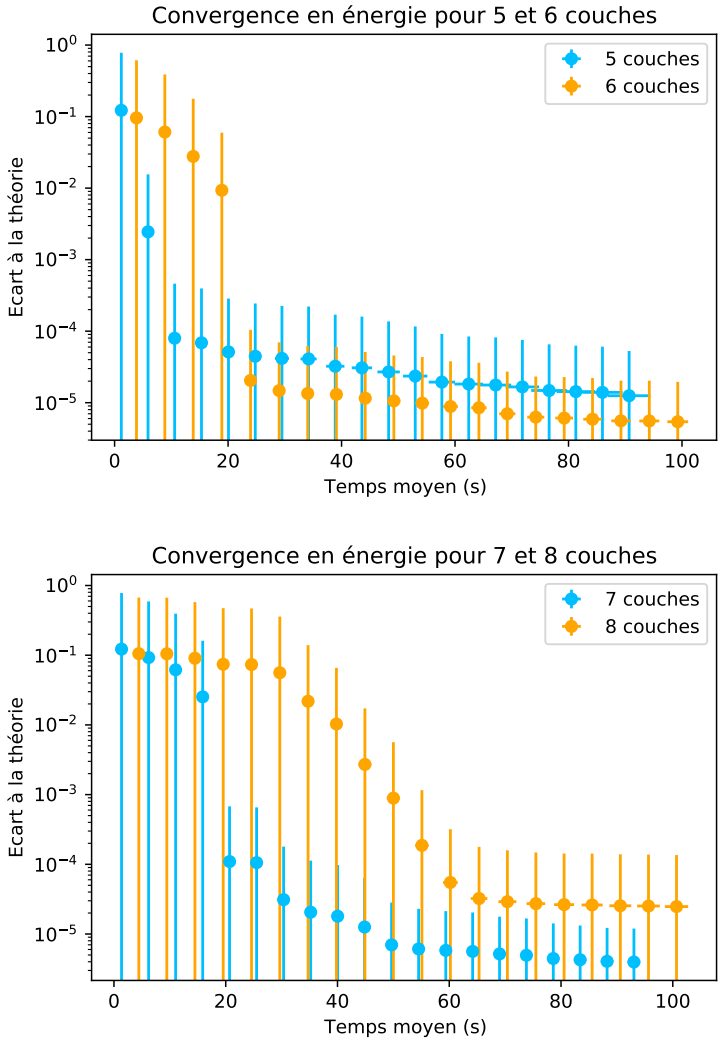

FIGURE 20 – Écart entre l'énergie du N.N. et la valeur théorique (0.5). Moyenne sur 30 runs. Environ 30600 paramètres.

5 Précision en fonction du temps : géométries équivalentes, comparaison du nombre de paramètres

Ici aussi le nombre de fits est de 10000 et on moyenne sur 30 runs.

5.1 1 couche

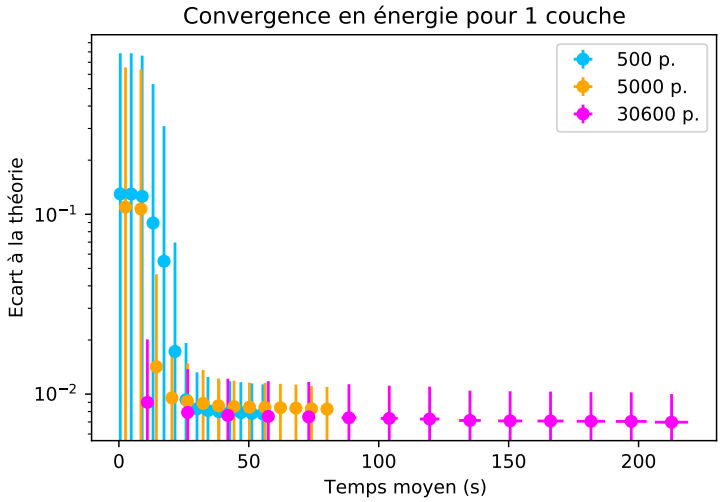

FIGURE 21 – Écart entre l'énergie du N.N. et la valeur théorique (0.5). Moyenne sur 30 runs.

5.2 2 couches

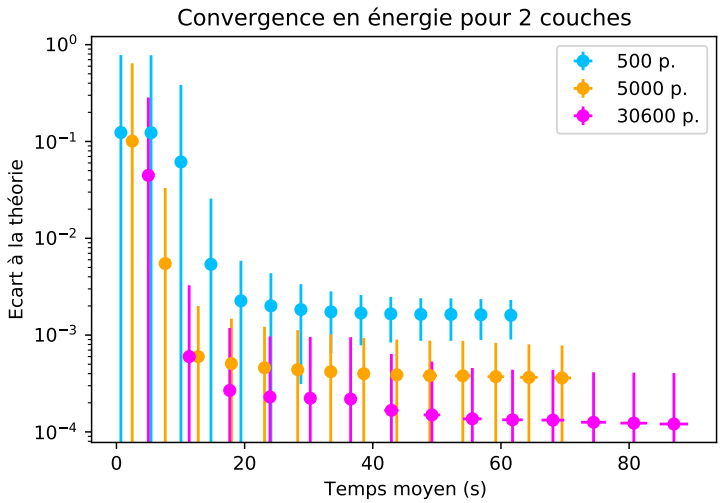

FIGURE 22 – Écart entre l'énergie du N.N. et la valeur théorique (0.5). Moyenne sur 30 runs.

5.3 3 couches

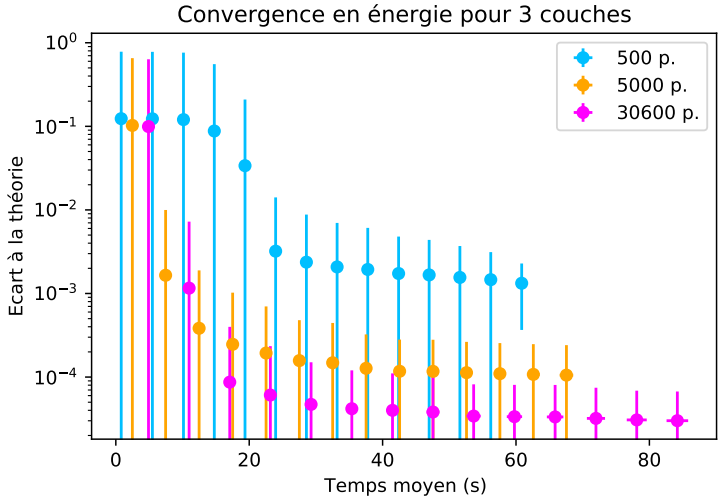

FIGURE 23 – Écart entre l'énergie du N.N. et la valeur théorique (0.5). Moyenne sur 30 runs.

5.4 4 couches

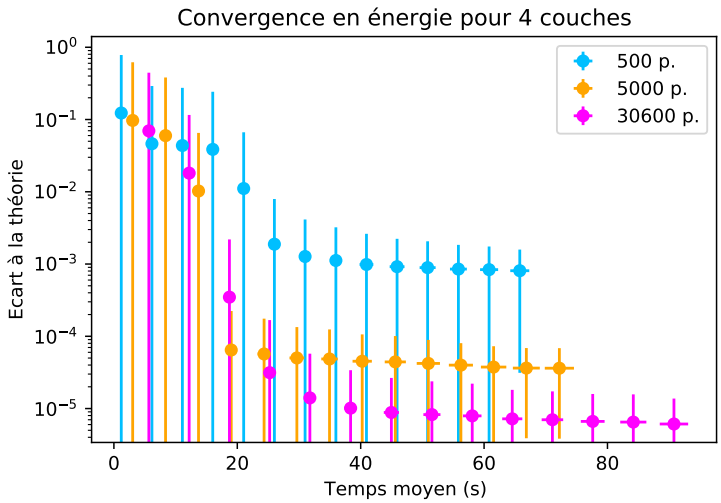

FIGURE 24 – Écart entre l'énergie du N.N. et la valeur théorique (0.5). Moyenne sur 30 runs.

5.5 5 couches

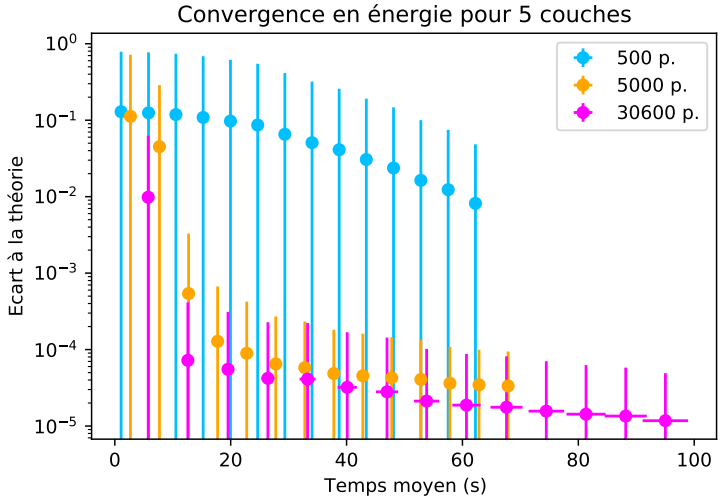

FIGURE 25 – Écart entre l'énergie du N.N. et la valeur théorique (0.5). Moyenne sur 30 runs.

5.6 6 couches

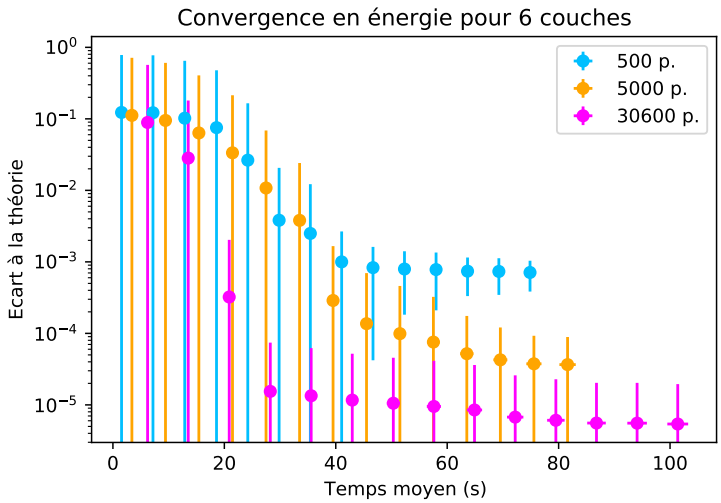

FIGURE 26 – Écart entre l'énergie du N.N. et la valeur théorique (0.5). Moyenne sur 30 runs.

5.7 7 couches

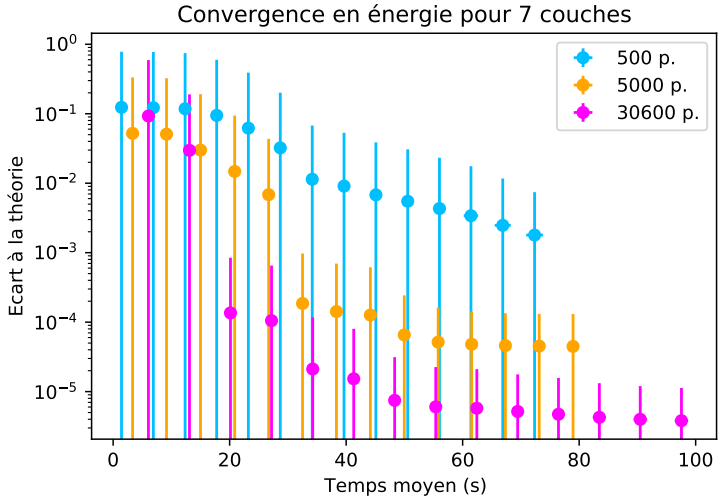

FIGURE 27 – Écart entre l'énergie du N.N. et la valeur théorique (0.5). Moyenne sur 30 runs.

5.8 8 couches

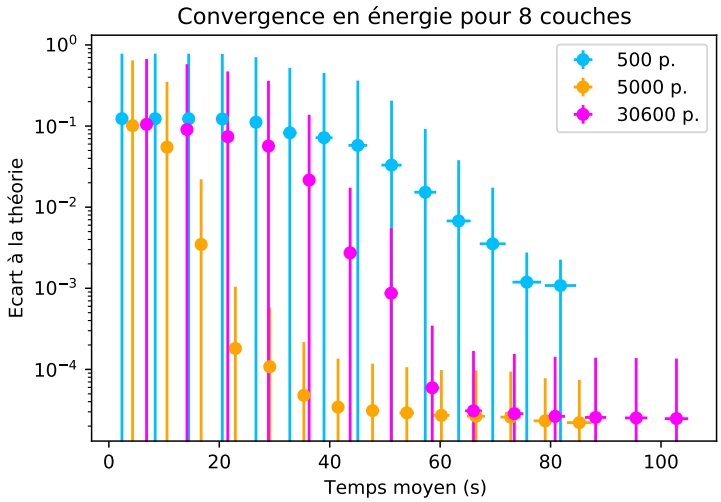

FIGURE 28 – Écart entre l'énergie du N.N. et la valeur théorique (0.5). Moyenne sur 30 runs.

5.9 Temps moyen pour 10.000 fits en fonction du nombre de couches

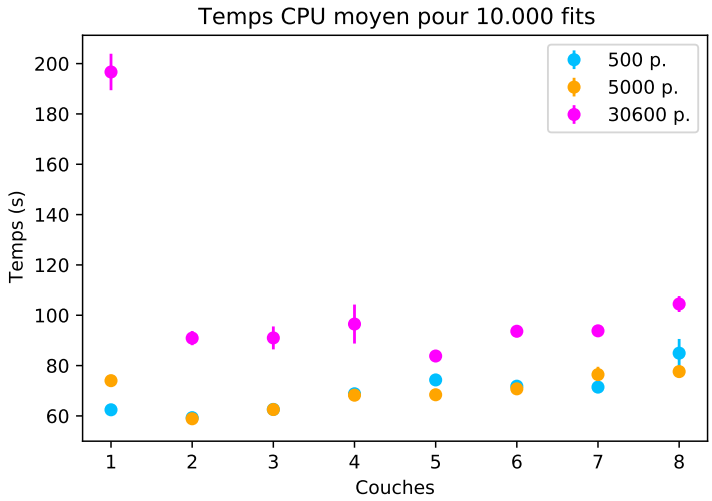

FIGURE 29 – Temps pour 10.000 fits, mesuré 30 fois puis moyenné.