Travaux numériques de Physique Quantique

Thomas Darley

June 8, 2021

A Particule dans un puits triangulaire

A.1 Résolution analytique

On rapelle la forme générale de l'équation de Schrödinger 1D:

$$H\Psi_n(x,t) = E_n\Psi_n(x,t)$$

Ici, l'Hamiltonien du système est, $\forall x > 0$:

$$H = -\frac{\hbar^2}{2m}\frac{\partial^2}{\partial x^2} + \alpha x$$

On considère une solution sous la forme $\Psi(x,t) = \psi(x)g(t)$, où $g(t) = e^{i\frac{E_n}{\hbar}t}$. ψ vérifie alors, $\forall x > 0$:

$$\frac{\hbar^2}{2m}\frac{d^2\psi_n}{dx^2}(x) = (\alpha x - E_n)\psi_n(x) \qquad \textbf{(1)}$$

Effectuons le changement de variable indiqué : $z = \left(\frac{2m\alpha}{\hbar^2}\right)^{1/3}\left(x - \frac{E_n}{\alpha}\right)$. On a d'abord, d'après la règle de la chaîne :

$$\frac{d^2}{dx^2} = \left(\frac{dz}{dx}\right)^2\frac{d^2}{dz^2} = \left(\frac{2m\alpha}{\hbar^2}\right)^{2/3}\frac{d^2}{dz^2}$$

L'équation 1 devient alors :

$$\left(\frac{\hbar^2}{2m\alpha}\right)^{1-2/3}\frac{d^2\psi_n}{dz^2}(z) = \psi_n(z)z\left(\frac{\hbar^2}{2m\alpha}\right)^{1/3}$$

Ou encore :

$$\frac{d^2\psi_n}{dz^2}(z) = \psi_n(z)z \qquad \textbf{(2)}$$

On retrouve bien l'équation d'Airy souhaitée, dont les solutions sont de la forme :

$$\psi_n : z \mapsto C \cdot A_i(z)$$

Où $C \in \mathbf{R}$ une constante de normalisation et A_i est la fonction d'Airy. On visualise cette fonction sur l'intervalle $[-10, 5]$ grâce à Python **(A1)** :

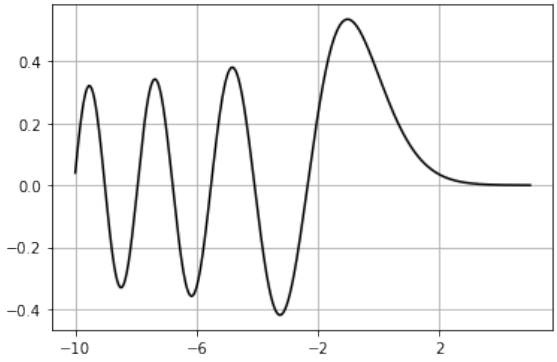

Figure 1: Fonction d'Airy

Ou, en remplaçant z par sa valeur,

$$\psi_n : x \mapsto C \cdot A_i\Big(\Big(\frac{2m\alpha}{\hbar^2}\Big)^{1/3}\Big(x - \frac{E_n}{\alpha}\Big)\Big) \qquad \textbf{(3)}$$

Vu que $\psi_n(0) = 0$, les énergies propres vérifient :

$$A_i\Big(-\Big(\frac{2m\alpha}{\hbar^2}\Big)^{1/3}\frac{E_n}{\alpha}\Big) = 0$$

En notant β_n le n-ième zéro de la fonction d'Airy, on a donc :

$$-\Big(\frac{2m\alpha}{\hbar^2}\Big)^{1/3}\frac{E_n}{\alpha} = \beta_n$$

Ou encore :

$$E_n = -\Big(\frac{\alpha^2\hbar^2}{2m}\Big)^{1/3}\beta_n$$

En utilisant l'approximation $\beta_n \approx -(\frac{3\pi(4n-1)}{8})^{2/3}$, on arrive aux valeurs d'énergies propres utilisées par la suite :

$$E_n = \Big(\frac{3\alpha\hbar\pi(4n-1)}{8\sqrt{2m}}\Big)^{2/3} \qquad \textbf{(4)}$$

A.2 Résolution numérique

On se place maintenant dans le cas où le potentiel triangulaire est lui-même dans un puits infini. Le puits aura une largeur a, et le potentiel affine atteindra V_0 en a, on a donc le nouveau Hamiltonien du système:

$$H^p = H + V_\infty$$

avec $V_\infty(x) = +\infty \ \forall x \in [-\infty, 0] \cup [a, \infty]$ et $V_\infty(x) = 0 \ \forall x \in [0, a]$.

On détermine alors numériquement H^p et les énergies propres normalisées (ε_n) par l'énergie fondamental du puits infini $E_1 = \frac{\pi^2\hbar^2}{2ma^2}$ (Annexe **(A2)**). On applique le code précédent et on trace le résultat pour $n = 0, 1, 2$ grâce au code **(A3)**.

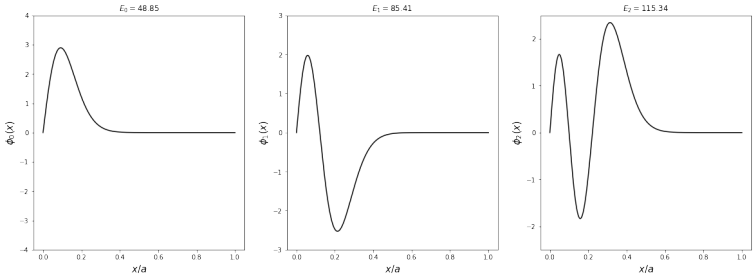

Figure 2: Trois premières fonctions propres du système et leurs énergies

On souhaite maintenant confronter cette simulation à la théorie. D'après **(4)**, on a (avec $\alpha = \frac{V_0}{a}$):

$$\varepsilon_n = \frac{E_n}{E_1} = \frac{\left(\frac{3V_0\hbar\pi(4n-1)}{8a\sqrt{2m}}\right)^{2/3}}{\frac{\pi^2\hbar^2}{2ma^2}} = \left(\frac{6mV_0a^2(4n-1)}{8\pi^2\hbar^2}\right)^{2/3} = \left(\frac{3}{8}v_0(4n-1)\right)^{2/3}$$

On peut également exprimer les fonctions d'onde en fonction de ε_n et $\frac{x}{a}$ d'après **(3)** :

$$\psi_n(x) = C \cdot A_i\left(\left(v_0\pi^2\right)^{1/3}\left(\frac{x}{a} - \frac{\varepsilon_n}{v_0}\right)\right)$$

On détermine C_n et on superpose la simulation à la théorie **(A4)** :

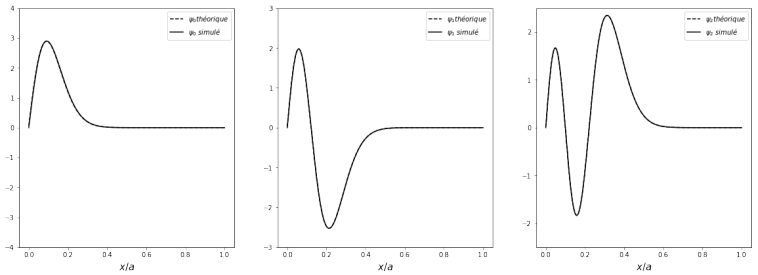

Figure 3: Comparaison des fonctions propres théoriques avec la simulation

On observe que la simulation colle quasi-parfaitement à la simulation effectuée avec $N = 50$.

Qu'en est-il des énergies ?

On peut également tracer les énergies normalisées en fonction de n. On obtient d'après la simulation, une courbe ayant deux tendances **(A5)** :

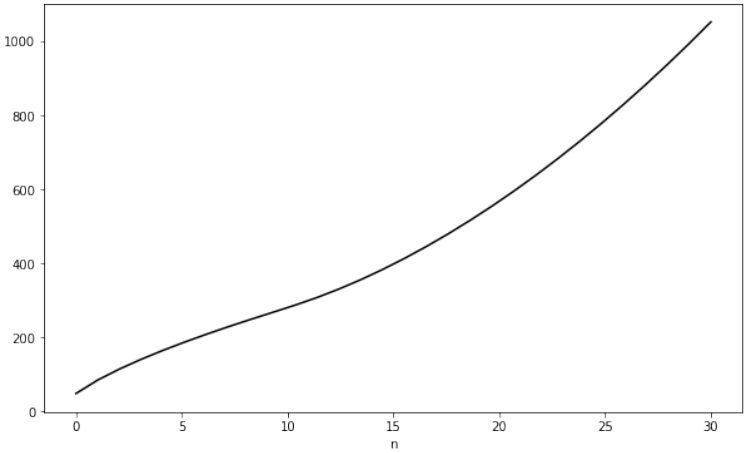

Figure 4: Les deux régimes de $\varepsilon(n)$

On peut fitter le régime des n petits par l'expression de $\varepsilon(n)$, proportionnelle à $n^{2/3}$. Pour les n grands, comme pour le puits infini, on a $\varepsilon(n) \propto n^2$. On obtient **(A6)** :

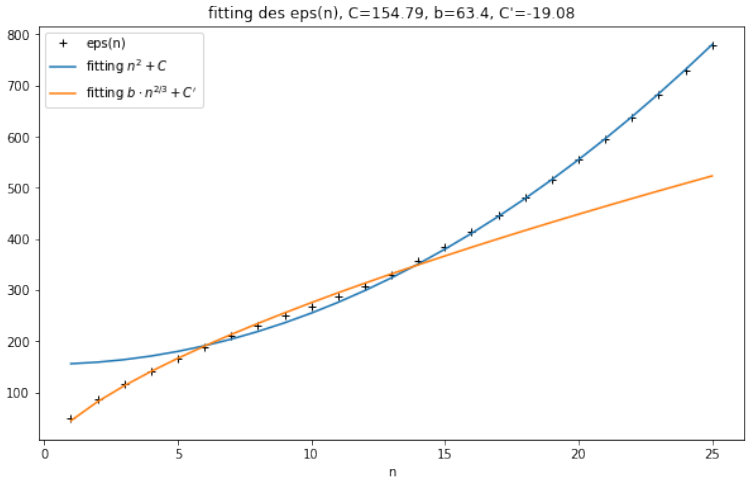

Figure 5: Fittings des régimes de $\varepsilon(n)$

On observe bien l'existence de 3 régimes, les deux précédents et un intermédiaire. On trace 3 fonctions d'onde aux ordres $(n = 3, 9, 20)$, **(A7)**:

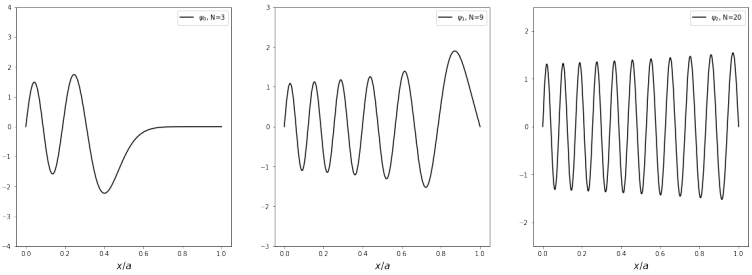

Figure 6: $\psi_2(x)$ pour $n = 3, 9, 20$

On observe que pour n faible, on a des diagrammes typiques du puits triangulaire et, plus n augmente, plus l'effet du potentiel triangulaire s'estompe et on retrouve les fonctions d'onde d'un puits infini.

On trace enfin les 5 premières densités de probabilité de présence, sur l'axe des énergies, rapellant la géométrie du puits. Le code **(A8)** et un peu de photoshop fournissent :

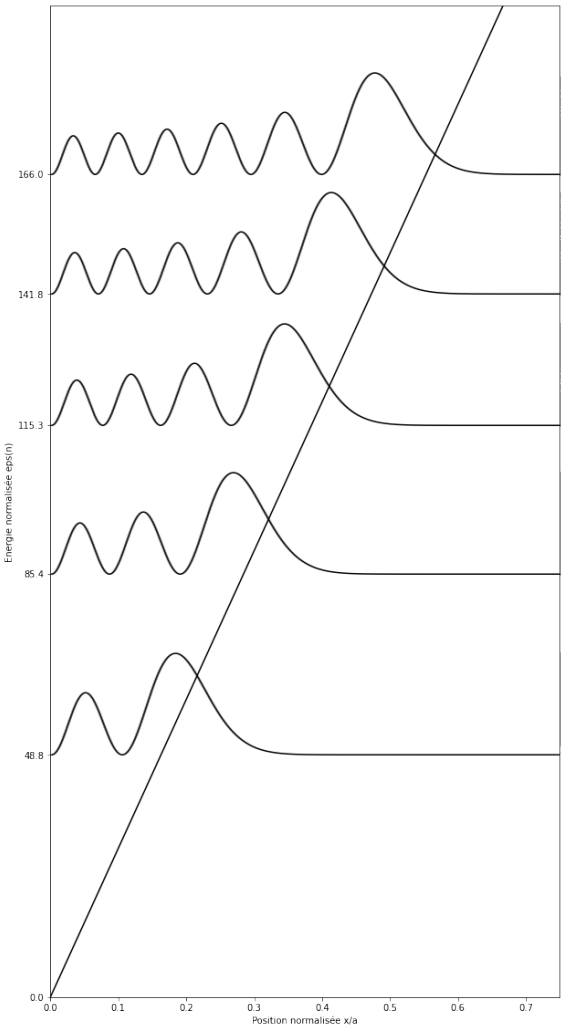

Figure 7: Probabilité $A|\psi_n(x)|^2$ pour les 5 premiers n

On observe bien que l'effet de la géométrie triangulaire du puits a de moins en moins d'impact plus n augmente. C'est ce qu'on observait sur la courbe des énergies.

B Résonance paramagnétique électronique dans le rubis

On étudie maintenant le phénomène de résonance dans un cristal. La théorie de l'énoncé fournit la forme de l'Hamiltonien dans $\vec{B_0}$ statique :

$$H = -\frac{2\pi D}{\hbar}(S_z^2 - \frac{S^2}{3}) + g\frac{eB_0}{2m}(S_z cos\theta + S_x sin\theta)$$

Où z est l'axe du cristal, $D = 5.73$ GHz et θ l'angle que fait le champ avec l'axe Oz.

Vu que S est fixé à $\frac{3}{2}$, et que m_s ne peut prendre que les valeurs demi-entières entre $-\frac{3}{2}$ et $\frac{3}{2}$, on a :

$$\left\{ |S, m_s\rangle \right\} = \left\{ |\frac{3}{2}, -\frac{3}{2}\rangle, |\frac{3}{2}, -\frac{1}{2}\rangle, |\frac{3}{2}, \frac{1}{2}\rangle, |\frac{3}{2}, \frac{3}{2}\rangle \right\}$$

On a, en outre, les effets suivants pour les opérateurs :

$\mathrm{S}_z |S, m_s\rangle = s\hbar |S, m_s\rangle$
$S^2 |S, m_s\rangle = s(s+1)\hbar^2 |S, m_s\rangle$

Malheureusement, S_x n'est pas valeur propre de $|S, m_s\rangle$, mais on peut passer par les opérateurs "ascenceurs", S_+ et S_- qui vérifient :

$\mathrm{S}_x = \frac{S_+ + S_-}{2}$
$S_+ |S, m_s\rangle = \hbar\sqrt{s(s+1) - m(m+1)} |S, m_s + 1\rangle$ (si possible, sinon = 0)
$S_- |S, m_s\rangle = \hbar\sqrt{s(s+1) - m(m-1)} |S, m_s - 1\rangle$ (si possible, sinon = 0)

Grâce à cela, on peut alors calculer tous les coefficients H_{ij} de la matrice H. On arrive, après calcul à (avec $\mu_B = \frac{e\hbar}{2m}$) :

$$\begin{pmatrix} -2\pi D\hbar - \frac{3}{2}g\mu_B B_0 cos\theta & \frac{\sqrt{3}}{2}g\mu_B B_0 sin\theta & 0 & 0 \\ \frac{\sqrt{3}}{2}g\mu_B B_0 sin\theta & 2\pi D\hbar - \frac{1}{2}g\mu_B B_0 cos\theta & g\mu_B B_0 sin\theta & 0 \\ 0 & g\mu_B B_0 sin\theta & 2\pi D\hbar + \frac{1}{2}g\mu_B B_0 cos\theta & \frac{\sqrt{3}}{2}g\mu_B B_0 sin\theta \\ 0 & 0 & \frac{\sqrt{3}}{2}g\mu_B B_0 sin\theta & -2\pi D\hbar + \frac{3}{2}g\mu_B B_0 cos\theta \end{pmatrix}$$

Valeurs propres

La matrice de l'Hamiltonien est tridiagonale mais pas diagonale. On propose le code **(B1)** qui calcule H, la diagonalise et plot les 4 énergies propres en fréquence, en fonction de θ et B.

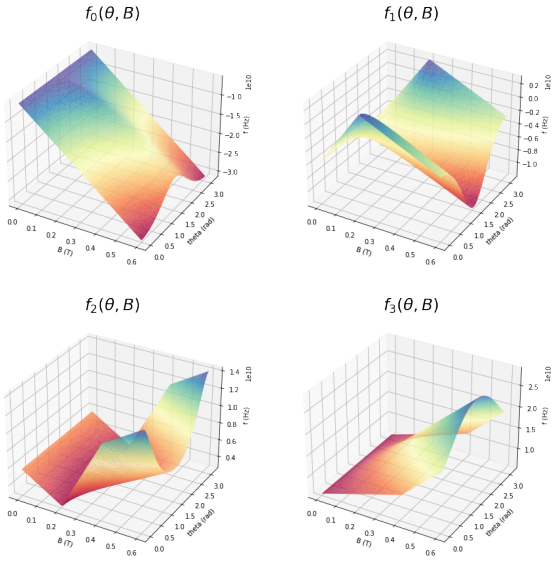

Figure 8: Allure de $f_n(\theta, B)$ pour $n = 0..3,\ B \in [0, 0.6]$ T, $\theta \in [0, \pi]$

On se place maintenant spécifiquement à $\theta = 50°$ (0.873 rad), on a alors **(B2)** :

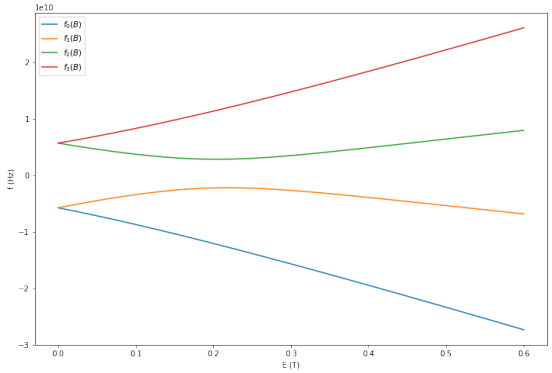

Figure 9: Allure de $f_n(50°, B)$ pour $n = 0..3,\ B \in [0, 0.6]$ T,

Vecteurs propres

A $B = 0$ T, la matrice H est diagonale, et on a trivialement les vecteurs propres suivants :

$$\psi_0 = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}, \psi_1 = \begin{bmatrix} 0 \\ 0 \\ 0 \\ 1 \end{bmatrix}, \psi_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ 0 \end{bmatrix}, \psi_3 = \begin{bmatrix} 0 \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

Pour $B = 0.6$ T, le code **(B3)** donne, arrondi au centième :

$$\psi_0 = \begin{bmatrix} 0.89 \\ 0.39 \\ -0.23 \\ -0.09 \end{bmatrix}, \psi_1 = \begin{bmatrix} -0.43 \\ 0.53 \\ -0.61 \\ -0.41 \end{bmatrix}, \psi_2 = \begin{bmatrix} 0.16 \\ -0.63 \\ -0.17 \\ -0.74 \end{bmatrix}, \psi_3 = \begin{bmatrix} -0.05 \\ 0.41 \\ 0.75 \\ -0.53 \end{bmatrix}$$

La fréquence de résonance f_r de la cavité vaut 9.19 GHz. Quand B est dans l'intervalle [0,0.6] T, toutes les fréquences propres passent par f_r à 50° d'après la Fig.9. On en déduit que 4 transitions seront visibles sur le graphe RPE. C'est bien ce qu'on obtient par la simulation (code **(B4)**) :

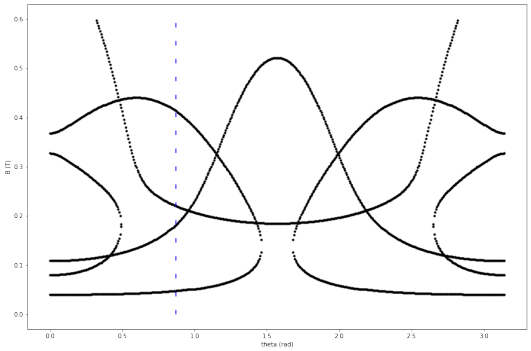

Figure 10: Spectre RPE simulé, verticale à $\theta = 50°$

On lit ce diagramme ainsi, à θ fixé, un trait vertical informe sur le nombre de transitions visibles, et à quels B les trouver. A B fixé, un trait horizontal informe sur le nombre de transitions visibles, et à quels θ les trouver.

Le trait vertical intersecte 4 fois les courbes, on observe bien 4 transitions.

On superpose enfin les résultats théoriques de 2018 avec la simulation. On obtient **(B5)** le graph :

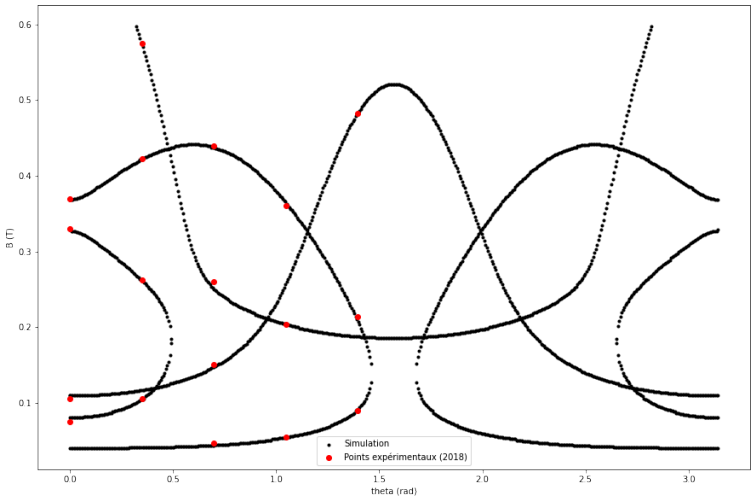

Figure 11: Simulation et expérience

Les résultats expérimentaux coïncident parfaitement avec la simulation, les étudiants de 2018 ont bien travaillé.

Annexes

Partie A

A1

```
#A1
from scipy import special
import numpy as np
import matplotlib.pyplot as plt

x=np.linspace(-10,5,10000)
plt.plot(x,special.airy(x)[0],'k')
plt.grid()
plt.xticks([-10,-6,-2,2])
```

A2

```
#A2
import numpy.linalg as alg
import matplotlib.pyplot as plt
import scipy
import numpy as np
from scipy.constants import hbar
from scipy.integrate import nquad

global a,v0,N,dx,m
m=1
a=1
v0=300
N=int(1e2)
dx=1e-6

def Psi(N,n_od): #n_od:ordre de l' tat  excit
    E_n= lambda n : (n*np.pi*hbar/a)**2/(2*m)
    v_tri= lambda x : v0/a*x
    delta= lambda n,m : int(n==m)
    intg= lambda n,m : nquad(lambda x : v_tri(x)*np.sin(n*np.pi*x)*
    np.sin(m*np.pi*x),[[0,1]],opts=[{'limit':100}])
    H_nm= lambda n,m : n**2*delta(n,m)+2*intg(n,m)[0]

    H=np.zeros((N,N))
    for i in range(N):
        for j in range(N):
            H[i,j]=H_nm(i+1,j+1)

    phi= lambda n,x : np.sqrt(2/a)*np.sin(n*np.pi*x/a)

    [spectre,vecs_p]=alg.eig(H)
    #tri des vecteurs propres
    ordre=spectre.argsort()
    vecs_p=vecs_p[:,ordre]
    spectre=spectre[ordre]

    vec=vecs_p[:,n_od]
    e=spectre[n_od]

```

```
    def f(x):
        f_=0
        for i in range(N):
            f_+=vec[i]*phi(i+1,x)
        return f_

    return f,e
```

A3

```
#A3
vec_x=np.linspace(0,a,int(1/dx))
fig,ax=plt.subplots(1,3,figsize=(20,6.66))

psi_0,e_0=Psi(N,0)
ax[0].plot(vec_x,psi_0(vec_x),'k')
ax[0].set_ylim([-4,4])
ax[0].set_ylabel('$\phi_0(x)$',fontsize=15)
ax[0].set_xlabel('$x/a$',fontsize=15)
ax[0].set_title('$E_0=$'+str(round(e_0,2)))

psi_1,e_1=Psi(N,1)
ax[1].plot(vec_x,-psi_1(vec_x),'k')
ax[1].set_ylim([-3,3])
ax[1].set_ylabel('$\phi_1(x)$',fontsize=15)
ax[1].set_xlabel('$x/a$',fontsize=15)
ax[1].set_title('$E_1=$'+str(round(e_1,2)))

psi_2,e_2=Psi(N,2)
ax[2].plot(vec_x,-psi_2(vec_x),'k')
ax[2].set_ylim([-2.5,2.5])
ax[2].set_ylabel('$\phi_2(x)$',fontsize=15)
ax[2].set_xlabel('$x/a$',fontsize=15)
ax[2].set_title('$E_2=$'+str(round(e_2,2)))
```

A4

```
#A4
epsi = lambda n :(3*v0*(4*n-1)/(8))**(2/3)
psi_th = lambda n,x : special.airy((v0*np.pi**2)**(1/3)*(x/a-epsi(n
    )/v0))[0]

C_0=max(abs(psi_0(vec_x)))/max(abs(psi_th(1,vec_x)))
C_1=max(abs(psi_1(vec_x)))/max(abs(psi_th(2,vec_x)))
C_2=max(abs(psi_2(vec_x)))/max(abs(psi_th(3,vec_x)))

vec_x=np.linspace(0,1,int(1/dx))
fig,ax=plt.subplots(1,3,figsize=(20,6.66))

ax[0].plot(vec_x,C_0*psi_th(1,vec_x),'k--',label='$\psi_0
    th orique$')
ax[1].plot(vec_x,-C_1*psi_th(2,vec_x),'k--',label='$\psi_1
    th orique$')
ax[2].plot(vec_x,C_2*psi_th(3,vec_x),'k--',label='$\psi_2
    th orique$')

ax[0].plot(vec_x,-psi_0(vec_x),'k-',label='$\psi_0 \ simul $')
ax[0].set_ylim([-4,4])
ax[0].set_xlabel('$x/a$',fontsize=15)
```

```

ax[1].plot(vec_x,-psi_1(vec_x),'k-',label='$\psi_1 \ simul $')
ax[1].set_ylim([-3,3])
ax[1].set_xlabel('$x/a$',fontsize=15)


ax[2].plot(vec_x,psi_2(vec_x),'k-',label='$\psi_2 \ simul $')
ax[2].set_ylim([-2.5,2.5])
ax[2].set_xlabel('$x/a$',fontsize=15)

ax[0].legend()
ax[1].legend()
ax[2].legend()
```

A5

```
#A5

N=75
n_od=0
E_n= lambda n : (n*np.pi*hbar/a)**2/(2*m)
v_tri= lambda x : v0/a*x
delta= lambda n,m : int(n==m)
intg= lambda n,m : nquad(lambda x : v_tri(x)*np.sin(n*np.pi*x)*np.
    sin(m*np.pi*x),[[0,1]],opts=[{'limit':100}])
H_nm= lambda n,m : n**2*delta(n,m)+2*intg(n,m)[0]

H=np.zeros((N,N))
for i in range(N):
    for j in range(N):
        H[i,j]=H_nm(i+1,j+1)

phi = lambda n,x : np.sqrt(2/a)*np.sin(n*np.pi*x/a)

[spectre,vecs_p]=alg.eig(H)
#tri des vecteurs propres
ordre=spectre.argsort()
vecs_p=vecs_p[:,ordre]
spectre=spectre[ordre]

plt.figure(figsize=[10,6])
plt.xlabel("n")
plt.plot(np.linspace(0,30,30),spectre[:30],'k-')
```

A6

```
#A6
from scipy.optimize import curve_fit

plt.figure(figsize=[10,6])

vec_n=np.linspace(1,25,25)
C=curve_fit(lambda n,C:n**2+C,vec_n[15:25],spectre[15:25])[0][0]
fit_h=lambda n:n**2+C
vec_fit_high=[fit_h(n+1) for n in range(0,25)]
plt.plot(vec_n,spectre[:25],'k+',label='eps(n)')
plt.plot(vec_n,vec_fit_high,label="fitting $n^2+C$")

```

```
b,cp=curve_fit(lambda n,b,C:b*n**(2/3)+C,vec_n[0:15],spectre[0:15])
    [0]
fit_l=lambda n:b*n**(2/3)+cp
vec_fit_low=[fit_l(n+1) for n in range(0,25)]
plt.plot(vec_n,vec_fit_low,label="fitting $b \cdot n^{2/3}+C'$")

plt.xlabel('n')
plt.legend()
plt.title('fitting des eps(n), C='+str(round(C,2))+', b='+str(round
    (b,2))+", C'="+str(round(cp,2)))
```

A7

```
#A7

vec_x=np.linspace(0,1,int(1/dx))
fig,ax=plt.subplots(1,3,figsize=(20,6.66))

ax[0].plot(vec_x,Psi(50,3)[0](vec_x),'k-',label='$\psi_0$, n=3')
ax[0].set_ylim([-4,4])
ax[0].set_xlabel('$x/a$',fontsize=15)


ax[1].plot(vec_x,Psi(50,10)[0](vec_x),'k-',label='$\psi_1$, n=9')
ax[1].set_ylim([-3,3])
ax[1].set_xlabel('$x/a$',fontsize=15)


ax[2].plot(vec_x,Psi(50,20)[0](vec_x),'k-',label='$\psi_2$, n=20')
ax[2].set_ylim([-2.5,2.5])
ax[2].set_xlabel('$x/a$',fontsize=15)

ax[0].legend()
ax[1].legend()
ax[2].legend()
```

A8

```
#A8.1

plt.figure(figsize=[10,20])
plt.yticks([0]+list(spectre[0:5]))
plt.plot(vec_x,v0/a*vec_x,'k')
plt.xlabel('Position normalis e x/a')
plt.ylabel('Energie normalis e eps(n)')
plt.ylim([0,200])
plt.xlim([0,0.75])

#A8.2
global A
A=1

fig,ax=plt.subplots(figsize=[10,2.25])
fig.patch.set_visible(False)
ax.axis('off')

n_plt=5

plt.plot(vec_x,A*abs(Psi(50,n_plt)[0](vec_x))**2,'k')
```

Partie B

B1

```
#B1
import numpy as np
from scipy.constants import hbar
import numpy.linalg as alg
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D

D=5.73E9
g=2
mu=9.274009994E-24

def H(B,theta):
    H=np.zeros([4,4])
    A=2*np.pi*D*hbar
    C=g*mu*B
    H[0,0]=-A-1.5*C*np.cos(theta)
    H[1,1]=A-.5*C*np.cos(theta)
    H[2,2]=A+.5*C*np.cos(theta)
    H[3,3]=-A+1.5*C*np.cos(theta)

    H[0,1]=np.sqrt(3)/2*C*np.sin(theta)
    H[1,0]=np.sqrt(3)/2*C*np.sin(theta)

    H[2,1]=C*np.sin(theta)
    H[1,2]=C*np.sin(theta)

    H[3,2]=np.sqrt(3)/2*C*np.sin(theta)
    H[2,3]=np.sqrt(3)/2*C*np.sin(theta)
    return H


def vp_H(B,theta):
    H_=H(B,theta)
    [spectre,vecs_p]=alg.eig(H_)
    ordre=spectre.argsort()
    vecs_p=vecs_p[:,ordre]
    spectre=spectre[ordre]
    return spectre

n_=100
theta_vec=np.linspace(0,np.pi,n_)
B_vec=np.linspace(0,0.6,n_)

fig = plt.figure(figsize=[15,15])

ax0 = fig.add_subplot(2,2,1,projection='3d')

data0=[]
for b_item in B_vec:
    for theta_item in theta_vec:
        data0.append([theta_item,b_item,vp_H(b_item,theta_item)
    [0]])

data0=np.array(data0) #theta,b,e
```

```
54
55
56 ax0.plot_trisurf(data0[:,1],data0[:,0],data0[:,2]/(2*np.pi*hbar),
       cmap=plt.cm.Spectral)
57 ax0.set_ylabel('theta (rad)')
58 ax0.set_xlabel('B (T)')
59 ax0.set_zlabel('f (Hz)')
60 ax0.set_title(r'$f_0(\theta,B)$',fontsize=25)
61
62
63
64 ax1 = fig.add_subplot(2,2,2,projection='3d')
65
66 data1=[]
67 for b_item in B_vec:
68     for theta_item in theta_vec:
69         data1.append([theta_item,b_item,vp_H(b_item,theta_item)
       [1]])
70
71 data1=np.array(data1) #theta,b,e
72
73
74 ax1.plot_trisurf(data1[:,1],data1[:,0],data1[:,2]/(2*np.pi*hbar),
       cmap=plt.cm.Spectral)
75 ax1.set_ylabel('theta (rad)')
76 ax1.set_xlabel('B (T)')
77 ax1.set_zlabel('f (Hz)')
78 ax1.set_title(r'$f_1(\theta,B)$',fontsize=25)
79
80
81
82 ax2 = fig.add_subplot(2,2,3,projection='3d')
83
84 data2=[]
85 for b_item in B_vec:
86     for theta_item in theta_vec:
87         data2.append([theta_item,b_item,vp_H(b_item,theta_item)
       [2]])
88
89 data2=np.array(data2) #theta,b,e
90
91
92 ax2.plot_trisurf(data2[:,1],data2[:,0],data2[:,2]/(2*np.pi*hbar),
       cmap=plt.cm.Spectral)
93 ax2.set_ylabel('theta (rad)')
94 ax2.set_xlabel('B (T)')
95 ax2.set_zlabel('f (Hz)')
96 ax2.set_title(r'$f_2(\theta,B)$',fontsize=25)
97
98
99
100 ax3 = fig.add_subplot(2,2,4,projection='3d')
101
102 data3=[]
103 for b_item in B_vec:
104     for theta_item in theta_vec:
```

```
        data3.append([theta_item,b_item,vp_H(b_item,theta_item)[3]])

data3=np.array(data3) #theta,b,e


ax3.plot_trisurf(data3[:,1],data3[:,0],data3[:,2]/(2*np.pi*hbar),cmap=plt.cm.Spectral)
ax3.set_ylabel('theta (rad)')
ax3.set_xlabel('B (T)')
ax3.set_zlabel('f (Hz)')
ax3.set_title(r'$f_3(\theta,B)$',fontsize=25)



plt.show()
```

B2

```
#B2
theta0=50*np.pi/180
B_vec2=np.linspace(0,0.6,1000)
E=[]
for i in range(4):
    e=[]
    for b_item in B_vec2:
        e.append(vp_H(b_item,theta0)[i])
    E.append(e)
E=np.array(E)
plt.figure(figsize=[12,8])
plt.plot(B_vec2,E[0]/(2*np.pi*hbar),label='$f_0(B)$')
plt.plot(B_vec2,E[1]/(2*np.pi*hbar),label='$f_1(B)$')
plt.plot(B_vec2,E[2]/(2*np.pi*hbar),label='$f_2(B)$')
plt.plot(B_vec2,E[3]/(2*np.pi*hbar),label='$f_3(B)$')
plt.legend()
plt.xlabel('E (T)')
plt.ylabel('f (Hz)')
```

B3

```
#B3
def vecp_H(B,theta,n):
    H_=H(B,theta)
    [spectre,vecs_p]=alg.eig(H_)
    ordre=spectre.argsort()
    vecs_p=vecs_p[:,ordre]
    return vecs_p[n]
print(np.round(vecp_H(0.6,theta0,0),2))
print(np.round(vecp_H(0.6,theta0,1),2))
print(np.round(vecp_H(0.6,theta0,2),2))
print(np.round(vecp_H(0.6,theta0,3),2))
```

B4

```
#B4

global n_p
n_p=500

```

```
def dE_list(theta,n):#n entre 0 et 2
    B=np.linspace(0,0.6,n_p)
    dE_vec=[]
    for B_item in B:
        energies=vp_H(B_item,theta)
        dE_vec.append(energies[n+1]-energies[n])
    return np.array(dE_vec)/(2*np.pi*hbar)

def dE2_list(theta,n):#n entre 0 et 1
    B=np.linspace(0,0.6,n_p)
    dE_vec=[]
    for B_item in B:
        energies=vp_H(B_item,theta)
        dE_vec.append(energies[n+2]-energies[n])
    return np.array(dE_vec)/(2*np.pi*hbar)

def dE3_list(theta):#n=0
    B=np.linspace(0,0.6,n_p)
    dE_vec=[]
    for B_item in B:
        energies=vp_H(B_item,theta)
        dE_vec.append(energies[3]-energies[0])
    return np.array(dE_vec)/(2*np.pi*hbar)

def transition_list(theta):
    B=np.linspace(0,0.6,n_p)
    fr=9.19E9
    trans=[]
    for n in range(3): #transition entre niveaux adjacents

        dE=dE_list(theta,n)
        for i in range(n_p-1):
            if (dE[i]>fr and dE[i+1]<fr) or (dE[i]<fr and dE[i+1]>
    fr):
                trans.append(B[i])

    for n in range(2): #transition 0-2,1-3
        dE2=dE2_list(theta,n)
        for i in range(n_p-1):
            if (dE2[i]>fr and dE2[i+1]<fr) or (dE2[i]<fr and dE2[i
    +1]>fr):
                trans.append(B[i])

    dE3=dE3_list(theta) #transition 0-3
    for i in range(n_p-1):
        if (dE3[i]>fr and dE3[i+1]<fr) or (dE3[i]<fr and dE3[i+1]>
    fr):
            trans.append(B[i])
    return trans

plt.figure(figsize=[15,10])
theta_vec=np.linspace(0,np.pi,n_p)
for theta in theta_vec:
    for trans in transition_list(theta):
        plt.scatter(theta,trans,marker=".",color='k')
plt.vlines(50*np.pi/180,0,0.6,colors='b',linestyles=(0,(5,15)))
plt.ylabel('B (T)')
```

```
plt.xlabel('theta (rad)')
plt.show()
```

B5

```
#B5

Bexp=np.array
    ([750,1050,3700,3300,1050,2620,4220,5750,460,1500,2600,4400,550,2040,3600,900,2140,4820])
    *1E-4
thetaexp=np.array
    ([0,0,0,0,20,20,20,20,40,40,40,40,60,60,60,80,80,80])*np.pi/180

plt.figure(figsize=[15,10])
theta_vec=np.linspace(0,np.pi,n_p)
for theta in theta_vec:
    for trans in transition_list(theta):
        plt.scatter(theta,trans,marker=".",color='k')

plt.scatter(theta_vec[0],transition_list(theta_vec[0])[0],marker="."
    ",color='k',label='Simulation')

plt.scatter(thetaexp,Bexp,color='r',label='Points exp rimentaux
    (2018)')
plt.ylabel('B (T)')
plt.xlabel('theta (rad)')
plt.legend()
plt.show()
```