Report 3: Event-based EPRB Experiment

Madhulan Suresh[1] **and Don Winter**[2]

[1]429481, madhulan.suresh@rwth-aachen.de
[2]431380, don.winter@rwth-aachen.de

Introduction

In this exercise we simulate the Einstein-Podolsky-Rosen-Bohm (EPRB) experiment with single photons in an event-based simulation. This experiment illustrated for the first time that "spooky action at a distance" actually exists. In other words, it provided experimental proof that two entangled photons behave non-classically by violating Bell's inequality. Our task in this exercise is to develop a computer simulation to replicate the EPRB experiment and its results.

Simulation Model and Method

In this section we first introduce the realization of the experiment in the lab, then analyze it from a theoretical viewpoint and finally elaborate on our simulation strategy.

Experiment

In the experiment, which can be seen in Fig. 1, a source emits pairs of entangled photons with random but orthogonal polarization. After arriving at the spatially separated stations $i = 1,2$, the photons pass through an electro-optical modulator which rotates the photon polarization by a discrete angle ϕ_j between 0 and 2π in 32 possible steps. From there two individual random number generators (RNGs) are used to project the photon to one of two possible states according to Malus' law. This state further determines which of two paths the photon will take to finally arrive at one of the detectors $D_{\pm,i}$, where it generates a signal $x_{n,i} = \pm 1$ for the nth generated photon pairs. Each station has an individual stopwatch which assigns a time tag to each generated signal. The coincidences are then counted within a coincidence window $W = 1ns$, i.e. the difference in arrival time of two photons $\tau = |t_{n,2} - t_{n,1}|$ must fall within the time window $\tau \leq W$. If this is not the case, we label the two photons as not coinciding. In general τ can be in the range $0 < \tau < T_0$ where $T_0 = 1\mu s$.

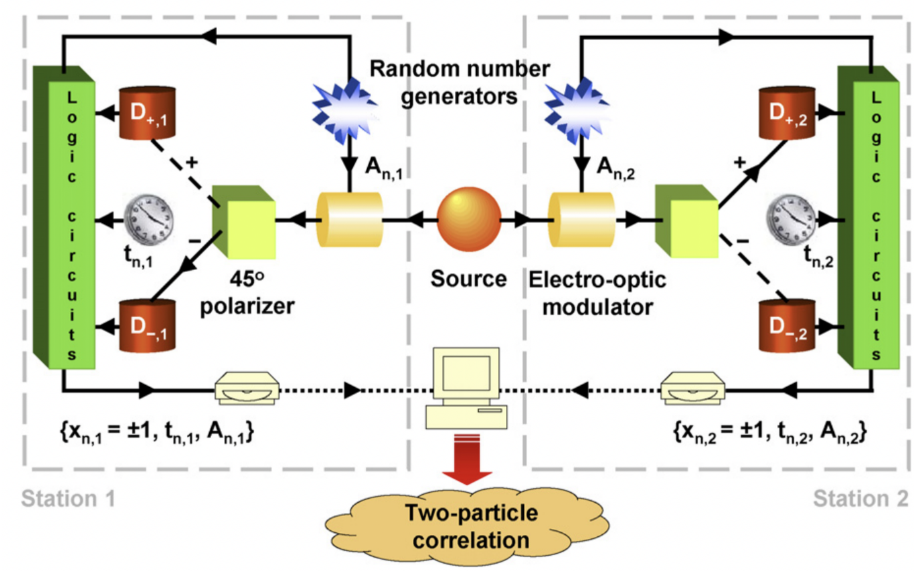

Figure 1. Schematic diagram of the EPRB experiment

Theory

The source produces two entangled photons in opposite directions. This we can describe by two vectors $\vec{S}_{n,1}$ and $\vec{S}_{n,2}$,

$$\vec{S}_{n,i} = [\cos(\xi_n + (i-1)\pi/2), \sin(\xi_n + (i-1)\pi/2)], \tag{1}$$

which represent the polarizations of the two photons incident on station 1 and 2. The angle ξ_n here can vary in a range between 0 and 2π. We use a RNG to randomly generate the value of $\xi_n = 2\pi r$ where $r = [0,1)$ in each event n to model the unpredictability of the polarizations of photons that the source produces. Inside the ith observation station the polarization vector of an incomming photon is rotated by an angle $\phi_j = 2\pi j/32$ from a discrete set of angles $\{\phi_j\}$. Furthermore, we use two uniformly drawn random numbers $a(a')$ and $b(b')$ to model the randomness in state projection according to Malus law, as mentioned before, obtaining the parameter α:

$$\alpha_{n,1} = \xi_n - [a(1+A_{n,1})/2 + (1-A_{n,1})a^{'}/2], \alpha_{n,1} = \xi_n - [b(1+A_{n,2})/2 + (1-A_{n,2})b^{'}/2] + \pi/2. \tag{2}$$

If $\cos 2\alpha > 1/2$ the photon causes $D_{+1,i}$ to click, otherwise $D_{-1,i}$ clicks. Thus, the detection of the photon generates the data $x_{n,i} = sign(\cos 2\alpha)$.

Similarly, we model the arrival as a photon, i.e. a detection event, as a random process caused by retardation of the photons through the waveplates. In our model we assume the the time delay $t_{n,i}$ is uniformly distributed within $[t_0, t_0 + T]$, thus:

$$t_{n,i} = T_0 r^{'} \sin^4 2\alpha_{n,i}, \tag{3}$$

where r' is the random variable. During the simulation we count every event in the corresponding bin, as well as the coincidence of two photons in a bin, i.e. when they arrive within the time window W at opposite detectors. We thus generate the dataset

$$\Upsilon_{N_i,i} = \{x_{n,i} = \pm 1, t_{n,i}, A_{n,i} = \pm 1 | n = 1, ..., N_i\}. \tag{4}$$

The time tag $t_{n,i}$ and time window, $t_{n,i} \leq W$ is used for counting the coincidences. To compute the coincidence, we use the given function

$$C_{xy}(a,b) = \sum_{n=1}^{N_1} \sum_{m=1}^{N_2} \delta_{x,x_{n,1}} \delta_{y,x_{m,2}} \delta_{a,\theta_{n,1}} \delta_{b,\theta_{m,2}} \Theta(W - |t_{n,1} - t_{m,2}|), \tag{5}$$

where $\Theta(t)$ is the Heaviside step function. Then we find the expectation value by taking the average of C_{xy} over each detection count $E_1(a)$ and $E_2(b)$, as well as the correlation $E_{12}(a,b)$. The values for the coincidences $C_{xy}(a,b)$ and correlation $E_{12}(a,b)$ depend on the time-tag resolution τ and the time window W used to identify the coincidences.

Simulation Results

After the simulation we obtain the generated data sets $\{\Upsilon_1, \Upsilon_2\}$. The coincidences depends on the the time tag $\tau \in [0, T_0]$, where $T_0 = 1000$ns, the time window $W = 1$ns and number of events $N = 10^6$. Fig. 2 shows a scatter plot of the expectation values of the correlations($E_{12}(a,b)$ without a time window (orange dots) and with a coincidence time window (blue dots). Fig. 3 shows a scatter plot of the product of expectation values of the individual detection events $E_1(a)E_2(b)$. The quantum theoretical

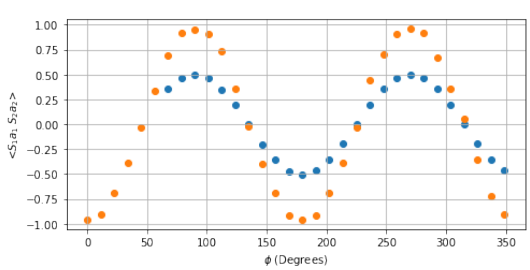

Figure 2. Plot of $E_{12}(a,b)$, orange dots-without time window and Blue dots-with time coincidence

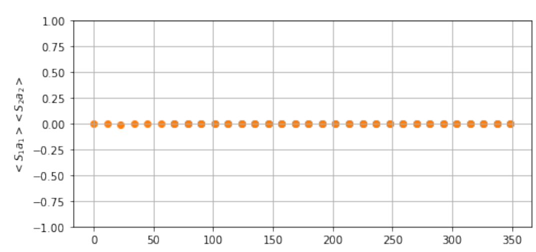

Figure 3. Plot of $E_1(a)E_2(b)$ as function of modulator angle.

description of the EPRB experiment predicts that $E_1(a) = 0$, $E_2(b) = 0$ and $E_{12}(a,b) = -\cos 2(a-b)$. From the figure, we can clearly see that both the plots are in excellent agreement with quantum theory. Bell's inequality states that

$$|S(a,a^{'},b,b^{'})| = |E(a,b) + E(a,b^{'}) - E(a^{'},b) + E(a^{'},b^{'})| \leq 2, \tag{6}$$

i.e. when this inequality is violated, we can't describe the particle correlations via a product state. So when making the time window $W \to \infty$, we can discard the time tag data from the coincidences to satisfy the inequality. Thus if the time window is small $W \to 0$, we get a state which is indistinguishable from the singlet state. Conversely, when $W \to \infty$ the correlations statisfy the inequality. We can see this difference in Fig. 2. When we introduced the time window, our model reproduces the correlation of a quantum state in the singlet state.

Discussion

Our EPRB experiment simulation produces data sets $\{\Upsilon_1, \Upsilon_2\}$, in accordance with Einstein's condition of local causality. The generated correlations $E(a,b) = -\cos 2(a-b)$ are characteristic for a maximally entangled quantum singlet state. The essential property of our simulation is that we use RNGs to arrive at expectation values instead of true quantum probability, as in the lab. Also, it paints a simple and realistic picture of the mechanism to reproduce the correlations which agree with the quantum state in the most entangled, i.e. singlet state.

Appendix

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

np.random.seed(9481)

count=np.zeros((2,2,2,361), dtype=int) # initialize counting array
tot=np.zeros((2,361), dtype=int)
E1=np.zeros((4,361))
E2=np.zeros((4,361))
E12=np.zeros((4,361))

n_steps=32
N_samples=100_000
HWP2=0    # intital angle of orientation of wave plate
T0=1_000  # maximum time delay
W=1  # time coincidence window in ns

cHWP2=np.cos(HWP2*np.pi/180)
sHWP2=np.sin(HWP2*np.pi/180)

def Analyzer(c, s, cHWP, sHWP, T0):
    c2=cHWP*c +sHWP*s  # the plane of polarization rotation
    s2= -sHWP*c+cHWP*s
    x=c2**2-s2**2
    y=2*c2*s2

    r=np.random.random() # Malus Law
    if x>2*r-1:
        j=0
    else:
        j=1
    r=np.random.random() # Fixing the time delay
    l=y**4 *T0*r
    return j,l

for ipsi0 in range(0,n_steps):
    cHWP1=np.cos(ipsi0*2*np.pi/n_steps)
    sHWP1=np.sin(ipsi0*2*np.pi/n_steps)
    for i in range(0, N_samples):
        r=np.random.random()
        c1=np.cos(r*2*np.pi)      # particle's polarization angles
        s1 = np.sin(r*2*np.pi)
        c2= -s1
        s2 = c1

        j1, l1 = Analyzer(c1, s1, cHWP1, sHWP1, T0)
        j2, l2 = Analyzer(c2, s2, cHWP2, sHWP2, T0)
        count[j1, j2, 0, ipsi0] += 1  # Malus Law without time window
        if(np.abs(l1 - l2) < W):  # Malus Law with time window
            count[j1, j2, 1, ipsi0] += 1
```

```python


for j in range(n_steps):  # Without time window, i=0 and with time coincidences, i=1
    for i in range(2):
        tot[i,j] = np.sum(count[:,:,i,j])
        E12[i,j] = count[0,0,i,j] + count[1,1,i,j] - count[1,0,i,j] - count[0,1,i,j] # Finding the expectation value of coincidence
        E1[i,j] = count[0,0,i,j] + count[0,1,i,j] - count[1,1,i,j] - count[1,0,i,j] # Expectation value of individual detection
        E2[i,j] = count[0,0,i,j] + count[1,0,i,j] - count[1,1,i,j] - count[0,1,i,j]
        if tot[i,j] > 0:
            E12[i,j] = E12[i,j] / tot[i,j]
            E1[i,j] = E1[i,j] / tot[i,j]
            E2[i,j] = E2[i,j] / tot[i,j]

for i in [0,1]: # Creating the csv file for plotting
    if i==0:
        f=open('eprb_no_time.csv', 'w')
    else:
        f=open('eprb_with_time.csv', 'w')

    f.write("Input Variables:\n" )
    f.write("NSamples = {}\n".format(N_samples))
    f.write( "HWP2 = {}\n".format(HWP2))
    f.write( "T0 = {}\n".format(T0))
    f.write("W = {}\n".format(W))
    f.write("iseed = 6247\n")
    f.write("Theta, S1.a, S2.b, S1.a_S2.b, theoryS, N++, N-+, N+-, N--, sum(N..)\n")
    for j in range(0, n_steps):
        S = 3*E12[i, j] - E12[1, j*3 % n_steps]
        r0=-np.cos(4*j*np.pi/n_steps)
        f.write ("{}, {}, {}, {}, {}, {}, {}, {}, {}, {}\n".format(j*360.0/n_steps, E1[i,j], E2[1,j], E12[i, j],
            r0, S, count[0,0,i,j], count[0,1,i, j], count[1, 0, i, j], count[1,1,1, j], tot[i, j]))

    f.close()

# Plotting the expectation values against the angle.
no_time = np.genfromtxt('eprb_no_time.csv', delimiter=',', skip_header=7, unpack=True)
with_time = np.genfromtxt('eprb_with_time.csv', delimiter=',', skip_header=7, unpack=True)

plt.scatter(no_time[0], no_time[3])
plt.scatter(with_time[0], with_time[3])
plt.xlabel(r'$\phi$ (Degrees)')
plt.ylabel(r'<$S_1a_1$ $S_2a_2$>')
plt.grid()
plt.show()

plt.scatter(no_time[0], no_time[1]*no_time[2])
plt.scatter(with_time[0], with_time[1]*with_time[2])
plt.ylim(-1, 1)
plt.xlabel(r'$\phi$ (Degrees)')
plt.ylabel(r'$<S_1a_1><S_2a_2>$')
plt.grid()
plt.show()
```