Tutorial on SIC-POVMs: Probability Distributions and Entanglement Properties

Nithyasri Srivathsan[1] and Blake C. Stacey[2]

[1] *Nanyang Technological University Singapore*

[2] *QBism Research Group, University of Massachusetts Boston*

(Dated: February 12, 2023)

In this tutorial, we present a hands-on exploration of quantum information theory, specifically the representation of quantum states by probability distributions for generalized measurements. This is a rich topic, with many and surprising connections to other areas, while still being accessible to students with a moderate amount of coding experience. We find patterns in the probability distributions of selected states of interest, both pure and mixed, as we explore relationships among their various properties of entanglement like Shannon entropy, von Neumann entropy, mutual information, and concurrence using computational and graphical methodologies.

I. BACKGROUND

Now and then, stories will pop up in the news about the latest hot new thing in quantum computers. If the story makes any attempt to explain why quantum computing is special or interesting, it often recycles a remark along the lines of, "A quantum bit can be both 0 and 1 simultaneously." This is rather like saying that Boston is at both the North Pole and the South Pole simultaneously. Something important has been lost. Our goal in this introductory section is to develop a mental picture for a *qubit,* the basic unit that quantum computers are typically regarded as built out of. To be more precise, we will develop a mental picture for the *mathematics* of a qubit, not for how to implement one in the lab. There are many ways to do so, and getting into the details of any one method would, for our purposes today, be a distraction. Instead, we will be brave and face the issue on a more abstract level.

A qubit is a thing that one *prepares* and that one *measures.* The mathematics of quantum theory tells us how to represent these actions algebraically. That is, it describes the set of all possible preparations, the set of all possible measurements, and how to compute the probability of getting a particular result from a chosen measurement given a particular preparation. To do something interesting, one would typically work with multiple qubits together, but we will start with a single one. And we will begin with the simplest kind of measurement, the *binary* ones. A binary test has two possible outcomes, which we can represent as 0 or 1, "plus" or "minus", "ping" and "pong", et cetera. In the lab, this could be sending an ion through a magnetic field and registering whether it swerved up or down; or, it could be sending a blip of light through a polarizing filter turned at a certain angle and registering whether there is or is not a flash. Or any of many other possibilities! The important thing is that there are two outcomes that we can clearly distinguish from each other.

For any physical implementation of a qubit, there are three binary measurements of special interest, which we can call the X test, the Y test and the Z test. Let us denote the possible outcomes of each test by $+1$ and -1, which turns out to be a convenient choice. The *expected value* of the X test is the average of these two possibilities, weighted by the

probability of each. If we write $P(+1|X)$ for the probability of getting the $+1$ outcome given that we do the X test, and likewise for $P(-1|X)$, then this expected value is

$$x = P(+1|X) \cdot (+1) + P(-1|X) \cdot (-1). \tag{1}$$

Because this is a weighted average of $+1$ and -1, it will always be somewhere in that interval. If for example we are completely confident that an X test will return the outcome $+1$, then $x = 1$. If instead we lay even odds on the two possible outcomes, then $x = 0$. Likewise,

$$y = P(+1|Y) \cdot (+1) + P(-1|Y) \cdot (-1), \tag{2}$$

and

$$z = P(+1|Z) \cdot (+1) + P(-1|Z) \cdot (-1). \tag{3}$$

To specify the preparation of a single qubit, all we have to do is pick a value for x, a value for y and a value for z. But not all combinations (x, y, z) are physically allowed. The valid preparations are those for which the point (x, y, z) lies on or inside the ball of radius 1 centered at the origin:

$$x^2 + y^2 + z^2 \leq 1. \tag{4}$$

We call this the *Bloch ball,* after the physicist Felix Bloch (1905–1983). The surface of the Bloch ball, at the distance exactly 1 from the origin, is the *Bloch sphere.* The points where the axes intersect the Bloch sphere — $(1, 0, 0)$, $(-1, 0, 0)$, $(0, 1, 0)$ and so forth — are the preparations where we are perfectly confident in the outcome of one of our three tests. Points in the interior of the ball, not on the surface, imply uncertainty about the outcomes of all three tests. But look what happens: If Alice is perfectly confident of what will happen should she choose to do an X test, then her expected values y and z must both be zero, meaning that she is *completely uncertain* about what might happen should she choose to do either a Y test or a Z test. There is an inevitable tradeoff between levels of uncertainty, baked into the shape of the theory itself.

We are now well-poised to improve upon the language in the news stories. The point that specifies the preparation of a qubit can be at the North Pole $(0, 0, 1)$, the South Pole $(0, 0, -1)$, or anywhere in the ball between them. We have a whole continuum of ways to be intermediate between completely confident that the Z test will yield $+1$ (all the way north) and completely confident that it will yield -1 (all the way south).

Now, there are other things one can do to a qubit. For starters, there are other binary measurements beyond just the X, Y and Z tests. Any pair of points exactly opposite each other on the Bloch sphere define a test, with each point standing for an outcome. The closer the preparation point is to an outcome point, the more probable that outcome. To be more specific, let's write the preparation point as (x, y, z) and the outcome point as (x', y', z'). Then the probability of getting that outcome given that preparation is

$$P = \frac{1}{2}(1 + xx' + yy' + zz'). \tag{5}$$

An interesting conceptual thing has happened here. We have encoded the preparation of a qubit by a set of expected values, i.e., a set of probabilities. Consequently, all those philosophical debates over what probability means will spill over into the arguments about what quantum mechanics means. Moreover, and not unrelatedly, we can ask, "Why *three* probabilities? Why is it the Bloch sphere, instead of the Bloch disc or the Bloch hypersphere?" It would be perfectly legitimate, mathematically, to require probabilities for only

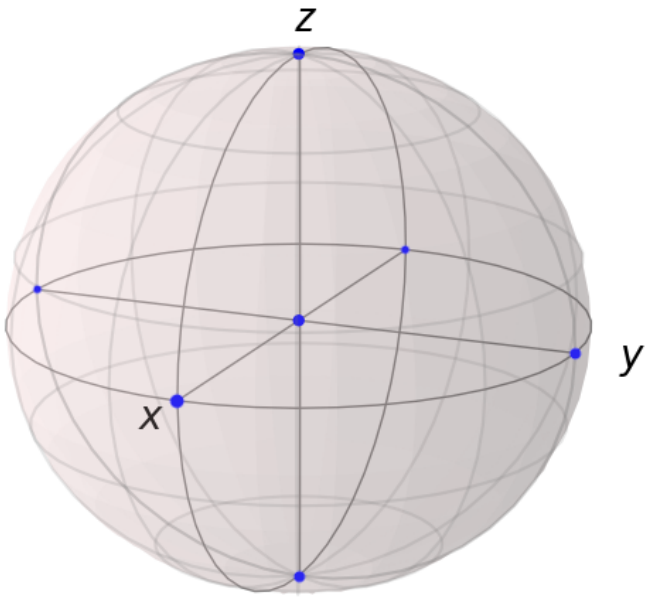


FIG. 1: Bloch ball, with the center point and the points where the axes intersect the outer sphere marked with dots

two tests in order to specify a preparation point, or to require more than three. That would not be quantum mechanics; the fact that three coordinates are needed to nail down the preparation of the simplest possible system is a structural fact of quantum theory. But is there a deeper truth from which that can be deduced?

One could go in multiple directions from here: What about tests with more than two outcomes? Systems composed of more than one qubit? Very quickly, the structures involved become more difficult to visualize, and familiarity with linear algebra — eigenvectors, eigenvalues and their friends — becomes a prerequisite. People have also tried a variety of approaches to understand what quantum theory might be derivable from. Any of those topics could justify something in between a blog post and a lifetime of study. In the following sections, we will take the path of using linear algebra to study quantum theory, with the aid of Python to automate some computations.

II. 1-QUBIT STATES

We begin by re-expressing what we covered above in more algebraic language, giving a lightning review of the quantum theory of 1-qubit systems. The preparation of such a system is encoded by a *density matrix,* a 2×2 matrix of complex numbers that satisfies three conditions. First, its trace is equal to 1; second, it is equal to its own conjugate transpose; and third, all of its eigenvalues are nonnegative. Such matrices are also known as *quantum states.* A state ρ is *pure* if $\rho = \rho^2$; otherwise, it is *mixed.*

A measurement on a qubit system is represented by a *positive operator-valued measure,* or POVM. A POVM is a set of matrices, one for each possible outcome of the measurement.

Like density matrices, each matrix in a POVM must be equal to its own conjugate transpose, and all of its eigenvalues must be nonnegative. Finally, the set of matrices making up the POVM must sum to the identity matrix. If this set is $\{E_1, \ldots, E_n\}$, for example, then

$$\sum_{j=1}^{n} E_j = I. \tag{6}$$

To calculate the probability of a particular measurement outcome given a quantum state, one uses the *Born rule*:

$$p(E_i) = \mathrm{tr}(\rho E_i). \tag{7}$$

All of the above will actually apply to quantum states of any dimension, that is, to $d \times d$ density matrices for any value of $d \geq 2$. However, focusing on $d = 2$ provides a convenient starting point. Any 1-qubit density matrix can be written in the form

$$\rho = \frac{1}{2}(I + x\sigma_x + y\sigma_y + z\sigma_z), \tag{8}$$

where σ_x, σ_y and σ_z are the Pauli matrices,

$$\sigma_x = \begin{pmatrix} 0 & 1 \\ 1 & 0 \end{pmatrix}, \sigma_y = \begin{pmatrix} 0 & -i \\ i & -1 \end{pmatrix}, \sigma_z = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix}. \tag{9}$$

The coefficients x, y and z are real numbers that obey the condition

$$x^2 + y^2 + z^2 \leq 1. \tag{10}$$

Consequently, any qubit density matrix ρ can be represented by a point (x, y, z) within a ball of radius 1, the Bloch ball we introduced earlier. Points on the surface of the Bloch ball correspond to pure states, while those on the interior correspond to mixed states.

Each of the Pauli matrices has two eigenvectors with eigenvalues ± 1. For example, the eigenvectors of σ_z are

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix}, \begin{pmatrix} 0 \\ 1 \end{pmatrix}. \tag{11}$$

We can construct the *projection operators* that project vectors onto these by taking their products with their conjugate transposes:

$$\begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 1 & 0 \end{pmatrix}^* = \begin{pmatrix} 1 & 0 \\ 0 & 0 \end{pmatrix}, \tag{12}$$

and likewise,

$$\begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} 0 & 1 \end{pmatrix}^* = \begin{pmatrix} 0 & 0 \\ 0 & 1 \end{pmatrix}. \tag{13}$$

These are both valid density matrices, with Bloch-ball coordinates $(0, 0, 1)$ and $(0, 0, -1)$ respectively. By the same logic, the eigenvectors of σ_x correspond to the points $(\pm 1, 0, 0)$, and the eigenvectors of σ_y correspond to the points $(0, \pm 1, 0)$. Together, these six points are the vertices of a regular octahedron inscribed in the Bloch ball.

Another interesting set of states is obtained by inscribing a regular tetrahedron in the Bloch ball instead. A convenient choice of coordinates is the following:

$$\pi_0 = \frac{1}{2}(I + \frac{1}{\sqrt{3}}(\sigma_x + \sigma_y + \sigma_z)), \tag{14}$$

$$\pi_1 = \frac{1}{2}(I + \frac{1}{\sqrt{3}}(\sigma_x - \sigma_y - \sigma_z)), \tag{15}$$

$$\pi_2 = \frac{1}{2}(I + \frac{1}{\sqrt{3}}(-\sigma_x + \sigma_y - \sigma_z)) \tag{16}$$

$$\pi_3 = \frac{1}{2}(I + \frac{1}{\sqrt{3}}(-\sigma_x - \sigma_y + \sigma_z)). \tag{17}$$

The sum of these four density matrices is just twice the identity matrix:

$$\sum_{i=0}^{3} \pi_i = 2I. \tag{18}$$

So, to make these quantum states into a POVM, it is necessary to scale them down by a factor of 2. This way, we get 4 operators,

$$E_i = \frac{1}{2}\pi_i, \tag{19}$$

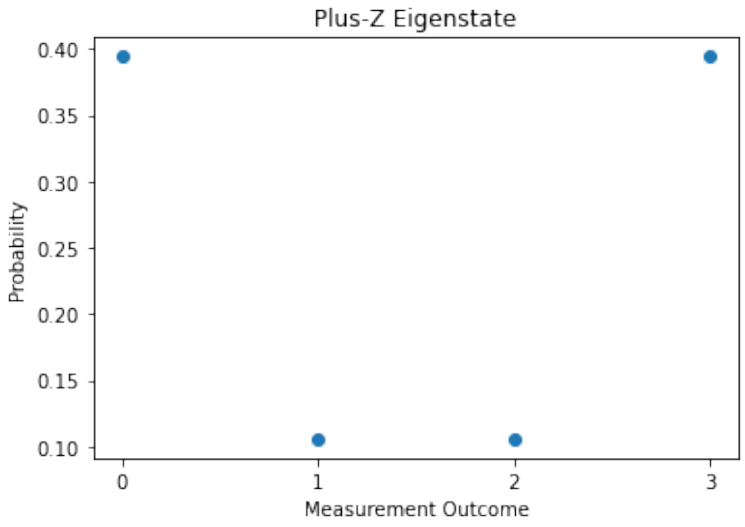
whose sum $\sum E_i$ is naturally the identity, I. The POVM is then simply the array of the 4 elements, $\{E_0, E_1, E_2, E_3\}$.

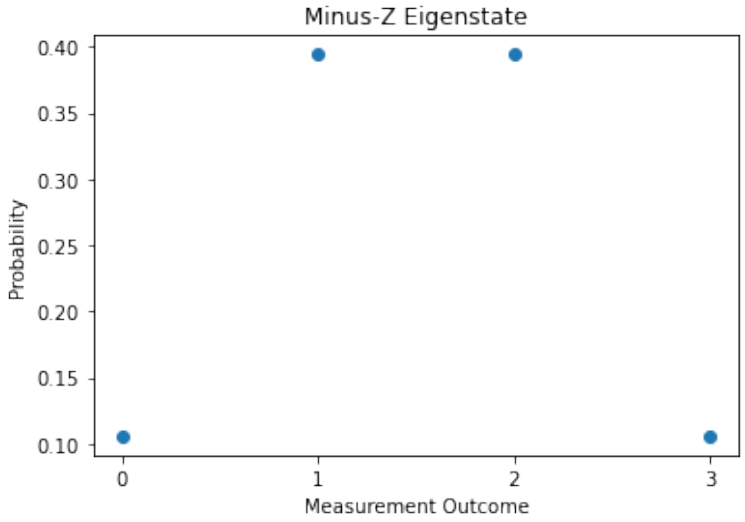
III. PROBABILITY DISTRIBUTIONS OF PAULI EIGENSTATES

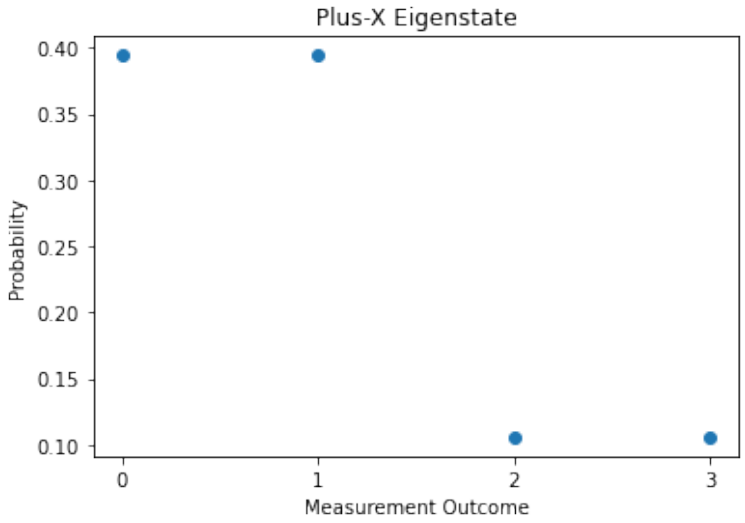
Now that we've successfully constructed the SIC-POVM, we can delve into the world of probabilistic representation that it opens up. All we need is the Born rule,

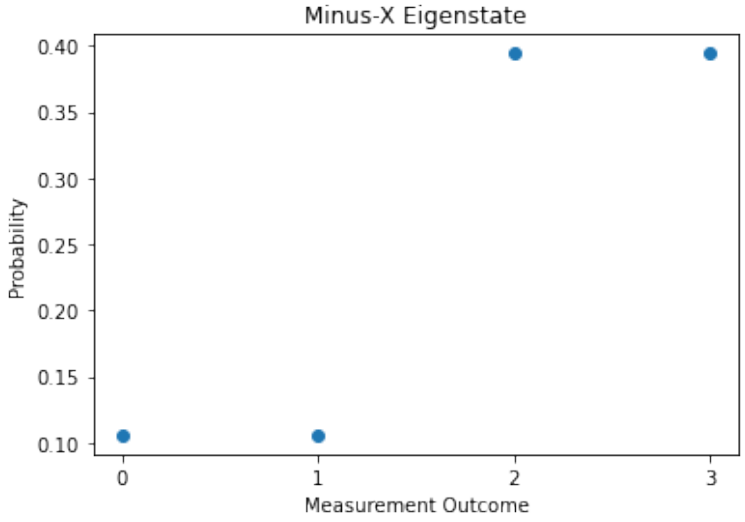
$$P_i = \mathrm{tr}(E_i \rho). \tag{20}$$

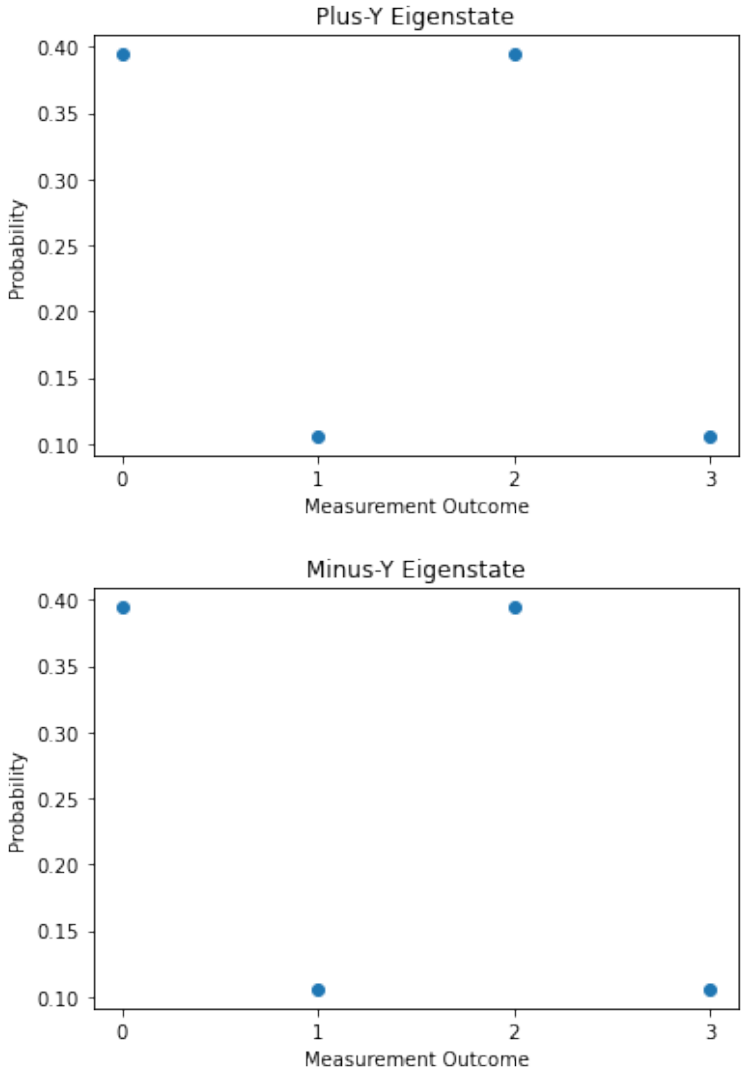
We can now construct scatter plots for all six Pauli eigenstates.




Minus-Z Eigenstate
Probability
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0
1
2
3
Measurement Outcome


Plus-X Eigenstate
Probability
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0
1
2
3
Measurement Outcome


Minus-X Eigenstate
Probability
0.40
0.35
0.30
0.25
0.20
0.15
0.10
0
1
2
3
Measurement Outcome



One thing is very evident — we observe a similar up-down pattern in their probability distributions. Each distribution seems to have only two distinct values, and all the distributions look like permutations of each other. We can get a sense for why this ought to be the case by noting that, if ρ is any Pauli eigenstate, and σ_j is any of the three Pauli matrices, then $\mathrm{tr}\rho\sigma_j$ will either be 0, +1, or −1. In the notation of the introduction, these traces-of-products are the expected values x, y and z:

$$x = \mathrm{tr}\rho\sigma_x,\ \ y = \mathrm{tr}\rho\sigma_y,\ \ z = \mathrm{tr}\rho\sigma_z. \tag{21}$$

So, the probabilities for our SIC outcomes are

$$P(0) = \frac{1}{4}(1 + \frac{1}{\sqrt{3}}(x + y + z)), \tag{22}$$

$$P(1) = \frac{1}{4}(1 + \frac{1}{\sqrt{3}}(x - y - z)), \tag{23}$$

$$P(2) = \frac{1}{4}(1 + \frac{1}{\sqrt{3}}(-x + y - z)) \tag{24}$$

$$P(3) = \frac{1}{4}(1 + \frac{1}{\sqrt{3}}(-x - y + z)). \tag{25}$$

These formulas work for any point (x, y, z) in the Bloch ball. (It's a bit of algebra to verify them, but they work out rather prettily, and the calculation is very worth doing!) When our density matrix is a Pauli eigenstate, two of the coordinates will be 0, and the other will be either $+1$ or -1.

IV. WERNER STATES

Mathematically, we can define Werner states as bipartite quantum states that do not change when an arbitrary unitary transformation is applied to each part:

$$\rho = (U \otimes U)\rho(U^\dagger \otimes U^\dagger). \tag{26}$$

Werner states are of conceptual interest, because they can exhibit quantum entanglement, yet the statistics they imply for measurement outcomes admit explanation in terms of local hidden variables. We refer to Werner's original paper for further discussion on this point [1]. In this section, we'll focus on 2-qubit Werner states, their construction, probability distribution, and their properties of entanglement.

A. Construction of Joint-POVM

In order to find a probabilistic representation of 2-qubit states, we need a 2-qubit reference POVM. There are many possibilities, but one easy choice is to build one out of a single-qubit SIC [2]. The matrices for our new POVM are tensor products of the matrices that we used before:

$$E_{ij} = E_i \otimes E_j. \tag{27}$$

B. Probability Distribution

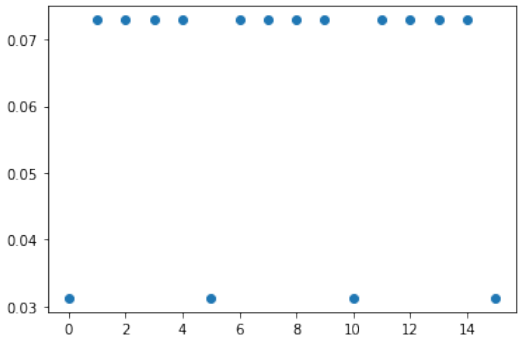
We, first, construct the Werner density matrix ρ using the formula:

$$\rho = \frac{I}{4}\lambda + (1-\lambda)\left|\psi^-\right\rangle\left\langle\psi^-\right|. \tag{28}$$

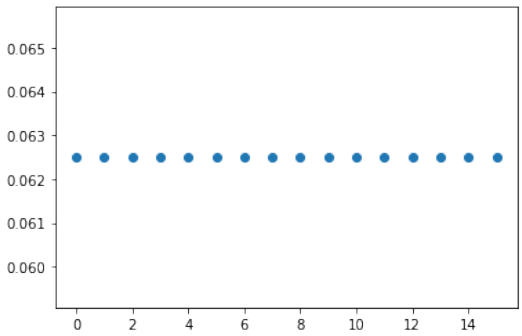
Here, we are using $|\psi^-\rangle$ to denote the Bell state

$$\left|\psi^-\right\rangle = \frac{1}{\sqrt{2}}(|z+\rangle \otimes |z-\rangle - |z-\rangle \otimes |z+\rangle). \tag{29}$$

Then, the probability distribution can then be obtained by iterating over the POVM elements and applying the Born rule. We shall now see an example case of probability distribution with the parameter $\lambda = 0$.



A similar up-down pattern is evident. We can try this for other values of λ within the interval $[0, 1]$, and we will observe a similar pattern except in the case where $\lambda = 1$, since that is the garbage state $\frac{1}{4}I$.



V. PROPERTIES OF ENTANGLEMENT

Having explored the world of probability distributions, this is a good time to stop ask ourselves if these probabilities relate to the physical properties of entanglement and further investigates plausible relationships.

A. Shannon Entropy

The first property of entanglement we shall investigate is the Shannon entropy.

Shannon entropy is a common and useful way of quantifying how "spread out" a probability distribution is. For a probability distribution $P(i)$, the Shannon entropy (also called the Shannon information and the Shannon index) is

$$H = -\sum P(i) \log P(i). \tag{30}$$

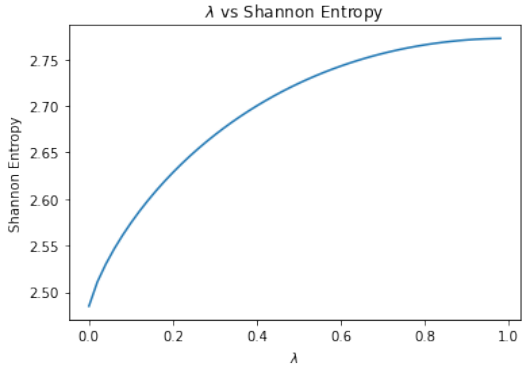
The logarithm can be taken to any base, though 2 and e are the most typical; changing the base just multiplies H by a constant. One little nuance here is the negative sign. Why include it? Well, any $P(i)$ will be a number in the interval $0 \leq P(i) \leq 1$, and so the logarithm of $P(i)$ will be negative or zero. So, putting a minus sign in front of the whole sum gives us a nonnegative final answer. (If $P(i)$ is ever zero, we declare that $P(i) \log P(i) = 0$.)

How big and small can the Shannon entropy be? For example, can the Shannon entropy ever be zero? Indeed it can. Suppose that our probability distribution is $P(1) = 1$ and

$P(i) = 0$ for all other values of i. Then every term in the sum will be zero, and so the Shannon entropy will be zero, indicating that the probability distribution is not spread out at all. What if, instead, the probability distribution is as spread out as possible? That is, suppose $P(i) = 1/n$ for all n possible values of i. Then

$$H = -\sum_{i=1}^{n} \left(\frac{1}{n}\right) \log\left(\frac{1}{n}\right) = -\log\left(\frac{1}{n}\right) = \log n. \tag{31}$$

We can apply the Shannon entropy to the probability distributions that represent Werner states by calculating H for a sequence of values of λ in the range $[0, 1]$. When $\lambda = 0$, we have the minimum value $H \approx 2.485$ and when $\lambda = 1$, we have the maximum value $H \approx 2.772$.



The parabolic relationship between λ and Shannon entropy indicates that as λ increases, the state becomes more completely mixed.

As we go from $\lambda = 0$ to $\lambda = 1$, we see a steep increase in the shannon entropy which shows that data is spreading out more and in terms of entanglement, we see that the states and getting more and more entangled, with the $\lambda = 1$ being the garbage state.

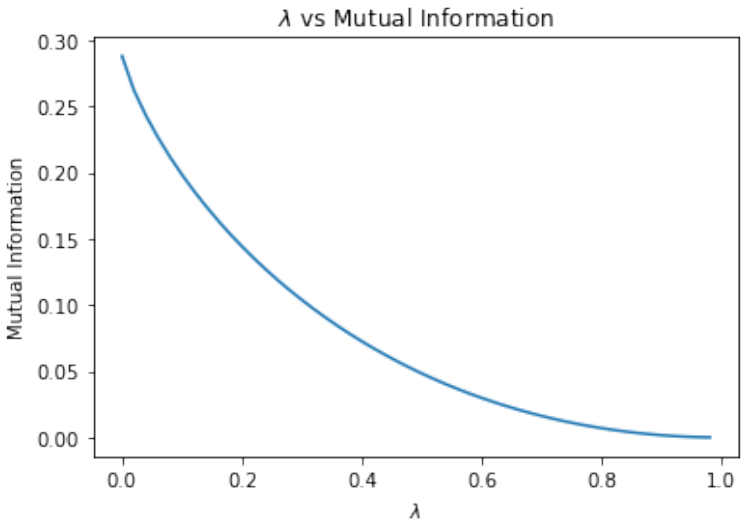
B. Mutual Information

In probability theory, mutual information is the measure of how much one random variable tells about an other. We can define it by the following formula:

$$M = \sum_{x} \sum_{y} P(x, y) \log \frac{P(x, y)}{P(x)P(y)}. \tag{32}$$

$P(x)$ and $P(y)$ are known as the *marginal* distributions of $P(x, y)$.

Like in the case of Shannon entropy, we track of the individual values of mutual information for each probability distribution for each value of λ in an array, and then we plot the results against the values of λ.



The parabolic relationship between λ and mutual information indicates that as λ increases, the result of a measurement upon one half of the two-qubit pair becomes less informative about what the result of a measurement upon the other half might be.

C. Concurrence

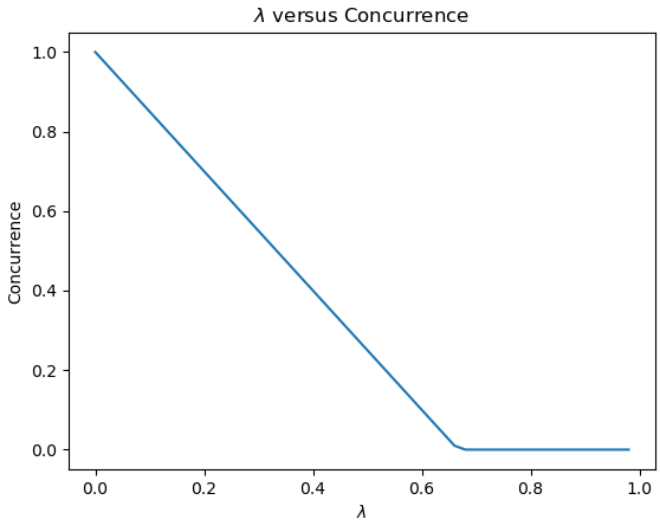
Concurrence is a measure of quantum entanglement [3]. Algorithmically, concurrence for a 2-qubit density matrix ρ can be computed by first finding the new matrix

$$\tilde{\rho} = (\sigma_y \otimes \sigma_y)\rho^*(\sigma_y \otimes \sigma_y), \tag{33}$$

where ρ^* is the matrix made by complex-conjugating each entry of ρ. Let e_i be the eigenvalues, in decreasing order, of $\tilde{\rho}$. Then the concurrence of ρ is

$$C = \max([0, \sqrt{e_3} - \sqrt{e_2} - \sqrt{e_1} - \sqrt{e_0}]). \tag{34}$$

A similar algorithm to the previous two cases can be used to create an array of values of concurrence for each value of λ which we then use to plot against λ.



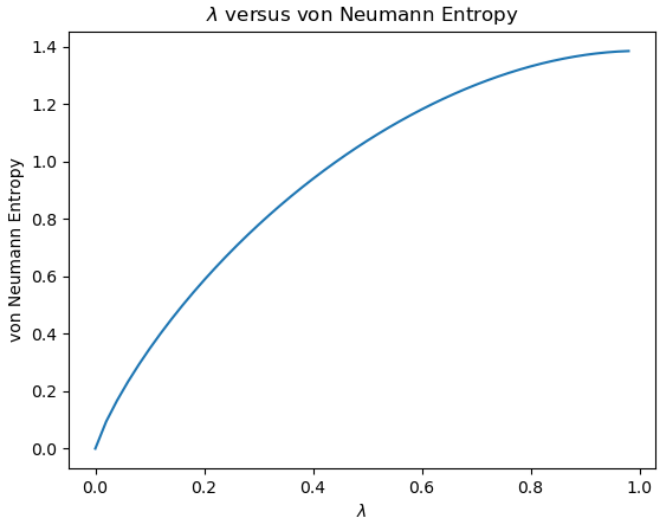
As λ increases, the states become more and more entangled. Looking carefully, we see a short lasting linear relationship between the two variables for $\lambda = [0,\ 0.02]$ which then progresses into a parabolic curve. This is in line with the fact that at $\lambda = 0$, the states are completely unentangled and are most separable at this stage. This slowly changes as we approach a value > 0.02.

D. Von Neumann Entropy

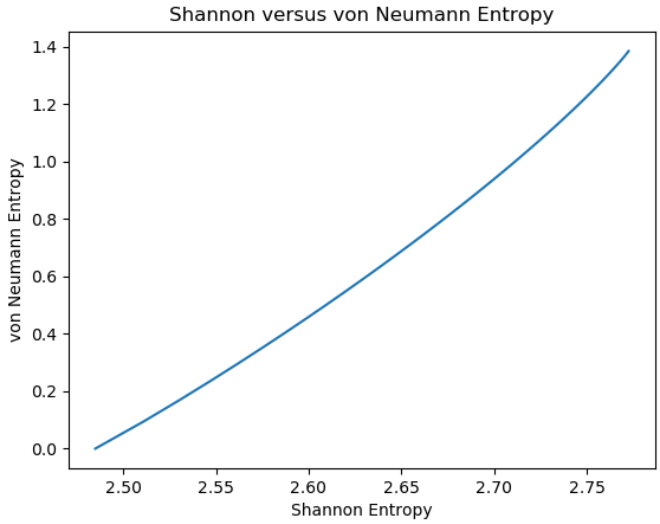
Von Neumann entropy is different from Shannon entropy in that its input is a density matrix instead of probabilities:

$$S(\rho) = -\mathrm{tr}\rho \log \rho. \tag{35}$$

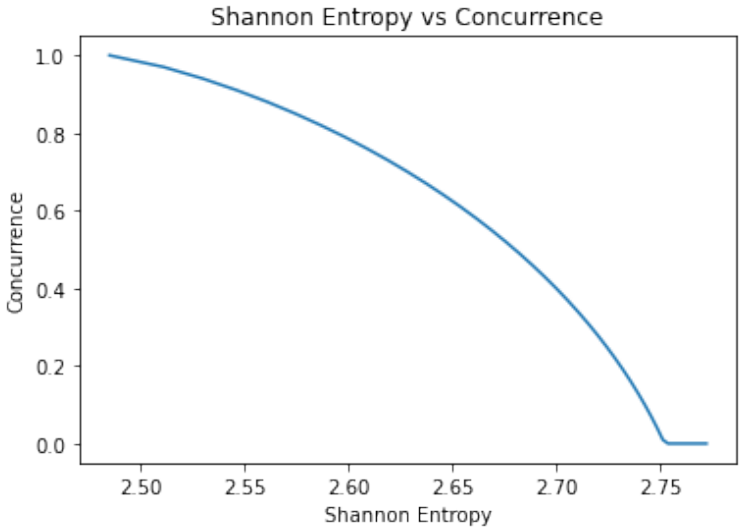
Equivalently, the von Neumann entropy can be calculated as the Shannon entropy of the eigenvalues of ρ. This suggests a question: How does the von Neumann entropy compare against the Shannon entropy of a probabilistic representation of a quantum state? We can get a taste of this by using Werner states and looping over λ as before. First, as λ increases, there's a steep increase in the value of von Neumann entropy.



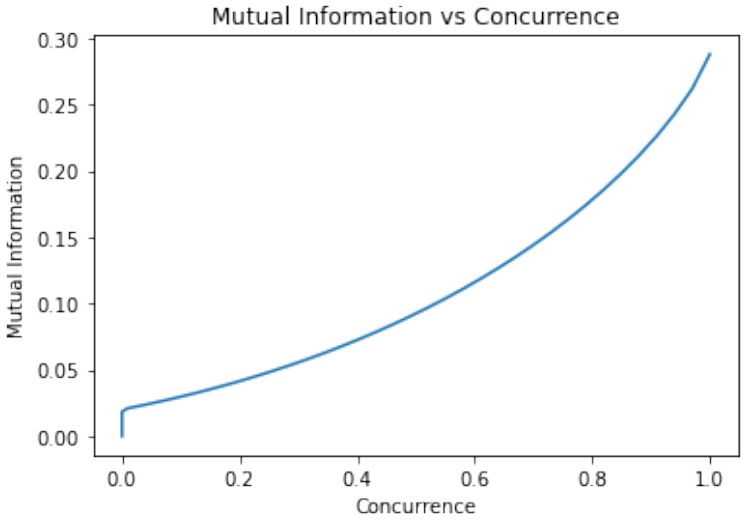
Until this point, we've investigated the various properties of entanglement and their relationships with λ. Now is a good point to think if the properties themselves are related to each other and in what way. For example, we can notice that Shannon entropy increases almost linearly with a slight steep when we plot it along von Neumann entropy.



Next, we find that as Shannon entropy increases, concurrence falls. One intricate feature of the graph to note here is the plateau in the curve from at the lower right. The Shannon entropy of the probabilistic representation of a Werner state is sensitive to changes in the λ parameter which do not register when we look at the concurrence.



Next, we can see that that mutual information and concurrence grow together over most of the range of λ. Again, we see that in the region where the concurrence is always zero, the mutual information still varies.



VI. DISCUSSION

Probabilistic representations of quantum states are a topic on the research frontier, yet interesting features of them can be computed with just a little code. Further patterns may yet arise when considering quantum states for three or more qubits, and generalizations of mutual information to higher-order relationships [4].

VII. EXAMPLE CODE

This code was written in Python version 3.8, using the NumPy 1.17 and Matplotlib 3.1 code libraries.

```
from numpy import *
from numpy.linalg import *
from pylab import *

def matrix_to_probabilities(rho, reference_POVM):
    # convert the given density matrix rho into a probability vector
    return [(R * rho).trace()[0,0].real for R in reference_POVM]

def bloch_coordinates_to_matrix(coordinates):
    # turn the given coordinates into a 2-by-2 density matrix
    return (1/2) * (I
                    + coordinates[0] * sigma_x
                    + coordinates[1] * sigma_y
                    + coordinates[2] * sigma_z)

def Shannon_entropy(probabilities):
    # find the Shannon entropy (in nats) of the given probability vector
    return -sum([p * log(p) for p in probabilities if p > 0.0])
```

```
def von_Neumann_entropy(rho):
    # calculate the von Neumann entropy of the given density matrix
    eigvals = [x.real for x in eig(rho)[0]]
    return Shannon_entropy(eigvals)

def concurrence(rho):
    # find the concurrence of the given two-qubit density matrix
    rho_star = rho.conj()
    YY = kron(sigma_y, sigma_y)
    rho_tilde = YY * rho_star * YY
    eigvals = [max([x.real, 0.0]) for x in eig(rho_tilde * rho)[0]]
    root_vals = [sqrt(x) for x in eigvals]
    root_vals.sort()
    root_vals.reverse()
    root_sum = root_vals[0] - root_vals[1] - root_vals[2] - root_vals[3]
    if root_sum > 0:
        return root_sum
    else:
        return 0.0

# define the basic matrices that we'll be working with repeatedly
I = eye(2)
sigma_x = matrix([[0, 1], [1, 0]])
sigma_y = matrix([[0, -1j], [1j, 0]])
sigma_z = matrix([[1, 0], [0, -1]])

# make a qubit SIC
SIC_coordinates = [(1, 1, 1) / sqrt(3),
                   (1, -1, -1) / sqrt(3),
                   (-1, 1, -1) / sqrt(3),
                   (-1, -1, 1) / sqrt(3)]
qubit_SIC = [(1/2) * bloch_coordinates_to_matrix(coordinates)
             for coordinates in SIC_coordinates]

# find the probabilistic representation of the Pauli eigenstates
# first, make a dictionary to pair their names with their Bloch
# coordinates
pauli_eigenstates = {"Plus-Z": (0, 0, 1),
                     "Minus-Z": (0, 0, -1),
                     "Plus-X": (1, 0, 0),
                     "Minus-X": (-1, 0, 0),
                     "Plus-Y": (0, 1, 0),
                     "Minus-Y": (0, -1, 0)}
# now, loop over them
for name, coordinates in pauli_eigenstates.items():
```

```
    rho = bloch_coordinates_to_matrix(coordinates)
    probabilities = matrix_to_probabilities(rho, qubit_SIC)
    plot(range(4), probabilities, "bo")
    xticks(range(4))
    xlabel("Measurement␣Outcome")
    ylabel("Probability")
    title(name + "␣Eigenstate")
    show()

# now, on to tensor products and entanglement
# we need a joint POVM for a reference measurement
tensorhedron_POVM = []
for i in range(4):
    for j in range(4):
        tensorhedron_POVM.append(kron(qubit_SIC[i], qubit_SIC[j]))

# let's make a Bell state
z_plus = matrix([[1], [0]])
z_minus = matrix([[0], [1]])
psi_Bell = (kron(z_plus, z_minus) - kron(z_minus, z_plus)) / sqrt(2)
rho_Bell = psi_Bell * psi_Bell.H

# ... and find its probabilistic representation
probabilities_Bell = matrix_to_probabilities(rho_Bell,
                                             tensorhedron_POVM)
plot(range(16), probabilities_Bell, "bo")
xticks(range(16))
xlabel("Measurement␣Outcome")
ylabel("Probability")
title("Bell␣State")
show()

# let's see how the Werner states look when we find their probabilistic
# representations
# first, we consider the Shannon entropy
list_of_Shannon_entropies = []
for param_lambda in arange(0, 1, 0.02):      # "lambda" is a Python keyword
    # construct the Werner state for the given parameter value
    Werner = eye(4) * (param_lambda / 4) + (1 - param_lambda) * rho_Bell
    # ... and find the Shannon entropy of its probabilistic representation
    probabilities = matrix_to_probabilities(Werner, tensorhedron_POVM)
    list_of_Shannon_entropies.append(Shannon_entropy(probabilities))
plot(arange(0, 1, 0.02), list_of_Shannon_entropies)
xlabel("$\lambda$")
ylabel("Shannon␣Entropy")
title("$\lambda$␣versus␣Shannon␣Entropy")
```

```
show()

# next, we tackle the mutual information, for which it is helpful to label
# the POVM outcomes by a pair of indices
list_of_mutual_informations = []
for param_lambda in arange(0, 1, 0.02):
    Werner = eye(4) * (param_lambda / 4) + (1 - param_lambda) * rho_Bell
    probability_dict = {}
    for i in range(4):
        for j in range(4):
            prob = (Werner * kron(qubit_SIC[i],
                                  qubit_SIC[j])).trace()[0,0].real
            if prob < 0:         # handle floating-point errors
                prob = 0
            probability_dict[(i,j)] = prob
    # evaluate the marginals
    prob_X = [sum([probability_dict[(i,j)] for j in range(4)])
              for i in range(4)]
    prob_Y = [sum([probability_dict[(i,j)] for i in range(4)])
              for j in range(4)]
    # calculate the mutual information
    mutual_information = 0.0
    for i in range(4):
        for j in range(4):
            prob = probability_dict[(i,j)]
            if prob > 0:
                mutual_information += prob * log(prob / (prob_X[i] * prob_Y[j]))
    list_of_mutual_informations.append(mutual_information)
plot(arange(0, 1, 0.02), list_of_mutual_informations)
xlabel("$\lambda$")
ylabel("Mutual Information")
title("$\lambda$ versus Mutual Information")
show()

# now, let's try relating these quantities to the concurrence
list_of_concurrences = []
for param_lambda in arange(0, 1, 0.02):
    Werner = eye(4) * (param_lambda / 4) + (1 - param_lambda) * rho_Bell
    list_of_concurrences.append(concurrence(Werner))
plot(arange(0, 1, 0.02), list_of_concurrences)
xlabel("$\lambda$")
ylabel("Concurrence")
title("$\lambda$ versus Concurrence")
show()

plot(list_of_Shannon_entropies, list_of_concurrences)
```

```
xlabel("Shannon Entropy")
ylabel("Concurrence")
title("Shannon Entropy versus Concurrence")
show()

plot(list_of_concurrences, list_of_mutual_informations)
xlabel("Concurrence")
ylabel("Mutual Information")
title("Mutual Information versus Concurrence")
show()

# and the von Neumann entropy
list_of_von_Neumann_entropies = []
for param_lambda in arange(0, 1, 0.02):
    Werner = eye(4) * (param_lambda / 4) + (1 - param_lambda) * rho_Bell
    list_of_von_Neumann_entropies.append(von_Neumann_entropy(Werner))
plot(arange(0, 1, 0.02), list_of_von_Neumann_entropies)
xlabel("$\lambda$")
ylabel("von Neumann Entropy")
title("$\lambda$ versus von Neumann Entropy")
show()

plot(list_of_Shannon_entropies, list_of_von_Neumann_entropies)
xlabel("Shannon Entropy")
ylabel("von Neumann Entropy")
title("Shannon versus von Neumann Entropy")
show()
```

---

[1] R. F. Werner, "Quantum states with Einstein-Podolsky-Rosen correlations admitting a hidden-variable model," *Physical Review A* **40** (1989), 4277–81.

[2] J. B. DeBrota, C. A. Fuchs and B. C. Stacey, "The varieties of minimal tomographically complete measurements," *International Journal of Quantum Information* **19** (2021), 204005, arXiv:1812.08762.

[3] W. K. Wootters, "Entanglement of Formation of an Arbitrary State of Two Qubits," *Physical Review Letters* **80** (1998), 2245, arXiv:quant-ph/9709029.

[4] B. C. Stacey, "Multiscale structure of more-than-binary variables," arXiv:1705.03927 (2017).