Computer simulation of Bell test experiment based on a classical wave model

Alvydas Jakeliunas
allvydas@gmail.com

Abstract

Here we present computer simulation of Bell test experiment
based on classical wave model of light.

The program consists of six modules. Each module is independent and can be executed separately. Independence of the modules is the exclusive property of this simulation which separates it from all known simulations like [2] based on a photon model of light. Especially important to note full independence between Alice's and Bob's modules.

Despite widely spread doubts about the possibility of making such a type of computer simulation [1] we get results that well agree with QM predictions and with data of real experiments [3][4].

Modules of the program:
Module N1 generates two files of light polarization.
Module N2 generates random numbers for Alice's polarizer.
Module N3 generates random numbers for Bob's polarizer.
Module N4 simulates Alice's measurement and generates output files.
Module N5 simulates Bob's measurement and generates output files.
Module N6 counts coincidence events and writes output.

Here is typical output from module N6:

```
-------------------------------------
E1 = 0.686508
E2 = -0.667946
E3 = 0.702381
E4 = 0.682156

E = E1 - E2 + E3 + E4 = 2.738991
-------------------------------------
```

The program is written in Python 3.xx. For testing, each module could be pasted into the separate .py file.

Module N1

```
# generates two files of light polarization
import random
import math

number_of_waves = 1000000

f_Alice = open('lightForAlice.txt', 'w')
f_Bob = open('lightForBob.txt', 'w')

for i in range(0, number_of_waves):
    polarization_forAlice = 2 * math.pi * random.random()
    polarization_forBob = polarization_forAlice - math.pi /2.0

    f_Alice.write('%f\n' % polarization_forAlice)
    f_Bob.write('%f\n' % polarization_forBob)

f_Alice.close()
f_Bob.close()

print("File 'lightForAlice.txt' is done")
print("File 'lightForBob.txt' is done")
```

Module N2

```
# random numbers for Alice
import random

number_of_waves = 1000000

f = open('randomNumbersForAlice.txt', 'w')

for i in range(0, number_of_waves):
    a_number = random.randrange(0, 2)
    f.write('%i\n' % a_number)

f.close()

print("File 'randomNumbersForAlice.txt' is done")
```

Module N3

```
# random numbers for Bob
import random

number_of_waves = 1000000
```

```
f = open('randomNumbersForBob.txt', 'w')

for i in range(0, number_of_waves):
    a_number = random.randrange(0, 2)
    f.write('%i\n' % a_number)

f.close()

print("File 'randomNumbersForBob.txt' is done")
```

Module N4

```
# Alice
import math

number_of_waves = 1000000
# Let minimum_energy_for_detection >> 1
minimum_energy_for_detection = 10.0

f_wave = open('lightForAlice.txt','r')
f_random = open('randomNumbersForAlice.txt', 'r')

f_output_D1 = open ('AliceOutput_D1.txt','w')
f_output_D2 = open ('AliceOutput_D2.txt','w')
f_output_polarizer = open ('AliceOutput_polarizer.txt','w')

# initial setup for detectors and polarizer
energy_at_detector_Alice_D1 = 0.0
energy_at_detector_Alice_D2 = 0.0

random_forAlice = int(f_random.readline().strip('\n'))

if random_forAlice == 0:
    angle_polarizer = 0.0
else:
     angle_polarizer = math.pi / 4.0


# run waves
for i in range(0, number_of_waves):

    angle_wave = float(f_wave.readline().strip('\n'))
    angle_wave_polarizer = angle_polarizer - angle_wave

    # add new portions of energy to detectors according to Malus law
    energy_for_Alice_D1 = math.cos(angle_wave_polarizer) ** 2
    energy_for_Alice_D2 = 1 - energy_for_Alice_D1
```

```
    energy_at_detector_Alice_D1 += energy_for_Alice_D1
    energy_at_detector_Alice_D2 += energy_for_Alice_D2

    if energy_at_detector_Alice_D1 > minimum_energy_for_detection:
        # detected "click" at detector D1
        f_output_D1.write('1\n')
    else:
        f_output_D1.write('0\n')

    if energy_at_detector_Alice_D2 > minimum_energy_for_detection:
        # detected "click" at detector D2
        f_output_D2.write('1\n')
    else:
        f_output_D2.write('0\n')

    f_output_polarizer.write('%i\n' % random_forAlice)

    if (energy_at_detector_Alice_D1 > minimum_energy_for_detection or
energy_at_detector_Alice_D2 > minimum_energy_for_detection):
        # reset polarizer and detectors
        random_forAlice = int(f_random.readline().strip('\n'))
        if random_forAlice == 0:
            angle_polarizer = 0.0
        else:
            angle_polarizer = math.pi / 4.0

        energy_at_detector_Alice_D1 = 0.0
        energy_at_detector_Alice_D2 = 0.0

f_wave.close()
f_random.close()
f_output_D1.close()
f_output_D2.close()
f_output_polarizer.close()

print("Alice's files for Counter are done")
```

Module N5

```
# Bob
import math

number_of_waves = 1000000
# Let minimum_energy_for_detection >> 1
minimum_energy_for_detection = 10.0

f_wave = open('lightForBob.txt','r')
```

```
f_random = open('randomNumbersForBob.txt', 'r')

f_output_D1 = open ('BobOutput_D1.txt','w')
f_output_D2 = open ('BobOutput_D2.txt','w')
f_output_polarizer = open ('BobOutput_polarizer.txt','w')

# initial setup for detectors and polarizer
energy_at_detector_Bob_D1 = 0.0
energy_at_detector_Bob_D2 = 0.0

random_forBob = int(f_random.readline().strip('\n'))

if random_forBob == 0:
   angle_polarizer = math.pi / 8.0
else:
    angle_polarizer = 3 * math.pi / 8.0


# run waves
for i in range(0, number_of_waves):

   angle_wave = float(f_wave.readline().strip('\n'))
   angle_wave_polarizer = angle_polarizer - angle_wave

   # add new portions of energy to detectors according to Malus law
   energy_for_Bob_D1 = math.cos(angle_wave_polarizer) **2
   energy_for_Bob_D2 = 1 - energy_for_Bob_D1

   energy_at_detector_Bob_D1 += energy_for_Bob_D1
   energy_at_detector_Bob_D2 += energy_for_Bob_D2

   if energy_at_detector_Bob_D1 > minimum_energy_for_detection:
      # detected "click" at detector D1
      f_output_D1.write('1\n')
   else:
      f_output_D1.write('0\n')

   if energy_at_detector_Bob_D2 > minimum_energy_for_detection:
      # detected "click" at detector D2
      f_output_D2.write('1\n')
   else:
      f_output_D2.write('0\n')

   f_output_polarizer.write('%i\n' % random_forBob)

   if (energy_at_detector_Bob_D1 > minimum_energy_for_detection or
energy_at_detector_Bob_D2 > minimum_energy_for_detection):
      # reset polarizer and detectors
      random_forBob = int(f_random.readline().strip('\n'))
```

```
        if random_forBob == 0:
            angle_polarizer = math.pi / 8.0
        else:
            angle_polarizer = 3 * math.pi / 8.0

        energy_at_detector_Bob_D1 = 0.0
        energy_at_detector_Bob_D2 = 0.0

f_wave.close()
f_random.close()
f_output_D1.close()
f_output_D2.close()
f_output_polarizer.close()

print("Bob's files for Counter are done")
```

Module N6

```
# Counter
number_of_waves = 1000000

f_Alice_D1 = open('AliceOutput_D1.txt','r')
f_Alice_D2 = open ('AliceOutput_D2.txt','r')
f_Alice_polarizer = open ('AliceOutput_polarizer.txt','r')

f_Bob_D1 = open('BobOutput_D1.txt','r')
f_Bob_D2 = open ('BobOutput_D2.txt','r')
f_Bob_polarizer = open ('BobOutput_polarizer.txt','r')

# coincedence for angles of setup 0 (Alice) and pi/8 (Bob)
coincedenceAlice1Bob1_angles_0_and_pi_by_8 = 0
coincedenceAlice1Bob2_angles_0_and_pi_by_8 = 0
coincedenceAlice2Bob1_angles_0_and_pi_by_8 = 0
coincedenceAlice2Bob2_angles_0_and_pi_by_8 = 0

# coincedence for angles of setup 0 (Alice) and 3*pi/8 (Bob)
coincedenceAlice1Bob1_angles_0_and_3pi_by_8 = 0
coincedenceAlice1Bob2_angles_0_and_3pi_by_8 = 0
coincedenceAlice2Bob1_angles_0_and_3pi_by_8 = 0
coincedenceAlice2Bob2_angles_0_and_3pi_by_8 = 0

#  coincedence for angles of setup pi/4 (Alice) and pi/8 (Bob)
coincedenceAlice1Bob1_angles_pi_by_4_and_pi_by_8 = 0
coincedenceAlice1Bob2_angles_pi_by_4_and_pi_by_8 = 0
coincedenceAlice2Bob1_angles_pi_by_4_and_pi_by_8 = 0
coincedenceAlice2Bob2_angles_pi_by_4_and_pi_by_8 = 0

# coincedence for angles of setup pi/4 (Alice) and 3*pi/8 (Bob)
```

```
coincedenceAlice1Bob1_angles_pi_by_4_and_3pi_by_8 = 0
coincedenceAlice1Bob2_angles_pi_by_4_and_3pi_by_8 = 0
coincedenceAlice2Bob1_angles_pi_by_4_and_3pi_by_8 = 0
coincedenceAlice2Bob2_angles_pi_by_4_and_3pi_by_8 = 0


for i in range(0, number_of_waves):
    n_Alice_D1 = int(f_Alice_D1.readline().strip('\n'))
    n_Alice_D2 = int(f_Alice_D2.readline().strip('\n'))
    n_Alice_polarizer = int(f_Alice_polarizer.readline().strip('\n'))

    n_Bob_D1 = int(f_Bob_D1.readline().strip('\n'))
    n_Bob_D2 = int(f_Bob_D2.readline().strip('\n'))
    n_Bob_polarizer = int(f_Bob_polarizer.readline().strip('\n'))

    if n_Alice_polarizer == 0 and n_Bob_polarizer == 0:
        if n_Alice_D1 == 1 and n_Bob_D1 == 1:
            coincedenceAlice1Bob1_angles_0_and_pi_by_8 +=1
        if n_Alice_D1 == 1 and n_Bob_D2 == 1:
            coincedenceAlice1Bob2_angles_0_and_pi_by_8 +=1
        if n_Alice_D2 == 1 and n_Bob_D1 == 1:
            coincedenceAlice2Bob1_angles_0_and_pi_by_8 +=1
        if n_Alice_D2 == 1 and n_Bob_D2 == 1:
            coincedenceAlice2Bob2_angles_0_and_pi_by_8 +=1

    if n_Alice_polarizer == 0 and n_Bob_polarizer == 1:
        if n_Alice_D1 == 1 and n_Bob_D1 == 1:
            coincedenceAlice1Bob1_angles_0_and_3pi_by_8 +=1
        if n_Alice_D1 == 1 and n_Bob_D2 == 1:
            coincedenceAlice1Bob2_angles_0_and_3pi_by_8 +=1
        if n_Alice_D2 == 1 and n_Bob_D1 == 1:
            coincedenceAlice2Bob1_angles_0_and_3pi_by_8 +=1
        if n_Alice_D2 == 1 and n_Bob_D2 == 1:
            coincedenceAlice2Bob2_angles_0_and_3pi_by_8 +=1

    if n_Alice_polarizer == 1 and n_Bob_polarizer == 0:
        if n_Alice_D1 == 1 and n_Bob_D1 == 1:
            coincedenceAlice1Bob1_angles_pi_by_4_and_pi_by_8 +=1
        if n_Alice_D1 == 1 and n_Bob_D2 == 1:
            coincedenceAlice1Bob2_angles_pi_by_4_and_pi_by_8 +=1
        if n_Alice_D2 == 1 and n_Bob_D1 == 1:
            coincedenceAlice2Bob1_angles_pi_by_4_and_pi_by_8 +=1
        if n_Alice_D2 == 1 and n_Bob_D2 == 1:
            coincedenceAlice2Bob2_angles_pi_by_4_and_pi_by_8 +=1

    if n_Alice_polarizer == 1 and n_Bob_polarizer == 1:
        if n_Alice_D1 == 1 and n_Bob_D1 == 1:
            coincedenceAlice1Bob1_angles_pi_by_4_and_3pi_by_8 +=1
        if n_Alice_D1 == 1 and n_Bob_D2 == 1:
```

```
            coincedenceAlice1Bob2_angles_pi_by_4_and_3pi_by_8 +=1
        if n_Alice_D2 == 1 and n_Bob_D1 == 1:
            coincedenceAlice2Bob1_angles_pi_by_4_and_3pi_by_8 +=1
        if n_Alice_D2 == 1 and n_Bob_D2 == 1:
            coincedenceAlice2Bob2_angles_pi_by_4_and_3pi_by_8 +=1

f_Alice_D1.close()
f_Alice_D2.close()
f_Alice_polarizer.close()

f_Bob_D1.close()
f_Bob_D2.close()
f_Bob_polarizer.close()

# E1, E2, E3, E4; values for Bell's theorem

# for setup angles 0 (Alice) and pi/8 (Bob)
E1 = ( (coincedenceAlice1Bob2_angles_0_and_pi_by_8
    + coincedenceAlice2Bob1_angles_0_and_pi_by_8
    - coincedenceAlice1Bob1_angles_0_and_pi_by_8
    - coincedenceAlice2Bob2_angles_0_and_pi_by_8) /
    (coincedenceAlice1Bob1_angles_0_and_pi_by_8
    + coincedenceAlice2Bob2_angles_0_and_pi_by_8
    + coincedenceAlice1Bob2_angles_0_and_pi_by_8
    + coincedenceAlice2Bob1_angles_0_and_pi_by_8) )

# for setup angles 0 (Alice) and 3*pi/8 (Bob)
E2 = ( (coincedenceAlice1Bob2_angles_0_and_3pi_by_8
    + coincedenceAlice2Bob1_angles_0_and_3pi_by_8
    - coincedenceAlice1Bob1_angles_0_and_3pi_by_8
    - coincedenceAlice2Bob2_angles_0_and_3pi_by_8) /
    (coincedenceAlice1Bob1_angles_0_and_3pi_by_8
    + coincedenceAlice2Bob2_angles_0_and_3pi_by_8
    + coincedenceAlice1Bob2_angles_0_and_3pi_by_8
    + coincedenceAlice2Bob1_angles_0_and_3pi_by_8) )

# for setup angles pi/4 (Alice) and pi/8 (Bob)
E3 = ( (coincedenceAlice1Bob2_angles_pi_by_4_and_pi_by_8
    + coincedenceAlice2Bob1_angles_0_and_pi_by_8
    - coincedenceAlice1Bob1_angles_pi_by_4_and_pi_by_8
    - coincedenceAlice2Bob2_angles_0_and_pi_by_8) /
    (coincedenceAlice1Bob1_angles_pi_by_4_and_pi_by_8
    + coincedenceAlice2Bob2_angles_0_and_pi_by_8
    + coincedenceAlice1Bob2_angles_0_and_pi_by_8
    + coincedenceAlice2Bob1_angles_0_and_pi_by_8) )

# for setup angles pi/4 (Alice) and 3*pi/8 (Bob)
E4 = ( (coincedenceAlice1Bob2_angles_pi_by_4_and_3pi_by_8
    + coincedenceAlice2Bob1_angles_pi_by_4_and_3pi_by_8
```

```
    - coincedenceAlice1Bob1_angles_pi_by_4_and_3pi_by_8
    - coincedenceAlice2Bob2_angles_pi_by_4_and_3pi_by_8) /
    (coincedenceAlice1Bob1_angles_pi_by_4_and_3pi_by_8
    + coincedenceAlice2Bob2_angles_pi_by_4_and_3pi_by_8
    + coincedenceAlice1Bob2_angles_pi_by_4_and_3pi_by_8
    + coincedenceAlice2Bob1_angles_pi_by_4_and_3pi_by_8) )

# by Bell's theorem
E = E1 - E2 + E3 + E4

#Output
print('coincedenceAlice1Bob1_angles_0_and_pi_by_8 = %i '
    % coincedenceAlice1Bob1_angles_0_and_pi_by_8)
print('coincedenceAlice1Bob2_angles_0_and_pi_by_8 = %i '
    % coincedenceAlice1Bob2_angles_0_and_pi_by_8)
print('coincedenceAlice2Bob1_angles_0_and_pi_by_8 = %i '
    % coincedenceAlice2Bob1_angles_0_and_pi_by_8)
print('coincedenceAlice2Bob2_angles_0_and_pi_by_8 = %i '
    % coincedenceAlice2Bob2_angles_0_and_pi_by_8)
print()
print('coincedenceAlice1Bob1_angles_0_and_3pi_by_8 = %i '
    % coincedenceAlice1Bob1_angles_0_and_3pi_by_8)
print('coincedenceAlice1Bob2_angles_0_and_3pi_by_8 = %i '
    % coincedenceAlice1Bob2_angles_0_and_3pi_by_8)
print('coincedenceAlice2Bob1_angles_0_and_3pi_by_8 = %i '
    % coincedenceAlice2Bob1_angles_0_and_3pi_by_8)
print('coincedenceAlice2Bob2_angles_0_and_3pi_by_8 = %i '
    % coincedenceAlice2Bob2_angles_0_and_3pi_by_8)
print()
print('coincedenceAlice1Bob1_angles_pi_by_4_and_pi_by_8 = %i '
    % coincedenceAlice1Bob1_angles_pi_by_4_and_pi_by_8)
print('coincedenceAlice1Bob2_angles_pi_by_4_and_pi_by_8 = %i '
    % coincedenceAlice1Bob2_angles_pi_by_4_and_pi_by_8)
print('coincedenceAlice2Bob1_angles_pi_by_4_and_pi_by_8 = %i '
    % coincedenceAlice2Bob1_angles_pi_by_4_and_pi_by_8)
print('coincedenceAlice2Bob2_angles_pi_by_4_and_pi_by_8 = %i '
        % coincedenceAlice2Bob2_angles_pi_by_4_and_pi_by_8)
print()
print('coincedenceAlice1Bob1_angles_pi_by_4_and_3pi_by_8 = %i '
    % coincedenceAlice1Bob1_angles_pi_by_4_and_3pi_by_8)
print('coincedenceAlice1Bob2_angles_pi_by_4_and_3pi_by_8 = %i '
    % coincedenceAlice1Bob2_angles_pi_by_4_and_3pi_by_8)
print('coincedenceAlice2Bob1_angles_pi_by_4_and_3pi_by_8 = %i '
    % coincedenceAlice2Bob1_angles_pi_by_4_and_3pi_by_8)
print('coincedenceAlice2Bob2_angles_pi_by_4_and_3pi_by_8 = %i '
    % coincedenceAlice2Bob2_angles_pi_by_4_and_3pi_by_8)
print()
print('E1 = %f ' % E1)
print('E2 = %f ' % E2)
```

```
print('E3 = %f ' % E3)
print('E4 = %f ' % E4)
print()
print('E = E1 - E2 + E3 + E4 = %f ' % E)

print('Done')
```

References

[1] Marian Kupczynski,
What do we learn from computer simulations of Bell experiments?
arXiv: 1611.03444 [quant-ph]

[2] K. De Raedt, H. De Raedt, K. Michielsen,
A computer program to simulate Einstein–Podolsky–Rosen–Bohm experiments with photons, Computer Physics Communications 176 (2007) 642–651

[3] G. Weihs, T. Jennewein, C. Simon, H. Weinfurter, A. Zeilinger,
Violation of Bell's inequality under strict Einstein locality conditions.
arXiv:quant-ph/9810080

[4] S. Groeblacher, T. Paterek, R. Kaltenbaek, An experimental test of non-local realism,
arXiv:0704.2529 [quant-ph]