Résolution d'équations différentielles à l'aide de réseaux de neurones

Matthieu Carreau*

Juillet 2022

1 Introduction

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.

2 Equation différentielle d'ordre 1

Soit Ψ une fonction à une variable dont la solution satisfait l'équation différentielle suivante où A désigne ?

$$\begin{cases} \dfrac{d\Psi(x)}{dx} + \cos(2\pi x) = 0 \\ \Psi(0) = A \end{cases} \tag{1}$$

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

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

Qu'est-ce-ça permet de savoir pour la suite ?

2.1 Solutions en séries de Fourier

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

*Encadré par Stam Nicolis et Pascal Thibaudeau

$$\begin{cases} \tilde{\Psi}(x) = A + \mathcal{N}(x, P) \\ \mathcal{N}(x, P) = \sum_{m=1}^{M} A_m \sin(2\pi m x) \end{cases} \quad (3)$$

P représente les coefficients $(A_m)_{m\in[\![1,M]\!]}$ qui sont les paramètres à ajuster. On cherche à obtenir la solution analytique, i.e $\forall m \in [\![1, M]\!], A_m = -\frac{1}{2\pi}\delta_1^m$.

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

$$E = \frac{1}{2}\sum_{i=1}^{N}(\sum_{m=1}^{M} 2\pi m A_m \cos(2\pi m x_i) + cos(2\pi x_i))^2 \quad (4)$$

On calcule alors la dérivée partielle de cette erreur par rapport à chaque paramètre A_l :

$$\frac{\partial E}{\partial A_l} = \sum_{i=1}^{N}(\sum_{m=1}^{M} 2\pi m A_m \cos(2\pi m x_i) + cos(2\pi x_i))2\pi l \cos(2\pi l x_i) \quad (5)$$

Les deux méthodes suivantes ont pour objectif de trouver les coefficient $(A_m)_{m\in[\![1,M]\!]}$ qui minimisent E.

2.1.1 Première méthode : inversion d'un système linéaire

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

$$\mathcal{M} = (r_{m,l})_{(m,l)\in[\![1,M]\!]^2}, \vec{A} = \begin{pmatrix} A_1 \\ A_2 \\ \vdots \\ A_M \end{pmatrix}, \vec{b} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_M \end{pmatrix} \quad (6)$$

$$\forall (m, l) \in [\![1, N]\!]^2, \begin{cases} r_{m,l} = 2\pi m l \sum_{i=1}^{N} cos(2\pi m x_i) cos(2\pi l x_i) \\ b_l = -l \sum_{i=1}^{N} cos(2\pi x_i) cos(2\pi l x_i) \end{cases} \quad (7)$$

On résoud le système linéaire associé en écrivant l'équation matricielle le représentant, soit

$$\mathcal{M}\vec{A} = \vec{b} \Leftrightarrow \vec{A} = \mathcal{M}^{-1}\vec{b} \quad (8)$$

(Attention pour certains paramètres comme $(M = 5, N = 10)$, $\mathcal{M}$ n'est pas inversible).

On initialise l'algorithme avec les paramètres suivants : $(M = 10, N = 100)$
On obtient les résultats suivants :
$A = [-1.59154943e-01, 4.74338450e-19, 1.08420217e-19, 2.71050543e-20, 2.74438675e-19, 1.05032085e-19, 9.82558219e-20, 5.92923063e-20, 5.42101086e-20, 5.42101086e-20]$

On constate comme attendu que le coefficient A_1 est très proche de $-\frac{1}{2\pi}$ (erreur relative de l'ordre de 10^{-16}), et que les autres coefficients ont une valeur absolue maximale de 1.1510^{-17}. On peut donc valider notre modèle.

2.1.2 Seconde méthode : descente de gradients

On définit les paramètres suivants :

$$\alpha > 0, \vec{A}^{(0)} = \begin{pmatrix} A_1^{(0)} \\ A_2^{(0)} \\ \vdots \\ A_M^{(0)} \end{pmatrix}, \vec{g}^{(0)} = \begin{pmatrix} \frac{\partial E^{(0)}}{\partial A_1^{(0)}} \\ \frac{\partial E^{(0)}}{\partial A_2^{(0)}} \\ \vdots \\ \frac{\partial E^{(0)}}{\partial A_M^{(0)}} \end{pmatrix}, \tag{9}$$

Puis on calcule itérativement :

$$\vec{A}^{(k+1)} = \vec{A}^{(k)} - \alpha \vec{g}^{(k)} \tag{10}$$

On initialise l'algorithme avec les paramètres suivants : $(M = 10, N = 100, V_0 = 1, \alpha = 10^{-5})$ On obtient au bout de 10000 itérations les résultats suivants :
$A = [-1.59154943e-01, 1.15066541e-17, 7.99406786e-18, 5.64963221e-18, 4.88181580e-18, 3.85811144e-18, 3.46024465e-18, 2.88560910e-18, 2.81846501e-18, 2.43172156e-18]$
On constate comme attendu que le coefficient A_1 est très proche de $-\frac{1}{2\pi}$ (erreur relative de l'ordre de 10^{-15}), et que les autres coefficients ont une valeur absolue maximale de 1.1510^{-17}. On peut donc valider notre modèle.

2.2 Solutions à l'aide d'un 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 :

$$\begin{cases} \tilde{\Psi}(x) = A + \mathcal{N}(x, P) \\ \mathcal{N}(x, P) = \sum_{j=1}^{H} v_j \sigma(w_j x + b_j) \end{cases} \tag{11}$$

$\mathcal{N}(x, P)$ correspond donc à la sortie d'un réseau de neurones dont l'architecture est présentée en figure 1, contenant une couche cachée intermèdiaire, qui réalise en sortie une somme pondérée de sigmoïdes, la fonction utilisée est $\forall x \in \mathbf{R}, \sigma(x) = \frac{1}{1+e^{-x}}$. Les paramètres P à ajuster sont désormais les coefficients $(w_j)_{j \in [\![1,H]\!]}$, $(b_j)_{j \in [\![1,H]\!]}$ et $(v_j)_{j \in [\![1,H]\!]}$.

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

$$E(P) = \sum_{i=1}^{N} (\frac{d\tilde{\Psi}}{dx}(x_i) + \cos(2\pi x))^2 \tag{12}$$

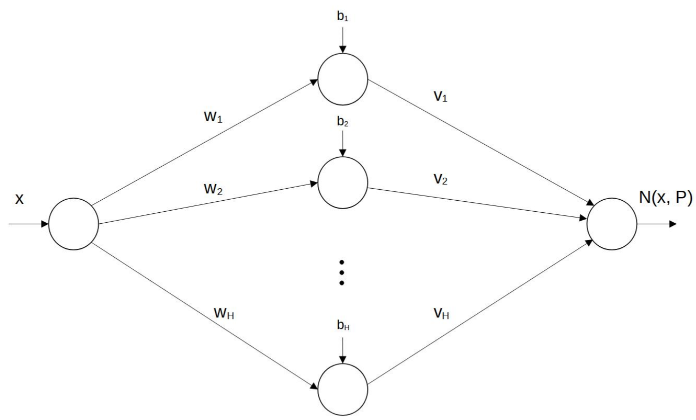

Figure 1 – Réseau de neurones

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.

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. Cela permet de valider notre modèle sur l'étude à une dimension.

3 Etude du cas de deux équations d'ordre 1 couplées, représentant un 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 $t \in [0, 1]$:

$$\left\{ \begin{array}{l} \frac{dv_x}{dt} = \omega v_y \\ \frac{dv_y}{dt} = -\omega v_x \end{array} \right. \left\{ \begin{array}{l} v_x(0) = V_0 \\ v_y(0) = 0 \end{array} \right. \tag{13}$$

Ce problème admet une unique solution analytique :

$$\left\{ \begin{array}{l} v_x(t) = V_0 \cos(2\omega t) \\ v_y(t) = -V_0 \sin(2\omega t) \end{array} \right. \tag{14}$$

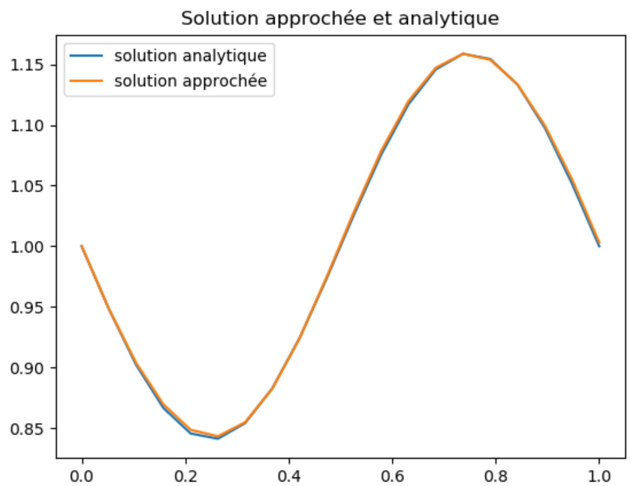

FIGURE 2 – estimation de la solution par un réseau de neurones

3.1 Recherche des 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 :

$$\begin{cases} \tilde{v}_x(t) = V_0 + \sum_{m=1}^{M} A_m(cos(\omega mt) - 1) + B_m \sin(\omega mt) \\ \tilde{v}_y(t) = \sum_{m=1}^{M} -A_m \sin(\omega mt) + B_m(cos(\omega mt) - 1) \end{cases} \tag{15}$$

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 : $\forall i \in [\![1,N]\!], t_i = \frac{i}{N-1}$:

$$E(P) = \sum_{i=1}^{N} (\frac{d\tilde{v}_x}{dt}(t_i) - \omega\tilde{v}_y(t_i))^2 + (\frac{d\tilde{v}_y}{dt}(t_i) + \omega\tilde{v}_x(t_i))^2 \tag{16}$$

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.

4 Conclusion et perspectives

Références

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

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