Computational Physics - Exercise 2

Joel Schumacher, 309231, joel.schumacher@rwth-aachen.de

May 24, 2017

Task 1

Introduction

In this task the behaviour of a 1 dimensional symmetric random walk is examined with respect to it's diffusion-like properties. The goal is to find the functional relationship between the variance $v = \langle x^2 \rangle - \langle x \rangle^2$and the number of steps of the random walk.

Simulation model and method

The random walk will be modelled by a position stored for each particle that is updated in every timestep by either being incremented or decremented randomly with a percentage of 50% each. The initial position will be 0 for each particle. With N_{jumps} being the total number of jumps simulated the 1D lattice has to have dimensions of $2 \cdot N_{jumps} \cdot \Delta x$, where Δx is the distance between each lattice site. Each final position may range from $-N_{jumps}$ to N_{jumps}.

Simulation results

In figure 1 the result of plotting the variance $v = \langle x^2 \rangle - \langle x \rangle^2$ over the number of jumps can be seen. A linear relationship of the form $v \approx N$ can clearly be observed. An additional plot showing the histograms of particle positions after a different number steps can be seen in figure 2. As expected from the theory discussed in the lecture, all distributions have a gaussian shape and in compliance with the first graph an increased width of each distribution for higher numbers of jumps N is observed.

Discussion

The linear relationship exhibited in figure 1 is in accordance with the expectation from the analytical considerations in that the gaussian distribution for the position for each particle has a mean of $\langle x \rangle = 0$ and the root mean square was determined as $\sqrt{\langle x^2 \rangle} = \sqrt{N} \Delta x \Leftrightarrow \langle x^2 \rangle = N \cdot \Delta x^2 = N$ so that it follows for the variance v:

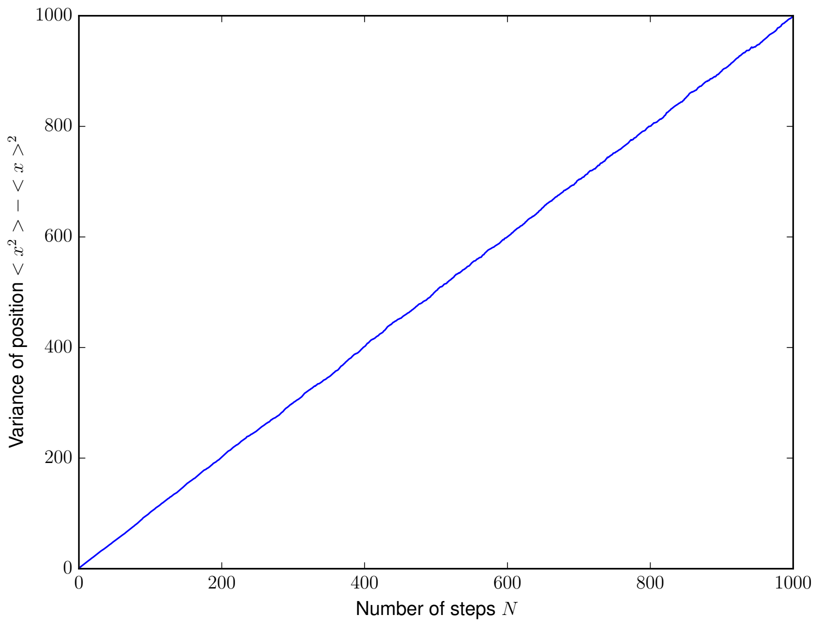

Figure 1: The variance of the positions v with respect to the number of jumps N

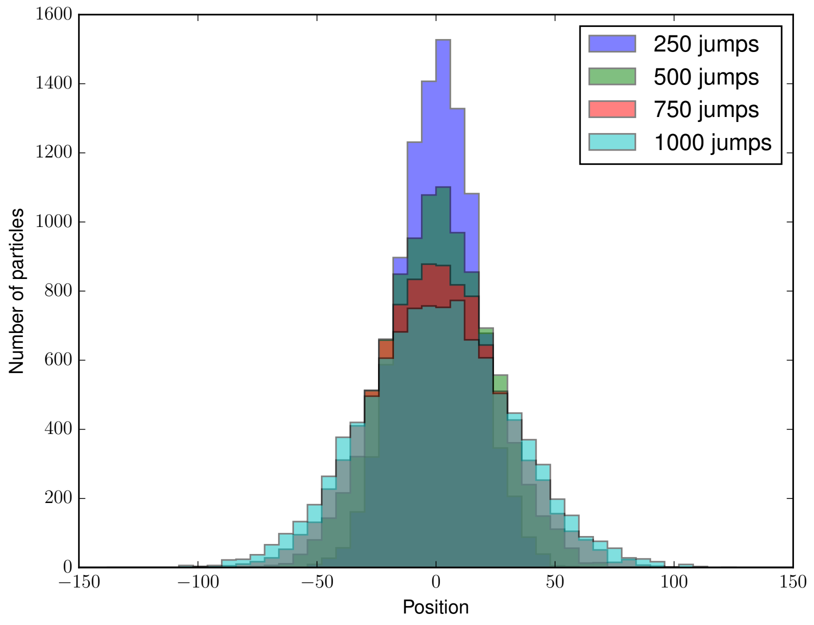

Figure 2: Histograms of the particle positions for different numbers of jumps

$$v = \langle x^2 \rangle - \langle x \rangle^2 = N$$

which is exactly what is visible in the plot. This behaviour corresponds to the aforementioned "diffusion" in that the particles' spatial uncertainty rises.

Task 2

Introduction

In this task a simulated realization of the Einstein–Podolsky–Rosen thought experiment is studied, specifically Bohm's variant of a system of two entangled spins. The EPR thought experiment, often called "EPR paradoxon" suggests non-locality or non-realism in quantum mechanical systems.

It consists of an entangled system of two spins, that may be well separated in space-time after being prepared in the entangled state. The orientation of one of the spins is unknown at the moment of measurement and according to the Copenhagen interpretation even indeterminate before measurement, therefore not element of physical reality. When one of the spins is measured information about the other system can be gained due to their entangled preparation, while never having measured that system itself. Local causality is assumed, meaning that the measurement of one may not influence the other system, because doing so instantly would violate relativity. That new gained information is then element of physical reality, but not accessible by the formalism of quantum mechanics when considering only that subsystem of the unmeasured spin. Einstein, Podolsky and Rosen concluded the incompleteness of quantum mechanics and the necessity for hidden variables. These ideas have been invalidated by experiments confirming Bell's theorem - "No physical theory of hidden variables can ever reproduce all of the predictions of quantum mechanics".

Simulation model and method

In the simulation a pair of photons is considered of which one has a random polarization and the second is chosen orthogonal to the first. The polarization of the first photon is then rotated by a random angle φ to simulate measuring both photons along a random direction each, Malus's law is applied and their polarization is measured along that axis (+1 or −1). The event counts of these measurements, referenced as "count" in code, c_{ij} with $i, j \in \{+, -\}$ are then used to determine the expectation values of each individual polarization/spin $E_1(a,b) = \langle \Psi | \sigma_1 \cdot a | \Psi \rangle$, $E_2(a,b) = \langle \Psi | \sigma_2 \cdot b | \Psi \rangle$:

$$E_1(a,b) = \frac{c_{--} + c_{-+} - c_{++} - c_{+-}}{c_{--} + c_{-+} + c_{++} + c_{+-}}$$

$$E_2(a,b) = \frac{c_{--} + c_{+-} - c_{++} - c_{-+}}{c_{--} + c_{+-} + c_{++} + c_{-+}}$$

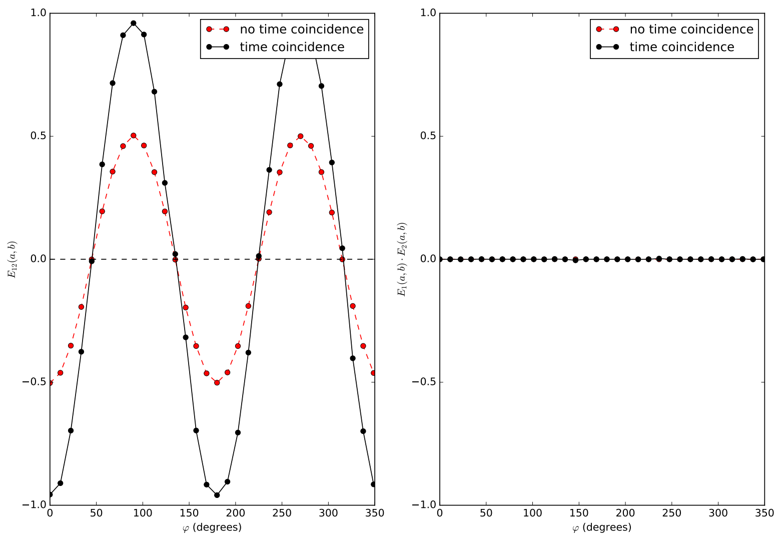

Figure 3: Expectation values of each individual polarization (right) and the correlation of the polarizations (left) over the angle difference of the analyzers φ. With time coincidence and without in black and red respectively.

One can also calculate the correlation $E_{12}(a,b) = \langle\Psi|\sigma_1 \cdot a\sigma_2 \cdot b|\Psi\rangle$:

$$E_1(a,b) = \frac{c_{++} + c_{--} - c_{+-} - c_{-+}}{c_{++} + c_{--} + c_{+-} + c_{-+}}$$

Additionally and optionally a time tag can be assigned to each photon that is determined by considering the retardation effects of the wave plates used to alter the polarization of the photons. To mimic the real experiment only temporally correlated pairs with $\Delta t < W$ are used where W is the time window to select two photons as a pair. This coincidence criterion is then applied to all events and all non-coinciding pairs are omitted to improve accuracy.

Simulation results

The results of this simulation can be seen in figure 3. The expectation value of the indiviual polarization vanishes for every angle and the expectation value of the correlation follows a sinusodial behaviour. When taking into account time coincidence of the photons, a more pronounced signal for the correlation is observed.

Discussion

The vanishing of the expectation value of the individual polarizations is in absolute accordance with the expectation, since for a uniformly distributed angle it is equally likely to point “up” or “down”.

The correlation predicted analytically for the quantum experiment is $E_{12}(a,b) = -a \cdot b = \cos(\sphericalangle(a,b))$, which is also in accordance with the experiment.

It is therefore possible to simulate a quantum mechanical system without directly incorporating the formalism of quantum mechanics itself and reproduce behaviour that is exhibited in this quantum system.

It should be noted that the measurements in the simulation are compatible with Einstein’s criterion of local causality, because we measure only one photon in the analyzer without then modifying the other photon using the result of that measurement. The EPR paradoxon is then resolved in this simulation, but through the usage of hidden variables.

Appendix

Program for Task 1

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

N_PART = 10000
N_JUMPS = 1000

part_pos = np.zeros(N_PART, dtype=np.int32)

# Matricle number 309231
np.random.seed(9231)

plt.rc('text', usetex=True)

var = np.zeros(N_JUMPS)
for i_jump in range(N_JUMPS):
    for i_part in range(N_PART):
        if np.random.random() < 0.5:
            part_pos[i_part] += 1
        else:
            part_pos[i_part] -= 1
    var[i_jump] = np.var(part_pos)

    if (i_jump+1) % 250 == 0:
        plt.hist(part_pos, bins=50, alpha=0.5,
            label="{} jumps".format(i_jump+1),
            histtype="stepfilled", range=(-150, 150))#
                range=(-N_JUMPS,N_JUMPS))
plt.legend()
plt.xlabel("Position")
plt.ylabel("Number of particles")
plt.savefig("walk_hist.pdf")
plt.show()

plt.plot(var)
plt.xlabel("Number of steps $N$")
plt.ylabel("Variance of position $<x^2> - <x>^2$")
plt.savefig("walk_var.pdf")
plt.show()
```

Program for Task 2

```
import math

import numpy as np
import matplotlib.pyplot as plt

def analyzer(c, s, cHWP, sHWP, T0):
    # modulator rotates polarization
    c2 =  cHWP*c + sHWP*s
    s2 = -sHWP*c + cHWP*s
    x = c2*c2 - s2*s2 # cos(2(x-a))
    y = 2*c2*s2 # sin(2(x-1))

    # Malus law
    r0 = np.random.random()
    if x > r0*2.0-1.0:
        j = 0 # +1 event
    else:
        j = 1 # -1 event

    # delay time: T0 * sin(2(theta1-x))**4
    l = y*y*y*y*T0*np.random.random()

    return j, l

nsteps = 32
nsamples = 200000
nsamples = 200000
HWP2 = 0
T0 = 1000
```

```
W = 1

cHWP2 = math.cos(HWP2/180*math.pi)
sHWP2 = math.sin(HWP2/180*math.pi)
count = np.zeros((2, 2, 2, nsteps))

for ipsi0 in range(nsteps):
    # loop over all different settings of electro-
        optic modulator
    cHWP1 = math.cos(ipsi0*2.0*math.pi/nsteps)
    sHWP1 = math.sin(ipsi0*2.0*math.pi/nsteps)

    for i in range(nsamples):
        # random polarization for one photon
        angle = np.random.random() * 2.0*math.pi
        c1 = math.cos(angle)
        s1 = math.sin(angle)
        # the other photon is orthogonal
        c2 = -s1
        s2 = c1

        j1, t1 = analyzer(c1, s1, cHWP1, sHWP1, T0)
        j2, t2 = analyzer(c2, s2, cHWP2, sHWP2, T0)

        count[j1, j2, 0, ipsi0] += 1
        if abs(t1-t2) < W:
            count[j1, j2, 1, ipsi0] += 1

# data analysis
tot = np.zeros((2, nsteps))
E1 = np.zeros((2, nsteps))
E2 = np.zeros((2, nsteps))
E12 = np.zeros((2, nsteps))
S = np.zeros((2, nsteps))
r0 = np.zeros(nsteps)
phi = np.zeros(nsteps)

for j in range(nsteps):
    phi[j] = j * 360.0 / nsteps
    r0[j] = -math.cos(2.0*j*2.0*math.pi/nsteps)

    for i in range(2): # temporally correlated or not
        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]
        E1[i,j] = count[0,0,i,j] + count[0,1,i,j] -
```

```python
            count[1,1,i,j] - count[1,0,i,j]
        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]

        S[i,j] = 3.0 * E12[i,j] - E12[i, j*3 % nsteps]

# plot
plt.rc('text', usetex=True)
f, axarr = plt.subplots(1, 2)

axarr[0].plot(phi, E12[0,:], label="no time coincidence", linestyle="--", marker="o", color="r")
axarr[0].plot(phi, E12[1,:], label="time coincidence", linestyle="-", marker="o", color="k")
axarr[0].axhline(0, linestyle="--", color="k")
axarr[0].set_xlabel(r"$\phi$ (degrees)")
axarr[0].set_ylabel(r"$<S_1 S_2>$")
axarr[0].set_ylim(-1, 1)
axarr[0].legend()

axarr[1].plot(phi, E1[0,:]*E2[0,:], label="no time coincidence", linestyle="--", marker="o", color="r")
axarr[1].plot(phi, E1[1,:]*E2[1,:], label="time coincidence", linestyle="-", marker="o", color="k")
axarr[1].set_xlabel(r"$\phi$ (degrees)")
axarr[1].set_ylabel(r"$<S_1><S_2>$")
axarr[1].set_ylim(-1, 1)
axarr[1].legend()
plt.savefig("ex2.pdf")
plt.show()
```