FSTC: Bachelor of Mathematics

Supervisor: Simon Campese

Random Walks

Yanis Bosch, Tim Seuré

Experimental mathematics 2
Winter semester 2019/20

Abstract

In the following, we will investigate random walks on $\mathbb{Z}$ and $\mathbb{Z}^2$. We will suggest some ways of representing them and analyse random variables connected to the random walks. In particular, our goal will be to experimentally examine their properties and check accordance with theoretical results.

We have initially chosen the topic *Runs in random sequences*, but upon studying runs in random walks, we have found other interesting properties that we wanted to report about. Therefore, we have entirely focused our mind on random walks in general.

Contents

1 One-dimensional random walk

1.1 Introduction

Casually described, consider a bunny with initial position $0 \in \mathbb{Z}$ and at each step, jumping either one unit to the left (interpreted as -1) with a probability of $p \in [0,1]$, or one unit to the right (interpreted as $+1$) with a probability of $q := 1 - p \in [0,1]$. The path this bunny treads will be described as a one dimensional random walk, or more concretely, a random walk on $\mathbb{Z}$.

In this section, we will be trying to analyse the following questions:

- How can the bunny's movement be illustrated?
- If the bunny does n jumps in total, how often does it return to the origin 0?
- If the bunny does n jumps in total, how often will it jump in a given pattern E? How many times will it jump until it has bounced in a given pattern E?
- How many times will the bunny have to jump until it has a distance N to the origin 0?

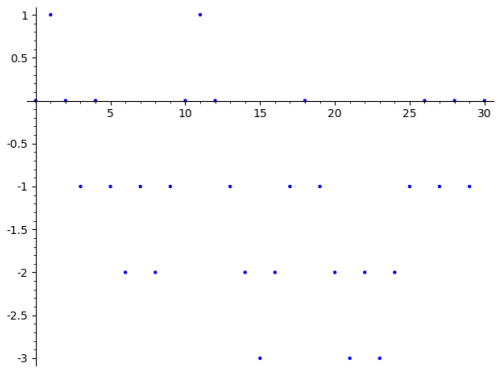
Firstly, let us give a more formal definition of what is meant by a random walk on $\mathbb{Z}$.

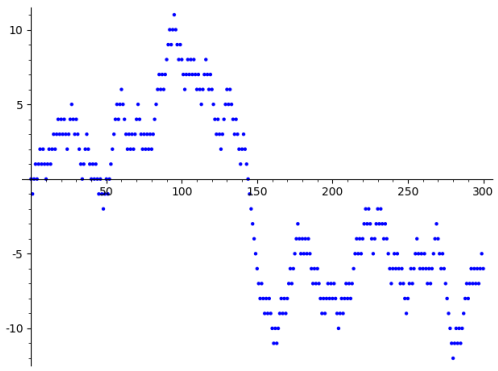
Definition 1 (Random walk on $\mathbb{Z}$)**.** Let $p \in [0,1]$ and $q := 1 - p$. Let $(X_i)_{i \in \mathbb{N}}$ be a sequence of independent and identically distributed random variables satisfying $\forall i \in \mathbb{N} : \mathbb{P}\{X_i = 1\} = p$ and $\mathbb{P}\{X_i = -1\} = q$. For $n \in \mathbb{N}$, let

$$S_n := \sum_{i=1}^{n} X_i.$$

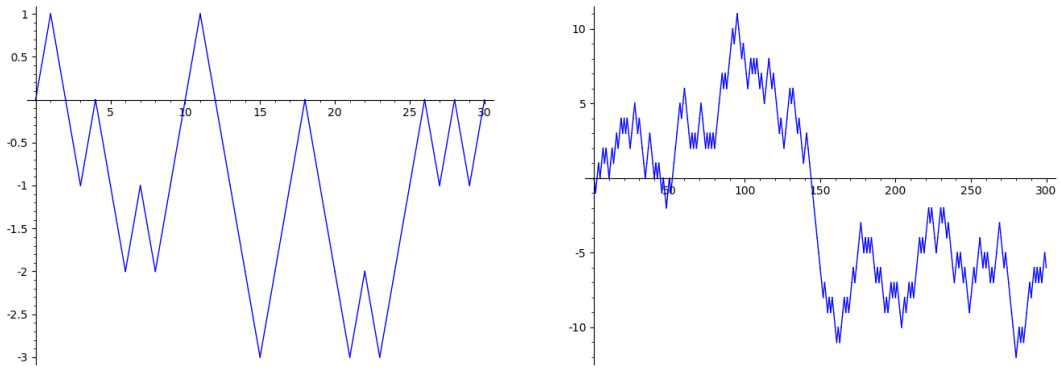
Then $(S_n)_{n \in \mathbb{N}}$ is called a *random walk on* $\mathbb{Z}$. If $p = \frac{1}{2}$, the random walk $(S_n)_{n \in \mathbb{N}}$ is *symmetric*.

Now, a random walk (S_n) can be illustrated by representing the points (n, S_n) in a two-dimensional orthogonal coordinate system, as illustrated in the following, where $p = \frac{1}{2}$ and we're representing random walks $(S_n)_{0 \le n \le m}$ for $m = 30, 300$:





The same random walks can be depicted by drawing the lines connecting the points (n, S_n) and $(n+1, S_{n+1})$:



The latter representation is the one we will be using for the following, as the first one might be confusing for small m.

1.2 Returning to zero

Informally speaking, the question we're asking is how often the bunny returns to its starting point, given the amount of jumps it puts into practice. It is clear that this question does not have a precise answer, as we are talking about a random event. Nonetheless, we will see that it can be approximated quite well. We will limit ourselves to the symmetric random walk, i.e. $p = \frac{1}{2}$.

Let us now mathematically formulate what we are trying to analyse.

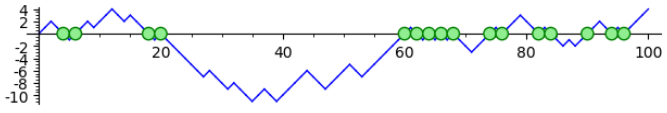
Definition 2 (Number of zeroes)**.** Let $(S_n)_{n\in\mathbb{N}}$ be a symmetric random walk on $\mathbb{Z}$. For a given $m \in \mathbb{N}$, let us define

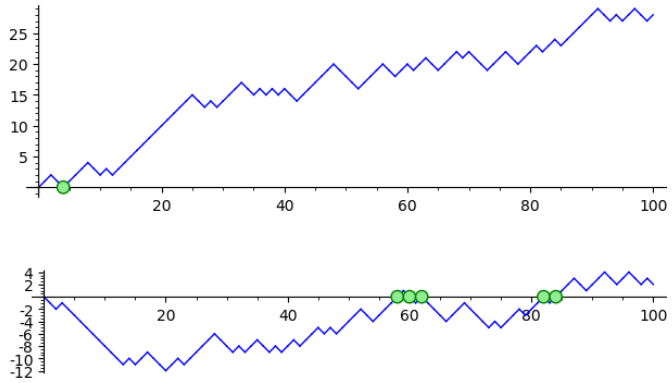
$$Y_m := \sum_{i=1}^{m} \mathbb{1}_{\{S_i=0\}}.$$

Y_m is called the *number of zeroes* of $(S_n)_{0\leq n\leq m}$.

Note that one always has $S_0 = 0$ by definition, but this zero is not counted in Y_m. Also, one could reduce the sum above to only account for those i that are even, as the random walk can never hit the x-axis after an odd amount of steps.

To investigate the problem, we shall illustrate some random walks $(S_n)_{0\leq n\leq 100}$ and mark the points $(n, 0)$ if $S_n = 0$.





Those graphs suggest that the number of zeroes seems to fluctuate strongly, i.e. the variance of Y_m is great. Let us represent the number of zeroes Y_{100} for 10 different random walks generated with Sage.

Y_{100}	7	2	5	7	4	6	9	1	3	8

Their arithmetic mean is 5.2. Let us now take a sample of 10 000 random walks $(S_n)_{0\leq n\leq 100}$ and compute, again using Sage, the arithmetic mean Y of their number of zeroes Y_{100}. One obtains $Y = 7.0403$. Redoing this gives $Y = 7.0447$, suggesting that we are indeed close the the expected value of Y_{100}. This is confirmed by the following

Theorem 1. *The expected value of the number of zeroes $Y_{2m}, m \geq 1$, is given by*

$$\mathbb{E}(Y_{2m}) = (2m+1)\frac{\binom{2m}{m}}{2^{2m}} - 1.$$

Proof. This theorem follows from the probability

$$\mathbb{P}(S_n = k) = \frac{\binom{n}{\frac{n+k}{2}}}{2^n}, \text{ where } -n \leq k \leq n.$$

A proof of this formula can be found in *Irrfahrten* [2]. Using this, combined with the equality $Y_{2m} = \sum_{i=1}^{m} \mathbb{1}_{\{S_{2i}=0\}}$ (which follows from the fact that S_{2i+1} is never equal to 0), we obtain:

$$\mathbb{E}(Y_{2m}) = \mathbb{E}\left(\sum_{i=1}^{m} \mathbb{1}_{\{S_{2i}=0\}}\right) = \sum_{i=1}^{m} \mathbb{E}\mathbb{1}_{\{S_{2i}=0\}} = \sum_{i=1}^{m} \mathbb{P}(S_{2i}=0) = \sum_{i=1}^{m} \frac{\binom{2i}{i}}{2^{2i}} = (2m+1)\frac{\binom{2m}{m}}{2^{2m}} - 1,$$

where the last equality can be shown inductively over $m \geq 1$:

$\underline{m = 1:}$ We have:

$$\sum_{i=1}^{1} \frac{\binom{2i}{i}}{2^{2i}} = \frac{1}{2} = (2\cdot 1 + 1)\frac{\binom{2\cdot 1}{1}}{2^{2\cdot 1}} - 1.$$

$\underline{m \to m+1:}$ Assume the formula holds for a given $m \geq 1$. Then:

$$\sum_{i=1}^{m+1} \frac{\binom{2i}{i}}{2^{2i}} = \sum_{i=1}^{m} \frac{\binom{2i}{i}}{2^{2i}} + \frac{\binom{2m+2}{m+1}}{2^{2m+2}} = (2m+1)\frac{\binom{2m}{m}}{2^{2m}} + \frac{\binom{2m+2}{m+1}}{2^{2m+2}} - 1 = \frac{4(2m+1)\binom{2m}{m} + \binom{2m+2}{m+1}}{2^{2m+2}} - 1,$$

where the second equality uses the induction hypothesis. Hence:

$$\begin{aligned}\sum_{i=1}^{m+1}\frac{\binom{2i}{i}}{2^{2i}} &= \frac{4(2m+1)(2m)!(m+1)^2+(2m+2)!}{(m+1)!(m+1)!\cdot 2^{2m+2}}-1\\ &= \frac{(2(m+1)+1)\,(2m+2)!}{(m+1)!(m+1)!\cdot 2^{2m+2}}-1\\ &= (2m+3)\frac{\binom{2m+2}{m+1}}{2^{2m+2}}-1,\end{aligned}$$

as desired. □

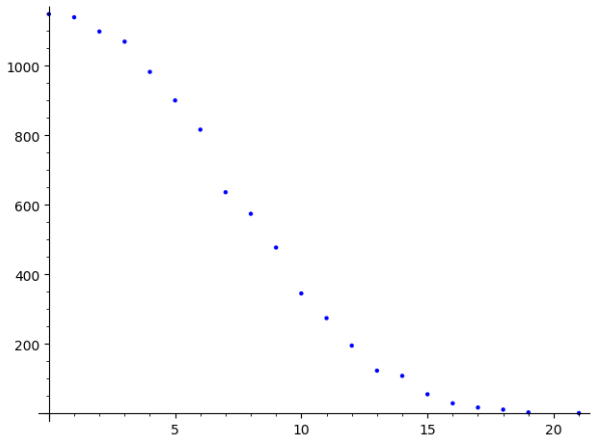
Note that $\mathbb{E}(Y_{2m+1}) = \mathbb{E}(Y_{2m})$, as there cannot be a zero at the $(2m+1)^{\text{st}}$ step.

We can now check that the previous values of Y were indeed close to the expected value of Y_{100}:

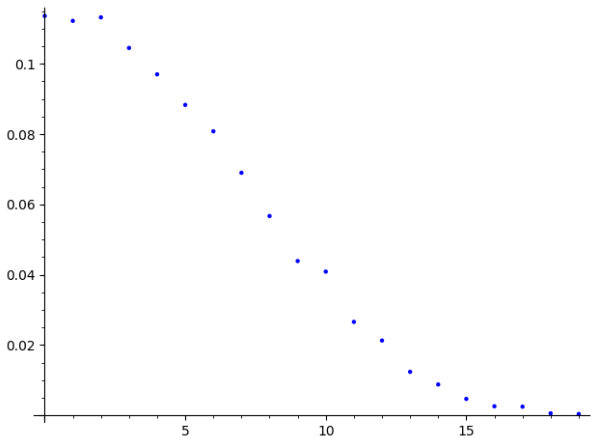
$$\mathbb{E}(Y_{100}) = \mathbb{E}(Y_{2\cdot 50}) = (2\cdot 50+1)\frac{\binom{2\cdot 50}{50}}{2^{2\cdot 50}} - 1 \approx 7.0385.$$

We will now compute N symmetric random walks of length n. Let $Z(k)$ be the amount of random walks whose number of zeroes is exactly k for $0 \le k \le n$. We will then plot the points $(k, Z(k))$ for $0 \le k \le n, Z(k) > 0$.

Here is an example for $N = 10\,000, n = 50$:



Next, let us plot the points $\left(k, \frac{Z(k)}{N}\right)$, where $\frac{Z(k)}{N}$ should be an approximation for the probability $\mathbb{P}\{Y_n = k\}$. The following again uses $N = 10\,000, n = 50$:



One shall not be mistaken if one thinks that this has similarities with the right half of the bell curve. In fact, one has the following theorem, also found in *Irrfahrten* [2]:

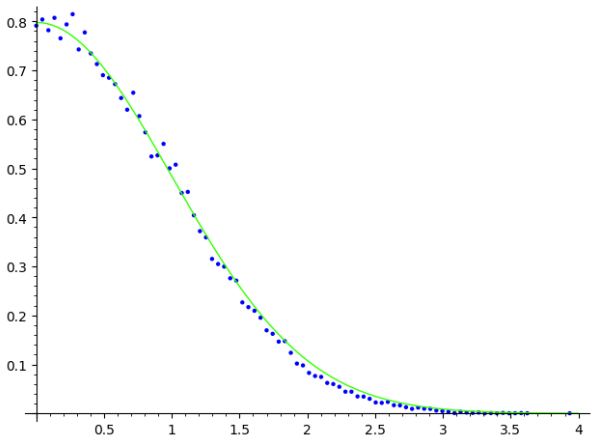
Theorem 2. $\forall x > 0$, *one has*

$$\lim_{n\to+\infty} \mathbb{P}\left(\frac{Y_{2n}}{\sqrt{2n}} \leq x\right) = \sqrt{\frac{2}{\pi}} \int_0^x \exp\left(-\frac{t^2}{2}\right) \mathrm{d}t.$$

This implies that our plot can, in a way, be normalized by representing the points $\left(\frac{k}{\sqrt{2n}}, \frac{\sqrt{2n}}{N} Z(k)\right)$ and should be approximated by the curve of the function

$$f : \begin{cases} \mathbb{R}_{\geq 0} \to \mathbb{R} \\ x \mapsto \sqrt{\frac{2}{\pi}} \exp\left(-\frac{x^2}{2}\right). \end{cases}$$

The following graph, which represents f and the points $\left(\frac{k}{\sqrt{2n}}, \frac{\sqrt{2n}}{N} Z(k)\right)$ for $n = 500$, $N = 60\,000$, shows how well the approximation behaves, even for small n:



1.3 Counting occurrences of a pattern E

The following is partially based on the definitions found in the book *An Introduction to Probability Theory and Its Applications* [1]. Some information on multithreading has been found on the websites [4] and [3].

Definition 3. Let $(S_n)_{n\in\mathbb{N}}$ be a random walk on $\mathbb{Z}$, $E = (X_i)_{0\leq i\leq m}$ a sequence where $X_i \in \{\pm 1\}$. Then we say that E *occurs at position* k, if E occurs in the subsequence $(S_n)_{0\leq n\leq k}$.

Note that if E occurs at position k, we count a second occurrence of E if and only if E occurs in the subsequence $(S_n)_{n>k}$. The same holds for third, fourth, ... occurrences of E. This ensures that the appearance of E qualifies as a recurrent event.

Definition 4. Let $(S_n)_{n\in\mathbb{N}}$ be a random walk on $\mathbb{Z}$, $E = (X_i)_{0\leq i\leq m}$ a sequence, where $X_i \in \{\pm 1\}$. We define the event

$$N_n := \text{"the number of occurences of } E \text{ in } (S_i)_{0\leq i\leq n}\text{"}.$$

We will now study the behaviour of N_n. To do so we will calculate N_n multiple times and plot the relative frequencies of the obtained values. The programming here was done using Python, the generation of our random variables was done using the NumPy and random modules. The plotting was achieved using the matplotlib module. We presented two functions to compute and plot the values of N_n: `run_st()` and `run_mt()` contained in the file `maths_exp_N_n_main.py`.

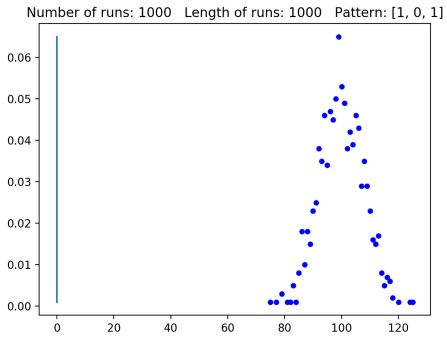
The first version used only one thread and the second one used multiple threads increasing the efficiency of our program for large computations. Our functions took the following inputs:

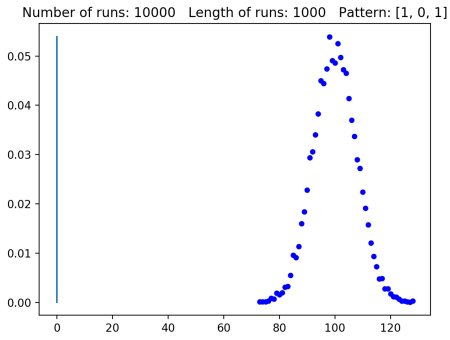
- N: number of values of N_n being computed
- n: length of the runs
- E: a pattern as described in definition 3
- p: probability associated to our random walks (default value: 0.5)
- t: number of threads used (only for `run_mt()`) (default value: 8)

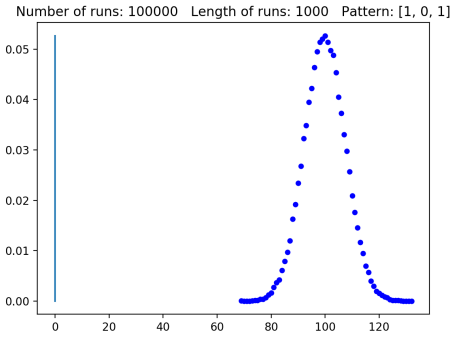
Let us call `run_mt()` with the following inputs for $N \in \left\{10^i : i \in \{3, 4, 5, 6\}\right\}$:

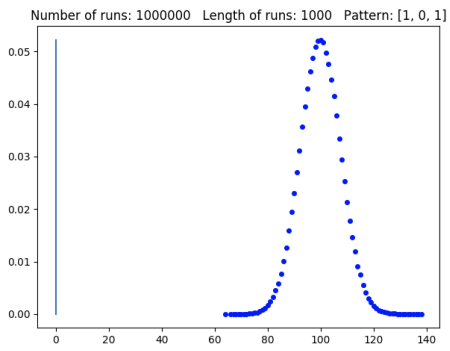
- $n = 1000$
- $E = [1, 0, 1]$
- $p = 0.5$
- $t = 8$

This yielded the following graphs:









We see that when $N \to +\infty$, our graph resembles more and more that of a normal distribution. We were given the following formulae to normalize our graph, i.e. to get an expectation of 0 and a standard deviation of 1:

$$\mu = \frac{1 - p^n}{(1-p) \times p^n}$$

$$\sigma = \sqrt{\frac{1}{((1-p) \times p^n)^2} - \frac{2 \times n + 1}{(1-p) \times p^n} - \frac{p}{(1-p)^2}}$$

$$N_n \mapsto \frac{N_n - n \times \mu}{\sigma \times \sqrt{n}}$$

But for any large values of n, Python rounded the values of some of the denominators to 0, causing zero division errors. Hence, we omitted the normalization for our program. Nevertheless, we can notice that the expectation seems to be around 100 for our inputs given as above.

Definition 5. Let $(S_n)_{n \in \mathbb{N}}$ be a random walk on $\mathbb{Z}$, $E = (X_i)_{0 \leq i \leq m}$ a sequence, where $X_i \in \{\pm 1\}$. We define the event

$$T_r := \text{"the minimal length of a subsequence } (S_i)_{0 \leq i \leq n} \text{ such that } E \text{ occurs } r \text{ times"}.$$

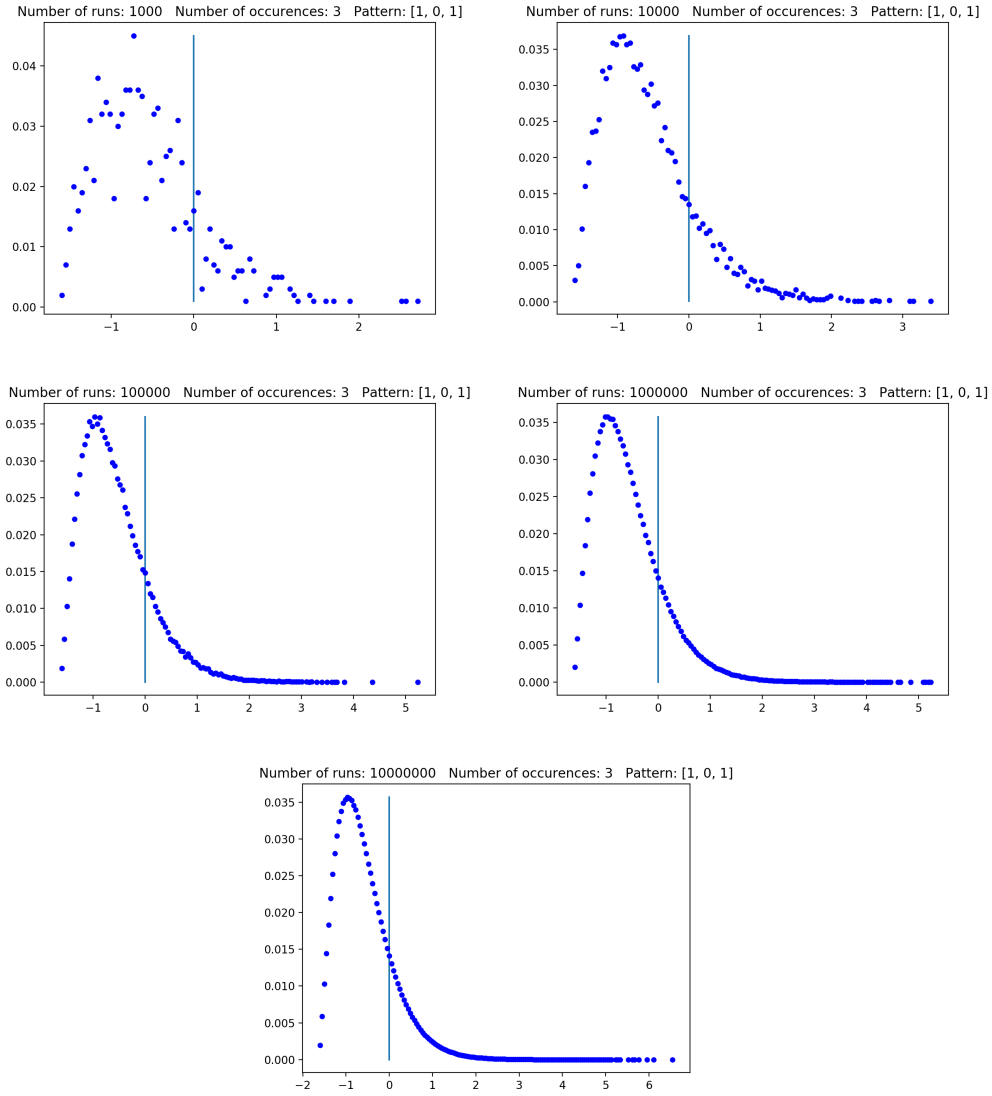
We now want to take a closer look at the behaviour of T_r. We will proceed exactly as before: This time we defined the functions `run_st()` and `run_mt()` contained in the file `maths_exp_T_r_main.py`. Just like before, `run_st()` was a single threaded version of `run_mt()` , which was multi-threaded. They took the following parameters:

- N: number of values of T_r being computed
- r: number of required occurrences of E
- E: a pattern as described in definition 3
- p: probability associated to our random walks (default value: 0.5)
- t: number of threads used (only for `run_mt()`) (default value: 8)

Let us call `run_mt()` with the following inputs for $N \in \left\{10^i : i \in \{3, 4, 5, 6, 7\}\right\}$:

- $r = 3$
- $E = [1, 0, 1]$
- $p = 0.5$
- $t = 8$

This yielded the following graphs:

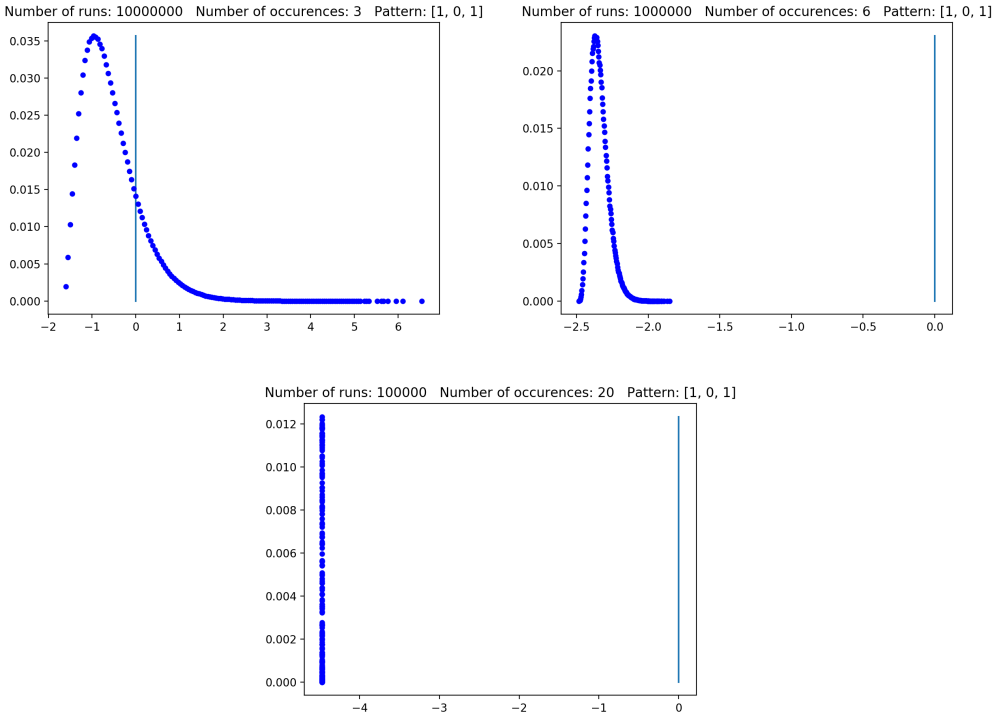


Again, for $N \to +\infty$, our graph starts to resemble that of a normal distribution. Yet we can notice two things:

- The graph seems to be slightly asymmetric, which is not the case for a normal distribution. This is due to the fact that the minimum possible result of T_r is r times the length of E, hence excluding any smaller results.

- The normalization does not centre our graph around the y-axis, i.e. the expectation after the normalization is not 0. We can notice that when we set r to larger values the graph is shifted further and further towards the left. We can also notice that the graph becomes more and more narrow, indicating that the standard deviation is not 1.

One explanation could be that the formula for our normalization does not depend on the length on E (which should impact the expectation) and that it does not depend on the number of 1's and -1's in E (which should also impact the expectation as for example with $p = 0.3$, $(-1, -1)$ will appear far less often than $(1, 1)$).



The formulae used for the normalization of our results were the following:

$$\mu = \frac{1-p^r}{(1-p)\times p^r}$$

$$\sigma = \sqrt{\frac{1}{((1-p)\times p^r)^2} - \frac{2\times r+1}{(1-p)\times p^r} - \frac{p}{(1-p)^2}}$$

$$T_r \mapsto \frac{T_r - r\times\mu}{\sigma\times\sqrt{r}}$$

Remark: After further looking into this topic, we've found out that the distribution we are dealing with is much more likely to be a Rayleigh-distribution.

1.4 Reaching distance N from the origin

Using the analogy of the bunny, the question we are asking is after how many jumps the bunny will be at a distance N from the origin. For this event we will not only limit ourselves to symmetric random walks: We will first consider random walks where $p = \frac{1}{2}$, and then briefly analyse random walks where for other $0 < p < 1$. Let's start with a more formal definition of the problem we are analysing.

Definition 6. Let $(S_n)_{n\in\mathbb{N}}$ be a random walk on $\mathbb{Z}$. Then we define

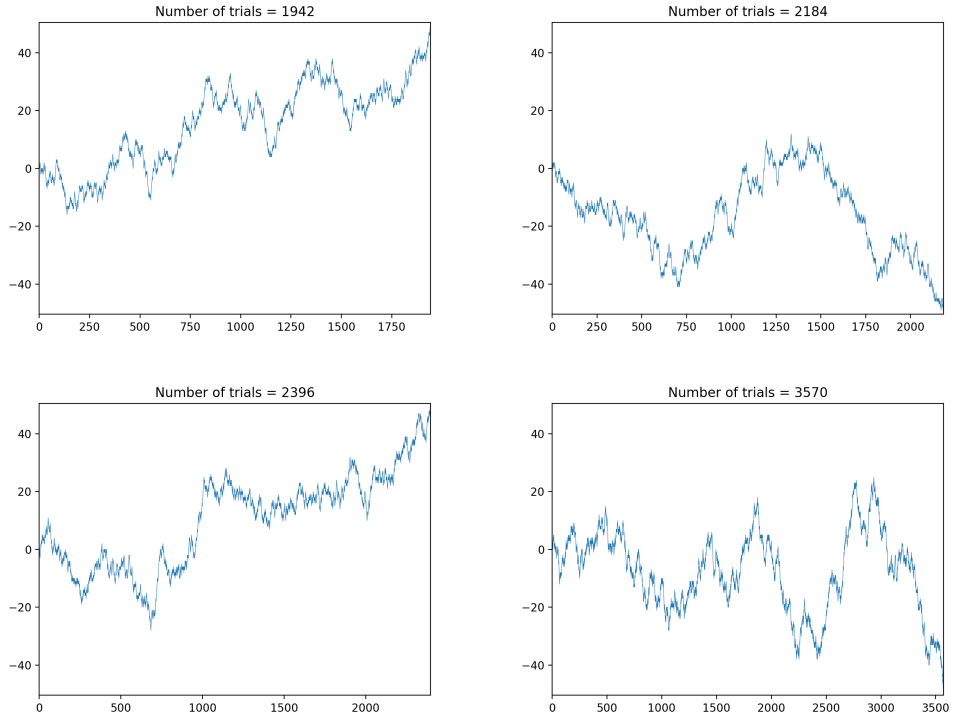
$$d_N := \min\{n \in \mathbb{N} : |S_n| = N\},$$

the first index at which our random walk is at a distance N from the origin.

For the following part we will be using the script `math_exp_random_run_over_N_main.py` coded in Python using the same modules as before. This time we will only use the `run_st()`, as the multi-threaded version turned out to be very inefficient. Multi-threading generates a lot of overhead, which is not alleviated by the multiple threads working simultaneously, as each thread only has to take care of small calculations (generation of random variables). In practice, it is around 100 times slower than the single threaded approach. However, as it does work, it can still be called using `run_mt()`. Both `run_st()` and `run_mt()` use the following as input:

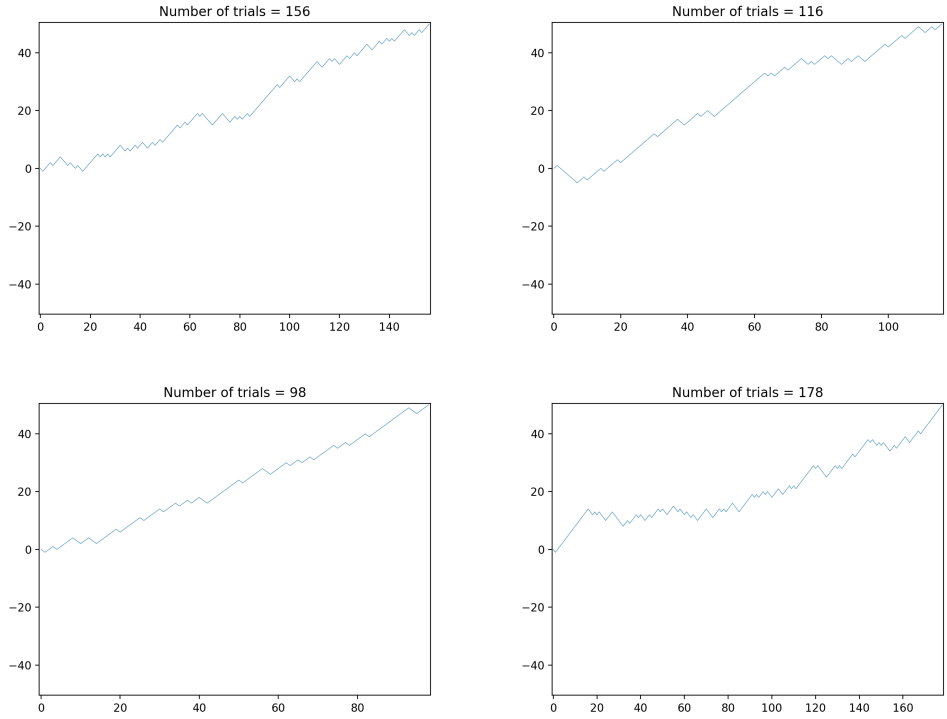
- N: the desired distance to the origin
- p: the associated probability to the random walks (default value: 0.5)

Case 1: $p = \frac{1}{2}$: We consider $N = 50$ and $p = \frac{1}{2}$. We obtain the following images:



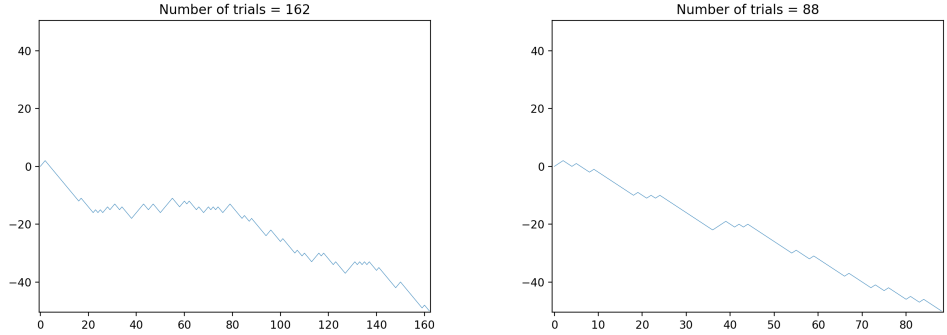
We notice that it takes quite a few steps until our random walk reaches a distance 50 from the origin. After 500 steps most walks will not even have reached a distance of 20 from the origin. It seems that on average it takes around 2 500 steps to reach the desired distance (the average length of our 4 runs here is 2 523). Computing the average of 1 000 runs gives us an average length of 2 514 steps, which is very close to the average over our 4 runs.

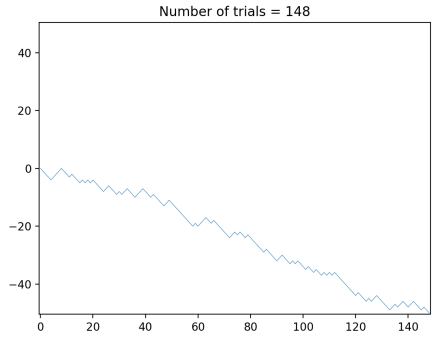
Case 2: $p < \frac{1}{2}$: We consider $N = 50$ and $p = 0.3$. We obtain the following images:

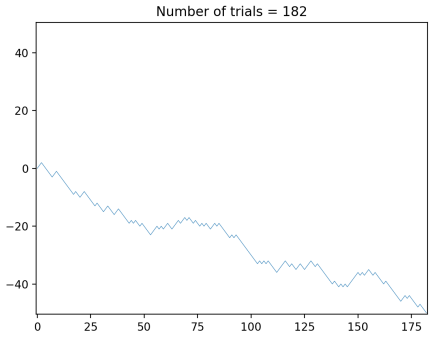


We immediately notice that it takes far fewer steps to reach a distance of 50 to the origin. Whereas in the case $p = \frac{1}{2}$ the number of steps was around N^2, here we notice that it is far closer to $2N$. This makes sense as with $p = \frac{1}{2}$, the random walk tends to stay around the origin, whereas with $p = 0.3$ the random run is far more likely to take a step upwards. This is also confirmed by the fact that our random walk reached $+N$ in all four of our runs, whereas in the case $p = \frac{1}{2}$, it reaches $+N$ only two out of four times. Calculating the average length of 1 000 random walks with associated probability 0.3 until it reaches a distance of $N = 50$ to the origin yields an average of 126 steps which is relatively close to our average of 137 over our 4 example runs.

Case 3: $p > \frac{1}{2}$: We consider $N = 50$ and $p = 0.7$. We obtain the following images:

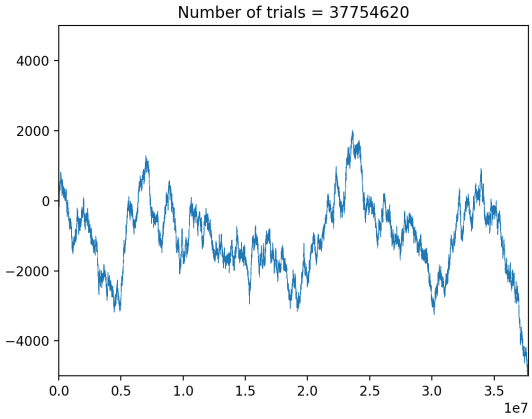




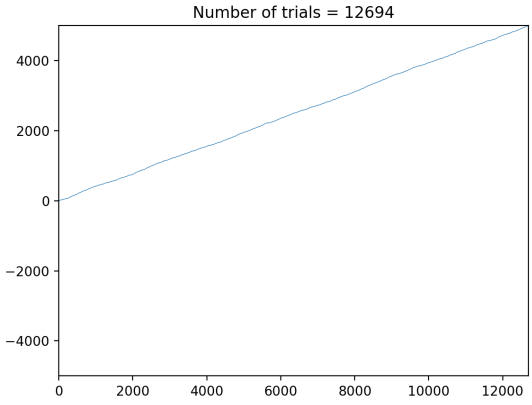


This case is analogous to the previous case because of symmetry.

What we have noticed is that no matter how big we choose N, the random walk will eventually reach a distance of N to the origin. This holds even for $p = \frac{1}{2}$, as one can see in the following example, where $N = 5\,000$. this specific random walk finally reached a distance of N to the origin after over $3.5 \cdot 10^7$ steps.



Much quicker was the case $N = 5\,000$, $p = 0.3$. Here, the random walk reached the wanted distance already after under $13\,000$ steps.



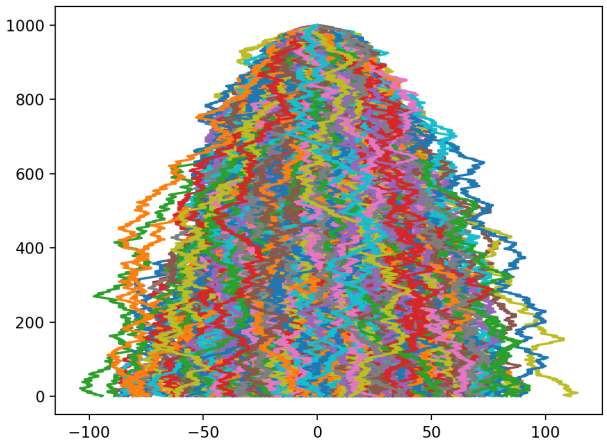
Lastly, we want to plot multiple random walks simultaneously. In addition, we can rotate the graph by 90 degrees to the right and we will notice that if we plot a lot of random walks, we get a shape resembling that of a normal distribution. Here again we presented two approaches using Python, one being multi-threaded, the other being single-threaded. Both approaches took the following as input:

- nbr: the number of random walks to plot
- N: the length of the plotted random walks
- p: the probability associated to the random walks (default value: 0.5)
- t: the number of threads used (only for `run_mt()`) (default value: 8)

We ran `run_mt()` with the following input:

- nbr $= 1000$
- $N = 1000$
- $p = \frac{1}{2}$

and got the graph below:



In order to see whether the multi-threaded approach yielded a faster computation time, we ran our functions `run_st()` and `run_mt()` with $N = \text{nbr} = 10^{(i+1)/2}$, where $i \in \{0, 1, 2, 3, 4, 5, 6, 7\}$. We got the following output in the IDLE Shell:

Multi-Threaded
— 0.060402870178222656 seconds —
Single-Threaded
— 0.010157108306884766 seconds —

Multi-Threaded
— 0.02314901351928711 seconds —

Single-Threaded
— 0.02015066146850586 seconds —

Multi-Threaded
— 0.06358504295349121 seconds —
Single-Threaded
— 0.05956912040710449 seconds —

Multi-Threaded
— 0.2403881549835205 seconds —
Single-Threaded
— 0.24015402793884277 seconds —

Multi-Threaded
— 1.0713260173797607 seconds —
Single-Threaded
— 1.1993682384490967 seconds —

Multi-Threaded
— 7.101611137390137 seconds —
Single-Threaded
— 8.303792715072632 seconds —

Multi-Threaded
— 60.43592023849487 seconds —
Single-Threaded
— 71.8449342250824 seconds —

Multi-Threaded
— 590.5307519435883 seconds —
Single-Threaded
— 732.1697857379913 seconds —

We notice that for our last call of `run_st()` and `run_mt()` the multi-threaded approach was more than 2 minutes quicker than the single-threaded approach. For smaller inputs though the single-threaded approach remained faster.

2 Two-dimensional random walk

As the two-dimensional random walk is much more complex than the one-dimensional one, we will unfortunately not be able to find the same level of detail in the answers to our questions.

In order not to repeat it during the whole section, we shall note this now: When we mention the radius of a square, we are talking about half of the square's side length. This is atypical, but simplifies the below notation.

2.1 Introduction

In this section, we will consider a bunny starting at the origin $(0,0) \in \mathbb{Z}^2$ and taking one jump after the other, either upwards, downwards, to the left, or to the right, each with a probability of $p = \frac{1}{4}$. We are interested in the following questions:

- How can we illustrate the bunny's movement?
- Given the amount of jumps the bunny does, how many different points does it reach?
- If the bunny jumps n times in total, what will be its maximal distance to the origin during its path?
- How many jumps does the bunny need in order to leave a given square centred at the origin?

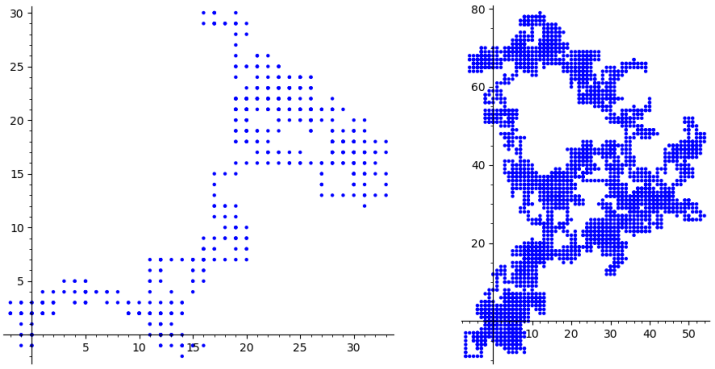
Again, let us start with a formal definition of the symmetric random walk on $\mathbb{Z}^2$.

Definition 7 (Symmetric random walk on $\mathbb{Z}^2$)**.** Let $(X_i)_{i\in\mathbb{N}}$ be a sequence of independent and identically distributed random variables satisfying $\forall i \in \mathbb{N}, \forall\{x,y\} \in \{\{0,\pm 1\}\} : \mathbb{P}\{X_i = (x,y)\} = \frac{1}{4}$. For $n \in \mathbb{N}$, let

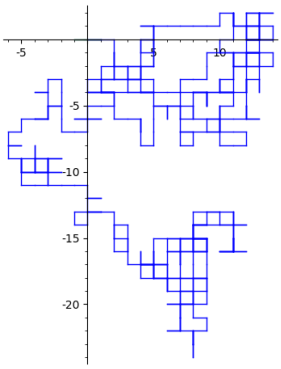
$$S_n := \sum_{i=1}^{n} X_i.$$

Then $(S_n)_{n\in\mathbb{N}}$ is called a *symmetric random walk on* $\mathbb{Z}^2$.

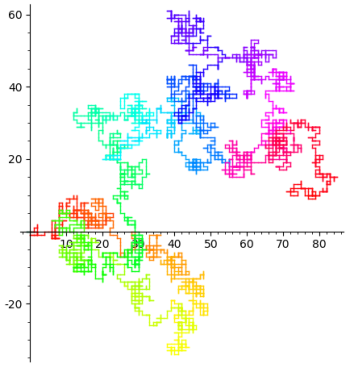
Such a walk can be illustrated by plotting the points S_n in $\mathbb{Z}^2$ in a two-dimensional orthogonal coordinate system, as shown in the following, where we're representing random walks $(S_n)_{0\leq n\leq m}$ for $m = 500, 5000$:



Now, instead of plotting the points, one can also draw the lines connecting S_n and S_{n+1}. The following is representing a random walk $(S_n)_{0 \leq n \leq 500}$:



In order to see the exact course of the random walk, we can add some colour to the lines, changing "continuously" as n increases. The below representation shows a random walk $(S_n)_{0 \leq n \leq 5000}$:



2.2 Number of points reached

How many different points does the bunny reach, given that it jumps n times in total? Let us formulate this question mathematically.

Definition 8 (Number of points reached)**.** Let $(S_n)_{n \in \mathbb{N}}$ be a symmetric random walk on $\mathbb{Z}^2$. For $m \in \mathbb{N}$, let

$$R_m := \#\{S_0, S_1, \ldots, S_m\}.$$

Then R_m is the *number of points reached* for $(S_n)_{0 \leq n \leq m}$.

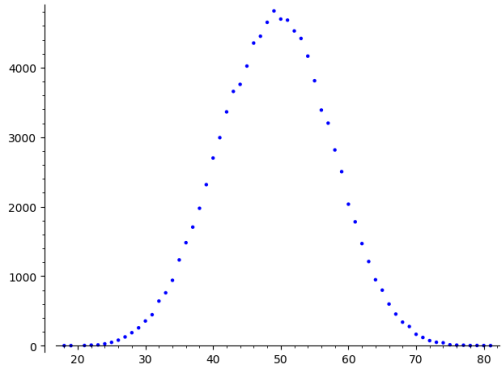
Hence, we are interested in the expected value of R_m. As before, it might make sense to approximate $\mathbb{E}(R_m)$ by computing some random walks of same length m, counting the number of points reached by them, then taking their average.

Using Sage, we get the following approximations for $\mathbb{E}(R_m)$, where we're always taking the average of 1 000 random walks:

m	10	50	100	500	1 000	10 000
$\approx \mathbb{E}(R_m)$	7.278	27.37	48.94	196.4	361.6	2 864

The next step should be to compute N symmetric random walks on $\mathbb{Z}^2$ of length m and count the amount $Z(k)$ of random walks that reach exactly k different points, where $0 \leq k \leq m$. Afterwards, we plot the points $(k, Z(k))$ for $Z(k) > 0$.

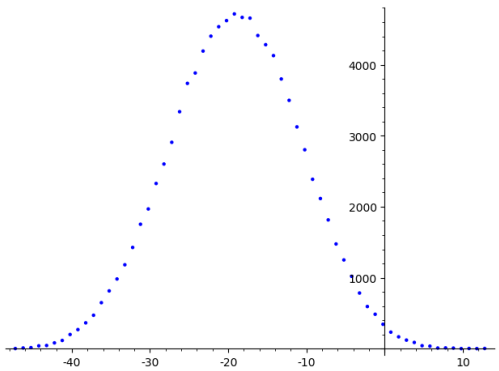
Using Sage, we get for $m = 100, N = 100\,000$:



Aiming to obtain a Gaussian bell curve, we are looking for a way to normalize this curve. As described in *Irrfahrten* [2], we have the asymptotic behaviour

$$\mathbb{E}(R_m) \sim \frac{\pi m}{\ln m}, \ m \to +\infty.$$

Unfortunately, as this clear curve only arises if N is much bigger than m, we cannot choose our m too big, as otherwise we would have problems computing. Choosing $m = 100$, which is small, and $N = 100\,000$, we get the following when plotting $\left(k - \frac{\pi m}{\ln m}, Z(k)\right)$:



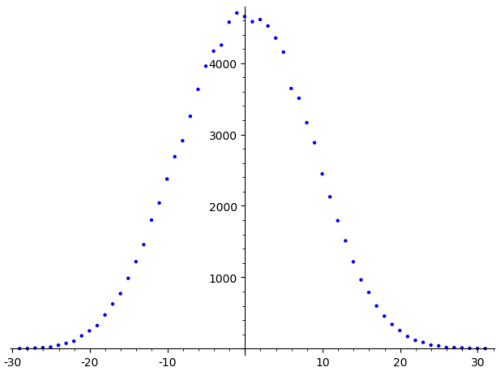
Hence, our m was chosen too small in order to centralize using the asymptotic behaviour of $\mathbb{E}(R_m)$. Fortunately, *Irrfahrten* [2] mentions tight upper and lower boundaries for $\mathbb{E}(R_{2m-1}), m \geq$

7, originally given by P. Erdős and A. Dvoretzky in *Some problems on random walk in space*, namely

$$\frac{2\pi m}{1.16\pi - 1 + \ln m} < \mathbb{E}\left(R_{2m-1}\right) < \frac{2\pi m}{1.066\pi - 1 + \ln m}.$$

Taking the average E_{2m-1} of those upper and lower bounds gives us a better approximation for the expected value of $\mathbb{E}(R_{2m-1})$, at least for small m.

Hence, we can now try to centralize our plotting using E_m by representing $(k - E_m, Z(k))$ for $m = 99$ (has to be odd by above formula), $N = 100\,000$:



We were not able to get closer to the bell curve than this, mostly due to the complexity of R_n and the fact that it is hard to grasp as a random variable (see its definition).

2.3 Maximal distance

Given that the bunny jumps n times in total, we're interested in the maximal Chebyshev-distance to the origin the bunny has during its walk. The following definition should fix this idea in a formal way.

Definition 9 (Maximal distance)**.** Let $(S_n)_{n\in\mathbb{N}}$ be a symmetric random walk on $\mathbb{Z}^2$ and

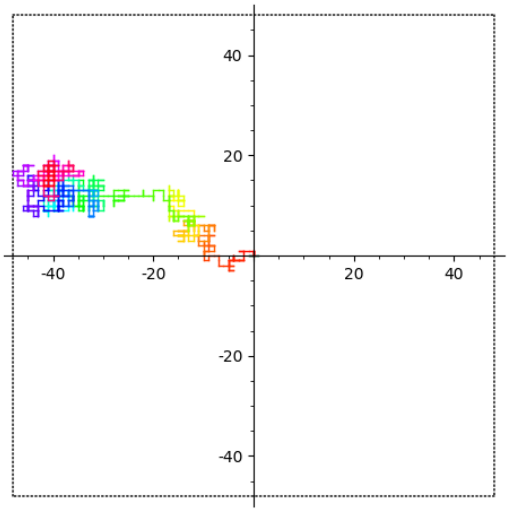
$$d : \begin{cases} \mathbb{Z}^2 \times \mathbb{Z}^2 \to \mathbb{Z}_{\geq 0} \\ ((a,b),(c,d)) \mapsto \max\{|a-c|, |b-d|\} \end{cases} \tag{2.1}$$

the Chebyshev-distance on $\mathbb{Z}^2$. For $m \in \mathbb{N}$, let

$$D_m := \max\{d(S_n, 0) : 0 \leq n \leq m\}.$$

Then D_m is the *maximal distance* of $(S_n)_{0\leq n\leq m}$ to the origin.

Geometrically speaking, we are interested in the radius of the smallest square (whose sides are parallel to the x- and y-axis) centred at the origin that contains our whole random walk. Below is a random walk $(S_n)_{0\leq n\leq 1\,000}$ where $D_{1000} = 48$, which equals the radius of the drawn square:



Our goal is to see how the expected value of D_m depends on m. For this, let us compute the average D_m of 1 000 random walks, which should be an approximation for $\mathbb{E}(D_m)$:

m	10	50	100	500	1 000	10 000
$\approx \mathbb{E}(D_m)$	3.091	7.232	10.342	23.58	33.63	107.9

What one notices is that the approximated values of $\mathbb{E}(D_m)$ are very close to $\sqrt{m}$. Hence, it might make sense to compute $\mathbb{E}(D_m)/\sqrt{m}$ and observe what we obtain.

m	10	50	100	500	1 000	10 000
$\approx \mathbb{E}(D_m)/\sqrt{m}$	0.9775	1.023	1.034	1.055	1.063	1.079

Hence, it appears that

$$\mathbb{E}(D_m) \sim \varphi(m)\sqrt{m},\ m \to +\infty,$$

where $\varphi : \mathbb{N} \to \mathbb{R}$ such that $\exists M \in \mathbb{N} : \forall m \geq M : \varphi(m) > 1$. We were not able to get a better estimation of $\mathbb{E}(D_m)$ for $m \to +\infty$.

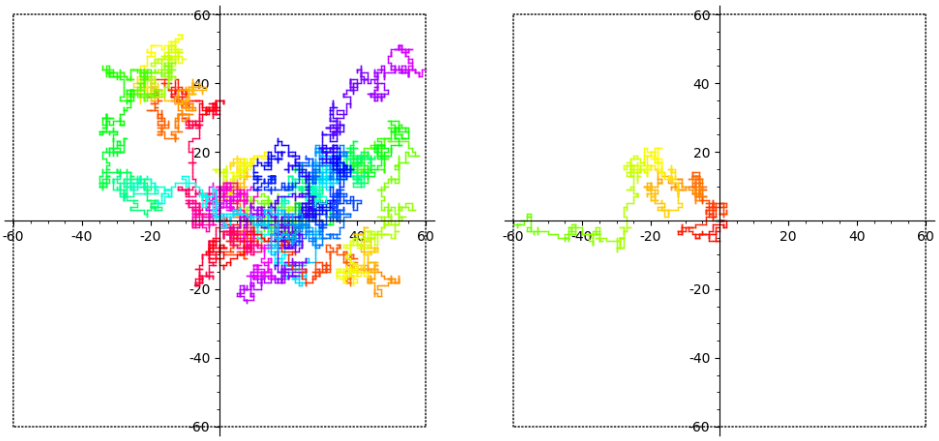
2.4 Leaving square

We want to know how long the bunny needs to leave an (open) square centred at the origin, whose sides are parallel to the x- and y-axis and whose radius equals $r \in \mathbb{N}$. In other words, we are interested in $\mathbb{E}(L_r)$, where

$$L_r := \min\{n \in \mathbb{N} : d(S_n, 0) \geq r\}.$$

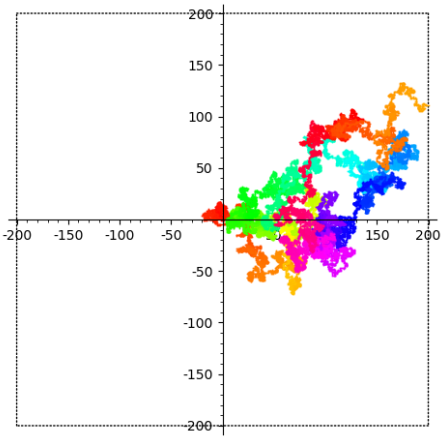
Here, we use the Chebyshev-distance function from (2.1).

For example, we can generate a random walk that will be drawn until it leaves the square of radius $r = 60$ centred at the origin, as is shown in the following 2 random walks:



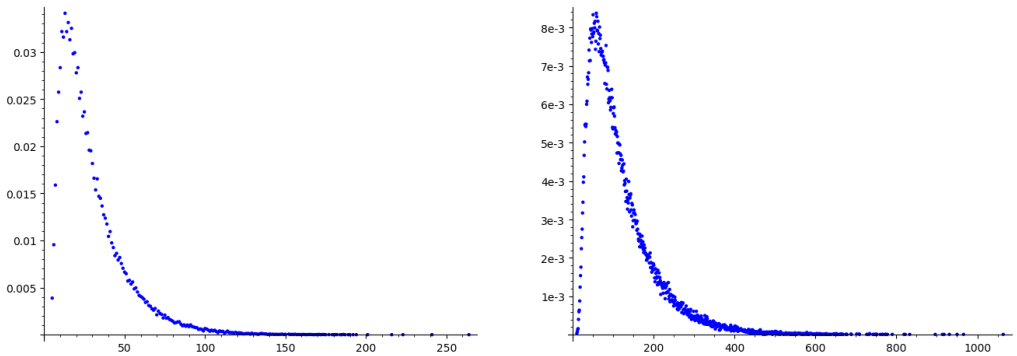
The first random walk needed 10 331 steps to leave the square, the second one only 951 steps. Hence, it is not wrong to assume that the variance of L_r is elevated.

Below is a random walk that needed 44 397 steps to leave the square with radius $r = 200$:



Now, for a fixed $r \in \mathbb{N}$, we compute N random walks on $\mathbb{Z}^2$ and count the amount $Z(k)$ of random walks that needed exactly k steps to leave the square or radius r centred at the origin. Afterwards, we plot the points $(k, Z(k)/N)$, where $Z(k)/N$ should be an approximation of the probability $\mathbb{P}(L_r = k)$. In order to get a meaningful plot, we shall choose N much greater to r, to emulate $N \to +\infty$.

Below are the plots of $(k, Z(k)/N)$, where we chose $r = 5, 10$ and $N = 100\,000$:



One notices similarities with the Rayleigh-distribution.

Bibliography

[1] W. Feller, *An Introduction to Probability Theory and Its Applications, Vol. 1 (v. 1)*, John Wiley and Sons (WIE), 3rd ed., 1968.

[2] N. Henze, *Irrfahrten – Faszination der Random Walks. Ein elementarer Einstieg in die stochastischen Prozesse*, Springer, 2nd ed., 2018.

[3] M. MCCURDY, *Python Multithreading and Multiprocessing Tutorial.*

[4] TutorialsPoint, *Python - Multithreaded Programming.*