Résolution d'équations différentielles par réseaux de neurones

Matthieu Carreau
Telecom Paris, Institut Polytechnique de Paris
F-91120, Palaiseau, France
matthieu.carreau@telecom-paris.fr

Supervisors:
Stam Nicolis
Institut Denis Poisson
Université de Tours, Université d'Orléans, CNRS (UMR7013)
Parc de Grandmont, F-37200, Tours, France
stam.nicolis@lmpt.univ-tours.fr

Pascal Thibaudeau
CEA Le Ripault
BP 16, F-37260, Monts, France
pascal.thibaudeau@cea.fr

Juillet 2022

Résumé

Faire un résumé de ce qu'il y a dans ce document

Table des matières

Chapitre 1

Introduction

Cette partie positionne le travail dans un contexte. Décrire le contexte. Dire ici ce que l'on doit faire et proposer un petit résumé des documents lus de façon à montrer en quoi ils sont pertients pour le problème posé. Par exemple : Bidulle et Machin dans la référence [1] ont montré que ... tandis que Truc et Chmuc dans la référence [2] ont prouvé que... Dans la section 2, je montrerai que... Dans la section 3, je montrerai... Enfin dans la section 4, je discuterai des résultats obtenus et proposerai quelques perspectives.

Chapitre 2

Equation différentielle d'ordre 1

Soit Ψ une fonction réelle à une variable dont la solution satisfait l'équation différentielle suivante, où ψ_0 désigne la valeur initiale de la fonction.

$$\begin{cases} \dfrac{d\Psi(x)}{dx} + \cos(2\pi x) = 0 \\ \Psi(0) = \psi_0 \end{cases} \tag{2.1}$$

On cherche à tester les méthodes présentées dans la section 1 sur l'équation (2.1), pour tout $x \in [0, 1]$. L'équation (2.1) et sa condition initiale donnée en $x = 0$, admet une solution analytique unique qui s'écrit

$$\Psi(x) = \psi_0 - \frac{1}{2\pi}\sin(2\pi x) \tag{2.2}$$

Cela nous permettra par la suite d'évaluer nos solutions numériques, en les comparant à cette solution analytique.

2.1 Solutions en séries de Fourier

On cherche des solutions numériques approchées de l'équation (2.1) sous la forme de séries de Fourier tronquées avec M harmoniques. On écrit pour cela la solution approchée $\tilde{\Psi}$ comme la somme de deux termes, le premier ψ_0 constant non ajustable vérifiant la condition initiale, et le deuxième $\mathcal{N}(x, \mathbf{A})$ dépendant des coefficients $(A_m)_{m\in[\![1,M]\!]}$ représentés par le vecteur $\mathbf{A}$, construit de façon à ne pas influencer la valeur initiale de la fonction.

$$\begin{cases} \tilde{\Psi}(x) = \psi_0 + \mathcal{N}(x, \mathbf{A}) \\ \mathcal{N}(x, \mathbf{A}) = \sum_{m=1}^{M} A_m \sin(2\pi m x) \end{cases} \tag{2.3}$$

On définit une fonction d'erreur pour ces solutions potentielles, à partir de la valeur de la dérivée de $\tilde{\Psi}$ aux N points suivants : $\forall i \in [\![1, N]\!], x_i = \frac{i-1}{N}$

$$E = \frac{1}{2} \sum_{i=1}^{N} (\sum_{m=1}^{M} 2\pi m A_m \cos(2\pi m x_i) + \cos(2\pi x_i))^2 \tag{2.4}$$

On cherche à présent le minimum de E en tant que fonction de $\mathbf{A}$. Une condition nécessaire sur $\mathbf{A}$ pour être un antécédent d'un minimum est

$$\forall l \in [\![1, M]\!], \frac{\partial E}{\partial A_l} = 0 \tag{2.5}$$

Or ces dérivées partielles sont données pour $l \in [\![1, M]\!]$ par :

$$\frac{\partial E}{\partial A_l} = \sum_{i=1}^{N} (\sum_{m=1}^{M} 2\pi m A_m \cos(2\pi m x_i) + \cos(2\pi x_i)) 2\pi l \cos(2\pi l x_i) \tag{2.6}$$

Les deux méthodes suivantes ont pour objectif de trouver les coefficients $(A_m)_{m \in [\![1,M]\!]}$ qui vérifient la condition 2.5, et de vérifier que le vecteur de coefficients trouvé correspond bien à la solution analytique, i.e $\forall m \in [\![1, M]\!], A_m = -\frac{1}{2\pi} \delta_1^m$.

2.1.1 Première méthode : inversion d'un système linéaire

On cherche à résoudre le système linéaire donné par : $\forall l \in [\![1, M]\!], \frac{\partial E}{\partial A_l} = 0$, on définit pour cela les matrices suivantes :

$$\mathcal{M} = (r_{m,l})_{(m,l) \in [\![1,M]\!]^2}, \mathbf{A} = \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_M \end{pmatrix}, \mathbf{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_M \end{pmatrix} \tag{2.7}$$

$$\forall (m,l) \in [\![1,N]\!]^2, \begin{cases} r_{m,l} = 2\pi ml \sum_{i=1}^{N} \cos(2\pi m x_i)\cos(2\pi l x_i) \\ b_l = -l \sum_{i=1}^{N} \cos(2\pi x_i)\cos(2\pi l x_i) \end{cases} \tag{2.8}$$

On souhaite alors résoudre le système linéaire en écrivant l'équation matricielle le représentant, c'est-à-dire :

$$\mathcal{M}\mathbf{A} = \mathbf{b} \Leftrightarrow \mathbf{A} = \mathcal{M}^{-1}\mathbf{b} \tag{2.9}$$

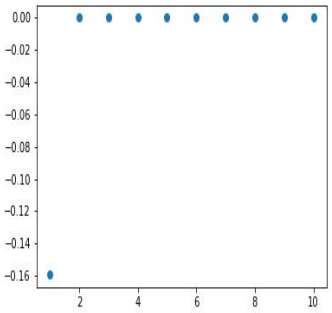
On réalise l'implémentation en python, en instanciant les matrices définies précédemment et à l'aide de la bibliothèque *numpy*. On peut constater que la matrice $\mathcal{M}$ n'est pas toujours inversible selon le choix de M et N. En effet, on peut montrer (à rajouter en annexe) que c'est nécessairement le cas lorsque $M > N$, mais on remarque également qu'elle ne l'est pas non plus lorsque M et N sont proches par exemple pour $M = N = 10$. Il serait interressant d'approfondir ce point pour savoir si cela se traduit par le fait que E admette plusieurs minimums locaux par exemple. Il semble que choisir $N \gg M$ soit suffisant pour que $\mathcal{M}$ soit inversible. On choisira alors $M = 10, N = 100$ pour la suite. On obtient alors les coefficients présentés en figure 2.1 avec les valeurs absolues des erreurs de chacun par rapport à la valeur théorique. On constate que les erreurs sur chaque coefficient est inférieure à 10^{-16}, on valide donc cette première méthode.

2.1.2 Seconde méthode : descente de gradients

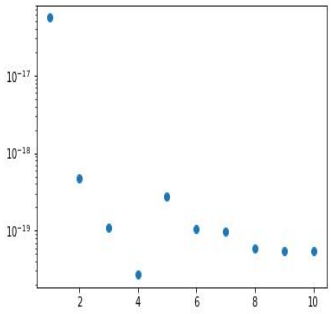
On définit les paramètres suivants :

$$\alpha > 0, \mathbf{A^{(0)}} = \begin{pmatrix} A_1^{(0)} \\ A_2^{(0)} \\ \vdots \\ A_M^{(0)} \end{pmatrix}, \mathbf{g^{(0)}} = \begin{pmatrix} \frac{\partial E^{(0)}}{\partial A_1^{(0)}} \\ \frac{\partial E^{(0)}}{\partial A_2^{(0)}} \\ \vdots \\ \frac{\partial E^{(0)}}{\partial A_M^{(0)}} \end{pmatrix}, \tag{2.10}$$

Puis on calcule itérativement :



(a) Coefficients trouvés



(b) Valeurs absolue de l'erreur pour chaque coefficient

FIGURE 2.1 – Résultats de la méthode de l'inversion de système

$$\mathbf{A^{(k+1)}} = \mathbf{A^{(k)}} - \alpha \mathbf{g^{(k)}} \tag{2.11}$$

On cherche à trouver le coefficient α optimal qui assure la convergence tout en maximisant la vitesse de convergence. On exprime tout d'abord le gradient en fonction de la matrice $\mathcal{M}$ et du vecteur $\mathbf{b}$ définis précédemment qui sont indépendants de $\mathbf{A}$ et de k :

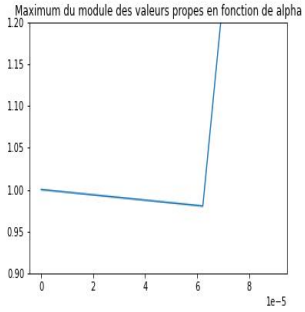
$$\mathbf{g^{(k)}} = \mathcal{M}\mathbf{A^{(k)}} - \mathbf{b} \tag{2.12}$$

Ainsi, l'équation de récurrence (2.11) se réécrit comme une suite arithmético-géométrique de vecteurs :

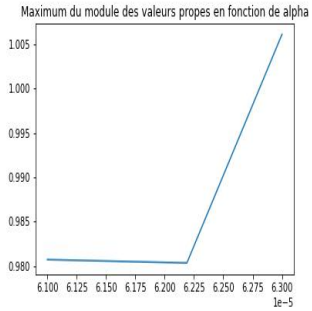
$$\mathbf{A^{(k+1)}} = (\mathcal{I}_\mathbf{M} - \alpha\mathcal{M})\mathbf{A^{(k)}} + \alpha\mathbf{b} \tag{2.13}$$

On en déduit que la suite converge si et seulement si la norme $(\mathcal{R}_\alpha^n)_{n\in\mathbb{N}}$ tend vers 0de le maximum du module des valeurs propres de la matrice $\mathcal{R}_\alpha = \mathcal{I}_M - \alpha\mathcal{M}$ est strictement inférieur à 1. De plus, elle convergera d'autant plus vit que ce maximum est faible. On trace donc ce maximum en fonction de α en figure 2.2. On en déduit la valeur critique $\alpha_c = 6.2807.10^{-5}$, pour laquelle le maximum des modules vaut 1, ainsi que la valeur $\alpha_{min} = 6.2189.10^{-5}$ pour laquelle le maximum des modules est minimum.

On éxécute l'algorithme en parallèle pour les deux valeurs de α trouvées précédemment ainsi que pour une valeur $\alpha_1 = 6.3.10^{-5}$, tel que le maximum des modules des valeurs propres de $\mathcal{R}_\alpha$ soit supérieur à 1. On montre



(a) $\alpha \in [0, 10^{-4}]$



(b) Au voisinage de l'intersection avec 1

FIGURE 2.2 – Maximum des valeurs propres de $\mathcal{R}_\alpha$ en fonction de α

l'évolution de l'erreur en fonction du nombre d'itérations en figure 2.3. On constate que α_{min} donne lieu à une décroissance exponentielle de l'erreur pendant les 2000 premières itérations, qui devient ensuite stationnaire. Tandis que α_1 donne une erreur qui croît exponentiellement car la norme de $(\mathcal{R}_\alpha^n)_{n\in\mathbb{N}}$ diverge exponentiellement. La valeur α_c donne une erreur constante, elle correspond au cas limite entre les 2 cas précédents.

On retient donc les résultats obtenus pour α_{min} que l'on montre en figure 2.4 avec les valeurs absolues des erreurs de chacun par rapport à la valeur théorique. On constate que les erreurs sur chaque coefficient est inférieure à 10^{-16}, on valide donc cette seconde méthode.

2.2 Solutions par réseau de neurones

On cherche à présent à utiliser un réseau de neurones pour approcher la solution de l'équation différentielle. On cherche désormais des solutions approchées sous la forme suivante :

$$\begin{cases} \tilde{\Psi}(x) = \psi_0 + \mathcal{N}(x, P) \\ \mathcal{N}(x, P) = \sum_{j=1}^{H} v_j \sigma(w_j x + b_j) \end{cases} \tag{2.14}$$

$\mathcal{N}(x, P)$ correspond donc à la sortie d'un réseau de neurones dont l'architecture est présentée en figure 2.5, contenant une couche cachée intermédiaire,

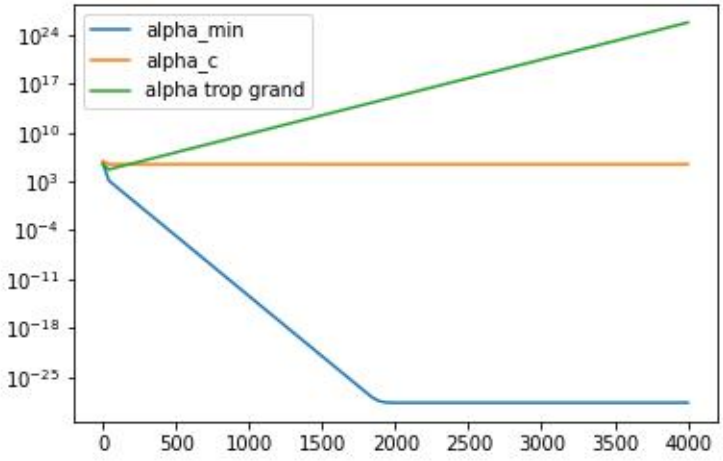
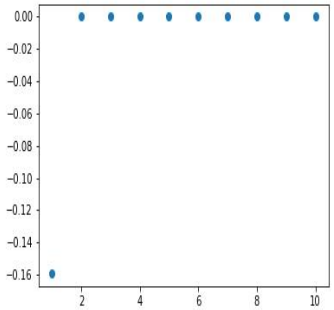
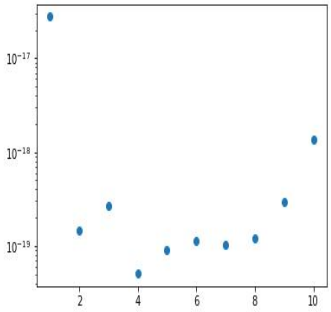


FIGURE 2.3 – Erreurs en fonction du nombre d'itérations pour 3 valeurs de α



(a) Coefficients trouvés



(b) Valeurs absolue de l'erreur pour chaque coefficient

FIGURE 2.4 – Résultats de la méthode de descnte de gradients

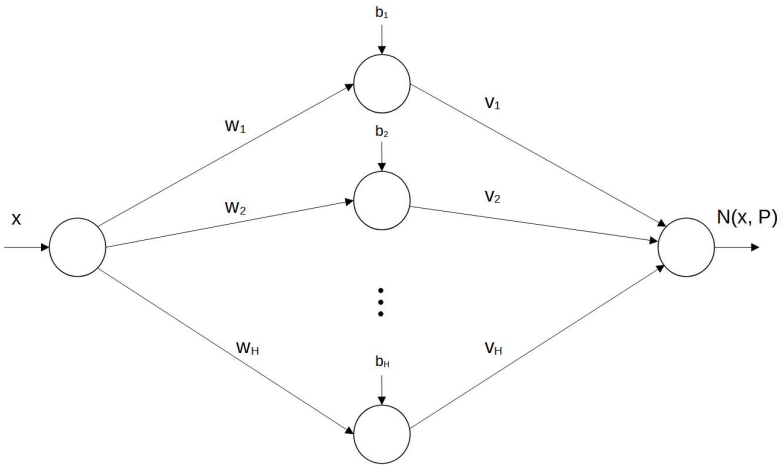


FIGURE 2.5 – Réseau de neurones

qui réalise en sortie une somme pondérée de sigmoïdes, la fonction utilisée est $\forall x \in \mathbf{R}, \sigma(x) = \dfrac{1}{1+e^{-x}}$. Les paramètres P à ajuster sont désormais les coefficients $(w_j)_{j\in[\![1,H]\!]}$, $(b_j)_{j\in[\![1,H]\!]}$ et $(v_j)_{j\in[\![1,H]\!]}$.

Peux-tu donner une référence pour quelqu'un qui cherche ce que tout ceci veut dire ?

On définit une nouvelle fonction d'erreur, calculée à partir des N points suivants : $\forall i \in [\![1, N]\!], x_i = \frac{i-1}{N-1}$

$$E(P) = \sum_{i=1}^{N} \left(\frac{d\tilde{\Psi}}{dx}(x_i) + \cos(2\pi x)\right)^2 \tag{2.15}$$

L'équation (2.15) est-elle correcte ?

On calcule ensuite les expressions analytiques des dérivées partielles de $E(P)$ par rapport à chaque paramètre ajustable, puis on cherche à minimiser cette erreur à l'aide de l'algorithme de descente de gradients.

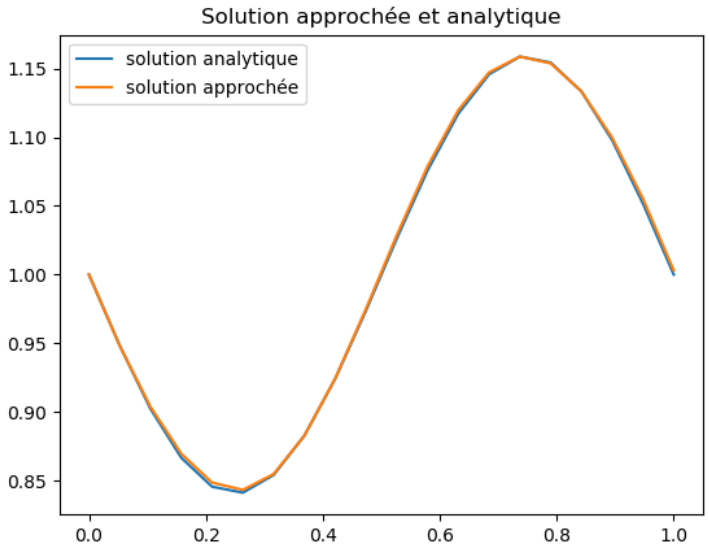


FIGURE 2.6 – estimation de la solution par un réseau de neurones

2.2.1 Résultats obtenus

On initialise l'algorithme avec les paramètres suivants : $(H = 4, N = 20)$ On obtient une erreur de $1,2.10^{-2}$ et une estimation visible en figure 2.6. Cela permet de valider notre modèle sur l'étude à une dimension.

Chapitre 3

Mouvement de précession

On s'intéresse désormais au problème de la précession d'un moment magnétique dans un champ magnétique constant. On le modélise par les équations suivantes pour $t \in [0, 1]$:

$$\begin{cases} \dfrac{dv_x}{dt} = \omega v_y \\ \dfrac{dv_y}{dt} = -\omega v_x \end{cases} \tag{3.1}$$

avec les conditions initiales

$$\begin{cases} v_x(0) = V_0 \\ v_y(0) = 0 \end{cases} \tag{3.2}$$

dont la solution analytique vaut

$$\begin{cases} v_x(t) = V_0 \cos(2\omega t) \\ v_y(t) = -V_0 \sin(2\omega t) \end{cases} \tag{3.3}$$

3.1 Solutions en séries de Fourier

On cherche des solutions numériques approchées sous la forme de séries de Fourier tronquées avec M harmoniques, en posant la forme suivante :

$$\begin{cases} \tilde{v}_x(t) = V_0 + \sum_{m=1}^{M} A_m(\cos(m\omega t) - 1) + B_m \sin(m\omega t) \\ \tilde{v}_y(t) = \sum_{m=1}^{M} -A_m \sin(m\omega t) + B_m(\cos(m\omega t) - 1) \end{cases} \tag{3.4}$$

Les coefficients $(A_m)_{m\in[\![1,M]\!]}$ et $(B_m)_{m\in[\![0,M]\!]}$ sont les paramètres à ajuster. On cherche à obtenir la solution analytique, i.e $\forall m \in [\![0, M]\!], A_m = \delta_1^m$ et $\forall m \in [\![1, M]\!], B_m = 0$. On remarque que le coefficient A_0 n'a aucune influence.

On définit une fonction d'erreur pour ces solutions potentielles, en s'interressant aux N points suivants : $\forall i \in [\![1, N]\!], t_i = \frac{i-1}{N}$:

$$E(P) = \sum_{i=1}^{N} (\frac{d\tilde{v}_x}{dt}(t_i) - \omega\tilde{v}_y(t_i))^2 + (\frac{d\tilde{v}_y}{dt}(t_i) + \omega\tilde{v}_x(t_i))^2 \tag{3.5}$$

On utilise ensuite la méthode de descente de gradients définie précédemment, en calculant les dérivées partielles suivantes : $(\frac{\partial E}{\partial A_l}, \frac{\partial E}{\partial B_l})_{l\in[\![1,M]\!]}$

3.1.1 Résultats obtenus

On initialise l'algorithme avec les paramètres suivants : $(M = 10, N = 100, V_0 = 1, \omega = 2\pi, \alpha = 10^{-6})$ On obtient au bout de 10000 itérations les résultats suivants :
$A = [1.00000255e{+}00, -1.23919994e{-}06, -2.20679520e{-}07, -9.12537244e{-}08, -4.93048945e{-}08, -3.05670278e{-}08, -2.06219718e{-}08, -1.47337251e{-}08, -1.09721848e-08, -8.43076573e-09]$,
$B = [-6.59235880e-07, 3.20274560e-07, 5.70352161e-08, 2.35847707e-08, 1.27429827e{-}08, 7.90013061e{-}09, 5.32980412e{-}09, 3.80797092e{-}09, 2.83579070e{-}09, 2.17895410e-09]$
On constate comme attendu que le coefficient A_0 est très proche de 1 (erreur relative inférieure de 2.5510^{-6}), et que les autres coefficients ont une valeur absolue maximale de 1.2410^{-6}. On peut donc valider notre modèle.

Chapitre 4

Conclusion et perspectives

Bibliographie

[1] C.Bidule and A.Machin, Journal of Computer Power **12** 123 (2020)

[2] C.Truc and T.Chmuc, (2020)