États quantiques

Équation de Schrödinger, états discrets et leurs propriétés.

Nous allons chercher les états discrets pour différents potentiels $U(x)$ en résolvant l'équation unidimensionnelle de Schrödinger :

$$\frac{\mathrm{d}^2 p(x)}{\mathrm{d}x^2} = \frac{2m}{\hbar^2}\,[U(x) - E]\;p(x)$$

($p(x)$ est l'amplitude de probabilité) comme cela fut introduit dans le cours de la physique de Feynman (§16-5 et §16-6). Pour cela, il faut tester plusieurs valeurs de l'énergie E afin d'obtenir les solutions $E = E_n$ sans divergence de $p(x)$ à l'infini. Évidemment, on peut utiliser des codes de module `scipy.optimize` pour trouver les solutions automatiquement, mais au début il est plus simple et plus clair de chercher les solutions manuellement. Notre code sera composé de trois fichiers (vous pouvez, évidemment, changer les noms des fichiers sans oublier de faire les changements correspondants dans le code) :

1. Le fichier *schroedinger_states.py* contient le code du menu `tkinter` qui contrôle le fonctionnement du programme et le code d'accumulation et de présentation des états quantiques obtenus. Vous pouvez télécharger ce fichier à partir de Moodle de notre cours. Le fonctionnement du menu est expliqué dans la présentation introductive et dans le 'Help'.

2. Le fichier *schroedinger_solve.py* contient la fonction *schroedinger_solve(...)* de solution de l'équation de Schrödinger en format réduit (x, U et E renormalisés) :

   $$\frac{\mathrm{d}^2 p}{\mathrm{d}x^2} = 2\;[U(x) - E]\;p(x)\;.$$

   La fonction *schroedinger_solve(...)* est appelée par la bouton 'Solve' du menu `tkinter`. C'est à vous de développer le code basé sur la fonction `odeint()` sans oublier le doc_string et les commentaires de son fonctionnement. Éventuellement vous pouvez ajouter ici le code de modification automatique de la valeur E, pour trouver une des valeurs E_n qui donne la solution recherchée (sans divergence).

3. Le fichier *schroedinger_matrices.py* (le code aussi à développer) contient deux fonctions :

   Une fonction `matrices_E_nm_nxm(...)` construit et imprime (dans la console et dans le fichier *schroedinger_matrices.txt*) le vecteur de valeurs E_n et les matrices $\langle n\,|\,m\rangle$ et $\langle n\,|\,x\,|\,m\rangle$;

   L'autre fonction `Aplus_Aminus(...)`, pour les puits paraboliques exclusivement, vérifie les propriétés des opérateurs d'annihilation et de création :

   $$\hat{a} = \frac{1}{\sqrt{2}}\left(x + \frac{\mathrm{d}}{\mathrm{d}x}\right)\;,\;\hat{a}^{+} = \frac{1}{\sqrt{2}}\left(x - \frac{\mathrm{d}}{\mathrm{d}x}\right)\;.$$

On commence le développement du code par la solution de l'équation différentielle de Schrödinger appelée par le bouton 'Solve'. Au début il faut rester avec le potentiel par défaut $U(x) =$ `x**2/2` donné par le menu `tkinter`, puisque pour ce potentiel on connaît les énergies des états stationnaires confinés $E_n = 1/2,\ 3/2,\ 5/2,...$ Rappelez-vous que, pour utiliser le solveur `odeint(fct, p0, x)` du module `scipy.integrate`, il faut transformer l'équation différentielle d'ordre 2 dans le système des équations d'ordre 1, donné par la fonction `fct(p,x)` (vous pouvez l'appeler 'schroedinger'). Prenez les 'conditions initiales' `p0` à $x = -x_{max}$ suivants : $p(-x_{max}) = 0$ et $\mathrm{d}p/\mathrm{d}x = 0.01$ (`p0 = [0., 0.01]`).

Les résultats (pas seulement p, mais aussi $\mathrm{d}p/\mathrm{d}x$) doivent être normalisés pour avoir

$$\int_{\infty}^{\infty} p_n(x)p_n(x)\mathrm{d}x = 1 \ .$$

Nous avons besoin d'avoir $\mathrm{d}p/\mathrm{d}x$ normalisé pour les opérateurs d'annihilation et de création.

Ayant obtenu les bonnes solutions, vous pouvez les accumuler et observer la figure intégrale en appuyant sur le bouton 'Add'. Vous pouvez recommencer l'accumulation après activation du bouton 'Clear'.

Il vous reste seulement de développer le code qui calcule les matrices

$$\langle n \,|\, m \rangle = \int_{\infty}^{\infty} p_n(x)p_m(x)\mathrm{d}x \ , \ \langle n \,|\, x \,|\, m \rangle = \int_{\infty}^{\infty} p_n(x)\, x\, p_m(x)\mathrm{d}x$$

et, pour le potentiel parabolique, les matrices correspondant aux opérateurs d'annihilation et de création. Imprimez les matrices dans la Console et dans un fichier ASCII. Visualisez l'effet de l'opérateur d'annihilation sur les niveaux de base. Présentez les résultats dans votre rapport et interprétez la signification de ces matrices.

Vous pouvez maintenant passer aux études des états quantiques dans les autres potentiels.

- Trouvez tous les états **confinés** pour les potentiels suivants :

  `-2*np.exp(-x**2)` et `-1*np.exp(-x**2)`.

- Étudiez l'effet tunnel entre deux puits de potentiels en fonction de l'épaisseur de la barrière qui les sépare. Il est suffisant de ne considérer que les états de base ainsi que deux épaisseurs différentes de la barrière. Ici une des difficultés de la programmation est de trouver l'expression pour le double puits.

- Afin de faire le dernier exercice au plus vite, il est conseillé de rechercher la valeur E automatiquement. Pour cela, il faut utiliser la fonction `odeint()` avec les arguments supplémentaires (à voir help) et l'inclure dans la fonction `p_end(E)` demandée par `fsolve(p_end, Einit)` de module `scipy.optimize`.

- *Pour finir animez la dynamique de la distribution de la probabilité des états mixtes

  $$P_{nm}(x,t) = \frac{1}{2}\left[p_n(x)e^{+iE_nt} + p_m(x)e^{+iE_mt}\right] \cdot \left[p_n(x)e^{-iE_nt} + p_m(x)e^{-iE_mt}\right] \ .$$

  Cet animation aidera vous à comprendre mieux la signification de la matrice $\langle n \,|\, x \,|\, m \rangle$. Un exemple pour $n = 4$ et $m = 5$ est présenté sur Moodle.