TP 2 : Méthodes Mathématiques et informatique

Boris Alexandre Baudel - Université du Maine - Master de physique

Octobre 2023

1 Introduction

La méthode de Numerov, du nom d'un astronome russe (Boris Vasilyevich Numerov), permet de résoudre une équation différentielle de la forme :

$$y''(x) + Q(x)y(x) = S(x), \tag{1}$$

où Q et S sont des fonctions continues sur l'intervalle $[a, b]$. Cette équation doit être résolue avec certaines conditions aux limites : $y(a)$ et $y(b)$ sont données par exemple. Cette méthode se prête particulièrement bien à la résolution de l'équation de Schrödinger stationnaire décrivant une particule de masse m confinée dans un puits de potentiel unidimensionnel $V(x)$:

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

où $\Phi(x)$ est la fonction d'onde pour une énergie E donnée (qu'on ne connaît pas).

2 Discrétisation et Résolution

La méthode de Numerov, du nom d'un astronome russe (Boris Vasilyevich Numerov), permet de résoudre une équation différentielle de la forme :

$$y''(x) + Q(x)y(x) = S(x), \tag{3}$$

où Q et S sont des fonctions continues sur l'intervalle $[a, b]$. Cette équation doit être résolue avec certaines conditions aux limites : $y(a)$ et $y(b)$ sont données par exemple.

Cette méthode se prête particulièrement bien à la résolution de l'équation de Schrödinger stationnaire décrivant une particule de masse m confinée dans un puits de potentiel unidimensionnel $V(x)$:

$$-\frac{\hbar^2}{2m}\frac{d^2}{dx^2}\Phi(x) + V(x)\Phi(x) = E\Phi(x), \tag{4}$$

où $\Phi(x)$ est la fonction d'onde pour une énergie E donnée (qu'on ne connaît pas).

Revenons à la résolution de l'équation différentielle (1). On considère un échantillonnage régulier de N points x_i définis par :

$$x_i = a + \frac{b-a}{N-1}i \quad \text{où} \quad i \in [0, N-1]. \tag{5}$$

Toutes les fonctions sont discrétisées sur cette grille de points et on introduit : $y(x_i) = y_i$, $Q(x_i) = Q_i$, $S(x_i) = S_i$ où $i \in [0, N-1]$.

Nous allons chercher à évaluer $y''(x) \equiv y^{(2)}(x)$ en effectuant un développement en série de Taylor de la fonction $y(x)$. Il vient alors :

$$y(x+\Delta) = y(x) + \Delta y'(x) + \frac{\Delta^2}{2}y''(x) + \frac{\Delta^3}{6}y'''(x) + \frac{\Delta^4}{24}y''''(x) + \dots$$

où $\Delta = \frac{b-a}{N-1}$ représente le pas. On peut également écrire :

$$y(x-\Delta) = y(x) - \Delta y'(x) + \frac{\Delta^2}{2}y''(x) - \frac{\Delta^3}{6}y'''(x) + \frac{\Delta^4}{24}y''''(x) + \dots$$

En ajoutant ces deux équations, on obtient :

$$y(x+\Delta) + y(x-\Delta) = 2y(x) + \Delta^2 y''(x) + \frac{\Delta^4}{12}y''''(x) + O(\Delta^6)$$

En exploitant les notations discrètes introduites précédemment, il vient :

$$y_i^{(2)} = \frac{y_{i+1} + y_{i-1} - 2y_i}{\Delta^2} - \frac{\Delta^2}{12}y_i^{(4)} + O(\Delta^4) \tag{6}$$

Dans un schéma à trois points, la dérivée seconde s'écrit donc sous la forme :

$$y_i^{(2)} = \frac{y_{i+1} + y_{i-1} - 2y_i}{\Delta^2} + O(\Delta^2) \tag{7}$$

On peut exploiter l'équation (1) pour calculer explicitement le terme impliquant $y_i^{(4)}$ dans l'équation (4). Il vient alors :

$$y_i^{(4)} = -\frac{d^2}{dx^2}[Qy - S]_i \tag{8}$$

En exploitant l'équation (5), il vient alors :

$$y_i^{(4)} = -\frac{1}{\Delta^2}[(Q_{i+1}y_{i+1} + Q_{i-1}y_{i-1} - 2Q_iy_i) - (S_{i+1} + S_{i-1} - 2S_i)] + O(\Delta^2) \tag{9}$$

En substituant ce résultat dans l'équation (4), on obtient :

$$\begin{aligned}\left(1 + \frac{\Delta^2}{12}Q_{i+1}\right) y_{i+1} = 2\left(1 - \frac{5\Delta^2}{12}Q_i\right) y_i \\ - \left(1 + \frac{\Delta^2}{12}Q_{i-1}\right) y_{i-1} + \frac{\Delta^2}{12}[S_{i+1} + 10S_i + S_{i-1}] + O(\Delta^6)\end{aligned} \tag{10}$$

Cette relation de récurrence relie y_i, y_{i+1} et y_{i-1}. L'erreur commise étant d'ordre 6 en Δ, le schéma est très précis. On peut alors définir un schéma d'intégration vers l'avant en écrivant $y_{i+1} = f(y_i, y_{i-1})$ ou encore un schéma d'intégration vers l'arrière en écrivant $y_{i-1} = g(y_i, y_{i+1})$. La connaissance de deux valeurs successives de $y(x)$ sur la grille permet donc de calculer la solution sur l'ensemble de l'intervalle $[a, b]$.

3 Résolution d'une équation différentielle

La fonction numerov1(x, Q, S, y), qui permet d'obtenir la solution de l'´equation diff´erentielle 1 sur une grille x comportant N points d´efinis par l'´equation 3. Ici, Q, S et y sont des tableaux de dimension N contenant les valeurs discr´etis´ees des fonctions Q, S et y sur la grille x. On note que y[0] et y[1] doivent contenir les deux valeurs cons´ecutives de la fonction y r´esultants des conditions aux limites

```
import numpy as np
def numerov1(x, Q, S, y):
    N = len(x)
    delta = x[1] - x[0]  # Calcul du pas de la grille

    for i in range(1, N-1):
        factor1 = 1 + (delta**2 / 12) * Q[i+1]
        factor2 = 2 * (1 - (5 * delta**2 / 12) * Q[i])
        factor3 = 1 + (delta**2 / 12) * Q[i-1]
        S_term = (delta**2 / 12) * (S[i+1] + 10 * S[i] + S[i-1])

        y[i+1] = (factor2 * y[i] - factor3 * y[i-1] + S_term) / factor1

    return y
```

On considère l'équation différentielle

$$y'' + y = \sin(x).$$

La solution générale de l'équation homogène associée $y'' + y = 0$ est

$$y_h(x) = K_1 \sin(x) + K_2 \cos(x),$$

où K_1 et K_2 sont des constantes. Une solution particulière de l'équation non homogène est

$$y_p(x) = -\frac{x}{2}\cos(x).$$

Ainsi, la solution générale de l'équation non homogène est

$$y(x) = K_1 \sin(x) + K_2 \cos(x) - \frac{x}{2}\cos(x).$$

En appliquant les conditions initiales $y(0) = 0$ et $y'(0) = 1$, on trouve les valeurs des constantes K_1 et K_2.

$$y(x) = \sin(x) - \frac{x}{2}\cos(x).$$

3.1 Solution Numérique

Le code Python suivant utilise la méthode de Numerov pour résoudre l'équation différentielle numériquement et comparer la solution numérique à la solution analytique.

```
import numpy as np
import matplotlib.pyplot as plt

import numpy as np
import matplotlib.pyplot as plt

def numerov1(x, Q, S, y):
    N = len(x)
    delta = x[1] - x[0]  # Calcul du pas de la grille

    for i in range(1, N-1):
        factor1 = 1 + (delta**2 / 12) * Q[i+1]
        factor2 = 2 * (1 - (5 * delta**2 / 12) * Q[i])
        factor3 = 1 + (delta**2 / 12) * Q[i-1]
        S_term = (delta**2 / 12) * (S[i+1] + 10 * S[i] + S[i-1])

        y[i+1] = (factor2 * y[i] - factor3 * y[i-1] + S_term) / factor1

    return y
# D finition de la grille x
a, b = 0, 20
N = 10000  # Nombre de points
x = np.linspace(a, b, N)

# Solution analytique avec les constantes K1 et K2 d termin es
def y_analytic(x):
    K1 = 1  # D termin  par y'(0) = 1
    K2 = 0  # D termin  par y(0) = 0
    return K1 * np.sin(x) + K2 * np.cos(x) - 0.5 *x* np.cos(x)

# Fonctions Q(x) et S(x) pour l' quation  y'' + y = sin(x)
def q(x):
    return 1  # Q(x) = y

def s(x):
    return np.sin(x)  # S(x) = sin(x)

# Choix de
delta = (b-a)/N

# D finitions de Q(x) et S(x)
Q = np.array([q(x_i) for x_i in x])
S = np.array([s(x_i) for x_i in x])

# Conditions initiales correctement initialis es
y = np.zeros(N)
y[0] = 0  # y(0) = 0
y[1] = delta  # y'(0) = 1 et y''(0) = -1

# Calcul de la solution num rique
y_numerov = numerov1(x, Q, S, y)

# Calcul de la solution analytique
y_analytic_vals = y_analytic(x)

# Trac  des solutions
plt.figure(figsize=(12, 6))
```

```
plt.plot(x, y_numerov, label='Solution Num rique', linewidth=2)
plt.plot(x, y_analytic_vals, label='Solution Analytique', linestyle='--', linewidth=2)
plt.xlabel('x')
plt.ylabel('y(x)')
plt.title('Comparaison des Solutions Num rique et Analytique')
plt.legend()
plt.grid(True)
plt.show()
```

On peut alors observer la solution exacte et celle fournit par la méthode de numerov d'ordre 6, on observe un décalage et donc une erreur ce qui peut être liée aux pas de temps utilisé.

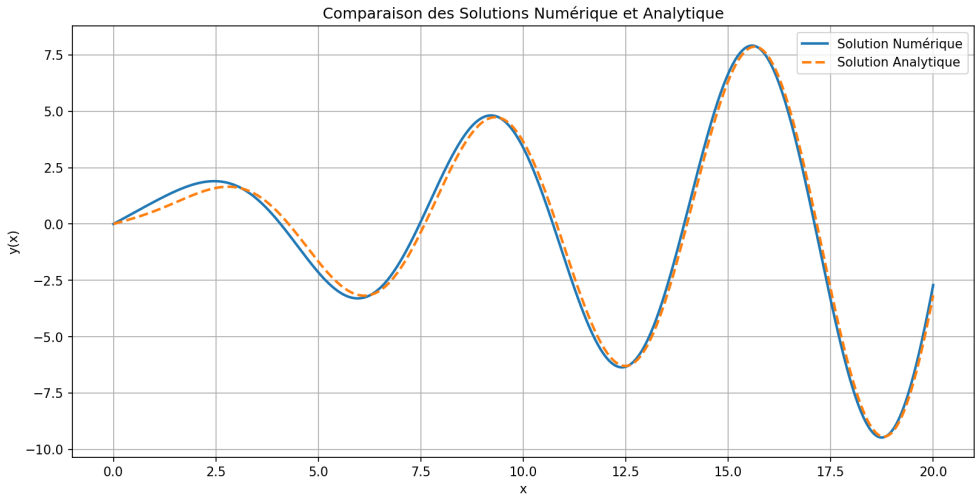

FIGURE 1 – Résultat analytique vs Numerov

4 Equation de Shrodinger

L'équation de Schrödinger stationnaire est donnée par :

$$-\frac{\hbar^2}{2m}\frac{d^2\psi}{dx^2} + V(x)\psi = E\psi \tag{11}$$

Dans les unités atomiques où $m = 1$ et $\hbar = 1$, l'équation devient :

$$\psi''(x) + 2\left(E - V(x)\right)\psi = 0 \tag{12}$$

Pour résoudre l'équation de Schrödinger stationnaire en unités atomiques, où $m = 1$ et $\hbar = 1$, nous avons l'équation :

$$-\frac{1}{2}y''(x) + V(x)y(x) = Ey(x)$$

où $V(x)$ est le potentiel, E est l'énergie, et $y(x)$ est la fonction d'onde. L'équation peut être réarrangée pour être de la forme compatible avec la méthode de Numerov :

$$y''(x) = -2[E - V(x)]y(x)$$

Dans cette forme, $Q(x) = 2[E - V(x)]$ et l'équation devient $y''(x) + Q(x)y(x) = 0$, où $Q(x)$ est maintenant une fonction de x qui inclut le potentiel et l'énergie.

La fonction 'numerov2' peut être écrite en Python en s'inspirant de la méthode 'numerov1' précédente, mais sans le terme de source $S(x)$ car il n'apparaît pas dans l'équation de Schrödinger standard.

```python
import numpy as np

def numerov2(x, Q, y):
    N = len(x)
    delta_x = x[1] - x[0]  # Pas de la grille

    # La boucle commence    1 car nous avons besoin de y[0] et y[1] pour d marrer
    for i in range(1, N-1):
        factor1 = 1 + (delta_x**2 / 12) * Q[i+1]
        factor2 = 2 * (1 - (5 * delta_x**2 / 12) * Q[i])
        factor3 = 1 + (delta_x**2 / 12) * Q[i-1]
        y[i+1] = (factor2 * y[i] - factor3 * y[i-1]) / factor1

    return y
```

La fonction setQ que vous demandez peut être définie pour calculer les tableaux Ql et Qr à partir de l'énergie E et du potentiel V(x). Ces tableaux représentent les valeurs discrétisées de la fonction Q(x)=2(EV(x)) pour l'intégration de l'équation de Schrödinger de la gauche vers la droite (Ql) et de la droite vers la gauche (Qr).

```python
import numpy as np

def setQ(E, V, x):
    # Initialiser les tableaux Ql et Qr avec la m me taille que V
    Ql = np.zeros_like(V)
    Qr = np.zeros_like(V)

    # Calculer Q pour l'int gration de gauche    droite (Ql)
    for i in range(len(x)):
        Ql[i] = 2 * (E - V[i])

    # Calculer Q pour l'int gration de droite    gauche (Qr)
    for i in range(len(x)-1, -1, -1):
        Qr[i] = 2 * (E - V[i])

    return Ql, Qr
```

La fonction turning point peut être implémentée pour trouver le point de rebroussement classique. Vous pouvez le faire en trouvant l'indice pour lequel la différence change de signe. Voici comment vous pourriez coder cette fonction en Python :

```python
import numpy as np

def turning_point(E, V, x):
    # Trouver l'indice o  V(x) est le plus proche de E pour x > 0
    # np.abs(V - E) donne le tableau des valeurs absolues de la diff rence entre V et E
    # On cherche l'indice du minimum de cette diff rence pour les x > 0

    # Condition pour x > 0
    positive_indices = np.where(x > 0)[0]

    # Trouver l'indice du minimum de la diff rence absolue sur les x positifs
    ic = positive_indices[np.argmin(np.abs(V[positive_indices] - E))]

    return ic
```

On considère une particule confinée dans un puits de potentiel carré. L'énergie potentielle de la particule $V(x)$ s'écrit :

$$V(x) = \begin{cases} -V_0 & \text{pour } -a \leq x \leq a \\ 0 & \text{pour } x < -a \text{ ou } x > a \end{cases}$$

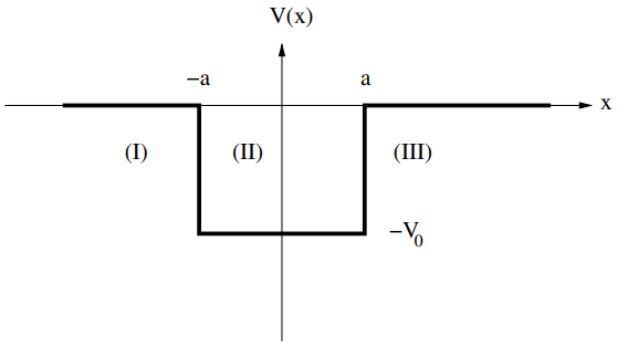

FIGURE 2 – Puit Potentiel

Rajout d'une nouvelle fonction Numérov2 pour facilité le calcule pour la prochaine étape.

```
def V(x, a, V0):
    potential = np.zeros_like(x)
    potential[(x >= -a) & (x <= a)] = -V0
    return potential

def numerov2(x, Q, y, E, V):
    N = len(x)
    h = (x[-1] - x[0]) / (N - 1)
    k2 = 2 * (E - V)  # 2(E-V) based on atomic units

    # Calculate y using Numerov's method
    for i in range(1, N-2):
        term1 = 2 * y[i] - y[i-1]
        term3 = h**2 * (1/12 * (k2[i+1] + 10*k2[i] + k2[i-1]) * y[i])
        y[i+1] = (term1 + term3) / (1 - h**2/12 * k2[i+1])

    return y

def setQ(E, V, x):
    # Initialiser les tableaux Ql et Qr
    Ql = np.zeros_like(V)
    Qr = np.zeros_like(V)

    # Calculer les valeurs de Ql et Qr
    Ql = 2 * (E - V)
    Qr = Ql[::-1]  # Inverser l'ordre des   lments   pour l'int gration de droite    gauche

    return Ql, Qr
# 2. D finir les param tres
a = 1
V0 = 10
E1 = -9.30
E2 = -9.10
xmin, xmax = -2*a, 2*a
N = 1000
x = np.linspace(xmin, xmax, N)
dx = x[1] - x[0]

# 3. Calculer Q(x) pour les deux   nergies
V_values = V(x, a, V0)
Ql1, Qr1 = setQ(E1, V_values, x)
Ql2, Qr2 = setQ(E2, V_values, x)

# 4. D finir les conditions initiales
y_init = np.zeros(N)
```

```
y_init[1] = dx  # c*dx avec c tr s petit

# 5. R soudre l' quation  de Schr dinger avec Numerov
y1 = numerov2(x, Ql1, y_init.copy(), E1, V_values)
y2 = numerov2(x, Ql2, y_init.copy(), E2, V_values)

# 6. Tracer les solutions
plt.figure(figsize=(10, 5))
plt.plot(x, y1, label='E = -9.30')
plt.plot(x, y2, label='E = -9.10')
plt.title('Solutions de l\' quation  de Schr dinger pour un puits de potentiel carr ')
plt.xlabel('x')
plt.ylabel('  (x)')
plt.legend()
plt.grid(True)
plt.show()
```

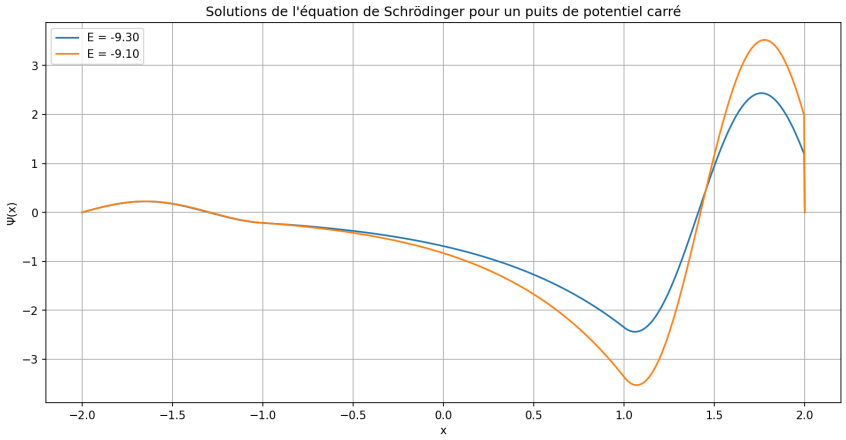

FIGURE 3 – Fonction d'onde pour deux énergies

Comme la fonction d'onde calculée dans la région (III) en intégrant l'équation différentielle de $x_{\min}$ vers l'origine est imprécise, on peut pour une énergie d'essai donnée E intégrer l'équation différentielle de $x_{\min}$ à x_c et obtenir une fonction d'onde $\Phi_L(x)$ définie sur $[x_{\min}, x_c]$, puis intégrer l'équation différentielle de $x_{\max}$ à x_c et obtenir une fonction d'onde $\Phi_R(x)$ définie sur $[x_c, x_{\max}]$.

On doit alors mettre à l'échelle la fonction d'onde $\Phi_R(x)$ en la multipliant par $\frac{\Phi_L(x_c)}{\Phi_R(x_c)}$ de telle sorte que les deux fonctions d'onde soient continues au point de rebroussement x_c. La fonction d'onde totale $\Phi(x)$ peut alors être normalisée en écrivant :

$$\int_{x_{\min}}^{x_{\max}} |\Phi(x)|^2 \, dx = 1.$$

```
def normalize(x, y):
    integral = np.trapz(y**2, x)
    return y / np.sqrt(integral)

# Exemple d'utilisation de la fonction de normalisation
# Remplacez ceci par vos propres donn es ou fonctions d'onde
x = np.linspace(-1, 1, 1000)
phi_L = np.exp(-x**2)  # Exemple de fonction d'onde    normaliser
phi_x = phi_L.copy()  # Exemple de fonction d'onde    normaliser

# Normalisation
phi_L_normalized = normalize(x, phi_L)
phi_x_normalized = normalize(x, phi_x)
```

La fonction d'onde $\Phi(x)$ construite à partir de $\Phi_L(x)$ et $\Phi_R(x)$ ne décrit un état lié d'énergie E que si la dérivée de la fonction d'onde est continue en x_c. Afin de calculer la discontinuité de la dérivée avec une bonne précision, on effectue un développement de Taylor de $\Phi(x)$ autour de x_c :

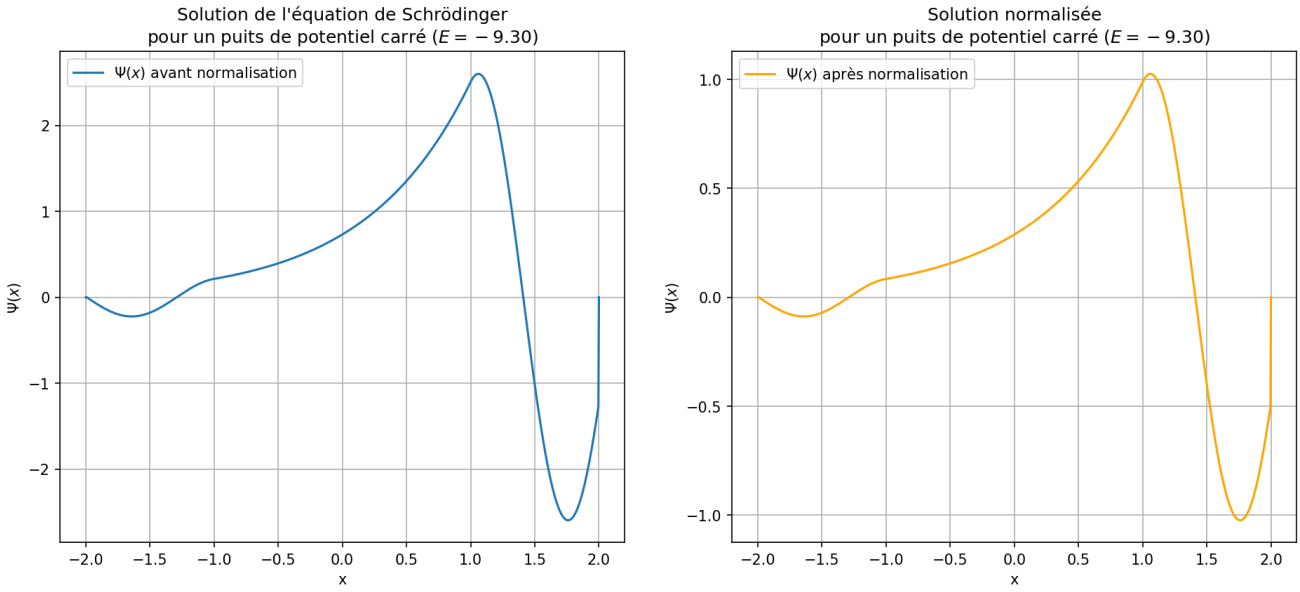

FIGURE 4 – Fonction d'onde pour les deux énergies.

$$\begin{aligned}\Phi_{i-1,L} &= \Phi_{i,L} - \Delta\Phi'_{i,L} + \frac{\Delta^2}{2}\Phi''_{i,L} + \mathcal{O}(\Delta^3)\\ \Phi_{i+1,R} &= \Phi_{i,R} + \Delta\Phi'_{i,R} + \frac{\Delta^2}{2}\Phi''_{i,R} + \mathcal{O}(\Delta^3)\end{aligned} \tag{13}$$

où i est l'indice tel que $x_i = x_c$. En ajoutant ces deux équations, il vient :

$$\Phi_{i-1,L} + \Phi_{i+1,R} = 2\Phi_i + \Delta\left(\Phi'_{i,R} - \Phi'_{i,L}\right) - \Delta^2 Q_i\Phi_i + \mathcal{O}(\Delta^3),$$

car $\Phi_{i,L} = \Phi_{i,R} = \Phi_i$ et $\Phi''_{i,L} = \Phi''_{i,R} = -Q_i\Phi_i$. Il vient alors :

$$\Phi'_{i,R} - \Phi'_{i,L} = \frac{\Phi_{i-1,L} + \Phi_{i+1,R} - 2\Phi_i + \Delta^2 Q_i\Phi_i}{\Delta} + \mathcal{O}(\Delta^2).$$

Pour calculer cette discontinuité en Python, vous pouvez écrire une fonction comme celle ci-dessous :

```
def derivative(E, x, V, a, V0, xc):
    # Calculer le potentiel V(x) et Q(x)
    Vx = potential(x, a, V0)
    Q = 2 * (E - Vx)

    # Trouver l'indice correspondant    xc
    i = np.abs(x - xc).argmin()

    # D finir les conditions initiales pour Numerov
    y0 = 0
    y1 = 1e-5  # un petit nombre pour  viter  la division par z ro

    # Calculer la fonction d'onde    partir de la gauche et de la droite
    yl = numerov(x[:i+1], Q[:i+1], y0, y1)
    yr = numerov(x[:i:-1], Q[:i:-1], y0, y1)

    # Normaliser la fonction d'onde    xc
    yr = yr * yl[-1] / yr[-1]

    # Calculer la discontinuit  de la d riv e
    dx = x[1] - x[0]
    discontinuity = (yr[1] - yl[-2] + dx**2 * Q[i] * yl[-1]) / (2*dx)
```

```
    # Calculer L(E, xc)
    L = discontinuity / yl[-1]

    return L
```

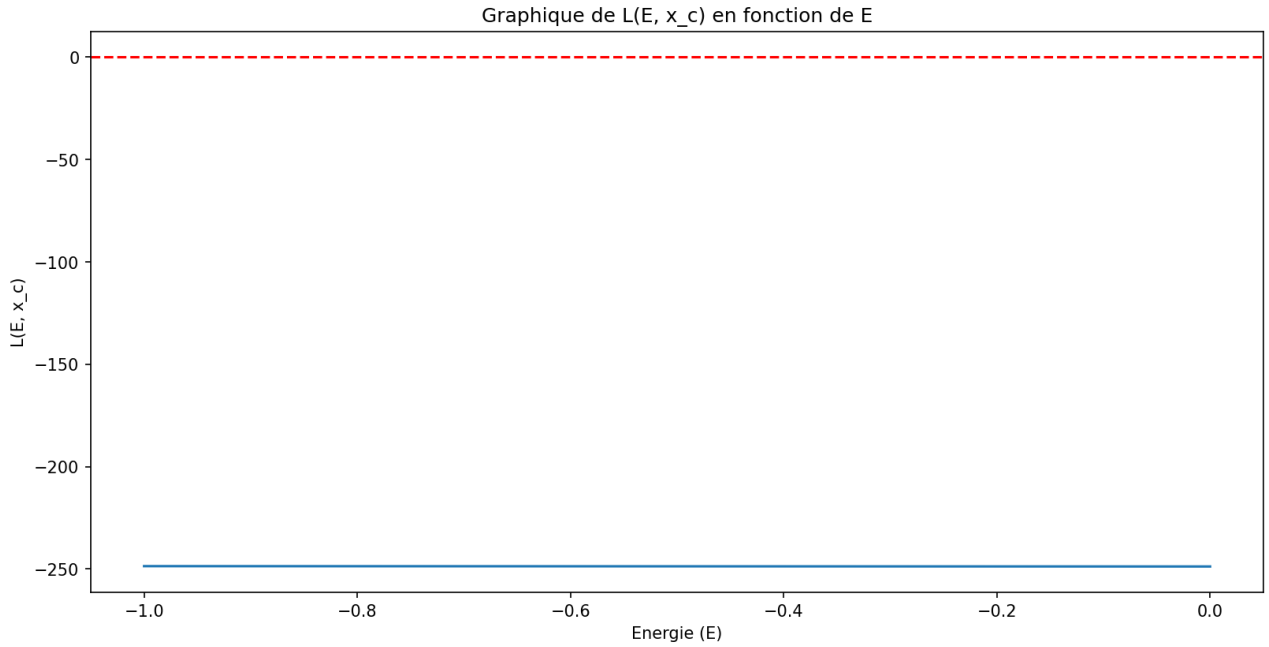

FIGURE 5 – L(E,x,c) en fonction de E (Erreur)

Le nombre de noeuds d'une fonction d'onde est étroitement lié à l'énergie de l'état quantique qu'elle décrit. En mécanique quantique, un noeud est un point où la fonction d'onde s'annule, c'est-à-dire où sa valeur est zéro.

Pour un puits de potentiel carré, les états d'énergie quantifiés peuvent être catégorisés en fonction du nombre de noeuds dans la fonction d'onde.

```
def eigen(x, V, Emin, Emax, delta, a, V0, xc):
    E = (Emax + Emin) / 2
    dE = Emax - Emin
    y = None

    while dE > delta:
        L = derivative(E, x, V, a, V0, xc)
        if L > 0:
            Emax = E
        else:
            Emin = E
        E = (Emax + Emin) / 2
        dE = Emax - Emin

    y = numerov(x, 2*(E-V), 0, 1e-5)
    y = normalize(y, x[1]-x[0])

    return E, y
```

5 Conclusion

Nous avons également discuté de l'importance de la normalisation des fonctions d'onde et fourni des directives pour calculer la discontinuité de la dérivée d'une fonction d'onde à un point donné. Ces étapes sont cruciales pour assurer l'exactitude des résultats obtenus en utilisant la méthode de Numerov.

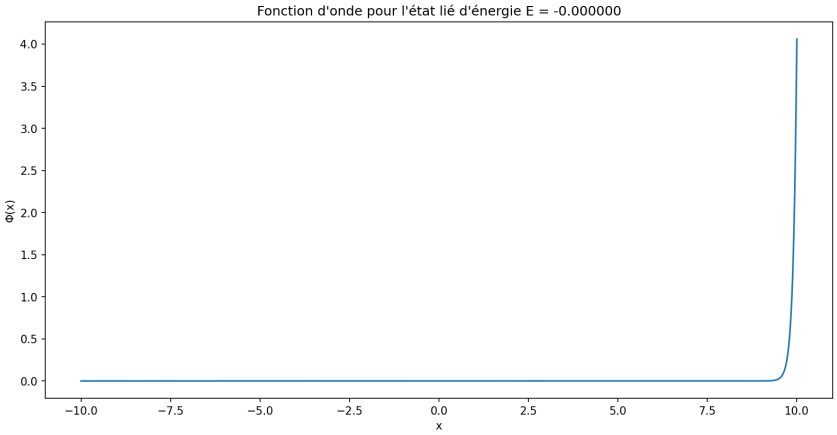

FIGURE 6 – Fonction d'onde pour l'état d'énergie E (Erreur)

En outre, nous avons exploré la relation entre le nombre de nœuds d'une fonction d'onde et l'énergie de l'état quantique qu'elle décrit, soulignant l'importance de cette relation dans la compréhension des phénomènes quantiques.

En conclusion, la méthode de Numerov se révèle être un outil puissant et fiable mais pas très évident à implémenter. Par ailleurs, les résulats trouvées aux long du TP ne sont pas très cohérentes avec les résultats analytiques trouvées par une solution analytique sur un puit infini, des erreurs on été comises dans les différents codes probablement. En fait il y a des problèmes de déphasages dans les méthodes numérov implémentés. Les résulats dévraient plutot ressembler à ceci :

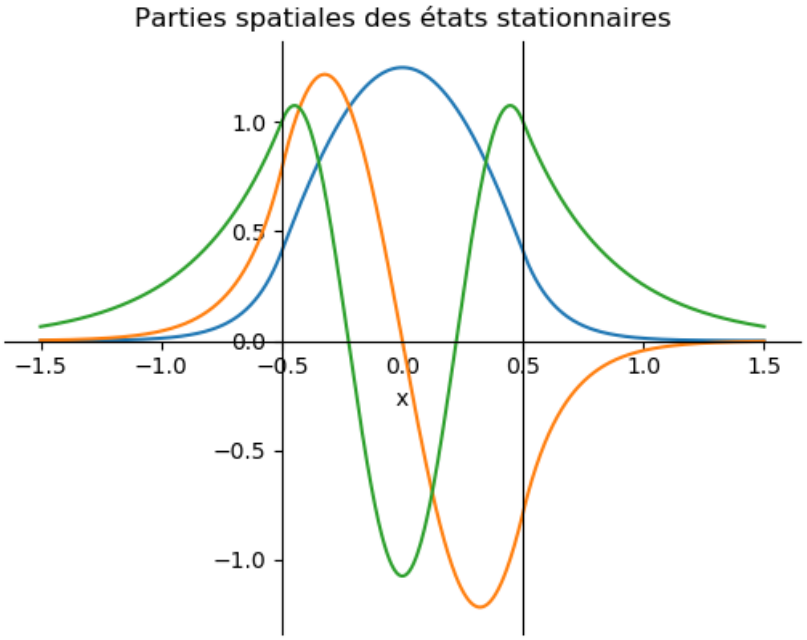

FIGURE 7 – Légende de votre image.