Computational Qunatum Physics and Applications

Dimitrios Psarras[1]

[1] Presenting author, dpsarras@auth.gr

June 13, 2022

ARISTOTLE
UNIVERSITY
OF THESSALONIKI

Table of Contents

Listing 6.6: Decay Sound

Problem

Simulate spontaneous decay. The code which also creates a sound each time a particle decays and plots the results.
The simulation was executed for:

- Radon-206 with decay constant $\lambda = 0.002533s^{-1}$
- Radon-220 with decay constant $\lambda = 0.0133s^{-1}$
- Radon-206 including the effect of stimulated emission with parameter $a = 0.01$
- Radon-220 including the effect of stimulated emission with parameter $a = 0.07$

Initialization


```
1 import vpython as vp
2 import random , winsound
3
4 lambda1 = 0.002533 # Decay constant
5 ns = [10., 100., 1000, 10000., 100000.]
6 tt = 6400
7
8 graph1 = vp.graph(title= "Spontaneous Decay of Radon-206 lambda=2.533e-3" , xtitle = "Time", ytitle
      = "log[N(t)]" , xmin=0 ,xmax=tt+10 , ymin=0 ,ymax=vp.log(ns[-1]+10)/vp.log(10))
```

```
8 graph1 = vp.graph(title= "Spontaneous Decay of Radon-220 lambda=0.0133" , xtitle = "Time", ytitle =
      "log[N(t)]" , xmin=0 ,xmax=tt+10 , ymin=0 ,ymax=vp.log(ns[-1]+10)/vp.log(10))
```

```
1 import vpython as vp
2 import random , winsound
3
4 lambda1 = 0.002533 # Decay constant
5 ns = [10., 100., 1000, 10000., 100000.]
6 tt = 6400
7
8 graph1 = vp.graph(title= "Spontaneous Decay of Radon-206 lambda=2.533e-3*0.01(DN(t)/DN(0))" ,
     xtitle = "Time", ytitle = "log[N(t)]" , xmin=0 ,xmax=tt+10 , ymin=0 ,ymax=vp.log(ns[-1]+10)/vp
     .log(10))
```

```
8 graph1 = vp.graph(title= "Spontaneous Decay of Radon-220 lambda=0.0133*0.07(DN(t)/DN(0))" , xtitle
     = "Time", ytitle = "log[N(t)]" , xmin=0 ,xmax=tt+10 , ymin=0 ,ymax=vp.log(ns[-1]+10)/vp.log
     (10))
```

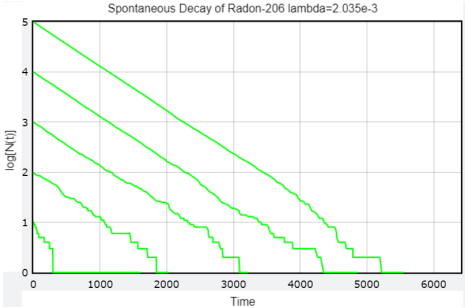
Simulation

```
for nn in ns:
    max_n = nn ; time_max = tt ; seed = 68111
    number = nloop = max_n # Initial value

    decayfunc = vp.gcurve( color = vp.color.green)
    decayfunc.plot( pos = ( 0 , vp.log(number)/vp.log(10) ) )
    for time in vp.arange( 0 , time_max + 1 ): # Time loop
        for atom in vp.arange ( 1 , number + 1 ): # Decay loop
            decay = random.random( )
            if ( decay < lambda1 ):
                nloop = nloop-1

                winsound.Beep( 600 , 100 ) # Sound beep
        number = nloop
        if number == 0:
            break
        decayfunc.plot( pos = ( time , vp.log(number)/vp.log(10) ) )
        vp.rate( 30 )
```

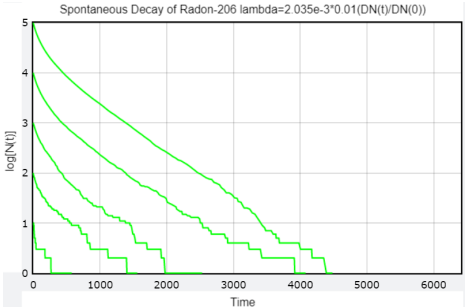
```
            l = lambda1 + (0.01*(number/max_n))
            if ( decay < l ):
                nloop = nloop-1
```

```
            l = lambda1 + (0.07*(number/max_n))
            if ( decay < l ):
                nloop = nloop-1
```

Figures for Radon 206

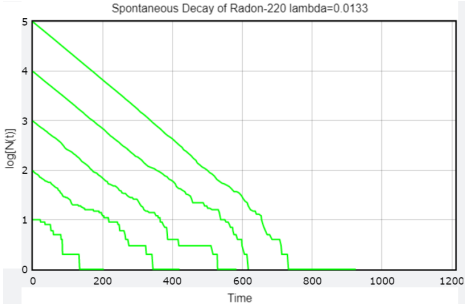


(a) The semilog plots show results of five simulations for Radon 206, all with the same decay constant, but differing initial numbers of atoms.

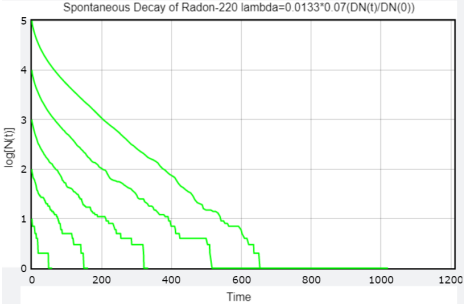


(b) The semilog plots show results of five simulations for Radon 206, all with the same decay constant including the effect of stimulated emission with parameter $a = 0.01$, but differing initial numbers of atoms.

Figures for Radon 220



(c) The semilog plots show results of five simulations for Radon 220, all with the same decay constant, but differing initial numbers of atoms.



(d) The semilog plots show results of five simulations for Radon 220, all with the same decay constant including the effect of stimulated emission with parameter $a = 0.07$, but differing initial numbers of atoms.

Listing 6.22: Qubits

Problem

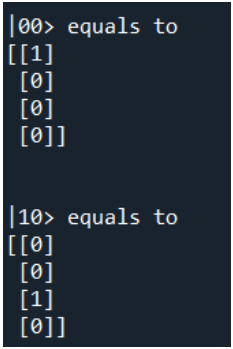
In quantum computing, information is stored in a two state system known as a quantum bit or qubit, for example, two spin states of a particle.
Through the code of this problem we want to compute the following:

- Compute the qubits as the direct products from a basis of $\mathbb{C}^4$
- Evaluate the tensor products $XA \otimes XB$, $YA \otimes YB$ and $ZA \otimes ZB$ as 4×4 matrices.
- Evaluate the Hamiltonian in the direct product space.
- Evaluate the eigenvalues of H and the corresponding eigenvectors.
- Using these eigenvectors states as new basis states, express the Hamiltonian matrix H in these basis states

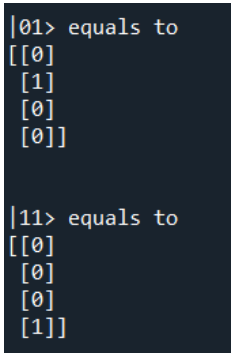
Calculation of Tensor Products

```
2  import numpy as np ; import numpy.linalg as lin
3
4  np.set_printoptions(suppress=True)
5
6  nmax = 4
7  H = np.zeros((nmax,nmax),float)
8
9  phi0 = np.array([[1],[0]])
10 phi1 = np.array([[0],[1]])
11 phi00 = np.kron(phi0,phi0)
12 phi10 = np.kron(phi1,phi0)
13 phi01 = np.kron(phi0,phi1)
14 phi11 = np.kron(phi1,phi1)
15 print(f"\n|00> equals to \n{phi00} \n" )
16 print(f"\n|10> equals to \n{phi10} \n" )
17 print(f"\n|01> equals to \n{phi01} \n" )
18 print(f"\n|11> equals to \n{phi11} \n" )
19
20 X = np.array([[0,1],[1,0]])
21 Y = np.array([[0,-1j],[1j,0]])
22 Z = np.array([[1,0],[0,-1]])
23 XAXB = np.kron(X,X)
24 YAYB = np.real(np.kron(Y,Y))
25 ZAZB = np.kron(Z,Z)
26 print(f"\nThe tensor product of XA and XB is \n{XAXB} \n" )
27 print(f"\nThe tensor product of YA and YB is \n{YAYB} \n" )
28 print(f"\nThe tensor product of ZA and ZB is \n{ZAZB} \n" )
```

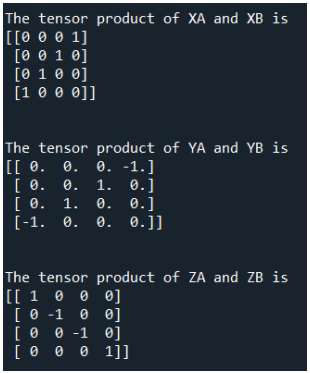
Printed results



(e) The states on which this Hamiltonian acts (The matrices of the Qubits).



(f) The states on which this Hamiltonian acts (The matrices of the Qubits).



(g) Results of the tensor products between $XA \otimes XB$, $YA \otimes YB$ and $ZA \otimes ZB$.

Calculation of Eigenvalues, Eigenvectors and Hamiltonian

```
30 SASB = XAXB + YAYB + ZAZB - 3*ZAZB # Hamiltonian
31
32 print(f"\nHamiltonian without mu^2/r^3 factor \n{SASB} \n" )
33
34 es , ev = lin.eig(SASB) # Eigenvalues and eigenvectors
35
36 print(f"Eigenvalues\n{es}\n")
37 print(f"Eigenvectors (in columns)\n{ev}\n" )
38
39 phi1 = (ev[0,0],ev[1,0],ev[2,0],ev[3,0]) # Eigenvectors
40 phi2 = (ev[0,3],ev[1,3],ev[2,3],ev[3,3])
41 phi3 = (ev[0,2],ev[1,2],ev[2,2],ev[3,2])
42 phi4 = (ev[0,1],ev[1,1],ev[2,1],ev[3,1])
43
44 basis = [phi1 , phi2 , phi3 , phi4 ] # Listeigenvectors
45
46 for i in range ( 0 ,nmax) : # Hamiltonian in new basis
47     for j in range ( 0 ,nmax) :
48         term = np.dot(SASB, basis[ i ] )
49         H[i,j] = np.dot(basis[j], term)
50
51 print (f"Hamiltonian in Eigenvector Basis \n{H}" )
```

Printed results

```
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.        ]]

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

Figure: The Hamiltonian in the direct product space, the eigenvalues of H and that the corresponding eigenvectors and the Hamiltonian matrix H in these basis states.