

# 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

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Equation différentielle d'ordre 1</b>	<b>3</b>
2.1	Solutions en séries de Fourier . . . . .	3
2.1.1	Première méthode : inversion d'un système linéaire . .	4
2.1.2	Seconde méthode : descente de gradients . . . . .	5
2.2	Solutions par réseau de neurones . . . . .	7
2.2.1	Résultats obtenus . . . . .	10
<b>3</b>	<b>Mouvement de précession</b>	<b>11</b>
3.1	Solutions en séries de Fourier . . . . .	11
<b>4</b>	<b>Conclusion et perspectives</b>	<b>15</b>

# 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 pertinents 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  $\Psi$  une fonction réelle à une variable dont la solution satisfait l'équation différentielle suivante, où  $\psi_0$  désigne la valeur initiale de la fonction.

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

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

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

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  $\psi_0$  constant non ajustable vérifiant la condition initiale, et le deuxième  $\mathcal{N}(x, \mathbf{A})$  dépendant des coefficients  $(A_m)_{m \in \llbracket 1, M \rrbracket}$  représentés par le vecteur  $\mathbf{A}$ , construit de façon à ne pas influencer la valeur initiale de la fonction.

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

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 :  $\forall i \in \llbracket 1, N \rrbracket, x_i = \frac{i-1}{N}$

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

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

$$\forall l \in \llbracket 1, M \rrbracket, \frac{\partial E}{\partial A_l} = 0 \quad (2.5)$$

Or ces dérivées partielles sont données pour  $l \in \llbracket 1, M \rrbracket$  par :

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

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

### 2.1.1 Première méthode : inversion d'un système linéaire

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

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

$$\forall (m, l) \in \llbracket 1, N \rrbracket^2, \begin{cases} r_{m,l} = 2\pi m l \sum_{i=1}^N \cos(2\pi m x_i) \cos(2\pi l x_i) \\ b_l = -l \sum_{i=1}^N \cos(2\pi x_i) \cos(2\pi l x_i) \end{cases} \quad (2.8)$$

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

$$\mathcal{M}\mathbf{A} = \mathbf{b} \Leftrightarrow \mathbf{A} = \mathcal{M}^{-1}\mathbf{b} \quad (2.9)$$

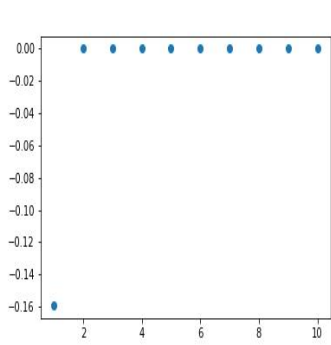
On réalise l'implémentation en python, en instanciant les matrices définies précédemment et à l'aide de la bibliothèque *numpy*. On peut constater que la matrice  $\mathcal{M}$  n'est pas toujours inversible selon le choix de  $M$  et  $N$ . En effet, on peut montrer (à rajouter en annexe) que c'est 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 intéressant 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

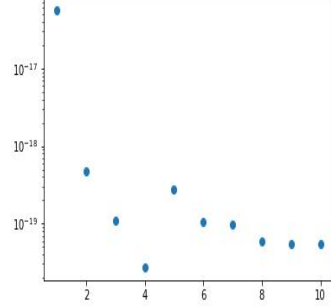
On définit les paramètres suivants :

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

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

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

On cherche à trouver le coefficient  $\alpha$  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  $k$  :

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

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

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

On en déduit que la suite converge si et seulement si la norme  $(\mathcal{R}_\alpha^n)_{n \in \mathbb{N}}$  tend vers 0. 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 vite que ce maximum est faible. On trace donc ce maximum en fonction de  $\alpha$  en figure 2.2. On en déduit la valeur critique  $\alpha_c = 6.2807 \cdot 10^{-5}$ , pour laquelle le maximum des modules vaut 1, ainsi que la valeur  $\alpha_{min} = 6.2189 \cdot 10^{-5}$  pour laquelle le maximum des modules est minimum.

On exécute l'algorithme en parallèle pour les deux valeurs de  $\alpha$  trouvées précédemment ainsi que pour une valeur  $\alpha_1 = 6.3 \cdot 10^{-5}$ , tel que le maximum des modules des valeurs propres de  $\mathcal{R}_\alpha$  soit supérieur à 1. On montre



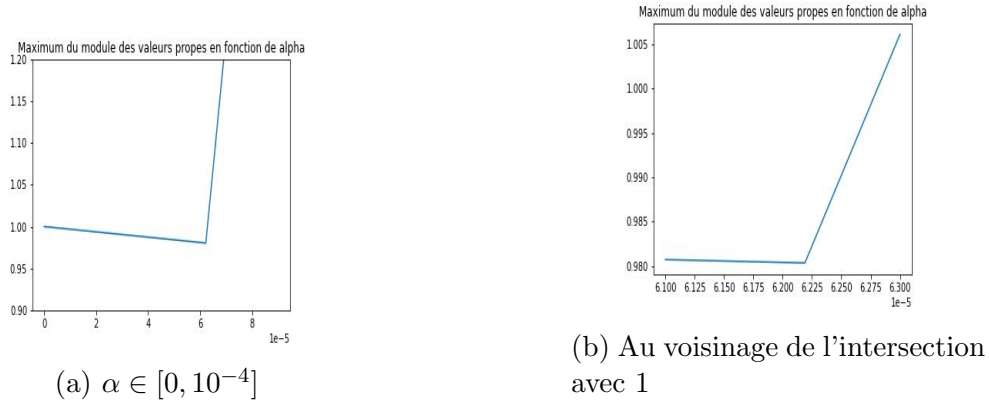


FIGURE 2.2 – Maximum des valeurs propres de  $\mathcal{R}_\alpha$  en fonction de  $\alpha$

l'évolution de l'erreur en fonction du nombre d'itérations en figure 2.3. On constate que  $\alpha_{min}$  donne lieu à une décroissance exponentielle de l'erreur pendant les 2000 premières itérations, qui devient ensuite stationnaire. Tandis que  $\alpha_1$  donne une erreur qui croît exponentiellement car la norme de  $(\mathcal{R}_\alpha^n)_{n \in \mathbb{N}}$  diverge exponentiellement. La valeur  $\alpha_c$  donne une erreur constante, elle correspond au cas limite entre les 2 cas précédents.

On retient donc les résultats obtenus pour  $\alpha_{min}$  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 :

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

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

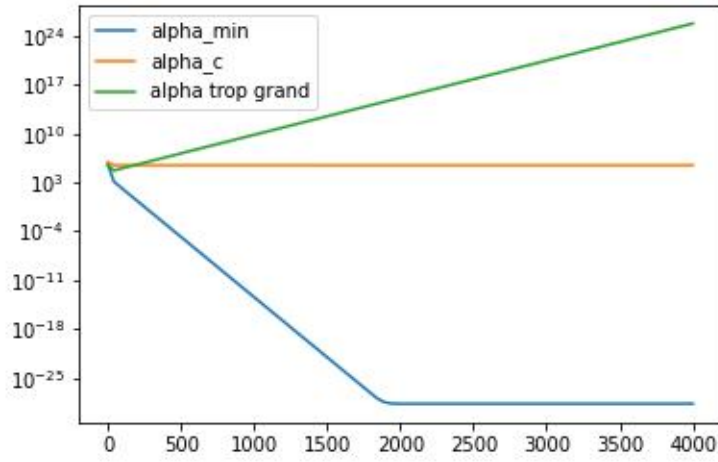
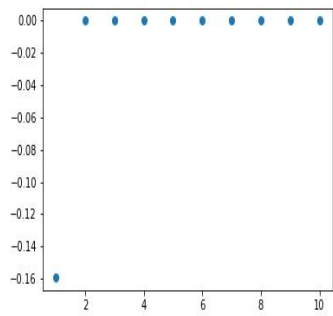
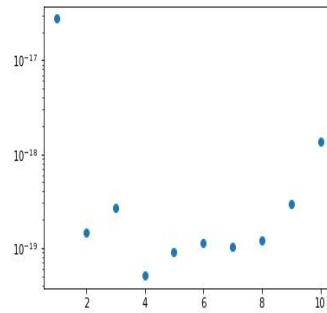


FIGURE 2.3 – Erreurs en fonction du nombre d'itérations pour 3 valeurs de  $\alpha$



(a) Coefficients trouvés



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

FIGURE 2.4 – Résultats de la méthode de descnte de gradients

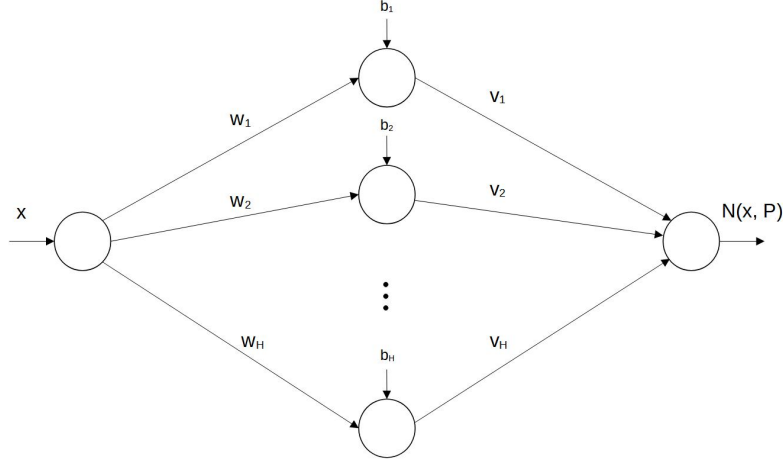


FIGURE 2.5 – Réseau de neurones

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

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 :  $\forall i \in \llbracket 1, N \rrbracket, x_i = \frac{i-1}{N-1}$

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

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.

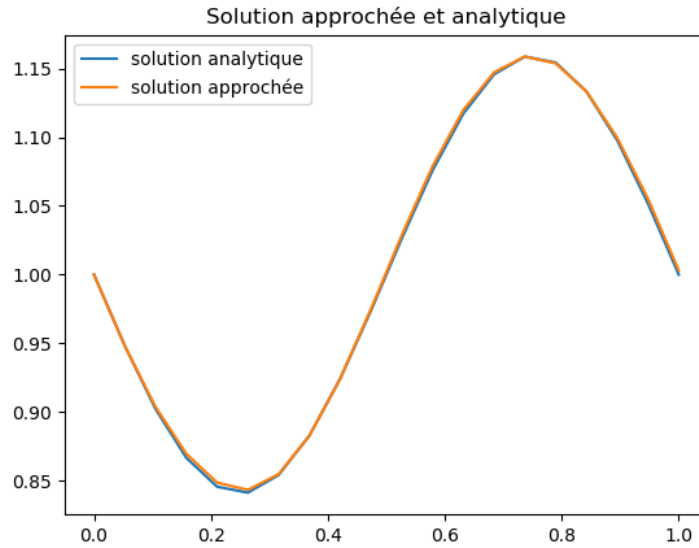


FIGURE 2.6 – estimation de la solution par un réseau de neurones

### 2.2.1 Résultats obtenus

On initialise l'algorithme avec les paramètres suivants : ( $H = 4, N = 20$ )  
On obtient une erreur de  $1,2 \cdot 10^{-2}$  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  $t \in [0, 1]$  :

$$\begin{cases} \frac{dv_x}{dt} = \omega v_y \\ \frac{dv_y}{dt} = -\omega v_x \end{cases} \quad (3.1)$$

avec les conditions initiales

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

dont la solution analytique vaut

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

### 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 :

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

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

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

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

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

On peut se ramener à l'écriture du cas à une dimension définissant le vecteur  $\mathbf{P}$  comme la concaténation des vecteurs  $\mathbf{A}$  et  $\mathbf{B}$  et le vecteur  $\mathbf{g}$  comme le vecteur contenant les dérivées partielles de  $E$  par rapport à chaque composante de  $\mathbf{P}$ .

$$\mathbf{P}^{(k)} = \begin{pmatrix} A_1^{(k)} \\ \vdots \\ A_M^{(k)} \\ B_1^{(k)} \\ \vdots \\ B_M^{(k)} \end{pmatrix}, \mathbf{g}^{(k)} = \begin{pmatrix} \frac{\partial E^{(k)}}{\partial A_1^{(k)}} \\ \vdots \\ \frac{\partial E^{(k)}}{\partial A_M^{(k)}} \\ \frac{\partial E^{(k)}}{\partial B_1^{(k)}} \\ \vdots \\ \frac{\partial E^{(k)}}{\partial B_M^{(k)}} \end{pmatrix}, \quad (3.6)$$

Il existe alors une nouvelle matrice  $\mathcal{M}$  d'ordre  $2M$  ainsi qu'un nouveau vecteur  $\mathbf{b}$  de taille  $2M$  tels que les équations définissant la descente de gradients soient les suivantes :

$$\mathbf{P}^{(k+1)} = \mathbf{P}^{(k)} - \alpha \mathbf{g}^{(k)} \quad (3.7)$$

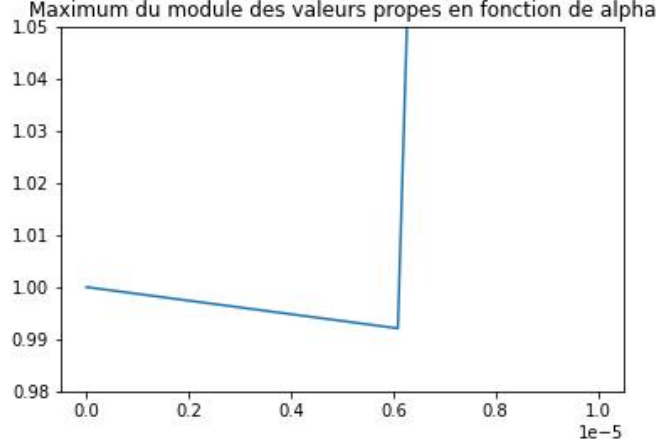


FIGURE 3.1 – Evolution du maximum du module en fonction de  $\alpha$

$$\mathbf{g}^{(k)} = \mathcal{M}\mathbf{P}^{(k)} - \mathbf{b} \quad (3.8)$$

$$\mathbf{P}^{(k+1)} = (\mathcal{I}_M - \alpha\mathcal{M})\mathbf{P}^{(k)} + \alpha\mathbf{b} \quad (3.9)$$

On réalise la même étude qu'en dimension 1 pour la recherche du coefficient  $\alpha_{min}$  qui minimise le maximum des modules des valeurs propres de  $\mathcal{R}_\alpha = \mathcal{I}_M - \alpha\mathcal{M}$ , ainsi que le coefficient  $\alpha_c$  à ne pas dépasser, à partir duquel ce maximum est supérieur à 1 et la suite diverge. On trouve les valeurs suivantes  $\alpha_{min} = 6.0906 \cdot 10^{-6}$  et  $\alpha_c = 6.1146 \cdot 10^{-6}$ , et l'évolution du maximum en fonction de alpha est montré en figure 3.1.

On exécute l'algorithme en choisissant la valeur  $\alpha_{min}$  précédente et on montre l'évolution de l'erreur en fonction du nombre d'itérations en figure 3.2. On constate comme dans le cas à 1 dimension que  $\alpha_{min}$  donne lieu à une décroissance exponentielle de l'erreur, cette fois pendant les 4500 premières itérations, qui devient ensuite quasiment stationnaire, de l'ordre de  $10^{-25}$ .

Les coefficients finaux de  $P$  ainsi que les valeurs absolues de leurs erreurs sont représentées en figure 3.3. On peut constater que l'on retrouve bien les coefficients attendus, avec une erreur au maximum de l'ordre de  $10^{-15}$ .

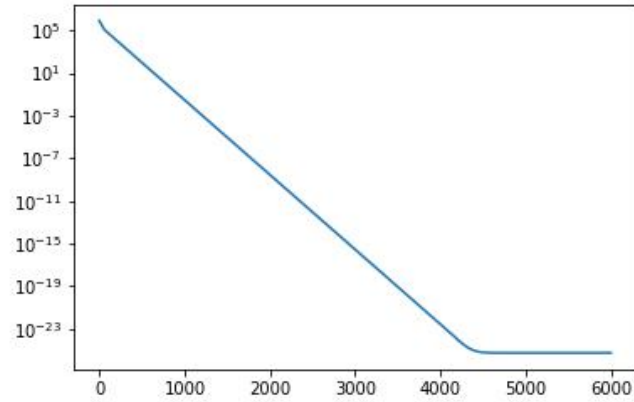
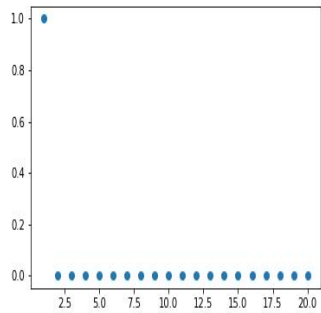
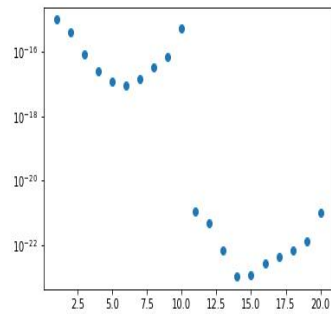


FIGURE 3.2 – Valeur de la fonction d'erreur en fonction du nombre d'itérations



(a) Coefficients trouvés



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

FIGURE 3.3 – Résultats de la méthode de descente de gradients pour le mouvement de précession



## 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)