Markov Chain Simulations

Moti Ben-Ari

`http://www.weizmann.ac.il/sci-tea/benari/`

February 24, 2023

Contents

Introduction

Simulations are an excellent way of understanding probability, especially, the behavior of processes of long duration. The programs enable the user to perform experiments by varying the parameters of problems interactively and analyzing the results, both printed and displayed in graphs. The simulations are of processes known as *Markov chains*, where the next state of the system depends only on the current state and not on the history of how the process got to the current state.

The definitions of the problems and their theoretical solutions are taken from the references at the end of the document.

The following problems are simulated: the *gambler's ruin* in Section 1, one-, two- and three-dimensional *random walk* in Section 2, the *Ehrenfest model* and the *two-state process* in Section 3.

Technical notes

The programs are written in the Python 3 language and use the `matplotlib` library to generate the graphs. Parameters directly related to the problems, such as the probability of success, can be modified interactively. Others, related to the simulation, such as the number of steps in a simulation, are defined in a module `configuration.py` containing just declarations of values that are easy to modify.

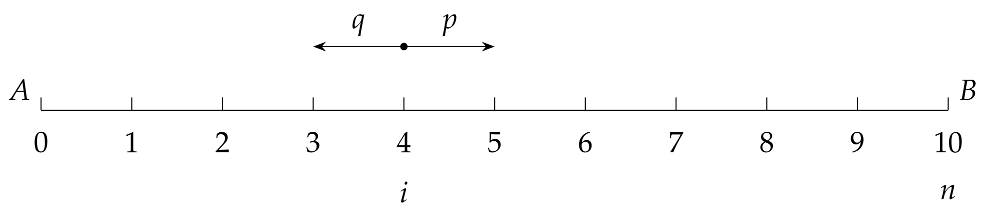
You need to install the Python (`https://www.python.org/downloads/`) although a knowledge of Python programming is not necessary.

To run in the IDLE or Thonny environments, change the configuration constant `CLOSE` to `True`. When the simulation is run multiple times, you will have to close each figure before running a new simulation. This is not necessary if the programs are run in Visual Studio Code or from the command line.

1 Gambler's Ruin

Problem Two players A and B compete in a contest. There is an initial finite capital of n units: A has i and B has $n - i$. They repeatedly play a game where the probability that A wins is p and the probability that B wins is $q = 1 - p$. The loser gives one unit to the winner. When one player has all n units the contest is terminated and that player is declared the winner.

1. Given initial parameters (p, n, i), what is the probability that A wins?
2. What is the expected duration of the game?



The most extensive presentation the gambler's ruin is in [5, Chapter 2] which includes the solution to the expected duration of the contest. Note that Privault asks for the probability that A is ruined, that is, that B wins. I follow other references which ask for A's probability of winning.

1.1 Theoretical results

Given (p, n, i) the probability that A wins the contest is:

$$P_A(p,n,i) = \left(\frac{1-r^i}{1-r^n}\right),$$

where $r = q/p$. By symmetry, the probability that B wins is:

$$P_B(p,n,i) = \left(\frac{1-(1/r)^{n-i}}{1-(1/r)^n}\right).$$

There are separate solutions for $p \neq 1/2$ and $p = 1/2$. For $p \neq 1/2$ the expected duration of the contest is:

$$E_{duration}(p,n,i) = \frac{1}{q-p}\left(i - n\frac{1-r^k}{1-r^n}\right).$$

For $p = 1/2$ the expected duration of the contest is:

$$E_{duration}(p,n,i) = i(n-1).$$

Of course the duration does not depend on which player wins. If A wins, the contest terminates for B also, and conversely.

1.2 Program structure

`configuration.py` contains declarations of variables which are intended to be constant.

`gambler_plot.py` contains the functions for plotting the histogram of the durations of all the runs of the simulation. If the simulations are run for multiple probabilities or initial values, a graph of the proportion of wins is also displayed.

`gamblers_ruin.py` is the main program which obtains the parameters, runs the simulations, prints the output and calls the plotting functions.

1.3 Running the simulations

The program asking the user how to run the simulations and then runs them in a loop. You can run the same simulation again with the saved parameters, enter new parameters, or run a sequence of simulations for a range of probabilities or initial values. Here is an output for 10000 simulations:

```
Probability = 0.45, capital = 20, initial = 8
Wins = 789, losses = 9211, limits exceeded = 0
Proportion of wins     = 0.0789
Probability of winning = 0.0732
Average duration  = 65
Expected duration = 65
```

A graph of the proportion of wins and the histogram of the durations are shown in Figures 1, 2. The vertical lines are the average durations.

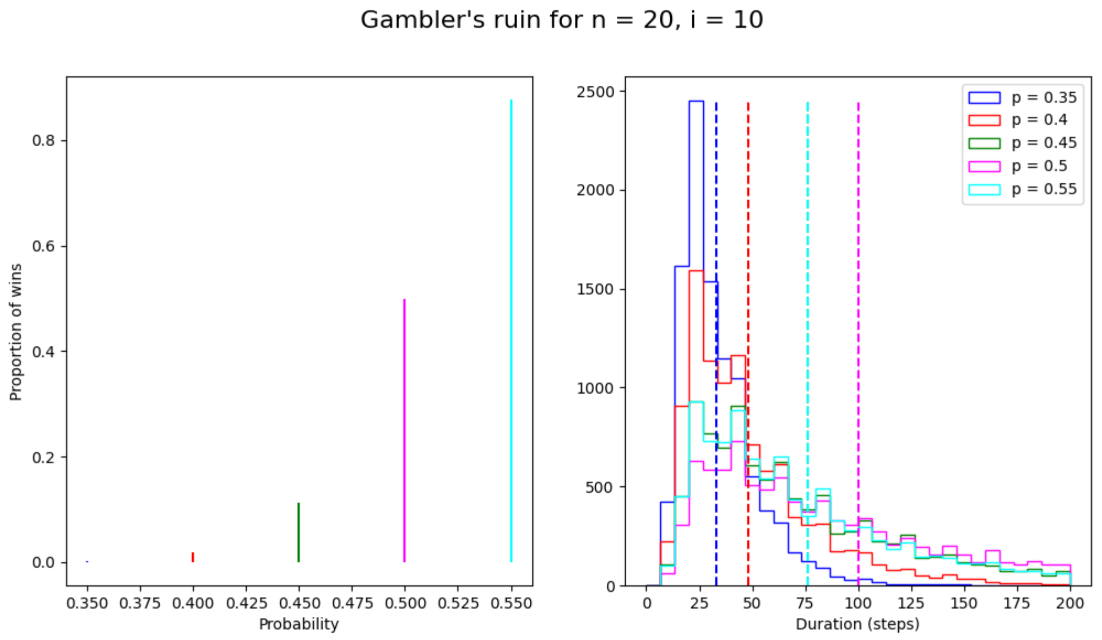


Figure 1: Proportion of wins and histogram for $n = 20, i = 10$ and multiple probabilities

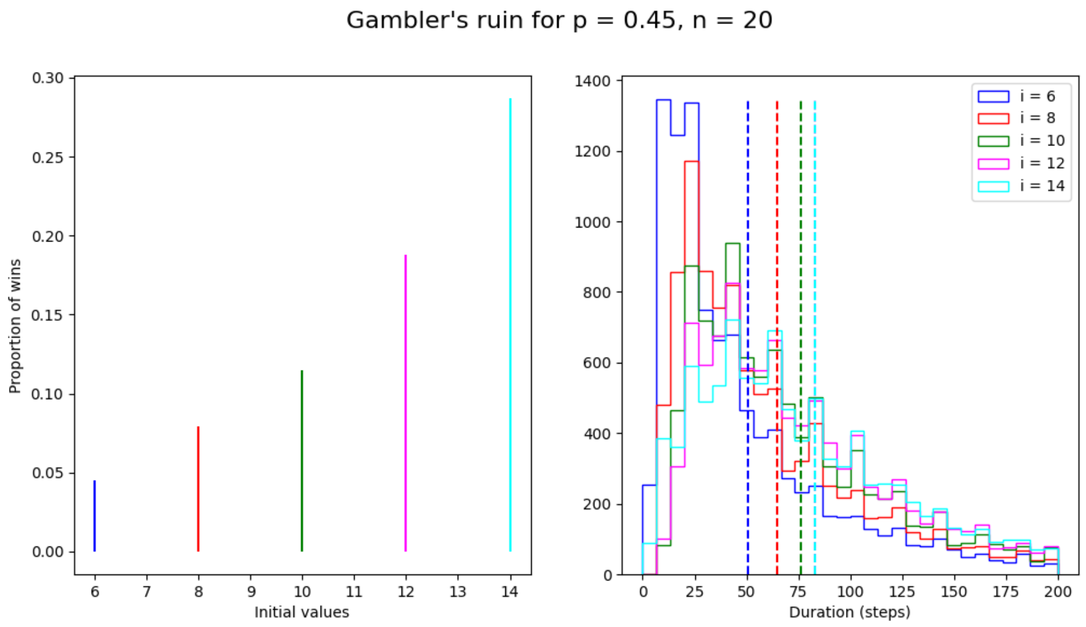
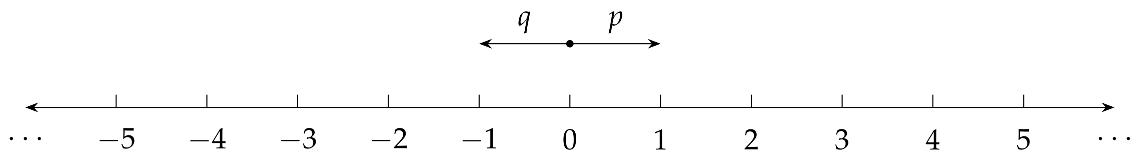


Figure 2: Proportion of wins and histogram for $p = 0.45, n = 20$ and multiple initial values

2 Random Walk (1D, 2D, 3D)

Problem A particle is placed at the origin of the x-axis. It repeatedly takes steps: right with probability p and left with probability $q = 1 - p$.

1. What is the probability that the particle will return to the origin?
2. What is the expected duration until the particle returns to the origin?



The clearest presentation of one-dimensional random walk is in [3], but the derivation of the expected duration is in [5].

2.1 Theoretical results

By symmetry let the first step be to the right.[1] The particle can only return to the origin after an even number of steps. Assume that $p = 1/2$. Let S_{2m} be the position of the particle after $2m$ steps. Then:

$$P(S_{2m} = 0) = \binom{2m}{m} \frac{1}{2^{2m}},$$

which by Stirling's formula is:

$$P(S_{2m} = 0) \approx \frac{1}{\sqrt{\pi m}}.$$

It can now be proved that the probability of a return to the origin is 1.

For $p \leq 1/2$, P_{origin}, the probability of a return to the origin, is 1 and for $p \geq 1/2$ the probability is (Figure 3):

$$P_{origin} = \frac{q}{p} = \frac{1-p}{p}.$$

E_{origin}, the expected duration until the first return to the origin, is infinite for $p \geq 1/2$ while for $p < 1/2$ it is:

$$E_{origin} = \frac{1}{q-p} = \frac{1}{1-2p}.$$

[1]This is equivalent to a first step to the left if p and q are exchanged.

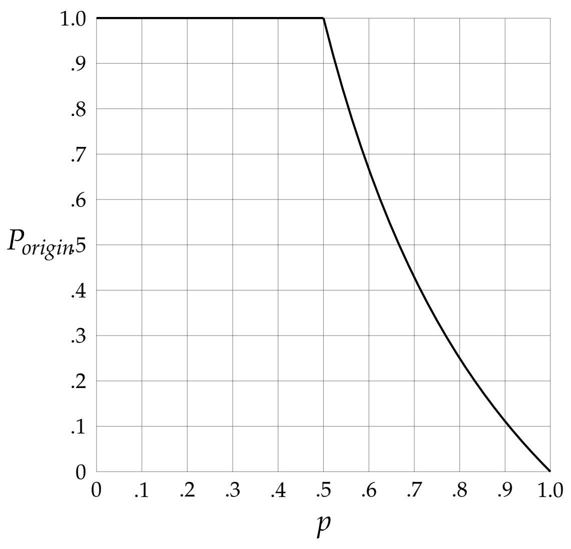


Figure 3: Graph of P_{origin}

2.2 Program structure

`configuration.py` contains declarations of variables which are intended to be constant.

`random_walk_plot.py` contains the functions for plotting a graph of the proportion of simulations that return to the origin and the mean durations if the simulation is run for multiple probabilities or limits.

`random_walk.py` is the main program which obtains the parameters, runs the simulations, prints the output and calls the plotting functions.

2.3 Running the simulations

The program asking the user how to run the simulations and then runs them in a loop. You can run the same simulation again with the saved parameters, enter new parameters, or run a sequence of simulations for a range of probabilities or limits. Here is an output of the simulation:

```
Probability = 0.50, step limit   = 1000
Proportion returning to origin   = 0.977
Probability of return to origin  = 1.000
Proportion reaching limit        = 0.023
Mean duration (steps)            = 49
Expected duration (steps)        = infinity
```

The proportion of wins in the simulation are very close to the theoretical probability, but the mean duration is far from infinite because the step limit was too small. The proportion of wins and the mean durations are shown in Figures 4 and 5.

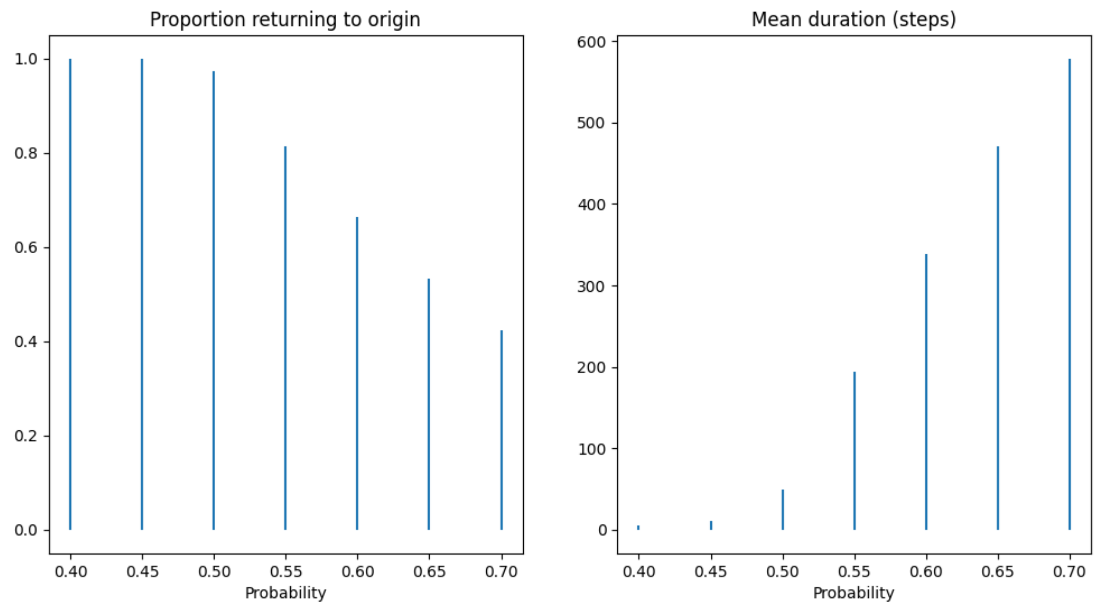


Figure 4: Proportion of returns to origin and and mean durations for multiple probabilities

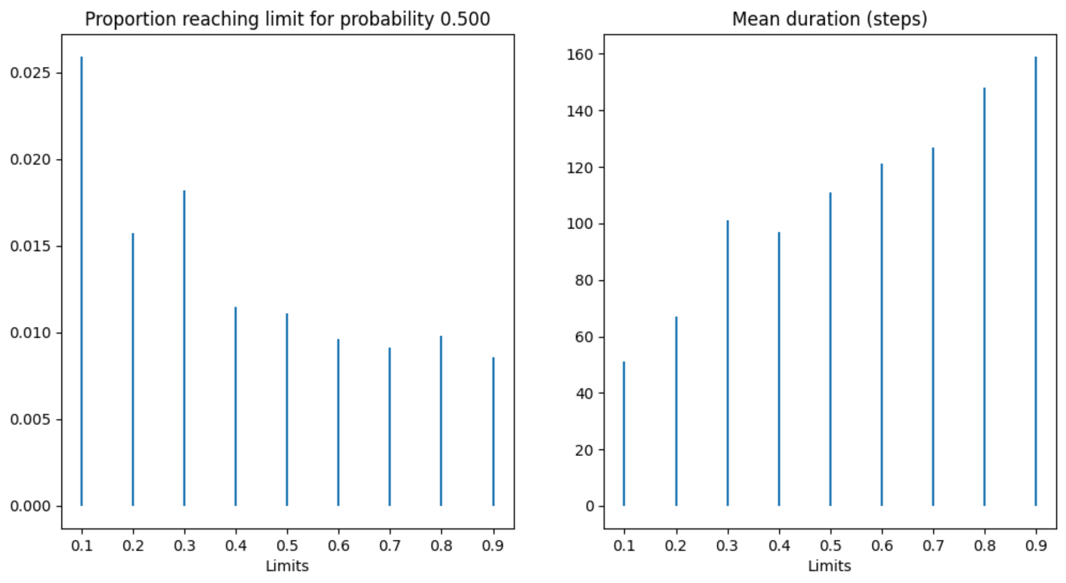


Figure 5: Proportion of returns to origin and mean durations for multiple limits

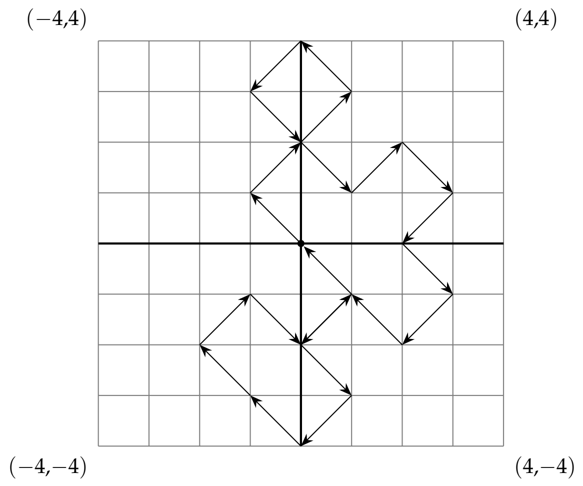


Figure 6: A 22-step two-dimensional random walk

2.4 Two-dimensional random walk

In a two-dimensional random walk [1, 4] a step of the particle consists of one step left or right on the x-axis with probability $1/2$ and simultaneously one step up or down on the y-axis also with probability $1/2$ (Figure 6).

The probability 1 the particle will return to the origin but the expected duration is infinite! Therefore, when you run the simulation with any reasonable limit on the number of steps, the proportion of returns to the origin will be much less than 1 and the mean duration will be quite large:

```
Limit                            = 100000
Proportion returning to origin   = 0.777
Proportion reaching limit        = 0.223
Mean duration (steps)            = 24133
```

The program structure is the same as for the one-dimensional random walk. You can enter the step limit parameter for each simulation interactively and can run the simulations for a range of limits expressed as percentages of the parameter. Figure 7 shows the proportion of returns to the origin and the mean durations for a range of limits.

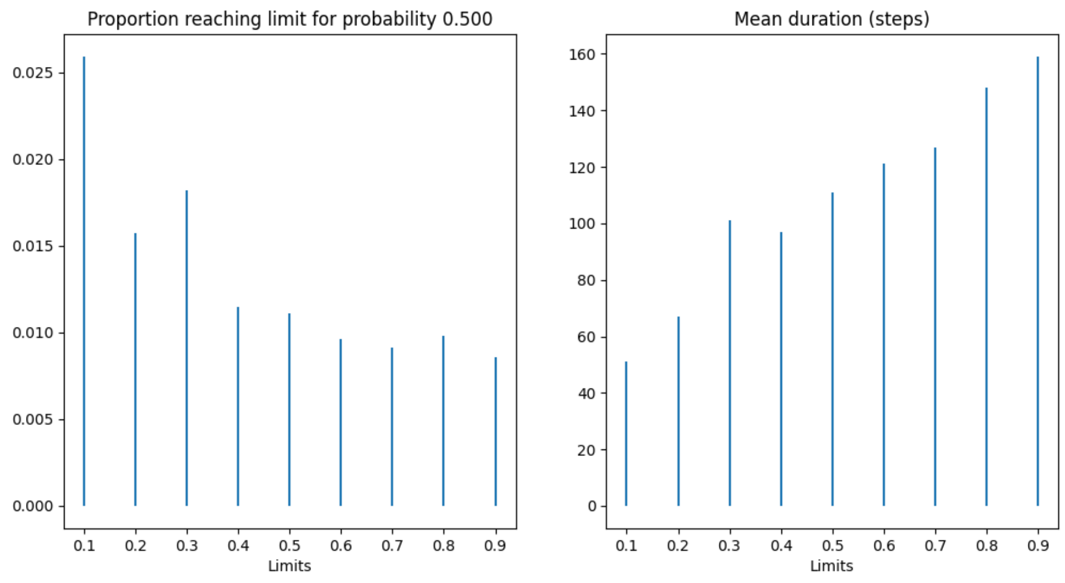


Figure 7: Proportion of returns to origin and and mean durations for multiple limits

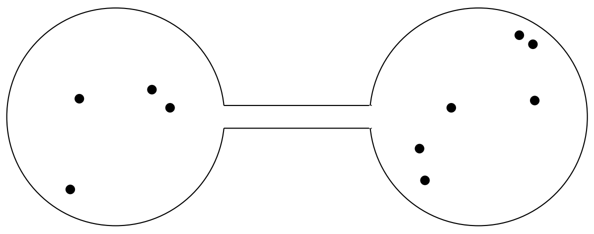
2.5 Three-dimensional random walk

The three-dimensional random walk [1, 4] adds a simultaneous step along the z-axis with probability $1/2$. However, the probability of a return to the origin is only 0.2379 so the simulations will show a large number of simulations reaching the limit:

```
Limit                           = 100000
Proportion returning to origin  = 0.370
Proportion reaching limit       = 0.630
Mean duration (steps)           = 63518
```

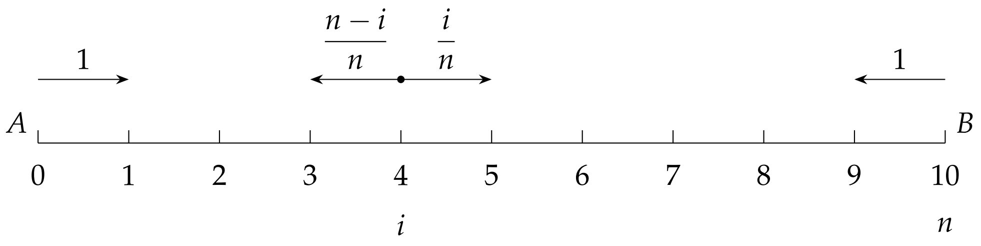
3 Ehrenfest Model and Two-State Process

Problem The Ehrenfest model is designed to model diffusion of particles between two containers. In the following diagram there are four particles in the left container and six particles in the right container for a total of $n = 10$ particles:



Repeated choose a particle at random with uniform distribution and move it to the other container. If there are i particles in the left container then the probability of choosing a particle from the left container is i/n and the probability of choosing a particle from the right container is $(n-i)/n$. If one container is empty the next particle must be chosen from the other container.

The problem is similar to the gambler's ruin except that: (a) the process never ends and (b) the probability of a left or right step changes with each step:



3.1 Theoretical results

The process is a *Markov chain*. Eventually, the process will reach a *stationary distribution*:

$$s_i = \binom{n}{i}\left(\frac{1}{n}\right)^n ,$$

where s_i is the proportion of time that the particle is at the i'th position.

3.2 Program structure

`configuration.py` contains declarations of variables which are intended to be constant.

`ehrenfest_plot.py` contains the functions for plotting the distribution, both the theoretical distribution and the result of the simulation.

`ehrenfest.py` is the main program which obtains the parameter n, runs the simulations, prints the output and calls the plotting functions.

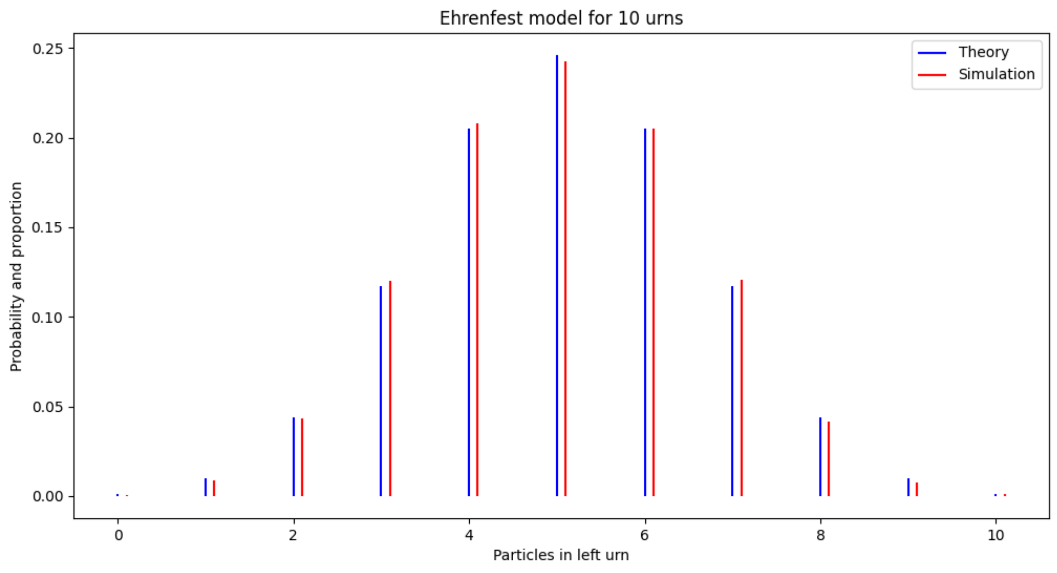


Figure 8: Stationary distribution for the Ehrenfest model

3.3 Running the simulations

The program asks the user how to run the simulation: with the saved value of n or with a new value of n. Here is an output of the simulation:

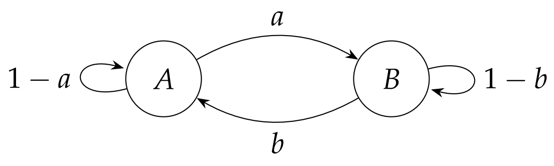
```
Total particles in urns = 10
Theoretical stationary distribution
[0.001 0.01  0.044 0.117 0.205 0.246 0.205 0.117 0.044 0.01  0.001]
Simulation stationary distribution
[0.001 0.009 0.044 0.12  0.208 0.243 0.205 0.121 0.042 0.008 0.001]
```

A graph of these distributions is shown in Figure 8; the theoretical distribution and the result of simulation are so close together that the lines are slightly offset.

3.4 The two-state process

The two-state process is similar to the Ehrenfest model in that the probabilities at each step are different and we are interested in the stationary probability distribution of the unbounded process.

There are two states A, B. When the process is in state A, with probability a it transitions to B and with probability $1 - a$ it remains in A. Similarly, the probability of a transition from B to A is b:


$$\begin{tikzcd} A \arrow[loop left, "1-a"] \arrow[r, bend left, "a"] & B \arrow[l, bend left, "b"] \arrow[loop right, "1-b"] \end{tikzcd}$$

The stationary distribution is:

$$\left[\frac{b}{a+b}, \frac{a}{a+b}\right].$$

Here is an output of the simulation:

```
Probabilities:  a = 0.500, b = 0.333
Theoretical stationary distribution: a = 0.400, b = 0.600
Simulation  stationary distribution: a = 0.402, b = 0.598
```

References

[1] Moti Ben-Ari. Mosteller's challenging problems in probability. `https://github.com/motib/probability-mosteller`, 2022.

[2] Joseph K. Blizstein and Jessica Hwang. *Introduction to Probability (Second Edition)*. CRC Press, 2019.

[3] K.C. Border. Lecture 16: Simple random walk. `http://www.math.caltech.edu/~2016-17/2term/ma003/Notes/Lecture16.pdf`, 2017.

[4] Frederick Mosteller. *Fifty Challenging Problems in Probability with Solutions*. Dover, 1965.

[5] Nicolas Privault. *Understanding Markov Chains: Examples and Applications (Second Edition)*. Springer, 2018.

[6] Sheldon Ross. *A First Course in Probability (Tenth Edition)*. Pearson, 2019.