A STATISTICAL ANALYSIS OF SNAKES & LADDERS

Gabriele Pinna
University of Birmingham
School of Physics and Astronomy
Total words: 3417

ABSTRACT

This paper investigates various features of the popular board game Snakes and Ladders. Three methods are used for the sake of comparison. The first is based on a simulation of the game using a random integer generator. The second relies on techniques from Markov chain theory. The last one is an optimisation of the second using the properties of absorbing Markov matrices. The three methods yield similar values for the average duration. The first method gives a duration of 39.236 turns, using 10^6 iterations. The second gives 39.225 turns, with 10^4 iterations, and the third returns 39.225 turns. Those calculations are based on the classic version.

Keywords Markov chain · Monte Carlo simulation · Snakes and Ladders

1 Introduction

Snakes and Ladders is a popular tabletop game; the most common board (Figure 1) is ten by ten with nine ladders and ten snakes.

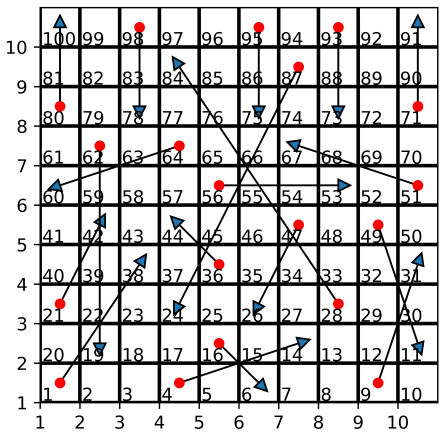


Figure 1: Traditional board with 10 snakes and 9 ladders substituted by arrows for simplicity.

The player rolls a dice (or rotates a spinner) and moves accordingly starting from outside the board (as if there were a square 0). The first player to reach the final square wins the game. The snakes and ladders randomise the process and lengthen the duration of the game. When a player lands at the base of a ladder, then the person climbs to the top of it. If a player lands at the top of a snake, the game piece ends up at the bottom of it. Three variations will be considered regarding the winning condition:

- Classic version: the player rolls the exact number that leads to the final square,
- Fast version: the player rolls a number that leads to the final square or beyond,
- Bounce back version: the player bounces back off the end of the board if overshooting occurred.

The principal aim is to determine the average duration of the game. This problem has been analysed by Althoen, King, and Schilling (1973) [1] using two approaches: a random number generator to simulate the game and Markov matrix theory. The present paper tries to analyse the mathematics behind the game, using three methods. The first method relies on a Monte Carlo simulation; the game is randomly simulated more than 10^4 times to acquire a sample of data. The second method relies on a probabilistic view based on Markov chains; the board is mapped onto a matrix whose elements represent the transition probabilities from a square to another. The third method is an optimisation of the second, based on the properties of the absorbing Markov matrices.

2 Monte Carlo simulation

A Monte Carlo simulation consists of generating random variables for modelling a real game situation. The simulation is made using the numpy.random.int function which generates random integer numbers on a specified interval, in that case, from 1 to 6. The approach aims to store the number of turns required to finish the game for each simulation. Consequently, samples of data are acquired to perform a statistical analysis. The relevant quantities are the mode, the average, the variance, the median, skewness, and kurtosis [1]. An interesting data point is the minimum, which is found to be 7. This corresponds to the shortest game, provided that the iteration is large enough. The data distribution is illustrated in Figure 2 trough a histogram plot.

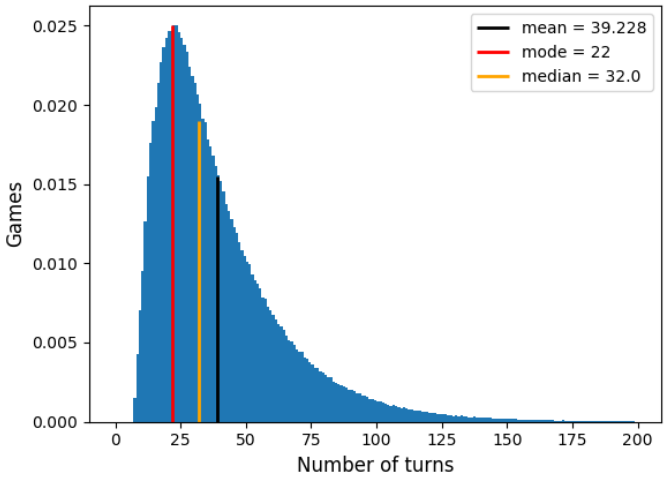


Figure 2: Normalised probability density function generated using Monte Carlo simulation with 10^6 iterations. The winning condition is classic. The standard deviation is $\sigma = 25.225$, the skewness is $s = 1.812$, and the kurtosis is $k = 8.168$.

The distribution is asymmetric with a positive skewness of $s = 1.812$. The mean is 39.228 turns, which is the average duration of the classic version of Snakes and Ladders. The function is used to calculate the cumulative distribution, illustrated in Figure 3, through integration.

[1] Consult Appendix 1 for definitions

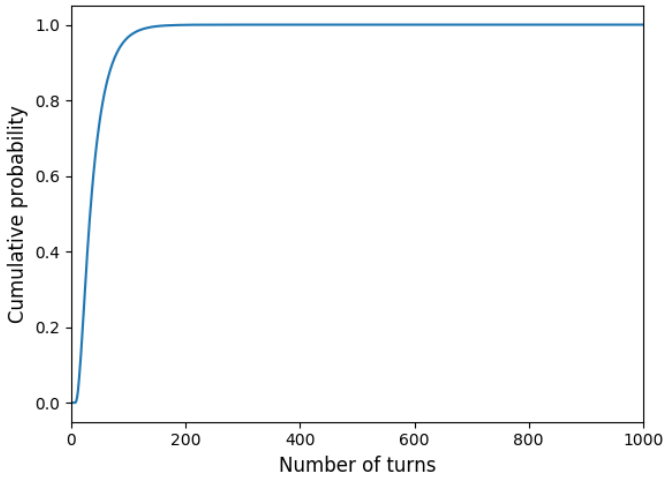


Figure 3: Cumulative distribution function generated using Monte Carlo simulation with 10^6 iterations. The version is classic.

Figure 3 shows how the cumulative probability of finishing the game tends to 1 as the number of turns increases; therefore the game will eventually end. The Monte Carlo method is used to compare the average duration of the three versions, as shown in Table 1.

Table 1: Average turns to finish the game.

Iterations	Average turns (classic)	Average turns (fast)	Average turns (bounce back)
10^3	39.386	37.408	43.072
10^4	38.897	35.765	43.164
10^5	39.275	35.942	43.290
10^6	39.236	35.828	43.338
2×10^6	39.273	35.822	43.346

The method confirms the intuition that the shortest variant of the game, on average, is the fast version. The variance is also calculated to asses how far the set of data is spread out. The result is presented in Table 2.

Table 2: Variance on the number of steps to finish the game.

Iterations	Variance (classic)	Variance (fast)	Variance (bounce back)
10^3	624.146	439.189	848.910
10^4	620.999	544.457	892.056
10^5	632.177	550.183	912.043
10^6	637.187	541.925	911.694
2×10^6	636.290	546.477	922.314

These results imply that the bounce back version has a more dispersed distribution. One of the major drawbacks of this approach is its instability. In fact, a stable result is produced for large iterations only. Consequently, the algorithm is computationally expensive.

3 Markov chain

3.1 Theory

A more efficient approach, which involves a probabilistic view, is considered. This is based on Markov chains which represent stochastic processes. A stochastic process is defined as a collection of random variables $X(t)$ ordered according to a set T [2], formally:

$$\{X(t), t \in T\}. \tag{1}$$

Usually, t is interpreted as time and $X(t)$ as a value observed at time t. A random variable can be informally thought as a real number from a set assigned to each outcome of a random experiment. A stochastic process is a Markov chain if the probability of a future event depends only on the present situation and not on the past. This condition can be expressed mathematically. Let $p(b|a)$ be the conditional probability of an event b occurring, given that the event a occurred. Let also $x_n = i_n$ be an event at a time n, then:

$$p(x_{n+1} = j|x_n = i_n, x_{n-1} = i_{n-1}, ..., x_1 = i_1, x_0 = i_0) = p(x_{n+1} = j|x_n = i). \tag{2}$$

Therefore, the knowledge of the past is redundant and does not affect the probability of a future event. This is known as Markov property which is satisfied by Snakes and Ladders. In fact, the transition probability does not depend on the path that leads to the present state. An extra assumption is needed, namely that the probability is time-independent. In that case, the Markov chain is known as homogeneous. Mathematically, this condition can be expressed as follows:

$$p(x_{n+1} = j|x_n = i) = p_{ij}, \tag{3}$$

where p_{ij} is the transition probability from state i to j which is n independent, and thus time-independent. A Markov chain can be represented using a transition (or Markov) matrix P, where the entry $(P)_{ij}$ corresponds to the transition probability p_{ij}. At this point, an example is examined to understand the concept.
Consider the following chain with 4 states.

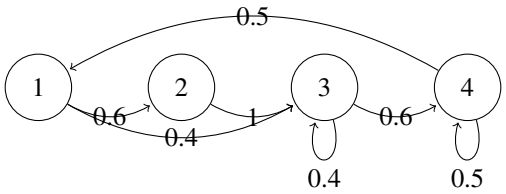


Figure 4: Example of a Markov chain

The Markov chain shown in Figure 4 is represented by the following 4×4 transition matrix:

$$P = \begin{pmatrix} 0 & 0.6 & 0.4 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0.4 & 0.6 \\ 0.5 & 0 & 0 & 0.5 \end{pmatrix}. \tag{4}$$

Since probabilities sum up to one, the sum of the elements in each row is also one. This is used to define a Markov matrix as a real square matrix with each row summing to one. Some authors represent it differently, requiring the column sum to be one. The first convention is used in this paper.

3.2 Application to Snakes & Ladders

A 101×101 matrix is used to describe the Snakes and Ladders board game. The size of the matrix is 101×101 because the initial state is equivalent to a square 0 located outside the board. This method requires matrix multiplications which is legitimate because the product of two transition matrices is a transition matrix. This can be proved as follows, let A and B be two transition matrices and $AB = C$, then, by definition:

$$\sum_j a_{ij} = 1, \tag{5}$$

$$\sum_k b_{jk} = 1. \tag{6}$$

In order for C to be a transition matrix, the sum of the elements of a row needs to be one. From the definition of matrix multiplication:

$$c_{ik} = \sum_j a_{ij} b_{jk}, \tag{7}$$

then,

$$\sum_k c_{ik} = \sum_j a_{ij} \sum_k b_{jk} = \sum_j a_{ij} = 1. \tag{8}$$

Therefore, the product of two transition matrices is a transition matrix. This result is fundamental because the Markov matrix describing the board is generated using the product between a matrix corresponding to the rolling of the dice and a matrix describing the location of the snakes and ladders. Let R be the matrix that represents the dice, then

$$(R)_{ij} = \frac{1}{6}\delta_{i,j-n}, \tag{9}$$

where $i \leq 95$ and $n = \{1, 2, ..., 6\}$. For $i \geq 96$ the dice matrix depends on the end-condition.
For the classic version, the remaining probability is added to the diagonal elements. This reflects the fact that the player does not move if a number that leads to past the final square is rolled. Therefore,

$$\begin{aligned} (R)_{ij} &= \frac{1}{6}\delta_{i,j-n}, \\ (R)_{ii} &= 1 - \sum_{j \neq i} r_{ij}. \end{aligned} \tag{10}$$

For the fast version, the remaining probability is added at the end of the row:

$$\begin{aligned} (R)_{i,j\neq 101} &= \frac{1}{6}\delta_{i,j-n}, \\ (R)_{i,101} &= 1 - \sum_{j \neq 101} r_{ij}. \end{aligned} \tag{11}$$

For the bounce back version, the remaining probability is redistributed as if the starting point is the final square:

$$(R)_{ij} = \begin{cases} \frac{1}{6}\delta_{i,j-n} & i \leq 95 \\ \frac{1}{6}(\delta_{i,j-n} + \delta_{i,202-j-n'}) & i \geq 96 \end{cases}. \tag{12}$$

Note that this way of defining the dice matrix for the bounce back version only works if the spinner range does not exceed the board size. If so, the player might bounce back off the first square as well as the final square. This requires extra calculations to determine how to redistribute the probability, which are beyond the scope of this paper.
The matrix corresponding to the snakes and ladders S is similarly defined. Let $\{a_k \to b_k\}_{k=1}^{n}$ be a sequence describing the starting (a_k) and the ending (b_k) positions of n snakes and ladders, then

$$(S)_{ij} = \begin{cases} (S)_{a_k+1,b_k+1} = 1 & \\ (S)_{i,i} = 1 & i \neq a_{k+1} \\ (S)_{i,j} = 0 & elsewhere \end{cases}. \tag{13}$$

The indices are shifted by one because the transition matrix is 101×101 rather than 100×100. It follows that the transition matrix M describing the board is given by:

$$M = RS. \tag{14}$$

Note that it is possible to define the M matrix as SR, in that case, the player waits for an additional turn to climb a ladder or to go down a snake. This version is not considered because it reproduces results very similar to those that will be obtained.
The evolution of the initial row vector $|p(0)\rangle$, after n turns, is computed by repeatedly multiplying it by M from the right. Thus,

$$|p(n)\rangle = |p(0)\rangle M^n, \tag{15}$$

where $|p(n)\rangle$ is the state vector after n turns. The entry $p(n)_i$ represents the probability of being in state i after n turns starting from state $|p(0)\rangle$. Consequently, the probability density and the cumulative distribution functions can be generated. The cumulative probability of finishing the game after n turns, from an initial state $|0\rangle$, is given by the last entry of $|0(n)\rangle$. Therefore, the evolution of the entry as a function of the matrix power n defines a vector $\boldsymbol{v}$ that

represents the discrete version of the cumulative distribution function (Figure 5). The index of the first non zero element of the vector v corresponds to the shortest game, which is found to be 7.

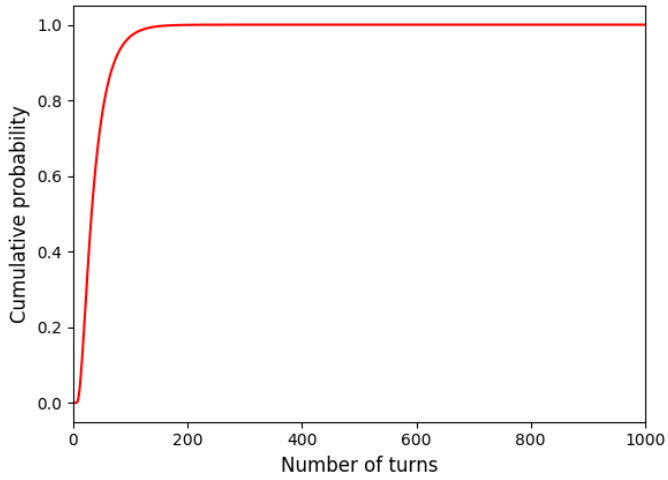


Figure 5: Cumulative distribution function generated via Markov transition matrices, using up to 10^4 matrix multiplications. The ending condition is classic.

The probability density function is obtained differentiating the cumulative distribution function. The finite difference of the cumulative vector $\boldsymbol{v}$ yields a new vector $\tilde{\boldsymbol{v}}$ which represents the discrete version of the probability density function (Figure 6).

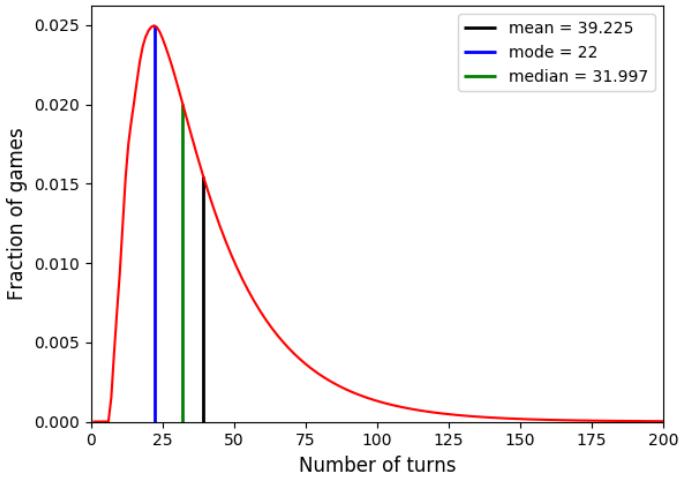


Figure 6: Probability density function generated via Markov transition matrices using up to 10^4 matrix multiplications. The ending condition is classic. The standard deviation is $\sigma = 25.225$. The skewness is $s = 1.822$ and the kurtosis is $k = 8.217$.

The expectation value is found using a dot product between $\tilde{\boldsymbol{v}}$ and a vector, of the same size, containing the integers starting from one, representing the number of turns. The average values, for the different versions, are shown in Table 3.

Table 3: Average duration.

Matrix power	Duration (classic)	Duration (fast)	Duration (bounce back)
10^2	35.2297073221362	32.9984742414180	35.9930614236645
2×10^2	39.1064602907142	35.7704354795212	42.8732196807671
10^3	39.2251223082352	35.8349384136953	43.3245974417222
10^4	39.2251223082316	35.8349384136926	43.3245974417248

The variances are illustrated in Table 4.

Table 4: Variance on the steps.

Matrix power	Variance (classic)	Variance (fast)	Variance (bounce back)
10^2	419.8630260359505	381.58232760330293	498.9582898279468
2×10^2	618.7177974268732	535.5268352789949	849.0016887274271
10^3	636.2984613451874	545.4000999857217	915.3193155118227
10^4	636.2984613124252	545.4000999576851	915.3193154997775

The accuracy is up to 13 decimal figures, to better visualise the stability of the method as the matrix power increases.

4 Absorbing Markov chains

4.1 Theory

The last method relies on the properties of absorbing Markov matrices. They describe Markov chains with one or more absorbing states which, if entered, cannot be left. This implies that the entry of the transition matrix P describing such a state satisfies $p_{ii} = 1$. The Snakes and Ladders board game has an absorbing state represented by square 100, thus $(M)_{101,101} = 1$. These matrices are advantageous to determine statistical quantities, including the mean and the variance [3]. An essential property is that the probability of reaching the absorbing state in n steps tends to 1 as $n \to \infty$. This can be proved considering two cases. If the process starts in an absorbing state, then, by definition, the probability of remaining in this state is 1. On the other hand, if the process begins in a transient state, it is possible to reach an absorbing state in at most m steps. Therefore, there exists a number p for which the probability of being absorbed, in at most m steps, is at least p, from each transient state. Consequently, the probability of not being in an absorbing state, after at most m steps, is at most $1 - p$. Consider km steps, the probability of not being in an absorbing state is at most $(1 - p)^k$, but for $k \to \infty$, $(1 - p)^k \to 0$. Hence, after infinite steps, an absorbing state is always reached [3]. An absorbing matrix M can be written in the canonical form, where the transient states and the absorbing states are aggregated. Suppose that there are s transient states and r absorbing states, then [3]:

$$P = \begin{pmatrix} Q_{s\times s} & R_{s\times r} \\ 0_{r\times s} & I_{r\times r} \end{pmatrix}. \tag{16}$$

The matrix Q corresponds to transitions to transient states only, the identity represents the absorbing states, and R represents transitions from the transient to the absorbing states. For calculation purposes, it is crucial to define the fundamental matrix N:

$$N = (I - Q)^{-1} = \sum_{k=0}^{\infty} Q^k. \tag{17}$$

This sum converges since Q^k tends to zero as k tends to infinity. The entry of the fundamental matrix $(N)_{ij}$ represents the average number of times the process is in state j, given that the chain started in state i. To prove this result, it is sufficient to compute the quantity $\langle n_{ij} \rangle$, which corresponds to the average number of times the process is in the transient state j starting from state i. It follows,

$$n_{ij} = \sum_{k=0}^{\infty} u_{ij}^k, \tag{18}$$

where u_{ij}^k is one if the process is in state j, starting from state i, after k steps and zero otherwise.
The average is

$$\langle n_{ij} \rangle = \sum_{k=0}^{\infty} \langle u_{ij}^k \rangle = \sum_{k=0}^{\infty} p_{ij}^{(k)} \rightarrow \sum_{k=0}^{\infty} Q^k \equiv N, \tag{19}$$

where $p_{ij}^{(k)}$ is the probability of reaching the transient state j, starting from the transient state i, after k steps. In the final equality, $p_{ij}^{(k)}$ is recognised as the matrix element $(Q^k)_{ij}$. Therefore, $\langle n_{ij} \rangle$ is the entry of the fundamental matrix.

4.2 Application to Snakes & Ladders

The fundamental matrix N plays a crucial role: it simplifies the analysis of Snakes and Ladders. In the present case, the matrix contains only one absorbing state: square 100. Using the fundamental matrix N, it is possible to find the most visited site starting from square 0. This is given by the entry with the highest value in the first row of N since this row represents transitions from square 0. The square is found to be 99, considering the classic version. This can be intuitively explained: if the player ends up in square 99, a roll of exactly 1 is required to win. On average, $5/6$ of the times the player stays in square 99 and every time counts as an extra visit. Similarly, this approach yields the least visited sites. The results are summarised in Table 5.

Table 5: Most and least visited squares.

Most/least visited square (classic)	Most/least visited square (fast)	Most/least visited square (bounce back)
99 (1.55 per game)	44 (1.07 per game)	84 (1.23 per game)
2 (0.17 per game)	99 (0.16 per game)	2 (0.17 per game)

The fundamental matrix can also be used to evaluate the average steps before reaching the absorbing state, starting from state i. This is just the row sum of the matrix N, given by:

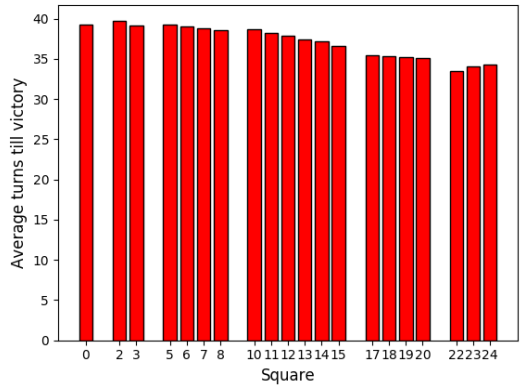
$$\tau = N\zeta. \tag{20}$$

where ζ is a column vector full of ones. The entry $(\tau)_i$ represents the average steps to reach square 100, starting from state i. Consequently, the exact average number of turns, for the three versions, can be directly found; the result is shown in Table 6.

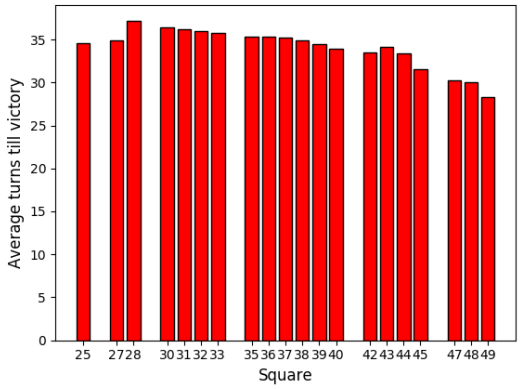
Table 6: Average turns to finish the game, using the properties of absorbing matrices.

Average turns (classic)	Average turns (fast)	Average turns (bounce back)
39.225	35.835	43.325

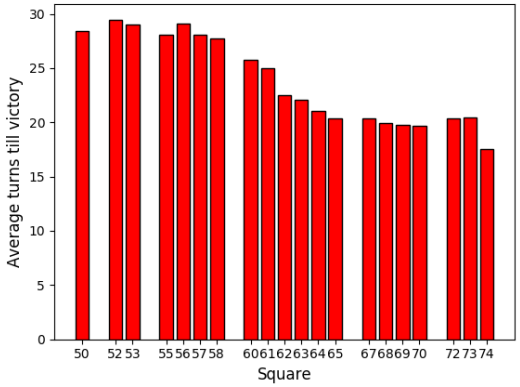
Additionally, the matrix τ can be used to order the squares accordingly to the average steps until victory. This is achieved through sorting the matrix by its indices using the numpy.argsort function; the result is shown in Figure 7.



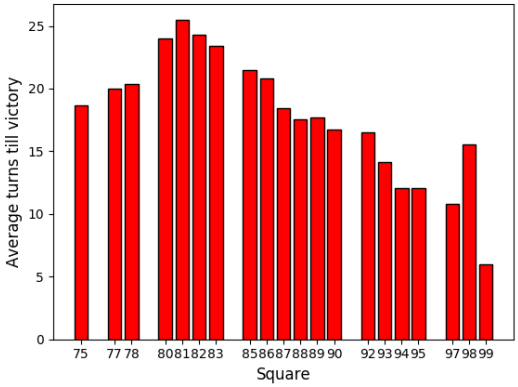
(a) Squares from 1 to 24



(b) Squares from 25 to 49



(c) Squares from 50 to 74



(d) Squares from 75 to 100

Figure 7: Average turns to reach square 100. The starting points of snakes or ladders are excluded since they are inaccessible.

The squares are ordered, considering the classic version, from the shortest to the longest average duration:

$$[94, 99, 97, 96, 74, 91, 92, 90, 75, 89, 88, 76, 70, 69, 77,$$
$$68, 72, 73, 78, 67, 66, 65, 79, 86, 63, 85, 84, 83, 82, 61, 60,$$
$$81, 59, 58, 55, 57, 50, 54, 53, 52, 48, 46, 45, 44, 42, 22, 40, 41,$$
$$23, 43, 39, 24, 25, 26, 38, 27, 37, 20, 19, 35, 18, 34, 17, 33,$$
$$32, 31, 30, 15, 29, 14, 13, 12, 11, 8, 10, 7, 6, 3, 0, 5, 2].$$

This result confirms the intuition; the further away from the initial squares, the shorter the game lasts. The starting points of snakes or ladders are not included because these squares are inaccessible: the player will be immediately transported to the other extremity. Alternatively, it is possible to define a new matrix M' where the rows and the columns, corresponding to transitions to and from these states, are deleted. Note that, if the matrix M is defined as SR (equation 14), these sites are accessible. In fact, in that case, if a player lands on these squares, an extra turn is needed to move to the other extremity of the snake or ladder.
The variance on the number of steps can also be determined using the fundamental matrix N. The formula is [2] [3]:

$$\sigma^2 = (2N - I)\tau - \tau_{sq}, \tag{21}$$

where τ_{sq} is the matrix τ with all the entries squared. The first entry of σ^2 is relevant; it represents the variance on the turns required to win, starting from square 0. The variances and the standard deviations are presented in Tables 7 and 8.

[2] Consult Appendix 2 for further details

Table 7: Variance using absorbing Markov matrices.

Variance (classic)	Variance (fast)	Variance (bounce back)
636.298	545.400	915.319

Table 8: Standard deviation, using absorbing Markov matrices.

Standard deviation (classic)	Standard deviation (fast)	Standard deviation (bounce back)
25.225	23.354	30.254

5 Optimisation of the game

The aim of this section is to determine the spinner range that minimises the average duration of the game, considering the classic 10×10 board shown in Figure 1. This is accomplished numerically by plotting the average duration for different spinner ranges. The result is shown in Figure 8.

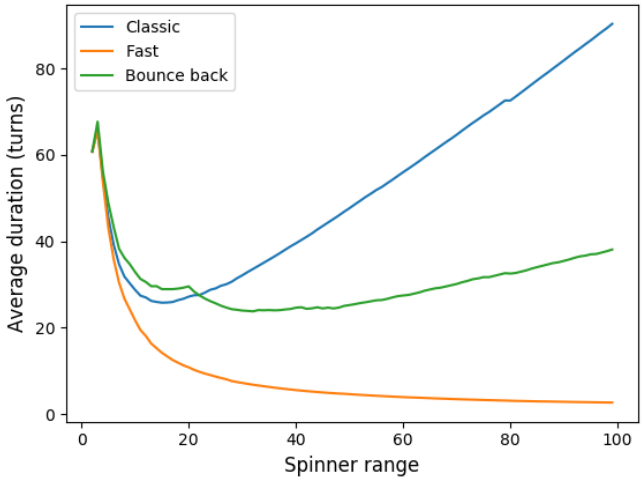


Figure 8: Duration of the game for different spinner ranges.

The average duration of the fast version decreases as the spinner range increases. This is expected; the player will get close to square 100 faster and overshooting is irrelevant. The classic and the bounce back versions have a non-trivial pattern. When the spinner range increases, for the classic version, it results in frequent turns in which a player does not move, since it needs to land exactly on square 100 to win. In the bounce back version, the player will move further back from square 100 more frequently. It is interesting to investigate which of these two versions is faster. It turns out that if the spinner range is less than 22, the bounce back version has a longer duration than the classic. Consequently, if the spinner range is greater than or equal to 22, the bounce back version is faster than the classic version. This analysis is done computationally, by finding the intersection of the curves describing the classic and the bounce back versions. The optimised version of the game is given by the spinner range corresponding to the location of the global minimum. The results are presented in Table 9.

Table 9: Spinner ranges associated with the shortest duration of the game.

Spinner range (classic)	Spinner range (fast)	Spinner range (bounce back)
15	∞	32

The maximum spinner range of the bounce back version is 100 in order to avoid multiple rebounds.

6 Analysis and Discussion

The first and second methods are approximate, nevertheless, they produce similar results. This is explained by the similarity of the probability density functions and the cumulative distribution functions. For the sake of comparison, it is useful to plot them in the same graph, as shown in Figures 9 and 10.

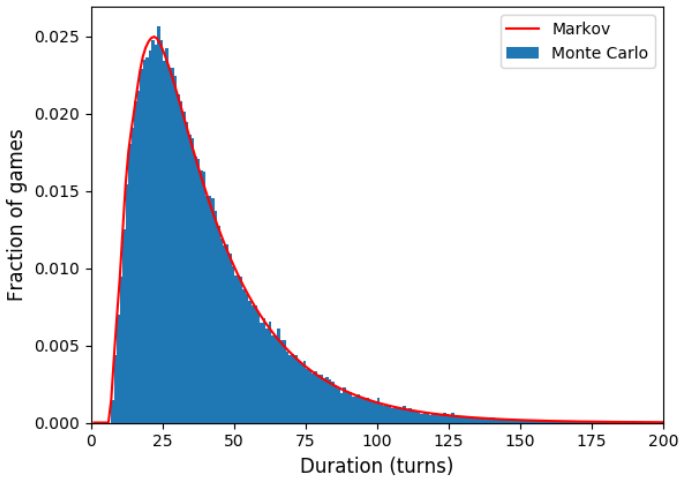


Figure 9: Comparison of the two PDFs obtained using Monte Carlo simulation and Markov matrices. The version is classic. The iterations are 10^5 and 10^3 for the Monte Carlo and the Markov matrix methods, respectively.

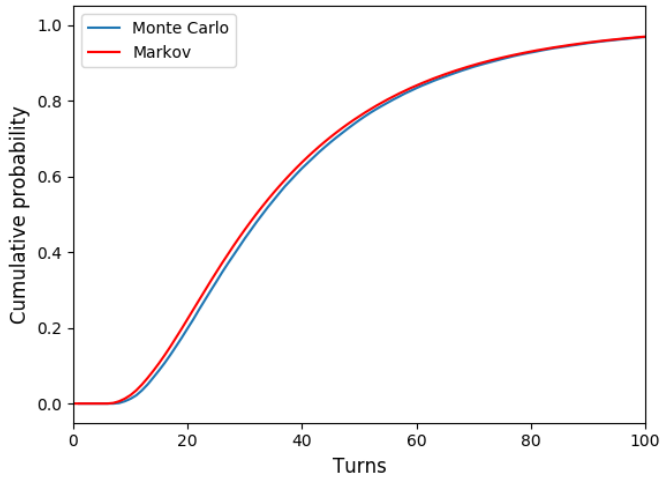


Figure 10: Comparison of the two CDFs obtained using Monte Carlo simulation and Markov matrices. The version is classic. The iterations are 10^5 and 10^3 for the Monte Carlo and the Markov matrix methods, respectively.

The most efficient method is the third, which uses the absorbing Markov matrices. In fact, it produces exact results, thus it can be used to evaluate the errors on the other methods. The relative error on the average duration, as a function of iterations, computed via Monte Carlo simulation, is shown in Figure 11.

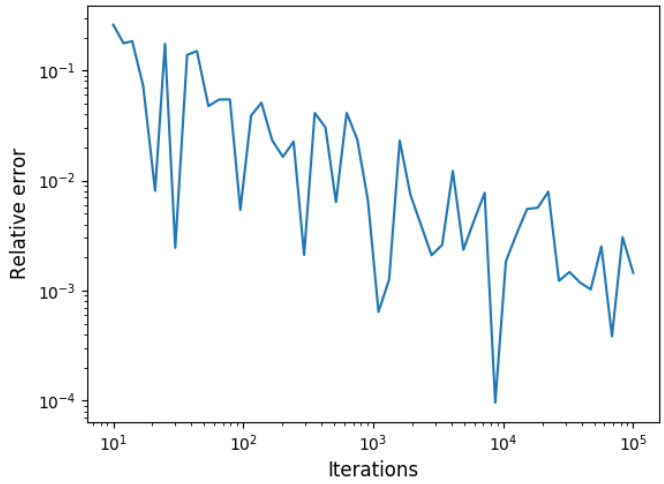


Figure 11: Fluctuations of the relative error on the mean, computed via Monte Carlo simulation, as a function of iterations. The version is classic.

This method is based on a random number generator, which explains the unstable oscillatory pattern. Nevertheless, the error, on average, gets smaller as the number of simulations increases. The same error analysis is performed on the other method[3]; the result is shown in Figure 12.

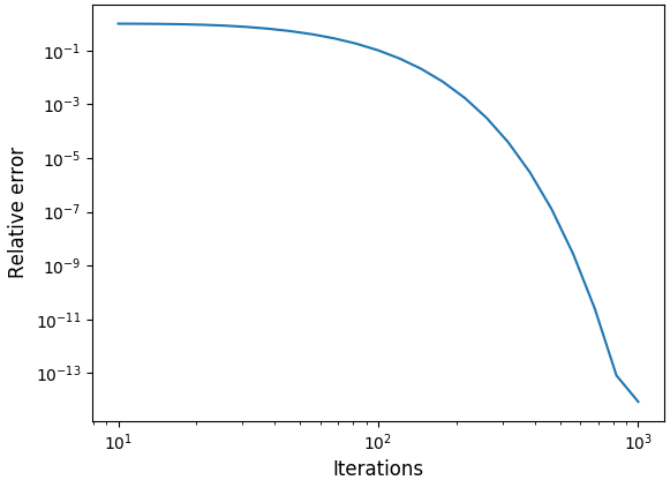


Figure 12: Relative error on the mean, calculated using Markov matrices, as a function of iterations. The version is classic.

In that case, the error decreases smoothly with the matrix power, as expected. This analysis shows that the Markov chain approach is better than the Monte Carlo simulation. In fact, the same accuracy is achieved using fewer iterations and the results are more stable. In general, the three methods seem to be logical, in fact, the average value of approximately 39.2, for the classic version, is consistent with the result presented by Althoen, King, and Schilling (1973) [1]. Furthermore, the optimised classic version of the game corresponding to a spinner range of 15 coincides with the work done by Cheteyan, Hengeveld, and Jones (2011) [4].

[3]Consult Appendix 3 for the error analysis on the variance, skewness, and kurtosis.

7 Conclusion

In this paper, the mathematics behind the popular board game Snakes and Ladders has been analysed. The first method is a Monte Carlo simulation using a random integer numbers generator provided by the function numpy.random.int. The second method relies on Markov matrix theory; the probabilistic evolution of the initial state vector is obtained by multiplying it by a transition matrix raised to the appropriate power. The second method is more accurate and less expensive than the first. However, the most efficient way to calculate the statistical quantities is provided by the third method, which relies on the absorbing Markov matrix formalism. This procedure produces exact results, therefore, it was used to examine the errors produced by the other methods.

References

[1] S. C. Althoen, L. King, and K. Schilling. How long is a game of snakes and ladders? *The Mathematical Gazette*, 77(478):71, 1993.

[2] John Lamperti. *Stochastic processes: a survey of the mathematical theory*. Springer-Verlag, 978-1-4684-9358-0, 1977.

[3] John G. Kemeny and James Laurie Snell. *Finite Markov chains: with a new appendix Generalization of a fundamental matrix"*. Springer-Verlag, 978-0-387-90192-3, 1983.

[4] Leslie A. Cheteyan, Stewart Hengeveld, and Michael A. Jones. Chutes and ladders for the impatient. *The College Mathematics Journal*, 42(1):2–8, 2011.

[5] Peter H. Westfall. Kurtosis as peakedness, 1905–2014.r.i.p. *The American Statistician*, 68(3):191–195, Mar 2014.

A Appendix 1: Statistical quantities

Given a set of N data values $\{x_n\}_{n=1}^{N}$, or a probability density function (for a continuous random variable), it is possible to calculate some statistical quantities such as the mode, the mean, the median and the momenta (variance, skewness, kurtosis).

Mode

The mode of a set of data values, Mo($\{x_n\}_{n=1}^{N}$), is defined as the value that appears most frequently. If x is a discrete random variable, then the mode is the value for which the probability mass function is maximised. If x is a continuous variable the mode is the value that maximises the probability density function.

Mean

The expectation value $E[x]$, or mean, of a discrete random variable x is defined as follows:

$$E[x] = \sum_n x_n p_n, \tag{22}$$

where p_n is the probability corresponding to x_n. If x is a continuous random variable, then:

$$E[x] = \int_R dx p(x) x, \tag{23}$$

where R is the domain in which x is defined and $p(x)$ is the probability density function.

Median

The median of an ordered discrete set is defined as the value that separates the lower half of the set from the upper half. The median $\tilde{x}$ of a continuous random variable, $x \in [a, b]$, is defined as:

$$\int_a^{\tilde{x}} dx p(x) = \int_{\tilde{x}}^{b} dx p(x) = \frac{1}{2}. \tag{24}$$

Standard deviation and variance

The standard deviation is a measure of how further apart are the values of a set of measures from the mean. It is defined as:

$$\sigma = \sqrt{E[x^2] - (E[x])^2}. \tag{25}$$

This definition covers both the discrete and the continuous case. The variance is defined as the square of the standard deviation.

Skewness

Skewness is a measure of the asymmetry of the probability distribution function. If a distribution is symmetric, such as a Gaussian, then the skewness is zero. The expression is given by:

$$s = E\Big[\Big(\frac{x-\mu}{\sigma}\Big)^3\Big], \tag{26}$$

where E is the expectation value operator, $\mu = E[x]$, and σ is the standard deviation. This formula can be simplified by applying the properties of the expectation value.

$$s = \frac{E[(x-\mu)^3]}{\sigma^3} = \frac{E[x^3 + 3x\mu^2 - 3x^2\mu - \mu^3]}{\sigma^3} = \frac{E[x^3] - 3\mu(E[x^2] - \mu^2) - \mu^3}{\sigma^3}, \tag{27}$$

hence:

$$s = \frac{E[x^3] - 3\mu\sigma^2 - \mu^3}{\sigma^3}. \tag{28}$$

Kurtosis

Kurtosis is a measure of the tendency of the process, described by a probability density function, to produce outliers, as highlighted by Westfall (2014) [5]. An outlier is a data point that differs significantly from other observations, thus it concerns the tail of the distribution. The formal definition is:

$$k = E\Big[\Big(\frac{x-\mu}{\sigma}\Big)^4\Big], \tag{29}$$

where E is the expectation value operator, $\mu = E[x]$ and σ is the standard deviation. This formula can be simplified by expanding the quartic binomial. The result is:

$$k = \frac{E[x^4] - 4\mu E[x^3] + 6\sigma^2\mu^2 + 3\mu^4}{\sigma^4}. \tag{30}$$

Note that some authors might use the excess kurtosis, obtained subtracting 3 (Gaussian kurtosis) from kurtosis.

B Appendix 2: Absorbing Markov matrices formalism

Variance on the number of turns to be absorbed

The fundamental matrix N, given by equation (17), can be used to find the variance on the number of steps required to reach the absorbing state. This proof follows the work done by Kemeny and Snell [3]. Let t_i be a function that counts the steps required to reach an absorbing state starting from the initial position i. If the chain is in an absorbing state then $t_i = 0$. The function t_i is generally given by:

$$t_i = \sum_j n_{ij}, \tag{31}$$

where n_{ij} represents the steps required to move from state i to j. The variance of this quantity is given by:

$$Var[t_i] = E[t_i^2] - E[t_i]^2. \tag{32}$$

The expectation of t_i is given by:

$$E[t_i] = \sum_j E[n_{ij}] \to N\zeta = \tau, \tag{33}$$

where N is the fundamental matrix and ζ a column vector full of ones. This result follows from the interpretation of the entries of the fundamental matrix N. In fact, $(N)_{ij}$ represents the number of times the process is in state j started from

state i. Therefore, the expectation is just a column vector with elements given by the row sum of the elements of N, which is just $\tau = N\zeta$. The expectation of t_j^2 is harder to find and requires extra calculations. The general formula is:

$$E[t_i^2] = \sum_j E[n_{ij}^2]. \tag{34}$$

This calculation involves only transitions from transient to absorbing states, therefore trivial transitions from absorbing to absorbing state are not considered. It is useful to split the calculation considering what happens after one step. Let T be the set of all possible transitions from a transient to a transient state in one step and $\tilde{T}$ the complementary set of T, then:

$$E[t_i^2] = \sum_{k \in T} p_{ik} + \sum_{k \in \tilde{T}} p_{ik} E[(t_k + 1)^2]. \tag{35}$$

The first part involves transitions in one step only. The second part corresponds to all possible remaining transitions which are done in $t_k + 1$ steps (one is added to take into account the first transition with probability p_{ik}). It follows:

$$E[t_i^2] = 1 + \sum_{k \in \tilde{T}} p_{ik}(E[t_k^2] + 2E[t_k]) \tag{36}$$

Switching to matrix representation, using $E[t_k^2] \to \tau_2$, $E[t_k] \to \tau$, $1 \to \zeta$, and $(Q)_{ik} = p_{ik}$:

$$\tau_2 = 1 + Q(\tau_2 + 2\tau), \tag{37}$$

hence:

$$\tau_2 = (1 - Q)^{-1}(2Q\tau + 1) = 2NQ\tau + N\zeta = 2(N - 1)\tau + \tau = (2N - 1)\tau. \tag{38}$$

The following result has been used:

$$NQ = Q + Q^2 + ... = N - 1. \tag{39}$$

Finally, the variance, in matrix representation, is given by:

$$\sigma^2 = 2(N - 1)\tau - \tau_{sq}, \tag{40}$$

where τ_{sq} is the matrix τ with all entries squared. This can be also written as:

$$\tau_{sq} = \tau^{\circ 2}, \tag{41}$$

where $\circ$ represents the Hadamard product notation. The Hadamard product between two matrices A and B is defined as:

$$(A \circ B)_{ij} = A_{ij} B_{ij}, \tag{42}$$

and the Hadamard power as:

$$(A^{\circ n})_{ij} = A_{ij}^n. \tag{43}$$

Skewness and kurtosis on the number of turns to be absorbed

A similar approach can be used to find the skewness. Using the definition presented in equation (28), it follows:

$$E[t_i^3] = 1 + \sum_{k \in \tilde{T}} p_{ik}(E[t_k^3] + 3E[t_k^2] + 3E[t_k^2]). \tag{44}$$

Switching to matrix representation:

$$\tau_3 = \zeta + Q(\tau_3 + 3\tau_{sq} + 3\tau), \tag{45}$$

where τ_3 is the matrix representation of $E[t_i^3]$. Using equations (38), (39), and (40):

$$\tau_3 = (1 - Q)^{-1}(\zeta + 3Q(\tau_{sq} + \tau)) = (6N(N - 1) + 1)\tau \tag{46}$$

Therefore, the skewness on the number of turns to be absorbed is:

$$s = [(6N(N - 1) + 1)\tau - 3\tau \circ \sigma^2 - \tau^{\circ 3}] \circ \sigma^{\circ(-3/2)}. \tag{47}$$

Using a similar approach, the kurtosis is:

$$k = \frac{\tau_4 - 4\tau_3\tau - 3\tau^2 + 6\tau_2\tau^2}{\sigma^{\circ 4}} \tag{48}$$

where,

$$\tau_4 = \tau + (N-1)(4\tau_3 + 6\tau_2 + 4\tau). \tag{49}$$

This result can be extended to any order. The important quantity to define is τ_n which represents $E[x^n]$. This is given, recursively, by:

$$\tau_n = \tau + (N-1)\sum_{i=1}^{n-1}\binom{n}{i}\tau_i. \tag{50}$$

Therefore the n^{th} order moment, in matrix representation, is given by:

$$\mu_n = E\Big[\Big(\frac{x-\mu}{\sigma}\Big)^n\Big] \rightarrow \frac{1}{\sigma^{\circ n}}\sum_{i=0}^{n}\binom{n}{i}\tau_i\tau^{\circ(n-i)}. \tag{51}$$

C Appendix 3: Errors on the variance, skewness, and kurtosis

The exact skewness and kurtosis can be found applying the properties of absorbing matrices. The results are reported in Tables 10 and 11.

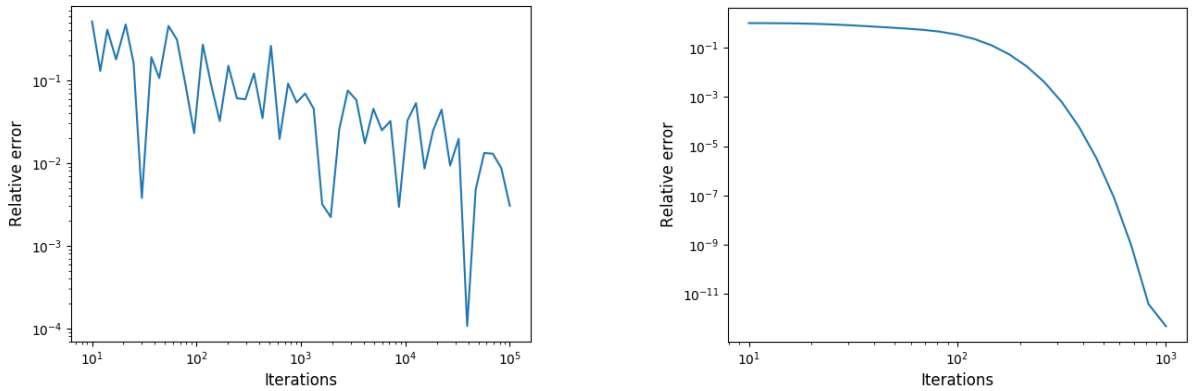
Table 10: Skewness on the number of turns, starting from square 0.

Skewness (classic)	Skewness (fast)	Skewness (bounce back)
1.822	1.876	1.910

Table 11: Kurtosis on the number of turns, starting from square 0.

Kurtosis (classic)	Kurtosis (fast)	Kurtosis (bounce back)
8.216	8.477	8.617

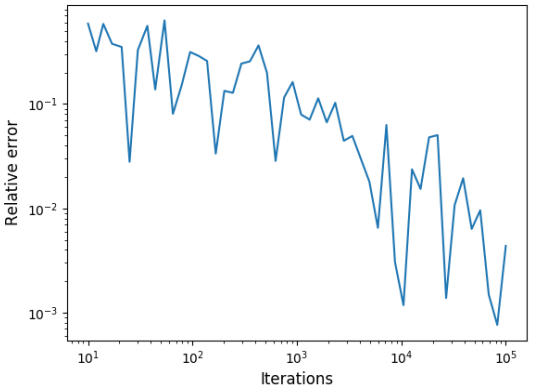
The errors on the variance, considering the classic version, using the Monte Carlo and the Markov matrix approaches are shown in Figure 13. The exact results for the variance are presented in the main text (Table 7).



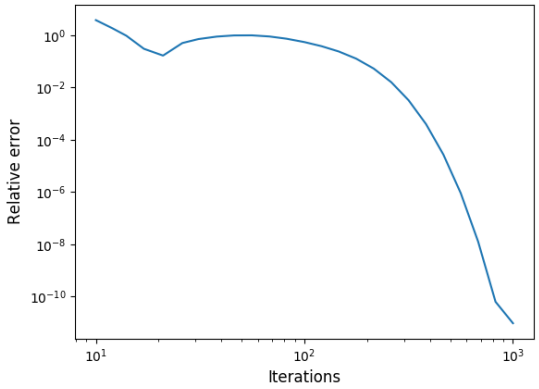
(a) Monte Carlo simulation (b) Markov matrix method

Figure 13: Relative error on the variance as a function of iterations. The version is classic.

The errors on the skewness and on the kurtosis, considering the classic version, are shown in Figures 14 and 15.

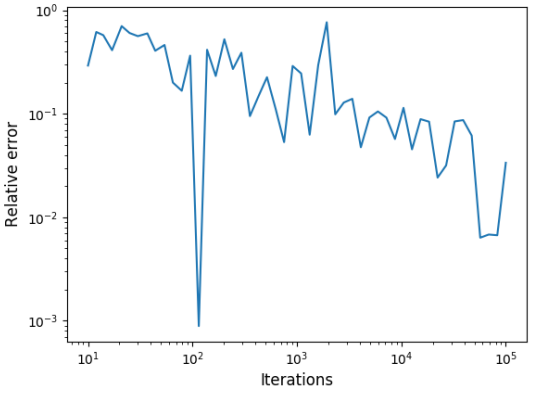


(a) Monte Carlo simulation

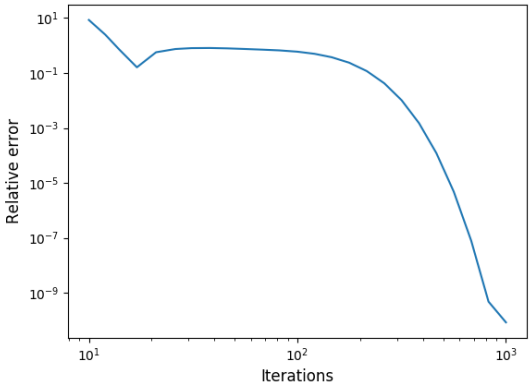


(b) Markov matrix method

Figure 14: Relative error on the skewness as a function of iterations. The version is classic.



(a) Monte Carlo simulation



(b) Markov matrix method

Figure 15: Relative error on the kurtosis as a function of iterations. The version is classic.

The above figures suggest that as the number of iterations increases the relative error decreases. Note that the higher the order of the momentum, the more iterations are needed to recover the same precision. Roughly, ten times more iterations are needed to compute a momentum of one order higher. This is expected since it requires the expectation value of the variable raised to a higher power.