Stochastic Processes and Simulations
Assignment 2
Random processes simulations

Theodor Jonsson (thjo0148@student.umu.se)
Simon Landin (sila0075@student.umu.se)
Oskar Ottander (osot0005@student.umu.se)

Stochastic Processes And Simulations
Teachers: Konrad Abramowicz, Mehdi Moradi & Jonas Gustafsson

Theodor Jonsson (thjo0148@student.umu.se)
Simon Landin (sila0075@student.umu.se)
Oskar Ottander (osot0005@student.umu.se)

December 7, 2022

Contents

1 Task 1: White Rat **2**
1.1 Task Description . . . 2
1.2 Theory . . . 3
1.2.1 Markov Chains . . . 3
1.3 Method . . . 4
1.3.1 Algorithm A . . . 5
1.3.2 Algorithm B . . . 5
1.3.3 Algorithm C . . . 6
1.4 Results . . . 7
1.4.1 Task 1A . . . 7
1.4.2 Task 1B . . . 7
1.4.3 Task 1C . . . 7
1.5 Conclusions . . . 7

2 Task 2: Back to origin **8**
2.1 Task Description . . . 8
2.2 Theory . . . 8
2.2.1 Central Limit Theorem . . . 9
2.3 Method . . . 9
2.3.1 Algorithm A . . . 9
2.3.2 Algorithm B . . . 10
2.3.3 Algorithm C . . . 10
2.4 Results . . . 11
2.4.1 Task 2A . . . 11
2.4.2 Task 2B . . . 11
2.4.3 Task 2C . . . 11
2.5 Conclusions . . . 12

3 Task 3: Decomposition **13**
3.1 Task Description . . . 13
3.2 Theory . . . 13
3.2.1 Homogeneous Poisson Processes . . . 13
3.2.2 Non-Homogeneous Poisson Processes . . . 14
3.3 Method . . . 16
3.3.1 Algorithm A . . . 17
3.3.2 Algorithm B . . . 18
3.3.3 Algorithm C . . . 18
3.4 Results . . . 19
3.5 Conclusions . . . 19

Theodor Jonsson (thjo0148@student.umu.se)
Simon Landin (sila0075@student.umu.se)
Oskar Ottander (osot0005@student.umu.se) December 7, 2022

Introduction

This is the lab report for Assignment 2 of Stochastic processes and Simulations during the autumn of 2022. The assignment consists of three different tasks regarding natural random processes. These tasks will be described in detail under each task section.

1 Task 1: White Rat

1.1 Task Description

A white rat is put into random cell of the maze (represented as a Markov chain in Figure 1) below. There are nine compartments with connections between the compartments as indicated. At each time step the rat moves through the compartments at random. That is, if there are k ways to leave a compartment, it chooses each of these with equal probability.

a) Consider long period of time (n=100 000 time steps). Use simulations to estimate the proportion of time spent by the rat in each cell.

b) Use Matlab to compute theoretical solution of the problem. Present necessary reasoning guaranteeing existence of the solution presented.

c) Assume now that a rat is placed in the position 5 initially (at time 0). Based on 1000 trials, what is the estimate of the expected time for the rat to return to its initial position? What is the standard error associated with your point estimate?

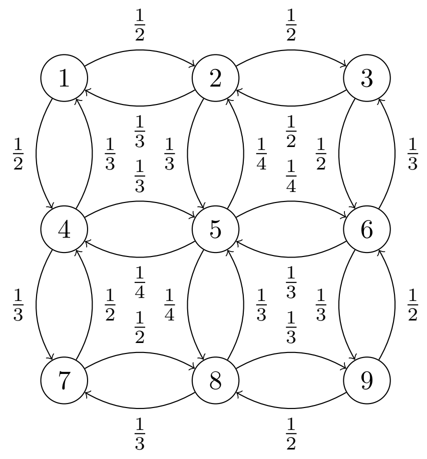

Figure 1: Visualization of the Markov chain constructed from the given conditions of equal probability of leaving to each adjacent cell. The values on each outgoing edge of the graph represents the probability that the rat chooses to travel via that edge.

1.2 Theory

The theory used in this exercise can be found in the lecture slides regarding Markov Chains.

1.2.1 Markov Chains

Definition 1.2.1: Markov Chains

A Markov Chain is a stochastic model that describes a sequence(chain) of possible events. The different events probability depend only on the previous state which can be expressed as

$$P(X_{n+1} = j|X_n = i, X_{n-1} = i_1, ...) = P(Xn + 1 = j|X_n = i) = p_{ij} \tag{1}$$

p_{ij} is the probability going from i to j are called transition probabilities.

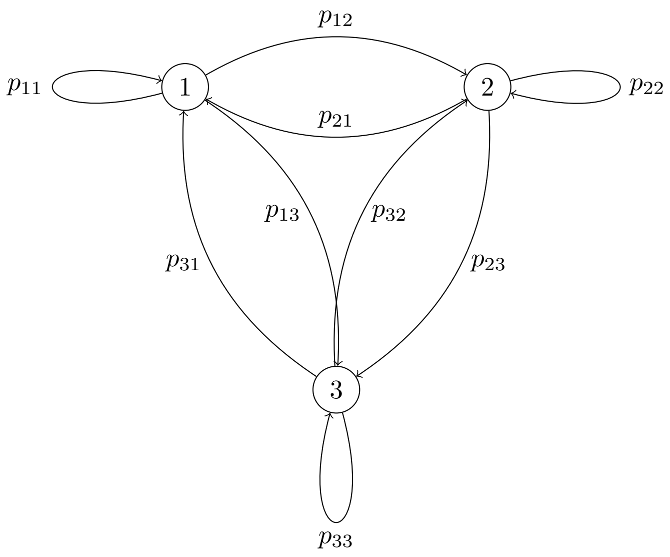

Figure 2: Figure 1 represents a simple Markov Chain and its corresponding transition probabilities.

The probabilities can also be arranged into the following matrix:

$$P = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1N} \\ a_{21} & a_{22} & \cdots & a_{2N} \\ \vdots & \vdots & & \vdots \\ a_{N1} & a_{N2} & \cdots & a_{NN} \end{bmatrix}$$

The row index corresponds to the state of chain at time n, and the column corresponds to the state in time n + 1. The total sum of each row has to equal to 1 since there has to be a transition.

$$\sum_{j=1}^{N} p_{ij} = 1 \tag{2}$$

The different types of Markov Chains are Irreducible Markov Chains:

Definition 1.2.2: Irreducible Markov Chain

The Markov Chain $X_n, n = 0, 1, 2, ...$ with space state $\{1, 2, ..., N\}$ is called irreducible if each state $j, j \in \{1, 2, ..., N\}$ can be reached in a finite time from any other state.

And Aperiodic Markov Chains:

Definition 1.2.3: Aperiodic Markov Chains

The Markov Chain $X_n, n = 0, 1, 2, ...$ with space state $\{1, 2, ..., N\}$ is aperiodic if for all states $j, j \in \{1, 2, ..., N\}$ there exists n such that

$$P(X_n = j|X_0 = j) > 0 \; and \; P(X_{n+1} = j|X_0 = j) > 0 \tag{3}$$

Theorem 1.2.1: Asymptotic distribution

:Let $X_n, n = 0, 1, 2, ...$ with space state $\{1, 2, ..., N\}$ be an irreducible and aperiodic Markov Chain. Then there exist an asymptotic distribution $\pi = (\pi_1, \pi_2, ..., \pi_N)$ Such that

$$\lim_{n\to\infty} P(X_n = j|X_0 = i) = \pi_j \; Independently \; of \; i, \tag{4}$$

or terms of transition matrix

$$\lim_{n\to\infty} P^n = \begin{bmatrix} \pi_1 & \pi_2 & \cdots & \pi_N \\ \pi_1 & \pi_2 & \cdots & \pi_N \\ \vdots & \vdots & & \vdots \\ \pi_1 & \pi_2 & \cdots & \pi_N \end{bmatrix} \tag{5}$$

1.3 Method

In our task the transition matrix is given by

$$P = \begin{bmatrix} 0 & 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0 \\ 1/3 & 0 & 1/3 & 0 & 1/3 & 0 & 0 & 0 & 0 \\ 0 & 1/2 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1/3 & 0 & 1/3 & 0 & 0 & 0 & 1/3 \\ 0 & 1/4 & 0 & 1/4 & 0 & 1/4 & 0 & 1/4 & 0 \\ 1/3 & 0 & 0 & 0 & 1/3 & 0 & 1/3 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 1/2 & 0 \\ 0 & 0 & 0 & 0 & 1/3 & 0 & 1/3 & 0 & 1/3 \\ 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 & 0 \end{bmatrix}$$

and a position is given by a 9-element row vector p with zeros everywhere except for a single index where there is a one.

We notice that our Markov chain is irreducible since we can reach all positions from everywhere.

We also see that it is periodic. Since we always need an even number of steps (as many steps up as down, and as many left as right) to get back to the place we started, we can never reach it in two consecutive time steps.

1.3.1 Algorithm A

For $N = 100\ 000$ we repeatedly update the position of an initially randomly placed rat. Each time the rat is put in a cell we update a counter for this cell, so that we after having run all the updates can compute the fraction of the time spent in each cell. This procedure is shown in Algorithm 1.

Algorithm 1: Simulate proportion of time spent in each cell.

```
1 $counter \leftarrow (0,0,0,0,0,0,0,0,0)$
2 $p \leftarrow rand(1:9)$
3 for $i \in 1:N$ do
4     $pos \leftarrow (0,0,0,0,0,0,0,0,0)$
5     $pos(p) \leftarrow 1$
6     $counter(p) \leftarrow counter(p) + 1$
7     $pos \leftarrow pos * P$
8     $p \leftarrow InverseTransform(pos)$
9 $fraction \leftarrow \frac{1}{N} * counter$
```

After running this, the $fraction$ will contain a vector of the proportion of time spent in each cell.

The *InverseTransform* function is a function which returns an index to the input vector with proportion equal to the probabilities in it. This is done using the inverse transform algorithm for discrete distributions. See Algorithm 2.

Algorithm 2: Inverse Transform given discrete probabilities.

```
  Input: $density$ A vector of probabilities for each position.
  Output: $i$ An index selected with probability proportional as to what is
          given in the $density$.
1 Function $InverseTransform(density)$
2     $U \sim \mathcal{U}(0,1)$                    % Uniformly distributed random number
3     $p \leftarrow 0$
4     $i \leftarrow 0$
5     while $p < U$ do
6         $i \leftarrow i + 1$
7         $p \leftarrow p + density[i]$
8     return $i$
```

1.3.2 Algorithm B

We set up the equation system and let `Matlab` solve it for us.

The equations will be (as a linear equation system):

$$\begin{bmatrix}\pi_1\\\pi_2\\\pi_3\\\pi_4\\\pi_5\\\pi_6\\\pi_7\\\pi_8\\\pi_9\end{bmatrix}^T \begin{bmatrix} 0 & 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 & 0\\ 1/3 & 0 & 1/3 & 0 & 1/3 & 0 & 0 & 0 & 0\\ 0 & 1/2 & 0 & 1/2 & 0 & 0 & 0 & 0 & 0\\ 0 & 0 & 1/3 & 0 & 1/3 & 0 & 0 & 0 & 1/3\\ 0 & 1/4 & 0 & 1/4 & 0 & 1/4 & 0 & 1/4 & 0\\ 1/3 & 0 & 0 & 0 & 1/3 & 0 & 1/3 & 0 & 0\\ 0 & 0 & 0 & 0 & 0 & 1/2 & 0 & 1/2 & 0\\ 0 & 0 & 0 & 0 & 1/3 & 0 & 1/3 & 0 & 1/3\\ 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 & 0 \end{bmatrix} = \begin{bmatrix}\pi_1\\\pi_2\\\pi_3\\\pi_4\\\pi_5\\\pi_6\\\pi_7\\\pi_8\\pi_9\end{bmatrix}^T$$

Which implies $\pi P - \pi = \pi(P-1) = 0$. We can also replace one column because of the requirement that the probabilities should add to 0; $\sum_{i=1}^{9} \pi_i = 1$.

$$\begin{bmatrix}\pi_1\\\pi_2\\\pi_3\\\pi_4\\\pi_5\\\pi_6\\\pi_7\\\pi_8\\\pi_9\end{bmatrix}^T \begin{bmatrix} -1 & 1/2 & 0 & 0 & 0 & 1/2 & 0 & 0 & 1\\ 1/3 & -1 & 1/3 & 0 & 1/3 & 0 & 0 & 0 & 1\\ 0 & 1/2 & -1 & 1/2 & 0 & 0 & 0 & 0 & 1\\ 0 & 0 & 1/3 & -1 & 1/3 & 0 & 0 & 0 & 1\\ 0 & 1/4 & 0 & 1/4 & -1 & 1/4 & 0 & 1/4 & 1\\ 1/3 & 0 & 0 & 0 & 1/3 & -1 & 1/3 & 0 & 1\\ 0 & 0 & 0 & 0 & 0 & 1/2 & -1 & 1/2 & 1\\ 0 & 0 & 0 & 0 & 1/3 & 0 & 2/3 & -1 & 1\\ 0 & 0 & 0 & 1/2 & 0 & 0 & 0 & 1/2 & 1 \end{bmatrix} = \begin{bmatrix}\pi_1\\\pi_2\\\pi_3\\\pi_4\\\pi_5\\\pi_6\\\pi_7\\\pi_8\\1\end{bmatrix}^T$$

Where the last column comes from the fact that the values in the stationary distribution has to sum to one, replacing one of the previous columns, and the rest to 0 because of the subtraction of I.

We solve this linear equation system using the Matlab `inv` command.

1.3.3 Algorithm C

For $N = 1000$ trials we place the rat in the central position, and count how many steps it takes for it to return. We then compute an estimate and standard error for this number. See Algorithm 3.

Algorithm 3: Estimate time until returning to center.

```
results ← list()                      % New empty list to store the simulation results.
for $i \in 1:N$ do
    $p \leftarrow 5$
    $first \leftarrow true$
    $count \leftarrow 0$
    while $first$ or $p \neq 5$ do
        $pos \leftarrow (0,0,0,0,0,0,0,0,0)$
        $pos(p) \leftarrow 1$
        $counter \leftarrow counter + 1$
        $pos \leftarrow pos * P$
        $p \leftarrow InverseTransform(pos)$
$\hat{\mu} \leftarrow \frac{1}{N}\sum_{i=1}^{N} results(i)$                      % Compute estimated mean.
$\hat{\sigma}^2 \leftarrow \frac{1}{N-1}\sum_{i=1}^{N}(results(i) - \hat{\mu})^2$    % Compute estimated variance.
$\hat{se} \leftarrow \frac{\hat{\sigma}}{\sqrt{N}}$                                  % Standard error.
```

1.4 Results

The following sections contain the results for each subtask.

1.4.1 Task 1A

We get the following result for the estimated fraction of time spent at each position.

$$\hat{\pi} = (0.0833, 0.1251, 0.0834, 0.1250, 0.1667, 0.1250, 0.0833, 0.1250, 0.0833)$$

1.4.2 Task 1B

We get the following result for the computed theoretical stationary distribution.

$$\pi = \begin{bmatrix} 0.0833 & 0.1250 & 0.0833 & 0.1250 & 0.1667 & 0.1250 & 0.0833 & 0.1250 & 0.0833 \end{bmatrix} = \begin{bmatrix} 1/12 & 1/8 & 1/12 & 1/8 & 1/6 & 1/8 & 1/12 & 1/8 \end{bmatrix}$$

1.4.3 Task 1C

The expected time taken until the rat gets back to the initial position during 1000 trials is 6.2600.

With standard error 0.1633.

1.5 Conclusions

The answers to the fraction of time spent at each position (the stationary distribution) matches between the analytical and estimated values. It also intuitively makes sense that the most time would be spent in the center (1/6), while the least likely position is the corners (1/12 each).

2 Task 2: Back to origin

2.1 Task Description

Consider a Brownian motion in two dimensions. This motion is named after the botanist Robert Brown, who first described the phenomenon, while looking through a microscope at pollen particle immersed in water. Mathematically, the process observed by Robert Brown can be described with help of two independent Brownian motions $\{W_1(t), t \geq 0\}$ and $\{W_2(t), t \geq 0\}$ with common parameter σ. The position of the pollen (particle) in Euclidean space as time progresses can be then described

$$(x(t), y(t)) = (W_1(t), W_2(t)), t \geq 0$$

so movement in each coordinate is simply described by one dimensional Brownian motion. Observe that we naturally have that the motion process starts in the origin (0,0). Now, assume that we put an absorbing bounding box determined by $x = -1, x = 1, y = -1$ and $y = -1$ (square with edges of length 2). When the process touches the boundaries it stops.

a) Assume that for both processes we have $\sigma = 1$. Describe the algorithm to simulate the above mentioned process. Plot an example of realisation of the process together with boundary. Use time discretisation step not bigger than $dt = 0.0001$ (you can decrease it further, depending on your computer capacity).

b) Using the result of a) construct a Monte Carlo algorithm to estimate the expected life time of such process (i.e., time before it hits the boundary). Provide a 95% confidence interval for the estimate. Choose dt and number of replicates N suitably to your computer capacity, but not bigger than $dt = 0.001$ and $N = 1000$, respectively.

c) Now consider increasing σ, i.e., increasing the volatility of the movement of the particle. Estimate the mean life time of such process for $\sigma = 1, 2, ..., 10$. Again, choose dt and number of replicates N suitably to your computer capacity. Investigate the relationship between the two quantities graphically. You can plot the relationships on original and log-log scale.

2.2 Theory

We are dealing with a particle moving in the x and y directions as by independent Brownian motions (W_1 and W_2). Brownian motion is defined as in Definition 2.2.1.

Definition 2.2.1: Brownian Motion

A stochastic process $\{X(t), t \geq 0\}$ is said to be a *Brownian motion* if

(i) $X(0) = 0$

(ii) It has independent stationary increments. (I.e. each disjoint increment of time gives an independent random variable with distribution depending only on the length of the increment).

(iii) $X(t) \sim \mathcal{N}(0, \sigma^2 t)\ \forall t$

The Brownian motion is said to be *standard* if $\sigma = 1$.

In order to construct an approximate Brownian motion, using increments of size Δt we use the formula (with $W(0) = 0$):

$$W(k\Delta t) = \sum_{i=1}^{k} X_i = W((k-1)\Delta t) + X_k$$

where each $X_i \sim \mathcal{N}(0, \sigma^2 \Delta t) = \sigma\sqrt{\Delta t}{\cdot}N(0,1)$

The information here is taken from the course lecture notes and slides.

2.2.1 Central Limit Theorem

In order to get a confidence interval when using Monte Carlo methods we need to know the distribution of the sum of a large number of independent identically distributed random variables. The central limit theorem states that this sum will be normally distributed.

Theorem 2.2.1: Central Limit Theorem

Let $X_1, X_2, \ldots, X_N$ be independent identically distributed random variables with $E(X_i) = \mu)$ and $Var(X_i) = \sigma^2 < \infty$, $\forall i$. Then

$$\frac{\frac{1}{N}\sum_{i=1}^{N} X_i - \mu}{\sigma/\sqrt{N}} \xrightarrow{D} \mathcal{N}(0,1) \;\; as \;\; N \to \infty$$

We let Φ denote the cumulative density function for the standard normal distribution, and Φ^{-1} its inverse.

We also make use of rules regarding how sums and scaling of normal distributions work to construct our confidence intervals.

The information here is taken from course slides.

2.3 Method

We use Algorithm 4 to update a given current position of a particle according to Brownian motion.

Algorithm 4: Brownian Motion update with given σ and Δt.

```
Input: $p = (x, y)$ the current particle position
Output: the next particle position
1 Function Update(p)
2     $X_x \leftarrow \mathcal{N}(0,1)$
3     $X_y \leftarrow \mathcal{N}(0,1)$
4     return $p + \sigma\Delta t(X_x, X_y)$
```

2.3.1 Algorithm A

In this task we let $\sigma = 1$. We also let $dt = 0.00001$.

We want to visualize a trajectory through the box with limits at ± 1 in x and y directions.

Starting at $p = (0, 0)$, we continue simulating the next step until we notice that we have reached the edge of the box, storing each position.

See Algorithm 5.

Algorithm 5: Visualization of Brownian motion trajectory.

```
$p \leftarrow (0,0)$
$history \leftarrow list()$                % An empty list.
while $p \in [0,1]^2$ do
    history.append($p$)                % Store the position.
    $p \leftarrow Update(p)$                % See Algorithm 4.
plot(history)                % Plot the trajectory
```

2.3.2 Algorithm B

We want to use a Monte Carlo algorithm to estimate the expected time until the particle reaches the boundary, and provide confidence intervals for this estimate.

In this task we again let $\sigma = 1$. We also let $dt = 0.00001$. We choose to run the Monte Carlo simulation for $N = 1000$ replications.

We run N iterations of the loop in Section 2.3.1, storing the number of steps taken for each.

By the Central Limit Theorem (2.2.1) we get a way to compute the approximate distribution of the results for large N, and we use this to compute the confidence interval using the data we got. (Note that 95% interval corresponds to values between 0.025 and 0.975 percentiles).

See Algorithm 6.

Algorithm 6: Creation of confidence intervals over number of steps until exiting.

```
$results \leftarrow list()$                % An empty list.
for $i \in 1 : N$ do
    $count \leftarrow 0$
    $p \leftarrow (0,0)$
    while $p \in [0,1]^2$ do
        $count \leftarrow count + 1$
        $p \leftarrow Update(p)$                % See Algorithm 4.
    results.append($count$)                % Store the count for the replicate.
$\hat{\mu} \leftarrow \frac{1}{N}\sum_{i=1}^{N} results[i]$
$\hat{\sigma}^2 \leftarrow \frac{1}{N-1}\sum_{i=1}^{N}(results[i] - \hat{\mu})^2$
$\hat{se} \leftarrow \frac{\sqrt{\hat{\sigma}^2}}{\sqrt{N}}$
$interval \leftarrow (\hat{\mu} + \Phi^{-1}(0.025) \cdot \hat{se},\ \hat{\mu} + \Phi^{-1}(0.975) \cdot \hat{se})$
```

2.3.3 Algorithm C

We repeat the procedure from Section 2.3.2 for each different value in $\sigma = 1, 2, \ldots, 10$, we then plot the results in log-log fashion.

See Algorithm 7.

Algorithm 7: Investigate effect of σ on time until exit.

```
1 $p \leftarrow (0, 0)$
2 $results \leftarrow list()$                          % An empty list.
3 $sigmas \leftarrow 1 : 10$ for $\sigma \in sigmas$ do
4     Run Algorithm 6
5     results.append($\hat{\mu}$)                       % Store the mean for each $\sigma$.
6 $plot(log_{10}(sigmas),\ log_{10}(results))$
```

2.4 Results

The following sections contain the results for each subtask.

2.4.1 Task 2A

In Figure 3 we see the Brownian motion process.

The red dot indicates the starting point, the red lines represents the bounding box, and the blue trail is the Brownian motion process steps.

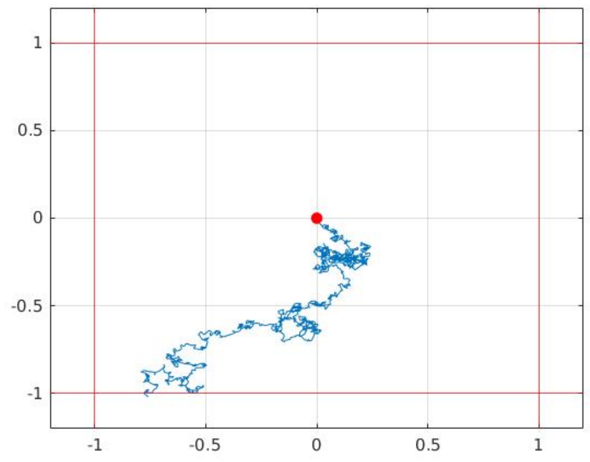

Figure 3: Visualization of Brownian motion.

2.4.2 Task 2B

We get the following 95% confidence interval for the time until exit: (0.5920, 0.6088).

The mean is about 0.6.

2.4.3 Task 2C

In Figure 4 we see the mean time until box being exited plotted against the size of σ. We use a log-log plot, and see that the graph forms a straight line, indicating a relationship between them with negative exponent (around -2).

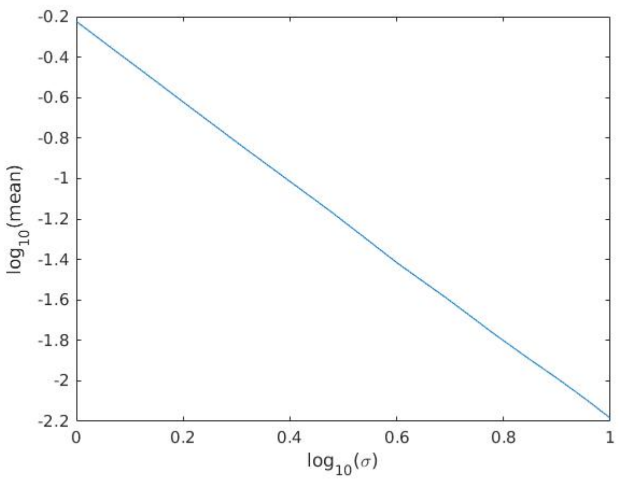

Figure 4: Log-log plot of mean times until exit against σ.

2.5 Conclusions

We see that the number of steps taken until increases in proportion to the decrease in Δt size. This results in the time taken (the product of the two) is constant, with using smaller Δt only leading to more accurate estimates of the mean time until exit (which is around 0.6 for $\sigma = 1$).

The time until exit T seems to be (by inspection of the log-log plot) related to the σ like a function similar to $T \approx a\sigma^k$ where a is about $0.6 \approx 10^{-0.22}$ and $k \approx -2$.

3 Task 3: Decomposition

3.1 Task Description

Consider a non-homogeneous Poisson process $\{N(t), t \geq 0\}$ with intensity function

$$\lambda(t) = 2 + 3\sqrt{t} + 1, t > 0 \tag{6}$$

We want to simulate the process on time interval for $t \in [0, 10]$.

a) Decompose the process into a homogeneous and non-homogeneous Poisson process. Construct two algorithms to simulate the homogeneous and non-homogeneous component of the process, respectively.

b) Use the results from a) to simulate trajectories of $N(t), t \in [0, 10]$. Plot 50 realisations of the process on a common plot together with the corresponding mean value function.

c) Use 10000 simulated realisations of the process to estimate E(N(10)) and V(N(10)). Do your estimates coincide with theoretical values (reason what those are)?

3.2 Theory

The theory used in this exercise can be found in the lecture slides regarding Homogeneous (HPP) and Non Homogeneous Poisson (NHPP).

3.2.1 Homogeneous Poisson Processes

We begin with the homogenous case.

Definition 3.2.1: Homogeneous Poisson Process

If $N(t)$ denotes the number of events occurring on the interval of $[0, t)$ where each event take place at random. Then $N(t)$ is a homogeneous Poisson process with constant intensity $\lambda > 0$ if the following criteria are satisfied.

1. There are no starting number of events, i.e. $N(0) = 0$
2. The number of events on two disconnected intervals are independent.
3. The distribution of $N(t)$ only depends on the length of the interval meaning that when the interval occurs does not alter the distribution.
4. On short intervals, the probability of events is proportional to the interval length.
5. No two events occur on exactly the same time.

From definition 3.2.1 the following useful properties can be derived:

Properties 3.2.1: Homogeneous Poisson Process

The number occurring events of a homogeneous Poisson process $N(t)$ with intensity $\lambda > 0$ on an interval of size t is Poisson distributed with intensity λt:

$$N(t+s) - N(s) \sim Po(\lambda t). \tag{7}$$

Since $N(0) = 0$, from definition 3.2.1, we have that:

$$N(t+0) - N(0) = N(t) \sim Po(\lambda t). \tag{8}$$

The distribution of $N(t)$ only depends on the length of the interval meaning that **when** the interval occurs, it does not alter the distribution. This is a crucial property of HPP since it relates the process into something well studied. The following properties of Poisson distributions are used in this task.

Properties 3.2.2: Poisson Distribution

Let X be Poisson distributed with intensity λ then we known:

$$E[X] = \lambda, \tag{9}$$

but also that the variance can be expressed as

$$V[X] = \lambda. \tag{10}$$

We can also calculate the probability that exactly k events occur using:

$$P\left(X = k\right) = \frac{e^{-\lambda}\lambda^{-k}}{k!}. \tag{11}$$

We can use equation 7 together with equation 11 to derive the following property of the Poisson process.

Property 3.2.1: Exponentially distributed time intervals

The elapsed time, T, between two events of a homogeneous Poisson process $\{N(t), t \geq 0\}$ with intensity λ, is exponentially distributed with rate λ:

$$T \sim exp(\lambda) \tag{12}$$

3.2.2 Non-Homogeneous Poisson Processes

A more generalized version of HPP is the Nonhomogeneous Poisson processes which is not constrained to constant intensities with respect to time. This new process does not satisfy all criteria of definition 3.2.1 or more specifically point 3 and 4 of the definition since the intensity is highly dependent on time. The nonhomogeneous process also obey a similar property as property 3.2.1 but modified to conform to the varying intensity over time.

Properties 3.2.3: Non Homogeneous increments

Let $N(t)$ describe the number of events occured until time point t of a non homogeneous Poisson process with intensity $\lambda(t)$. Then the expected value of $N(t)$ can be expressed by the **mean-value function** $\Lambda(t)$:

$$E[N(t)] = \Lambda(t) = \int_0^t \lambda(\alpha) d\alpha \tag{13}$$

The modified version of property 3.2.1 for a non homogeneous Poisson process can be expressed as:

$$N(t+s) - N(s) \sim Po\left(\Lambda(t+s) - \Lambda(s)\right) = Po(\int_s^{t+s} \lambda(\alpha) d\alpha). \tag{14}$$

Using the property of Poisson processes that $N(0) = 0$ then

$$N(t+0) - N(0) = N(t) \sim Po\left(\Lambda(t) - \Lambda(0)\right) = Po(\int_0^t \lambda(\alpha) d\alpha). \tag{15}$$

The non homogeneous Poisson process also conforms to super positional properties described as:

Property 3.2.2: Non Homogeneous Super Position

If $N_1(t)$ and $N_2(t)$ are two non homogeneous Poisson processes with intensities $\lambda_1(t)$ and $\lambda_2(t)$ respectively then

$$N(t) = N_1(t) + N_2(t)$$

is also a non homogeneous Poisson process but with intensity:

$$\lambda(t) = \lambda_1(t) + \lambda_2(t).$$

3.3 Method

We construct two algorithms to simulate a homogeneous and non homogeneous Poisson processes, see Algorithm 8 and Algorithm 9. With these algorithms we can easily study various characteristics of Poisson processes.

The nonhomogeneous Poisson process with intensity $\lambda(t) = 2 + \frac{3}{\sqrt{t+1}}$ is decomposed into its homogeneous and nonhomogeneous parts: $\lambda(t) = \lambda_H + \lambda_N(t)$, where $\lambda_H = 2$, and $\lambda_N(t) = \frac{3}{\sqrt{t+1}}$.

The notation used throughout these exercises are as follows:

- $\lambda_H = 2$ - Constant intensity w.r.t. time, used for homogeneous Poisson processes.
- $\lambda_N(t) = \frac{3}{\sqrt{t+1}}$ - Intensity which depend on time, used for non homogeneous Poisson processes.
- $N_H(t)$ - Number of events in a homogeneous Poisson process with intensity λ_H till time t
- $S_H(t)$ - Homogeneous Poisson process with intensity λ_H
- $N_H(t)$ - Number of events in a Non homogeneous Poisson process with intensity $\lambda_N(t)$ till time t
- S_N - Non homogeneous Poisson process with intensity $\lambda_N(t)$
- $T = 10$ - Time to simulate the process.
- $\Lambda(t) = \int_0^t \lambda_N(\alpha) d\alpha = 6\sqrt{t+1}$ - Mean value function of the given $\lambda_N(t)$, see properties 3.2.3.
- $\lambda(t) = \lambda_H + \lambda_N(t)$ - Intensity for the combined processes.
- S - The combined Poisson process with intensity $\lambda(t)$
- $N(t)$ - Number of events in the combined Poisson process with intensity $\lambda(t)$ till time t

3.3.1 Algorithm A

Algorithm 8:
Simulate trajectory of homogeneous Poisson process with intensity λ

```
Input:
    $\lambda > 0$ the constant intensity of the process.
    $T$ the amount of time to simulate.
Output:
    $S$ homogeneous Poisson process.
Function homogeneous ($\lambda$,$T$)
    $t \leftarrow 0$                                    % Time always starts at 0.
    $S_H \leftarrow list()$                             % An empty list.
    while $t < T$ do
        $U \sim \mathcal{U}(0,1)$                       % Random uniform number representing probability
        $X \leftarrow -\frac{1}{\lambda}\log(U)$        % ITM of exponential distribution.
        $t \leftarrow t + X$                            % Take a time step.
        if $t < T$ then
            $S_H.append(t)$                             % An event occurred at time $t$.
    return $S_H$
```

The algorithm above demonstrates how a homogeneous Poisson process can be simulated. This is done since we know that the intermediate time of two events are exponentially distributed with rate λ, see property 3.2.1, the same as the sought after Poisson process'.

The algorithm for the non homogeneous counterpart is similar but differs on some key points.

Algorithm 9:
Simulate trajectory of non homogeneous Poisson process with intensity $\lambda(t)$.

```
Input:
    $\lambda(t) > 0$ the varying intensity of the process as a function of time.
    $T$ the amount of time to simulate.
    $\lambda_{max}$ maximum value of $\lambda(t)$ on $t \in [0, T]$
Output:
    $S$ non homogeneous Poisson process.
Function non_homogeneous($\lambda(t)$,$T$,$\lambda_{max}$)
    $t \leftarrow 0$                                    % Time always starts at 0.
    $S_N \leftarrow list()$                             % An empty list.
    while $t < T$ do
        $U \sim \mathcal{U}(0,1)$                       % Random uniform number representing probability
        $X \leftarrow -\frac{1}{\lambda}\log(U)$        % ITM of exponential distribution.
        $t \leftarrow t + X$                            % Take a time step.
        $U \sim \mathcal{U}(0,1)$ if $t < T$ and $U < \frac{\lambda(t)}{\lambda_{max}}$ then
            $S_N.append(t)$                             % An event occurred at time $t$.
    return $S_N$
```

The algorithm requires the maximum value of $\lambda(t)$ to be calculated before hand in order to use the acceptance and rejection style method of simulating the non homogeneous process.

3.3.2 Algorithm B

We want to, with $N_{reps} = 50$ replicates, visualize the process over time $[0, T]$, where $T = 10$. Plot the trajectory (the cumulative value of the number of events) for each of the N_{reps} simulated runs from 0 to T.

Algorithm 10:
Simulate trajectory of combined Poisson process S

```
Input:
    $T$ amount of time units to simulate for.
Output:
    $S$ the combined process of $S_H(T)$ and $S_N(T)$
Function simulate(T)
    $\lambda_H \leftarrow 2$                                   % The constant intensity of $N(t)$
    $\lambda_N(t) \leftarrow \frac{3}{\sqrt{t+1}}$             % The varying intensity as function of time.
    $\lambda_{max} \leftarrow 3$                               % The maximum intensity of $\lambda_N(t)$
    $S_H \leftarrow homogeneous(\lambda_H, T)$
    $S_N \leftarrow non\_homogeneous(\lambda_N(t), \lambda_{max}, T)$
    $S \leftarrow Union(S_H, S_N)$                             % Join the two lists.
    $S \leftarrow sort(S)$                                     % Sort the list w.r.t. time.
    return $S$
```

The algorithm above simulate two distributions, one homogeneous S_H using λ_H and one non homogeneous with intensity $\lambda_N(t)$. We the use the property of super position, see property 3.2.2, to construct a combined non homogeneous Poisson process S. S is then sorted for time continuity. In order to be able to see whether the values are reasonable we also plot the theoretical mean values.

Algorithm 11: Visualize Poisson process trajectories.

```
$results_S \leftarrow list()$                    % The simulated results
$results_T \leftarrow list()$                    % Expected value at a fixed number of time points.
for $i \in 1 : N_{reps}$ do
    $S \leftarrow simulate(T)$                   % Simulate combined process
    $results_S.append(S)$
for $t \in (0, T)$ do
    $E = \Lambda(t) - \Lambda(0)$                % Calculate the theoretically expected value
    $results_T.append(E)$
$plot\_stairs(results_S)$                        % Plot the simulation as stair plot
$plot(results_T)$                                % Plot the theoretically expected value
```

3.3.3 Algorithm C

We want to, with $N_{reps} = 10000$ replicates, estimate mean and variance of the results of the process for $T = 10$.

To do this we loop N times and store the number of events from simulation calls for the process. Then we compute the mean and variance of these results.

Algorithm 12: Estimate mean and variance of Poisson process.

```
1 $N_{reps} = 10000$
2 $results \leftarrow list()$
3 for $i \in 1 : N_{reps}$ do
4     $S \leftarrow simulate(T)$                % Simulate combined process
5     $N \leftarrow length(S)$                  % The length of $S$ is the number of events
6     $results.append(N)$
7 $\hat{\sigma}^2 \leftarrow var(results)$
8 $\hat{\mu} \leftarrow mean(results)$
```

3.4 Results

Using the decomposed Poisson process simulation algorithm (see Algorithm 10) and running it as in Algorithm 11 we get the data shown in the following plots.

In Figure 5 we see the 50 Poisson process simulations stair plot trajectories.

We see that the trajectories are following the theoretically expected mean as expected.

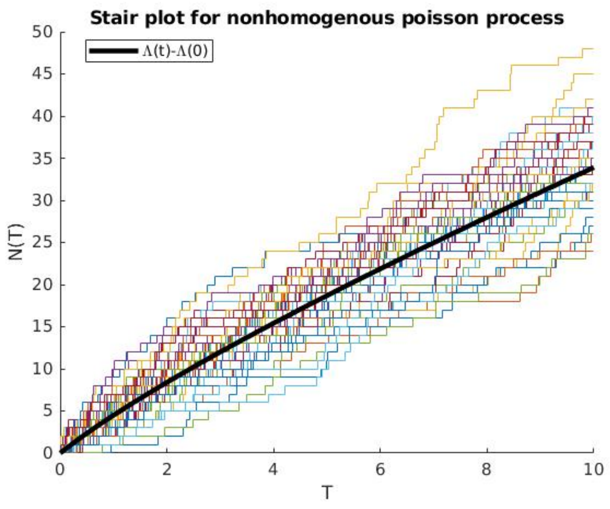

Figure 5: Stair plot over 50 poisson processes over time.

Running the simulation for $N = 100\,000$ (with each $T = 10$) we get the following result:

```
Approximative mean: 33.912
Approximate variance: 33.580
```

We see that this corresponds well to the theoretical value of them both: $\Lambda(T) - \Lambda(0) = \mu = \sigma^2 \approx 33.900$

3.5 Conclusions

The simulated process seems to follow the values expected from the theoretical formulas. This indicates that the algorithms works as intended giving an accurate simulation of the posed distribution.