Quantum Randomness Vs Classical Randomness

OVERVIEW

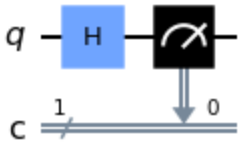
We demonstrate quantum supremacy in generating true random numbers. Classical random number generator is only ‘Pseudo Random’, in the sense these values have a hidden logic behind them, and we do recognize the fact that anything with logic can’t be called random.
There are various applications of random numbers. It is widely used in data sampling, statistics, risk analysis, cryptography and other fields.

STEPS

1. We created our own quantum random number generator
2. We run a Monte Carlo, ‘Hit or Miss’ simulation to give an example of quantum supremacy.

1. Quantum Random Number Generator

We use a simple circuit with only one qubit, and we repeat it 30 times to generate one random number. The circuit consists of putting the qubit in equal superposition through hadamard gate and then measuring it, just that.



To generate the number, each value we get (0 or 1), and the position corresponding to a number between 0 and 29 we make a binary number, then dividing by 2e9 the result will be between 0 and 0.5368709115. Finally, we run the circuit again and if the measure is 0, the number will be negative, and if it’s 1 the number will be positive.

For our experiment, we make it for 2 variables, x and y, and the process to generate the values is the same.

2. Hit Or Miss Monte Carlo Simulation

Monte Carlo simulations have wide applications. One example is, if we were to determine the number of days it would take to complete a project then we can run a Monte Carlo simulation for that. To have better results a Monte Carlo simulation depends on effective random numbers(excellent randomness). We believed quantum systems could increase the quality of randomness.

So, one good example of a Monte Carlo simulation we used to describe this better quality quantum present is ‘Hit or Miss’. The idea behind this is that the dots randomly fall within the area of a circle inscribed in a square. Does it hit the area of the circle or miss it? These chances are better predicted by quantum random numbers than classical ones.

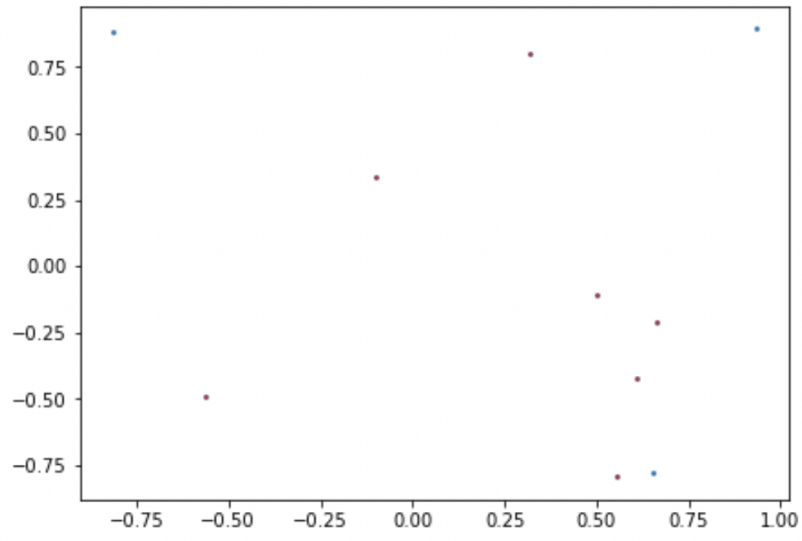
Here is the **classical** result and code :

```
import numpy
import matplotlib.pyplot as plt
```

```
#n is number of dots
n=10
#x and y coordinates radius
x=numpy.random.uniform(low=-1, high=1, size=[n,1])
y=numpy.random.uniform(low=-1, high=1, size=[n,1])

#equation of circle is used , 1 is the radius square
# r^2

inside_bool= x**2+y**2<1

#approximation

approx_pi= 4*numpy.sum(inside_bool)/n
print('Pi: {}, approximation:{}'.format(numpy.pi, approx_pi))

#toggle with n value

x_in= x[inside_bool]
y_in= y[inside_bool]

plt.figure(figsize=[7,5])
plt.scatter(x,y, s=3)
plt.scatter(x_in, y_in, color='r', s=1)
plt.show()
```

Here are the classical results for 10 dots.

```
Pi: 3.141592653589793, approximation:2.8
```



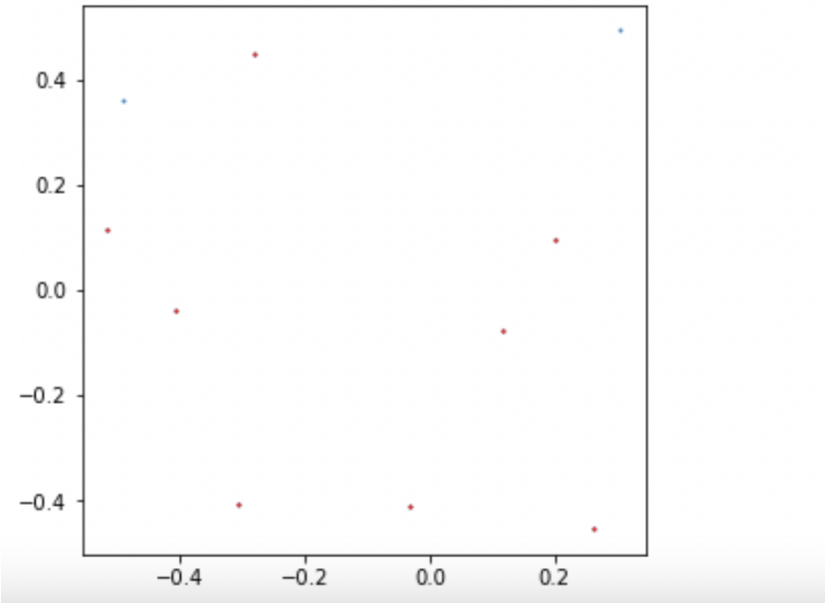
Now the **Quantum Code** :

```
a=[-0.305606019,
-0.2822788985,
0.260500853,
-0.4918202565,
-0.4081671015,
-0.0336017705,
0.303636028,
-0.516360504,
0.2004155665,
0.114934775]
b=[-0.404670965,
0.451808428,
-0.454247895,
0.359854425,
-0.0385920205,
-0.4095557895,
0.4946134635,
0.1149128235,
0.0956408665,
-0.0765905335]

x = numpy.ndarray(shape=(len(a),1), dtype=float, buffer = numpy.array(a))
y = numpy.ndarray(shape=(len(b),1), dtype=float, buffer = numpy.array(b))

#equation of circle is used

inside_bool= x**2+y**2<0.28743434
```

```
#approximation

approx_pi= 4*numpy.sum(inside_bool)/n
print('Pi: {}, approximation:{}'.format(numpy.pi, approx_pi))

#toggle with n value

x_in= x[inside_bool]
y_in= y[inside_bool]

plt.figure(figsize=[5,5])
plt.scatter(x,y, s=1)
plt.scatter(x_in, y_in, color='r', s=1)
plt.show()
```

The Quantum Result :

```
Pi: 3.141592653589793, approximation:3.2
```



We can see that the approximate for the quantum system is closer to pi value.

Conclusion

From the above results and testing, it has been shown that quantum computers are better at generating random numbers as it doesn't use any logic and depend on natural physics. Though they may be more prone to error, current work done by researchers and scientists in error current may help give more efficient and accurate results.

Contributors :

Neelanjana Anne - 301_Anne

Bartolomé Riera - 1810_Riera

Jay

Resources :

https://github.com/GINARTeam/Diehard-statistical-test

https://www.youtube.com/watch?v=X5kdy29rX1U

https://towardsdatascience.com/monte-carlo-simulations-with-python-part-1-f5627b7d60b0

https://www.geeksforgeeks.org/estimating-value-pi-using-monte-carlo/