Techniques de Machine Learning dans l'amélioration d'intégrateurs numériques pour les équations différentielles stochastiques.

Dorian Coudiere sous la supervision d'Adrien Laurent

June 4, 2024

Sommaire

Equation différentielle stochastique

EDS:

$$dX = f(X)dt + \sigma(X)dW(t) \quad X(0) = x \in \mathbb{R}^d$$

Equation différentielle stochastique

EDS:

$$dX = f(X)dt + \sigma(X)dW(t) \quad X(0) = x \in \mathbb{R}^d$$

Flot exact : $Y(h) = \varphi_h(Y_0)$

Equation différentielle stochastique

EDS:

$$dX = f(X)dt + \sigma(X)dW(t) \quad X(0) = x \in \mathbb{R}^d$$

Flot exact : $Y(h) = \varphi_h(Y_0)$

Intégrateur : $Y_{n+1} = \phi_h^{f,\sigma}(Y_n)$

Equation différentielle stochastique

EDS:

$$dX = f(X)dt + \sigma(X)dW(t) \quad X(0) = x \in \mathbb{R}^d$$

Flot exact : $Y(h) = \varphi_h(Y_0)$

Intégrateur : $Y_{n+1} = \phi_h^{f,\sigma}(Y_n)$

Euler-Maruyama : $Y_{n+1} = Y_n + hf(Y_n) + \sqrt{h}\sigma(Y_n)\xi \quad \xi \sim N(0,1)$

Equation différentielle stochastique

EDS:

$$dX = f(X)dt + \sigma(X)dW(t) \quad X(0) = x \in \mathbb{R}^d$$

Flot exact : $Y(h) = \varphi_h(Y_0)$

Intégrateur : $Y_{n+1} = \phi_h^{f,\sigma}(Y_n)$

Euler-Maruyama : $Y_{n+1} = Y_n + hf(Y_n) + \sqrt{h}\sigma(Y_n)\xi \quad \xi \sim N(0,1)$

EDS Modifiée :

$$dX = \tilde{f}(X)dt + \tilde{\sigma}(X)dW(t) \quad X(0) = x \in \mathbb{R}^d$$

Equation différentielle stochastique

EDS:

$$dX = f(X)dt + \sigma(X)dW(t) \quad X(0) = x \in \mathbb{R}^d$$

Flot exact : $Y(h) = \varphi_h(Y_0)$

Intégrateur : $Y_{n+1} = \phi_h^{f,\sigma}(Y_n)$

Euler-Maruyama : $Y_{n+1} = Y_n + hf(Y_n) + \sqrt{h}\sigma(Y_n)\xi \quad \xi \sim N(0,1)$

EDS Modifiée :

$$dX = \tilde{f}(X)dt + \tilde{\sigma}(X)dW(t) \quad X(0) = x \in \mathbb{R}^d$$

$$X_{n+1} = \varphi_h(X_n) = \phi_h^{\tilde{f},\tilde{\sigma}}(X_n)$$

Equation différentielle stochastique

EDS:

$$dX = f(X)dt + \sigma(X)dW(t) \quad X(0) = x \in \mathbb{R}^d$$

Flot exact : $Y(h) = \varphi_h(Y_0)$

Intégrateur : $Y_{n+1} = \phi_h^{f,\sigma}(Y_n)$

Euler-Maruyama : $Y_{n+1} = Y_n + hf(Y_n) + \sqrt{h}\sigma(Y_n)\xi \quad \xi \sim N(0,1)$

EDS Modifiée :

$$dX = \tilde{f}(X)dt + \tilde{\sigma}(X)dW(t) \quad X(0) = x \in \mathbb{R}^d$$

$$X_{n+1} = \varphi_h(X_n) = \phi_h^{\tilde{f},\tilde{\sigma}}(X_n)$$

Proposition

On peut écrire les champs modifiés comme des B-séries

Equation différentielle stochastique

EDS:

$$dX = f(X)dt + \sigma(X)dW(t) \quad X(0) = x \in \mathbb{R}^d$$

Flot exact : $Y(h) = \varphi_h(Y_0)$

Intégrateur : $Y_{n+1} = \phi_h^{f,\sigma}(Y_n)$

Euler-Maruyama : $Y_{n+1} = Y_n + hf(Y_n) + \sqrt{h}\sigma(Y_n)\xi \quad \xi \sim N(0,1)$

EDS Modifiée :

$$dX = \tilde{f}(X)dt + \tilde{\sigma}(X)dW(t) \quad X(0) = x \in \mathbb{R}^d$$

$$X_{n+1} = \varphi_h(X_n) = \phi_h^{\tilde{f},\tilde{\sigma}}(X_n)$$

Proposition

On peut écrire les champs modifiés comme des B-séries

- $\tilde{f}_h(x) = f(x) + \sum_{i=1}^{\infty} h^i f_i(x)$
- $\tilde{\sigma}_h(x) = \sigma(x) + \sum_{i=1}^{\infty} h^i \sigma_i(x)$

EDS Linéaire

EDS Linéaire :

$$dX = \lambda X dt + \mu X dW \quad \lambda \in \mathbb{R} \quad \mu \in \mathbb{R} \quad X(0) = x$$

EDS Linéaire

EDS Linéaire :

$$dX = \lambda X dt + \mu X dW \quad \lambda \in \mathbb{R} \quad \mu \in \mathbb{R} \quad X(0) = x$$

Application de l'intégrateur numérique modifié :

EDS Linéaire

EDS Linéaire :

$$dX = \lambda X dt + \mu X dW \quad \lambda \in \mathbb{R} \quad \mu \in \mathbb{R} \quad X(0) = x$$

Application de l'intégrateur numérique modifié :

Euler Maruyama : $X(h) = x + h\tilde{f}(x) + \sqrt{h\tilde{\sigma^2}(x)}\xi \quad \xi \sim N(0,1)$

EDS Linéaire

EDS Linéaire :

$$dX = \lambda X dt + \mu X dW \quad \lambda \in \mathbb{R} \quad \mu \in \mathbb{R} \quad X(0) = x$$

Application de l'intégrateur numérique modifié :

Euler Maruyama : $X(h) = x + h\tilde{f}(x) + \sqrt{h\tilde{\sigma^2}(x)}\xi \quad \xi \sim N(0,1)$

- $E[X(h)] = x + h\tilde{f}(x) = x + hf(x) + h^2 f_1(x) + h^3 f_2(x) + O(h^4)$

EDS Linéaire

EDS Linéaire :

$$dX = \lambda X dt + \mu X dW \quad \lambda \in \mathbb{R} \quad \mu \in \mathbb{R} \quad X(0) = x$$

Application de l'intégrateur numérique modifié :

Euler Maruyama : $X(h) = x + h\tilde{f}(x) + \sqrt{h\tilde{\sigma^2}(x)}\xi \quad \xi \sim N(0,1)$

- $E[X(h)] = x + h\tilde{f}(x) = x + hf(x) + h^2 f_1(x) + h^3 f_2(x) + O(h^4)$
- $Var[X(h)] = h\tilde{\sigma^2}(x) = h\sigma^2(x) + h^2\sigma_1^2(x) + h^3\sigma_2^2(x) + O(h^4)$

EDS Linéaire

EDS Linéaire :

$$dX = \lambda X dt + \mu X dW \quad \lambda \in \mathbb{R} \quad \mu \in \mathbb{R} \quad X(0) = x$$

Application de l'intégrateur numérique modifié :

Euler Maruyama : $X(h) = x + h\tilde{f}(x) + \sqrt{h\tilde{\sigma^2}(x)}\xi \quad \xi \sim N(0,1)$

- $E[X(h)] = x + h\tilde{f}(x) = x + hf(x) + h^2 f_1(x) + h^3 f_2(x) + O(h^4)$
- $Var[X(h)] = h\tilde{\sigma^2}(x) = h\sigma^2(x) + h^2\sigma_1^2(x) + h^3\sigma_2^2(x) + O(h^4)$

Solution exact :

$$X(t) = e^{(\lambda - \frac{\mu^2}{2})t + \mu W(t)} x$$

EDS Linéaire

EDS Linéaire :

$$dX = \lambda X dt + \mu X dW \quad \lambda \in \mathbb{R} \quad \mu \in \mathbb{R} \quad X(0) = x$$

Application de l'intégrateur numérique modifié :

Euler Maruyama : $X(h) = x + h\tilde{f}(x) + \sqrt{h\tilde{\sigma^2}(x)}\xi \quad \xi \sim N(0,1)$

- $E[X(h)] = x + h\tilde{f}(x) = x + hf(x) + h^2 f_1(x) + h^3 f_2(x) + O(h^4)$
- $Var[X(h)] = h\tilde{\sigma^2}(x) = h\sigma^2(x) + h^2\sigma_1^2(x) + h^3\sigma_2^2(x) + O(h^4)$

Solution exact :

$$X(t) = e^{(\lambda - \frac{\mu^2}{2})t + \mu W(t)} x$$

- $E[X(t)] = e^{\lambda t} x$
- $Var[X(t)] = e^{2\lambda t}(e^{\mu^2 t} - 1)x^2$

EDS Linéaire

EDS Linéaire :

$$dX = \lambda X dt + \mu X dW \quad \lambda \in \mathbb{R} \quad \mu \in \mathbb{R} \quad X(0) = x$$

Application de l'intégrateur numérique modifié :

Euler Maruyama : $X(h) = x + h\tilde{f}(x) + \sqrt{h\tilde{\sigma}^2(x)}\xi \quad \xi \sim N(0,1)$

- $E[X(h)] = x + h\tilde{f}(x) = x + hf(x) + h^2 f_1(x) + h^3 f_2(x) + O(h^4)$
- $Var[X(h)] = h\tilde{\sigma}^2(x) = h\sigma^2(x) + h^2\sigma_1^2(x) + h^3\sigma_2^2(x) + O(h^4)$

Solution exact :

$$X(t) = e^{(\lambda - \frac{\mu^2}{2})t + \mu W(t)} x$$

- $E[X(t)] = e^{\lambda t} x$
- $Var[X(t)] = e^{2\lambda t}(e^{\mu^2 t} - 1)x^2$

Fonctions modifiés :

- $f_1(x) = \frac{\lambda^2}{2}x$
- $f_2(x) = \frac{\lambda^3}{6}x$

EDS Linéaire

EDS Linéaire :

$$dX = \lambda X dt + \mu X dW \quad \lambda \in \mathbb{R} \quad \mu \in \mathbb{R} \quad X(0) = x$$

Application de l'intégrateur numérique modifié :

Euler Maruyama : $X(h) = x + h\tilde{f}(x) + \sqrt{h\tilde{\sigma^2}(x)}\xi \quad \xi \sim N(0,1)$

- $E[X(h)] = x + h\tilde{f}(x) = x + hf(x) + h^2 f_1(x) + h^3 f_2(x) + O(h^4)$
- $Var[X(h)] = h\tilde{\sigma^2}(x) = h\sigma^2(x) + h^2\sigma_1^2(x) + h^3\sigma_2^2(x) + O(h^4)$

Solution exact :

$$X(t) = e^{(\lambda - \frac{\mu^2}{2})t + \mu W(t)} x$$

- $E[X(t)] = e^{\lambda t} x$
- $Var[X(t)] = e^{2\lambda t}(e^{\mu^2 t} - 1)x^2$

Fonctions modifiés :

- $f_1(x) = \frac{\lambda^2}{2}x$
- $f_2(x) = \frac{\lambda^3}{6}x$
- $\sigma_1^2(x) = (\frac{\mu^4}{2} + 2\lambda\mu^2)x^2$
- $\sigma_2^2(x) = (\frac{\mu^6}{6} + \lambda\mu^4 + 2\lambda^2\mu^2)x^2$

Sommaire

Création du Réseau de neuronnes

$$y_1 = \varphi_h^{f,\sigma}(y_0) = \phi_h^{\tilde{f},\tilde{\sigma}} \rightarrow \phi_h^{f_{app},\sigma_{app}}$$

Création du Réseau de neuronnes

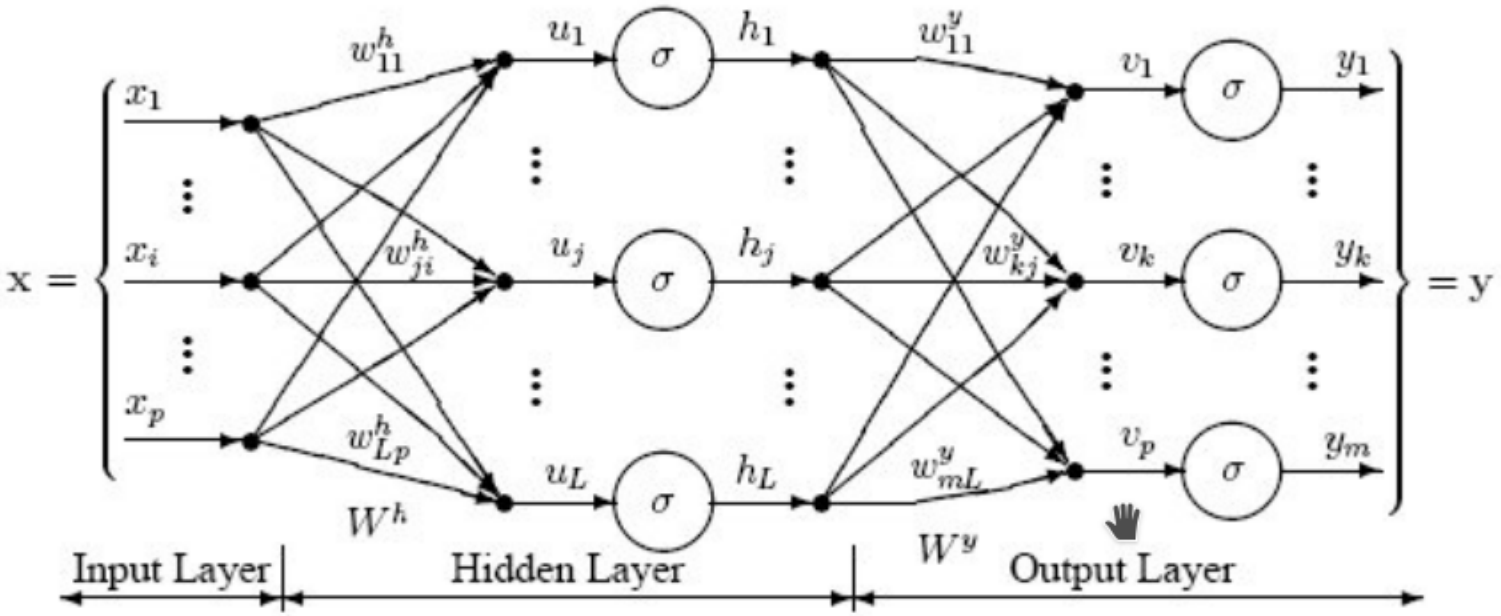
$$y_1 = \varphi_h^{f,\sigma}(y_0) = \phi_h^{\tilde{f},\tilde{\sigma}} \rightarrow \phi_h^{f_{app},\sigma_{app}}$$

- $\tilde{f}_h(y) = f(y) + \sum_{i=1}^{n} h^i f_i(y) \rightarrow f_{app}(y) = f(y) + \sum_{i=1}^{n} h^i MLP_{1,i}(y) + R_1$
- $\tilde{\sigma}_h(y) = \sigma(y) + \sum_{i=1}^{n} h^i \sigma_i(y) \rightarrow \sigma_{app}(y) = \sigma(y) + \sum_{i=1}^{n} h^i MLP_{2,i}(y) + R_2$

Création du Réseau de neuronnes

$$y_1 = \varphi_h^{f,\sigma}(y_0) = \phi_h^{\tilde{f},\tilde{\sigma}} \rightarrow \phi_h^{f_{app},\sigma_{app}}$$

- $\tilde{f}_h(y) = f(y) + \sum_{i=1}^{n} h^i f_i(y) \rightarrow f_{app}(y) = f(y) + \sum_{i=1}^{n} h^i MLP_{1,i}(y) + R_1$
- $\tilde{\sigma}_h(y) = \sigma(y) + \sum_{i=1}^{n} h^i \sigma_i(y) \rightarrow \sigma_{app}(y) = \sigma(y) + \sum_{i=1}^{n} h^i MLP_{2,i}(y) + R_2$



Stratégie d'apprentissage

Etapes de l'apprentissage de notre modèle :

Stratégie d'apprentissage

Etapes de l'apprentissage de notre modèle :

- On construit un Dataset $(Y0_i, h_i)_{1 \leq i \leq N}$

Stratégie d'apprentissage

Etapes de l'apprentissage de notre modèle :

- On construit un Dataset $(Y0_i, h_i)_{1 \leq i \leq N}$
- On calcule pour chaque point du dataset une approximation quasi exact de la valeur réel $(Y1_i)_{1 \leq i \leq N}$

Stratégie d'apprentissage

Etapes de l'apprentissage de notre modèle :

- On construit un Dataset $(Y0_i, h_i)_{1 \leq i \leq N}$
- On calcule pour chaque point du dataset une approximation quasi exact de la valeur réel $(Y1_i)_{1 \leq i \leq N}$
- On calcule la solution approchée par notre réseau en simulant $\phi_h^{\tilde{f},\tilde{\sigma}}(Y0)$

Stratégie d'apprentissage

Etapes de l'apprentissage de notre modèle :

- On construit un Dataset $(Y0_i, h_i)_{1 \leq i \leq N}$
- On calcule pour chaque point du dataset une approximation quasi exact de la valeur réel $(Y1_i)_{1 \leq i \leq N}$
- On calcule la solution approchée par notre réseau en simulant $\phi_h^{\tilde{f},\tilde{\sigma}}(Y0)$
- On calcule l'erreur entre la solution exact et la solution donnée par notre réseau de neuronne

Stratégie d'apprentissage

Etapes de l'apprentissage de notre modèle :

- On construit un Dataset $(Y0_i, h_i)_{1 \leq i \leq N}$
- On calcule pour chaque point du dataset une approximation quasi exact de la valeur réel $(Y1_i)_{1 \leq i \leq N}$
- On calcule la solution approchée par notre réseau en simulant $\phi_h^{\tilde{f},\tilde{\sigma}}(Y0)$
- On calcule l'erreur entre la solution exact et la solution donnée par notre réseau de neuronne
- On met à jour les différents paramètres des MLP par rétropropagation

Fonction de perte

Definition

Une fonction de perte est une fonction calculant l'erreur entre la prévision du modèle et la solution exact

Fonction de perte

Definition

Une fonction de perte est une fonction calculant l'erreur entre la prévision du modèle et la solution exact

Exemple :

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

Fonction de perte

Definition

Une fonction de perte est une fonction calculant l'erreur entre la prévision du modèle et la solution exact

Exemple :

- MSE : $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
- MAE : $\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$

Fonction de perte

Definition

Une fonction de perte est une fonction calculant l'erreur entre la prévision du modèle et la solution exact

Exemple :

- MSE : $\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$
- MAE : $\frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i|$

Construction de notre fonction :

$$Loss_{train} = \frac{1}{K}\sum_{k=0}^{K-1}\left[\frac{|E[\phi_{h^{(k)}}^{f_{app}(.,h^{(k)}),\sigma_{app}(.,h^{(k)})}(y_0^{(k)})] - E[\varphi_{h^{(k)}}^{f,\sigma}(y_0^{(k)})]|}{h^{(k)p+1}} + \frac{|Cov[\phi_{h^{(k)}}^{f_{app}(.,h^{(k)}),\sigma_{app}(.,h^{(k)})}(y_0^{(k)})] - Cov[\varphi_{h^{(k)}}^{f,\sigma}(y_0^{(k)})]|}{h^{(k)p+1}}\right]$$

Hyperparamètres

Définition

Un hyperparamètre est un paramètre définit en amont de l'apprentissage et régissant le processus.

Hyperparamètres

Définition

Un hyperparamètre est un paramètre définit en amont de l'apprentissage et régissant le processus.

- Nombres de couches cachés

Hyperparamètres

Définition

Un hyperparamètre est un paramètre définit en amont de l'apprentissage et régissant le processus.

- Nombres de couches cachés

Proposition

Toute fonction continue définie sur un ensemble compact peut être approchée d'aussi près que l'on veut par un MLP à une couche cachée avec activation non linéaire.

Hyperparamètres

Définition

Un hyperparamètre est un paramètre définit en amont de l'apprentissage et régissant le processus.

- Nombres de couches cachés

Proposition

Toute fonction continue définie sur un ensemble compact peut être approchée d'aussi près que l'on veut par un MLP à une couche cachée avec activation non linéaire.

- Nombres de neurones par couches cachés

Hyperparamètres

Définition

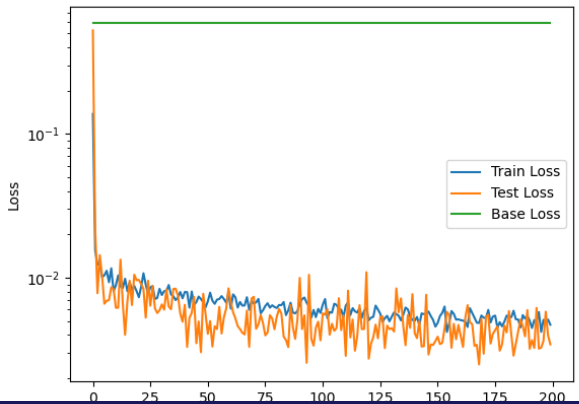
Un hyperparamètre est un paramètre définit en amont de l'apprentissage et régissant le processus.

- Nombres de couches cachés

Proposition

Toute fonction continue définie sur un ensemble compact peut être approchée d'aussi près que l'on veut par un MLP à une couche cachée avec activation non linéaire.

- Nombres de neurones par couches cachés



1 CC
100 Neurones

Hyperparamètres

Définition

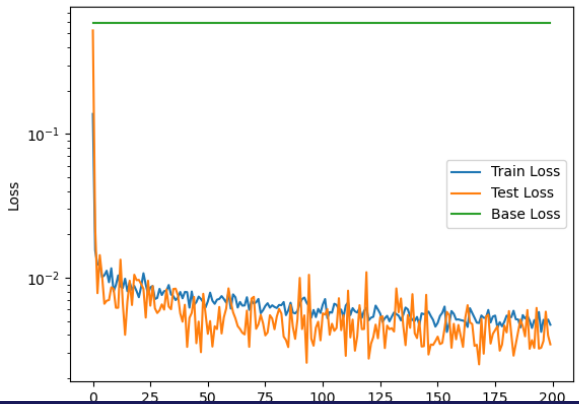
Un hyperparamètre est un paramètre définit en amont de l'apprentissage et régissant le processus.

- Nombres de couches cachés

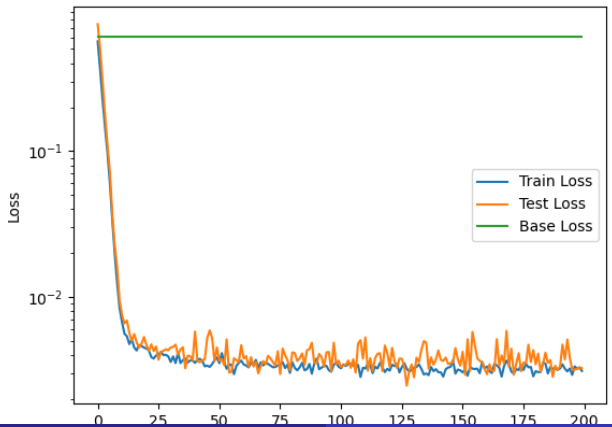
Proposition

Toute fonction continue définie sur un ensemble compact peut être approchée d'aussi près que l'on veut par un MLP à une couche cachée avec activation non linéaire.

- Nombres de neurones par couches cachés



1 CC
100 Neurones



2 CC
50 Neurones

Hyperparamètres

- Fonction d'activation
  - Sigmoïde : $f(x) = \frac{1}{1+e^{-x}}$

Hyperparamètres

- Fonction d'activation
  - Sigmoïde : $f(x) = \frac{1}{1+e^{-x}}$
  - Tanh : $f(x) = \tanh(x)$

Hyperparamètres

- Fonction d'activation
  - Sigmoïde : $f(x) = \frac{1}{1+e^{-x}}$
  - Tanh : $f(x) = \tanh(x)$
  - ReLU :

$$f(x) = \begin{cases} 0 \text{ si } x < 0 \\ x \text{ si } x \geq 0 \end{cases}$$

Hyperparamètres

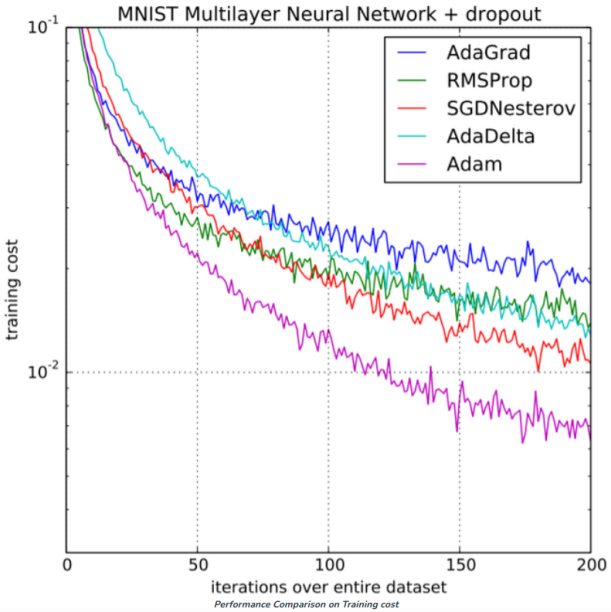
- Fonction d'activation
  - Sigmoïde : $f(x) = \frac{1}{1+e^{-x}}$
  - Tanh : $f(x) = \tanh(x)$
  - ReLU :
    $$f(x) = \begin{cases} 0 \text{ si } x < 0 \\ x \text{ si } x \geq 0 \end{cases}$$
- Algorithme d'optimisation

Hyperparamètres

- Fonction d'activation
  - Sigmoïde : $f(x) = \frac{1}{1+e^{-x}}$
  - Tanh : $f(x) = \tanh(x)$
  - ReLU :

$$f(x) = \begin{cases} 0 \text{ si } x < 0 \\ x \text{ si } x \geq 0 \end{cases}$$

- Algorithme d'optimisation

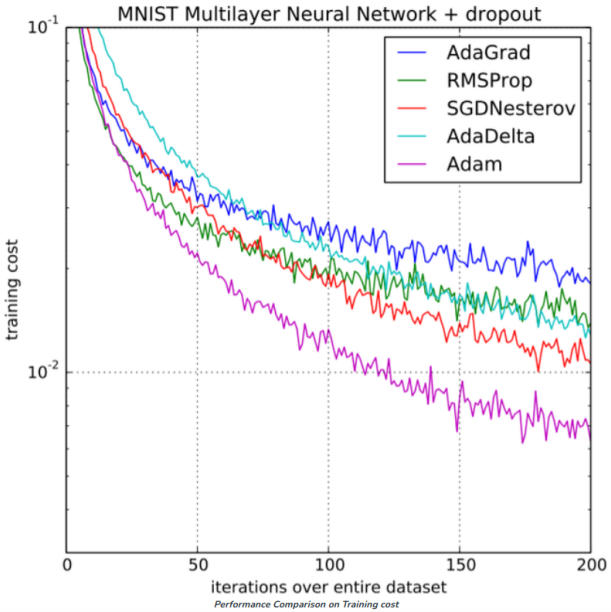


Hyperparamètres

- Fonction d'activation
  - Sigmoïde : $f(x) = \frac{1}{1+e^{-x}}$
  - Tanh : $f(x) = \tanh(x)$
  - ReLU :

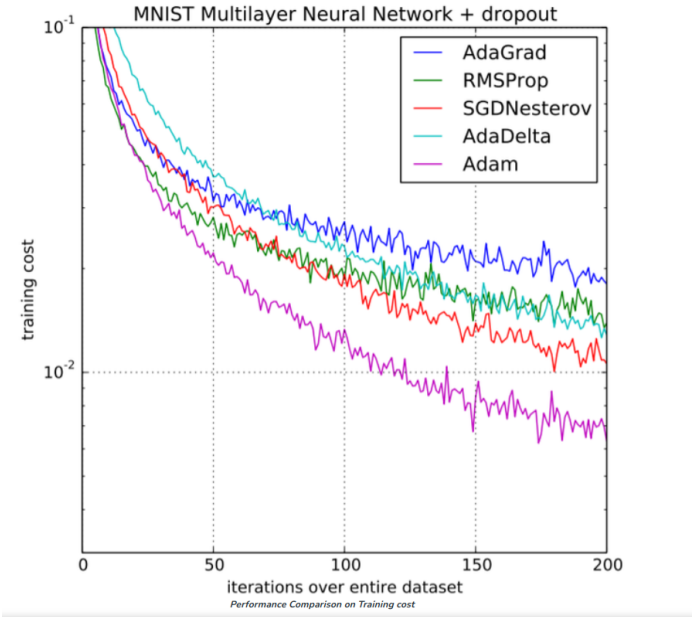
$$f(x) = \begin{cases} 0 \text{ si } x < 0 \\ x \text{ si } x \geq 0 \end{cases}$$

- Algorithme d'optimisation



Hyperparamètres

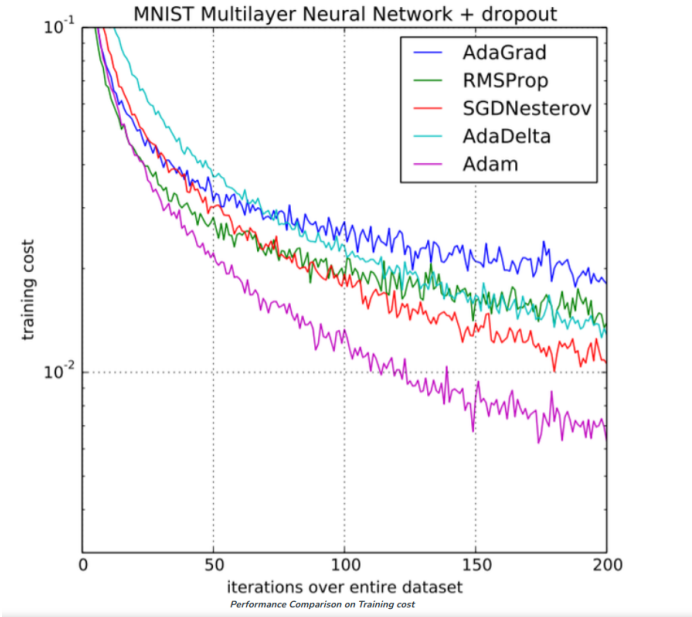
- Fonction d'activation
  - Sigmoïde : $f(x) = \frac{1}{1+e^{-x}}$
  - Tanh : $f(x) = \tanh(x)$
  - ReLU :
$$f(x) = \begin{cases} 0 \text{ si } x < 0 \\ x \text{ si } x \geq 0 \end{cases}$$
- Algorithme d'optimisation



- Learning rate

Hyperparamètres

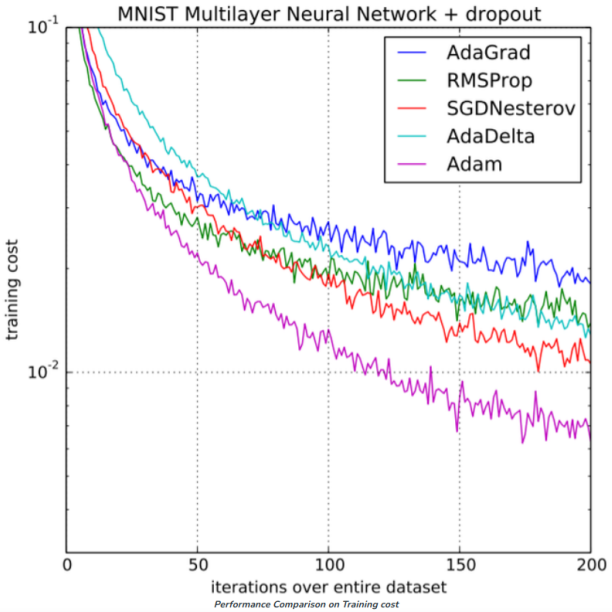
- Fonction d'activation
  - Sigmoïde : $f(x) = \frac{1}{1+e^{-x}}$
  - Tanh : $f(x) = \tanh(x)$
  - ReLU :
    $$f(x) = \begin{cases} 0 \text{ si } x < 0 \\ x \text{ si } x \geq 0 \end{cases}$$
- Algorithme d'optimisation



- Learning rate
  $$W_{i+1} = W_i - \eta \frac{\partial L}{\partial W_i}$$

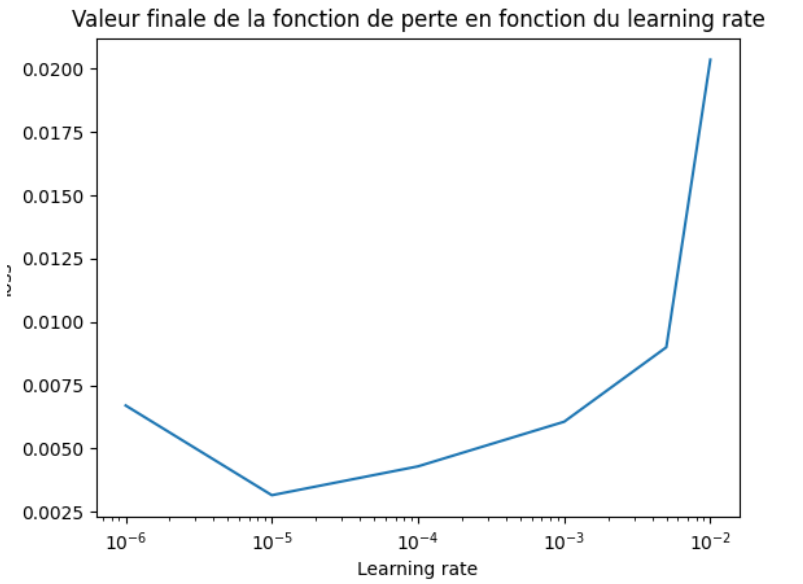
Hyperparamètres

- Fonction d'activation
  - Sigmoïde : $f(x) = \frac{1}{1+e^{-x}}$
  - Tanh : $f(x) = \tanh(x)$
  - ReLU :
$$f(x) = \begin{cases} 0 \text{ si } x < 0 \\ x \text{ si } x \geq 0 \end{cases}$$
- Algorithme d'optimisation



- Learning rate

$$W_{i+1} = W_i - \eta \frac{\partial L}{\partial W_i}$$



Sommaire

Sommaire

Étude de l'amélioration

$$dX = \lambda X dt + \mu X dW \quad \lambda \in \mathbb{R} \quad \mu \in \mathbb{R} \quad X(0) = x$$

Étude de l'amélioration

$dX = \lambda X dt + \mu X dW \quad \lambda \in \mathbb{R} \quad \mu \in \mathbb{R} \quad X(0) = x$

- $\lambda = 1$ $\mu = 0.1$

Étude de l'amélioration

$dX = \lambda X dt + \mu X dW \quad \lambda \in \mathbb{R} \quad \mu \in \mathbb{R} \quad X(0) = x$

•$\lambda = 1$ $\mu = 0.1$ • Domaine d'entrainement : [0,2]

Étude de l'amélioration

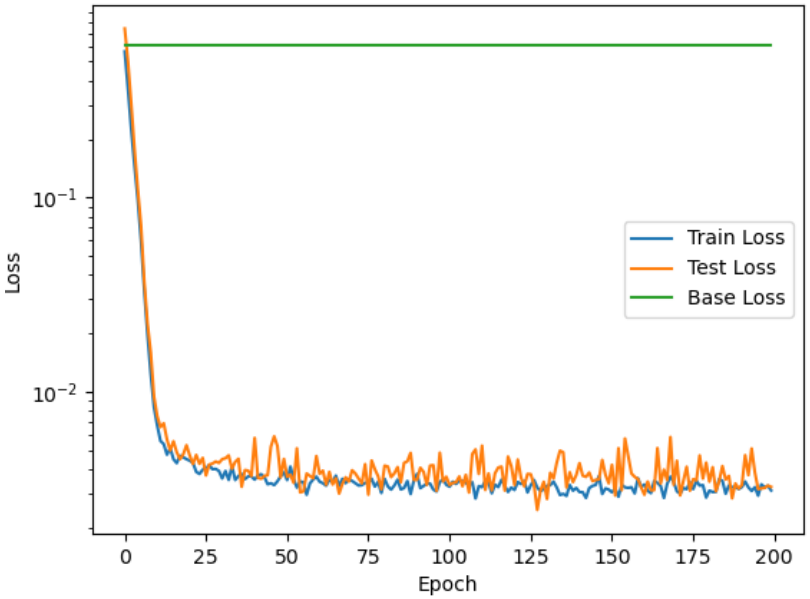
$dX = \lambda X dt + \mu X dW \quad \lambda \in \mathbb{R} \quad \mu \in \mathbb{R} \quad X(0) = x$

- $\lambda = 1$ $\mu = 0.1$
- Domaine d'entrainement : [0,2]
- Intégrateur numérique : Euler-Maruyama

Étude de l'amélioration

$dX = \lambda X dt + \mu X dW \quad \lambda \in \mathbb{R} \quad \mu \in \mathbb{R} \quad X(0) = x$

- $\lambda = 1$ $\mu = 0.1$
- Domaine d'entrainement : [0,2]
- Intégrateur numérique : Euler-Maruyama
- Amélioration de l'ordre cherchée : 2 $\quad \tilde{f} = f + hf_1 + h^2 f_2$

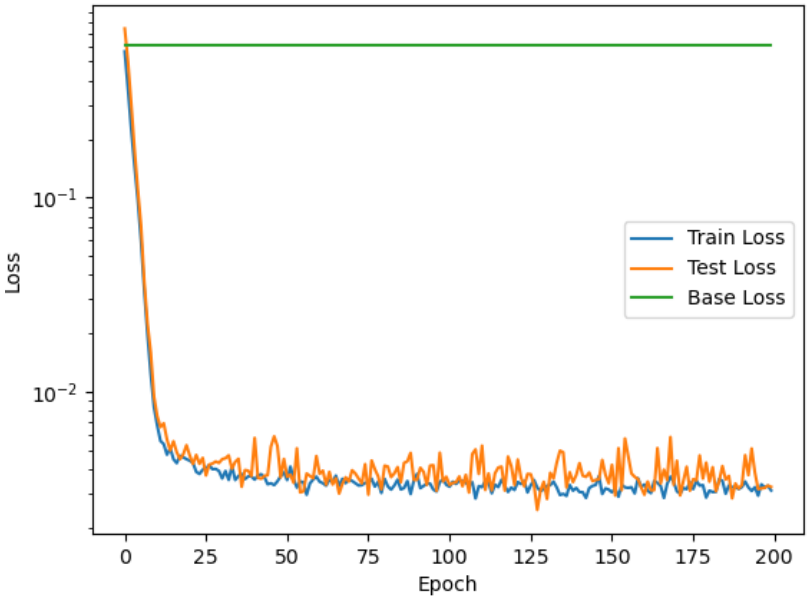


- Diminution de la perte par 100

Étude de l'amélioration

$dX = \lambda X dt + \mu X dW \quad \lambda \in \mathbb{R} \quad \mu \in \mathbb{R} \quad X(0) = x$

- $\lambda = 1$ $\mu = 0.1$
- Domaine d'entrainement : [0,2]
- Intégrateur numérique : Euler-Maruyama
- Amélioration de l'ordre cherchée : 2 $\quad \tilde{f} = f + hf_1 + h^2 f_2$

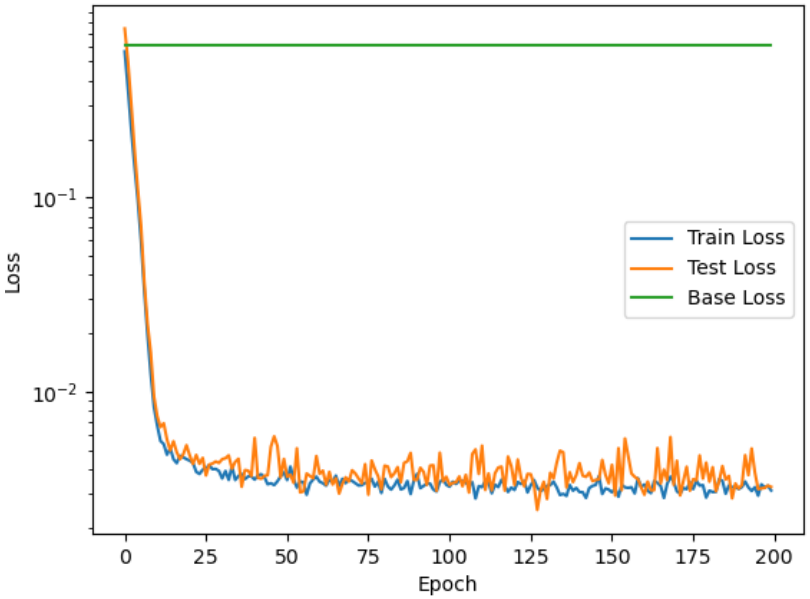


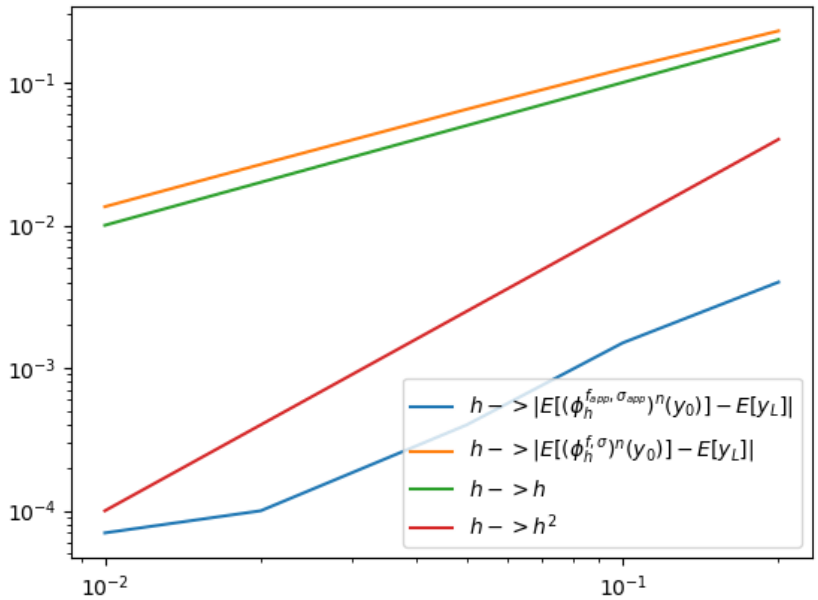
- Diminution de la perte par 100
- Pas d'overfitting

Étude de l'amélioration

$dX = \lambda X dt + \mu X dW \quad \lambda \in \mathbb{R} \quad \mu \in \mathbb{R} \quad X(0) = x$

- $\lambda = 1$ $\mu = 0.1$ • Domaine d'entrainement : [0,2]
- Intégrateur numérique : Euler-Maruyama
- Amélioration de l'ordre cherchée : 2 $\quad \tilde{f} = f + hf_1 + h^2 f_2$



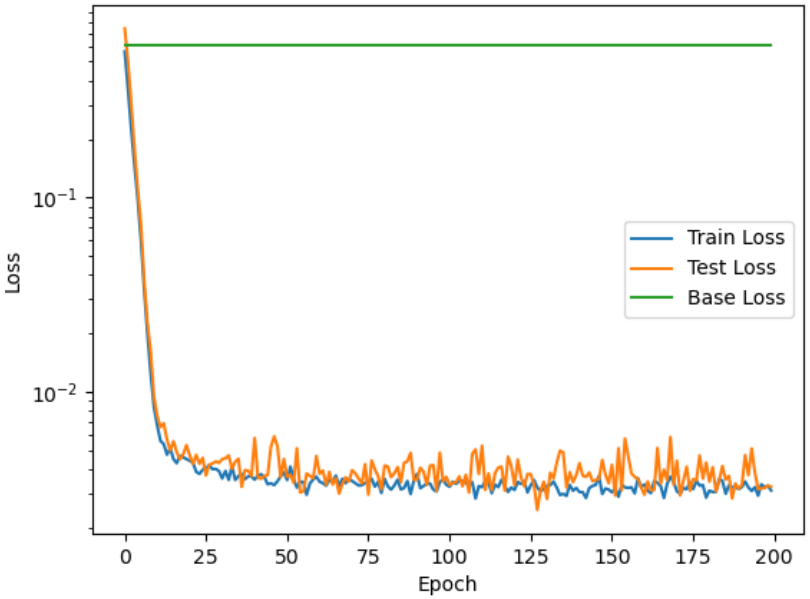


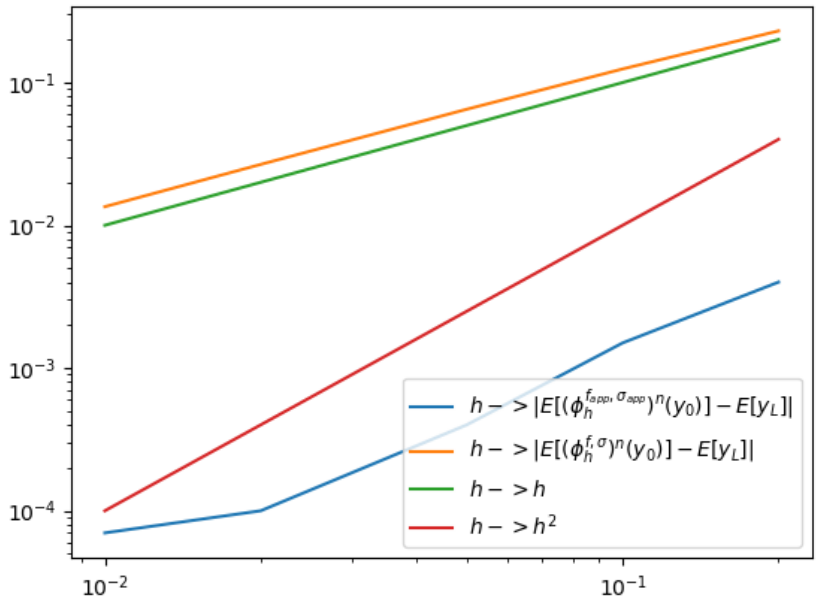
- Diminution de la perte par 100
- Pas d'overfitting

Étude de l'amélioration

$dX = \lambda X dt + \mu X dW \quad \lambda \in \mathbb{R} \quad \mu \in \mathbb{R} \quad X(0) = x$

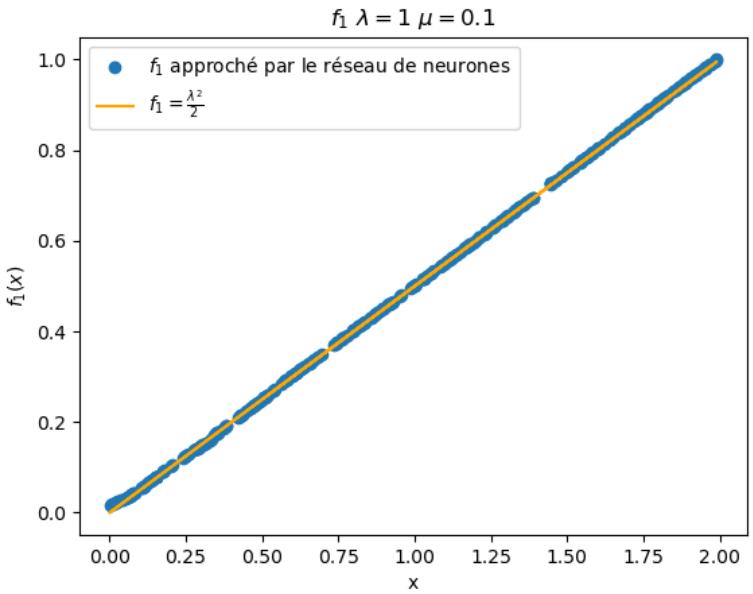
- $\lambda = 1$ $\mu = 0.1$ • Domaine d'entrainement : [0,2]
- Intégrateur numérique : Euler-Maruyama
- Amélioration de l'ordre cherchée : 2 $\quad \tilde{f} = f + hf_1 + h^2 f_2$

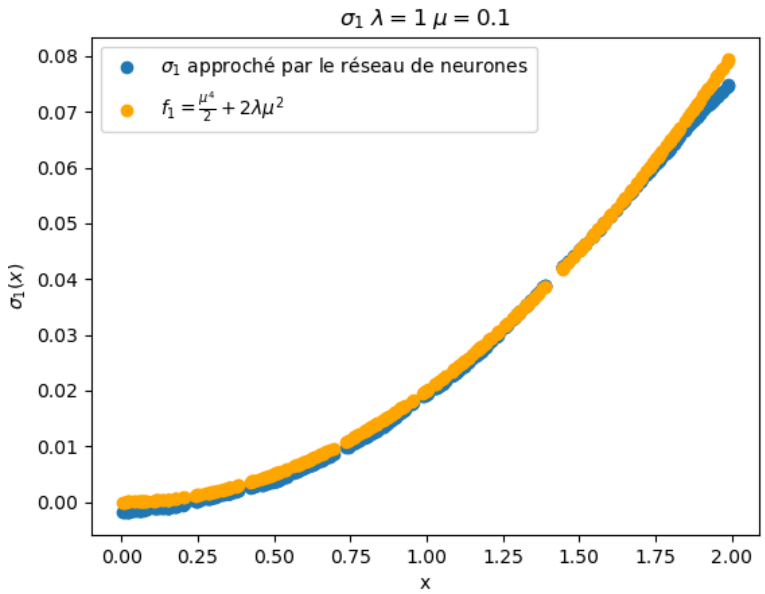




- Diminution de la perte par 100
- Pas d'overfitting

Étude des MLP





Sommaire

Pendule stochastique

$$\begin{cases} dX = f(q,p)(dt + \circ dW) \\ f(q,p) = J\nabla H(q,p) = \begin{pmatrix} p \\ -\sin(q) \end{pmatrix} \\ H(q,p) = \frac{p^2}{2} + \sin(q) \end{cases}$$

Pendule stochastique

$$\begin{cases} dX = f(q,p)(dt + \circ dW) \\ f(q,p) = J\nabla H(q,p) = \begin{pmatrix} p \\ -\sin(q) \end{pmatrix} \\ H(q,p) = \frac{p^2}{2} + \sin(q) \end{cases}$$

MidPoint Stochastique : $X_1 = X_0 + f(\frac{X_0+X_1}{2})h + \sigma(\frac{X_0+X_1}{2})\sqrt{h}\xi$

Pendule stochastique

$$\begin{cases} dX = f(q,p)(dt + \circ dW) \\ f(q,p) = J\nabla H(q,p) = \begin{pmatrix} p \\ -\sin(q) \end{pmatrix} \\ H(q,p) = \frac{p^2}{2} + \sin(q) \end{cases}$$

MidPoint Stochastique : $X_1 = X_0 + f(\frac{X_0+X_1}{2})h + \sigma(\frac{X_0+X_1}{2})\sqrt{h}\xi$

Domaine d'entrainement :

$[-\pi, \pi]x[-1.5, 1.5]$

Pendule stochastique

$$\begin{cases} dX = f(q,p)(dt + \circ dW) \\ f(q,p) = J\nabla H(q,p) = \begin{pmatrix} p \\ -\sin(q) \end{pmatrix} \\ H(q,p) = \frac{p^2}{2} + \sin(q) \end{cases}$$

MidPoint Stochastique : $X_1 = X_0 + f(\frac{X_0+X_1}{2})h + \sigma(\frac{X_0+X_1}{2})\sqrt{h}\xi$

Domaine d'entrainement :

$$[-\pi,\pi]x[-1.5,1.5] \rightarrow \begin{pmatrix} \frac{\pi}{3}\cos(\theta) + \epsilon_1 \\ \sin(\theta) + \epsilon_2 \end{pmatrix} \theta \in [0,2\pi], \epsilon_1\epsilon_2 \in [-0.2,0.2]$$

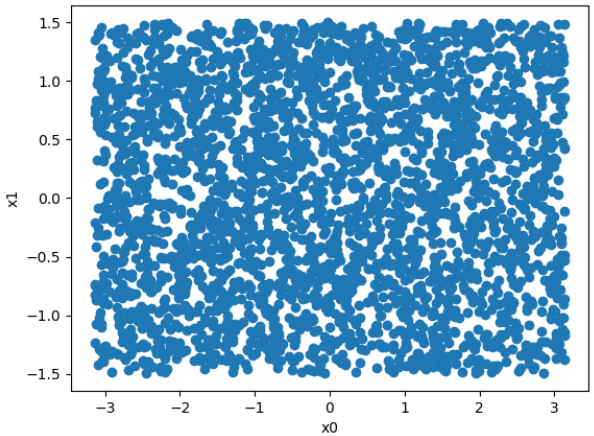
Pendule stochastique

$$\begin{cases} dX = f(q,p)(dt + \circ dW) \\ f(q,p) = J\nabla H(q,p) = \begin{pmatrix} p \\ -\sin(q) \end{pmatrix} \\ H(q,p) = \frac{p^2}{2} + \sin(q) \end{cases}$$

MidPoint Stochastique : $X_1 = X_0 + f(\frac{X_0+X_1}{2})h + \sigma(\frac{X_0+X_1}{2})\sqrt{h}\xi$

Domaine d'entrainement :

$$[-\pi,\pi]x[-1.5,1.5] \rightarrow \begin{pmatrix} \frac{\pi}{3}\cos(\theta) + \epsilon_1 \\ \sin(\theta) + \epsilon_2 \end{pmatrix} \theta \in [0,2\pi], \epsilon_1\epsilon_2 \in [-0.2,0.2]$$



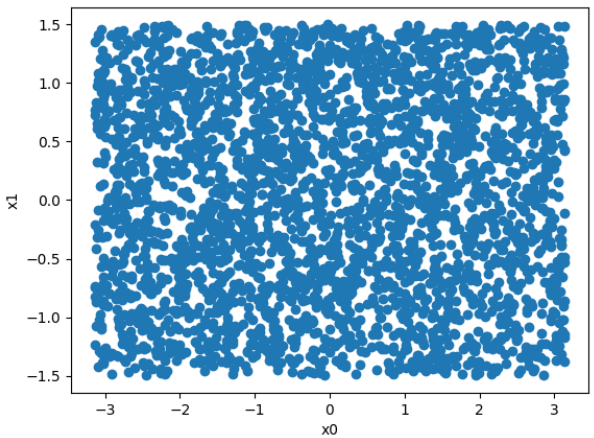
Pendule stochastique

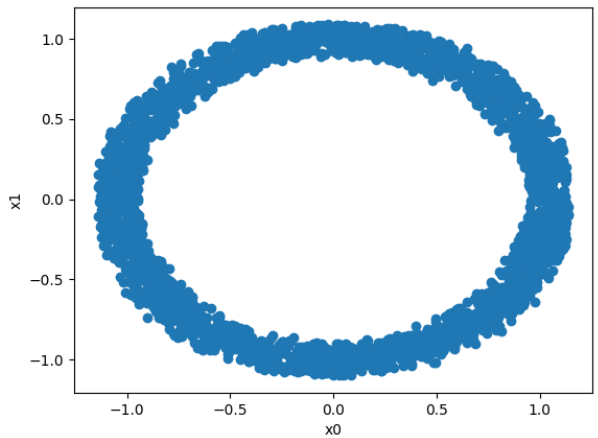
$$\begin{cases} dX = f(q,p)(dt + \circ dW) \\ f(q,p) = J\nabla H(q,p) = \begin{pmatrix} p \\ -\sin(q) \end{pmatrix} \\ H(q,p) = \frac{p^2}{2} + \sin(q) \end{cases}$$

MidPoint Stochastique : $X_1 = X_0 + f(\frac{X_0+X_1}{2})h + \sigma(\frac{X_0+X_1}{2})\sqrt{h}\xi$

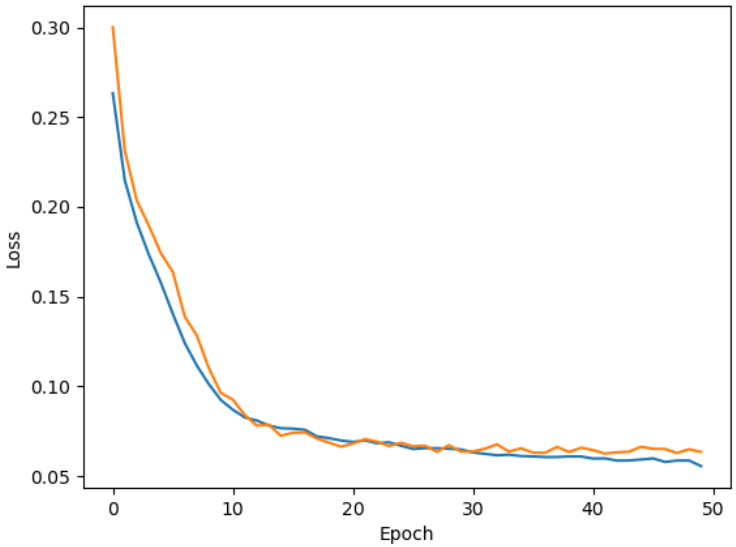
Domaine d'entrainement :

$$[-\pi,\pi]x[-1.5,1.5] \rightarrow \begin{pmatrix} \frac{\pi}{3}\cos(\theta) + \epsilon_1 \\ \sin(\theta) + \epsilon_2 \end{pmatrix} \quad \theta \in [0,2\pi], \epsilon_1\epsilon_2 \in [-0.2,0.2]$$

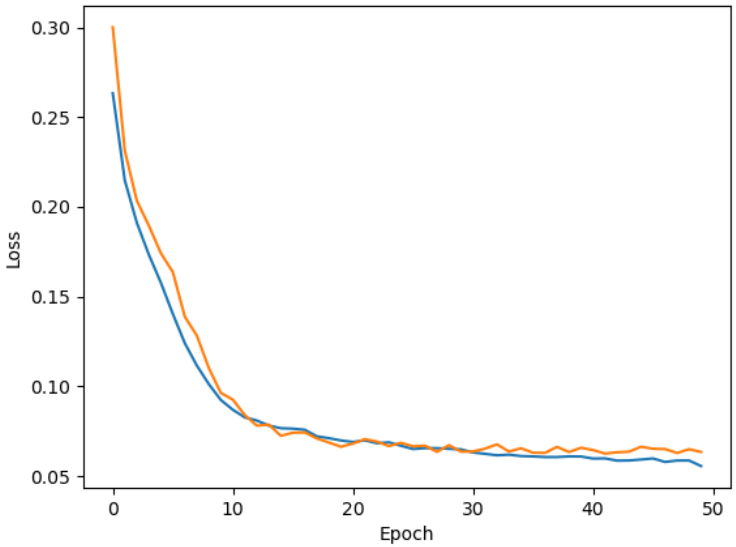




Étude de l'amélioration

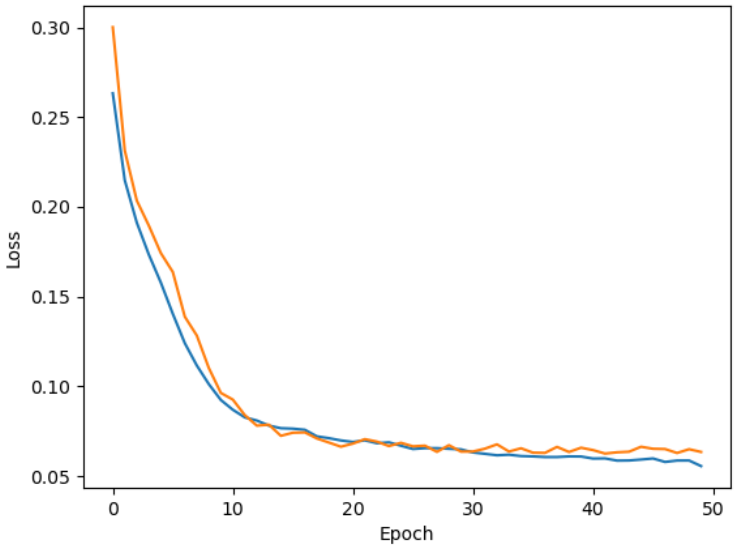


Étude de l'amélioration



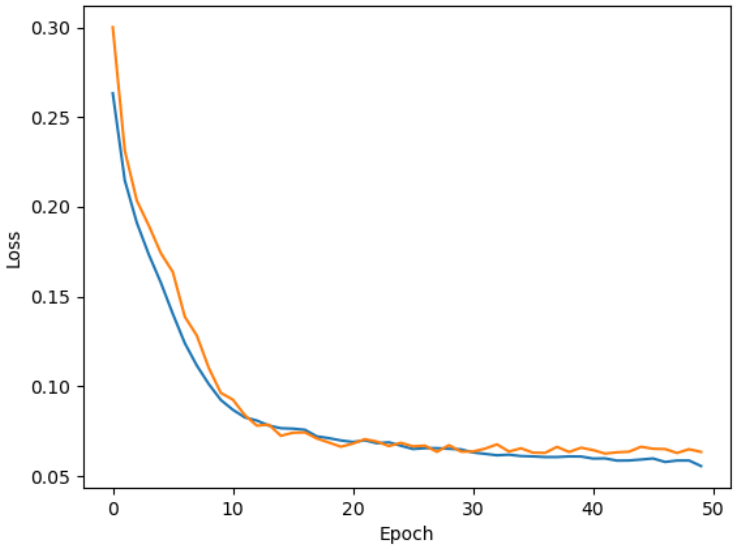
- Diminution de la perte par 10

Étude de l'amélioration



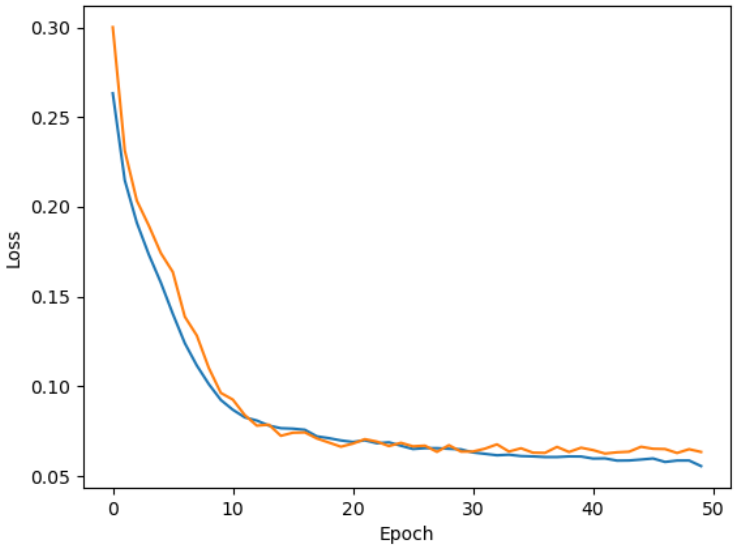
- Diminution de la perte par 10
- Apprentissage stable

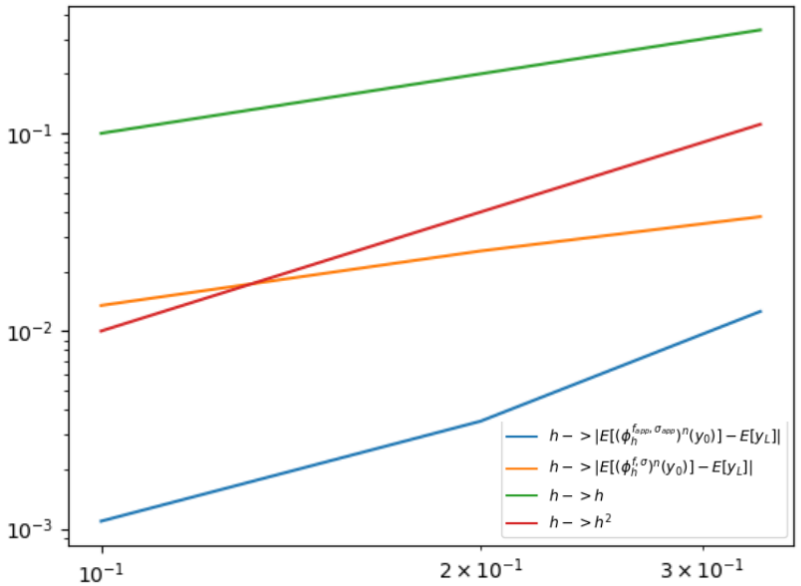
Étude de l'amélioration



- Diminution de la perte par 10
- Apprentissage stable
- Pas d'overfitting $\rightarrow$ Dropout

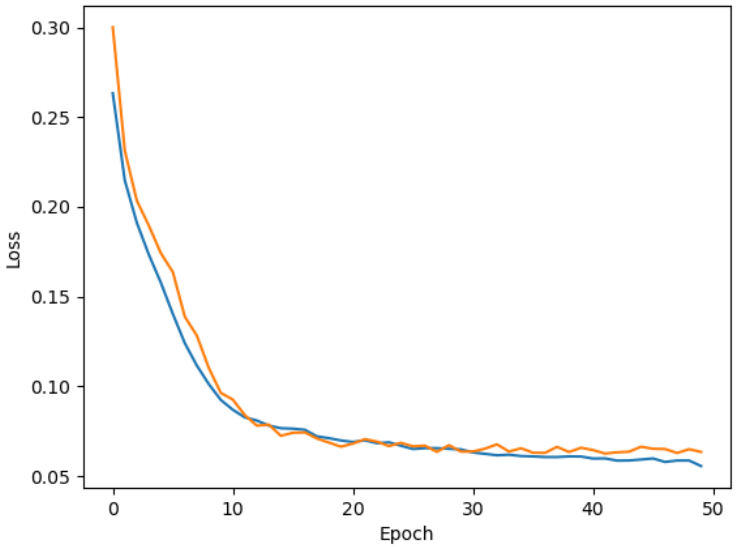
Étude de l'amélioration



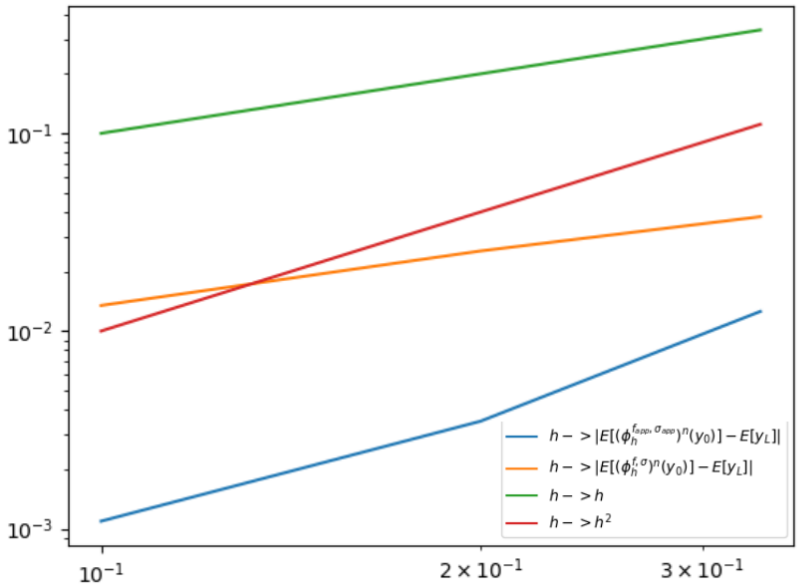


- Diminution de la perte par 10
- Apprentissage stable
- Pas d'overfitting $\rightarrow$ Dropout

Étude de l'amélioration

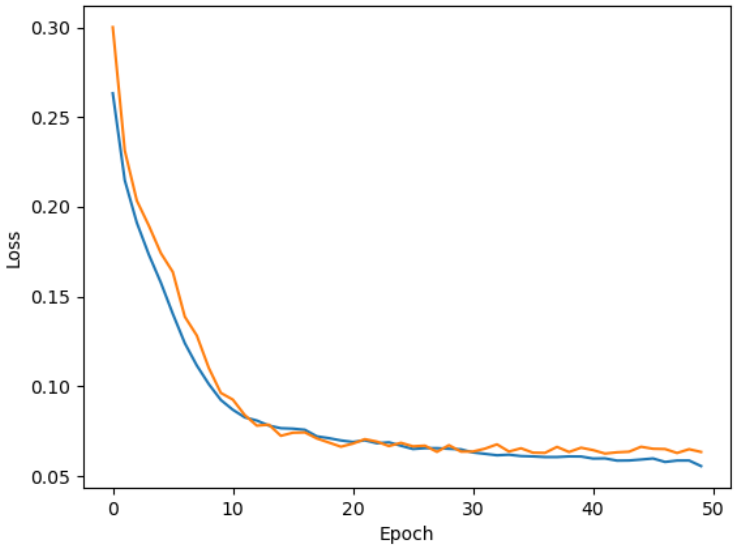


- Diminution de la perte par 10
- Apprentissage stable
- Pas d'overfitting $\rightarrow$ Dropout

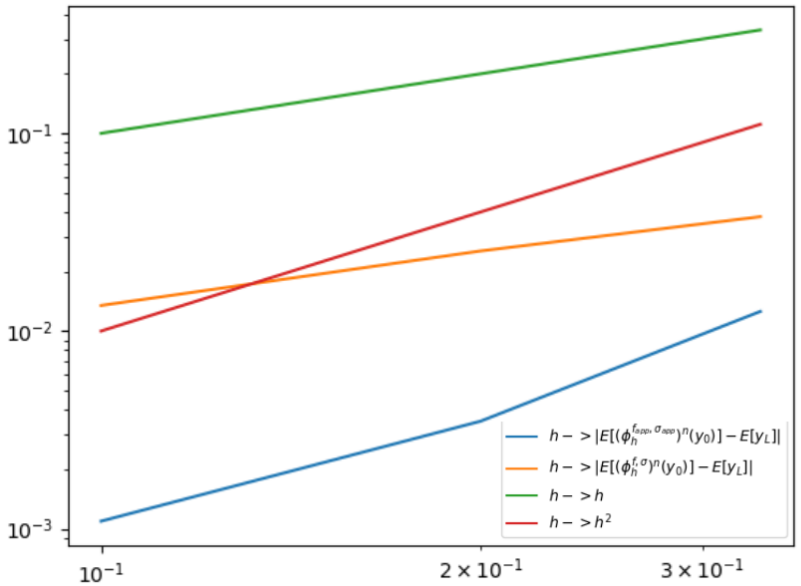


- Plage de h assez réduite

Étude de l'amélioration

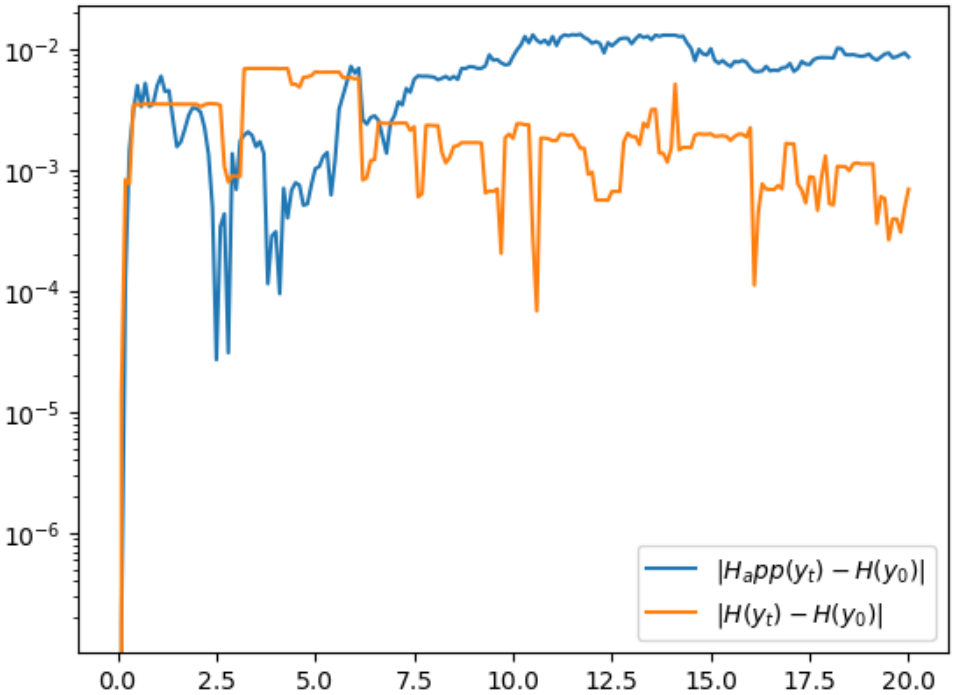


- Diminution de la perte par 10
- Apprentissage stable
- Pas d'overfitting $\rightarrow$ Dropout

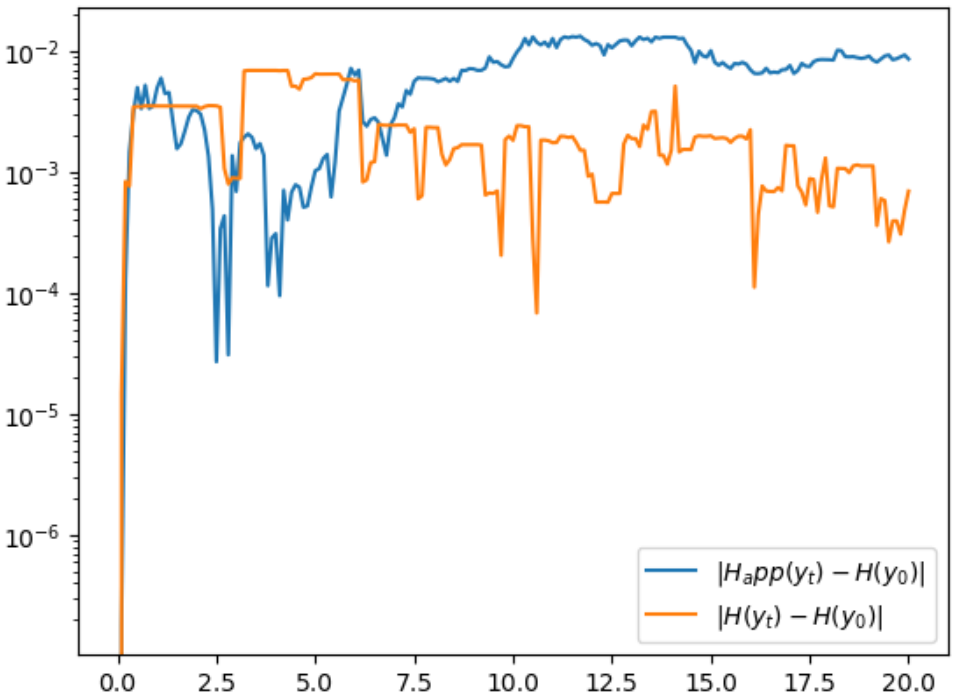


- Plage de h assez réduite
- Augmentation de l'ordre

Hamiltonien

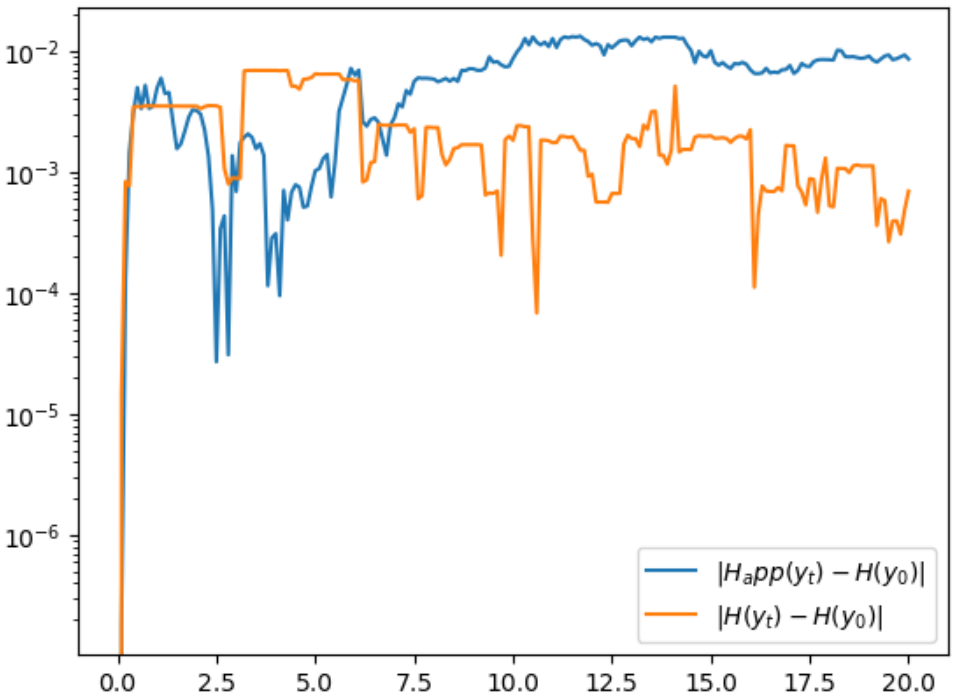


Hamiltonien



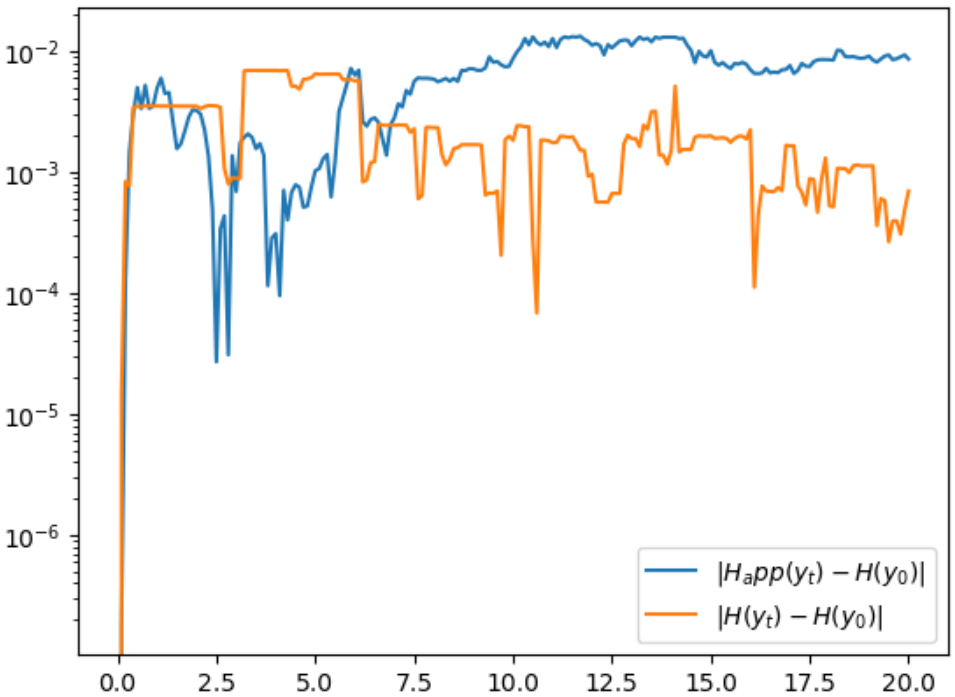
- Point Milieu Simple : Oscillation autour de la valeur initiale

Hamiltonien



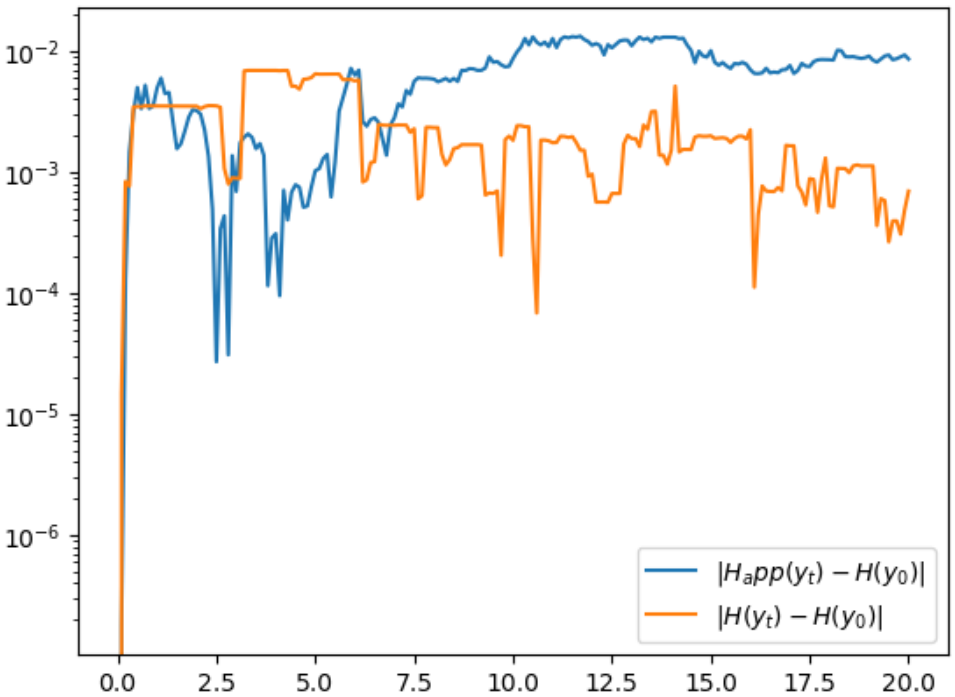
- Point Milieu Simple : Oscillation autour de la valeur initiale
- Point Milieu Modifié : Oscillation pour des temps faibles puis divergence faible de l'erreur.

Hamiltonien



- Point Milieu Simple : Oscillation autour de la valeur initiale
- Point Milieu Modifié : Oscillation pour des temps faibles puis divergence faible de l'erreur.
- Contrairement aux EDOs, pas vraiment une méthode symplectique

Hamiltonien



- Point Milieu Simple : Oscillation autour de la valeur initiale
- Point Milieu Modifié : Oscillation pour des temps faibles puis divergence faible de l'erreur.
- Contrairement aux EDOs, pas vraiment une méthode symplectique
- Solution potentielle : Réseaux de neuronnes Hamiltonien

Sommaire

Conclusion et suite

Résumé :

- Définition d'un modèle approximant le champs modifié

Conclusion et suite

Résumé :

- Définition d'un modèle approximant le champs modifié
- Réglage des hyperparamètres du modèle

Conclusion et suite

Résumé :

- Définition d'un modèle approximant le champs modifié
- Réglage des hyperparamètres du modèle
- Apprentissage sur 2 cas tests significatifs

Conclusion et suite

Résumé :

- Définition d'un modèle approximant le champs modifié
- Réglage des hyperparamètres du modèle
- Apprentissage sur 2 cas tests significatifs

Suite du travail :

Conclusion et suite

Résumé :

- Définition d'un modèle approximant le champs modifié
- Réglage des hyperparamètres du modèle
- Apprentissage sur 2 cas tests significatifs

Suite du travail :

- Etudes d'autres exemples

Conclusion et suite

Résumé :

- Définition d'un modèle approximant le champs modifié
- Réglage des hyperparamètres du modèle
- Apprentissage sur 2 cas tests significatifs

Suite du travail :

- Etudes d'autres exemples
  - Bruit Additif : $dX = f(X)dt + \sigma dW, \sigma \in \mathbb{R}$

Conclusion et suite

Résumé :

- Définition d'un modèle approximant le champs modifié
- Réglage des hyperparamètres du modèle
- Apprentissage sur 2 cas tests significatifs

Suite du travail :

- Etudes d'autres exemples
  - Bruit Additif : $dX = f(X)dt + \sigma dW, \sigma \in \mathbb{R}$
  - EDS Standard

Conclusion et suite

Résumé :

- Définition d'un modèle approximant le champs modifié
- Réglage des hyperparamètres du modèle
- Apprentissage sur 2 cas tests significatifs

Suite du travail :

- Etudes d'autres exemples
  - Bruit Additif : $dX = f(X)dt + \sigma dW, \sigma \in \mathbb{R}$
  - EDS Standard
- Test d'augmentations d'ordres plus grands

Conclusion et suite

Résumé :

- Définition d'un modèle approximant le champs modifié
- Réglage des hyperparamètres du modèle
- Apprentissage sur 2 cas tests significatifs

Suite du travail :

- Etudes d'autres exemples
  - Bruit Additif : $dX = f(X)dt + \sigma dW, \sigma \in \mathbb{R}$
  - EDS Standard
- Test d'augmentations d'ordres plus grands
- Réduire la complexité en temps et en mémoire