Université catholique de Louvain

Ecole de physique

Simulation numérique en physique [lphy2371]

Méthodes spectrales

Auteurs :
Valéry Materne
Arnaud Schils

Enseignant :
Pr. Bernard Piraux

Décembre 2016

Première partie

Exercice d'introduction : 3 méthodes spectrales

1.1 Introduction

Nous considérons l'équation différentielle suivante :

$$u_{xx}(x) + u_x(x) - 2u(x) + 2 = 0 \tag{1}$$

sur le domaine $-1 \leq x \leq 1$ et avec les condition aux frontières $u(-1) = u(1) = 0$.

Sa solution analytique exacte est :

$$u(x) = 1 - \frac{\sinh(2)e^x + \sinh(1)e^{-2x}}{\sinh(3)} \ . \tag{2}$$

Nous souhaitons obtenir une approximation de celle-ci par un développement tronqué de polynômes de Tchebychev :

$$v(x) = \sum_{k=0}^{N} a_k T_k(x) \ . \tag{3}$$

1.2 Propriétés des polynômes de Tchebychev

Le polynôme de Tchebychev de degré n, $T_n(x)$, est défini par

$$T_n(x) = \cos(n \arccos(x)) \ , \tag{4}$$

$$T_n(\pm 1) = (\pm 1)^n \ . \tag{5}$$

Relation d'orthogonalité :

$$\int_{-1}^{1} T_m(x) T_n(x) (1 - x^2)^{-1/2} dx = \frac{\pi}{2} c_n \delta_{mn} \tag{6}$$

avec $c_0 = 2$ et $c_n = 1$ pour $n > 0$.

Relations de récurrence :

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \quad , n \geq 1 \ , \tag{7}$$

$$2T_n(x) = \frac{T'_{n+1}(x)}{n+1} - \frac{T'_{n-1}(x)}{n-1} \quad , n \geq 2 \ , \tag{8}$$

avec $T_0(x) = 1$, $T_1(x) = x$ et $T'_0(x) = 0$, $T'_1(x) = T_0(x)$, $T'_2(x) = 4T_1(x)$.

1.3 Calcul des coefficients du développement du résidu

Nous injectons la série tronquée (3) dans l'équation différentielle de départ (1) et nous appelons le résultat : le résidu $R(x)$. Nous le redéfinissons selon une nouvelle série tronquée de polynômes de Tchebychev :

$$R(x) = v_{xx}(x) + v_x(x) - 2v(x) + 2 \quad (9)$$

$$= \sum_{k=0}^{N} A_k T_k(x) \; . \quad (10)$$

Nous calculons ensuite ces nouveaux coefficients A_k en fonction des a_k. Pour ce faire nous commençons par exprimer les dérivées de v en fonction des polynômes de Tchebychev.

On recherche les formes suivantes :

$$v_x(x) = \sum_{k=0}^{N} b_k T_k(x) \quad (11)$$

$$v_{xx}(x) = \sum_{k=0}^{N} c_k T_k(x) \quad (12)$$

$$(13)$$

Dérivée première $v_x(x)$

Pour ce faire, on part de l'équation (3), on a :

$$v_x(x) = \sum_{k=0}^{N} a_k T'_k(x) \; . \quad (14)$$

On doit trouver l'expression des $T'_k(x)$ en fonction des $T_k(x)$ pour $k \geq 0$. Pour ce faire, on utilise la relation de récurrence (8) et le fait que $T'_0(x) = 0$, $T'_1(x) = T_0(x)$ et $T'_2(x) = 4T_1(x)$. Si on pose $k = n + 1$, on obtient :

$$T'_k(x) = k \left(2T_{k-1}(x) + \frac{T'_{k-2}(x)}{k-2} \right) \; . \quad (15)$$

En la réinjectant dans son terme de droite, par récurrence, on obtient deux cas :

k impair

$$T'_k(x) = 2k \left(T_{k-1}(x) + T_{k-3}(x) + \cdots + T_4(x) + T_2(x) + T_0(x)/2 \right) \; , \quad (16)$$

k pair

$$T'_k(x) = 2k \left(T_{k-1}(x) + T_{k-3}(x) + \cdots + T_5(x) + T_3(x) + T_1(x) \right) \; . \quad (17)$$

On peut réécrire les coefficients b_k en fonction des a_k sous la forme :

$$\begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{N-1} \\ b_N \end{pmatrix} = D \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{N-1} \\ a_N \end{pmatrix} \tag{18}$$

où D est appelée la matrice dérivée.

On va maintenant définir cette matrice. En égalant les équations (11) et (14) et en remplaçant les expressions de $T'_k(x)$ par (16) et (17), on obtient les valeurs de b_k suivantes :

k impair

$$b_k = 2 \sum_{n=k-1}^{\frac{N-1}{2}} (2n) a_{2n} \ , \ si \ N \ est \ impair \tag{19}$$

$$b_k = 2 \sum_{n=k-1}^{\frac{N}{2}-1} (2n) a_{2n} \ , \ si \ N \ est \ pair \tag{20}$$

k pair

$k \geq 2$

$$b_k = 2 \sum_{n=k/2}^{\frac{N-1}{2}} (2n+1) a_{2n+1} \ , \ si \ N \ est \ impair \tag{21}$$

$$b_k = 2 \sum_{n=k/2}^{\frac{N}{2}-1} (2n+1) a_{2n+1} \ , \ si \ N \ est \ pair \tag{22}$$

Pour le cas $k = 0$, b_0 est égal à la moitié de (21) ou (22) selon la parité de N.

A partir de ces séries, il est possible de définir les éléments de la matrice D par l'expression explicite suivante :

$$D_{ij} = \begin{cases} 2j\alpha_i & \text{si } i < j,\ i+j \text{ impair} \ , \\ 0 & \text{sinon} \ , \end{cases} \tag{23}$$

avec

$$\alpha_i = \begin{cases} \frac{1}{2} & \text{si } i = 0 \ , \\ 1 & \text{sinon} \ , \end{cases} \tag{24}$$

où $0 \leq i, j \leq N$.

On a implémenté cette matrice dans Matlab, valide pour tout N, voici le cas $N = 4$:

$$\begin{pmatrix} 0 & 1 & 0 & 3 & 0 \\ 0 & 0 & 8 & 0 & 8 \\ 0 & 0 & 0 & 6 & 0 \\ 0 & 0 & 0 & 0 & 8 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} . \tag{25}$$

Dérivée seconde $v_{xx}(x)$

On part de l'équation (3), on a :

$$v_{xx}(x) = \sum_{k=0}^{N} a_k T_k''(x) . \tag{26}$$

Or selon définition de $v_x(x)$ en fonction des $T_k(x)$, on a :

$$v_{xx}(x) = \left(\sum_{k=0}^{N} a_k T_k'(x) \right)' = \left(\sum_{k=0}^{N} b_k T_k(x) \right)' = \sum_{k=0}^{N} b_k T_k'(x) . \tag{27}$$

On peut donc réexprimer $v_{xx}(x)$ en fonction des $T_k(x)$ par la relation de récurrence (8) comme lors du calcul de la dérivée première, on a alors

$$v_{xx}(x) = \sum_{k=0}^{N} c_k T_k(x) \tag{28}$$

avec les coefficients c_k :

$$\begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{N-1} \\ c_N \end{pmatrix} = D \cdot \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{N-1} \\ b_N \end{pmatrix} = D^2 \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{N-1} \\ a_N \end{pmatrix} . \tag{29}$$

On défini les éléments de la matrice D^2 à partir de ceux de D, on obtient l'expression explicite suivante :

$$D^2_{ij} = \begin{cases} j(i+j)(j-i)\alpha_i & \text{si } i < j,\ i+j \text{ pair} , \\ 0 & \text{sinon} , \end{cases} \tag{30}$$

avec

$$\alpha_i = \begin{cases} \frac{1}{2} & \text{si } i = 0 , \\ 1 & \text{sinon} , \end{cases} \tag{31}$$

où $0 \leq i, j \leq N$.

Pour l'implémentation dans Matlab valable pour tout N, on peut également simplement appliquée deux fois la matrice dérivée D (multiplication matricielle) sur le vecteur contenant les a_k. Cette opération est cependant numériquement plus lente. On obtient pour $N = 4$:

$$\begin{pmatrix} 0 & 0 & 4 & 0 & 32 \\ 0 & 0 & 0 & 24 & 0 \\ 0 & 0 & 0 & 0 & 48 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix} . \tag{32}$$

Expression des coefficients du résidu

On peut réécrire le résidu $R(x)$:

$$\begin{aligned} R(x) &= v_{xx}(x) + v_x(x) - 2v(x) + 2 \,, & (33) \\ &= \sum_{k=0}^{N} (c_k + b_k - 2a_k)\, T_k(x) + 2T_0(x) \,, & (34) \\ &= \sum_{k=0}^{N} A_k T_k(x) \,, & (35) \end{aligned}$$

avec les A_k défini en fonction des a_k :

$$\begin{pmatrix} A_0 \\ A_1 \\ \vdots \\ A_{N-1} \\ A_N \end{pmatrix} = \left(D^2 + D - 2\mathbb{I}\right) \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{N-1} \\ a_N \end{pmatrix} + \begin{pmatrix} 2 \\ 0 \\ \vdots \\ 0 \\ 0 \end{pmatrix} . \tag{36}$$

Après implémentation dans Matlab, on obtient pour le cas $N = 4$:

$$\begin{pmatrix} A_0 \\ A_1 \\ A_2 \\ A_3 \\ A_4 \end{pmatrix} = \begin{pmatrix} -2 & 1 & 4 & 3 & 32 \\ 0 & -2 & 4 & 24 & 8 \\ 0 & 0 & -2 & 6 & 48 \\ 0 & 0 & 0 & -2 & 8 \\ 0 & 0 & 0 & 0 & -2 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} + \begin{pmatrix} 2 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} . \tag{37}$$

1.4 Conditions aux frontières

On impose les conditions aux frontières et en utilisant la définition (5), on a :

$$v(-1) = \sum_{k=0}^{N} a_k T_k(-1) = \sum_{k=0}^{N} a_k (-1)^k = 0 \ , \tag{38}$$

$$v(1) = \sum_{k=0}^{N} a_k T_k(1) = \sum_{k=0}^{N} a_k = 0 \ . \tag{39}$$

C'est à dire :

$$C \cdot \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{N-1} \\ a_N \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \ , \tag{40}$$

avec les éléments de la matrice C qui sont défini par l'expression explicite suivante :

$$C_{ij} = \begin{cases} (-1)^j & \text{si } i = 0 \ , \\ 1 & \text{si } i = 1 \ , \end{cases} \tag{41}$$

où $i = 0, 1$ et $0 \leq j \leq N$.

Après implémentation dans Matlab, on obtient pour le cas $N = 4$:

$$\begin{pmatrix} 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \end{pmatrix} \tag{42}$$

1.5 Méthodes spectrales

Nous souhaitons obtenir un résidu nul, c'est-à-dire $A_k = 0$ pour tout k, tout en satisfaisant les conditions aux frontières (40). Nous avons donc un système surdéterminé à résoudre. En effet, il y a $N + 1$ équations pour les A_k et deux équations pour les conditions aux frontières, alors qu'il n'y a que $N + 1$ paramètres a_k à déterminer.

Nous allons résoudre ce système surdéterminé par trois méthodes spectrales différentes. Celles-ci diffèrent sur la façon dont elles l'approximent.

1.5.1 Méthode Tau

On a besoin que l'expression de $R(x)$ soit orthogonale à $T_k(x)$ pour $k = 0, 1, .., N - 2$:

$$\int_{-1}^{1} \frac{R(x)T_k(x)}{\sqrt{1-x^2}}dx = 0 \ , \tag{43}$$

c'est-à-dire, étant donné l'équation d'orthogonalité des polynômes de Tchebychev (6), $A_k = 0$ pour $k = 0, 1, ..N-2$.

On prend les $N-1$ premières équations de (36) avec $A_k = 0$ et on y ajoute les deux équations des conditions aux frontières (40). On obtient un nouveau système linéaire déterminé à $N+1$ équations et $N+1$ inconnues (a_k) à résoudre. Après implémentation dans Matlab du cas général (valable pour tout N), voici l'exemple pour $N = 4$:

$$\begin{pmatrix} -2 & 1 & 4 & 3 & 32 \\ 0 & -2 & 4 & 24 & 8 \\ 0 & 0 & -2 & 6 & 48 \\ 1 & -1 & 1 & -1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{pmatrix} \cdot \begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} = \begin{pmatrix} -2 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix} \ . \tag{44}$$

Après résolution de ce système avec Matlab (package `LAPACK` et non commande `inv()`) , on obtient :

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} = \begin{pmatrix} 0.2724 \\ -0.0444 \\ -0.2562 \\ 0.0444 \\ -0.0162 \end{pmatrix} \ . \tag{45}$$

1.5.2 Méthode Galerkin

A partir des polynômes de Tchebychev, on définit des nouvelles fonctions de base qui satisfont les conditions aux frontières :

$$\Phi_2(x) = T_2(x) - T_0(x) \ , \tag{46}$$
$$\Phi_3(x) = T_3(x) - T_1(x) \ , \tag{47}$$
$$\vdots \tag{48}$$
$$\Phi_N(x) = T_N(x) - \begin{cases} T_0(x) & \text{si N pair ,} \\ T_1(x) & \text{si N impair .} \end{cases} \tag{49}$$

On veut que $R(x)$ soit orthogonale à $\Phi_l(x)$ avec $l = 2, 3, .., N$:

$$\int_{-1}^{1} \frac{R(x)\Phi_l(x)}{\sqrt{1-x^2}}dx = 0 \ . \tag{50}$$

En remplaçant l'expression de $R(x)$ en fonction des $T_k(x)$, on a :

$$\sum_{k=0}^{N} A_k \int_{-1}^{1} \frac{T_k(x)\Phi_l(x)}{\sqrt{1-x^2}} dx = 0 \ , \tag{51}$$

c'est-à-dire :

$$\sum_{k=0}^{N} A_k \begin{cases} \int_{-1}^{1} \frac{T_k(x)(T_l(x)-T_0(x))}{\sqrt{1-x^2}} dx = 0 & \text{si l pair ,} \\ \int_{-1}^{1} \frac{T_k(x)(T_l(x)-T_1(x))}{\sqrt{1-x^2}} dx = 0 & \text{si l impair .} \end{cases} \tag{52}$$

On obtient donc les conditions :

$$\begin{cases} A_l - 2A_0 = 0 & \text{si l pair ,} \\ A_l - A_1 = 0 & \text{si l impair ,} \end{cases} \tag{53}$$

où $l = 2, 3, ..., N$.

On peut réécrire ces conditions sous forme matricielle, on a donc les $N-1$ équations suivantes :

$$G \cdot \begin{pmatrix} A_0 \\ A_1 \\ \vdots \\ A_{N-1} \\ A_N \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \ . \tag{54}$$

avec

$$G_{ij} = \begin{cases} -2 & \text{si } j = 0, \ i \text{ pair ,} \\ -1 & \text{si j =1, } i \text{ impair ,} \\ 1 & \text{si } j - i = 2 \ , \end{cases} \tag{55}$$

où $i = 0, ..., N-2$ et $j = 0, ..., N$.

On remplace les A_k par leurs expressions (36) et on ajoute les deux équations des conditions aux frontières (40) pour obtenir un système déterminé à $N+1$ équations et $N+1$ inconnues.

Après implémentation dans Matlab du cas général (valable pour tout N), on obtient le résultat suivant pour $N = 4$:

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} = \begin{pmatrix} 0.2741 \\ -0.0370 \\ -0.2593 \\ 0.0370 \\ -0.0148 \end{pmatrix} \ . \tag{56}$$

1.5.3 Méthode Collocation (pseudo-spectrale)

On prend $R(x_i) = 0$ avec $x_i = \cos\left(\frac{i\pi}{N}\right)$ pour $i = 1, ..., N-1$. On peut donc réécrire l'expression du résidu en terme de polynômes de Tchebychev (10) et selon leur définition (4), on obtient :

$$R(x_i) = \sum_{k=0}^{N} A_k T_k(x_i) = \sum_{k=0}^{N} A_k T_k\left(\cos\left(\frac{i\pi}{N}\right)\right) = \sum_{k=0}^{N} A_k \cos\left(\frac{ki\pi}{N}\right) = 0 \ , \tag{57}$$

et sous forme matricielle :

$$P \cdot \begin{pmatrix} A_0 \\ A_1 \\ \vdots \\ A_{N-1} \\ A_N \end{pmatrix} = \begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix} \ , \tag{58}$$

avec

$$P_{ij} = \cos\left(\frac{ij\pi}{N}\right) \ , \tag{59}$$

où $i = 1, ..., N-1$ et $j = 0, ..., N$.

On remplace les A_k par leurs expressions (36) et on ajoute les deux équations des conditions aux frontières (40) pour obtenir un système déterminé à $N+1$ équations et $N+1$ inconnues.

Après implémentation dans Matlab du cas général (valable pour tout N), on obtient le résultat suivant pour $N = 4$:

$$\begin{pmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \end{pmatrix} = \begin{pmatrix} 0.2743 \\ -0.0371 \\ -0.2600 \\ 0.0371 \\ -0.0143 \end{pmatrix} \ . \tag{60}$$

1.6 Méthode aux différences finies

On va résoudre le problème initial par la méthode aux différences finies pour pouvoir comparer son efficacité par rapport aux méthodes spectrales. Pour ce faire, on discrétise l'équation différentielle de départ (1). Les dérivées sont remplacées par leurs expressions en différences finies. Soient $u(x_i) \equiv U_i$ et h le pas d'espace, on a :

$$\frac{U_{i+1} - 2U_i + U_{i-1}}{h^2} + \frac{U_{i+1} - U_{i-1}}{2h} - U_i + 2 + \mathcal{O}(h^2) = 0 \ , \tag{61}$$

où $i = 1, ..., N$. Nous avons pris des différences centrées pour les dérivées avec leur formule à trois points d'ordre $\mathcal{O}(h^2)$.

On défini $x_i = ih$ avec $i = 0, ..., N+1$ où $(N+1)h$ est la longueur du domaine $-1 \leq x \leq 1$. La discrétisation est donc réalisée sur $N + 2$ points de grille. Les conditions aux frontières sont les suivantes :

$$U_0 = U_{N+1} = 0 \, . \tag{62}$$

En reformulant l'équation (61), nous obtenons l'équation tronquée suivante :

$$(2 + h)U_{i+1} - 4(h^2 + 1)U_i + (2 - h)U_{i-1} + 4h^2 = 0 \, . \tag{63}$$

Sous forme matricielle, en tenant compte des conditions aux frontières, on obtient un système linéaire à matrice tri-diagonale :

$$\begin{pmatrix} -4(h^2+1) & (2+h) & & & & \\ (2-h) & -4(h^2+1) & (2+h) & & & \\ & \ddots & \ddots & \ddots & & \\ & & (2-h) & -4(h^2+1) & (2+h) & \\ & & & (2-h) & -4(h^2+1) & \end{pmatrix} \cdot \begin{pmatrix} U_1 \\ U_2 \\ \vdots \\ U_{N-1} \\ U_N \end{pmatrix} = \begin{pmatrix} -4h^2 \\ -4h^2 \\ \vdots \\ -4h^2 \\ -4h^2 \end{pmatrix} \, . \tag{64}$$

On le résoud sous Maltab avec le package `LAPACK` et non la commande `inv()` qui est source d'erreur.

1.7 Comparatif des différentes méthodes

Pour comparer l'efficacité des différentes méthodes spectrales, on a mesuré l'erreur absolue maximale commise par celles-ci par rapport à la solution analytique pour un nombre de polynômes de Tchebychev ($N + 1$, équation (3)) donné. Le maximum de l'erreur a été pris parmi mille mesures d'erreur, entre la solution numérique et la solution analytique, équidistantes sur le domaine de l'équation différentielle $-1 \leq x \leq 1$.

L'erreur absolue maximale de la méthode aux différences finies d'ordre $\mathcal{O}(h^2)$ a également été mesurée. Cette mesure a été prise pour différents nombres de points de grille équidistants sur l'intervalle du domaine de définition de l'équation différentielle. Le maximum de l'erreur a été pris parmi toutes les valeurs d'erreur, entre la solution numérique et la solution analytique, aux différents points de grille.

On constate sur le graphique de la figure 1 que l'erreur maximale des méthodes spectrales, en fonction du nombre de polynômes utilisé, décroit bien plus rapidement que celle pour la méthode aux différences finies, fonction du nombre de points de grille utilisé pour la discrétisation. Sur ce graphique, l'erreur maximale est représentée sur une échelle logarithmique. Les méthodes spectrales tendent donc vers la solution exacte de manière exponentielle. Tandis que l'erreur de la méthode aux différences finies tend vers zéro de manière polynomiale.

En outre, les méthodes Galerkin et Collocation ont une précision un peu meilleur que la méthode Tau. En particulier pour un faible nombre de polynômes utilisés dans la série tronquée (3).

Enfin, on note qu'à partir de 18 polynômes utilisés pour les méthodes spectrales, l'erreur maximale atteint un minimum à environ 10^{-15}. Ce minimum représente l'erreur machine due à la représentation des nombres limité sous Matlab.

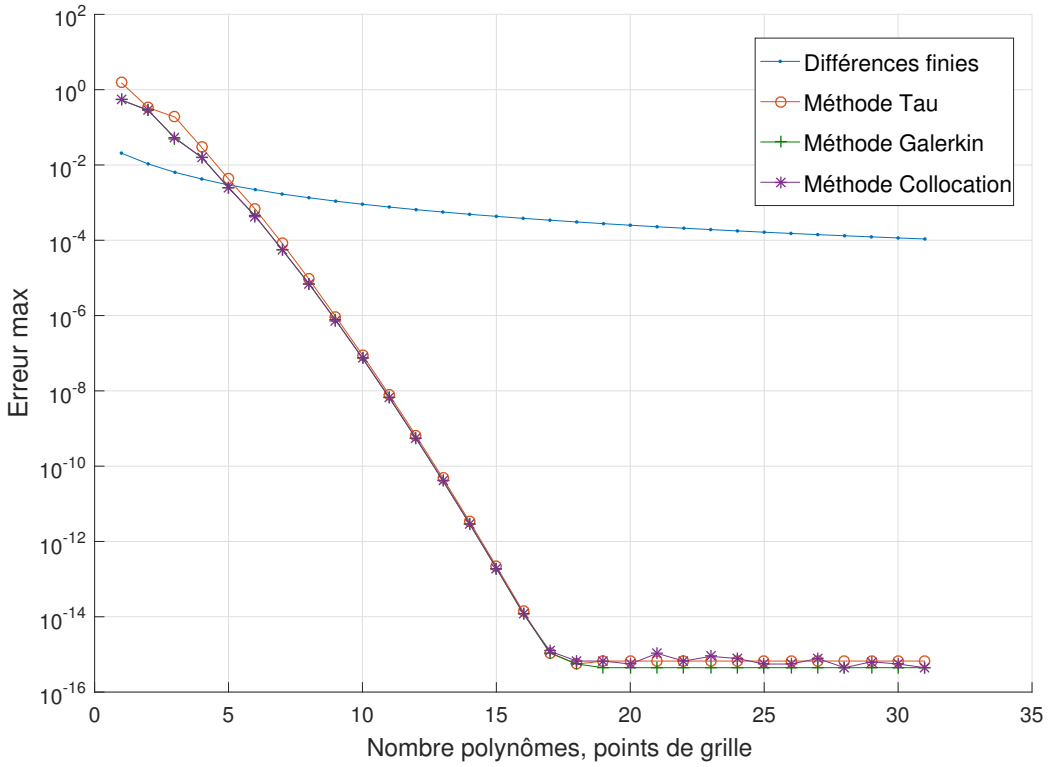

FIGURE 1 – Comparaison de l'erreur maximale des différentes méthodes numériques avec la solution analytique.

Deuxième partie

Exercice sur l'équation de Schrödinger

2.1 Equation de Schrödinger stationnaire

2.1.1 Introduction

Dans les unités atomiques, l'équation de Schrödinger stationnaire à une dimension s'écrit :

$$\left[\frac{1}{2}p^2 + V(x)\right] |\phi(x)\rangle = E\,|\phi(x)\rangle \tag{1}$$

avec $m_e = \hbar = |e| = \frac{1}{4\pi\epsilon_0} = 1$.

On remplace le carré de l'opérateur impulsion par $p^2 = -\frac{\partial^2}{\partial x^2}$ et on utilise le potentiel suivant :

$$V(x) = -V_0 e^{-\alpha x^2} \ . \tag{2}$$

Pour résoudre cette équation (1), on utilise un développement tronqué de fonctions d'Hermite :

$$\phi(x) = \sum_{k=0}^{N} a_k \varphi_k(x) \ , \tag{3}$$

où les fonctions d'Hermite :

$$\varphi_k(x) = c_k H_k(x) e^{-\frac{x^2}{2}} \ , \tag{4}$$

avec $c_k = \left(2^k k! \sqrt{\pi}\right)^{-1/2}$ et les polynômes d'Hermite $H_k(x)$, forment une base complète. Le choix des fonctions de bases utilisés a été guidé par la physique du problème traité.

On a les propriétés suivantes, relation d'orthogonalité des fonctions d'Hermite :

$$\int_{-\infty}^{+\infty} \varphi_m(x)\varphi_n(x)dx = \delta_{mn} \ , \tag{5}$$

et les relations de récurrence des polynômes d'Hermite :

$$H_{k+1}(x) = 2xH_k(x) - 2kH_{k-1}(x) \ , \tag{6}$$

$$H'_k(x) = 2kH_{k-1}(x) \ , \tag{7}$$

pour $k \geq 1$ avec $H_0(x) = 1$ et $H_1(x) = 2x$.

2.1.2 Résolution

On applique le bra $\langle\varphi_n(x)|$ à l'équation (1). On a :

$$\langle\varphi_n(x)|\left[\frac{1}{2}p^2+V(x)\right]|\phi(x)\rangle = \langle\varphi_n(x)|\,E\,|\phi(x)\rangle \quad (8)$$

$$= E\sum_{k=0}^{N} a_k \int_{-\infty}^{+\infty} \varphi_n(x)\varphi_k(x)dx \quad (9)$$

$$= E\sum_{k=0}^{N} a_k \delta_{nk} \quad (10)$$

$$= Ea_n \quad (11)$$

On traite ensuite le membre de gauche en deux parties.

Terme énergie cinétique

$$\langle\varphi_n(x)|\,\frac{1}{2}p^2\,|\phi(x)\rangle = -\frac{1}{2}\sum_{k=0}^{N} a_k \int_{-\infty}^{+\infty} \varphi_n(x)\frac{\partial^2}{\partial x^2}\varphi_k(x)dx\,. \quad (12)$$

avec

$$\frac{\partial^2}{\partial x^2}\varphi_k(x) = \frac{1}{2}\sqrt{k(k-1)}\varphi_{k-2}(x) - \left(k+\frac{1}{2}\right)\varphi_k(x) + \frac{1}{2}\sqrt{(k+1)(k+2)}\varphi_{k+2}(x) \quad (13)$$

pour $k \geq 2$. On a obtenu cette forme à l'aide des relations de récurrence (6) et (7). En tenant compte de la relation d'orthogonalité (5), (12) devient :

$$-\frac{1}{2}\sum_{k=0}^{N} a_k \left(\frac{1}{2}\sqrt{k(k-1)}\delta_{n,k-2} - \left(k+\frac{1}{2}\right)\delta_{nk} + \frac{1}{2}\sqrt{(k+1)(k+2)}\delta_{n,k+2}\right) \quad (14)$$

$$= -\frac{1}{2}\left(\frac{1}{2}\sqrt{(n+1)(n+2)}a_{n+2} - \left(n+\frac{1}{2}\right)a_n + \frac{1}{2}\sqrt{n(n-1)}a_{n-2}\right)\,. \quad (15)$$

pour $n \geq 2$. Pour $n = 0, 1$, on omet le dernier terme de cette expression.

Terme énergie potentiel

$$\langle\varphi_n(x)|\,V(x)\,|\phi(x)\rangle = \sum_{k=0}^{N} a_k \int_{-\infty}^{+\infty} \varphi_n(x)V(x)\varphi_k(x)dx\,. \quad (16)$$

Il n'est pas possible de trouver une solution analytique à cette intégrale. Nous allons donc utiliser la quadrature de Gauss-Hermite comme approximation.

Quadrature de Gauss-Hermite. La quadrature de Gauss-Hermite permet d'approximer l'intégrale de la forme suivante :

$$\int_{-\infty}^{+\infty} f(x)e^{-x^2}dx \ . \tag{17}$$

Dans ce cas :

$$\begin{aligned} \int_{-\infty}^{+\infty} f(x)e^{-x^2}dx &= \sum_{i=1}^{N} w_i f(x_i) + C_N f^{(2N)}(\xi) \ , & (18)\\ &\approx \sum_{i=1}^{N} w_i f(x_i) & (19) \end{aligned}$$

où N est le nombre de points d'échantillonnage utilisé, x_i sont les racines des polynômes d'Hermite et w_i sont les poids associés.

Expression de l'intégrale du terme énergie potentiel. On a donc :

$$\begin{aligned} &\int_{-\infty}^{+\infty} \varphi_n(x)V(x)\varphi_k(x)dx & (20)\\ &= -\int_{-\infty}^{+\infty} c_n H_n(x)e^{-\frac{x^2}{2}} V_0 e^{-\alpha x^2} c_k H_k(x)e^{-\frac{x^2}{2}}dx \ , & (21)\\ &= -\int_{-\infty}^{+\infty} V_0 c_n c_k H_n(x)H_k(x)e^{-(1+\alpha)x^2}dx \ , & (22)\\ &= \int_{-\infty}^{+\infty} \left(-\frac{V_0 c_n c_k}{\sqrt{1+\alpha}} H_n\left(\frac{u}{\sqrt{1+\alpha}}\right) H_k\left(\frac{u}{\sqrt{1+\alpha}}\right)\right) e^{-u^2}du \ . & (23) \end{aligned}$$

On a utilisé le changement de variable $x = \frac{u}{\sqrt{1+\alpha}}$ à la dernière ligne. On retrouve une expression de la forme (17). On résoud cette intégrale en utilisant la quadrature de Gauss-Hermite. Cela se fait aisément dans Matlab à l'aide de la fonction `hermquad.m`. Celle-ci donne les x_i et les w_i pour N fixé. On prendra N suffisamment élevé pour que $f^{(2N)}(\xi) = 0$. C'est à dire au moins le nombre de fonction d'Hermite $N+1$ dans (3).

2.1.3 Solution sous Maltab

Dans le potentiel, si l'on prend comme valeur $V_0 = 3$ et $\alpha = 1$, on obtient deux états liés :

$$\begin{cases} E_0 = -1.963720 \ , \\ E_1 = -0.369890 \ . \end{cases} \tag{24}$$

Avec notre implémentation sous Matlab, on obtient ces deux valeurs d'énergie à six décimales exactes en prenant $N \geq 50$. Si l'on prend un α plus petit, on peut obtenir plus d'états liés.

2.2 Equation de Schrödinger dépendant du temps

$$i\hbar\frac{\partial}{\partial t}\left|\Psi(x,t)\right\rangle = \left[\frac{1}{2}p^2 + V(x) + A(t)\cdot p\right]\left|\Psi(x,t)\right\rangle \ . \tag{25}$$

avec

$$\begin{cases} p^2 = -\frac{\partial^2}{\partial x^2} \\ V(x) = -V_0 e^{-\alpha x^2} \\ A(t) = A_0 f(t-t_0)\sin(\omega(t-t_0)) \end{cases} , \tag{26}$$

on prend $t_0 = 0$, $V_0 = 3$ et $\alpha = 1$.
L'enveloppe :

$$f(t-t_0) = \cos^2\left(\frac{\omega(t-t_0)}{2K}\right) \tag{27}$$

pour $\frac{-\pi K}{\omega} < t - t_0 < \frac{\pi K}{\omega}$.
La fonction d'onde solution de l'équation de Schrödinger dépendante du temps est :

$$\Psi(x,t) = \sum_{n=0}^{N} b_n(t)\phi_n(x)e^{-iE_n t} \ , \tag{28}$$

où

$$\phi_n(x) = \sum_{k=0}^{N} a_{k,n}\varphi_k(x) \ , \tag{29}$$

et avec $\Psi(x, t = t_0) = \Psi(x, 0) = \phi_0(x)$.
On a la relation d'orthogonalité des fonctions d'onde solutions de l'équation de Schrödinger stationnaire :

$$\int_{-\infty}^{+\infty} \phi_m(x)\phi_n(x)dx = \delta_{mn} \ . \tag{30}$$

2.2.1 Résolution

On injecte (28) dans (25), on obtient :

$$\sum_{n=0}^{N} \phi_n(x)\dot{b}_n(t)e^{-iE_n t} = -A(t)\sum_{n=0}^{N} \frac{\partial\phi_n(x)}{\partial x} b_n(t)e^{-iE_n t} \ . \tag{31}$$

On applique le bra $\langle\phi_m(x)|\, e^{iE_m t}$ à cette équation et avec la relation d'orthogonalité (30), on a :

$$\dot{b}_m(t) = -A(t) \int \sum_{n=0}^{N} \phi_m^*(x) \frac{\partial \phi_n(x)}{\partial x} b_n(t) e^{-i(E_n - E_m)t} dx \ . \tag{32}$$

On pose :

$$I_{m,n} = \int \phi_m^*(x) \frac{\partial \phi_n(x)}{\partial x} dx \ . \tag{33}$$

Pour résoudre cette intégrale, On commence par réécrire son intégrant à l'aide des fonctions d'Hermite :

$$\phi_m^*(x) \frac{\partial \phi_n(x)}{\partial x} = \sum_{k=0}^{N} a_{k,m}^* \varphi_k^* \cdot \left(\sum_{l=1}^{N} a_{l,n} \sqrt{\frac{l}{2}} \varphi_{l-1}(x) - \sum_{l=0}^{N-1} a_{l,n} \sqrt{\frac{l+1}{2}} \varphi_{l+1} \right) , \tag{34}$$

où on a pris :

$$\varphi_l'(x) = \sqrt{\frac{l}{2}} \varphi_{l-1}(x) - \sqrt{\frac{l+1}{2}} \varphi_{l+1}(x) \ . \tag{35}$$

Après intégration, avec la relation orthogonalité des fonctions d'Hermites (5) et en posant $l = k+1$ dans premier terme de la parenthèse et $l = k-1$ dans le second :

$$I_{m,n} = \sum_{k=0}^{N-1} a_{k,m}^* a_{k+1,n} \sqrt{\frac{k+1}{2}} - \sum_{k=1}^{N} a_{k,m}^* a_{k-1,n} \sqrt{\frac{k}{2}} \ . \tag{36}$$

pour $m, n = 0, 1, .., N$. Où l'on peut remarquer que la première somme termine à $N-1$ et la seconde commence à 1, car $k = 0, 1, .., N$ doit être respecté.
On a donc finalement :

$$\dot{b}_m(t) = -A(t) \sum_{n=0}^{N} I_{m,n} b_n(t) e^{-i(E_n - E_m)t} \ . \tag{37}$$

pour $m = 0, .., N$.

2.2.2 Solution sous Matlab

On implémente cette fonction dans Matlab avec Ode15s pour trouver les $b(t)$. On aura sous forme matricielle :

$$\frac{d}{dt} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{N-1} \\ b_N \end{pmatrix} = -A(t) \cdot M(t) \cdot \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{N-1} \\ b_N \end{pmatrix} , \tag{38}$$

où $M(t)$ est le produit d'Hadamard (commande Matlab : .*) des matrices $I_{m,n}$ et $e^{-i(E_n-E_m)t}$.

On prend les conditions initiales : $b_0 = 1$ et $b_n = 0$ pour $n > 0$. C'est-à-dire que l'état initial est dans le fondamental.

Calculons

$$|b_0(t)|^2 = b_0^*(t)b_0(t) \tag{39}$$

$$|b_1(t)|^2 = b_1^*(t)b_1(t) \tag{40}$$

pour $t_0 = 0$, $V_0 = 3$, $\alpha = 1$, $A_0 = 1$, $\omega = 1$ et $K = 1$.

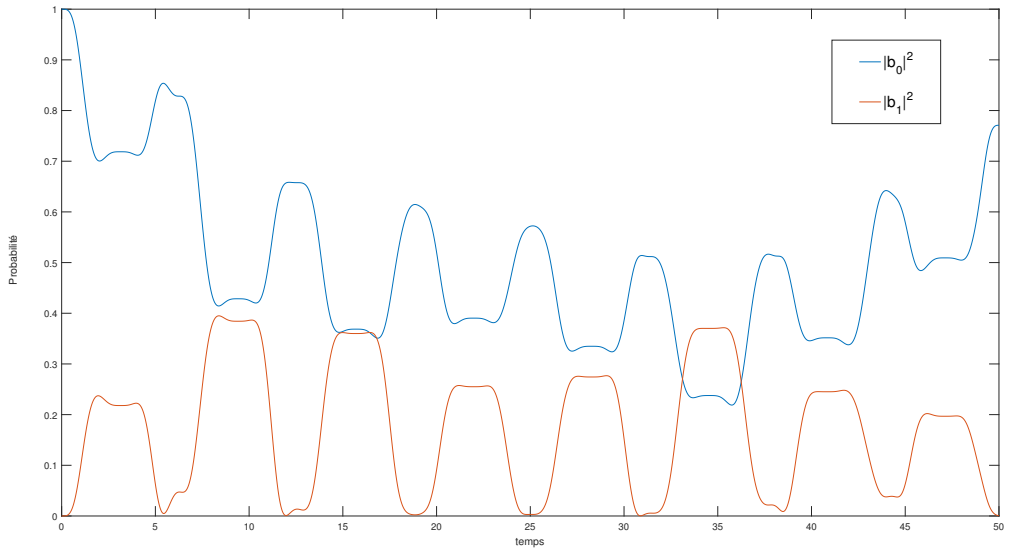

FIGURE 2 – $|b_0(t)|^2$ et $|b_1(t)|^2$

2.2.3 Fin - pas achevé

correction,notes en rouge!!

Or

$$\sum_n |b_n(t)|^2 = 1$$

donc probabilité d'ionisation :

$$1 - |b_0(t)|^2 - |b_1(t)|^2$$

graphes avec proba état ds les 2 états et

la probabilité d'ionisation en fonction de w .

Si w est egale différence d'energie , c'est à dire résonance.

regarder proba de rester ds etat fondamental en fonction du temps et d'etre ds état ionisation en fonction du temps.

Normalement les 2 courbles oscille en opposition de phase. c'est oscillations de rabi