ΥΠΟΛΟΓΙΣΤΙΚΗ ΚΒΑΝΤΙΚΗ ΦΥΣΙΚΗ ΚΑΙ ΕΦΑΡΜΟΓΕΣ (ΥΦΥ201)
LISTINGS 6.14 , 6.22

ΑΡΙΣΤΟΤΕΛΕΙΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΟΝΙΚΗΣ
ΧΑΡΙΤΟΠΟΥΛΟΥ ΔΕΣΠΟΙΝΑ
Α.Ε.Μ.:4405
ΕΠΙΒΛΕΠΟΝΤΕΣ ΚΑΘΗΓΗΤΕΣ: ΜΟΥΣΤΑΚΙΔΗΣ Χ., ΧΑΤΖΗΣΑΒΒΑΣ Κ.

LISTING 6.14

- Υπολογίζει το κυματοπακέτο για φορτισμένο σωματίδιο σε πηγάδι αρμονικού ταλαντωτή επίσης εκτεθιμένο σε ηλεκτρικό πεδίο.
- Το αρχικό κυματοπακέτο που περιγράφει σωματίδιο με φορτίο q ,δεσμευμένο σε αρμονικό ταλαντωτή :

$$\psi(x, t=0) = e^{-(x/0.5)^2/2}\, e^{ipx}$$

Αλλιώς εκφρασμένο το κυματοπακέτο :

$$\psi(x,t) = \mathrm{Re}\psi(x,t) + i\,\mathrm{Im}\psi(x,t)$$

Όπου το πραγματικό μέρος γράφεται : $\psi_{Re} = e^{-0.5\left(\frac{x}{0.5}\right)^2} \cos(k_0 x)$

Και το φανταστικό : $\psi_{Im} = e^{-0.5\left(\frac{x}{0.5}\right)^2} \sin(k_0 x)$

- Έχουμε, επομένως, από την εξίσωση του Schrödinger δύο μερικές διαφορικές εξισώσεις (PDEs):

$$\frac{\partial \mathrm{Re}\psi(x,t)}{\partial t} = -\frac{1}{2m}\frac{\partial^2 \mathrm{Im}\psi(x,t)}{\partial x^2} + V(x)\mathrm{Im}\psi(x,t)$$

$$\frac{\partial \mathrm{Im}\psi(x,t)}{\partial t} = +\frac{1}{2m}\frac{\partial^2 \mathrm{Re}\psi(x,t)}{\partial x^2} - V(x)\mathrm{Re}\psi(x,t)$$

Τη λύση των οποίων υπολογίζουμε μέσω της μεθόδου των πεπερασμένων διαφορών (finite difference), και έτσι θα έχουμε τις εξής εξισώσεις:

$$\mathrm{Re}\psi(x,t+\tfrac{1}{2}\Delta t) = \mathrm{Re}\psi(x,t-\tfrac{1}{2}\Delta t) + [2\beta + V(x)\,\Delta t]\mathrm{Im}\psi(x,t) - \beta[\mathrm{Im}\psi(x+\Delta x,t) + \mathrm{Im}\psi(x-\Delta x,t)],$$

$$\mathrm{Im}\psi(x,t+\tfrac{1}{2}\Delta t) = \mathrm{Im}\psi(x,t-\tfrac{1}{2}\Delta t) - [2\beta+V(x)\Delta t]\mathrm{Re}\psi(x,t) + \beta[\mathrm{Re}\psi(x+\Delta x,t) + \mathrm{Re}\psi(x-\Delta x,t)]$$

Όπου $\beta = \frac{\Delta t}{\Delta x^2}$

- Στόχος μας είναι να δημιουργήσουμε ένα διαδραστικό plot (animation plot) στην python που να περιγράφει την κίνηση του κυματοπακέτου (την pdf του).

Ως δεδομένα έχουμε:

- E=70
- q=1
- k_0=5.5 × π
- x : (-6,6) με βήμα dx=0.06
- $dt = \frac{dx^2}{8}$

Λίγα λόγια για τον κώδικα

- Αρχικά, για τη δημιουργία διαδραστικού plot απαιτείται η ενεργοποίηση με την εντολή :

```
# Enable interactive plot
%matplotlib notebook
```

- Έπειτα εισάγουμε τις βιβλιοθήκες που θα χρησιμοποιήσουμε:

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib.animation import FuncAnimation
```

- Μετά δημιουργούμε αρχική κατάσταση για το animation plot. Καλούμε την εντολή subplot και ορίζουμε έναν άξονα ax:

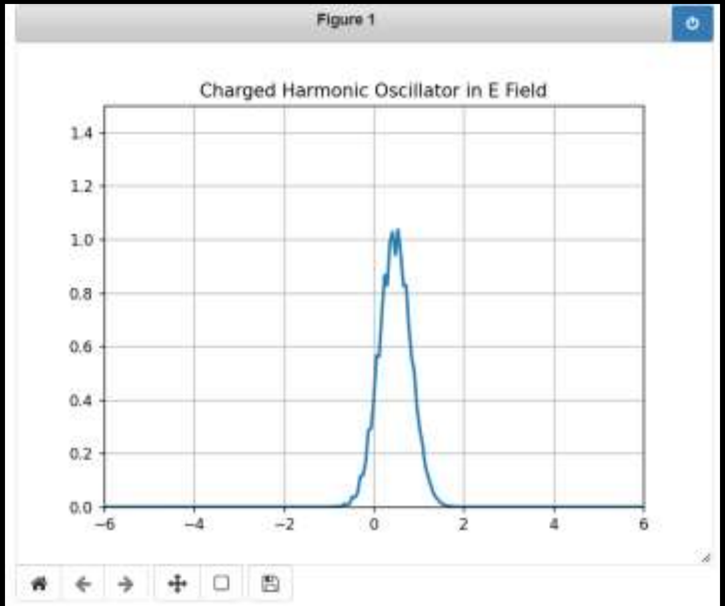
```
dx = 0.06
dx2 = dx*dx
k0 = 5.5*np.pi
dt = dx2/8.
xmax = 6.
xs = np.arange(-xmax,xmax+dx/2,dx) # x array
psr = np.exp(-0.5*(xs/0.5)**2)*np.cos(k0*xs) # Re psi
psi = np.exp(-0.5*(xs/0.5)**2)*np.sin(k0*xs) # Im psi
E = 70  # E field
v = 25.0*xs**2 - E*xs
fig = plt.figure()
ax = fig.add_subplot(111, autoscale_on=False, xlim=(-xmax,xmax), ylim=(0,1.5))
ax.grid()
plt.title('Charged Harmonic Oscillator in E Field')
line , = ax.plot(xs, psr*psr+psi*psi,lw=2)
```

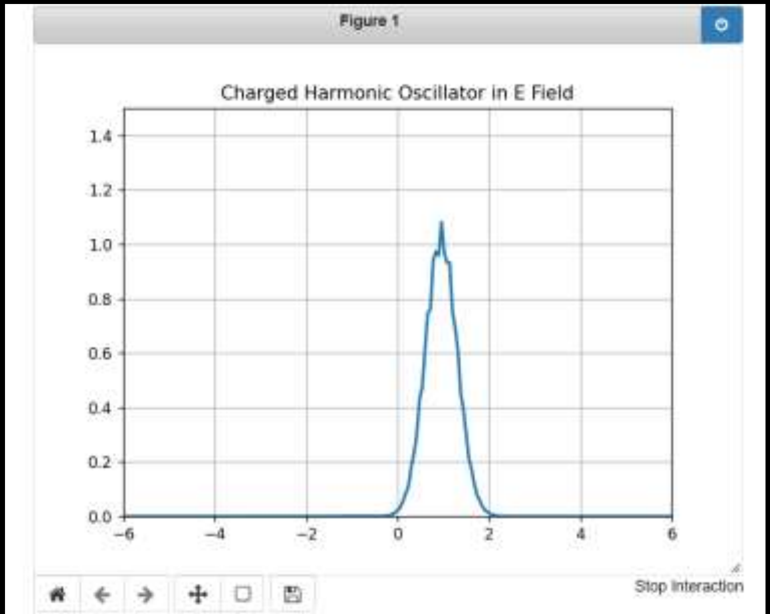
- Στη συνέχεια, δημιουργούμε μια συνάρτηση animate( ) η οποία θα καλείται από την FuncAnimation:

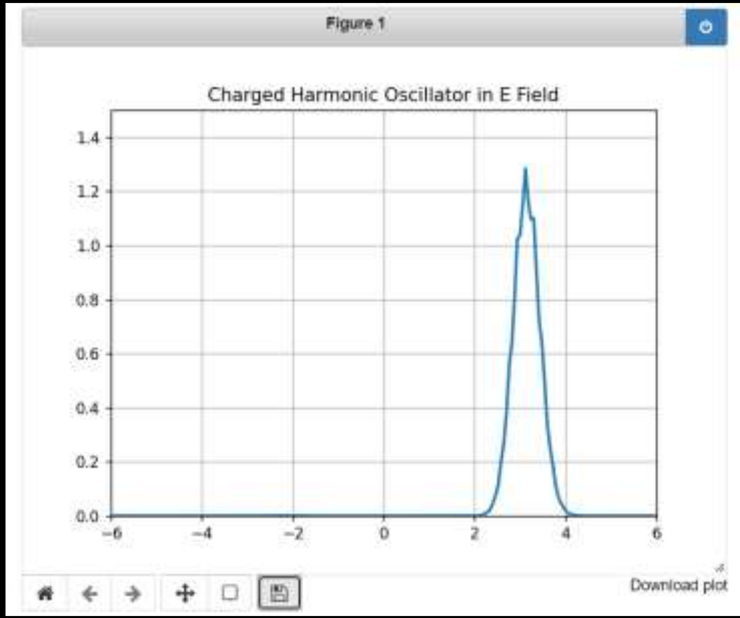
```
def animate(dum):
    psr[1:-1]=psr[1:-1]-(dt/dx2)*(psi[2:]+psi[:-2]-2*psi[1:-1])+dt*v[1:-1]*psi[1:-1]
    psi[1:-1]=psi[1:-1]+(dt/dx2)*(psr[2:]+psr[:-2]-2*psr[1:-1])-dt*v[1:-1]*psr[1:-1]
    line.set_data(xs, psr**2+psi**2)
    return line ,

ani = FuncAnimation(fig, animate, frames=100, blit=True)
plt.show()
```

- Τέλος, με την εντολή show έχουμε την δημιουργία του animation plot:







LISTING 6.22

- Υπολογίζει τη Χαμιλτονιανή, τις ιδιοτιμές και τα ιδιοδιανύσματα για entangled κβαντικές καταστάσεις.

- Entangled qubits: qubits που αλληλεπιδρούν μεταξύ τους.
- Qubit: σύστημα δύο καταστάσεων γνωστό ως quantum bit, στο οποίο αποθηκεύεται πληροφορία (quantum computing)
- Α και Β κβαντικά συστήματα >> διεμπλεκόμενα (entangled) >> συσχέτιση μεταξύ των τιμών κάποιων ιδιοτήτων του Α με τις αντίστοιχες ιδιότητες του Β.

If state $|\alpha\rangle$ belongs to a Hilbert space H_1, and state $|\beta\rangle$ belongs to a Hilbert space H_2, then the states are entangled if the state H in the Hilbert space of both states cannot be expressed as a tensor (direct) product of the two states.

- Έχουμε δύο μαγνητικά δίπολα A και B. Η Χαμιλτονιανή έχει τη μορφή:

$$H = \frac{\mu^2}{r^3}(\vec{\sigma}_A \cdot \vec{\sigma}_B - 3Z_A Z_B)$$

- Όπου :

$$\vec{\sigma}_A = X_A \hat{\mathbf{i}} + Y_A \hat{\mathbf{j}} + Z_A \hat{\mathbf{k}}$$

$$X \equiv \sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \quad Y \equiv \sigma_y = \begin{pmatrix} 0 & -i \\ i & 0 \end{pmatrix}, \quad Z \equiv \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}$$

Λίγα λόγια για τον κώδικα

- Αρχικά, εισάγουμε τις βιβλιοθήκες που θα χρησιμοποιήσουμε:

```
import numpy as np
import numpy.linalg as nl
```

- Έπειτα, εισάγουμε τους πίνακες $X_A X_B$, $Y_A Y_B$, $Z_A Z_B$, υπολογίζουμε τη Χαμιλτονιανή (χωρίς τον όρο $\frac{\mu^2}{r^3}$) και υπολογίζουμε τις ιδιοτιμές και τα ιδιοδιανύσματα της Χαμιλτονιανής χρησιμοποιώντας το πακέτο numpy.linalg της γραμμικής άλγεβρας:

```
nmax=4
H = np.zeros((nmax,nmax),float)
XAXB = np.array([[0, 0, 0, 1],[0, 0, 1, 0],[0, 1, 0, 0],[1, 0, 0, 0]])
YAYB = np.array([[0, 0, 0, -1],[0, 0, 1, 0],[0, 1, 0, 0],[-1, 0, 0, 0]])
ZAZB = np.array([[1, 0, 0, 0],[0, -1, 0, 0],[0, 0, -1, 0],[0, 0, 0, 1]])
SASB = XAXB + YAYB + ZAZB - 3*ZAZB
print('\n Hamiltonian without mu^2/r^3 factor: \n', SASB ,'\n')
es , ev = nl.eig(SASB)
print('Eigenvalues:\n', np.round(es,2) ,'\n')
print('Eigenvectors(in columns):\n' , ev , '\n' )
```

```
 Hamiltonian without mu^2/r^3 factor:
 [[-2  0  0  0]
 [ 0  2  2  0]
 [ 0  2  2  0]
 [ 0  0  0 -2]]

Eigenvalues:
 [ 4.  0. -2. -2.]

Eigenvectors(in columns):
 [[ 0.          0.          1.          0.        ]
 [ 0.70710678  0.70710678  0.          0.        ]
 [ 0.70710678 -0.70710678  0.          0.        ]
 [ 0.          0.          0.          1.        ]]
```

- Ιδιοτιμές : 4, 0, -2, -2
- Ιδιοδιανύσματα : (0, 0.70710678, 0.70710678, 0) , (0, 0.70710678, - 0.70710678, 0) , (1, 0, 0, 0) , (0, 0, 0, 1)

$$\phi_1 = \frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\1\\0\end{pmatrix} = \frac{|01\rangle + |10\rangle}{\sqrt{2}}, \qquad \phi_2 = \begin{pmatrix}0\\0\\0\\1\end{pmatrix} = |11\rangle,$$

$$\phi_4 = \frac{1}{\sqrt{2}}\begin{pmatrix}0\\1\\-1\\0\end{pmatrix} = \frac{|01\rangle - |10\rangle}{\sqrt{2}}, \qquad \phi_3 = \begin{pmatrix}1\\0\\0\\0\end{pmatrix} = |00\rangle.$$

- Έπειτα, χρησιμοποιώντας τα ιδιοδιανύσματα αυτά ως καινούργιες βασικές καταστάσεις εκφράζουμε τον Χαμιλτονιανό πίνακα , ο οποίος είναι διαγώνιος με τα στοιχεία της διαγωνίου να είναι οι ιδιοτιμές:

$$H = \begin{pmatrix} \langle\phi_1|H|\phi_1\rangle & \langle\phi_1|H|\phi_2\rangle & \langle\phi_1|H|\phi_3\rangle & \langle\phi_1|H|\phi_4\rangle \\ \langle\phi_2|H|\phi_1\rangle & \langle\phi_2|H|\phi_2\rangle & \langle\phi_2|H|\phi_3\rangle & \langle\phi_2|H|\phi_4\rangle \\ \langle\phi_3|H|\phi_1\rangle & \langle\phi_3|H|\phi_2\rangle & \langle\phi_3|H|\phi_3\rangle & \langle\phi_3|H|\phi_4\rangle \\ \langle\phi_4|H|\phi_1\rangle & \langle\phi_4|H|\phi_2\rangle & \langle\phi_4|H|\phi_3\rangle & \langle\phi_4|H|\phi_4\rangle \end{pmatrix}$$

```python
phi1 = (ev[0,0], ev[1,0], ev[2,0], ev[3,0]) # Eigenvectors
phi4 = (ev[0,1], ev[1,1], ev[2,1], ev[3,1])
phi3 = (ev[0,2], ev[1,2], ev[2,2], ev[3,2])
phi2 = (ev[0,3], ev[1,3], ev[2,3], ev[3,3])
basis=[phi1, phi2 , phi3 , phi4] # List eigenvectors

for i in range(0 , nmax): # Hamiltonian in new basis
     for j in range(0 , nmax):
          term     = np.dot(SASB,basis[i])
          H[i , j] = np.round(np.dot(basis[j],term),2)
print(" Hamiltonian in Eigenvector Basis: \n " ,H)
```

```
Hamiltonian in Eigenvector Basis:
 [[ 4.  0.  0.  0.]
[ 0. -2.  0.  0.]
[ 0.  0. -2.  0.]
[ 0.  0.  0.  0.]]
```

Σας ευχαριστώ για τον χρόνο σας