Python On Resonance (PyOR)

Everybody can simulate NMR

Author: Vineeth Thalakottoor

Email: vineethfrancis.physics@gmail.com

Why to develop PyOR when there are a lot of NMR simulation packages available? Why to reinvent the wheel?

The short answer: "The Pleasure of Finding Things Out" .

Why Python??

The answer is: it's free.

What is the purpose of this NMR Simulation tutorial?

Since it will take some time for me to make public the source code of PyOR, I taught to start a series of small NMR simulation tutorials so that the you (audience) can learn how to use PyOR to simulate NMR experiments when I upload the beta version (source code will be hidden) and first version later (source code will be open) in github.

And I want to show simulating NMR experiment from scratch can be really fun.

Tutorial 1: Spin Operators for Single spin

If you ask me what is the heart of PyOR?, my answer is the function which generate Spin Operators, "SpinOperator()". "SpinOperator()" can generate spin operators of a NMR system with any number of spins (if your pc support you) with any spin quantum number. This is my favorite function and any one can code it easly if you have a introductory quantum mechanics book.

With Spin Operators defiend, half the job is done. Let begin.

Load Python packages and define path to the source file "PythonOnResonance.py"

In [1]:
```
pathSource = '/path to/PyOR/Source'
```

In [2]:
```
from IPython.display import display, HTML
display(HTML("<style>.container { width:100% !important; }</style>"))
import sys
sys.path.append(pathSource)

import PythonOnResonance as PyOR

import time
```

```
import numpy as np
import matplotlib.pyplot as plt
from matplotlib import rc
%matplotlib notebook
import sympy as sp
from sympy import *
```

Generating Spin System

In [3]:
```
"""
Define Spin quantum numbers of your spins
""";

Slist1 = [2]

"""
Try also
Slist1 = [1/2]
Slist1 = [1]
Slist1 = [3/2]
Slist1 = [2]
Slist1 = [5/2]
So on ...
and compare with results in:
https://easyspin.org/easyspin/documentation/spinoperators.html
""";
```

In [4]:
```
"""
Define Planck constant equals 1.
Because NMR spectroscopists are more interested to write Energy in frequency units.
if False then hbarEQ1 = hbar
""";

hbarEQ1 = True
```

In [5]:
```
"""
Generate Spin Operators
""";

System = PyOR.Numerical_MR(Slist1,hbarEQ1)

"""
Sx, Sy and Sz Operators
""";
Sx,Sy,Sz = System.SpinOperator()

"""
S+ and S- Operators
""";
Sp,Sm = System.PMoperators(Sx,Sy)
```

Information about the functions

In [6]:
```
help(System.SpinOperator)
```

```
Help on method SpinOperator in module PythonOnResonance:
```

```
SpinOperator() method of PythonOnResonance.Numerical_MR instance
    Generate spin operators for all spins: Sx, Sy and Sz
    INPUT
    -----
    nill

    OUTPUT
    ------
    Sx : array [Sx of spin 1, Sx of spin 2, Sx of spin 3, ...]
    Sy : array [Sy of spin 1, Sy of spin 2, Sy of spin 3, ...]
    Sz : array [Sz of spin 1, Sz of spin 2, Sz of spin 3, ...]
```

In [7]:

```
help(System.PMoperators)
```

```
Help on method PMoperators in module PythonOnResonance:

PMoperators(Sx, Sy) method of PythonOnResonance.Numerical_MR instance
    Generate spin operators for all spins: Sp (Sx + j Sy) and Sm (Sx - j Sy)
    INPUT
    -----
    Sx, Sy

    OUTPUT
    ------
    Sp : array [Sp of spin 1, Sp of spin 2, Sp of spin 3, ...]
    Sm : array [Sm of spin 1, Sm of spin 2, Sm of spin 3, ...]
```

Representation of Spin Operators

In PyOR you visualize a matrix (Hamitonians, Density matrix and Operators) in various forms.

In [8]:

```
"""
Matrix represenatation (numpy)
""";
print("Sx = ", Sx)
print("Sy = ", Sx)
print("Sz = ", Sx)
print("Sp = ", Sp)
print("Sm = ", Sm)
```

```
Sx =  [[[0.        +0.j 1.        +0.j 0.        +0.j 0.        +0.j
   0.        +0.j]
  [1.        +0.j 0.        +0.j 1.22474487+0.j 0.        +0.j
   0.        +0.j]
  [0.        +0.j 1.22474487+0.j 0.        +0.j 1.22474487+0.j
   0.        +0.j]
  [0.        +0.j 0.        +0.j 1.22474487+0.j 0.        +0.j
   1.        +0.j]
  [0.        +0.j 0.        +0.j 0.        +0.j 1.        +0.j
   0.        +0.j]]]
Sy =  [[[0.        +0.j 1.        +0.j 0.        +0.j 0.        +0.j
   0.        +0.j]
  [1.        +0.j 0.        +0.j 1.22474487+0.j 0.        +0.j
   0.        +0.j]
  [0.        +0.j 1.22474487+0.j 0.        +0.j 1.22474487+0.j
   0.        +0.j]
  [0.        +0.j 0.        +0.j 1.22474487+0.j 0.        +0.j
   1.        +0.j]
  [0.        +0.j 0.        +0.j 0.        +0.j 1.        +0.j
   0.        +0.j]]]
Sz =  [[[0.        +0.j 1.        +0.j 0.        +0.j 0.        +0.j
```

```
  0.         +0.j]
 [1.         +0.j 0.         +0.j 1.22474487+0.j 0.         +0.j
  0.         +0.j]
 [0.         +0.j 1.22474487+0.j 0.         +0.j 1.22474487+0.j
  0.         +0.j]
 [0.         +0.j 0.         +0.j 1.22474487+0.j 0.         +0.j
  1.         +0.j]
 [0.         +0.j 0.         +0.j 0.         +0.j 1.         +0.j
  0.         +0.j]]]
Sp =  [[[0.         +0.j 2.         +0.j 0.         +0.j 0.         +0.j
  0.         +0.j]
 [0.         +0.j 0.         +0.j 2.44948974+0.j 0.         +0.j
  0.         +0.j]
 [0.         +0.j 0.         +0.j 0.         +0.j 2.44948974+0.j
  0.         +0.j]
 [0.         +0.j 0.         +0.j 0.         +0.j 0.         +0.j
  2.         +0.j]
 [0.         +0.j 0.         +0.j 0.         +0.j 0.         +0.j
  0.         +0.j]]]
Sm =  [[[0.         +0.j 0.         +0.j 0.         +0.j 0.         +0.j
  0.         +0.j]
 [2.         +0.j 0.         +0.j 0.         +0.j 0.         +0.j
  0.         +0.j]
 [0.         +0.j 2.44948974+0.j 0.         +0.j 0.         +0.j
  0.         +0.j]
 [0.         +0.j 0.         +0.j 2.44948974+0.j 0.         +0.j
  0.         +0.j]
 [0.         +0.j 0.         +0.j 0.         +0.j 2.         +0.j
  0.         +0.j]]]
```

In [9]:

```
"""
Matrix represenatation (Sympy)
In Sx[0], 0 means the index of the spin (0th spin or first spin).
""";
Matrix(Sx[0])
```

Out[9]:

$$\begin{bmatrix} 0 & 1.0 & 0 & 0 & 0 \\ 1.0 & 0 & 1.22474487139159 & 0 & 0 \\ 0 & 1.22474487139159 & 0 & 1.22474487139159 & 0 \\ 0 & 0 & 1.22474487139159 & 0 & 1.0 \\ 0 & 0 & 0 & 1.0 & 0 \end{bmatrix}$$

In [10]:

```
Matrix(Sy[0])
```

Out[10]:

$$\begin{bmatrix} 0 & -1.0i & 0 & 0 & 0 \\ 1.0i & 0 & -1.22474487139159i & 0 & 0 \\ 0 & 1.22474487139159i & 0 & -1.22474487139159i & 0 \\ 0 & 0 & 1.22474487139159i & 0 & -1.0i \\ 0 & 0 & 0 & 1.0i & 0 \end{bmatrix}$$

In [11]:

```
Matrix(Sz[0])
```

Out[11]:

$$\begin{bmatrix} 2.0 & 0 & 0 & 0 & 0 \\ 0 & 1.0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1.0 & 0 \\ 0 & 0 & 0 & 0 & -2.0 \end{bmatrix}$$

In [12]:
```
Matrix(Sp[0])
```

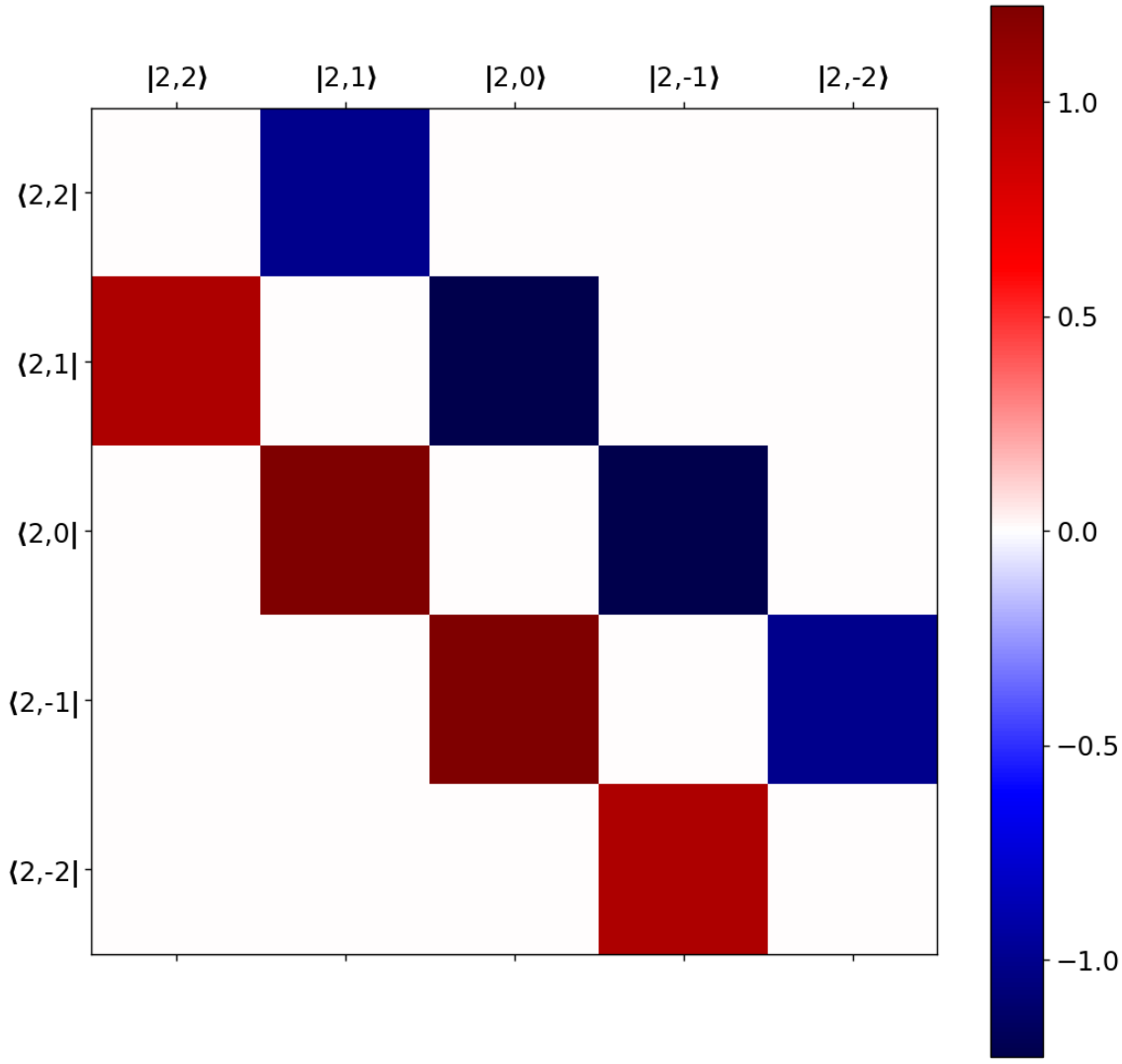
Out[12]:
$$\begin{bmatrix} 0 & 2.0 & 0 & 0 & 0 \\ 0 & 0 & 2.44948974278318 & 0 & 0 \\ 0 & 0 & 0 & 2.44948974278318 & 0 \\ 0 & 0 & 0 & 0 & 2.0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

In [13]:
```
Matrix(Sm[0])
```

Out[13]:
$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 2.0 & 0 & 0 & 0 & 0 \\ 0 & 2.44948974278318 & 0 & 0 & 0 \\ 0 & 0 & 2.44948974278318 & 0 & 0 \\ 0 & 0 & 0 & 2.0 & 0 \end{bmatrix}$$

In [14]:
```
"""
Matrix Represenatation (matplotlib)
""";

System.MatrixPlot(1,Sy[0].imag)
```

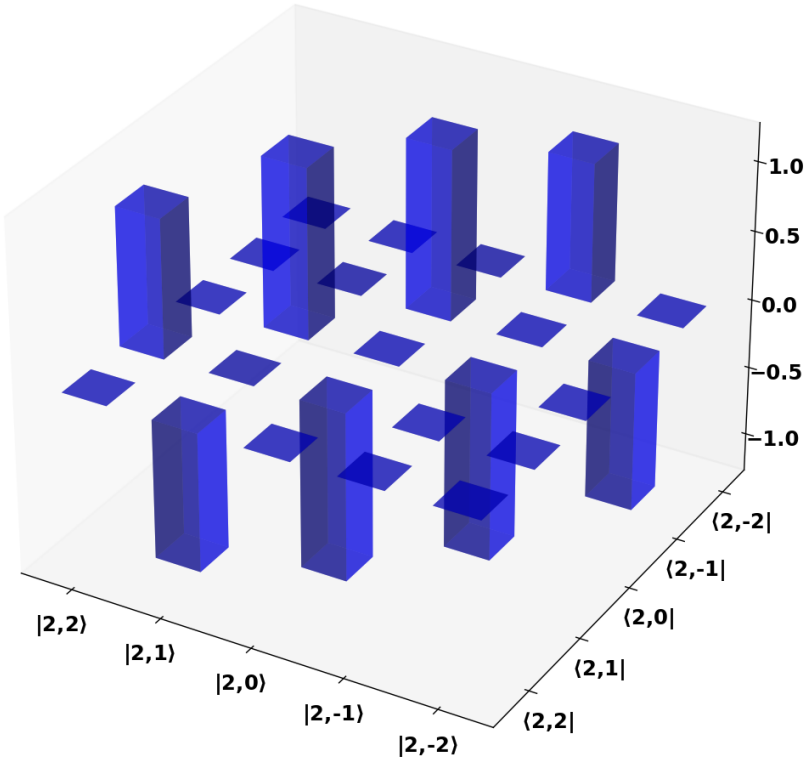



```
/media/HD2/Vineeth/PostDoc_Simulations/Github/PyOR_G/Source/PythonOnResonance.py:588: User
Warning: FixedFormatter should only be used together with FixedLocator
  ax.set_xticklabels([''] + labelx)
/media/HD2/Vineeth/PostDoc_Simulations/Github/PyOR_G/Source/PythonOnResonance.py:589: User
Warning: FixedFormatter should only be used together with FixedLocator
  ax.set_yticklabels([''] + labely)
```

In [15]:

```
"""
Matrix 3D Represenatation (matplotlib)
""";

System.MatrixPlot3D(2,Sy[0].imag)
```



Next lecture: Spin Operators for multi spin system

In this lecture you will see how to use Spin Operators more clearly

Any suggestion? write to me

vineethfrancis.physics@gmail.com

In [ ]: