Techniques d'intelligence artificielle pour les équations différentielles autonomes

Stage de Master 2 - Rapport

Stage encadré par Philippe CHARTIER, Mohammed LEMOU et Florian MEHATS

Remerciements

Je remercie Philippe CHARTIER, Mohammed LEMOU et Florian MEHATS pour l'encadrement, l'aide et les conseils apportés tout au long du stage, mais aussi pour l'intérêt qu'ils m'ont fait porter pour ce domaine.

Introduction

L'intelligence artificielle, outre l'application à des problèmes complexes tels que le traitement d'image, du signal ou encore la classification des données [1],[2], peut s'appliquer à l'étude de problèmes d'évolution. En effet, il est possible d'utiliser des techniques d'IA sur des équations différentielles - voire aux dérivées partielles lorsqu'elles sont discrétisées en espace - autonomes $\dot{y} = f(y)$ afin de retrouver le champ de vecteurs f via l'étude de trajectoires [3],[4] y compris lorsque ces dernières sont perturbées par un bruit [5], afin d'imiter la trajectoire (discrétisée en temps) d'une solution d'EDO [6] ou lorsque l'on veut simplifier un modèle physique afin de réduire le coût de calcul [7]

L'objectif de ce stage consiste en l'application de méthodes d'intelligence artificielle à des équations différentielles, c'est-à-dire que, étant donné une EDO $\dot{y} = f(y)$, il s'agit de retrouver le champ de vecteurs f de manière approchée à partir de données sur la trajectoire de la solution partant d'une donnée initiale y_0, et en connaissant une donnée "finale" y_1. En réalité, il est nécessaire de connaître plus d'une trajectoire (donc on se donne plusieurs conditions initiales $y_0^{(k)}$ et plusieurs données $y_1^{(k)}$) afin de pouvoir calculer f. C'est là que les méthodes d'IA entrent en jeu, puisque les données sont souvent nombreuses. Par ailleurs, f est retrouvé de manière approchée, c'est-à-dire que des méthodes numériques pour les EDO entrent en jeu [8],[9]. Une part du travail est consacrée à l'étude de la précision avec laquelle le champ de vecteurs f est retrouvé et le lien avec l'ordre de la méthode numérique, en particulier lorsque l'on rajoute des points intermédiaires entre y_0 et y_1, permettant d'apporter plus de précision.

La première partie traite de l'introduction des méthodes d'intelligence artificielle pour l'étude d'EDO. Il y est introduit la définition de fonction de perte *Loss*, qui mesure l'écart entre des points de la trajectoire réelle et de la trajectoire calculée par une IA, la notion de réseaux de neurones, notamment le PMC [2], ainsi que le gradient stochastique, technique d'optimisation largement employée dans le cadre du stage afin de minimiser la fonction *Loss* [1],[10].

La seconde partie se concentre sur le cas linéaire et aborde quatre méthodes numériques classiques des EDO de petit ordre [8],[9] permettant de retrouver le champ de vecteurs. Cette partie accorde une place importante à la convexité de la fonction *Loss*, critère très apprécié en optimisation.

La troisième partie propose toujours d'étudier l'approximation de f dans le cas linéaire, en fonction de certaines méthodes numériques mieux adaptées aux algorithmes d'optimisation que l'on peut trouver dans la littérature [3],[8],[9]

Enfin, la quatrième partie aborde le cas où f n'est plus linéaire, et met en évidence une nouvelle structure de réseau de neurones [1][7], adaptée à l'étude de systèmes dynamiques complexes utilisé dans d'autres disciplines comme la physique ou la biologie [3],[7]. Les méthodes numériques employées reprennent celles utilisées dans le cas linéaire [3],[8],[9].

Table des matières

Partie 1

Intoduction et outils

1.1 Position du problème

Soit $f : \mathbb{R}^d \to \mathbb{R}^d$ un champ de vecteurs continu et localement lipschitzien. Soient $h > 0, N \in \mathbb{N}^*$ et $t_0 = 0 < t_1 < \cdots < t_N = h$. On considère le problème de Cauchy suivant:

$$\begin{cases} \dot{y} = f(y) \\ y(0) = y_0 \end{cases}$$

et on suppose que, pour tout $j \in [\![0, N]\!]$, $y_{\frac{t_j}{h}} = \phi_{t_j}(y_0)$ est connu, où ϕ est le flot associé à l'équation différentielle $\dot{y} = f(y)$.

L'idée du travail réalisé est de retrouver le champ de vecteurs f à partir des données y_j de la solution à différents temps. Concrètement, il s'agit de retrouver le champ de vecteurs f à partir d'observations sur la trajectoire partant de y_0.

Bien entendu, plusieurs observations sont menées, à partir de K données initiales $y_0^{(k)}$, pour $k \in [\![0, K-1]\!]$, donnant ainsi, pour tout $j \in [\![1, N]\!]$, $y^{(k)}_{\frac{t_j}{h}} = \phi_{t_j}(y_0^{(k)})$.

La stratégie est d'approcher le champ de vecteurs f aussi précisément que possible afin de pouvoir reconstituer toute la trajectoire. C'est là que les techniques d'IA entrent en jeu:

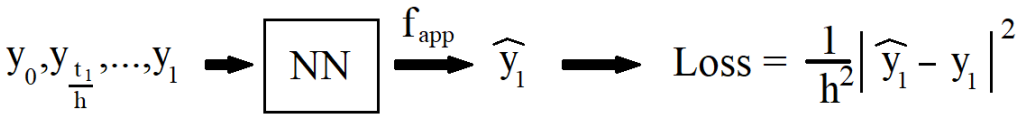

Figure 1.1: Illustration de la technique d'IA

- Les données $y_{\frac{t_j}{h}}$ sont entrées dans un réseau de neurones (NN), muni de paramètres.

- Il en ressort un champ de vecteurs approché f_{app} (calculé en fonction des paramètres de NN), ainsi qu'une valeur en h prédite de la solution, notée $\hat{y_1}$, faisant intervenir f_{app} au moyen d'un schéma numérique donné. Par exemple, si on opte pour le schéma d'Euler explicite, on aura $\hat{y_1} = y_0 + hf_{app}(y_0)$.

- On évalue la différence $\hat{y_1} - y_1$ à l'aide le la fonction de perte (Loss). C'est cette fonction que l'on va chercher à minimiser en les paramètres du réseau de neurones, faisant intervenir des méthodes d'optimisation

On dispose de K données, mais on ne les utilise pas toutes afin d'optimiser la fonction de perte. Une partie seulement le sera ($K_0 < K$ données). C'est la phase d'apprentissage (ou entraînement), qui consiste à minimiser cette fonction:

$$Loss_{Training} = \frac{1}{K_0} \sum_{k=0}^{K_0-1} \frac{1}{h^2} \left| \hat{y_1}^{(k)} - y_1^{(k)} \right|^2$$

Le reste des données servira à vérifier que les paramètres du réseau de neurones approchent correctement f, et que les autres différences $\hat{y_1^{(k)}} - y_1^{(k)}$ ne sont pas trop grandes non plus. C'est la phase de test, qui consiste à vérifier que la fonction suivante n'est pas trop grande par rapport à la fonction $Loss_{Training}$:

$$Loss_{Test} = \frac{1}{K - K_0} \sum_{k=K_0}^{K-1} \frac{1}{h^2} \left| \hat{y_1}^{(k)} - y_1^{(k)} \right|^2$$

Remarque. - *Lorsque la fonction $Loss_{Test}$ devient trop grande par rapport à $Loss_{Training}$, on dit qu'il y a surapprentissage.*

- *En notant $p \in]0,1[$ la proportion de données qui servent à l'entraînement, on a $K_0 = \lfloor pK_0 \rfloor$*

1.2 Réseaux de neurones

Dans cette section, nous allons détailler le fonctionnement d'un réseau de neurones, tel qu'il est utilisé dans le cadre du stage (ce qui ne correspond donc pas au cas général).

1.2.1 Couche de neurones

Une couche de neurones peut être vue comme une application $\mathbb{R}^d \to \mathbb{R}^\zeta$ qui envoie un vecteur vers un autre vecteur, et dont les paramètres sont les suivants:

- Une matrice $W \in \mathcal{M}_{\zeta,d}(\mathbb{R})$, appelée **matrice poids**

- Un vecteur $b \in \mathbb{R}^\zeta$ appelé **poids du biais** (le biais a la valeur 1)

- Une fonction $\sigma : \mathbb{R} \to \mathbb{R}$ appelée **fonction d'activation**

On dit que la couche comporte ζ neurones, et voici son fonctionnement:

- On prend un vecteur $x \in \mathbb{R}^d$, appelé **entrée**

- On calcule le vecteur $y = Wx + b \in \mathbb{R}^\zeta$

- On calcule le vecteur $x' = \Sigma(y) \in \mathbb{R}^\zeta$ en appliquant la fonction d'activation σ à chacune des composantes de y: $\Sigma((y_1, \dots, y_\zeta)^T) = (\sigma(y_1), \dots, \sigma(y_\zeta))^T$

- On obtient un vecteur $x' \in \mathbb{R}^\zeta$, appelée **sortie**

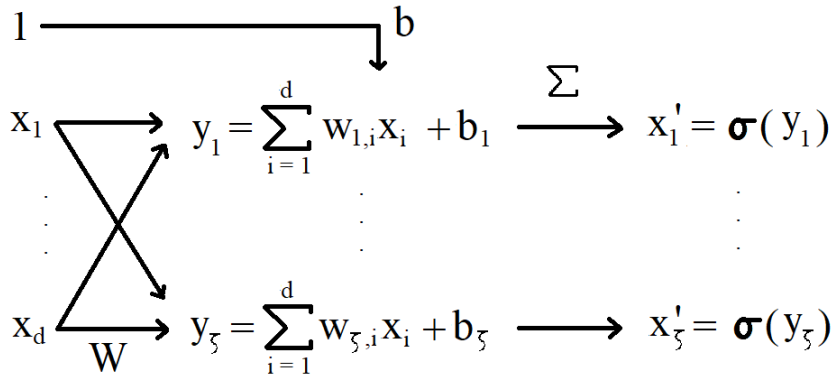

Figure 1.2: Illustration du fonctionnement de la couche de neurones

Remarque. *Pour tout $j \in [\![1, \zeta]\!]$, le vecteur $(w_{j,1}, \dots, w_{j,d}) \in \mathbb{R}^d$ est appelé* ***poids du neurone*** *j.*

1.2.2 Perceptron Multi-Couche (PMC)

Un PMC est un type de réseau de neurones utilisé dans le cadre du stage. Il se compose d'une succession de couches dans lesquelles la sortie de la j-ème couche correspond à l'entrée de la $(j+1)$-ème couche:

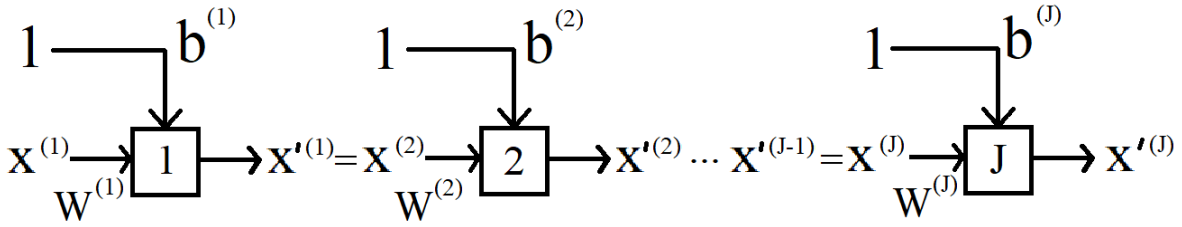

Figure 1.3: Illustration du fonctionnement d'un PMC comportant J couches de neurones, l'entrée, la sortie et le poids du biais associés à la couche j sont respectivement notés $x^{(j)}$, $x'^{(j)}$ et $b^{(j)}$

Dans le cadre du stage, en notant $W^{(j)}$ la matrice poids associée à la couche j, nous obtenons une fonction Loss qui dépendra des paramètres $W^{(1)}$,...,$W^{(J)}$,$b^{(1)}$,...,$b^{(J)}$. La stratégie est d'optimiser en ces paramètres.

1.3 Algorithme du gradient stochastique (SGD)

On cherche à minimiser la fonction en les paramètres du réseau de neurones, que l'on note W:

$$Loss_{Training}(W) \quad = \quad \frac{1}{K_0} \sum_{k=0}^{K_0-1} L(W,k)$$

où $L(W,k) = \frac{1}{h^2} \left| \hat{y_1^{(k)}} - y_1^{(k)} \right|^2$.

On se donne un réel $\alpha > 0$, appelé **taux d'apprentissage** (*learning rate* en anglais). L'algorithme du gradient stochastique est donné par la boucle suivante, et s'effectue en un nombre d'itérations que l'on note N_{iter}:

$(W)_0$ est choisi au hasard

Pour tout $t \in [\![0, N_{iter} - 1]\!]$:

- $i_t \sim \mathcal{U}([\![0, K_0 - 1]\!])$
- $(W)_{t+1} = (W)_t - \alpha \nabla L((W)_t, i_t)$

Cet algorithme présente l'avantage de ne pas avoir à calculer le gradient de $Loss_{Training}$ à chaque itération, ce qui serait coûteux, en particulier lorsque K_0 devient grand, ce qui est le cas lorsque nous avons beaucoup de données, tout en assurant de descendre dans suffisamment de directions afin de faire converger l'algorithme.

Remarques. - *Dans le cadre du stage, il a été observé que l'algorithme converge si les fonctions $L(\cdot, k)$ sont (strictement) convexes sur le domaine sur lequel se trouvent les $(W)_t$.*

- *Il existe des variantes de l'algorithme SGD. Par exemple, il est possible de calculer la moyenne de plusieurs gradients de fonctions $L(\cdot, k)$ à chaque itération (petits lots), ou bien de faire décroître le taux d'apprentissage au cours des itérations.*

Partie 2

Cas linéaire: Première formulation

2.1 Position du problème

Dans cette partie, on va se restreindre au cas linéaire, i.e. de la forme:

$$\begin{cases} \dot{y} = Ay \\ y(0) = y_0 \end{cases}$$

où $A \in \mathcal{M}_d(\mathbb{R})$

Par ailleurs, on considèrera que $N = 1$, c'est-à-dire que l'on connaît la solution du problème de Cauchy qu'en $t = 0$ et $t = h$, cette dernière étant donnée par: $y_1^{(k)} = e^{hA} y_0^{(k)}$.

Ainsi, on utilise un réseau neuronal comportant une couche de d neurones, sans biais, ce qui se traduit par une matrice poids $W \in \mathcal{M}_d(\mathbb{R})$. Ainsi, nous avons:

$$\hat{y_1^{(k)}} = A_h y_0^{(k0)}$$

où A_h est la matrice d'itération, qui dépend du schéma numérique employé:

- Pour la **méthode d'Euler Explicite**, on a:

$$A_h = I_d + hW$$

donnant ainsi:

$$L(W,k) = \frac{1}{h^2} \left| \left(I_d + hW - e^{hA} \right) y_0^{(k)} \right|^2$$

- Pour la **méthode d'Euler Implicite**, on a:

$$A_h = (I_d - hW)^{-1}$$

donnant ainsi:

$$L(W,k) = \frac{1}{h^2} \left| \left((I_d - hW)^{-1} - e^{hA} \right) y_0^{(k)} \right|^2$$

- Pour la **méthode du Point Milieu**, on a:

$$A_h = \left(I_d - \frac{h}{2}W\right)^{-1}\left(I_d + \frac{h}{2}W\right)$$

donnant ainsi:

$$L(W,k) = \frac{1}{h^2}\left|\left(\left(I_d - \frac{h}{2}W\right)^{-1}\left(I_d + \frac{h}{2}W\right) - e^{hA}\right)y_0^{(k)}\right|^2$$

- Pour la **méthode de Runge-Kutta 2**, on a:

$$A_h = I_d + hW + \frac{h^2}{2}W^2$$

donnant ainsi:

$$L(W,k) = \frac{1}{h^2}\left|\left(I_d + hW + \frac{h^2}{2}W^2 - e^{hA}\right)y_0^{(k)}\right|^2$$

Remarque. *Si on écrit une fonction Loss de la façon suivante:*

$$L(W,k) = \frac{1}{h^2}\left|My_0^{(k)}\right|^2$$

alors la condition pour que $Loss_{Training}(W) = 0 \Leftrightarrow M = 0$ *(minimisant ainsi la Loss) est d'avoir* d *données* $y_0^{(k)}$ *linéairement indépendantes. Ainsi, peu de données suffisent dans le cas linéaire.*

2.2 Méthode d'Euler Explicite

Proposition (Fonction *Loss* - Méthode d'Euler Explicite)

Soit $k \in [\![0, K-1]\!]$. La fonction

$$L(\cdot,k) : W \mapsto \frac{1}{h^2}\left|\left(I_d + hW - e^{hA}\right)y_0^{(k)}\right|^2$$

est convexe sur $\mathcal{M}_d(\mathbb{R})$ et est presque partout strictement convexe sur $\mathcal{M}_d(\mathbb{R})$.

Démonstration. *$L(\cdot,k)$ est une forme quadratique en W, donc est une application régulière. Tout d'abord, calculons la différentielle de $L(\cdot,k)$. Soient $W, H \in \mathcal{M}_d(\mathbb{R})$:*

$$\begin{aligned} L(W+H,k) &= \frac{1}{h^2}\left|\left(I_d + h(W+H) - e^{hA}\right)y_0^{(k)}\right|^2 \\ &= \frac{1}{h^2}\left|\left(I_d + hW - e^{hA}\right)y_0^{(k)}\right|^2 + \frac{2}{h}\langle\left(I_d + hW - e^{hA}\right)y_0^{(k)} | Hy_0^{(k)}\rangle + \left|Hy_0^{(k)}\right|^2 \end{aligned}$$

Donc, on retient le terme linéaire en H pour obtenir la différentielle (le terme restant est un terme quadratique):

$$dL(W,k) : H \mapsto \frac{2}{h}\langle \left(I_d + hW - e^{hA}\right) y_0^{(k)} | H y_0^{(k)} \rangle$$

- *Tout d'abord, montrons la convexité. Soient $W_1, W_2 \in \mathcal{M}_d(\mathbb{R})$:*

$$\begin{aligned}
L(W_1,k) + dL(W_1,k)\cdot(W_2 - W_1) &= \frac{1}{h^2}\left|\left(I_d + hW_1 - e^{hA}\right) y_0^{(k)}\right|^2 \\
&+ \frac{2}{h}\langle \left(I_d + hW_1 - e^{hA}\right) y_0^{(k)} | (W_2 - W_1) y_0^{(k)} \rangle \\
&\leqslant \frac{1}{h^2}\left|\left(I_d + hW_1 - e^{hA}\right) y_0^{(k)}\right|^2 \\
&+ \frac{2}{h}\langle \left(I_d + hW_1 - e^{hA}\right) y_0^{(k)} | (W_2 - W_1) y_0^{(k)} \rangle \\
&+ \left|(W_2 - W_1) y_0^{(k)}\right|^2 \\
&\leqslant \frac{1}{h^2}\left|\left(I_d + hW_1 - e^{hA}\right) y_0^{(k)} + h(W_2 - W_1) y_0^{(k)}\right|^2 \\
&\leqslant \frac{1}{h^2}\left|\left(I_d + hW_2 - e^{hA}\right) y_0^{(k)}\right|^2 \\
&\leqslant L(W_2,k)
\end{aligned}$$

Donc $L(\cdot,k)$ est convexe sur $\mathcal{M}_d(\mathbb{R})$.

- *Montrons la stricte convexité presque partout sur $\mathcal{M}_d(\mathbb{R})$.*

Cette propriété se montre en reprenant le raisonnement précédent, et en utilisant le fait que, pour presque tous $W_1, W_2 \in \mathcal{M}_d(\mathbb{R}), y_0^{(k)} \in \mathbb{R}^d$:

$$\left|(W_2 - W_1) y_0^{(k)}\right|^2 > 0$$

puisque les matrices non inversibles sont de mesure nulle dans $\mathcal{M}_d(\mathbb{R})$ (pour la mesure de Lebesgue), et que $y_0^{(k)} \neq 0$ presque partout sur $\mathbb{R}^d$. Ainsi, on a, pour presque tous $W_1, W_2 \in \mathcal{M}_d(\mathbb{R})$, $y_0^{(k)} \in \mathbb{R}^d$:

$$L(W_1,k) + dL(W_1,k)\cdot(W_2 - W_1) < L(W_2,k)$$

ce qui montre la stricte convexité presque partout sur $\mathcal{M}_d(\mathbb{R})$

■

La stricte convexité presque partout fournit l'existence d'au plus un minimiseur W_{Min} qui annule la fonction $L(\cdot,k)$, et à fortiori $Loss_{Training}$, cette matrice étant la matrice recherchée pour approximer A.

Proposition (Méthode d'Euler Explicite - Ordre de convergence)

L'apprentissage pour la méthode d'Euler Explicite est d'ordre 1:

$$W_{Min} \underset{h\to 0}{=} A + \mathcal{O}(h)$$

Démonstration. *On rappelle que:*

$$L(W,k) = \frac{1}{h^2}\left|\left(I_d + hW - e^{hA}\right) y_0^{(k)}\right|^2$$

On a:

$$W_{Min} = \frac{e^{hA} - I_d}{h}$$

On fait un développement limité à l'ordre 2:

$$\begin{aligned} W_{Min} &\underset{h\to 0}{=} \frac{1}{h}\left(I_d + hA + \mathcal{O}(h^2) - I_d\right) \\ W_{Min} &\underset{h\to 0}{=} A + \mathcal{O}(h) \end{aligned}$$

■

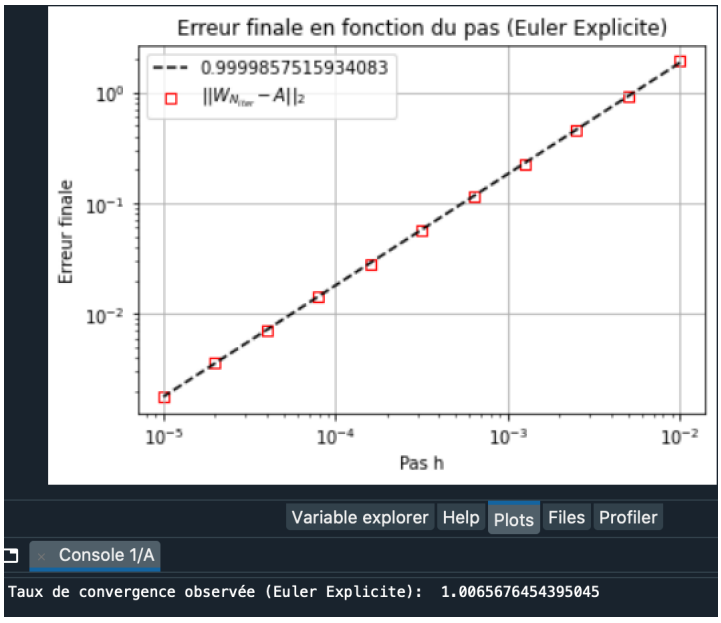

Figure 2.1: Courbe de convergence en échelle log-log pour la méthode d'Euler Explicite, obtenue avec une matrice A prise au hasard dans $\mathcal{M}_d([-10,10])$ dans le cas $d = 2$, $N_{iter} = 500$ et $\alpha = 10^{-3}$

2.3 Méthode d'Euler Implicite

Proposition (Fonction *Loss* - Méthode d'Euler Implicite en dimension 1)

Soit $R > 0$. Soient $k \in [\![0, K-1]\!]$ et $y_0^{(k)} \neq 0$. La fonction

$$L(\cdot, k) : W \mapsto \frac{1}{h^2}\left(\frac{1}{1-hW} - e^{hA}\right)^2 y_0^{(k)^2}$$

est strictement convexe sur $[-R, R]$ dès que h est assez petit.

Démonstration. *On a, pour tout* $W \in \mathbb{R}\backslash\left\{\frac{1}{h}\right\}$

$$L(W,k) = \frac{1}{h^2}\left(\frac{1}{1-hW} - e^{hA}\right)^2 y_0^{(k)^2}$$

Cette fonction est deux fois différentiable sur $\mathbb{R}\backslash\left\{\frac{1}{h}\right\}$ *et on a:*

$$\begin{aligned}\frac{\partial L}{\partial W}(W,k) &= \frac{2}{h(1-hW)^2}\left(\frac{1}{1-hW} - e^{hA}\right) y_0^{(k)^2} \\ \frac{\partial^2 L}{\partial W^2}(W,k) &= \frac{2}{(1-hW)^3}\left(\frac{2}{1-hW} - e^{hA}\right) y_0^{(k)^2}\end{aligned}$$

Prenons $h < \frac{1}{R}$. *Soit* $W \in [-R, R]$. *On a* $W \leqslant R < \frac{1}{h}$, *donc* $\frac{1}{1-hW} > 0$. *Donc* $L(\cdot, k)$ *est strictement convexe si et seulement si:*

$$\frac{2}{1-hW} - e^{hA} > 0$$

(stricte positivité de la dérivée seconde). Or, nous avons:

$$\begin{aligned}\frac{2}{1-hW} - e^{hA} > 0 &\Leftrightarrow 1 - hW < 2e^{-hA} \\ &\Leftrightarrow 2e^{-hA} + hW - 1 > 0\end{aligned}$$

Or, un développement limité en h *assure que:*

$$\begin{aligned}2e^{-hA} + hW - 1 &\underset{h\to 0}{=} 2(1-hA) + hW - 1 + \mathcal{O}(h^2) \\ &\underset{h\to 0}{=} 1 + h(W - 2A) + \mathcal{O}(h^2)\end{aligned}$$

Donc, si h *est assez petit, on a:*

$$2e^{-hA} + hW - 1 > 0$$

Donc $L(\cdot, k)$ *est strictement convexe sur* $[-R, R]$.

■

Étudions maintenant une matrice W_{Min} telle que $Loss_{Training}(W_{Min}) = 0$:

Proposition (Méthode d'Euler Implicite - Ordre de convergence)

L'apprentissage pour la méthode d'Euler Implicite est d'ordre 1:

$$W_{Min} \underset{h\to 0}{=} A + \mathcal{O}(h)$$

Démonstration. *On rappelle que:*

$$L(W,k) = \frac{1}{h^2}\left|\left((I_d - hW)^{-1} - e^{hA}\right) y_0^{(k)}\right|^2$$

On a comme unique possibilité pour W_{Min}:

$$W_{Min} = \frac{I_d - e^{-hA}}{h}$$

On fait un développement limité à l'ordre 2:

$$\begin{aligned} W_{Min} &\underset{h\to 0}{=} \frac{1}{h}\left(I_d - I_d + hA + \mathcal{O}(h^2)\right) \\ W_{Min} &\underset{h\to 0}{=} A + \mathcal{O}(h) \end{aligned}$$

■

Remarque. *En dimension 1, le point critique de $L(\cdot,k)$ vérifie $\frac{1}{1-hW} - e^{hA} = 0$. donc, au voisinage de ce point critique:*

$$\begin{aligned} \frac{\partial^2 L}{\partial W^2}(W,k) &= \frac{2{y_0^{(k)}}^2}{(1-hW)^3}\cdot\frac{1}{1-hW} \\ &= \frac{2{y_0^{(k)}}^2}{(1-hW)^4} \\ &> 0 \end{aligned}$$

donc on y a de la convexité stricte, la convergence vers ce point critique, minimiseur via la méthode du gradient y est garantie. De façon plus générale, en partant d'un point quelconque, la méthode de gradient va converger vers le point critique, et ce grâce à la convexité sur tout segment lorsque h est assez petit.

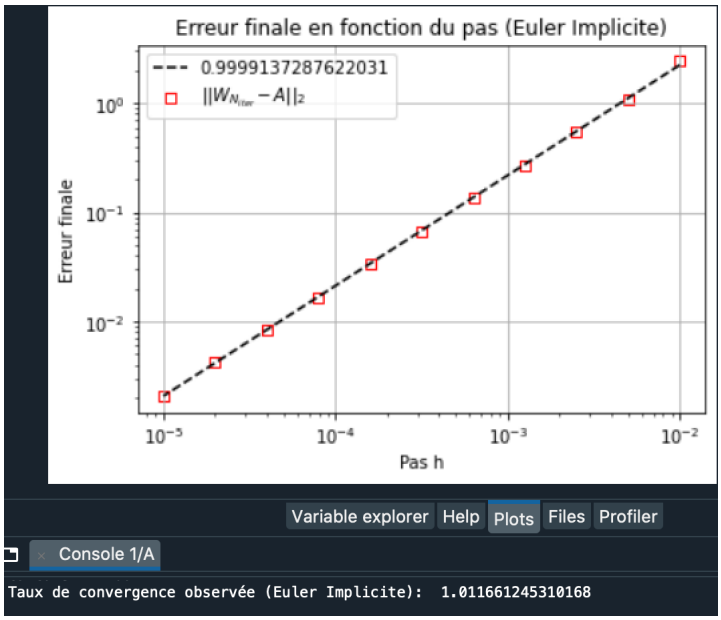

Figure 2.2: Courbe de convergence en échelle log-log pour la méthode d'Euler Implicite, obtenue avec une matrice A prise au hasard dans $\mathcal{M}_d([-10, 10])$ dans le cas $d = 2$, $N_{iter} = 500$ et $\alpha = 10^{-3}$

2.4 Méthode du Point Milieu

Proposition (Fonction *Loss* - Méthode du Point Milieu en dimension 1)

Soit $R > 0$. Soient $k \in [\![0, K-1]\!]$ et $y_0^{(k)} \neq 0$. La fonction

$$L(\cdot, k) : W \mapsto \frac{1}{h^2}\left(\frac{1+\frac{h}{2}W}{1-\frac{h}{2}W} - e^{hA}\right)^2 \left(y_0^{(k)}\right)^2$$

est strictement convexe sur $[-R, R]$ dès que h est assez petit.

Démonstration. *On a, pour tout* $W \in \mathbb{R}\setminus\left\{\frac{2}{h}\right\}$*:*

$$L(W,k) = \frac{1}{h^2}\left(\frac{1+\frac{h}{2}W}{1-\frac{h}{2}W} - e^{hA}\right)^2 \left(y_0^{(k)}\right)^2$$

Cette fonction est deux fois différentiable sur $\mathbb{R}\setminus\left\{\frac{2}{h}\right\}$ *et on a:*

$$\begin{aligned}
\frac{\partial L}{\partial W}(W,k) &= \frac{2}{h\left(1-\frac{h}{2}W\right)^2}\left(\frac{1+\frac{h}{2}W}{1-\frac{h}{2}W} - e^{hA}\right)\left(y_0^{(k)}\right)^2 \\
\frac{\partial^2 L}{\partial W^2}(W,k) &= \frac{2}{\left(1-\frac{h}{2}W\right)^2}\left(y_0^{(k)}\right)^2 + \frac{2}{\left(1-\frac{h}{2}W\right)^3}\left(\frac{1+\frac{h}{2}W}{1-\frac{h}{2}W} - e^{hA}\right)\left(y_0^{(k)}\right)^2
\end{aligned}$$

Pour tout $W \neq \frac{2}{h}$*:*

$$\frac{\partial^2 L}{\partial W^2}(W,k) = \frac{2}{\left(1-\frac{h}{2}W\right)^4}\left(\left(1-\frac{h}{2}W\right)^2+1+\frac{h}{2}W-\left(1-\frac{h}{2}W\right)e^{hA}\right)\left(y_0^{(k)}\right)^2$$

Si $h < \frac{2}{R}$, alors $W \leqslant R < \frac{2}{h}$, donc $\frac{1}{\left(1-\frac{h}{2}W\right)^4} > 0$. Donc $L(\cdot,k)$ est strictement convexe si et seulement si:

$$\left(1-\frac{h}{2}W\right)^2+1+\frac{h}{2}W-\left(1-\frac{h}{2}W\right)e^{hA} > 0$$

(stricte positivité de la dérivée seconde).

On fait un développement limité au voisinage de $h=0$:

$$\begin{aligned}
\left(1-\frac{h}{2}W\right)^2+1+\frac{h}{2}W-\left(1-\frac{h}{2}W\right)e^{hA} &\underset{h\to 0}{=} 1-hW+1+\frac{h}{2}W+\mathcal{O}(h^2) \\
&\quad - \left(1-\frac{h}{2}W\right)\left(1+hA+\mathcal{O}(h^2)\right) \\
\left(1-\frac{h}{2}W\right)^2+1+\frac{h}{2}W-\left(1-\frac{h}{2}W\right)e^{hA} &\underset{h\to 0}{=} 2-\frac{h}{2}W-1+\frac{h}{2}W-hA+\mathcal{O}(h^2) \\
\left(1-\frac{h}{2}W\right)^2+1+\frac{h}{2}W-\left(1-\frac{h}{2}W\right)e^{hA} &\underset{h\to 0}{=} 1-hA+\mathcal{O}(h^2)
\end{aligned}$$

Donc, si h est assez petit:

$$\left(1-\frac{h}{2}W\right)^2+1+\frac{h}{2}W-\left(1-\frac{h}{2}W\right)e^{hA} > 0$$

Donc $L(\cdot,k)$ est strictement convexe sur $[-R,R]$.

■

On s'intéresse à une matrice W_{Min} qui minimise $Loss_{Training}$ en l'annulant.

Proposition (Méthode du Point Milieu - Ordre de convergence)

L'apprentissage pour la méthode du Point Milieu est d'ordre 2:

$$W_{Min} \underset{h\to 0}{=} A+\mathcal{O}(h^2)$$

Démonstration. *On rappelle que:*

$$L(W,k) = \frac{1}{h^2}\left|\left(\left(I_d-\frac{h}{2}W\right)^{-1}\left(I_d+\frac{h}{2}W\right)-e^{hA}\right)y_0^{(k)}\right|^2$$

On a ainsi:

$$\left(I_d - \frac{h}{2}W_{Min}\right)^{-1}\left(I_d + \frac{h}{2}W_{Min}\right) \;=\; e^{hA}$$

On applique le logarithme matriciel:

$$hA \;=\; \log\left\{\left(I_d - \frac{h}{2}W_{Min}\right)^{-1}\left(I_d + \frac{h}{2}W_{Min}\right)\right\}$$

On utilise les propriétés du logarithme (valables pour les matrices):

$$A \;=\; \frac{1}{h}\left\{\log\left(I_d + \frac{h}{2}W_{Min}\right) - \log\left(I_d - \frac{h}{2}W_{Min}\right)\right\}$$

On fait un développement limité à l'ordre 3:

$$\begin{aligned} A &\underset{h\to 0}{=} \frac{1}{h}\left\{\frac{h}{2}W_{Min} - \frac{1}{2}\left(\frac{h}{2}W_{Min}\right)^2 + \mathcal{O}(h^3) - \left(-\frac{h}{2}W_{Min} - \frac{1}{2}\left(-\frac{h}{2}W_{Min}\right)^2 + \mathcal{O}(h^3)\right)\right\} \\ &\underset{h\to 0}{=} \frac{1}{h}\left\{hW_{Min} + \mathcal{O}(h^3)\right\} \\ &\underset{h\to 0}{=} W_{Min} + \mathcal{O}(h^2) \end{aligned}$$

Donc on a:

$$W_{Min} \underset{h\to 0}{=} A + \mathcal{O}(h^2)$$

■

Remarque. *En dimension 1, le point critique de $L(\cdot, k)$ vérifie $\frac{1+\frac{h}{2}W}{1-\frac{h}{2}W} - e^{hA} = 0$. donc, au voisinage de ce point critique:*

$$\begin{aligned} \frac{\partial^2 L}{\partial W^2}(W,k) &= \frac{2y_0^{(k)^2}}{\left(1-\frac{h}{2}W\right)^2} \\ &> 0 \end{aligned}$$

Donc on y a de la stricte convexité, ce qui garantit la convergence de la méthode de gradient vers ce point critique. De façon plus générale, en partant d'un point quelconque, la méthode de gradient va converger vers le point critique, et ce grâce à la convexité sur tout segment lorsque h est assez petit.

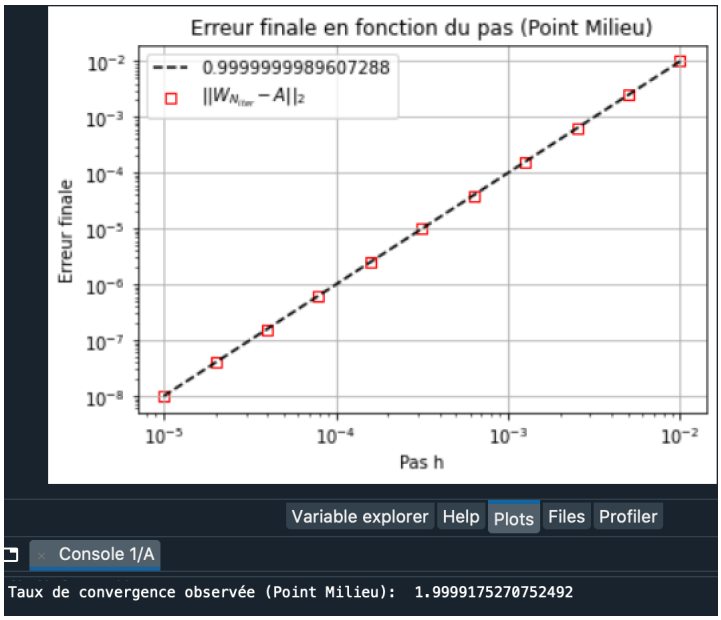

Figure 2.3: Courbe de convergence en échelle log-log pour la méthode du Point Milieu, obtenue avec une matrice A prise au hasard dans $\mathcal{M}_d([-10, 10])$ dans le cas $d = 2$, $N_{iter} = 500$ et $\alpha = 10^{-3}$

2.5 Méthode de Rung-Kutta 2

Proposition (Fonction *Loss* - Méthode de Runge-Kutta 2 en dimension 1)

Soient $k \in [\![0, K-1]\!]$ et $y_0^{(k)} \neq 0$. La fonction

$$L(\cdot, k) : W \mapsto \frac{1}{h^2}\left(1 + hW + \frac{h^2}{2}W^2 - e^{hA}\right)^2 y_0^{(k)2}$$

est strictement convexe sur $\mathbb{R}$ si et seulement si $A < 0$ et $h \geqslant \frac{1}{|A|}\log(2)$

Démonstration. *On a, pour tout $W \in \mathbb{R}$:*

$$L(W, k) = \frac{1}{h^2}\left(1 + hW + \frac{h^2}{2}W^2 - e^{hA}\right)^2 \left(y_0^{(k)}\right)^2$$

Cette fonction est deux fois différentiable sur $\mathbb{R}$ et on a:

$$\begin{aligned}
\frac{\partial L}{\partial W}(W, k) &= \frac{2}{h}(1 + hW)\left(1 + hW + \frac{h^2}{2}W^2 - e^{hA}\right)\left(y_0^{(k)}\right)^2 \\
\frac{\partial^2 L}{\partial W^2}(W, k) &= 2\left(2 - e^{hA} + 3hW + \frac{3}{2}h^2W^2\right)\left(y_0^{(k)}\right)^2 \\
&= 2\left[\frac{3}{2}(1 + hW)^2 + \frac{1}{2} - e^{hA}\right]\left(y_0^{(k)}\right)^2
\end{aligned}$$

Une condition nécéssaire et suffisante de stricte convexité est que, pour tout $W \in \mathbb{R}$, $\frac{\partial^2 L}{\partial W^2}(W,k) > 0$, *i.e.* $\Delta < 0$, *où* Δ *est le discriminant associé à la fonction polynomiale de degré 2* $\frac{\partial^2 L}{\partial W^2}(\cdot,k)$. Δ *est donné par:*

$$\Delta = 12h^2 y_0^{(k)^4}\left(2e^{hA}-1\right)$$

Donc:

$$\begin{aligned}\frac{\partial^2 L}{\partial W^2}(\cdot,k) > 0 \text{ sur } \mathbb{R} &\Leftrightarrow \Delta < 0\\ &\Leftrightarrow 2e^{hA}-1<0\\ &\Leftrightarrow hA < -\log(2)\end{aligned}$$

- *Si* $A \geqslant 0$, *c'est impossible*
- *Si* $A<0$, $h \geqslant \frac{\log(2)}{|A|}$

■

Remarque. *Ce résultat nécéssite de ne pas prendre* h *trop petit et on n'a pas de convexité globale quand* $A \geqslant 0$

Proposition (Fonction *Loss* - Méthode de Runge-Kutta 2 en dimension 1)

Soit $R>0$. Soient $k \in [\![0, K-1]\!]$ et $y_0^{(k)} \neq 0$. Si on a:

$$h < \min\left\{\frac{1}{R}\left(1-\frac{1}{\sqrt{2}}\right), \frac{1}{|A|}\log\left(\frac{5}{4}\right)\right\}$$

alors la fonction:

$$L(\cdot,k) : W \mapsto \frac{1}{h^2}\left(1+hW+\frac{h^2}{2}W^2-e^{hA}\right)^2\left(y_0^{(k)}\right)^2$$

est strictement convexe sur $[-R,+\infty[$.

Démonstration. - *Si* $h < \frac{1}{|A|}\log\left(\frac{5}{4}\right)$ *donc* $e^{hA} \leqslant e^{h|A|} < \frac{5}{4}$, *soit* $\frac{1}{2} - e^{hA} > -\frac{3}{4}$

- *De plus, si* $h \leqslant \frac{1}{R}\left(1-\frac{1}{\sqrt{2}}\right)$, *alors, pour tout* $W \in [-R,0[$:

$$h \leqslant \frac{1}{|W|}\left(1-\frac{1}{\sqrt{2}}\right) = \frac{1}{W}\left(\frac{1}{\sqrt{2}}-1\right)$$

$$\begin{aligned}h &\leqslant \frac{1}{|W|}\left(1-\frac{1}{\sqrt{2}}\right)\\ h &\leqslant \frac{1}{W}\left(\frac{1}{\sqrt{2}}-1\right)\end{aligned}$$

Donc $hW \geqslant \frac{1}{\sqrt{2}} - 1$ *i.e.* $1 + hW \geqslant \frac{1}{\sqrt{2}}$. *Ainsi, on a:*

$$(1 + hW)^2 \quad \geqslant \quad \frac{1}{2}$$

Donc, pour tout $W \in [-R, 0[$*:*

$$\frac{\partial^2 L}{\partial W^2}(W, k) = 2\left[\frac{3}{2}\underbrace{(1 + hW)^2}_{\geqslant \frac{1}{2}} + \underbrace{\frac{1}{2} - e^{hA}}_{> -\frac{3}{4}}\right] > 0$$

Donc $L(\cdot, k)$ *est strictement convexe sur* $[-R, 0[$.

- *Enfin,* $h < \frac{1}{|A|}\log\left(\frac{5}{4}\right) \leqslant \frac{1}{|A|}\log(2)$ *donc* $hA \leqslant h|A| < \log(2)$ *i.e.* $e^{hA} < 2$ *soit* $\frac{1}{2} - e^{hA} \geqslant -\frac{3}{2}$.

  Donc, si $W \geqslant 0$*:*

$$\frac{\partial^2 L}{\partial W^2}(W, k) = 2\left[\frac{3}{2}\underbrace{(1 + hW)^2}_{\geqslant 1} + \underbrace{\frac{1}{2} - e^{hA}}_{> -\frac{3}{2}}\right] > 0$$

Donc $L(\cdot, k)$ *est strictement convexe sur* $[0, +\infty[$.

Donc $L(\cdot, k)$ *est strictement convexe sur* $[-R, +\infty[$.

■

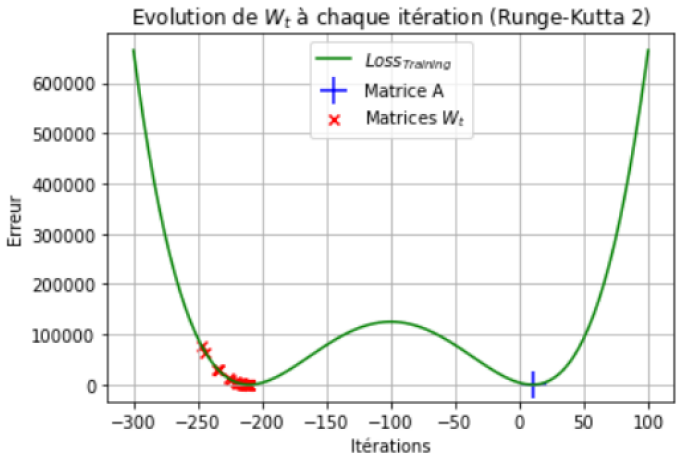

Figure 2.4: Lorsque h n'est pas assez petit, l'algorithme du gradient stochastique donne la convergence vers le mauvais minimiseur de la *Loss*

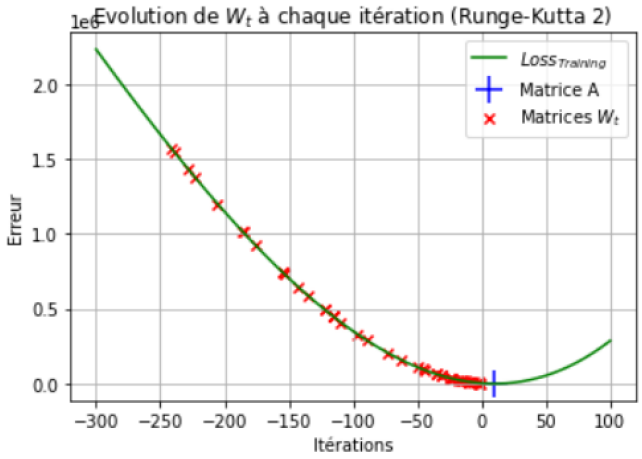

Figure 2.5: Lorsque h est suffisamment petit, l'algorithme du gradient stochastique donne la convergence vers le bon minimiseur de la *Loss*, et ce grâce à la convexité.

Proposition (Fonction *Loss* - Méthode de Runge-Kutta 2 en dimension supérieure)

Soit $R > 0$. Soient $k \in [\![0, K-1]\!]$ et $y_0^{(k)} \neq 0$. La fonction

$$L(\cdot, k) : W \mapsto \frac{1}{h^2} \left| I_d + hW + \frac{h^2}{2} W^2 - e^{hA} \right|^2 y_0^{(k)^2}$$

est convexe sur $\mathbb{B}_{||\cdot||}(0, R)$ où $||\cdot||$ est une norme matricielle quelconque.

Démonstration. *On a, pour tout $W \in \mathcal{M}_d(\mathbb{R})$:*

$$\begin{aligned} L(W,k) &= \frac{1}{h^2} y_0^{(k)^T} \left(I_d + hW + \frac{h^2}{2} W^2 - e^{hA} \right)^T \left(I_d + hW + \frac{h^2}{2} W^2 - e^{hA} \right) y_0^{(k)} \\ &\underset{h\to 0}{=} \frac{1}{h^2} y_0^{(k)^T} \left(h(W-A) + \mathcal{O}(h^2) \right)^T \left(h(W-A) + \mathcal{O}(h^2) \right) y_0^{(k)} \\ &\underset{h\to 0}{=} \left| (W-A) y_0^{(k)} \right|^2 + \mathcal{O}(h) \end{aligned}$$

Donc, si h est assez petit, $L(\cdot, k)$ ressemble à une forme quadratique en $W \to 0_d$, donc on a de la convexité autour de cette matrice.

■

On s'intéresse à une matrice W_{Min} qui minimise $Loss_{Training}$ en l'annulant.

Proposition (Méthode de Runge-Kutta 2 - Ordre de convergence)

L'apprentissage pour la méthode de Runge-Kutta 2 est d'ordre 2:

$$W_{Min} \underset{h\to 0}{=} A + \mathcal{O}(h^2)$$

Démonstration. *On rappelle que:*

$$L(W,k) = \frac{1}{h^2}\left|\left(I_d + hW + \frac{h^2}{2}W^2 - e^{hA}\right) y_0^{(k)}\right|^2$$

On a ainsi:

$$I_d + hW_{Min} + \frac{h^2}{2}W_{Min}^2 = e^{hA}$$

On applique le logarithme matriciel:

$$\begin{aligned} hA &= \log\left\{I_d + hW_{Min} + \frac{h^2}{2}W_{Min}^2\right\} \\ A &= \frac{1}{h}\log\left\{I_d + hW_{Min} + \frac{h^2}{2}W_{Min}^2\right\} \end{aligned}$$

On fait un développement limité à l'ordre 3:

$$\begin{aligned} A &\underset{h\to 0}{=} \frac{1}{h}\left\{hW_{Min} + \frac{h^2}{2}W_{Min}^2 - \frac{1}{2}\left(hW_{Min} + \frac{h^2}{2}W_{Min}^2\right)^2 + \mathcal{O}(h^3)\right\} \\ &\underset{h\to 0}{=} \frac{1}{h}\left\{hW_{Min} + \frac{h^2}{2}W_{Min}^2 - \frac{h^2}{2}W_{Min}^2 - \frac{h^3}{2}W_{Min}^3 - \frac{h^4}{8}W_{Min}^4 + \mathcal{O}(h^3)\right\} \\ &\underset{h\to 0}{=} \frac{1}{h}\left\{hW_{Min} + \mathcal{O}(h^3)\right\} \\ &\underset{h\to 0}{=} W_{Min} + \mathcal{O}(h^2) \end{aligned}$$

Donc on a:

$$W_{Min} \underset{h\to 0}{=} A + \mathcal{O}(h^2)$$

■

Remarque. *En dimension 1, le point critique de $L(\cdot,k)$ vérifie $1+hW+\frac{1}{2}h^2W^2-e^{hA}=0$ i.e. $\frac{1}{2}\left[1+(1+hW)^2\right]=e^{hA}$ donc, au voisinage de ce point critique:*

$$\frac{\partial^2 L}{\partial W^2}(W,k) = 2{y_0^{(k)}}^2(1+hW)^2 > 0$$

Donc on y a de la stricte convexité, ce qui garantit la convergence de la méthode de gradient vers ce point critique. De façon plus générale, en partant d'un point quelconque, la méthode de gradient va converger vers le point critique, et ce grâce à la convexité sur tout segment lorsque h est assez petit.

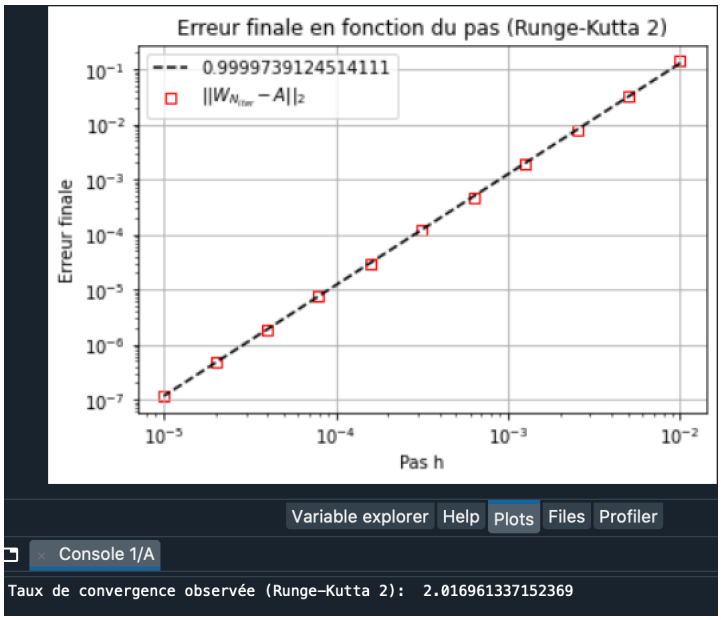

Figure 2.6: Courbe de convergence en échelle log-log pour la méthode de Runge-Kutta 2, obtenue avec une matrice A prise au hasard dans $\mathcal{M}_d([-10, 10])$ dans le cas $d = 2$, $N_{iter} = 500$ et $\alpha = 10^{-3}$

Partie 3

Cas linéaire: Nouvelle formulation et méthodes supplémentaires

3.1 Position du problème

Dans cette partie, on va toujours se restreindre au cas linéaire:

$$\begin{cases} \dot{y} = Ay \\ y(0) = y_0 \end{cases}$$

où $A \in \mathcal{M}_d(\mathbb{R})$

En revanche, on considèrera que $N \geqslant 1$ quelconque, c'est-à-dire que l'on connaît la solution du problème de Cauchy en des temps $0 = t_0 < t_1 < \ldots < t_N = h$, i.e. on connaît la solution en des points intermédiaires, cette dernière étant donnée par: $y^{(k)}_{\frac{t_j}{h}} = e^{t_j A} y_0^{(k)}$.

Ainsi, on utilise toujours un réseau neuronal comportant une couche de d neurones, sans biais, ce qui se traduit par une matrice poids $W \in \mathcal{M}_d(\mathbb{R})$. Cependant, nous allons considérer ces formes pour le vecteur $\hat{y_1}^{(k)}$:

$$\hat{y_1}^{(k)} = \sum_{j=0}^{N} \alpha_j y^{(k)}_{\frac{t_j}{h}} + W \sum_{j=0}^{N} \beta_j h_j y^{(k)}_{\frac{t_j}{h}} \tag{3.1}$$

avec $h_0 + \cdots + h_n = h$. Ainsi, la fonction $L(\cdot, k)$ est donnée, pour tout $W \in \mathcal{M}_d(\mathbb{R})$ par:

$$L(W,k) = \frac{1}{h^2} \left| \sum_{j=0}^{N} \alpha_j y^{(k)}_{\frac{t_j}{h}} - y_1^{(k)} + W \sum_{j=0}^{N} \beta_j h_j y^{(k)}_{\frac{t_j}{h}} \right|^2$$

- Pour la **méthode d'Euler Explicite**, on a:

$$\hat{y_1}^{(k)} = y_0^{(k)} + hW y_0^{(k)}$$

donnant ainsi:

$$L(W,k) = \frac{1}{h^2} \left| \left(I_d + hW - e^{hA}\right) y_0^{(k)} \right|^2$$

- Pour la **méthode d'Euler Implicite**, on a:

$$\hat{y_1}^{(k)} = y_0^{(k)} + hWy_1^{(k)}$$

donnant ainsi:

$$L(W,k) = \frac{1}{h^2}\left|\left((I_d - hW)e^{hA} - I_d\right)y_0^{(k)}\right|^2$$

- Pour la **méthode du Point Milieu**, on a:

$$\hat{y_1}^{(k)} = y_0^{(k)} + \frac{h}{2}W\left(y_0^{(k)} + y_1^{(k)}\right)$$

donnant ainsi:

$$L(W,k) = \frac{1}{h^2}\left|\left(e^{hA} - I_d - \frac{h}{2}W(I_d + e^{hA})\right)y_0^{(k)}\right|^2$$

3.2 Convexité des fonctions *Loss*

L'écriture de $y_1^{(k)}$ sous la forme (3.1) permet d'obtenir une fonction $L(\cdot,k)$ sous la forme d'une forme quadratique en W:

$$L(W,k) = \frac{1}{h^2}\left|u^{(k)} + Wv^{(k)}\right|^2$$

Proposition (Fonction *Loss* - Forme quadratique en W)

Soit $k \in [\![0, K-1]\!]$. La fonction

$$L(\cdot,k) : W \mapsto \frac{1}{h^2}\left|u^{(k)} + Wv^{(k)}\right|^2$$

est convexe sur $\mathcal{M}_d(\mathbb{R})$ et est presque partout strictement convexe sur $\mathcal{M}_d(\mathbb{R})$.

Démonstration. *Cette propriété se montre de la même manière que pour la fonction Loss associée à la méthode d'Euler Explicite dans la partie précédente, en utilisant la différentielle de la fonction.*

■

Remarque. *Dans la plupart des cas rencontrés, la fonction Loss s'écrit sous la forme:*

$$L(W,k) = \frac{1}{h^2}\left|By_0^{(k)} + WCy_0^{(k)}\right|^2$$

où $B, C \in \mathcal{M}_d(\mathbb{R})$. Ainsi, la propriété précédente de stricte convexité donne l'existence d'au plus un minimiseur, que l'on peut expliciter, dans le cas où la matrice C est inversible, par:

$$W_{Min} \quad = \quad -BC^{-1}$$

car ce minimiseur annule la fonction $L(\cdot, k)$.

3.3 Méthodes à deux points

Dans cette sous-partie, on suppose que seules $y_0^{(k)}$ et $y_1^{(k)}$ sont données ($N = 1$). On peut en déduire les trois méthodes principales: Euler Explicite, Euler Implicite et Point Milieu.

Proposition (Méthodes d'Euler et du Point Milieu - Ordres de convergence)

- L'apprentissage pour la méthode d'Euler Explicite est d'ordre 1:

$$W_{Min} \underset{h\to 0}{=} A + \mathcal{O}(h)$$

- L'apprentissage pour la méthode d'Euler Implicite est d'ordre 1:

$$W_{Min} \underset{h\to 0}{=} A + \mathcal{O}(h)$$

- L'apprentissage pour la méthode du Point Milieu est d'ordre 2:

$$W_{Min} \underset{h\to 0}{=} A + \mathcal{O}(h^2)$$

Démonstration. - *Pour la méthode d'Euler Explicite, on raisonne comme dans la proposition de la partie précédente.*

- *Pour la méthode d'Euler Implicite, même si la Loss est différente, on a toujours:*

$$W_{Min} \quad = \quad \frac{I_d - e^{hA}}{h}$$

et un développement limité à l'ordre 2 donne le résultat souhaité, comme dans la partie précédente

- *Pour la méthode du Point Milieu, on a:*

$$W_{Min} \quad = \quad \frac{2}{h}(I_d + e^{hA})^{-1}(e^{hA} - I_d)$$

qui annule $L(\cdot, k)$. On fait un développement limité à l'ordre 3:

$$
\begin{aligned}
W_{min} &\underset{h\to 0}{=} \frac{2}{h}\left(2I_d + hA + \frac{h^2}{2}A^2 + \mathcal{O}(h^3)\right)^{-1}\left(hA + \frac{h^2}{2}A^2 + \mathcal{O}(h^3)\right) \\
&\underset{h\to 0}{=} \frac{1}{h}\left(I_d + \frac{h}{2}A + \frac{h^2}{4}A^2 + \mathcal{O}(h^3)\right)^{-1}\left(hA + \frac{h^2}{2}A^2 + \mathcal{O}(h^3)\right) \\
&\underset{h\to 0}{=} \frac{1}{h}\left[I_d - \frac{h}{2}A - \frac{h^2}{4}A^2 + \left(\frac{h}{2}A + \frac{h^2}{4}A^2\right)^2 + \mathcal{O}(h^3)\right]\left(hA + \frac{h^2}{2}A^2 + \mathcal{O}(h^3)\right) \\
&\underset{h\to 0}{=} \frac{1}{h}\left(I_d - \frac{h}{2}A + \mathcal{O}(h^3)\right)\left(hA + \frac{h^2}{2}A^2 + \mathcal{O}(h^3)\right) \\
&\underset{h\to 0}{=} \frac{1}{h}\left(hA - \frac{h^2}{2}A^2 + \frac{h^2}{2}A^2 + \mathcal{O}(h^3)\right) \\
&\underset{h\to 0}{=} \frac{1}{h}\left(hA + \mathcal{O}(h^3)\right) \\
&\underset{h\to 0}{=} A + \mathcal{O}(h^2)
\end{aligned}
$$

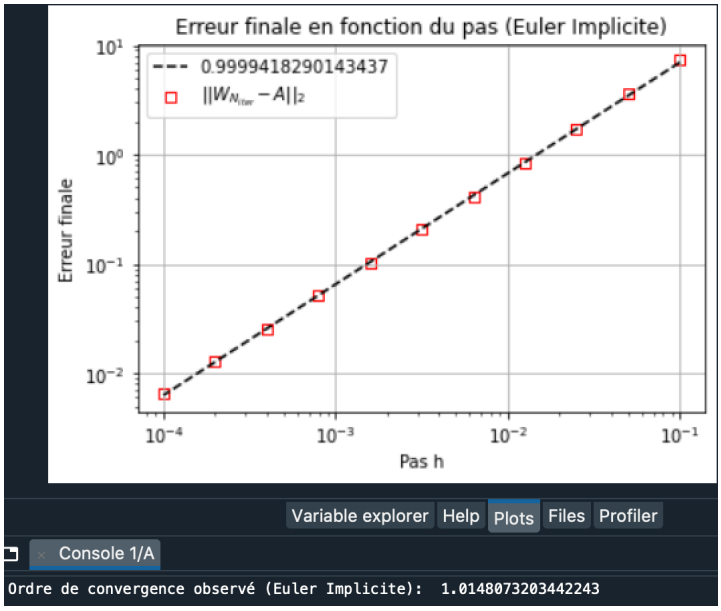

Figure 3.1: Courbe de convergence en échelle log-log pour la méthode d'Euler Implicite (*Loss* convexe), obtenue avec une matrice A prise au hasard dans $\mathcal{M}_d([-10, 10])$ dans le cas $d = 2$, $N_{iter} = 500$ et $\alpha = 10^{-3}$

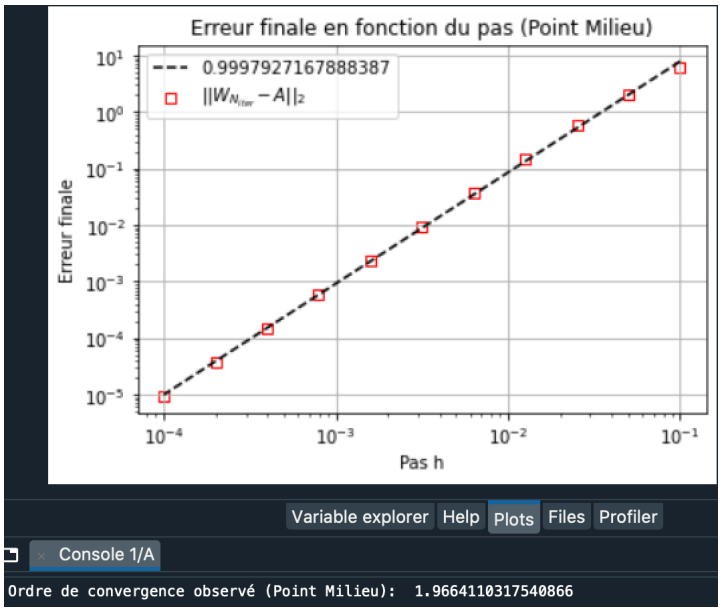

Figure 3.2: Courbe de convergence en échelle log-log pour la méthode du Point Milieu (*Loss* convexe), obtenue avec une matrice A prise au hasard dans $\mathcal{M}_d([-10, 10])$ dans le cas $d = 2$, $N_{iter} = 500$ et $\alpha = 10^{-3}$

La proposition suivante donne l'odre maximal que l'on peut atteindre avec la forme (3.1) pour $\hat{y_1}^{(k)}$:

Proposition (Ordre de convergence maximal avec deux points)

Si on prend $\hat{y_1}^{(k)}$ de la forme:

$$\hat{y_1}^{(k)} = \alpha_0 y_0^{(k)} + \alpha_1 y_1^{(k)} + hW\left(\beta_0 y_0^{(k)} + \beta_1 y_1^{(k)}\right)$$

Alors, l'apprentissage est, au mieux, d'ordre 2.

Démonstration. *On a, pour tout* $k \in [\![0, K-1]\!]$*:*

$$L(W,k) = \frac{1}{h^2}\left|\left[\alpha_0 I_d + (\alpha_1 - 1)e^{hA} + hW\left(\beta_0 I_d + \beta_1 e^{hA}\right)\right] y_0^{(k)}\right|^2$$

On a ainsi:

$$\begin{aligned} W_{Min} &= -\frac{1}{h}\left(\beta_0 I_d + \beta_1 e^{hA}\right)^{-1}\left(\alpha_0 I_d + (\alpha_1 - 1)e^{hA}\right) \\ &\underset{h\to 0}{\sim} -\frac{1}{h}\left[(\beta_0+\beta_1)I_d + \beta_1 hA\right]^{-1}\left[(\alpha_0+\alpha_1-1)I_d + (\alpha_1-1)hA\right] \end{aligned} \tag{3.2}$$

On doit avoir $\alpha_0 + \alpha_1 = 1$ *pour que le troisième facteur de* (3.2) *soit en* $\mathcal{O}(h)$ *et "compense" le premier facteur en* $\frac{1}{h}$*. on a ainsi:*

$$W_{Min} \underset{h\to 0}{\sim} \frac{\alpha_0}{\beta_0+\beta_1}\left[I_d - \frac{\beta_1}{\beta_0+\beta_1}hA + \mathcal{O}(h^2)\right] A \underset{h\to 0}{\sim} \frac{\alpha_0}{\beta_0+\beta_1}A$$

On doit avoir $\alpha_0 = \beta_0 + \beta_1$. L'objectif est de monter en ordre. Comme on a:

$$\left\{ \begin{array}{rcl} \alpha_0 + \alpha_1 & = & 1 \\ \beta_0 + \beta_1 & = & \alpha_0 \end{array} \right.$$

On obtient:

$$W_{Min} = -\frac{1}{h}\left[\beta_0 I_d + \beta_1 e^{hA}\right]^{-1}\left[\alpha_0 I_d - \alpha_0 e^{hA}\right]$$

On fait un développement limité à l'ordre 4:

$$\begin{array}{rcl} W_{Min} & \underset{h\to 0}{=} & \left[(\beta_0+\beta_1)I_d + \beta_1 hA + \beta_1\frac{h^2}{2}A^2 + \beta_1\frac{h^3}{6}A^3 + \mathcal{O}(h^4)\right]^{-1} \\ & \cdot & \alpha_0\left[A + \frac{h}{2}A^2 + \frac{h^2}{6}A^3 + \frac{h^3}{24}A^4 + \mathcal{O}(h^4)\right] \end{array}$$

En posant $b = \frac{\alpha_0}{\beta_0+\beta_1}$, on a:

$$\begin{array}{rcl} W_{Min} & \underset{h\to 0}{=} & \frac{\alpha_0}{\beta_0+\beta_1}\left[I_d + b\left(hA + \frac{h^2}{2}A^2 + \frac{h^3}{6}A^3 + \mathcal{O}(h^4)\right)\right]^{-1} \\ & \cdot & \left[A + \beta_1\frac{h}{2}A^2 + \frac{h^2}{6}A^3 + \frac{h^3}{24}A^4 + \mathcal{O}(h^4)\right] \\ W_{Min} & \underset{h\to 0}{=} & \left[I_d + b\left(hA + \frac{h^2}{2}A^2 + \frac{h^3}{6}A^3 + \mathcal{O}(h^4)\right)\right]^{-1} \\ & \cdot & \left[A + \frac{h}{2}A^2 + \frac{h^2}{6}A^3 + \frac{h^3}{24}A^4 + \mathcal{O}(h^4)\right] \end{array}$$

En écrivant W_{Min} sous forme d'une série de Taylor, on a, à l'ordre 4:

$$\begin{array}{rcl} W_{min} & \underset{h\to 0}{=} & A + \left(\frac{1}{2} - b\right)hA^2 + \left(b^2 - b + \frac{1}{6}\right)h^2A^3 \\ & + & \left(-b^3 + \frac{3}{2}b^2 - \frac{7}{12}b + \frac{1}{24}\right)h^3A^4 + \mathcal{O}(h^4) \end{array}$$

Si on veut de l'ordre 2, on doit avoir $b = \frac{1}{2}$, et les conditions pour avoir de l'ordre 2 sont:

$$\left\{ \begin{array}{rcl} \alpha_0 + \alpha_1 & = & 1 \\ \beta_0 + \beta_1 & = & \alpha_0 \\ \beta_0 & = & \beta_1 \end{array} \right. \Leftrightarrow \left\{ \begin{array}{rcl} \alpha_0 & = & 2\beta_0 \\ \alpha_0 + \beta_1 & = & \alpha_0 \\ \beta_1 & = & \beta_0 \end{array} \right.$$

Ce qui correspond à la méthode du Point Milieu.

En plus de ces conditions, si on veut avoir de l'ordre 3, on doit avoir:

$$b^2 - b + \frac{1}{6} = 0$$

ce qui est impossible avec $b = \frac{1}{2}$, donc on ne peut pas monter en ordre.

■

Remarques. - *si* $\alpha_0 + \alpha_1 - 1 = \alpha_1 - 1 = 0$, *alors* $(\alpha_0, \alpha_1) = (0, 1)$*:*

$$\hat{y_1}^{(k)} = y_1^{(k)} + hW\left(\beta_0 y_0^{(k)} + \beta_1 y_1^{(k)}\right)$$

$$L(W,k) = \frac{1}{h^2}\left|hW\left(\beta_0 y_0^{(k)} + \beta_1 y_1^{(k)}\right)\right|^2$$

Donc $W_{Min} = 0$*, ce qui n'approche pas forcément* A

- *Si* $\beta_0 + \beta_1 = 0$ *et* $\alpha_0 + \alpha_1 = 1$*, alors, dans l'expression* (3.2)*, on a:*

$$\begin{aligned} W_{Min} &\underset{h\to 0}{=} \frac{1-\alpha_1}{h\beta_1} I_d + \mathcal{O}(1) \text{ si } \beta_1 \neq 0 \\ &\underset{h\to 0}{=} \frac{\alpha_0}{h\beta_1} I_d + \mathcal{O}(1) \end{aligned}$$

Cela impose $\alpha_0 = 0$*, donc* $\alpha_1 = 1$*, et on se ramène au cas précédent, et si* $\beta_1 = 0$*, alors* $\beta_0 = 0$ *et* $\hat{y_1}^{(k)}$ *ne dépend pas de* W*.*

3.4 Méthodes Multi-pas (explicites) à pas constant

3.4.1 Ordre de convergence

On suppose que, pour tout $j \in [\![0, N]\!]$, $t_j = \frac{jh}{N}$, et que la solution y est connue en les points t_j, i.e. pour tout $k \in [\![0, K-1]\!]$:

$$y_{\frac{j}{N}}^{(k)} = e^{\frac{j}{N}A} y_0^{(k)}$$

Posons alors:

$$\hat{y_1^{(k)}} = \sum_{i=0}^{N-1} \alpha_i y_{1-\frac{i+1}{N}}^{(k)} + \frac{h}{N} W \sum_{i=0}^{N-1} \beta_i y_{1-\frac{i+1}{N}}^{(k)}$$

On obtient ainsi:

$$L(W,k) = \frac{1}{h^2}\left|\sum_{i=0}^{N-1} \alpha_i y_{1-\frac{i+1}{N}}^{(k)} - y_1^{(k)} + \frac{h}{N} W \sum_{i=0}^{N-1} \beta_i y_{1-\frac{i+1}{N}}^{(k)}\right|^2$$

et, comme minimiseur:

$$W_{Min} = -\frac{N}{h}\left(\sum_{i=0}^{N-1} \beta_i e^{\frac{N-i-1}{N}hA}\right)^{-1}\left(\sum_{i=0}^{N-1} \alpha_i e^{\frac{N-i-1}{N}hA} - e^{hA}\right)$$

Proposition (Méthode Multi-pas à pas constant - Ordre de convergence)

Si on a:

-
$$\sum_{i=0}^{N-1} \alpha_i = 1$$

- Pour tout $q \in [\![1, p]\!]$:

$$\sum_{i=0}^{N-1} i^q \alpha_i - q i^{q-1} \beta_i \quad = \quad (-1)^q$$

Alors:

$$W_{Min} \underset{h\to 0}{=} A + \mathcal{O}(h^p)$$

Démonstration. *On a:*

$$\begin{aligned} W_{Min} &= -\frac{N}{h}\left(\sum_{i=0}^{N-1} \beta_i e^{\frac{N-i-1}{N}hA}\right)^{-1}\left(\sum_{i=0}^{N-1} \alpha_i e^{\frac{-i}{N}hA} - e^{hA}\right) \\ W_{Min} &= -\frac{N}{h}\left(e^{\frac{N-1}{N}hA}\sum_{i=0}^{N-1} \beta_i e^{-\frac{i}{N}hA}\right)^{-1}\left(e^{\frac{N-1}{N}hA}\left\{\sum_{i=0}^{N-1} \alpha_i e^{-\frac{i}{N}hA} - e^{\frac{h}{N}A}\right\}\right) \end{aligned}$$

Les matrices commutent (polynômes en A), donc on peut simplifier par $e^{\frac{N-1}{N}hA}$:

$$W_{Min} = -\frac{N}{h}\left(\sum_{i=0}^{N-1} \beta_i e^{\frac{N-i-1}{N}hA}\right)^{-1}\left(\sum_{i=0}^{N-1} \alpha_i e^{\frac{-i}{N}hA} - e^{\frac{h}{N}A}\right)$$

On fait un développement limité à l'ordre p du numérateur et du dénominateur:

$$\begin{aligned} W_{Min} \underset{h\to 0}{=} & -\frac{N}{h}\left(\sum_{i=0}^{N-1}\beta_i \sum_{q=0}^{p}\left(-\frac{i}{N}A\right)^q \frac{h^q}{q!} + \mathcal{O}(h^{p+1})\right)^{-1} \\ & \cdot \left(\sum_{i=0}^{N-1}\sum_{q=0}^{p}\left[\alpha_i\left(-\frac{i}{N}A\right)^q - \left(\frac{1}{N}A\right)^q\right]h^q + \mathcal{O}(h^{q+1})\right) \\ \underset{h\to 0}{=} & -\frac{N}{h}\left(\sum_{q=0}^{p}\frac{1}{q!}\left(\frac{A}{N}\right)^q\left[\sum_{i=0}^{N-1}(-i)^q\beta_i\right]h^q + \mathcal{O}(h^{p+1})\right)^{-1} \\ & \cdot \left(\sum_{q=0}^{p}\frac{1}{q!}\left(\frac{A}{N}\right)^q\left[\sum_{i=0}^{N-1}(-i)^q\alpha_i - 1\right]h^q + \mathcal{O}(h^{p+1})\right) \end{aligned}$$

Dans le deuxième facteur, on sépare la somme en deux parties: un premier terme pour $q = 0$ et un second terme pour q allant de 1 à p:

$$\begin{aligned}
W_{Min} \underset{h\to 0}{=} & -N\left(\sum_{q=0}^{p}\frac{1}{q!}\left(\frac{A}{N}\right)^q\left[\sum_{i=0}^{N-1}(-i)^q\beta_i\right]h^q+\mathcal{O}(h^{p+1})\right)^{-1}\\
& \cdot\quad\left(\frac{1}{h}\sum_{i=0}^{N-1}\alpha_i-\frac{1}{h}+\sum_{q=1}^{p}\frac{1}{q!}\left(\frac{A}{N}\right)^q\left[\sum_{i=0}^{N-1}(-i)^q\alpha_i-1\right]h^{q-1}+\mathcal{O}(h^p)\right)\\
\underset{h\to 0}{=} & -\frac{N}{h}\left(\sum_{q=0}^{p}\frac{1}{q!}\left(\frac{A}{N}\right)^q\left[\sum_{i=0}^{N-1}(-i)^q\beta_i\right]h^q+\mathcal{O}(h^{p+1})\right)^{-1}\cdot\left(\sum_{i=0}^{N-1}\alpha_i-1\right)\\
& -\quad N\left(\sum_{q=0}^{p}\frac{1}{q!}\left(\frac{A}{N}\right)^q\left[\sum_{i=0}^{N-1}(-i)^q\beta_i\right]h^q+\mathcal{O}(h^{p+1})\right)^{-1}\\
& \cdot\quad\frac{A}{N}\left(\sum_{q=1}^{p}\frac{1}{q!}\left(\frac{A}{N}\right)^{q-1}\left[\sum_{i=0}^{N-1}(-i)^q\alpha_i-1\right]h^{q-1}+\mathcal{O}(h^p)\right)
\end{aligned}$$

à la dernière ligne, on fait le changement d'indice $q \mapsto q-1$ dans la somme:

$$\begin{aligned}
W_{Min} \underset{h\to 0}{=} & -\frac{N}{h}\left(\sum_{q=0}^{p}\frac{1}{q!}\left(\frac{A}{N}\right)^q\left[\sum_{i=0}^{N-1}(-i)^q\beta_i\right]h^q+\mathcal{O}(h^{p+1})\right)^{-1}\cdot\left(\sum_{i=0}^{N-1}\alpha_i-1\right)\\
& +\quad A\left(\sum_{q=0}^{p-1}\frac{1}{q!}\left(\frac{A}{N}\right)^q\left[-\sum_{i=0}^{N-1}(-i)^q\beta_i\right]h^q+\mathcal{O}(h^p)\right)^{-1}\\
& \cdot\quad\left(\sum_{q=0}^{p-1}\frac{1}{q!}\left(\frac{A}{N}\right)^q\frac{1}{q+1}\left[\sum_{i=0}^{N-1}(-i)^{q+1}\alpha_i-1\right]h^q+\mathcal{O}(h^p)\right)
\end{aligned}$$

Donc, si on a ces conditions:

-

$$\sum_{i=0}^{N-1}\alpha_i-1=0$$

- *Pour tout $q \in [\![0, p-1]\!]$:*

$$-\sum_{i=0}^{N-1}(-i)^q\beta_i = \frac{1}{q+1}\left(\sum_{i=0}^{N-1}(-i)^{q+1}\alpha_i-1\right)$$

Alors on a:

$$\begin{aligned}
W_{Min} &\underset{h\to 0}{=} A\left(I_d+\mathcal{O}(h^p)\right)\\
W_{Min} &\underset{h\to 0}{=} A+\mathcal{O}(h^p)
\end{aligned}$$

en particulier, la deuxième condition est équivalente à:

$$\forall q \in [\![1,q]\!],\ -\sum_{i=0}^{N-1}(-i)^{q-1}\beta_i = \frac{1}{q}\left(\sum_{i=0}^{N-1}(-i)^q\alpha_i - 1\right)$$

$$\Leftrightarrow \forall q \in [\![1,q]\!],\ \sum_{i=0}^{N-1}(-1)^q i^q\alpha_i + q(-1)^{q-1}i^{q-1}\beta_i = 1$$

$$\Leftrightarrow \forall q \in [\![1,q]\!],\ \sum_{i=0}^{N-1} i^q\alpha_i - qi^{q-1}\beta_i = (-1)^q$$

■

Remarque. *Concrètement, pour un problème avec $N+1$ points donnés $y_0^{(k)}, y_{\frac{1}{N}}^{(k)}, \dots, y_1^{(k)}$, pour avoir l'ordre p, il faut que:*

$$\begin{aligned}
\alpha_0 + \alpha_1 + \alpha_2 + \cdots + \alpha_{N-1} &= 1 \\
\alpha_1 + 2\alpha_2 + (N-1)\alpha_{N-1} - (\beta_0 + \cdots + \beta_{N-1}) &= -1 \\
\vdots &\ \vdots \\
\alpha_1 + 2^p\alpha_2 + \cdots + (N-1)^p\alpha_{N-1} - p\left(\beta_1 + 2^{p-1}\beta_2 + \cdots + (N-1)^{p-1}\beta_{N-1}\right) &= (-1)^p
\end{aligned}$$

3.4.2 Exemples

- **Méthode de Nyström**: On se place dans le cas $N = 2$ (1 point intermédiaire donné), et on prend les coefficients:

$$(\alpha_0, \alpha_1, \beta_0, \beta_1) = (0, 1, 2, 0)$$

Ainsi, avec:

$$\hat{y_1^{(k)}} = y_0^{(k)} + \frac{h}{2}W\left(2y_{\frac{1}{2}}^{(k)}\right)$$

l'apprentissage est d'ordre 2:

$$W_{Min} \underset{h\to 0}{=} A + \mathcal{O}(h^2)$$

- **Une méthode d'ordre 3**: On se place dans le cas $N = 2$ (1 point intermédiaire donné), et on prend les coefficients:

$$(\alpha_0, \alpha_1, \beta_0, \beta_1) = (-4, 5, 4, 2)$$

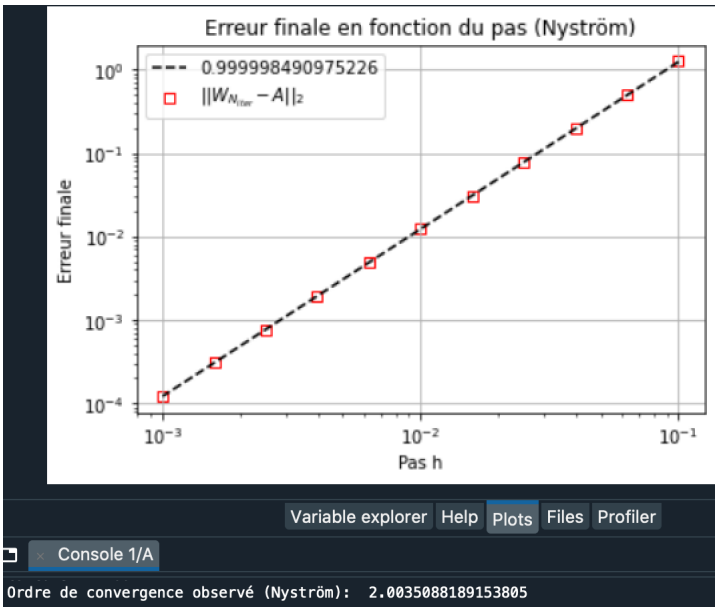

Figure 3.3: Courbe de convergence en échelle log-log pour la méthode de Nyström (*Loss* convexe), obtenue avec une matrice A prise au hasard dans $\mathcal{M}_d([-10, 10])$ dans le cas $d = 2$, $N_{iter} = 500$ et $\alpha = 10^{-3}$

Ainsi, avec:

$$\hat{y_1^{(k)}} = -4y_{\frac{1}{2}}^{(k)} + 5y_0^{(k)} + \frac{h}{2}W\left(4y_{\frac{1}{2}}^{(k)} + 2y_0^{(k)}\right)$$

l'apprentissage est d'ordre 3:

$$W_{Min} \underset{h\to 0}{=} A + \mathcal{O}(h^3)$$

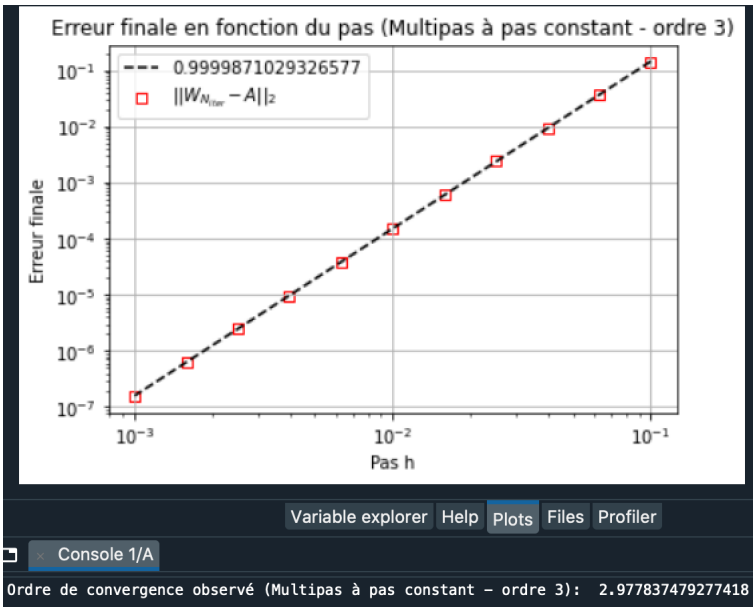

Figure 3.4: Courbe de convergence en échelle log-log pour la méthode Multipas à pas constant d'ordre 3 (*Loss* convexe), obtenue avec une matrice A prise au hasard dans $\mathcal{M}_d([-10, 10])$ dans le cas $d = 2$, $N_{iter} = 500$ et $\alpha = 10^{-3}$

- **Méthode de Milne**: On se place dans le cas $N = 4$ (3 points intermédiaire donné), et on prend les coefficients:

$$(\alpha_0, \alpha_1, \alpha_2, \alpha_3, \beta_0, \beta_1, \beta_2, \beta_3) \quad = \quad \left(0, 0, 0, 1, \frac{8}{3}, -\frac{4}{3}, \frac{8}{3}, 0\right)$$

Ainsi, avec:

$$\hat{y_1^{(k)}} \quad = \quad y_0^{(k)} + \frac{h}{4} W \left(\frac{8}{3} y_{\frac{3}{4}}^{(k)} - \frac{4}{3} y_{\frac{1}{2}}^{(k)} + \frac{8}{3} y_{\frac{1}{4}}^{(k)}\right)$$

l'apprentissage est d'ordre 4:

$$W_{Min} \underset{h \to 0}{=} A + \mathcal{O}(h^4)$$

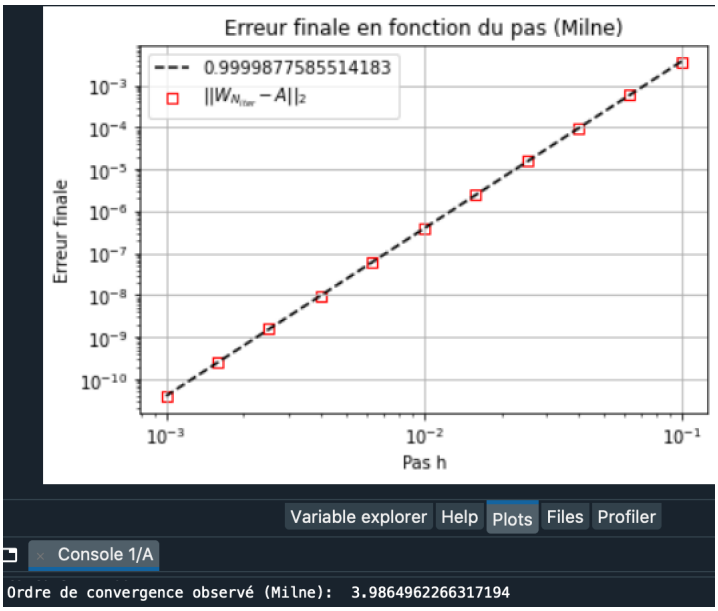

Figure 3.5: Courbe de convergence en échelle log-log pour la méthode de Milne (*Loss* convexe), obtenue avec une matrice A prise au hasard dans $\mathcal{M}_d([-10, 10])$ dans le cas $d = 2$, $N_{iter} = 500$ et $\alpha = 10^{-3}$

- **Une méthode d'ordre 5**: On se place dans le cas $N = 3$ (2 points intermédiaire donné), et on prend les coefficients:

$$(\alpha_0, \alpha_1, \alpha_2, \beta_0, \beta_1, \beta_2) \quad = \quad (-18, 9, 10, 9, 18, 3)$$

Ainsi, avec:

$$\hat{y_1^{(k)}} \quad = \quad -18y_{\frac{2}{3}}^{(k)} + 9y_{\frac{1}{3}}^{(k)} + 10y_0^{(k)} + \frac{h}{3}W\left(9y_{\frac{2}{3}}^{(k)} + 18y_{\frac{1}{3}}^{(k)} + 3y_0^{(k)}\right)$$

l'apprentissage est d'ordre 5:

$$W_{Min} \underset{h\to 0}{=} A + \mathcal{O}(h^5)$$

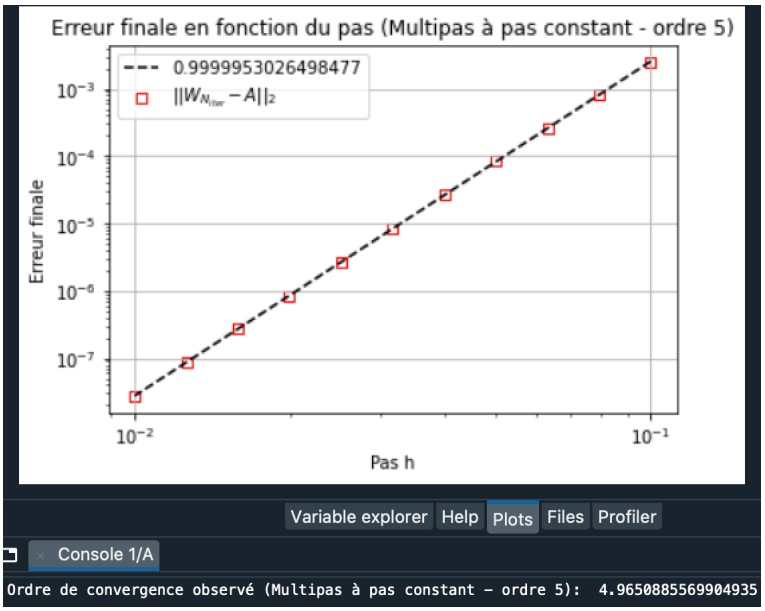

Figure 3.6: Courbe de convergence en échelle log-log pour la méthode Multipas à pas constant d'ordre 5 (*Loss* convexe), obtenue avec une matrice A prise au hasard dans $\mathcal{M}_d([-10, 10])$ dans le cas $d = 2$, $N_{iter} = 1000$ et $\alpha = 10^{-4}$

3.5 Méthodes Multi-pas à pas non constant

On suppose la solution connue aux temps $0 = t_0 < t_1 < \cdots < t_N = h$, où cette foix on ne suppose plus forcément que $t_j = \frac{jh}{N}$:

$$y^{(k)}_{\frac{t_j}{h}} = e^{t_j A} y_0^{(k)}$$

3.5.1 Ordre de convergence

On considère les polynômes interpolateurs de Lagrange:

$$\forall j \in [\![0, N]\!], \quad L_{j,N}(t) = \prod_{\substack{0 \leqslant i \leqslant N \\ i \neq j}} \frac{t - t_i}{t_j - t_i} \in \mathbb{R}_N[t]$$

Ainsi, pour tout $i \in [\![0, N]\!]$, $L_{j,N}(t_i) = \delta_{i,j}$
Soit ainsi:

$$P_N^{(k)}(t) = \sum_{j=0}^{N} L_{j,N}(t) W y^{(k)}_{\frac{t_j}{h}} \in \mathbb{R}[t]$$

$t \mapsto P_N^{(k)}(t)$ doit approcher $t \mapsto W e^{tA} y_0^{(k)}$ sur $[0, h]$ par interpolation aux points $t_0, t_1, \cdots, t_N$.
Posons, pour tout $k \in [\![0, K-1]\!]$:

$$\begin{aligned}\hat{y_1}^{(k)} &= y_0^{(k)} + \int_0^h P_N^{(k)}(t)\mathrm{d}t \\ &= y_0^{(k)} + W\sum_{j=0}^{N}\int_0^h L_{j,N}(t)\mathrm{d}t y_{\frac{t_j}{h}}^{(k)}\end{aligned}$$

On a ainsi:

$$L(W,k) = \frac{1}{h^2}\left|(I_d - e^{hA})y_0^{(k)} + W\sum_{j=0}^{N}\left(\int_0^h L_{j,N}(t)\mathrm{d}t\right)e^{t_jA}y_0^{(k)}\right|^2$$

Proposition (Méthode Multi-pas à pas non constant - Ordre de convergence)

Si $A \in GL_d(\mathbb{R})$ (ce qui est le cas presque partout dans $\mathcal{M}_d(\mathbb{R})$), l'apprentissage pour cette méthode est d'ordre $N+1$:

$$W_{Min} \underset{h\to 0}{=} A + \mathcal{O}(h^{N+1})$$

Démonstration. *Tout d'abord, notons que:*

$$W_{Min} = \left(\sum_{j=0}^{N}\int_0^h L_{j,N}(t)\mathrm{d}t e^{t_jA}\right)^{-1}\left(e^{hA} - I_d\right)$$

Soit la matrice de polynômes:

$$Q_N = \sum_{i=0}^{N} e^{t_jA}L_{j,N}(X) \in \mathbb{R}_N[X]^{d\times d}$$

Soit $t \in [0,h]$. Par le théorème de Taylor, il existe $\xi \in]t,h[$ tel que:

$$e^{tA} - Q_N(t) = \frac{1}{(N+1)!}A^{N+1}e^{\xi A}\pi_N(\xi)$$

avec:

$$\pi_N(\xi) = \prod_{j=0}^{N}(\xi - t_j)$$

Or, on a:

$$\begin{aligned}W_{Min} &= \left(\int_0^h Q_N(t)\mathrm{d}t\right)^{-1}\left(e^{hA} - I_d\right) \\ &= \left(\int_0^h e^{tA}\mathrm{d}t - \int_0^h e^{tA} - Q_N(t)\mathrm{d}t\right)^{-1}\left(e^{hA} - I_d\right)\end{aligned}$$

Donc, pour tout $t \in [0,h]$:

$$\left|\left|e^{tA} - Q_N(t)\right|\right| \leqslant \frac{1}{(N+1)!} ||A||^{N+1} e^{h||A||} h^{N+1}$$

Or, on a:

$$\int_0^h e^{tA} \mathrm{d}t = A^{-1}\left(e^{hA} - I_d\right)$$

Notons que toutes les matrices utilisées commutent entre elles, puisque ce sont des polynômes en A, donc on a:

$$\begin{aligned} W_{Min} &= \left(A^{-1}\left(e^{hA} - I_d\right) - \int_0^h e^{tA} - Q_N(t)\mathrm{d}t\right)^{-1}\left(e^{hA} - I_d\right) \\ &= A\left(e^{hA} - I_d\right)^{-1}\left(I_d - A\left(e^{hA} - I_d\right)^{-1}\int_0^h e^{tA} - Q_N(t)\mathrm{d}t\right)^{-1}\left(e^{hA} - I_d\right) \\ &= A\left(I_d - A\left(e^{hA} - I_d\right)^{-1}\int_0^h e^{tA} - Q_N(t)\mathrm{d}t\right)^{-1} \end{aligned}$$

Or, on a:

$$\begin{aligned} A\left(e^{hA} - I_d\right)^{-1} &\underset{h\to 0}{\sim} \frac{1}{h} I_d \\ \int_0^h e^{tA} - Q_N(t)\mathrm{d}t &\underset{h\to 0}{=} \mathcal{O}(h^{N+2}) \end{aligned}$$

Donc on a:

$$W_{Min} \underset{h\to 0}{=} A\left(I_d + \mathcal{O}(h^{N+1})\right)^{-1} \underset{h\to 0}{=} A + \mathcal{O}(h^{N+1})$$

■

3.5.2 Exemples

- **Une méthode d'ordre 3**

  On prend $N = 2$ (1 point intermédiaire), et on connaît les solutions en $0, \frac{h}{3}$ et h. L'apprentissage est d'ordre 3:

$$W_{Min} \underset{h\to 0}{=} A + \mathcal{O}(h^3)$$

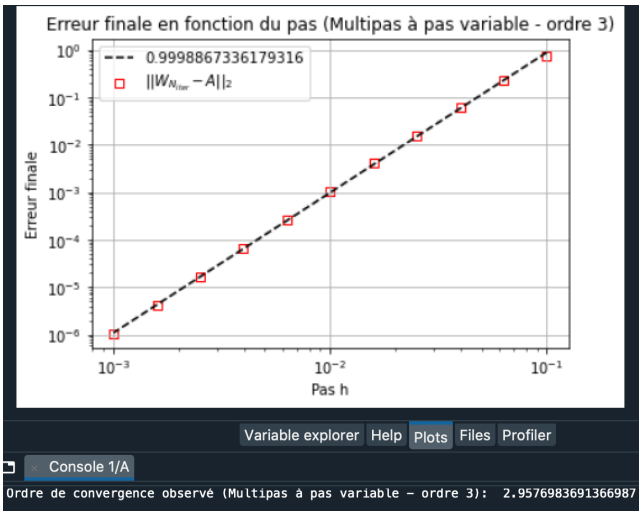

Figure 3.7: Courbe de convergence en échelle log-log pour la méthode Multipas à pas variable d'ordre 3 (*Loss* convexe), obtenue avec une matrice A prise au hasard dans $\mathcal{M}_d([-10, 10])$ dans le cas $d = 2$, $N_{iter} = 500$ et $\alpha = 10^{-3}$

- **Une méthode d'ordre 4**

  On prend $N = 3$ (2 points intermédiaires), et on connaît les solutions en $0, \frac{h}{4}, \frac{h}{2}$ et h. L'apprentissage est d'ordre 4:

$$W_{Min} \underset{h \to 0}{=} A + \mathcal{O}(h^4)$$

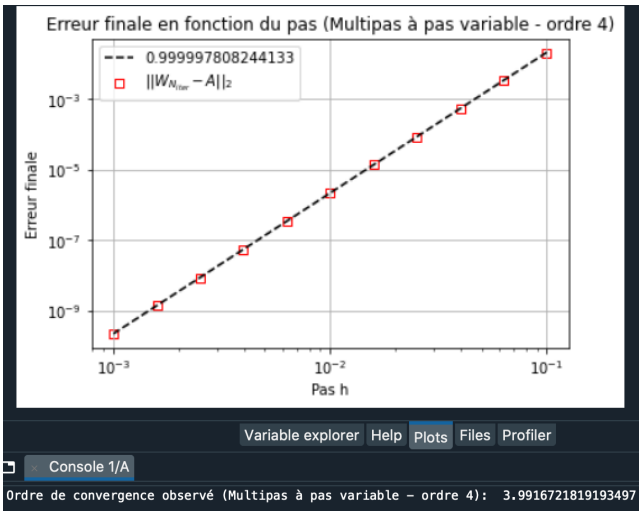

Figure 3.8: Courbe de convergence en échelle log-log pour la méthode Multipas à pas variable d'ordre 4 (*Loss* convexe), obtenue avec une matrice A prise au hasard dans $\mathcal{M}_d([-10, 10])$ dans le cas $d = 2$, $N_{iter} = 500$ et $\alpha = 10^{-3}$

3.6 Généralisation

3.6.1 Ordre de convergence

Dans cette sous-partie, on veut déterminer l'ordre d'apprentissage directement à partir de l'ordre de la méthode numérique, tant qu'elle est sous la forme (3.1). Par exemple, vu que les méthodes d'Euler sont d'ordre 1, l'apprentissage est d'ordre 1, pour le Point Milieu, qui est une méthode d'ordre 2, l'apprentissage est lui aussi d'ordre 2.

Théorème (Ordre de convergence - Cas linéaire)

Si on prend $\hat{y_1}$ de la forme (3.1):

$$\hat{y_1} = \sum_{j=0}^{N} \alpha_j y_{\frac{t_j}{h}} + W \sum_{j=0}^{N} \beta_j h_j y_{\frac{t_j}{h}}$$

où la méthode est d'ordre p, c'est-à-dire que, si on pose:

$$\widetilde{y_1} = \sum_{j=0}^{N} \alpha_j y_{\frac{t_j}{h}} + A \sum_{j=0}^{N} \beta_j h_j y_{\frac{t_j}{h}}$$

on a:

$$\widetilde{y_1} - y_1 \underset{h\to 0}{=} \mathcal{O}(h^{p+1})$$

et si on suppose que:

$$\sum_{j=0}^{N} \beta_j \frac{h_j}{h} \neq 0$$

Alors, l'apprentissage est d'ordre p:

$$W_{Min} \underset{h\to 0}{=} A + \mathcal{O}(h^p)$$

Démonstration. *On a cette expression pour la fonction $L(\cdot)$:*

$$L(W) = \frac{1}{h^2} \left| \sum_{j=0}^{N} \alpha_j y_{\frac{t_j}{h}} - y_1 + W \sum_{j=0}^{N} \beta_j h_j y_{\frac{t_j}{h}} \right|^2$$

Remarque. *Par souci d'allègement des notations, l'indice (k) de la donnée a été supprimé. Dans le cas d'une vraie donné $y_0^{(k)}$, on écrirait $L(W, k)$.*

Ainsi, on a:

$$\begin{aligned} W_{Min}\left(\sum_{j=0}^{N}\beta_j h_j y_{\frac{t_j}{h}}\right) &= y_1 - \sum_{j=0}^{N}\alpha_j y_{\frac{t_j}{h}} \\ (W_{Min}-A)\left(\sum_{j=0}^{N}\beta_j h_j y_{\frac{t_j}{h}}\right) &= y_1 - \sum_{j=0}^{N}\alpha_j y_{\frac{t_j}{h}} - A\left(\sum_{j=0}^{N}\beta_j h_j y_{\frac{t_j}{h}}\right) \\ &= y_1 - \widetilde{y_1} \end{aligned}$$

W_{Min} peut-être construite à partir de A et de $e^{t_j A}$ par somme, produit ou inversion, donc à partir de polynômes en A, c'est pour cette raison que l'on peut faire commuter les facteurs des produits. Ainsi, on a:

$$\begin{aligned} \sum_{j=0}^{N}\beta_j h_j W_{Min} e^{t_j A} y_0 &= y_1 - \widetilde{y_1} + \sum_{j=0}^{N}\beta_j h_j A e^{t_j A} y_0 \\ W_{Min}\left(\sum_{j=0}^{N}\beta_j h_j e^{t_j A}\right) y_0 &= A\left(\sum_{j=0}^{N}\beta_j h_j e^{t_j A}\right) y_0 + y_1 - \widetilde{y_1} \\ W_{Min} y_0 &= A y_0 + \left(\sum_{j=0}^{N}\beta_j h_j e^{t_j A}\right)^{-1}\left(y_1 - \widetilde{y_1}\right) \\ W_{Min} y_0 &= A y_0 + \left(\sum_{j=0}^{N}\beta_j \frac{h_j}{h} e^{t_j A}\right)^{-1}\frac{1}{h}\left(y_1 - \widetilde{y_1}\right) \end{aligned}$$

Comme $\sum_{j=0}^{N}\beta_j \frac{h_j}{h} \neq 0$, on a:

$$\sum_{j=0}^{N}\beta_j \frac{h_j}{h} e^{t_j A} \underset{h\to 0}{\longrightarrow} \left(\sum_{j=0}^{N}\beta_j \frac{h_j}{h}\right) I_d$$

qui est une matrice inversible (perturbation d'un multiple non nul de l'identité). Par ailleurs, comme on a $\widetilde{y_1} - y_1 \underset{h\to 0}{=} \mathcal{O}(h^{p+1})$, on obtient ainsi:

$$W_{Min} \underset{h\to 0}{=} A + \mathcal{O}(h^p)$$

■

3.6.2 Exemples

Une méthode d'ordre 2

Si on prend cette méthode numérique:

$$\begin{aligned} \hat{y_1} &= y_0 + \frac{h}{4} W(y_0 + 2y_{\frac{1}{2}} + y_1) \\ \widetilde{y_1} &= y_0 + \frac{h}{4} A(y_0 + 2y_{\frac{1}{2}} + y_1) \end{aligned}$$

Alors on a:

$$\begin{aligned} \widetilde{y_1} - y_1 &= \left(I_d - e^{hA}\right) y_0 + \frac{h}{4} A \left(I_d + 2e^{\frac{h}{2}A} + e^{hA}\right) y_0 \\ &= \left(I_d - e^{hA}\right) y_0 + \frac{h}{4} A \left(I_d + e^{\frac{h}{2}A}\right)^2 y_0 \end{aligned}$$

On a donc:

$$\widetilde{y_1} - y_1 \underset{h \to 0}{=} \mathcal{O}(h^3)$$

qui est de l'ordre 3 localement, ainsi, l'apprentissage est d'ordre 2:

$$W_{min} \underset{h \to 0}{=} A + \mathcal{O}(h^2)$$

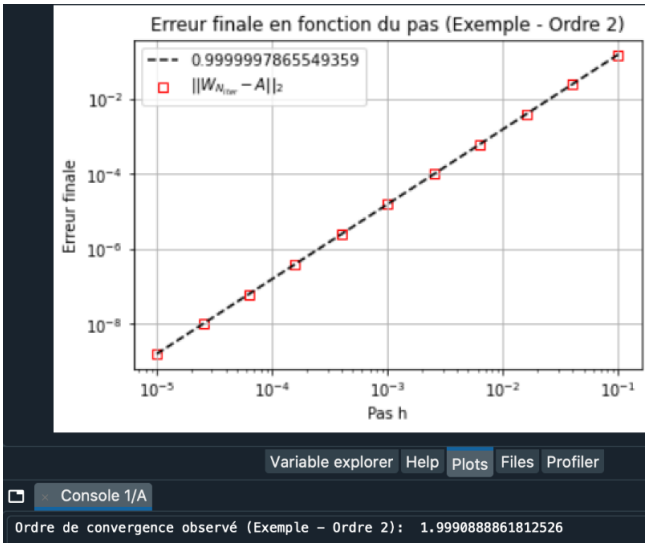

Figure 3.9: Courbe de convergence en échelle log-log pour l'exemple, obtenue avec une matrice A prise au hasard dans $\mathcal{M}_d([-10, 10])$ dans le cas $d = 2$, $N_{iter} = 1000$ et $\alpha = 10^{-3}$.

Une méthode d'ordre 4

Si on prend cette méthode numérique:

$$\begin{aligned}\hat{y_1} &= y_0 + \frac{h}{8}W(y_0 + 3y_{\frac{1}{3}} + 3y_{\frac{2}{3}} + y_1)\\ \widetilde{y_1} &= y_0 + \frac{h}{8}A(y_0 + 3y_{\frac{1}{3}} + 3y_{\frac{2}{3}} + y_1)\end{aligned}$$

Alors on a:

$$\begin{aligned}\widetilde{y_1} - y_1 &= \left(I_d - e^{hA}\right) y_0 + \frac{h}{8}A\left(I_d + 3e^{\frac{h}{3}A} + 3e^{\frac{2h}{3}A} + e^{hA}\right) y_0\\ &= \left(I_d - e^{hA}\right) y_0 + \frac{h}{8}A\left(I_d + e^{\frac{h}{3}A}\right)^3 y_0\end{aligned}$$

On a donc:

$$\widetilde{y_1} - y_1 \underset{h\to 0}{=} \mathcal{O}(h^5)$$

qui est de l'ordre 5 localement, ainsi, l'apprentissage est d'ordre 4:

$$W_{min} \underset{h\to 0}{=} A + \mathcal{O}(h^4)$$

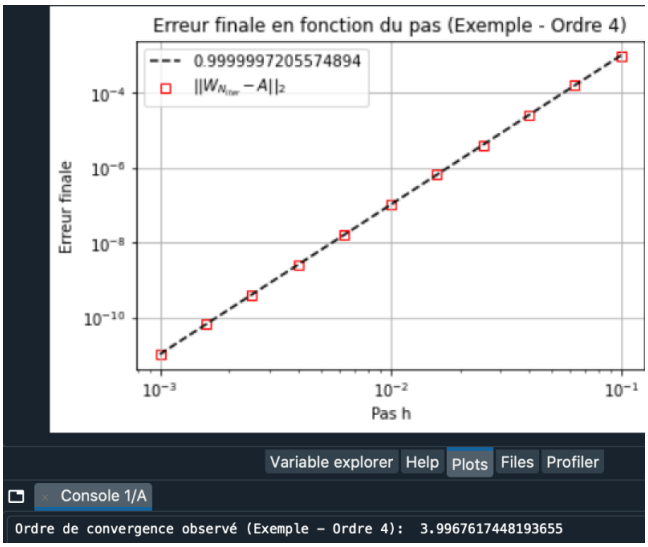

Figure 3.10: Courbe de convergence en échelle log-log pour l'exemple, obtenue avec une matrice A prise au hasard dans $\mathcal{M}_d([-10, 10])$ dans le cas $d = 2$, $N_{iter} = 1000$ et $\alpha = 10^{-3}$.

Partie 4

Cas non linéaire

Dans cette partie, nous allons nous intéresser au cas général:

$$\left\{ \begin{array}{rcl} \dot{y} & = & f(y) \\ y(0) & = & y_0 \end{array} \right.$$

et on suppose que l'on connaît K données $y_0^{(k)}$ (conditions initiales) et $y_1^{(k)} = \phi_h(y_0^{(k)})$ (solution au temps h, où h est le pas de la méthode numérique choisie).

4.1 Structure du réseau de neurones

Afin d'avoir une idée du réseau de neurones le plus simple à créer, nous allons utiliser le théorème d'universalité:

Théorème (Théorème d'universalité)

Soit $g \in \mathcal{C}(\mathbb{R}^d, \mathbb{R})$, soit $\sigma : \mathbb{R} \longrightarrow \mathbb{R}$ une fonction continue non polynomiale. Alors, pour tout $\varepsilon > 0$, il existe $g_{app} \in \mathcal{M}_\sigma$ telle que, pour tout $\Omega \subset \mathbb{R}^d$ compact:

$$||g - g_{app}||_{L^\infty(\Omega)} \leqslant \varepsilon$$

avec:

$$\mathcal{M}_\sigma = Vect\left\{x \mapsto \sigma(w \cdot x + b), w \in \mathbb{R}^d, b \in \mathbb{R}\right\}$$

Remarques. - *σ est appelée fonction d'activation. On peut par exemple choisir une sigmoïde:*

$$\sigma(x) = \frac{1}{1 + e^{-x}}$$

mais on retiendra, dans le cadre du stage, la tangente hyperbolique:

$$\sigma(x) = \tanh(x)$$

- *Vu que notre champ de vecteurs est à valeurs dans $\mathbb{R}^d$ et que le théorème s'applique à une fonction à valeurs dans $\mathbb{R}$, on doit appliquer ce théorème à chaque composante de f.*

Ainsi, avec ce théorème, on peut construire un réseau de neurones qui approchera correctement notre champ de vecteurs f par un champ approché f_{app}. Voici la structure du réseau de neurones, qui comporte deux couches:

- Une première couche de ζ neurones, et donc les paramètres sont $W^{(1)} \in \mathcal{M}_{\zeta,d}(\mathbb{R})$ (matrice poids des ζ neurones), ainsi qu'un biais de poids $w_0^{(1)} \in \mathbb{R}^\zeta$
- Une seconde couche de d neurones, et donc les paramètres sont $W^{(2)} \in \mathcal{M}_{d,\zeta}(\mathbb{R})$ (matrice poids des ζ neurones), ne comportant pas de biais.

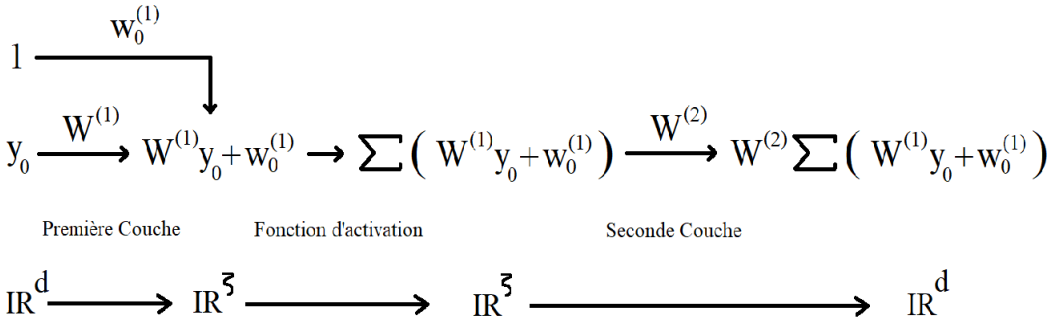

Figure 4.1: Illustration du fonctionnement du réseau de neurones permettant d'approcher f_{app} pour le cas général. La fonction $\Sigma : \mathbb{R}^\zeta \to \mathbb{R}^\zeta$ applique la fonction σ à chacune des composantes du vecteur qu'elle prend en entrée

Ainsi, le champ de vecteur approché f_{app} s'écrit:

$$f_{app}(y_0) = W^{(2)}\Sigma\left(W^{(1)}y_0 + w_0^{(1)}\right)$$

Par exemple, pour la méthode d'Euler Explicite, on a cette valeur prédite en h:

$$\hat{y_1^{(k)}} = y_0^{(k)} + hf_{app}\left(y_0^{(k)}\right)$$

ainsi que cette fonction *loss*:

$$L\left(W^{(1)}, w_0^{(1)}, W^{(2)}, k\right) = \frac{1}{h^2}\left|y_0^{(k)} - y_1^{(k)} + hf_{app}\left(y_0^{(k)}\right)\right|^2$$

4.2 Ordre de convergence

Dans cette sous-partie, on considèrera une certaine méthode numérique multi-pas. On considère également que le champ de vecteur f recherché peut être approché de manière aussi précise que l'on souhaite par un champ f_{app} (en d'autres termes, on peut utiliser

autant de neurones que l'on veut dans le théorème d'universalité). on considère alors un vecteur $\hat{y_1}$ donnant la valeur prédite en h du problème de Cauchy:

$$\begin{cases} \dot{y} &= f(y) \\ y(0) &= y_0 \end{cases}$$

à l'aide du champ de vecteurs f_{app}. Ce vecteur $\hat{y_1}$ utilise une certaine méthode numérique, en faisant intervenir f_{app}, ainsi que $y_{\frac{t_j}{h}}$, via notre réseau de neurones.

On introduit la fonction *Loss*:

$$L(f_{app}) = \frac{1}{h^2} \left| \hat{y_1} - y_1 \right|^2$$

Soit $f^*_{app} \in \mathcal{C}^0(\mathbb{R}^d, \mathbb{R}^d)$ tel que $L(f^*_{app}) = 0$ (on suppose que l'on peut avoir une précision aussi fine que possible, en pratique, il est compliqué d'avoir une *Loss* nulle). L'objectif de cette sous-partie est d'établir un lien entre l'ordre de la méthode numérique utilisé et l'ordre d'apprentissage, c'est-à-dire étudier la différence entre les champs de vecteurs f^*_{app} (champ de vecteurs appris) et f (champ de vecteurs réel).

4.2.1 Première formulation

Dans cette section, on souhaite déterminer l'ordre d'apprentissage directement à partir de l'ordre de la méthode numérique, tant que cette dernière est de la forme:

$$\hat{y_1} = \sum_{j=0}^{N} \alpha_j y_{\frac{t_j}{h}} + h \sum_{j=0}^{N} \beta_j \frac{h_j}{h} f_{app}\left(y_{\frac{t_j}{h}}\right) \tag{4.1}$$

Théorème (Ordre de convergence dans le cas général - Première formulation)

Si on prend $\hat{y_1}$ de la forme (4.1):

$$\hat{y_1} = \sum_{j=0}^{N} \alpha_j y_{\frac{t_j}{h}} + h \sum_{j=0}^{N} \beta_j \frac{h_j}{h} f_{app}\left(y_{\frac{t_j}{h}}\right)$$

où la méthode est d'ordre p, c'est-à-dire que, si on pose:

$$\widetilde{y_1} = \sum_{j=0}^{N} \alpha_j y_{\frac{t_j}{h}} + h \sum_{j=0}^{N} \beta_j \frac{h_j}{h} f\left(y_{\frac{t_j}{h}}\right)$$

on a:

$$\widetilde{y_1} - y_1 \underset{h\to 0}{=} \mathcal{O}(h^{p+1})$$

et si on suppose que:

$$\sum_{j=0}^{N} \beta_j \frac{h_j}{h} \neq 0$$

Alors, l'apprentissage est d'ordre p, autrement dit, pour tout $y_0 \in \mathbb{R}^d$, et pour h assez petit:

$$f^*_{app}(y_0) \underset{h\to 0}{=} f(y_0) + \mathcal{O}(h^p)$$

Démonstration. *On a:*

$$\hat{y_1} = \sum_{j=0}^{N} \alpha_j y_{\frac{t_j}{h}} + h \sum_{j=0}^{N} \beta_j \frac{h_j}{h} f_{app}\left(y_{\frac{t_j}{h}}\right)$$

avec $\frac{h_j}{h}$ constant (la position des t_j ne change pas par rapport à h). Soit la fonction Loss:

$$\begin{aligned} L(f_{app}) &= \frac{1}{h^2} |\hat{y_1} - y_1|^2 \\ &= \frac{1}{h^2} \left| \sum_{j=0}^{N} \alpha_j y_{\frac{t_j}{h}} - y_1 + h \sum_{j=0}^{N} \beta_j \frac{h_j}{h} f_{app}\left(y_{\frac{t_j}{h}}\right) \right|^2 \end{aligned}$$

On a ainsi:

$$h \sum_{j=0}^{N} \beta_j \frac{h_j}{h} f^*_{app}\left(y_{\frac{t_j}{h}}\right) = y_1 - \sum_{j=0}^{N} \alpha_j y_{\frac{t_j}{h}}$$

Donc:

$$\begin{aligned} h\sum_{j=0}^{N}\beta_j\frac{h_j}{h}\left[f^*_{app}-f\right]\left(y_{\frac{t_j}{h}}\right) &= y_1-\sum_{j=0}^{N}\alpha_j y_{\frac{t_j}{h}}-h\sum_{j=0}^{N}\beta_j\frac{h_j}{h}f\left(y_{\frac{t_j}{h}}\right) \\ &= y_1-\widetilde{y_1} \end{aligned}$$

Ainsi, on a:

$$\begin{aligned} \sum_{j=0}^{N}\beta_j\frac{h_j}{h}\left[f^*_{app}-f\right]\left(y_{\frac{t_j}{h}}\right) &= \frac{1}{h}\left(y_1-\widetilde{y_1}\right) \\ \varphi_h\left(f^*_{app}-f\right)(y_0) &= \frac{1}{h}\left(y_1-\widetilde{y_1}\right) \end{aligned}$$

où l'application (linéaire) φ_h est donnée par:

$$\begin{array}{rccc} \varphi_h: & \mathcal{C}^0(\mathbb{R}^d,\mathbb{R}^d) & \longrightarrow & \mathcal{C}^0(\mathbb{R}^d,\mathbb{R}^d) \\ & g & \longmapsto & \sum_{j=0}^{N}\beta_j\frac{h_j}{h}g\left(\phi_{t_j}(y_0)\right) \end{array}$$

On a ainsi:

$$\varphi_h \underset{h\to 0}{\longrightarrow} \left(\sum_{j=0}^{N}\beta_j\frac{h_j}{h}\right)I_d$$

qui est bien une application inversible puisque $\sum_{j=0}^{N}\beta_j\frac{h_j}{h}\neq 0$. On a ainsi:

$$f^*_{app}(y_0) = f(y_0)+\frac{1}{h}\varphi_h^{-1}\left(\phi_h-F_h\right)(y_0)$$

où F_h est le champ de vecteurs donné par:

$$F_h = \left(\sum_{j=0}^{N}\alpha_j\phi_{t_j}\right)I_d+h\sum_{j=0}^{N}\beta_j\frac{h_j}{h}\left(f^*_{app}\circ\phi_{t_j}\right)$$

Or, on a:

$$\varphi_h^{-1}\left(F_h-\phi_h\right)(y_0) \underset{h\to 0}{\sim} \frac{1}{\sum_{j=0}^{N}\beta_j\frac{h_j}{h}}\left(\phi_h-F_h\right)(y_0)$$

Donc on a:

$$\begin{aligned} f^*_{app}(y_0) &\underset{h\to 0}{=} f(y_0)+\frac{1}{h\sum_{j=0}^{N}\beta_j\frac{h_j}{h}}\left(\phi_h-F_h\right)(y_0) \\ &\underset{h\to 0}{=} f(y_0)+\frac{1}{h\sum_{j=0}^{N}\beta_j\frac{h_j}{h}}\left(y_1-\widetilde{y_1}\right) \end{aligned}$$

D'où:

$$f^*_{app}(y_0) \underset{h\to 0}{=} f(y_0)+\mathcal{O}(h^p)$$

■

4.2.2 Nouvelle formulation

Dans cette section, on souhaite déterminer l'ordre d'apprentissage directement à partir de l'ordre de la méthode numérique, tant que cette dernière est de la forme:

$$\hat{y_1} = \sum_{j=0}^{N} \alpha_j y_{\frac{t_j}{h}} + h f_{app}\left(\sum_{j=0}^{N} \beta_j \frac{h_j}{h} y_{\frac{t_j}{h}}\right) \tag{4.2}$$

Théorème (Ordre de convergence dans le cas général - Nouvelle formulation)

Si on prend $\hat{y_1}$ de la forme (4.2):

$$\hat{y_1} = \sum_{j=0}^{N} \alpha_j y_{\frac{t_j}{h}} + h f_{app}\left(\sum_{j=0}^{N} \beta_j \frac{h_j}{h} y_{\frac{t_j}{h}}\right)$$

où la méthode est d'ordre p, c'est-à-dire que, si on pose:

$$\tilde{y_1} = \sum_{j=0}^{N} \alpha_j y_{\frac{t_j}{h}} + h f\left(\sum_{j=0}^{N} \beta_j \frac{h_j}{h} y_{\frac{t_j}{h}}\right)$$

on a:

$$\tilde{y_1} - y_1 \underset{h\to 0}{=} \mathcal{O}(h^{p+1})$$

et si on suppose que:

$$\sum_{j=0}^{N} \beta_j \frac{h_j}{h} \neq 0$$

Alors, l'apprentissage est d'ordre p, autrement dit, pour tout $y_0 \in \mathbb{R}^d$, et pour h assez petit:

$$f^*_{app}(y_0) \underset{h\to 0}{=} f(y_0) + \mathcal{O}(h^p)$$

Démonstration. *On a:*

$$\hat{y_1} = \sum_{j=0}^{N} \alpha_j y_{\frac{t_j}{h}} + h f_{app}\left(\sum_{j=0}^{N} \beta_j \frac{h_j}{h} y_{\frac{t_j}{h}}\right)$$

avec $\frac{h_j}{h}$ constant (la position des t_j ne change pas par rapport à h). Soit la fonction Loss:

$$\begin{aligned} L(f_{app}) &= \frac{1}{h^2}\left|\hat{y_1} - y_1\right|^2 \\ &= \frac{1}{h^2}\left|\sum_{j=0}^{N}\alpha_j y_{\frac{t_j}{h}} - y_1 + hf_{app}\left(\sum_{j=0}^{N}\beta_j\frac{h_j}{h}y_{\frac{t_j}{h}}\right)\right|^2 \end{aligned}$$

On a ainsi:

$$hf^*_{app}\left(\sum_{j=0}^{N}\beta_j\frac{h_j}{h}y_{\frac{t_j}{h}}\right) = y_1 - \sum_{j=0}^{N}\alpha_j y_{\frac{t_j}{h}}$$

Donc:

$$h\left[f^*_{app} - f\right]\left(\sum_{j=0}^{N}\beta_j\frac{h_j}{h}y_{\frac{t_j}{h}}\right) = y_1 - \sum_{j=0}^{N}\alpha_j y_{\frac{t_j}{h}} - hf\left(\sum_{j=0}^{N}\beta_j\frac{h_j}{h}y_{\frac{t_j}{h}}\right) = y_1 - \widetilde{y_1}$$

Ainsi, on a:

$$\left[f^*_{app} - f\right]\left(\sum_{j=0}^{N}\beta_j\frac{h_j}{h}y_{\frac{t_j}{h}}\right) = \frac{1}{h}\left(y_1 - \widetilde{y_1}\right) \underset{h\to 0}{=} \mathcal{O}(h^p)$$

De plus, puisque l'on a:

$$\sum_{j=0}^{N}\beta_j\frac{h_j}{h} \neq 0$$

alors on peut changer $\sum_{j=0}^{N}\beta_j\frac{h_j}{h}y_{\frac{t_j}{h}}$ *en* y_0*. En effet:*

$$\sum_{j=0}^{N}\beta_j\frac{h_j}{h}y_{\frac{t_j}{h}} = \left(\sum_{j=0}^{N}\beta_j\frac{h_j}{h}\phi_{t_j}\right)(y_0)$$

et:

$$\sum_{j=0}^{N}\beta_j\frac{h_j}{h}\phi_{t_j} \underset{h\to 0}{\longrightarrow} \left(\sum_{j=0}^{N}\beta_j\frac{h_j}{h}\right)I_d$$

qui est inversible, donc $\sum_{j=0}^{N}\beta_j\frac{h_j}{h}\phi_{t_j}$ *est inversible quand* h *est assez petit (c'est une perturbation d'un multiple non nul de l'identité). De plus, le terme* $y_1 - \widetilde{y_1}$ *reste d'ordre* $p+1$*.*

■

Bibliographie

[1] S. Mallat, Notes des Cours de Stéphane Mallat Chaire "Sciences des Données" du Collège de France, Cours 2019: L'apprentissage par réseaux de neurones profonds. Notes sur tout le cours 2019 par J-E. Campagne

[2] Site OpenClassrooms d'initiation au Deep Learning, https://openclassrooms.com/fr/courses/5801891-initiez-vous-au-deep-learning

[3] M. Raissi, P. Perdikaris, G.E. Karniadakis, Multistep neural networks for data-driven discovery of nonlinear dynamical systems, arXiv preprint, arXiv:1801.01236.

[4] Steven L. Brunton, Joshua L. Proctor, and J. Nathan Kutz. "Discovering governing equations from data by sparse identification of nonlinear dynamical systems". In: Proceedings of the National Academy of Sciences 113.15 (2016), pp. 3932–3937. ISSN: 0027-8424

[5] Duong Nguyen et al. EM-like Learning Chaotic Dynamics from Noisy and Partial Observations. 2019. arXiv: 1903.10335

[6] Ricky T. Q. Chen et al. "Neural Ordinary Differential Equations". In: Advances in Neural Information Processing Systems. Ed. by S. Bengio et al. Vol. 31. Curran Associates, Inc., 2018.

[7] F. Regazzoni, L. Dedè, and A. Quarteroni. "Machine learning for fast and reliable solution of time-dependent differential equations". In: Journal of Computational Physics 397 (2019), p. 108852. ISSN: 0021-9991.

[8] J-P. Demailly, Analyse numérique et équations différentielles, coll. Grenoble Sciences, 1999.

[9] E. Hairer, S. P. Nørsett and G. Wanner. Solving Ordinary Differential Equations I, 2008, ISBN 978-3-540-56670-0.

[10] Pierre Gillot, Akka Zemmari1, Jenny Benois-Pineau et Yurii Nesterov. Algorithmes de Descente de Gradient Stochastique avec le filtrage des paramètres pour l'entraînement des réseaux à convolution profonds.