Report 2: Event-based simulation of random walk

Don Winter[1] and Madhulan Suresh[2]

[1]don.winter@rwth-aachen.de, Mat. No. 431380
[2]madhulan.suresh@rwth-aachen.de, Mat. No. 429481

Introduction

In this exercise we simulate a symmetric one-dimensional random walk in an event-based simulation. In this variant of the random walk a particle is starting at the origin ($x_0 = 0$) and is only allowed to move either left or right by step size Δx. Which direction a particle takes is decided via a fair coin flip which can result in the two outcomes $X = \pm 1$ with equal probability, s.t. the particle moves $x \to x + X\Delta x$. Each coin flip and subsequent movement action is called an *event*. To simulate a random walk N events are executed sequentially, s.t. the position of a particle is updated after each event. The probability of ending up at final location x after N steps is thus binomially distributed, which we can use to analytically calculate statistical quantities. For this exercise however, we are asked to simulate $N = 1000$ events for $N_{part} = 10000$ particles and from this calculate the statistical quantities numerically, i.e.we are asked to calculate and plot the variance $\langle x^2 \rangle - \langle x \rangle^2$ as a function of N, which we are then to compare to the corresponding analytical result.

Simulation model and method

For a single particle, we model the final position x of the random walker as the sum of N random variables X_i which can take on the values ± 1 with a probability $I\!P(X_i = \pm 1) = 0.5$,

$$x = x_0 + X_1 + ... + X_N, \tag{1}$$

where for our case $x_0 = 0$. For N successive events we would expect

$$\langle x \rangle = \sum_{i=1}^{N} \langle X_i \rangle = \sum_{i=1}^{N} -1 I\!P(X_i = -1) + 1 I\!P(X_i = +1) = 0, \tag{2}$$

since the outcomes $X_i = \pm 1$ are equally likely and $I\!P(X_i)$ is symmetric around $x_0 = 0$. Furthermore, for the expectation of the squared final position we have

$$\langle x^2 \rangle = \langle (\sum_{i=1}^{N} X_i)^2 \rangle = \langle \sum_{i}^{N} \sum_{j}^{N} X_i X_j \rangle = \sum_{i}^{N} \sum_{j}^{N} \langle X_i X_j \rangle = N + \sum_{i \neq j} \langle X_i X_j \rangle = N, \tag{3}$$

since for independent random variables $\langle X + Y \rangle = \langle X \rangle + \langle Y \rangle$, as well as $\langle X_i^2 \rangle = 1$ and since $X_i = -1$ and $X_i = +1$ are equally likely also $X_i X_j = 1$ and $X_i X_j = -1$ for $i \neq j$ are equally likely, thus $\langle X_i X_j \rangle = 0$. So, for the analytical value of the variance we can conclude that

$$Var[x] = \langle x^2 \rangle - \langle x \rangle^2 = N. \tag{4}$$

To answer the technical question posed in the exercise statement, we have to choose a lattice of size $2N = 2000$ positions, as in the worst case scenario particles can end up at $-N$ or N, respectively. However, we'd like to emphasize that the probability to reach these positions is with $0.5^N \sim 10^{-301}$ highly improbable. In practice, we'd see that 99% of our values actually fall within $[-82, 82]$, i.e. in a range less than one order of magnitude. This result can be obtained by observing that the random variables X_i are i.i.d. resulting in a binomial distribution centered around $x_0 = 0$ after N trials, which approaches the standard normal distribution[1] with standard deviation $\sigma_x = \sqrt{Var[x]} = \sqrt{N}$. Thereby, in order to account for 99% of all x-values, corresponding to a z-score of 2.57, we have $x = x_0 \pm 2.57\sigma_x = 0 \pm 2.57\sqrt{1000} \approx \pm 81.3$, yielding the confidence interval $[-82, 82]$

[1] By the law of large numbers. This can also be shown analytically by using Stirling's formula on the logarithm of the binomial distribution for large N (that is large in comparison to step size Δx) which we are however not going to prove here for brevity.

Due to independence of particle walks, we simulated each of the N_{part} particles in parallel. Thus, in the simulation we only looped over the number of steps N, creating a vector $\vec{r}$ of N_{part} random values between 0 and 1 drawn from a uniform distribution. Then, we used numpy's advanced indexing to add the previous column of final positions $\vec{x}_{i-1}$ to the current column $\vec{x}_i$, while adding (subtracting) $\Delta x = 1$ where the corresponding element of $\vec{r}$ is larger or equal[2] (smaller) than 0.5, which we subsequently store in the ith column of our result matrix. By this prodecure, we iteratively filled our final $N_{part} \times N$ matrix one column $\vec{x}_i$ at a time, starting with the initial position $\vec{x}_0 = \vec{0}$ at $i = 0$.

Simulation results

In order to plot the desired variance as a function of the number of events (steps), we summed over the rows of our output matrix and devided by N_{part} to get the expectation (i.e. average) value $\langle x \rangle$ per event. We did the same for the squared output matrix, in order to get $\langle x^2 \rangle$, and finally squaring $\langle x \rangle$ and taking the difference $\langle x^2 \rangle$ - $\langle x \rangle^2$ yielded the variance, which we than plotted against N in Fig. 1. We also plotted the analytical result $Var[x] = N$. As we can see, the two agree reasonable well.

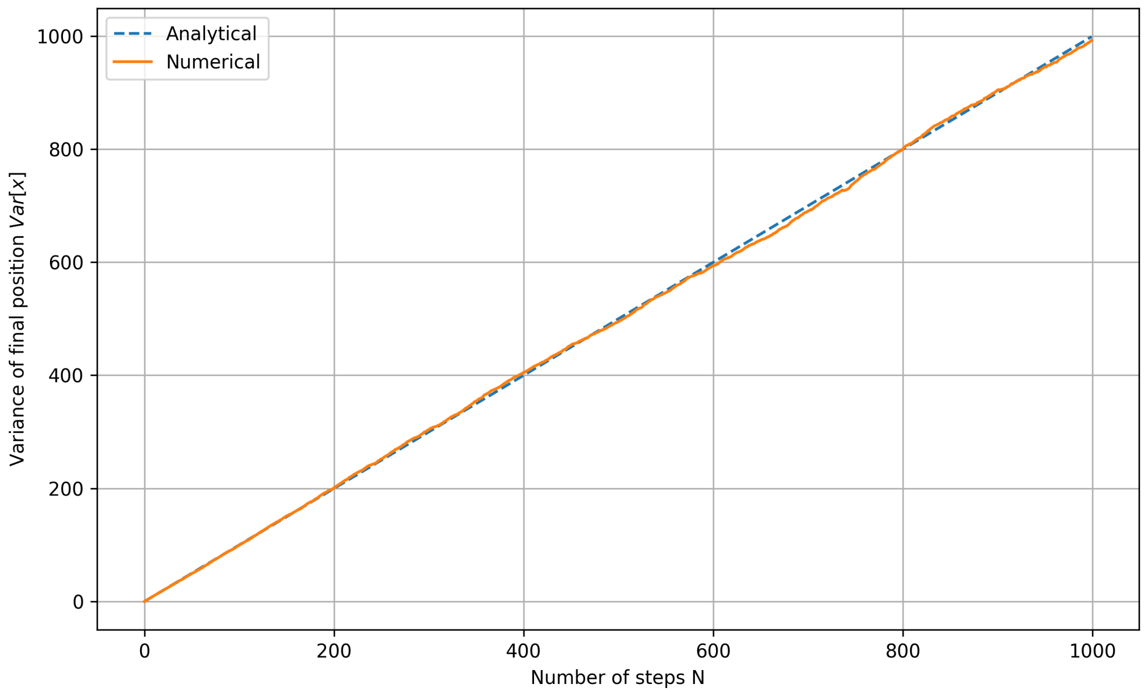

Figure 1. Variance as function of number of steps for a simple random walk

In order to verify the 99% confidence interval we calculated analytically at the end of last section also by our numerical data, we plot a histogram of the last column of our resulting matrix partitioned into 300 bins in Fig. 2. We can indeed see a normal distribution (imperfect resolution due to binning) where most of the occurances are in the interval $[-82, +82]$. Even if we consider also the remaining 1% not included in the confidence interval, we see that there are no values outside of $[-150, 150]$, which supports our claim that most of the very distant final positions are in practice never reached.

Discussion

In summary, we analyzed the one-dimensional symmetric random walk from first principles, in order to derive an expression for expectation values and the variance. We then ran an event-based simulation of N_{part} particles performing a random walk to compute the same values numerically. Finally, we compared the two results. In conclusion, our numerical simulation results nicely match the theoretical values.

[2] Since the random number is drawn from $[0, 1)$ we have to include equality for the upper half of the interval.

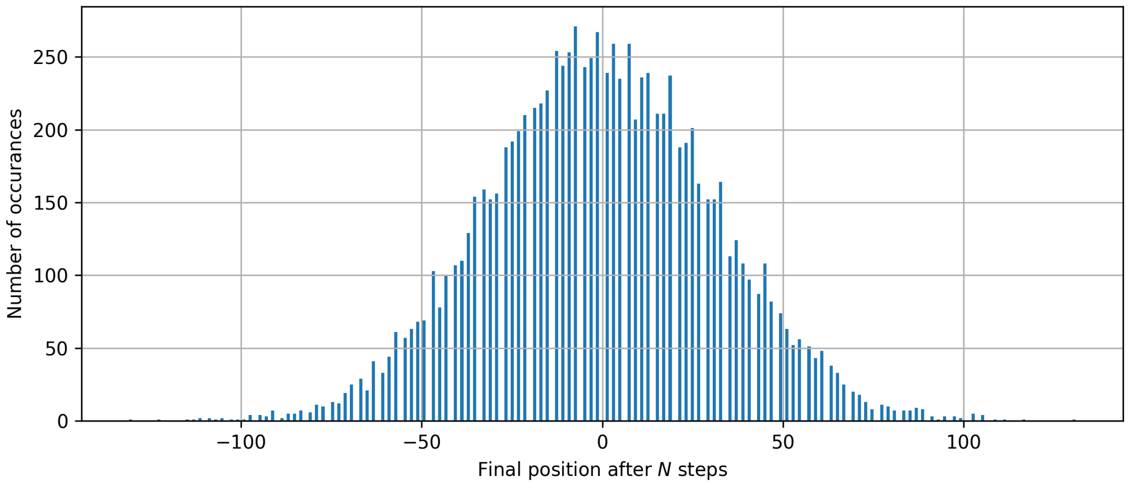

Figure 2. Histogram of final position x of N_{part} particles after N steps

Appendix

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

n_steps = 1_000
n_particles = 10_000
np.random.seed(1380) # Set seed
X = np.zeros((n_particles, n_steps)) # initialize result matrix

for i in range(1,n_steps):
    r = np.random.uniform(size=(n_particles)) # random vector
    X[r>0.5,i] += X[r>=0.5,i-1] + 1 # Add previous value +1 for each "Heads"
    X[r<0.5,i] += X[r<0.5,i-1] - 1 # Add previous value -1 for each "Tails"

avg = np.sum(X, axis=0) / n_particles # Calculate expectation value of x numerically
var = np.sum(X**2, axis=0) / n_particles - avg ** 2 # Calculate variance numerically

# Plot variance as function of no. of events
plt.figure(figsize=(10,6))
plt.plot(range(n_steps), range(n_steps), '--', label="Analytical") # Analytical variance f(n)=n
plt.plot(range(n_steps), var, label="Numerical")
plt.xlabel('Number of steps N')
plt.ylabel(r'Variance of final position $Var[x]$')
plt.grid()
plt.legend()
plt.savefig("fig", dpi=300)

# Plot histogram of final position values
plt.figure(figsize=(10,4))
plt.hist(X[:,-1], bins=300)
plt.xlabel(r'Final position after $N$ steps')
plt.ylabel(r'Number of occurances')
plt.grid()
plt.savefig("fig2", dpi=300)
```