Projet de Physique Numérique

Dynamique Quantique

Travail présenté dans le cadre du cours LU3PY124 : Physique expérimentale et numérique

Travail réalisé par Arthur ANTOINE, Antonin CARBIENER et Lucas BOISTAY

Mars 2023

Table des matières

Première partie

Introduction au problème

L'équation de Schrödinger, conçue par le physicien autrichien Erwin Schrödinger en 1925, est une équation fondamentale en mécanique quantique. Elle permet en effet de décrire l'évolution temporelle d'une particule quantique (non-relativiste [1]) de la même manière que la seconde loi de Newton pour une particule classique.

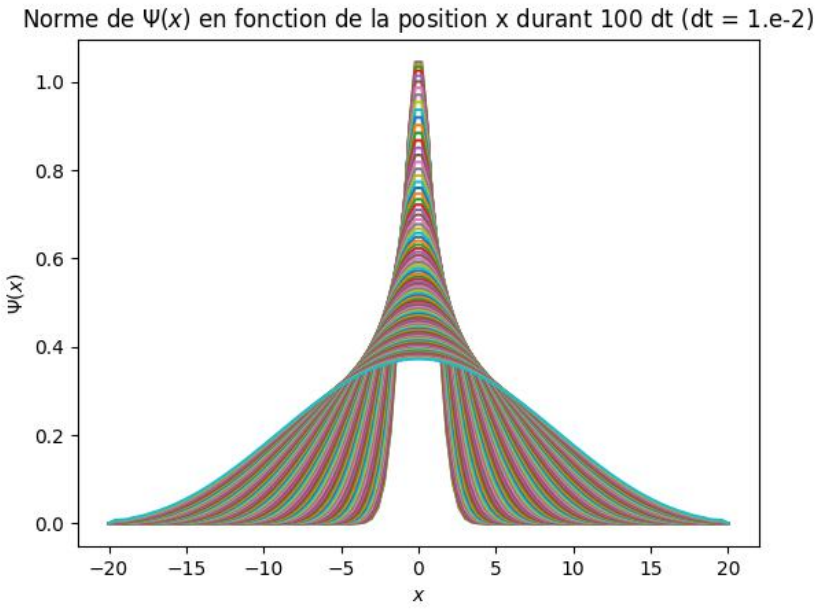


FIGURE 1 – Norme de la fonction d'onde d'une particule (représentée ici par un paquet d'onde gaussien) en fonction de la position qui évolue avec le temps. Méthode : RK4

Si nous connaissons l'état initial d'une particule représenté par un vecteur Ψ, nous pouvons alors déterminer son état à tout moment t et déterminer sa probabilité de présence à une position x et à un moment t : $|\Psi(x,t)|^2$.

Lors de ce projet, nous allons mettre en place une résolution numérique de l'équation de Schrödinger dépendante du temps :

$$i\hbar\frac{\partial}{\partial t}\Psi(x,t) = \hat{H}(t)\Psi(x,t) \tag{1}$$

où $\hat{H}(t)$ est l'hamiltonien du système [2].

1. C'est-à-dire avec une vitesse très inférieure à celle de la vitesse de la lumière $v \ll c$.
2. L'hamiltonien d'un système étant la représentation sous forme de matrice de son énergie totale (en classique, on a par exemple pour une particule se déplaçant à une dimension et soumise à un potentiel V(x) :

$$H = \frac{p^2}{2m} + V(x)$$

avec x la position et p l'impulsion (aussi appelé quantité de mouvement) tel que $p = m\frac{\partial x}{\partial t}$.

L'utilité finale sera de résoudre cette équation avec des potentiels différents (puit de potentiel, barrière de potentiel, potentiel harmonique...), ou différents états initiaux (état stationnaire, superposition d'états stationnaires, paquet d'onde gaussien...).
De plus, nous allons devoir discuter de la discrétisation, que celle-ci soit spatiale ou temporelle, et de quelle manière elle affecte notre simulation.

Nous commencerons par une résolution des états stationnaires, pour ensuite traiter la résolution numérique temporelle.

Deuxième partie
Méthodes

1 Simplification de l'équation de Schrödinger

Ici, nous nous intéresserons au cas d'une particule de masse m soumise à l'action d'un potentiel unidimensionnel indépendant du temps. On obtient :

$$i\hbar\frac{\partial}{\partial t}\Psi(x,t) = \left[-\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + V(x)\right]\Psi(x,t) \tag{2}$$

que l'on peut simplifier en prenant un système d'unité différent [3], on obtient :

$$i\frac{\partial}{\partial T}\Psi(X,T) = \left[-\frac{1}{2}\frac{\partial^2}{\partial X^2} + V(X)\right]\Psi(X,T) \tag{3}$$

2 États stationnaires

Dans un premier temps, nous considérerons uniquement les états stationnaires du problème, ce qui permettra de traiter d'abord la discrétisation spatiale.

2.1 Obtenir l'hamiltonien du système à partir du potentiel

Nous choisissons une longueur L, et un nombre de pas N. Nous obtenons alors $dx = \frac{2L}{N-1}$ [4] qui sera notre pas spatial. Nous pouvons ensuite créer un vecteur x de dimension N.

De même pour V, le potentiel indépendant du temps, que l'on définit sur un vecteur de dimension N.

$$x = \begin{pmatrix} -L \\ -L + dx \\ -L + 2\cdot dx \\ \vdots \\ L - dx \\ L \end{pmatrix} \qquad V = \begin{pmatrix} V_0 \\ V_1 \\ \vdots \\ V_{N-1} \\ V_N \end{pmatrix}$$

3. Nos professeurs nous ont précisé que puisque nous n'utiliserons pas d'application numérique à un système réel, nous n'avons pas besoin de trouver les valeurs des nouvelles variables (X, M, T). Nous retirons donc les unités et on pose $\hbar = 1$.

4. On utilise ici $2L$ car nous voulons que notre simulation couvre l'espace de -L à L, et N-1 pour contenir -L et L.

Nous pouvons ensuite déterminer que H est une matrice tridiagonale symétrique [5] telle que :

$$H = \begin{pmatrix} \frac{2}{dx^2}+V_0 & -\frac{1}{dx^2} & 0 & 0 & 0 & \dots & 0 \\ -\frac{1}{dx^2} & \frac{2}{dx^2}+V_1 & -\frac{1}{dx^2} & 0 & 0 & \dots & 0 \\ 0 & -\frac{1}{dx^2} & \frac{2}{dx^2}+V_2 & -\frac{1}{dx^2} & 0 & \dots & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ \dots & 0 & -\frac{1}{dx^2} & \frac{2}{dx^2}+V_i & -\frac{1}{dx^2} & 0 & \dots \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & \dots & 0 & 0 & 0 & -\frac{1}{dx^2} & \frac{2}{dx^2}+V_{n-1} \end{pmatrix} \tag{4}$$

Simplifions les calculs en utilisant les fonctions déjà fournies par Numpy [6], nous obtenons la fonction suivante qui nous donne H pour tout potentiel indépendant du temps donné et ensuite calcule les états stationnaires en diagonalisant H :

```python
# On cree la fonction pour former les etats stationnaires a partir de V
def get_psi_statio(potentiel):
    """
    Return (w,v) with w the eigenvalues of H and v eigenvectors
    :param potentiel: potential to find w and v
    :return: (w,v)
    """
    d,e = get_matrice(potentiel)
    #On recupere la matrice de la formule
    #donne par l'equation 2

    ## Puis, on calcule w et v respectivement les valeurs propres
    ## et vecteurs propres de H
    w, v = eigh_tridiagonal(d,e)
    v0 = np.zeros((N,N))
    for i in range(N):
        for y in range(N): #On echange ligne et colonnes
            v0[i][y] = v[y][i]
    # On doit desormais normaliser les vecteurs propres :
    for vec in v0:
        normalize(vec,dx)
    return w,v0
```

Listing 1 – get_psi_statio() - Fichier statio.ipynb

Cette fonction nous renvoie w le vecteur composé des énergies propres E_n et v_0 la matrice contenant les vecteurs propres de H.

2.2 Différences entre les potentiels et formation de ceux-ci

Avant de réaliser les différentes formes de potentiel, comparons les valeurs propres entre le calcul numérique et théorique autour d'une particule dans un puits de potentiel infini.

La spécificité d'un puits de potentiel infini est que le potentiel est infini après les bornes. C'est-à-dire que la particule ne peut se trouver après ces bornes. Or lorsque l'on calcule la fonction d'onde d'une particule libre avec un ordinateur, nous incluons indirectement un potentiel infini de chaque

5. Cette formule nous a été donné par notre professeur, et est trouvable sur le TP1 2023 de 3P122.
6. Regarder la vidéo sur YouTube expliquant comment utiliser la diagonalisation d'une matrice tridiagonale [1].

côté de la courbe. La particule ne peut s'y trouver car l'ordinateur ne calcule pas en dehors des limites que nous avons fixées précédemment.

Nous avons facilement les valeurs propres numériques en appliquant la fonction *get_psi(V)* sur un potentiel constant et nul. Pour les valeurs propres théoriques, nous utilisons la formule [7] :

$$E_p = (\frac{\pi(p+1)}{2})^2$$

Nous allons de plus ajouter le carré de la différence entre les deux courbes obtenues afin de bien voir l'écart entre les deux méthodes.

Nous pouvons désormais aussi calculer analytiquement $\Psi(x)$ théorique. Pour le puit de potentiel infini, nous allons utiliser la fonction suivante :

$$\Psi(x) = \sqrt{\frac{2}{L}} cos\left(\frac{(p+1)\pi x}{L} + \pi p/2\right) \quad \text{(si p est pair)}$$

$$\Psi(x) = \sqrt{\frac{2}{L}} cos\left(\frac{(p+1)\pi x}{L}\right) \quad \text{(si p impair)}$$

Et pour le potentiel harmonique :

$$\Psi(x) = \frac{1}{\sqrt{2^p p! \sqrt{\pi}\sigma}} e^{-\frac{x^2}{2\sigma^2}} \mathcal{H}(\frac{x}{\sigma})(-1)^p \tag{5}$$

où ici $\mathcal{H}$ est la fonction Hermite et où $\sigma = 2^{1/4}$.

Ensuite, nous pourrons appliquer la résolution des états stationnaires pour des potentiels plus complexes, tels que le double puits de potentiels ou un potentiel périodique.

3 Évolution temporelle

Nous souhaitons désormais résoudre l'équation de Schrödinger dépendante du temps.
Nous posons de même T le temps final à atteindre et un nombre de pas temporel M. Nous obtenons alors $dt = \frac{T}{M-1}$ qui sera notre pas temporel.

3.1 Méthode d'Euler explicite

Nous allons tout d'abord tenter d'utiliser la méthode d'Euler explicite [8] :

$$\Psi(x, t+1) = \Psi(x,t) + dt \cdot f(\Psi(x,t)) \tag{6}$$

avec

$$f(\Psi(x,t)) = \frac{\partial}{\partial t}\Psi(x,t) = -iH\Psi(x,t)$$

donné par l'équation 1.

Nous avons dû utiliser des nombres complexes avec NumPy ce qui a été possible à l'aide du site [3].

7. De même, cette formule est visible sur le poly de TP1-3P122

8. Voir le site qui explique dans les détails les mathématiques derrière la méthode d'Euler et comment l'implémenter dans son programme [2].

Cette méthode n'est pas la plus efficace comme ne le verrons dans la partie "Résultats". Elle est très gourmande en ressource car demande un dt minime, et donc une grande quantité d'itération pour atteindre un temps T.

3.2 Méthode de Runge-Kutta d'ordre 4

Par la suite, nous utiliserons une autre méthode : celle de Runge-Kutta d'ordre 4 [9]. Nous avons pour cet algorithme :

$$\Psi(x,t+1) = \Psi(x,t) + dt \cdot (\frac{1}{6}k_1 + \frac{1}{3}k_2 + \frac{1}{3}k_3 + \frac{1}{6}k_4) \tag{7}$$

avec :

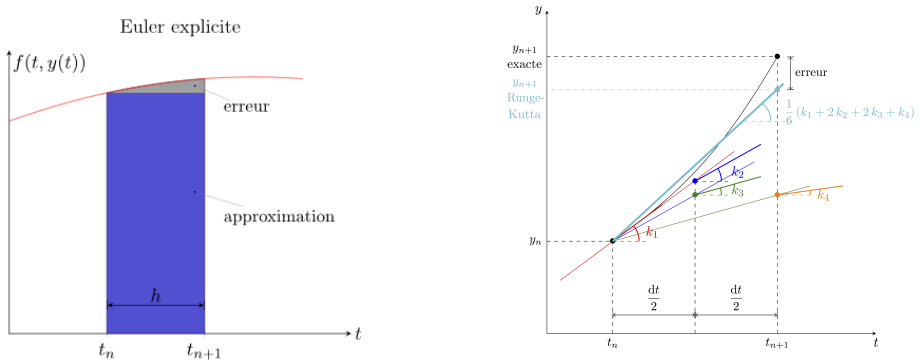
$$\begin{aligned} k_1 &= f(\Psi(x,t)) \\ k_2 &= f(\Psi(x + \frac{1}{2}dt \cdot k_1, t + \frac{dt}{2})) \\ k_3 &= f(\Psi(x + \frac{1}{2}dt \cdot k_2, t + \frac{dt}{2})) \\ k_4 &= f(\Psi(x + dt \cdot k_3, t + dt)) \end{aligned}$$

Ici, puisque H est indépendant du temps, nous pouvons donc avoir :

$$\begin{aligned} k_1 &= f(\Psi(x,t)) \\ k_2 &= f(\Psi(x + \frac{1}{2}dt \cdot k_1, t)) \\ k_3 &= f(\Psi(x + \frac{1}{2}dt \cdot k_2, t)) \\ k_4 &= f(\Psi(x + dt \cdot k_3, t)) \end{aligned}$$

Nous ferons remarquer que cet algorithme est bien plus efficace qu'Euler. En effet, pour un même dt, même faible, celui-ci nous permet d'obtenir un résultat cohérent. Ceci est dû au fait que la méthode de Runge-Kutta d'ordre 4 est une étape supplémentaire dans le raffinement du calcul de l'intégrale. Au lieu d'utiliser la méthode du rectangle à gauche comme Euler, nous utiliserons la méthode des trapèzes pour Runge-Kutta d'ordre 2 et la méthode de Simpson pour celle d'ordre 4 (voir Figure 2).

9. Voir le site [4]



(a) Méthode d'Euler explicite (b) Méthode de RK4 et ses différents composants

FIGURE 2 – Différences entre les méthodes d'Euler et de Runge-Kutta à l'ordre 4

3.3 Amélioration de la vitesse : Numba

Tous ces calculs sont gourmands en ressources. En effet, on itère M fois, sur des vecteurs et des matrices de dimensions N.

Si l'on souhaite obtenir des simulations efficaces et précises, nous devons trouver un moyen d'améliorer la vitesse de calcul.

Nous utiliserons donc la librairie Numba, qui accélère la vitesse de Python et lui permet d'atteindre une efficacité proche de cette du langage C. Durant nos tests, nous avons pu constater une différence de vitesse de rendu de l'ordre de 4x supérieure.

3.4 Cas particuliers intéressants d'évolution temporelle

Une fois la bonne méthode sélectionnée, nous pouvons choisir des états initiaux différents.

Tout d'abord, nous pouvons utiliser un état stationnaire pour voir l'efficacité de la méthode d'Euler. En effet, l'état stationnaire doit rester constant avec le temps, puisqu'on a $H|E_n\rangle = E_n|E_n\rangle$. Puisque notre vecteur est normalisé, il reste constant.

Nous pouvons alors déterminer la "finesse" de notre dt, en effet, si $\Psi(x)$ varie rapidement [10], notre dt est bien trop grand.

Ensuite, nous pouvons nous intéresser à une somme d'états stationnaires. La résolution numérique dans un potentiel nul [11] est très simple à calculer analytiquement, ce qui nous permet de comparer les résultats théoriques aux résultats numériques.

En effet, nous pouvons trouver que si :

10. En réalité, $\Psi(x)$ va forcément varier à un moment, même avec une méthode parfaite ; en effet, les valeurs ne sont jamais exactes, ce sont des approximations. Tout d'abord dû au *float* et deuxièmement car la diagonalisation de la matrice ne peut être parfaite. Donc ici notre but est plutôt de voir à partir de quelle itération notre simulation commence à atteindre ses limites.

11. Un potentiel nul peut se rapporter à une barrière de potentiel très large, on prend alors $L \gg 1$

$$\Psi(x, t = 0) = \sum_n c_n |E_n\rangle$$

$$\Psi(x, t) = \sum_n c_n \cdot e^{-\frac{iE_n}{\hbar}t} \cdot |E_n\rangle$$

Enfin, nous allons initier la simulation avec un paquet d'onde gaussien, ce qui représente de manière bien plus réaliste une particule. Nous pourrons alors la mettre dans un potentiel choisi (par exemple un double puit de potentiel harmonique) qui n'a pas de résolution analytique simple.

Pour simuler un paquet d'onde gaussien, nous utilisons la formule [12] :

$$\Psi(x, t = 0) = Ce^{-(\frac{\sigma(x-x_0)}{2})^2} \cdot e^{ikx}$$

où C est une constante qui dépend de la normalisation de Ψ, nous n'avons pas à la définir car celle-ci sera normalisée à chaque itération par la fonction *normalize()*.

Troisième partie

Résultats et analyses

Dans cette partie, nous allons vous présenter les résultats numériques sous formes de graphes, qui seront dans un premier temps comparés aux résultats théoriques.

Nous n'avons pas mis d'unité sur les graphiques. En effet, comme vu à l'équation adimensionnelle (3), nous avons pu remplacer x, m et t en de nouvelles variables.

1 États stationnaires

1.1 Énergies numérique et théorique pour un puit de potentiel infini

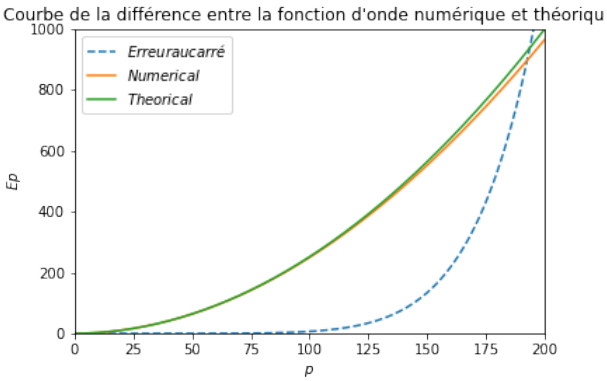


FIGURE 3 – Fichier : *energie_num_vs_theoric_n1000*

12. Nous avons trouvé une formule avec un σ en numérateur, même si il est normalement utilisé en dénominateur. Cela ne posera pas de problème ici car σ n'a pas de sens physique dans notre simulation mais nous préférons le signaler.

Nous voyons qu'aux petites énergies, la courbe théorique et numérique sont extrêmement proches, la courbe d'erreur est nulle. Mais pour $p > 100$, la différence s'accentue et l'erreur augmente exponentiellement. Nous concluons donc que la fonction numérique n'est utilisable qu'à basse énergie.

1.2 Puits infini

Nous obtenons alors pour 3 niveaux d'énergie :

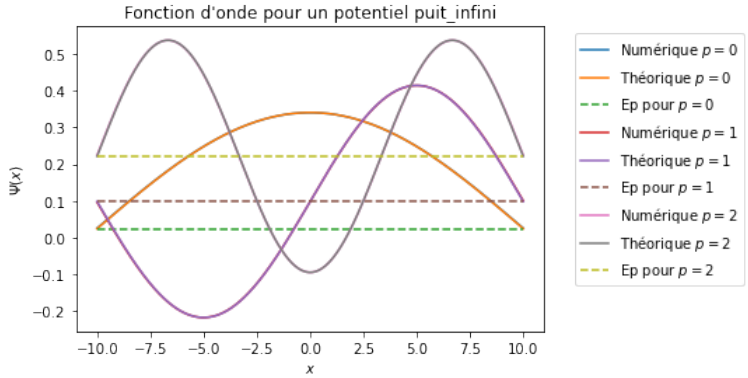


FIGURE 4 – Fichier : *puit_ infini*

Nous ne distinguons pas de différence entre la courbe théorique et numérique. La méthode numérique est donc fiable pour une particule dans un puits infini. Mais ce n'est que le potentiel le plus simpliste, nous allons donc comparer maintenant avec un potentiel plus complexe.

1.3 Potentiel harmonique

Après computation, nous obtenons :

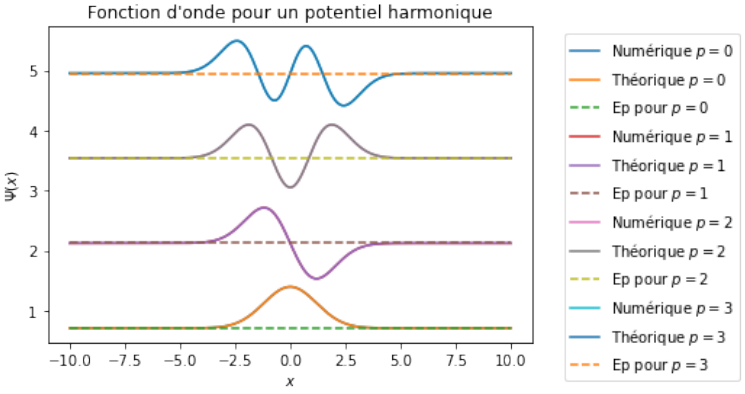


FIGURE 5 – Fichier : *harmonique*

Nous remarquons une faible variance entre la courbe théorique et numérique qui s'accentue avec l'augmentation de l'énergie. La forme reste toutefois la même dans les deux cas.

1.4 Double potentiel harmonique

Nous pouvons aussi vérifier la précision de notre fonction numérique via un double puits harmonique. Si nous utilisons deux fois la fonction du puits harmonique, on peut faire apparaître la forme

de psi dans ce potentiel. En appliquant notre fonction numérique, nous trouvons :

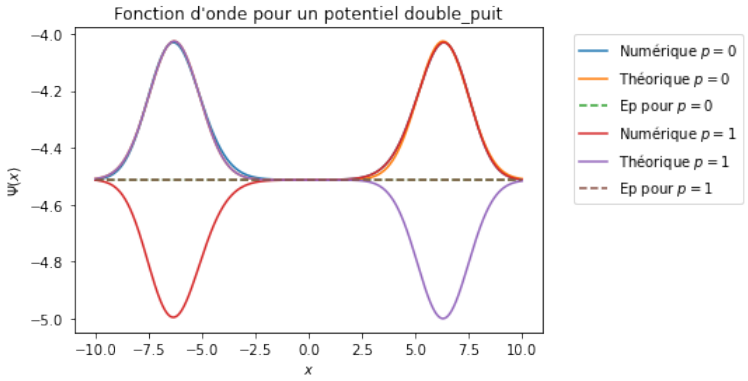


FIGURE 6 – Fichier : *double_puit*

Nous voyons distinctement que nos deux fonctions commencent à être trop différentes. Ce résultat est compréhensible car la fonction théorique de ce potentiel à double puits harmonique a été calculé comme une simple somme de deux puits harmoniques, et il n'y a pas de couplage pris en compte. Nous pouvons tout de même voir que la forme et le signe restent les mêmes.

1.5 Potentiel periodique

Finalement, mettons notre fonction numérique réellement à l'épreuve avec un potentiel harmonique périodique. Après la computation habituelle, nous avons :

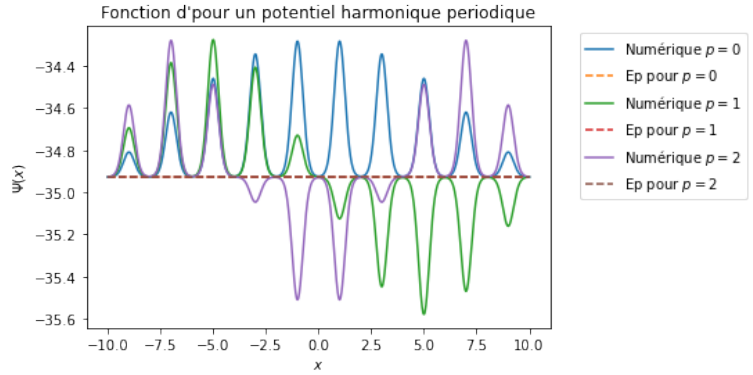


FIGURE 7 – Fichier : *periodique*

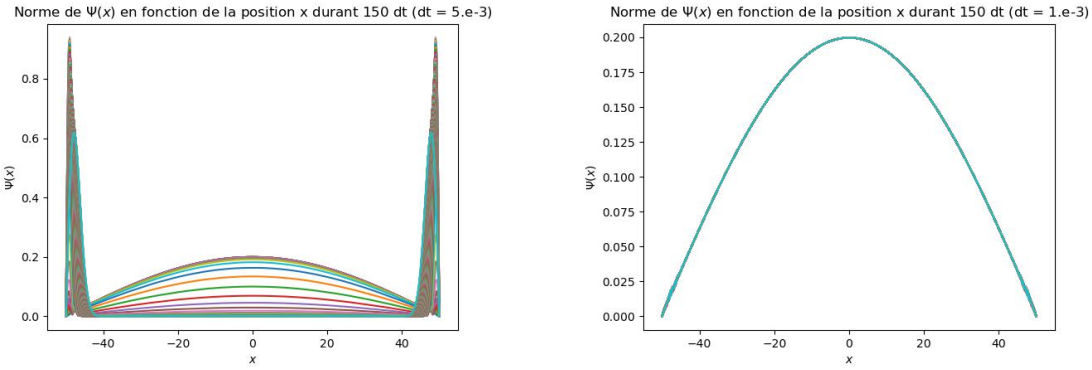
La forme de la courbe est bien périodique mais il est facile de remarquer que la forme générale de la courbe suit la forme de la courbe de la particule dans un puits de potentiel infini. C'est compréhensible car les limites de computations font office de potentiel infini. Nous avons donc deux courbes en une : la courbe du potentiel périodique et la courbe du puits infini.

2 Évolution temporelle

2.1 Évolution temporelle d'un état stationnaire

Puisque nous ne pouvons pas afficher de *GIF* sur ce PDF, nous allons représenter l'évolution avec différentes courbes. Chaque courbe est une nouvelle itération de $\Psi(x)$, et puisque dt est faible, nous devrions comprendre l'évolution de celui-ci. Pour les évolutions temporelles finales, nous vous proposerons de jeter un coup d'oeil aux GIFs correspondants.

Pour $|E_1\rangle$, nous obtenons avec Euler la figure [13] 8.



(a) M = 1000 (b) M = 5000

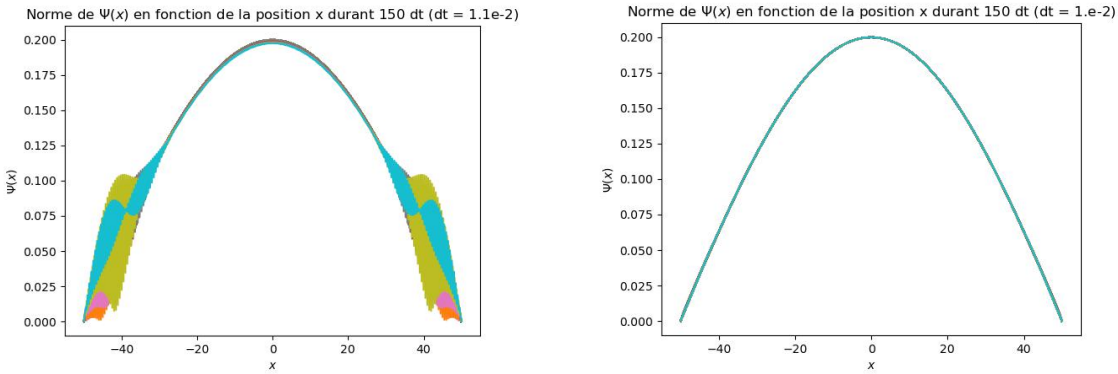
FIGURE 8 – Différences pour la Méthode d'Euler entre un $dt = 5 \cdot 10^{-3}$ et un $dt = 1 \cdot 10^{-3}$ - Fichier : *psi_fonction_de_x_euler_etat_statio0*

Nous voyons bien ici que pour un $dt = 5 \cdot 10^{-3}$ qui est déjà plutôt faible, l'état stationnaire ne reste pas constant. Il y a trop d'erreurs ! Par contre, pour un $dt = 10^{-3}$, celui-ci reste plutôt constant (nous remarquons par contre des petites modifications aux niveaux des extrémités qui amèneront rapidement à des grosses différences).

Utilisons donc la méthode de Runge-Kutta à l'ordre 4 et regardons ses résultats pour les mêmes valeurs de dt et durant un même temps $150 \cdot dt$. Nous obtenons la figure 9.

Nous pouvons rapidement remarquer sur 9 que notre simulation est bien plus précise, et ce, pour dt beaucoup plus petit. La quantité de calcul (ou nombres d'itérations) sera donc beaucoup plus faible [14].

13. Les fichiers finissent tous par *.png* et ils sont suivis des paramètres (M, durée etc...).
14. On doit itérer M fois pour atteindre T.



(a) M = 448 (b) M = 500

FIGURE 9 – Différences pour la Méthode de RK4 entre un $dt = 1.1 \cdot 10^{-2}$ et un $dt = 1 \cdot 10^{-2}$ - Fichier : *psi_fonction_de_x_rk4_etat_statio0*

2.2 Évolution temporelle d'une somme d'états stationnaires

Ici, notre but est de comparer l'évolution temporelle d'une somme d'états stationnaires entre une évolution théorique et une numérique. Prenons donc

$$|\Psi(x,t)\rangle = 2 \cdot |E_1\rangle + |E_2\rangle$$

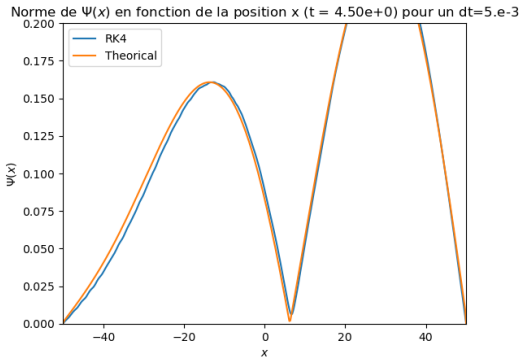


FIGURE 10 – Différence entre la norme théorique et numérique d'un vecteur d'état $|\Psi\rangle$ à t=4,5. Méthode : RK4 - Fichier : *anim_stationnaire_diff_theorical2*

Ici, nous voyons le résultat final de la simulation, c'est-à-dire la dernière itération sur les 1000. Nous pouvons remarquer que la différence est très faible, ce qui nous conforte dans l'idée que notre simulation est plutôt efficace [15].

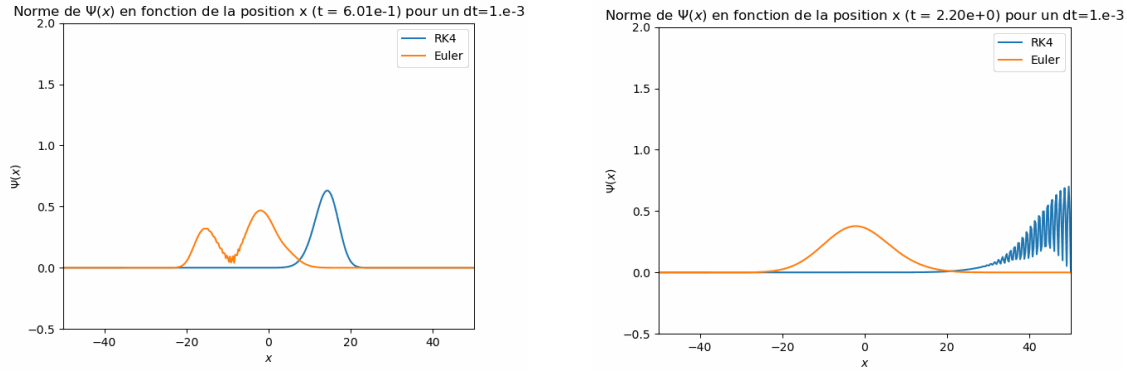
Nous pouvons donc conclure cette partie sur la vérification de notre simulation temporelle. Celle-ci est valide et on peut désormais passer à des états initiaux plus complexes qui n'ont pas de résolution analytique simple.

15. Nous pouvons voir ici que l'on a choisi un dt plus petit que précédemment. Même si un $dt = 10^{-2}$ nous suffisait pour la conservation d'un état stationnaire sur une centaine d'itérations, ici nous avons besoin de bien plus de précisions.

2.3 Évolution temporelle d'un paquet d'onde gaussien

Dans cette partie, nous allons construire un état initial $\Psi(x)$ sous forme d'un paquet d'onde gaussien.

Tout d'abord, nous prenons le cas d'une particule libre avec un déplacement vers les x croissants. Nous obtenons [16] la figure 11.



(a) Ici la méthode d'Euler se transforme par erreur et ne représente plus l'état réel de Ψ.

(b) On remarque ici que la fonction provenant de RK4 rencontre les bordure de la simulation.

FIGURE 11 – Photos de l'animation *anim_gauss_pot_nul_séparé* prises à différents t pour expliquer certains comportements.

Nous pouvons remarquer que la méthode de RK4 est bien plus précise, puisque la fonction d'onde dans la méthode d'Euler se "brise" (a) rapidement et semble reprendre une forme de paquet d'onde gaussien fixe.

Par contre, avec notre méthode de RK4, la particule va "rebondir" contre le mur (b). Le mur est un potentiel infini ce qui empêche l'onde de se transmettre et elle va donc repartir vers les x décroissants. Il est à ce point difficile de savoir si le bruit visible lors du rebond est physiquement valable ou si ce n'est dû qu'à l'approximation spatiale dx.

Nous pouvons le voir avec un graphique qui représente la fonction d'onde avec la méthode RK4 uniquement.

16. Voir GIF : anim_gauss_pot_nul_séparé_M_5000

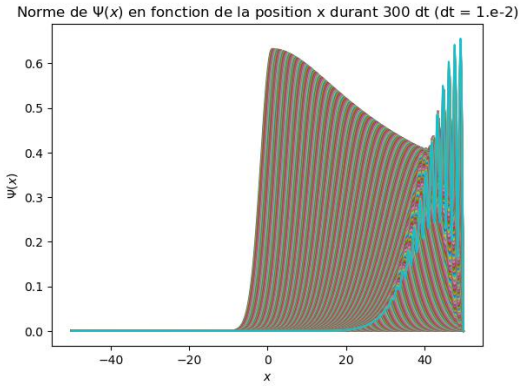


FIGURE 12 – Paquet d'onde gaussien libre se déplaçant vers les x positifs. Méthode : RK4 - Fichier : *psi_fonction_de_x_rk4_gaussienne*

Prenons désormais un potentiel harmonique. Classiquement, une particule avec une certaine quantité d'impulsion devrait se mouvoir dans la "vallée" jusqu'à atteindre un point où $V(x) = E$ puis repartir dans le sens contraire, et ce, à infini.

Voyons ce que nous donne la simulation quantique sur la figure 13.

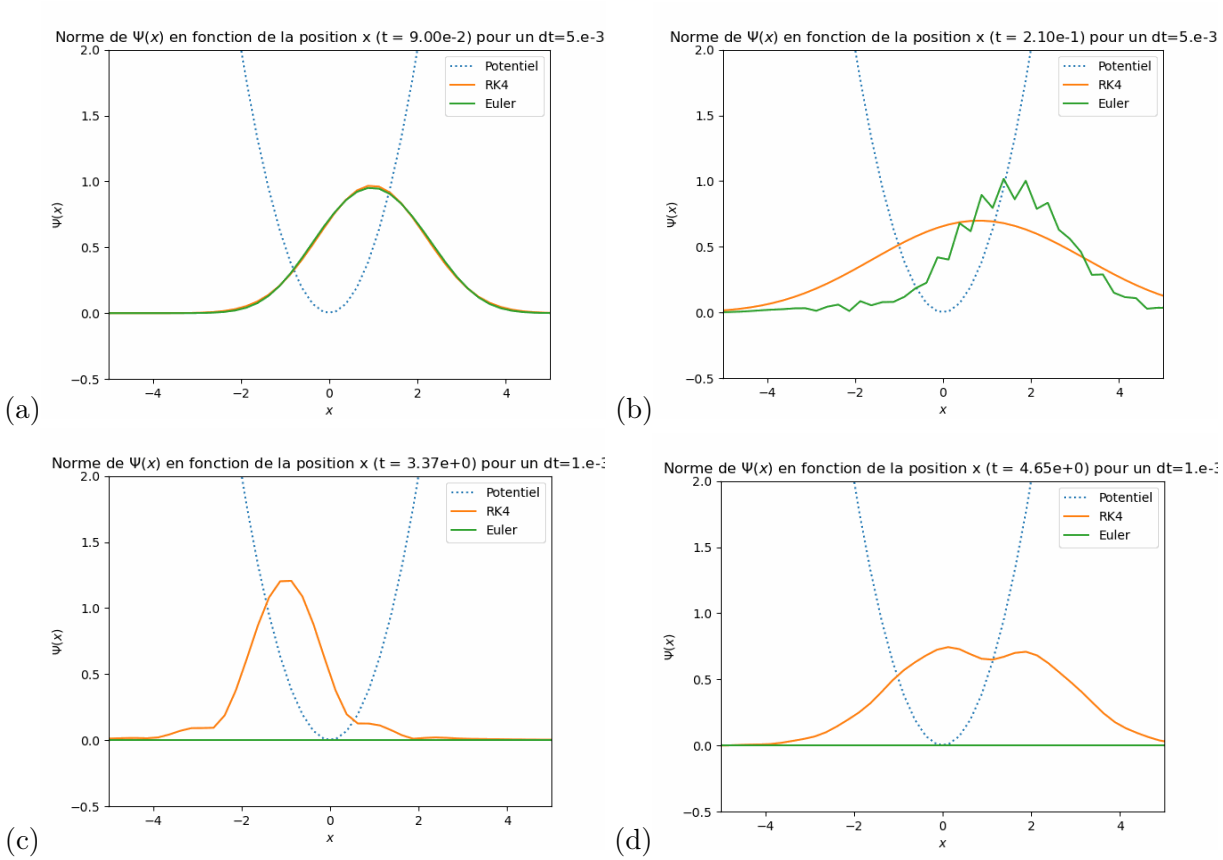


FIGURE 13 – Fichier : *anim_gauss_potentiel_double_puit*

Nous remarquerons ici encore une fois que la méthode d'Euler ne suffit pas. Pour RK4, nous remarquons bien le mouvement attendu, la fonction d'onde parait "glisser" dans le puits de potentiel.

On remarque tout de même quelques approximations. En effet, on peut voir apparaître une "onde évanescente" provenant des bordures qui est le rebond de notre fonction d'onde contre les murs de notre simulation.

De plus, on remarque que lorsque notre fonction d'onde atteint le maximum de $|x|$, sa forme gaussienne n'est plus si précise. On peut alors faire remarquer que notre dx n'est pas si faible, mais le diminuer ne ferait qu'augmenter exponentiellement le temps de rendu de notre simulation.

Quatrième partie
Conclusion

Durant ce projet de physique numérique, nous avons pu mettre en place une résolution numérique de l'équation de Schrödinger. Tout d'abord en résolvant les états propres de l'Hamiltonien, puis en intégrant l'équation 1 à l'aide de plusieurs méthodes (Euler, RK4).

Il n'est pas simple de se représenter la mécanique quantique, ou même de trouver des solutions analytiques à certaines de ses applications. Mais à l'aide des outils numériques, nous pouvons résoudre des problèmes complexes.

Cinquième partie

Sources

Références

[1] Vidéo youtube expliquant comment créer une matrice tridiagonale avec scipy. https://www.youtube.com/watch?v=tsK72kSgPoI.

[2] Méthode d'euler explicite. https://www.f-legrand.fr/scidoc/docmml/numerique/euler/euler/euler.html.

[3] Explication sur comment utiliser les nombres complexes dans les matrices avec numpy. https://www.moonbooks.org/Articles/How-to-create-a-matrix-of-complex-numbers-in-python-using-numpy-/.

[4] Explication sur comment utiliser et implémenter runge-kutta d'ordre 4 et ses différences avec euler. https://femto-physique.fr/analyse-numerique/runge-kutta.php.