20191108—3.2_Basic_Operations_on_Quantum_Objects

February 3, 2020

1 Om

2 20191108—3.2_Basic_Operations_on_Quantum_Objects

2.1 QuTiP

2.1.1 User Guide

- (http://qutip.org/docs/4.1/guide/guide.html)

  -3.2 Basic Operations on Quantum Objects (Upto 20 pages in qutip doc book) -3.2.1 First things first -3.2.2 The quantum object class -3.2.3 Functions operating on Qobj class

2.2 3.2 Basic Operations on Quantum Objects

2.2.1 3.2.1 First things first

- Warning: Do not run QuTiP from the installation directory.
- To load the qutip modules, we must first call the import statement:

[1]:
```
from qutip import *
```

that will load all of the user available functions. Often, we also need to import the NumPy and Matplotlib libraries with:

[2]:
```
import numpy as np
import matplotlib.pyplot as plt
```

Note that, in the rest of the documentation, functions are written using – qutip.module.function() – notation which links to the corresponding function in the – QuTiP API: Functions. – However, in calling import *, we have already loaded all of the QuTiP modules. Therefore, we will only need the function name and not the complete path when calling the function from the interpreter prompt, Python script, or Jupyter notebook.

2.2.2 3.2.2 The quantum object class

Introduction

- The key difference between classical and quantum mechanics lies in the use of operators instead of numbers as variables.
- Moreover, we need to specify state vectors and their properties.

- Therefore, in computing the dynamics of quantum systems we need a data structure that is capable of encapsulating the properties of a quantum operator and ket/bra vectors.
- The quantum object class, qutip.Qobj, accomplishes this using matrix representation.

To begin, let us create a blank Qobj:

[3]:
```
Qobj()
```

[3]:
```
Quantum object: dims = [[1], [1]], shape = (1, 1), type = bra
```

$$\begin{pmatrix} 0.0 \end{pmatrix}$$

`where we see the blank Qobj object with dimensions, shape, and data. Here the data corresponds to a 1x1- dimensional matrix consisting of a single zero entry.`

`Hint: By convention, Class objects in Python such as Qobj() differ from functions in the use of a beginning capital letter.`

`We can create a Qobj with a user defined data set by passing a list or array of data into the Qobj:`

[4]:
```
# Just Observe the difference between with print() and without
Qobj([[1],[2],[3],[4],[5],[6]])
```

[4]:
```
Quantum object: dims = [[6], [1]], shape = (6, 1), type = ket
```

$$\begin{pmatrix} 1.0 \\ 2.0 \\ 3.0 \\ 4.0 \\ 5.0 \\ 6.0 \end{pmatrix}$$

[5]:
```
print(Qobj([[1],[2],[3],[4],[5],[6]]))
```

```
Quantum object: dims = [[6], [1]], shape = (6, 1), type = ket
Qobj data =
[[1.]
 [2.]
 [3.]
 [4.]
 [5.]
 [6.]]
```

[6]:
```
Qobj(np.array([[1,2,3,4,5]]))
```

[6]:
```
Quantum object: dims = [[1], [5]], shape = (1, 5), type = bra
```

$$\begin{pmatrix} 1.0 & 2.0 & 3.0 & 4.0 & 5.0 \end{pmatrix}$$

```
[7]: print(Qobj(np.array([[1,2,3,4,5]])))
```

```
Quantum object: dims = [[1], [5]], shape = (1, 5), type = bra
Qobj data =
[[1. 2. 3. 4. 5.]]
```

Here in the above we can see the dims, shape, type and - what all these called.? and why these are different from actual python.?

```
[8]: # random
     Qobj(np.random.rand(5,5))
```

[8]: Quantum object: dims = [[5], [5]], shape = (5, 5), type = oper, isherm = False

$$\begin{pmatrix} 0.713 & 0.649 & 0.492 & 0.530 & 0.544 \\ 0.886 & 0.716 & 0.572 & 0.925 & 0.847 \\ 0.736 & 0.252 & 0.683 & 0.720 & 0.608 \\ 0.660 & 0.257 & 0.176 & 0.743 & 0.353 \\ 0.084 & 0.374 & 0.656 & 0.332 & 0.836 \end{pmatrix}$$

Here - What is type called OPERATOR ? - What is isherm and is False ?

Notice how both the dims and shape change according to the input data. Although dims and shape appear to have the same function, the difference will become quite clear in the section on tensor products and partial traces.

Note: If you are running QuTiP from a python script you must use the print function to view the Qobj attributes.

Next topic is about - Quantum States and Quantum Operators (States and Operators)

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```

```
[ ]:
```