HW4_Code

February 9, 2023

[ ]:
```
import random
import numpy as np
from qutip import *
from qutip.measurement import measure

from qiskit.visualization import array_to_latex
```

0.0.1 Question 1)

A)

[ ]:
```
b = 13

# Try finding k from 0 to 200
for k in range(0, 200):
    if (k*b) % 180 == 1:
        break
print("Our found value of k is {}".format(k))
```

```
Our found value of k is 97
```

0.0.2 Question 3)

[ ]:
```
H = ket("H") # horizontal state
V = ket("V") # vertical state
D = 1/np.sqrt(2) * (H + V) # Diagonal state (+45)
A = 1/np.sqrt(2) * (H - V) # Anti-Diagonal state (-45)

# Helper function to convert a basis into a string representation
def basis_to_str(B):
    if B == H:
        return "H"
    elif B == V:
        return "V"
    elif B == D:
        return "D"
    elif B == A:
        return "A"
```

```
    else:
        return "UNKNOWN"
```

[ ]:
```
def GetRandomBasis(N_p, basis_states):
    basis = {}
    for i in range(1, N_p+1):
        basis[i] = random.choice(basis_states)
    return basis
```

A)

[ ]:
```
# Function modeling Alice sending a message to Bob with N_p number of photons
def AliceMessage(N_p, basis):
    sent_message = {}
    for i in range(1, N_p+1):
        sent_message[i] = basis[i]
    return sent_message
```

Example of Alice sending 5 photons:

[ ]:
```
alice_message = AliceMessage(5, GetRandomBasis(5, [H,D]))
alice_message
```

[ ]:
```
{1: Quantum object: dims = [[2], [1]], shape = (2, 1), type = ket
 Qobj data =
 [[1.]
  [0.]],
 2: Quantum object: dims = [[2], [1]], shape = (2, 1), type = ket
 Qobj data =
 [[0.70710678]
  [0.70710678]],
 3: Quantum object: dims = [[2], [1]], shape = (2, 1), type = ket
 Qobj data =
 [[0.70710678]
  [0.70710678]],
 4: Quantum object: dims = [[2], [1]], shape = (2, 1), type = ket
 Qobj data =
 [[1.]
  [0.]],
 5: Quantum object: dims = [[2], [1]], shape = (2, 1), type = ket
 Qobj data =
 [[0.70710678]
  [0.70710678]]}
```

B)

[ ]:
```
bob_basis = GetRandomBasis(5, [V,A])
print(bob_basis)
```

```
{1: Quantum object: dims = [[2], [1]], shape = (2, 1), type = ket
Qobj data =
[[ 0.70710678]
 [-0.70710678]], 2: Quantum object: dims = [[2], [1]], shape = (2, 1), type =
ket
Qobj data =
[[ 0.70710678]
 [-0.70710678]], 3: Quantum object: dims = [[2], [1]], shape = (2, 1), type =
ket
Qobj data =
[[0.]
 [1.]], 4: Quantum object: dims = [[2], [1]], shape = (2, 1), type = ket
Qobj data =
[[0.]
 [1.]], 5: Quantum object: dims = [[2], [1]], shape = (2, 1), type = ket
Qobj data =
[[0.]
 [1.]]}
```

Now that Bob has a basis, lets create a function that measures Alice's message

[ ]:
```
def BobReceiveMessage(N_p, message, measurement_basis):
    measurements = {}
    for i in range(1, N_p+1):
        # Create a density matrix out of our basix
        cur_basis_op = ket2dm(measurement_basis[i])

        # Measure the projected eigen value/state with the given basis
        eigen_val, eigen_state = measure(message[i], cur_basis_op)
        measurements[i] = round(eigen_val)
    return measurements

BobReceiveMessage(5, alice_message, bob_basis)
```

[ ]:
```
{1: 1, 2: 1, 3: 0, 4: 1, 5: 1}
```

C) First, lets generate an 100 photon message from Alice and have Bob measure it with a randomly chosen basis

[ ]:
```
N_p = 100
alice_basis = GetRandomBasis(N_p, [H,D])
alice_message = AliceMessage(N_p, alice_basis)
bob_basis = GetRandomBasis(N_p, [V,A])
bob_measurements = BobReceiveMessage(N_p, alice_message, bob_basis)
```

Now, lets display all of the states Bob got a 'click' with

[ ]:
```
def GetCorrectBits(bob_measurements, bob_basis):
    correct_bits = []
    for i in range(1, N_p+1):
        if bob_measurements[i] == 1:
            bit = 1 if bob_basis[i] == V else 0
            correct_bits.append(bit)
            print("Bob got click for photon {:02d} with basis {}. Sent bit was␣
 ↪{}.".format(i, basis_to_str(bob_basis[i]), bit))

    print("\nNumber of correct bits Bob recieved: {}".format(len(correct_bits)))
    return correct_bits

correct_bits = GetCorrectBits(bob_measurements, bob_basis)
```

```
Bob got click for photon 05 with basis A. Sent bit was 0.
Bob got click for photon 06 with basis A. Sent bit was 0.
Bob got click for photon 09 with basis V. Sent bit was 1.
Bob got click for photon 14 with basis V. Sent bit was 1.
Bob got click for photon 17 with basis V. Sent bit was 1.
Bob got click for photon 18 with basis V. Sent bit was 1.
Bob got click for photon 20 with basis V. Sent bit was 1.
Bob got click for photon 21 with basis A. Sent bit was 0.
Bob got click for photon 38 with basis V. Sent bit was 1.
Bob got click for photon 40 with basis A. Sent bit was 0.
Bob got click for photon 41 with basis A. Sent bit was 0.
Bob got click for photon 42 with basis V. Sent bit was 1.
Bob got click for photon 43 with basis V. Sent bit was 1.
Bob got click for photon 45 with basis A. Sent bit was 0.
Bob got click for photon 51 with basis V. Sent bit was 1.
Bob got click for photon 54 with basis A. Sent bit was 0.
Bob got click for photon 58 with basis A. Sent bit was 0.
Bob got click for photon 71 with basis A. Sent bit was 0.
Bob got click for photon 73 with basis V. Sent bit was 1.
Bob got click for photon 74 with basis A. Sent bit was 0.
Bob got click for photon 80 with basis A. Sent bit was 0.
Bob got click for photon 91 with basis V. Sent bit was 1.
Bob got click for photon 95 with basis V. Sent bit was 1.
Bob got click for photon 99 with basis A. Sent bit was 0.
Bob got click for photon 100 with basis A. Sent bit was 0.

Number of correct bits Bob recieved: 25
```

Cool! Now Bob can broadcast his basis to Alice and she'll know which of her bits were sent correctly. We also see that the accepted error rate is roughly 25% for B92.

Lets now assume they create a private key from the first half of the correct bits. This corresponds to:

[ ]:
```
correct_bits[:len(correct_bits)//2]
```

[ ]:
```
[0, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1]
```

D) Eve knows for sure what state Alice must've sent if she gets a click. If she gets no click, however, then the best she can do is just randomly send either a 0 or 1

[ ]:
```
def Eavesdrop(N_p, message, measurement_basis):
    eve_message = {}
    measurements = {}
    for i in range(1, N_p+1):
        # Create a density matrix out of our basix
        cur_basis_op = ket2dm(measurement_basis[i])

        # Measure the projected eigen value/state with the given basis
        eigen_val, eigen_state = measure(message[i], cur_basis_op)
        measurements[i] = round(eigen_val)

        # Got a click. We for sure know the state
        if measurements[i] == 1:
            if measurement_basis[i] == V:
                eve_message[i] = D
            else:
                eve_message[i] = H
        # Don't have a click, so we pick randomly
        elif measurements[i] == 0:
            eve_message[i] = random.choice([H, D])

    return eve_message
```

E)

[ ]:
```
N_p = 100
alice_basis = GetRandomBasis(N_p, [H,D])
alice_message = AliceMessage(N_p, alice_basis)

eve_basis = GetRandomBasis(N_p, [V,A])
eve_message = Eavesdrop(N_p, alice_message, eve_basis)

bob_basis = GetRandomBasis(N_p, [V,A])
bob_measurements = BobReceiveMessage(N_p, eve_basis, bob_basis)
```

[ ]:
```
correct_bits = GetCorrectBits(bob_measurements, bob_basis)
```

```
Bob got click for photon 03 with basis V. Sent bit was 1.
Bob got click for photon 06 with basis V. Sent bit was 1.
Bob got click for photon 10 with basis A. Sent bit was 0.
```

```
Bob got click for photon 13 with basis V. Sent bit was 1.
Bob got click for photon 15 with basis A. Sent bit was 0.
Bob got click for photon 16 with basis V. Sent bit was 1.
Bob got click for photon 17 with basis A. Sent bit was 0.
Bob got click for photon 18 with basis V. Sent bit was 1.
Bob got click for photon 19 with basis A. Sent bit was 0.
Bob got click for photon 20 with basis V. Sent bit was 1.
Bob got click for photon 21 with basis V. Sent bit was 1.
Bob got click for photon 24 with basis A. Sent bit was 0.
Bob got click for photon 25 with basis A. Sent bit was 0.
Bob got click for photon 26 with basis A. Sent bit was 0.
Bob got click for photon 28 with basis A. Sent bit was 0.
Bob got click for photon 29 with basis A. Sent bit was 0.
Bob got click for photon 30 with basis A. Sent bit was 0.
Bob got click for photon 31 with basis A. Sent bit was 0.
Bob got click for photon 32 with basis A. Sent bit was 0.
Bob got click for photon 34 with basis A. Sent bit was 0.
Bob got click for photon 35 with basis V. Sent bit was 1.
Bob got click for photon 36 with basis V. Sent bit was 1.
Bob got click for photon 37 with basis A. Sent bit was 0.
Bob got click for photon 38 with basis A. Sent bit was 0.
Bob got click for photon 39 with basis V. Sent bit was 1.
Bob got click for photon 40 with basis V. Sent bit was 1.
Bob got click for photon 41 with basis V. Sent bit was 1.
Bob got click for photon 42 with basis A. Sent bit was 0.
Bob got click for photon 43 with basis V. Sent bit was 1.
Bob got click for photon 44 with basis V. Sent bit was 1.
Bob got click for photon 46 with basis A. Sent bit was 0.
Bob got click for photon 47 with basis A. Sent bit was 0.
Bob got click for photon 48 with basis V. Sent bit was 1.
Bob got click for photon 49 with basis V. Sent bit was 1.
Bob got click for photon 50 with basis V. Sent bit was 1.
Bob got click for photon 52 with basis V. Sent bit was 1.
Bob got click for photon 53 with basis V. Sent bit was 1.
Bob got click for photon 54 with basis V. Sent bit was 1.
Bob got click for photon 55 with basis V. Sent bit was 1.
Bob got click for photon 56 with basis A. Sent bit was 0.
Bob got click for photon 58 with basis V. Sent bit was 1.
Bob got click for photon 60 with basis A. Sent bit was 0.
Bob got click for photon 61 with basis V. Sent bit was 1.
Bob got click for photon 63 with basis A. Sent bit was 0.
Bob got click for photon 64 with basis V. Sent bit was 1.
Bob got click for photon 66 with basis V. Sent bit was 1.
Bob got click for photon 67 with basis V. Sent bit was 1.
Bob got click for photon 68 with basis V. Sent bit was 1.
Bob got click for photon 70 with basis V. Sent bit was 1.
Bob got click for photon 71 with basis V. Sent bit was 1.
Bob got click for photon 72 with basis V. Sent bit was 1.
```

```
Bob got click for photon 73 with basis V. Sent bit was 1.
Bob got click for photon 74 with basis A. Sent bit was 0.
Bob got click for photon 75 with basis V. Sent bit was 1.
Bob got click for photon 76 with basis V. Sent bit was 1.
Bob got click for photon 78 with basis A. Sent bit was 0.
Bob got click for photon 79 with basis A. Sent bit was 0.
Bob got click for photon 80 with basis V. Sent bit was 1.
Bob got click for photon 81 with basis A. Sent bit was 0.
Bob got click for photon 83 with basis A. Sent bit was 0.
Bob got click for photon 85 with basis A. Sent bit was 0.
Bob got click for photon 86 with basis V. Sent bit was 1.
Bob got click for photon 87 with basis V. Sent bit was 1.
Bob got click for photon 88 with basis A. Sent bit was 0.
Bob got click for photon 89 with basis A. Sent bit was 0.
Bob got click for photon 90 with basis V. Sent bit was 1.
Bob got click for photon 91 with basis V. Sent bit was 1.
Bob got click for photon 92 with basis A. Sent bit was 0.
Bob got click for photon 93 with basis V. Sent bit was 1.
Bob got click for photon 95 with basis V. Sent bit was 1.
Bob got click for photon 97 with basis A. Sent bit was 0.
Bob got click for photon 98 with basis V. Sent bit was 1.
Bob got click for photon 100 with basis A. Sent bit was 0.

Number of correct bits Bob recieved: 73
```

F) Lets attempt to improve our eavesdropping. This can simply be done by having Eve measure in Alice's basis instead of in Bob's basis. If she sucessfully measures in the parallel basis to the incoming state, she's gaurunteed to get a click and knows the state. If she gets a 0, then she just simply sends a random state. This will significantly decrease our error rate down to roughly 25%.

[ ]:
```python
def Eavesdrop_Improved(N_p, message, measurement_basis):
    eve_message = {}
    measurements = {}
    for i in range(1, N_p+1):
        # Create a density matrix out of our basix
        cur_basis_op = ket2dm(measurement_basis[i])

        # Measure the projected eigen value/state with the given basis
        eigen_val, eigen_state = measure(message[i], cur_basis_op)
        measurements[i] = round(eigen_val)

        # Got a click. We for sure know the state
        if measurements[i] == 1:
            if measurement_basis[i] == D:
                eve_message[i] = D
            else:
                eve_message[i] = H
```

```
        # Don't have a click, so we pick randomly
        elif measurements[i] == 0:
            eve_message[i] = random.choice([H, D])

    return eve_message
```

[ ]:
```
N_p = 1000
alice_basis = GetRandomBasis(N_p, [H,D])
alice_message = AliceMessage(N_p, alice_basis)

eve_basis = GetRandomBasis(N_p, [H,D])
eve_message = Eavesdrop_Improved(N_p, alice_message, eve_basis)

bob_basis = GetRandomBasis(N_p, [V,A])
bob_measurements = BobReceiveMessage(N_p, eve_basis, bob_basis)
```

[ ]:
```
correct_bits = GetCorrectBits(bob_measurements, bob_basis)
```

```
Bob got click for photon 05 with basis V. Sent bit was 1.
Bob got click for photon 13 with basis V. Sent bit was 1.
Bob got click for photon 20 with basis V. Sent bit was 1.
Bob got click for photon 22 with basis A. Sent bit was 0.
Bob got click for photon 23 with basis V. Sent bit was 1.
Bob got click for photon 24 with basis V. Sent bit was 1.
Bob got click for photon 26 with basis A. Sent bit was 0.
Bob got click for photon 31 with basis V. Sent bit was 1.
Bob got click for photon 32 with basis A. Sent bit was 0.
Bob got click for photon 40 with basis V. Sent bit was 1.
Bob got click for photon 41 with basis V. Sent bit was 1.
Bob got click for photon 43 with basis A. Sent bit was 0.
Bob got click for photon 45 with basis V. Sent bit was 1.
Bob got click for photon 49 with basis V. Sent bit was 1.
Bob got click for photon 58 with basis V. Sent bit was 1.
Bob got click for photon 65 with basis V. Sent bit was 1.
Bob got click for photon 73 with basis V. Sent bit was 1.
Bob got click for photon 76 with basis V. Sent bit was 1.
Bob got click for photon 77 with basis V. Sent bit was 1.
Bob got click for photon 78 with basis V. Sent bit was 1.
Bob got click for photon 81 with basis A. Sent bit was 0.
Bob got click for photon 85 with basis V. Sent bit was 1.
Bob got click for photon 89 with basis V. Sent bit was 1.
Bob got click for photon 97 with basis V. Sent bit was 1.
Bob got click for photon 100 with basis V. Sent bit was 1.

Number of correct bits Bob recieved: 25
```