TP 4

Valeurs propres et vecteurs propres : Résolution de l'équation de Schrödinger par un calcul de différences finies

L'équation de Schrödinger $H\psi = E\psi$ est l'équation maîtresse de la mécanique quantique : résoudre un problème de physique quantique revient en général à résoudre cette équation dans le cas considéré. Malheureusement, même dans les cas simples, on se heurte à des problèmes techniques souvent ardus quand on cherche à le faire ; fort souvent même, il n'y a pas de solution analytique du tout ! Dans le présent exercice, on étudiera une méthode simple, peu exigeante sur le plan de la programmation (il s'agit essentiellement de faire appel à un sous-programme de bibliothèque déjà existant) pour tenter de résoudre numériquement des problèmes dont la solution est connue (puits infini, oscillateur harmonique) afin de contrôler la méthode, puis dans des cas où la solution n'est pas connue. Ce sera aussi l'occasion de se familiariser à peu de frais avec cet objet étrange qu'est la fonction d'onde ψ.

On se restreint ici à l'équation de Schrödinger stationnaire (ou non-dépendante du temps) à une particule et à une dimension. Il s'agit donc de chercher les états propres d'une particule dans un potentiel stationnaire [1].

Dans ces conditions, l'équation de Schrödinger s'écrit :

$$-\frac{\hbar^2}{2m}\frac{d^2\psi(x)}{dx^2} + V(x)\,\psi(x) = E\,\psi(x) \tag{4.1}$$

où V et ψ sont des fonctions de l'abscisse x, $V(x)$ est l'énergie potentielle de la particule au point d'abscisse x, $\psi(x)$ la fonction d'onde qui décrit l'état de la particule et E l'énergie associée à cet état ; m est la masse de la particule et $\hbar$ la constante de Plank. Résoudre l'équation (4.1) revient à trouver les fonctions d'ondes $\psi(x)$ et les énergies E pour lesquelles l'équation est vérifiée : il s'agit d'une équation différentielle du second ordre dont on connaît les solutions analytiques dans certains cas ($V = Cst$, $V = \frac{1}{2}kx^2$, ...), mais reste insoluble dans bien des cas d'où la recherche de solutions numériques.

Afin de simplifier les notations, on utilisera un système d'unités *ad hoc* dans lequel $\frac{\hbar^2}{2m} = 1$, donc l'équation (4.1) devient :

$$-\frac{d^2\psi(x)}{dx^2} + V(x)\,\psi(x) = E\,\psi(x) \tag{4.2}$$

1. voir le cours de physique quantique.

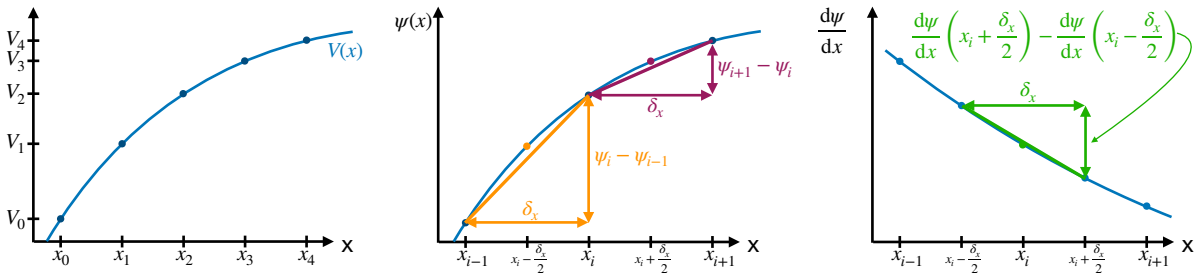

FIGURE 4.1 – Gauche : Illustration de la discrétisation de x ainsi que des conséquences pour le potentiel V et la fonction d'onde ψ. Centre : La dérivée première de ψ est approchée à $x_i \pm \delta_x/2$. Droite : La dérivée seconde de ψ est approchée à x_i.

4.1 La méthode des différences finies.

Le problème est légèrement différent de celui de la résolution d'une (ou plusieurs) équation(s) du mouvement où l'on calcule de proche en proche les valeurs que prend la ou les fonctions recherchées : ici, on veut simultanément *toutes* les valeurs de $\psi(x)$ et de surcroît E est aussi une inconnue. Les méthodes de type Runge-Kutta ne conviennent donc pas, il faut en trouver d'autres.

La méthode des différences finies en est une possible. Elle consiste en une discrétisation du problème, c'est-à-dire que l'on fait l'approximation de remplacer la variable continue x par une variable discrète x_i :

$$x_i = -L/2 + i \cdot \delta_x, \; i \in [0, n-1]$$

où i est un indice entier. La droite infinie est évidemment remplacée par un segment fini de longueur $L = (n-1)\delta_x$, centré autour de zéro, on couvre alors l'intervalle $[-L/2, L/2]$.

Cela entraîne que δ doit être "petit", c'est-à-dire sensiblement plus petit qu'une distance "typique" sur laquelle la fonction d'onde varie sensiblement. Par ailleurs, L doit être "presque" infini, soit, plus grand que le domaine dans lequel la particule peut se déplacer. Autrement dit, n doit être "grand".

Notre objectif est maintenant de discrétiser l'équation de Schrödinger, c'est-à-dire en particulier la fonction d'onde ainsi que sa dérivée seconde. On pose :

$$\begin{aligned} \psi_i &= \psi(x_i) \\ V_i &= V(x_i) \end{aligned}$$

pour la fonction d'onde et le potentiel discrétisés. Pour la dérivée première de ψ, prise un demi-intervalle au delà de x_i, on peut trouver une expression approchée :

$$\frac{d\psi}{dx}\left(x_i + \frac{\delta_x}{2}\right) \sim \frac{\psi_{i+1} - \psi_i}{\delta_x}$$

De même on trouve pour la dérivée prise un demi-intervalle en deçà de x_i :

$$\frac{d\psi}{dx}\left(x_i - \frac{\delta_x}{2}\right) \sim \frac{\psi_i - \psi_{i-1}}{\delta_x}$$

En utilisant ces expressions, nous pouvons ensuite en déduire une expression approchée pour la dérivée seconde de ψ prise en x_i :

$$\frac{d^2\psi}{dx^2}(x_i) \sim \frac{\frac{d\psi}{dx}\left(x_i + \frac{\delta_x}{2}\right) - \frac{d\psi}{dx}\left(x_i - \frac{\delta_x}{2}\right)}{\delta_x} = \frac{\psi_{i+1} - 2\psi_i + \psi_{i-1}}{\delta_x^2}$$

Muni de ces expressions approchées, nous pouvons établir une forme discrétisée de l'équation de Schrödinger :

$$-\frac{\psi_{i+1} - 2\psi_i + \psi_{i-1}}{\delta_x^2} + V_i\psi_i = E\psi_i \tag{4.3}$$

Dans la suite on va étudier trois problèmes qui correspondent à des potentiels V(x) différents :

1. le puits carré infini : Il suffit de fixer la valeur du potentiel à zéro partout sur l'intervalle, $V(x) = 0$.
2. le potentiel harmonique : $V(x) = \frac{m}{2}\omega^2 x^2$. Ici on prendra $\omega = \sqrt{\frac{1}{m}}$ (voir ci-dessous).
3. un double puits, paramétré par $V(x) = a(x - r_1)(x - r_2)(x - r_3)(x - r_4)$, soit un polynôme de degré 4 dont les racines sont r_1, r_2, r_3, r_4.

1. Écrire trois fonctions (une par problème) qui rendent la valeur $V(x)$. Lorsqu'on voudra ensuite changer la forme du potentiel il suffira de modifier l'appel de la fonction, à l'exclusion du reste du programme. Ces fonctions sont alors de cette forme :

```
1 double potentiel_puits(double x){
2     return ...
3 }
```

⇒ Rendre un seul script python qui contient les résultats des éxercices 1 et 2.

2. Définir un intervalle $[-L/2, L/2]$, le diviser en n segments de longueur δ_x et remplir les tableaux x (contenant les x_i) et V (contenant les valeurs de potentiel V_i). On peut utiliser des tableaux numpy classiques (structure de données `ndarray`). Tracer ensuite les trois potentiels définis dans l'exercise précédent dans une seule figure. Choisir $L = 5, n = 100$ et $a = 1, r_1 = -r_4 = 2$, $r_2 = -r_3 = 0.5$.
⇒ Rendre une seule figure avec l'inscription des axes et la légende.

4.2 Valeurs propres-vecteurs propres.

On considère maintenant que la fonction d'onde discrétisée est un vecteur :

$$\psi = \begin{pmatrix} \psi_0 \\ \psi_1 \\ \vdots \\ \psi_i \\ \vdots \\ \psi_{n-2} \\ \psi_{n-1} \end{pmatrix}$$

On peut écrire l'expression de la matrice $n \times n$ notée $\mathbf{H}$ telle que l'équation de Schrödinger discrétisée s'écrive maintenant sous forme matricielle :

$$\mathbf{H}\psi = E\psi$$

On appellera **H** l'Hamiltonien discrétisé : ce n'est plus un opérateur comme dans l'équation (4.1), mais une matrice [2].

Autrement dit, l'équation de Schrödinger discrétisée s'écrit pour tous les ψ_i :

$$\sum_{j=0}^{n-1} H_{ij} \cdot \psi_j = E \cdot \psi_i \tag{4.4}$$

Si on regroupe l'équation 4.3 par les différents ψ_i, on obtient :

$$\begin{aligned} -\frac{\psi_{i+1} - 2\psi_i + \psi_{i-1}}{\delta_x^2} + V_i\psi_i &= E\psi_i \\ -\frac{1}{\delta_x^2}\psi_{i-1} + \left(\frac{2}{\delta_x^2} + V_i\right)\psi_i - \frac{1}{\delta_x^2}\psi_{i+1} &= E\psi_i \end{aligned} \tag{4.5}$$

En comparant l'équation 4.4 avec l'équation 4.5 on obtient :

$$\begin{aligned} H_{ii} &= \frac{2}{\delta_x^2} + V_i \\ H_{i(i-1)} &= -\frac{1}{\delta_x^2} \\ H_{i(i+1)} &= -\frac{1}{\delta_x^2} \end{aligned}$$

Donc concrètement, la matrice **H** a cette forme :

$$\mathbf{H} = \begin{pmatrix} \frac{2}{\delta_x^2} + V_0 & -\frac{1}{\delta_x^2} & 0 & 0 & 0 & \dots & 0 \\ -\frac{1}{\delta_x^2} & \frac{2}{\delta_x^2} + V_1 & -\frac{1}{\delta_x^2} & 0 & 0 & \dots & 0 \\ 0 & -\frac{1}{\delta_x^2} & \frac{2}{\delta_x^2} + V_2 & -\frac{1}{\delta_x^2} & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & 0 & -\frac{1}{\delta_x^2} & \frac{2}{\delta_x^2} + V_i & -\frac{1}{\delta_x^2} & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & 0 & -\frac{1}{\delta_x^2} & \frac{2}{\delta_x^2} + V_{n-2} & -\frac{1}{\delta_x^2} \\ 0 & \dots & 0 & 0 & 0 & -\frac{1}{\delta_x^2} & \frac{2}{\delta_x^2} + V_{n-1} \end{pmatrix}$$

C'est une matrice tridiagonale symétrique.

Les extrémités de l'intervalle, x_0 et x_{n-1}, doivent faire l'objet d'une attention particulière. On fera l'approximation supplémentaire que $\psi_{-1} = \psi_n = 0$, c'est-à-dire que la fonction d'onde « à l'infini » est nulle : En $i = 0$ on n'a pas le terme en ψ_{i-1} et en $i = n-1$, le terme ψ_{i+1} est absent. Cela revient à postuler que la fonction d'onde est nulle aux deux extrémités de l'intervalle fini considéré. La conséquence en est que la particule ne doit pas s'aventurer vers les extrémités de l'intervalle et que donc l'extension de sa fonction d'onde doit rester à peu près centrée et petite devant la taille de l'intervalle.

2. Pour mieux comprendre le lien entre la matrice **H** et l'équation (4.1), il est utile d'écrire l'opérateur hamiltonien en représentation de position : L'opérateur hamiltonien est défini comme $\hat{H} = \frac{\hat{P}^2}{2m} + V(\hat{X})$ avec l'opérateur impulsion $\hat{P}$ et l'opérateur de position $\hat{X}$. On fait alors le choix de la "représentation de position $|x\rangle$" définie par $\hat{X}|x\rangle = x|x\rangle$ où x est maintenant une variable. Dans cette représentation on obtient $\langle x|P|\psi\rangle = -i\hbar\frac{d\psi(x)}{dx}$ et $\langle x|V(X)|\psi\rangle = V(x)\psi(x)$ où $\langle x|\psi\rangle = \psi(x)$ est la fonction d'onde.

3. Quel sens physique cela peut-il avoir ? Quelle est la conséquence pour le potentiel effectif qu'on simule avec cette approche ?

Le problème mathématique à résoudre est maintenant devenu une équation aux valeurs propres : les valeurs que peut prendre E sont les valeurs propres de la matrice $\mathbf{H}$ et le vecteur ψ est le vecteur propre asssocié à chaque valeur propre. Comme la matrice est $n \times n$, il doit y avoir n valeurs propres et n vecteurs propres solution de l'équation de Schrödinger discrétisée.

4.3 Mise en œuvre numérique.

Il n'est pas question d'écrire une fonction capable de résoudre un problème aux valeurs propres ! Il existe de nombreuses bibliothèques de programmes déjà écrits et testés : ils sont fiables et efficaces, il est inutile de réinventer la roue ! On utilisera ici la bibliothèque EIGEN, qui est spécialement écrite pour le C++ et décrite dans le poly de cours C++ de 3P002. EIGEN permet non seulement de travailler avec des vecteurs et matrices (= tableaux 2D), mais aussi de diagonaliser des matrices pour résoudre un problème aux valeurs et vecteurs propres. Pour faciliter la tâche, voici une fonction qui utilise EIGEN pour diagonaliser une matrice carrée symétrique `z` (téléchargeable sur moodle) :

```
1 void solve(MatrixXd &z, VectorXd &d, MatrixXd &v){
2    SelfAdjointEigenSolver<MatrixXd> eigensolver(z);
3    if (eigensolver.info() != Success){
4         cout << "ERROR in SelfAdjointEigenSolver" << endl;
5         abort();
6    }
7    d = eigensolver.eigenvalues();
8    v = eigensolver.eigenvectors();
9    cout << "La diagonalisation s'est bien passée" << endl;
10 }
```

Cette fonction `solve()` fournit les valeurs propres dans le vecteur (= tableau 1D) `d` et les vecteurs propres dans la matrice (= tableau 2D) `v`. On l'appelle comme ceci :

```
1 int n = 100;
2 MatrixXd z(n,n);
3 ... // remplissage de z
4 VectorXd eigenvalues;
5 MatrixXd eigenvectors;
6 solve(z, eigenvalues, eigenvectors);
```

Ici les objets `z,␣eigenvalues` et `eigenvectors` sont passés à la fonction `solve()` via des *références* comme indiqué par les & dans l'entête de la fonction `solve()`. Ceci permet à `solve()` de lire et écrire sur ces objets. Voir le chapitre sur EIGEN dans le poly de cours C++ de 3P002 pour en savoir plus.

Pour pouvoir utiliser EIGEN, il faut ajouter aussi ceci au début du programme :

```
#include "Eigen/Dense"
using namespace std;
using namespace Eigen;
```

Puis on doit encore copier ou décompresser le dossier “Eigen” dans le dossier de votre programme, voir le poly de cours C++ de 3P002.

Enfin, pour la compilation il est très utile ici d’ajouter l’option “-O2” (grand O pas zéro !) à la ligne de commande du compilateur, pour activer les optimisations automatiques du code du 2ème niveau. Ceci peut accélérer la diagonalisation de matrice d’un facteur 10 à 20 pour atteindre des performances similaires à d’autres bibliothèques très utilisées dans le domaine comme par exemple LAPACK qui est écrite pour Fortran.

Il ne reste plus qu’à remplir la matrice `z` qui représente $\mathbf{H}$ en forme discrète. Sa taille `n*n` dépend du nombre d’éléments du vecteur ψ. Les vecteurs propres sont enregistrés colonne par colonne dans la matrice `eigenvectors` avec la fonction `solve()`.

Pour la sortie des vecteurs propres dans un fichier, on peut profiter des fonctionnalités d’EIGEN :

```
ofstream fichier("TP6_q3.res");
MatrixXd results(n,n+1);
results << x, eigenvectors;
fichier << results;
fichier.close();
```

Comme on doit ici écrire les positions `x[i]` en première colonne, on crée d’abord une matrice “results” avec n+1 colonnes, puis on la remplit avec le vecteur de n éléments `x` et la matrice de taille n*n `eigenvectors` d’en haut. Ensuite l’écriture de “results” sur le fichier se fait très facilement, pas besoin d’une seule boucle ! Il faut juste faire attention que `x` n’est pas un tableau C classique, mais bien un objet du type `VectorXd`.

On remarquera que les fonctions d’ondes calculées ne sont pas normalisées comme les fonctions d’ondes théoriques avec

$$\int \psi^2 dx = 1$$

On peut le faire, si souhaité, avec cette fonction par exemple sur la matrice `eigenvectors` :

```
// Using the normalize() function of Eigen and vector-operations
  ↪  on each column using the colwise() function.
// The multiplication with the scalar 1/sqrt(dx) can be done for
  ↪  all elements of the matrix with a single command:
void normalize(MatrixXd &m, double dx)
{
    m.colwise().normalize(); // columns of m are already
  ↪  normalized here, but to be shure we do it again here.
    m = m / sqrt(dx);
}
```

Lors de ce TP seront générés beaucoup de fichiers *.res différents en fonction du potentiel et des paramètres `L` et `n`. Pour garder une trace de quel fichier correspond à quoi, il est utile de directement inclure les valeurs des paramètres dans le nom de fichier. Pour faire cela d'une manière automatique, on peut utiliser les "string-stream" pour définir le nom de fichier, c'est-à-dire les *flux* qui permettent d'assembler une chaîne de caractères. Voir cet exemple :

```
#include <fstream> //file-stream
#include <sstream> //string-stream

// dans main():
ostringstream filename;
filename << "q4_harm_valeurs_n" << n << "_L" << (int)L << ".res";
ofstream fichier(filename.str().c_str());
```

Ici le nom de fichier (`filename`) indique qu'il s'agit de valeurs propres de la question 4 pour un potentiel harmonique avec les paramètres `n` et `L`. Ce qui donne par exemple : `q4_harm_valeurs_n100_L20.res`. On utilise ici la même syntaxe que pour les `cout`, on a seulement remplacé l'objet `cout` par l'objet `filename` qui est du type `ostringstream` ("output string stream" => pour créer une chaîne de caractères). Ensuite pour récupérer de l'objet `filename` une chaîne de caractères classique du langage C, il faut d'abord extraire un objet du type `string` avec `filename.str()`, puis de celui-ci la chaîne de caractères classiques avec `c_str()`, ce qui donne en une seule ligne : `filename.str().c_str()`.

4. Écrire une fonction qui rend la valeur $V(x)$. Pour le puits infini, il suffit de mettre 0, pour le potentiel harmonique, x^2. Lorsqu'on voudra ensuite changer la forme du potentiel, il suffira de modifier cette fonction, à l'exclusion du reste du programme. Cette fonction est alors de cette forme :

```
double potentiel(double x){
    return ... ;
}
```

5. Écrire un programme principal qui :

1. définisse un intervalle $[-L/2, L/2]$, le divise en n segments de longueur δx et remplisse les tableaux `x[i]` (contenant les x_i) et `V[i]` (contenant les valeurs de potentiel V_i). On peut soit utiliser des tableaux C classiques, soit des objets du type `VectorXd` d'EIGEN. Pour faciliter l'écriture sur un fichier, il est conseillé d'utiliser la 2ème variante : `VectorXd␣x(n),␣V(n)`. L'accès aux tableaux C et les vecteurs d'EIGEN se fait de la même façon avec `x[i]` et `V[i]`. Prenez ces valeurs pour commencer : $L = 5$ et $n = 100$.
2. crée le tableau 2D `MatrixXd␣z(n,n)`, l'initialise à zéro, puis le remplisse avec les éléments diagonaux et sous-diagonaux de **H**.
3. en recherche les valeurs propres et les vecteurs propres et écrive les valeurs propres dans un fichier et quelques vecteurs propres dans un autre.

Rappel : La compilation de ce programme se fait avec l'option `-O2` pour optimiser le code :
`c++ -Wall -Wextra -O2 mon_prog.cpp -o mon_prog`
où `mon_prog.cpp` est évidemment le nom de *votre* programme, que vous pouvez choisir librement...

4.4 Étude numérique.

4.4.1 Instructions pour gnuplot

- Utiliser uniquement des scripts gnuplot dans ce TP, voir poly de cours C++ de 3P002. Il y aura un très grand nombre de graphes à créer et des scripts gnuplot permettent de gagner du temps et de garder une trace sur comment vous avez généré vos graphes. **Vos scripts gnuplot doivent être imprimés et joints à votre compte rendu.**
- Faites une sortie sur fichier *.ps, puis imprimer ces fichiers seulement quand vous êtes satisfait du résultat.
- Nota bene : Lors de ce TP seront générés beaucoup de fichiers *.pdf différents en fonction du potentiel. Pour garder une trace de quel fichier correspond à quoi, il est utile de directement inclure les informations saliants dans le nom de fichier, par exemple `q6_puits_valeurs_n100_L5.pdf` correspondrait aux valeurs propres de la question 6 pour un puits de potentiel avec $n = 100,\ L = 5$.
- Pour ouvrir plusieurs fenêtres en même temps avec gnuplot sous Linux, il y a la commande `set term X11 0` pour la première fenêtre, puis `set term X11 1` pour la 2ème etc. Ces commandes sont à mettre à l'endroit de votre scripts où vous voulez ouvrir une nouvelle fenêtre sans oublier d'incrémenter l'identifiant de la fenêtre.
- Pour tracer la courbe théorique de E_p ou les fonctions d'ondes théoriques gnuplot permet d'entrer directement les formules mathématique. Par exemple pour E_p du puits carré infini : `plot pi*pi*x*x/(L*L)` si on définit la variable `L` plus haut dans votre script gnuplot : `L=5`
- Comme pour le langage C, il faut faire attention avec gnuplot aux divisions entre valeurs entières. Par exemple pour tracer une fonction d'onde théorique, il faudra plutôt écrire `sqrt(2.0/L)` que `sqrt(2/L)` si `L` est définit en étant une variable entière avec l'affectation `L=5`. En gnuplot on n'écrit pas explicitement le type d'une variable, il dépend de la valeur affecté.

4.4.2 Puits carré infini.

Rappels théoriques[3] *:*

Si on considère une particule dans un puits carré infini de largeur L, les fonctions d'onde solutions de l'équation de Schrödinger indépendante du temps sont nulles en dehors du puits et ont la forme suivante dans le puits

$$\psi_p^{\text{theo}}(x) \quad = \quad \sqrt{\frac{2}{L}} \sin \frac{(p+1)\pi(x+\frac{L}{2})}{L}$$

3. Reportez-vous au cours de physique quantique.

Les énergies de ces états s'écrivent :

$$E_p^{\text{theo}} = \frac{\hbar^2}{2m}\left(\frac{\pi(p+1)}{L}\right)^2$$

Rappel : on choisit ici $\frac{\hbar^2}{2m} = 1$.

6. Essayer votre programme avec $n = 100$ et $L = 5$, afin de comparer le résultat obtenu avec le résultat théorique : Tracer les énergies propres E_p en fonction de p avec, sur le même graphe, la courbe théorique E_p^{theo}. Vérifier l'accord avec la courbe théorique et discuter le résultat.
⇒ Rendre une seule figure avec les deux courbes.

Ensuite, on souhaite comparer également les fonctions d'onde. On remarquera que les fonctions d'ondes calculées ne sont pas normalisées comme les fonctions d'ondes théoriques avec

$$\int \psi^2 dx = 1$$

7. Tracer les premières fonctions d'onde ($p \in [0,2]$) obtenues par votre programme : Pour cela tracer sur le même graphe les premières ($p \in [0,2]$) fonctions d'onde *au carré*, décalées verticalement d'une valeur proportionnelle à leur énergie E_p. Attention à ne pas confondre les lignes et les colonnes - les vecteurs propres se trouvent dans les **colonnes** de la matrice v. Discuter le résulat qualitativement eu égard des résultats attendus pour un puits carré infini.
⇒ Rendre une seule figure avec les trois courbes décalées ainsi qu'un code C++ qui contient votre programme complet.

Astuces gnuplot :

- Pour faire ce graphe complexe, on peut s'aider en utilisant la modification des données à la volée de gnuplot (voir aussi le poly de cours C++ de 3P002). Voici un exemple où il faudra encore insérer vos valeurs de E_p et ajuster un facteur d'échelle. On y voit que pour faire le carré de ψ_p on utilise `($2)**2` (pas possible sous C++ !). Le mieux est de mettre ces instructions gnuplot dans un script gnuplot :

```
E0 = ...  # la valeur de l'energie pour p=0
E1 = ...  # la valeur de l'energie pour p=1
E2 = ...  # la valeur de l'energie pour p=2
scale = ... # un facteur d'echelle entre l'amplitude des
            # fonctions d'ondes et l'energie
plot 'vecteurs.res' u 1:(scale*($2)**2 + E0) title 'Fondamental' w l
replot 'vecteurs.res' u 1:(scale*($3)**2 + E1) title '1er excite' w l
replot 'vecteurs.res' u 1:(scale*($4)**2 + E2) title '2nd excite' w l
```

8. Comparer l'accord entre les fonctions d'onde numériques et théoriques pour $p = 1$ et un ordre plus élevé dans le domaine où l'accord pour E_p n'est pas bon, par exemple $p = 55$. Attention à ne pas mélanger les fonctions d'onde avec leurs valeurs au carré. Faites votre diagnostic.
⇒ Rendre deux figures avec l'inscription des axes et la légende - un graphe par valeur de p. Faites attention à rendre des figures avec des courbes lisibles. Si nécessaire, ajuster

l'ordonnée ou modifier l'échelle de la courbe.

Sur deux autres graphes comparer l'accord entre les fonctions d'onde numériques et théoriques pour $p = 1$ et un ordre plus élevé dans le domaine où l'accord pour E_p n'est pas bon, par exemple $p = 55$ (un graphe par valeur de p). Faites votre diagnostic. L'amplitude et le signe de la fonction d'onde numérique peuvent être différents comparés à la fonction d'onde théorique, si on ne normalise pas la fonction d'onde numérique (voir plus haut).

Refaire cette dernière analyse de comparaison des résultats numériques et théoriques (pour E_p et ψ_p) avec différentes valeurs de n et de L, par exemple : $n = 200$ & $L = 5$, $n = 100$ & $L = 10$ et $n = 200$ & $L = 10$. Comparer les résultats.

4.4.3 Potentiel harmonique.

Rappels théoriques :
Le deuxième cas dont la solution est connue est le potentiel harmonique [4] :

$$V(x) = \frac{1}{2}kx^2 = \frac{m}{2}\omega^2 x^2, \quad \omega = \sqrt{\frac{k}{m}}$$

Les énergies propres s'écrivent maintenant :

$$E_p^{\text{theo}} = \left(p + \frac{1}{2}\right)\hbar\omega, \quad p = 0, 1, 2, 3, \ldots$$

L'état fondamental, pour $p = 0$ a donc une énergie $E_0 = \hbar\omega/2$.

9. Modifier votre programme pour étudier le potentiel harmonique : on prendra $k = 1$, soit $\hbar\omega = \sqrt{2m}\sqrt{k/m} = \sqrt{2}$. Il s'agit de la forme de potentiel que vous avez déjà mise en œuvre dans l'exercice 1. Pour rappel : Il suffit de changer l'appel de la fonction du potentiel.
Répéter les opérations de l'exercice 6 avec $n = 100$ et $L = 5$: tracer les énergies propres numériques E_p avec la courbe théorique E_p^{theo}. Commenter le résultat.
⇒ Rendre une seule figure avec deux courbes.

10. Refaire le même calcul, toujours avec $n = 100$ mais cette fois-ci avec $L = 20$. Commenter sur l'impact de δ_x et n par rapport aux résultats obtenus - diagnostic.
⇒ Rendre une seule figure avec deux courbes.

11. Tracer sur le même graphe le potentiel $V(x)$ ainsi que les premières ($p \in [0, 2]$) fonctions d'onde au carré (numériques), décalées verticalement d'une valeur égale à leur énergie E_p. (On pourra multiplier l'amplitude des fonctions d'onde, avant décalage de E_p, par un facteur pour éviter leur chevauchement et ainsi rendre le graphe bien lisible). Qu'observe-t-on quand à la "largeur" de la fonction d'onde ? Commenter.
⇒ Rendre une seule figure avec quatre courbes - le potentiel ainsi que les trois fonctions d'onde au carré.

4.4.4 Double puits.

L'intérêt de cet exercice n'est bien sûr pas de chercher à résoudre un problème dont la solution analytique est connue (comme le puits infini ou l'oscillateur harmonique).

4. Se reporter de nouveau au cours de physique quantique.

Cela permet juste de tester le programme que l'on a écrit et d'illustrer le cours de physique quantique.

Toutefois, le même programme, à très peu de choses près, permet de résoudre des problèmes sensiblement plus compliqués, pour lesquels les solutions analytiques sont pour l'essentiel inaccessibles. On remplace le potentiel harmonique maintenant par :

$$V(x) = a(x - r_1)(x - r_2)(x - r_3)(x - r_4)$$

soit un polynôme de degré 4, dont les racines sont r_1, r_2, r_3, r_4.
Selon le paramètrage du potentiel, il permet de simuler un double puits symétrique ou dissymétrique. Pour la suite, prenez $n = 1000$ et $L = 20$.

12. Comme précédemment, tracer sur un seul graphe les solutions obtenues des premiers états pour $a = 1$ et $r_1 = -2$, $r_2 = -0.5$, $r_3 = 0.5$, $r_4 = 2$, ainsi que le potentiel $V(x)$. C'est le paramètrage de votre troisième fonction de l'éxercice 1 et correspond à un double puits symétrique. Quel est le sens physique d'un tel potentiel, quelles appliquations pourrait-il avoir ? Commenter ensuite sur la forme des premiers états propres du système. Retrouvez-vous dans vos résultats des effets que vous connaissez déjà, par exemple de la physique moléculaire ?
⇒ Rendre une seule figure avec plusieurs courbes. Vérifier que toutes les courbes sont bien lisibles, qu'elles sont déplacées proportionellement à leurs énergies et qu'elles sont étiquetées dans la légende.

4.4.5 *Pour aller plus loin :* Double puits particuliers

13. Tracer les solutions obtenues pour $a = 400$ et $r_1 = -2$, $r_2 = -0.5$, $r_3 = 0.5$, $r_4 = 2$ (double puits symétrique plus profond). Commenter et, bien sûr, comparer avec le cas précédent.

14. Tracer maintenant les solutions obtenues pour $a = 1$ et $r_1 = -2$, $r_2 = -0.5$, $r_3 = 0$, $r_4 = 2$, ainsi que le potentiel $V(x)$. C'est un potentiel correspondant à un double puits dissymétrique. Commenter et, bien sûr, comparer avec le double puits symétrique avec $a = 1$.