Mathematical Queueing models and their applications in everyday problems

Mariia Sinkevich
Supervised by Dr. Nicos Georgiou

School of Mathematics and Physical Sciences
Department of Mathematics
University of Sussex

August 2023

Contents

Introduction

Queues have become an integral part of our daily lives in today's world. However, most people do not realize the amount of time they spend waiting in queues for services or in traffic jams. Statistics reveal that people, on average, spend five years of their lives waiting in queues and dealing with traffic congestion.

Have you ever found yourself pondering these questions while waiting in line: How much longer will I have to wait? Should I switch to another line? Why is the line next to me moving faster? In this report, we'll take a closer look at these common queries and explore the complexities of traffic congestion. Additionally, we'll examine different models that can help us better comprehend these systems and their behaviours.

In the first chapter, we delved into the study of Markov chain theory considered various theorems and proved them. We studied the types of queues and the conditions of their formation. You can find practical examples of queues and their analysis using mathematics, in particular Markov chains. At the end of the chapter, we provide the first simulation Python model of a simple queue and the results obtained.

In the first chapter, we explored the theory of Markov chains and proved various theorems. We also studied different types of queues and the conditions that cause them to form. You can find practical examples of queues including its mathematical analysis. Lastly, we provided a simulation model of a simple queue and its results.

The second chapter focuses on Poisson processes, which are the foundation of queueing systems. We covered both the necessary theory and practical applications of Poisson processes, including types of queues, arrival distributions, and customer waiting times. We also derived the conditions and initial parameters for queue formation. Additionally, we used the obtained theory and data to develop computer programs to study the behaviour of two types of Markov queues, M/M/1 and M/M/k.

Within the traffic and modelling chapter, we created simulations to investigate traffic queues on roads. Our approach involved utilizing Poisson processes to model localized traffic and the actions of cars on roadways. A developed Python program generates simulated traffic and presents visual animations of particle flow patterns. By fine-tuning the program's parameters, we can scrutinize traffic behaviour and congestion. Furthermore, we developed a program for the height function that provides data on the amount of cars crossing specific road sections. This data helps us identify areas with weaker traffic flow and observe how traffic jams come about.

Join us on this analytical journey where theory meets code and modelling reveals the operational principles driving traffic dynamics.

Chapter 1

Markov Chains

Consider a discrete-time stochastic process $X_n, n = 0, 1, 2, ...$, where each X_i takes values is some countable set $S = \{s_1, s_2, ..., s_N, ...\}$. We call the possible values for X_n the states of the system and S is the state space.

The Markov property states that to make predictions of the behavior of a system in the future, it suffices to consider only the present state of the system and not the past history.

$$\mathbb{P}\{X_{n+1} = i_{n+1}|X_n = i_n, X_{n-1} = i_{n-1}, ..., X_0 = i_0\} = \mathbb{P}\{X_{n+1} = i_{n+1}|X_n = i_n\} \quad (1.1)$$

The quantities $\mathbb{P}\{X_{n+1} = i_{n+1}|X_n = i_n\} = p_n(i_{n+1}, i_n)$ are called the transition probabilities.

The last probability above we symbolise by $p_n(i_{n+1}, i_n)$ and we call it the transition probability from state i_n to state i_{n+1} at time n. If the transition probability does not depend on the time index n we simply write $p(i_n, i_{n+1})$ and the chain is called time homogeneous.

Definition: Time homogeneous Markov Chain

Definition. A Markov chain $\{X_n\}_{n\in\mathbb{N}_0}$ is time homogeneous if and only if for any $n \in N$, $\mathbb{P}\{X_n = j|X_{n-1} = i\} = p(i, j)$, independent of n. The conditional probabilities $p(i, j)$ are called transition probabilities.

If the state space S has cordiality $N = |S|$ we can construct the $N \times N$ matrix

$$P = \{p_{ij} = p(i, j) = \mathbb{P}\{X_n = j|X_{n-1} = i\}\}_{1\leq i,j\leq N} \quad (1.2)$$

Definition: Stochastic matrix

A square matrix $P_{N\times N} = p_{ij}, 1 \leq i, j \leq N$ is a stochastic matrix if:

1. $0 \leq p_{i,j} \leq 1, \quad \forall\, 1 \leq i, j \leq N$
2. $\sum_{j=1}^{N} p_{i,j} = 1, \quad \forall\, 1 \leq i \leq N$

Any matrix satisfying the previous properties can be the transition matrix for a Markov chain.

EXAMPLE: I.I.D. Random variables

A sequence of independent and identically distributed (I.I.D.) random variables is a Markov chain, albeit a somewhat trivial one. Suppose we have a discrete random variable X taking values in $S = 1, 2, ..., k$ with probability $\mathbb{P}\{X = i\} = p_i$. If we generate an i.i.d. sequence $X_0, X_1, ...$ of random variables with this probability mass function, then it is a Markov chain with transition matrix.

$$P = \begin{pmatrix} p_1 & p_2 & \cdots & p_k \\ p_1 & p_2 & \cdots & p_k \\ \vdots & \vdots & \ddots & \\ p_1 & p_2 & \cdots & p_k \end{pmatrix}$$

EXAMPLE: Random walk

Assume the model for the state of a phone where $X_n = 0$ means that the phone is free at time n and $X_n = 1$ means that the phone is busy. We assume that during each time interval there is a probability p that a call comes in. If the phone is busy during that period, the incoming call does not get through and assuming that the phone system can put one caller on hold. Hence at anytime the number of callers in the system is in the set $S = 0, 1, 2$. Any call will be completed during a time interval with probability q and a new caller will come in with probability p, unless the system is already full.

To model this we set $p(0, 0) = 1 - p$, $p(0, 1) = p$, $p(0, 2) = 0$ since a caller comes in with probability p (again we are assuming only one caller arrives during any time period). $p(2, 0) = 0$, $p(2, 1) = q$, $p(2, 2) = 1 - q$ since no new callers may arrive if there are two callers in the system, and both calls may not end simultaneously.

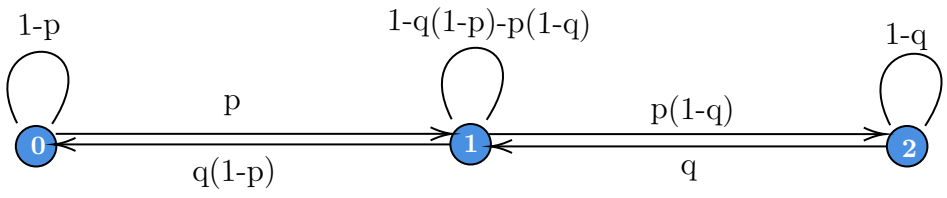
If there is exactly one caller in the system, it is a little more complicated. The state of the system goes from 1 to 0 if the current call is completed and no new callers enter the system, i.e., $p(1, 0) = q(1 - p)$.

Similarly, the state goes from 1 to 2 if the current call is not completed but a new call arrives, i.e., $p(1, 2) = p(1 - q)$.

Since the rows must add to 1, $p(1, 1) = 1 - q(1 - p) - p(1 - q)$ and hence

$$P = \begin{pmatrix} 1-p & p & 0 \\ q(1-p) & 1-q(1-p)-p(1-q) & p(1-q) \\ 0 & q & 1-q \end{pmatrix}$$

The above matrix can be represented graphically as follows:



Proposition

Let X_n be a time-homogeneous Markov chain on a finite state space S with cardinality N and with transition matrix P. Then the m-step transition

$$p_{ij}^{(m)} = \mathbb{P}\{X_m = j | X_0 = i\} = (P^m)_{ij}$$

Furthermore, if $\mu_i = \mathbb{P}\{X_0 = i\}$ the initial distribution of the process, then

$$\mathbb{P}\{X_m = i\} = ((\mu_1, \mu_2, ..., \mu_N)P^m)_i$$

1.1 Classification of states of a Markov Chain

We say that a site j is accessible from site i, denoted by $i \to j$ if there is $n \geq 0$ so that $p^{(n)}(i,j) > 0$. We say that j communicates with i, denoted by $i \leftrightarrow j$ if there exists $m, n \geq 0$ so that $p^{(n)}(i,j) > 0$ and $p^{(m)}(j,i) > 0$. In particular this implies that $i \to j$ and $j \to i$.

LEMMA: The communication relation $\leftrightarrow$ is an equivalence relation for the states of the Markov chain. It is

1. Reflexive: $x \leftrightarrow x$ for any $x \in S$.
2. Symmetric: $x \leftrightarrow y$ implies $y \leftrightarrow x$ for any $x, y \in S$.
3. Transitive: $x \leftrightarrow y$ and $y \leftrightarrow z$ imply $x \leftrightarrow z$ for any $x, y, z \in S$.

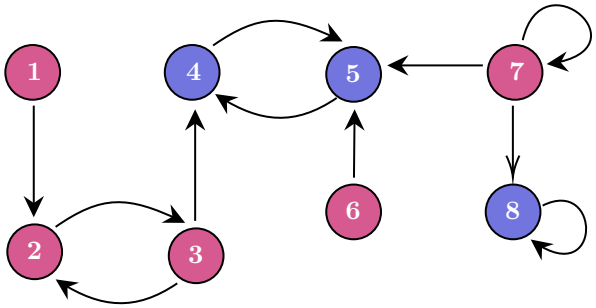
The states of a Markov Chain can be classified based on the transition probability P_{ij} of its transition matrix.

1. A state j is absorbing if it is certain to return to itself in one transition. The transition probability of an absorbing state is $P_{jj} = 1$.
2. A state j is transient if it can reach another state but cannot be reached back from another state. The transition probability in this case will follow the condition $\lim_{n\to\infty} P^{(n)} = 0$, for all i.
3. A state j is recurrent if the probability of being revisited from other states is 1. This is possible only if the state is not transient.
4. A state j is periodic with period $t > 1$ if a return is possible only in $t, 2t, 3t, ...$ steps. The transition probability of a periodic state is $P_{ij}^{(n)} = 0$ when n is not divisible by t.

Definition:

A Markov chain is called **irreducible** if it consists of a single recurrent communication class. Otherwise the state is called **reducible**.

EXAMPLE:



States 1 and 6 are defined as transient. States 2 and 3 are also transient states as the Markov chain can enter from state 2 and move to one from the other for a number of time steps but it will eventually exit from state 3 to 4.

States 4 and 5 are recurrent states . Once the Markov chain reaches state 4 then all subsequent transitions will take it from one to the other. It is also worth mentioning that each state 4 or 5 is reached after two time steps thus these states are defined also as periodic with period 2. Positive recurrent states are the states whose mean recurrence time is finite so states 4 and 5 are characterized positive recurrent as well. Recurrent states with infinite recurrent time are known as null recurrent states and this is feasible when the Markov chain has a infinite number of states.

State 7 is a transient state. Once the Markov chain enters state 7 may, for some finite number of time steps, remain in state 7, but eventually it will move on to either state 5 or state 8. Finally state 8 is a recurrent state but also it is defined as absorbing. It is evident that if a Markov chain enters state 8 it will remain there forever and thus $P_{8,8} = 1$.

1.1.1 Periodicity

Definition:

The period $d(i)$ of a state i is defined by

$$d(i) = gcd\{n : p^{(n)}(i,i) > 0\}$$

We call i **periodic** if $d(i) > 1$ and **aperiodic** if $d(i) = 1$. A chain is called aperiodic if the period of all states is 1.

Theorem:

Let $\{X_n\}_{n\in N_0}$ be an irreducible chain. Then every state has the same period. If the chain is reducible, then each communication class may have different periods.

Proposition:

Let $\{X_n\}_{n\in N_0}$ be an irreducible Markov chain on a finite state space S with period $d \geq 1$. Then there exists an integer M such that $p^{(md)}(i,i) > 0$ for all $m > M$ and all $i \in S$.

1.2 Countable Markov Chains

Consider (time-homogeneous) Markov chains with a countably infinite state space. Let X_n denote a Markov chain. Some of that which was described for finite-state Markov chains holds equally well in the infinite case. We again can speak of the transition matrix, but in this case it becomes an infinite matrix. We will choose not to use the matrix notation here, but simply write the transition probabilities as $p(x,y) = \mathbb{P}\{X_1 = y|X_0 = x\}, x, y \in S$.The transition probabilities are non negative and the "rows" add up to 1, i.e.,for each $x \in S$

$$\sum_{y\in S} p(x,y) = 1$$

We have chosen to use x, y, z for elements of the state space S. We also define the n-step transition probabilities $p_n(x,y) = \mathbb{P}\{X_n = y|X_0 = x\}$

Theorem: Chapman–Kolmogorov equation

If $0 < m, n < \infty$,

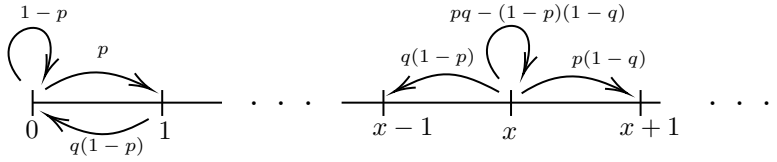
$$p^{(m+n)}(x,y) = \mathbb{P}\{X_{m+n} = y|X_0 = x\} =$$
$$= \sum_{z\in S} \mathbb{P}\{X_{m+n} = y, X_m = z|X_0 = x\} = \sum_{z\in S} p^{(m)}(x,z)p^{(n)}(z,y)$$

EXAMPLE: Queueing Model

Let X_n be the number of customers waiting in line for some service. We think of the first person in line as being serviced while all others are waiting their turn. During each time interval there is a probability p that a new customer arrives. With probability q, the service for the first customer is completed and that customer leaves the queue. We put no limit on the number of customers waiting in line. This is a Markov chain with state space $\{0, 1, 2, ...\}$ and transition probabilities(from the Example)

$$p(x, x-1) = q(1-p), p(x,x) = qp + (1-q)(1-p)$$
$$p(x, x+1) = p(1-q), x > 0$$
$$p(0,0) = 1-p, p(0,1) = p$$

As in the case of finite Markov chains, our goal will be to understand the behavior for large time. Some of the ideas for finite chains apply equally well to the infinite case. For example, the notion of communication classes applies equally well here. Again, we call a Markov chain irreducible if all the states communicate. All the examples discussed in this chapter are irreducible except for a couple of cases where all the states but one communicate and that one state x is absorbing, $p(x,x) = 1$. The example above is aperiodic.



1.3 Recurrence & Transience

For any state x define the integer random variable

$$N_x = \sum_{n=0}^{\infty} \mathbb{1}\{X_n = x\}$$

This random variable counts how many times the chain hits state x. Denote by $\mathbb{P}_x = \mathbb{P}|X_0 = x$.

Definition: Recurrent & Transient states

State x is called recurrent if and only if

$$\mathbb{P}\{N_x = \infty | X_0 = x\} = \mathbb{P}_x\{N_x = \infty\} = 1$$

which is to say that the probability of infinitely many returns to x, having started at x, is 1. Otherwise, state x is called transient.

Theorem: Recurrent & Transient states

Let $\{X_n\}$ a Markov chain on S and let $x \in S$.

1. A state x is recurrent if and only if $\mathbb{P}\{X_n = x \text{ for some } n \geq 1 | X_0 = x\} = 1$.
2. A state x is transient if and only if the probability above is strictly less than 1.

PROOF:

1. ($\Longrightarrow$) By the definition of recurrence,

   $1 = \mathbb{P}_x\{N_x = \infty\} = \mathbb{P}_x\{X_n = x \text{ for infinitely many } n\} \leq \mathbb{P}_x\{X_n = x \text{for some } n > 1\} \leq 1.$

   Therefore we must have equality throughout.

   ($\Longleftarrow$) Suppose that the probability equals 1. Then we can write it as a statement for the hitting time $T_x^{(1)} = min\{k \geq 1 : X = x\}$, namely,

   $$\mathbb{P}\{T_x^{(1)} < \infty | X_0 = x\} = 1$$

   $T_x^{(1)}$ is the time of first return to x and it is a stopping time. Therefore, by the Strong Markov Property, the process $Y_n = X_{T_x^{(1)}+n}$ is a Markov chain that starts

from x with the same transition P and its evolution is independent of anything up to time $T^{(1)}$. Then for this chain we have

$$\mathbb{P}\{Y_n = x \text{ for some } n \geq 1 | Y_0 = x\} = 1$$

This implies that the time of first return for the chain Y_n is finite almost surely. That's the time of second return for the original chain $\{X_n\}_{n\in\mathbb{N}_0}$,

$$T_x^{(2)} = min\{k \geq T_x^{(1)} + 1 : X_k = x\} < \infty$$

So the chain hits x twice in finite time. We can iterate this argument as many times as necessary to conclude that with probability 1,

$$T_x^{(l)} = min\{k \geq T_x^{(l-1)} + 1 : X_k = x\} < \infty, \text{ for any } l \geq 2$$

But then, we can bound

$$\mathbb{P}_x\{N_x \geq l\} = \mathbb{P}_x\{T_x^{(l)} < \infty\} = 1$$

Letting l tend to infinity we see that $\mathbb{P}_x\{N_x = \infty\} = 1$ as required.

2. ($\Longrightarrow$) Assume the state is transient. If $\mathbb{P}_x\{X_n = x \text{ for some } n > 1\} = 1$, by (1) above, the state x is recurrent which is a contradiction. Therefore it has to be $\mathbb{P}_x\{X_n = x \text{ for some } n > 1\} < 1$.

   ($\Longleftarrow$) Assume that $\mathbb{P}_x\{X_n = x \text{ for some } n > 1\} = \rho < 1$, $\mathbb{P}_x\{N_x = \infty\} \leq \rho < 1$. Therefore, by the definition, the state is transient.

Theorem:

Let x be a transient state and y any state in S. Then

1. The expected number of visits to state x, starting from x is

$$\mathbb{E}[N_x | X_0 = x] = \mathbb{E}_x(N_x) = \frac{1}{1 - f(x)} < \infty$$

2. The $\mathbb{P}_x\{N_x = \infty\} = 0$ and it can never happen that $\mathbb{P}_x\{N_x = \infty\} \in (0, 1)$ for any state x.

3. If we know that $\mathbb{P}_y\{N_y = \infty\} > 0$ it has to be that y is recurrent and $\mathbb{P}_y\{N_y = \infty\} = 1$.

PROOF:

1. $\mathbb{E}(N_x) = \sum_{k=1}^{\infty} \mathbb{P}_x\{N_x \geq k\} = \sum_{k=1}^{\infty} (\rho_x)^{k-1} = \frac{1}{1-\rho_x} < \infty$

2. Since the expected value of N_x is finite, it cannot take the value ∞ with positive probability, otherwise the expected value would be infinite. The rest follows by the definition of recurrence and transience.

3. Follows from (2).

Corollary: Let x in S and consider the infinite series $\sum_{n=0}^{\infty} p^{(n)}(x,x)$

1. If the series above diverges, then x is recurrent.
2. If the series above converges, then x is transient.

PROOF: From the previous theorem we see that $\mathbb{E}_x(N_x)$ is finite if and only if x is a transient state. But

$$\infty > \mathbb{E}_x(N_x) = \mathbb{E}_x(\sum_{n=0}^{\infty} \mathbb{1}\{X_n = x\}) = \sum_{n=0}^{\infty} \mathbb{E}_x(\mathbb{1}\{X_n = x\}) = \sum_{n=0}^{\infty} p^{(n)(x,x)}$$

Lets consider another method for determining recurrence or transience. Suppose X_n is an irreducible Markov chain and consider a fixed state which we will denote z. For each state x, we set

$$\alpha(x) = \mathbb{P}\{X_n = z \text{ for some } n \geq 0 | X_0 = x\}$$

Clearly,$\alpha(z) = 1$. If the chain is recurrent, then $\alpha(x) = 1$ for all x. However,if the chain is transient there must be states x with $\alpha(x) < 1$. In fact if the chain is transient there must be points "farther and farther" away from z with $\alpha(x)$ as small as we like.
If $x \neq z$, then

$$\alpha(x) = \mathbb{P}\{X_n = z \text{ for some } n \geq 0 | X_0 = x\} = \mathbb{P}\{X_n = z \text{ for some } n \geq 1 | X_0 = x\} =$$
$$= \sum_{y \in S} \mathbb{P}\{X_1 = y | X_0 = x\} \mathbb{P}\{X_n = z \text{ for some } n \geq 1 | X_1 = y\} = \sum_{y \in S} p(x,y)\alpha(y)$$

Summarizing, $\alpha(x)$ satisfies the following:

$$0 \geq \alpha(x) \geq 1,$$
$$\alpha(z) = 1, \inf\{\alpha(x) : x \in S\} = 0,$$
$$\alpha(x) = \sum_{y \in S} p(x,y)\alpha(y), x \neq z$$

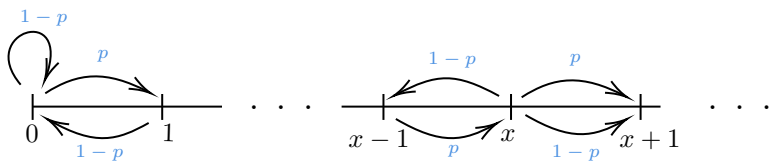
Theorem:

An irreducible Markov chain is transient if and only if for any z we can find a function $\alpha(x)$ satisfying properties above.

EXAMPLE:

Consider a queuing model. Probability that one customer comes is p, leaves with probability $1-p$

$$p(0,0) = 1-p,\ p(x,x+1) = p \text{ and } p(x,x-1) = 1-p \text{ for } x > 0$$



If we can find such a function α that follows all 3 properties, then the chain is transient, otherwise it is recurrent. Set $z = 0$, $x > 0$. From condition (3)

$$\alpha(x) = \sum_{y \in S} p(x,y)\alpha(y), x \neq z$$

$$\alpha(x) = \sum_{y \in S} \pi(y)p(y,x), \sum_{x \in S} \alpha(x) = 1$$

$$\alpha(x) = \alpha(x-1)(1-p) + \alpha(x+1)p$$

This is a second order linear recursion.

$$\alpha(x) = \begin{cases} C_1 + C_2(\frac{1-p}{p})^x & p \neq \frac{1}{2} \\ C_1 + C_2 x & p = \frac{1}{2} \end{cases}$$

From the second condition $\alpha(z) = 1$, $\alpha(0) = 1$

$$\alpha(x) = \begin{cases} 1 - C_2 + C_2(\frac{1-p}{p})^x & p \neq \frac{1}{2} \\ 1 + C_2 x & p = \frac{1}{2} \end{cases}$$

If $p = \frac{1}{2}$: $\alpha(x) = 1 + C_2 x$, $x > 0$ the first condition does not hold. $0 \leq a(x) \leq 1$, $C_2 \neq 0$. Given Markov chain is infinite then $\alpha(x)$ becomes unbounded when $x \to \infty$. If $C_2 = 0$ condition 2 fails since $\alpha(x) = 1$, $inf\{\alpha(x) : x \in S\} = 1 \neq 0$. Hence the chain is recurrent for $p = \frac{1}{2}$. If $p < \frac{1}{2}$:

$$\alpha(x) = 1 - C_2 + C_2(\frac{1-p}{p})^x$$

By the first condition:

$$0 \leq 1 - C_2 + C_2(\frac{1-p}{p})^x \leq 1$$

$$1 - C_2 \leq C_2(\frac{1-p}{p})^x \leq C_2$$

$$\frac{1-C_2}{C_2} \leq (\frac{1-p}{p})^x \leq 1$$

If $C_2 \neq 0$, $x \to \infty$ and $\frac{1-p}{p} > 1 \Leftrightarrow p < \frac{1}{2}$ then $(\frac{1-p}{p})^x$ becomes unbounded as $x \to \infty$. If $C_2 = 0$ condition 2 fails. Therefore the chain is recurrent for $p < \frac{1}{2}$
If $x \to \infty$ and $\frac{1-p}{p} < 1 \Leftrightarrow p > \frac{1}{2}$ then $(\frac{1-p}{p})^x \to 0$ as $x \to \infty$. By condition 2 $\alpha(x) = 1 - C_2 + C_2(\frac{1-p}{p})^x \to 0$ as $x \to \infty$. $1 - C_2 = 0$, $C_2 = 1$. Then

$$\alpha(x) = (\frac{1-p}{p})^x \ , x \geq 0$$

Thus by theorem, MC is transient when $p > \frac{1}{2}$ and recurrent otherwise.

1.4 Long time behaviour, stationary distributions

Definition: Invariant distribution

Let $\{X_n\}_{n \in N_0}$ be a Markov chain on S. A non-negative vector π is called the invariant distribution for the chain, if and only if

$$\pi P = \pi, \sum_{i \in S} \pi_i = 1$$

LEMMA: Let $\{X_n\}_{n\in N_0}$ be a Markov chain on a finite state space S with transition matrix P. Then P has at least one eigenvalue equal to 1.

PROOF: Consider the column vector $1 = (1, ..., 1)^T$ and multiply by matrix P from the left,

$$(P1)^i = \sum_{j\in S} p(i,j)1 = \sum_{j\in S} p(i,j) = 1 = 1_i$$

because the matrix is stochastic. Therefore P1 = 1 and we found a right 1-eigenvector. In particular this shows that the matrix has an eigenvalue 1. The following theorem from linear algebra tells us when we can find invariant distributions.

Theorem: Perron Frobenius Theorem

Let P be a stochastic matrix with all entries positive. Then

1. 1 is a simple eigenvalue for P.
2. The left 1-eigenvector of P can be made to have all positive entries (therefore it can be made into an invariant distribution by multiplying with a constant).
3. All other eigenvalues λ have $|\lambda| < 1$.

Theorem

Let $\{X_n\}$ be an irreducible, aperiodic Markov chain on a finite state space S and transition matrix P. Then

1. There exists $N \in \mathbb{N}$ so that P^N has strictly positive entries.
2. X_n has a unique invariant distribution π with $\pi_i > 0$ for all i.
3. The $\lim_{n\to\infty} P^n$ exists and it is a matrix where every row is π.
4. Let $\mu^{(0)}$ denote any initial probability vector. Then $\lim_{n\to\infty} \mu^{(0)} P^n = \pi$. We say the chain forgets its initial distribution.

PROOF:

1. From the assumptions we have a single closed communication class and it is therefore recurrent. Thus, for any pair $(i,j) \in SS$, we can find an integer n_{ij} so that $p^{(n_{ij})}(i,j) > 0$. Moreover, by Proposition we can find an M so that $p^{(m)}(i,i) > 0$ for all $m > M$ and all $i \in S$. Let $N = M + \sum_{(i,j)\in SS} n_{ij}$. Then, for any (i,j)
$$p^{(N)}(i,j) \geq p^{(n_{ij})}(i,j)p^{(N-n_{ij})}(j,j) > 0$$
. Therefore every entry is positive. The last inequality in the display above is since $N - n_{ij} > M$.

2. By the Perron-Frobenius Theorem, the matrix P^N has a single eigenvalue equal to 1, all other eigenvalues satisfy $|\lambda| < 1$ and there is a unique left 1-eigenvector π with all entries positive. The eigenvalues of P are 1 and $\mu = \sqrt{N}\lambda$ for all λ eigenvalues of P^N. As such, 1 is a simple eigenvalue of P and all other eigenvalues satisfy $|\mu| < 1$. Furthermore, P^N and P share the same eigenvectors, so π is

a left 1-eigenvector of P. Multiply π with an appropriate constant to turn it into an invariant distribution. Invariant distribution π is also unique. The eigenvalue 1 is simple, therefore the eigenspace is 1-dimensional and it is spanned by the vector π.

3. $$Q^{-1}PQ = \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & & & \\ \vdots & & M & \\ 0 & & & \end{pmatrix}$$

   Perform a Jordan decomposition on matrix P. The change of basis matrix Q has as the first column the vector 1 (the right 1-eigenvector) while Q^{-1} has as the first row the invariant distribution π. Furthermore, all eigenvalues of matrix M have absolute values strictly less than 1. Therefore $\lim_{n\to\infty} M^n = 0$, the matrix with all entries 0. Then

$$\lim_{n\to\infty} P^n = Q \begin{pmatrix} 1 & 0 & \dots & 0 \\ 0 & & & \\ \vdots & & \lim_{n\to\infty} M^n & \\ 0 & & & \end{pmatrix} Q^{-1} =$$

$$= Q \begin{pmatrix} \pi_1 & \pi_2 & \dots & \pi_K \\ 0 & & & \\ \vdots & & 0 & \\ 0 & & & \end{pmatrix} = \begin{pmatrix} \pi_1 & \pi_2 & \dots & \pi_K \\ \pi_1 & \pi_2 & \dots & \pi_K \\ \vdots & \vdots & \ddots & \\ \pi_1 & \pi_2 & \dots & \pi_K \end{pmatrix}$$

4. Follows immediately by (3).

Theorem: Periodic irreducible chain

Suppose $\{X_n\}$ is a periodic, irreducible chain with period d on a finite space S. Then

1. The state space S can be partitioned into d disjoint sets, $A_1, A_2, ..., A_d$ such that the chain moves from a state in A_i to a state in $A_{i+1}(A_d \to A_1)$ in consecutive time steps.
2. For very large n, matrices $P_{n+1}, P_{n+2}, ..., P_{n+d}$ have non-zero entries at different coordinates. The position of the non-zero entries repeat every d consecutive powers.
3. The transition matrix P has a simple eigenvalue 1 and it has an invariant distribution π.
4. The are d eigenvalues $z_1, z_2, ..., z_d$ overall, with $|z_i| = 1$, satisfying $z^d = 1$, that are all simple. I.e. the matrix P has complex eigenvalues. All other eigenvalues have $|\lambda| < 1$.
5. Given any initial distribution $\mu^{(0)}$

$$\lim_{n\to\infty} \mu^{(0)}(\frac{1}{d}(P^{dn} + P^{dn+1} + ... + P^{dn+d-1})) = \pi$$

Theorem: Transient to transient

Suppose x is a transient state for $\{X_n\}$ on a state space S. Then, $\lim_{n\to\infty} \mathbb{P}_y\{X_n = x\} = 0$.

PROOF: If $y \nrightarrow x$ then it is immediate that $\mathbb{P}_y\{X_n = x\} = 0$. Now assume $y \to x$ and therefore y is also transient, either in the same class as x or from a different transient class. Then we can find an m so that $p^{(m)}(y, x) > 0$.

Case 1: $y \leftrightarrow x$: Then there exists a k so that $p^(k)(x, y) > 0$. If $y = x$, and since x is transient, then by Corollary, the series $\sum_{n=0}^{\infty} p^{(n)}(x, x)$ converges. As such, it must be $\lim_{n\to\infty} p^{(n)}(x, x) = 0$. Assume $y \neq x$, then for any n large enough

$$p^{(n)}(y, y) \geq p^{(n-m)}(y, x)p^{(k)}(x, y)$$

Since $p^{(n)}(x, x) \to 0$ then so does the right-hand side. But $p^{(k)}(x, y) > 0$ and does not change with n so it must be that $\lim_{n\to\infty} p^{(n-m)}(x, y) = 0$.

Case 2: $y \longrightarrow x$, $x \nrightarrow y$: Define T_{C_x} to be the hitting time of the class of x, starting from y. Then

$$\lim_{n\to\infty} \mathbb{P}_y\{X_n = x\} = \lim_{n\to\infty} (\mathbb{P}_y\{X_n = x, T_{C_x} < \infty\} + \mathbb{P}_y\{X_n = x, T_{C_x} = \infty\})$$

The second probability is always 0, since if we never hit the class of x we will never hit x. The first probability is

$$\lim_{n\to\infty} \mathbb{P}_y\{X_n = x, T_{C_x} < \infty\} = \lim_{n\to\infty} \sum_{u\in C_x} \mathbb{P}_y\{X_n = x, T_{C_x} \leq n | X_{T_{C_x}} = u\}\mathbb{P}_y\{X_{T_{C_x}} = u\} =$$

$$= \sum_{u\in C_x} \sum_{m=1}^{\infty} \lim_{n\to\infty} \mathbb{P}_y\{X_n = x, T_{C_x} = m | X_{T_{C_x}} = u\}\mathbb{P}_y\{X_{T_{C_x}} = u\} =$$

$$= \sum_{u\in C_x} \sum_{m=1}^{\infty} \lim_{k\to\infty} \mathbb{P}_y\{X_{m+k} = x, T_{C_x} = m | X_{T_{C_x}} = u\}\mathbb{P}_y\{X_{T_{C_x}} = u\} =$$

$$= \sum_{u\in C_x} \lim_{k\to\infty} \mathbb{P}_y\{X_{k+T_{C_x}} = x, T_{C_x} = m | X_{T_{C_x}} = u\}\mathbb{P}_y\{X_{T_{C_x}} = u\} =$$

$$= \sum_{u\in C_x} \lim_{k\to\infty} \mathbb{P}_u\{X_k = x\} \sum_{m=1}^{\infty} \mathbb{P}_y\{T_{C_x} = m, X_m = u\} \leq$$

$$\leq \sum_{u\in C_x} \lim_{k\to\infty} \mathbb{P}_u\{X_k = x\} \sum_{m=1}^{\infty} \mathbb{P}_y\{T_{C_x} = m\} =$$

$$= \sum_{u\in C_x} \lim_{k\to\infty} \mathbb{P}_u\{X_k = x\}\mathbb{P}_y\{T_{C_x} < \infty\} \leq$$

$$\leq \sum_{u\in C_x} \lim_{k\to\infty} \mathbb{P}_u\{X_k = x\} \to 0$$

Theorem: Transient to recurrent

Assume $\{X_n\}_{n\in\mathbf{N}}$ has k recurrent classes $R_1, ...R_k$. Let $\pi^{(i)}$ denote the invariant probability for the stochastic sub-matrix that corresponds to R_i. For any $y \in R_i$ and x transient $\lim_{n\to\infty} \mathbb{P}\{X_n = y | X_0 = x\} = \alpha_{R_i}(x)\pi_y^{(i)}$.

1.5 Return times

Let X_n be an irreducible Markov chain on a finite space S. Define

$$Y_n(j) = \sum_{m=0}^{n} \mathbb{1}\{X_m = j\}$$

which represents the number of visits to state j. Then the limiting expected proportion of time the chain spent on state j up to time n as $n \to \infty$ is

$$\lim_{n\to\infty} \frac{1}{n+1}\mathbb{E}(Y_n(j)|X_0 = i) = \lim_{n\to\infty} \frac{1}{n+1} \sum_{m=0}^{n} \mathbb{P}\{X_m = j|X_0 = i\} = \pi_j$$

LEMMA: The unique invariant distribution π represents the proportion of time that a chain spends on a given state in the long-run.

The time of first return to state x is

$$T_x^{(1)} = inf\{k \geq 1 : X_k = x\}, X_0 = x$$

The time of the k-th return is defined as

$$T_x^{(k)} = inf\{k > T_x^{(k-1)} : X_k = x\}$$

Proposition

Let X_n be an irreducible chain on a finite state space S with unique invariant distribution π

$$\mathbb{E}_x(T_x^{(1)}) = \frac{1}{\pi_x}$$

PROOF: Define the inverse process

$$L_n = max\{k \in \mathbb{N} : T_x^{(k)} \leq n\} = max\{k \in \mathbb{N} : Y_n(x) > k\}$$

Let $k_{\mathcal{E},n} = \frac{n}{(1+\mathcal{E})\mathbb{E}(Z_i)}$

$$\mathbb{P}\{L_n > k_{\mathcal{E},n}\} = \mathbb{P}\{T_x^{k_{\mathcal{E},n}} < n\} = \mathbb{P}\{\sum_{i=1}^{k_{\mathcal{E},n}} Z_i < n\} =$$

$$= \mathbb{P}\{{k_{\mathcal{E},n}}^{-1} \sum_{i=1}^{k_{\mathcal{E},n}} Z_i < (1+\mathcal{E})\mathbb{E}(Z_i)\} =$$

$$= \mathbb{P}\{{k_{\mathcal{E},n}}^{-1} \sum_{i=1}^{k_{\mathcal{E},n}} Z_i - \mathbb{E}(Z_1) < \mathcal{E}\mathbb{E}(Z_1)\} \to 1 \text{ by the Law of Large Numbers.}$$

Similar arguments show that with probability tending to 1 as n grows, $L_n < \frac{n}{(1-\mathcal{E})\mathbb{E}(Z_1)}$, i.e. overall we have the equivalent formulation

$$1 = \lim_{n\to\infty} \mathbb{P}\{T_x^{k_{\mathcal{E},n}} < n < T_x^{k_{-\mathcal{E},n}}\} = \lim_{n\to\infty} \mathbb{P}\{k_{\mathcal{E},n} < Y_n(x) < k_{-\mathcal{E},n}\} =$$

$$= \lim_{n\to\infty} \mathbb{P}\{(1-\mathcal{E})\mathbb{E}(Z_1) < \frac{n}{Y_n(x)} < (1+\mathcal{E})\mathbb{E}(Z_1)\} = \lim_{n\to\infty} \mathbb{P}\{|\frac{n}{Y_n(x)} - \mathbb{E}(Z_1)| < \delta\}$$

Then we have that $\frac{Y_n(x)}{n} \to \frac{1}{\mathbb{E}(Z_1)}$ in probability so there is a subsequence converging in L_1.

1.6 Infinite state space

Definition: Invariant distribution for infinite chains

Let X_n be an irreducible Markov chain on a countable space S. Then $\{\pi(y)\}_{y\in S}$ is an invariant probability distribution on S for X_n if and only if $0 < \pi(x)$ for all $x \in S$ and

$$\pi(x) = \sum_{y\in S} \pi(y)p(y,x), \sum_{x\in S} \pi(x) = 1$$

Definition: Null vs positive recurrence

We say that a state x is null recurrent if and only if x is recurrent and $\mathbb{E}_x(T_x^{(1)}) = \infty$. We say that a state x is positive recurrent if and only if $\mathbb{E}_x(T_x^{(1)}) < \infty$.

Combining this definition with the previous theorem, we conclude that an infinite Markov chain with at least one positive recurrent state has at least one invariant distribution with strictly positive entries on coordinates corresponding to the positive recurrent states.

Theorem

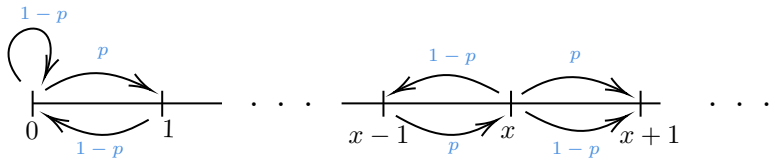
Let X_n be an irreducible recurrent Markov chain. Then

1. Either all states are positive recurrent or all states are null recurrent, i.e. null or positive recurrence are class properties.
2. The chain is positive recurrent if and only if it has an invariant distribution π and in that case $\pi(x) > 0$ for all $x \in S$.

Any irreducible, positive recurrent chain has a unique invariant distribution.

EXAMPLE: Consider the previous example. Probability that 1 customer comes is p, leaves with probability $1-p$

$$p(0,0) = 1-p,\ p(x,x+1) = p \text{ and } p(x,x-1) = 1-p \text{ for } x > 0$$



By the definition the invariant distribution for a given Markov Chain

$$\pi(x) = \sum_{y\in S} \pi(y)p(y,x), \sum_{x\in S} \pi(x) = 1$$

$$\pi(x) = \pi(x-1)p + \pi(x+1)(1-p)$$

This is a second order linear recursion.

$$\pi(x) = \begin{cases} C_1 + C_2(\frac{p}{1-p})^x & p \neq \frac{1}{2} \\ C_1 + C_2 x & p = \frac{1}{2} \end{cases}$$

By the MC model we see that

$$\pi(0) = \pi(0)(1-p) + \pi(1)(1-p)$$

$$\pi(1)(1-p) = \pi(0)p \ , \ \pi(1) = \pi(0)\frac{p}{1-p}$$

Plugging this into the above system of equations gives:

$$\pi(x) = \begin{cases} C_2(\frac{p}{1-p})^x & p \neq \frac{1}{2} \\ C_1 & p = \frac{1}{2} \end{cases}$$

For $p \neq \frac{1}{2}$:

$$\pi(x) = C_2(\frac{p}{1-p})^x$$

$$\sum_{x \in S} \pi(x) = 1 \Rightarrow \sum_{x \in S} C_2(\frac{p}{1-p})^x = 1$$

$$C_2 \sum_{x=0}^{\infty} (\frac{p}{1-p})^x = C_2 \frac{1}{1 - \frac{p}{1-p}} = C_2 \frac{1-p}{1-2p} = 1$$

$$C_2 = \frac{1-2p}{1-p}$$

If $|\frac{p}{1-p}| < 1 \Leftrightarrow p < \frac{1}{2}$

Therefore $\pi(x) = \frac{1-2p}{1-p}(\frac{p}{1-p})^x$ if $p < \frac{1}{2}$. In this case the chain is positive recurrent.

If $p > \frac{1}{2}$, $\sum_{x=0}^{\infty}(\frac{p}{1-p})^x$ diverges. In the previous example we proved that this Markov chain is transient when $p > \frac{1}{2}$.

For $p = \frac{1}{2}$: $\pi(x) = C_1 \ \sum_{x \in S} \pi(x) = \sum_{x=0}^{\infty} \pi C_1 \neq 1$.

The given Markov chain is not positive recurrent (and not transient by the previous example), thus it is null recurrent for $p = \frac{1}{2}$. To summarise:

positive recurrent if $p < \frac{1}{2}$,
null recurrent if $p = \frac{1}{2}$,
transient if $p > \frac{1}{2}$.

1.6.1 Simulation Model

Following the previous example, let's create a simulation model of the queue and see how the queue behaves with different initial parameters. In one time step, one object joins or leaves the system. Only one event can occur at each time step.

Let's write a function to simulate this system. For each time step we produce a random number U that is uniformly distributed with parameters 0 and 1, $U \sim Unif[0,1]$.

If U is less than p (initial probability), then a new object enters the model. If U is greater than or equal to p and the number of objects in the queue is greater than

0, one object leaves the queue. If the number of objects is 0 and U is greater than or equal to p, the number of objects does not change. We continue the loop until we reach the required number of steps (time steps).

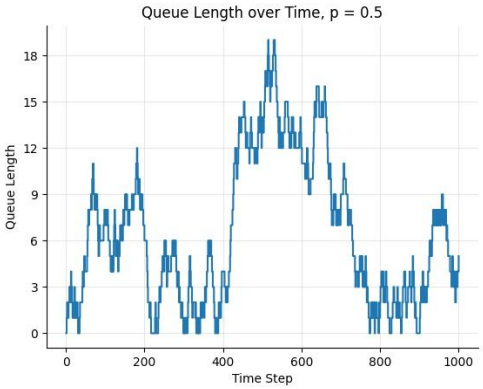
Below you can find an example of this algorithm in the Python programming language:

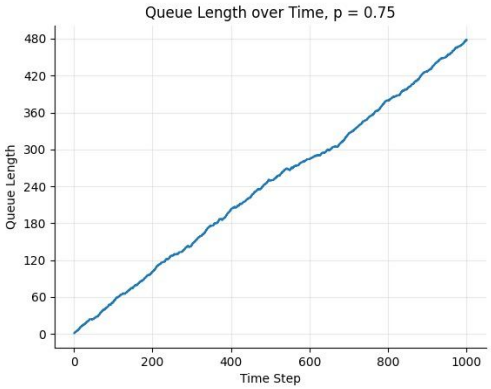
```
import random
import matplotlib.pyplot as plt
from collections import Counter
from pylab import MaxNLocator


def markov_queue_simulation(p, num_steps):  # p - probability
    queue = []  # Initialize an empty queue
    num_customers = 0  # Number of customers in the queue
    queue_lengths = []  # List to store the queue lengths over time

    for _ in range(num_steps):
        # Check if a customer arrives
        if random.random() < p:
            queue.append(1)  # Add a customer to the queue
            num_customers += 1

        # Check if a customer departs
        if num_customers > 0 and random.random() <= 1 - p:
            queue.pop(0)  # Remove the first customer from the queue
            num_customers -= 1

        queue_lengths.append(num_customers)  # Store the current queue length

        print(f"Time Step {_ + 1}: Queue Length = {num_customers}")

    # Plot the queue length chart
    plt.plot(range(1, num_steps + 1), queue_lengths, drawstyle='steps-post')
    plt.xlabel('Time Step')
    plt.ylabel('Queue Length')
    plt.title('Queue Length over Time, p = ' + p)

    plt.gca().spines['top'].set_visible(False)
    plt.gca().spines['right'].set_visible(False)
    plt.grid(color='lightgray', alpha=0.5, zorder=1)
    a = plt.gca()
    a.yaxis.set_major_locator(MaxNLocator(integer=True))
    plt.show()

    # Calculate the number of occurrences for each queue length
    queue_length_counts = Counter(queue_lengths)

    # Calculate the probabilities of each queue length
    total_occurrences = sum(queue_length_counts.values())
    queue_length_probabilities = {
        length: count / total_occurrences
        for length, count in queue_length_counts.items()
    }

    # Print the probabilities of each queue length
    print("Queue Length Probabilities:")
    for length, probability in queue_length_probabilities.items():
        print(f"Length {length}: Probability {probability}")
    return queue_length_counts


# Example usage:
p = 0.5  # Probability of customer arrival
```

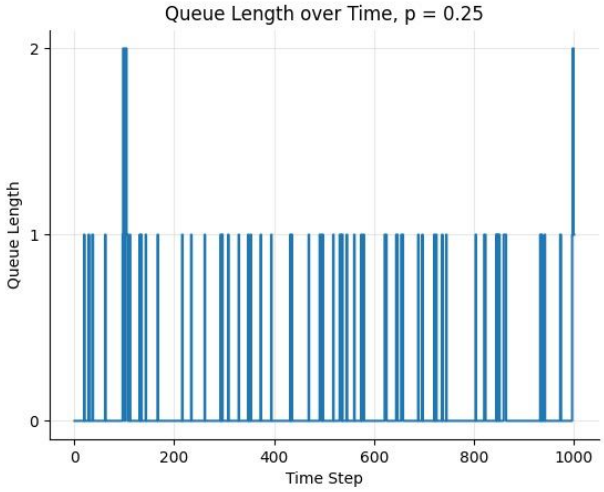
```
num_steps = 780  # Number of simulation steps

queue_length_counts = markov_queue_simulation(p, num_steps)
```

This code has a plotting function, which shows the behaviour of the queue over time. The program also outputs the invariant distribution using the Monte Carlo method, considering the results obtained.

Examples of outputs with various probabilities p are given below.







For a positive recurring Markov Chain the invariant distribution is

Queue Length Probabilities:
Length 0: Probability 0.913
Length 1: Probability 0.081
Length 2: Probability 0.006

Chapter 2

Poisson Process

Consider X_t the number of customers arriving in a line by time t, $t \in \mathbb{R}_0^+$.

1. The number of customers arriving during one time interval does not affect the number arriving during a different time interval. For $s_1 \leq t_1 \leq s_2 \leq t_2 \leq \ \leq s_n \leq t_n$, the random variables $X_{t_1} - X_{s_1}, ..., X_{t_n} - X_{s_n}$ are independent
2. The "average" rate at which customers arrive remains constant. Let λ be the rate at which customers arrive, i.e., on the average we expect λt customers in time t. In a small time interval $[t, t + \Delta t]$, we expect that a new customer arrives with rate about $\lambda \Delta t$
3. Customers arrive one at a time. The probability that more than one customer comes in during a small time interval is significantly smaller than this.

$$\mathbb{P}\{X_{t+\Delta t} = X_t\} = 1 - \lambda \Delta t + o(\Delta t) \tag{8.1}$$

$$\mathbb{P}\{X_{t+\Delta t} = X_t + 1\} = \lambda \Delta t + o(\Delta t) \tag{8.2}$$

$$\mathbb{P}\{X_{t+\Delta t} \geq X_t + 2\} = o(\Delta t) \tag{8.3}$$

A stochastic process X_t with $X_0 = 0$ satisfying these assumptions is called a Poisson process with rate parameter λ.

$$P_k(t) = \mathbb{P}\{X_t = k\}$$

Note that $P_0(0) = 1$ and $P_k(0) = 0$, $k > 0$. Equations (8.1) - (8.3) can be used to give a system of differential equations for $P_k(t)$. The definition of the derivative gives

$$P_k'(t) = \lim_{\Delta t \to 0} \frac{1}{\Delta t}(\mathbb{P}\{X_{t+\Delta t} = k\} - \mathbb{P}\{X_t = k\})$$

$$\begin{aligned}\mathbb{P}\{X_{t+\Delta t} = k\} = \mathbb{P}\{X_t = k\}\mathbb{P}\{X_{t+\Delta t} = k | X_t = k\} + \\ + \mathbb{P}\{X_t = k-1\}\mathbb{P}\{X_{t+\Delta t} = k | X_t = k-1\} + \\ + \mathbb{P}\{X_t \leq k-2\}\mathbb{P}\{X_{t+\Delta t} = k | X_t \leq k-2\} = \\ = P_k(t)(1 - \lambda \Delta t) + P_{k-1}(t)\lambda \Delta t + o(\Delta t)\end{aligned}$$

$$P_k'(t) = \lim_{\Delta t \to 0} \frac{1}{\Delta t}(P_k(t)(1 - \lambda \Delta t) + P_{k-1}(t)\lambda \Delta t + o(\Delta t) - P_k(t)) = \lambda P_{k-1}(t) - \lambda P_k(t)$$

$$P_k'(t) = \lambda P_{k-1}(t) - \lambda P_k(t)$$

For $k = 0$ the differential equation $P_0'(t) = -\lambda P_o(t), \qquad P_0(0) = 1$ has solution $P_0(t) = e^{-\lambda t}$

To solve for $k > 0$ consider $f_k(t) = e^{\lambda t} P_k(t)$ Then $f_0(t) = 1$ and the differential equation becomes

$$f_k'(t) = \lambda f_{k-1}(t), \quad f_k(0) = 0$$

Inductively we can prove that the solution is

$$f_k(t) = \lambda^k t^k / k!$$

Then

$$P_k(t) = e^{-\lambda t}(\lambda t)^k / k!$$

i.e.,X_t has a Poisson distribution with parameter λt.

Another way to view the Poisson process is to consider the waiting times between customers. Let $T_n,\ n = 1, 2, ...$ be the time between the arrivals of the $(n-1)$st and nth customers. Let $Y_n = T_1 + \ + T_n$ be the total amount of time until n customers arrive.

$$Y_n = \inf\{t : X_t = n\}, \quad T_n = Y_n - Y_{n-1}$$

The T_i should be independent, identically distributed random variables. One property that the T_i should satisfy is the loss of memory property: if we have waited s time units for a customer and no one has arrived, the chance that a customer will come in the next t time units is exactly the same as if there had been some customers before.

$$\mathbb{P}\{T_i \geq s + t | T_i \geq s\} = \mathbb{P}\{T_i \geq t\}$$

The only real-valued functions satisfying $f(s + t) = f(s)f(t)$ are of the form $f(t) = e^{-bt}$. Hence the distribution of T_i must be an exponential distribution with parameter b. Let's calculate the value of b. For large t values we expect for there to be about λt customers. Hence, $Y_{\lambda t} \approx t$. But $Y_n \approx n\mathbb{E}(T_i) = n/b$. Hence $\lambda = b$ and $T_i \sim Exp(\lambda)$.

2.1 Finite State Space

Suppose $T_1, ..., T_n$ are independent random variables, each exponential with rates $b_1, ..., b_n$, respectively. Intuitively, we can think of n alarm clocks which will go off at times $T_1, ..., T_n$. Consider the first time when any of the alarm clocks goes off; more precisely, consider the random variable

$$T = min\{T_1, ..., T_n\}$$

$$\mathbb{P}\{T \geq t\} = P\{T_1 \geq t, ..., T_n \geq t\} =$$
$$= \mathbb{P}\{T_1 \geq t\}\mathbb{P}\{T_2 \geq t\}...\mathbb{P}\{T_n \geq t\} = e^{-b_1 t}e^{-b_2 t}...e^{-b_n t} = e^{-(b_1+...+b_n)t}$$

Hence, T has an exponential distribution with parameter $b_1 + ... + b_n$. Moreover, it is easy to give the probabilities for which of the clocks goes off first,

$$\mathbb{P}\{T_1 = T\} = \int_0^\infty \mathbb{P}\{T_2 > t, ..., T_n > t\} d\mathbb{P}\{T_1 = T\} =$$
$$= \int_0^\infty e^{-(b_1+...+b_n)t} b_1 e^{-b_1 t} dt = \frac{b_1}{b_1 + ... + b_n}$$

In other words, the probability that the ith clock goes off first is the ratio of b_i to $b_1 + \ + b_n$. If we are given an infinite sequence of exponential random variables $T_1, T_2, ...$, with parameters b_1, b_2,..., the same result holds provided that $b_1 + b_2 + < \infty$

Suppose now that we have a finite state space S. We will define a continuous-time process X_t on S that has the Markov property,

$$\mathbb{P}\{X_t = y | X_r, 0 \leq r \leq s\} = \mathbb{P}\{X_t = y | X_s\},$$

and that is time-homogeneous,

$$\mathbb{P}\{X_t = y | X_s = x\} = \mathbb{P}\{X_t - s = y | X_0 = x\}$$

For each $x, y \in S$,$x \neq y$ we assign a nonnegative number $\alpha(x, y)$ that we think of as the rate at which the chain changes from state x to state y. We let $\alpha(x)$ denote the total rate at which the chain is changing from state x, i.e.,

$$\alpha(x) = \sum_{y \neq x} \alpha(x, y)$$

A time-homogeneous continuous-time Markov chain with rates α is a stochastic process X_t taking values in S satisfying

$$\mathbb{P}\{X_{t+\Delta t} = x | X_t = x\} = 1 - \alpha(x)\Delta t + o(\Delta t)$$

$$\mathbb{P}\{X_{t+\Delta t} = x | X_t = y\} = \lambda(y, x)\Delta t + o(\Delta t), \quad x \neq y$$

In other words, the probability that the chain in state y jumps to a different state x in a small time interval of length Δt is about $\lambda(y, x)\Delta t$. If we let $p_x(t) = \mathbb{P}\{X_t = x\}$,then the equations above can be shown to give a system of linear differential equations,

$$p_x'(t) = \lim_{\Delta t \to 0} \frac{\mathbb{P}\{X_{t+\Delta t} = x\} - \mathbb{P}\{X_t = x\}}{\Delta t} =$$

$$= \lim_{\Delta t \to 0} \frac{\sum_y \mathbb{P}\{X_{t+\Delta t} = x | X_t = y\}\{X_t = y\} - \sum_y \mathbb{P}\{X_t = x | X_t = y\}\{X_t = y\}}{\Delta t} =$$

$$= \lim_{\Delta t \to 0} \frac{\sum_y [\mathbb{P}\{X_{t+\Delta t} = x | X_t = y\} - \mathbb{P}\{X_t = x | X_t = y\}]}{\Delta t}\{X_t = y\} =$$

$$= \lim_{\Delta t \to 0} [\frac{(\mathbb{P}\{X_{t+\Delta t} = x | X_t = x\} - 1)p_x(t)}{\Delta t} + \frac{(\mathbb{P}\{X_{t+\Delta t} = x | X_t = x - 1\} - 0)p_{x-1}(t)}{\Delta t}] =$$

$$= -\lambda p_x(t) + \lambda p_{x-1}(t)$$

If we impose an initial condition, $p_x(0)$, $x \in S$, then we can solve the system. This system is often written in matrix form. Let A be the matrix whose (x, y) entry equals $\alpha(x, y)$ if $x \neq y$ and equals $-\alpha(x)$ if $x = y$. Then if $\bar{p}(t)$ denotes the vector of probabilities, the system can be written

$$\bar{p}'(t) = \bar{p}(t)A$$

The matrix A is called the infinitesimal generator of the chain. Note that the row sums of A equal 0, the nondiagonal entries of A are nonnegative, and the diagonal entries are nonpositive. From differential equations, we can give the solution

$$\bar{p}'(t) = \bar{p}(0)e^{tA}$$

We can also write this in terms of transition matrices. Let $p_t(x, y) = \mathbb{P}X_t = y | X_0 = x$ and let Pt be the matrix whose (x, y) entry is $p_t(x, y)$. The system of differential equations can be written as a single matrix equation:

$$\frac{d}{dt}\mathbb{P}_t = P_t A, \quad P_0 = I$$

$$P_t = e^{tA}$$

2.2 Birth-and-Death Processes

Consider a large class of infinite state space, continuous-time Markov chains that are known by the name of birth-and-death processes.The state space will be $\{0, 1, 2, ...\}$, and changes of state will always be from n to $n + 1$ or n to $n - 1$. Intuitively we can view the state of the system as the size of a population that can increase or decrease by 1 by a "birth" or a "death,"respectively. To describe the chain, we give birth rates λ_n,$n = 0, 1, 2, ...$ and death rates μ_n, $n = 1, 2, 3, ...$ If the population is currently n, then new individuals arrive at rate λ_n and individuals leave at rate μ_n (note if the population is 0 there can be no deaths, so $\mu_n = 0$).
Let X_t denote the state of the chain at time t, then

$$\mathbb{P}\{X_{t+\Delta t} = n | X_t = n\} = 1 - (\mu_n + \lambda_n)\Delta t + o(\Delta t)$$

$$\mathbb{P}\{X_{t+\Delta t} = n + 1 | X_t = n\} = \lambda_n \Delta t + o(\Delta t)$$

$$\mathbb{P}\{X_{t+\Delta t} = n - 1 | X_t = n\} = \mu_n \Delta t + o(\Delta t)$$

EXAMPLE: Markovian Queueing Models
Suppose X_t denotes the number of people on line for some service. We assume that people arrive at a rate λ; more precisely, the arrival rate of customers follows a Poisson process with rate λ. Customers are also serviced at an exponential rate μ.

(a) **M/M/1 queue** In this case there is one server and only the first person in line is being serviced. This gives a birth-and-death process with $\lambda_n = \lambda$ and $\mu_n = \mu (n \geq 1)$. The two M in the notation refer to the fact that both the arrival and the service times are exponential and hence the process is Markovian. The 1 denotes the fact that there is one server.

(b) **M/M/k queue** In this case there are k servers and anyone in the first k positions in the line can be served. If there are k people being served, and each one is served at rate μ, then the rate at which people are leaving the system is $k\mu$. This gives a birth-and-death process with $\lambda_n = \lambda$ and

$$\mu_n = \begin{cases} n\mu & \text{if } n \leq k \\ k\mu & \text{if } n \geq k \end{cases}$$

(c) **M/M/∞ queue** In this case there are an infinite number of servers,so everyone in line has a chance of being served. In this case $\lambda_n = \lambda$ and $\mu_n = n\mu$.

We call the birth-and-death chain irreducible if all the states communicate. It is not very difficult to see that this happens if and only if all the $\lambda_n (n \geq 0)$ and all the $\mu_n (n \geq 1)$ are positive. An irreducible chain is recurrent if one always returns to a state; otherwise, it is called transient. For any birth-and-death process, there is a discrete-time Markov chain on $\{0, 1, 2, ...\}$ that follows the continuous-time chain "when it moves." It has transition probabilities

$$p(n, n-1) = \frac{\mu_n}{\mu_n + \lambda_n}, p(n, n+1) = \frac{\lambda_n}{\mu_n + \lambda_n}$$

The continuous-time chain is recurrent if and only if the corresponding discrete-time chain is recurrent. Let $a(n)$ be the probability that the chain starting at state n ever reaches state 0. Note that $a(0) = 1$ and the value of $a(n)$ is the

same whether one considers the continuous-time or the discrete-time chain. a(n) satisfies

$$a(n)(\mu_n + \lambda_n) = a(n-1)\mu_n + a(n+1)\lambda_n, n > 0$$

If the chain is transient, $a(n) \to 0$ as $n \to \infty$. If the chain is recurrent, no solution of this equation will exist with $a(0) = 1, 0 \leq a(n) \leq 1$, $a(n) \to 0$ as $n \to \infty$.

We now give a necessary and sufficient condition for a birth-and-death chain to be transient. We will try to find the function $a(n)$.

$$a(n) - a(n+1) = \frac{\mu_n}{\lambda_n}[a(n-1) - a(n)], n \geq 1$$

If we continue, we get

$$a(n) - a(n+1) = \frac{\mu_1 ... \mu_n}{\lambda_1 ... \lambda_n}[a(0) - a(1)]$$

$$a(n+1) = [a(n+1) - a(0)] + a(0) = \sum_{j=0}^{n}[a(j+1) - a(j)] + 1 =$$

$$= [a(1) - 1]\sum_{j=0}^{n}\frac{\mu_1 ... \mu_n}{\lambda_1 ... \lambda_n} + 1$$

where the $j = 0$ term of the sum equals 1 by convention. We can find a nontrivial solution if the sum converges.

Proposition

The birth-and-death chain is transient if and only if

$$\sum_{n=1}^{\infty}\frac{\mu_1 ... \mu_n}{\lambda_1 ... \lambda_n} < \infty$$

EXAMPLE:

$M/M/1$ **queue**:

$$\sum_{n=1}^{\infty}\frac{\mu_1 ... \mu_n}{\lambda_1 ... \lambda_n} = \sum_{n=1}^{\infty}\frac{\mu}{\lambda}^n$$

The sum converges if $\mu < \lambda$

$M/M/k$ **queue**: $\lambda_n = \lambda$ and $\mu_n = \begin{cases} n\mu & \text{if } n \leq k \\ k\mu & \text{if } n \geq k \end{cases}$

$$\sum_{n=1}^{\infty}\frac{\mu_1 ... \mu_n}{\lambda_1 ... \lambda_n} = \sum_{n=1}^{\infty}\frac{k!}{k^k}(\frac{\mu k}{\lambda})^n$$

for any $n \geq k$ In this case the sum is finite and the chain is transient if and only if $k\mu < \lambda$

$M/M/\infty$ **queue**:

$\lambda_n = \lambda$, $\mu_n = n\mu$

$$\sum_{n=1}^{\infty}\frac{\mu_1 ... \mu_n}{\lambda_1 ... \lambda_n} = \sum_{n=1}^{\infty}n!(\frac{\mu}{\lambda})^n = \infty$$

Hence, for all values of μ and λ the chain is recurrent.

These three results can be summarized by saying that the queuing models are transient (and hence the lines grow longer and longer) if and only if the (maximal) service rate is strictly less than the arrival rate.

An irreducible chain is positive recurrent if there exists a probability distribution $\pi(n)$ such that

$$\lim_{t\to\infty}\{X_t = n | X_0 = m\} = \pi(n), \text{ for all } m$$

Otherwise a recurrent chain is called null recurrent.
If the system is in the limiting probability, i.e., if $\mathbb{P}_n(t) = \pi(n)$, then $\mathbb{P}'_n(t)$ should equal 0.

$$\mathbb{P}'_n(t) = \mu_{n+1}\mathbb{P}_{n+1}(t) + \lambda_{n-1}\mathbb{P}_{n-1}(t) - (\mu_n + \lambda_n)\mathbb{P}_n(t)$$

$$0 = \mu_{n+1}\pi(n+1) + \lambda_{n-1}\pi(n-1) - (\mu_n + \lambda_n)\pi(n)$$

For the case of discrete-time chains, we can find π by solving these equations. If we can find a probability distribution that satisfies the previous equation, then the chain is positive recurrent and that distribution is the unique equilibrium distribution.

First, the equation for $n = 0$ gives

$$\pi(1) = \frac{\lambda_0}{\mu_1}\pi(0)$$

For $n \geq 1$:

$$\mu_{n+1}\pi(n+1) - \lambda_n\pi(n) = \mu_n\pi(n) - \lambda_{n-1}\pi(n-1)$$

If we iterate this equation, we get

$$\mu_{n+1}\pi(n+1) - \lambda_n\pi(n) = \mu_1\pi(1) - \lambda_0\pi(0) = \mu_1\frac{\lambda_0}{\mu_1}\pi(0) - \lambda_0\pi(0) = 0$$

Thus, $\pi(n+1) = \frac{\lambda_n}{\mu_{n+1}}\pi(n)$. By iterating we get:

$$\pi(n) = \frac{\lambda_0 ... \lambda_{n-1}}{\mu_1 ... \mu_n}\pi(0)$$

We now impose the condition that π be a probability measure. We can arrange this if and only if $\sum_n \pi(n) < \infty$

Proposition

A birth-and-death chain is positive recurrent if and only if

$$q = \sum_{n=0}^{\infty}\frac{\lambda_0 ... \lambda_{n-1}}{\mu_1 ... \mu_n} < \infty$$

Then the invariant probability is given by

$$\pi(n) = \frac{\lambda_0 ... \lambda_{n-1}}{\mu_1 ... \mu_n}q^{-1}$$

EXAMPLE: $M/M/1$ **queue:**

$$\sum_{n=0}^{\infty} \frac{\lambda_0 ... \lambda_{n-1}}{\mu_1 ... \mu_n} = \sum_{n=0}^{\infty} \frac{\lambda^n}{\mu^n} = \frac{\mu}{\mu - \lambda} \text{ if } \lambda < \mu$$

Hence, the equilibrium distribution is

$$\pi(n) = \frac{\mu}{\mu - \lambda} (\frac{\lambda}{\mu})^n$$

$M/M/k$ **queue:** $\lambda_n = \lambda$ and $\mu_n = \begin{cases} n\mu & \text{if } n \leq k \\ k\mu & \text{if } n \geq k \end{cases}$

$$\sum_{n=0}^{\infty} \frac{\lambda_0 ... \lambda_{n-1}}{\mu_1 ... \mu_n} = \sum_{n=0}^{k} \frac{\lambda^n}{(n\mu)^n} + \sum_{n=k}^{\infty} \frac{\lambda^n}{(k\mu)^n}$$

The first sum is finite.
Hence, the queue is positive recurrent if $\sum_{n=k}^{\infty} \frac{\lambda^n}{(k\mu)^n} < \infty$, $\lambda < k\mu$
$M/M/\infty$ **queue:**

$$\sum_{n=0}^{\infty} \frac{\lambda_0 ... \lambda_{n-1}}{\mu_1 ... \mu_n} = \sum_{n=0}^{\infty} \frac{1}{n!} (\frac{\lambda}{\mu})^n = e^{\lambda/\mu}$$

Hence, the chain is positive recurrent for all λ, μ and has equilibrium distribution

$$\pi(n) = e^{-\lambda/\mu} \frac{(\lambda/\mu)^n}{n!1}$$

2.2.1 Simulations

M/M/1 queue

Given function simulates a continuous time Markov chain for a queue and returns information about each customer's arrival and departure. Arguments:

a (float): Parameter for the exponential distribution of customer arrivals.
b (float): Parameter for the exponential distribution of customer departures.
simulation_time (float): The simulation time.
Returns:
tuple: A tuple containing two lists: arrival_times and departure_times.
arrival_times (list): List of arrival times for each customer.
departure_times (list): List of departure times for each customer.

```
1
2  def simulate_continuous_time_markov_chain_queue(a, b, simulation_time):
3
4      current_time = 0.0
5      num_customers = 0
6      arrival_times = []
7      departure_times = []
8
9      while current_time < simulation_time:
10         # Generate time until next arrival and departure
11         time_until_next_arrival = np.random.exponential(scale=1 / a)
12         time_until_next_departure = np.random.exponential(scale=1 / b)
13
14         if current_time + min(time_until_next_arrival,
       time_until_next_departure) > simulation_time:
15             break
16
17         if time_until_next_arrival < time_until_next_departure:
```

```
            num_customers += 1
            arrival_times.append(current_time)
            current_time += time_until_next_arrival
        else:
            if num_customers > 0:  # Check if there are customers in the queue
                num_customers -= 1
                departure_times.append(current_time)
            current_time += time_until_next_departure


    return arrival_times, departure_times
```

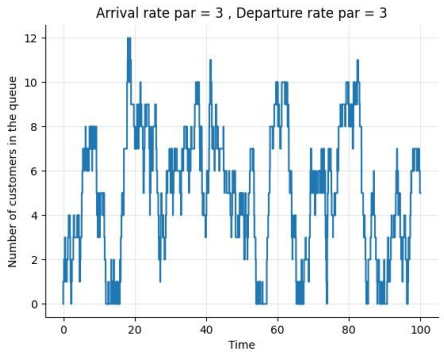
To analyse the results, let's create a function that converts the obtained results and plots the graphs. The function calculates the invariant distribution as well. Plots the behaviour of the queue over time.

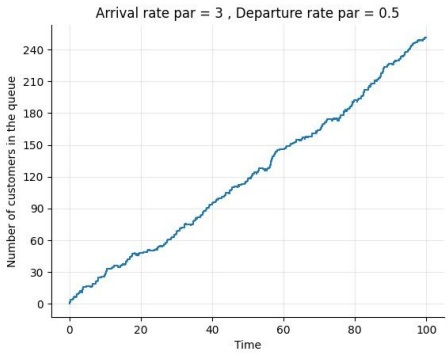
Arguments:
arrival_times (list): List of arrival times for each customer.
departure_times (list): List of departure times for each customer.
simulation_time (float): The simulation time.

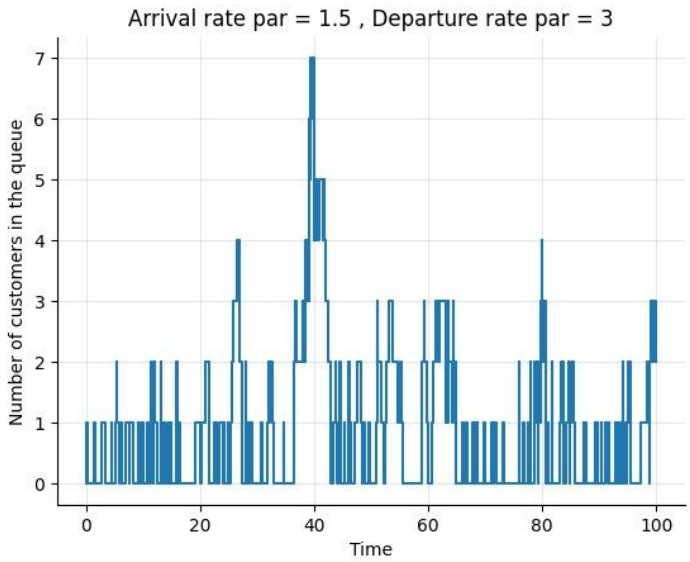
```


def plot_queue_behavior(arrival_times, departure_times, simulation_time):

    events = [(time, 'arrival') for time in arrival_times] + [(time, 'departure') for time in departure_times]
    events.sort(key=lambda x: x[0])

    time_points = [0]
    queue_sizes = [0]

    for event in events:
        time_points.append(event[0])

        if event[1] == 'arrival':
            queue_sizes.append(queue_sizes[-1] + 1)
        else:
            queue_sizes.append(max(queue_sizes[-1] - 1, 0))  # Ensure the queue size is never less than 0

    time_points.append(simulation_time)
    queue_sizes.append(queue_sizes[-1])

    plt.step(time_points, queue_sizes, where='post')
    plt.xlabel('Time')
    plt.ylabel('Number of customers in the queue')

    plt.title('Arrival rate par = '+ str(a) + ' , Departure rate par = ' + str(b))
    plt.gca().spines['top'].set_visible(False)
    plt.gca().spines['right'].set_visible(False)
    plt.grid(color='lightgray', alpha=0.5, zorder=1)
    a1 = plt.gca()
    a1.yaxis.set_major_locator(MaxNLocator(integer=True))
    plt.show()

    # Calculate the number of occurrences for each queue length
    queue_length_counts = Counter(queue_sizes)

    # Calculate the probabilities of each queue length
    total_occurrences = sum(queue_length_counts.values())
    queue_length_probabilities = {
        length: count / total_occurrences
        for length, count in queue_length_counts.items()
```

```
    }
    print("Queue Length Probabilities:")
    for length, probability in queue_length_probabilities.items():
        print(f"Length {length}: Probability {probability}")
    return queue_length_counts
```

These examples provide a clear demonstration of how the code functions according to the original parameters.







For a positive recurring Markov Chain the invariant distribution is
Queue Length Probabilities:

Length 0: Probability 0.2806451612903226
Length 1: Probability 0.4
Length 2: Probability 0.1774193548387097
Length 3: Probability 0.06774193548387097
Length 4: Probability 0.03225806451612903
Length 5: Probability 0.02258064516129032
Length 6: Probability 0.012903225806451613
Length 7: Probability 0.0064516129032258064

M/M/k queue This function is identical to the previous function which simulates a continuous time M/M/1 queue. In this model we have added a parameter k which indicates the number of servants.

Arguments:

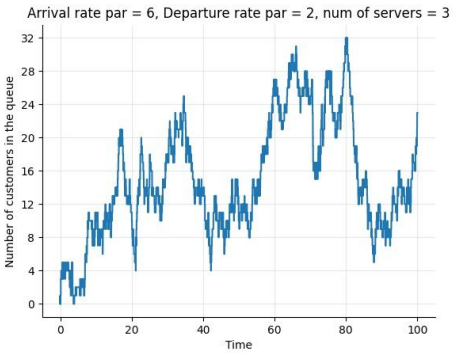
a (float): Parameter for the exponential distribution of customer arrivals.
b (float): Parameter for the exponential distribution of customer departures.
k (float): Parameter for the the number of servants.
simulation_time (float): The simulation time.

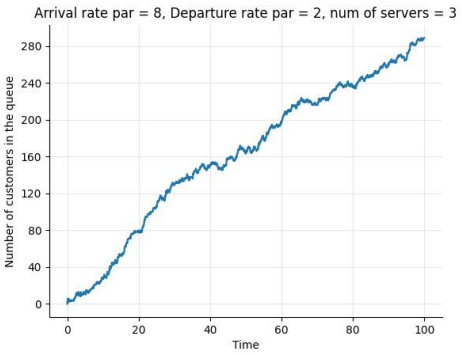
Returns:

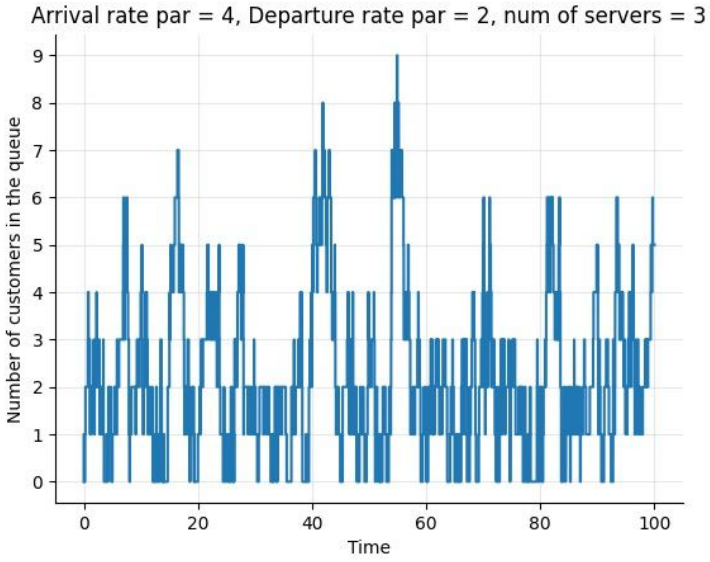
tuple: A tuple containing two lists: arrival_times and departure_times.
arrival_times (list): List of arrival times for each customer.
departure_times (list): List of departure times for each customer.

```
def simulate_continuous_time_markov_chain_queue(a, b, k, simulation_time):

    current_time = 0.0
    num_customers = 20
    arrival_times = []
    departure_times = []


    while current_time < simulation_time:
        if num_customers > 0: c = b * min(k, num_customers)
        # Generate time until the next arrival and departure

        time_until_next_arrival = np.random.exponential(1 / a)
        time_until_next_departure = np.random.exponential(1 / c)

        if current_time + min(time_until_next_arrival,
    time_until_next_departure) > simulation_time:
            break

        if time_until_next_arrival < time_until_next_departure:
            num_customers += 1
            arrival_times.append(current_time)
            current_time += time_until_next_arrival
        else:
            if num_customers > 0:  # Check if there are customers in the
    queue
                num_customers -= 1
                departure_times.append(current_time)
            current_time += time_until_next_departure

    return arrival_times, departure_times
```

The plotting function is identical to M/M/1 simulation code.






Arrival rate par = 4, Departure rate par = 2, num of servers = 3
Number of customers in the queue
Time
0
1
2
3
4
5
6
7
8
9
0
20
40
60
80
100

Chapter 3

Traffic modelling & Animations

The fast growing number of vehicles on highways or networks of roads and the related economical and social implications: pollution and energy control, prevention of car crashes, etc., constantly motivate an intense research activity in the field of traffic flow experiments, analysis and modelling.

In this paragraph, we proceed to model a single stream of cars. Using the Python programming language, we aim to develop a program that allows us to study the dynamics of car traffic. By manipulating various parameters, we aim to gain valuable insights into the complex behaviour of transport systems.

Traffic flow in certain scenarios can be approximated using a Poisson distribution due to the nature of arrival and departure events, especially when the arrivals are relatively rare and random. Let's break down why traffic flow in a simple scenario like a single-line traffic situation, modeled as an M/M/1 queue, follows a Poisson distribution.

The Poisson distribution comes into play due to the following key assumptions and characteristics of the system:

1. **Independent Arrivals:** The vehicles arriving at a traffic junction or point on the road are assumed to be independent events. This means that the arrival of one vehicle does not affect the likelihood of another vehicle arriving in the immediate future.

2. **Constant Arrival Rate:** The arrival rate of vehicles follows a constant rate over time, implying that the probability of a vehicle arriving in a short time interval is proportional to the length of that interval.

3. **Rare Events:** The Poisson process is used to model rare and random events that occur independently over time or space. When applied to traffic flow, this means that cars arrive in a manner that is unpredictable and relatively infrequent.

4. **Memoryless Property:** Both the exponential distribution (for inter-arrival times) and the exponential distribution (for service times) exhibit the memoryless property. This means that the time until the next event (arrival or departure) is not influenced by how much time has already elapsed.

Based on the M/M/1 queue code, the following function was written to simulate the single-line movement of cars. We can study the behaviour of queues by setting parameters:

- input parameter - parameter with which the car joins the flow
- movement parameter - the parameter with which the car moves to a free space forward, if there is a free space.
- exit parameter - the parameter with which the car leaves the flow

```
def simulate_traffic(particles, input_par, exit_par, movement_par):
    print(particles)  # Print current state
    next_particles = particles.copy()  # Create a copy of the particles
    list
    move_probs = particles.copy()  # Create another copy of the particles
    list to store movement probabilities

    # Calculate movement probabilities for each particle pair
    for i in range(len(move_probs) - 1):
        if particles[i] == 1 and particles[i + 1] == 0:
            move_probs[i] = np.random.exponential(1 / movement_par)
        else:
            move_probs[i] = 0

    # Calculate movement probability for the last particle
    if particles[len(move_probs) - 1] == 1:
        move_probs[len(move_probs) - 1] = np.random.exponential(1 /
    exit_par)

    # Calculate movement probability for the first particle
    if particles[0] == 0:
        move_probs[0] = np.random.exponential(1 / input_par)

    # Find the lowest movement probability that is greater than 0
    lowest_prob = min(n for n in move_probs if n > 0)

    # Update particle positions based on movement probabilities
    for i in range(len(move_probs) - 1):
        if move_probs[i] == lowest_prob and particles[i] == 1 and
    particles[i + 1] == 0:
            next_particles[i] = 0
            next_particles[i + 1] = 1

    # Update the last particle position if its movement probability is the
     lowest
    if move_probs[len(move_probs) - 1] == lowest_prob and particles[len(
    move_probs) - 1] == 1:
        next_particles[len(move_probs) - 1] = 0

    # Update the first particle position if its movement probability is
    the lowest
    if move_probs[0] == lowest_prob and particles[0] == 0:
        next_particles[0] = 1

    return next_particles, lowest_prob  # Return the updated particles and
     the lowest movement probability
```

This function simulates a basic traffic scenario in which the list of particles represents the positions of cars in a single-lane line. Each position can be occupied by a car (1) or empty (0). The function calculates traffic probabilities based on Poisson process conditions, updates the positions of the cars and returns the updated positions along with the lowest traffic probability.

To visually illustrate how traffic evolves based on initial parameters, an animation function has been created using the TKINTER package. This function collaborates with the SIMULATE TRAFFIC function, visually representing the movement of particles (cars) on screen. It uses the SIMULATE TRAFFIC output to showcase dynamic traffic changes at different time intervals. This combined approach provides both quantitative insights from the simulation and qualitative visual understanding, enhancing our grasp of traffic dynamics under varying conditions

```
def update_animation(next_particles, simulation_time, current_time):
    animation_speed = 1000  # milliseconds
    global canvas

    if len(next_particles) == 0:
        return  # Animation ends when no more particle lists are available

    # Clear the canvas
    canvas.delete("all")

    # Draw the particles
    particle_size = 20  # size of each particle
    x_offset = 50  # horizontal offset for the first particle
    y_offset = 50  # vertical offset for the particles
    input_par = 1  # input parameter
    exit_par = 1  # exit parameter
    movement_par = 1  # movement parameter

    next_particles, lowest_prob = simulate_traffic(next_particles,
    input_par, exit_par, movement_par)  # Get the next particle list
    current_time += lowest_prob
    print(current_time)
    for j, particle in enumerate(next_particles):
        color = "royalblue" if particle == 1 else "white"
        canvas.create_rectangle(
            x_offset, y_offset,
            x_offset + particle_size, y_offset + particle_size,
            fill=color
        )
        x_offset += particle_size + 10  # spacing between particles

    if current_time > simulation_time: return
    # Update the window
    canvas.update()

    # Schedule the next update
    canvas.after(round(animation_speed * lowest_prob), update_animation,
    next_particles, simulation_time, current_time)
```

Please find below an example of a traffic simulation implemented through written code. All of the input, output, and movement parameters equal to one. If a square is blue, it indicates that the spot is taken by a particle. On the other hand, if a square is white, it signifies that the area is available.

In order to view the animation in action, it is recommended to use Adobe Acrobat Reader when accessing this document.

The provided animation serves as a vivid illustration of the process of traffic jam formation. In this demonstration, we take into account three crucial parameters: input, movement, and output. Specifically, the input parameter is set at 15, the movement parameter is designated as 10, and the output parameter maintains a value of 1.

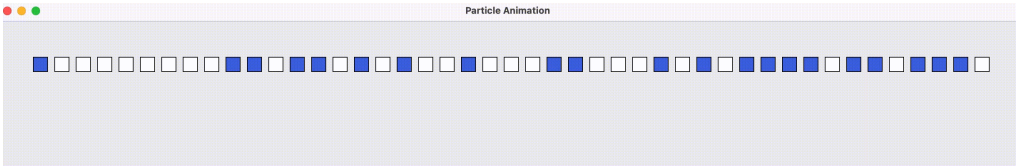


Figure 3.1: Animated Traffic Simulation

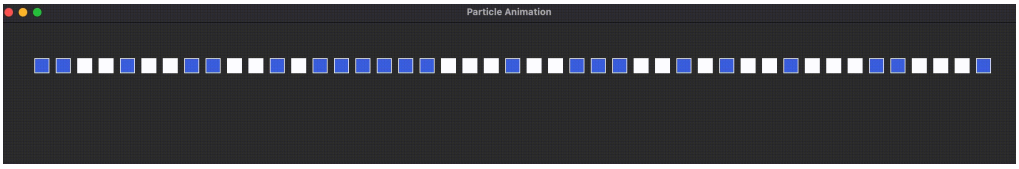


Figure 3.2: Animated Traffic Simulation: Traffic Jam

It is noteworthy that due to the substantial disparity between the output parameter and the input/movement parameters, a noticeable accumulation of particles transpires, rapidly giving rise to a traffic jam scenario. The animation has been intentionally accelerated to highlight this progression.

For a more comprehensive understanding of how different initial parameters can influence queue behaviour, I encourage you to employ the aforementioned functions. These functions provide the means to simulate traffic patterns with diverse initial parameter settings, showing the connection between these parameters and the queue dynamics.

3.1 Height Function

Within this section, we will introduce the Height function, which displays the quantity of particles that have successfully traversed the specified stages.

Tracking and plotting pass-stage information provide a quantitative and visual representation of a system's behavior and performance. This information is essential for understanding, optimizing, and effectively managing the system, whether it's a traffic flow simulation, a manufacturing process, or any other dynamic system.

```
def simulate_traffic(particles, num, input_par, exit_par, movement_par):
    particle_lists = []
    stage_pass = particles.copy()
    for l in range(100):
        particle_lists.append(particles)
        next_particles = particles.copy()
        move_probs = particles.copy()
        for i in range(len(move_probs) - 1):
            if particles[i] == 1 and particles[i + 1] == 0:
                move_probs[i] = np.random.exponential(1 / movement_par)
            else:
                move_probs[i] = 0
        if particles[len(move_probs) - 1] == 1:
            move_probs[len(move_probs) - 1] = np.random.exponential(1 /
    exit_par)

        if particles[0] == 0:
            move_probs[0] = np.random.exponential(1 / input_par)
        lowest_prob = min(n for n in move_probs if n > 0)

        for i in range(len(move_probs) - 1):
```

```
            if move_probs[i] == lowest_prob and particles[i] == 1 and
    particles[i + 1] == 0:
                next_particles[i] = 0
                next_particles[i + 1] = 1
                stage_pass[i] += 1

        if move_probs[len(move_probs) - 1] == lowest_prob and particles[
    len(move_probs) - 1] == 1:
            next_particles[len(move_probs) - 1] = 0
            stage_pass[len(move_probs) - 1] += 1
        if move_probs[0] == lowest_prob and particles[0] == 0:
            next_particles[0] = 1

        print(stage_pass)
        if particles == next_particles:
            break
        particles = next_particles
        l += 1

        fig = plt.figure(figsize=(10, 5))
        x = list(range(num))
        plt.xlabel('Stages')
        plt.ylabel('Pass Count')
        plt.title('Pass Count at Different Stages')
        plt.yticks(range(0, max(stage_pass) + 1, 1))
        plt.xticks(range(0, 50, 2))
        plt.tight_layout()  # Ensures labels/titles are not cut off
        bar_width = 0.8  # Adjust the bar width for better spacing
        bars = plt.bar(x, stage_pass, width=bar_width, align='center',
    alpha=0.7, color='skyblue')
        plt.grid(True, axis='y', linestyle='--', alpha=0.7)  # Add grid
    lines for y-axis
        plt.show()
        sample_file_name = "sample" + str(l)
        fig.savefig(results_dir + sample_file_name)

    return particle_lists, stage_pass
```

The function is based on the SIMULATE TRAFFIC function from the traffic section. We have incorporated a pass count feature to the SIMULATE TRAFFIC function. The simulation runs for a maximum of 100 (changeable) iterations or until the particle positions converge.

The function also includes a plotting section to visualize the pass counts at different stages during each iteration. The resulting plots are saved with filenames indicating the iteration number. The function returns a list of particle positions over time and the final stage pass counts.

The next packages were used:

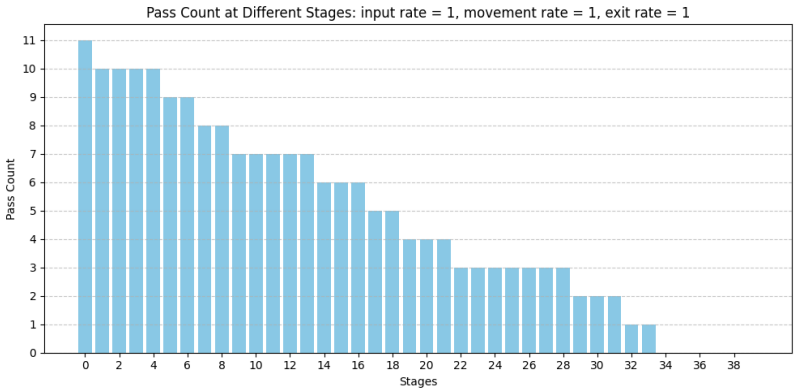
import numpy as np
import matplotlib.pyplot as plt
import os

The example of usage(main function):

```
# Example usage

num = 20  # number of particles
initial_particles = [0] * num  # initial empty list
# Parameters:
input_par = 1
exit_par = 1
movement_par = 1

# Find a directory to save the graphs
```

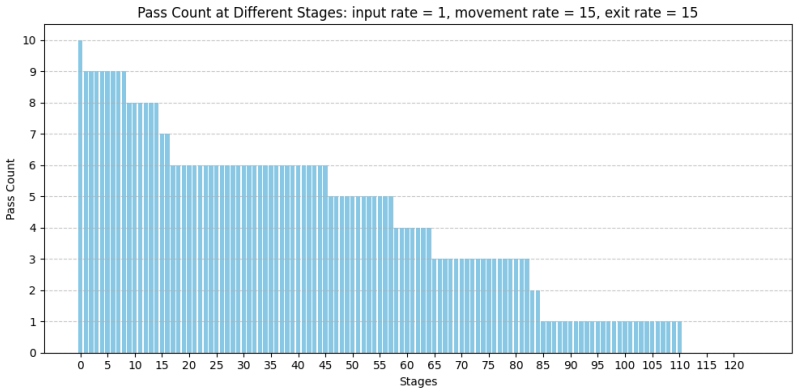
```
script_dir = os.path.dirname(__file__)
results_dir = os.path.join(script_dir, 'Results_function/')
if not os.path.isdir(results_dir):
    os.makedirs(results_dir)

# Run the function
simulate_traffic(initial_particles, num, input_par, exit_par, movement_par
    )
```

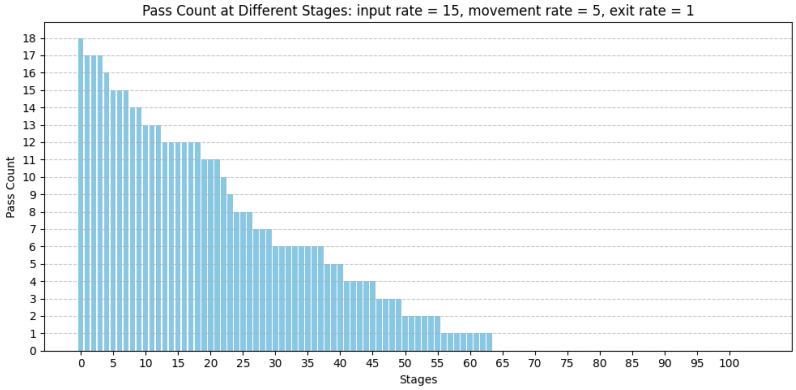
OUTPUT EXAMPLES:

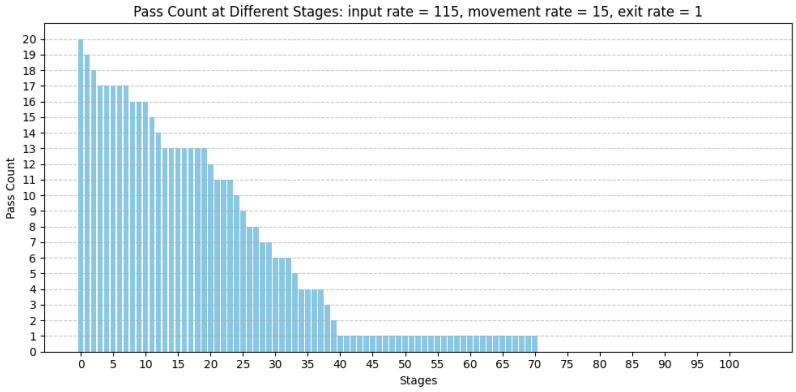


The graph displays the output of the program, which was initiated with parameters set equal to one. The parameters are identical to those used in the animation 3.2. The graph depicts a null recurrent flow of cars, moving evenly without causing any traffic congestion.

The following three graphs are examples of the height function for transient flows, which after a while create traffic jams.




Pass Count at Different Stages: input rate = 15, movement rate = 5, exit rate = 1
Pass Count
Stages


Pass Count at Different Stages: input rate = 115, movement rate = 15, exit rate = 1
Pass Count
Stages

References

[1] Hermann Rost. "Non-equilibrium behaviour of a many particle process: Density profile and local equilibria". In: *Zeitschrift für Wahrscheinlichkeitstheorie und Verwandte Gebiete* 58 (1981), pp. 41–53.

[2] Sidney I Resnick. *Adventures in stochastic processes*. Springer Science & Business Media, 1992.

[3] Sanjay K. Bose. "Advanced Queueing Theory". In: *An Introduction to Queueing Systems*. Boston, MA: Springer US, 2002, pp. 89–142.

[4] "Nonlinear hydrodynamic models of traffic flow in the presence of tollgates". In: *Mathematical and Computer Modelling* 35.5 (2002), pp. 549–559. ISSN: 0895-7177.

[5] "An ODE traffic network model". In: *Journal of Computational and Applied Mathematics* 203.2 (2007). Special Issue: The first Indo-German Conference on PDE, Scientific Computing and Optimization in Applications, pp. 419–436. ISSN: 0377-0427.

[6] Nicos Georgiou, Rohini Kumar, and Timo Seppalainen. "TASEP with discontinuous jump rates". In: *arXiv preprint arXiv:1003.3218* (2010).

[7] Saif Eddin Jabari and Henry X. Liu. "A stochastic model of traffic flow: Theoretical foundations". In: *Transportation Research Part B: Methodological* 46.1 (2012), pp. 156–174. ISSN: 0191-2615.

[8] Gregory F Lawler. *Introduction to stochastic processes*. CRC Press, 2018.

[9] Renátó Besenczi et al. "Large-scale simulation of traffic flow using Markov model". In: *PLOS ONE* (2021). DOI: 10.1371/journal.pone.0246062.

[10] E. Scalas J. Butt N. Georgiou. "Queuing models with Mittag-Leffler inter-event times". In: *Fract Calc Appl Anal 26, 1465–1503* (2023).