Particule confinée à une dimension

1. Introduction

Ce document montre comment obtenir les états stationnaires pour une particule confinée dans un potentiel unidimensionnel $V(x)$. On considère l'équation de Schrödinger pour la fonction d'onde $\psi(x,t)$ de la particule

$$i\hbar\frac{\partial\psi(x,t)}{\partial t} = -\frac{\hbar^2}{2m}\frac{\partial^2\psi(x,t)}{\partial x^2} + V(x)\psi(x,t) \tag{1}$$

On recherche les états stationnaires d'énergie E, définis par :

$$\psi(x,t) = e^{-i\frac{E}{\hbar}t}\phi(x) \tag{2}$$

On doit alors résoudre l'équation de Schrödinger indépendante du temps :

$$-\frac{\hbar^2}{2m}\frac{d^2\phi(x)}{dx^2} + V(x)\phi(x) = E\phi(x) \tag{3}$$

La fonction d'onde recherchée est donc une fonction propre de l'opérateur hamiltonien défini par :

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

Les énergies E possibles sont des valeurs propres de cet opérateur.

2. Méthode numérique

On pose $m = 1$ et $\hbar = 1$.

On suppose que la particule reste confinée dans un intervalle $[a,b]$. La fonction d'onde est donc nulle en dehors de cet intervalle, et sur les bornes a et b. On considère un échantillonnage régulier de N points x_n sur cet intervalle. On note Δx le pas de x de cette subdivision.

La discrétisation de la dérivée seconde par différences finies donne :

$$\frac{2\phi_n - \phi_{n-1} - \phi_{n+1}}{2(\Delta x)^2} + V_n\phi_n = E\phi_n \tag{5}$$

Sur le bord $x = a$, correspondant à l'indice $n = 0$, on applique la condition limite $\varphi_{-1} = 0$. De même, sur le bord $x = b$, on applique $\varphi_N = 0$. On pose :

$$\alpha = \frac{1}{(\Delta x)^2} \tag{6}$$

L'opérateur hamiltonien appliqué à la fonction d'onde discrétisée est :

$$H = \begin{pmatrix} \alpha + V_0 & -\frac{\alpha}{2} & & & \\ -\frac{\alpha}{2} & \alpha + V_1 & -\frac{\alpha}{2} & & \\ & -\frac{\alpha}{2} & \alpha + V_2 & -\frac{\alpha}{2} & \\ & & & & \\ & & & -\frac{\alpha}{2} & \alpha + V_{N-1} \end{pmatrix} \tag{7}$$

Il s'agit d'une matrice tridiagonale symétrique.

En notant Φ la matrice colonne contenant les échantillons $\phi_{;n}$, on a finalement :

$$H\Phi = E\Phi \tag{8}$$

Les énergies sont donc les valeurs propres de la matrice H. Les solutions stationnaires sont les vecteurs propres correspondants.

3. Fonction python

La fonction suivante prend en argument les échantillons de x et du potentiel V, et le nombre de valeurs propres souhaités (les énergies les plus basses). Elle renvoit les énergies et les vecteurs propres correspondants. Les fonctions utilisées sont disponibles avec scipy (version 0.11 ou plus) compilé avec le support de BLAS et LAPACK.

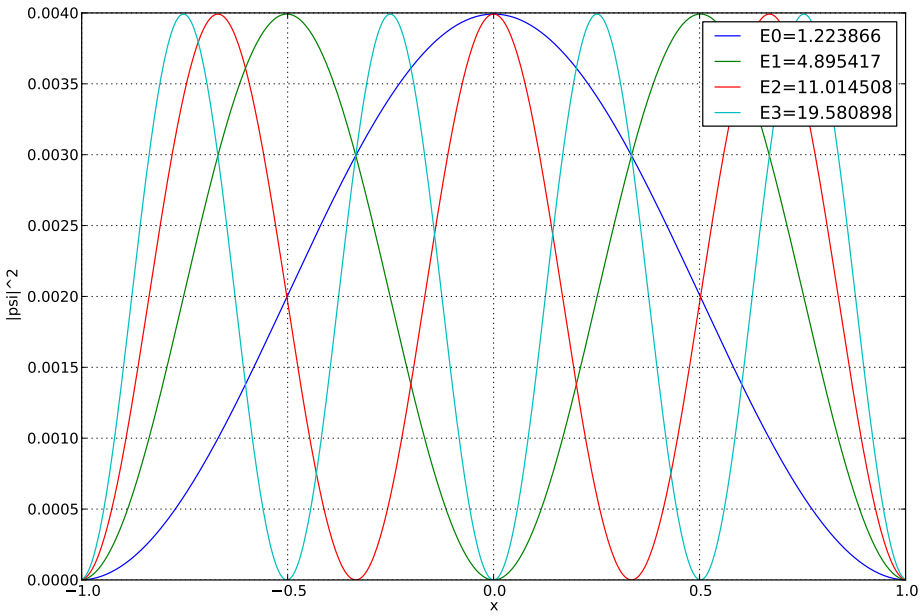
```
import numpy
import scipy.sparse as sparse
import scipy.sparse.linalg as linalg
from matplotlib.pyplot import *

def etats(x,V,nE,tol=0):
    N = V.size
    if x.size!=N:
        raise Exception("tailles de x et V incompatibles")
    alpha = 1.0/(x[1]-x[0])**2
    H = sparse.diags([-alpha/2,V+alpha,-alpha/2],[-1,0,1],shape=(N,N))
    E,phi = linalg.eigs(H,nE,which='SM',tol=tol)
    return numpy.real(E),phi
```

4. Exemples

4.a. Particule dans un puits infini carré

Le potentiel est nul sur tout l'intervalle.

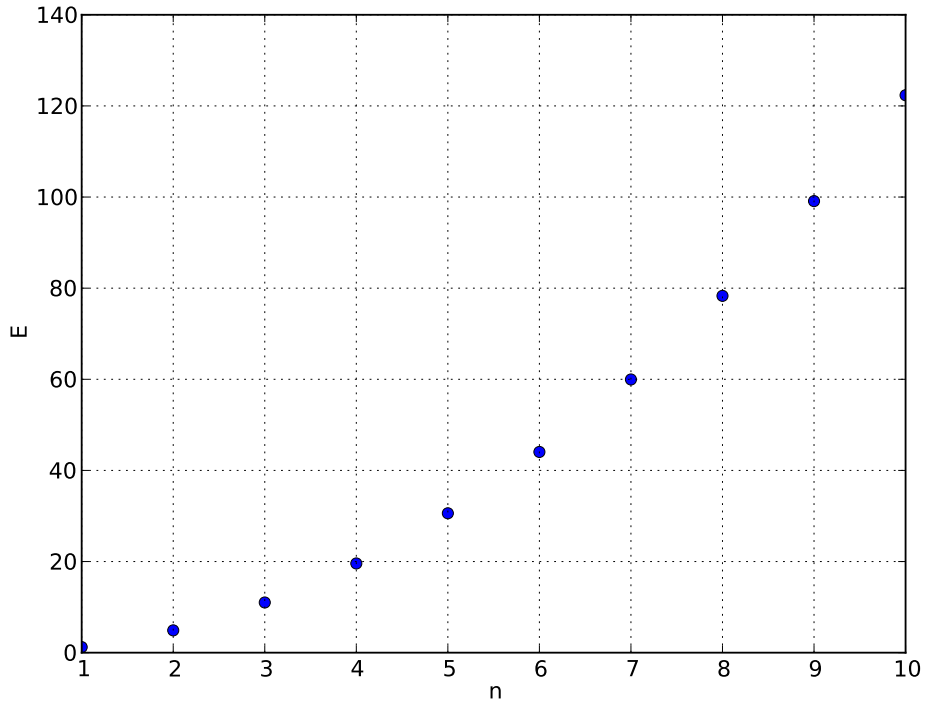
```
x = numpy.linspace(-1.0,1.0,500)
V = numpy.zeros(x.size)
E,phi = etats(x,V,4)
figure(figsize=(12,8))
for i in range(E.size):
    plot(x,numpy.square(numpy.absolute(phi[:,i])),label="E%d=%f"%(i,E[i]))
xlabel("x")
ylabel("|psi|^2")
grid()
legend(loc='upper right')
```



On retrouve les états stationnaires pour une particule dans une boîte, identiques aux modes propres d'une corde vibrante (ou d'une cavité).

On trace aussi la répartition des énergies :

```
nE = 10
E,phi = etats(x,V,nE)
figure()
plot(range(1,nE+1),E,'o')
xlabel("n")
ylabel("E")
grid()
```



Les énergies sont réparties en n^2 :

$$E_n = n^2 \frac{\hbar^2 \pi^2}{2mL^2} = n^2 \frac{\pi^2}{8} = 1{,}23\, n^2 \tag{9}$$

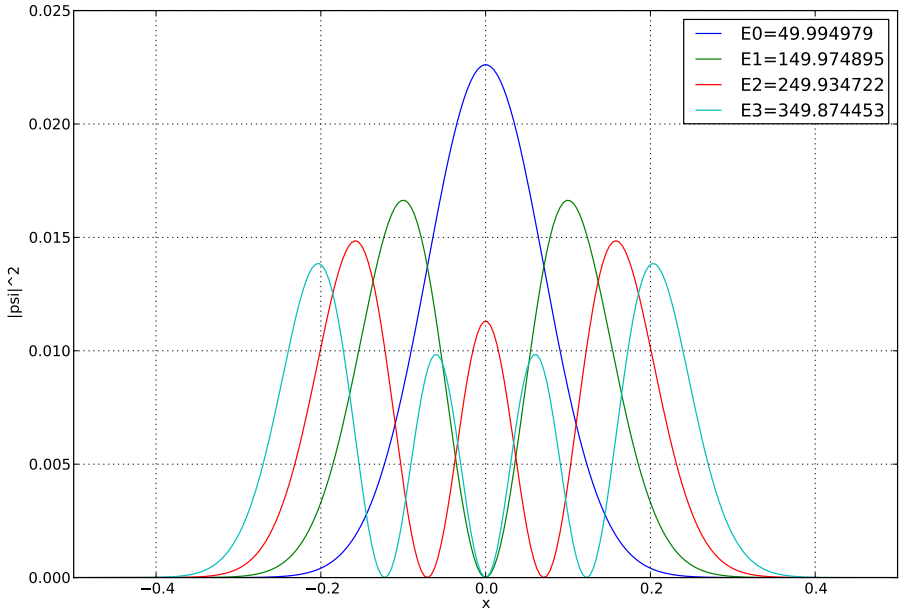
4.b. Oscillateur harmonique

Le potentiel d'un oscillateur harmonique est :

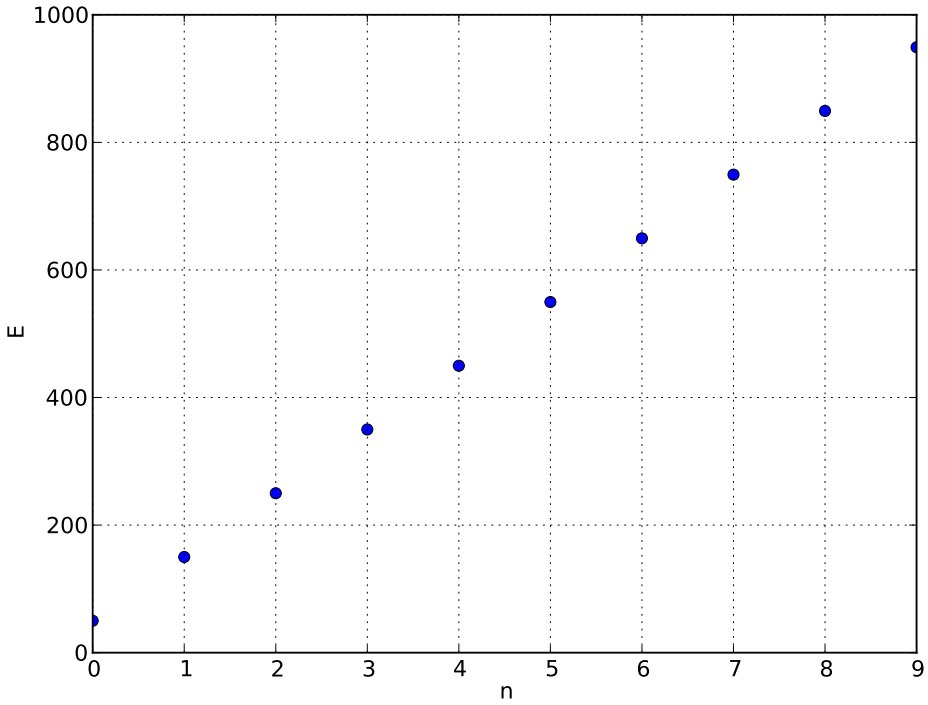
$$V(x) = \frac{1}{2} m\omega^2 x^2 \tag{10}$$

On se place sur l'intervalle $[-1, 1]$. En $x = -1$ et $x = 1$ la fonction d'onde est fixée à zéro, ce qui revient à prendre un potentiel infini en ces points. La constante ω doit être choisie assez grande pour que les états obtenus aient leur énergie dans la partie harmonique du potentiel.

```
omega = 100
x = numpy.linspace(-1.0,1.0,500)
V = x*x*0.5*omega**2
E,phi = etats(x,V,4)
figure(figsize=(12,8))
for i in range(E.size):
    plot(x,numpy.square(numpy.absolute(phi[:,i])),label="E%d=%f"%(i,E[i]))
xlabel("x")
ylabel("|psi|^2")
grid()
legend(loc='upper right')
axis([-0.5,0.5,0,0.025])
```




```
E,phi = etats(x,V,10)
figure()
plot(E,'o')
xlabel("n")
ylabel("E")
grid()
```

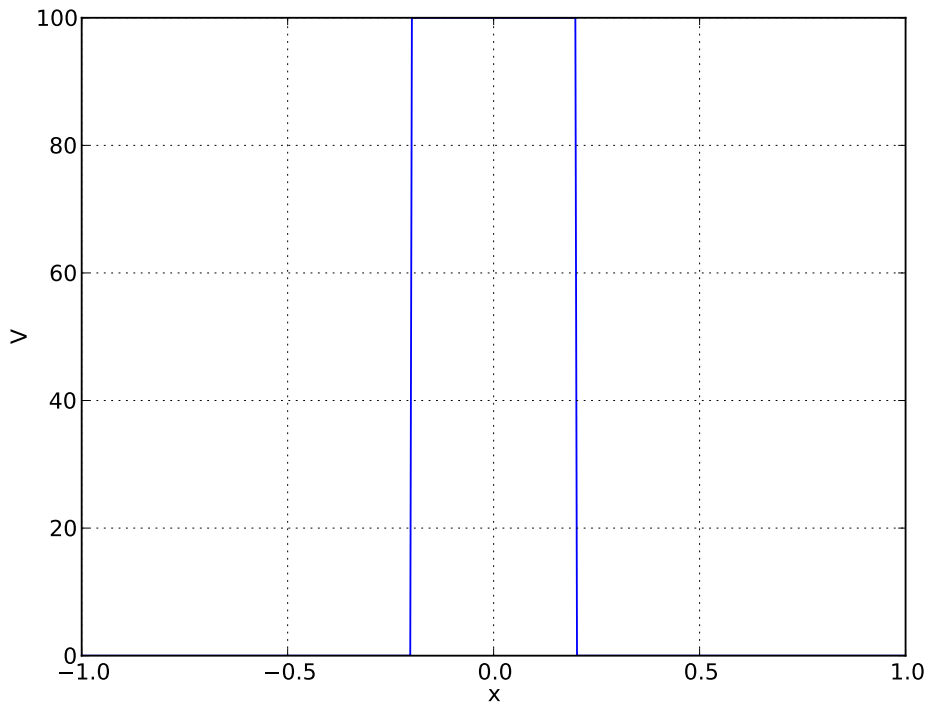


Les niveaux d'énergie sont régulièrement espacés de ω. On retrouve la répartition des énergies de l'oscillateur harmonique :

$$E_n = (n + \frac{1}{2})\hbar\omega \tag{11}$$

4.c. Puit de potentiel double

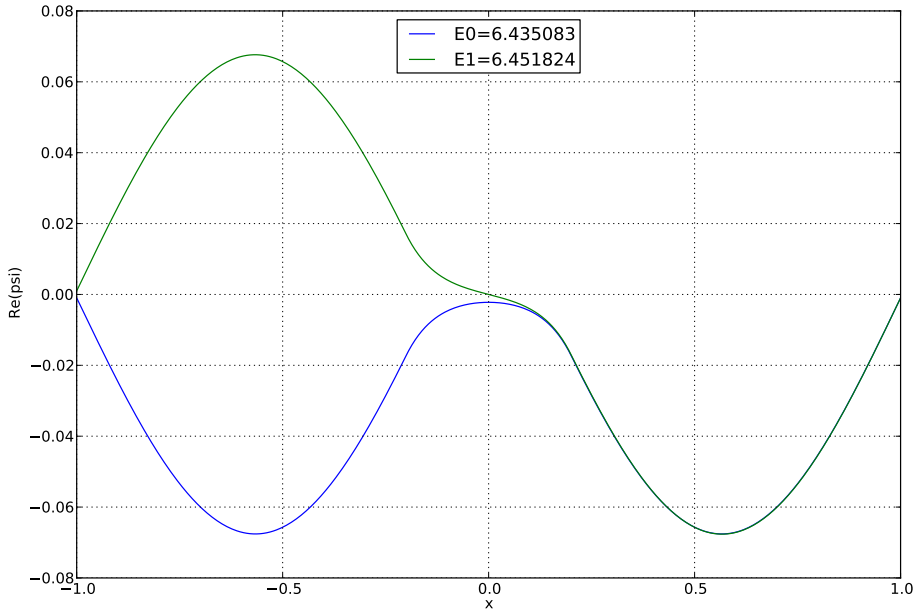
```
x = numpy.linspace(-1.0,1.0,500)
V = numpy.zeros(x.size)
nL = int(x.size*0.1)
m = int(x.size/2)
V[m-nL:m+nL] = 100.0
figure()
plot(x,V)
xlabel("x")
ylabel("V")
grid()
```



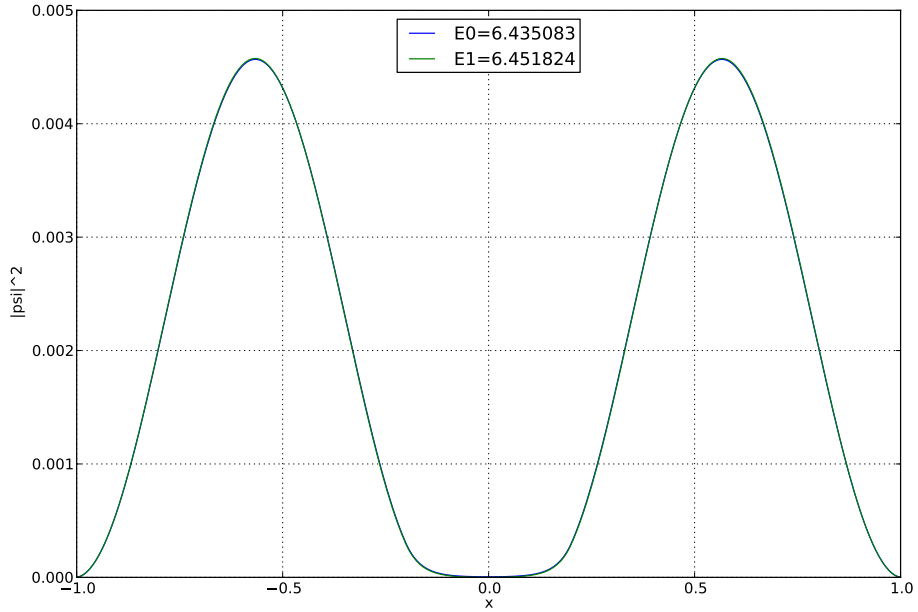

```
E,phi = etats(x,V,15)
```

On trace la partie réelle de la fonction d'onde et son module au carré pour les deux premiers états :

```
figure(figsize=(12,8))
i=0
plot(x,numpy.real(phi[:,i]),label="E%d=%f"%(i,E[i]))
i=1
plot(x,numpy.real(phi[:,i]),label="E%d=%f"%(i,E[i]))
xlabel("x")
ylabel("Re(psi)")
grid()
legend(loc='upper center')
```



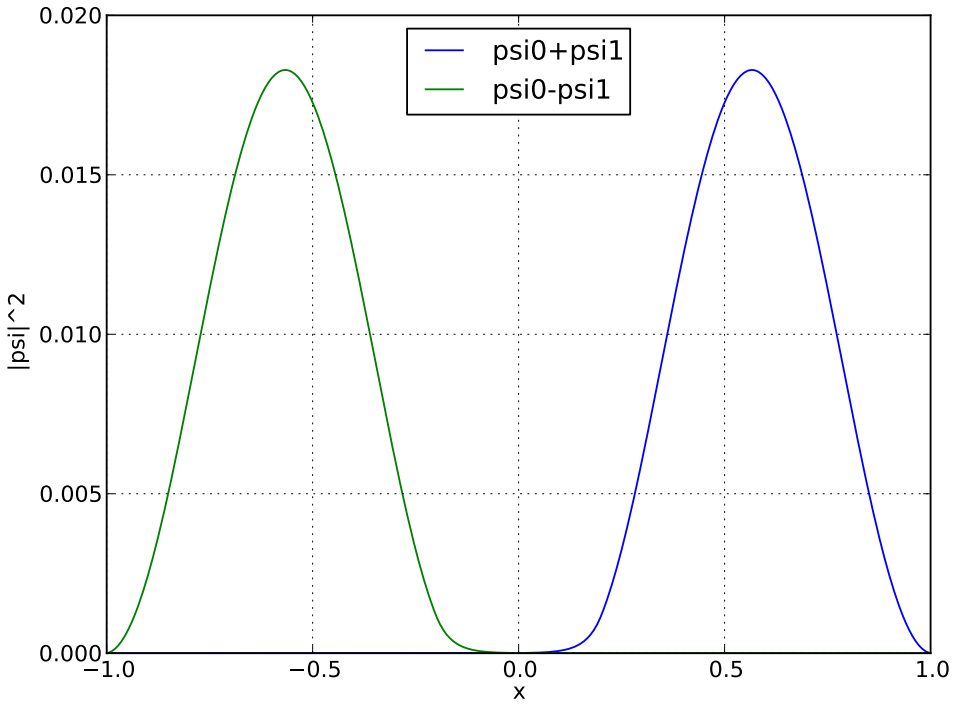

```
figure(figsize=(12,8))
i=0
plot(x,numpy.square(numpy.absolute(phi[:,i])),label="E%d=%f"%(i,E[i]))
i=1
plot(x,numpy.square(numpy.absolute(phi[:,i])),label="E%d=%f"%(i,E[i]))
xlabel("x")
ylabel("|psi|^2")
grid()
legend(loc='upper center')
```



Les deux premiers états des énergies très proches. Pour le premier, la fonction d'onde est symétrique. Pour le deuxième, elle est antisymétrique. La densité de probabilité est la même pour les deux états, avec une particule délocalisée sur les deux puits.

En faisant la somme (ou la différence) de ces deux états, on obtient une particule localisée seulement dans un des puits (état non stationnaire) :

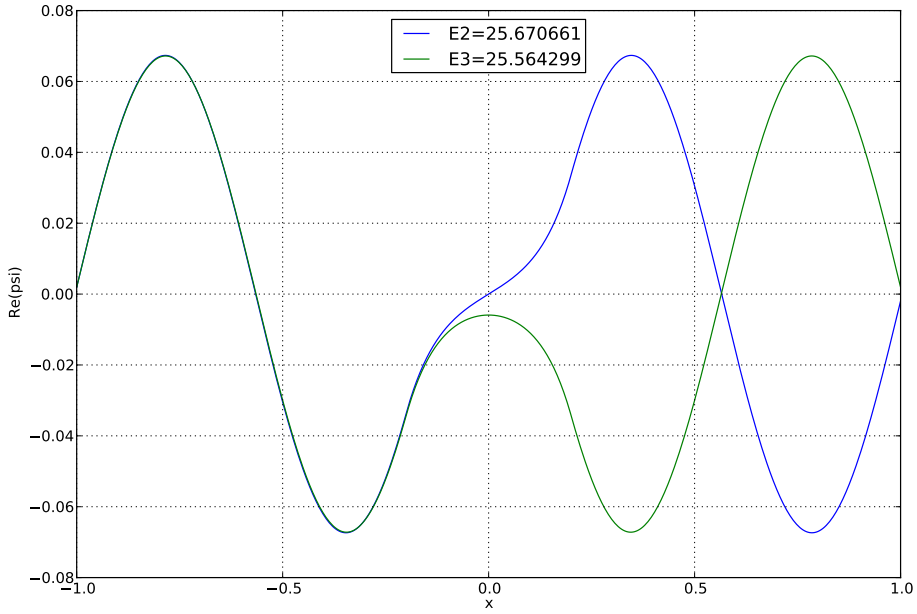
```
phiD = phi[:,0]+phi[:,1]
phiG = phi[:,0]-phi[:,1]
figure()
plot(x,numpy.square(numpy.absolute(phiD)),label="psi0+psi1")
plot(x,numpy.square(numpy.absolute(phiG)),label="psi0-psi1")
xlabel("x")
ylabel("|psi|^2")
grid()
legend(loc='upper center')
```



Voici les deux états suivants :

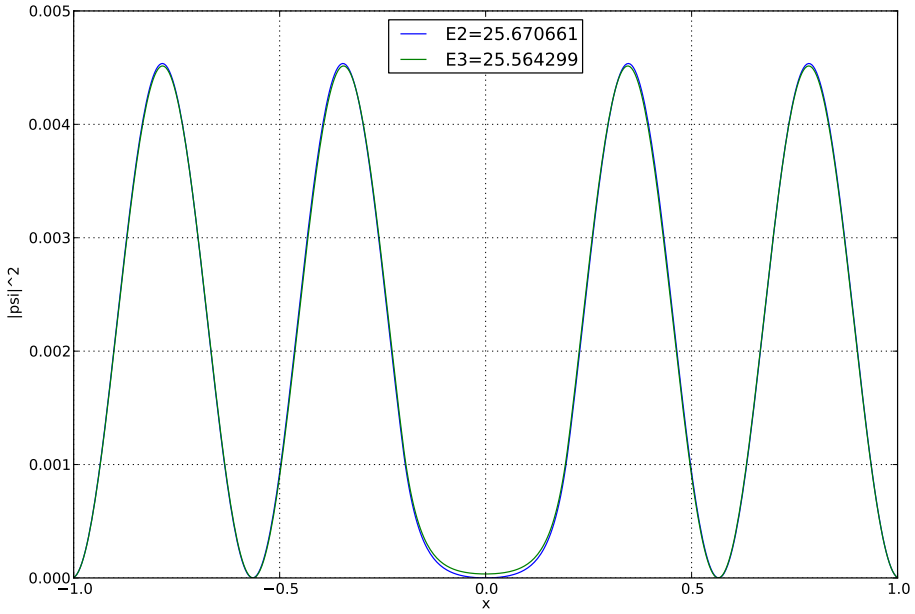
```
figure(figsize=(12,8))
i=2
plot(x,numpy.real(phi[:,i]),label="E%d=%f"%(i,E[i]))
i=3
plot(x,numpy.real(phi[:,i]),label="E%d=%f"%(i,E[i]))
xlabel("x")
ylabel("Re(psi)")
grid()
```

```
legend(loc='upper center')
```



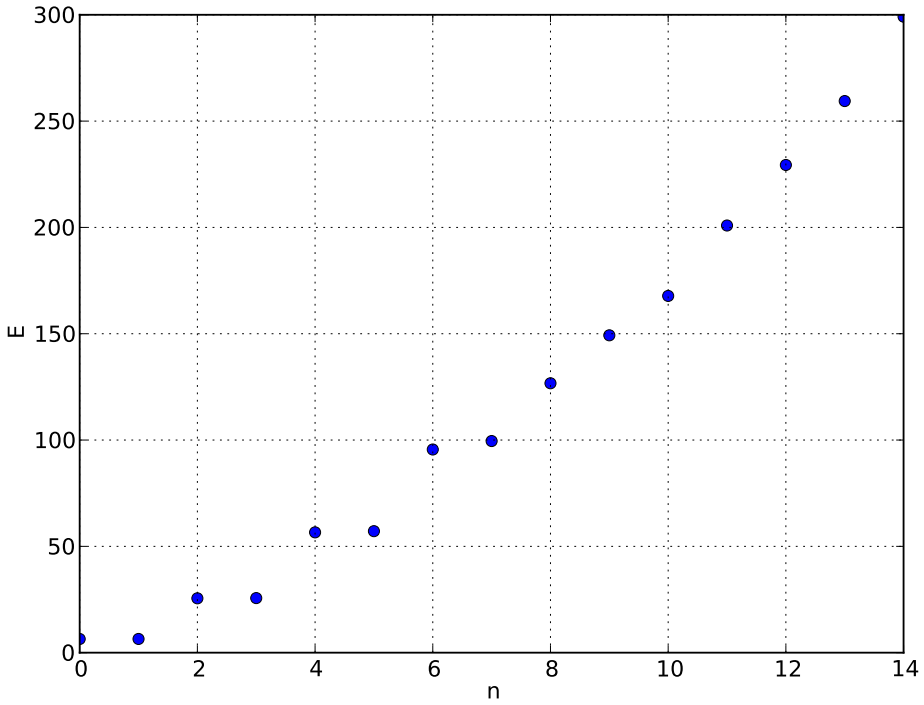
Ces deux niveaux n'apparaissent pas dans l'ordre d'énergie croissante. L'état symétrique est toujours celui de plus basse énergie.

```
figure(figsize=(12,8))
i=2
plot(x,numpy.square(numpy.absolute(phi[:,i])),label="E%d=%f"%(i,E[i]))
i=3
plot(x,numpy.square(numpy.absolute(phi[:,i])),label="E%d=%f"%(i,E[i]))
xlabel("x")
ylabel("|psi|^2")
grid()
legend(loc='upper center')
```



Voici les niveaux d'énergie :

```
E = sorted(E)
figure()
plot(E,'o')
xlabel("n")
ylabel("E")
grid()
```



Seuls les 8 premiers niveaux ont une énergie inférieure à la hauteur de la barrière séparant les deux puits. Pour les énergies supérieures à la hauteur, on retrouve la répartition d'un puit simple. En augmentant la hauteur de la barrière, on peut bien sûr augmenter le nombre d'états d'énergie inférieure à cette hauteur.

4.d. Potentiel périodique

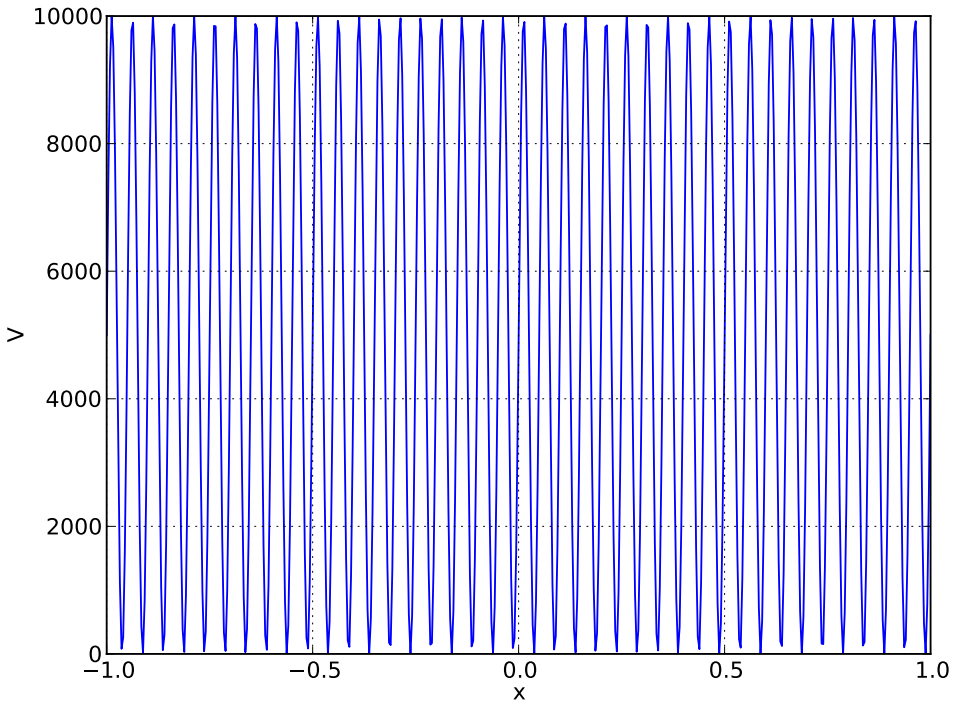
On modifie la matrice H pour appliquer une condition limite périodique sur la fonction d'onde.

```
def etats_periodique(x,V,nE,tol=0):
    N = V.size
    if x.size!=N:
        raise Exception("tailles de x et V incompatibles")
    alpha = 1.0/(x[1]-x[0])**2
    H = sparse.diags([-alpha/2,V+alpha,-alpha/2],[-1,0,1],shape=(N,N))
    H = H.toarray()
    H[0,N-1] = -alpha/2
    H[N-1,0] = -alpha/2
    E,phi = linalg.eigs(H,nE,which='SM',tol=tol)
    return numpy.real(E),phi
```

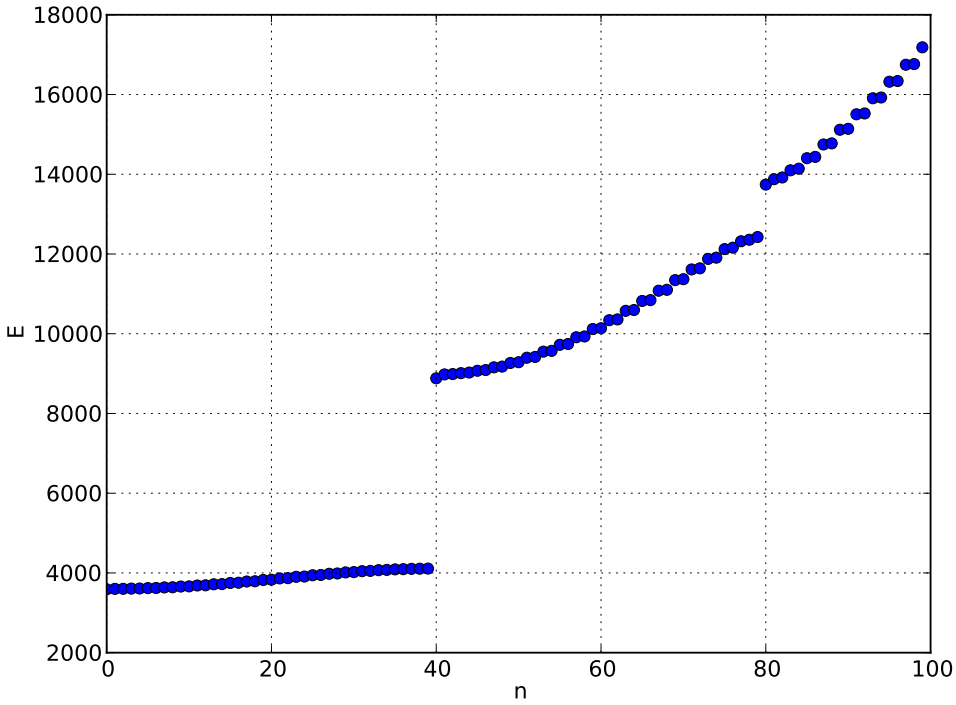
Voici un exemple avec un potentiel sinusoïdal :

```
x = numpy.linspace(-1.0,1.0,500)
V = (numpy.sin(40*numpy.pi*x)+1)*0.5*10000
figure()
```

```
plot(x,V)
xlabel("x")
ylabel("V")
grid()
```

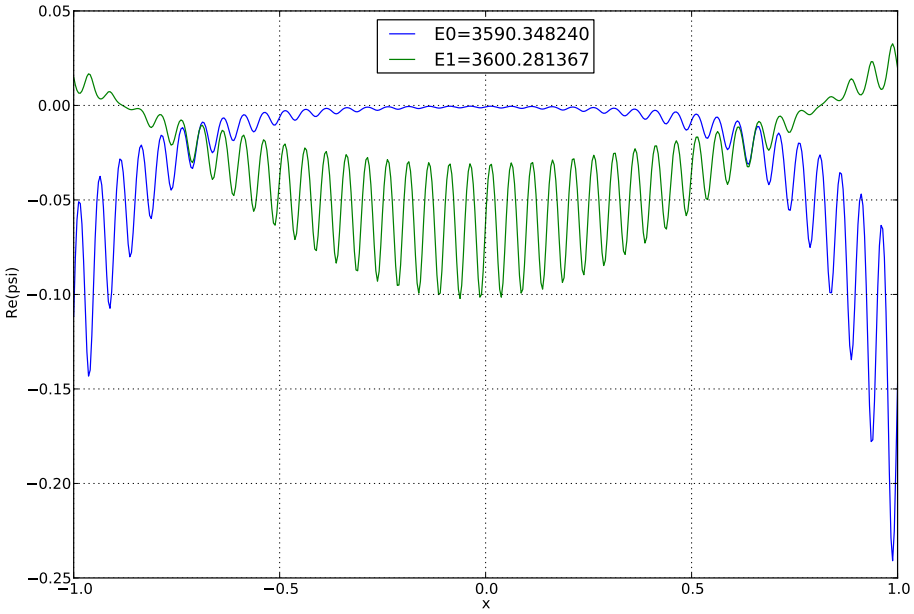



```
E,phi = etats_periodique(x,V,100)
E = sorted(E)
figure()
plot(E,'o')
xlabel("n")
ylabel("E")
grid()
```



Les énergies se répartissent par bandes. Les bandes d'énergie s'observent pour les électrons de conduction dans un métal. Ces électrons sont en effet soumis à un potentiel périodique de la part des cations du réseau cristallin.

```
figure(figsize=(12,8))
i=0
plot(x,numpy.real(phi[:,i]),label="E%d=%f"%(i,E[i]))
i=1
plot(x,numpy.real(phi[:,i]),label="E%d=%f"%(i,E[i]))
xlabel("x")
ylabel("Re(psi)")
grid()
legend(loc='upper center')
```


E0=3590.348240
E1=3600.281367
0.05
0.00
−0.05
−0.10
−0.15
−0.20
−0.25
Re(psi)
−1.0
−0.5
0.0
0.5
1.0
x