Interval Markov chains: Performance measures and sensitivity analysis

Mingmei Teo

Thesis submitted for the degree of

Master of Philosophy

in

Applied Mathematics

at

The University of Adelaide

School of Mathematical Sciences

December 2013

Contents

Abstract

There is a vast literature on Markov chains where point estimates of transition and initial probabilities are used to calculate various performance measures. However, using these point estimates does not account for the associated uncertainty in estimate. If these point estimates are used, then the best outcome possible would be an approximate solution. Hence, it would be beneficial if there was a way to allow for some uncertainty in the parameters and to carry this through the calculations.

One method of incorporating variation is to place bounds on the parameters and use these intervals rather than a single point estimate. By considering the intervals that contain point estimates, it is possible to control the amount of variation allowed. When these intervals are used in calculations, the results obtained are also intervals containing the true solution. Hence, allowing for an approximation of the result as well as a margin of error to be obtained.

One of the objectives of this thesis is to develop and investigate different methods of calculating intervals for various performance measures (for example, mean hitting times and expected total costs) for Markov chains when intervals are given for the parameters instead of point estimates. We develop a numerical method for obtaining intervals for the performance measures for general unstructured interval Markov chains through the use of optimisation techniques. During this development, we found a connection between interval Markov chains and Markov decision

processes and exploited it to obtain a form for our solution. Further, we also considered structured interval Markov chains, such as interval birth and death processes, and obtained analytic results for the classes of processes considered.

Following from the idea of structured Markov chains, we considered the Markovian SIR (*susceptible-infectious-recovered*) epidemic model and looked to extend the concepts developed for the unstructured interval Markov chains. Two important performance measures, namely the mean final epidemic size and mean epidemic duration, were of interest to us and we were able to prove analytic results for the mean final epidemic size. For the mean epidemic duration, we modified the numerical method for general unstructured interval Markov chains to calculate bounds on this performance measure.

The other objective of this thesis was to investigate if it was possible to use interval analysis as an alternative to sensitivity analysis. We explored this in the context of the SIR model, where the true value of the parameters of the model may not be known. Hence, if one were to be careful when using point estimates, one would consider using sensitivity analysis which explores the parameter space around the chosen estimates. We considered a distribution on the parameter estimates and used the methods developed in the early chapters of the thesis, to calculate intervals for performance measures. Using these intervals, we developed a method to obtain an approximate cumulative distribution function of the performance measure. This approximate cumulative distribution function was found to very closely resemble the cumulative distribution function obtained from extensive simulations.

Signed Statement

Acknowledgements

First, to my supervisors, Professor Nigel Bean and Dr Joshua Ross. Thank you for all your help, support and guidance throughout the past couple of years. I am immensely grateful for your patience in explaining concepts and ideas to me, whenever I knocked on your door, as well as the time and effort you have invested to make this project a thoroughly enjoyable experience. I really do not think I would have been able to achieve all of this without your help and for that, the both of you are the best supervisors one could have.

To everyone involved in the stochastic group, I thank you for your help during stochastic coffee. You have all contributed to my project in one way or another by allowing me to bounce ideas off you and providing helpful suggestions to solve my countless problems. In particular, I would like to thank Dr Giang Nguyen for her help with Markov decision processes. Without her help in identifying the connection with Markov decision processes, I think I would still be stuck with my proof.

I would like to thank everyone in the School of Mathematical Sciences for making these two years fun and enjoyable. There are undoubtably countless people to thank but here are a number I would like to thank specifically. To Kate, Jess, Vincent, Nic, Caitlin, Nicola and David A, thank you for all your help and also for the fun we have had the past couple of years. Further, I cannot forget the three people who have played a massive role over the past year in helping me get to this point. A big thank you to Ben, David and Hayden for their immense support and friendship, especially over the past six months, to enable me to not only finish my thesis but also have an amazing time doing so.

Finally, a big thank you to my parents for your unconditional love and support for the past 21 years of my life. Thank you for encouraging me to pursue my Masters and also being my chef, chauffeur and everything in between, especially over the last few months. This would all not have been possible without the both of you.

Chapter 1

Introduction

It is common in many applications to only know a best estimate of the parameters of a model. Hence, when we use these estimates in our models, there will inherently exist some error or uncertainty in the output. If we want to be careful when using estimates of parameters, then sensitivity analysis is a technique commonly used to account for the lack of exact knowledge of the parameter values. The idea of sensitivity analysis to explore the parameter space around the chosen parameter values which may involve a fair bit of computation. Instead, we look to account for the uncertainty in the parameters by placing bounds, or intervals, around the best estimates. Thus, one of the objectives of this project is to investigate if performing computation with intervals allows us to avoid the need for sensitivity analysis as we have already taken into account the uncertainty in the parameters through the use of intervals. The other main objective of this project is to develop the necessary techniques to obtain interval calculations of the performance measures of interest when provided with intervals around the parameters of our model.

Since Markov chains allow us to model processes which evolve randomly over time and are used in many areas, the overall underlying theme of this thesis is the incorporation of intervals in Markov chains and the development of methods to obtain bounds on performance measures such as the expected total cost or mean hitting time. Throughout the exploration and development of these concepts, we investigate the relationships between these models, optimisation problems and Markov decision processes. Furthermore, we exploit these relationships to obtain analytic results for our problem of obtaining bounds on the performance measures of interest.

In the next chapter, we provide background material on Markov chains, intervals and other concepts required in later chapters and also discuss previous work published in this area. We begin the discussion of our work in Chapters 3 and 4 where we focus on unstructured discrete-time and continuous-time interval Markov chains and consider calculating a corresponding interval of expected total costs. We explore useful theoretical properties as well as develop a complete numerical method for obtaining the interval of expected total costs.

Birth and death chains are commonly used in many applications. Hence, in Chapter 5, we consider incorporating intervals into these structured Markov chains and exploring ways to obtain results for the interval of expected total costs. The first type of birth and death chain follows from the discrete-time interval Markov chains in Chapter 3 as the rows of the transition probability matrix are independent. Then, we shift our attention away from the row independence of transition probability matrices to consider discrete-time constant parameter birth and death processes where the rows of the transition probability matrix are dependent. Hence, these processes no longer directly relate to the processes in Chapters 3 and 4.

The concept of row dependence in the transition probability matrix is explored further in Chapter 6. This chapter considers a specific application of our work on interval continuous-time Markov chains and extends it to allow for row dependence in epidemic models. One of the aims of this project is to determine if it is possible to use interval analysis to investigate model sensitivity, this is also discussed in Chapter 6. In Chapter 7, we discuss possible extensions of the work in this thesis.

Chapter 2

Background

In the previous chapter, we gave a brief introduction to Markov chains and intervals as well as providing reasons why we are interested in combining these two concepts. Here, we provide a more in-depth explanation of Markov chains, intervals and other background material necessary for easy reading of further chapters. We also present the previous work in this area and establish the notation used throughout this thesis.

2.1 Markov chains

Markov chains are an interesting type of mathematical model as they allow us to model processes which evolve randomly over time. Unlike some models of random processes, Markov chains are tractable and thus, they have been used in a wide variety of areas. These include the modelling of biological processes, telecommunication networks, traffic systems and the list goes on [16]. Hence, it is of no doubt that Markov chains are of great interest to us and here, we consider such processes evolving in either discrete time or continuous time.

2.1.1 Discrete-time Markov chains

Let S be a subset of the non-negative integers, $\mathbb{Z}_+ = \{0, 1, \ldots\}$. A discrete-time random process can be thought of as a set of random variables $X_m \in S$ at time points $m \in \mathbb{Z}_+$. We often describe X_m as the state of the process. Then, a discrete-time Markov chain is a discrete-time random process where the next state of the process depends only on the current state of the process and not on any past history. This describes the Markov property and is the key to tractability. Note that a sample path of a discrete-time Markov chain is an observed sequence of outcomes representing the state visited by the process at each time step.

The Markov property is stated more formally as

$$P(X_{m+1} = j | X_0 = x_0, X_1 = x_1, \ldots, X_m = i) = P(X_{m+1} = j | X_m = i),$$

for $i, j, x_k \in S$ and $k = 0, 1, \ldots, m-1$.

Due to the Markov property, we are able to focus our attention on $P(X_{m+1} = j | X_m = i)$ and note that in general it depends on m. However, this need not be the case and leads to the definition of *time-homogeneous* discrete-time Markov chains. The idea of *time homogeneity* is that the probability of moving from state i to state j in a single time step only depends on the states i and j and not on the actual time m. So this means that for a time-homogeneous discrete-time Markov chain, the probability of moving from state i to state j in a single time step is given by,

$$P_{ij} = P(X_{m+1} = j | X_m = i) = P(X_1 = j | X_0 = i), \qquad \text{for all } i, j \in S \text{ and } m \in \mathbb{Z}_+.$$

These probabilities P_{ij} are known as the *one-step transition probabilities* and together, they form a *one-step transition probability matrix* P. Note that the matrix P is a *stochastic matrix* which means that the rows of the matrix P sum to 1. This *one-step transition probability matrix* easily extends to m steps through the

Chapman-Kolmogorov equations [20] which are given by,

$$P_{ij}^{(m+\ell)} = \sum_{k \in S} P_{ik}^{(m)} \, P_{kj}^{(\ell)}, \qquad \text{for all } m, \ell \geq 0 \text{ and } i, j \in S,$$

where $P_{ij}^{(m)} = P(X_m = j | X_0 = i)$ is the probability of moving from state i to state j in m steps. These equations can be written in matrix form as follows,

$$P^{(m+\ell)} = P^{(m)} \, P^{(\ell)},$$

where $P^{(m)}$ is the m-step transition probability matrix.

Hence, by induction, we have that the probability of moving from state i to state j in m steps, $P_{ij}^{(m)} = P(X_m = j | X_0 = i)$, is given by $(P^m)_{ij}$. This gives us an easy way of calculating the probability of being in state j after a certain number of time steps m, given that the process started in state i, for all $i, j \in S$.

This also allows us to classify the states of a Markov chain. To do this, we present the following definitions which arise from Norris [16].

Definition 2.1.1. *For $i, j \in S$, if for some $m \geq 0$ we have $P_{ij}^{(m)} > 0$, then we say that i* leads to *j and denote this by $i \to j$.*

Definition 2.1.2. *States i and j* communicate *if $i \to j$ and $j \to i$ and we denote this by $i \leftrightarrow j$.*

Furthermore, we note that communication is an equivalence relation since:

- $i \leftrightarrow i$,
- $i \leftrightarrow j \Rightarrow j \leftrightarrow i$,
- $i \leftrightarrow j$, $j \leftrightarrow k \Rightarrow k \leftrightarrow i$.

Hence, $\leftrightarrow$ partitions the state space S into communicating classes, which contain states that are able to communicate with each other, but not with states outside of the communicating class. That is, if the process leaves the communicating class a and goes to another communicating class b, then there is no path from the communicating class b back to the original communicating class a.

Definition 2.1.3. *A Markov chain is* irreducible *if the entire state space S is a single communicating class.*

That is, a Markov chain is irreducible if every state in the state space S is able to communicate with each other.

We can pursue this idea further by considering closed classes.

Definition 2.1.4. *A class $C \subseteq S$ is* closed *if for any $i \in C$ and $i \to j$, implies that $j \in C$.*

This means that once the process enters any state in the closed class, the process cannot escape from this closed class.

Hence, we can define an absorbing state as follows.

Definition 2.1.5. *A state i is* absorbing *if $\{i\}$ is a closed class.*

This means that once the process enters state i it can never leave, and we say that the process is absorbed into state i.

A state i of a Markov chain can be further classified as either *recurrent*, which means that the process returns to state i with probability 1, or *transient*, which means the process returns to state i with probability less than 1. We note that transience/recurrence is a solidarity property, in the sense that all states in a communicating class are either transient or recurrent, together.

Now that we have established the required definitions for a discrete-time Markov chain, let us consider a discrete-time Markov chain, X_m, with $n + 1$ states, $S = \{0, 1, \ldots, n\}$ with state 0 an absorbing state. Note that this discrete-time Markov chain will be used for the remainder of this section. Since we have an absorbing state, it would be useful to determine the probability that the process is absorbed into the absorbing state 0, given that the process starts in state i, for $i \in S$. The presentation below follows from Norris [16].

Probability of absorption into state 0

Let the random variable $N : \Omega \to \{0, 1, 2, \ldots\} \cup \{\infty\}$ describe the *hitting time*, which is the number of steps required to reach state 0. That is,

$$N(\omega) = \inf\{m \geq 0 : X_m(\omega) = 0\}.$$

Note here that we use infimum instead of minimum because for realisations, ω, of the discrete-time Markov chain, such a value of m may not exist. Thus, for this case, we define the infimum of the empty set to be ∞.

Now, let a_i be the probability the process is ever absorbed into state 0, conditional on starting in state $i \in S$. That is,

$$a_i = P(N < \infty | X_0 = i), \qquad \text{for } i \in S.$$

The following theorem gives us a way of calculating the absorption probability a_i, for all $i \in S$.

Theorem 2.1.1 (Theorem 1.3.2. of Norris [16])**.** *The vector of absorption probabilities $(a_i : i \in S)$ is the minimal non-negative solution to the system of linear equations*

$$a_i = \begin{cases} 1, & \text{for } i = 0, \\ \displaystyle\sum_{j \in S} P_{ij} a_j, & \text{for } i \neq 0. \end{cases} \tag{2.1.1}$$

(Minimality means that if $\mathbf{x} = (x_i : i \in S)$ is another solution with $x_i \geq 0$ for all i, then $x_i \geq a_i$ for all i.)

Let us define the following,

- P_s is the sub-matrix of the transition probability matrix P containing the transition probabilities P_{ij} for $i, j = 1, \ldots, n$,
- $\mathbf{P}_0$ is the column vector containing the probabilities P_{i0} for $i = 1, \ldots, n$, and

- $\mathbf{a}$ is the column vector containing the absorption probabilities a_j for $j = 1, \ldots, n$.

Hence, we can write equation (2.1.1) in vector/matrix form as

$$\begin{aligned}\mathbf{a} &= \begin{bmatrix}\mathbf{P}_0 & P_s\end{bmatrix}\begin{bmatrix}1\\ \mathbf{a}\end{bmatrix}\\ &= P_s\,\mathbf{a} + \mathbf{P}_0.\end{aligned}$$

Here, we take a cautious approach to solving for $\mathbf{a}$.

$$\begin{aligned}\mathbf{a} &= P_s\,\mathbf{a} + \mathbf{P}_0\\ &= P_s\left(P_s\,\mathbf{a} + \mathbf{P}_0\right) + \mathbf{P}_0, && \text{by self substitution,}\\ &= P_s^2\,\mathbf{a} + P_s\,\mathbf{P}_0 + \mathbf{P}_0\\ &\;\vdots\\ &= P_s^\ell\,\mathbf{a} + \sum_{m=0}^{\ell-1} P_s^m\,\mathbf{P}_0, && \text{for some positive integer } \ell.\end{aligned}$$

Let $\ell \to \infty$ and choose the minimal non-negative solution, so that

$$\mathbf{a} = \sum_{m=0}^{\infty} P_s^m\,\mathbf{P}_0.$$

We will often write this as $\mathbf{a} = (I - P_s)^{-1}\,\mathbf{P}_0$, since if P_s is finite-dimensional and has spectral radius less than 1, then we know that $\sum_{m=0}^{\infty} P_s^m = (I - P_s)^{-1}$, where I is the identity matrix of the same size as P_s.

Mean hitting time

A common performance measure for discrete-time Markov chains is the mean hitting time. Recall that we let N be the random variable describing the number of steps required to reach state 0. Let $\nu_i = E\left[N|X_0 = i\right]$ be the expected number of steps needed to reach state 0, conditional on starting in state i, which is the mean hitting time from state i.

As with the absorption probabilities, we have the following theorem which allows us to calculate ν_i for $i \in S$.

Theorem 2.1.2 (Theorem 1.3.5. of Norris [16])**.** *The vector of mean hitting times* $(\nu_i : i \in S)$ *is the minimal non-negative solution to the system of linear equations*

$$\nu_i = \begin{cases} 0, & \textit{for } i = 0, \\ \sum_{j \neq 0} P_{ij}\nu_j + 1, & \textit{for } i \neq 0. \end{cases} \tag{2.1.2}$$

As before, minimal non-negative means if there is another solution to the system of linear equations with $x_i \geq 0$, then $x_i \geq \nu_i$ for $i \in S$.

Let us define the following,

- $\mathbf{1}$ is the column vector of size $n \times 1$ containing ones, and
- $\boldsymbol{\nu}$ is the column vector containing the mean hitting times ν_j for $j = 1, \dots, n$.

Hence, we can write equation (2.1.2) in vector/matrix form as

$$\boldsymbol{\nu} = P_s\,\boldsymbol{\nu} + \mathbf{1}.$$

Following the same approach as for $\boldsymbol{a}$, we have

$$\begin{aligned} \boldsymbol{\nu} &= P_s\,\boldsymbol{\nu} + \mathbf{1} \\ &= P_s\left(P_s\,\boldsymbol{\nu} + \mathbf{1}\right) + \mathbf{1}, && \text{by self substitution,} \\ &= P_s^2\,\boldsymbol{\nu} + P_s\,\mathbf{1} + \mathbf{1} \\ &\;\vdots \\ &= P_s^{\ell}\,\boldsymbol{\nu} + \sum_{m=0}^{\ell-1} P_s^m\,\mathbf{1}, && \text{for some positive integer } \ell. \end{aligned}$$

Let $\ell \to \infty$ and choose the minimal non-negative solution, so that

$$\boldsymbol{\nu} = \sum_{m=0}^{\infty} P_s^m\,\mathbf{1}.$$

As before, we will often write this as $\boldsymbol{\nu} = (I - P_s)^{-1}\mathbf{1}$, since if P_s is finite-dimensional and has spectral radius less than 1, then we know that $\sum_{m=0}^{\infty} P_s^m = (I - P_s)^{-1}$, where I is the identity matrix of the same size as P_s.

Expected total costs

Now, let us extend this idea further. Instead of calculating the mean hitting times, let us calculate the expected total costs instead. We define the following, let

- $c_j \in \mathbb{R}$ be the cost per visit to state j, for $j \in S$,
- K be the random variable describing the total cost incurred before the process is absorbed into state 0, and
- $\chi_i = E[K|X_0 = i]$ be the expected total cost incurred before the process is absorbed into state 0, conditional on starting in state $i \in S$.

The calculation of expected total costs is a generalisation of the mean hitting times whereby the latter corresponds to $c_j = 1$ for $j = 1, \ldots, n$. Therefore, we replace the vector of ones, $\mathbf{1}$, with a vector of costs, $\mathbf{c}$. Hence, we can calculate the expected total costs vector $\boldsymbol{\chi}$ using,

$$\boldsymbol{\chi} = \sum_{m=0}^{\infty} P_s^m \mathbf{c}.$$

Note $\boldsymbol{\chi} = (I - P_s)^{-1}\mathbf{c}$, if P_s is finite-dimensional and has spectral radius less than 1.

2.1.2 Continuous-time Markov chains

A continuous-time Markov chain is similar in nature to a discrete-time Markov chain, except it evolves at all $t \geq 0$ rather than at integer time points $m = 0, 1, \ldots$. Let S be a subset of the non-negative integers, $\mathbb{Z}_+ = \{0, 1, \ldots\}$. A continuous-time random process can be thought of as a set of random variables $X(t) \in S$, for all time $t \geq 0$. Then a continuous-time Markov chain is a continuous-time random process satisfying the Markov property. Recall from Section 2.1.1 that the Markov property states that the probability that the process moves to a given state does not depend on the previous states visited by the process but only on the current state of the process. This is stated more formally in continuous time as

$$P(X(t+s) = j | X(s) = i, X(u) = k, u < s) = P(X(t+s) = j | X(s) = i),$$

for $i, j, k \in S$ and $s, t \geq 0$.

Hence, the sample behaviour of a continuous-time Markov chain is that after some random time t the process jumps to another state. This time t is known as the *holding time* and once this time ends, the process instantaneously jumps to another state j of the system with some probability that we shall present later.

As with discrete-time Markov chains, we will only consider *time-homogeneous* continuous-time Markov chains. For a time-homogeneous continuous-time Markov chain, the probability of being in state j at time $t + s$, given the chain was in state i at time s is given by,

$$P_{ij}(t) = P(X(t+s) = j | X(s) = i), \qquad \text{for } i, j \in S \text{ and } s, t \geq 0,$$

and this probability is independent of the actual time s. If we consider $P_{ij}(t)$ for all $i, j \in S$, these probabilities form the matrix, $P(t)$.

Here, we note that all definitions which allow for the classification of the states in a discrete-time Markov chain, as specified in Section 2.1.1, carry over analogously for continuous-time Markov chains.

Knowing $P(t)$, for all $t \geq 0$, is analogous to knowing P^m, for all m, in the discrete time case. In the latter case, it was sufficient to record P, from which P^m can be constructed. So is there a compact form which is analogous to P in the discrete time case? For continuous-time Markov chains, the *generator matrix* Q plays this role.

Generator matrix Q

The definition and properties of Q follow from Ross [20] and Norris [16] respectively.

The generator matrix $Q = (q_{ij} : i, j \in S)$ is a matrix defined by

$$Q = \lim_{h \to 0^+} \frac{P(h) - I}{h},$$

where I is the identity matrix of the same size as $P(h)$. Hence Q has the following properties,

- $0 \leq -q_{ii}$, for all $i \in S$,
- $q_{ij} \geq 0$, for all $i \in S$ and $j \neq i$, and
- $\sum_{j \in S} q_{ij} = 0$, for all $i \in S$.

This Q matrix is used to specify the *rates* of a continuous-time Markov chain, where q_{ij}, for $j \neq i$, is the rate of moving from state i to state j. Let us consider a finite state continuous-time Markov chain, $X(t)$, with a generator matrix Q and $n+1$ states, $S = \{0, 1, \ldots, n\}$ where state 0 is an absorbing state. It is common to let $q_i = -q_{ii} = \sum_{j \neq i} q_{ij}$, which is the rate of leaving state i. Since S is finite, we know that q_i also finite for all states $i \in S$.

Recall that the *holding time* is the random time t the process spends in a state before moving to another state. To make this idea more concrete, we define the holding time for state i, t_i, to be the random time the process spends in state i

which has an exponential distribution with mean

$$E[t_i] = \begin{cases} \dfrac{1}{q_i}, & \text{if } q_i > 0, \\ \infty, & \text{if } q_i = 0. \end{cases}$$

Recall that we have previously used the Chapman-Kolmogorov equations for discrete-time Markov chains in Section 2.1.1. A similar version of the Chapman-Kolmogorov equations apply in continuous time.

Theorem 2.1.3 (Lemma 6.3 of Ross [20]). *For all $s \geq 0$, $t \geq 0$,*

$$P_{ij}(t+s) = \sum_{k \in S} P_{ik}(t) P_{kj}(s).$$

The Chapman-Kolmogorov equations are useful as they help us derive the Kolmogorov Forward and Backward differential equations, which are given in matrix form as

$$\begin{aligned} \frac{dP(t)}{dt} &= P(t)Q, \qquad \text{(forward)} \\ \frac{dP(t)}{dt} &= QP(t), \qquad \text{(backward).} \end{aligned}$$

Thus, when we are given the transition rates Q, we can solve either of the above differential equations to determine the transition probability matrix $P(t)$, where

$$P(t) = e^{Qt},$$

when S is finite, as assumed herein.

As before with discrete-time Markov chains, we are interested in the probability of absorption into state 0, the mean hitting times and the expected total cost of continuous-time Markov chains. In Chapter 3, instead of working with these concepts in continuous time, we choose to exploit our work on discrete-time Markov chains in the early parts of the chapter. To do so, we utilise the concept of *uniformisation* as a way of converting a continuous-time Markov chain to a discrete-time Markov chain without any loss of information.

Uniformisation

Let us first present some definitions and explanations, both arising from Ross [20], which discuss how uniformisation allows us to convert a continuous-time Markov chain to a discrete-time Markov chain without losing any information.

Definition 2.1.6 (Uniformisable)**.** *A continuous-time Markov chain, $X(t)$, is* uniformisable *if*

$$\sup_j (q_j) < \infty.$$

Consider a uniformisable continuous-time Markov chain, $X(t)$, with generator matrix Q.

Definition 2.1.7 (Uniformisation constant)**.** *The* uniformisation constant *is any*

$$u \geq \overline{q} = \sup_i (q_i)\,.$$

Note here that since S is finite, we have $q_i < \infty$ for all $i \in S$ and so $u \geq \overline{q} = \max_i (q_i)$ for all $i \in S$.

Any such u allows us to define an associated discrete-time Markov chain of the continuous-time Markov chain, $X(t)$.

Definition 2.1.8 (Associated discrete-time Markov chain)**.** *Let X_m^u be the associated discrete-time Markov chain of the continuous-time Markov chain, $X(t)$, with transition probability matrix P_u, given by*

$$P_u = I + \frac{1}{u}Q,$$

for any $u \geq \overline{q}$.

So how does this transformation work and allow the associated discrete-time Markov chain to retain all the information from the continuous-time Markov chain? We know that $P_{ij}(t)$ completely describes the dynamics of the continuous-time Markov chain and is defined to be the probability of being in state j at time $t + s$, given the chain was in state i at time s. Unlike $P_{ij}^{(m)}$ for discrete-time Markov chains, which are the m-step transition probabilities, in the continuous time setting, we do not know how many transitions have occurred in the time interval t. The process might have been through any number of transitions in the time interval and we can model the number of transitions within the time interval t using a Poisson process with rate u. However, recall that $u \geq \overline{q} = \max_i(q_i)$ and q_i is rate of leaving state i. This means that by using u, the rate of leaving each state i has actually increased. So how does this transformation work? We introduce fictitious transitions into the system which allow the process to remain in state i. We know that the probability of moving from state i to state j, for all $j \in S$ and $j \neq i$, is given by $\frac{q_{ij}}{u}$. Hence, in the transformation which gives us P_u, the actual transitions (leaving state i) occur with probability $\frac{q_i}{u}$ and the fictitious transitions (leaving and then immediately returning to state i) occur with probability $\left(1 - \frac{q_i}{u}\right)$.

Recall that $P(t) = e^{Qt}$ and if we apply the transformation $P_u = I + \frac{1}{u}Q$, then we have

$$\begin{aligned} P(t) &= e^{Qt} \\ &= e^{u(P_u - I)t}, \qquad \text{since } P_u = I + \frac{1}{u}Q, \\ &= e^{-ut}e^{P_u ut} \\ &= e^{-ut}\sum_{m=0}^{\infty}\frac{(P_u ut)^m}{m!} \\ &= \sum_{m=0}^{\infty}(P_u)^m e^{-ut}\frac{(ut)^m}{m!}. \end{aligned}$$

This last expression can be interpreted as follows. Consider the $(i, j)^{th}$ element of $P(t)$ which is given by,

$$P_{ij}(t) = \sum_{m=0}^{\infty}(P_u)_{ij}^m e^{-ut}\frac{(ut)^m}{m!}, \qquad t \geq 0,$$

where $(P_u)_{ij}^m$ is the probability of moving from state i to state j in m steps in the associated discrete-time Markov chain and $e^{-ut}\frac{(ut)^m}{m!}$ describes the probability of having m events in a Poisson process with rate u over an interval of length t. Since we do not know how many transitions might occur in the time interval t, we sum over all possible values of m. Thus, we have not lost any information when converting from a continuous-time Markov chain to the associated discrete-time Markov chain.

Now recall that, as in the discrete time case, we are interested in the probability of absorption into state 0, the mean hitting times and the expected total costs of continuous-time Markov chains. Since we are dealing with these performance measures in discrete-time through the use of uniformisation, we simply state the relevant theorems and definitions continuous-time Markov chains which arise from Norris [16]. Note that for the remainder of this section, we consider a uniformisable continuous-time Markov chain, $X(t)$, with generator matrix Q and $n+1$ states, $S = \{0, 1, \dots, n\}$ where state 0 is an absorbing state, unless otherwise stated.

Definition 2.1.9 (Hitting time). *The* hitting time *of state 0 is the random variable T defined by*

$$T(\omega) = \inf\{t \geq 0 : X(t, \omega) = 0\}$$

with the usual convention that the infimum of the empty set is ∞.

Thus, we have that the probability of absorption into state 0, conditional on starting in state $i \in S$, is given by,

$$a_i = P(T < \infty | X(0) = i), \qquad \text{for } i \in S.$$

The following theorem gives us a way to calculate a_i for $i \in S$.

Theorem 2.1.4 (Theorem 3.3.1. of Norris [16]). *The vector of absorption probabilities $(a_i : i \in S)$ is the minimal non-negative solution to the system of linear equations*

$$\begin{cases} a_i = 1, & \text{for } i = 0, \\ \displaystyle\sum_{j \in S} q_{ij} a_j = 0, & \text{for } i \neq 0. \end{cases}$$

Recall that we are interested is the mean hitting time. Let $\tau_i = E\left[T|X(0) = i\right]$ be the expected time to hit state 0, conditional on starting in state i, which is the mean hitting time from state i in the continuous time setting. The following theorem allows us to obtain values for τ_i.

Theorem 2.1.5 (Theorem 3.3.3. of Norris [16])**.** *Assume that $q_i > 0$ for all $i \neq 0$. The vector of mean hitting times $(\tau_i : i \in S)$ is the minimal non-negative solution to the system of linear equations*

$$\begin{cases} \tau_i = 0, & \textit{for } i = 0, \\ \sum_{j\in S} q_{ij}\tau_j = -1, & \textit{for } i \neq 0. \end{cases}$$

Another performance measure of interest to us is the expected total cost. Let

- c_j be the cost per unit time of being in state j, for $j \in S$,
- K be the random variable describing the total cost incurred before the process is absorbed into state 0, and
- $\chi_i = E[K|X(0) = i]$ be the expected total cost incurred before the process is absorbed into state 0, conditional on starting in state $i \in S$.

The following theorem gives us a way to calculate the vector of expected total costs.

Theorem 2.1.6. *The vector of expected total costs is the minimal non-negative solution to the system of linear equations*

$$\begin{cases} \chi_i = 0, & \textit{for } i = 0, \\ \sum_{j\in S} q_{ij}\chi_j = -c_i, & \textit{for } i \neq 0. \end{cases}$$

Note that this theorem is based on Theorem 4.2.4. of Norris [16] where we consider uniformisable continuous-time Markov chains of the form specified earlier for this section.

As will be required later in this thesis (Chapter 6), we note that the expected total cost χ_i, for $i \neq 0$, can be written in matrix form, since $\chi_0 = 0$, as

$$Q_s\boldsymbol{\chi} = -\mathbf{c},$$

where

- Q_s is the sub-matrix of the generator matrix Q containing the rates q_{ij} for $i, j = 1, \ldots, n$,
- $\mathbf{c}$ is the column vector containing the costs c_j for $j = 1, \ldots, n$, and
- $\boldsymbol{\chi}$ is the column vector containing the expected total costs χ_j for $j = 1, \ldots, n$.

Hence, the expected total costs vector $\boldsymbol{\chi}$ is easily obtained by

$$\boldsymbol{\chi} = -Q_s^{-1}\mathbf{c},$$

provided Q_s is finite-dimensional and has spectral radius less than zero.

Since the mean hitting times is the expected total cost where $\mathbf{c}$ is a vector of ones, the mean hitting times vector $\boldsymbol{\tau}$ is easily obtained by

$$\boldsymbol{\tau} = -Q_s^{-1}\mathbf{1},$$

again provided Q_s is finite-dimensional and has spectral radius less than zero.

Now that we have considered the background required for Markov chains, let us discuss the background for the other major concept, intervals and interval arithmetic.

2.2 Intervals

There are many situations where we round off answers or give approximate solutions to equations that we solve. Should we accept these approximate solutions or would it be better if we could provide bounds in which the exact solution lies? If we can provide bounds on the solution, then we can be certain that we have captured the true solution up to a stated precision. There are also situations when an experiment is performed and the experimental values may only be known to a certain degree of accuracy. If we use these experimental results in further calculations, we would only be able to obtain approximate answers. However, if we could bound the true value and use these bounds in our calculation, we can then be certain that the true solution lies within its bounds. Using this method, not only are we able to give an approximation to the true solution, but we can also specify the margin of error for the approximation.

Based on this idea of bounding, it is of interest to consider the use of intervals. Interval analysis [14] allows us to analyse and perform calculations on intervals which contain the true solution. There are a variety of intervals that exist in mathematics but we focus primarily on closed intervals. Hence, whenever we mention *intervals*, we mean *closed intervals*. We present the following definitions and examples in this Section 2.2 which arise from Moore *et al.* [14].

An interval $[a, b]$ can be represented in set notation as follows,

$$[a, b] = \{x \in \mathbb{R} : a \leq x \leq b\}.$$

Following the notation used in [14], an interval and its endpoints are denoted by capital letters. So the interval X is represented as $X = [\underline{X}, \overline{X}]$, where $\underline{X}$ is the lower bound and $\overline{X}$ is the upper bound of the interval.

One of the things we can do with intervals is to perform basic arithmetic operations on intervals such as addition, subtraction, multiplication and division. Since interval representation is interchangeable with set notation representation, all arithmetic operations on the intervals X and Y can be represented in set notation as

$$X \odot Y = \{x \odot y : x \in X \ , \ y \in Y\},$$

where $\odot$ represents the arithmetic operator and we assume for division that $0 \notin Y$.

Although this specifies the set we are interested in, it is not a convenient way of obtaining an interval. An easier method of performing arithmetic operations is to use the endpoints of the intervals X and Y. We detail these methods below.

- Addition

$$X + Y = \left[\underline{X} + \underline{Y}, \overline{X} + \overline{Y}\right],$$

- Subtraction

$$X - Y = \left[\underline{X} - \overline{Y}, \overline{X} - \underline{Y}\right],$$

- Multiplication

$$X \cdot Y = [\min S, \max S], \quad \text{where } S = \left\{\underline{X}\,\underline{Y}, \underline{X}\overline{Y}, \overline{X}\underline{Y}, \overline{X}\,\overline{Y}\right\},$$

- Division

$$X/Y = X \cdot (1/Y), \quad \text{where } 1/Y = \{y : 1/y \in Y\backslash\{0\}\} = \left[1/\overline{Y}, 1/\underline{Y}\right].$$

We can also extend intervals to vectors and matrices. An interval vector is a vector whose elements are intervals. For example, the 1×2 interval vector $\mathbf{X}$ is expressed as follows

$$\mathbf{X} = (X_1, X_2) = \left(\left[\underline{X}_1, \overline{X}_1\right], \left[\underline{X}_2, \overline{X}_2\right]\right).$$

Similarly, an interval matrix A is a matrix whose elements are intervals and can be expressed, for example, as

$$A = \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix}$$

where $A_{ij} = \left[\underline{A}_{ij}, \overline{A}_{ij}\right]$ for $i, j = 1, 2$.

An interesting phenomenon that occurs in interval analysis is *interval dependency* which affects the bounds of the interval we are calculating. Consider the following example.

We define $X^2 = \{x^2 : x \in X\}$ which can be expressed in interval notation as

$$X^2 = \begin{cases} \left[\underline{X}^2, \overline{X}^2\right], & 0 \leq \underline{X} \leq \overline{X}, \\ \left[\overline{X}^2, \underline{X}^2\right], & \underline{X} \leq \overline{X} \leq 0, \\ \left[0, \max\left\{\underline{X}^2, \overline{X}^2\right\}\right], & \underline{X} < 0 < \overline{X}. \end{cases}$$

In normal arithmetic, we know that $x^2 = x \cdot x$. However, when dealing with interval arithmetic, due to *interval dependency*, $X^2 \neq X \cdot X$.

For example, consider $X = [-1, 1]$. This gives

$$X^2 = [-1, 1]^2 = [0, 1] \qquad \text{whilst} \qquad X \cdot X = [-1, 1] \cdot [-1, 1] = [-1, 1].$$

Here, we note that $X^2 \subseteq X \cdot X$. The reason for this is due to the fact that when we calculate the interval $X \cdot X$, we are able to choose two different x values from the interval, but when we calculate X^2, we use a single value of x from the interval. This is clearly illustrated in set notation, since

$$X^2 = \{x^2 : x \in X\} \qquad \text{whilst} \qquad X \cdot X = \{x_1 \cdot x_2 : x_1 \in X, x_2 \in X\}.$$

Hence, care must be taken when dealing with intervals.

Interval arithmetic is not restricted to the basic arithmetic operations, instead, functions of intervals can also be considered. We have already encountered an example of this when considering X^2. It would be ideal if we could obtain exact bounds for any interval function. That is, we would like to be able to compute the set of function values when we vary x over its interval X. If we have a function

f of a single variable x, then the set obtained when we apply the formula $f(x)$ to $x \in X$, for some interval X, is given by

$$f(X) = \{f(x) : x \in X\}.$$

This is known as the *image set.*

Two interesting and key ideas when dealing with interval functions are *interval extension* and *united extension.* The following definitions have been adapted from Moore *et al.* [14] to make explicit interval notation.

Definition 2.2.1. *F is an* interval extension *of f on X, if for all degenerate interval arguments $[x, x] \subseteq X$, F agrees with f:*

$$F([x, x]) = [f(x), f(x)], \qquad \textit{for } x \in X.$$

Definition 2.2.2. *If we have an interval extension of f, obtained by applying the formula directly using intervals, which yields the desired set image, then this extension is termed the* united extension *of f. That is, if $F(X)$ is an interval extension of $f(x)$ and $F(X) = f(X)$, then $F(X)$ is the* united extension *of f.*

For example, if we consider the following functions

$$f(x) = x(1 - x) \quad \text{and} \quad g(x) = \tfrac{1}{4} - \left(x - \tfrac{1}{2}\right)^2, \quad x \in [0, 1].$$

As the x values range between 0 and 1, the values of $f(x)$ and $g(x)$ range from 0 to $\frac{1}{4}$. Hence, we have

$$f([0, 1]) = g([0, 1]) = [0, \tfrac{1}{4}].$$

Now if we form interval extensions of f and g, we obtain

$$F(X) = X \cdot (1 - X), \quad X = \left[\underline{X}, \overline{X}\right],$$

and

$$G(X) = \tfrac{1}{4} - \left(X - \tfrac{1}{2}\right)^2, \quad X = \left[\underline{X}, \overline{X}\right].$$

Thus, giving us $F([0,1]) = [0,1]$ and $G([0,1]) = \left[0, \frac{1}{4}\right]$. Both are interval extensions of their respective functions, but $F(X) \neq f(X)$ whilst $G(X) = g(X) = f(X)$. So we have found the united extension of the function g. We also know that f is equivalent to g in ordinary arithmetic. Hence, we have found the united extension of f as well.

This is just one of many cases where applying the formula directly to intervals does not result in the desired set image. Another thing of note, as shown in the example above, is that there are many cases where although two expressions may be equivalent in ordinary arithmetic, when we extend to intervals, these expressions may no longer be equivalent. Thus, re-emphasising the need to be careful when dealing with intervals. Hence, we define *minimal intervals* to be intervals which contain the true solution and have the smallest width.

Interval arithmetic and analysis is not restricted to analytic calculations by hand. There are software packages that implement interval arithmetic. One such package is INTLAB [21] which is used within the MATLAB environment. Like other available packages, INTLAB rounds the answer outwardly before displaying results. This means that the final displayed answer (given to a fixed number of decimal places) will contain the true solution.

This gives us a basic understanding of interval analysis and we use the above concepts and ideas as we progress.

2.3 Incorporating intervals into Markov chains

In Sections 2.1 and 2.2, we discussed the important ideas of intervals and Markov chains separately. Now, we look to combine intervals with Markov chains. We focus on discrete-time Markov chains and note that the incorporation of intervals follows in a similar way for continuous-time Markov chains.

Consider a discrete-time Markov chain, X_m, with $n+1$ states, $S = \{0, 1, \ldots, n\}$ with state 0 an absorbing state. Recall that for a general discrete-time Markov chain, the transition probability of moving from state i to state j in a single time step is given by $P(X_{m+1} = j|X_m = i)$ and depends on the time step m. These probabilities form the transition probability matrix ${}_mP$. To incorporate intervals into a discrete-time Markov chain, we can specify an interval of probabilities for the transition probability matrix, ${}_mP$. These discrete-time Markov chains are now known as *discrete-time interval Markov chains* and they have an interval transition probability matrix denoted by ${}_m\mathbb{P}$ which is represented as follows,

$$
{}_m\mathbb{P} = \begin{bmatrix}
\left[{}_m\underline{P}_{00}, {}_m\overline{P}_{00}\right] & \left[{}_m\underline{P}_{01}, {}_m\overline{P}_{01}\right] & \cdots & \left[{}_m\underline{P}_{0n}, {}_m\overline{P}_{0n}\right] \\
\left[{}_m\underline{P}_{10}, {}_m\overline{P}_{10}\right] & \left[{}_m\underline{P}_{11}, {}_m\overline{P}_{11}\right] & \cdots & \left[{}_m\underline{P}_{1n}, {}_m\overline{P}_{1n}\right] \\
\vdots & \vdots & \ddots & \vdots \\
\left[{}_m\underline{P}_{n0}, {}_m\overline{P}_{n0}\right] & \left[{}_m\underline{P}_{n1}, {}_m\overline{P}_{n1}\right] & \cdots & \left[{}_m\underline{P}_{nn}, {}_m\overline{P}_{nn}\right]
\end{bmatrix}.
$$

This appears to be a simple replacement of point estimates with intervals in the transition probability matrix. However, there exist some intricacies in this replacement as we are dealing with Markov chains. Thus, there are a number of ways we can incorporate intervals into the transition probability matrix. The choice we make depends on the idea of preserving the assumption of time homogeneity which can be blurred when we deal with interval probabilities.

Recall from Section 2.1 that a time-homogenous Markov chain has the following property

$$
P_{ij} = P(X_{m+1} = j|X_m = i) = P(X_1 = j|X_0 = i), \qquad \text{for all } i, j \in S \text{ and } m \in \mathbb{Z}_+.
$$

This tells us that the probability of moving from state i to state j in a single time step only depends on the states i and j and not on the actual time m. Hence, the one-step transition probability matrix, P, is constant over time.

This is where there lies some ambiguity when incorporating intervals to Markov chains and because of this, we have created our own terminology to describe time homogeneity and interval probabilities. We decided on the following terminology for interval transition probability matrices where we make a distinction between a time-homogeneous Markov chain and a time-homogeneous interval matrix.

Time-homogeneous interval matrix: The same interval matrix is used at each time step. This is a simple extension of a time-homogeneous Markov chain to the intervals where we replace point estimates with intervals. Under this class, we have two subclasses.

> **One-sample (Time-homogeneous Markov chain):** The same value initially chosen from the interval matrix is used at each time step. This is a direct correspondence with a time-homogeneous Markov chain where the one-step transition probability matrix is constant over time.
>
> **Re-sampled (Time-inhomogeneous Markov chain):** A different value from the interval matrix can be chosen at each time step. This is different from the standard time-homogenous Markov chain as the one-step transition probability matrix is not constant over time, but the intervals that contain the values are constant over time.

Time-inhomogeneous interval matrix: A different interval matrix is used at each time step. This is an extension of a time-inhomogeneous Markov chain and was explored by Hartfiel [8] and Hartfiel and Seneta [9]. The model considered is known as *Markov set-chains.*

If time homogeneity is a property of the underlying stochastic system, then we would like to preserve this property. The idea of holding the intervals in a transition probability matrix constant but allowing for variation in the choice of the elements within the intervals was explored by Škulj [23, 22]. Although this upholds the idea of time homogeneity for interval matrices, it does not have a direct link with the time-homogeneous Markov chain that is widely described in the literature of Markov chains. Instead, we are more interested in the one-sample time-homogeneous interval matrix, or time-homogeneous Markov chain, which was explored by Kozine and Utkin [12], Campos *et al.* [3], Crossman *et al.* [5, 6] and Blanc and den Hertog [2].

2.4 Literature review

In this section, we explain what other researchers have managed to accomplish through their use of intervals when dealing with Markov chains. As mentioned previously, we are interested in the one-sample time-homogeneous interval matrix. Hence, in the following, we focus on the works of Kozine and Utkin [12], Campos *et al.* [3], Crossman *et al.* [5, 6] and Blanc and den Hertog [2].

Both Kozine and Utkin [12] and Campos *et al.* [3] looked at calculating the probability of a Markov chain being in state $i = 1, \ldots, n$ at time step $k = 1, 2, \ldots$ which can be calculated as follows,

$$\mathbf{x}_k = \mathbb{P}\mathbf{x}_{k-1},$$

where $\mathbf{x}_k$ is a vector of probabilities of being in states $i = 1, \ldots, n$ at time step k and $\mathbb{P}$ is the transition probability matrix. Under normal circumstances, we have

$$\begin{aligned} \mathbf{x}_k &= \mathbb{P}\mathbf{x}_{k-1} \\ &= \mathbb{P}^2\mathbf{x}_{k-2} \\ &\vdots \\ &= \mathbb{P}^k\mathbf{x}_0. \end{aligned}$$

However, when dealing with intervals, the above formulations may not be equivalent. Recall in Section 2.2, we saw that $X^2 \neq X \cdot X$. Thus, the same care must be taken here as $\mathbb{P}^2$ may not be equal to $\mathbb{P} \times \mathbb{P}$.

Kozine and Utkin [12] developed a method of calculating $\mathbf{x}_k$ at each time step by using $\mathbf{x}_{k-1}$ and this method returned the upper and lower bounds for these probabilities. Although some papers [23, 22, 5] state that Kozine and Utkin [12] consider a time-homogeneous Markov chain (i.e. with a one-sample time-homogeneous interval matrix), this is not entirely true when their method is used to calculate the upper and lower bounds of the interval. In the original formulation of their problem, they allow for some variation in the interval transition probability matrix. That is, instead of choosing the same value of P_{ij} at each time step, they allow this value to change within the intervals, which remain constant. The confusion arises when we compare their original formulation of the problem and the revised formulation used to calculate the upper and lower bounds for the probabilities. In the revised formulation, they use the same upper and lower bounds of the interval transitional probability matrix. Hence, making it seem like they are just taking a single sample and using the same P_{ij} in all their calculations. However, the problem lies with the idea we discussed in Section 2.2 where $X^2 \neq X \cdot X$ for intervals. By using $\mathbf{x}_{k-1}$ to calculate $\mathbf{x}_k$, it brings in the idea of re-sampling as instead of calculating $\mathbf{x}_k = \mathbb{P}\mathbf{x}_{k-1} = \mathbb{P}^2\mathbf{x}_{k-2}$, the calculation performed is $\mathbf{x}_k = \mathbb{P}\mathbf{x}_{k-1} = \mathbb{P} \times \mathbb{P}\mathbf{x}_{k-2}$.

Since they are actually re-sampling from the intervals, as the number of time steps increases they will obtain wider intervals for the probabilities of being state i, than necessary. This was shown and stated in the paper by Campos *et al.* [3]. Since the Kozine and Utkin [12] method produces exact intervals after a single time step, we can use it to calculate bounds on the probabilities $\mathbf{x}_k$ when we have $\mathbb{P}^{k-1}$ and $\mathbf{x}_0$. It is this idea that was explored by Campos *et al.* [3] to reduce the width of the intervals at the second time step. They modified the Kozine and Utkin [12] method by developing an algorithm to calculate $\mathbb{P}^2$ and then used it to calculate $\mathbf{x}_2 = \mathbb{P}^2\mathbf{x}_0$. By combining their algorithm and the method by Kozine and Utkin [12], they were able to obtain exact bounds on the probabilities $\mathbf{x}_2$. Campos *et al.* [3] note that the general problem of calculating exact intervals for 3-step transition probabilities using the algorithm they have developed is NP-hard. However, they hold out hope that a sub-problem may not be NP-hard.

Crossman *et al.* [5] considered the long term behaviour and quasi-stationary behaviour in the framework of a time-homogeneous birth-death processes. They discuss the idea of time-homogeneity carefully and explicitly state that they consider constant transition probabilities and constant bounds on the transition probabilities. Hence, this corresponds to our definition of a one-sample time-homogeneous interval matrix. This is of interest to us as they consider the birth and death processes, which we consider in Chapter 5, where the rows of the transition probability matrix are independent. The other paper by Crossman *et al.* [6] also considers the quasi-stationary behaviour for the one-sample time-homogeneous interval matrix as well as the re-sampled time-homogeneous interval matrix. However, as these papers consider the quasi-stationary distribution only, we do not go into further detail of their work.

Blanc and den Hertog [2] considered the same problem as the one of interest to us in a discussion paper in 2008. They incorporated uncertainty in the transition probabilities by considering lower and upper bounds on the transition probabilities, as we have done, in so-called *box uncertainty*, but also study another form of uncertainty, *ellipsoidal uncertainty*. Since we are more interested in the first type of uncertainty, we analyse their work in that area in more detail. Blanc and den Hertog proceeded to investigate a method to obtain intervals on the following performance measures for Markov chains: limiting distributions, mean sojourn times in transient states, absorption probabilities for absorbing states and probabilities of being in states after m steps. As will be discussed in Chapter 3, we are interested in the mean hitting times and the expected total costs of Markov chains. Hence, we examine Blanc and den Hertog's work in greater detail in Section 3.2 after presenting our problem of interest.

2.5 Markov decision processes

A Markov decision process (MDP) is a way to model decision making processes where we want to optimise a pre-defined objective in a stochastic environment. It will be useful for us in Section 3.5 where we represent our interval calculation as a Markov decision process problem.

2.5.1 Terminology

Here, we define the necessary terms used to describe a Markov decision process, including decision times, states, actions, rewards and transition probabilities. We present the following definitions which arise from Puterman [18].

Definition 2.5.1. Decision times *are time points at which a decision is made. We will only consider discrete time points and let T denote a set of discrete time points. That is, let $T = \{0, 1, 2, \dots, N\}$ where N can be finite or infinite which corresponds to a* finite-horizon *or* infinite-horizon *problem, respectively.*

Definition 2.5.2. *The state space of the system contains the* states *the process occupies at each time step. Let s denote a particular state of the system and let S denote the set of states. Note it is assumed that S does not vary with time. Hence, the set of states, S, remains the same at every time point.*

Definition 2.5.3. Actions *are what the decision maker can choose to do at each decision time. For each state $s \in S$ and time step $t \in T$, the decision maker chooses an action a from the set of allowable actions, A_s, for that particular state. Note that A_s does not vary with time.*

There are two possible ways to choose an action from the action set. We can either choose actions randomly or deterministically. By choosing actions deterministically, this means that an action is chosen with probability 1. On the other hand, if an action is chosen randomly, this means that an action a is chosen with some non-trivial probability. Thus, it is not guaranteed that the same action will *be chosen when the same state is visited at a later time step.*

Definition 2.5.4. *Given that an action $a \in A_s$ is chosen in state s at time step t, the decision maker receives some* reward *denoted by $r_t(s, a)$. This is some real-valued function which can be positive if some reward is earned or negative if some cost is incurred by the decision maker. The value of the reward is dependent on the current state s, the action a chosen from the set of actions and the time step t.*

Definition 2.5.5. Transition probabilities *describe the probability of moving from one state to another. Let $P_t(j|s, a)$ be the probability of moving to state $j \in S$ in the next time step given action a has been chosen and the process is currently in state s at time step t. It is usually assumed that*

$$\sum_{j \in S} P_t(j|s, a) = 1.$$

The reason a Markov decision process is stated to be "Markovian" is because the rewards and transition probabilities depend only on the current state and the chosen action, not on any past history of the process. In addition to the above terms, there are two other important concepts which are important for MDPs.

Definition 2.5.6. *A* decision rule *describes how an action is selected in each state at a specified time point. Depending on how actions are chosen and how the past history of the process is incorporated into decision rules, they can be* deterministic *or* randomised *as well as* Markovian *or* history dependent. *As was discussed earlier, actions can be chosen deterministically or randomly. This corresponds to a deterministic decision rule or a randomised decision rule respectively. Furthermore, a decision rule can be history dependent or Markovian depending on the amount of past information incorporated when the action is chosen. If no history is required and the choice of action only depends on the current state of the system, then the decision rule is said to be Markovian. On the other hand, if the choice of action depends on the previous states of the process and/or previous actions, then* *the decision rule is said to be history dependent.*

Definition 2.5.7. *A* policy *is a sequence of decision rules and specifies the decision rule to be used at all time points. Furthermore, a policy can also be* stationary *or* non-stationary*. A stationary policy means that the choice of decision rule is independent of time. Hence, the same decision rule is chosen at each time step. On the other hand, a non-stationary policy is one where the choice of decision rule depends on either the time or the past history.*

These concepts are of great importance and will be referred to in Section 3.5.

2.6 Optimisation essentials

We present the following concepts and definitions which follow from Chong and Zak [4].

An optimisation problem involves finding an optimal value of a given function on a constrained or unconstrained region. Here, we will only consider constrained optimisation problems of the form

$$\min f(\mathbf{x})$$

subject to

$$\begin{aligned} h_i(\mathbf{x}) &= 0, \qquad \text{for } i = 1, \ldots, p, \\ g_j(\mathbf{x}) &\leq 0, \qquad \text{for } j = 1, \ldots, m, \end{aligned} \tag{2.6.3}$$

where $\mathbf{x} \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}$, $h_i : \mathbb{R}^n \to \mathbb{R}$, $g_j : \mathbb{R}^n \to \mathbb{R}$ and $p \leq n$. Here, we have stated the minimisation problem but note that the maximisation problem is the same as minimising $-f(\mathbf{x})$.

After solving the above optimisation problem, we obtain an optimal solution of the objective function f. There are two types of optimal solutions, *local* and *global*.

Definition 2.6.1 (Local minimum/maximum)**.** $\mathbf{x}^*$ *is a* local minimum *of the constrained region if for some* $\varepsilon > 0$, $f(\mathbf{x}^*) \leq f(\mathbf{x})$ *for all feasible* $\mathbf{x}$ *such that* $||\mathbf{x} - \mathbf{x}^*|| < \varepsilon$. *Similarly,* $\mathbf{x}^*$ *is a* local maximum *of the constrained region if for some* $\varepsilon > 0$, $f(\mathbf{x}^*) \geq f(\mathbf{x})$ *for all feasible* $\mathbf{x}$ *such that* $||\mathbf{x} - \mathbf{x}^*|| < \varepsilon$.

This definition basically tells us that within some region of the entire constrained space, $\mathbf{x}^*$ is an optimal solution on that region. To obtain *strict* local optima, replace the inequalities with strict inequalities.

Definition 2.6.2 (Global minimum/maximum)**.** $\mathbf{x}^*$ *is a* global minimum *of the constrained region if* $f(\mathbf{x}^*) \leq f(\mathbf{x})$ *for all* $\mathbf{x}$ *in the entire constrained region and* $\mathbf{x}^*$ *is a* global maximum *of the constrained region if* $f(\mathbf{x}^*) \geq f(\mathbf{x})$ *for all* $\mathbf{x}$ *in the entire constrained region.*

This tells us that $\mathbf{x}^*$ is the optimal solution on the entire constrained region. Note that the global optimum may not be unique as we have not imposed strict inequalities. Hence, to ensure that a unique global optimum has been obtained, we replace the inequalities in Definition 2.6.2 with strict inequalities.

A useful property of the functions f, h_i and g_j is *convexity* which simplifies the method used to solve the optimisation problem. First, we define a *convex set.*

Definition 2.6.3 (Convex set)**.** *A set* $C \subseteq \mathbb{R}^n$ *is convex if for all* $\mathbf{x}, \mathbf{y} \in C$ *and for all* $\alpha \in [0, 1]$,

$$\mathbf{x} + \alpha(\mathbf{y} - \mathbf{x}) \in C.$$

This means that for a set C to be convex, any line joining two points in C must lie within C.

Definition 2.6.4 (Convex function)**.** *A function* f *defined on a convex set* $C \subseteq \mathbb{R}^n$ *is* convex *if*

$$f(\mathbf{x} + \alpha(\mathbf{y} - \mathbf{x})) \leq f(\mathbf{x}) + \alpha(f(\mathbf{y}) - f(\mathbf{x})),$$

for all $\mathbf{x}, \mathbf{y} \in C$ *and* $\alpha \in [0, 1]$.

This may not be the easiest and most practical way to check if a function is convex. Instead, Theorem 2.6.1 might provide an easier way to check for convexity. Before we present the theorem, we require some definitions.

Definition 2.6.5 (Positive semidefinite)**.** *A real, symmetric matrix A is positive semidefinite if and only if* $\mathbf{z}^T A\mathbf{z} \geq 0$ *for all* $\mathbf{z}$.

Other equivalent ways of testing if a real, symmetric matrix A is positive semidefinite are:

- A is positive semidefinite if and only if all eigenvalues of A are non-negative,
- A is positive semidefinite if and only if all principal minors are greater than or equal to 0, where principal minors are $k \times k$ subdeterminants formed by successively removing a row and its associated column.

Now, let us consider the first and second partial derivatives of the function f. The *gradient* of f consists of the (first) partial derivatives of f and is given by the column vector

$$\nabla f(\mathbf{x}) = \begin{bmatrix} \dfrac{\partial f}{\partial x_1} \\ \dfrac{\partial f}{\partial x_2} \\ \vdots \\ \dfrac{\partial f}{\partial x_n} \end{bmatrix}.$$

The Hessian of f is a square matrix containing the second partial derivatives of f and is represented as follows

$$Hf(\mathbf{x}) = \begin{bmatrix} \dfrac{\partial^2 f}{\partial x_1^2} & \dfrac{\partial^2 f}{\partial x_1 \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_1 \partial x_n} \\ \dfrac{\partial^2 f}{\partial x_2 \partial x_1} & \dfrac{\partial^2 f}{\partial x_2^2} & \cdots & \dfrac{\partial^2 f}{\partial x_2 \partial x_n} \\ \vdots & \vdots & \ddots & \vdots \\ \dfrac{\partial^2 f}{\partial x_n \partial x_1} & \dfrac{\partial^2 f}{\partial x_n \partial x_2} & \cdots & \dfrac{\partial^2 f}{\partial x_n^2} \end{bmatrix}.$$

Note that the Hessian is symmetric if the second partial derivatives of f are continuous.

Now we state a method to check for convexity of the function f using the Hessian matrix, Hf.

Theorem 2.6.1. *Provided the second partial derivatives of f exist (that is, Hf exists), f is convex on the convex set C if and only if $Hf(\mathbf{x})$ is positive semidefinite on C (that is, if and only if $\mathbf{z}^T Hf(\mathbf{x})\mathbf{z} \geq 0$ for all $\mathbf{x} \in C$ and for all $\mathbf{z}$).*

After solving the optimisation problem and obtaining a solution to the objective function, how do we know that this solution is optimal? First and second-order optimality conditions allow us to determine if a point in the feasible region is optimal. Before we state these conditions, let us define some terms needed for the optimality conditions. The presentation below follows from Chong and Zak [4].

Definition 2.6.6 (Active)**.** *$I(\mathbf{x}) = \{j : g_j(\mathbf{x}) = 0\}$ is the set of* active *constraints.*

Definition 2.6.7 (Regular point)**.** *Let $\mathbf{x}^*$ satisfy $h_i(\mathbf{x}^*) = 0$, for $i = 1, \ldots, p$, $g_j(\mathbf{x}^*) \leq 0$, for $j = 1, \ldots, m$, and let $I(\mathbf{x}^*)$ be the set of active constraints at $\mathbf{x}^*$. Then $\mathbf{x}^*$ is a* regular point *if the set of vectors $\Big\{\nabla h_i(\mathbf{x}^*)$, for all $i = 1, \ldots, p$ and $\nabla g_j(\mathbf{x}^*), j \in I(\mathbf{x}^*)\Big\}$ are linearly independent.*

The following theorem gives us the first-order necessary condition for a point $\mathbf{x}^*$ to be a local minimiser of f.

Theorem 2.6.2 (Karush-Kuhn-Tucker (KKT) Theorem)**.** *Let f, h_i, for all $i = 1, \ldots, p$ and g_j, for all $j = 1, \ldots, m$, be sufficiently differentiable (that is, ∇f, ∇h_i, for all $i = 1, \ldots, p$ and ∇g_j, for all $j = 1, \ldots, m$ exist). Let $\mathbf{x}^*$ be a regular point and a local minimiser of $f(\mathbf{x})$ such that $h_i(\mathbf{x}) = 0$, for all $i = 1, \ldots, p$ and $g_j(\mathbf{x}) \leq 0$, for all $j = 1, \ldots, m$. Then, there exists $\mu_i \in \mathbb{R}$, for all $i = 1, \ldots, p$ and $\lambda_j \in \mathbb{R}$, for all $j = 1, \ldots, m$ such that:*

1. $\nabla f(\mathbf{x}^*) + \sum_{i=1}^{p} \mu_i \nabla h_i(\mathbf{x}^*) + \sum_{j=1}^{m} \lambda_j \nabla g_j(\mathbf{x}^*) = \mathbf{0}$,

2. $\lambda_i g_i(\mathbf{x}^*) = 0$, *for all* $i = 1, \ldots, m$, *and*

3. $\lambda_i \geq 0$ *for all* $i = 1, \ldots m$.

A definition required for the second-order conditions is the *Lagrangian function.*

Definition 2.6.8 (Lagrangian function)**.** *The Lagrangian function is given by*

$$l(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = f(\mathbf{x}) + \sum_{i=1}^{p} \mu_i h_i(\mathbf{x}) + \sum_{j=1}^{m} \lambda_j g_j(\mathbf{x}).$$

Then, we let $\mathbf{L}(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\lambda})$ be the Hessian matrix of $l(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\lambda})$ with respect to $\mathbf{x}$:

$$\mathbf{L}(\mathbf{x}, \boldsymbol{\mu}, \boldsymbol{\lambda}) = Hf(\mathbf{x}) + \sum_{i=1}^{p} \mu_i Hh_i(\mathbf{x}) + \sum_{j=1}^{m} \lambda_j Hg_j(\mathbf{x}),$$

where $Hf(\mathbf{x})$ is the Hessian matrix of f at $\mathbf{x}$ and similarly for $Hh_i(\mathbf{x})$ and $Hg_j(\mathbf{x})$.

In the theorem for the second-order sufficient conditions, we use the following set:

$$\begin{aligned} T(\mathbf{x}^*, \boldsymbol{\lambda}) = \Big\{ \mathbf{y} : \nabla h_i(\mathbf{x}^*)^T \mathbf{y} = 0, \ \text{for all } i = 1, \ldots, p \\ \text{and } \nabla g_j(\mathbf{x}^*)^T \mathbf{y} = 0, j \in \tilde{I}(\mathbf{x}^*, \boldsymbol{\lambda}) \Big\}, \end{aligned}$$

where $\tilde{I}(\mathbf{x}^*, \boldsymbol{\lambda}) = \{i : g_i(\mathbf{x}^*) = 0, \lambda_i > 0\}$.

Now that we have defined all terms required, we state the second-order sufficient conditions for a point $\mathbf{x}^*$ to be a strict local minimiser of f.

Theorem 2.6.3 (Second-Order Sufficient Conditions)**.** *Suppose that* f, h_i, *for all* $i = 1, \ldots, p$ *and* g_j, *for all* $j = 1, \ldots, m$, *are sufficiently differentiable (that is,* Hf, Hh_i, *for all* $i = 1, \ldots, p$ *and* Hg_j, *for all* $j = 1, \ldots, m$ *exist) and there exists a feasible point* $\mathbf{x}^* \in \mathbb{R}^n$ *and* $\mu_i \in \mathbb{R}$, *for all* $i = 1, \ldots, p$ *and* $\lambda_j \in \mathbb{R}$, *for all* $j = 1, \ldots, m$ *such that:*

1. $\nabla f(\mathbf{x}^*) + \sum_{i=1}^{p} \mu_i \nabla h_i(\mathbf{x}^*) + \sum_{j=1}^{m} \lambda_j \nabla g_j(\mathbf{x}^*) = \mathbf{0}$,

2. $\lambda_i g_i(\mathbf{x}^*) = 0$, *for all* $i = 1, \ldots, m$,

3. $\lambda_i \geq 0$ *for all* $i = 1, \ldots m$*, and*

4. For all $\boldsymbol{y} \in T(\mathbf{x}^*, \boldsymbol{\lambda}), \boldsymbol{y} \neq \mathbf{0}$*, we have* $\boldsymbol{y}^T \mathbf{L}(\mathbf{x}^*, \boldsymbol{\mu}, \boldsymbol{\lambda})\boldsymbol{y} > 0$*.*

Then, $\mathbf{x}^*$ *is a strict local minimiser of* f *subject to* $h_i(\mathbf{x}) = 0$*, for all* $i = 1, \ldots, p$ *and* $g_j(\mathbf{x}) \leq 0$*,* $j = 1, \ldots, m$*.*

This concludes the background chapter which states definitions, notation and other material required in later chapters. Throughout all chapters, we will endeavour to re-state or refer the reader back to any useful background material to allow for easier reading. In the next chapter, we focus on developing theoretical properties for calculating the interval of expected total costs for discrete-time and continuous-time interval Markov chains.

Chapter 3

Markov chains: Analytic investigation

Markov chains are used in a wide variety of areas to model processes which evolve randomly over time. The tractable nature of Markov chains is a reason for their attractiveness as it allows various performance measures to be easily obtained. In Chapter 2, we discussed the common practice of using point estimates of transition probabilities in the transition probability matrix and also introduced the idea of using intervals to allow us to take into account uncertainty in the estimates of the transition probabilities. The combination of intervals and Markov chains are known as interval Markov chains.

In this chapter, we first consider *discrete-time interval Markov chains* and seek to obtain intervals for the expected total costs. To do so, we explore theoretical properties which simplify our problem and investigate if an analytic solution exists. After exploring theoretical properties for discrete-time interval Markov chains, we turn our attention to *continuous-time interval Markov chains* and consider the use of uniformisation (Section 2.1.2) to exploit the theoretical properties developed for discrete-time interval Markov chains and the transformations required to obtain the desired performance measures.

3.1 Description of problem

Consider a discrete-time interval Markov chain $(X_m, m \in \mathbb{Z}_+)$ with $n+1$ states, $S = \{0, 1, \ldots, n\}$, where state 0 is an absorbing state. Assume X_m has an interval transition probability matrix,

$$\mathbb{P} = \left[\begin{array}{c|ccc} [1,1] & [0,0] & \cdots & [0,0] \\ \hline [\underline{P}_{10}, \overline{P}_{10}] & & & \\ \vdots & & \mathbb{P}_s & \\ [\underline{P}_{n0}, \overline{P}_{n0}] & & & \end{array}\right]$$

$$= \left[\begin{array}{c|cccccc} [1,1] & [0,0] & \cdots & \cdots & \cdots & [0,0] \\ \hline [\underline{P}_{10}, \overline{P}_{10}] & [\underline{P}_{11}, \overline{P}_{11}] & \cdots & \cdots & \cdots & [\underline{P}_{1n}, \overline{P}_{1n}] \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ [\underline{P}_{i0}, \overline{P}_{i0}] & [\underline{P}_{i1}, \overline{P}_{i1}] & \cdots & [\underline{P}_{ij}, \overline{P}_{ij}] & \cdots & [\underline{P}_{in}, \overline{P}_{in}] \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ [\underline{P}_{n0}, \overline{P}_{n0}] & [\underline{P}_{n1}, \overline{P}_{n1}] & \cdots & \cdots & \cdots & [\underline{P}_{nn}, \overline{P}_{nn}] \end{array}\right]$$

where

$$0 \le \underline{P}_{ij} \le 1, \qquad \text{for all } i = 1, \ldots, n \text{ and } j = 0, \ldots, n, \tag{3.1.1}$$

$$0 \le \overline{P}_{ij} \le 1, \qquad \text{for all } i = 1, \ldots, n \text{ and } j = 0, \ldots, n, \tag{3.1.2}$$

$$\underline{P}_{ij} \le \overline{P}_{ij}, \qquad \text{for all } i = 1, \ldots, n \text{ and } j = 0, \ldots, n, \tag{3.1.3}$$

and there exists a realisation

$$P_{ij} \in \left[\underline{P}_{ij}, \overline{P}_{ij}\right], \quad \text{for all } i = 1, \ldots, n \text{ and } j = 0, \ldots, n, \tag{3.1.4}$$

such that

$$\sum_{j=0}^{n} P_{ij} = 1, \quad \text{for all } i = 1, \ldots, n. \tag{3.1.5}$$

The above conditions ensure that:

- the end-points of the intervals are valid probabilities (3.1.1 - 3.1.2);
- the lower bound of the intervals are not bigger than the upper bounds (3.1.3); and,
- there exists probabilities within the intervals that sum to 1 for each row of the matrix (3.1.4 - 3.1.5).

That is, these conditions ensure that there exists a valid stochastic matrix with its elements contained within the specified intervals. We now seek to evaluate the *interval expected total cost* of a discrete-time interval Markov chain.

Recall that χ_i is defined to be the expected total cost incurred before the process is absorbed into state 0, conditional on starting in state i. We can calculate it using,

$$\boldsymbol{\chi} = (I - P_s)^{-1}\,\mathbf{c},$$

where I is the identity matrix of the same size as P_s, $\mathbf{c}$ is a vector of costs per visit and P_s is the sub-matrix of the transition probability matrix P. However, recall that there are requirements on P_s to ensure that $(I - P_s)^{-1}$ is well defined. A sufficient condition is that P_s is finite-dimensional and has a spectral radius less than 1. The matrix P_s is finite-dimensional since our state space, S, is finite.

For P_s to have a spectral radius of 1, there must exist an absorbing state or absorbing sub-class, under P_s, in $\{1, 2, \ldots, n\}$. Equivalently, this means that the probability of absorption to state 0 is not equal to 1 for some state $j \in S$, since there is a chance that the process might be absorbed into the absorbing state or absorbing sub-class, under P_s, in $\{1, 2, \ldots, n\}$ as opposed to being absorbed into state 0. To ensure that P_s has a spectral radius less than 1 and hence $(I - P_s)^{-1}$ to be well defined, we require the probability of absorption to state 0 from all states $j \in S$ to equal 1.

Let us define the *interval expected total cost* to be

$$[\underline{\boldsymbol{\chi}}, \overline{\boldsymbol{\chi}}] = (I - \mathbb{P}_s)^{-1} \mathbf{c}, \tag{3.1.6}$$

where $\underline{\boldsymbol{\chi}}$ and $\overline{\boldsymbol{\chi}}$ represent the lower and upper bound on the expected total costs respectively and $\mathbb{P}_s$ is the sub-matrix of the interval matrix $\mathbb{P}$. As before with the non-interval case, we have the same problem that $(I - \mathbb{P}_s)^{-1}$ may not be well defined. Unlike the non-interval case where there is a single P_s matrix to work with, for the interval version, $\mathbb{P}_s$ represents a range of possible P_s matrices. To ensure that every possible P_s has a spectral radius less than 1, stringent constraints are needed on $\mathbb{P}$, for example, $\underline{P}_{ij} > 0$ for all $i, j = 1, \ldots, n$. However, this greatly reduces the range of problems we can consider. We could also consider imposing other constraints on the problem which are less restrictive but these may be too difficult to handle both analytically and numerically.

Hence, we consider problems with an interval matrix $\mathbb{P}$ such that there exists a realisation $P \in \mathbb{P}$ with sub-matrix P_s that has a spectral radius less than 1. This means that there exists a matrix $P \in \mathbb{P}$ such that $(I - P_s)^{-1}$ is well defined and also allows us to obtain a solution for $\boldsymbol{\chi}$. Thus, as long as this property holds, we have a well-defined problem which ensures that we are able to find the interval expected total cost for the discrete-time interval Markov chain.

3.2 Possible methods to obtain interval expected total costs

There are a number of methods which could be used to calculate the interval expected total cost for a given discrete-time interval Markov chain. However, recall that we want to calculate minimal intervals, which are intervals of smallest width containing the true solution.

A possible method for calculating the interval expected total cost is to solve the interval system of linear equations corresponding to (3.1.6) using INTLAB (Section 2.2). However, the methods used by INTLAB are iterative. This causes problems with imposing the required constraints on the interval transition probability matrix, $\mathbb{P}$, as well as preserving the same choice of P_{ij} at each time step. Recall that we need to preserve the choice of P_{ij} at each time step to retain the idea of a time-homogeneous discrete-time Markov chain as well as ensuring intervals are as tight as possible (due to the difference between $X \cdot X$ and X^2). The other important condition required is to ensure that each realisation of the interval transition probability matrix has a row sum of 1. However, we are unable to ensure that this condition is satisfied when using interval arithmetic and INTLAB to solve the problem. Hence, using INTLAB to calculate the interval expected total cost does not satisfy our requirements and so another method has to be developed.

We decided to solve for the interval expected total cost using optimisation techniques. That is, we seek to minimise and maximise the expected total cost for each state of our interval Markov chain. Note that this is the classical method of finding the expected total cost without intervals. This then gives us the lower and upper bounds on the expected total cost. Thus, for n states, this means solving a total of $2n$ optimisation problems, each of which are described in Section 3.2.1.

First, recall that we briefly discussed the work of Blanc and den Hertog in Section 2.4. Here, we examine their method in greater detail for obtaining the mean sojourn times in transient states as this is directly related to our problem, which is to obtain the expected total costs for a discrete-time interval Markov chain. The following presentation goes through the method developed by Blanc and den Hertog and hence follows closely the presentation of the authors [2].

Consider a transition probability matrix with the following form,

$$P = \begin{pmatrix} I & O \\ R & Q \end{pmatrix},$$

where Q is a matrix containing the transition probabilities of moving from one transient state to another, R is a matrix containing the transition probabilities of moving from a transient state to an absorbing state, O is a matrix of zeros as this represents the probabilities of leaving an absorbing state and I is the identity matrix representing the probabilities of remaining in absorbing states. Since row sums of a transition probability matrix must sum to 1, $Q\mathbf{1} + R\mathbf{1} = \mathbf{1}$.

The problem we are interested in is Blanc and den Hertog's calculation of the mean sojourn time from some state k in the set of transient states, given there is some initial distribution $\mathbf{q}_0$ over the set of transient states. Since they are interested in obtaining intervals on the mean sojourn times, optimisation problems are considered to minimise and maximise the mean sojourn time given a set of constraints. The optimisation problem of interest is of the form

$$\min_{(Q,R)\in\mathcal{U}} \left\{ \mathbf{q}_0^T \left(I - Q\right)^{-1} \mathbf{c} | Q\mathbf{1} + R\mathbf{1} = \mathbf{1} \right\},$$

where $\mathcal{U}$ is an interval uncertainty region defined by,

$$\underline{Q} \leq Q \leq \overline{Q} \qquad \text{and} \qquad \underline{R} \leq R \leq \overline{R}.$$

Since we require $Q\mathbf{1} + R\mathbf{1} = \mathbf{1}$, this implies that the upper and lower bounds on Q and R should also satisfy the following condition,

$$\underline{Q}\mathbf{1} + \underline{R}\mathbf{1} \leq Q\mathbf{1} + R\mathbf{1} \leq \overline{Q}\mathbf{1} + \overline{R}\mathbf{1}.$$

Furthermore, it is noted that since the matrix R is not of interest itself, the above constraint can be reduced to $\mathbf{1} - \overline{R}\mathbf{1} \leq Q\mathbf{1} \leq \mathbf{1} - \underline{R}\mathbf{1}$.

The novelty of Blanc and den Hertog's method is the idea of introducing new variables to transform the above non-linear optimisation problem into a linear optimisation problem. The benefit of this transformation is that linear programs are typically easier and quicker to solve than non-linear optimisation problems.

The non-linearity of the problem appears in $(I - Q)^{-1}$ which forms part of the objective function. Thus, to remove this, the vector $\mathbf{v}^T = \mathbf{q}_0^T (I - Q)^{-1}$ is introduced to transform the optimisation problem into the form,

$$\min_{\mathbf{v},Q} \left\{ \mathbf{v}^T \mathbf{c} | \mathbf{v}^T (I - Q) = \mathbf{q}_0^T,\, \underline{Q} \le Q \le \overline{Q},\, \mathbf{1} - \overline{R}\mathbf{1} \le Q\mathbf{1} \le \mathbf{1} - \underline{R}\mathbf{1} \right\}.$$

Next, a new matrix Ξ with elements,

$$\xi_{ij} = v_i (I_{ij} - q_{ij}), \qquad \text{for } i, j \in \mathcal{T},$$

where $\mathcal{T}$ represents the set of transient states, can be introduced to transform the non-linear constraints $\mathbf{v}^T (I - Q) = \mathbf{q}_0^T$ to linear constraints of the form $\mathbf{1}^T \Xi = \mathbf{q}_0^T$.

Furthermore, imposing this transformation on the original variables does not change the linearity of the row sums. The new constraints on the row sums are given by,

$$v_i \sum_{\ell \in \mathcal{A}} \underline{r}_{i\ell} \le \sum_{j \in \mathcal{T}} \xi_{ij} \le v_i \sum_{\ell \in \mathcal{A}} \overline{r}_{i\ell}\,, \qquad i \in \mathcal{T},$$

where $\mathcal{A}$ represents the set of absorbing states.

Hence, the full linear program can be written as,

$$\min_{\mathbf{v},\Xi} \left\{ \mathbf{v}^T \mathbf{c} \,\middle|\, \mathbf{1}^T \Xi = \mathbf{q}_0^T,\, v_i (I_{ij} - \overline{q}_{ij}) \le \xi_{ij} \le v_i (I_{ij} - \underline{q}_{ij}),\, i, j \in \mathcal{T}, \right.$$
$$\left. v_i \sum_{\ell \in \mathcal{A}} \underline{r}_{i\ell} \le \sum_{j \in \mathcal{T}} \xi_{ij} \le v_i \sum_{\ell \in \mathcal{A}} \overline{r}_{i\ell},\, i \in \mathcal{T} \right\}.$$

For the above linear problem, there are $N^2 + N$ decision variables, N equality constraints and $2 \times (N^2 + N)$ inequality constraints, where N is the number of transient states for the Markov chain. Note that a constraint of the form $a \le x \le b$

is considered to be 2 separate inequality constraints, $a \leq x$ and $x \leq b$. Hence, resulting in $2 \times (N^2 + N)$ inequality constraints as opposed to the $N^2 + N$ inequality constraints reported by [2].

Unfortunately, we found this unpublished discussion paper by Blanc and den Hertog [2] just as this thesis was being completed. Thus, we were unable to fully utilise this method. Throughout the duration of this project, we have explored the theoretical properties (Chapter 3) and developed our own numerical method (Chapter 4) for the original non-linear optimisation problem, which is discussed in more detail in Section 3.2.1. Therefore, we have only provided a numerical comparison of Blanc and den Hertog's method with the method we developed in Section 4.10.

3.2.1 Optimisation problems

Since we seek to minimise and maximise the expected total cost for each state, the objective function we are interested in is given as follows for each state $k = 1, \ldots, n$,

$$\chi_k = \left[(I - P_s)^{-1} \mathbf{c}\right]_k ,$$

where

$$P_s = \begin{bmatrix} P_{11} & \cdots & \cdots & \cdots & P_{1n} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ P_{i1} & \cdots & P_{ij} & \cdots & P_{in} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ P_{n1} & \cdots & \cdots & \cdots & P_{nn} \end{bmatrix},$$

is a realisation of the interval $\mathbb{P}_s$ matrix and P_{ij} are the decision variables for this problem.

Thus, this is the function we seek to minimise and maximise subject to the following constraints,

$$\begin{aligned} 1-\overline{P}_{i0} \leq \sum_{j=1}^{n} P_{ij} \leq 1-\underline{P}_{i0}, & \qquad \text{for } i=1,\ldots,n, \\ \underline{P}_{ij} \leq P_{ij} \leq \overline{P}_{ij}, & \qquad \text{for } i,j=1,\ldots,n. \end{aligned}$$

We note that these conditions follow from (3.1.1 - 3.1.5) as we combine the following equations,

$$\begin{aligned} \sum_{j=0}^{n} P_{ij} = 1, & \qquad \text{for } i=1,\ldots,n, \\ \underline{P}_{i0} \leq P_{i0} \leq \overline{P}_{i0}, & \qquad \text{for all } i=1,\ldots,n, \end{aligned}$$

to remove the redundancy in the original constraints.

Here, we state the minimisation problem formally and note that the maximisation problem follows by maximising instead of minimising the objective function.

For each state $k=1,\ldots,n$, we want to solve the following problem to obtain the lower bounds on the expected total costs.

$$\min \chi_k = \left[(I-P_s)^{-1}\mathbf{c}\right]_k,$$

subject to

$$\begin{aligned} 1-\overline{P}_{i0} \leq \sum_{j=1}^{n} P_{ij} \leq 1-\underline{P}_{i0}, & \qquad \text{for } i=1,\ldots,n, \\ \underline{P}_{ij} \leq P_{ij} \leq \overline{P}_{ij}, & \qquad \text{for } i,j=1,\ldots,n. \end{aligned}$$

Recall that two important and useful properties commonly used in optimisation problems are the gradient and the Hessian. Here, we investigate analytic expressions for the gradient and Hessian of χ_k.

Gradient

The following theorem provides us with a method to calculate the gradient of the objective function for each state k of the Markov chain, for $k = 1, \dots, n$. First, we introduce an ordering of the variables. As the gradient is an $n^2 \times 1$ vector where each element is found by differentiating the objective function with respect to an element in P_s, we need some way to order the variables. That is, we need an ordering to represent an $n \times n$ matrix using an $n^2 \times 1$ vector. The ordering we use is to consider stacking the rows of the P_s matrix alongside each other, which is the lexicographic ordering. Thus, the $n^2 \times 1$ vector is represented by,

$$\Big[(1,1),(1,2),\dots,(1,n),(2,1),\dots,(2,n),(3,1),\dots,(n-1,n),(n,1),\dots,(n,n)\Big]^T,$$

where (i,j) represents the ij^{th} element of the P_s matrix.

Theorem 3.2.1. *For each state $k = 1, \dots, n$, the gradient of χ_k is an $n^2 \times 1$ vector and is given by*

$$\begin{aligned}\nabla \chi_k &= \left[\frac{\partial \chi_k}{\partial P_{ij}}\right]_{i,j=1,\dots,n} \\ &= \Big[\left[(I-P_s)^{-1}\right]_{ki}\left[(I-P_s)^{-1}\mathbf{c}\right]_j\Big]_{i,j=1,\dots,n},\end{aligned}$$

ordered as above. That is, the gradient is of the form:

$$\nabla \chi_k = \begin{bmatrix} \left[(I-P_s)^{-1}\right]_{k1}\left[(I-P_s)^{-1}\mathbf{c}\right]_1 \\ \vdots \\ \left[(I-P_s)^{-1}\right]_{k1}\left[(I-P_s)^{-1}\mathbf{c}\right]_n \\ \vdots \\ \left[(I-P_s)^{-1}\right]_{kn}\left[(I-P_s)^{-1}\mathbf{c}\right]_1 \\ \vdots \\ \left[(I-P_s)^{-1}\right]_{kn}\left[(I-P_s)^{-1}\mathbf{c}\right]_n \end{bmatrix}.$$

Proof. To prove the above theorem, we need the following properties of matrix differentiation [17]:

$$\frac{\partial \left(\mathbf{X}^{-1}\right)_{kl}}{\partial X_{ij}} = -\left(\mathbf{X}^{-1}\right)_{ki}\left(\mathbf{X}^{-1}\right)_{jl}, \qquad \text{for all } i,j,k \text{ and } l, \tag{3.2.7}$$

$$\partial(\mathbf{XY}) = (\partial\mathbf{X})\mathbf{Y} + \mathbf{X}(\partial\mathbf{Y}). \tag{3.2.8}$$

For our case, let $\mathbf{X} = (I - P_s)$ and so we can re-write the objective function as

$$\chi_k = \left[\mathbf{X}^{-1}\mathbf{c}\right]_k, \qquad \text{for } k = 1, \ldots, n.$$

Therefore,

$$\frac{\partial \left(\mathbf{X}^{-1}\right)_{kl}}{\partial P_{ij}} = \frac{\partial \left(\mathbf{X}^{-1}\right)_{kl}}{\partial X_{ij}}\frac{dX_{ij}}{dP_{ij}}, \qquad \text{for all } i,j,k \text{ and } l. \tag{3.2.9}$$

Furthermore, note that

$$\frac{dX_{ij}}{dP_{ij}} = \frac{d\left(I - P_s\right)_{ij}}{dP_{ij}} = -1. \tag{3.2.10}$$

Thus, for each state $k = 1, \ldots, n$ and for $i, j = 1, \ldots, n$, we have

$$\begin{aligned}
\left[\nabla\chi_k\right]_{(i,j)} &= \frac{\partial\chi_k}{\partial P_{ij}} \\
&= \frac{\partial\left(\mathbf{X}^{-1}\mathbf{c}\right)_k}{\partial P_{ij}} \\
&= \left\{\left(\frac{\partial\mathbf{X}^{-1}}{\partial P_{ij}}\right)\mathbf{c}\right\}_k + \left\{\mathbf{X}^{-1}\left(\frac{\partial\mathbf{c}}{\partial P_{ij}}\right)\right\}_k, && \text{from (3.2.8)} \\
&= \sum_{\ell=1}^{n} c_\ell \frac{\partial\left(\mathbf{X}^{-1}\right)_{k\ell}}{\partial P_{ij}} + 0, && \text{since } \frac{\partial\mathbf{c}}{\partial P_{ij}} = \mathbf{0} \\
&= \sum_{\ell=1}^{n} c_\ell \frac{\partial\left(\mathbf{X}^{-1}\right)_{k\ell}}{\partial X_{ij}}\frac{dX_{ij}}{dP_{ij}}, && \text{from (3.2.9)} \\
&= \sum_{\ell=1}^{n} c_\ell \left\{-\left(\mathbf{X}^{-1}\right)_{ki}\left(\mathbf{X}^{-1}\right)_{j\ell}\right\}(-1), && \text{from (3.2.7) and (3.2.10)} \\
&= \sum_{\ell=1}^{n} c_\ell \left(\mathbf{X}^{-1}\right)_{ki}\left(\mathbf{X}^{-1}\right)_{j\ell} \\
&= \left(\mathbf{X}^{-1}\right)_{ki}\sum_{\ell=1}^{n} c_\ell \left(\mathbf{X}^{-1}\right)_{j\ell} \\
&= \left(\mathbf{X}^{-1}\right)_{ki}\left(\mathbf{X}^{-1}\mathbf{c}\right)_j .
\end{aligned}$$

Thus, the gradient of χ_k is given by

$$\nabla \chi_k = \left[\left[(I-P_s)^{-1}\right]_{ki}\left[(I-P_s)^{-1}\mathbf{c}\right]_j\right]_{i,j=1,\ldots,n}, \qquad \text{for } k=1,\ldots,n.$$

□

Hessian

The Hessian of the objective function, χ_k, can be calculated using the following theorem.

Theorem 3.2.2. *For each state k, the Hessian of the objective function, χ_k, is an $n^2 \times n^2$ matrix and is given by*

$$\begin{aligned} H\,\chi_k &= \left[\frac{\partial^2 \chi_k}{\partial P_{\ell m}\partial P_{ij}}\right]_{i,j=1,\ldots,n \text{ and } \ell,m=1,\ldots,n} \\ &= \Big[\left[(I-P_s)^{-1}\right]_{k\ell}\left[(I-P_s)^{-1}\right]_{mi}\left[(I-P_s)^{-1}\mathbf{c}\right]_j \\ &\quad + \left[(I-P_s)^{-1}\right]_{ki}\left[(I-P_s)^{-1}\right]_{j\ell}\left[(I-P_s)^{-1}\mathbf{c}\right]_m\Big]_{i,j=1,\ldots,n \text{ and } \ell,m=1,\ldots,n}, \end{aligned}$$

where i and j are in lexicographic order for rows, and ℓ and m are in lexicographic order for columns, which is the same ordering as used in the gradient.

Proof. As with the proof of the gradient, we require the same properties of matrix differentiation to prove the analytic form of the Hessian.

Define $\mathbf{X} = (I - P_s)$ and we have for state $k = 1, \ldots, n$,

$$\chi_k = \left[\mathbf{X}^{-1}\mathbf{c}\right]_k \qquad \text{and} \qquad \nabla \chi_k = \left[\left(\mathbf{X}^{-1}\right)_{ki}\left(\mathbf{X}^{-1}\mathbf{c}\right)_j\right]_{i,j=1,\ldots,n}.$$

Recall that the Hessian is the second derivative of the objective function. Hence to obtain the Hessian, we can either differentiate the objective function twice or differentiate the gradient once to give us the following. For each state $k = 1, \ldots, n$

and for $i, j = 1, \ldots, n$, we have

$$\begin{aligned}
\left[H\,\chi_k\right]_{(i,j),(\ell,m)} &= \frac{\partial^2 \chi_k}{\partial P_{\ell m} \partial P_{ij}} \\
&= \frac{\partial}{\partial P_{\ell m}} \frac{\partial \chi_k}{\partial P_{ij}} \\
&= \frac{\partial}{\partial P_{\ell m}} \frac{\partial \left(\mathbf{X}^{-1}\mathbf{c}\right)_k}{\partial P_{ij}} \\
&= \frac{\partial}{\partial P_{\ell m}} \left\{\left(\mathbf{X}^{-1}\right)_{ki} \left(\mathbf{X}^{-1}\mathbf{c}\right)_j\right\}, \qquad \text{from gradient } \nabla\,\chi_k \\
&= \frac{\partial}{\partial X_{\ell m}} \left\{\left(\mathbf{X}^{-1}\right)_{ki} \left(\mathbf{X}^{-1}\mathbf{c}\right)_j\right\} \frac{dX_{\ell m}}{dP_{\ell m}}, \qquad \text{from (3.2.9)} \\
&= \left[\left(\frac{\partial \left(\mathbf{X}^{-1}\right)_{ki}}{\partial X_{\ell m}}\right) \left(\mathbf{X}^{-1}\mathbf{c}\right)_j + \left(\mathbf{X}^{-1}\right)_{ki} \left(\frac{\partial \left(\mathbf{X}^{-1}\mathbf{c}\right)_j}{\partial X_{\ell m}}\right)\right](-1), \\
&\quad \text{from (3.2.8) and (3.2.10)} \\
&= -\left[\left(\frac{\partial \left(\mathbf{X}^{-1}\right)_{ki}}{\partial X_{\ell m}}\right) \left(\mathbf{X}^{-1}\mathbf{c}\right)_j + \left(\mathbf{X}^{-1}\right)_{ki} \left(\frac{\partial}{\partial X_{\ell m}} \sum_{r=1}^{n} c_r \left(\mathbf{X}^{-1}\right)_{jr}\right)\right] \\
&= -\left[\left(\frac{\partial \left(\mathbf{X}^{-1}\right)_{ki}}{\partial X_{\ell m}}\right) \left(\mathbf{X}^{-1}\mathbf{c}\right)_j + \left(\mathbf{X}^{-1}\right)_{ki} \sum_{r=1}^{n} c_r \frac{\partial \left(\mathbf{X}^{-1}\right)_{jr}}{\partial X_{\ell m}}\right] \\
&= -\Bigg[\left\{-\left(\mathbf{X}^{-1}\right)_{k\ell} \left(\mathbf{X}^{-1}\right)_{mi}\right\} \left(\mathbf{X}^{-1}\mathbf{c}\right)_j + \\
&\qquad \left(\mathbf{X}^{-1}\right)_{ki} \sum_{r=1}^{n} c_r \left\{-\left(\mathbf{X}^{-1}\right)_{j\ell} \left(\mathbf{X}^{-1}\right)_{mr}\right\}\Bigg], \qquad \text{from (3.2.7)} \\
&= \left(\mathbf{X}^{-1}\right)_{k\ell} \left(\mathbf{X}^{-1}\right)_{mi} \left(\mathbf{X}^{-1}\mathbf{c}\right)_j + \left(\mathbf{X}^{-1}\right)_{ki} \left(\mathbf{X}^{-1}\right)_{j\ell} \sum_{r=1}^{n} c_r \left(\mathbf{X}^{-1}\right)_{mr} \\
&= \left(\mathbf{X}^{-1}\right)_{k\ell} \left(\mathbf{X}^{-1}\right)_{mi} \left(\mathbf{X}^{-1}\mathbf{c}\right)_j + \left(\mathbf{X}^{-1}\right)_{ki} \left(\mathbf{X}^{-1}\right)_{j\ell} \left(\mathbf{X}^{-1}\mathbf{c}\right)_m .
\end{aligned}$$

Hence, by replacing $\mathbf{X}$ with $(I - P_s)$, we get

$$\begin{aligned}
H\,\chi_k = \Big[&\left[(I-P_s)^{-1}\right]_{k\ell} \left[(I-P_s)^{-1}\right]_{mi} \left[(I-P_s)^{-1}\mathbf{c}\right]_j \\
&+ \left[(I-P_s)^{-1}\right]_{ki} \left[(I-P_s)^{-1}\right]_{j\ell} \left[(I-P_s)^{-1}\mathbf{c}\right]_m \Big]_{i,j=1,\ldots,n \text{ and } \ell,m=1,\ldots,n} .
\end{aligned}$$

□

Recall that we mentioned in Section 2.6, if our objective function is convex then the method used to solve the optimisation problem is simplified. Also recall that for a function to be convex, we require the function to be defined on a convex set. All constraints of the problem are either linear constraints or bounds on the decision variables P_{ij}. Since each constraint is a linear function of P_{ij} and all linear functions are convex, these constraints form a convex region. Thus, the objective function is defined on a convex set. Now, we need to determine if the objective function is convex.

Recall from Section 2.6, that a function is convex on a convex set if and only if the Hessian of the function is positive semidefinite on the convex set. To check if the Hessian is positive semidefinite, we can check that all eigenvalues of the Hessian are non-negative. We will show, by counter-example, that not all eigenvalues are non-negative.

Let us consider a discrete-time interval Markov chain with the following interval transition probability matrix,

$$\mathbb{P} = \begin{bmatrix} [1,1] & [0,0] & [0,0] \\ [0.1,0.2] & [0,0.3] & [0.4,0.7] \\ [0.2,0.3] & [0.1,0.4] & [0.5,0.6] \end{bmatrix}.$$

We are interested in obtaining the interval expected total cost for state $k = 1$ with costs $c_i = 1$. Thus, this problem reduces to obtaining the interval mean hitting time for the discrete-time Markov chain. The optimisation problem we are interested in solving is

$$\chi_1 = \left[(I - P_s)^{-1}\mathbf{1}\right]_1,$$

subject to

$$\begin{aligned} 1 - \overline{P}_{i0} \leq \sum_{j=1}^{n} P_{ij} \leq 1 - \underline{P}_{i0}, & \qquad \text{for } i = 1, \ldots, n, \\ \underline{P}_{ij} \leq P_{ij} \leq \overline{P}_{ij}, & \qquad \text{for } i, j = 1, \ldots, n, \end{aligned}$$

for the given interval transition probability matrix above.

Let us consider a realisation P_s of the interval $\mathbb{P}_s$ matrix,

$$P_s = \begin{bmatrix} 0.1 & 0.7 \\ 0.1 & 0.6 \end{bmatrix}.$$

The Hessian matrix of the objective function χ_1 is

$$H\chi_1 = \begin{bmatrix} 14.4327 & 8.3644 & 25.2573 & 14.6377 \\ 8.3644 & 3.2802 & 27.7147 & 17.6309 \\ 25.2573 & 27.7174 & 44.2003 & 48.5055 \\ 14.6377 & 17.6309 & 48.5055 & 51.6626 \end{bmatrix},$$

which has eigenvalues: $-13.3626, 0, 11.9202$ and 115.0182. Since the first eigenvalue is negative, our objective function χ_1 is not convex. Hence, we have a counter-example for the convexity of the general objective function and can conclude that the objective function

$$\chi_k = \left[(I - P_s)^{-1}\,\mathbf{c}\right]_k,$$

is not necessarily convex.

Since the objective function is non-convex, we now seek other useful analytic properties to simplify the problem and also investigate if there is a simple analytic solution before considering possible numerical methods to solve the optimisation problems.

3.3 Simplification of the optimisation problems

Recall that the only constraints in our optimisation problems are the linear inequality constraints,

$$1 - \overline{P}_{i0} \le \sum_{j=1}^{n} P_{ij} \le 1 - \underline{P}_{i0}, \qquad \text{for } i = 1, \ldots, n,$$

and the bounds on the decision variables,

$$\underline{P}_{ij} \le P_{ij} \le \overline{P}_{ij}, \qquad \text{for } i, j = 1, \ldots, n.$$

Also recall that we were able to remove the redundancy in the original constraints and reduce the number of decision variables for our problem by combining constraints. Now, we would like to investigate if it is possible to simplify the problem further. We can simplify the problem by reducing the two linear inequality constraints to a single linear equality constraint for each row $i = 1, \ldots, n$ of the P_s matrix. By doing this, we obtain slightly different constraints for the minimisation and maximisation problems. Note that from now on we will consider non-negative costs, unless otherwise stated. Without the non-negativity condition, the following simplifications do not hold. Furthermore, we note that most practical applications involve the use of non-negative costs and in particular, applications of interest to us, such as the mean hitting time and final epidemic size (Chapter 6), have non-negative costs.

Let us first consider the minimisation problem. The idea behind the reduction of two linear inequality constraints to a single linear equality constraint is based on minimising the expected total cost. By minimising the expected total cost, this means we want to maximise the probability of direct absorption from any state, that is, set $P_{i0} = \overline{P}_{i0}$, as no further costs are incurred after absorption. Similarly for the maximisation problem, we want to maximise the expected total cost. Thus, we want to minimise the probability of direct absorption from any state, that is, setting $P_{i0} = \underline{P}_{i0}$, to continue incurring cost. This results in the following linear equality constraints for the minimisation and maximisation problems.

For the minimisation problem, the linear equality constraint is given by,

$$\sum_{j=1}^{n} P_{ij} = 1 - \overline{P}_{i0}, \qquad \text{for } i = 1, \ldots, n,$$

whilst for the maximisation problem, we have,

$$\sum_{j=1}^{n} P_{ij} = 1 - \underline{P}_{i0}, \qquad \text{for } i = 1, \ldots, n.$$

We now provide proofs for these simplifications.

3.3.1 Minimisation problem

First, consider the minimisation problem and we want to prove the following theorem.

Theorem 3.3.1. *For the minimisation problem, the following constraint*

$$\sum_{j=1}^{n} P_{ij} = 1 - \overline{P}_{i0}, \qquad \textit{for } P_{ij} \in \left[\underline{P}_{ij}, \overline{P}_{ij}\right] \textit{ and } i = 1, \ldots, n,$$

must be satisfied at the minimum expected total cost, χ_k, for any state $k = 1, \ldots, n$.

The above theorem tells us that the minimum expected total cost, χ_k, for any state $k = 1, \ldots, n$, is obtained when the optimal value for P_{i0}, denoted by P^*_{i0}, is equal to $\overline{P}_{i0}$, for all $i = 1, \ldots, n$.

Proof. Consider a realisation $P \in \mathbb{P}$ with sub-matrix P_s that has a spectral radius less than 1 and $P_{\ell 0} < \overline{P}_{\ell 0}$ for some fixed $\ell \in \{1, \ldots, n\}$. We show that such a realisation can only result in a (global) minimum expected total cost if there also exists a realisation with the same expected total cost that has $P_{\ell 0} = \overline{P}_{\ell 0}$.

Consider the problem of minimising the expected total cost, χ_k, for a fixed state $k \in \{1, \ldots, n\}$ and also fix the state $\ell \in \{1, \ldots, n\}$. A given realisation $P \in \mathbb{P}$ must obey the following conditions,

$$\begin{aligned} 1 - \overline{P}_{i0} \le \sum_{j=1}^{n} P_{ij} &\le 1 - \underline{P}_{i0}, &&\text{for } i = 1, \ldots, n, \\ \underline{P}_{ij} \le P_{ij} &\le \overline{P}_{ij}, &&\text{for } i, j = 1, \ldots, n, \end{aligned}$$

where $P_{\ell 0} < \overline{P}_{\ell 0}$. Now define $P(\varepsilon)$ and therefore $P_s(\varepsilon)$, for all $\varepsilon \in [0, \overline{\varepsilon}]$, as follows:

$$\begin{aligned}
P_{\ell 0}(\varepsilon) &= P_{\ell 0} + \varepsilon, \\
P_{\ell j}(\varepsilon) &= P_{\ell j} - \varepsilon\, \alpha_j, && \text{for } j = 1, \ldots, n, \\
P_{ij}(\varepsilon) &= P_{ij}, && \text{for } i \neq \ell, j = 1, \ldots, n,
\end{aligned}$$

where

- $\overline{\varepsilon} = \overline{P}_{\ell 0} - P_{\ell 0}$ so that $\underline{P}_{\ell 0} \leq P_{\ell 0}(\varepsilon) \leq \overline{P}_{\ell 0}$ for all $\varepsilon \in [0, \overline{\varepsilon}]$, and
- choose $\alpha_j \geq 0$, for $j = 1, \ldots, n$, such that $\sum\limits_{j=1}^{n} \alpha_j = 1$ and $P_{\ell j} - \overline{\varepsilon}\alpha_j \geq \underline{P}_{\ell j}$.

We know that as we change the value of ε, not only are the probabilities $P_{\ell j}$ for $j = 0, 1, \ldots, n$ affected, but the expected total costs, χ_m, for each state $m = 1, \ldots, n$ may also change. Hence, we make the dependence explicit and denote these by $\chi_m(\varepsilon)$, for $m = 1, \ldots, n$.

Thus, the expected total costs equation is given by,

$$(I - P_s(\varepsilon))\, \boldsymbol{\chi}(\varepsilon) = \mathbf{c}.$$

Recall we are interested in showing that the minimum $\chi_k(\varepsilon)$ occurs when $P_{\ell 0}(\varepsilon)$ is at its maximum value, that is, when $\varepsilon = \overline{\varepsilon} = \overline{P}_{\ell 0} - P_{\ell 0}$. To show that this holds, we consider the sign of $\dfrac{d\,\chi_k(\varepsilon)}{d\,\varepsilon}$.

$$\begin{aligned}
\frac{d\,\chi_k(\varepsilon)}{d\,\varepsilon} &= \sum_{i=1}^{n}\sum_{j=1}^{n} \frac{\partial\,\chi_k(\varepsilon)}{\partial P_{ij}(\varepsilon)} \frac{dP_{ij}(\varepsilon)}{d\,\varepsilon}, \quad \text{by the chain rule,} \\
&= \sum_{j=1}^{n} \frac{\partial\,\chi_k(\varepsilon)}{\partial P_{\ell j}(\varepsilon)} \frac{dP_{\ell j}(\varepsilon)}{d\,\varepsilon}, \quad \text{since } \frac{dP_{ij}(\varepsilon)}{d\,\varepsilon} = 0, \text{ for } i \neq \ell \text{ and } j = 1, \ldots, n, \\
&= \sum_{j=1}^{n} \frac{\partial\,\chi_k(\varepsilon)}{\partial P_{\ell j}(\varepsilon)} \left(-\alpha_j\right), \quad \text{since } \frac{dP_{\ell j}(\varepsilon)}{d\,\varepsilon} = -\alpha_j, \text{ for } j = 1, \ldots, n, \\
&= -\sum_{j=1}^{n} \alpha_j \Big[\nabla\,\chi_k(\varepsilon)\Big]_{(\ell,j)},
\end{aligned}$$

where $\Big[\nabla \chi_k(\varepsilon)\Big]_{(\ell,j)} = \dfrac{\partial \chi_k(\varepsilon)}{\partial P_{\ell j}(\varepsilon)}$ and is the element of the gradient vector, $\nabla \chi_k(\varepsilon)$, corresponding to the $(\ell, j)^{th}$ element of the matrix $P_s(\varepsilon)$, and

$$\Big[\nabla \chi_k(\varepsilon)\Big]_{(\ell,j)} = \left[(I - P_s(\varepsilon))^{-1}\right]_{k\ell} \left[(I - P_s(\varepsilon))^{-1}\,\mathbf{c}\right]_j .$$

Hence for $\nabla \chi_k(\varepsilon)$ to exist, $(I - P_s(\varepsilon))^{-1}$ must exist. Furthermore, we know that $(I - P_s(\varepsilon))^{-1} = \displaystyle\sum_{m=0}^{\infty} P_s^m(\varepsilon)$ and $P_s(\varepsilon)$ is a matrix containing probabilities. This means that $P_s(\varepsilon)$ is a non-negative matrix and $P_s^m(\varepsilon)$ is also a non-negative matrix for all m. Thus, all entries in $(I - P_s(\varepsilon))^{-1}$ are non-negative. Since $\nabla \chi_k(\varepsilon)$ is composed of elements of $(I - P_s(\varepsilon))^{-1}$ and sums of non-negative costs multiplied with elements of $(I - P_s(\varepsilon))^{-1}$, this means all elements of the gradient vector, $\nabla \chi_k(\varepsilon)$, are non-negative.

Thus, we have that $\nabla \chi_k(\varepsilon) \geq 0$ everywhere $(I - P_s(\varepsilon))^{-1}$ exists. This means

$$\begin{aligned} \frac{d\,\chi_k(\varepsilon)}{d\,\varepsilon} &= -\sum_{j=1}^{n} \alpha_j \Big[\nabla \chi_k(\varepsilon)\Big]_{(\ell,j)} \\ &\leq 0, \end{aligned}$$

everywhere $(I - P_s(\varepsilon))^{-1}$ exists since $\alpha_j \geq 0$ and $\Big[\nabla \chi_k(\varepsilon)\Big]_{(\ell,j)} \geq 0$.

Let us first assume that $(I - P_s(\varepsilon))^{-1}$ exists for all $\varepsilon \in [0, \overline{\varepsilon}]$. This means that $\dfrac{d\,\chi_k(\varepsilon)}{d\,\varepsilon} \leq 0$ for all $\varepsilon \in [0, \overline{\varepsilon}]$. Hence, without loss of generality, we can choose to set $\varepsilon = \overline{\varepsilon}$ and hence $P_{\ell 0} = \overline{P}_{\ell 0}$ as a condition for the optimum.

However, what if $(I - P_s(\varepsilon))^{-1}$ does not exist for all $\varepsilon \in [0, \overline{\varepsilon}]$? Recall from Section 3.1 that we consider problems with an interval matrix, $\mathbb{P}$, where there exists a realisation $P \in \mathbb{P}$ with sub-matrix P_s that has a spectral radius less than 1. Let $\rho(P_s)$ denote the spectral radius of the matrix P_s. By this choice of P, we have that $\rho(P_s(0)) < 1$ and as ε increases, all the elements of $P_s(\varepsilon)$ are non-increasing. Now, Corollary 2.1.5 of Berman and Plemmons [1] states that if two matrices A and B are such that $0 \leq A \leq B$, then $\rho(A) \leq \rho(B)$, where matrix inequalities are interpreted elementwise.

Since $P_s(\varepsilon) \leq P_s(0)$ for all $\varepsilon \in [0, \overline{\varepsilon}]$, this tells us that $\rho(P_s(\varepsilon)) \leq \rho(P_s(0))$. Thus, we have that if $\rho(P_s(0)) < 1$, then $\rho(P_s(\varepsilon)) < 1$ for all $\varepsilon \in [0, \overline{\varepsilon}]$ and so $(I - P_s(\varepsilon))^{-1}$ exists for all $\varepsilon \in [0, \overline{\varepsilon}]$.

Therefore, we have shown that any realisation $P \in \mathbb{P}$, with $\rho(P_s) < 1$ and $P_{\ell 0} < \overline{P}_{\ell 0}$ for some $\ell \in \{1, \ldots, n\}$, can result in a (global) minimum expected total cost only if there exists another realisation with $P_{\ell 0} = \overline{P}_{\ell 0}$ that has the same expected total cost. □

3.3.2 Maximisation problem

Now, consider the maximisation problem and we want to prove the following theorem.

Theorem 3.3.2. *For the maximisation problem, the following constraint*

$$\sum_{j=1}^{n} P_{ij} = 1 - \underline{P}_{i0}, \qquad \text{for } P_{ij} \in \left[\underline{P}_{ij}, \overline{P}_{ij}\right] \text{ and } i = 1, \ldots, n,$$

must hold at the maximum expected total cost, χ_k, for any state $k = 1, \ldots, n$.

The above theorem tells us that the maximum expected total cost, χ_k, for any state $k = 1, \ldots, n$, is obtained when the optimal value for P_{i0}, denoted by P^*_{i0}, is equal to $\underline{P}_{i0}$, for all $i = 1, \ldots, n$.

Proof. We note that this proof follows in a similar manner to the minimisation proof with a few differences.

Consider a realisation $P \in \mathbb{P}$ with sub-matrix P_s that has a spectral radius less than 1 and $P_{\ell 0} > \underline{P}_{\ell 0}$ for some fixed $\ell \in \{1, \ldots, n\}$. We show that such a realisation can only result in a (global) maximum expected total cost if there also exists a realisation with the same expected total cost that has $P_{\ell 0} = \underline{P}_{\ell 0}$.

Consider the problem of maximising the expected total cost, χ_k, for a fixed state $k \in \{1, \ldots, n\}$ and also fix the state $\ell \in \{1, \ldots, n\}$. A given realisation $P \in \mathbb{P}$

must obey the following conditions,

$$1 - \overline{P}_{i0} \leq \sum_{j=1}^{n} P_{ij} \leq 1 - \underline{P}_{i0}, \qquad \text{for } i = 1, \dots, n,$$

$$\underline{P}_{ij} \leq P_{ij} \leq \overline{P}_{ij}, \qquad \text{for } i, j = 1, \dots, n,$$

where $P_{\ell 0} > \underline{P}_{\ell 0}$. Now define $P(\varepsilon)$ and therefore $P_s(\varepsilon)$, for all $\varepsilon \in [0, \overline{\varepsilon}]$, as follows:

$$\begin{aligned} P_{\ell 0}(\varepsilon) &= P_{\ell 0} - \varepsilon, \\ P_{\ell j}(\varepsilon) &= P_{\ell j} + \varepsilon\, \alpha_j, \qquad \text{for } j = 1, \dots, n, \\ P_{ij}(\varepsilon) &= P_{ij}, \qquad \text{for } i \neq \ell, j = 1, \dots, n, \end{aligned}$$

where

- $\overline{\varepsilon} = P_{\ell 0} - \underline{P}_{\ell 0}$ so that $\underline{P}_{\ell 0} \leq P_{\ell 0} \leq \overline{P}_{\ell 0}$ for all $\varepsilon \in [0, \overline{\varepsilon}]$, and
- choose $\alpha_j \geq 0$, for $j = 1, \dots, n$, such that $\sum_{j=1}^{n} \alpha_j = 1$ and $P_{\ell j} + \overline{\varepsilon}\alpha_j \leq \overline{P}_{\ell j}$.

As before, we make the dependence of the expected total costs, χ_m for each state $m = 1, \dots, n$, on ε explicit and denote these by $\chi_m(\varepsilon)$, for $m = 1, \dots, n$.

Thus, the expected total costs equation is given by,

$$(I - P_s(\varepsilon))\, \boldsymbol{\chi}(\varepsilon) = \mathbf{c}.$$

Recall we are interested in showing that the maximum $\chi_k(\varepsilon)$ occurs when $P_{\ell 0}(\varepsilon)$ is at its minimum value, that is, when $\varepsilon = \overline{\varepsilon} = P_{\ell 0} - \underline{P}_{\ell 0}$. To show that this holds, we consider the sign of $\dfrac{d\, \chi_k(\varepsilon)}{d\, \varepsilon}$.

$$\begin{aligned} \frac{d\, \chi_k(\varepsilon)}{d\, \varepsilon} &= \sum_{i=1}^{n} \sum_{j=1}^{n} \frac{\partial\, \chi_k(\varepsilon)}{\partial P_{ij}(\varepsilon)} \frac{dP_{ij}(\varepsilon)}{d\, \varepsilon}, \qquad \text{by the chain rule,} \\ &= \sum_{j=1}^{n} \frac{\partial\, \chi_k(\varepsilon)}{\partial P_{\ell j}(\varepsilon)} \frac{dP_{\ell j}(\varepsilon)}{d\, \varepsilon}, \qquad \text{since } \frac{dP_{ij}(\varepsilon)}{d\, \varepsilon} = 0, \text{ for } i \neq \ell \text{ and } j = 1, \dots, n, \\ &= \sum_{j=1}^{n} \frac{\partial\, \chi_k(\varepsilon)}{\partial P_{\ell j}(\varepsilon)} (\alpha_j), \qquad \text{since } \frac{dP_{\ell j}(\varepsilon)}{d\, \varepsilon} = \alpha_j, \text{ for } j = 1, \dots, n, \\ &= \sum_{j=1}^{n} \alpha_j \Big[\nabla\, \chi_k(\varepsilon)\Big]_{(\ell, j)}, \end{aligned}$$

where $\left[\nabla \chi_k(\varepsilon)\right]_{(\ell,j)} = \dfrac{\partial \chi_k(\varepsilon)}{\partial P_{\ell j}(\varepsilon)} \geq 0$ for all $\ell, j = 1, \ldots, n$, as before, everywhere $(I - P_s(\varepsilon))^{-1}$ exists. This means

$$\begin{aligned} \frac{d\,\chi_k(\varepsilon)}{d\,\varepsilon} &= \sum_{j=1}^{n} \alpha_j \left[\nabla \chi_k(\varepsilon)\right]_{\ell j} \\ &\geq 0, \end{aligned}$$

everywhere $(I - P_s(\varepsilon))^{-1}$ exists since $\alpha_j \geq 0$ and $\left[\nabla \chi_k(\varepsilon)\right]_{\ell j} \geq 0$.

Let us first assume that $(I - P_s(\varepsilon))^{-1}$ exists for all $\varepsilon \in [0, \overline{\varepsilon}]$. This tells us that $\dfrac{d\,\chi_k(\varepsilon)}{d\,\varepsilon} \geq 0$ for all $\varepsilon \in [0, \overline{\varepsilon}]$. Hence, without loss of generality, to maximise the expected total cost we would choose to set $\varepsilon = \overline{\varepsilon}$ and hence $P_{\ell 0} = \underline{P}_{\ell 0}$ as a condition for the optimum.

However, what if $(I - P_s(\varepsilon))^{-1}$ does not exist for all $\varepsilon \in [0, \overline{\varepsilon}]$? Recall that we consider problems with an interval matrix, $\mathbb{P}$, where there exists a realisation $P \in \mathbb{P}$ with sub-matrix P_s that has a spectral radius less than 1. Let $\rho(P_s)$ denote the spectral radius of the matrix P_s. By the choice of P, we have that $\rho(P_s(0)) < 1$ and as ε increases, all the elements of $P_s(\varepsilon)$ are non-decreasing.

Hence, unlike the minimisation problem, here we have $P_s(0) \leq P_s(\varepsilon)$ for all $\varepsilon \in [0, \overline{\varepsilon}]$, which tells us that $\rho(P_s(0)) \leq \rho(P_s(\varepsilon))$. Although we know that $\rho(P_s(0)) < 1$, we cannot be sure that $\rho(P_s(\varepsilon)) < 1$. Instead, we have two possibilities to consider, $\rho(P_s(\varepsilon)) < 1$ and $\rho(P_s(\varepsilon)) = 1$. Note that the maximum value of $\rho(P_s(\varepsilon))$ is 1 because P_s is a sub-matrix of P, which is a stochastic matrix, and we know that the spectral radius of a stochastic matrix can be at most 1 [1]. If $\rho(P_s(\varepsilon)) < 1$ for all $\varepsilon \in [0, \overline{\varepsilon}]$, this means that $(I - P_s(\varepsilon))^{-1}$ exists for all $\varepsilon \in [0, \overline{\varepsilon}]$ which we have already considered.

However, if $\rho(P_s(\varepsilon)) = 1$ for some $\varepsilon \in [0, \overline{\varepsilon}]$, then $(I - P_s(\varepsilon))^{-1}$ will not exist. We note that the only way for $\rho(P_s(\varepsilon))$ to equal 1 is if there is a recurrent class in $\{1, \ldots, n\}$ under $P_s(\varepsilon)$. Since we have assumed that $\rho(P_s(0)) < 1$, $P_s(0)$ cannot have a recurrent class in $\{1, \ldots, n\}$ under $P_s(0)$. Also, as $P_s(\varepsilon)$ is non-decreasing, the communication between states in $P_s(\varepsilon)$ cannot be cut and so a recurrent class in $\{1,\ldots,n\}$ under $P_s(\varepsilon)$ cannot be formed for $\varepsilon \in [0,\overline{\varepsilon})$. It is only possible to form a recurrent class in $\{1,\ldots,n\}$ under $P_s(\varepsilon)$ at $\varepsilon = \overline{\varepsilon}$ because $P_{\ell 0}(\varepsilon) = P_{\ell 0} - \varepsilon$, so the only chance for $P_{\ell 0} = 0$ is if $\varepsilon = \overline{\varepsilon}$. This tells us that $\rho(P_s(\varepsilon))$ only can equal 1 when $\varepsilon = \overline{\varepsilon}$, in which case $(I - P_s(\varepsilon))^{-1}$ does not exist only at $\varepsilon = \overline{\varepsilon}$.

However as $\varepsilon \to \overline{\varepsilon}$, we know that $\rho(P_s(\varepsilon)) \to 1$ and so $(I - P_s(\varepsilon))^{-1}$ exists for $\varepsilon \in [0,\overline{\varepsilon})$. This means $\dfrac{d\,\chi_k(\varepsilon)}{d\,\varepsilon} \geq 0$ for $\varepsilon \in [0,\overline{\varepsilon})$, which tells us that as $\varepsilon \to \overline{\varepsilon}$, $\chi_k(\varepsilon)$ increases. There are 2 possible situations, depending on the choice of k and the communication structure of the $P_s(\overline{\varepsilon})$ matrix. Either $\chi_k(\overline{\varepsilon}) = \infty$, in which case $\chi_k(\varepsilon)$ increases without bound and hence no maximum exists; or $\chi_k(\overline{\varepsilon})$ is finite.

Therefore, we have shown that any realisation $P \in \mathbb{P}$, with $\rho(P_s) < 1$ and $P_{\ell 0} > \underline{P}_{\ell 0}$ for some $\ell \in \{1,\ldots,n\}$, can result in a finite (global) maximum expected total cost only if there exists another realisation with $P_{\ell 0} = \underline{P}_{\ell 0}$ that results in the same expected total cost. Further, if no (global) maximum exists, then the maximum expected total cost with $P_{\ell 0} = \underline{P}_{\ell 0}$ will also be undefined. $\square$

Hence, we can now state both the minimisation and maximisation problems formally as follows, for $k = 1,\ldots,n$,

$$\min \chi_k = \left[(I - P_s)^{-1}\,\mathbf{c}\right]_k,$$

subject to

$$\sum_{j=1}^{n} P_{ij} = 1 - \overline{P}_{i0}, \qquad \text{for } i = 1,\ldots,n,$$

$$\underline{P}_{ij} \leq P_{ij} \leq \overline{P}_{ij}, \qquad \text{for } i,j = 1,\ldots,n,$$

and,

$$\max \chi_k = \left[(I - P_s)^{-1}\,\mathbf{c}\right]_k,$$

subject to

$$\sum_{j=1}^{n} P_{ij} = 1 - \underline{P}_{i0}, \qquad \text{for } i = 1, \ldots, n,$$

$$\underline{P}_{ij} \le P_{ij} \le \overline{P}_{ij}, \qquad \text{for } i, j = 1, \ldots, n.$$

Recall that we mentioned previously, if costs are not non-negative, then Theorems 3.3.1 and 3.3.2 do not hold. Hence, we are unable to set $P_{i0} = \overline{P}_{i0}$ for the minimisation problem and $P_{i0} = \underline{P}_{i0}$ for the maximisation problem. However, all theoretical properties of the problem and any numerical method developed can be reworked accordingly to solve problems without the non-negativity condition.

3.4 Development of an analytic solution for the mean hitting times

Before we consider an analytic solution for general costs, let us first investigate if it is possible to find an analytic solution for the simpler problem of mean hitting times, where all costs are 1.

Recall that we have been able to simplify our problem by thinking about minimising and maximising the expected total cost physically. That is, we considered what the process should be doing at each state to minimise and maximise the expected total cost. Thus, we would like to investigate if it is possible to use the same logic to determine a simple analytic solution to the mean hitting times problem. Here, we consider the minimisation problem and note that the maximisation problem follows in a similar manner. We first consider the simplest and smallest problem, which is a three state discrete-time interval Markov chain.

3.4.1 Analytic solution to a three state discrete-time interval Markov chain

Consider the minimisation problems for a three state discrete-time interval Markov chain with absorbing state 0. We seek to minimise the mean hitting times for state 1 and state 2, by solving the following minimisation problems,

$$\min \nu_1 = \left[(I - P_s)^{-1}\,\mathbf{1}\right]_1,$$

subject to

$$\begin{aligned} P_{i1}+P_{i2} &= 1 - \overline{P}_{i0}, &\qquad \text{for } i = 1,2,\\ \underline{P}_{ij} \leq P_{ij} &\leq \overline{P}_{ij}, &\qquad \text{for } i,j = 1,2, \end{aligned}$$

and

$$\min \nu_2 = \left[(I - P_s)^{-1}\,\mathbf{1}\right]_2,$$

subject to

$$\begin{aligned} P_{i1}+P_{i2} &= 1 - \overline{P}_{i0}, &\qquad \text{for } i = 1,2,\\ \underline{P}_{ij} \leq P_{ij} &\leq \overline{P}_{ij}, &\qquad \text{for } i,j = 1,2. \end{aligned}$$

We would like to be able to find an analytic solution to the above problems.

Recall the idea behind the simplification of the minimisation problem in Section 3.3 was to maximise the probability of direct absorption from any state as it corresponds to minimising the mean hitting times. If we extend this intuition further and assume, without loss of generality, that $\overline{P}_{10} > \overline{P}_{20}$, then there is a higher chance of direct absorption from state 1 than from state 2. Hence if possible, we would prefer to be in state 1 as opposed to state 2 and so we want to maximise P_{11} and P_{21} subject to the constraints,

$$\begin{aligned} P_{11} + P_{12} &= 1 - \overline{P}_{10},\\ P_{21} + P_{22} &= 1 - \overline{P}_{20}. \end{aligned}$$

Then, we allocate the remaining probabilities to P_{12} and P_{22}. We would like to prove that this intuitive idea is the analytic solution to both minimisation problems.

Let us assume, without loss of generality, that

$$\overline{P}_{10} \geq \overline{P}_{20},$$

and note that

$$(I - P_s)^{-1} = \sum_{m=0}^{\infty} P_s^m.$$

Hence, minimising the objective function $\nu_k = \left[(I - P_s)^{-1}\mathbf{1}\right]_k$ is the same as minimising the k^{th} element of the vector $\boldsymbol{\nu} = \sum_{m=0}^{\infty} P_s^m \begin{pmatrix} 1 \\ 1 \end{pmatrix}$.

If we consider $\sum_{m=0}^{\infty} P_s^m \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, we note that minimising this sum is quite challenging. Instead, we seek to minimise the relevant terms of $P_s^0 \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, $P_s^1 \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, $P_s^2 \begin{pmatrix} 1 \\ 1 \end{pmatrix}$, etc. If the argmin of each such sub-problem of $P_s^m \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ is the same, then the sum of the minima is the minimum of the sum. This means that to minimise ν_k, it would only involve minimising the k^{th} element of each term of the sum $\sum_{m=0}^{\infty} P_s^m \begin{pmatrix} 1 \\ 1 \end{pmatrix}$ as opposed to minimising the k^{th} element of the entire sum.

Define, for all $m \geq 1$,

$$f_1^{(m)} = \min_{P_s} \left[P_s \begin{pmatrix} f_1^{(m-1)} \\ f_2^{(m-1)} \end{pmatrix} \right]_1 \qquad \text{and} \qquad f_2^{(m)} = \min_{P_s} \left[P_s \begin{pmatrix} f_1^{(m-1)} \\ f_2^{(m-1)} \end{pmatrix} \right]_2,$$

where

$$\begin{pmatrix} f_1^{(0)} \\ f_2^{(0)} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \end{pmatrix}.$$

The following lemma tells us that we can minimise both $f_1^{(m)}$ and $f_2^{(m)}$ simultaneously.

Lemma 3.4.1.

$$\begin{pmatrix} f_1^{(m)} \\ f_2^{(m)} \end{pmatrix} = \min_{P_s} \left[P_s \begin{pmatrix} f_1^{(m-1)} \\ f_2^{(m-1)} \end{pmatrix} \right], \quad \textit{for all } m.$$

Proof. We know that

$$f_1^{(m)} = \min_{P_s} \left[P_s \begin{pmatrix} f_1^{(m-1)} \\ f_2^{(m-1)} \end{pmatrix} \right]_1, \; f_2^{(m)} = \min_{P_s} \left[P_s \begin{pmatrix} f_1^{(m-1)} \\ f_2^{(m-1)} \end{pmatrix} \right]_2 \text{ and } P_s = \begin{bmatrix} P_{11} & P_{12} \\ P_{21} & P_{22} \end{bmatrix}.$$

Hence, we have

$$\begin{aligned} f_1^{(m)} &= \min_{P_s} \left(P_{11} f_1^{(m-1)} + P_{12} f_2^{(m-1)} \right), \\ f_2^{(m)} &= \min_{P_s} \left(P_{21} f_1^{(m-1)} + P_{22} f_2^{(m-1)} \right). \end{aligned}$$

Now, we see that $f_1^{(m)}$ only depends on row 1 of P_s and $f_2^{(m)}$ only depends on row 2 of P_s.

Therefore,

$$\begin{pmatrix} f_1^{(m)} \\ f_2^{(m)} \end{pmatrix} = \min_{P_s} \left[P_s \begin{pmatrix} f_1^{(m-1)} \\ f_2^{(m-1)} \end{pmatrix} \right], \quad \text{for all } m.$$

□

Another thing of interest is that we know there is at least as great a chance of direct absorption from state 1 as state 2, since we have assumed without loss of generality that $\overline{P}_{10} \geq \overline{P}_{20}$, and so it seems likely that $f_1^{(m)} \leq f_2^{(m)}$ for all m. This gives us the following theorem which we shall prove by induction.

Theorem 3.4.2.

$$f_1^{(m-1)} \leq f_2^{(m-1)} \Rightarrow f_1^{(m)} \leq f_2^{(m)}, \quad \textit{for all } m.$$

We require the following lemma to help us in the proof of the above theorem.

Lemma 3.4.3.

$$f_1^{(m)} \leq f_2^{(m)} \iff (P_{11} - P_{21})f_1^{(m-1)} + (P_{12} - P_{22})f_2^{(m-1)} \leq 0.$$

Proof. We have

$$\begin{aligned} f_1^{(m)} &= P_{11}f_1^{(m-1)} + P_{12}f_2^{(m-1)}, \\ f_2^{(m)} &= P_{21}f_1^{(m-1)} + P_{22}f_2^{(m-1)}. \end{aligned}$$

Substitute these equations into $f_1^{(m)} \leq f_2^{(m)}$ to give

$$\begin{aligned} f_1^{(m)} \leq f_2^{(m)} &\iff P_{11}f_1^{(m-1)} + P_{12}f_2^{(m-1)} \leq P_{21}f_1^{(m-1)} + P_{22}f_2^{(m-1)} \\ &\iff (P_{11} - P_{21})f_1^{(m-1)} + (P_{12} - P_{22})f_2^{(m-1)} \leq 0. \end{aligned}$$

□

We now prove Theorem 3.4.2 by induction using Lemma 3.4.3 as it is an easier expression to work with.

Proof. Consider $m = 1$. We know that $f_1^{(0)} = f_2^{(0)} = 1$, so $f_1^{(0)} \leq f_2^{(0)}$ and we want to show that $f_1^{(1)} \leq f_2^{(1)}$. From Lemma 3.4.3, we know that

$$f_1^{(1)} \leq f_2^{(1)} \iff (P_{11} - P_{21})f_1^{(0)} + (P_{12} - P_{22})f_2^{(0)} \leq 0.$$

We also have

$$\begin{aligned} \begin{pmatrix} f_1^{(1)} \\ f_2^{(1)} \end{pmatrix} &= \min_{P_s} \left[P_s \begin{pmatrix} f_1^{(0)} \\ f_2^{(0)} \end{pmatrix} \right] \\ &= \min_{P_s} \left[P_s \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right]. \end{aligned}$$

This means we want to minimise $P_{11}+P_{12}$ and $P_{21}+P_{22}$ subject to the constraints on P_s.

Recall the following conditions for the minimisation problem,

$$\begin{aligned} P_{11}+P_{12} &= 1-\overline{P}_{10}, \\ P_{21}+P_{22} &= 1-\overline{P}_{20}, \end{aligned}$$

and we have assumed

$$\begin{aligned} \overline{P}_{10} &\geq \overline{P}_{20} \\ \Rightarrow 1-\overline{P}_{10} &\leq 1-\overline{P}_{20} \\ \Rightarrow P_{11}+P_{12} &\leq P_{21}+P_{22} \\ \Rightarrow (P_{11}+P_{12})-(P_{21}+P_{22}) &\leq 0. \end{aligned}$$

Now consider

$$\begin{aligned} (P_{11}-P_{21})f_1^{(0)}+(P_{12}-P_{22})f_2^{(0)} &= (P_{11}-P_{21})\times 1+(P_{12}-P_{22})\times 1 \\ &= (P_{11}+P_{12})-(P_{21}+P_{22}) \\ &\leq 0, \end{aligned}$$

which is the condition in Lemma 3.4.3 and so $f_1^{(1)} \leq f_2^{(1)}$.

Now, assume $f_1^{(i)} \leq f_2^{(i)}$ and we want to prove $f_1^{(i+1)} \leq f_2^{(i+1)}$. That is, by Lemma 3.4.3, we want to prove that $(P_{11}-P_{21})f_1^{(i)}+(P_{12}-P_{22})f_2^{(i)} \leq 0$.

Recall that

$$\begin{pmatrix} f_1^{(i+1)} \\ f_2^{(i+1)} \end{pmatrix} = \min_{P_s}\left[P_s\begin{pmatrix} f_1^{(i)} \\ f_2^{(i)} \end{pmatrix}\right].$$

From $\overline{P}_{10} \geq \overline{P}_{20}$ and above we have

$$(P_{11}-P_{21})+(P_{12}-P_{22}) \leq 0.$$

So we have the following possible cases:

1. $(P_{11} - P_{21}) \leq 0$ and $(P_{12} - P_{22}) \leq 0$,
2. $(P_{11} - P_{21}) \geq 0$ and $(P_{12} - P_{22}) \leq 0$ with $P_{11} - P_{21} \leq -(P_{12} - P_{22})$,
3. $(P_{11} - P_{21}) \leq 0$ and $(P_{12} - P_{22}) \geq 0$ with $P_{12} - P_{22} \leq -(P_{11} - P_{21})$.

Case 1: $(P_{11} - P_{21}) \leq 0$ and $(P_{12} - P_{22}) \leq 0$.

We know that

$$(P_{11} - P_{21})f_1^{(i)} + (P_{12} - P_{22})f_2^{(i)} \leq 0$$

because we are considering $(P_{11} - P_{21}) \leq 0$ and $(P_{12} - P_{22}) \leq 0$ and we also know that $f_1^{(i)} \geq 0$ and $f_2^{(i)} \geq 0$. Thus, this case holds immediately.

Case 2: $(P_{11} - P_{21}) \geq 0$ and $(P_{12} - P_{22}) \leq 0$.

We know from the induction hypothesis that $0 \leq f_1^{(i)} \leq f_2^{(i)}$. Hence, we have

$$\begin{aligned}(P_{11} - P_{21})f_1^{(i)} + (P_{12} - P_{22})f_2^{(i)} &\leq (P_{11} - P_{21})f_2^{(i)} + (P_{12} - P_{22})f_2^{(i)} \\ &= (P_{11} - P_{21} + P_{12} - P_{22})f_2^{(i)} \\ &\leq 0,\end{aligned}$$

since we know $(P_{11} - P_{21} + P_{12} - P_{22}) \leq 0$ and $f_2^{(i)} \geq 0$. So this case holds.

Case 3: $(P_{11} - P_{21}) \leq 0$ and $(P_{12} - P_{22}) \geq 0$.

Consider

$$\begin{aligned}(P_{11} - P_{21})f_1^{(i)} + (P_{12} - P_{22})f_2^{(i)} &= (P_{11} - P_{21}) \left[\min_{P_s} \left(P_{11}f_1^{(i-1)} + P_{12}f_2^{(i-1)}\right)\right] \\ &\quad + (P_{12} - P_{22}) \left[\min_{P_s} \left(P_{21}f_1^{(i-1)} + P_{22}f_2^{(i-1)}\right)\right] \\ &\leq (P_{11} - P_{21}) \left[P_{11}f_1^{(i-1)} + P_{12}f_2^{(i-1)}\right] \\ &\quad + (P_{12} - P_{22}) \left[P_{21}f_1^{(i-1)} + P_{22}f_2^{(i-1)}\right] \\ &\leq (P_{11} - P_{21}) \left[P_{11}f_1^{(i-1)} + P_{12}f_1^{(i-1)}\right] \\ &\quad + (P_{12} - P_{22}) \left[P_{21}f_1^{(i-1)} + P_{22}f_2^{(i-1)}\right] \\ &\text{since } f_1^{(i-1)} \leq f_2^{(i-1)} \text{ and } (P_{11} - P_{21})P_{12} \leq 0.\end{aligned}$$

Therefore,

$$\begin{aligned}(P_{11}-P_{21})f_1^{(i)}+(P_{12}-P_{22})f_2^{(i)} &\leq (P_{11}-P_{21})P_{11}f_1^{(i-1)}+P_{11}P_{12}f_1^{(i-1)}\\ &\quad -P_{21}P_{22}f_1^{(i-1)}+P_{22}(P_{12}-P_{22})f_2^{(i-1)}\\ &= \left[P_{11}(P_{11}-P_{21}+P_{12})-P_{21}P_{22}\right]f_1^{(i-1)}\\ &\quad +P_{22}(P_{12}-P_{22})f_2^{(i-1)}\\ &\leq \left[P_{11}P_{22}-P_{21}P_{22}\right]f_1^{(i-1)}+P_{22}(P_{12}-P_{22})f_2^{(i-1)}\\ &\text{since } P_{11}+P_{12}\leq P_{21}+P_{22} \text{ and } f_1^{(i-1)}\geq 0.\end{aligned}$$

So,

$$\begin{aligned}(P_{11}-P_{21})f_1^{(i)}+(P_{12}-P_{22})f_2^{(i)} &\leq P_{22}(P_{11}-P_{21})f_1^{(i-1)}+P_{22}(P_{12}-P_{22})f_2^{(i-1)}\\ &= P_{22}\left[(P_{11}-P_{21})f_1^{(i-1)}+(P_{12}-P_{22})f_2^{(i-1)}\right]\\ &\leq 0,\end{aligned}$$

since $P_{22}\geq 0$ and $(P_{11}-P_{21})f_1^{(i-1)}+(P_{12}-P_{22})f_2^{(i-1)}\leq 0$ by the induction hypothesis.
Thus, this case holds as well.

We have shown that for all possible cases

$$(P_{11}-P_{21})f_1^{(i)}+(P_{12}-P_{22})f_2^{(i)}\leq 0,$$

which is equivalent to showing that

$$f_1^{(i+1)}\leq f_2^{(i+1)},$$

by Lemma 3.4.3. Hence, this tells us that by assuming $f_1^{(i)}\leq f_2^{(i)}$, we have shown $f_1^{(i+1)}\leq f_2^{(i+1)}$.

Therefore, we have proved by induction that

$$f_1^{(m)}\leq f_2^{(m)},\quad \text{for all } m.$$

□

Now let us consider the matrix P_s^* that achieves

$$\begin{aligned}\begin{pmatrix} f_1^{(m)} \\ f_2^{(m)} \end{pmatrix} &= \min_{P_s} \left[P_s \begin{pmatrix} f_1^{(m-1)} \\ f_2^{(m-1)} \end{pmatrix} \right] \\ &= \min \begin{bmatrix} P_{11} f_1^{(m-1)} + P_{12} f_2^{(m-1)} \\ P_{21} f_1^{(m-1)} + P_{22} f_2^{(m-1)} \end{bmatrix}.\end{aligned}$$

Since we have proved that $f_1^{(m-1)} \leq f_2^{(m-1)}$, for us to minimise both $f_1^{(m)}$ and $f_2^{(m)}$ simultaneously, we would want to maximise P_{11} and P_{21} and place the remaining probabilities into P_{12} and P_{22}. Thus,

$$P_s^* = \begin{bmatrix} P_{11}^* & 1 - \overline{P}_{10} - P_{11}^* \\ P_{21}^* & 1 - \overline{P}_{20} - P_{21}^* \end{bmatrix}$$

where P_{11}^* and P_{21}^* are the largest possible values within their respective intervals subject to constraints.

Note that this is independent of m. That is, the choice of P_s^* remains the same for all possible values of m when we consider minimising both $f_1^{(m)}$ and $f_2^{(m)}$ for all m.

Hence, we have that

$$\begin{aligned}\begin{pmatrix} \underline{\nu}_1 \\ \underline{\nu}_2 \end{pmatrix} &= \min \sum_{m=0}^{\infty} P_s^m \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\ &= \sum_{m=0}^{\infty} \min \left[P_s^m \begin{pmatrix} 1 \\ 1 \end{pmatrix} \right] \\ &= \sum_{m=0}^{\infty} (P_s^*)^m \begin{pmatrix} 1 \\ 1 \end{pmatrix} \\ &= \sum_{m=0}^{\infty} \begin{pmatrix} f_1^{(m)} \\ f_2^{(m)} \end{pmatrix}.\end{aligned}$$

This gives us an analytic solution to the matrix P_s^* which results in the minimum mean hitting times and we can simply use

$$\begin{pmatrix} \underline{\nu}_1 \\ \underline{\nu}_2 \end{pmatrix} = \left(I - P_s^*\right)^{-1} \mathbf{1},$$

to obtain the minimum values for ν_1 and ν_2.

3.4.2 Counterexample for a four state discrete-time interval Markov chain

Now, we would like to extend our method of finding the minimum mean hitting times to a four state discrete-time interval Markov chain. Recall that to find the minimum mean hitting time for each state, we want to

$$\min \boldsymbol{\nu} = \left(I - P_s\right)^{-1} \mathbf{1},$$

subject to the following constraints

$$\begin{aligned} P_{i1} + P_{i2} + P_{i3} &= 1 - \overline{P}_{i0}, && \text{for } i = 1, 2, 3, \\ \underline{P}_{ij} \leq P_{ij} &\leq \overline{P}_{ij}, && \text{for } i, j = 1, 2, 3. \end{aligned}$$

As in the case of a three state discrete-time interval Markov chain, we know it is possible to calculate $\boldsymbol{\nu}$ as follows,

$$\boldsymbol{\nu} = \begin{pmatrix} \nu_1 \\ \nu_2 \\ \nu_3 \end{pmatrix} = \left(I - P_s\right)^{-1} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \sum_{m=0}^{\infty} P_s^m \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

As before, we minimise ν_1, ν_2 and ν_3 separately. That is, we want to minimise each element of $P_s^m \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}$ separately for all m.

First, let us assume, without loss of generality, that

$$\overline{P}_{10} \geq \overline{P}_{20} \geq \overline{P}_{30}.$$

Define, for all $m \geq 1$,

$$f_1^{(m)} = \min_{P_s} \left[P_s \begin{pmatrix} f_1^{(m-1)} \\ f_2^{(m-1)} \\ f_3^{(m-1)} \end{pmatrix} \right]_1, \quad f_2^{(m)} = \min_{P_s} \left[P_s \begin{pmatrix} f_1^{(m-1)} \\ f_2^{(m-1)} \\ f_3^{(m-1)} \end{pmatrix} \right]_2$$

$$\text{and } f_3^{(m)} = \min_{P_s} \left[P_s \begin{pmatrix} f_1^{(m-1)} \\ f_2^{(m-1)} \\ f_3^{(m-1)} \end{pmatrix} \right]_3,$$

where

$$\begin{pmatrix} f_1^{(0)} \\ f_2^{(0)} \\ f_3^{(0)} \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix}.$$

Lemma 3.4.4.

$$\begin{pmatrix} f_1^{(m)} \\ f_2^{(m)} \\ f_3^{(m)} \end{pmatrix} = \min_{P_s} \left[P_s \begin{pmatrix} f_1^{(m-1)} \\ f_2^{(m-1)} \\ f_3^{(m-1)} \end{pmatrix} \right], \quad \textit{for all } m.$$

Proof. This follows directly from the proof of Lemma 3.4.1. □

Since we have assumed that $\overline{P}_{10} \geq \overline{P}_{20} \geq \overline{P}_{30}$, there is a higher chance of direct absorption from state 1 than states 2 and 3 and from state 2 than state 3. Hence, it seems likely that $f_1^{(m)} \leq f_2^{(m)} \leq f_3^{(m)}$ for all m.

However, for the four state discrete-time interval Markov chain, this property does not hold for our conjectured optimum. Let us consider a four state discrete-time interval Markov chain with the following interval transition probability matrix,

$$\mathbb{P} = \begin{bmatrix} [1,1] & [0,0] & [0,0] & [0,0] \\ [0.3,0.35] & [0,0.1] & [0,0] & [0,1] \\ [0.2,0.3] & [0,1] & [0,1] & [0,1] \\ [0.1,0.2] & [0,0] & [0,0.3] & [0,1] \end{bmatrix}.$$

For $m = 1$ and $f^{(0)} = [1,1,1]^T$, we have

$$\begin{aligned} \begin{pmatrix} f_1^{(1)} \\ f_2^{(1)} \\ f_3^{(1)} \end{pmatrix} &= \min_{P_s} \left[P_s \begin{pmatrix} f_1^{(0)} \\ f_2^{(0)} \\ f_3^{(0)} \end{pmatrix} \right] \\ &= \min_{P_s} \left[P_s \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} \right]. \end{aligned}$$

So we would like to minimise the row sum of P_s subject to the constraints on P_s.

Our method tells us to maximise P_{11}, P_{21} and P_{31} followed by P_{12}, P_{22} and P_{32} and finally P_{13}, P_{23} and P_{33}. By applying this method, we obtain the following matrix

$$P_s^{(1)} = \begin{bmatrix} 0.1 & 0 & 0.55 \\ 0.7 & 0 & 0 \\ 0 & 0.3 & 0.5 \end{bmatrix}.$$

Thus, giving us

$$\begin{aligned} f^{(1)} &= P_s^{(1)} f^{(0)} \\ &= \begin{bmatrix} 0.1 & 0 & 0.55 \\ 0.7 & 0 & 0 \\ 0 & 0.3 & 0.5 \end{bmatrix} \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \\ &= \begin{bmatrix} 0.65 \\ 0.7 \\ 0.8 \end{bmatrix}. \end{aligned}$$

Here, we have $f_1^{(1)} \leq f_2^{(1)} \leq f_3^{(1)}$.

Now, consider $m = 2$ with $f^{(1)} = [0.65, 0.7, 0.8]^T$. Again, by minimising the row sum of P_s subject to the constraints on P_s and following our method, we obtain the matrix,

$$P_s^{(2)} = \begin{bmatrix} 0.1 & 0 & 0.55 \\ 0.7 & 0 & 0 \\ 0 & 0.3 & 0.5 \end{bmatrix}.$$

Thus, giving us

$$\begin{aligned} f^{(2)} &= P_s^{(2)} f^{(1)} \\ &= \begin{bmatrix} 0.1 & 0 & 0.55 \\ 0.7 & 0 & 0 \\ 0 & 0.3 & 0.5 \end{bmatrix} \begin{bmatrix} 0.65 \\ 0.7 \\ 0.8 \end{bmatrix} \\ &= \begin{bmatrix} 0.505 \\ 0.455 \\ 0.610 \end{bmatrix}. \end{aligned}$$

However, now we have $f_1^{(2)} \geq f_2^{(2)}$.

If we consider $m = 3$, minimise the row sum of P_s and use our method, we obtain the matrix

$$P_s^{(3)} = \begin{bmatrix} 0.1 & 0 & 0.55 \\ 0 & 0.7 & 0 \\ 0 & 0.3 & 0.5 \end{bmatrix} \neq P_s^{(2)}.$$

Hence, unlike the previous case, we find the minimum is not independent of the step m, and so we are unable to bring the minimum into the sum, as P_s is dependent on m. So the method of proof we used for the previous case is not going to work. By using our numerical approach, we found that the optimum value of $\boldsymbol{\nu}$ is obtained when $P_s^{(3)}$ is used which is not our conjectured choice of $P_s = P_s^{(2)}$. Thus, we have a counter-example for the four state discrete-time interval Markov chain.

We have been able to develop an analytic method for calculating the tightest possible intervals which contain the true solution to our problem for a three state discrete-time interval Markov chain. However, when we include an extra state, we find a counter-example which shows that our method fails to find the bounds for mean hitting times. It appears that analytic results of performance measures we consider cannot be obtained for general large interval Markov chains.

Hence, we now seek to identify analytic properties which may assist in determining the optimal solution. One such useful property would be the form of the optimal solution to the optimisation problems, which we now investigate using Markov decision processes.

3.5 Markov decision processes

Recall that we have laid down the framework for Markov decision processes (MDPs) in Section 2.5. In this section, we would like to represent our optimisation problem as an MDP problem and as such, we require a mapping from one to the other.

3.5.1 Mapping

Without loss of generality, we consider the maximisation problem and note that equivalent theorems hold for the minimisation problem. To obtain upper bounds on the expected total costs, we solve maximisation problems of the form,

$$\max \chi_k = \left[(I - P_s)^{-1}\,\mathbf{c}\right]_k,$$

subject to

$$\begin{aligned} \sum_{j=1}^{n} P_{ij} &= 1 - \underline{P}_{i0}, && \text{for } i = 1, \ldots, n, \\ \underline{P}_{ij} \leq P_{ij} &\leq \overline{P}_{ij}, && \text{for } i, j = 1, \ldots, n. \end{aligned}$$

Lemma 3.5.1. *Our maximisation problem is a Markov decision process restricted to only consider Markovian decision rules and stationary policies.*

Proof. We know from the Section 2.5 that the following terms describe a Markov decision process: decision times, states, actions, rewards and transition probabilities. Hence, we need to be able to show how we can describe our maximisation problem using these terms.

Recall that we are considering a discrete-time interval Markov chain with $n + 1$ states where state 0 is an absorbing state. This discrete-time interval Markov chain is the underlying Markov chain of the maximisation problem. Hence, the set of transient states of the discrete-time interval Markov chain correspond with the rows of the P_s matrix in the maximisation problem and also coincide with the set of states in the underlying discrete-time Markov chain in the MDP framework. Since we have n transient states in the underlying discrete-time Markov chain of the maximisation problem, this means we have a total of n states in our MDP problem.

Consider the feasible region of the maximisation problem. We note that the constraints are row-based as they only involve row sums and bounds on the P_s matrix.

That is, there is no interaction between the rows of the P_s matrix when we deal with the constraints on the problem. Hence, in the following, we will make extensive use of the feasible region for each row and introduce the notation F_i to describe the feasible region for row i, where $i = 1, \ldots, n$. Since F_i describes the feasible region for row i, it also represents the possible vectors for the i^{th} row of the P_s matrix.

We know that F_i has a finite number of vertices as there are two possible choices for each decision variable, P_{ij}, in row i, either the lower bound, $\underline{P}_{ij}$, or the upper bound, $\overline{P}_{ij}$. Thus, for a row with n decision variables, there is an upper bound of 2^n possible vertices.

Furthermore, since the only constraints on the problem are upper and lower bounds as well as linear constraints, these form a convex hull. We know that a convex hull is composed of a set of vertices as well as points within the hull which are a convex combination of the vertices. Hence, F_i is composed of a set of vertices as well as points which can be written as a convex combination of vertices.

Now, we choose to let each vertex in F_i correspond to an action of the MDP when in state i. Thus, in the MDP framework, the feasible region F_i can be recovered by taking convex combinations of actions as they correspond to convex combinations of vertices.

Other terms needing to be mapped are decision times, rewards and transition probabilities. Recall that we move from one state to another at each time step of the underlying discrete-time Markov chain of our maximisation problem. Hence, a time step in the underlying discrete-time Markov chain maps to a decision time in the MDP problem as a decision is made at each time step to decide which state to move into next. We note that we have an infinite-horizon Markov decision process as we seek to maximise the expected total cost incurred before the process reaches the absorbing state, conditional on starting in some other state. Hence, this does not correspond to maximising our objective in the MDP problem up to a certain time step. Instead, we allow the process to continue until absorption which means that the time steps of our problem are infinite. As such, our problem is classified as an infinite-horizon Markov decision process.

The rewards are c_i for state i, regardless of the choice of action. Whenever a decision is made, the discrete-time Markov chain moves to some state $i \in \{0, 1, \ldots, n\}$ and the total cost increases by c_i (where $c_0 = 0$) with each move to state i. Lastly, we consider the transition probabilities. Recall that the transition probabilities of an MDP problem describe the probability of moving from one state to another given an action has been chosen. We have also previously defined each action in state i for $i = 1, \ldots, n$ to be a vertex of the feasible region F_i. Thus, each action a has an associated (possibly dishonest) probability distribution vector, $\mathbf{P}_i^{(a)}$, with elements P_{ij} corresponding to the values at the vertex. This vector contains the transition probabilities of moving from the current state i to all states of the underlying discrete-time Markov chain given that action a has been chosen. When an action a is chosen in state i, the corresponding probability distribution vector, $\mathbf{P}_i^{(a)}$, is inserted into the matrix, P_s. Hence, when considering all states $i = 1, \ldots, n$, once an action has been chosen for each state, the probability distribution vectors, $\mathbf{P}_i^{(a)}$, for each state form the P_s matrix for the maximisation problem.

Finally, as our maximisation problem fundamentally involves a discrete-time Markov chain, which state the process moves to next can only depend on the current state of the process. Thus, we know that the decision rule must be Markovian. We also know that due to the structure of our maximisation problem, we are guaranteed to have a stationary policy as the optimal P_s matrix remains constant over time since our problem is a one-sample time-homogeneous interval Markov chain. This means that the choice of decision rule is independent of time which describes a stationary policy. Hence, our maximisation problem is a Markov decision process restricted to only consider Markovian decision rules and stationary policies. □

We can now consider theorems for MDPs with stationary policies and Markovian decision rules to help us prove that the form of the optimal solution is as described in Theorem 3.5.2.

Theorem 3.5.2. *There exists an optimal solution of the maximisation problem where row i of the optimal matrix, P_s^*, represents a vertex of F_i for all $i = 1, \ldots, n$.*

Proof. By Lemma 3.5.1, we know that we are able to consider the maximisation problem as a Markov decision process with Markovian decision rules and a stationary policy. Hence, to show that the optimal solution occurs at a vertex, we need only show that there is no extra benefit to having randomised decision rules as opposed to deterministic decision rules. The following proposition gives us this.

Proposition 3.5.3. *For all $\mathbf{v} \in V$,*

$$\sup_{d \in D^{MD}} \{\mathbf{r}_d + P_d \mathbf{v}\} = \sup_{d \in D^{MR}} \{\mathbf{r}_d + P_d \mathbf{v}\},$$

where

- *V is the set of bounded real valued functions on S with componentwise partial order and supremum norm. Since we only deal with finite S, for each $\mathbf{v} \in V$, the supremum norm gives us the largest element in the vector $\mathbf{v}$. Also, a componentwise partial order means that for $\mathbf{u} \in V$ and $\mathbf{v} \in V$, $\mathbf{u} \geq \mathbf{v}$ if $\mathbf{u}(s) \geq \mathbf{v}(s)$ for all $s \in S$,*
- *D^{MD} is the set of decision rules which are Markovian and deterministic,*
- *D^{MR} is the set of decision rules which are Markovian and randomised,*
- *$\mathbf{r}_d$ is the vector of rewards obtained for the choice of decision rule, d,*
- *P_d is the transition probability matrix corresponding to the decision rule d.*

Note that this proposition is based on Proposition 6.2.1. of Puterman [18] restricted to the case without discounting, that is, when $\lambda = 1$ and depends on the following assumptions.

Assumption 3.5.1 (Assumption 6.0.1. of Puterman [18])**.** *Stationary rewards and transition probabilities; $r(s, a)$ and $p(j|s, a)$ do not vary from one time step to another.*

Assumption 3.5.2 (Assumption 6.0.2. of Puterman [18])**.**
Bounded rewards; $|r(s, a)| \leq M < \infty$ for all $a \in A_s$ and $s \in S$.

Assumption 3.5.3 (Assumption 6.0.4. of Puterman [18])**.**
Discrete state space; S is finite or countable.

We know that these assumptions hold based on the construction of our maximisation problem.

From Section 6.2 in Puterman [18] and knowing that our action set, A_s, is finite for all states s, we see that the following system of equations are the *optimality equations*,

$$v(s) = \max_{a \in A_s} \left\{ r(s, a) + \sum_{j \in S} p(j|s, a) v(j) \right\}, \qquad \text{for all } s \in S.$$

The above system of equations can be expressed in vector notation as,

$$\mathbf{v} = \max_{d \in D^{MD}} \{\mathbf{r}_d + P_d \mathbf{v}\}.$$

They correspond with our maximisation problem as we calculate the expected total costs using

$$\boldsymbol{\chi} = \mathbf{c} + P_s \boldsymbol{\chi}.$$

In equations, $\mathbf{r}_d = \mathbf{c}$ and P_d corresponds to the matrix P_s. When maximising, we try to find the optimal P_s matrix which results in the largest value for $\boldsymbol{\chi}$. Thus, in the MDP language, we are trying to find the optimal decision rule which maximises the expected total costs.

Proposition 3.5.3 compares Markovian deterministic rules with Markovian randomised rules and tells us that there is nothing to be gained from having randomised decision rules as opposed to deterministic decision rules. Thus, the optimal solution to the optimisation problem is attained when we have decision rules which are deterministic and Markovian.

Recall from Definitions 2.5.3 and 2.5.6 that a deterministic decision rule corresponds to choosing actions deterministically which means that a single action is chosen in each state with probability 1. Since each action in state i of the MDP problem corresponds to a vertex in the feasible region F_i, for $i = 1, \ldots, n$, this means deterministic decision rules ensure that each row of the optimal matrix will correspond to a vertex of F_i. Thus, we have shown that there exists an optimal solution of the maximisation problem where row i of the optimal matrix, P_s^*, represents a vertex of F_i for all $i = 1, \ldots, n$. $\square$

3.6 Continuous-time interval Markov chains

In this section, we consider *continuous-time interval Markov chains* and investigate the use of uniformisation to exploit the theoretical properties obtained for discrete-time interval Markov chains.

First, consider a standard, non-interval continuous-time Markov chain, $X(t)$, with a discrete state space $S = \{0, 1, \ldots, n\}$, where state 0 is an absorbing state, and generator matrix,

$$Q = \begin{bmatrix} 0 & 0 & \cdots & 0 \\ q_{10} & q_{11} & \cdots & q_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ q_{n0} & q_{n1} & \cdots & q_{nn} \end{bmatrix}$$

where

- $q_{ij} \geq 0$, for all $i \in S$ and $i \neq j$, is the rate of moving from state i to state j,
- $0 \leq q_i < \infty$ for all $i \in S$ where $q_i = -q_{ii} = \sum_{j \neq i} q_{ij}$ is the total rate of leaving state i, and
- $\sum_{j \in S} q_{ij} = 0$ (row sum constraint).

As before with discrete-time Markov chains, where we accounted for the uncertainty of the transition probabilities by using an interval transition probability matrix, we do the same for continuous-time Markov chains.

Consider a continuous-time interval Markov chain, $X(t)$, with $n+1$ states, $S = \{0, 1, \ldots, n\}$, where state 0 is an absorbing state and interval generator matrix,

$$\mathbb{Q} = \begin{bmatrix} [0,0] & [0,0] & \cdots & [0,0] \\ \left[\underline{q}_{10}, \overline{q}_{10}\right] & \left[\underline{q}_{11}, \overline{q}_{11}\right] & \cdots & \left[\underline{q}_{1n}, \overline{q}_{1n}\right] \\ \vdots & \vdots & \ddots & \vdots \\ \left[\underline{q}_{n0}, \overline{q}_{n0}\right] & \left[\underline{q}_{n1}, \overline{q}_{n1}\right] & \cdots & \left[\underline{q}_{nn}, \overline{q}_{nn}\right] \end{bmatrix},$$

where

$$\begin{aligned} &0 \leq \underline{q}_{ij} \leq \overline{q}_{ij}, && \text{for all } i = 1, \ldots, n \text{ and } j \neq i, \\ &-\infty < \underline{q}_{ii} \leq \overline{q}_{ii} \leq 0, && \text{for all } i = 1, \ldots, n, \text{ and} \\ &\sum_{j \in S} q_{ij} = 0, && \text{for } q_{ij} \in \left[\underline{q}_{ij}, \overline{q}_{ij}\right] \text{ and } i = 1, \ldots, n. \text{ (row sum constraint)} \end{aligned}$$

To assist us with notation later, we also specify an interval on q_i which is given by $\left[\underline{q}_i, \overline{q}_i\right] = -\left[\underline{q}_{ii}, \overline{q}_{ii}\right] = \left[\sum_{j\neq i} \underline{q}_{ij}, \sum_{j\neq i} \overline{q}_{ij}\right]$. Like discrete-time interval Markov chains, we are interested in obtaining an interval expected total cost for each state of an interval continuous-time Markov chain.

Recall that χ_k is the expected total cost incurred before the process is absorbed into state 0, conditional on starting in state k. From Section 2.1.2, we know that the expected total cost vector, $\boldsymbol{\chi}$, composed of the elements χ_k, for $k = 1, \ldots, n$, can be calculated using

$$\boldsymbol{\chi} = -Q_s^{-1}\mathbf{c},$$

where Q_s is the sub-matrix of the generator matrix Q and $\mathbf{c}$ is the column vector containing the cost per unit time of being in each state. However, to ensure that Q_s^{-1} is well defined, we only require Q_s to have a spectral radius less than zero as it is finite-dimensional. Since we have already dealt with this problem for discrete-time Markov chains in Section 3.1, instead of evaluating the above expression for continuous-time Markov chains, we borrow the theoretical properties obtained previously by considering uniformisation. In the following, we consider applying uniformisation to continuous-time interval Markov chains.

3.6.1 Uniformisation for continuous-time interval Markov chains

Recall from Section 2.1.2 that uniformisation allows us to transform a continuous-time Markov chain to a discrete-time Markov chain without losing any information and we also discussed the transformation used to obtain a transition probability matrix from a generator matrix. Here, we extend this transformation to interval generator matrices and interval transition probability matrices as well as extending definitions for uniformisation to continuous-time interval Markov chains.

A continuous-time interval Markov chain, $X(t)$, is *uniformisable* if

$$\sup_i \left(\overline{q}_i\right) < \infty.$$

Then, assuming that we have a uniformisable continuous-time interval Markov chain, $X(t)$, with interval generator matrix, $\mathbb{Q}$, the *uniformisation constant* is any

$$u \geq \overline{q} = \sup_i \left(\overline{q}_i\right).$$

Since S is finite, this means $\overline{q}_i < \infty$ for all $i = 1, \ldots, n$ and so $X(t)$ is uniformisable. Hence, we have that $u \geq \overline{q} = \max_i \left(\overline{q}_i\right)$. Then any such u allows us to define the associated discrete-time interval Markov chain of the continuous-time interval Markov chain, $X(t)$.

Let X_m^u be the associated discrete-time interval Markov chain of the continuous-time interval Markov chain, $X(t)$, with interval transition probability matrix $\mathbb{P}_u$, given by

$$\mathbb{P}_u = I + \frac{1}{u}\mathbb{Q},$$

for any $u \geq \overline{q}$.

Since we have required $-\infty < \underline{q}_{ii} \leq \overline{q}_{ii} \leq 0$ which means that $0 \leq \underline{q}_i \leq \overline{q}_i < \infty$ for all $i = 1, \ldots, n$ in our definition of a continuous-time interval Markov chain, we know that the continuous-time interval Markov chains we are interested in are uniformisable. Thus, we can obtain a uniformisation constant, u, and apply the transformation $\mathbb{P}_u = I + \frac{1}{u}\mathbb{Q}$ to give us an interval transition probability matrix for the associated discrete-time interval Markov chain.

Hence, we now have a discrete-time interval Markov chain with interval transition probability matrix,

$$\mathbb{P}_u = \begin{bmatrix} [1,1] & [0,0] & \cdots & \cdots & \cdots & [0,0] \\ [\underline{P}_{10}, \overline{P}_{10}] & [\underline{P}_{11}, \overline{P}_{11}] & \cdots & \cdots & \cdots & [\underline{P}_{1n}, \overline{P}_{1n}] \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ [\underline{P}_{i0}, \overline{P}_{i0}] & [\underline{P}_{i1}, \overline{P}_{i1}] & \cdots & [\underline{P}_{ij}, \overline{P}_{ij}] & \cdots & [\underline{P}_{in}, \overline{P}_{in}] \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ [\underline{P}_{n0}, \overline{P}_{n0}] & [\underline{P}_{n1}, \overline{P}_{n1}] & \cdots & \cdots & \cdots & [\underline{P}_{nn}, \overline{P}_{nn}] \end{bmatrix}$$

where

$$[\underline{P}_{ij}, \overline{P}_{ij}] = \left[\frac{\underline{q}_{ij}}{u}, \frac{\overline{q}_{ij}}{u}\right], \qquad \text{for } i = i, \ldots, n \text{ and } j \neq i,$$
$$[\underline{P}_{ii}, \overline{P}_{ii}] = \left[1 - \frac{\overline{q}_i}{u}, 1 - \frac{\underline{q}_i}{u}\right], \qquad \text{for } i = i, \ldots, n.$$

Since we have a discrete-time interval Markov chain of the same form as before (Section 3.1), all theoretical properties we have developed now apply for this case and the numerical method (Chapter 4) used to solve for the expected total cost, χ_k, for each state $k = 1, \ldots, n$, of the discrete-time interval Markov chain can also be used here. However, this gives us the expected total cost for the associated discrete-time interval Markov chain instead of the original continuous-time interval Markov chain we are interested in. Thus, we need an appropriate transformation to obtain results for the continuous-time interval Markov chain.

3.6.2 Transforming discrete-time expected total costs to continuous-time

Let χ_k be the expected total cost incurred before the process is absorbed into state 0, conditional on starting in state k for the continuous-time interval Markov chain. Then, let χ_k^u be the expected total cost incurred before the process is absorbed into state 0, conditional on starting in state k for the associated discrete-time interval Markov chain with uniformisation constant, u.

Recall that we have applied uniformisation to transform the continuous-time interval Markov chain to its associated discrete-time Markov chain. Then solved for the expected total costs, χ_k^u, for states $k = 1, \ldots, n$, for the associated discrete-time Markov chain. The following theorem gives us the transformation of the expected total costs for the associated discrete-time interval Markov chain, χ_k^u, to obtain the expected total costs for the original continuous-time interval Markov chain, χ_k.

Theorem 3.6.1. *For $k = 1, \ldots, n$, the expected total cost χ_k of a continuous-time interval Markov chain with costs per unit time c_i (for $i = 1, \ldots n$) can be obtained from the expected total cost χ_k^u of its associated discrete-time interval Markov chain with costs per visit c_i (for $i = 1, \ldots, n$) by applying the following transformation,*

$$\chi_k = \frac{\chi_k^u}{u},$$

where u is the uniformisation constant.

Proof. Let $X(t)$ be a continuous-time interval Markov chain and X_m^u be its associated discrete-time interval Markov chain with uniformisation constant u.

For state $k = 1, \ldots, n$, we define the following terms. Let

- Y_ℓ be the random variable describing the amount of time the continuous-time interval Markov chain spent in the ℓ^{th} step, where $Y_\ell \sim Exp(u)$ due to uniformisation,
- A_i be the random variable describing the amount of time the continuous-time interval Markov chain spends in state i before absorption,
- N_i is the random variable describing the number of visits to state i before absorption. Note that a visit in this context includes transitions from state i to itself, and
- A_{ij} is the random variable describing the amount of time spent in state i on the j^{th} visit.

Thus, the total cost incurred before the process is absorbed into state 0 for the continuous-time interval Markov chain is given by

$$K = \sum_{i=1}^{n} c_i A_i.$$

Conditioning on the initial state k and taking expectation gives,

$$\begin{aligned}
\chi_k &= E\left[K|X(0)=k\right] \\
&= E\left[\sum_{i=1}^{n} c_i A_i \middle| X(0)=k\right].
\end{aligned}$$

Now, let

$$A_i = \sum_{j=1}^{N_i} A_{ij}.$$

Hence,

$$\begin{aligned}
\chi_k &= E\left[\sum_{i=1}^{n} c_i \sum_{j=1}^{N_i} A_{ij} \middle| X(0)=k\right] \\
&= \sum_{i=1}^{n} E\left[c_i \sum_{j=1}^{N_i} A_{ij} \middle| X(0)=k\right] \\
&= \sum_{i=1}^{n} E_{N_i}\left[E\left[c_i \sum_{j=1}^{N_i} A_{ij} \middle| X(0)=k, N_i\right] \middle| X_0^u = k\right] \\
&= \sum_{i=1}^{n} E_{N_i}\left[c_i \sum_{j=1}^{N_i} E\left[A_{ij}|X(0)=k, N_i\right] \middle| X_0^u = k\right] \\
&= \sum_{i=1}^{n} E_{N_i}\left[c_i \sum_{j=1}^{N_i} E\left[Y_\ell\right] \middle| X_0^u = k\right] \\
&= E\left[Y_\ell\right] \sum_{i=1}^{n} E_{N_i}\left[c_i N_i | X_0^u = k\right] \\
&= \frac{\chi_k^u}{u},
\end{aligned}$$

since $\sum_{i=1}^{n} E_{N_i}\left[c_i N_i | X_0^u = k\right] = \chi_k^u$ and $Y_\ell \sim Exp(u)$ for all ℓ which means $E\left[Y_\ell\right] = \frac{1}{u}$.

□

In this section, we have obtained the expected total cost for a continuous-time interval Markov chain by considering uniformisation and the required transformation to obtain our desired results. Figure 3.6.1 provides a graphical representation of the steps we use to obtain the expected total cost for a continuous-time interval Markov chain using uniformisation.

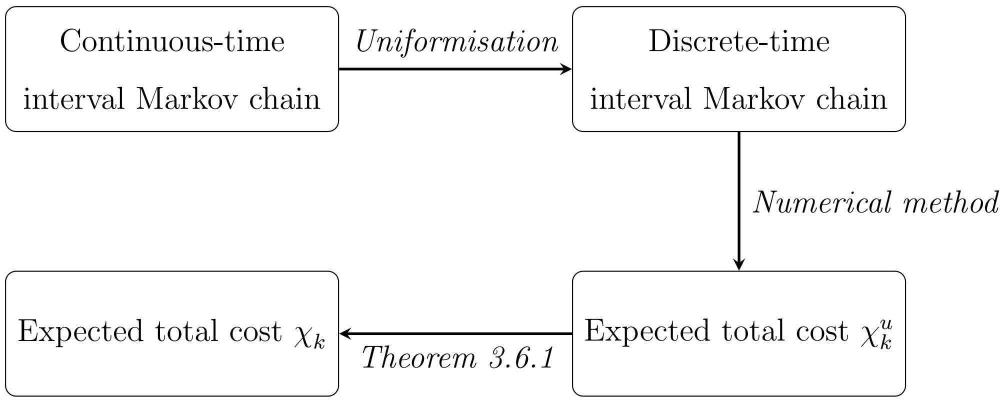
Continuous-time interval Markov chain
Uniformisation
Discrete-time interval Markov chain
Numerical method
Expected total cost χ_k^u
Theorem 3.6.1
Expected total cost χ_k

Figure 3.6.1: Graphical representation of steps used to obtain expected total cost for a continuous-time interval Markov chain.

In Sections 3.3 and 3.4, we have been able to reduce the number of decision variables in our optimisation problems. Although we were not able to determine an analytic solution for general interval Markov chains, the intuitive idea behind the analytic solution for the three state interval Markov chain can be used as a starting point for an iterative numerical optimisation routine. We explain in detail how this idea is used in Chapter 4. Section 3.5 also provides us with a useful analytic property which we exploit when finding and checking the solution of the optimisation problem obtained using numerical techniques. Lastly, Section 3.6 explored continuous-time interval Markov chains and obtained a method for calculating the expected total cost using uniformisation and our discrete-time interval Markov chain results.

Chapter 4

Numerical Method

Here, we describe the numerical method to solve the minimisation problem. Note that a similar method is used to solve the maximisation problem and any differences in the method of solving are explicitly considered in the relevant sections.

We know that an optimisation problem involves finding an optimal value of a given function on a region which may or may not be constrained. Thus, a minimisation problem involves finding a minimum value for a given objective function subject to constraints (if they exist). Recall the minimisation problem we are interested in solving is

$$\min \chi_k = \left[(I - P_s)^{-1} \mathbf{c} \right]_k ,$$

subject to

$$\begin{aligned} &\sum_{j=1}^{n} P_{ij} = 1 - \overline{P}_{i0}, && \text{for } i = 1, \dots, n, \\ &\underline{P}_{ij} \le P_{ij} \le \overline{P}_{ij}, && \text{for } i, j = 1, \dots, n. \end{aligned} \tag{4.0.1}$$

Here, we have a non-linear objective function with linear constraints and bounds on the decision variables. Hence, our problem fits into the framework of MATLAB's `fmincon` function which solves minimisation problems of the form

$$\min_{\mathbf{x}} f(\mathbf{x})$$

subject to

$$\begin{aligned} k(\mathbf{x}) &\leq 0, \\ keq(\mathbf{x}) &= 0, \\ A\mathbf{x} &\leq b, \\ Aeq\mathbf{x} &= beq, \\ lb \leq \mathbf{x} &\leq ub, \end{aligned}$$

where $f(\mathbf{x})$ represents a linear or non-linear objective function, $k(\mathbf{x})$ and $keq(\mathbf{x})$ are non-linear functions and $A\mathbf{x}$ and $Aeq\mathbf{x}$ are linear functions.

However, before we can solve the minimisation problem using `fmincon`, some pre-processing of the intervals for the problem is required.

4.1 Pre-processing

The idea behind the need to pre-process the intervals for the minimisation problem is to ensure that the true solution is always contained within the intervals. Another reason for implementing pre-processing is to reduce the number of decision variables, which shrinks the size of the feasible region and reduces the search space for the optimisation algorithm; hence, making the problem easier to solve. The rest of this section is dedicated to explaining the following concepts: *degeneracy*, *outward rounding* and *coherence*, which form the basis for the pre-processing. We will also discuss how these concepts are implemented for our problem. Note that in this pre-processing section, we consider the full interval transition probability matrix $\mathbb{P}$.

4.1.1 Degeneracy

Before we consider how degeneracy is dealt with in our problem, we have to understand what it means for an interval to be degenerate.

Definition 4.1.1. *[14] An interval, $X = \left[\underline{X}, \overline{X}\right]$, is* degenerate *if $\underline{X} = \overline{X}$.*

That is, a degenerate interval is an interval with zero width.

Thus, in taking degeneracy into account, we simplify our problem by removing constant variables from the decision variables and hence make it easier to solve. In the following, we detail the identification of degenerate intervals and leave the discussion of the simplification of our problem to Section 4.3.5.

Since we are dealing with floating point numbers, we are limited by numerical precision. This makes it unlikely for the lower and upper bounds of an interval to be exactly equal. Hence, we relax the definition of a degenerate interval to allow for numerical precision and define an interval X to be *numerically degenerate* if the width of the interval is within some acceptable tolerance of zero. For our problem, we have specified this tolerance to be $TolDegenerate$. Thus, an interval is numerically degenerate if $\left(\overline{X} - \underline{X}\right) \leq TolDegenerate$. The choice of default value for $TolDegenerate = 10^{-8}$ and will be discussed in greater detail in Section 4.4. However, we note here that the default value chosen is significantly greater than machine epsilon ($2^{-52} \approx 2 \times 10^{-16}$) because optimisation routines have trouble adequately exploring intervals which are too narrow. Now we provide the full procedure used to identify degenerate intervals in our problem and detail the method used to deal with them. We also consider the consequences of this in our problem.

Procedure used to implement degeneracy:

- Identify degenerate intervals and set their bounds to be the midpoints of the intervals. That is,

$$new\underline{P}_{ij} = new\overline{P}_{ij} = P_{ij}^M, \qquad \text{where } P_{ij}^M = \frac{\underline{P}_{ij} + \overline{P}_{ij}}{2}.$$

- We need to ensure that any changes made to degenerate intervals still result in a valid interval matrix. That is, there still exist realisations of the interval matrix with row sums within some tolerance of 1. We have defined this tolerance to be $TolEqual$ and choose the default value to be $TolEqual = 10^{-13}$. Note that reasons for this choice of default value will be discussed in detail in Section 4.4.
- If we have a row of degenerate intervals, we must ensure that the sum of these degenerate intervals is within $TolEqual$ of 1. That is, we require

  $$1 - TolEqual \leq \sum_j P_{ij}^M \leq 1 + TolEqual.$$

  We note that shifting the upper and lower bounds of degenerate intervals to their midpoints can result in two ways of violating this condition.

  1. The row sum of degenerate intervals are too small to result in a row sum within $TolEqual$ of 1.
  2. The row sum of degenerate intervals exceeds $(1 + TolEqual)$.

  Hence, we need to shift these degenerate intervals away from their midpoint to allow their sum to be within $TolEqual$ of 1. To do so, we allocate the amount needed for the row sum to be within $TolEqual$ of 1 proportionally to all intervals with non-zero width. That is, we perform the following modification to the degenerate intervals.

  Let Δ be the amount needed to bring the row sum to 1. Hence,

  $$\Delta = \sum_j P_{ij}^M - 1.$$

Then, if $|\Delta| > TolEqual$, we allocate Δ, as follows, to bring the sum of degenerate intervals to within $TolEqual$ of 1.

$$newP_{ij} = P_{ij}^{M} - \Delta \times \frac{\overline{P}_{ij} - \underline{P}_{ij}}{\sum_{j}(\overline{P}_{ij} - \underline{P}_{ij})}, \qquad \text{for } j = 0, \ldots, n.$$

This takes care of both cases 1 and 2 above: If $\Delta > TolEqual$, we need to subtract the extra amount from each degenerate interval with non-zero width; on the other hand, if $\Delta < -TolEqual$, we need to add the extra amount to each degenerate interval with non-zero width. Since Δ is negative for the second case, we have the effect of adding the extra amount.

- If not all intervals in a row are degenerate, we need to ensure that for each row, the lower bounds sum to a value less than or equal to 1 and the upper bounds sum to a value greater than or equal to 1. Let D be the set of degenerate intervals. Then, we require the following to hold,

$$\sum_{j \notin D} \underline{P}_{ij} + \sum_{j \in D} P_{ij}^{M} \leq 1,$$
$$\sum_{j \notin D} \overline{P}_{ij} + \sum_{j \in D} P_{ij}^{M} \geq 1.$$

Let us consider the condition on the upper bounds. If this condition is not satisfied, we allocate the difference between the row sum and 1 proportionally to each non-zero degenerate interval. This increases the value of each degenerate interval by some amount which enables the row sum to be within $TolEqual$ of 1.

Let $\Delta = \left| \sum_{j \notin D} \overline{P}_{ij} + \sum_{j \in D} P_{ij}^{M} - 1 \right|$.

If $\Delta > TolEqual$, we allocate Δ to the non-zero degenerate intervals using the following method to increase the row sum to at least 1.

$$newP_{ij} = P_{ij}^M + \Delta \times \frac{\overline{P}_{ij} - \underline{P}_{ij}}{\sum_j (\overline{P}_{ij} - \underline{P}_{ij})}, \qquad \text{for } j \in D.$$

If the condition on the lower bounds is violated, we use the same method as for the upper bounds with a few modifications. Here, we replace the upper bounds in Δ with lower bounds and we subtract from P_{ij}^M for $j \in D$, to decrease the row sum.

4.1.2 Outward rounding

Another important concept we need to consider is the idea of *outward rounding*. *Outward rounding* is a procedure performed on intervals where the left endpoint is moved to the left (on the number line) and the right endpoint is moved to the right [14]. The idea behind performing outward rounding is to ensure that intervals used in further calculations always contain the true solution. This is necessary due to the representation of floating point numbers on a computer and in MATLAB.

There are a variety of ways to represent a floating point number in MATLAB. Since the default format used in MATLAB is the double precision format, this will be the one of focus to us. The double precision format uses the IEEE 754 standard for double precision. This means that any number stored in double precision in MATLAB uses 64 bits where 1 bit is used for the sign of the number, 11 bits for the exponent and 52 bits for the mantissa. Note that examples used to explain floating point numbers and outward rounding will be expressed in hexadecimal. That is, instead of writing out the full binary representation in base 2, we can use a hexadecimal representation in base 16 which groups 4 bits into one hexadecimal digit. For example,

$$0111 = 7,$$
$$1011 = b,$$

where the expression on the left is in binary and on the right is hexadecimal.

Without loss of generality, we demonstrate the double precision format on the number $\frac{1}{10}$ which is represented in hexadecimal [13] as

$$\frac{1}{10} = 3fb999999999999a.$$

Recall that a number is stored in double precision using 64 bits with the first bit used for the sign, the next 11 bits used for the exponent and the remaining 52 bits used for the mantissa. Hence, the hexadecimal representation can be split up into the separate components as

$$\overbrace{3fb}^{\text{sign and exponent}} \qquad \overbrace{999999999999a}^{\text{mantissa}},$$

and the separation of the sign and exponent is given in binary as,

$$3fb = \overbrace{\ 0\ }^{\text{sign}}\overbrace{01111111011}^{\text{exponent}}.$$

First, consider the sign. Since the sign is represented by the first bit, this bit is 0 for positive numbers and 1 for negative numbers. For example,

$$\frac{1}{10} = 3fb999999999999a \qquad \text{whilst} \qquad -\frac{1}{10} = bfb999999999999a.$$

The only difference between the two is the first bit in the first hexadecimal number as $3 = 0011$ and $b = 1011$.

Now, consider the exponent which is represented by the next 11 bits. A change in the last bit of the exponent represents a doubling of the floating point number.

For example,

$$\begin{aligned}\frac{1}{20} &= 3fa999999999999a,\\ \frac{1}{10} &= 3fb999999999999a,\\ \frac{1}{5} &= 3fc999999999999a.\end{aligned}$$

In these examples, we see that the exponent has increased by 1 from $3fa$ to $3fb$ then $3fc$ and with each increase, the floating point number is doubled.

Since the double precision format used by MATLAB uses 64 bits and 52 bits are used for the mantissa, this means that the numbers represented are accurate up to $2^{-52} \approx 2.2 \times 10^{-16}$. For example, we know that $\frac{1}{10} = 0.1$. However, this cannot be represented exactly using 64 bits. Instead, the number used is the nearest representable number which in MATLAB is given by

$$\begin{aligned}\frac{1}{10} &= 3fb999999999999a \qquad \text{(in hexadecimal)}\\ &= 0.10000000000000000555.\end{aligned}$$

The default method used in IEEE 754 to round non-representable floating point numbers is "Rounding to nearest, ties to even". This means, as mentioned above, that a floating point number is rounded to its nearest representable number. If a floating point number lies exactly between two representable numbers, it is rounded to the one with an even (*i.e.* 0) least significant bit. We revisit the example of $\frac{1}{10}$ to show how this rounding works.

If we consider the hexadecimal representation of $\frac{1}{10}$, we have the following possible representations,

$$\begin{aligned}3fb9999999999999 &= 0.09999999999999999167,\\ 3fb999999999999a &= 0.10000000000000000555,\\ 3fb999999999999b &= 0.10000000000000001943,\end{aligned}$$

where the expression on the left is the hexadecimal representation and on the right is the decimal representation. Since $3fb999999999999a = 0.10000000000000000555$ is the closest representable number to $\frac{1}{10} = 0.1$, it is the number used in numerical calculations.

Hence, if we wanted an interval with a lower bound of 0.1, we would not be able to obtain such a lower bound without any modifications because once this number is entered into MATLAB, it is rounded to a number larger than 0.1. This means the interval specified in MATLAB would not contain 0.1 and we would never be able to use a value as low as 0.1 in our calculations. Thus, we need to perform outward rounding of the intervals to ensure that we always contain the true solution which means ensuring that all input values are always attainable.

Recall that the idea of outward rounding is to round the lower bounds of intervals to a representable machine number smaller than itself and round the upper bounds to a machine number larger than itself. Thus, resulting in a slightly wider interval that contains the previous bounds. The inbuilt MATLAB function `eps(x)` is a useful function to perform this procedure as it returns the absolute value of the distance between the number x and the next larger machine number of the same precision [13]. Note that precision does not depend on the mantissa, but does depend on the exponent. In fact, the absolute value of `eps` decreases with the exponent, while the relative value remains constant, for example,

$$\begin{aligned} \texttt{eps}(0.25) &= 2^{-54}, \\ \texttt{eps}(0.5) &= 2^{-53}, \\ \texttt{eps}(1) &= 2^{-52}, \\ \texttt{eps}(2) &= 2^{-51}. \end{aligned}$$

Let $\underline{P}_{ij}^{(0)}$ and $\overline{P}_{ij}^{(0)}$ represent the lower and upper bound of the $(i,j)^{th}$ entry of the original transition probability matrix. We obtain the outwardly rounded bounds for the transition probabilities by applying the following,

$$\left[\underline{P}_{ij}^{(1)}, \overline{P}_{ij}^{(1)}\right] = \left[\underline{P}_{ij}^{(0)} - \texttt{eps}\left(\underline{P}_{ij}^{(0)}\right), \overline{P}_{ij}^{(0)} + \texttt{eps}\left(\overline{P}_{ij}^{(0)}\right)\right],$$

for all $i = 1, \ldots, n$ and $j = 0, \ldots, n$.

If we have a degenerate interval, we do not outward round the bounds since they are meant to be equal. Thus, $\underline{P}_{ij}^{(1)} = \underline{P}_{ij}^{(0)}$ and $\overline{P}_{ij}^{(1)} = \overline{P}_{ij}^{(0)}$. Also, as it is not possible for probabilities to be less than 0 or greater than 1, we do not outward round bounds which are 0 or 1. Here we note that 0 and 1 are perfectly represented in double precision and so no outward rounding is required.

4.1.3 Coherence

The idea of coherence is that every element within the interval is attainable. Consider a given interval matrix $\mathbb{P}$. For every interval, it must be possible to fix any value in that interval and still achieve realisations of the interval matrix with row sums of 1. For example, let us consider the element P_{ij} for states $i = 1, \ldots, n$ and $j = 0, \ldots, n$. Coherence implies that for every $P_{ij} \in \left[\underline{P}_{ij}, \overline{P}_{ij}\right]$, there exists elements $P_{ik} \in \left[\underline{P}_{ik}, \overline{P}_{ik}\right]$ where $k \neq j$, such that $\sum_{j=0}^{n} P_{ij} = 1$ for $i = 1, \ldots, n$ [3].

Since we do not know if our bounds are coherent, we modify them to ensure they are coherent. Coherence tightens the bounds of the interval to ensure that every element within the interval is attainable. Since degenerate intervals are the tightest possible intervals, applying coherence to these intervals will not change their bounds. Let $P_{ij}^{(1)}$ represent the $(i,j)^{th}$ entry of the outwardly-rounded transition probability matrix and $P_{ij}^{(2)}$ represent the coherent probabilities. Then, for all non-degenerate intervals, the lower and upper bounds are shifted according to the following rules [12],

$$\underline{P}_{ij}^{(2)} = \min\left\{\max\left(\underline{P}_{ij}^{(1)}, 1 - \sum_{\substack{k=0\\k\neq j}}^{n} \overline{P}_{ik}^{(1)}\right), \overline{P}_{ij}^{(1)}\right\},$$

$$\overline{P}_{ij}^{(2)} = \max\left\{\min\left(\overline{P}_{ij}^{(1)}, 1 - \sum_{\substack{k=0\\k\neq j}}^{n} \underline{P}_{ik}^{(1)}\right), \underline{P}_{ij}^{(1)}\right\},$$

for all $i = 1, \ldots, n$ and $j = 0, \ldots, n$.

The reason we take the minimum and maximum of the bounds is to ensure that we do not exceed the previous bounds due to numerical errors from performing floating point arithmetic. We know that coherence tightens the intervals and so there is no reason for the intervals to be moved outside the previous bounds.

4.1.4 Pre-processing method

Now that we have discussed and explained the concepts involved in the pre-processing stage, we need a method of pre-processing which combines these ideas of degeneracy, outward rounding and coherence in a suitable manner to fulfil the aim of pre-processing. That is, we want to develop a method which ensures that with each modification, the true solution is still contained within the intervals.

Our pre-processing method:

- We first apply degeneracy to ensure that degenerate intervals are not outwardly rounded.
- Next, we apply outward rounding to ensure that the true solution is contained within the intervals.
- Then, we apply coherence to tighten the intervals and ensure that row sums of 1 are attainable for every element in the intervals.

- Since coherence may have caused intervals to become degenerate, we apply degeneracy again.

- Now we note that coherence and degeneracy affect one another as coherence can cause intervals to become degenerate and degeneracy can have an impact on the row sum equalling 1. Thus, coherence and degeneracy are applied alternatively until no more intervals become degenerate.

With this, the pre-processing stage is complete and we can now move on to consider the optimisation problems.

4.2 Separation of minimisation and maximisation problems

Since we have been able to pre-process the intervals for the full interval transition probability matrix, we can now consider the optimisation problems we are interested in solving. Recall from Section 3.3, we have split our optimisation problem into a minimisation and a maximisation problem as we proved that the optimal value for P_{i0} can always be at the upper or lower bound for the minimisation and maximisation problem, respectively. Hence, we can set these P_{i0} intervals to be degenerate and remove them from the set of decision variables. However, there are some consequences which need to be considered. By forcing these intervals to be degenerate, we affect the row sums of the interval $\mathbb{P}$ matrix. Recall that degeneracy has an impact on coherence and vice-versa. Hence, the coherence-degeneracy loop used in the pre-processing method has to be applied again, separately, for both the minimisation and maximisation problems. This loop continues until no more changes are made to the intervals.

4.3 MATLAB fmincon formulation

Now that we have pre-processed the intervals, we will use MATLAB's `fmincon` function to solve the minimisation problem. This function allows us to specify an objective function, constraints on the decision variables and also bounds on the decision variables which makes it very useful for our purposes. We note that `fmincon` allows for user-specification of the optimisation algorithm. Depending on this choice, it may also allow for further optional inputs, such as, specifying the gradient of the objective function and the Hessian used in `fmincon`. Another input required for `fmincon` is an initial point from which the algorithm starts its search.

4.3.1 Choice of optimisation algorithm

There are four different algorithms to choose from in `fmincon`: `interior-point`, `trust-region-reflective`, `active-set` and `sqp`. For our problem, we have chosen to use `fmincon`'s `sqp` (sequential quadratic programming) algorithm to find the minimum. One of the reasons for choosing the `sqp` algorithm is because we are unable to satisfy the requirements for the `trust-region-reflective` algorithm. This algorithm does not allow for the use of equality constraints and upper and lower bounds on the variables [13]; both of which we have in our minimisation problem. Hence, it is not appropriate for our problem.

A couple of advantages the `sqp` algorithm has over the `active-set` algorithm are that `sqp` satisfies bounds at each iteration of the algorithm and it uses more efficient linear algebra routines as compared to the `active-set` algorithm [13]. We also note that since we have established Theorem 3.5.2 in Section 3.5, we know that the optimal solution to the problem lies at a vertex of the feasible region. Hence, if we chose the `interior-point` algorithm, some post-processing is needed to ensure that we lie on the boundary of the parameter space as this algorithm ensures the optimal solution lies within the feasible region. On the other hand, the `sqp` algorithm does not require any major post-processing of the solution. Both methods have been implemented on a large variety of examples and their solutions are consistent. Hence, we have chosen to use the `sqp` algorithm. Note that concepts used in the `interior-point` algorithm are discussed where necessary.

`sqp` (sequential quadratic programming) algorithm [7, 13, 15]

Consider the general non-linear optimisation problem,

$$\min_{\mathbf{x}} f(\mathbf{x})$$

subject to

$$\begin{aligned} h_i(\mathbf{x}) &= 0, \qquad \text{for } i = 1, \ldots, n, \\ g_j(\mathbf{x}) &\leq 0, \qquad \text{for } j = 1, \ldots, m, \end{aligned}$$

where h_i represents the equality constraints, g_j represents inequality constraints and the bounds on $\mathbf{x}$ are re-written in the form of inequality constraints.

The basic idea behind the `sqp` algorithm is to break down the non-linear optimisation problem into a sequence of quadratic programming (QP) subproblems. The QP subproblems are of the form

$$\min_{d \in \mathbb{R}^n} \frac{1}{2} d^T H_k d + \nabla f(\mathbf{x}_k)^T d$$

subject to

$$\begin{aligned} \nabla h_i(\mathbf{x}_k)^T d + h_i(\mathbf{x}_k) &= \mathbf{0}, \qquad i = 1, \ldots, n, \\ \nabla g_j(\mathbf{x}_k)^T d + g_j(\mathbf{x}_k) &\leq \mathbf{0}, \qquad j = 1, \ldots, m, \end{aligned}$$

where H_k is a positive-definite estimate of the Hessian of the Lagrangian, calculated using a quasi-Newton method. At each iteration of the algorithm, the QP subproblem is solved using an active set strategy. This solution, d_k, can then be used together with the step length parameter, α_k, to calculate the next point, $\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k d_k$. The step length parameter α_k is determined using a line search method where α_k is accepted when it results in a sufficient decrease in a *merit function*. A merit function balances the desire of minimising the objective function with the need to satisfy the constraints of the problem. This continues iteratively until convergence is attained.

4.3.2 Gradient

MATLAB's `fmincon` function allows us to specify the gradient of the objective function. This enables us to obtain a more accurate value to machine precision for the gradient, as opposed to an approximation obtained using a finite differencing method, thus providing us with more confidence in our results. Recall that Theorem 3.2.1 gives us an analytic expression for the gradient of the objective function and we use this form to calculate the gradient and supply it to `fmincon`.

To ensure that our calculation of the gradient is correct, we can specify an additional option in `fmincon`. This option allows us to check that the value of our gradient is within `1e-06` (tolerance specified by MATLAB) of the gradient calculated by MATLAB using the finite differencing method. If our gradient is not within `1e-06`, MATLAB stops computation and returns an error. Thus, it is a useful check to ensure that we are using the correct gradient in `fmincon`. This option was used throughout the testing phase to ensure that the gradient, as specified in Theorem 3.2.1, matched the approximation from MATLAB.

4.3.3 Hessian

As before with the gradient, we would like to provide a Hessian to `fmincon` which would enable us to obtain a more accurate result to machine precision. However, the `sqp` algorithm does not allow for a Hessian to be supplied to the `fmincon` function. Instead, MATLAB uses a quasi-Newton positive-definite approximation of the Hessian matrix when it iterates to find the optimal solution. It is noted that the Hessian that `fmincon` uses can be quite inaccurate. However, we cannot do any better when using this algorithm.

Since `fmincon`'s approximation of the Hessian may not be accurate at all times, it would be beneficial to supply a Hessian into the function. One of the algorithms which allows us to do so is the `interior-point` algorithm. This algorithm was used in the testing phase to compare with results from the `sqp` algorithm and so we provide a discussion of the Hessian used in `fmincon` and the derivation of the Hessian used for the `interior-point` algorithm.

The Hessian used by the `fmincon` function in its algorithm to find the optimal solution is the Hessian of the Lagrangian. The Lagrangian is given by

$$L = f + \sum_i \lambda_i k_i + \sum_j \lambda_j keq_j,$$

where f is the objective function, k_i are the non-linear inequality constraints, keq_j are the non-linear equality constraints and λ_i are the Lagrange multipliers. Thus, the Hessian of the Lagrangian is composed of the second derivatives of the Lagrangian.

We would like to develop a method to analytically calculate the Hessian of the Lagrangian. In the definition of the Lagrangian specified above, there were both non-linear inequality and non-linear equality constraints. However in our problem, we do not have any of these constraints and so the Lagrangian reduces to the objective function. Thus, we are interested in obtaining an analytic expression for the Hessian of the objective function. Recall that we have already proven an analytic form of the Hessian of the objective function in Theorem 3.2.2. This analytic form is provided to the `fmincon` function and used to give a more accurate result to machine precision.

4.3.4 Initial point

A required input to the `fmincon` function is an initial point which is used as a starting point for the search for the optimal objective function value. Instead of using a randomly chosen feasible initial point, we decided to use an initial point that we intuitively expect to be, or at least to be very close to, the optimal point.

Recall, the P_s matrix contains the probabilities of moving around the transient states of the Markov chain and is given by

$$P_s = \begin{bmatrix} P_{11} & \cdots & \cdots & \cdots & P_{1n} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ P_{i1} & \cdots & P_{ij} & \cdots & P_{in} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ P_{n1} & \cdots & \cdots & \cdots & P_{nm} \end{bmatrix}.$$

These probabilities, P_{ij}, are the decision variables of the problem for which we need to calculate initial values to input into `fmincon`. Hence, these are the initial values we are seeking to determine by using the intuitive method described below instead of randomly generating initial values.

Recall that we have a minimisation and a maximisation problem and so each problem has a different initial point which needs to be calculated. Let us first consider the minimisation problem. Let $P_s^{\min}$ denote the initial matrix for the minimisation problem. The intuitive method used to calculate the values in $P_s^{\min}$ is based on a similar idea to the one used in Section 3.3 where we formed the minimisation and maximisation problems. Since we are considering the minimisation problem, we are looking to minimise the expected total costs. That is, we want the process to enter the absorbing state as quickly as possible to avoid incurring extra cost. Hence, we would like the process to move to states with a higher chance of direct absorption into state 0. However, there is a trade-off between maximising the chance of direct absorption and minimising the cost incurred when the process moves around the states.

Recall from Section 4.2, we forced the P_{i0} interval to be degenerate. Hence, in the following, we will use P_{i0} instead of specifying the lower or upper bounds of the interval as they are equal.

To account for the trade-off between maximising the chance of direct absorption, P_{i0}, and minimising the cost incurred, c_i, we sort the values $\frac{c_i}{P_{i0}}$, for $i = 1, \ldots, n$, in ascending order. This is the ordering used to maximise the columns in $P_s^{\min}$. Note that if $P_{i0} = 0$ or both $P_{i0} = c_i = 0$, then we may encounter problems with the ratio $\frac{c_i}{P_{i0}}$ when sorting. Hence, these states are placed at the end of the ordering and so processed last.

Let

$$\begin{aligned} O^{\min} &= \left[\frac{c_i}{P_{i0}}, i = 1, \ldots, n, \text{ sorted in ascending order}\right], \\ r^{\min} &= \left[\text{corresponding states in the vector } O^{\min}\right]. \end{aligned}$$

For example, if we had

$$\begin{aligned} c_1 &= 1.1, & c_2 &= 2, & c_3 &= 1.2, \\ P_{10} &= 0.1, & P_{20} &= 0.25, & P_{30} &= 0.2, \end{aligned}$$

then

$$\begin{aligned} O^{\min} &= \left[\frac{c_i}{P_{i0}}, i = 1, 2, 3, \text{ sorted in ascending order}\right] \\ &= \left[\frac{1.2}{0.2}, \frac{2}{0.25}, \frac{1.1}{0.1}\right] \\ &= [6, 8, 11] \end{aligned}$$

and

$$\begin{aligned} r^{\min} &= \left[\text{corresponding states in the vector } O^{\min}\right] \\ &= [3, 2, 1]. \end{aligned}$$

These orderings are used to form the following initial matrix for the minimisation problem,

$$P_s^{\min} = \begin{bmatrix} P_{11}^{\min} & \cdots & \cdots & \cdots & P_{1n}^{\min} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ P_{i1}^{\min} & \cdots & P_{ij}^{\min} & \cdots & P_{in}^{\min} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ P_{n1}^{\min} & \cdots & \cdots & \cdots & P_{nm}^{\min} \end{bmatrix},$$

where

$$P_{ir(j)}^{\min} = \min\left\{ \overline{P}_{ir(j)}, \max\left(\underline{P}_{ir(j)}, 1 - P_{i0} - \sum_{k=1}^{j-1} P_{ir(k)}^{\min} - \sum_{k=j+1}^{n} \underline{P}_{ir(k)} \right) \right\},$$

for $i, j = 1, \ldots, n$ and $r(j)$ is the j^{th} element of the vector $r^{\min}$.

A similar procedure is applied for maximisation problems where we seek to maximise the expected total costs. Let $P_s^{\max}$ represent the initial matrix for the maximisation problem. Here, we want the process to avoid being absorbed for as long as possible which means preferring that the process moves to states with a smaller chance of direct absorption into state 0. However, like the minimisation problem, there is a trade-off between minimising the chance of direct absorption and maximising the cost incurred when the process moves around the states. Thus, we sort the values $\dfrac{c_i}{P_{i0}}$, for $i = 1, \ldots, n$ in descending order. This is also the order in which the columns of $P_s^{\max}$ are maximised. Note that if $P_{i0} = 0$ or both $P_{i0} = c_i = 0$, then we may encounter problems with the ratio $\dfrac{c_i}{P_{i0}}$ when sorting. Hence, for these states i, they are placed at the end of the ordering and so processed last.

In the same way as the minimisation problem, we let

$$O^{\max} = \left[\frac{c_i}{P_{i0}}, i = 1, \ldots, n, \text{ sorted in descending order} \right],$$

$$r^{\max} = \left[\text{corresponding states in the vector } O^{\max} \right].$$

These orderings dictate the formation of the following initial matrix for the maximisation problem,

$$P_s^{\max} = \begin{bmatrix} P_{11}^{\max} & \cdots & \cdots & \cdots & P_{1n}^{\max} \\ \vdots & \ddots & \vdots & \vdots & \vdots \\ P_{i1}^{\max} & \cdots & P_{ij}^{\max} & \cdots & P_{in}^{\max} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ P_{n1}^{\max} & \cdots & \cdots & \cdots & P_{nm}^{\max} \end{bmatrix},$$

where

$$P_{ir(j)}^{\max} = \min\left\{ \overline{P}_{ir(j)}, \max\left(\underline{P}_{ir(j)}, 1 - P_{i0} - \sum_{k=1}^{j-1} P_{ir(k)}^{\max} - \sum_{k=j+1}^{n} \underline{P}_{ir(k)} \right) \right\},$$

for $i, j = 1, \ldots, n$ and $r(j)$ is the j^{th} element of the vector $r^{\max}$.

Note that minimums and maximums are used to ensure that the initial probabilities lie within their bounds. Furthermore for degenerate intervals, since the interval has no width and both bounds are equal, the initial point is the same as the lower and upper bounds.

An important consideration is that we need an initial point with a probability 1 of absorption into state 0. That is, starting in our chosen state k, there will be some path from state k to state 0. This is important because `fmincon` requires an initial point which has a finite objective function value. Since our objective function is the expected total cost, the only way to have an infinite expected total cost is if the probability of eventual absorption into state 0 is less than 1. We also need to calculate a new initial point because we want to return a solution with a probability of absorption equal to 1. If our initial point does not have a probability 1 of absorption into state 0, we are unable to start `fmincon` from that point and, if possible, would not want to return such a solution.

Recall from Section 2.1.1 that the probability of absorption into state 0 is obtained by calculating

$$\mathbf{a} = \sum_{m=0}^{\infty} P_s^m \mathbf{P}_0,$$

where

- P_s is the sub-matrix of the transition probability matrix P containing the transition probabilities P_{ij} for $i, j = 1, \ldots, n$,
- $\mathbf{P}_0$ is the column vector containing the probabilities P_{i0} for $i = 1, \ldots, n$, and
- $\mathbf{a}$ is the column vector containing the absorption probabilities a_j for $j = 1, \ldots, n$.

Hence, before we use the initial point obtained via the above method, we first check that the probability of absorption into state 0 is 1 for the state k we are interested in. That is, we check if the k^{th} element of the vector $\mathbf{a}$ is 1. If the probability is not 1, we must find another initial point which satisfies this property.

However, we note that the interval transition probability matrix of interest may not contain a realisation P_s with probability of absorption into state 0 equal to 1 from every state. Hence, we first perform a reachability analysis, which basically investigates if the process is able to reach state 0 from any other state in the system.

Let Q be a matrix containing 1s and 0s where $Q_{ij} = 1$ if $\underline{P}_{ij} > 0$ and $Q_{ij} = 0$ if $\underline{P}_{ij} = 0$. Then, let the matrix $R = I + Q + Q^2 + \ldots + Q^n$, where I is the identity matrix of the same size as Q. Each element in the matrix R, R_{ij}, contains the total number of possible paths from state i to state j in $0, 1, 2, \ldots, n$ steps. Note that as we have a total of n states, taking n steps ensures that all non-looping paths have been considered (as well as many looping paths).

Hence, for a path to exist from any state i, where $i = 0, \ldots, n$, in the system to state 0, we require that the first column of R is strictly positive. That is, there exists at least one path from any state $i = 0, \ldots, n$ to the absorbing state 0. If there is no path from every state to state 0, the interval transition probability matrix is rejected and we return an error to the user. On the other hand, if there exists a path from every state to state 0, we calculate a new initial point in the following way. For each row i of the P_s matrix,

1. Find the minimum non-zero interval width, denoted by *min width*.
2. Find all non-degenerate intervals with a lower bound of 0, and let z be the number of such intervals.
3. Pre-allocate all such elements to be $\frac{1}{z} \times$ *min width*. Thus, we have a vector $\mathbf{b}$ with $b_i = \frac{1}{z}$ for non-degenerate elements in the interval transition probability matrix with a lower bound of 0 and $b_i = 0$ for all other elements in the vector.
4. Subtract the sum of the vector $\mathbf{b}$ from the row sum, which has the effect of removing the pre-allocated values from the row sum.
5. Then, using our previous method, we obtain a vector $\mathbf{d}$ containing the intuitive probabilities for the current row without the pre-allocation.
6. Finally, we add the pre-allocated values to the intuitive probabilities, that is, sum vectors $\mathbf{b}$ and $\mathbf{d}$ to obtain the initial values for the current row i in our initial matrix.

Using this modified method, we obtain an initial point with probability of absorption into state 0 equal to 1.

4.3.5 Removal of degenerate intervals from the problem

In the pre-processing stages, we identified degenerate intervals in the interval $\mathbb{P}$ matrix and set their upper and lower bounds to be equal. Since degenerate intervals have zero width, there is only one choice for the decision variables of these intervals. Thus, there is no need to optimise these decision variables and they can be removed from the problem. This has the effect of shrinking the dimension of the search space for the optimisation routine.

A consequence of removing decision variables corresponding to degenerate intervals from the problem is that their respective constraints must be either removed from the problem or recalculated. This is done to ensure that the constraints for the remaining decision variables are still valid.

4.4 Tolerances

Since `fmincon` is an iterative method, there needs to be some way to determine when to stop iterating, that is to determine when the optimal solution has been found. This is where we introduce the idea of tolerances. Tolerances allow us to specify the accuracy of our solution and determine when the `fmincon` function should stop iterating. Here, we provide a brief explanation of some of the tolerances used in `fmincon` as well as discuss our choices of default values for these tolerances. Before we discuss the tolerances used in `fmincon`, we first consider the two tolerances which we have defined previously in Section 4.1.1 as well as a tolerance we use to check for *active constraints* (Section 2.6).

4.4.1 TolDegenerate

Recall in Section 4.1.1, we have previously defined $TolDegenerate$ to be the tolerance on degenerate intervals. That is, an interval $X = \left[\underline{X}, \overline{X}\right]$ is numerically degenerate if $\left(\overline{X} - \underline{X}\right) \leq TolDegenerate$. Since we know that optimisation routines fail if the interval is too narrow, we have chosen a default value which is much greater than machine epsilon. Recall from Section 4.3.5 that we remove degenerate intervals from the problem. Thus, any interval with width less than $TolDegenerate$ is deemed to be numerically degenerate and removed from the problem. This means these decision variables are fixed and we do not need to use `fmincon` to obtain their optimal values. Hence, we have chosen to set the default value for $TolDegenerate = 10^{-8}$.

4.4.2 TolEqual

In Section 4.1.1, we also defined $TolEqual$ to be the tolerance on how close a numerically calculated value lies to its true value. For example, $TolEqual$ was used to check that the row sums of degenerate intervals in the $\mathbb{P}$ matrix was within $TolEqual$ of 1. Another area in which we use $TolEqual$ is in Section 4.5, where the optimal solution is purified by moving values within $TolEqual$ of their bound to their bound.

Hence, the reason for having this tolerance is due to numerical precision. Recall we have discussed floating point numbers in Section 4.1.2. We know that when arithmetic operations are performed on floating point numbers, we are not guaranteed to obtain the exact values we desire. Instead, we obtain values which lie very close to the true value. Thus, we require some tolerance to allow for this error. We have chosen the default value for $TolEqual = 10^{-13}$ to allow for this numerical precision. It is not realistic to assume that the accumulation of errors when performing floating point arithmetic will lie within machine precision, hence, we have chosen $TolEqual$ to be larger than machine precision. However, $TolEqual$ cannot be too large or we would be taking into account more than numerical precision. Thus, we have chosen $TolEqual = 10^{-13}$ as a suitable balance between allowing for error and encompassing too much variation.

4.4.3 TolActive

TolActive is a tolerance on active constraints. Recall in Section 2.6 that we defined active constraints to be the inequality constraints where equality holds. For example, if the inequality constraints are $g_j(\mathbf{x}) \leq 0$, then the active constraints at $\mathbf{x}'$ are the constraints where $g_j(\mathbf{x}') = 0$. Due to numerical precision as well as the tolerances to which `fmincon` stops iterating, it would not be appropriate to expect $g_j(\mathbf{x}') = 0$. Instead, active constraints are those where $|g_j(\mathbf{x}') - 0| <= TolActive$, that is, active constraints are constraints which are within *TolActive* of 0. We have chosen the default value for $TolActive = 10^{-8}$ due to its relationship with *TolDegenerate* and *TolX* (see below).

4.4.4 TolX

As stated in the MATLAB documentation [13], let x_i be the solution of the current iteration and x_{i+1} be the solution of the next iteration. Then, we have that:

> `TolX` is a lower bound on the size of a step, meaning the norm of $(x_i - x_{i+1})$. If the solver attempts to take a step that is smaller than `TolX`, the iterations end. `TolX` is sometimes used as a relative bound, meaning iterations end when $|(x_i - x_{i+1})| < \texttt{TolX} * (1 + |x_i|)$, or a similar relative measure.

In `fmincon`, the default value for *TolX* is 10^{-6} but we have chosen to set $TolX = 10^{-8}$. Note that this is the same value as the default for *TolDegenerate* which is the tolerance on degenerate intervals. The reason for this is that the step size taken cannot be bigger than the width of the intervals. If *TolX* is larger than *TolDegenerate*, then the largest possible step from the upper bound to the lower bound or vice-versa would result in a step size smaller than *TolX*. Thus causing `fmincon` to stop.

4.4.5 TolFun

In the MATLAB documentation [13], $TolFun$ is defined as follows:

> `TolFun` is a lower bound on the change in the value of the objective function during a step. If $|f(x_i) - f(x_{i+1})| <$ `TolFun`, the iterations end. `TolFun` is sometimes used as a relative bound, meaning iterations end when $|f(x_i) - f(x_{i+1})| <$ `TolFun`$*(1 + |f(x_i)|)$, or a similar relative measure.
>
> `TolFun` is also a bound on the first-order optimality measure. If the optimality measure is less than `TolFun`, the iterations end. `TolFun` can also be used as a relative bound.
>
> The first-order optimality measure is the maximum of the following two norms:
>
> $$\begin{aligned}\|\nabla_x L(x,\lambda)\| = \Bigg\| &\nabla f(x) + A^T\lambda_{ineqlin} + Aeq^T\lambda_{eqlin} \\ &+ \sum_i \lambda_{ineqnonlin,i}\nabla k_i(x) + \sum_i \lambda_{eqnonlin,i}\nabla keq_i(x)\Bigg\|,\end{aligned} \tag{4.4.2}$$
>
> and
>
> $$\left\| \overrightarrow{|\ell_i - x_i|\lambda_{lower,i}}, \overrightarrow{|x_i - u_i|\lambda_{upper,i}}, \overrightarrow{|(Ax-b)_i|\lambda_{ineqlin,i}}, \overrightarrow{|c_i(x)|\lambda_{ineqnonlin,i}} \right\|, \tag{4.4.3}$$
>
> where the notation $\overrightarrow{|\ell_i - x_i|\lambda_{lower,i}}$ represents the vector obtained by multiplying the vectors $|\ell - \mathbf{x}|$ and λ_{lower} elementwise. Note that the norm of the vectors in (4.4.2) and (4.4.3) is the infinity norm (maximum). The subscripts on the Lagrange multipliers correspond to solver Lagrange multiplier structures, where
>
> - `lower`, associated with lower bounds

- `upper`, associated with upper bounds
- `eqlin`, associated with linear equalities
- `ineqlin`, associated with linear inequalities
- `eqnonlin`, associated with nonlinear equalities
- `ineqnonlin`, associated with nonlinear inequalities

> The summations in (4.4.2) range over all constraints. If a bound is ±`Inf`, that term is not constrained, so it is not part of the summation.

After much testing, we decided to set $TolFun = 10^{-6}$. Since we perform our own first-order optimality check, we are not too particular about this check and simply require `fmincon` to return a feasible solution.

4.4.6 TolCon

MATLAB's documentation [13] has the following explanation for $TolCon$:

> `TolCon` is an upper bound on the magnitude of any constraint function. If a solver returns a point x with $c(x) >$ `TolCon` or $|ceq(x)| >$ `TolCon`, the solver reports that the constraints are violated at x. `TolCon` can also be a relative bound.
>
> Note: `TolCon` operates differently from other tolerances. If `TolCon` is not satisfied (i.e., if the magnitude of the constraint function exceeds `TolCon`), the solver attempts to continue, unless it is halted for another reason. A solver does not halt simply because `TolCon` is satisfied.

Since we want the maximum constraint violation to be as small as possible, we have chosen to set $TolCon = 10^{-12}$. Furthermore, we require $TolCon$ to be smaller than $TolActive$ as we need to account for possible optimal values returned by `fmincon` which lie outside their bounds.

All of the concepts discussed in Sections 4.3 and 4.4 are specified as options or input variables in the `fmincon` function. Then the optimal expected total cost, χ_k, is returned together with the optimal P_s matrix containing the decision variables for the chosen state k, for $k = 1, \ldots, n$.

4.5 Purification

Now that we have an optimal expected total cost and an optimal P_s matrix, we perform a purification of the decision variables in the optimal P_s matrix. The idea behind this process is the same as that of purification for interior point methods in linear programming problems. For our problem, we want to ensure that all decision variables in the optimal P_s matrix are sufficiently close to satisfying all of their constraints. Hence, given an optimal P_s matrix, we first move any decision variables lying outside the interval to its closest bound. This may occur due to numerical precision and tolerances.

Next, recall from Theorem 3.5.2 that we have proven the optimal P_s matrix must be a vertex. Also recall from Section 4.4, *TolActive* is the tolerance on active constraints and so any value within *TolActive* of its bound is considered to be active or at its bound. Hence, we move any optimal values within *TolActive* of its bounds to its closest bound.

After moving all of these optimal values to their bounds, it is unlikely for the row sum, $\sum_j P_{ij}$, to remain within *TolEqual* of 1. Thus, we need to shift variables so that the important row sum constraint, that is, $\sum_j P_{ij}$ lies within *TolEqual* of 1, is satisfied.

Procedure:

- Move optimal values outside their bounds to their bounds.

- Move optimal values within $TolActive$ of their bounds to their bounds.
- Check row sum constraint.
- If row sum constraint not satisfied, we move certain variables.

This gives us two possible types of variables we can move,

1. Free variables, which are variables x such that $|x - lb| > TolActive$ and $|x - ub| > TolActive$. That is, the optimal value x is not within $TolActive$ of its bounds.
2. Active variables, all the optimal variables which were initially active (*i.e.* at one of their bounds) and also the variables which were moved to their bounds (*i.e.* either outside the interval or within $TolActive$ of the bounds).

Ideally, we want to allocate the amount needed to bring the row sum to 1 in the above order.

Within each variable, we have two sub-cases,

1. Positive slack $\left(\sum_j P_{ij} - 1 > TolEqual\right)$,
2. Negative slack $\left(\sum_j P_{ij} - 1 < -TolEqual\right)$.

In the following, we detail methods to deal with these sub-cases for both free and active variables.

1. Free variables

Since we know that the P_s matrix should be at a vertex to be optimal (Theorem 3.5.2), we should consider moving these free variables to a bound first. Note that the reason we have free variables is due to the solution returned by `fmincon` being an interior point. Thus, we would add or subtract from these free variables depending on whether a positive or negative amount is needed to bring the row sum to 1.

a) For a positive amount

Here, there is too much probability mass. Hence, to reduce the mass and ensure that the row sums are within $TolEqual$ of 1, we need to decrease the value of the free variables.

Method:

For each row without a row sum within $TolEqual$ of 1,

- Calculate the gradient for the current optimal values.
- Using only the gradients corresponding to the free variables, find the largest gradient. That is, find the maximum of *gradient* as every element of the gradient vector is positive. The reason for this is that we want to move the variable which causes the largest decrease in the objective function, as we are considering the minimisation problem. Note that a similar logic and method is used for the maximisation problem.
- Move the variable corresponding to the largest gradient as much as possible without exceeding its lower bound.
- Repeat this process until row sums are within $TolEqual$ of 1, that is, $\sum_j P_{ij} - 1 < TolEqual$, or we have no more free variables left to move.

b) For a negative amount

Here, the variables are not large enough for the row sum to be 1. Hence, we need to increase the value of the variables not at their upper bounds which in turn increases the row sum and pushes it to be within $TolEqual$ of 1. We apply the same method as the one for the positive amount case, except we move the variable corresponding to the smallest gradient (for a minimisation problem) towards its upper bound without exceeding the bound.

2. Active variables

If there are no free variables, then there is no choice but to move active variables to enable the row sum to be within $TolEqual$ of 1.

a) For a positive amount

Here, there is too much probability mass. Since all variables are active, they must either lie on their lower or upper bound. To decrease the row sum, we need to decrease the value of the variables. This cannot be done to variables at their lower bounds as this would take the variables outside their bounds. Thus, we can only subtract from variables at their upper bounds.

Method:

For each row without a row sum within $TolEqual$ of 1,

- Calculate the gradient for the current optimal values.
- Using only the gradients corresponding to variables which are at their upper bound, find the largest gradient. That is, find the maximum of *gradient* as every element of the gradient vector is positive. The reason for this is that we want to move the variable which causes the largest decrease in the objective function, as we are considering the minimisation problem. For the maximisation problem, a similar logic and method is used.
- Move the variable corresponding to the largest gradient as much as possible without exceeding its lower bound.
- Repeat this process until row sums are within $TolEqual$ of 1, that is, $\sum_j P_{ij} - 1 < TolEqual$, or we have no more active variables left to move.
- Note that if there are no more active variables to move, the row

sum should be within *TolEqual* of 1. The reason for this is because coherence and degeneracy have been applied.

b) For a negative amount

Here, the variables are not large enough to enable a row sum of 1. Similar to the positive amount case, there are only active variables and so increasing the row sum involves increasing the value of the variables. Only variables at their lower bounds are considered for movement because increasing variables at their upper bound will result in pushing the variable outside their intervals. The same method as the one developed for the positive amount case is used. The differences between the two procedures are that we consider gradients corresponding to variables at their lower bounds for the negative amount case and also move the variable with the smallest gradient (for a minimisation problem) towards its upper bound without exceeding the bound.

After purification has been completed, we check that the optimal matrix P_s^* has a probability of absorption equal to 1 to provide the user with more information about the optimal matrix. If the probability of absorption is not equal to 1 for P_s^*, we notify the user of this and do not check first-order optimality conditions as $(I - P_s^*)^{-1}$ will be undefined. (see Section 3.1).

4.6 Optimality conditions

Recall that the minimisation problem we solve, for a given state k, using `fmincon` is given by,

$$\min \chi_k = \left[(I - P_s)^{-1}\mathbf{c}\right]_k,$$

subject to

$$\begin{aligned}
h_i(P_s) &= \sum_{j=1}^{n} P_{ij} - \big(1 - \overline{P}_{i0}\big) = 0, && i = 1, \ldots, n, \\
\ell_{ij}(P_s) &= \underline{P}_{ij} - P_{ij} \leq 0, && i, j = 1, \ldots, n, \\
u_{ij}(P_s) &= P_{ij} - \overline{P}_{ij} \leq 0, && i, j = 1, \ldots, n.
\end{aligned}$$

First-order optimality conditions allow us to determine if a point in the feasible region is optimal. Recall that we have defined the Karush-Kuhn-Tucker (KKT) conditions, which are first-order necessary conditions, for a point P_s^* to be a local minimiser of the objective function, χ_k. Here, we re-state Theorem 2.6.2 using the terms and constraints of our optimisation problem and refer the reader back to Section 2.6 for definitions of terms used.

Theorem 4.6.1 (Karush-Kuhn-Tucker (KKT) Theorem)**.** *Let χ_k, for all $k = 1, \ldots, n$, h_i, ℓ_{ij}, u_{ij}, for all $i, j = 1, \ldots, n$ be sufficiently differentiable (that is, $\nabla \chi_k$, for all $k = 1, \ldots, n$, ∇h_i, $\nabla \ell_{ij}$, ∇u_{ij}, for all $i, j = 1, \ldots, n$ exist). Let P_s^* be a regular point and a local minimiser of $\chi_k(P_s)$ such that $h_i(P_s) = 0$, $\ell_{ij}(P_s) \leq 0$, $u_{ij}(P_s) \leq 0$, for all $i, j = 1, \ldots, n$. Then, there exists $\mu_i \in \mathbb{R}$, $\lambda_{\ell_{ij}} \in \mathbb{R}$ and $\lambda_{u_{ij}} \in \mathbb{R}$ for all $i, j = 1, \ldots, n$ such that:*

1. $\nabla \chi_k(P_s^*) + \sum_{i=1}^{n} \mu_i \nabla h_i(P_s^*) + \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_{\ell_{ij}} \nabla \ell_{ij}(P_s^*) + \sum_{i=1}^{n} \sum_{j=1}^{n} \lambda_{u_{ij}} \nabla u_{ij}(P_s^*) = \mathbf{0}$,
2. $\lambda_{\ell_{ij}} \ell_{ij}(P_s^*) = 0$ *and* $\lambda_{u_{ij}} u_{ij}(P_s^*) = 0$, *for all* $i, j = 1, \ldots, n$, *and*
3. $\lambda_{\ell_{ij}}, \lambda_{u_{ij}} \geq 0$ *for all* $i, j = 1, \ldots n$.

We note that for inequality constraints ℓ_{ij} and u_{ij}, at least one of the constraint or the corresponding λ must be zero. Hence, we can specify $L(P_s)$ and $U(P_s)$ to be the sets of active lower bound and upper bound constraints at P_s respectively, that is, $L(P_s) = \{(i, j) : \ell_{ij}(P_s) = 0\}$ and $U(P_s) = \{(i, j) : u_{ij}(P_s) = 0\}$. Further, we can consider for each $i = 1, \ldots, n$, the sets of active lower bound and upper bound constraints for each row i of the P_s matrix given by $L_i(P_s) = \{j : \ell_{ij}(P_s) = 0\}$ and

$U_i(P_s) = \{j : u_{ij}(P_s) = 0\}$ respectively. Thus, the first condition in Theorem 4.6.1 reduces to

$$\nabla \chi_k(P_s^*) + \sum_{i=1}^{n} \mu_i \nabla h_i(P_s^*) + \sum_{(i,j)\in L(P_s^*)} \lambda_{\ell_{ij}} \nabla \ell_{ij}(P_s^*) + \sum_{(i,j)\in U(P_s^*)} \lambda_{u_{ij}} \nabla u_{ij} = \mathbf{0}.$$

Recall that a regular point, P_s^*, as defined in Section 2.6, is a feasible point satisfying the requirement that the set of vectors

$$\Big\{\nabla h_i(P_s^*) \text{ for all } i = 1, \ldots, n,\ \nabla \ell_{ij}(P_s^*) \text{ for all } (i,j) \in L(P_s^*) \\ \text{and } \nabla u_{ij}(P_s^*) \text{ for all } (i,j) \in U(P_s^*)\Big\}$$

are linearly independent.

We note that for our problem, the requirement of a regular point in the KKT conditions, Theorem 4.6.1, is not satisfied. However, the KKT conditions are necessary conditions under the regularity assumption, not sufficient conditions. This means that of all regular points only those that satisfy the KKT conditions are candidates to be a local minimiser of the function. There may be other irregular points, either satisfying the KKT conditions 1, 2 and 3, or not, which could be a local minimiser. Further, any regular point that satisfies the KKT conditions is not necessarily a local minimiser. Thus, second-order sufficient conditions (Theorem 2.6.3) are used to check that P_s^* is a minimiser of the objective function.

Here, we re-state Theorem 2.6.3 using the terms and constraints for our optimisation problem and refer the reader back to Section 2.6 for definition of terms used.

The Lagrangian function for this optimisation problem is given by,

$$l(P_s, \boldsymbol{\mu}, \boldsymbol{\lambda_\ell}, \boldsymbol{\lambda_u}) = \chi_k(P_s) + \sum_{i=1}^{n} \mu_i h_i(P_s) + \sum_{(i,j)\in L(P_s)} \lambda_{\ell_{ij}} \ell_{ij}(P_s) + \sum_{(i,j)\in U(P_s)} \lambda_{u_{ij}} u_{ij}(P_s),$$

and so the Hessian matrix of $l(P_s, \boldsymbol{\mu}, \boldsymbol{\lambda_\ell}, \boldsymbol{\lambda_u})$ with respect to P_s is

$$\mathbf{L}(P_s, \boldsymbol{\mu}, \boldsymbol{\lambda_\ell}, \boldsymbol{\lambda_u}) = H\,\chi_k(P_s) + \sum_{i=1}^{n} \mu_i H h_i(P_s) + \sum_{(i,j)\in L(P_s)} \lambda_{\ell_{ij}} H\ell_{ij}(P_s) + \sum_{(i,j)\in U(P_s)} \lambda_{u_{ij}} H u_{ij}(P_s),$$

where $H\,\chi_k(P_s)$ is the Hessian matrix of χ_k at P_s and similarly for $H\ell_{ij}(P_s)$ and $Hu_{ij}(P_s)$.

Here we also consider the set $T(P_s^*, \boldsymbol{\mu}, \boldsymbol{\lambda_\ell}, \boldsymbol{\lambda_u})$ which is used in Theorem 4.6.2.

$$\begin{aligned} T(P_s^*, \boldsymbol{\mu}, \boldsymbol{\lambda_\ell}, \boldsymbol{\lambda_u}) = \Big\{ \mathbf{y} :& \nabla h_i(P_s^*)^T \mathbf{y} = 0, \text{for all } i = 1, \ldots, n, \\ & \nabla \ell_{ij}(P_s^*)^T \mathbf{y} = 0, \text{for all } (i,j) \in \tilde{L}(P_s^*, \boldsymbol{\lambda_\ell}), \text{ and} \\ & \nabla u_{ij}(P_s^*)^T \mathbf{y} = 0, \text{for all } (i,j) \in \tilde{U}(P_s^*, \boldsymbol{\lambda_u}) \Big\}, \end{aligned}$$

where we have $\tilde{L}(P_s^*, \boldsymbol{\lambda_\ell}) = \{(i,j) : \ell_{ij}(P_s^*) = 0, \lambda_{\ell_{ij}} > 0\}$ and $\tilde{U}(P_s^*, \boldsymbol{\lambda_u}) = \{(i,j) : u_{ij}(P_s^*) = 0, \lambda_{u_{ij}} > 0\}$.

The second-order sufficient conditions for a point P_s^* to be a strict local minimiser of χ_k is given in the following theorem.

Theorem 4.6.2 (Second-Order Sufficient Conditions)**.** *Suppose that χ_k, for all $k = 1, \ldots, n$, h_i, ℓ_{ij}, u_{ij}, for all $i, j = 1, \ldots, n$ are sufficiently differentiable (that is, $H\,\chi_k$, for all $k =, 1, \ldots, n$, Hh_i, $H\ell_{ij}$, Hu_{ij}, for all $i, j = 1, \ldots, n$ exist) and there exists a feasible point P_s^* and $\mu_i \in \mathbb{R}$, $\lambda_{\ell_{ij}} \in \mathbb{R}$ and $\lambda_{u_{ij}} \in \mathbb{R}$ for all $i, j = 1, \ldots, n$ such that:*

1. $\nabla\,\chi_k(P_s^*) + \sum_{i=1}^{n} \mu_i \nabla h_i(P_s^*) + \sum_{(i,j)\in L(P_s^*)} \lambda_{\ell_{ij}} \nabla \ell_{ij}(P_s^*) + \sum_{(i,j)\in U(P_s^*)} \lambda_{u_{ij}} \nabla u_{ij} = \mathbf{0}$,
2. $\lambda_{\ell_{ij}} \ell_{ij}(P_s^*) = 0$ *and* $\lambda_{u_{ij}} u_{ij}(P_s^*) = 0$, *for all* $i, j = 1, \ldots, n$,
3. $\lambda_{\ell_{ij}}, \lambda_{u_{ij}} \geq 0$ *for all* $i, j = 1, \ldots n$, *and*
4. *For all* $\boldsymbol{y} \in T(P_s^*, \boldsymbol{\mu}, \boldsymbol{\lambda_\ell}, \boldsymbol{\lambda_u}), \boldsymbol{y} \neq \mathbf{0}$, *we have* $\boldsymbol{y}^T \mathbf{L}(P_s^*, \boldsymbol{\mu}, \boldsymbol{\lambda_\ell}, \boldsymbol{\lambda_u}) \boldsymbol{y} > 0$.

Then, P_s^* is a strict local minimiser of χ_k subject to $h_i(\mathbf{x}) = 0$, $\ell_{ij}(P_s) \leq 0$, $u_{ij}(P_s) \leq 0$ for all $i, j = 1, \ldots, n$.

However, recall that we proved in Theorem 3.5.2, that the optimal solution occurs at the vertex of the feasible region. Also recall from Section 3.5 that the rows of the P_s matrix are independent. Hence, we can consider each row separately and discuss what it means for a row of the P_s matrix to represent a vertex of the feasible region for that row. At a vertex solution for a row of P_s, this means that the row has at least $(n-1)$ active constraints as at most one element in a row of the P_s matrix can be free. If we consider a vertex solution, which satisfies the KKT conditions 1, 2 and 3, for row i of the P_s matrix, then

$$T(P_s^*, \boldsymbol{\mu}, \boldsymbol{\lambda_\ell}, \boldsymbol{\lambda_u}) = \{\mathbf{0}\},$$

if all active constraints j have $\lambda_{\ell_{ij}} > 0$ and $\lambda_{u_{ij}} > 0$. That is, the union of the sets $\tilde{L}(P_s^*, \boldsymbol{\lambda_\ell})$ and $\tilde{U}(P_s^*, \boldsymbol{\lambda_u})$ contains all active constraints j. If $T(P_s^*, \boldsymbol{\mu}, \boldsymbol{\lambda_\ell}, \boldsymbol{\lambda_u}) = \{\mathbf{0}\}$, then it tells us that there are no feasible directions to move in to result in a smaller objective function. Hence, the vertex is a local minimum.

We note that the only way for $T(P_s^*, \boldsymbol{\mu}, \boldsymbol{\lambda_\ell}, \boldsymbol{\lambda_u}) \neq \{\mathbf{0}\}$ is when $\lambda_{\ell_{ij}} = 0$ or $\lambda_{u_{ij}} = 0$ for some active constraint j. However, we note that throughout extensive testing of the code, we never observed this to occur. Through some testing, it appears that unless the initial point occurs at a vertex and has an active constraint j with $\lambda_{\ell_{ij}} = 0$ or $\lambda_{u_{ij}} = 0$, the `sqp` algorithm used in `fmincon` does not move to these points when iterating to obtain the minimum. Since such an occurrence has not been observed, we do not detail our method for checking second-order optimality conditions for the cases where $T(P_s^*, \boldsymbol{\mu}, \boldsymbol{\lambda_\ell}, \boldsymbol{\lambda_u}) \neq \{\mathbf{0}\}$ when P_s^* is at a vertex of the feasible region.

Recall that for the common case of a vertex solution with $T(P_s^*, \boldsymbol{\mu}, \boldsymbol{\lambda_\ell}, \boldsymbol{\lambda_u}) = \{\mathbf{0}\}$, we do not need to check second-order optimality conditions as there are no feasible directions which decrease the objective function. Hence, we just need to check that a solution is optimal by checking that it satisfies the KKT conditions 1, 2 and 3.

Recall when we discussed the tolerance $TolFun$ in Section 4.4, we presented the way MATLAB calculates its first-order optimality measure which is based upon the KKT conditions 1, 2 and 3. This first-order optimality measure is used to determine when the solver iterations end. Since we have placed quite a loose tolerance on the first-order optimality measure ($TolFun = 10^{-6}$) and `fmincon` returns approximate solutions which depend on tolerances, we have implemented our own method for ensuring that the KKT conditions 1, 2 and 3 in Theorem 4.6.1, are satisfied. By doing so, it will also allow us to be more confident with our results.

4.6.1 Checking optimality

To check that the KKT conditions 1, 2 and 3 are satisfied, the Lagrange multipliers, μ_i, $\lambda_{\ell_{ij}}$ and $\lambda_{u_{ij}}$ for $i, j = 1, \ldots, n$, are calculated by solving the following equation,

$$\nabla \chi_k(P_s^*) + \sum_{i=1}^{n} \mu_i \nabla h_i(P_s^*) + \sum_{(i,j)\in A(P_s^*)} \lambda_{\ell_{ij}} \nabla \ell_{ij}(P_s^*) + \sum_{(i,j)\in A(P_s^*)} \lambda_{u_{ij}} \nabla u_{ij} = \mathbf{0},$$

and ensuring that $\lambda_{\ell_{ij}}, \lambda_{u_{ij}} \geq 0$. Since h_i is an equality constraint, it must hold at all times. Thus, ∇h_i exists in the equation for all i. On the other hand, $\nabla \ell_{ij}$ and ∇u_{ij} only appear when constraints ℓ_{ij} or u_{ij} are active.

We know that the following inequalities hold for the minimisation problem and all possible P_s matrices,

$$\begin{aligned}
\nabla\chi_k(P_s) &\geq 0, && \text{for all } k = 1, \dots, n, \\
\nabla h_i(P_s) &= 1, && \text{for all } i = 1, \dots, n, \\
\nabla \ell_{ij}(P_s) &= \begin{cases} -1, & \text{if } (i,j) \in L(P_s), \\ 0, & \text{otherwise,} \end{cases} \\
\nabla u_{ij}(P_s) &= \begin{cases} 1, & \text{if } (i,j) \in U(P_s), \\ 0, & \text{otherwise.} \end{cases}
\end{aligned}$$

There are a variety of cases which need to be considered when solving for the Lagrange multipliers. Furthermore, we note that as each row of the P_s matrix is independent of one another, we have blocks of constraints associated with different rows of the P_s matrix. This reduces our problem and allows us to consider these blocks separately. Thus, for each row, we consider the following cases: n active constraints, $n-1$ active constraints and less than $n-1$ active constraints. Within these cases, we have further sub-cases to consider, which we explain in the following. First, let us fix row i, for $i = 1, \dots, n$.

1. n active constraints

This corresponds to an underconstrained problem as we have more variables than equations. Hence, we can choose any solution for $\mu_i, \lambda_{\ell_{ij}}, \lambda_{u_{ij}}$ as long as $\lambda_{\ell_{ij}}, \lambda_{u_{ij}} \geq 0$ for all $i, j = 1, \dots, n$.

Case 1: All lower bounds active. Here, the active constraints are h_i and ℓ_{ij}, that is, $\sum_{j=1}^{n} P_{ij} = \left(1 - \overline{P}_{i0}\right)$ and $P_{ij} = \underline{P}_{ij}$. Thus, we want to solve the following system of equations, for all $j \in L_i(P_s^*)$,

$$\mu_i - \lambda_{\ell_{ij}} = -\Big[\nabla\chi_k(P_s^*)\Big]_{(i,j)},$$

where $\Big[\nabla\,\chi_k(P_s^*)\Big]_{(i,j)}$ is the element of the gradient vector, $\nabla\,\chi_k(P_s^*)$, corresponding to the $(i,j)^{th}$ element of the P_s^* matrix, and is known to be non-negative. Here, we can set $\mu_i = 0$ and solve for the remaining $\lambda_{\ell_{ij}}$, which are non-negative.

Case 2: If the only active constraints are h_i and u_{ij}, that is, $\displaystyle\sum_{j=1}^{n} P_{ij} = \big(1 - \overline{P}_{i0}\big)$ and $P_{ij} = \overline{P}_{ij}$, then we have a degenerate case since all upper bounds are active. This tells us we have degenerate intervals as decreasing any P_{ij} will not lead to a row sum of 1. As we have dealt with degenerate intervals, this case should never occur.

Case 3: Here, the active constraints are h_i as well as some upper and lower bound constraints, u_{ij} and ℓ_{ij}, that is, $\displaystyle\sum_{j=1}^{n} P_{ij} = \big(1 - \overline{P}_{i0}\big)$, some $P_{ij} = \underline{P}_{ij}$ and the other $P_{ij} = \overline{P}_{ij}$. Thus, for all $j \in L_i(P_s^*)$ and $m \in U_i(P_s^*)$, we want to solve

$$\mu_i - \lambda_{\ell_{ij}} = -\left[\nabla\,\chi_k(P_s^*)\right]_{(i,j)} \qquad \text{and} \qquad \mu_i + \lambda_{u_{im}} = -\left[\nabla\,\chi_k(P_s^*)\right]_{(i,m)}.$$

Let

$$\begin{aligned} a &= -\max\Big(\Big|\left[\nabla\,\chi_k(P_s^*)\right]_{(i,m)}\Big|\Big), &\qquad \text{for all } m \in U_i(P_s^*),\\ b &= -\min\Big(\Big|\left[\nabla\,\chi_k(P_s^*)\right]_{(i,j)}\Big|\Big), &\qquad \text{for all } j \in L_i(P_s^*). \end{aligned}$$

Here, we require

$$b \leq a.$$

If this condition is not satisfied, then we do not have a solution satisfying KKT conditions 1, 2 and 3. Assuming $b \leq a$, we need $\mu \in [b,a]$ and without loss of generality, we set

$$\mu_i = \frac{a+b}{2},$$

and solve for the remaining $\lambda_{\ell_{ij}}$ and $\lambda_{u_{im}}$, guaranteeing that $\lambda_{\ell_{ij}}$ and $\lambda_{u_{im}}$ are all non-negative.

2. $n-1$ **active constraints**

For this case, there should be a unique solution for the Lagrange multipliers. Hence, we solve the system of linear equations to obtain the unique values for μ_i, $\lambda_{\ell_{ij}}$ and $\lambda_{u_{ij}}$.

3. Less than $n-1$ **active constraints**

This corresponds to row i of P_s^* not being a vertex of the feasible region. Since we have an overconstrained problem, that is, there are more constraints than variables for this case, we are generally unable to obtain a solution for μ_i. For all $j \notin L_i(P_s^*) \cup U_i(P_s^*)$, we have $\mu_i = -\left[\nabla \chi_k(P_s^*)\right]_{(i,j)}$ and as there are less than $n-1$ active constraints, there must be at least 2 such j. Therefore, unless $\left[\nabla \chi_k(P_s^*)\right]_{(i,j)}$ are equal for all such j, no solution exists and P_s^* is not first-order optimal. However, if $\left[\nabla \chi_k(P_s^*)\right]_{(i,j)}$ are equal for all such j and other constraints ensure that $\lambda_{\ell_{ij}}, \lambda_{u_{ij}} \geq 0$, then the point is first-order optimal.

The above method is used to obtain the Lagrange multiplier values for the minimisation problem. We note the Lagrange multipliers for the maximisation problem can be found in a similar way by taking extra care with signs.

After obtaining the Lagrange multiplier values, we need to determine if P_s^* is first-order optimal. Recall from Theorem 4.6.1 that we require $\lambda_{\ell_{ij}}, \lambda_{u_{ij}} \geq 0$ for all $i, j = 1, \dots, n$. Thus, we simply check that $\lambda_{\ell_{ij}}, \lambda_{u_{ij}} \geq 0$ for all $i, j = 1, \dots, n$ for the point P_s^* to be optimal.

4.6.2 Checking solution is at a vertex

Recall from Theorem 3.5.2, that we are free to choose the optimal solution to be at a vertex of the feasible region. Thus, we choose to check that the optimal P_s^*

satisfies this condition. A possible reason for P_s^* to not be at a vertex is if there are equivalent vertex solutions but `fmincon` has returned a non-vertex optimal solution instead. Although we purify the solution from `fmincon`, if an element in P_s^* does not lie within $TolActive$ of its bounds, we do not move it. Hence, it is possible to have a first-order optimal solution which is not a vertex solution.

If this check fails, that is, P_s^* is not at a vertex, we move the elements of P_s^* lying within their bounds to their bounds to form a vertex solution whilst ensuring that the row sum constraint, that is, $\sum_j P_{ij}$ is within $TolEqual$ of 1, remains satisfied. Recall that each row of the P_s matrix is independent of one another. Thus, we can consider each row separately and only modify the rows with elements not forming a vertex of the feasible region. Here, we assume that there are only 2 free variables in each row. Hence, each row has a maximum of two possible vertices. In the following, we detail the method used to shift P_s^* to a vertex of the feasible region. Note that it is possible to generalise this method to consider more than 2 free variables in each row, however we do not present that here.

Method:

1. Find the first row with more than 1 free variable and find the free variables in this row.
2. For the first vertex, we increase the first free variable and decrease the second free variable until one of them reaches their upper and lower bound respectively.
   - Calculate the distance between the first free variable and its upper bound, d_1.
   - Calculate the distance between the second free variable and its lower bound, d_2.

- Obtain the minimum of d_1 and d_2, $d = \min(d_1, d_2)$, and this is the most we can move either variable to ensure that the row sum constraint remains fulfilled and both variables are still within their bounds.

- Move each variable in the appropriate direction by the minimum distance, d.

- Note that by construction, we can be sure that the row sum constraint remains satisfied and all variables lie within their bounds.

3. For the second vertex, we decrease the first free variable and increase the second free variable until one of them reaches their lower and upper bound respectively, using a method analogous to the one for the first vertex. The only difference is the direction we move the variables. Instead of the distance between the first free variable and its upper bound, here, we consider the distance between the first free variable and its lower bound and the distance between the second free variable and its upper bound.

4. Calculate the objective function value at each vertex.

   - If the objective function values are the same, then without loss of generality, we choose the first vertex solution and replace the current optimal solution with the first vertex solution.

   - If the objective function values are different, then we choose the vertex solution which optimises the objective function and replace the current optimal solution with this vertex solution.

5. Re-run the algorithm starting at this vertex solution.

The reason for running `fmincon` again at the new vertex solution is because when we consider a new P_s^* matrix, it is possible for the gradient vector to change. Hence, this new P_s^* matrix may not be an optimal solution.

4.6.3 What if `fmincon` fails?

Due to tolerances on the `fmincon` function, there may be times when it fails to find a first-order optimal solution. So what can we do in this instance? One possibility is to consider linearising the objective function using a Taylor polynomial expansion about the point P_s^*. For any $n \times n$ matrix M, define $vec(M)$ to be the $n^2 \times 1$ vector arranged in the usual order $(M_{11}, M_{12}, \ldots, M_{1n}, \ldots, M_{n1}, \ldots, M_{nn})^T$. Hence, we consider the following linear program instead,

$$\chi_k(P_s) = \chi_k(P_s^*) + \big(vec(P_s) - vec(P_s^*)\big)^T \nabla\, \chi_k(P_s^*),$$

subject to the constraints (4.0.1). Since we expect the solution obtained from `fmincon` to be close to the optimal, $\big(vec(P_s) - vec(P_s^*)\big)$ should be small and so we expect this approximation to be reasonable.

Since $vec(P_s^*)$ is a constant, we can reduce the objective function to

$$vec(P_s)^T \nabla\, \chi_k(P_s^*).$$

We use `linprog` to solve this linear program and then perform purification as well as checking optimality conditions for this new optimal point.

4.7 Other methods to solve this problem

- Sequential linear programming: This is a promising method as it returns comparable results to the ones obtained using our method. Furthermore, as we show in Section 4.10, the sequential linear programming method runs much quicker than our method and Blanc and den Hertog's [2] method even for larger problems. We note that the idea of solving an optimisation problem sequentially is well established in the second-order case, where we sequentially approximate a non-linear optimisation problem by quadratic problems.

However, this is not well established in the first-order case, where we linearise the non-linear objective function. Hence, more work needs to be performed to ensure the robustness of this sequential linear programming method.

One obvious reason why sequential linear programming may not work is that such an approach cannot return an optimal solution at any point other than a vertex. In Section 3.5, we proved that the optimal vertex solution cannot be improved upon by consideration of interior points. Thus, our problem seems to be a suitable candidate for sequential linear programming techniques.

However, we do not know whether that is sufficient, or whether other issues could arise. Since the computational time is much quicker than the other two methods and returns comparable results, it is definitely a possible future extension to this work.

- MDP methods: Since we have been able to prove that our optimisation problems are Markov decision processes with some restrictions (Lemma 3.5.1), we could consider using MDP methods to find the optimal solution. However, we would like to be able to use our method for structured problems. As we will discuss in Chapter 5, when considering structured Markov chains, we do not have a Markov decision process. Thus, MDP methods cannot be used for structured Markov chains and so we do not consider MDP methods here.

4.8 Post-processing

Recall that we minimise and maximise for each of the n states separately. Thus, a total of $2n$ optimisation problems are solved in this process. Just as we have done throughout this chapter, we consider minimising and maximising for a state k, for $k = 1, \dots, n$, of a discrete-time interval Markov chain. Hence, after obtaining optimal solutions which occur at vertices of the feasible region, we perform some post-processing of the minimum and maximum solutions for each state $k = 1, \ldots, n$ before returning a solution to the user. Due to tolerances on `fmincon`, the minimum and maximum expected total cost returned by the `fmincon` function are approximations of the true minimum and maximum values. Thus, the purpose of the post-processing stage is to ensure that we bound the true minimum and maximum expected total cost for the interval discrete-time Markov chain.

The basic method we apply is to consider intervals centred at the minimum and maximum P_s matrices with widths relative to the tolerance specified in `fmincon`. Since we are unable to use `fmincon` again on these smaller intervals due to tolerances, we choose to use INTLAB for calculations. Although we have previously stated the problems with using INTLAB's method of solution, we know that the intervals returned by INTLAB are conservative. Furthermore, as these new intervals we are considering are very narrow, we would not expect the conservative intervals returned by INTLAB to be too wide. We check that this is true by calculating width of the minimum and maximum interval expected total cost returned by INTLAB. In the following, we expand and provide more detail on the basic method described above.

Using the optimal P_s matrices, P_s^*, obtained from `fmincon` for one of our n problems, we can calculate intervals for the expected total cost of the chosen state of the Markov chain. Since we have specified a tolerance on the decision variables, $TolX$, in `fmincon`, we add and subtract the relative tolerance, $P_s^* \times TolX$, from P_s^* to give an interval $\mathbb{P}_s^*$ matrix, where $\mathbb{P}_s^* = [P_s^* (1 - TolX), P_s^* (1 + TolX)]$. This procedure is performed on both the minimum and maximum P_s^* matrices returned in the previous method.

Before we use these interval $\mathbb{P}_s^*$ matrices to calculate the interval expected total costs, we first ensure that the elements in the interval $\mathbb{P}_s^*$ matrices do not exceed the user-specified interval $\mathbb{P}_s$ matrix. To do so, we take the intersection of the interval $\mathbb{P}_s^*$ matrices with the outwardly rounded user-specified interval $\mathbb{P}_s$ matrix. This gives us interval $\mathbb{P}_s^*$ matrices for the minimisation and maximisation problems.

Let us consider the minimisation problem and note that the maximisation problem follows analogously. We calculate the minimum interval expected total cost vector, $\boldsymbol{\chi} = (I - \mathbb{P}_s^*)^{-1}\,\mathbf{c}$, using INTLAB's backslash operator, $\boldsymbol{\chi} = (I - \mathbb{P}_s^*)\backslash\mathbf{c}$, where $\mathbb{P}_s^*$ is the interval matrix for the minimisation problem. Then, we extract the appropriate element corresponding to the state k we have chosen to consider. As discussed earlier, we use INTLAB to give us conservative intervals and since the intervals in our $\mathbb{P}_s^*$ interval matrix should be very narrow $(2 \times TolX)$, the expansion should be minor.

To determine the variability about the minimum solution, we calculate the difference between the upper and lower bounds of the calculated interval expected total cost for the minimum interval $\mathbb{P}_s^*$ matrix. If the width of this interval is small, this tells us that there is a small variability about the minimum expected total cost. This method is also used to determine the variability about the maximum expected total cost.

Finally, we want to return an interval expected total cost for a chosen state of an interval Markov chain, given an interval transition probability matrix, $\mathbb{P}$. This interval that we return has a lower bound corresponding to the lower bound of the minimum interval expected total cost and an upper bound corresponding to the upper bound of the maximum interval expected total cost.

4.9 Comparison with intervals obtained from INTLAB

Recall from Section 3.2 that we could have calculated the interval expected total costs vector,

$$[\underline{\chi}, \overline{\chi}] = (I - \mathbb{P}_s^*)^{-1}\, \mathbf{c},$$

directly using INTLAB (Section 2.2). However, we noted that as the methods used by INTLAB are iterative, it does not allow us to impose the time-homogeneous property. Furthermore, we wanted to ensure that the interval expected total costs we obtain are as tight as possible. Here, we provide a comparison of the interval mean hitting time, that is, the interval expected total cost, with $\mathbf{c} = \mathbf{1}$, obtained using INTLAB and our method.

Consider the following discrete-time interval Markov chain with interval transition probability matrix,

$$\mathbb{P} = \begin{bmatrix} [1,1] & [0,0] & [0,0] \\ 0.3 \pm \varepsilon & 0.4 \pm \varepsilon & 0.3 \pm \varepsilon \\ 0.25 \pm \varepsilon & 0.5 \pm \varepsilon & 0.25 \pm \varepsilon \end{bmatrix}.$$

We would like to calculate the interval mean hitting time vector, $\boldsymbol{\nu}$, for the above interval transition probability matrix with $\varepsilon = 0.05$.

Using our numerical method, we obtain an interval mean hitting time vector of the form

$$\boldsymbol{\nu} = \begin{bmatrix} [2.9577, 4.2858] \\ [3.0985, 4.4898] \end{bmatrix}.$$

If we use INTLAB to directly calculate (3.1.6), we obtain

$$\boldsymbol{\nu} = \begin{bmatrix} [1.5108, 5.4892] \\ [1.5828, 5.7506] \end{bmatrix}.$$

Recall from Section 3.2 that we want to calculate minimal intervals, that is, intervals of smallest width containing the true solution. Hence, from these intervals, we observe that the intervals obtained using our method are tighter than the ones obtained using INTLAB and thus, preferred.

Now let us compare the intervals obtained from both methods as we increase the value of ε (see Table 4.9). We observe from this that as ε increases, that is, the uncertainty in our parameters increases, the width of the mean hitting times intervals increases as well. Hence, the larger the uncertainty in our parameters, the more uncertain we are in the performance measure. The other important observation from Table 4.9 is that for $\varepsilon = 0.1$, the intervals returned by INTLAB contain negative numbers. Since it does not make sense for the mean hitting time to be negative, the intervals obtained by INTLAB are unsatisfactory for our needs.

ε	Our method	INTLAB
0.01	$\begin{bmatrix}[3.3762, 3.6333]\\ [3.5369, 3.8063]\end{bmatrix}$	$\begin{bmatrix}[3.2293, 3.7707]\\ [3.3831, 3.9502]\end{bmatrix}$
0.05	$\begin{bmatrix}[2.9577, 4.2858]\\ [3.0985, 4.4898]\end{bmatrix}$	$\begin{bmatrix}[1.5108, 5.4892]\\ [1.5828, 5.7506]\end{bmatrix}$
0.1	$\begin{bmatrix}[2.5609, 5.5264]\\ [2.6829, 5.7895]\end{bmatrix}$	$\begin{bmatrix}[-5.3530, 12.3530]\\ [-5.6079, 12.9412]\end{bmatrix}$

Table 4.9.1: Table comparing mean hitting times obtained from our method with those obtained from INTLAB for increasing values of ε.

4.10 Comparison with the method of Blanc and den Hertog

Recall from Section 3.2 that we discussed in detail the method developed by Blanc and den Hertog [2]. Here, we provide a numerical comparison of their method with ours and show that for a small problem, our method is faster but for larger problems, solving a linear program is more efficient as expected.

Blanc and den Hertog's formulation of the mean hitting times problem is described by the following linear optimisation problem,

$$\min \mathbf{v}^T \mathbf{1},$$

subject to

$$\begin{aligned}
\mathbf{1}^T \Xi &= \mathbf{e}_k^T, & \\
v_i(I_{ij} - \overline{p}_{ij}) - \xi_{ij} &\leq 0, & \text{for } i, j = 1, \dots, n, \\
\xi_{ij} - v_i(I_{ij} - \underline{p}_{ij}) &\leq 0, & \text{for } i, j = 1, \dots, n, \\
v_i \, \underline{p}_{i0} - \sum_{j=1}^{n} \xi_{ij} &\leq 0, & \text{for } i = 1, \dots, n, \\
\sum_{j=1}^{n} \xi_{ij} - v_i \, \overline{p}_{i0} &\leq 0, & \text{for } i = 1, \dots, n,
\end{aligned}$$

where $\mathbf{1}$ is a vector of ones and $\mathbf{e}_k = [0, \dots, 0, 1, 0, \dots, 0]^T$ is a vector of zeros with 1 in the k^{th} element. The decision variables for this problem are v_i and ξ_{ij}. To obtain solutions in the form of the original non-linear optimisation problem, we need to transform ξ_{ij} to obtain the elements p_{ij} of the P_s matrix, but no transformation is required on v_i.

This problem was solved using MATLAB's `linprog` function with the Simplex algorithm. We recorded how long it took to obtain bounds on the mean hitting time and the corresponding optimal P_s matrices for states $k = 1, \dots, n$ in the

Markov chain. Then, we compared the computational time taken for Blanc and den Hertog's method and our method to solve the problem and obtain the desired outputs. Figure 4.10.1 is a plot of the computational time for an increasing number of states (3 to 30). For each state, 100 random interval transition probability matrices were considered and the average time taken to solve each problem is plotted. We can see from Figure 4.10.1 that as the number of states increases, the method by Blanc and den Hertog is faster than our method. For less than 25 states, our method obtains the desired solutions faster than the method by Blanc and den Hertog. However, for much larger state spaces, solving a linear program is much quicker than our method. Recall from Section 4.7 that sequential linear programming can be used to solve the optimisation problems. This was implemented in MATLAB and the computational time for the sequential linear programming method is also shown on the plot of computational times. We see that it solves the optimisation problems much more quickly than the other two methods.

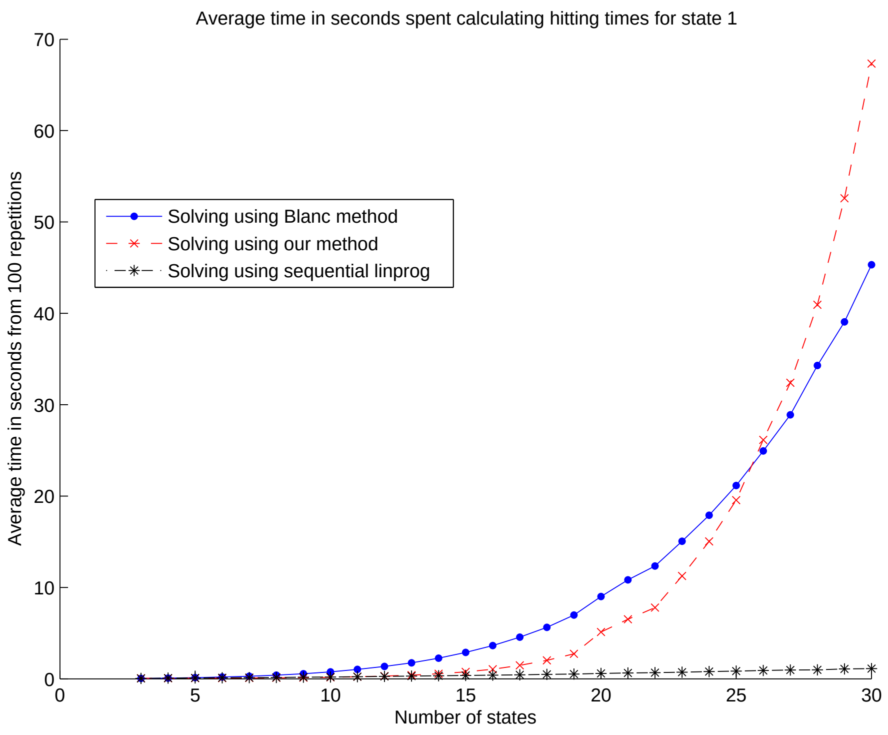

Figure 4.10.1: Plot of computational times for Blanc and den Hertog's method, our method and sequential linear programming.

4.11 Application to continuous-time Markov chains

Recall from Section 3.6 that to obtain the expected total cost for a continuous-time interval Markov chain, we first use uniformisation to transform the continuous-time interval Markov chain to its associated discrete-time interval Markov chain. Then, we solve for the expected total cost for the associated discrete-time interval Markov chain before using Theorem 3.6.1 to obtain the expected total cost for the continuous-time interval Markov chain. In the following, we discuss our choice of a uniformisation constant and describe the outward rounding steps used in the transformation from a continuous-time interval Markov chain to its associated discrete-time interval Markov chain.

4.11.1 Choice of uniformisation constant

The uniformisation constant u is defined, in Section 3.6, to be any real number such that $u \geq m = \max_i (\overline{q}_i)$. Since we are able to choose any real number u, we have decided to choose u such that $u \geq \max_i (\overline{q}_i)$ and ensure that u has an exact floating point representation. Thus, u is a known constant and since it has an exact floating point representation, there is no need for an interval representation of u. The reason for choosing an exact floating point representation is due to numerical precision. If u does not have an exact floating point representation, we would need to use an interval representation to ensure that we contain the true value of u.

Recall that the transformation required to obtain an interval transition probability matrix for the associated discrete-time interval Markov chain from an interval generator matrix of a continuous-time interval Markov chain is given by, $\mathbb{P}_u = I + \frac{1}{u}\mathbb{Q}$. Thus, if we have an interval representation of u, then we would need to obtain an interval representation of $\frac{1}{u}$ and multiply this by the interval generator matrix $\mathbb{Q}$ whilst ensuring that at each step the true solution is contained within the intervals. This increases the complexity of the transformation and calculations involved. Since there is no restriction on the choice of u other than $u \geq \max_i (\overline{q}_i)$, we can choose to make the calculations easier by requiring an exact floating point representation of u and still obtain the desired results.

Since $u \geq \max_i (\overline{q}_i)$, we look to obtain an exact floating point number larger than $\max_i (\overline{q}_i)$. Recall from Section 4.1.2, that the inbuilt MATLAB function `eps(x)` returns the absolute value of the distance between the number x and the next larger machine number of the same precision [13]. Hence, we can use this function to obtain an exact floating point representation of u by apply the following,

$$u = \max_i (\overline{q}_i) + \texttt{eps}\left(\max_i (\overline{q}_i)\right).$$

4.11.2 Outward rounding

As we have discussed in Section 4.1.2, we outwardly round intervals to ensure that the true solution is always contained within the intervals. The method for outward rounding is quite simple where the left endpoint is moved to the left (on the number line) and the right endpoint is moved to the right [14]. Here, we consider the use of outward rounding on the interval generator matrix $\mathbb{Q}$ and also on the transformation from a continuous-time interval Markov chain to its associated discrete-time interval Markov chain.

Given an interval generator matrix $\mathbb{Q}$ of a continuous-time interval Markov chain, we perform the following transformation, $\mathbb{P}_u = I + \frac{1}{u}\mathbb{Q}$, to obtain an interval transition probability matrix, $\mathbb{P}_u$, for the associated discrete-time interval Markov chain. We want to ensure that the interval transition probability matrix, $\mathbb{P}_u$, contains the true solution. Hence, we apply the following method to ensure that this holds.

Let `out(X)` represent the outwardly rounded interval X. Then, our method is as follows.

1. Outward round the interval generator matrix $\mathbb{Q}$.

$$\mathbb{Q}_1 = \texttt{out}\left(\mathbb{Q}\right).$$

2. Outward round each step of the transformation $\mathbb{P}_u = I + \frac{1}{u}\mathbb{Q}$, to obtain

$$\mathbb{P}_u = \texttt{out}\left(I + \texttt{out}\left(\frac{\mathbb{Q}_1}{u}\right)\right),$$

where u is the uniformisation constant and has an exact floating point representation.

This gives us an interval transition probability matrix, $\mathbb{P}_u$, and we can now use the numerical method (including pre- and post-processing steps) developed in this chapter, for discrete-time interval Markov chains, to obtain interval expected total costs for each state $k = 1, \dots, n$ of the associated discrete-time interval Markov chain. By applying the transformation in Theorem 3.6.1, we obtain the interval expected total costs for each state $k = 1, \dots, n$ of the continuous-time interval Markov chain.

This concludes the discussion of our numerical method for a general discrete-time interval Markov chain as well as the required extensions for continuous-time interval Markov chain. In the next chapter, we consider interval birth and death chains and prove analytic results for these.

Chapter 5

Interval birth and death chains: An analytic approach

Due to their properties, birth and death chains are commonly used in many applications, such as ecological modelling and epidemic modelling (which we explore in Chapter 6). Hence, it is of interest to us to incorporate intervals into these structured Markov chains, investigate them in detail and explore ways to calculate various performance measures, such as the expected total costs, for these chains. Furthermore, recall from Chapter 3 that we were unable to prove analytic results on the expected total cost for general Markov chains. Instead, in Chapter 4, we developed a numerical method to obtain minimal intervals on the expected total cost for general Markov chains. Here, we are interested in obtaining an analytic solution for this class of discrete-time birth and death processes. In this chapter, we define a discrete-time interval birth and death process as well as a discrete-time interval constant-parameter birth and death process and prove analytic results for both classes of processes.

5.1 Discrete-time interval birth and death process

Consider a discrete-time interval Markov chain $(X_m, m \in \mathbb{Z}_+)$ with $n+1$ states, $S = \{0, 1, \ldots, n\}$ where state 0 is an absorbing state. Recall from Section 3.1 that X_m has an interval transition probability matrix of the form,

$$\mathbb{P} = \begin{bmatrix} [1,1] & [0,0] & \cdots & \cdots & \cdots & [0,0] \\ [\underline{P}_{10}, \overline{P}_{10}] & [\underline{P}_{11}, \overline{P}_{11}] & \cdots & \cdots & \cdots & [\underline{P}_{1n}, \overline{P}_{1n}] \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ [\underline{P}_{i0}, \overline{P}_{i0}] & [\underline{P}_{i1}, \overline{P}_{i1}] & \cdots & [\underline{P}_{ij}, \overline{P}_{ij}] & \cdots & [\underline{P}_{in}, \overline{P}_{in}] \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ [\underline{P}_{n0}, \overline{P}_{n0}] & [\underline{P}_{n1}, \overline{P}_{n1}] & \cdots & \cdots & \cdots & [\underline{P}_{nn}, \overline{P}_{nn}] \end{bmatrix},$$

where

$$\begin{aligned} &\sum_{j=0}^{n} P_{ij} = 1, && \text{for } P_{ij} \in [\underline{P}_{ij}, \overline{P}_{ij}] \text{ and } i = 1, \ldots, n, \\ &0 \le \underline{P}_{ij} \le 1, && \text{for all } i = 1, \ldots, n \text{ and } j = 0, \ldots, n, \\ &0 \le \overline{P}_{ij} \le 1, && \text{for all } i = 1, \ldots, n \text{ and } j = 0, \ldots, n, \\ &\underline{P}_{ij} \le \overline{P}_{ij}, && \text{for all } i = 1, \ldots, n \text{ and } j = 0, \ldots, n. \end{aligned}$$

Now let us investigate a modified version of the above discrete-time interval Markov chain by introducing structure to the transition probability matrix.

Consider a discrete-time interval birth and death process $(X_m, m \in \mathbb{Z}_+)$ with the same set of states as the general discrete-time interval Markov chain described above. The difference between these two processes is the structure of the transition probability matrix. Due to the birth and death structure, the birth and death process is only able to move to neighbouring states. That is, if the process is in state i at time t, then at time $t+1$, it can remain in state i or move to either of its neighbouring states $i-1$ or $i+1$ with probability γ_i, μ_i and λ_i respectively. Note that these probabilities depend on the state the process is in at time t. Hence, the interval transition probability matrix, $\mathbb{P}$, for a discrete-time interval birth and death process is given by,

$$\begin{bmatrix} [1,1] & [0,0] & [0,0] & [0,0] & \cdots & \cdots & [0,0] \\ [\underline{\mu}_1,\overline{\mu}_1] & [\underline{\gamma}_1,\overline{\gamma}_1] & [\underline{\lambda}_1,\overline{\lambda}_1] & [0,0] & \cdots & \cdots & [0,0] \\ [0,0] & [\underline{\mu}_2,\overline{\mu}_2] & [\underline{\gamma}_2,\overline{\gamma}_2] & [\underline{\lambda}_2,\overline{\lambda}_2] & [0,0] & \cdots & [0,0] \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \vdots \\ [0,0] & \cdots & \cdots & [0,0] & [\underline{\mu}_{n-1},\overline{\mu}_{n-1}] & [\underline{\gamma}_{n-1},\overline{\gamma}_{n-1}] & [\underline{\lambda}_{n-1},\overline{\lambda}_{n-1}] \\ [0,0] & \cdots & \cdots & \cdots & [0,0] & [\underline{\mu}_n,\overline{\mu}_n] & [\underline{\gamma}_n+\underline{\lambda}_n,\overline{\gamma}_n+\overline{\lambda}_n] \end{bmatrix},$$

where

$$\begin{aligned} &0<\underline{\mu}_i\le\overline{\mu}_i\le 1, && \text{for } i=1,\cdots,n,\\ &0\le\underline{\gamma}_i\le\overline{\gamma}_i\le 1, && \text{for } i=1,\cdots,n,\\ &0\le\underline{\lambda}_i\le\overline{\lambda}_i\le 1, && \text{for } i=1,\cdots,n,\\ &\mu_i+\gamma_i+\lambda_i=1, && \text{for } \mu_i\in[\underline{\mu}_i,\overline{\mu}_i],\gamma_i\in[\underline{\gamma}_i,\overline{\gamma}_i],\lambda_i\in[\underline{\lambda}_i,\overline{\lambda}_i] \text{ and } i=1,\cdots,n. \end{aligned}$$

Note here that we assume $0<\underline{\mu}_i$ for $i=1,\dots,n$ to ensure that there is always a positive probability of moving from state i to state $i-1$, for $i=1,\dots,n$. This means that the probability of being absorbed into state 0 is always 1 from any state k, for $k=1,\dots,n$.

Here, we note that for the rest of the chapter we consider a finite state space. However, everything generalises to an infinite state space with the removal of the upper boundary.

Recall from Chapter 3 that we are interested in obtaining an interval for the expected total cost for each state of the process. We obtained lower and upper bounds on the expected total cost for each state by minimising and maximising the expected total cost, χ_k, for each state $k=1,\dots,n$ separately. In Sections 5.1.1 and

5.1.2, we will detail proofs for the analytic form of the minimum and maximum P_s matrices. From these matrices, we can easily solve for the minimum and maximum expected total cost for each state by solving $\chi_k = \left[(I - P_s)^{-1}\,\mathbf{c}\right]_k$.

First, recall that the structure of the birth and death process only allows movement between neighbouring states. This means that the process can only be absorbed into state 0 from state 1. Thus, it seems likely that the expected total cost from state k would be greater than or equal to the expected total cost from state $k-1$, for $k = 1, \ldots, n$. The following theorem gives us this result.

Theorem 5.1.1. *Let χ_k be the expected total cost incurred before the process is absorbed into state 0, conditional on starting in state $k = 1, \ldots, n$. For a discrete-time interval birth and death process,*

$$\chi_k \geq \chi_{k-1}, \qquad \textit{for } k = 1, \ldots, n.$$

Proof. Let K_j be the random variable describing the total cost incurred before the process first hits state j. Then we have

- K_0 is the random variable describing the total cost incurred before the process first hits state 0, and
- K_{k-1} is the random variable describing the total cost incurred before the process first hits state $k-1$.

Recall, we have defined χ_k to be the expected total cost incurred before the process is absorbed into state 0, conditional on starting in state k. That is, $\chi_k = E[K_0|X_0 = k]$. Hence, we have

$$\begin{aligned}\chi_k &= E[K_0|X_0 = k] \\ &= E[K_{k-1}|X_0 = k] + E[K_0|X_0 = k-1], \qquad \text{for } k = 1, \ldots, n,\end{aligned}$$

because given that the process starts in state k, the structure of the birth and death process implies that the process must pass through state $k-1$ for it to be absorbed into state 0, and also the Markov property tells us that the next state of the process only depends on the current state the process is in and not on any of the states previously visited.

Thus, this means

$$\begin{aligned} \chi_k &\geq E[K_0|X_0 = k-1] \\ &= \chi_{k-1}, \qquad \text{for } k = 1, \ldots, n. \end{aligned}$$

$\square$

5.1.1 Minimisation problem

Consider the minimisation problem which has the form,

$$\min \chi_k = \left[(I - P_s)^{-1} \mathbf{c} \right]_k ,$$

where

$$P_s = \begin{bmatrix} \gamma_1 & \lambda_1 & 0 & \cdots & \cdots & \cdots & 0 \\ \mu_2 & \gamma_2 & \lambda_2 & 0 & \cdots & \cdots & 0 \\ 0 & \mu_3 & \gamma_3 & \lambda_3 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \mu_{n-2} & \gamma_{n-2} & \lambda_{n-2} & 0 \\ 0 & \cdots & \cdots & 0 & \mu_{n-1} & \gamma_{n-1} & \lambda_{n-1} \\ 0 & \cdots & \cdots & \cdots & 0 & \mu_n & \gamma_n + \lambda_n \end{bmatrix},$$

and subject to the constraints

$$\begin{aligned} &\underline{\mu}_i \leq \mu_i \leq \overline{\mu}_i, && \text{for } i = 1, \cdots, n, \\ &\underline{\gamma}_i \leq \gamma_i \leq \overline{\gamma}_i, && \text{for } i = 1, \cdots, n, \\ &\underline{\lambda}_i \leq \lambda_i \leq \overline{\lambda}_i, && \text{for } i = 1, \cdots, n, \\ &\mu_i + \gamma_i + \lambda_i = 1, && \text{for } i = 1, \cdots, n. \qquad \text{(row sum constraint)} \end{aligned}$$

We are interested in obtaining an analytic solution to the above problem which is given by the following theorem.

Theorem 5.1.2. *The minimum expected total cost, χ_k, for $k = 1, \dots, n$ for the discrete-time interval birth and death process is obtained by solving $\chi_k = \left[(I - P_s^*)^{-1}\mathbf{c}\right]_k$ where the optimal P_s matrix, P_s^*, is given by,*

$$P_s^* = \begin{bmatrix} \gamma_1^* & \lambda_1^* & 0 & \cdots & \cdots & \cdots & 0 \\ \mu_2^* & \gamma_2^* & \lambda_2^* & 0 & \cdots & \cdots & 0 \\ 0 & \mu_3^* & \gamma_3^* & \lambda_3^* & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \mu_{n-2}^* & \gamma_{n-2}^* & \lambda_{n-2}^* & 0 \\ 0 & \cdots & \cdots & 0 & \mu_{n-1}^* & \gamma_{n-1}^* & \lambda_{n-1}^* \\ 0 & \cdots & \cdots & \cdots & 0 & \mu_n^* & 1 - \mu_n^* \end{bmatrix},$$

and

$$\begin{aligned} \mu_i^* &= \overline{\mu}_i, && \textit{for } i = 1, \dots, n, \\ \gamma_i^* &= \min\left(\overline{\gamma}_i, 1 - \overline{\mu}_i - \underline{\lambda}_i\right), && \textit{for } i = 1, \dots, n-1, \\ \lambda_i^* &= 1 - \overline{\mu}_i - \gamma_i^*, && \textit{for } i = 1, \dots, n-1. \end{aligned}$$

Proof. From Theorem 5.1.1, we know that the expected total costs for each state are ordered according to $\chi_k \geq \chi_{k-1}$, for $k = 1, \dots, n$. This means the process is better to move to state $i-1$ from state i, for $i = 1, \dots, n$, in order to minimise χ_k, for $k = 1, \dots, n$. Thus, we seek to maximise the probability of moving from state i to $i-1$, that is, maximise μ_i for $i = 1, \dots, n$. Also, we note that due to the structure of the birth and death process, the process is better to remain in state i as opposed to moving to state $i+1$ as it would require at least one extra transition to get back to state i. Hence, after maximising the probability of moving from state i to state $i-1$, μ_i, we seek to maximise the probability of remaining in state i, γ_i, and set λ_i, which is the probability of moving from state i to $i+1$, to be the remaining probability. The two boundary cases follow similarly.

Note that although μ_1 is not within P_s, since μ_1 corresponds to P_{10} for a general discrete-time interval Markov chain, we know that $\mu_1^* = \overline{\mu}_1$ from Theorem 3.3.1. Now let us consider the elements in P_s. Recall that we are interested in maximising μ_i. Thus, we can set $\mu_i^* = \overline{\mu}_i$, for all $i = 1, \ldots, n$ as coherence (Section 4.1.3) ensures that every element within the interval is attainable. Since no element has been allocated in row i, for $i = 1, \ldots, n$, we can choose to set $\mu_i^* = \overline{\mu}_i$, for all $i = 1, \ldots, n$ and still be certain that the row sum constraint will be satisfied.

Next, we seek to maximise the probability of remaining in state i, γ_i, for $i = 1, \ldots, n-1$, subject to the row sum constraint. Ideally, we would want to set $\gamma_i^* = \overline{\gamma}_i$. However, as we have already set $\mu_i^* = \overline{\mu}_i$ for $i = 1, \ldots, n$, choosing $\gamma_i^* = \overline{\gamma}_i$ may result in the row sum constraint being violated. Hence, we set $\gamma_i^* = \min\left(\overline{\gamma}_i, 1 - \overline{\mu}_i - \underline{\lambda}_i\right)$ for $i = 1, \ldots, n-1$. The second term ensures that if $\overline{\gamma}_i$ cannot be chosen, then the largest value of γ_i within its interval is chosen that ensures the row sum constraint is satisfied.

Finally, we allocate the remaining probability required for the row sum constraint to be fulfilled in λ_i. Hence, $\lambda_i^* = 1 - \overline{\mu}_i - \gamma_i^* \in \left[\underline{\lambda}_i, \overline{\lambda}_i\right]$ for $i = 1, \ldots, n-1$.

Thus, the following optimal values,

$$\begin{aligned} \mu_i^* &= \overline{\mu}_i, && \text{for } i = 1, \ldots, n, \\ \gamma_i^* &= \min\left(\overline{\gamma}_i, 1 - \overline{\mu}_i - \underline{\lambda}_i\right), && \text{for } i = 1, \ldots, n-1, \\ \lambda_i^* &= 1 - \overline{\mu}_i - \gamma_i^*, && \text{for } i = 1, \ldots, n-1, \end{aligned}$$

will result in the minimal value of χ_k, for $k = 1, \ldots, n$. □

5.1.2 Maximisation problem

Now, consider the maximisation problem which has the form,

$$\max \chi_k = \left[(I - P_s)^{-1} \mathbf{c}\right]_k,$$

where

$$P_s = \begin{bmatrix} \gamma_1 & \lambda_1 & 0 & \cdots & \cdots & \cdots & 0 \\ \mu_2 & \gamma_2 & \lambda_2 & 0 & \cdots & \cdots & 0 \\ 0 & \mu_3 & \gamma_3 & \lambda_3 & 0 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \mu_{n-2} & \gamma_{n-2} & \lambda_{n-2} & 0 \\ 0 & \cdots & \cdots & 0 & \mu_{n-1} & \gamma_{n-1} & \lambda_{n-1} \\ 0 & \cdots & \cdots & \cdots & 0 & \mu_n & \gamma_n + \lambda_n \end{bmatrix},$$

and subject to the constraints

$$\begin{aligned} \underline{\mu}_i \le \mu_i \le \overline{\mu}_i, &\qquad \text{for } i = 1, \cdots, n, \\ \underline{\gamma}_i \le \gamma_i \le \overline{\gamma}_i, &\qquad \text{for } i = 1, \cdots, n, \\ \underline{\lambda}_i \le \lambda_i \le \overline{\lambda}_i, &\qquad \text{for } i = 1, \cdots, n, \\ \mu_i + \gamma_i + \lambda_i = 1, &\qquad \text{for } i = 1, \cdots, n. \qquad \text{(row sum constraint)} \end{aligned}$$

As before with the minimisation problem, we are interested in obtaining an analytic solution to the above problem.

Theorem 5.1.3. *The maximum expected total cost, χ_k, for $k = 1, \ldots, n$ for the discrete-time interval birth and death process is obtained by solving $\chi_k = \left[(I - P_s^*)^{-1}\,\mathbf{c}\right]_k$ where the optimal P_s matrix, P_s^*, is given by,*

$$P_s^* = \begin{bmatrix} \gamma_1^* & \lambda_1^* & 0 & \cdots & \cdots & \cdots & 0 \\ \mu_2^* & \gamma_2^* & \lambda_2^* & 0 & \cdots & \cdots & 0 \\ 0 & \mu_3^* & \gamma_3^* & \lambda_3^* & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \mu_{n-2}^* & \gamma_{n-2}^* & \lambda_{n-2}^* & 0 \\ 0 & \cdots & \cdots & 0 & \mu_{n-1}^* & \gamma_{n-1}^* & \lambda_{n-1}^* \\ 0 & \cdots & \cdots & \cdots & 0 & \mu_n^* & 1 - \mu_n^* \end{bmatrix},$$

and

$$\begin{aligned}
\lambda_i^* &= \overline{\lambda}_i, && \textit{for } i = 1, \dots, n-1,\\
\gamma_i^* &= \min\left(\overline{\gamma}_i, 1 - \underline{\mu}_i - \overline{\lambda}_i\right), && \textit{for } i = 1, \dots, n-1,\\
\mu_i^* &= 1 - \overline{\lambda}_i - \gamma_i^*, && \textit{for } i = 1, \dots, n-1,\\
\mu_n^* &= \underline{\mu}_m.
\end{aligned}$$

Proof. Since we are interested in maximising the expected total cost, χ_k, for $k = 1, \dots, n$, we want to avoid absorption to state 0 as much as possible. From Theorem 5.1.1, we know that the expected total costs for each state are ordered according to $\chi_k \geq \chi_{k-1}$, for $k = 1, \dots, n$. Thus, to maximise χ_k for $k = 1, \dots, n$, the process is better to move to state $i + 1$ from state i, for $i = 1, \dots, n$. This means we seek to maximise the probability of moving from state i to state $i + 1$, that is, maximise λ_i. Due to the structure of the birth and death process and to avoid absorption into state 0, it makes sense for the process to remain in state i as opposed to moving to state $i - 1$ as remaining in state i ensures that there is at least one extra transition to get to state 0. Thus, after maximising the probability of moving from state i to $i+1$, λ_i, we seek to maximise the probability of remaining in state i, γ_i, and set μ_i, which is the probability of moving from state i to $i - 1$, to be the remaining probability. Note that both boundary cases follow similarly.

For the boundary at $i = n$, since there are no λ_n and γ_n values, we seek to maximise the probability of remaining in state n. That is, maximise $1 - \mu_n$ which is the same as minimising μ_n. Hence, we set $\mu_n^* = \underline{\mu}_n$. For the boundary at $i = 1$, we note that although μ_1 is not within P_s, since μ_1 corresponds to P_{10} for a general discrete-time interval Markov chain, we know that $\mu_1^* = \underline{\mu}_1$ from Theorem 3.3.2.

Now let us consider the elements in P_s. Recall that we are interested in maximising λ_i. Thus, we can set $\lambda_i^* = \overline{\lambda}_i$, for all $i = 1, \dots, n-1$ as coherence (Section 4.1.3) ensures that every element within the interval is attainable. Since no element has been allocated in row i, for $i = 1, \dots, n-1$, we can choose to set $\lambda_i^* = \overline{\lambda}_i$, for all

$i = 1, \ldots, n-1$ and still be certain that the row sum constraint will be satisfied.

Next, we seek to maximise the probability of remaining in state i, γ_i, for $i = 1, \ldots, n-1$, subject to the row sum constraint. Ideally, we would want to set $\gamma_i^* = \overline{\gamma}_i$. However, as we have already set $\lambda_i^* = \overline{\lambda}_i$ for $i = 1, \ldots, n-1$, choosing $\gamma_i^* = \overline{\gamma}_i$ may result in the row sum constraint being violated. Hence, we set $\gamma_i^* = \min\left(\overline{\gamma}_i, 1 - \underline{\mu}_i - \overline{\lambda}_i\right)$ for $i = 1, \ldots, n-1$. The second term ensures that if $\overline{\gamma}_i$ cannot be chosen, then the largest value of γ_i within its interval is chosen that ensures the row sum constraint is satisfied.

Finally, we allocate the remaining probability to μ_i to ensure that the row sum constraint is fulfilled. Hence, we set $\mu_i^* = 1 - \overline{\lambda}_i - \gamma_i^* \in \left[\underline{\mu}_i, \overline{\mu}_i\right]$ for $i = 1, \ldots, n-1$.

Thus, the following optimal values,

$$\begin{aligned}
\lambda_i^* &= \overline{\lambda}_i, && \text{for } i = 1, \ldots, n-1,\\
\gamma_i^* &= \min\left(\overline{\gamma}_i, 1 - \underline{\mu}_i - \overline{\lambda}_i\right), && \text{for } i = 1, \ldots, n-1,\\
\mu_i^* &= 1 - \overline{\lambda}_i - \gamma_i^*, && \text{for } i = 1, \ldots, n-1,\\
\mu_n^* &= \underline{\mu}_m,
\end{aligned}$$

will result in the maximal value of χ_k, for $k = 1, \ldots, n$. □

From Theorems 5.1.2 and 5.1.3, we get analytic forms of P_s^* for the minimisation and maximisation problems. Then, using these P_s^* in $\chi_k = \left[(I - P_s^*)^{-1}\,\mathbf{c}\right]_k$, for $k = 1, \ldots, n$, will give us the minimum and maximum expected total cost for each state k of the discrete-time interval birth and death process.

5.2 Discrete-time interval constant-parameter birth and death process

Since we have obtained an analytic solution of the expected total cost for each state of a discrete-time interval birth and death process, we now look to obtain a similar analytic solution for the discrete-time interval constant-parameter birth and death process. In the standard, non-interval case, a discrete-time constant-parameter birth and death process is a specific case of the discrete-time birth and death process. The only difference between the two is that the transition probabilities for the discrete-time constant-parameter birth and death process are constant across the states of the process, as the name suggests.

The discrete-time interval constant-parameter birth and death process $(X_m, m \in \mathbb{Z}_+)$ with $n+1$ states, $S = \{0, 1, \dots, n\}$, has the interval transition probability matrix,

$$\mathbb{P} = \begin{bmatrix} [1,1] & [0,0] & [0,0] & [0,0] & \cdots & \cdots & [0,0] \\ [\underline{\mu},\overline{\mu}] & [\underline{\gamma},\overline{\gamma}] & [\underline{\lambda},\overline{\lambda}] & [0,0] & \cdots & \cdots & [0,0] \\ [0,0] & [\underline{\mu},\overline{\mu}] & [\underline{\gamma},\overline{\gamma}] & [\underline{\lambda},\overline{\lambda}] & [0,0] & \cdots & [0,0] \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ \vdots & & \ddots & \ddots & \ddots & \ddots & \vdots \\ [0,0] & \cdots & \cdots & [0,0] & [\underline{\mu},\overline{\mu}] & [\underline{\gamma},\overline{\gamma}] & [\underline{\lambda},\overline{\lambda}] \\ [0,0] & \cdots & \cdots & \cdots & [0,0] & [\underline{\mu},\overline{\mu}] & [\underline{\gamma}+\underline{\lambda},\overline{\gamma}+\overline{\lambda}] \end{bmatrix},$$

where

$$\begin{aligned} &0 < \underline{\mu} \leq \overline{\mu} \leq 1, \\ &0 \leq \underline{\gamma} \leq \overline{\gamma} \leq 1, \\ &0 \leq \underline{\lambda} \leq \overline{\lambda} \leq 1, \\ &\mu + \gamma + \lambda = 1, \qquad \text{for } \mu \in [\underline{\mu},\overline{\mu}], \gamma \in [\underline{\gamma},\overline{\gamma}], \lambda \in [\underline{\lambda},\overline{\lambda}]. \quad \text{(row sum constraint)} \end{aligned}$$

As with the discrete-time interval birth and death process, we ensure $0 < \underline{\mu}$ so that there is a positive probability of moving from state i to state $i-1$, for $i = 1, \ldots, n$. This means that the probability of absorption into state 0 is 1 from every state k, for $k = 1, \ldots, n$.

First, we note that the discrete-time interval constant-parameter birth and death process is not a special case of the discrete-time interval birth and death process, unlike in the standard, non-interval case. The reason for this is that there are only 3 parameters in the discrete-time interval constant-parameter birth and death process and the rows of the interval $\mathbb{P}$ matrix are not independent for this process. On the other hand, for the discrete-time interval birth and death process, there are 3 parameters for each row of the interval $\mathbb{P}$ matrix and hence, each row is independent of the other.

Unlike the other processes we have considered previously (the discrete-time interval Markov chain and the discrete-time interval birth and death process), the discrete-time interval constant-parameter birth and death process is the first process we encounter where the rows in the interval $\mathbb{P}$ matrix are not independent. Hence, the previous theoretical properties, such as the connection with Markov decision processes, and numerical methods we have developed and discussed for the other processes, cannot be immediately applied to this process. The other method by Blanc and den Hertog [2], which we considered in Chapters 2 and 4, also requires the rows of the $\mathbb{P}$ matrix to be independent. Thus, we are unable to use their method for this process. Instead, new theory must be developed to allow for dependence between rows. However, the general principles still remain where optimisation techniques are used to obtain the minimum and maximum objective function values on a constrained region.

Recall that we are interested in the problem of obtaining an analytic solution for the expected total cost, χ_k, for $k = 1, \ldots, n$, of the discrete-time interval constant-parameter birth and death process. First, we consider the minimisation problem. The following theorem gives us an analytic form for the optimal P_s^* for the minimisation problem.

Theorem 5.2.1. *The minimum expected total cost, χ_k, for $k = 1, \dots, n$, for the discrete-time interval constant-parameter birth and death process is obtained by solving $\chi_k = \left[(I - P_s^*)^{-1}\mathbf{c}\right]_k$, where the optimal matrix P_s^* is given by,*

$$P_s^* = \begin{bmatrix} \gamma^* & \lambda^* & 0 & \cdots & \cdots & \cdots & 0 \\ \mu^* & \gamma^* & \lambda^* & 0 & \cdots & \cdots & 0 \\ 0 & \mu^* & \gamma^* & \lambda^* & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \mu^* & \gamma^* & \lambda^* & 0 \\ 0 & \cdots & \cdots & 0 & \mu^* & \gamma^* & \lambda^* \\ 0 & \cdots & \cdots & \cdots & 0 & \mu^* & 1-\mu^* \end{bmatrix},$$

and

$$\begin{aligned} \mu^* &= \overline{\mu}, \\ \gamma^* &= \min\left(\overline{\gamma}, 1 - \overline{\mu} - \underline{\lambda}\right), \\ \lambda^* &= 1 - \overline{\mu} - \gamma^*. \end{aligned}$$

Proof. Theorem 5.1.1 was stated for the discrete-time interval birth and death process, but did not make any use of the independence between rows and so also applies to the discrete-time interval constant-parameter birth and death process. Hence, from Theorem 5.1.1, we know that $\chi_k \geq \chi_{k-1}$ for $k = 1, \dots, n$. Like the proof of Theorem 5.1.2, the process is better to move to state $i-1$ from state i, for $i = 1, \dots, n$, to minimise χ_k. This means we want to maximise μ followed by γ then allocate the remaining probability to λ. This applies to states $i = 2, \dots, n-1$ as these rows contain μ, γ and λ. However, for the boundary cases, $i = 1$ and $i = n$, the P_s matrix does not explicitly contain all of μ, γ and λ. We now consider these cases.

For $i = 1$, we note that μ is not within P_s as it is the probability of absorption to state 0 from state 1. Since we are interested in minimising χ_k, this means we seek to maximise the probability of absorption to state 0, that is, we want to maximise μ first. Next, we note that $\chi_k \geq \chi_{k-1}$ for $k = 1, \ldots, n$, so we want to maximise γ next and allocate the remaining probability to λ. Similarly for $i = n$, we seek to maximise μ first as $\chi_k \geq \chi_{k-1}$ for $k = 1, \ldots, n$. Then, set $\gamma + \lambda$ to be the remaining probability. Since it does not matter what values we have for γ and λ, as long as they are feasible, and sum up to the required total, we choose to maximise γ next and then allocate the rest to λ. Thus for all cases, we choose to maximise μ, then γ and finally λ subject to the row sum constraint.

Since we seek to maximise μ first, we can set $\mu^* = \overline{\mu}$ because of coherence (Section 4.1.3) which ensures that every element within the interval is attainable. Next, we maximise γ subject to the row sum constraint. Thus, we choose $\gamma^* = \min\left(\overline{\gamma}, 1 - \overline{\mu} - \underline{\lambda}\right)$. This ensures that the row sum constraint is satisfied whilst making γ as large as possible. Finally, we set λ to be the remaining probability. Hence, $\lambda^* = 1 - \overline{\mu} - \gamma^* \in \left[\underline{\lambda}, \overline{\lambda}\right]$.

Thus, we have

$$\begin{aligned}
\mu^* &= \overline{\mu}, \\
\gamma^* &= \min\left(\overline{\gamma}, 1 - \overline{\mu} - \underline{\lambda}\right), \\
\lambda^* &= 1 - \overline{\mu} - \gamma^*.
\end{aligned}$$

□

Just like the minimisation problem, we have the following theorem which gives us an analytic solution for the maximum expected total cost.

Theorem 5.2.2. *The maximum expected total cost, χ_k, for $k = 1, \ldots, n$, for the discrete-time interval constant-parameter birth and death process is obtained by* *solving* $\chi_k = \left[(I - P_s^*)^{-1}\mathbf{c}\right]_k$*, where the optimal matrix* P_s^* *is given by,*

$$P_s^* = \begin{bmatrix} \gamma^* & \lambda^* & 0 & \cdots & \cdots & \cdots & 0 \\ \mu^* & \gamma^* & \lambda^* & 0 & \cdots & \cdots & 0 \\ 0 & \mu^* & \gamma^* & \lambda^* & 0 & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & \cdots & 0 & \mu^* & \gamma^* & \lambda^* & 0 \\ 0 & \cdots & \cdots & 0 & \mu^* & \gamma^* & \lambda^* \\ 0 & \cdots & \cdots & \cdots & 0 & \mu^* & 1-\mu^* \end{bmatrix},$$

and

$$\begin{aligned} \lambda^* &= \overline{\lambda}, \\ \gamma^* &= \min\left(\overline{\gamma}, 1 - \underline{\mu} - \overline{\lambda}\right), \\ \mu^* &= 1 - \overline{\lambda} - \gamma^*. \end{aligned}$$

Proof. We note that the proof for this choice of P_s^* for maximisation problem follows analogously from the proof of the minimisation problem. □

In this chapter, we have explored structured Markov chains and obtained analytic solutions for the expected total cost for both the discrete-time interval birth and death process as well as the discrete-time interval constant-parameter birth and death process. In the last chapter of this thesis, we consider applying the techniques we have developed to an SIR epidemic model where we allow the parameters to vary within intervals. As with this chapter, we look to obtain analytic solutions to the performance measures of interest. Finally, we end with an investigation of using intervals as an alternative to sensitivity analysis and apply the method developed to the performance measures of interest.

Chapter 6

Markovian SIR model and sensitivity analysis

In this chapter, we consider a Markovian SIR (*susceptible-infectious-recovered*) epidemic model where we allow uncertainty in the parameter values by incorporating intervals into our model. We note that the SIR model, like the discrete-time constant-parameter birth and death process, is not a row-independent model as there are only 2 parameters, β and γ, throughout the entire model. Hence the work in Chapters 3 and 4, in particular the connection with Markov decision processes and numerical methods developed, cannot be immediately applied to this model. Instead, we extend the ideas and techniques developed in previous chapters to allow for the row dependence in the SIR model.

Common performance measures for an SIR epidemic model include the *mean final epidemic size* and the *mean epidemic duration* of an epidemic. We obtain an analytic solution for the mean final epidemic size and develop a numerical method for the mean epidemic duration as an analytic solution is not obtainable. Last, we consider the use of intervals as an alternative to sensitivity analysis and develop a method to accomplish this before applying this method to our epidemic model.

Our use of intervals for sensitivity analysis is related to recent work by Roberts [19] where he considers the deterministic SIR model with R_0, the *basic reproduction number* [10]. He considers a symmetric distribution on R_0 and looked to evaluate the consequences of this uncertainty on the dynamics of the epidemic, in particular, the mean final epidemic size.

6.1 Markovian SIR epidemic model

An SIR model allows us to model the spread of a disease through a population, where the individuals of the population are classified as susceptible (S), infectious (I) or recovered (R). The SIR model is a continuous-time Markov chain, $X(t)$, with transition rates

$$\begin{aligned} q_{(s,i),(s-1,i+1)} &= \beta i \frac{s}{N-1}, && \text{(infection)} \\ q_{(s,i),(s,i-1)} &= \gamma i, && \text{(recovery)} \end{aligned} \tag{6.1.1}$$

where $\beta > 0$ is the effective transmission rate parameter (susceptible becoming infectious), $\gamma > 0$ is the rate of recovery of an infectious individual (infectious becoming recovered), N is the population size, i denotes the number of infectious individuals and s denotes the number of susceptible individuals. Note that we require $\beta > 0$ and $\gamma > 0$ to have an SIR model. If either β or γ is 0, then we would not have an SIR model but an IR (*infectious-recovered*) or SI (*susceptible-infectious*) model instead.

The state space, S, of this system is,

$$S = \{(s,i) : 0 \leq s, i \leq N, 0 \leq s+i \leq N\},$$

where the set of states $A = \{(s,0) : s = 0, \ldots, N\}$ correspond to the absorbing states and the other states of the system, denoted by C, are the transient states. The set A corresponds to the absorbing states since there are no more infectious individuals to continue the spread of the disease; thus, ending the epidemic. For further information regarding the SIR model and other epidemic models, see [10] and [11].

We are interested in calculating various performance measures for the SIR model such as the mean final epidemic size and mean epidemic duration. However, we note that the current form of the SIR model is a two-dimensional model. Hence to allow for calculations to be performed with the SIR model, we need a mapping which converts the current two-dimensional state-space to a one-dimensional state-space. A possible mapping is

$$(s,i) \to y = Ni - \tfrac{1}{2}i(i-3) + s + 1, \tag{6.1.2}$$

which allows us to form a generator matrix, Q, for the SIR model, where the first $N+1$ rows of the matrix correspond to the absorbing states of the model. Thus, bringing this problem back in line with the form for a standard continuous-time Markov chain with generator matrix Q.

The following is an example of the Q matrix for a population of $N = 2$ individuals with the original two-dimensional states displayed,

$$Q = \begin{array}{c} \\ (0,0) \\ (1,0) \\ (2,0) \\ (0,1) \\ (1,1) \\ (0,2) \end{array} \!\!\! \begin{array}{c} \begin{array}{cccccc} (0,0) & (1,0) & (2,0) & (0,1) & (1,1) & (0,2) \end{array} \\ \left(\begin{array}{cccccc} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ \gamma & 0 & 0 & -\gamma & 0 & 0 \\ 0 & \gamma & 0 & 0 & -(\gamma+\beta) & \beta \\ 0 & 0 & 0 & 2\gamma & 0 & -2\gamma \end{array} \right) \end{array}.$$

As before, we are interested in extending this model to allow for an interval representation of the parameters, β and γ, in the SIR model.

Consider an interval SIR model with a population of N individuals. Since we are interested in allowing intervals around the parameters β and γ, the transition rates for the interval SIR model are given by (6.1.1) where $\beta \in \left[\underline{\beta}, \overline{\beta}\right]$ and $\gamma \in \left[\underline{\gamma}, \overline{\gamma}\right]$. These transition rates form the interval generator matrix, $\mathbb{Q}$, for the interval SIR model. Note that we impose the restriction that $\underline{\beta}, \underline{\gamma} > 0$ to be consistent with the model defined in equation (6.1.1).

Now that we have defined an interval SIR epidemic model, we are interested in calculating various performance measures such as the *interval mean final epidemic size* and the *interval mean epidemic duration* of an interval SIR epidemic model. First, we consider the mean final epidemic size.

6.2 Mean final epidemic size

Let Z be the random variable describing the total number of infections that have occurred over the course of the epidemic. Then, the mean final size of an epidemic, $\zeta_k = E\left[Z | X(0) = k\right]$ for $k \in S$, is the expected total number of infections that have occurred over the course of the epidemic, conditional on starting in state k.

To enable us to calculate the mean final epidemic size easily, we introduce a new state 0 to the system. Thus, giving us a new state space $S' = \{0\} \cup S$. We make this state the new absorbing state of the process by specifying that $c_0 = 0$ and $q_{a,0} = 1$ for all $a \in A$ and all other $q_{i,0} = q_{0,j} = 0$. This effectively means that the states in A are now transient. However, as we have set $q_{a,0} = 1$ for all $a \in A$ and all other $q_{i,0} = q_{0,j} = 0$, this means that once the process enters any state $a \in A$, it will move to state 0 after spending a mean time of 1 in the state a and incur some cost c_a, for $a \in A$; this cost corresponds to the number of recovered individuals in the state a. For example, consider a population of size 4 and the absorbing state, $(1, 0)$, meaning the number of susceptible individuals is 1 and the number of infectious individuals is 0. The number of recovered individuals in the absorbing state $(1, 0)$ for a population of size 4 is 3 and so for this example, we set $c_{(1,0)} = 3$.

Thus, the mean final epidemic size, ζ_k for every state $k \in S$, of a standard, non-interval SIR model can be calculated using

$$\boldsymbol{\zeta} = -Q_*^{-1}\mathbf{c}, \tag{6.2.3}$$

where Q_* is the new Q matrix just described, but restricted to the original state space S and $\mathbf{c}$ contains the number of recovered individuals in the absorbing states. Then, it is simply a matter of using the mapping (6.1.2) to obtain the mean final epidemic size for the state of interest to us. Recall that as we have defined the parameters $\beta, \gamma > 0$, this calculation is well defined and Q_*^{-1} exists.

Here, we consider an example to show the form of Q_* and $\mathbf{c}$. For a population of size $N = 2$, the corresponding Q_* matrix and vector $\mathbf{c}$ are given by,

$$Q_* = \begin{bmatrix} -1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 0 & 0 \\ \gamma & 0 & 0 & -\gamma & 0 & 0 \\ 0 & \gamma & 0 & 0 & -(\gamma+\beta) & \beta \\ 0 & 0 & 0 & 2\gamma & 0 & -2\gamma \end{bmatrix} \quad \text{and} \quad \mathbf{c} = \begin{bmatrix} 2 \\ 1 \\ 0 \\ 0 \\ 0 \\ 0 \end{bmatrix}.$$

As in previous chapters, we look to obtain interval versions of the performance measures of interest. Here, the performance measure of interest is the *interval mean final epidemic size* and so we follow the logic used in the standard, non-interval SIR model to enable easy calculation of the *interval mean final epidemic size.* The main thing to note is that for the interval $\mathbb{Q}$ matrix, $q_{a,0} = [1, 1]$ for all $a \in A$ and all other $q_{i,0} = q_{0,j} = [0, 0]$.

We define the *interval mean final epidemic size* to be

$$[\underline{\boldsymbol{\zeta}}, \overline{\boldsymbol{\zeta}}] = -\mathbb{Q}_*^{-1}\mathbf{c}, \tag{6.2.4}$$

where $\mathbb{Q}_*$ is the new interval $\mathbb{Q}$ matrix just described, but restricted to the original state space S and $\mathbf{c}$ is a non-interval vector containing the number of recovered individuals in the absorbing states. As we have previously required $\underline{\beta}, \underline{\gamma} > 0$ in the interval generator matrix $\mathbb{Q}$ for the interval SIR model, $\mathbb{Q}_*^{-1}$ is well defined and hence so is (6.2.4).

Recall that we choose to solve minimisation and maximisation problems to obtain the lower and upper bounds for the mean final epidemic size. Hence for each state $k \in S$, we want to minimise and maximise the following objective function,

$$\zeta_k = -\left[Q_*^{-1}\mathbf{c}\right]_k,$$

subject to the constraints

$$\underline{\beta} \leq \beta \leq \overline{\beta},$$
$$\underline{\gamma} \leq \gamma \leq \overline{\gamma}.$$

Let us first consider the minimisation problem. Intuitively, if β is small and γ is large, then this means there is a slower rate of infection and a higher rate of recovery. Hence, if $\beta = \underline{\beta}$ and $\gamma = \overline{\gamma}$, then one would expect the mean final epidemic size to be minimised.

For the maximisation problem, the opposite situation occurs. For a larger value of β, there is a higher rate of infection, and a smaller value of γ means recovery occurs at a slower rate. Thus, we expect the mean final epidemic size to be maximised when $\beta = \overline{\beta}$ and $\gamma = \underline{\gamma}$.

We formalise this intuition in the following theorem.

Theorem 6.2.1. *For an interval SIR model, the minimum mean final epidemic size, $\underline{\zeta}_k$ for state $k \in S$, occurs when $\beta = \underline{\beta}$ and $\gamma = \overline{\gamma}$ and the maximum mean final epidemic size, $\overline{\zeta}_k$ for state $k \in S$, occurs when $\beta = \overline{\beta}$ and $\gamma = \underline{\gamma}$.*

Proof. Recall that the transition rates for the SIR model are given as follows,

$$\begin{aligned} q_{(s,i),(s-1,i+1)} &= \beta i \frac{s}{N-1}, && \text{(infection)} \\ q_{(s,i),(s,i-1)} &= \gamma i. && \text{(recovery)} \end{aligned}$$

To form the interval generator matrix $\mathbb{Q}$ for the SIR model, we use the mapping (6.1.2) to convert the two-dimensional state space to a one-dimensional state space and also allow β and γ to vary within the intervals $\left[\underline{\beta}, \overline{\beta}\right]$ and $\left[\underline{\gamma}, \overline{\gamma}\right]$ respectively. Hence, we know that for a realisation $Q \in \mathbb{Q}$, its parameters β and γ must obey the following conditions,

$$\begin{aligned} \underline{\beta} \leq \beta \leq \overline{\beta}, \\ \underline{\gamma} \leq \gamma \leq \overline{\gamma}. \end{aligned}$$

First consider minimising the mean final epidemic size ζ_k for a fixed state $k \in S$. Since the costs of all states in the set of transient states C are 0 and the mean time spent in each state in the set of absorbing states A is 1, we can convert our continuous-time Markov chain into a discrete-time Markov chain by considering the jump chain without affecting the value of $\underline{\zeta}_k$.

For a given realisation $Q \in \mathbb{Q}$, the one-step transition probabilities for the associated jump chain can be calculated as follows:

$$p_{j,k} = -\frac{q_{j,k}}{q_{j,j}}, \qquad j,k \in S, k \neq j, \qquad \text{and} \qquad p_{j,j} = 0, \qquad j \in S. \tag{6.2.5}$$

We also note that the mean final epidemic size is the total number of infections during the epidemic. Thus, minimising the mean final epidemic size corresponds to minimising the total number of infections, which can be achieved by minimising the probability of infection for every state $i \in S$ of the system. Since the SIR model consists of only two parameters β and γ, which relate to every state, minimising the mean final epidemic size involves minimising the probability of infection for each state simultaneously, if possible.

Consider the probability of infection, p_{inf}, for each state of the SIR model. Using the transformation (6.2.5), we have

$$\begin{aligned} p_{inf} &= \frac{\dfrac{\beta is}{N-1}}{\gamma i + \dfrac{\beta is}{N-1}} \\ &= \frac{\beta is}{(N-1)\gamma i + \beta is}. \end{aligned}$$

Since we are seeking to minimise the probability of infection and know that p_{inf} depends on the parameters, β and γ, we study the change in p_{inf} as we vary β and γ in their respective intervals. That is, we consider the partial derivatives of p_{inf} with respect to β and γ.

First, we look at the partial derivative of p_{inf} with respect to β,

$$\begin{aligned} \frac{\partial p_{inf}}{\partial \beta} &= \frac{is\left[(N-1)\gamma i + \beta is\right] - \beta is(is)}{\left[(N-1)\gamma i + \beta is\right]^2} \\ &= \frac{is\left[(N-1)\gamma i\right]}{\left[(N-1)\gamma i + \beta is\right]^2} \\ &\geq 0, \end{aligned}$$

for all values of $(s, i) \in S$ and N as we know these variables are all non-negative and all values of $\beta, \gamma > 0$.

Now, consider the partial derivative of p_{inf} with respect to γ,

$$\begin{aligned} \frac{\partial p_{inf}}{\partial \gamma} &= \frac{-\beta is\left[(N-1)i\right]}{\left[(N-1)\gamma i + \beta is\right]^2} \\ &\leq 0, \end{aligned}$$

for all values of $(s, i) \in S$ and N as we know these variables are all non-negative and all values of $\beta, \gamma > 0$.

Since we have shown that $\dfrac{\partial p_{inf}}{\partial \beta} \geq 0$, this means that as we decrease β, the probability of infection, p_{inf}, decreases. Hence, the minimum probability of infection occurs when $\beta = \underline{\beta}$, for all states.

We have also seen that $\frac{\partial p_{inf}}{\partial \gamma} \leq 0$ which means that as γ increases, the probability of infection, p_{inf}, decreases. Thus, the minimum probability of infection occurs when $\gamma = \overline{\gamma}$, for all states.

If we combine the above results and recall that the minimum mean final epidemic size, $\underline{\zeta}_k$ for $k \in S$, corresponds to minimising the probability of infection in all states simultaneously, then we have that $\underline{\zeta}_k$ is achieved when $\beta = \underline{\beta}$ and $\gamma = \overline{\gamma}$.

Now, if we look to maximise the mean final epidemic size, this means we seek to maximise the total number of infections which can be achieved by maximising the probability of infection, p_{inf}, for every state of the system simultaneously. We have already shown that $\frac{\partial p_{inf}}{\partial \beta} \geq 0$ and $\frac{\partial p_{inf}}{\partial \gamma} \leq 0$. Hence, it follows that the maximum mean final epidemic size occurs when $\beta = \overline{\beta}$ and $\gamma = \underline{\gamma}$. $\square$

6.3 Mean epidemic duration

The other performance measure of interest for the SIR model is the mean epidemic duration. Let T be the random variable describing the epidemic duration, that is, the time taken to first reach an absorbing state. Then, the mean epidemic duration, $\tau_k = E\left[T|X(0) = k\right]$ for $k \in C$, is the expected time taken to first reach an absorbing state, conditional on starting in state k. We calculate the mean epidemic duration, τ_k for every state $k \in C$, of a standard, non-interval SIR model, using

$$\boldsymbol{\tau} = -Q_c^{-1}\mathbf{1}, \tag{6.3.6}$$

where Q_c is Q restricted to the set of transient states, C, of the system and $\boldsymbol{\tau}$ is the mean epidemic duration vector containing the mean epidemic duration for each transient state $k \in C$. Then, it is simply a matter of using the mapping (6.1.2) to obtain the mean epidemic duration for the state of interest to us. Recall that as we have defined the parameters $\beta, \gamma > 0$, this calculation is well defined and Q_c^{-1} exists.

Here we consider an example of the form of the Q_c matrix for a population of $N = 2$ individuals,

$$Q_c = \begin{bmatrix} -\gamma & 0 & 0 \\ 0 & -(\gamma+\beta) & \beta \\ 2\gamma & 0 & -2\gamma \end{bmatrix}.$$

Like the mean final epidemic size, we are interested in obtaining the *interval mean epidemic duration* which is defined to be

$$[\underline{\boldsymbol{\tau}}, \overline{\boldsymbol{\tau}}] = -\mathbb{Q}_c^{-1}\mathbf{1}, \tag{6.3.7}$$

where $\mathbb{Q}_c$ is the interval generator matrix $\mathbb{Q}$ restricted to the set of transient states C. Again, since we require $\underline{\beta}, \underline{\gamma} > 0$ in the interval generator matrix $\mathbb{Q}$ for the interval SIR model, $\mathbb{Q}_c^{-1}$ is well defined and hence so is (6.3.7).

We solve the above problem by using optimisation techniques, where we solve a minimisation problem to obtain the lower bounds and solve a maximisation problem to obtain the upper bounds on the mean epidemic duration. Unlike the mean final epidemic size, we are not able to prove analytic results for the mean epidemic duration. We also note that the minimum and maximum mean epidemic duration do not necessarily occur at the intuitive values. Hence, we consider a numerical method instead. In the following, we consider the minimisation problem and note that the maximisation problem follows similarly.

For each state $k \in C$, we would like to solve the following problem,

$$\min \tau_k = -\left[Q_c^{-1}\mathbf{1}\right]_k,$$

subject to

$$\underline{\beta} \leq \beta \leq \overline{\beta},$$

$$\underline{\gamma} \leq \gamma \leq \overline{\gamma}.$$

6.3.1 Numerical method

The numerical method for the mean epidemic duration closely follows the numerical method presented in Chapter 4. Hence, we direct the reader back to Chapter 4 for details of concepts used and only explicitly discuss the differences between the two methods.

Pre-processing

As discussed in Chapter 4, before we solve the minimisation problem, we first need to pre-process the intervals to ensure that the true solution is always contained within the intervals. We also consider pre-processing to reduce the number of parameters in the model, which shrinks the search space of the optimisation algorithm; hence, making the problem easier to solve. The concepts considered in the pre-processing stage for the numerical method in Chapter 4 were *degeneracy*, *outward rounding* and *coherence*. For this method, we check and remove degenerate intervals from the problem as well as apply outward rounding to the intervals $\left[\underline{\beta}, \overline{\beta}\right]$ and $\left[\underline{\gamma}, \overline{\gamma}\right]$. Note that since we do not have an equality constraint, all elements in the intervals $\left[\underline{\beta}, \overline{\beta}\right]$ and $\left[\underline{\gamma}, \overline{\gamma}\right]$ can be attained. Hence, there is no need for coherence to be applied.

MATLAB `fmincon` formulation

After pre-processing the intervals on β and γ, we look to solve the minimisation problem using MATLAB's `fmincon` function. Recall from Section 4.3 that the `fmincon` function allows us to specify the objective function and constraints on our problem. Furthermore, it allows for user-specification of the optimisation algorithm, specifying a gradient of the objective function, an initial point from which the algorithm starts its search and tolerances which determine when the `fmincon` function stops iterating. In the following, we present the form of the gradient for the SIR model and consider the initial point sent to the `fmincon` function. We do not explicitly discuss the choice of optimisation algorithm or tolerances as these follow from Chapter 4.

Gradient

Since the `fmincon` function allows for user-specification of the gradient of the objective function, we have chosen to use this option for the same reasons detailed in Section 4.3.2.

Recall that the objective function for each state k, for $k \in C$, of the SIR model is given by,

$$\tau_k = -\left[Q_c^{-1}\mathbf{1}\right]_k .$$

The following theorem specifies the gradient of the objective function for each state $k \in C$.

Theorem 6.3.1. *For each state $k \in C$, the gradient of τ_k is a 2×1 vector given by*

$$\begin{aligned}\nabla\tau_k &= \begin{bmatrix} \dfrac{\partial\tau_k}{\partial\beta} \\ \dfrac{\partial\tau_k}{\partial\gamma} \end{bmatrix} \\ &= \begin{bmatrix} \left\{Q_c^{-1}\dfrac{\partial Q_c}{\partial\beta}Q_c^{-1}\mathbf{1}\right\}_k \\ \left\{Q_c^{-1}\dfrac{\partial Q_c}{\partial\gamma}Q_c^{-1}\mathbf{1}\right\}_k \end{bmatrix}.\end{aligned}$$

Proof. First, let us state a few properties of matrix differentiation [17] required in the proof:

$$\partial(\mathbf{XY}) = (\partial\mathbf{X})\mathbf{Y} + \mathbf{X}(\partial\mathbf{Y}), \tag{6.3.8}$$

$$\partial\left(\mathbf{X}^{-1}\right) = -\mathbf{X}^{-1}\left(\partial\mathbf{X}\right)\mathbf{X}^{-1}. \tag{6.3.9}$$

Consider the partial derivative of τ_k with respect to β.

$$\begin{aligned}
\frac{\partial \tau_k}{\partial \beta} &= -\frac{\partial\, [Q_c^{-1}\mathbf{1}]_k}{\partial \beta} \\
&= -\left\{\frac{\partial Q_c^{-1}}{\partial \beta}\mathbf{1}\right\}_k - \left\{Q_c^{-1}\frac{\partial \mathbf{1}}{\partial \beta}\right\}_k, && \text{from (6.3.8)} \\
&= -\left\{-Q_c^{-1}\,\frac{\partial Q_c}{\partial \beta}\,Q_c^{-1}\mathbf{1}\right\}_k, && \text{using (6.3.9) and } \frac{\partial \mathbf{1}}{\partial \beta} = \mathbf{0} \\
&= \left\{Q_c^{-1}\,\frac{\partial Q_c}{\partial \beta}\,Q_c^{-1}\mathbf{1}\right\}_k.
\end{aligned}$$

The derivation of the partial derivative of τ_k with respect to γ follows similarly to the one presented for β. □

We note that the form of the partial derivatives look complicated. However, we know from the transition rates of the SIR model that the matrix Q_c is linear in β and γ. Hence, $\frac{\partial Q_c}{\partial \beta}$ is the Q_c matrix with $\beta = 1$ and $\gamma = 0$, and $\frac{\partial Q_c}{\partial \gamma}$ is the Q_c matrix with $\beta = 0$ and $\gamma = 1$.

Initial point

Since we do not know the shape of the function for the mean epidemic duration, we cannot be sure if there are local minima or maxima in the interior of the feasible region. Hence, to ensure that we obtain the global minima for the minimisation problem, we consider all vertices of the feasible region as initial points. If there are no degenerate intervals, then we have four initial points: $\underline{\beta}$ and $\underline{\gamma}$, $\underline{\beta}$ and $\overline{\gamma}$, $\overline{\beta}$ and $\underline{\gamma}$, $\overline{\beta}$ and $\overline{\gamma}$. If β or γ lies within a degenerate interval, then we are only left with two initial points.

Since we have multiple initial points, we consider each initial point separately and provide it to `fmincon`. As a result, we obtain a set of optimal solutions, each of which corresponds to an initial point. Then, we take the minimum of this set of optimal solutions to be the minimum on the feasible region.

Recall from Chapter 4 that after obtaining an optimal solution from `fmincon`, we purify the optimal decision variables to ensure that they occurred at a vertex of the feasible region (due to Theorem 3.5.2). However, Theorem 3.5.2 does not hold for the SIR model. Hence, the purification step must be ignored for the SIR model.

Optimality conditions

To ensure that the solution obtained from `fmincon` is in fact optimal, we check that first and second-order optimality conditions are satisfied. First, we state the minimisation problem we are interested in solving.

For a given state $k \in C$,

$$\min \tau_k = -\left[Q_c^{-1}\mathbf{1}\right]_k,$$

subject to

$$\begin{aligned} \ell_1 &= \underline{\beta} - \beta \leq 0, \\ \ell_2 &= \underline{\gamma} - \gamma \leq 0, \\ u_1 &= \beta - \overline{\beta} \leq 0, \\ u_2 &= \gamma - \overline{\gamma} \leq 0. \end{aligned}$$

Recall in Section 2.6, we defined the first-order necessary conditions, or the Karush-Kuhn-Tucker (KKT) conditions, for a point to be a local minimiser of the objective function, τ_k. Here, we re-state Theorem 2.6.2 using the terms and constraints for our optimisation problem and refer the reader back to Section 2.6 for definitions of terms used. Let $\mathbf{x} = (\beta, \gamma)$.

Theorem 6.3.2 (Karush-Kuhn-Tucker (KKT) Theorem)**.** *Let τ_k, for all $k \in C$, ℓ_i, u_i, for all $i = 1, 2$ be sufficiently differentiable (that is, $\nabla\tau_k$, for all $k \in C$, $\nabla\ell_i$, ∇u_i, for all $i = 1, 2$ exist). Let $\mathbf{x}^* = (\beta^*, \gamma^*)$ be a regular point and a local* *minimiser of $\tau_k(\mathbf{x})$ such that $\ell_i(\mathbf{x}) \leq 0, u_i(\mathbf{x}) \leq 0$ for all $i = 1, 2$. Then, there exists $\lambda_{\ell_i} \in \mathbb{R}$ and $\lambda_{u_i} \in \mathbb{R}$ for all $i = 1, 2$ such that:*

1. $\nabla \tau_k(\mathbf{x}^*) + \sum_{i=1}^{2} \lambda_{\ell_i} \nabla \ell_i(\mathbf{x}^*) + \sum_{i=1}^{2} \lambda_{u_i} \nabla u_i(\mathbf{x}^*) = \mathbf{0}$,
2. $\lambda_{\ell_i} \ell_i(\mathbf{x}^*) = 0$ *and* $\lambda_{u_i} u_i(\mathbf{x}^*) = 0$, *for all* $i = 1, 2$, *and*
3. $\lambda_{\ell_i}, \lambda_{u_i} \geq 0$ *for all* $i = 1, 2$.

For the inequality constraints ℓ_i and u_i, at least one of the constraint or the corresponding λ must be zero. Hence, we can specify $L(\mathbf{x})$ and $U(\mathbf{x})$ to be the set of active lower bound and upper bound constraints at $\mathbf{x}$ respectively, that is, $L(\mathbf{x}) = \{i : \ell_i(\mathbf{x}) = 0\}$ and $U(\mathbf{x}) = \{i : u_i(\mathbf{x}) = 0\}$. Hence, the first condition in Theorem 6.3.2 reduces to

$$\nabla \tau_k(\mathbf{x}^*) + \sum_{i \in L(\mathbf{x}^*)} \lambda_{\ell_i} \nabla \ell_i(\mathbf{x}^*) + \sum_{i \in U(\mathbf{x}^*)} \lambda_{u_i} \nabla u_i(\mathbf{x}^*) = \mathbf{0}.$$

From Section 2.6, we recall that for $\mathbf{x}^*$ to be a regular point, we require the set of vectors

$$\Big\{ \nabla \ell_i(\mathbf{x}^*) \text{ for all } i \in L(\mathbf{x}^*), \nabla u_j(\mathbf{x}^*) \text{ for all } j \in U(\mathbf{x}^*) \Big\}$$

to be linearly independent.

However, we note that we do not have a regular point for our problem. As discussed in Section 4.6, instead, we consider second-order sufficient conditions to check that $\mathbf{x}^*$ is a minimiser of the objective function. Here, we re-state Theorem 2.6.3 using the terms and constraints for our optimisation problem and refer the reader back to Section 2.6 for definitions of terms used.

The Lagrangian function for this optimisation problem is given by,

$$l(\mathbf{x}, \boldsymbol{\lambda}_{\boldsymbol{\ell}}, \boldsymbol{\lambda}_{\boldsymbol{u}}) = \tau_k(\mathbf{x}) + \sum_{i=1}^{2} \lambda_{\ell_i} \ell_i(\mathbf{x}) + \sum_{i=1}^{2} \lambda_{u_i} u_i(\mathbf{x}),$$

and so the Hessian matrix of $l(\mathbf{x}, \boldsymbol{\lambda_\ell}, \boldsymbol{\lambda_u})$ with respect to $\mathbf{x}$ is

$$\mathbf{L}(\mathbf{x}, \boldsymbol{\lambda_\ell}, \boldsymbol{\lambda_u}) = H\tau_k(\mathbf{x}) + \sum_{i=1}^{2} \lambda_{\ell_i} H\ell_i(\mathbf{x}) + \sum_{i=1}^{2} \lambda_{u_i} Hu_i(\mathbf{x}),$$

where $H\tau_k(\mathbf{x})$ is the Hessian matrix of τ_k at $\mathbf{x}$ and similarly for $H\ell_i(\mathbf{x})$ and $Hu_i(\mathbf{x})$.

Here we also consider the set $T(\mathbf{x}^*, \boldsymbol{\lambda_\ell}, \boldsymbol{\lambda_u})$ which is used in Theorem 6.3.3.

$$T(\mathbf{x}^*, \boldsymbol{\lambda_\ell}, \boldsymbol{\lambda_u}) = \Big\{ \mathbf{y} : \nabla \ell_i(\mathbf{x}^*)^T \mathbf{y} = 0 \text{ for all } i \in \tilde{L}(\mathbf{x}^*, \boldsymbol{\lambda_\ell}) \\ \text{and } \nabla u_j(\mathbf{x}^*)^T \mathbf{y} = 0 \text{ for all } j \in \tilde{U}(\mathbf{x}^*, \boldsymbol{\lambda_u}) \Big\},$$

where $\tilde{L}(\mathbf{x}^*, \boldsymbol{\lambda_\ell}) = \{i : \ell_i(\mathbf{x}^*) = 0, \lambda_{\ell_i} > 0\}$ and $\tilde{U}(\mathbf{x}^*, \boldsymbol{\lambda_u}) = \{i : u_i(\mathbf{x}^*) = 0, \lambda_{u_i} > 0\}$.

The second-order sufficient conditions for a point $\mathbf{x}^*$ to be a strict local minimiser of τ_k is given in the following theorem.

Theorem 6.3.3 (Second-Order Sufficient Conditions)**.** *Suppose that τ_k, for all $k \in C$, ℓ_i, u_i, for all $i = 1, 2$ are sufficiently differentiable (that is, $H\tau_k$, for all $k \in C$, $H\ell_i$, Hu_i, for all $i = 1, 2$ exist) and there exists a feasible point $\mathbf{x}^*$ and $\lambda_{\ell_i} \in \mathbb{R}$ and $\lambda_{u_i} \in \mathbb{R}$ for all $i = 1, 2$ such that:*

1. $\nabla \tau_k(\mathbf{x}^*) + \sum_{i=1}^{2} \lambda_{\ell_i} \nabla \ell_i(\mathbf{x}^*) + \sum_{i=1}^{2} \lambda_{u_i} \nabla u_i(\mathbf{x}^*) = \mathbf{0}$,
2. $\lambda_{\ell_i} \ell_i(\mathbf{x}^*) = 0$ *and* $\lambda_{u_i} u_i(\mathbf{x}^*) = 0$, *for all* $i = 1, 2$,
3. $\lambda_{\ell_i}, \lambda_{u_i} \geq 0$ *for all* $i = 1, 2$, *and*
4. *For all* $\boldsymbol{y} \in T(\mathbf{x}^*, \boldsymbol{\lambda_\ell}, \boldsymbol{\lambda_u}), \boldsymbol{y} \neq \mathbf{0}$, *we have* $\boldsymbol{y}^T \mathbf{L}(\mathbf{x}^*, \boldsymbol{\lambda_\ell}, \boldsymbol{\lambda_u}) \boldsymbol{y} > 0$.

Then, $\mathbf{x}^*$ *is a strict local minimiser of* τ_k *subject to* $\ell_i(\mathbf{x}) \leq 0, u_i(\mathbf{x}) \leq 0$ *for all* $i = 1, 2$.

As discussed previously in Section 4.6, due to the tolerances (for example, $TolFun = 10^{-6}$) placed on `fmincon` and the desire to be more confident with our results, we have implemented our own method for ensuring that the optimality conditions are satisfied. In the following sections, we detail two methods. The first checks that the KKT optimality conditions 1, 2 and 3 are satisfied and the second checks that second-order sufficient conditions are satisfied.

Checking the KKT optimality conditions 1, 2 and 3

We check that the first-order optimality conditions are satisfied by solving for the Lagrange multipliers, λ_{ℓ_i} and λ_{u_i}, for $i = 1, 2$, using the following equation,

$$\nabla \tau_k(\mathbf{x}^*) + \sum_{i \in L(\mathbf{x})} \lambda_{\ell_i} \nabla \ell_i(\mathbf{x}^*) + \sum_{i \in U(\mathbf{x})} \lambda_{u_i} \nabla u_i(\mathbf{x}^*) = \mathbf{0},$$

and ensuring that $\lambda_{\ell_i}, \lambda_{u_i} \geq 0$. Furthermore, we know that for all possible vectors $\mathbf{x}$ in the feasible region,

$$\nabla \ell_1 = \begin{bmatrix} -1 \\ 0 \end{bmatrix}, \quad \nabla \ell_2 = \begin{bmatrix} 0 \\ -1 \end{bmatrix}, \quad \nabla u_1 = \begin{bmatrix} 1 \\ 0 \end{bmatrix}, \quad \nabla u_2 = \begin{bmatrix} 0 \\ 1 \end{bmatrix}.$$

In the following, we assume both β and γ lie within non-degenerate intervals. Hence, they are not removed from the optimisation problem and both are considered when checking the KKT conditions 1, 2 and 3. Note that for problems where either β or γ are degenerate, the cases needing to be considered are subsets of the cases for non-degenerate intervals on both parameters.

There are a number of cases which need to be considered when solving for the Lagrange multipliers. These include 2 active constraints, 1 active constraint and no active constraints.

1. 2 active constraints

There are 3 sub-cases within this case that need to be considered.

Case 1: Both upper bounds are active. That is, $\beta = \overline{\beta}$ and $\gamma = \overline{\gamma}$. Thus, we want to solve the following to obtain λ_{u_1} and λ_{u_2},

$$\begin{aligned}\lambda_{u_1}\nabla u_1(\mathbf{x}^*) + \lambda_{u_2}\nabla u_2(\mathbf{x}^*) &= -\nabla\tau_k(\mathbf{x}^*),\\ \Rightarrow \begin{bmatrix}1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\lambda_{u_1}\\ \lambda_{u_2}\end{bmatrix} &= -\nabla\tau_k(\mathbf{x}^*).\end{aligned}$$

Case 2: Both lower bounds are active. That is, $\beta = \underline{\beta}$ and $\gamma = \underline{\gamma}$. Thus, we want to solve the following to obtain λ_{ℓ_1} and λ_{ℓ_2},

$$\begin{aligned}\lambda_{\ell_1}\nabla \ell_1(\mathbf{x}^*) + \lambda_{\ell_2}\nabla \ell_2(\mathbf{x}^*) &= -\nabla\tau_k(\mathbf{x}^*),\\ \Rightarrow \begin{bmatrix}-1 & 0\\ 0 & -1\end{bmatrix}\begin{bmatrix}\lambda_{\ell_1}\\ \lambda_{\ell_2}\end{bmatrix} &= -\nabla\tau_k(\mathbf{x}^*).\end{aligned}$$

Case 3: One upper and one lower bound active. For example, $\beta = \underline{\beta}$ and $\gamma = \overline{\gamma}$ or $\beta = \overline{\beta}$ and $\gamma = \underline{\gamma}$. For the first example, we want to solve the following to obtain λ_{ℓ_1} and λ_{u_2},

$$\begin{aligned}\lambda_{\ell_1}\nabla u_1(\mathbf{x}^*) + \lambda_{u_2}\nabla u_2(\mathbf{x}^*) &= -\nabla\tau_k(\mathbf{x}^*),\\ \Rightarrow \begin{bmatrix}-1 & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}\lambda_{u_1}\\ \lambda_{u_2}\end{bmatrix} &= -\nabla\tau_k(\mathbf{x}^*).\end{aligned}$$

We note that a similar equation is used to solve the second example.

For all the above cases, we determine if $\mathbf{x}^*$ obeys the first-order optimal KKT conditions 1, 2 and 3 by checking $\lambda_{\ell_i}, \lambda_{u_i} \geq 0$ for all $i = 1, 2$.

2. 1 active constraint

For this case, we have that the optimal point $\mathbf{x}^*$ lies on a boundary of the feasible region. The boundary could be any one of these: $\beta = \underline{\beta}$, $\beta = \overline{\beta}$, $\gamma = \overline{\gamma}$ or $\gamma = \underline{\gamma}$.

Let us consider $\beta = \underline{\beta}$. To obtain a value for λ_{ℓ_1}, we solve

$$\begin{aligned} \lambda_{\ell_1} \nabla \ell_1(\mathbf{x}^*) &= -\nabla \tau_k(\mathbf{x}^*), \\ \Rightarrow \lambda_{\ell_1} \begin{bmatrix} -1 \\ 0 \end{bmatrix} &= -\nabla \tau_k(\mathbf{x}^*). \end{aligned}$$

To determine if the point $\mathbf{x}^*$ obeys the first-order optimal KKT conditions 1, 2 and 3, we check that $\lambda_{\ell_1} \geq 0$ and also check that the second element of $\nabla \tau_k$ is zero.

For the other possible active constraints, we solve for λ_{ℓ_i} or λ_{u_i} and check first-order optimality in an analogous way

3. No active constraints

Here, the optimal point $\mathbf{x}^*$ lies in the interior of the feasible region. By condition 2 of the KKT conditions (Theorem 6.3.2), we know that $\lambda_{\ell_i} = 0$ and $\lambda_{u_i} = 0$ for $i = 1, 2$. Hence, condition 1 of the KKT conditions reduces to $\nabla \tau_k(\mathbf{x}^*) = \mathbf{0}$. This means we need to check that both elements of the gradient are zero at $\mathbf{x}^*$ to check that we have a first-order optimal point.

Checking second-order optimality conditions

After determining that $\mathbf{x}^*$ is a first-order optimal point, we need to check second-order optimality conditions to ensure we are at a minimum as opposed to a maximum or saddle point, which are also first-order optimal points.

Before we detail the method to check for second-order optimality, recall from the second-order sufficient conditions (Theorem 6.3.3) that the 4^{th} condition requires the Hessian of the Lagrangian which is given by,

$$\mathbf{L}(\mathbf{x}, \boldsymbol{\lambda_\ell}, \boldsymbol{\lambda_u}) = H\tau_k(\mathbf{x}) + \sum_{i=1}^{2} \lambda_{\ell_i} H\ell_i(\mathbf{x}) + \sum_{i=1}^{2} \lambda_{u_i} Hu_i(\mathbf{x}).$$

We note that since $\ell_i(\mathbf{x})$ and $u_i(\mathbf{x})$ are linear functions of $\mathbf{x}$, this means $H\ell_i(\mathbf{x}) = \mathbf{0}$ and $Hu_i(\mathbf{x}) = \mathbf{0}$. Thus, the Hessian of the Lagrangian reduces to

$$\mathbf{L}(\mathbf{x}) = H\tau_k(\mathbf{x}),$$

and we have removed $\boldsymbol{\lambda}_{\boldsymbol{\ell}}, \boldsymbol{\lambda}_{\boldsymbol{u}}$ as the Hessian of the Lagrangian no longer depends on these values.

Since the 4^{th} condition of the second-order sufficient conditions (Theorem 6.3.3) requires computation of $\mathbf{L}(\mathbf{x})$, we need a form for the Hessian of the objective function, $H\tau_k(\mathbf{x})$.

Theorem 6.3.4. *For each state $k \in C$, the Hessian of the objective function, τ_k, is a 2×2 matrix and is given by*

$$H\tau_k = \begin{bmatrix} \dfrac{\partial^2 \tau_k}{\partial \beta^2} & \dfrac{\partial^2 \tau_k}{\partial \gamma \partial \beta} \\ \dfrac{\partial^2 \tau_k}{\partial \beta \partial \gamma} & \dfrac{\partial^2 \tau_k}{\partial \gamma^2} \end{bmatrix}$$

where

$$\begin{aligned}
\frac{\partial^2 \tau_k}{\partial \beta^2} &= -2\left\{Q_c^{-1}\frac{\partial Q_c}{\partial \beta}Q_c^{-1}\frac{\partial Q_c}{\partial \beta}Q_c^{-1}\mathbf{1}\right\}_k, \\
\frac{\partial^2 \tau_k}{\partial \gamma \partial \beta} &= \frac{\partial^2 \tau_k}{\partial \beta \partial \gamma} \\
&= -\left\{Q_c^{-1}\frac{\partial Q_c}{\partial \gamma}Q_c^{-1}\frac{\partial Q_c}{\partial \beta}Q_c^{-1}\mathbf{1}\right\}_k - \left\{Q_c^{-1}\frac{\partial Q_c}{\partial \beta}Q_c^{-1}\frac{\partial Q_c}{\partial \gamma}Q_c^{-1}\mathbf{1}\right\}_k, \\
\frac{\partial^2 \tau_k}{\partial \gamma^2} &= -2\left\{Q_c^{-1}\frac{\partial Q_c}{\partial \gamma}Q_c^{-1}\frac{\partial Q_c}{\partial \gamma}Q_c^{-1}\mathbf{1}\right\}.
\end{aligned}$$

Proof. The following proof requires the same properties of matrix differentiation as the ones used for the gradient given in (6.3.8) and (6.3.9).

Note that since Q_c is linear in β and γ, we have

$$\frac{\partial^2 Q_c}{\partial \beta^2} = \frac{\partial^2 Q_c}{\partial \gamma^2} = \frac{\partial^2 Q_c}{\partial \beta \partial \gamma} = \frac{\partial^2 Q_c}{\partial \gamma \partial \beta} = \mathbf{0}. \tag{6.3.10}$$

Consider the second-order partial derivative of τ_k with respect to β.

$$\begin{aligned}
\frac{\partial^2 \tau_k}{\partial \beta^2} &= \frac{\partial}{\partial \beta}\frac{\partial \tau_k}{\partial \beta} \\
&= \frac{\partial}{\partial \beta}\left(\left\{Q_c^{-1}\frac{\partial Q_c}{\partial \beta}Q_c^{-1}\mathbf{1}\right\}_k\right) \\
&= \left\{\frac{\partial Q_c^{-1}}{\partial \beta}\frac{\partial Q_c}{\partial \beta}Q_c^{-1}\mathbf{1}\right\}_k + \left\{Q_c^{-1}\frac{\partial^2 Q_c}{\partial \beta^2}Q_c^{-1}\mathbf{1}\right\}_k + \left\{Q_c^{-1}\frac{\partial Q_c}{\partial \beta}\frac{\partial Q_c^{-1}}{\partial \beta}\mathbf{1}\right\}_k \\
&\quad + \left\{Q_c^{-1}\frac{\partial Q_c}{\partial \beta}Q_c^{-1}\frac{\partial \mathbf{1}}{\partial \beta}\right\}_k, \qquad \text{from (6.3.8)} \\
&= \left\{-Q_c^{-1}\frac{\partial Q_c}{\partial \beta}Q_c^{-1}\frac{\partial Q_c}{\partial \beta}Q_c^{-1}\mathbf{1}\right\}_k + \left\{Q_c^{-1}\times 0\times Q_c^{-1}\mathbf{1}\right\}_k \\
&\quad + \left\{Q_c^{-1}\frac{\partial Q_c}{\partial \beta}\left(-Q_c^{-1}\right)\frac{\partial Q_c}{\partial \beta}Q_c^{-1}\mathbf{1}\right\}_k + \left\{Q_c^{-1}\frac{\partial Q_c}{\partial \beta}Q_c^{-1}\times 0\right\}_k, \\
&\qquad \text{using (6.3.9), (6.3.10) and } \frac{\partial \mathbf{1}}{\partial \beta} = \mathbf{0} \\
&= -2\left\{Q_c^{-1}\frac{\partial Q_c}{\partial \beta}Q_c^{-1}\frac{\partial Q_c}{\partial \beta}Q_c^{-1}\mathbf{1}\right\}_k.
\end{aligned}$$

The derivation of $\frac{\partial^2 \tau_k}{\partial \gamma \partial \beta}$ and $\frac{\partial^2 \tau_k}{\partial \gamma^2}$ follows similarly to the derivation of $\frac{\partial^2 \tau_k}{\partial \beta^2}$. $\square$

Recall that we are interested in checking that our first-order optimal solution satisfies the second-order sufficient conditions given in Theorem 6.3.3. We note that conditions 1, 2 and 3 of Theorem 6.3.3 are the same as the KKT conditions 1, 2 and 3 in Theorem 6.3.2. Since we know that we have a first-order optimal point, all that remains to be checked for this point to be second-order optimal is condition 4 of Theorem 6.3.3 which states that for all $\boldsymbol{y} \in T(\mathbf{x}^*, \boldsymbol{\lambda}_{\boldsymbol{\ell}}, \boldsymbol{\lambda}_{\boldsymbol{u}}), \boldsymbol{y} \neq \mathbf{0}$, we have $\boldsymbol{y}^T\mathbf{L}(\mathbf{x}^*, \boldsymbol{\lambda}_{\boldsymbol{\ell}}, \boldsymbol{\lambda}_{\boldsymbol{u}})\boldsymbol{y} > 0$.

Hence, we require the set

$$\begin{aligned}
T(\mathbf{x}^*, \boldsymbol{\lambda}_{\boldsymbol{\ell}}, \boldsymbol{\lambda}_{\boldsymbol{u}}) = \Big\{\mathbf{y} : \nabla \ell_i(\mathbf{x}^*)^T\mathbf{y} = 0 \text{ for all } i \in \tilde{L}(\mathbf{x}^*, \boldsymbol{\lambda}_{\boldsymbol{\ell}}) \\
\text{and } \nabla u_j(\mathbf{x}^*)^T\mathbf{y} = 0 \text{ for all } j \in \tilde{U}(\mathbf{x}^*, \boldsymbol{\lambda}_{\boldsymbol{u}})\Big\},
\end{aligned}$$

where $\tilde{L}(\mathbf{x}^*, \boldsymbol{\lambda}_{\boldsymbol{\ell}}) = \{i : \ell_i(\mathbf{x}^*) = 0, \lambda_{\ell_i} > 0\}$ and $\tilde{U}(\mathbf{x}^*, \boldsymbol{\lambda}_{\boldsymbol{u}}) = \{i : u_i(\mathbf{x}^*) = 0, \lambda_{u_i} > 0\}$. We note that this set depends on the number of active constraints with positive

Lagrange multipliers, λ_{ℓ_i} and λ_{u_i} for $i = 1, 2$. Thus again, 3 cases need to be considered.

1. 2 active constraints

As both β and γ are at their bounds, we have 4 possible sub-cases: $\beta = \underline{\beta}$ and $\gamma = \underline{\gamma}$, $\beta = \underline{\beta}$ and $\gamma = \overline{\gamma}$, $\beta = \overline{\beta}$ and $\gamma = \underline{\gamma}$ and $\beta = \overline{\beta}$ and $\gamma = \overline{\gamma}$.

Let us consider the first case where $\beta = \underline{\beta}$ and $\gamma = \underline{\gamma}$. Then, we have the following sub-cases.

a. If $\lambda_{\ell_1} > 0$ and $\lambda_{\ell_2} > 0$,

$$\tilde{L}(\mathbf{x}^*, \boldsymbol{\lambda_\ell}) = \{1, 2\} \qquad \text{and} \qquad T(\mathbf{x}^*, \boldsymbol{\lambda_\ell}, \boldsymbol{\lambda_u}) = \{\mathbf{0}\}.$$

This means there are no feasible directions to move in to result in a smaller objective function. Hence, we are at the minimum.

b. If $\lambda_{\ell_1} > 0$ and $\lambda_{\ell_2} = 0$,

$$\tilde{L}(\mathbf{x}^*, \boldsymbol{\lambda_\ell}) = \{1\} \qquad \text{and} \qquad T(\mathbf{x}^*, \boldsymbol{\lambda_\ell}, \boldsymbol{\lambda_u}) = \{[0, y_2]^T : y_2 \in \mathbb{R}\}.$$

Here, we need to check $\boldsymbol{y}^T\mathbf{L}(\mathbf{x}^*, \boldsymbol{\lambda_\ell}, \boldsymbol{\lambda_u})\boldsymbol{y} > 0$ which corresponds to checking that the $(2, 2)$ element of $H\tau_k(\mathbf{x}^*)$ is positive.

c. If $\lambda_{\ell_1} = 0$ and $\lambda_{\ell_2} > 0$,

$$\tilde{L}(\mathbf{x}^*, \boldsymbol{\lambda_\ell}) = \{2\} \qquad \text{and} \qquad T(\mathbf{x}^*, \boldsymbol{\lambda_\ell}, \boldsymbol{\lambda_u}) = \{[y_1, 0]^T : y_1 \in \mathbb{R}\}.$$

Here, we need to check $\boldsymbol{y}^T\mathbf{L}(\mathbf{x}^*, \boldsymbol{\lambda_\ell}, \boldsymbol{\lambda_u})\boldsymbol{y} > 0$ which corresponds to checking that the $(1, 1)$ element of $H\tau_k(\mathbf{x}^*)$ is positive.

d. If $\lambda_{\ell_1} = 0$ and $\lambda_{\ell_2} = 0$, $\tilde{L}(\mathbf{x}^*, \boldsymbol{\lambda_\ell}) = \{\}$. This means we simply need to check that $\mathbf{L}(\mathbf{x}^*, \boldsymbol{\lambda_\ell}, \boldsymbol{\lambda_u}) = H\tau_k(\mathbf{x}^*)$ is positive definite.

Note that for the other cases of β and γ at their bounds, similar logic follows.

2. 1 active constraint

Here, either β or γ is at one of their bounds. This means that at most one of λ_{ℓ_i} or λ_{u_i} is positive with the others zero, or all Lagrange multipliers could be zero.

For the case of at most one of λ_{ℓ_i} or λ_{u_i} is positive with the others zero, this corresponds to either case (b) or (c) from Case 1. Hence, we just need to check that the appropriate element of $H\tau_k(\mathbf{x}^*)$ is positive.

On the other hand, for the case where all Lagrange multipliers are zero, this corresponds to Case 1 (d). Thus, we check that $H\tau_k(\mathbf{x}^*)$ is positive definite.

3. No active constraints

For this case, all Lagrange multipliers must be zero to satisfy condition 2 of Theorems 6.3.2 and 6.3.3. Hence, this corresponds to Case 1 (d) and we check that $H\tau_k(\mathbf{x}^*)$ is positive definite.

If the above conditions, where we consider either the appropriate element of $H\tau_k(\mathbf{x}^*)$ is positive or $H\tau_k(\mathbf{x}^*)$ is positive definite, are satisfied, then we know that the first-order optimal point, $\mathbf{x}^*$, obeys the second-order sufficient conditions (Theorem 6.3.3). Hence, the point $\mathbf{x}^*$ is a strict local minimiser of the objective function τ_k.

Recall from Chapter 4 that we checked if the optimal solution satisfied the condition of being at a vertex of the feasible region (Theorem 3.5.2). Since Theorem 3.5.2 does not hold for the SIR model, there is no need to include this step here. Using the same method as detailed in Chapter 4, we post-process the solutions before returning the interval mean epidemic duration to the user.

6.4 Sensitivity analysis

The idea behind sensitivity analysis is to determine how a performance measure changes as we vary the inputs, or parameters, of the model. The standard method for performing sensitivity analysis is to consider values around a chosen parameter estimate and evaluate the performance measure at each of these values. This then gives us an idea of how variable the performance measure is to the parameter estimates. Since we are dealing with intervals, it is possible to specify an interval of parameter estimates, then use interval calculations to obtain bounds on the performance measure. However, this only gives us a snapshot of information on the performance measure. Instead, we would like to be able to elicit more detailed information about the performance measure from the parameters.

To use a model for an application, we must calibrate the model to the system via available data. This results in a distribution on the parameters which allows for greater flexibility as any type of distribution could be considered. For example, if nothing is known about the true value of the parameters aside from bounds on the parameter, then a uniform distribution on the parameters could be used which recovers something akin to a traditional sensitivity analysis. Thus, if we considered a distribution on the parameter estimates, then the corresponding distribution on the performance measures is of great interest. This distribution would give us a lot more information about the performance measure as we would know where the largest variability of the performance measure lies as opposed to bounds which simply tell us the minimum and maximum of the performance measure on some interval of the parameter estimates. Hence, given some distribution on the parameter estimates, we are interested in obtaining the cumulative distribution function (CDF) of the performance measure. In the following, we present a method which allows us to use the intervals obtained for a given performance measure to create an approximate CDF for the performance measure.

Let X be a continuous random variable with CDF $F_X(x) = P(X \leq x), x \in \mathbb{R}$. Consider an interval of X values, denoted by $\left[\underline{W}, \overline{W}\right]$, such that $(\alpha \times 100)\,\%$ of the distribution lies within this interval. That is,

$$F_X\left(\overline{W}\right) - F_X\left(\underline{W}\right) = P\left(\underline{W} \leq X \leq \overline{W}\right) = \alpha.$$

Now, consider the transformation $Y = g(X)$. Applying this transformation to the interval $\left[\underline{W}, \overline{W}\right]$ gives the interval $\left[\underline{Y}, \overline{Y}\right] = g\left(\left[\underline{W}, \overline{W}\right]\right)$, which is the image set (Section 2.2) where $\underline{Y}$ and $\overline{Y}$ are the smallest and largest values of Y, respectively, on the interval $\left[\underline{W}, \overline{W}\right]$. Hence, we have that

$$\begin{aligned} P\left(\underline{Y} \leq Y \leq \overline{Y} \cap \underline{W} \leq X \leq \overline{W}\right) &= P\left(\underline{W} \leq X \leq \overline{W}\right) \\ &= \alpha. \end{aligned}$$

Furthermore, we note that this extends to

$$\begin{aligned} F_Y\left(\overline{Y}\right) - F_Y\left(\underline{Y}\right) &= P\left(\underline{Y} \leq Y \leq \overline{Y}\right) \\ &\geq P\left(\underline{W} \leq X \leq \overline{W}\right) \\ &= \alpha, \end{aligned} \tag{6.4.11}$$

where $F_Y(y) = P(Y \leq y), y \in \mathbb{R}$ is the CDF of Y.

Hence the probability that the performance measure lies in the interval calculation must be at least the probability that the parameter lies in its chosen interval. We can make use of this information in our method to obtain an approximate CDF of Y using the intervals calculated.

Method to obtain an approximate CDF using intervals

Let X be a continuous random variable with CDF, $F_X(x)$, and let Y be the transformed continuous random variable, $Y = g(X)$, with CDF, $F_Y(y)$, where g is some performance measure (or function) of interest. Consider m non-overlapping

$(\alpha_j \times 100)\%$ intervals on the distribution of X, denoted by $\left[\underline{X}_j, \overline{X}_j\right]$ for $j = 1, \ldots, m$[1]. For example, if we consider a uniform distribution for X on $[0, 1]$, then for $\alpha_1 = \alpha_2 = 0.5$, the non-overlapping intervals we consider are $[0, 0.5]$ and $[0.5, 1]$. From these non-overlapping intervals on X, we obtain corresponding intervals on Y which we denote by $I_j = \left[\underline{Y}_j, \overline{Y}_j\right] = g\left(\left[\underline{X}_j, \overline{X}_j\right]\right)$, for $j = 1, \ldots, m$.

Since we know $F_X\left(\overline{X}_j\right) - F_X\left(\underline{X}_j\right) = \alpha_j$, this means that $F_Y\left(\overline{Y}_j\right) - F_Y\left(\underline{Y}_j\right) \geq \alpha_j$ by (6.4.11). Hence, each interval I_j contributes at least the probability α_j to the CDF. Thus, each interval I_j has density at least $d_j = \dfrac{\alpha_j}{\overline{Y}_j - \underline{Y}_j}$.

Then, we define P to be the ordered set of unique endpoints of the intervals,

$$\begin{aligned} P &= \bigcup_j \{\underline{Y}_j, \overline{Y}_j\} \\ &= \{b_i : i = 1, \ldots, n\}, \end{aligned}$$

where n is the total number of points in the set.

For example, if $\left[\underline{Y}_1, \overline{Y}_1\right] = [0, 0.04]$ and $\left[\underline{Y}_2, \overline{Y}_2\right] = [0, 0.06]$, then

$$\begin{aligned} P &= \left\{\underline{Y}_1, \overline{Y}_1\right\} \cup \left\{\underline{Y}_2, \overline{Y}_2\right\} \\ &= \{0, 0.04, 0.06\}. \end{aligned}$$

Now, define b_i^+ as a value slightly larger than b_i but less than b_{i+1}. That is,

$$b_i^+ \in (b_i, b_{i+1}), \qquad \text{for } i = 1, \ldots, n-1.$$

Note that we do not consider b_n^+ as it lies outside the largest endpoint, b_n, and hence outside the range of our data. Then, for all i, we calculate the approximate probability density,

$$d\left(b_i^+\right) = \sum_{j : b_i^+ \in I_j} d_j.$$

[1]Since all such intervals are closed, the term non-overlapping intervals should be interpreted as meaning that the intersection of any pair of intervals has Lebesgue measure zero.

Recall that we are interested in obtaining an approximate CDF of Y. Since we are considering continuous random variables, an approximate CDF of Y is found by integrating an approximate probability density function (pdf) of Y. From our calculations, we can obtain an approximate pdf of Y by assuming that the approximate probability density between consecutive endpoints is constant. Hence, we integrate this approximate pdf of Y to obtain an approximate CDF of Y. The cumulative probabilities $F_Y(b_i)$ for this approximate CDF can be calculated as follows,

$$F_Y(b_i) = P(Y \leq b_i) = \sum_{k<i} d\left(b_k^+\right) \times (b_{k+1} - b_k), \qquad \text{for } i = 1, \dots, n.$$

By assuming a constant density between consecutive endpoints, we have assumed that the increase in the approximate CDF is linear between consecutive endpoints, b_i. Therefore, we can plot the cumulative probabilities $F_Y(b_i)$ at each b_i, for $i = 1, \dots, n$, before joining these points to get the approximate CDF of Y.

In the following, we present an approximate CDF of Y for the mean final epidemic size and mean epidemic duration using our method of intervals and compare our approximate CDF to that achieved by simulation.

SIR epidemic model

Consider a population of $N = 200$ individuals with a constant rate of recovery of an infectious individual, $\gamma = 1$, and consider a uniform distribution on the effective transmission rate parameter, $\beta \sim \text{Uniform}[1, 4]$. We obtained 100000 samples of β and calculated the mean final epidemic size using equation (6.2.3) and the mean epidemic duration using equation (6.3.6), for each sample. We provided these to MATLAB's `ksdensity` function to obtain kernel density estimates of the CDFs for the mean final epidemic size and the mean epidemic duration.

We consider non-overlapping 5% intervals on β and use our method to obtain approximate CDFs for the mean final epidemic size (Figure 6.4.1) and the mean epidemic duration (Figure 6.4.2) for the above SIR model. On the same figures, we also plot the CDFs obtained using MATLAB's `ksdensity` function to compare with our approximate CDFs.

From the CDFs of the mean final epidemic size and the mean epidemic duration (Figures 6.4.1 and 6.4.2), we observe that the approximate CDFs obtained using our method very closely resemble the CDFs obtained using `ksdensity`.

The other comparison performed was based on computation times. Using 100000 simulations to obtain a CDF using `ksdensity` takes a lot more computational time than our method. Hence, we are interested in comparing the approximate CDF obtained using simulations which take a similar amount of computational time as our method. For the mean final epidemic size, we found that it took a similar amount of time to perform 39 simulations and obtain an approximate CDF as it did for our method to obtain an approximate CDF using non-overlapping 5% intervals. Similarly for the mean epidemic size, performing 320 simulations took a similar amount of time as our method to obtain an approximate CDF.

In Figures 6.4.1 and 6.4.2, we have also included the plot of the mean final epidemic size CDF using 39 simulations and the plot of the mean epidemic duration using 320 simulations. From these plots, we see that over a similar computational time, our approximate CDF plots very closely resemble the CDF from the extensive simulation (100000 samples). On the other hand, the CDFs formed using much smaller samples are less accurate than our method. Hence, giving us evidence that our method is potentially quite useful especially in regards to computational time and accuracy of the approximate CDF.

In Sections 6.1, 6.2 and 6.3, we have investigated the use of intervals in the SIR model, proved analytic results for an interval mean final epidemic size and developed a numerical method to obtain an interval mean epidemic duration for a given interval SIR model. Lastly, in the final section, we investigated the idea of

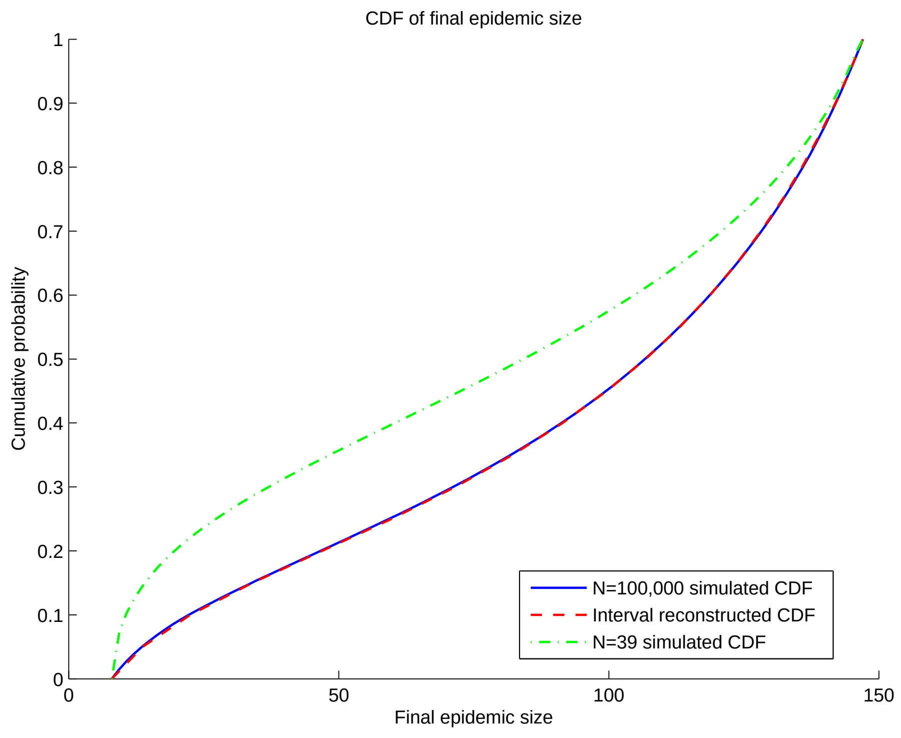

Figure 6.4.1: Comparison of CDFs of mean final epidemic size for N $= 200$, $\gamma = 1$ and $\beta \sim$ Uniform$[1, 4]$.

using intervals as an alternative to traditional sensitivity analysis and compared our method of obtaining an approximate CDF using intervals with the CDF obtained using kernel density estimates from simulations. In what remains (the final chapter), we conclude and summarise our findings as well as consider possible extensions and future work.

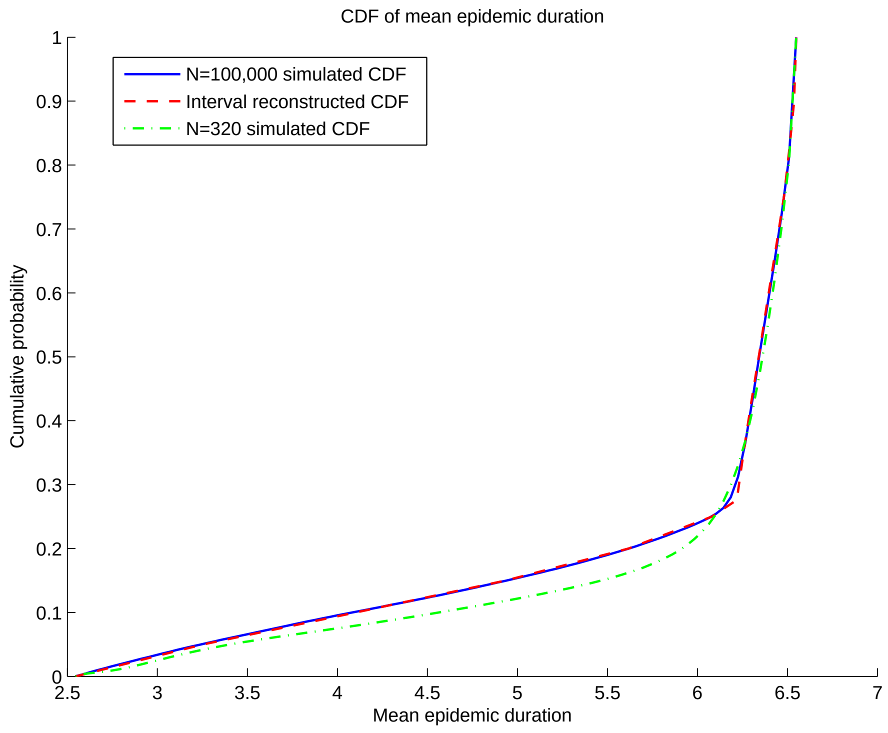

Figure 6.4.2: Comparison of CDFs of mean epidemic duration for N $= 200$, $\gamma = 1$ and $\beta \sim$ Uniform$[1, 4]$.

Chapter 7

Conclusions

7.1 Summary

In this thesis, we explored a method for accounting for the uncertainty in parameter estimates through the use of intervals and applied this to Markov chains. One of the main objectives was to develop methods to obtain intervals on various performance measures, such as the expected total cost and mean hitting times, for interval Markov chains. We first considered discrete-time interval Markov chains and the problem of obtaining intervals for the expected total costs for these interval Markov chains. Since solving these interval systems of equations directly did not satisfy the required constraints on the problem, we needed to develop a different method to solve for the interval expected total costs and we chose to do so using optimisation techniques. Since analytic solutions are desired, we first investigated methods to obtain analytic solutions. Throughout this investigation for an analytic solution, we obtained various theoretical properties of the problem which simplified our problem greatly. Furthermore, it was through this investigation that we found a connection with Markov decision processes and proved the form of the optimal solution for our problem.

However, as we were unable to obtain an analytic solution to our problem, we developed a numerical method to obtain the interval expected total cost vector for a discrete-time interval Markov chain. In the numerical method, we applied the theoretical properties found to simplify our problem as well as to purify the approximate solutions obtained from MATLAB's `fmincon` function. Aside from discrete-time interval Markov chains, we also considered continuous-time interval Markov chains where we used the concept of uniformisation to convert a continuous-time interval Markov chain to a discrete-time interval Markov chain without losing any information. This meant we were able to transform any uniformisable continuous-time interval Markov chain to its associated discrete-time interval Markov chain and use the techniques we developed previously to solve for the interval expected total cost for the associated discrete-time interval Markov chain. Then, all that was needed was a transformation back to obtain the interval expected costs for the continuous-time interval Markov chain.

Since birth and death processes are commonly used in many applications, we also considered incorporating intervals into these structured Markov chains. As before, we were interested in obtaining the interval expected total costs for these processes and were able to prove analytic results for these. Two birth and death processes were considered. The first being a discrete-time interval birth and death process where the rows of the interval transition probability matrix are independent. The second discrete-time interval birth and death process considered was the discrete-time interval constant-parameter birth and death process. For this process, there were only 3 parameters in the transition probability matrix and the rows of the interval transition probability matrix were not independent. Hence, the previous theory and numerical methods developed could not be immediately applied to this process and new theory was developed.

We continued with the trend of constant parameter models, by considering an SIR epidemic model. For this model, we investigated analytic and numerical results for the performance measures of interest, the interval mean final epidemic size and the interval mean epidemic duration. We were able to prove analytic results for the interval final epidemic size but for the interval mean epidemic duration a numerical method was developed.

Finally to conclude our investigation, we considered the other main objective of this thesis which was to investigate if interval analysis can be used in place of sensitivity analysis. Hence, instead of considering traditional sensitivity analysis, we considered a distribution around our parameters and managed to obtain an approximate CDF using intervals on performance measures. The approximate CDF we obtained closely resembled the CDF obtained through extensive simulations. Finally, to show the worth of our method of obtaining approximate CDFs, we compared our approximate CDF with the approximate CDF obtained using simulations which took a similar amount of computational time as ours. We observed through the above comparison that our method performed better in terms of accuracy as our approximate CDF was closer to the CDF obtained using extensive simulations.

7.2 Future work

A potentially interesting extension to this thesis is to explore the use of sequential linear programming to solve our optimisation problems. In Chapter 4, we observed that the sequential linear programming method outperformed our numerical method and the method by Blanc and den Hertog [2] very significantly when comparing computational times and also returned comparable results. We noted in Section 4.7 that solving non-linear optimisation problems using sequential quadratic programming is well established but linearising the non-linear objective function and solving it sequentially is not well established at least in part because sequential linear programming can only return vertex solutions. Thus is appropriate in our problem, hence, with additional theoretical investigation and justification to ensure the robustness of this sequential linear programming method, this method would be preferred due to its faster computational times. A possible immediate avenue to explore is to consider the non-negativeness of the gradient for our non-linear optimisation problem. We know that the gradient for our problem is non-negative at all points in the feasible region and this may explain the observed successes of the sequential linear programming method. Another possible avenue is to re-visit the work by Blanc and den Hertog [2]. Since they were able to convert the original non-linear optimisation problem into a linear programming problem, this may explain why taking a linearisation of the non-linear objective function and solving it sequentially seems to work.

Another potential area for investigation is to extend our method for obtaining an approximate CDF to consider distributions on more parameters. Recall that our method as specified in Section 6.4 is one-dimensional. For the SIR epidemic model, the parameter γ was fixed and we only considered a uniform distribution on the parameter β. Since our method for obtaining an approximate CDF for the performance measures performs well when compared to the CDF obtained through extensive simulations, it is of interest to extend our method further to handle more than one-dimension. For the case of two parameters, we would be considering non-overlapping grids instead of non-overlapping intervals, then obtaining intervals on the performance measure corresponding to these grids. However, we note that this process might potentially be quite time consuming and so computationally efficient methods will need to be developed to obtain an approximate CDF when we consider distributions on several parameters.

Bibliography

[1] A. Berman and R.J. Plemmons. *Nonnegative matrices in the mathematical sciences.* Computer science and applied mathematics. Academic Press, 1979.

[2] H. Blanc and D. den Hertog. On Markov chains with uncertain data. *CentER Discussion Paper. Available at SSRN 1138144*, 2008.

[3] M.A. Campos, G.P. Dimuro, A.C. da Rocha Costa, and V. Kreinovich. Computing 2-Step Predictions for Interval-Valued Finite Stationary Markov Chains. Technical report, University of Texas at El Paso, 2004.

[4] E.K.P. Chong and S.H. Zak. *An Introduction to Optimization.* Wiley Series in Discrete Mathematics and Optimization. Wiley, 2008.

[5] R.J. Crossman, P. Coolen-Schrijner, and F.P.A. Coolen. Time-homogeneous Birth-death Processes with Probability Intervals and Absorbing State. *Journal of Statistical Theory and Practice*, 3(1):103–118, 2009.

[6] R.J. Crossman, P. Coolen-Schrijner, D. Škulj, and F.P.A. Coolen. Imprecise Markov Chains with an Absorbing State. In *Proceedings of the Sixth International Symposium on Imprecise Probabilities: Theory and Applications, Durham, UK*, pages 119–128, 2009.

[7] P.E. Gill, W. Murray, and M.H. Wright. *Practical optimization.* Academic Press, 1981.

[8] D.J. Hartfiel. *Markov set-chains.* Lecture Notes in Mathematics. Springer, 1998.

[9] D.J. Hartfiel and E. Seneta. On the theory of Markov set-chains. *Advances in Applied Probability*, 26(4):947–964, 1994.

[10] M.J. Keeling and P. Rohani. *Modeling Infectious Diseases in Humans and Animals.* Princeton University Press, 2008.

[11] M.J. Keeling and J.V. Ross. On methods for studying stochastic disease dynamics. *Journal of The Royal Society Interface*, 5(19):171–181, 2008.

[12] I.O. Kozine and L.V. Utkin. Interval-Valued Finite Markov Chains. *Reliable Computing*, 8:97–113, 2002.

[13] MATLAB. *MATLAB and Optimization Toolbox.* The MathWorks Inc., Natick, Massachusetts, Version 8.0 (R2012b).

[14] R.E. Moore, R.B. Kearfott, and M.J. Cloud. *Introduction to Interval Analysis.* Society for Industrial and Applied Mathematics, 2009.

[15] J. Nocedal and S.J. Wright. *Numerical optimization.* Springer Series in Operations Research and Financial Engineering. Springer, 2006.

[16] J.R. Norris. *Markov Chains.* Cambridge Series in Statistical and Probabilistic Mathematics. Cambridge University Press, 2007.

[17] K.B. Petersen and M.S. Pedersen. The Matrix Cookbook. November 2008.

[18] M.L. Puterman. *Markov Decision Processes: Discrete Stochastic Dynamic Programming.* A Wiley-Interscience publication. Wiley, 1994.

[19] M.G. Roberts. Epidemic models with uncertainty in the reproduction number. *Journal of Mathematical Biology*, 66(7):1463–1474, 2013.

[20] S.M. Ross. *Introduction to Probability Models.* Probability and Statistics. Academic Press, 2007.

[21] S.M. Rump. INTLAB - INTerval LABoratory. In Tibor Csendes, editor, *Developments in Reliable Computing*, pages 77–104. Kluwer Academic Publishers, Dordrecht, 1999. `http://www.ti3.tuhh.de/rump/`.

[22] D. Škulj. Finite Discrete Time Markov Chains with Interval Probabilities. In *Soft Methods for Integrated Uncertainty Modelling*, volume 37 of *Advances in Soft Computing*, pages 299–306. Springer Berlin/Heidelberg, 2006.

[23] D. Škulj. Discrete Time Markov Chains with Interval Probabilities. *International Journal of Approximate Reasoning*, 50(8):1314–1329, 2009.