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 $u(-1) = u(1) = 0$.

Nous souhaitons approximer la solution analytique exacte de cette équation différentielle :

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

par un développement tronqué de polynôme 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

$$\begin{aligned} T_n(x) &= \cos(n \arccos(x)) \ , && (4) \\ T_n(\pm 1) &= (\pm 1)^n \ . && (5) \end{aligned}$$

Relation d'orthogonalité :

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

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

Relations de récurrence :

$$\begin{aligned} T_{n+1}(x) &= 2xT_n(x) - T_{n-1}(x) && , n \geq 1 \ , && (7) \\ 2T_n(x) &= \frac{T'_{n+1}(x)}{n+1} - \frac{T'_{n-1}(x)}{n-1} && , n \geq 2 \ , && (8) \end{aligned}$$

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 :

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

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

$$\quad (13)$$

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 :

<u>k impair</u>

$$T_k'(x) = 2k\left(T_{k-1}(x) + T_{k-3}(x) + \cdots + T_4(x) + T_2(x) + T_0(x)/2\right) \; , \quad (16)$$

<u>k pair</u>

$$T_k'(x) = 2k\left(T_{k-1}(x) + T_{k-3}(x) + \cdots + T_5(x) + T_3(x) + T_1(x)\right) \; . \quad (17)$$

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 :

<u>k impair</u>

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

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

<u>k pair</u>
<u>$k \geq 2$</u>

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

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

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 $R(x)$:

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

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 :

$$
\begin{aligned}
v(-1) &= \sum_{k=0}^{N} a_k T_k(-1) = \sum_{k=0}^{N} a_k (-1)^k = 0\ , && (38)\\
v(1) &= \sum_{k=0}^{N} a_k T_k(1) = \sum_{k=0}^{N} a_k = 0\ . && (39)
\end{aligned}
$$

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 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

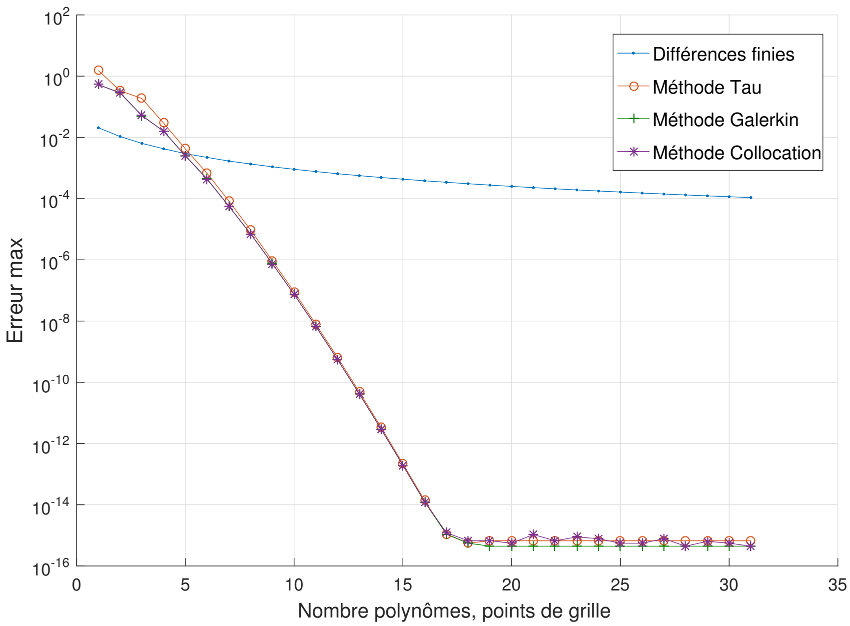

FIGURE 1 – Comparaison selon méthode de l'erreur maximale avec la solutions analytique

Deuxième partie

Exercice sur l'équation de Schrödinger

2.1 équation de Schrödinger stationnaire

2.2 équation de Schrödinger dépendant du temps

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

avec

$$A(t) = A_0 f(t-t_0)\sin(\omega(t-t_0)) \tag{65}$$

on prend $t_0 = 0$
L'enveloppe :

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

pour $\frac{-\pi k}{\omega} < t - t_0 < \frac{\pi k}{\omega}$.

$$\sum_{n=0}^{N}\phi_n(x)\dot{b}_n(t)e^{-iE_nt} = -A(t)\sum_{m=0}^{N}\frac{\partial\phi_m(x)}{\partial x}b_m(t)e^{-iE_mt} \ . \tag{66}$$

on multiplie par $\phi_n^*(x)e^{iE_nt}$
On a

$$\dot{b}_n(t) = -A(t)\int\sum_{m=0}^{N}\phi_n^*(x)\frac{\partial\phi_m(x)}{\partial x}b_m(t)e^{-i(E_m-E_n)t}dx \ . \tag{67}$$

où

$$I_{m,n} = \int\phi_m^*(x)\frac{\partial\phi_n(x)}{\partial x}dx \ . \tag{68}$$

avec

$$\phi_n(x) = \sum_{k=0}^{N}a_{k,n}\varphi_k(x) \ . \tag{69}$$

c'est parti :

$$\phi_m^*(x)\frac{\partial\phi_n(x)}{\partial x} = \sum_{k=0}^{N}a_{k,m}^*\varphi_k^*\cdot\left(\sum_{l=1}^{N}a_{l,n}\sqrt{\frac{l}{2}}\varphi_{l-1}(x) - \sum_{l=0}^{N-1}a_{l,n}\sqrt{\frac{l+1}{2}}\varphi_{l+1}\right) \ . \tag{70}$$

avec

$$\varphi_l'(x) = \sqrt{\frac{l}{2}}\varphi_{l-1}(x) - \sqrt{\frac{l+1}{2}}\varphi_{l+1}(x) \ . \tag{71}$$

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

$$I_{m,n} = \sum_{k=0}^{N-1} a^*_{k,m} a_{k+1,n} \sqrt{\frac{k+1}{2}} - \sum_{k=1}^{N} a^*_{k,m} a_{k-1,n} \sqrt{\frac{k}{2}} \; . \tag{72}$$

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 implémente cette fonction dans Matlab avec Ode15s pour trouver les $b(t)$:

$$\dot{b}_m(t) = -A(t) \sum_{n=0}^{N} I_{m,n} b_n(t) e^{-i(E_m - E_n)t} \; . \tag{73}$$

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

on aura sous forme matricielle :

$$\frac{d}{dt} \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{N-1} \\ b_N \end{pmatrix} = -A(t) \cdot M(t) \cdot \begin{pmatrix} b_0 \\ b_1 \\ \vdots \\ b_{N-1} \\ b_N \end{pmatrix} \; . \tag{74}$$

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

On prend conditions initiales : $b_0 = 1$ et $b_n = 0$ pour $n > 0$. Etat initial dans le fondamental.

2.2.1 résultats

Calculons

$$|b_0(t)|^2 = b_0^*(t) b_0(t)$$

$$|b_1(t)|^2 = b_1^*(t) b_1(t)$$

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