# Computational Physics - Exercise 2: Event-based simulation I

Torben Steegmann Chih-Chun Kuo

## Introduction

The random walk is one of the event-based simulation models with numerous applications in various fields, such as physics, brain research, and the economy. In this exercise, we implement a symmetric 1D random walk and investigate its statistical properties, specifically the variance.

## Simulation model and method

A symmetric 1D random walk is a random walk on a one dimensional lattice, where each step the chosen direction has the same probability. Additionally, the length of each step has the same size.

We can model such a walk by computing a few subtasks.

(1) First of all, we define a function `move()` which simulates the movement of a particle in a 1D random walk. It generates a random step of +1 (to the right) or −1 (to the left) with equal probability using `random.randint(0, 1)` with the random seed 3227.

(2) Then we initialize an array `position` to keep track of the positions of all particles. Each element of this array represents the position of a single particle.

(3) Then we apply the following nested loops to iterate over each step and each particle. For each particle, the `move()` function is called to generate a random step, and the particle's position is updated accordingly. After each step, the mean and mean square values of the particle positions are calculated and stored in the respective arrays `mean` and `ms`.

```
for j in range(N_jump):
```

```
    for i in range(N_particle):
        position[i] += move()

    mean[j] = np.sum(position) / N_particle
    ms[j] = np.sum(position**2) / N_particle
```

For this exercise, we are specifically interested in the variance of such a simulation. The variance $\langle x^2 \rangle - \langle x \rangle^2$ measures the spread of data points alround their mean value. Due to the symmetry of the walk, we expect the variance to equal the amount of steps taken. The final subtask to be calculated is hence as follows.

(4) The variance of the particle positions is calculated using the formula $\langle x^2 \rangle - \langle x \rangle^2$ and the resulting data points are plotted in Figure 2.

## Simulation results

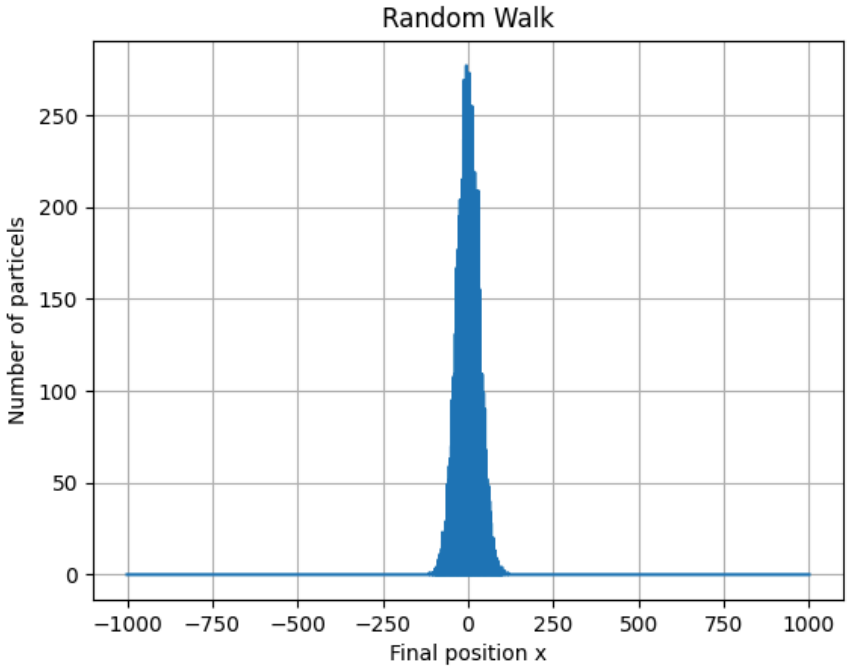


Figure 1: The end positions of the $N_{\text{part}} = 10000$ particles after 1000 steps. We see the shape resembles a Gaussian distribution.

Running the described program and plotting the final position x of all particles after 1000 steps results in the graph depicted in Figure 1. We see the shape of the end position resembles a Gaussian distribution,

centered around the starting position 0. Plotting these results according to their variance results in Figure 2. Even though the resulting graph is not perfectly linear, the variance has a clear linear correspondence to the number of steps taken. The slight imperfections in linearity are a result of the inherent randomness of such a simulation.

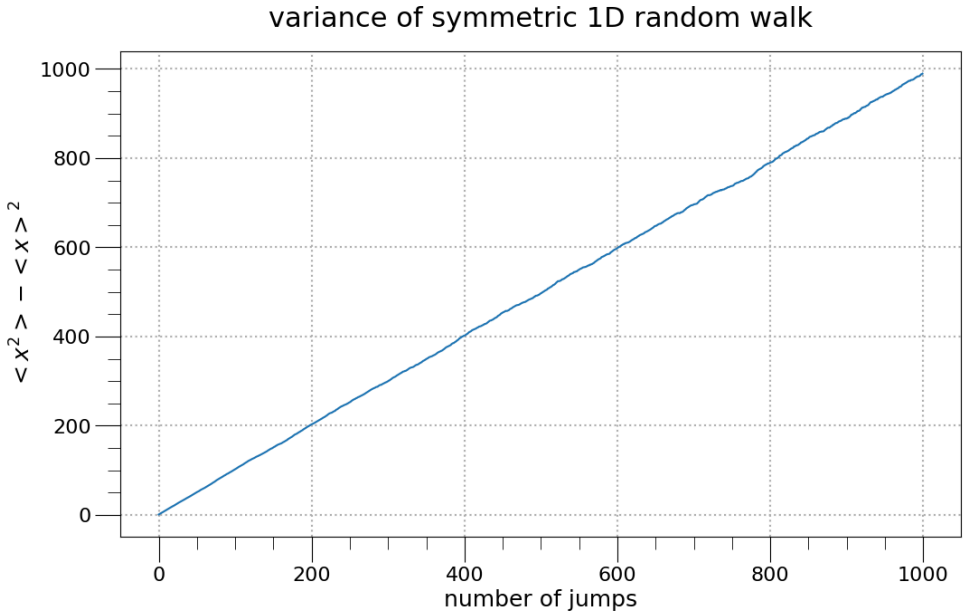


Figure 2: The variance of a symmetric 1D random walk of 10000 particles. The x-axis depicts the number of jumps a particle has made from 0 to 1000. The y-axis depicts the measured variance $\langle x^2 \rangle - \langle x \rangle^2$. We can see a clear linear correspondense which matches the analytical solution.

When simulating such a walk on a 1D lattice one would have to have a lattice length of at least 2N+1. Although the probability that a particle makes N steps in the same direction is incredebly slim, it is non-zero. Since that direction can either be positive or negative we need a length of 2N and 1 additional length for 0.

## Discussion

We know that coin tossing follows a binomial distribution, which means

$$Y_{ij} \sim \text{Binomial}(i, p = 0.5),$$

where $Y_{ij}$ represents the random variable of heads we get by tossing i times coins for particle j. We then define the sum of the results of all particles

$$Y_i := Y_{i1} + Y_{i2} + ... + Y_{iN_{\text{particle}}}.$$

So

$$Y_i \sim \text{Binomial}\big(i \cdot N_{\text{particle}}, p = 0.5\big),$$

where the expectation value is

$$E(Y_i) = i \cdot N_{\text{particle}} \cdot p,$$

and the variance is

$$\text{Var}(Y_i) = i \cdot N_{\text{particle}} \cdot p(1-p).$$

Then we define $X_{ij}$ as the random variable representing the position of particle j after i jumps, so

$$X_i = X_{i1} + X_{i2} + ... + X_{iN_{\text{particle}}}$$

The relation between the sum of the positions of all particles and coin tossing can be expressed as

$$X_i = Y_i - \big(i \cdot N_{\text{particle}} - Y_i\big) = 2Y_i - i \cdot N_{\text{particle}},$$

so

$$E(X_i) = 2E(Y_i) - i \cdot N_{\text{particle}} = i \cdot N_{\text{particle}} - i \cdot N_{\text{particle}} = 0$$

$$\text{Var}(X_i) = 2^2 E(Y_i) = 4i \cdot N_{\text{particle}} \cdot p(1-p) = i.$$

In Figure 2, the simulation result for the variance of the random walk closely mirrors the analytical prediction, as described by

$$\text{Var}(X_i) = i,$$

exhibiting a clear linear relationship.

# Appendix

```
import random
import numpy as np

random.seed(3227) #set random seed

# Function to simulate the movement of a particle
def move():
    # Generate a random integer: 0 for left, 1 for right
    if random.randint(0, 1) == 1:
        return 1  # Move right
    else:
        return -1  # Move left

# Parameters for the simulation
N_particle = 10000
N_jump = 1000

# Initialize arrays to store mean and ms values after each
jump
ms = np.zeros(N_jump)
mean = np.zeros(N_jump)

# Array to store positions of all particles
position = np.zeros(N_particle)


for j in range(N_jump):

    for i in range(N_particle):
        # Update particle position by making a random move
        position[i] += move()

     # Calculate mean and ms values of particle positions
after each jump
    mean[j] = np.sum(position) / N_particle
    ms[j] = np.sum(position**2) / N_particle
```

```
# Calculate the variance of particle positions using the
mean and rms values
variance = ms - mean**2

# plotting and formatting
import matplotlib.pyplot as plt

fig = plt.figure(figsize = (11,7))
ax = fig.add_subplot(1,1,1)

plt.plot(variance, label="variance")
# plt.plot(mean, label="mean")

plt.grid(linestyle = "dotted", linewidth = 1.5)
# plt.legend()
plt.title("variance of symmetric 1D random walk", fontsize
= 22, pad = 20)
plt.xlabel("number of jumps", fontsize = 18)
plt.ylabel("$<x^2>-<x>^2$", loc = "center", fontsize = 18)

plt.subplots_adjust(top=0.89,
bottom=0.125,
left=0.13,
right=0.975,
hspace=0.2,
wspace=0.2)
plt.minorticks_on()
plt.tick_params(length=20,    axis='both',    labelsize=16,
which='major')
plt.tick_params(length=10,  axis='both', which='minor')

plt.show()
```