UNIVERSITY OF COPENHAGEN
FACULTY OR DEPARTMENT

Master Thesis in Actuarial Mathematics

Malte Lindholm Stoltze Rasmussen

Bootstrap and statistical inference on phase-type distributions

Supervisor: Mogens Bladt

Submitted on: 9 June 2017

Contents

Abstract

This thesis deals with statistical inference on phase-type distributions. As one of the main contributions, an approach for obtaining well-defined sample distributions for the estimated parameter values is provided and implemented. This approach involves a combination of nonparametric bootstrap with the EM algorithm and provides an alternative for deriving (empirical) confidence intervals for phase-type distributions, which may be used in statistical inference.

In addition, this work focuses on the denseness of continuous phase-type distributions in the positive real axis. A numerical study of the denseness property is carried out, and a new way of approximating non-negative continuous distributions with phase-type distributions is provided.

Keywords: phase-type distributions; EM algorithm; Bootstrap

Preface

This master thesis was submitted at the University of Copenhagen in partial fulfilling the requirements for acquiring a Master degree in Actuarial Mathematics.

The project has been supervised by Mogens Bladt, guest professor at University of Copenhagen (Department of Mathematical Sciences).

Malte Lindholm Stoltze Rasmussen
June, 2017

Notation

$1_{\{x\in A\}}$,$1_A(x)$,$1\{x \in A\}$	:	indicator function for A.
$\delta_x(y)$	:	Kronecker delta, function which is one if $x = y$ and zero otherwise.
$\delta_{i,j}, \delta_{ij}$	:	number which is one if $i = j$, zero otherwise.
$X, Y, ...$	:	random variables
$X_{i,n}$	:	the i'th order statistic out of n.
$\boldsymbol{\alpha}, \boldsymbol{\pi}$,...	:	bold lower case greek letters denote *row* vectors
α_i, π_i	:	elements of vectors $\boldsymbol{\alpha}, \boldsymbol{\pi}, ...$
$\boldsymbol{s}$,$\boldsymbol{t}$,...	:	bold lower case roman letters denote *column* vectors
$s_i, t_i, ...$	:	elements of $\boldsymbol{s}, \boldsymbol{t}, ...$
$\boldsymbol{e}_i$	:	the i'th unit vector of the standard basis for $\mathbb{R}^n$.
$\boldsymbol{e}$	:	a vector with all entries equal to one.
$\boldsymbol{a}'$	:	transposed of the vector $\boldsymbol{a}$.
$\Gamma, \boldsymbol{S}, ...$	:	bold upper case letters of *any* kind denote matrices
γ_{ij}, s_{ij},...	:	elements of matrices $\Gamma, \boldsymbol{S}, ...$
$\mathbb{N}$	:	the natural numbers $\{0, 1, 2, ...\}$
$\mathbb{Z}$	:	the set of integers
$\mathbb{R}, \mathbb{R}^n$	:	Real numbers in one and n dimensions.
$\mathbb{P}$	:	Probability (measure)
$\mathbb{P}_i$	:	$\mathbb{P}(\cdot\|X_0 = i)$
$\mathbb{E}$	:	Expectation
$\mathbb{E}_i$	:	$\mathbb{E}(\cdot\|X_0 = i)$
a.s.	:	almost surely
a.e.	:	almost everywhere
$\xrightarrow{\mathbb{P}}$	:	convergence in probability
$\xrightarrow{d}$	:	convergence in distribution, weak convergence
$\sim, \stackrel{d}{=}$	:	distributed as, equality in law.
$\min(a, b)$	:	minimum of a and b
$\max(a, b)$	:	maximum of a and b
$[x]$	:	integer part of $x \in \mathbb{R}$, i.e. largest integer which is smaller or equal to x.
δ_{ij}	:	$1\{i = j\}$, indicator for i equals j.
$\boldsymbol{I}$	:	identity matrix
$\mathbf{0}$	:	zero matrix
i.i.d.	:	independent identically distributed

L_X	:	Laplace transform of the random variable X, i.e. $L_X(\theta) = \mathbb{E}(e^{-\theta X})$.
M_X	:	moment generating function of the random variable X, i.e. $M_X(\theta) = \mathbb{E}(e^{\theta X})$.
$\mathrm{PH}(\boldsymbol{\alpha}, \boldsymbol{S})$, $\mathrm{PH}_p(\boldsymbol{\alpha}, \boldsymbol{S})$	:	p–dimensional phase–type distribution (representation) with initial distribution $\boldsymbol{\alpha}$ and sub–intensity matrix $\boldsymbol{S}$.
$X \sim \exp(\lambda)$	:	X exponentially distributed with intensity λ (mean $1/\lambda$).
$X \sim \mathrm{Pois}(\lambda)$	:	X Poisson distributed with intensity λ (mean $1/\lambda$).
$\mathrm{Er}_n(\lambda)$	:	n'th order Erlang distribution with intensity λ, i.e. the distribution of $X_1 + ... + X_n$ where $X_1, ..., X_n$ i.i.d. $\sim \exp(\lambda)$.
$:=$	:	defined to be, assigned.
$\propto$	:	proportional to
$y-$	:	$\lim_{x \uparrow y} x$

Chapter 1

Introduction

Phase-type distributions model the time until absorption of a Markov chain with one absorbing state and finitely many transient states. They can be defined in terms of a Markov chain in discrete time or continuous time, depending on whether a discrete or continuous phase-type distribution is considered. Phase-type distributions can be represented by a sub-intensity matrix $\boldsymbol{T}$ and an initial probability vector $\boldsymbol{\pi}$, where $(\boldsymbol{\pi}, \boldsymbol{T})$ singularly comprises all information about the distribution.

The fact that we often can express functionals in closed-form makes them mathematically tractable. Any distribution with support on $\mathbb{N} \cup \{0\}$ or $\mathbb{R}^+$ can be approximated arbitrarily closely by a phase-type distribution, so at the same time we can obtain a high level of generality. Due to the versatility of phase-type distributions, they have shown to be advantageous to work with in different research fields.

The expectation maximization algorithm (EM algorithm) is an iterative method for finding parameters in statistical models where the model depends on some unknown latent variable. This is done by finding the maximum likelihood estimate. The algorithm alternates between calculating the log likelihood based on the given estimates of the parameters, and then finding new parameters by maximizing the expected log likelihood function. These two steps are referred to as the expectation step and maximization step. The EM algorithm can be used to fit phase-type distributions to both uncensored and interval censored sample data, but also theoretical distributions. The EM algorithm is a mathematically convenient tool for estimating phase-type distributions, partly because no evaluation of the derivatives is required. Furthermore, since phase-type distributions are a subclass of exponential families, the expectation step simplifies to calculating conditional expectations of the sufficient statistics, instead of the whole log likelihood function. These sufficient statistics can furthermore be expressed explicitly in terms of matrix exponentials.

Regarding statistical inference on phase-type distributions, there are issues with respect to identification and over-parameterization. Confidence regions will in general not be well-defined due to a general non-unique phase-type representation.

This project accommodates a numerical study of this denseness property. By using this denseness property, a new approximation formula will be proposed and the applicability of this approximation will be examined.

The main contribution of this master thesis is an approach, to achieve well defined confidence regions on non-unique phase-type distributions, by combining nonparametric bootstrap with the EM algorithm. This approach will be implemented by extension and modification of an already existing program, called the EMpht program. This program was developed for fitting phase-type distributions to incomplete or complete data by the use of the EM algorithm. Furthermore, new features will be added to the original EMpht program which will build the foundation for a new type of numerical study.

The structure of this thesis is the following. Since Markov chains are essential for understanding phase-type distributions, the theory of Markov chains is introduced in chapter 2. In chapter 3 phase-type distributions are introduced including their relation to Markov chains. Chapter 4 introduces the EM algorithm, both in general context and in relation to phase-type distributions. Chapter 5 contains a proof for denseness of continuous phase-type distributions in $\mathbb{R}^+$ together with a numerical study of this denseness property. Chapter 6 contains an introduction to nonparametric bootstrap and an implementation of this aforementioned approach for achieving well defined confidence regions.

Chapter 2

Markov chains

Stochastic models are used for modeling random processes, evolving over time or space. A stochastic process, with state-space E, is a family of random variables $\{X_t : t \in \mathcal{I}\}$ on E where $\mathcal{I}$ is either the set of natural numbers $\mathbb{N}$ or $[0, \infty)$. Markov chains are subclasses of stochastic processes which are characterized by the the so-called Markov dependency. The behavior of the future of the process, given its past, only depends on information about the current state. A keystone for understanding and applying Markov chains is to understand this memoryless property, also called forgetfulness property. Section 2.1 introduces this memoryless property. The 2 subsequent sections introduce Markov chains in discrete and continuous time, respectively.

2.1 Memoryless property

A continuous random variable has the memoryless property if, for $\forall x, y \geq 0$, it holds that

$$\mathbb{P}(T > x + y | T > x) = \mathbb{P}(T > y). \tag{2.1}$$

Likewise, a discrete random variable has the memoryless property if 2.1 holds for all nonnegative integers x and y. Let T be an exponential distributed random variable with rate parameter λ, such that $\mathbb{P}(T > x) = e^{-\lambda x}$. The calculations below shows that the memoryless property holds for exponential distributions:

$$\begin{aligned}\mathbb{P}(T > x+y | T > x) &= \frac{\mathbb{P}(T > x+y, T > x)}{\mathbb{P}(T > x)} \\ &= \frac{\mathbb{P}(T > x+y)}{\mathbb{P}(T > x)} \\ &= \frac{e^{-\lambda(x+y)}}{e^{-\lambda x}} \\ &= e^{-\lambda y} \\ &= \mathbb{P}(T > y)\end{aligned} \tag{2.2}$$

Now assume that T is a memoryless continuous random variable. Since $\mathbb{P}(T > y)$ is monotone decreasing with y, it holds that

$$\begin{aligned}\frac{\mathbb{P}(T > x+y)}{\mathbb{P}(T > x)} &= \mathbb{P}(T > x+y | T > x) \\ &= \mathbb{P}(T > y).\end{aligned} \tag{2.3}$$

Rearranging 2.3 gives

$$\mathbb{P}(T > x+y) = \mathbb{P}(T > x)\mathbb{P}(T > y). \tag{2.4}$$

Set $x = 1$ and $y = 1$. Then

$$\begin{aligned}\mathbb{P}(T > 2) &= \mathbb{P}(T > 1)\mathbb{P}(T > 1) \\ &= (\mathbb{P}(T > 1))^2,\end{aligned} \tag{2.5}$$

and consequently

$$\mathbb{P}\left(T > \frac{1}{2}\right) = (\mathbb{P}(T > 1))^{\frac{1}{2}}. \tag{2.6}$$

Set $\lambda = -\ln(\mathbb{P}(T > 1))$ such that, for $z \in \mathbb{Q}$, it holds that

$$\begin{aligned}\mathbb{P}(T > z) &= (\mathbb{P}(T > 1))^z \\ &= e^{\ln(\mathbb{P}(T>1))z} \\ &= e^{-\lambda z}.\end{aligned} \tag{2.7}$$

Since 2.7 holds for $z \in \mathbb{Q}$ and $\mathbb{P}(T > z)$ is monotone decreasing in z, 2.7 implies that $\mathbb{P}(T > z)$ is exponential over its whole domain. Since this is the tail probability for the exponential distribution, we conclude that the exponential distribution is the only continuous distribution, for which the memoryless property holds.

Now, let T be a geometric distributed random variable with success parameter p, such that $\mathbb{P}(T > x) = (1-p)^x$. The calculations below shows that the memoryless property also holds for geometric distributions:

$$\begin{aligned}\mathbb{P}(T > x+y | T > x) &= \frac{\mathbb{P}(T > x+y)}{\mathbb{P}(T > x)} \\ &= \frac{(1-p)^{x+y}}{(1-p)^x} \\ &= (1-p)^y \\ &= \mathbb{P}(T > y)\end{aligned} \tag{2.8}$$

Let X be a discrete random variable with support on $\mathbb{N}$. Then

$$\mathbb{P}(X \geq s+t) = \mathbb{P}(X \geq s)\mathbb{P}(X > t) \tag{2.9}$$

holds for $s, t \in \mathbb{N}$. Summing over all $s \in \mathbb{N}$ in 2.9 and setting $t = k$ gives

$$\begin{aligned}\sum_{s=1}^{\infty} \mathbb{P}(X \geq s+k) &= \mathbb{P}(X > k) \sum_{s=1}^{\infty} \mathbb{P}(X \geq s) \\ &= \mathbb{P}(X > k)\left(1 + \sum_{s=2}^{\infty} \mathbb{P}(X \geq s)\right) \\ &= \mathbb{P}(X \geq k+1)\left(1 + \sum_{s=2}^{\infty} \mathbb{P}(X \geq s)\right).\end{aligned} \tag{2.10}$$

Likewise, summing over all $t \in \mathbb{N}$ and setting $s = k$ in 2.9 gives

$$\sum_{t=1}^{\infty} \mathbb{P}(X \geq t+k) = \mathbb{P}(X \geq k) \sum_{t=1}^{\infty} \mathbb{P}(X > t) = \mathbb{P}(X \geq k) \sum_{t=2}^{\infty} \mathbb{P}(X \geq t). \tag{2.11}$$

We see that left hand side of 2.10 and 2.11 are the same. Setting right hand side of 2.10 and 2.11 equal to each other, and rearranging, gives

$$\begin{aligned}\mathbb{P}(X \geq k) \sum_{t=2}^{\infty} \mathbb{P}(X \geq t) &= \mathbb{P}(X \geq k+1)\left(1 + \sum_{s=2}^{\infty} \mathbb{P}(X \geq s)\right) \\ \Leftrightarrow \sum_{t=2}^{\infty} \mathbb{P}(X \geq t) &= \frac{\mathbb{P}(X \geq k+1)}{\mathbb{P}(X \geq k) - \mathbb{P}(X \geq k+1)}.\end{aligned} \tag{2.12}$$

Calculate 2.12 for $k = 1$:

$$\begin{aligned}\sum_{t=2}^{\infty} \mathbb{P}(X \geq t) &= \frac{\mathbb{P}(X \geq 2)\,\mathbb{P}(X \geq 1)}{1 - \mathbb{P}(X \geq 2)} \\ &= \frac{\mathbb{P}(X \geq 2)}{1 - \mathbb{P}(X \geq 2)} \\ &= \frac{1 - \mathbb{P}(X = 1)}{\mathbb{P}(X = 1)}\end{aligned} \tag{2.13}$$

Insert 2.13 into 2.12 and solve for $\mathbb{P}(X \geq k+1)$:

$$\begin{aligned}\mathbb{P}(X \geq k)\,\frac{1 - \mathbb{P}(X = 1)}{\mathbb{P}(X = 1)} &= \mathbb{P}(X \geq k+1)\left(1 + \frac{1 - \mathbb{P}(X = 1)}{\mathbb{P}(X = 1)}\right) \\ &= \frac{\mathbb{P}(X \geq k+1)}{\mathbb{P}(X = 1)} \Leftrightarrow \\ \mathbb{P}(X \geq k+1) &= \mathbb{P}(X \geq k)\,(1 - \mathbb{P}(X = 1))\end{aligned} \tag{2.14}$$

By inserting 2.14 iteratively we derive an explicit expression for $\mathbb{P}(X \geq k+1)$:

$$\begin{aligned}\mathbb{P}(X \geq k+1) &= \mathbb{P}(X \geq k-1)\,(1 - \mathbb{P}(X = 1))^2 \\ &= \mathbb{P}(X \geq 1)\,(1 - \mathbb{P}(X = 1))^k \\ &= (1 - \mathbb{P}(X = 1))^k\end{aligned} \tag{2.15}$$

Setting $p = \mathbb{P}(X = 1)$ then gives

$$\mathbb{P}(X \geq k) = (1-p)^{k-1}, \tag{2.16}$$

from which we derive the point probability

$$\begin{aligned}\mathbb{P}(X = k) &= \mathbb{P}(X \geq k) - \mathbb{P}(X \geq k+1) \\ &= \mathbb{P}(X \geq k) - \mathbb{P}(X \geq k)\,(1-p) \\ &= p\mathbb{P}(X \geq k) \\ &= p(1-p)^{k-1}.\end{aligned} \tag{2.17}$$

Finally, we conclude that a memoryless discrete random variable has to be geometric distributed. Hence, the geometric distribution is the only discrete distribution where the memoryless property holds.

2.2 Discrete time Markov chains

A discrete time stochastic process $\{X_n\}_{n\geq 0}$ with finite or countable state–space E is a Markov chain if, for all $n \in \mathbb{N}$ and $i, j, i_0, i_1, \ldots, i_{n-1} \in E$, the Markov property in discrete time holds, which says

$$\mathbb{P}(X_{n+1} = j \mid X_n = i, X_{n-1} = i_{n-1}, \ldots, X_0 = i_0) = \mathbb{P}(X_{n+1} = j \mid X_n = i)\,. \tag{2.18}$$

In other words, a discrete time stochastic process is a Markov chain if it given its past only depends on the present. The discrete time Markov chain is time-homogeneous if the right hand side 2.18 does not depend on the time n. We denote the right hand side a transition probability and use the notation

$$p_{ij} = \mathbb{P}\left(X_{n+1} = j \mid X_n = i\right). \tag{2.19}$$

We arrange the transition probabilities in a transition matrix $\boldsymbol{P} = \left\{p_{ij}\right\}_{i,j\in E}$ and let the row vector $\boldsymbol{\pi} = \{\pi_i\}_{i\in E}$ denote the initial distribution of X_0. $p_{ij}^{(n)} = \mathbb{P}\left(X_n = j \mid X_0 = i\right)$ denotes the n step transition probability, given initiation in state i. We let $\boldsymbol{P}^n = \left\{p_{ij}^{(n)}\right\}_{i,j\in E}$ denote the corresponding n-step probability matrix. The whole distribution of the process is therefore completely determined by $\boldsymbol{\pi}$ and $\boldsymbol{T}$. It follows directly from the matrix multiplication $\boldsymbol{P}^{m+n} = \boldsymbol{P}^m\boldsymbol{P}^n$ that the n-step transition probabilities satisfy

$$p_{ij}^{(n+m)} = \sum_{k\in E} p_{ik}^{(n)} p_{kj}^{(m)}. \tag{2.20}$$

This result is the known as the Chapman-Kolmogorov Theorem.

A stopping time $\tau \in \mathbb{N}\cup\{+\infty\}$ for the Markov chain $\{X_n\}_{n\in\mathbb{N}}$ is a random variable with the property $\{\tau = n\} \in \mathcal{F}_n = \sigma(X_0, X_1, \ldots, X_n)$ for all n. Hence, $\mathcal{F}_n$ is the natural filtration. Define

$$\mathcal{F}_\tau = \{A \in \mathcal{F} \mid A \cap \{\tau = n\} \in \mathcal{F}_n, n \in \mathbb{N}\cup\{+\infty\}\}. \tag{2.21}$$

We use the definition of stopping time for the strong Markov property theorem which makes sure that the Markov property also holds for certain random times. The strong Markov property says that, if τ is a stopping time for the Markov chain $\{X_n\}_{n\in\mathbb{N}}$, then

$$\mathbb{P}\left(X_{\tau+1} = j \mid X_\tau = i, X_{\tau-1} = i_{n-1}, \ldots, X_0 = i_0\right) = \mathbb{P}\left(X_1 = j \mid X_0 = X_\tau\right). \tag{2.22}$$

We say that state i leads to state j, and write $i \to j$, if

$$\exists n \geq 1 : \; p_{ij}^{(n)} > 0. \tag{2.23}$$

State i and state j communicates if $i \to j$ and $j \to i$, and we write $i \sim j$, where the relation $\sim$ defines an equivalence relation on the state space. We can partition the state-space into disjoint equivalence classes where every state, within the corresponding equivalence class, communicates. State i is recurrent if

$$\mathbb{P}\left(T_i < \infty \mid X_0 = i\right) = 1, \tag{2.24}$$

where

$$T_i = \inf\{n \geq 1 \mid X_n = i\} \tag{2.25}$$

is the recurrence time. Conversely, state i transient if

$$\mathbb{P}\left(T_i < \infty \mid X_0 = i\right) < 1. \tag{2.26}$$

Recurrence and transience are class properties, meaning, that if an element of an equivalence class is recurrent (transient), then all the states of the equivalence class are recurrent (transient). If all of its states communicate, the Markov chain consists of one class and we say that the Markov chain is irreducible.

A row vector $\boldsymbol{v}$ is a stationary measure of a Markov chain $\{X_n\}_{n\in\mathbb{N}}$, with transition matrix $\boldsymbol{P}$, if

$$\boldsymbol{v} = \boldsymbol{v}\boldsymbol{P}. \tag{2.27}$$

Consequently, if v is a stationary probability measure of the Markov chain $\{X_n\}_{n\in\mathbb{N}}$, and $X_n \sim v$, then it also holds that $X_{n+1} \sim v$. Actually if a state i is recurrent, the expected number of visits $v_j = \mathbb{E}_j(\sum_{n=0}^{T_i-1} 1\{X_n = j\})$ to state j, between two consecutive visits to state i, defines a stationary measure $v = (v_j)_{j\in E}$ for the Markov chain $\{X_n\}_{n\in\mathbb{N}}$. It can be shown that there exists a stationary measure if a Markov chain is irreducible and recurrent and, furthermore, if all stationary measures are proportional. Hence, if the sum of the entries of the stationary measure is finite, then we can derive a stationary distribution by normalizing the stationary vector $\pi = \frac{v}{\sum_k v_k}$.

Let $L_{d(i)} = \{d(i),\ 2d(i),\ 3d(i), 4d(i),\ \cdots\}$ and let $T_i = \inf\{n \geq 1 \mid X_n = i\}$ be the aforementioned recurrence time. We define periodicity of a discrete time Markov chain as the largest integer $d(i)$ such that

$$\mathbb{P}_i(T_i \in L_{d(i)}) = 1. \tag{2.28}$$

We say that the Markov chain is aperiodic if the period is one.

We say that a Markov chain $\{X_n\}_{n\in\mathbb{N}}$ is ergodic if it is irreducible, aperiodic, and positive recurrent $\mathbb{E}(T_i) < \infty$. Consequently, the n-step probabilies $p_{ij}^{(n)}$ will converge to the corresponding probabilities of the stationary distribution $\pi = \{\pi_i\}_{i\in E}$, i.e.,

$$\sup_j |p_{ij}^{(n)} - \pi_j| \to 0 for n \to \infty \tag{2.29}$$

We refer to the latter as the ergodic theorem.

2.3 Continuous time Markov chain

Now that discrete phase-type distributions have been described briefly, a slightly more comprehensive study of continuous phase-type distributions will be covered.

A continuous time stochastic process is a Markov jump process with state-space E, if for all $0 < t_1 < t_2 < \cdots < t_n$ and $i_0, i_1, \ldots, i_n \in E$, it holds that

$$\mathbb{P}\left(X_{t_n} = i_n \,\middle|\, X_{t_{n-1}} = i_{n-1}, \ldots, X_{t_1} = i_1, X_{t_0} = i_0\right) = \mathbb{P}\left(X_{t_n} = i_n \,\middle|\, X_{t_{n-1}} = i_{n-1}\right). \tag{2.30}$$

The continuous time Markov chain is time-homogeneous if $\mathbb{P}(X_{t+h} = j \mid X_t = i)$ only depends on h. If that's the case, we let $p_{ij}(h) = \mathbb{P}(X_{t+h} = j \mid X_t = i)$ denote the h-step transition probabilities which we arrange in h-step transition matrix $\boldsymbol{P}(h) = \left\{p_{ij}(h)\right\}_{i,j\in E}$.

Denote by τ_i the holding time, or sojourn time, in state i before jumping to the next state j. Time-homogeneity and the Markov property results in a memoryless continuous distributed holding time on $[0, \infty)$, confirmed by the calculations below:

$$\begin{aligned}\mathbb{P}(\tau_i > s + t | \tau_i > s) &= \mathbb{P}(X(u) = i \text{ for } u \in [0, s + t] | X(u) = i \text{ for } u \in [0, s]) \\ &= \mathbb{P}(X(u) = i \text{ for } u \in [0, t] | X(0) = i) \\ &= \mathbb{P}(\tau_i > t)\end{aligned} \tag{2.31}$$

It was proven in section 2.1 that if a random variable contains the memoryless property, then the random variable is exponentially distributed. Hence the holding time τ_i has an exponential distribution $\tau_i \sim \exp(\lambda_i)$ with some state-specific parameter λ_i. If the transition intensity is zero, then the state is said to be an absorbing state, and the process never leaves after the process has jumped into the absorbing state. We let $T_1, T_2, \ldots, T_n$ denote the jumping times of the Markov chain, and use the convention $T_0 = 0$, such that $T_k = \sum_{i=0}^{k} \tau_i$. Then a discrete time Markov chain $\{Y_n\}_{n \in \mathbb{N}}$ can be defined where $Y_n = X(T_n)$. We call $\{Y_n\}_{n \in \mathbb{N}}$ the embedded Markov chain and let $\boldsymbol{U} = \left\{u_{ij}\right\}_{i,j \in E}$ denote the corresponding transition matrix. Given that the process sojourn in state i, the probability of jumping to another state, during the time interval $[t, t + h)$, is given by

$$\begin{aligned}1 - p_{ii}(h) &= 1 - \mathbb{P}(X(h) = i | X(0) = i) \\ &\leq 1 - \mathbb{P}(\tau_1 > h | X(0) = i).\end{aligned} \tag{2.32}$$

The inequality is true since the second conditional probability is to a set that excludes any indirect transition from state i to state i, transitted through other states. In other words, the last set excludes the possibility of sojourning in other states than i during a time period of lenght h. However, for h infinitesimal small, the possibility of any indirect transition becomes zero such that the first probability converges to the second probability. Hence, by using the exponential distributions of the waiting times we can, for h small enough, write

$$\begin{aligned}1 - p_{ii}(h) &= 1 - \mathbb{P}(\tau_1 > h | X(0) = i) \\ &= 1 - e^{-\lambda_i h} \\ &= \lambda_i h + o(h).\end{aligned} \tag{2.33}$$

Likewise, we have

$$\begin{aligned}p_{ij}(h) &= \mathbb{P}(X(h) = j | X(0) = i) \\ &\to \mathbb{P}(\tau_1 \leq h,\ X(\tau_1) = j,\ \tau_2 - \tau_1 > h | X(0) = i)\end{aligned} \tag{2.34}$$

for $h \to 0$. Let $Y_n = X(T_n)$ be the embedded Markov process, and let the conditional transition probability from state i to state j be given by u_{ij}, where u_{ij} is the entry of

the ith row and the jth column in the transition matrix of the embedded Markov chain. Moreover, since the waiting time is exponentially distributed, it follows that

$$\begin{aligned} p_{ij}(h) &= \mathbb{P}(\tau_1 \leq h,\ X(\tau_1) = j,\ \tau_2 - \tau_1 > h | X(0) = i) \\ &= (1 - e^{-\lambda_i h}) u_{ij} e^{-\lambda_j h} \\ &= (\lambda_i h + o(h)) u_{ij} (1 - \lambda_j h + o(h)) \\ &= \lambda_i u_{ij} h + o(h). \end{aligned} \tag{2.35}$$

As defined above the transition intensities λ_i are the rates at which the system leaves state i and λ_{ij} the rate at which the system leaves state i to state j. By using 2.38, the transition intensities can be expressed as

$$\lambda_i = \lim_{h \to 0} (1 - p_{ii}(h)) / h \tag{2.36}$$

and

$$\lambda_{ij} = \lim_{h \to 0} p_{ij}(h) / h \ = \lambda_i u_{ij}, \tag{2.37}$$

where u_{ij} is the probability of jumping from state i to j in the embedded Markov chain. From the definition of the embedded Markov chain, it is clear that $u_{ii} = 0$. Summing over all the states different from i gives the following relation between the transition intensities:

$$\sum_{j \neq i} \lambda_{ij} = \sum_{j \neq i} \lambda_i u_{ij} = \lambda_i, \tag{2.38}$$

since $\sum_{j \neq i} u_{ij} = 1$. Setting $\lambda_i = -\lambda_{ii}$, we can arrange the transition intensities in an infinitesimal generator, or intensity matrix, $\mathbf{\Lambda}$, defined by

$$\mathbf{\Lambda} = \begin{cases} -\lambda_i \text{ for } i = j \\ \lambda_{ij} \text{ for } i \neq j \end{cases}, \tag{2.39}$$

where λ_i and λ_{ij} are the transition intensities defined above.

We have the following important relations between the transition intensities and the transition probabilities:

$$P(t) = e^{\mathbf{\Lambda} t} \tag{2.40}$$

This relation will be used frequently in the next chapter.

2.4 Bibliographic references

The memoryless property is introduced in [18]. Prove that the exponential distribution and geometric distribution is the only memoryless continuous and discrete distribution respectively is inspired by [2] and [19]. Proof of memoryless property and the relation between transition intensities transition probabilities are based on [38]. Proof that

the exponential distribution is the only continuous distribution with the memoryless property are inspired by [2]. Markov chains are based on the pioneering work from Andrey Markov and was introduced back in 1906. Inspiration has also been found in [8] and [17] when introducing the contents of this section.

Chapter 3

Phase-type distributions

In section 3.1 discrete phase-type distributions and its theory are described briefly. Since continuous phase-type distributions will be the main subject in the estimation later on, less attention is given to the discrete phase-type distributions. In section 3.2 a more comprehensive introduction to continuous phase-type distributions are carried out. This includes derivation of the density, distribution, moments of arbitrary order and the Laplace transformation on closed-form expressions. Section 3.3 includes maximum likelihood estimation of phase-type distributions, both discrete and continuous. In section 3.4 examples of sub-classes of phase-type distributions are given.

3.1 Discrete phase-type distributions

Let $\{X_n\}_{n\in\mathbb{N}}$ be a discrete time Markov chain with state-space $E = \{1,\ 2, \ldots, p+1\}$. We assume the states $1, 2, \ldots, p$ are transient and $p+1$ is an absorbing state. Furthermore, we assume that the Markov chain does not start in the absorbing state, such that the initial distribution is

$$\boldsymbol{\pi} = \begin{pmatrix} \pi_1 & \pi_2 & \ldots & \pi_p & 0 \end{pmatrix}, \tag{3.1}$$

where $\pi_i = \mathbb{P}(X_0 = i)$. It then follows that the transition matrix is given by

$$\boldsymbol{P} = \begin{pmatrix} \boldsymbol{T} & \boldsymbol{t} \\ \boldsymbol{0} & 1 \end{pmatrix}, \tag{3.2}$$

where $\boldsymbol{T}$ is a $p \times p$-dimensional sub-transition matrix. $\boldsymbol{t}$ is a p-dimensional vector where entry i contains the probability of jumping from state i to the absorbing state $p+1$, also called the exit probability. Since $\boldsymbol{P}$ is a probability matrix the elements of the rows sum to 1, such that $\boldsymbol{P}\overline{\boldsymbol{e}} = \boldsymbol{e}$ where $\overline{\boldsymbol{e}}$ is a $p+1$-dimensional vector of 1s. Hence

$$\begin{aligned} \boldsymbol{T}\boldsymbol{e} + \boldsymbol{t} = \boldsymbol{e} \Leftrightarrow \\ \boldsymbol{t} = (\boldsymbol{I} - \boldsymbol{T})\,\boldsymbol{e} \end{aligned} \tag{3.3}$$

where $\boldsymbol{e}$ is a p-dimensional vector of 1s. Let

$$\tau = \inf\{n \geq 1 \mid X_n = p+1\} \tag{3.4}$$

be the time until absorption of the discrete time Markov chain $\{X_n\}_{n\in\mathbb{N}}$ described above and let the initial distribution $\boldsymbol{\pi}$ and the transition matrix $\boldsymbol{P}$ given be by 3.1 and 3.2 respectively. Then τ is a discrete phase-type distributed random variable with initial distribution $\boldsymbol{\pi}$ and sub-transition matrix $\boldsymbol{T}$. We use the notation $\tau \sim DPH_p(\boldsymbol{\pi}, \boldsymbol{T})$ or $\tau \sim DPH(\boldsymbol{\pi}, \boldsymbol{T})$.

$\boldsymbol{I} - \boldsymbol{T}$ is invertible if all the states in the sub-transition matrix are transient. If $DPH_p(\boldsymbol{\pi}, \boldsymbol{T})$ is a phase-type representation then $\boldsymbol{U} = \{u_{ij}\} = (\boldsymbol{I} - \boldsymbol{T})^{-1}$ is called its Green matrix. u_{ij} is the expected time spent in state j, of the underlying Markov jump process, prior to absorption and given initiation in state i.

As mentioned in the introduction, some functionals of phase-type distributions have mathematical convenient closed-form expressions. The density f and distribution F of $\tau \sim DPH_p(\boldsymbol{\pi}, \boldsymbol{T})$ is given by

$$f(n) = \boldsymbol{\pi}\boldsymbol{T}^{n-1}\boldsymbol{t} \tag{3.5}$$

for $n \geq 1$ and

$$F(n) = 1 - \boldsymbol{\pi}\boldsymbol{T}^{n}\boldsymbol{e} \tag{3.6}$$

for $n \geq 1$ respectively. The n'th order moment of a phase-type distributed random variable $\tau \sim PH_p(\boldsymbol{\pi}, \boldsymbol{T})$ is given by

$$E(\tau^n) = n!\boldsymbol{\pi}\boldsymbol{U}^n\boldsymbol{e} = n!\boldsymbol{\pi}(-\boldsymbol{T})^n\boldsymbol{e}, \tag{3.7}$$

where $n!\boldsymbol{U}^n_{ij}$ is the expected volume generated by the underlying Markov jump process in state j given initiation in state i. If $\tau \sim DPH(\pi, \boldsymbol{T})$ the probability generating function is given by

$$\mathbb{E}(z^\tau) = z\boldsymbol{\pi}(\boldsymbol{I} - z\boldsymbol{T})^{-1}\boldsymbol{t} = \boldsymbol{\pi}(z^{-1}\boldsymbol{I} - \boldsymbol{T})^{-1}\boldsymbol{t}, \tag{3.8}$$

which is a rational function and exists at least for $|z| \leq 1$.

The discrete phase-type distribution is closed under convolution, meaning that, if $\tau_1 \sim DPH_p(\boldsymbol{\alpha}, \boldsymbol{S})$ and $\tau_2 \sim DPH_q(\boldsymbol{\beta}, \boldsymbol{T})$ are independent, then

$$\tau_1 + \tau_2 \sim DPH_{p+q}\left(\begin{pmatrix} \boldsymbol{\alpha} & \boldsymbol{0} \end{pmatrix}, \begin{pmatrix} \boldsymbol{S} & \boldsymbol{\beta} \\ 0 & \boldsymbol{T} \end{pmatrix} \right). \tag{3.9}$$

Let τ_1 and τ_2 have the densities f_1 and f_2, respectively and the same distributions as above. If $p \in [0,1]$ then $f(x) = pf_1(x) + (1-p)f_2(x)$ is the density for the phase-type distribution with representation

$$DPH_{p+q}\left((p\boldsymbol{\alpha}(1-p)\boldsymbol{\beta}), \begin{pmatrix} \boldsymbol{S} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{T} \end{pmatrix} \right). \tag{3.10}$$

Hence discrete time phase-type distributions is closed under convex mixtures.

3.2 Continuous phase-type distributions

Let $\{X_t\}_{t\geq 0}$ be a Markov jump process with state-space $E = \{1, 2, \ldots, p+1\}$. We assume the states $1, 2, \ldots, p$ are transient and $p+1$ is an absorbing state, and we refer to the transient states as the phases. Furthermore we assume that the Markov jump process does not start in the absorbing state, such that the initial distribution is

$$\boldsymbol{\pi} = \begin{pmatrix} \pi_1 & \pi_2 & \ldots & \pi_p & 0 \end{pmatrix}, \tag{3.11}$$

where $\pi_i = \mathbb{P}(X_0 = i)$. It then follows that the intensity matrix or infinitesimal generator $\boldsymbol{\Lambda}$ is given by

$$\boldsymbol{\Lambda} = \begin{pmatrix} \boldsymbol{T} & \boldsymbol{t} \\ \mathbf{0} & 0 \end{pmatrix}. \tag{3.12}$$

$\boldsymbol{T}$ is a $p \times p$-dimensional sub-intensity matrix, also called the phase-type generator. $\boldsymbol{t}$ is a p-dimensional vector called the exit vector, where entry i contains the conditional intensity t_i from state i to the absorbing state, called the exit rate. Since $\boldsymbol{\Lambda}$ is an intensity matrix the elements of the rows sum to 0, as shown in section 2.3. Therefore $\boldsymbol{\Lambda}\bar{\boldsymbol{e}} = \mathbf{0}$ where $\bar{\boldsymbol{e}}$ is a $p+1$-dimensional vector of 1s. Consequently,

$$\boldsymbol{Te} - \boldsymbol{t} = \mathbf{0} \Leftrightarrow \boldsymbol{t} = -\boldsymbol{Te} \tag{3.13}$$

where $\boldsymbol{e}$ is a p-dimensional vector of 1s. Let

$$\tau = \inf\{t > 0 \mid X_t = p+1\} \tag{3.14}$$

be time until absorption of the Markov jump process $\{X_t\}_{t\geq 0}$ with an initial distribution $\boldsymbol{\pi}$ and an intensity matrix $\boldsymbol{\Lambda}$, given by 3.11 and 3.12 respectively. Then τ is a phase-type distributed random variable with initial distribution $\boldsymbol{\pi}$ and sub-intensity matrix $\boldsymbol{T}$. We use the notation $\tau \sim PH_p(\boldsymbol{\pi}, \boldsymbol{T})$ or $\tau \sim PH(\boldsymbol{\pi}, \boldsymbol{T})$.

It can be proven that a sub-intensity matrix is invertible if all the states in the sub-intensity matrix are transient. Therefore, if $PH_p(\boldsymbol{\pi}, \boldsymbol{T})$ is a phase-type representation, then the corresponding sub-intensity matrix $\boldsymbol{T}$ is invertible. If $PH_p(\boldsymbol{\pi}, \boldsymbol{T})$ is a phase-type representation then $\boldsymbol{U} = \{u_{ij}\} = (-\boldsymbol{T})^{-1}$ is called its Green matrix. u_{ij} is the expected time spent in state j of the underlying Markov jump process, prior to absorption, and given initiation in state i.

Using 2.40 in last chapter, the transition probability restricted to the transient states can be calculated as

$$p_{ij}(s) = \mathbb{P}(X(s) = j | X(0) = i) = (e^{\boldsymbol{T}s})_{ij}. \tag{3.15}$$

By using the law of total probability twice and that

$$\sum_{j=1}^{p} \mathbb{P}(\tau \in [s,\ s+ds) | X(s) = j) = t_j ds, \tag{3.16}$$

the density f of $\tau \sim PH_p(\boldsymbol{\pi}, \boldsymbol{T})$ can be derived by

$$\begin{aligned} f(s)ds &= \mathbb{P}(\tau \in [s,\ s+ds)) \\ &= \sum_{j=1}^{p} \mathbb{P}(\tau \in [s,\ s+ds) | X(s) = j) \mathbb{P}(X(s) = j) \\ &= \sum_{j=1}^{p} \mathbb{P}(\tau \in [s,\ s+ds) | X(s) = j) \sum_{i=1}^{p} \mathbb{P}(X(s) = j | X(0) = i) \mathbb{P}(X(0) = i) \\ &= \sum_{j=1}^{p} t_j ds \sum_{i=1}^{p} (e^{\boldsymbol{T}s})_{ij} \pi_i \\ &= \sum_{i=1}^{p} \sum_{j=1}^{p} \pi_i (e^{\boldsymbol{T}s})_{ij} t_j ds \\ &= \boldsymbol{\pi} e^{\boldsymbol{T}s} \boldsymbol{t} ds, \end{aligned} \tag{3.17}$$

so

$$f(u) = \boldsymbol{\pi} e^{\boldsymbol{T}u} \boldsymbol{t}. \tag{3.18}$$

It is clear that, the probability of absorption before time s, is the same as, the probability that the underlying process is in the absorbing state at time s. Furthermore, since $\{X(s) = j\}$ are pairwise disjoint for $j \in \{1, \ldots, p\}$ we have

$$\bigcap_{j=1}^{p} \{X(s) = j\} = \emptyset. \tag{3.19}$$

Using the law of total probability and sigma additivity the cumulative probability function of the continuous phase-type distributions can be derived:

$$
\begin{aligned}
F(s) &= \mathbb{P}(\tau \leq s) \\
&= \mathbb{P}(X(s) \in \{1+p\}) \\
&= 1 - \mathbb{P}(X(s) \in \{1, \ldots, p\}) \\
&= 1 - \mathbb{P}\left(\bigcup_{j=1}^{p} \{X(s) = j\}\right) \\
&= 1 - \sum_{j=1}^{p} \mathbb{P}(X(s) = j) + \mathbb{P}\left(\bigcap_{j=1}^{p} \{X(s) = j\}\right) \\
&= 1 - \sum_{j=1}^{p} \mathbb{P}(X(s) = j) \\
&= 1 - \sum_{i,j=1}^{p} \mathbb{P}(X(s) = j | X(0) = i)\mathbb{P}(X(0) = i) \\
&= 1 - \sum_{i,j=1}^{p} p_{ij}^{s} \pi_i \\
&= 1 - \sum_{i,j=1}^{p} \pi_i (e^{\boldsymbol{T}s})_{ij} \\
&= 1 - \boldsymbol{\pi} e^{\boldsymbol{T}s} \boldsymbol{e}
\end{aligned}
\tag{3.20}
$$

Using integration by parts, the moment can be calculated:

$$
\begin{aligned}
\mathbb{E}(\tau) &= \int_0^\infty s f(s) ds \\
&= \int_0^\infty s \boldsymbol{\pi} e^{\boldsymbol{T}s} \boldsymbol{t} ds \\
&= \left[s \boldsymbol{\pi} e^{\boldsymbol{T}s} \boldsymbol{T}^{-1} \boldsymbol{t} \right]_0^\infty - \int_0^\infty \boldsymbol{\pi} e^{\boldsymbol{T}s} \boldsymbol{T}^{-1} \boldsymbol{t} ds \\
&= \lim_{s \to \infty} \boldsymbol{\pi} e^{\boldsymbol{T}s} \boldsymbol{T}^{-1} \boldsymbol{t} - \int_0^\infty \boldsymbol{\pi} e^{\boldsymbol{T}s} \boldsymbol{T}^{-1} \boldsymbol{t} ds
\end{aligned}
\tag{3.21}
$$

It can be shown that all the eigenvalues of $\boldsymbol{T}$ are negative and as a result of that $\boldsymbol{T}$ is negative definite. Since $\boldsymbol{T}$ is negative definite, $\lim_{s \to \infty} \boldsymbol{\pi} e^{\boldsymbol{T}s} \boldsymbol{T}^{-1} \boldsymbol{t} = 0$. Consequently,

$$
\begin{aligned}
\mathbb{E}(\tau) &= -\int_0^\infty \boldsymbol{\pi} e^{\boldsymbol{T}s} \boldsymbol{T}^{-1} \boldsymbol{t} ds \\
&= -\left[\boldsymbol{\pi} e^{\boldsymbol{T}s} \boldsymbol{T}^{-2} \boldsymbol{t} \right]_0^\infty \\
&= -\lim_{s \to \infty} \boldsymbol{\pi} e^{\boldsymbol{T}s} \boldsymbol{T}^{-2} \boldsymbol{t} + \boldsymbol{\pi} e^{\boldsymbol{T}0} \boldsymbol{T}^{-2} \boldsymbol{t}.
\end{aligned}
\tag{3.22}
$$

By the same argument as before, $\lim_{s \to \infty} \boldsymbol{\pi} e^{\boldsymbol{T}s} \boldsymbol{T}^{-2} \boldsymbol{t} = 0$, so 3.22 reduces to

$$\begin{aligned}\mathbb{E}(\tau) &= \boldsymbol{\pi}\boldsymbol{T}^{-2}\boldsymbol{t} \\ &= \boldsymbol{\pi}\boldsymbol{T}^{-2}(-\boldsymbol{T}\boldsymbol{e}) \\ &= -\boldsymbol{\pi}\boldsymbol{T}^{-1}\boldsymbol{e}\end{aligned} \tag{3.23}$$

Using integration by parts iteratively and 3.23, we derive the n'th order moment of a phase-type distributed random variable $\tau \sim PH_p(\boldsymbol{\pi}, \boldsymbol{T})$. The nth order moment is given by

$$\begin{aligned}\mathbb{E}(\tau^n) &= \int_0^\infty s^n f(s) ds \\ &= \int_0^\infty s^n \boldsymbol{\pi} e^{\boldsymbol{T}s}\boldsymbol{t} ds \\ &= -\int_0^\infty n s^{n-1}\boldsymbol{\pi} e^{\boldsymbol{T}s}\boldsymbol{T}^{-1}\boldsymbol{t} ds \\ &= -n\boldsymbol{T}^{-1}\int_0^\infty s^{n-1}\boldsymbol{\pi} e^{\boldsymbol{T}s}\boldsymbol{t} ds \\ &= -n\boldsymbol{T}^{-1}(-1)^{n-2}(n-1)!\boldsymbol{T}^{-(n-2)}\int_0^\infty s\boldsymbol{\pi} e^{\boldsymbol{T}s}\boldsymbol{t} ds \\ &= -n\boldsymbol{T}^{-1}(-1)^{n-2}(n-1)!\boldsymbol{T}^{-(n-2)}\mathbb{E}(\tau) \\ &= -n\boldsymbol{T}^{-1}(-1)^{n-2}(n-1)!\boldsymbol{T}^{-(n-2)}(-\boldsymbol{\pi}\boldsymbol{T}^{-1}\boldsymbol{e}) \\ &= (-1)^n n!\boldsymbol{\pi}\boldsymbol{T}^{-n}\boldsymbol{e},\end{aligned} \tag{3.24}$$

where $n!\boldsymbol{T}_{ij}^{-n}$ is the expected volume generated by the underlying Markov jump process in state j given initiation in state i. The Laplace transform of a phase-type distribution is given by

$$\begin{aligned}\mathbb{E}(e^{-s\tau}) &= \int_0^\infty e^{-sx} f(x) dx \\ &= \int_0^\infty e^{-sx}\boldsymbol{\pi} e^{\boldsymbol{T}x}\boldsymbol{t} dx \\ &= \int_0^\infty \boldsymbol{\pi} e^{-sx}\boldsymbol{I} e^{\boldsymbol{T}x}\boldsymbol{t} dx \\ &= \int_0^\infty \boldsymbol{\pi} e^{-s\boldsymbol{I}x} e^{\boldsymbol{T}x}\boldsymbol{t} dx \\ &= \int_0^\infty \boldsymbol{\pi} e^{(-s\boldsymbol{I}+\boldsymbol{T})x}\boldsymbol{t} dx \\ &= \boldsymbol{\pi}(s\boldsymbol{I}-\boldsymbol{T})^{-1}\boldsymbol{t}.\end{aligned} \tag{3.25}$$

The Laplace transform of a phase-type distribution $\boldsymbol{\pi}(s\boldsymbol{I}-\boldsymbol{T})^{-1}\boldsymbol{t}$ is a rational function. In fact, phase-type distributions constitute a subclass of the so-called matrix exponential distributions, which are distributions with rational Laplace transformations. Matrix

exponential distributions can also be represented by a vector $\boldsymbol{\pi}$ and a matrix $\boldsymbol{T}$ but, as opposed to phase-type distributions, no restrictions on the parameter values are given.

If we can write the exit rate as $\boldsymbol{t} = -\boldsymbol{T}\boldsymbol{e} = \lambda \boldsymbol{e}$ for some constant $\lambda > 0$ then $\tau \sim \exp(\lambda)$. This shows that the phase-type distribution may be over-parameterized and that the dimension p does not necessarily reflect the true order of a phase-type distribution.

Furthermore, we say that the phase-type representation is irreducible if all phases have a positive probability of being visited before absorption. For our phase-type representation to be irreducible, no superfluous phases are allowed which means that each transient state must have a positive probability of being visited. Assuming that the phase-type distribution does not have an atom at zero, an irreducible representation is equivalent to an irreducible sub-intensity matrix. If the sub-intensity matrix is reducible, superfluous phases can just be removed to make it irreducible.

Continuous phase-type distribution is also closed under convolution. Analogous to the discrete case, let $\tau_1 \sim PH_p(\boldsymbol{\alpha}, \boldsymbol{S})$ and $\tau_2 \sim PH_q(\boldsymbol{\beta}, \boldsymbol{T})$ be independent. Then it holds that

$$\tau_1 + \tau_2 \sim PH_{p+q}\left(\begin{pmatrix} \boldsymbol{\alpha} & \boldsymbol{0} \end{pmatrix}, \begin{pmatrix} \boldsymbol{S} & \boldsymbol{\beta} \\ 0 & \boldsymbol{T} \end{pmatrix} \right). \tag{3.26}$$

Assuming τ_1 and τ_2 have the densities f_1 and f_2, respectively, it furthermore holds that, if $p \in [0, 1]$, then $f(x) = pf_1(x) + (1 - p)f_2(x)$ is the density for the phase-type distribution with representation

$$PH_{p+q}\left((p\boldsymbol{\alpha}(1-p)\boldsymbol{\beta}), \begin{pmatrix} \boldsymbol{S} & \boldsymbol{0} \\ \boldsymbol{0} & \boldsymbol{T} \end{pmatrix} \right). \tag{3.27}$$

Hence, continuous time phase-type distributions is also closed under convex mixtures.

3.3 Likelihood function for phase-type distribution

Likelihood functions for phase-type distributions will now be derived. The continuous and discrete case will be treated separately.

Continuous phase-type distribution

Consider one observation x as a realization of an underlying Markov jump process $\{X_t\}_{0<t\leq\tau}$ to a phase-type distributed random variable $\tau \sim PH_p(\pi, \boldsymbol{T})$. Then there is a one-to-one correspondence between $\{x_t\}_{0<t\leq\tau}$ and $\{(i_k, u_k)\}_{k=0,1,\ldots,n}$ where $i_0, i_1, \ldots, i_n$ are the states visited prior to absorption and $u_0, u_1, \ldots, u_n$ are the time spend in the corresponding states. From the results in section 2.1 regarding the memoryless property, we know that u_k are realizations of exponential distributed variables with rate parameters $\lambda_{i_k i_k} = -t_{i_k i_k}$ where $t_{i_k i_k}$ is entry (i_k, i_k) in the sub-intensity matrix $\boldsymbol{T}$. i_k is a realization of the embedded Markov chain with transition probabilities $p_{ij} = -\frac{t_{ij}}{t_{ii}}$ for $i, j = 1, 2, \ldots, p$ and $i \neq j$. Moreover, $p_{i,p+1} = -\frac{t_i}{t_{ii}}$ where t_i is the i'th entry in the exit rate vector.

Let N_{ij} be the number of observed jumps from state i to state j, Z_i be the total time spent in state i and N_i be the number of realizations which exits to the absorbing state

from state i. By some rearrangements, and by the use of 2.37, we can calculate the complete likelihood function as

$$\begin{aligned} L_c(\theta; x) &= \pi_{i_0} \left(\prod_{k=0}^{n-1} \lambda_{i_k} e^{-\lambda_{i_k} u_k} p_{i_k i_{k+1}} \right) \lambda_{i_n} e^{-\lambda_{i_n} u_n} p_{i_n, p+1} \\ &= \pi_{i_0} \left(\prod_{k=0}^{n-1} e^{-u_k \left(\sum_{j \neq i_k} t_{i_k j} + t_{i_k} \right)} t_{i_k i_{k+1}} \right) e^{-u_n \left(\sum_{j \neq i_n} t_{i_n j} + t_{i_n} \right)} t_{i_n, p+1} \\ &= \pi_{i_0} \left(\prod_{i=1}^{p} \prod_{j=1,\ j \neq i}^{p} t_{ij}^{N_{ij}} e^{t_{ij} Z_i} \right) \prod_{i=1}^{p} t_i^{N_i} e^{t_i Z_i}. \end{aligned} \tag{3.28}$$

Notice that, since we only consider a single realization, N_i can at most be 1.

Now consider N iid. observations $x_1, x_2, \ldots, x_N$ where x_i is a realization of the underlying Markov jump process $\{X_t\}_{0<t\leq\tau_i}$ to a phase-type distributed random variable τ_i. Let B_i be the number of observed processes which initiate in state i. Then we can calculate the complete likelihood as the product of the individual complete likelihood functions:

$$\begin{aligned} L_c(\theta; x) &= \prod_{i=1}^{N} L_c(\theta; x_i) \\ &= \left(\prod_{i=1}^{p} \pi_i^{B_i} \right) \left(\prod_{i=1}^{p} \prod_{j=1,\ j \neq i}^{p} t_{ij}^{N_{ij}} e^{t_{ij} Z_i} \right) \left(\prod_{i=1}^{p} t_i^{N_i} e^{t_i Z_i} \right) \end{aligned} \tag{3.29}$$

Taking the natural logarithm to the likelihood function gives

$$\begin{aligned} \ell_c(\theta; x_1, \ldots, x_N) &= \log L_c(\theta; x_1, .., x_N) \\ &= \sum_{i=1}^{p} B_i \log(\pi_i) p + \sum_{i=1}^{p} \sum_{j=1, j \neq i}^{p} N_{ij} \log(t_{ij}) \\ &\quad - \sum_{i=1}^{p} \sum_{j=1, j \neq i}^{p} Z_i t_{ij} + \sum_{i=1}^{p} N_i \log(t_i) - \sum_{i=1}^{p} Z_i t_i \end{aligned} \tag{3.30}$$

Notice that the log likelihood function is linear in the sufficient statistics. Hence, it suffices to calculate the sufficient statistics in maximum likelihood problems for phase-type distributions.

To find the maximum likelihood estimators, the complete log-likelihood function must be maximized under the condition that the initial probabilities sum up to one. From the Lagrange function

$$H(\pi_1, \ldots, \pi_p) = \sum_{j=1}^{p} B_i \log(\pi_i) - c \left(\sum_{j=1}^{p} \pi_i - 1 \right), \tag{3.31}$$

we calculate the gradient

$$\nabla H(\pi_1, \ldots, \pi_p) = \begin{pmatrix} \frac{B_1}{\pi_1} - c \\ \vdots \\ \frac{B_p}{\pi_p} - c \end{pmatrix}. \tag{3.32}$$

Setting this gradient equal to a 0 vector, gives p equations. Now, take an arbitrary equation and isolate π_i:

$$\pi_i = B_i/c \tag{3.33}$$

Summing over all entries shows that the Lagrange multiplier c equals N since $\sum_{i=1}^{N} B_i = N$ and $\sum_{i=1}^{N} \pi_i = 1$ per definition. Inserting $c = N$ into 3.33 gives the estimate

$$\widehat{\pi_i} = \frac{B_i}{N}. \tag{3.34}$$

The use of classical optimization techniques can be applied to find the remaining estimators, which is given by

$$\begin{aligned} \hat{t}_{ij} &= \frac{N_{ij}}{Z_i} \\ \hat{t}_i &= \frac{N_i}{Z_i} \\ \widehat{\pi_i} &= \frac{B_i}{N} \end{aligned} \tag{3.35}$$

From the second order condition we see that

$$\begin{aligned} \frac{\partial^2 \ell_c}{\partial t_{ij}^2} &= -\frac{N_{ij}}{t_{ij}^2} < 0 \\ \frac{\partial^2 \ell_c}{\partial t_i^2} &= -\frac{N_i}{t_i^2} < 0 \\ \frac{\partial^2 \ell_c}{\partial \pi_i^2} &= -\frac{B_i}{\pi_i^2} < 0. \end{aligned} \tag{3.36}$$

Hence the Hessian matrix is negative definite which confirms that this is in fact a maximum.

Discrete phase-type distribution

Now consider discrete phase-type distributions. Let $x_1, x_2, \ldots, x_N$ be i.i.d. realizations of a discrete time Markov chain until the time of absorption. By an argument analogous to the continuous case, a formula for the complete likelihood function for the discrete phase-type distribution can be derived.

$$L_c(\theta; x_1, \ldots, x_N) = \prod_{i=1}^{p} \pi_i^{B_i} \prod_{i=1}^{p} \prod_{j=1}^{p} t_{ij}^{N_{ij}} \prod_{i=1}^{p} t_i^{N_i} \tag{3.37}$$

where $\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{T})$. B_i is the number of realasations which initiates in state i, N_{ij} is the total number of transitions from state i to state j and N_i as the number of transitions from state i to the absorbing state. Taking the natural logarithm to 3.37 will show that the sufficient statistics are linear in the log likelihood function, just as in the continuous case. maximizing the log-likelihood function under the condition that the entries in the initial probability vector sum to one, gives the estimates can be found as,

$$\hat{\pi}_i = \frac{B_i}{N}, \tag{3.38}$$

$$\hat{t}_{ij} = \frac{N_{ij}}{\sum_{j=1}^{p} N_{ij} + N_i}, \tag{3.39}$$

$$\hat{t}_i = \frac{N_i}{\sum_{j=1}^{p} N_{ij} + N_i} \tag{3.40}$$

for $i, j = 1, \cdots, p$. By calculating the second order derivatives we easily verify that the stationary point found is, in fact, a maximum.

3.4 Special cases of phase-type distribution

This section includes some distributions which constitute a subclass of phase-type distributions. By using the closure properties of phase-type distributions, we will see how any series-parallel arrangements of exponential distributions or geometrical distributions can be expressed in terms of a phase-type representation. Moreover, an interpretation of their correspondence underlying Markov chain is given.

3.4.1 Special cases of continuous phase-type distributions

All special cases below constitute subclass of continuous phase-type distributions.

Exponential distribution

The exponential distribution can be seen as an one order continuous phase-type distributed random variable with representation $\tau \sim PH_1(1, -\lambda)$ and exit rate λ. The rest of the phase-type representations in this subsection follows from the closure properties of phase-type distributed random variables.

Hyperexponential distribution

A hyperexponential distributed random variable $\tau = \sum_{i=1}^{p} 1_{\{N=i\}}(X_i)$ is a mixture of exponential distributed random variables $X_j \sim \exp\left(\lambda_j\right)$, such that the density is given by

$$f_\tau(x) = \sum_{i=j}^{p} \pi_j f_j(x) = \sum_{i=j}^{p} \pi_j \lambda_j e^{-\lambda_j x}. \tag{3.41}$$

This have phase-type representation

$$\tau \sim PH_p\left(\begin{pmatrix} \pi_1 & \dots & \pi_p & 0 \end{pmatrix}, \begin{pmatrix} -\lambda_1 & 0 & \cdots & 0 \\ 0 & -\lambda_2 & \cdots & 0 \\ \vdots & \vdots & \ddots & 0 \\ 0 & 0 & 0 & -\lambda_p \end{pmatrix}\right), \tag{3.42}$$

with exit vector

$$\boldsymbol{t} = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_p \end{pmatrix}. \tag{3.43}$$

Hence, we have p parallel phases as initial phases, and the process will only visit one of these phases one time before it terminates in the absorbing state.

Erlang-p distribution

The Erlang-p distribution is the sum of p exponential distributed variables with similar rate parameters, such that if $X_j \sim \exp(\lambda)$ for $j \in \{1, \dots, p\}$ then $\tau = \sum_{j=1}^{p} X_j \sim Er_p(\lambda)$ with density

$$f_\tau(x) = \sum_{j=1}^{p} f_j(x) = \frac{\lambda^p}{(p-1)!} x^{p-1} e^{-\lambda x} \tag{3.44}$$

Furthermore, the Erlang-p distribution is also a special case of the gamma distribution $\Gamma(p, \beta)$, where p is restricted to the natural numbers. This corresponds to a process which initiates in state 1 and progress through the phases sequentially before terminating in the absorbing state. Hence, the Erlang-p distribution have phase-type representation

$$\tau \sim PH_p\left(\begin{pmatrix} 1 & 0 & \dots & 0 \end{pmatrix}, \begin{pmatrix} -\lambda & \lambda & \cdots & 0 & 0 \\ 0 & -\lambda & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & 0 & -\lambda & \lambda \\ 0 & 0 & 0 & 0 & -\lambda \end{pmatrix}\right) \tag{3.45}$$

with exit vector

$$\boldsymbol{t} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ \lambda \end{pmatrix} \tag{3.46}$$

A Hypoexponential Distribution is a generalization of Erlang distribution where the rates are allowed to differ.

Cox distribution

A Cox distribution, also known as branching Erlang distribution and Coxian distribution, is a special case of a phase-type distribution, where the transient states are ordered. The process progress through the phases sequentially, with the exception that the process at any state can jump directly to the absorbing state. This have phase-type representation $X \sim PH_p(\boldsymbol{e}_1, \boldsymbol{T})$

$$\boldsymbol{T} = \begin{pmatrix} -(\lambda_1 + t_1) & \lambda_1 & \cdots & 0 & 0 \\ 0 & -(\lambda_2 + t_2) & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & 0 & -\left(\lambda_{p-1} + t_{p-1}\right) & \lambda_{p-1} \\ 0 & 0 & 0 & 0 & -\left(\lambda_p + t_p\right) \end{pmatrix}, \tag{3.47}$$

with exit vector

$$\boldsymbol{t} = \begin{pmatrix} t_1 \\ t_2 \\ \vdots \\ t_p \end{pmatrix}. \tag{3.48}$$

Generalized Erlang-p distribution

In the generalized Erlang distribution, a hyper-exponential distribution is combined with a Erlang-p distribution, resulting in a mixture of Erlang-p distributions. This can be presented as a phase-type distribution where the underlying process progress sequentially through one of p parallel levels, each containing a series of p phases, and afterwards terminating in the absorbing state.

If the waiting time $X_{i,j}$ in state i is exponential distributed i.e. $X_{i,j} \sim \exp\left(\lambda_j^i\right)$ then $X_j = \sum_{i=1}^{p} X_{i,j} \sim Er_p\left(\lambda_j^1, \ldots, \lambda_j^p\right)$ and the initial probability of starting in state k is π_k such that $\tau = \sum_{i=1}^{p} 1_{\{N=i\}}(X_i)$. In the special case where $\lambda_j^1 = \cdots = \lambda_j^p = \lambda_j$ the absorption time τ then follows a p-mixture of Erlang-p distributions with density given by

$$f_\tau(x) = \sum_{i=1}^{p} \pi_i f_i(x) = \sum_{i=1}^{p} \pi_i \frac{{\lambda_j}^p}{(p-1)!} x^{p-1} e^{-\lambda_j x} \tag{3.49}$$

In the general case let $\left\{\boldsymbol{T}_{ij}\right\}_{i,j\in\{1,\ldots p\}}$ be diagonal $p \times p$-matrices, $\boldsymbol{0}_{p\times p}$ be diagonal $p \times p$-matrices of zeros and $\boldsymbol{T}$ be a $p^2 \times p^2$-matrix. The sub-intensity matrix $\boldsymbol{T}$ can be decomposed into diagonal block matrices

$$\boldsymbol{T} = \begin{pmatrix} \boldsymbol{T}_{11} & \boldsymbol{T}_{12} & \cdots & \boldsymbol{0}_{p\times p} & \boldsymbol{0}_{p\times p} \\ 0 & \boldsymbol{T}_{22} & \cdots & \boldsymbol{0}_{p\times p} & \boldsymbol{0}_{p\times p} \\ \vdots & \vdots & \ddots & \boldsymbol{T}_{p-2,p-1} & \boldsymbol{0}_{p\times p} \\ \boldsymbol{0}_{p\times p} & \boldsymbol{0}_{p\times p} & \boldsymbol{0}_{p\times p} & \boldsymbol{T}_{p-1,p-1} & \boldsymbol{T}_{p-1,p} \\ \boldsymbol{0}_{p\times p} & \boldsymbol{0}_{p\times p} & \boldsymbol{0}_{p\times p} & \boldsymbol{0}_{p\times p} & \boldsymbol{T}_{p,p} \end{pmatrix} \tag{3.50}$$

where

$$\{\boldsymbol{T}_{ii}\}_{i\in\{1,\dots p\}} = \left\{\begin{pmatrix} -\lambda_1^i & 0 & \cdots & 0 & 0 \\ 0 & -\lambda_2^i & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & 0 & -\lambda_{p-1}^i & 0 \\ 0 & 0 & 0 & 0 & -\lambda_p^i \end{pmatrix}\right\}_{i\in\{1,\dots p\}} \tag{3.51}$$

and

$$\{\boldsymbol{T}_{i,i+1}\}_{i\in\{1,\dots p-1\}} = \left\{\begin{pmatrix} \lambda_1^i & 0 & \cdots & 0 & 0 \\ 0 & \lambda_2^i & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & 0 & \lambda_{p-1}^i & 0 \\ 0 & 0 & 0 & 0 & \lambda_p^i \end{pmatrix}\right\}_{i\in\{1,\dots p-1\}} \tag{3.52}$$

Furthermore let $\boldsymbol{t}_p$ be p-dimensional vectors, $\mathbf{0}_{p\times 1}$ and $\mathbf{1}_{p\times 1}$ be p-dimensional vectors of zeros and ones respectively and $\boldsymbol{t}$ be a p^2-dimensional vector. The exit vector is then given by

$$\boldsymbol{t} = \begin{pmatrix} \mathbf{0}_{p\times 1} \\ \vdots \\ \mathbf{0}_{p\times 1} \\ \boldsymbol{t}_p \end{pmatrix} \tag{3.53}$$

where

$$\boldsymbol{t}_p = \begin{pmatrix} \lambda_1 \\ \lambda_2 \\ \vdots \\ \lambda_p \end{pmatrix} \tag{3.54}$$

such that $\boldsymbol{T}_{ii} + \boldsymbol{T}_{i,i+1} = \mathbf{0}_{p\times p}$ for $i \in \{1, \dots p-1\}$ and $\boldsymbol{T}_{p,p}\mathbf{1}_{p\times 1} - \boldsymbol{t}_p = \mathbf{0}_{p\times 1}$. Finally let $\boldsymbol{\pi}_1$ be a p-dimensional row vector of initial probabilities, $\mathbf{0}_{1\times p}$ be p-dimensional row vectors of zeros and $\boldsymbol{\pi}$ be a p^2-dimensional row vector where

$$\boldsymbol{\pi} = \begin{pmatrix} \boldsymbol{\pi}_1 & \mathbf{0}_{1\times p} & \dots & \mathbf{0}_{1\times p} \end{pmatrix} \tag{3.55}$$

A generalized Erlang-k distributed random variable X then have phase-type representation $X \sim PH_{p^2}(\boldsymbol{\pi}, \boldsymbol{T})$ where $\boldsymbol{\pi}$ and $\boldsymbol{T}$ are given by and respectively.

3.4.2 Special cases of discrete phase-type distributions

Geometric distribution

Analogously to a the phase-type representation of an exponential distributed random variable, a geometric distributed random variables can be seen as an one order discrete phase-type distributed random variable with representation $\tau \sim DPH_1(1, -p)$ and exit probability p.

Negative binomial distribution

Since a negative binomial distributed random variable with success probability p and scale parameter n is a convolution of n geometric distributed random variables with success probability p, we can analogously to a continuous phase-type representation of an Erlang-n distributed use the discrete phase-type representation

$$\tau \sim DPH_n\left(\begin{pmatrix} 1 & 0 & \dots & 0 \end{pmatrix}, \begin{pmatrix} -p & p & \cdots & 0 & 0 \\ 0 & -p & \cdots & 0 & 0 \\ \vdots & \vdots & \ddots & 0 & 0 \\ 0 & 0 & 0 & -p & p \\ 0 & 0 & 0 & 0 & -p \end{pmatrix}\right) \tag{3.56}$$

and exit probability vector

$$\boldsymbol{t} = \begin{pmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ p \end{pmatrix}. \tag{3.57}$$

3.5 Bibliographic references

Phase-type distribution was first introduced by Neuts in [27]. The prior theory which builded the foundation for the phase-type distributions was however introduced by Erlang in [16] and Jensen in [24]. Other important contributions to the theory of phase-type distributions are, among other, [3], [29] and [28].

The derivation of density function, cumulative distribution function, nth order moments and Laplace transform of continuous phase-type distribution is partly based on [17] and [38]. Inspiration has also been found in [8]. Examples of phase-type distributions such as exponential distribution, geometric distribution, negative binomial distribution, Erlang distribution, coxian distribution can also be found in [10]. [33]. In [10] these aforementioned distributions are explored with their phases but their sub-intensity matrices, exit vectors and initial probability vectors are not derived.

Chapter 4

EM algorithm

The EM algorithm uses an iterative approach to derive maximum likelihood estimates of the parameters, which can be used when data is incomplete due to missing observations, censored observations etc.

The EM algorithm consists of an E-step (expectation step) and a M-step (maximization step). After initialization with some starting values, the EM algorithm alternates between, calculating the conditional expectation of the complete log likelihood function given the observed data in the expectation step, and finding the parameters which maximizes the calculated complete log likelihood function in the maximization step. Since maximum likelihood estimations require evaluation of the score vector and the Fisher information matrix, direct maximization of the likelihood is not always the most mathematically convenient approach. The EM algorithm can be an alternative to direct maximum likelihood, when the maximization problem cannot be solved analytically, or when the maximization problem is analytically tedious. Before describing its applications to phase-type distributions, we introduce the EM algorithm in general context. It will be shown that by using this iterative approach, the incomplete log likelihood function will increase for each iteration and thereby converge to local maximum, global maximum or saddle point.

Section 4.1 starts out by introducing the EM algorithm in general context. In the subsequent three sections estimation of phase-type distributions, by using the EM algorithm, are introduced for uncensored, censored and discrete data respectively. A large part of these sections contain derivation of the conditional sufficient statistics which will be written on an explicit form, expressed in terms of an exponential matrix. Section 4.4 include an approach for numeric calculation of exponential matrices. Last section in chapter 4 introduces approximation of continuous nonnegative distributions with continuous phase-type distributions, by using the EM algorithm.

4.1 Proof of the EM algorithm

Let $\boldsymbol{X}$ and $\boldsymbol{Y} = \boldsymbol{y}$ denote complete data and incomplete observed data respectively and assume that we have a many-to-one map $\psi(\boldsymbol{X}) = \boldsymbol{Y}$. Similarly, let L_c and ℓ_c denote the

complete data likelihood and log-likelihood functions respectively, and let L and ℓ denote the incomplete likelihood and log-likelihood functions. Furthermore, let $\boldsymbol{\theta}_n$ denote the estimated parameter vector of the nth iteration and let $\boldsymbol{\theta}$ denote the true parameter vector. By using a conditional argument, we can write the incomplete likelihood function as

$$\begin{aligned} L(\boldsymbol{\theta};\boldsymbol{y}) &= \int f_{Y|X=x}(\boldsymbol{y};\boldsymbol{\theta})\, f_X(\boldsymbol{x};\boldsymbol{\theta})\, d\boldsymbol{x} \\ &= \int f_{Y|X=x}(\boldsymbol{y};\boldsymbol{\theta}) \left(\frac{f_{X|Y=y}(\boldsymbol{x};\boldsymbol{\theta}_n)\, f_X(\boldsymbol{x};\boldsymbol{\theta})}{f_{X|Y=y}(\boldsymbol{x};\boldsymbol{\theta}_n)} \right) d\boldsymbol{x} \\ &= \int f_{X|Y=y}(\boldsymbol{x};\boldsymbol{\theta}_n) \left(\frac{f_{Y|X=x}(\boldsymbol{y};\boldsymbol{\theta})\, f_X(\boldsymbol{x};\boldsymbol{\theta})}{f_{X|Y=y}(\boldsymbol{x};\boldsymbol{\theta}_n)} \right) d\boldsymbol{x} \\ &= \mathbb{E}\left(\frac{f_{Y|X=x}(\boldsymbol{y};\boldsymbol{\theta})\, f_X(\boldsymbol{x};\boldsymbol{\theta})}{f_{X|Y=y}(\boldsymbol{x};\boldsymbol{\theta}_n)} \;\middle|\; \boldsymbol{Y} = \boldsymbol{y} \right). \end{aligned} \tag{4.1}$$

Since the logarithm is a concave function, it follows by Jensen's inequality that $\mathbb{E}(\log(\dots)) \geq \log(\mathbb{E}(\dots))$. By using Jensen's inequality, we can calculate the difference between the true incomplete log likelihood function and the incomplete log likelihood function in the nth iteration:

$$\begin{aligned} &\log L(\boldsymbol{\theta};\boldsymbol{y}) - \log L(\boldsymbol{\theta}_n;\boldsymbol{y}) \\ &= \log L(\boldsymbol{\theta};\boldsymbol{y}) - \log f_{\boldsymbol{Y}}(\boldsymbol{y};\boldsymbol{\theta}_n) \\ &= \log\left(\mathbb{E}\left(\frac{f_{Y|X=x}(\boldsymbol{y};\boldsymbol{\theta})\, f_X(\boldsymbol{x};\boldsymbol{\theta})}{f_{X|Y=y}(\boldsymbol{x};\boldsymbol{\theta}_n)} \;\middle|\; \boldsymbol{Y} = \boldsymbol{y} \right) \right) - \log f_{\boldsymbol{Y}}(\boldsymbol{y};\boldsymbol{\theta}_n) \\ &\geq \mathbb{E}\left(\log\left(\frac{f_{Y|X=x}(\boldsymbol{y};\boldsymbol{\theta})\, f_X(\boldsymbol{x};\boldsymbol{\theta})}{f_{X|Y=y}(\boldsymbol{x};\boldsymbol{\theta}_n)} \right) \;\middle|\; \boldsymbol{Y} = \boldsymbol{y} \right) - \log f_{\boldsymbol{Y}}(\boldsymbol{y};\boldsymbol{\theta}_n) \\ &= \int f_{X|Y=y}(\boldsymbol{x};\boldsymbol{\theta}_n) \log\left(\frac{f_{Y|X=x}(\boldsymbol{y};\boldsymbol{\theta})\, f_X(\boldsymbol{x};\boldsymbol{\theta})}{f_{X|Y=y}(\boldsymbol{x};\boldsymbol{\theta}_n)} \right) d\boldsymbol{x} - \log f_{\boldsymbol{Y}}(\boldsymbol{y};\boldsymbol{\theta}_n) \\ &= \int f_{X|Y=y}(\boldsymbol{x};\boldsymbol{\theta}_n) \log\left(\frac{f_{Y|X=x}(\boldsymbol{y};\boldsymbol{\theta})\, f_X(\boldsymbol{x};\boldsymbol{\theta})}{f_{X|Y=y}(\boldsymbol{x};\boldsymbol{\theta}_n)\, f_{\boldsymbol{Y}}(\boldsymbol{y};\boldsymbol{\theta}_n)} \right) d\boldsymbol{x} \end{aligned} \tag{4.2}$$

Define $\Delta(\boldsymbol{\theta},\boldsymbol{\theta}_n;\boldsymbol{y})$ as the right hand side above. Furthermore, define

$$\begin{aligned} g(\boldsymbol{\theta},\ \boldsymbol{\theta}_n;\boldsymbol{y}) &:= \log L(\boldsymbol{\theta}_n;\boldsymbol{y}) + \Delta(\boldsymbol{\theta},\ \boldsymbol{\theta}_n;\boldsymbol{y}) \\ &\leq \log L(\boldsymbol{\theta};\boldsymbol{y}), \end{aligned} \tag{4.3}$$

where the inequality follows from 4.2. Applying the definition of $\Delta(\boldsymbol{\theta},\boldsymbol{\theta}_n;\boldsymbol{y})$, we see from 4.3 that

$$
\begin{aligned}
g\left(\boldsymbol{\theta}_n, \boldsymbol{\theta}_n; \boldsymbol{y}\right) &= \log L\left(\boldsymbol{\theta}_n; \boldsymbol{y}\right) \; + \Delta\left(\boldsymbol{\theta}_n, \; \boldsymbol{\theta}_n; \boldsymbol{y}\right) \\
&= \log L\left(\boldsymbol{\theta}_n; \boldsymbol{y}\right) \; + \int f_{X|Y=y}\left(\boldsymbol{x}; \boldsymbol{\theta}_n\right) \log\left(\frac{f_{Y|X=x}\left(\boldsymbol{y}; \boldsymbol{\theta}\right) f_X\left(\boldsymbol{x}; \boldsymbol{\theta}_n\right)}{f_{X|Y=y}\left(\boldsymbol{x}; \boldsymbol{\theta}_n\right) f_Y\left(\boldsymbol{y}; \boldsymbol{\theta}_n\right)}\right) d\boldsymbol{x} \\
&= \log L\left(\boldsymbol{\theta}_n; \boldsymbol{y}\right) \; + \int f_{X|Y=y}\left(\boldsymbol{x}; \boldsymbol{\theta}_n\right) \log\left(\frac{f_{X,Y}\left(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}_n\right)}{f_{X,Y}\left(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}_n\right)}\right) d\boldsymbol{x} \\
&= \log L\left(\boldsymbol{\theta}_n; \boldsymbol{y}\right) \; + \int f_{X|Y=y}\left(\boldsymbol{x}; \boldsymbol{\theta}_n\right) \log\left(1\right) d\boldsymbol{x} \\
&= \log L\left(\boldsymbol{\theta}_n; \boldsymbol{y}\right) .
\end{aligned}
\tag{4.4}
$$

From 4.4, we conclude that $g\left(\boldsymbol{\theta}_n, \boldsymbol{\theta}_n; \boldsymbol{y}\right)$ equals the complete log likelihood function in the nth iteration. According to 4.3, $g\left(\boldsymbol{\theta}_n, \boldsymbol{\theta}_n; \boldsymbol{y}\right)$ has an upper bound, given by the incomplete log likelihood function evaluated in the true parameter values $\log L\left(\boldsymbol{\theta}; \boldsymbol{y}\right)$. Now, initiate with an arbitrary $\boldsymbol{\theta}_0$ and define

$$
\begin{aligned}
\boldsymbol{\theta}_{n+1} &\coloneqq \operatorname{argmax}_{\boldsymbol{\theta}} g\left(\boldsymbol{\theta}, \; \boldsymbol{\theta}_n; \boldsymbol{y}\right) \\
&= \operatorname{argmax}_{\boldsymbol{\theta}} \left(\log L\left(\boldsymbol{\theta}_n; \boldsymbol{y}\right) \; + \Delta\left(\boldsymbol{\theta}, \; \boldsymbol{\theta}_n; \boldsymbol{y}\right)\right) \\
&= \operatorname{argmax}_{\boldsymbol{\theta}} \Delta\left(\boldsymbol{\theta}, \; \boldsymbol{\theta}_n; \boldsymbol{y}\right) \\
&= \operatorname{argmax}_{\boldsymbol{\theta}} \int f_{X|Y=y}\left(\boldsymbol{x}; \boldsymbol{\theta}_n\right) \log\left(\frac{f_{Y|X=x}\left(\boldsymbol{y}; \boldsymbol{\theta}\right) f_X\left(\boldsymbol{x}; \boldsymbol{\theta}\right)}{f_{X|Y=y}\left(\boldsymbol{x}; \boldsymbol{\theta}_n\right) f_Y\left(\boldsymbol{y}; \boldsymbol{\theta}_n\right)}\right) d\boldsymbol{x} \\
&= \operatorname{argmax}_{\boldsymbol{\theta}} \int f_{X|Y=y}\left(\boldsymbol{x}; \boldsymbol{\theta}_n\right) \log\left(f_{X|Y=y}\left(\boldsymbol{y}; \boldsymbol{\theta}\right) f_X\left(\boldsymbol{x}; \boldsymbol{\theta}\right)\right) d\boldsymbol{x} .
\end{aligned}
\tag{4.5}
$$

Second and fourth equalities in 4.5 follows directly from the definitions given above. Third equality in 4.5 follows from the fact that the first term is constant with respect to $\boldsymbol{\theta}$. Likewise the last equality in 4.5 follows from the fact that, the denominator is constant with respect to $\boldsymbol{\theta}$. Using that

$$
\begin{aligned}
&\int f_{X|Y=y}\left(\boldsymbol{x}; \boldsymbol{\theta}_n\right) \log\left(f_{Y|X=x}\left(\boldsymbol{y}; \boldsymbol{\theta}\right) f_X\left(\boldsymbol{x}; \boldsymbol{\theta}\right)\right) d\boldsymbol{x} \\
&= \int f_{X|Y=y}\left(\boldsymbol{x}; \boldsymbol{\theta}_n\right) \log\left(f_{X,Y}\left(\boldsymbol{x}, \boldsymbol{y}; \boldsymbol{\theta}\right)\right) d\boldsymbol{x} \\
&\propto \int f_{X|Y=y}\left(\boldsymbol{x}; \boldsymbol{\theta}_n\right) \log\left(f_X\left(\boldsymbol{x}; \boldsymbol{\theta}\right)\right) d\boldsymbol{x} \\
&= \mathbb{E}_{\boldsymbol{\theta}_n}\left(\ell_c\left(\boldsymbol{X}; \boldsymbol{\theta}\right) | \boldsymbol{Y} = \boldsymbol{y}\right) \\
&= \log L\left(\boldsymbol{\theta}_n; \boldsymbol{y}\right) ,
\end{aligned}
\tag{4.6}
$$

where the proportionality follows from the fact that $\boldsymbol{Y}$ is a function of $\boldsymbol{X}$, it follows that

$$
\boldsymbol{\theta}_{n+1} = \operatorname{argmax}_{\boldsymbol{\theta}} \log L\left(\boldsymbol{\theta}_n; \boldsymbol{y}\right) . \tag{4.7}
$$

Hence,

$$\log L(\boldsymbol{\theta}_{n+1}; \boldsymbol{y}) \geq \log L(\boldsymbol{\theta}_n; \boldsymbol{y}) \tag{4.8}$$

As seen, the likelihood function will increase for each iteration. Since the likelihood sequence $\{\log L(\boldsymbol{\theta}_n; \boldsymbol{y})\}_{n\in\mathbb{N}}$ is bounded by $L(\boldsymbol{\theta}; \boldsymbol{y})$, it will converge to a local, possibly global, maximum or saddle point.

The algorithm can be summarized the following way:

0. Initialize with some arbitrary parameter vector $\boldsymbol{\theta}_0$ and let $n = 0$.
1. Calculate $h : \boldsymbol{\theta} \to \ell(\boldsymbol{\theta}_n; \boldsymbol{y}) = \mathbb{E}_{\boldsymbol{\theta}_n}(\ell_c(\boldsymbol{\theta}; X) \mid \boldsymbol{Y} = \boldsymbol{y})$
2. Calculate $\boldsymbol{\theta}_{n+1} = \text{argmax}_{\boldsymbol{\theta}} h(\boldsymbol{\theta})$
3. Set $n = n + 1$ and GOTO 1

4.2 EM algorithm for continuous phase-type distributions

In section 3.3 the complete log likelihood function for phase-type distributions was calculated, see 3.30. Since the complete log likelihood function is linear in the sufficient statistics, it suffices to calculate the conditional expectations of the sufficient statistics. It is clear that the conditional expectation of any of these sufficient statistics given is the sum of the conditional expectations given $Y_i = y_i$, for $i = 1, ..., n$, where n is the number of observations. Hence, we only need to calculate the conditional expectations of the sufficient statistics, for a single realization, since we can just add them up afterwards. Considering a single data point $Y = y$, the underlying Markov chain will process to the absorbing state at the time y. The conditional expectation of the sufficient statistics from only single realizations are calculated below. In section 3.2 the density of continuous phase-type distributions was derived. Recall the formula:

$$f(y) = P(Y \in dy) = \boldsymbol{\pi} e^{\boldsymbol{T}y} \boldsymbol{t} \tag{4.9}$$

Consequently,

$$\mathbb{P}(Y \in dy \mid X_0 = i) = \boldsymbol{e}_i' e^{\boldsymbol{T}y} \boldsymbol{t}, \tag{4.10}$$

since this just corresponds to a phase-type distributed random variable with initial probability vector $\boldsymbol{e}_i'$. Next recall that

$$F(s) = 1 - \boldsymbol{\pi} e^{\boldsymbol{T}s} \boldsymbol{e}, \tag{4.11}$$

and by the same argument

$$\mathbb{P}(Y \leq s | X_0 = i) = 1 - \boldsymbol{e}_i' e^{\boldsymbol{T}s} \boldsymbol{e}. \tag{4.12}$$

These formulas will be used frequently in the following two sections.

Recall that B_i is the number of initiations of the underlying Markov chain in state i. Considering a single realization, B_i is then an indicator function which activates, if the underlying Markov chain initiates in state i:

$$B_i = 1\{X_0 = i\} \tag{4.13}$$

By using 4.12, we calculate $\mathbb{E}(B_i \mid Y = y)$ as

$$\begin{aligned}
\mathbb{E}(B_i \mid Y = y) &= \mathbb{E}(1\{X_0 = i\} \mid Y = y) \\
&= \mathbb{P}(X_u = i \mid Y = y) \\
&= \frac{\mathbb{P}(X_0 = i, Y \in dy)}{\mathbb{P}(Y \in dy)} \\
&= \frac{\mathbb{P}(X_0 = i)\mathbb{P}(Y \in dy | X_0 = i)}{\mathbb{P}(Y \in dy)} \\
&= \frac{\boldsymbol{\pi}_i \boldsymbol{e}_i' e^{\boldsymbol{T}y}\boldsymbol{t}}{\boldsymbol{\pi} e^{\boldsymbol{T}y}\boldsymbol{t}}.
\end{aligned} \tag{4.14}$$

Next, recall that Z_i is the total amount of time spent in state i. Again, since we are considering a single realization, we can write Z_i as

$$Z_i = \int_0^\tau 1\{X_u = i\}\, du. \tag{4.15}$$

From 4.12 we get

$$\mathbb{P}(Y \in dy \mid X_u = i) = \mathbb{P}(Y \in d(y-u) \mid X_0 = i) = \boldsymbol{e}_i' e^{\boldsymbol{T}(y-u)}\boldsymbol{t}. \tag{4.16}$$

By using 4.12 and 4.9, we calculate $\mathbb{E}(Z_i \mid Y = y)$ as

$$\begin{aligned}
\mathbb{E}(Z_i | Y = y) &= \mathbb{E}\left(\int_0^y 1\{X_u = i\}\, du \;\middle|\; Y = y \right) \\
&= \int_0^y \mathbb{P}(X_u = i | Y = y) du \\
&= \int_0^y \frac{\mathbb{P}(Y \in dy | X_u = i)\mathbb{P}(X_u = i)}{\mathbb{P}(Y \in dy)} du \\
&= \int_0^y \frac{\boldsymbol{e}_i' e^{\boldsymbol{T}(y-u)}\boldsymbol{t}\boldsymbol{\pi} e^{\boldsymbol{T}u}\boldsymbol{e}_i}{\boldsymbol{\pi} e^{\boldsymbol{T}y}\boldsymbol{t}} du.
\end{aligned} \tag{4.17}$$

N_i is the number of jumps from state i to the absorbing state, which for a single data point amounts to an indicator function. This is the equivalent to the underlying Markov chain being in state i, just before absorption a time $y-$. Consequently, we have the formula

$$N_i = 1\left\{X_{y-} = i\right\}. \tag{4.18}$$

Analogously to the other calculations we get

$$\begin{aligned}\mathbb{E}(N_i|Y=y) &= \mathbb{P}(X_{y-}=i|Y=y)\\ &= \frac{\mathbb{P}(Y\in dy|X_{y-}=i)\mathbb{P}(X_{y-}=i)}{\mathbb{P}(Y\in dy)}\\ &= \frac{\boldsymbol{\pi}e^{\boldsymbol{T}y}\boldsymbol{e}_i t_i}{\boldsymbol{\pi}e^{\boldsymbol{T}y}\boldsymbol{t}}.\end{aligned} \tag{4.19}$$

N_{ij} is the number of jumps from state i to state j. N_{ij} is more tedious to calculate compared to the other sufficient statistics. Regarding N_{ij}, by the law of total probability, and the fact that N_{ij} and $\{Y\in[y,y+dy)\}$ are conditionally independent given X_{y-}, we have that

$$\begin{aligned}&\mathbb{E}\left(N_{ij}1\left\{Y\in[y,y+dy)\right\}\right)\\ &=\sum_\ell \mathbb{E}\left(N_{ij}1\left\{X_{y-}=\ell, Y\in[y,y+dy)\right\}\right)\\ &=\sum_\ell \mathbb{E}\left(N_{ij}\,\middle|\, X_{y-}=\ell, Y\in[y,y+dy)\right)\mathbb{P}\left(X_{y-}=\ell, Y\in[y,y+dy)\right)\\ &=\sum_\ell \mathbb{E}\left(N_{ij}\,\middle|\, X_{y-}=\ell, Y\in[y,y+dy)\right)\underbrace{\mathbb{P}\left(Y\in[y,y+dy)\,\middle|\, X_{y-}=\ell\right)}_{t_\ell dy}\mathbb{P}\left(X_{y-}=\ell\right)\\ &=\sum_\ell \mathbb{E}\left(N_{ij}\,\middle|\, X_{y-}=\ell\right)\mathbb{P}\left(X_{y-}=\ell\right)t_\ell dy\\ &=\sum_\ell \mathbb{E}\left(N_{ij}1\left\{X_{y-}=\ell\right\}\right)t_\ell dy.\end{aligned} \tag{4.20}$$

Before we continue, notice that $\boldsymbol{e}_i\,\boldsymbol{e}_j'$ is a matrix which have the value one in row i and column j and zeroes in all the remaining entries. Consider the expression

$$\begin{aligned}&\mathbb{E}\left(N_{ij}1\left\{X_{y-}=\ell\right\}\,\middle|\, X_0=k\right)\\ &=\boldsymbol{e}_k'\int_0^y e^{\boldsymbol{T}s}t_{ij}\boldsymbol{e}_i\,\boldsymbol{e}_j'e^{\boldsymbol{T}(y-s)}ds\boldsymbol{e}_\ell\\ &=t_{ij}\boldsymbol{e}_j'\int_0^y e^{\boldsymbol{T}(y-s)}\boldsymbol{e}_\ell\boldsymbol{e}_k'e^{\boldsymbol{T}s}ds\boldsymbol{e}_i.\end{aligned} \tag{4.21}$$

The intuition behind first equality in 4.22 is this: The expected number of jumps in $[s, s+ds)$ can, at most, be one such that the number of jumps can be written as an indicator function $1\{X_s=i, X_{s+ds}=j\}$. Taking the expectations, then gives the probability $\mathbb{P}\{X_s=i, X_{s+ds}=j\}$ which, per definition, is $t_{ij}ds$. The probability of being in state i at time s, given initiation in state k, is $\boldsymbol{e}_k'e^{\boldsymbol{T}s}\boldsymbol{e}_i$. Reaching state l at time $y-$, from state j, at

the time $s + sd$, has the same probability to being in state l at time $y - s$, given initiation in state j. This is due to the Markov property. This event has the probability

$$\boldsymbol{e}_j' e^{\boldsymbol{T}((y-)-(s+ds))}\boldsymbol{e}_\ell = \boldsymbol{e}_j' e^{\boldsymbol{T}(y-s)}\boldsymbol{e}_\ell, \tag{4.22}$$

due to continuity. To summarize, the probability of jumping from state i to state j given initiation in state k, and jumping to absorption from state l, is then $\boldsymbol{e}_k' e^{\boldsymbol{T}s}\boldsymbol{e}_i \boldsymbol{e}_j' e^{\boldsymbol{T}(y-s)} ds \boldsymbol{e}_\ell$. Summing over all s in $(0, y)$ then gives 4.22. Using 4.20 and 4.22, we finally derive the conditional expectation of N_{ij}:

$$\begin{aligned}
\mathbb{E}\left(N_{ij} \,\middle|\, Y = y\right) &= \frac{\mathbb{E}(N_{ij} 1\{Y \in dy\})}{f_Y(y)} \\
&= \sum_\ell \frac{\mathbb{E}\left(N_{ij} 1\left\{X_{y-} = \ell\right\} t_\ell\right)}{f_Y(y)} \\
&= \sum_{\ell,\, k} \pi_k \frac{\mathbb{E}\left(N_{ij} 1\left\{X_{y-} = \ell\right\} \,\middle|\, X_0 = k\right)}{f_Y(y)} t_\ell \\
&= \sum_{\ell,\, k} \pi_k \frac{t_{ij}\boldsymbol{e}_j' \int_0^y e^{\boldsymbol{T}(y-s)}\boldsymbol{e}_\ell \boldsymbol{e}_k' e^{\boldsymbol{T}s} ds \boldsymbol{e}_i}{\boldsymbol{\pi} e^{\boldsymbol{T}y}\boldsymbol{t}} t_\ell
\end{aligned} \tag{4.23}$$

We have now calculated the conditional expectations of the sufficient statistics in a single data point. However, regarding N_{ij} and Z_i, the expression includes an integral which can be analytically intractable to solve. However, as shown below, it is possible to derive explicit expressions, written in terms of matrix exponentials. Consider the expression

$$\boldsymbol{J}(y; \boldsymbol{\pi}, \boldsymbol{T}) = \int_0^y e^{\boldsymbol{T}(y-u)}\boldsymbol{t}\boldsymbol{\pi} e^{\boldsymbol{T}u} du. \tag{4.24}$$

Calculation of the derivative gives

$$\begin{aligned}
&\frac{\boldsymbol{J}(y+h; \boldsymbol{\pi}, \boldsymbol{T}) - \boldsymbol{J}(y; \boldsymbol{\pi}, \boldsymbol{T})}{h} \\
&= \frac{1}{h}\left(\int_0^{y+h} e^{\boldsymbol{T}(y+h-u)}\boldsymbol{t}\boldsymbol{\pi} e^{\boldsymbol{T}u} du - \int_0^y e^{\boldsymbol{T}(y-u)}\boldsymbol{t}\boldsymbol{\pi} e^{\boldsymbol{T}u} du\right) \\
&= \int_0^y \frac{1}{h}\left(e^{\boldsymbol{T}(y+h-u)} - e^{\boldsymbol{T}(y-u)}\right)\boldsymbol{t}\boldsymbol{\pi} e^{\boldsymbol{T}u} du + \frac{1}{h}\int_y^{y+h} e^{\boldsymbol{T}(y-u)}\boldsymbol{t}\boldsymbol{\pi} e^{\boldsymbol{T}u} du \\
&\to \int_0^y \frac{\partial e^{\boldsymbol{T}(y-u)}}{\partial y}\boldsymbol{t}\boldsymbol{\pi} e^{\boldsymbol{T}u} du + \frac{\partial}{\partial y}\left(\int_0^y e^{\boldsymbol{T}(y-u)}\boldsymbol{t}\boldsymbol{\pi} e^{\boldsymbol{T}u} du\right) \\
&= \int_0^y \boldsymbol{T} e^{\boldsymbol{T}(y-u)}\boldsymbol{t}\boldsymbol{\pi} e^{\boldsymbol{T}u} du + e^{\boldsymbol{T}(y-y)}\boldsymbol{t}\boldsymbol{\pi} e^{\boldsymbol{T}y} \\
&= \int_0^y \boldsymbol{T} e^{\boldsymbol{T}(y-u)}\boldsymbol{t}\boldsymbol{\pi} e^{\boldsymbol{T}u} du + \boldsymbol{t}\boldsymbol{\pi} e^{\boldsymbol{T}y}.
\end{aligned} \tag{4.25}$$

Hence,

$$\frac{\partial \boldsymbol{J}\left(y;\boldsymbol{\pi},\boldsymbol{T}\right)}{\partial y} = \int_0^y \boldsymbol{T}e^{\boldsymbol{T}(y-u)}\boldsymbol{t}\boldsymbol{\pi}e^{\boldsymbol{T}u}du + \boldsymbol{t}\boldsymbol{\pi}e^{\boldsymbol{T}y}. \tag{4.26}$$

Notice that matrix differentiation happens entry-wise. Consequently, we have

$$\begin{aligned}
&\frac{\partial}{\partial y}\left(\begin{pmatrix} e^{\boldsymbol{T}y} & \boldsymbol{J}\left(y;\boldsymbol{\pi},\boldsymbol{T}\right) \\ 0 & e^{\boldsymbol{T}y} \end{pmatrix}\right) \\
&= \begin{pmatrix} \boldsymbol{T}e^{\boldsymbol{T}y} & \int_0^y \boldsymbol{T}e^{\boldsymbol{T}(y-u)}\boldsymbol{t}\boldsymbol{\pi}e^{\boldsymbol{T}u}du + \boldsymbol{t}\boldsymbol{\pi}e^{\boldsymbol{T}y} \\ 0 & \boldsymbol{T}e^{\boldsymbol{T}y} \end{pmatrix} \\
&= \begin{pmatrix} \boldsymbol{T} & \boldsymbol{t}\boldsymbol{\pi} \\ 0 & \boldsymbol{T} \end{pmatrix}\begin{pmatrix} e^{\boldsymbol{T}y} & \int_0^y e^{\boldsymbol{T}(y-u)}\boldsymbol{t}\boldsymbol{\pi}e^{\boldsymbol{T}u}du \\ 0 & e^{\boldsymbol{T}y} \end{pmatrix} \\
&= \begin{pmatrix} \boldsymbol{T} & \boldsymbol{t}\boldsymbol{\pi} \\ 0 & \boldsymbol{T} \end{pmatrix}\begin{pmatrix} e^{\boldsymbol{T}y} & \boldsymbol{J}\left(y;\boldsymbol{\pi},\boldsymbol{T}\right) \\ 0 & e^{\boldsymbol{T}y} \end{pmatrix}.
\end{aligned} \tag{4.27}$$

Hence, we have a matrix differential equation with boundary condition

$$\begin{pmatrix} e^{\boldsymbol{T}0} & \boldsymbol{J}\left(0;\boldsymbol{\pi},\boldsymbol{T}\right) \\ 0 & e^{\boldsymbol{T}0} \end{pmatrix} = \boldsymbol{I}, \tag{4.28}$$

from which we obtain the solution

$$\begin{aligned}
\exp\left(\begin{pmatrix} \boldsymbol{T} & \boldsymbol{t}\boldsymbol{\pi} \\ 0 & \boldsymbol{T} \end{pmatrix}y\right) &= \begin{pmatrix} e^{\boldsymbol{T}y} & \boldsymbol{J}\left(y;\boldsymbol{\pi},\boldsymbol{T}\right) \\ 0 & e^{\boldsymbol{T}y} \end{pmatrix} \\
&= \begin{pmatrix} e^{\boldsymbol{T}y} & \int_0^y e^{\boldsymbol{T}(y-u)}\boldsymbol{t}\boldsymbol{\pi}e^{\boldsymbol{T}u}du \\ 0 & e^{\boldsymbol{T}y} \end{pmatrix}.
\end{aligned} \tag{4.29}$$

By using these types of matrix exponentials of block-partioned matrices, we can write an explicit expression for the sufficient statistics:

$$\exp\left(\begin{pmatrix} \boldsymbol{T} & \boldsymbol{e}\boldsymbol{\pi} \\ 0 & \boldsymbol{T} \end{pmatrix}y\right) = \begin{pmatrix} e^{\boldsymbol{T}y} & \int_0^y e^{\boldsymbol{T}(y-u)}\boldsymbol{e}\boldsymbol{\pi}e^{\boldsymbol{T}u}du \\ 0 & e^{\boldsymbol{T}y} \end{pmatrix} \tag{4.30}$$

Using this formula on 4.23 and 4.17, we derive explicit expressions of the sufficient statistics written in terms of matrix exponentials:

$$\mathbb{E}\left(Z_i \,\middle|\, Y = y\right) = \frac{\boldsymbol{J}(y;\boldsymbol{\pi},\boldsymbol{T})_{ii}}{\boldsymbol{\pi}e^{\boldsymbol{T}y}\boldsymbol{t}} \tag{4.31}$$

$$\mathbb{E}\left(N_{ij} \,\middle|\, Y = y\right) = t_{ij}\frac{\boldsymbol{J}(y;\boldsymbol{\pi},\boldsymbol{T})_{ii}}{\boldsymbol{\pi}e^{\boldsymbol{T}y}\boldsymbol{t}} \tag{4.32}$$

Numeric calculations of these matrix exponentials, can for instance be done by using the Runge Kutta approach, which will be described in section 4.5.

Regarding the maximization step we recall from section 3.3 that the parameter values which maximizes the complete log likelihood function are given by 3.35. Taking the conditional expectation of these values corresponds to the maximization step in the EM algorithm.
All of this can be summarized in EM algorithm for phase-type distributions below:

0. Initialize with som arbitrary $(\boldsymbol{\pi}_0, \boldsymbol{T}_0)$

1. Calculate

$$\mathbb{E}\left(B_i \mid \boldsymbol{Y} = \boldsymbol{y}\right) = \sum_{i=1}^{N} \frac{\boldsymbol{\pi}_i \boldsymbol{e}_i' e^{\boldsymbol{T}y_i} \boldsymbol{t}}{\boldsymbol{\pi} e^{\boldsymbol{T}y_i} \boldsymbol{t}} \tag{4.33}$$

$$\mathbb{E}\left(Z_i \mid \boldsymbol{Y} = \boldsymbol{y}\right) = \sum_{i=1}^{N} \frac{\boldsymbol{J}(y_k; \boldsymbol{\pi}, \boldsymbol{T})_{ii}}{\boldsymbol{\pi} e^{\boldsymbol{T}y_k} \boldsymbol{t}}, \tag{4.34}$$

$$\mathbb{E}\left(N_{ij} \;\middle|\; \boldsymbol{Y} = \boldsymbol{y}\right) = \sum_{i=1}^{N} t_{ij} \frac{\boldsymbol{J}(y_k; \boldsymbol{\pi}, \boldsymbol{T})_{ji}}{\boldsymbol{\pi} e^{\boldsymbol{T}y_k} \boldsymbol{t}}, \tag{4.35}$$

$$\mathbb{E}\left(N_i \mid \boldsymbol{Y} = \boldsymbol{y}\right) = \sum_{i=1}^{N} \frac{\boldsymbol{\pi} e^{\boldsymbol{T}y_k} \boldsymbol{e}_i t_i}{\boldsymbol{\pi} e^{\boldsymbol{T}y_k} \boldsymbol{t}} \tag{4.36}$$

   where $\boldsymbol{J}\left(y_k; \boldsymbol{\pi}, \boldsymbol{T}\right) = \int_0^y e^{\boldsymbol{T}(y-u)} \boldsymbol{t} \boldsymbol{\pi} e^{\boldsymbol{T}u} du$.

2. Calculate

$$\hat{t}_{ij} = \frac{\mathbb{E}\left(N_{ij} \;\middle|\; \boldsymbol{Y} = \boldsymbol{y}\right)}{\mathbb{E}\left(Z_i \mid \boldsymbol{Y} = \boldsymbol{y}\right)}, \tag{4.37}$$

$$\hat{t}_i = \frac{\mathbb{E}\left(N_i \mid \boldsymbol{Y} = \boldsymbol{y}\right)}{\mathbb{E}\left(Z_i \mid \boldsymbol{Y} = \boldsymbol{y}\right)}, \tag{4.38}$$

$$\widehat{\pi}_i = \frac{\mathbb{E}\left(B_i \mid \boldsymbol{Y} = \boldsymbol{y}\right)}{N} \tag{4.39}$$

3. Set $\boldsymbol{\pi} := \hat{\boldsymbol{\pi}}, \boldsymbol{T} := \hat{\boldsymbol{T}}, \boldsymbol{t} := \hat{\boldsymbol{t}}, n := n + 1$ and GOTO 1

If we initialize some entries in the initial probability vector or sub-intensity matrix with 0, the entries considered will remain zero. That allows for estimating subclasses of phase-type distributions such as Erlang distributions, hyperexponential distributions and the other sub-classes, presented in section 3.4.

With a slightly modification, this EM algorithm is also applicable when a phase-type distribution has an atom at zero. All we have to do is setting π_{p+1} equal to the proportion of observations in the dataset, which contains the value zero. Afterwards, we eliminate the zeros from the data and use the EM algorithm to fit the phase-type distribution to the remaining data.

4.3 EM algorithm for interval censored continuous phase-type distributed data

The EM algorithm can be applied on censored data by modifying the expectation step. Analogous to the last section, the conditional expectation of the sufficient statistics are calculated for a single realization. We calculate the conditional expectation for each censored data point, using a censored formula which will be derived in this section. Phase-type distributions can then be fitted on a sample, mixed with censored and uncensored observations, by combining the use of the censored formula and the uncensored formula. If the data point is uncensored, the uncensored formula is applied and vice versa. Finally, the conditional expectations for the sufficient statistics in each data point is summed up over the formula for individual data points.

We can calculate the conditional expectation for N_{ij} as

$$\begin{aligned}
&\mathbb{E}\left(N_{ij}(t) \;\middle|\; Y \in (s,t]\right) \\
&= \frac{\mathbb{E}\left(N_{ij}(t)\, 1_{(s,t]}(Y)\right)}{P(Y \in (s,t])} \\
&= \frac{\mathbb{E}\left(N_{ij}(t)\right) - \mathbb{E}\left(N_{ij}(t)\, 1\{Y \le s\}\right) - \mathbb{E}\left(N_{ij}(t)\, 1\{Y > t\}\right)}{\boldsymbol{\pi} e^{\boldsymbol{T}s}\boldsymbol{e} - \boldsymbol{\pi} e^{\boldsymbol{T}t}\boldsymbol{e}}.
\end{aligned} \tag{4.40}$$

This formula can be used for the other sufficient statistics as well, just by replacing N_{ij} with the corresponding statistic. However, the formula simplifies for some of the sufficient statistics, as will be shown later in this section.

First the conditional expectation of N_{ij} is derived. Consider the expression

$$\begin{aligned}
&\mathbb{E}\left(N_{ij}(t+dt) - N_{ij}(t)\right) \\
&= \mathbb{E}\left(\mathbb{E}\left(N_{ij}(t+dt) - N_{ij}(t) \;\middle|\; X_t\right)\right) \\
&= \sum_{k\in\mathbb{N}} \mathbb{E}\left(N_{ij}(t+dt) - N_{ij}(t) \;\middle|\; X_t = k\right)\mathbb{P}(X_t = k).
\end{aligned} \tag{4.41}$$

Since we only consider one realization, on an infinitesimal interval in 4.41, there can only be a jump from state i to state j during $[t, t+dt)$ if the process is in state i at the beginning of the time interval. As a result of that, all the other terms on the right hand side of 4.41 cancels out. Furthermore, there can at most be one jump. Therefore 4.41 can be expressed in terms of an indicator function which activates when the underlying Markov process jumps from state i to state j, during this infinitesimal time interval. Hence,

$$\begin{aligned}
\mathbb{E}\left(N_{ij}(t+dt) - N_{ij}(t)\right) &= \mathbb{E}\left(N_{ij}(t+dt) - N_{ij}(t) \;\middle|\; X_t = i\right)\mathbb{P}(X_t = i) \\
&= \mathbb{E}\left(1\{X_{t+dt} = j \mid X_t = i\}\right)\mathbb{P}(X_t = i) \\
&= \mathbb{P}(X_{t+dt} = j \mid X_t = i)\,\mathbb{P}(X_t = i) \\
&= t_{ij}dt\mathbb{P}(X_t = i).
\end{aligned} \tag{4.42}$$

Taking the derivative with respect to t gives

$$\frac{\partial \mathbb{E}\left(N_{ij}\left(t+dt\right)-N_{ij}\left(t\right)\right)}{\partial t}=t_{ij}\mathbb{P}(X_t=i). \tag{4.43}$$

By using $\mathbb{E}\left(N_{ij}\left(0\right)\right)=0$ and 4.43, we then have

$$\begin{aligned}\mathbb{E}\left(N_{ij}\left(t\right)\right) &= \int_0^t t_{ij}\mathbb{P}\left(X_s=i\right)ds \\ &= \int_0^t t_{ij}\boldsymbol{\pi}e^{\boldsymbol{T}s}\boldsymbol{e}_i ds.\end{aligned} \tag{4.44}$$

Using the result in the E step from last section, the uncensored expression in 4.40 can be calculated as

$$\begin{aligned}\mathbb{E}\left(N_{ij}\left(t\right)1\left\{Y\le t\right\}\right) &= \int_0^t \mathbb{E}\left(N_{ij}\left(t\right)\,\middle|\, Y=y\right)f_Y\left(y\right)dy \\ &= \int_0^t\int_0^y \frac{\boldsymbol{e}_j'e^{\boldsymbol{T}(y-u)}\boldsymbol{t}\boldsymbol{\pi}e^{\boldsymbol{T}u}\boldsymbol{e}_i}{\boldsymbol{\pi}e^{\boldsymbol{T}y}\boldsymbol{t}}t_{ij}\boldsymbol{\pi}e^{\boldsymbol{T}y}\boldsymbol{t}dudy \\ &= \int_0^t\int_0^y \boldsymbol{e}_j'e^{\boldsymbol{T}(y-u)}\boldsymbol{t}\boldsymbol{\pi}e^{\boldsymbol{T}u}\boldsymbol{e}_i t_{ij}dudy \\ &= \int_0^t\int_u^t \boldsymbol{e}_j'e^{\boldsymbol{T}(y-u)}\boldsymbol{t}\boldsymbol{\pi}e^{\boldsymbol{T}u}\boldsymbol{e}_i t_{ij}dydu \\ &= \int_0^t \boldsymbol{e}_j'\left[e^{\boldsymbol{T}y}\boldsymbol{T}^{-1}\right]_0^{t-u}\boldsymbol{t}\boldsymbol{\pi}e^{\boldsymbol{T}u}\boldsymbol{e}_i t_{ij}du \\ &= \int_0^t \boldsymbol{e}_j'\left(e^{\boldsymbol{T}(t-u)}\boldsymbol{T}^{-1}-\boldsymbol{T}^{-1}\right)\boldsymbol{t}\boldsymbol{\pi}e^{\boldsymbol{T}u}\boldsymbol{e}_i t_{ij}du \\ &= \int_0^t \boldsymbol{e}_j'\left(e^{\boldsymbol{T}(t-u)}\boldsymbol{I}-\boldsymbol{I}\right)\boldsymbol{T}^{-1}\boldsymbol{t}\boldsymbol{\pi}e^{\boldsymbol{T}u}\boldsymbol{e}_i t_{ij}du.\end{aligned} \tag{4.45}$$

Using that $\boldsymbol{T}\boldsymbol{e}=-\boldsymbol{t}$, which is equivalent to $\boldsymbol{T}^{-1}\boldsymbol{t}=-\boldsymbol{e}$, and $\boldsymbol{e}_j'\boldsymbol{I}\boldsymbol{e}=\boldsymbol{e}_j'\boldsymbol{e}=1$, we get

$$\begin{aligned}&\mathbb{E}\left(N_{ij}\left(t\right)1\left\{Y\le t\right\}\right) \\ &= \int_0^t \boldsymbol{e}_j'\left(\boldsymbol{I}-e^{\boldsymbol{T}(t-u)}\right)\boldsymbol{e}\boldsymbol{\pi}e^{\boldsymbol{T}u}\boldsymbol{e}_i t_{ij}du \\ &= \int_0^t \boldsymbol{e}_j'\boldsymbol{I}\boldsymbol{e}\boldsymbol{\pi}e^{\boldsymbol{T}u}\boldsymbol{e}_i t_{ij}du-\int_0^t \boldsymbol{e}_j'e^{\boldsymbol{T}(t-u)}\boldsymbol{e}\boldsymbol{\pi}e^{\boldsymbol{T}u}\boldsymbol{e}_i t_{ij}du \\ &= \int_0^t \boldsymbol{\pi}e^{\boldsymbol{T}u}\boldsymbol{e}_i t_{ij}du-\int_0^t \boldsymbol{e}_j'e^{\boldsymbol{T}(t-u)}e\boldsymbol{\pi}e^{\boldsymbol{T}u}\boldsymbol{e}_i t_{ij}du.\end{aligned} \tag{4.46}$$

Consequently, the remaining term in the numerator in 4.40 can be derived from 4.44 and 4.46:

$$
\begin{aligned}
&\mathbb{E}\left(N_{ij}(t)\,1\{Y>t\}\right)\\
&=\mathbb{E}\left(N_{ij}(t)\right)-\mathbb{E}\left(N_{ij}(t)\,1\{Y\le t\}\right)\\
&=\int_0^t t_{ij}\boldsymbol{\pi}e^{\boldsymbol{T}s}\boldsymbol{e}_i ds-\left(\int_0^t \boldsymbol{\pi}e^{\boldsymbol{T}u}\boldsymbol{e}_i t_{ij}du-\int_0^t \boldsymbol{e}'_j e^{\boldsymbol{T}(t-u)}e\boldsymbol{\pi}e^{\boldsymbol{T}u}\boldsymbol{e}_i t_{ij}du\right)\\
&=\int_0^t \boldsymbol{e}'_j e^{\boldsymbol{T}(t-u)}e\boldsymbol{\pi}e^{\boldsymbol{T}u}\boldsymbol{e}_i t_{ij}du
\end{aligned}
\tag{4.47}
$$

Finally insert 4.44, 4.46 and 4.47 into 4.40:

$$
\mathbb{E}(N_{ij}|Y\in(s,t])=\frac{t_{ij}\int_s^t \boldsymbol{\pi}e^{\boldsymbol{T}u}\boldsymbol{e}_i du-\int_s^t \boldsymbol{e}'_j e^{\boldsymbol{T}(t-u)}e\boldsymbol{\pi}e^{\boldsymbol{T}u}\boldsymbol{e}_i t_{ij}du}{\boldsymbol{\pi}e^{\boldsymbol{T}s}e-\boldsymbol{\pi}e^{\boldsymbol{T}t}e}
\tag{4.48}
$$

By an argument essentially identical to $N_{ij}(t)$, we calculate

$$
\begin{aligned}
\mathbb{E}\left(N_i(t)\right)&=\mathbb{E}\left(N_{i,p+1}(t)\right)\\
&=\int_0^t t_i\boldsymbol{\pi}e^{\boldsymbol{T}s}\boldsymbol{e}_i ds.
\end{aligned}
\tag{4.49}
$$

Furthermore, by the results in the last section we have

$$
\begin{aligned}
&\mathbb{E}\left(N_i(t)\,1\{Y\le t\}\right)\\
&=\int_0^t \mathbb{E}\left(N_i(y)\mid Y=y\right)f_Y(y)\,dy\\
&=\int_0^t \frac{\boldsymbol{\pi}e^{\boldsymbol{T}y}\boldsymbol{e}_i t_i}{\boldsymbol{\pi}e^{\boldsymbol{T}y}\boldsymbol{t}}\boldsymbol{\pi}e^{\boldsymbol{T}y}\boldsymbol{t}dy\\
&=\int_0^t \boldsymbol{\pi}e^{\boldsymbol{T}y}\boldsymbol{e}_i t_i dy.
\end{aligned}
\tag{4.50}
$$

Notice that $\mathbb{E}(N_i(t)1\{Y>t\})=0$ since absorption has not taken place yet. Therefore, we have

$$
\begin{aligned}
\mathbb{E}(N_i(t)|Y\in(s,t])&=\frac{\mathbb{E}\left(N_i(t)\right)-\mathbb{E}\left(N_i(t)\,1\{Y\le s\}\right)-\mathbb{E}\left(N_i(t)\,1\{Y>t\}\right)}{\boldsymbol{\pi}e^{\boldsymbol{T}s}\boldsymbol{e}-\boldsymbol{\pi}e^{\boldsymbol{T}t}\boldsymbol{e}}\\
&=\frac{\mathbb{E}\left(N_i(t)\right)-\mathbb{E}\left(N_i(t)\,1\{Y\le s\}\right)}{\boldsymbol{\pi}e^{\boldsymbol{T}s}\boldsymbol{e}-\boldsymbol{\pi}e^{\boldsymbol{T}t}\boldsymbol{e}}\\
&=\frac{t_i\int_s^t \boldsymbol{\pi}e^{\boldsymbol{T}u}\boldsymbol{e}_i du}{\boldsymbol{\pi}e^{\boldsymbol{T}s}\boldsymbol{e}-\boldsymbol{\pi}e^{\boldsymbol{T}t}\boldsymbol{e}}.
\end{aligned}
\tag{4.51}
$$

Concerning $Z_i(t)$, we use the exact same method again:

$$\begin{aligned}\mathbb{E}(Z_i(t)) &= \mathbb{E}\left(\int_0^t 1\{X_u = i\}\, du\right) \\ &= \int_0^t \mathbb{P}(X_u = i) du \\ &= \int_0^t \boldsymbol{\pi} e^{\boldsymbol{T}u} \boldsymbol{e}_i du\end{aligned} \tag{4.52}$$

Analogous to the formula 4.51, we calculate the terms for Z_i as

$$\begin{aligned}\mathbb{E}(Z_i(s)1\{Y \le s\}) &= \mathbb{E}\left(\int_0^s 1\{X_u = i, Y \le s\}\, du\right) \\ &= \int_0^s \mathbb{P}(X_u = i, Y \le s) du \\ &= \int_0^s \mathbb{P}(Y \le s | X_u = i)\mathbb{P}(X_u = i) du \\ &= \int_0^s \mathbb{P}(Y \le s - u | X_0 = i)\mathbb{P}(X_u = i) du \\ &= \int_0^s (1 - \boldsymbol{e}_i' e^{\boldsymbol{T}(s-u)} e)\boldsymbol{\pi} e^{\boldsymbol{T}u} \boldsymbol{e}_i du \\ &= \int_0^s \boldsymbol{\pi} e^{\boldsymbol{T}u} \boldsymbol{e}_i du - \int_0^s \boldsymbol{e}_i' e^{\boldsymbol{T}(s-u)} \boldsymbol{e}\boldsymbol{\pi} e^{\boldsymbol{T}u} \boldsymbol{e}_i du\end{aligned} \tag{4.53}$$

and

$$\begin{aligned}\mathbb{E}\left(Z_i(t)\, 1\{Y > t\}\right) &= \mathbb{E}\left(Z_i(t)\right) - \mathbb{E}\left(Z_i(t)\, 1\{Y \le t\}\right) \\ &= \int_0^t \boldsymbol{\pi} e^{\boldsymbol{T}u} \boldsymbol{e}_i du - \left(\int_0^t \boldsymbol{\pi} e^{\boldsymbol{T}u} \boldsymbol{e}_i du - \int_0^t \boldsymbol{e}_i' e^{\boldsymbol{T}(s-u)} \boldsymbol{e}\boldsymbol{\pi} e^{\boldsymbol{T}u} \boldsymbol{e}_i du\right) \\ &= \int_0^t \boldsymbol{e}_i' e^{\boldsymbol{T}(s-u)} \boldsymbol{e}\boldsymbol{\pi} e^{\boldsymbol{T}u} \boldsymbol{e}_i du\end{aligned} \tag{4.54}$$

From 4.52, 4.53 and 4.54, we finally obtain the uncensored formula for Z_i:

$$\mathbb{E}(Z_i(t) | Y \in (s, t]) = \frac{\int_s^t \boldsymbol{\pi} e^{\boldsymbol{T}u} \boldsymbol{e}_i du - \int_s^t \boldsymbol{e}_i' e^{\boldsymbol{T}(s-u)} \boldsymbol{e}\boldsymbol{\pi} e^{\boldsymbol{T}u} \boldsymbol{e}_i du}{\boldsymbol{\pi} e^{\boldsymbol{T}s} \boldsymbol{e} - \boldsymbol{\pi} e^{\boldsymbol{T}t} \boldsymbol{e}} \tag{4.55}$$

The unconditional expectation of B_i is given by

$$\begin{aligned}\mathbb{E}(B_i) &= \mathbb{E}(1\{X_0 = i\}) \\ &= \mathbb{P}(X_0 = i) \\ &= \pi_i.\end{aligned} \tag{4.56}$$

We easily calculate the remaining terms in the censored formula as

$$\begin{aligned}\mathbb{E}(B_i 1\{Y \leq s\}) &= \mathbb{E}(1\{X_0 = i\}1\{Y \leq s\}) \\ &= \mathbb{P}(Y \leq s | X_0 = i)\mathbb{P}(X_0 = i) \\ &= \pi_i(1 - e_i' e^{\boldsymbol{T}s}\boldsymbol{e}),\end{aligned} \tag{4.57}$$

and

$$\mathbb{E}(B_i 1\{Y > s\}) = \pi_i e_i' e^{\boldsymbol{T}s}\boldsymbol{e}. \tag{4.58}$$

Hence, from 4.56, 4.57 and 4.58, we obtain the censored formula for B_i:

$$\mathbb{E}(B_i | Y \in (s,t]) = \frac{\pi_i e_i' e^{\boldsymbol{T}s}\boldsymbol{e} - \pi_i e_i' e^{\boldsymbol{T}t}\boldsymbol{e}}{\boldsymbol{\pi} e^{\boldsymbol{T}s}\boldsymbol{e} - \boldsymbol{\pi} e^{\boldsymbol{T}t}\boldsymbol{e}} \tag{4.59}$$

As mentioned in the beginning of the section, the formulas in the last section are replaced with the censored formulas 4.14, 4.17, 4.19 and 4.23, when the individual data points are censored. With the exception of this modification of the expectation step, the EM algorithm for censored observations is unchanged. By using the relation between integrals and matrix exponential in 4.30, it is also possible to write the conditional expectations of the sufficient statistics 4.14, 4.17, 4.19 and 4.23 on explicit form. This makes the EM algorithm tractable for estimation of phase-type distributions. However, as one might expect, the rate of convergence of the EM algorithm decreases with the number of censored observations in the dataset.

4.4 EM algorithm for discrete phase-type distributions

In this section the EM algorithm of discrete phase-type distributions are derived. As mentioned in section 3.3, the complete likelihood function for discrete phase-type distributions was calculated. By taking the natural logarithm to the complete likelihood function derived in section 3.3, we would see that the complete log likelihood function is linear in the sufficient statistics, just as in the scenario with continuous phase-type distributions. Therefore, it suffices to calculate the conditional expectation of the sufficient statistics given data. As in the two last sections, only a single data point is considered.

Recall, that B_i is the number of realizations which initiate in state i. Hence, when we consider one observation, B_i is just an indicator function which activates when the process starts in state i, i.e., $B_i = 1\{x_0 = i\}$. Calculating the expected value of B_i conditioned on our data, gives

$$\begin{aligned}&\mathbb{E}\left(B_i \mid Y = y\right) \\ &= \mathbb{E}\left(1\left\{X_0 = i\right\} \mid Y = y\right) = \mathbb{P}\left(X_0 = i \mid Y = y\right) \\ &= \frac{\mathbb{P}(Y = y | X_0 = i)\mathbb{P}(X_0 = i)}{\mathbb{P}(Y = y)} \\ &= \frac{\pi_i \boldsymbol{e}_i' \boldsymbol{T}^{y_k - 1} t}{\boldsymbol{\pi}\boldsymbol{T}^{y_k - 1} t}.\end{aligned} \tag{4.60}$$

In the 4^{th} equality we use that $\mathbb{P}(Y = y \mid X_0 = i) = \boldsymbol{e}_i' \boldsymbol{T}^{y_k - 1} t$, since we initiate in state i with probability one. In other words, $Y|X_0 = i \sim \ DPH_p\left(\boldsymbol{e}_i', \boldsymbol{T}\right)$.

When we calculate N_{ij} for one observation we get:

$$\begin{aligned}
\mathbb{E}\left(N_{ij} \;\middle|\; Y = y\right) &= \mathbb{E}\left(\sum_{k=0}^{y-1} 1\left\{X_k = i, X_{k+1} = j\right\} \;\middle|\; Y = y\right) \\
&= \mathbb{E}\left(1\{y \geq 2\} \sum_{k=0}^{y-2} 1\left\{X_k = i, X_{k+1} = j\right\} \;\middle|\; Y = y\right) \\
&= 1\{y \geq 2\} \sum_{k=0}^{y-2} \mathbb{P}\left(X_k = i, X_{k+1} = j \mid Y = y\right)
\end{aligned} \tag{4.61}$$

In the 2nd equality we use that the time of absorption must not be before time 2, in order to have a transition between two transient states i and j, since given absorption in time y a jump in time $y - 1$ must be to the absorbing state. Using Markov property we can rewrite the inner term as

$$\begin{aligned}
&\mathbb{P}\left(X_k = i, X_{k+1} = j \mid Y = y\right) \\
&= \frac{\mathbb{P}(Y = y | X_k = i, X_{k+1} = j)\mathbb{P}(X_k = i, X_{k+1} = j)}{\mathbb{P}(Y = y)} \\
&= \frac{\mathbb{P}\left(Y = y \mid X_k = i, X_{k+1} = j\right)\mathbb{P}\left(X_{k+1} = j \mid X_k = i\right)\mathbb{P}\left(X_k = i\right)}{\mathbb{P}\left(Y = y\right)} \\
&= \frac{\mathbb{P}\left(Y = y \mid X_{k+1} = j\right)\mathbb{P}\left(X_{k+1} = j \mid X_k = i\right)\mathbb{P}\left(X_k = i\right)}{\mathbb{P}\left(Y = y\right)}
\end{aligned} \tag{4.62}$$

Hence, we get

$$\begin{aligned}
&\mathbb{E}\left(N_{ij} \;\middle|\; Y = y\right) \\
&= 1\{y \geq 2\} \sum_{k=0}^{y-2} \frac{\mathbb{P}\left(Y = y \mid X_{k+1} = j\right)\mathbb{P}\left(X_{k+1} = j \mid X_k = i\right)\mathbb{P}\left(X_k = i\right)}{\mathbb{P}(Y = y)} \\
&= 1\{y \geq 2\} \sum_{k=0}^{y-2} \frac{\boldsymbol{e}_j' \boldsymbol{T}^{y-(k+1)-1} \boldsymbol{t} \boldsymbol{\pi} \boldsymbol{T}^k \boldsymbol{e}_i t_{ij}}{\boldsymbol{\pi} \boldsymbol{T}^{y-1} \boldsymbol{t}}.
\end{aligned} \tag{4.63}$$

In the numerator we use that

$$\begin{aligned}
\mathbb{P}\left(Y = y \mid X_{k+1} = j\right) &= \mathbb{P}\left(Y = y - (k+1) \mid X_0 = j\right) \\
&= \boldsymbol{e}_j' \boldsymbol{T}^{y-(k+1)-1} \boldsymbol{t}.
\end{aligned} \tag{4.64}$$

Similarly, for N_i we calculate

$$
\begin{aligned}
&\mathbb{E}\left(N_i \mid Y = y\right) \\
&= \mathbb{E}\left(\sum_{k=0}^{y-1} 1\left\{X_k = i, X_{k+1} = p+1\right\} \;\middle|\; Y = y\right) \\
&= \mathbb{E}\left(1\left\{X_{y-1} = i, X_y = p+1\right\} \;\middle|\; Y = y\right) \\
&= \mathbb{P}\left(X_{y-1} = i, X_y = p+1 \;\middle|\; Y = y\right) \\
&= \frac{\mathbb{P}(Y = y | X_{y-1} = i, X_y = p+1)\mathbb{P}\left(X_y = p+1 \;\middle|\; X_{y-1} = i\right)\mathbb{P}\left(X_{y-1} = i\right)}{\mathbb{P}(Y = y)}.
\end{aligned} \tag{4.65}
$$

In the second equality we use the fact that we know the underlying discrete time Markov chain will jump to the absorbing state $p+1$ in time y, so the remaining terms cancel out. Fourth equality follows easily from calculations analogous to 4.62. Now since $\mathbb{P}\left(X_y = p+1\right) = \mathbb{P}\left(Y = y\right)$ we have

$$
\begin{aligned}
\mathbb{P}\left(Y = y \;\middle|\; X_{y-1} = i, X_y = p+1\right) &= \mathbb{P}\left(Y = y \;\middle|\; X_{y-1} = i,\ Y = y\right) \\
&= 1
\end{aligned} \tag{4.66}
$$

so

$$
\begin{aligned}
\mathbb{E}\left(N_i \mid Y = y\right) &= \frac{\mathbb{P}\left(Y = y \;\middle|\; X_{y-1} = i\right)\mathbb{P}\left(X_{y-1} = i\right)}{\mathbb{P}(Y = y)} \\
&= \frac{\boldsymbol{\pi}\boldsymbol{T}^{y-1}\boldsymbol{e}_i t_i}{\boldsymbol{\pi}\boldsymbol{T}^{y-1}\boldsymbol{t}}.
\end{aligned} \tag{4.67}
$$

We have now calculated all the conditional sufficient statistics. Based on these calculations and the maximum likelihood estimates from section 3.3, we derive the EM algorithm for discrete phase-type distributions.

0. Initialize with some arbitrary $(\boldsymbol{\pi}_0, \boldsymbol{T}_0)$
1. (E step) Calculate

$$
\mathbb{E}(B_i | \boldsymbol{Y} = \boldsymbol{y}) = \sum_{k=1}^{N} \frac{\pi_i \boldsymbol{e}_i' \boldsymbol{T}^{y_k - 1}\boldsymbol{t}}{\boldsymbol{\pi}\boldsymbol{T}^{y_k - 1}\boldsymbol{t}} \tag{4.68}
$$

$$
\mathbb{E}(N_{ij} | \boldsymbol{Y} = \boldsymbol{y}) = \sum_{k=1}^{N} 1\{y_k \geq 2\} \sum_{l=0}^{y_k - 2} \frac{\boldsymbol{e}_j' \boldsymbol{T}^{y_k - (l+1) - 1}\boldsymbol{t}\boldsymbol{\pi}\boldsymbol{T}^{l}\boldsymbol{e}_i t_{ij}}{\boldsymbol{\pi}\boldsymbol{T}^{y_k - 1}\boldsymbol{t}} \tag{4.69}
$$

$$
\mathbb{E}(N_i | \boldsymbol{Y} = \boldsymbol{y}) = \sum_{k=1}^{N} \frac{\boldsymbol{\pi}\boldsymbol{T}^{y_k - 1}\boldsymbol{e}_i t_i}{\boldsymbol{\pi}\boldsymbol{T}^{y_k - 1}\boldsymbol{t}} \tag{4.70}
$$

$$
\tag{4.71}
$$

2. (M step) Let

$$\hat{\pi}_i = \frac{\mathbb{E}(B_i|\boldsymbol{Y}=\boldsymbol{y})}{N} \qquad \text{for } i = 1, \cdots, p \tag{4.72}$$

$$\hat{t}_{ij} = \frac{\mathbb{E}(N_{ij}|\boldsymbol{Y}=\boldsymbol{y})}{\sum_{j=1}^{p}\mathbb{E}(N_{ij}|\boldsymbol{Y}=\boldsymbol{y}) + \mathbb{E}(N_i|\boldsymbol{Y}=\boldsymbol{y})}, \qquad \text{for } i, j = 1, \cdots, p \tag{4.73}$$

$$\hat{t}_i = \frac{\mathbb{E}(N_i|\boldsymbol{Y}=\boldsymbol{y})}{\sum_{j=1}^{p}\mathbb{E}(N_{ij}|\boldsymbol{Y}=\boldsymbol{y}) + \mathbb{E}(N_i|\boldsymbol{Y}=\boldsymbol{y})}, \qquad \text{for } i = 1, \cdots, p \tag{4.74}$$

and let $\hat{\boldsymbol{\pi}} = (\hat{\pi}_1, \ldots, \hat{\pi}_p)$, $\hat{\boldsymbol{T}} = \{\hat{t}_{ij}\}_{i,j=1,\ldots,p}$, and $\hat{\boldsymbol{t}} = (\hat{t}_1, \ldots, \hat{t}_p)'$.

3. Assign $\boldsymbol{\pi} := \hat{\boldsymbol{\pi}}, \boldsymbol{T} := \hat{\boldsymbol{T}}, \boldsymbol{t} := \hat{\boldsymbol{t}}$, and GOTO 1.

4.5 Numerical estimation

The expectation step can be numerically challenging to calculate, due to the matrix exponentials. Several methods have been proposed to calculate matrix exponentials numerically. One frequently used method for calculations of matrix exponentials in the EM algorithm is the Runge-Kutta method. This method will be described below. Define

$$\begin{aligned} a(y_k|\boldsymbol{\theta}) &= \boldsymbol{\pi}e^{\boldsymbol{T}y_k} \\ b(y_k|\boldsymbol{\theta}) &= e^{\boldsymbol{T}y_k}\boldsymbol{t} \\ c(y_k,\ i|\boldsymbol{\theta}) &= \int_0^{y_k} \boldsymbol{\pi}e^{\boldsymbol{T}u}\boldsymbol{e}_i e^{\boldsymbol{T}(y_k-u)}\boldsymbol{t}du \end{aligned} \tag{4.75}$$

for $i = 1, \ldots, p$. Hence $a(y_k|\boldsymbol{\theta})$ is a p dimensional row vector and $b(y_k|\boldsymbol{\theta})$ and $c(y_k|\boldsymbol{\theta})$ is a p dimensional vector. Consequently, the ith entry in the vectors are given by

$$\begin{aligned} a_i\left(y_k \mid \boldsymbol{\theta}\right) &= \boldsymbol{\pi}e^{\boldsymbol{T}y_k}\boldsymbol{e}_i \\ b_i(y_k|\boldsymbol{\theta}) &= \boldsymbol{e}_i' e^{\boldsymbol{T}y_k}\boldsymbol{t} \\ c_i(y_k,\ i|\boldsymbol{\theta}) &= \int_0^{y_k} \boldsymbol{\pi}e^{\boldsymbol{T}u}\boldsymbol{e}_i\boldsymbol{e}_i' e^{\boldsymbol{T}(y_k-u)}\boldsymbol{t}du. \end{aligned} \tag{4.76}$$

The conditional expectations in the expectations step in the EM algorithm can then be expressed in terms of these vector functions:

$$\begin{aligned} \mathbb{E}\left(B_i \mid Y_k = y_k\right) &= \frac{\pi_i \boldsymbol{e}_i' e^{\boldsymbol{T}y_k}\boldsymbol{t}}{\boldsymbol{\pi}e^{\boldsymbol{T}y_k}\boldsymbol{t}} \\ &= \frac{\pi_i b_i\left(y_k \mid \boldsymbol{\theta}\right)}{\boldsymbol{\pi}b\left(y_k \mid \boldsymbol{\theta}\right)} \end{aligned} \tag{4.77}$$

$$\begin{aligned} \mathbb{E}\left(Z_i \mid Y_k = y_k\right) &= \frac{\int_0^{y_k} \boldsymbol{\pi}e^{\boldsymbol{T}u}\boldsymbol{e}_i\boldsymbol{e}_i' e^{\boldsymbol{T}(y_k-u)}\boldsymbol{t}du}{\boldsymbol{\pi}e^{\boldsymbol{T}y_k}\boldsymbol{t}} \\ &= \frac{c_i\left(y_k, i \mid \boldsymbol{\theta}\right)}{\boldsymbol{\pi}b\left(y_k \mid \boldsymbol{\theta}\right)} \end{aligned} \tag{4.78}$$

$$\begin{aligned}\mathbb{E}\left(N_i \mid Y_k = y_k\right) &= \frac{t_i \boldsymbol{\pi} e^{\boldsymbol{T} y_k} \boldsymbol{e}_i}{\boldsymbol{\pi} e^{\boldsymbol{T} y_k} \boldsymbol{t}} \\ &= \frac{t_i a_i(y_k|\boldsymbol{\theta})}{\boldsymbol{\pi} b(y_k|\boldsymbol{\theta})}\end{aligned} \tag{4.79}$$

$$\begin{aligned}\mathbb{E}\left(N_{ij} \;\middle|\; Y_k = y_k\right) &= \frac{t_{ij} \int_0^{y_k} \boldsymbol{\pi} e^{\boldsymbol{T} u} \boldsymbol{e}_i \boldsymbol{e}_j' e^{\boldsymbol{T}(y_k - u)} \boldsymbol{t} du}{\boldsymbol{\pi} e^{\boldsymbol{T} y_k} \boldsymbol{t}} \\ &= \frac{t_{ij} c_j\left(y_k, i \mid \boldsymbol{\theta}\right)}{\boldsymbol{\pi} b\left(y_k \mid \boldsymbol{\theta}\right)}\end{aligned} \tag{4.80}$$

Differentiating 4.76 with respect to y_k and using 4.26, gives

$$\begin{aligned}a_i'\left(y \mid \boldsymbol{\theta}\right) &= \boldsymbol{\pi} e^{\boldsymbol{T} y_k} \boldsymbol{T} \boldsymbol{e}_i \\ b_i'(y_k|\boldsymbol{\theta}) &= \boldsymbol{e}_i' \boldsymbol{T} e^{\boldsymbol{T} y_k} \boldsymbol{t} \\ c_i'\left(y_k,\ i \mid \boldsymbol{\theta}\right) &= t_{ij} c\left(y,\ i \mid \boldsymbol{\theta}\right) + t_i a_i\left(y \mid \boldsymbol{\theta}\right) t.\end{aligned} \tag{4.81}$$

Using matrix notation gives the following equations:

$$\begin{aligned}a'\left(y \mid \boldsymbol{\theta}\right) - \ a(y|\boldsymbol{\theta}) \boldsymbol{T} \ &= \boldsymbol{0} \\ b'\left(y \mid \boldsymbol{T}\right) - \ \boldsymbol{T} b(y|\boldsymbol{\theta}) &= \boldsymbol{0} \\ c'(y,\ i|\boldsymbol{\theta}) - \boldsymbol{T} c\left(y,\ i \mid \boldsymbol{\theta}\right) - a_i(y|\boldsymbol{\theta}) \boldsymbol{t} &= \boldsymbol{0}\end{aligned} \tag{4.82}$$

Combining these with the initial conditions $a(0|\boldsymbol{\theta}) = \boldsymbol{\pi}$, $b(0|\boldsymbol{\theta}) = \boldsymbol{t}$, and $c(0,\ i|\boldsymbol{\theta}) = 0$ for $i = 1, 2, \ldots, p$ gives $p(p+1)$ homogeneous differential equations. These differential equations can be solved numerically to derive an explicit solution to the matrix exponentials in the expectation step.

4.6 Approximating phase–type distributions to continuous distributions using the EM algorithm

This section elaborate on how to use the EM algorithm to fit phase-type distributions to continuous theoretical distributions. Suppose we simulate observations from the theoretical distribution. If the observations are drawn independently N times from the theoretical distribution, the dataset will consist of N independent and identically distributed observations. First consider the estimated values of the initial probability vector. By inserting the expectations step into the maximization step, we get

$$\hat{\pi}_i \ = \frac{1}{N} \sum_{k=1}^{N} \frac{\pi_i \boldsymbol{e}_i' e^{\boldsymbol{T} y_k} \boldsymbol{t}}{\boldsymbol{\pi} e^{\boldsymbol{T} y_k} \boldsymbol{t}}. \tag{4.83}$$

Since the observations are independent and identically distributed it follows by the law of large numbers that

$$\begin{aligned}\hat{\pi}_i &\to \mathbb{E}(\frac{\pi_i \boldsymbol{e}_i' e^{\boldsymbol{T}} \boldsymbol{t}}{\boldsymbol{\pi} e^{\boldsymbol{T}Y} \boldsymbol{t}}) \\ &= \int_0^\infty \frac{\pi_i \boldsymbol{e}_i' e^{\boldsymbol{T}y} \boldsymbol{t}}{\boldsymbol{\pi} e^{\boldsymbol{T}y} \boldsymbol{t}} h(y) dy.\end{aligned} \tag{4.84}$$

Likewise, t_{ij} can be estimated by

$$\begin{aligned}\hat{t}_{ij} &= \frac{\mathbb{E}(N_{ij}|\boldsymbol{Y} = \boldsymbol{y})}{\mathbb{E}(Z_i|\boldsymbol{Y} = \boldsymbol{y})} \\ &= \frac{\frac{1}{N}\mathbb{E}(N_{ij}|\boldsymbol{Y} = \boldsymbol{y})}{\frac{1}{N}\mathbb{E}(Z_i|\boldsymbol{Y} = \boldsymbol{y})} \\ &= \frac{\frac{1}{N}\sum_{k=1}^N \int_0^{y_k} \frac{\boldsymbol{e}_j' e^{\boldsymbol{T}(y_k-u)} \boldsymbol{t} \boldsymbol{\pi} e^{\boldsymbol{t}u} \boldsymbol{e}_i}{\boldsymbol{\pi} e^{\boldsymbol{T}y_k} \boldsymbol{t}} t_{ij} du}{\frac{1}{N}\sum_{k=1}^N \int_0^{y_k} \frac{\boldsymbol{\pi} e^{\boldsymbol{t}u} \boldsymbol{e}_i \boldsymbol{e}_i' e^{\boldsymbol{T}(y_k-u)} \boldsymbol{t}}{\boldsymbol{\pi} e^{\boldsymbol{T}y_k} \boldsymbol{t}} du} \\ &\to \frac{\int_0^\infty t_{ij} \int_0^y \boldsymbol{\pi} e^{\boldsymbol{t}u} \boldsymbol{e}_i \boldsymbol{e}_j' e^{\boldsymbol{T}(y-u)} \boldsymbol{t} du h(y) dy}{\int_0^\infty \int_0^y \boldsymbol{\pi} e^{\boldsymbol{t}u} \boldsymbol{e}_i \boldsymbol{e}_i' e^{\boldsymbol{T}(y-u)} \boldsymbol{t} du h(y) dy} \\ &= \frac{\int_0^\infty \frac{t_{ij}}{\boldsymbol{\pi} e^{\boldsymbol{t}y} \boldsymbol{t}} \int_0^y \boldsymbol{\pi} e^{\boldsymbol{t}u} \boldsymbol{e}_i \boldsymbol{e}_j' e^{\boldsymbol{T}(y-u)} \boldsymbol{t} du h(y) dy}{\int_0^\infty \frac{1}{\boldsymbol{\pi} e^{\boldsymbol{t}y} \boldsymbol{t}} \int_0^y \boldsymbol{\pi} e^{\boldsymbol{t}u} \boldsymbol{e}_i \boldsymbol{e}_i' e^{\boldsymbol{T}(y-u)} \boldsymbol{t} du h(y) dy}.\end{aligned} \tag{4.85}$$

A formula for t_i can be derived analogous to t_{ij}. The integrals can then be numerically approximated by simple discretization, which means conversion of the integral to a weighted sum. For numerical implementation, an upper limit has to be specified. Consequently, heavy-tailed theoretical distributions require more computational power since numerical integration has to be done with a higher upper limit on the weighted sum, in order to have a good fit. We end this section by summarizing the algorithm:

0.: *Initialize with some "arbitrary"* $(\pi_0, \boldsymbol{T}_0)$ *and let* $n = 0$.
1.: *Keep iterating*

$$(\boldsymbol{\pi}_{n+1})_i = \int_0^\infty \frac{(\boldsymbol{\pi}_n)_i \boldsymbol{e}_i' e^{\boldsymbol{T}_n y} \boldsymbol{t}_n}{\boldsymbol{\pi}_n e^{\boldsymbol{T}_n y} \boldsymbol{t}_n} h(y) dy \tag{4.86}$$

$$(\boldsymbol{T}_{n+1})_{ij} = \frac{\int_0^\infty \frac{(\boldsymbol{T}_n)_{ij}}{\boldsymbol{\pi}_n e^{\boldsymbol{T}_n y} \boldsymbol{t}_n} \int_0^y \boldsymbol{\pi}_n e^{\boldsymbol{T}_n u} \boldsymbol{e}_i \boldsymbol{e}_j' e^{\boldsymbol{T}_n (y-u)} \boldsymbol{t}_n du h(y) dy}{\int_0^\infty \frac{1}{\boldsymbol{\pi}_n e^{\boldsymbol{T}_n y} \boldsymbol{t}_n} \int_0^y \boldsymbol{\pi}_n e^{\boldsymbol{T}_n u} \boldsymbol{e}_i \boldsymbol{e}_i' e^{\boldsymbol{T}_n (y-u)} \boldsymbol{t}_n du h(y) dy} \tag{4.87}$$

$$(\boldsymbol{t}_{n+1})_j = \frac{\int_0^\infty \frac{\boldsymbol{\pi}_n e^{\boldsymbol{T}_n y} \boldsymbol{e}_i}{\boldsymbol{\pi}_n e^{\boldsymbol{T}_n y} \boldsymbol{t}_n} (\boldsymbol{t}_n)_i h(y) dy}{\int_0^\infty \frac{1}{\boldsymbol{\pi}_n e^{\boldsymbol{T}_n y} \boldsymbol{t}_n} \int_0^y \boldsymbol{\pi}_n e^{\boldsymbol{T}_n u} \boldsymbol{e}_i \boldsymbol{e}_i e^{\boldsymbol{T}_n (y-u)} \boldsymbol{t}_n du h(y) dy} \tag{4.88}$$

until convergence.

4.7 Bibliographic references

The EM algorithm was first introduced by Dempster (1977) [12] as an iterative method for finding maximum likelihood estimates when data is incomplete. Fitting phase-type distributions using the EM algorithm, was introduced by Asmussen et.al [4] for complete data and was later extended to censored observations in [31]. Horvath et.alin [22] derived formulas for fitting phase-type distributions to discrete and continuous distributions. The EM algorithm, applied to discrete phase-type distributions was also carried out in [9]. Calculations of sufficient statistics are primarily based on [8]. Numerically estimation of matrix exponential is based on [17]. The Runge-Kutta method was originally developed around 1900 in [25].

Chapter 5

Denseness of phase-type distributions

In section 5.1 we will prove that any continuous distribution with positive support can be well-approximated by a continuous phase-type distribution, when the dimension of the phase-type distribution is sufficiently high. Section 5.2 contains a numeric study of this denseness property. This approximation constitute a finite mixture of Erlang distributions, which will be used to approximate different theoretical distributions. We end this chapter with an evaluation in section 5.3.

5.1 Denseness of phase-type distributions

We will now prove that continuous phase-type distributions are dense in $[0, \infty)$. The following lemma will be used in the proof of the denseness property:

Lemma 5.1.1 *Let $\{F_{n,\theta}\}_{n\in\mathbb{N}}$ be a family of distributions and u be some bounded and continuous function. Furthermore, let $X_n \sim F_{n,\theta}$ have a distribution where, for all $n \in \mathbb{N}$, the following assumptions hold:*

$$\mathbb{E}(X_n) = \theta \tag{5.1}$$

$$\textit{The variance is finite and } \sigma_n^2(\theta) \to 0 \textit{ for } n \to \infty \tag{5.2}$$

Then for any sequence of random variables $\{X_n\}_{n\in\mathbb{N}}$, with the assumptions specified in 5.1 and 5.2, it holds that

$$X_n \xrightarrow{d} \theta \textit{ for } n \to \infty, \tag{5.3}$$

which is equivalent to

$$\mathbb{E}(X_n) \to u(\theta) \textit{ for } \ n \to \infty. \tag{5.4}$$

Lemma 5.1.1 will now be proved. By the triangle inequality, we have

$$\begin{aligned}
&|\mathbb{E}\left(u\left(X_n\right)\right) - u\left(\theta\right)| \\
&= \left| \int_0^\infty u\left(x\right) dF_{n,\theta}\left(x\right) - u\left(\theta\right) \right| \\
&\leq \int_0^\infty |\, u\left(x\right) - u\left(\theta\right)| \, dF_{n,\theta}\left(x\right) \\
&= \int_{\{x:|x-\theta|<\delta\}} |u\left(x\right) - u\left(\theta\right)| \, dF_{n,\theta}\left(x\right) + \int_{\{x:|x-\theta|\geq\delta\}} |u\left(x\right) - u\left(\theta\right)| \, dF_{n,\theta}\left(x\right).
\end{aligned} \tag{5.5}$$

Since the function is continuous, we have

$$\exists \delta > 0 \forall \varepsilon > 0 : |u\left(x\right) - u\left(\theta\right)| < \varepsilon \Rightarrow |x - \theta| < \delta. \tag{5.6}$$

Without loss of generality, we can choose a δ such that $|u\left(x\right) - u\left(\theta\right)| < \frac{\varepsilon}{2}$. Applying this inequality on the first term in 5.5 gives

$$\begin{aligned}
\int_{\{x:|x-\theta|<\delta\}} |u\left(x\right) - u\left(\theta\right)| \, dF_{n,\theta}\left(x\right) &\leq \int_{\{x:|x-\theta|<\delta\}} \frac{\varepsilon}{2} dF_{n,\theta}\left(x\right) \\
&= \frac{\varepsilon}{2} \mathbb{P}\left(|X_n - \theta| < \delta\right) \\
&\leq \frac{\varepsilon}{2}.
\end{aligned} \tag{5.7}$$

Since the functions is bounded, there exist a constant, for example $M/2$, such that $|u\left(x\right)| < M/2$, and consequently $|u\left(x\right) - u\left(\theta\right)| < M$. Therefore, we can then write the second term in 5.5 as

$$\begin{aligned}
\int_{\{x:|x-\theta|\geq\delta\}} |\, u\left(x\right) - u\left(\theta\right)| \, dF_{n,\theta}\left(x\right) &\leq \int_{\{x:|x-\theta|\geq\delta\}} M dF_{n,\theta}\left(x\right) \\
&< M\mathbb{P}\left(|X_n - \theta| \geq \delta\right).
\end{aligned} \tag{5.8}$$

By using that the variance is finite, we get

$$
\begin{aligned}
\mathbb{P}\left(|X_n - \theta| \geq \delta\right) &= \int_{\{|X_n-\theta|\geq\delta\}} dF_{n,\theta}\left(x\right) \\
&\leq \int_{\{|X_n-\theta|\geq\delta\}} \frac{|X_n-\theta|^2}{\delta^2} dF_{n,\theta}\left(x\right) \\
&= \frac{1}{\delta^2}\int_{\{|X_n-\theta|\geq\delta\}} \left(X_n - \mathbb{E}\left(X_n\right)\right)^2 dF_{n,\theta}\left(x\right) \\
&= \frac{1}{\delta^2}\int_{\{|X_n-\theta|\geq\delta\}} \sigma_n^2\left(\theta\right) dF_{n,\theta}\left(x\right) \\
&= \frac{\sigma_n^2\left(\theta\right)}{\delta^2}\mathbb{P}\left(|X_n-\theta|\geq\delta\right) \\
&\leq \frac{\sigma_n^2\left(\theta\right)}{\delta^2},
\end{aligned}
\tag{5.9}
$$

where the inequality $\mathbb{P}\left(|X_n - \theta| \geq \delta\right) \leq \frac{\sigma_n^2(\theta)}{\delta^2}$ in 5.9 is a special case of the so-called Chebyshev's inequality. Since the variance converges to zero $\sigma_n^2\left(\theta\right) \to 0$ for $n \to \infty$ we have, by the definition of point-wise convergence, that

$$
\exists n_0 \in \mathbb{N} \forall n \geq n_0 : \ \sigma_n^2\left(\theta\right) \leq \frac{\varepsilon}{M2}. \tag{5.10}
$$

Inserting 5.10 and 5.9 into 5.8, gives

$$
\begin{aligned}
\int_{\{x:|x-\theta|\geq\delta\}} |\, u\left(x\right) - u\left(\theta\right)| \, dF_{n,\theta}\left(x\right) &< M\frac{\sigma_n^2\left(\theta\right)}{\delta^2} \\
&\leq M\frac{\varepsilon}{M2} \\
&= \frac{\varepsilon}{2}.
\end{aligned}
\tag{5.11}
$$

Finally, insert 5.11 and 5.7 into 5.5:

$$
\exists \delta > 0 \forall \varepsilon > 0 : |\mathbb{E}\left(u\left(X_n\right)\right) - u\left(\theta\right)| < \varepsilon \Rightarrow |x - \theta| < \delta \tag{5.12}
$$

By the definition of convergence in distribution $X_n \xrightarrow{d} X$, so the lemma is proved. Any continuous, bounded and positive function can therefore, for n large enough, be well-approximated by any distribution where the assumptions in 5.2 and 5.1 hold.

Next, consider a family of Poisson distributions. Let $X_n = \frac{1}{n}Y_n$ where $Y_n \sim \text{Pois}\left(nt\right)$. Then X_n fulfill the requirements above since

$$
\begin{aligned}
\mathbb{E}\left(X_n\right) &= \mathbb{E}\left(\frac{Y_n}{n}\right) \\
&= \frac{1}{n}\mathbb{E}\left(Y_n\right) \\
&= t
\end{aligned}
\tag{5.13}
$$

$$
\begin{aligned}
Var\left(X_n\right) &= Var\left(\frac{Y_n}{n}\right) \\
&= \frac{1}{n^2} Var\left(Y_n\right) \\
&= \frac{t}{n} \to 0 \text{ for } n \to \infty
\end{aligned}
\tag{5.14}
$$

The density of X_n is given by $P\left(Y_n = k\right) = e^{-nt}(nt)^k/k!$. Therefore, by lemma 5.1.1,

$$
\begin{aligned}
\mathbb{E}\left(F\left(X_n\right)\right) &= \mathbb{E}\left(F\left(\frac{Y_n}{n}\right)\right) \\
&= \sum_{k=0}^{\infty} F\left(\frac{k}{n}\right) e^{-nt} \frac{(nt)^k}{k!} \\
&\to F\left(t\right) \text{ for } n \to \infty.
\end{aligned}
\tag{5.15}
$$

Now, consider an infinite mixture of Erlang distributions:

$$
F_n\left(t\right) = \sum_{k=1}^{\infty} \left(F\left(\frac{k}{n}\right) - F\left(\frac{k-1}{n}\right)\right)\left(1 - \sum_{i=0}^{k-1} \frac{e^{-nt}(nt)^i}{i!}\right) \tag{5.16}
$$

By rearranging the formula, we get

$$
\begin{aligned}
F_n\left(t\right) &= \sum_{k=1}^{\infty} \left(F\left(\frac{k}{n}\right) - F\left(\frac{k-1}{n}\right)\right)\left(1 - \sum_{i=0}^{k-1} \frac{e^{-nt}(nt)^i}{i!}\right) \\
&= \sum_{k=1}^{\infty} \left(F\left(\frac{k}{n}\right) - F\left(\frac{k-1}{n}\right)\right) - \sum_{k=1}^{\infty} \left(F\left(\frac{k}{n}\right) - F\left(\frac{k-1}{n}\right)\right) \sum_{i=0}^{k-1} \frac{e^{-nt}(nt)^i}{i!}.
\end{aligned}
\tag{5.17}
$$

Consider the first term in 5.17. Since the support of F is $(0, \infty)$ and the sum is telescoping, we have

$$
\sum_{k=1}^{\infty} \left(F\left(\frac{k}{n}\right) - F\left(\frac{k-1}{n}\right)\right) = \lim_{k \to \infty} F\left(\frac{k}{n}\right) - F\left(\frac{0}{n}\right) = 1. \tag{5.18}
$$

Insert 5.18 into 5.17 and rearrange:

$$
\begin{aligned}
F_n\left(x\right) &= 1 - \sum_{k=1}^{\infty} \left(F\left(\frac{k}{n}\right) - F\left(\frac{k-1}{n}\right)\right) \sum_{i=0}^{k-1} \frac{e^{-nt}(nt)^i}{i!} \\
&= 1 - \sum_{i=0}^{\infty} \frac{e^{-nt}(nt)^i}{i!} \sum_{k=i+1}^{\infty} \left(F\left(\frac{k}{n}\right) - F\left(\frac{k-1}{n}\right)\right)
\end{aligned}
\tag{5.19}
$$

Applying the same rewriting in 5.18 on the last summation in 5.19 gives

$$\sum_{k=i+1}^{\infty}\left(F\left(\frac{k}{n}\right)-F\left(\frac{k-1}{n}\right)\right)=\lim_{k\to\infty}F\left(\frac{k}{n}\right)\;-\;F\left(\frac{i}{n}\right)=1-F\left(\frac{i}{n}\right). \tag{5.20}$$

Finally, inserting 5.20 into 5.19 gives

$$\begin{aligned}
F_n(x) &= 1-\sum_{i=0}^{\infty}\frac{e^{-nt}(nt)^i}{i!}\left(1-F\left(\frac{i}{n}\right)\right)\\
&= 1-\underbrace{\sum_{i=0}^{\infty}\frac{e^{-nt}(nt)^i}{i!}}_{=1}+\sum_{i=0}^{\infty}\frac{e^{-nt}(nt)^i}{i!}F\left(\frac{i}{n}\right)\\
&= \sum_{i=0}^{\infty}F\left(\frac{i}{n}\right)\frac{e^{-nt}(nt)^i}{i!}\\
&= \mathbb{E}\left(F\left(X_n\right)\right),
\end{aligned} \tag{5.21}$$

where X_n is the Poisson distributed random variable from before. $\sum_{i=0}^{\infty}\frac{e^{-nt}(nt)^i}{i!}=1$, since that corresponds to summing over all the probability mass in a Poisson distribution. Consequently, by 5.15, we get

$$F_n(t)=\sum_{k=0}^{\infty}F\left(\frac{k}{n}\right)e^{-nt}\frac{(nt)^k}{k!}\to F(t)\;\text{ for } n\to\infty. \tag{5.22}$$

In other words, an infinite mixture of Erlang distributions with the same setup as in 5.16 is dense in $(0,\infty)$. We will now show that this denseness property also holds for finite mixtures of Erlang distributions. This is achieved by truncating the infinite mixture above m/n and normalize it to obtain a distribution. By this procedure, we obtain the following mixture:

$$F_{m,n}(t)=F\left(\frac{m}{n}\right)^{-1}\sum_{k=1}^{m}\left(F\left(\frac{k}{n}\right)-F\left(\frac{k-1}{n}\right)\right)\left(1-\sum_{i=0}^{k-1}\frac{e^{-nt}(nt)^i}{i!}\right) \tag{5.23}$$

Define

$$T_{p,m}(t)=\sum_{k=p}^{m}\left(F\left(\frac{k}{n}\right)-F\left(\frac{k-1}{n}\right)\right)\left(1-\sum_{i=0}^{k-1}\frac{e^{-nt}(nt)^i}{i!}\right). \tag{5.24}$$

Re-arranging the terms and using the triangle inequality, gives

$$
\begin{aligned}
&\left|F_n(t) - F_{m,n}(t)\right| \\
&= \left|\lim_{m\to\infty} T_{1,m}(t) - F\left(\frac{m}{n}\right)^{-1} T_{1,m}(t)\right| \\
&= \left|\left(1 - F\left(\frac{m}{n}\right)^{-1}\right) T_{1,m}(t) + \lim_{p\to\infty} T_{m+1,p}(t)\right| \\
&\leq \left|\left(1 - F\left(\frac{m}{n}\right)^{-1}\right) T_{1,m}(t)\right| + \left|\lim_{p\to\infty} T_{m+1,p}(t)\right| \\
&= \left|\left(1 - F\left(\frac{m}{n}\right)^{-1}\right) \sum_{k=1}^{m} \left(F\left(\frac{k}{n}\right) - F\left(\frac{k-1}{n}\right)\right)\left(1 - \sum_{i=0}^{k-1} \frac{e^{-nt}(nt)^i}{i!}\right)\right| \\
&+ \left|\sum_{k=m+1}^{\infty} \left(F\left(\frac{k}{n}\right) - F\left(\frac{k-1}{n}\right)\right)\left(1 - \sum_{i=0}^{k-1} \frac{e^{-nt}(nt)^i}{i!}\right)\right| \\
&\leq \left|\left(1 - F\left(\frac{m}{n}\right)^{-1}\right) \sum_{k=1}^{m} \left(F\left(\frac{k}{n}\right) - F\left(\frac{k-1}{n}\right)\right)\right| + \left|\sum_{k=m+1}^{\infty} \left(F\left(\frac{k}{n}\right) - F\left(\frac{k-1}{n}\right)\right)\right|.
\end{aligned}
\tag{5.25}
$$

Inserting 5.20 into 5.25, gives

$$
\left|F_n(t) - F_{m,n}(t)\right| \leq \left|\left(1 - F\left(\frac{m}{n}\right)^{-1}\right) F\left(\frac{m}{n}\right)\right| + 1 - F\left(\frac{m}{n}\right) \to 0. \tag{5.26}
$$

Finally, we can conclude that there exists a finite mixture of Erlang distributions which is dense in $(0, \infty)$. In other words, every distribution with a positive support can, for m large enough, be well-approximated by a mixture of Erlang distributions. Since Erlang mixtures constitute a subclass of phase-type distributions, the denseness in $(0, \infty)$ holds for phase-type distributions as well.

This denseness property can easily be extended to include distributions with an atom in zero. A distribution with support on $[0, \infty)$ will have an atom at zero with some probability, say p. We can modify the mixture in 5.23 such that with probability p we have an atom at zero, and with probability $1 - p$, we have the mixture distribution in 5.23. Hence, this denseness property can further be generalized to $[0, \infty)$.

5.2 Phase-type approximation of theoretical functions

In last section it was proved that any distribution with a nonnegative support can be approximated arbitrarily closely by a phase-type distribution. Recall the following formula:

$$
F_{m,n}(t) = F\left(\frac{m}{n}\right)^{-1} \sum_{k=1}^{m} \left(F\left(\frac{k}{n}\right) - F\left(\frac{k-1}{n}\right)\right)\left(1 - \sum_{i=0}^{k-1} \frac{e^{-nt}(nt)^i}{i!}\right) \tag{5.27}
$$

From now on we will refer to equation 5.27 as the denseness formula. This is a finite mixture of Erlang distributions where the shape parameter, in the underlying Erlang distribution, sequentially increases with 1. An Erlang distribution with shape parameter k in this mixture has the corresponding weight $(F(\frac{k}{n}) - F(\frac{k-1}{n}))/F(\frac{m}{n})$. Recall that $F(\frac{m}{n})^{-1}$ is just a normalization constant. We will now give this Erlang mixture a phase-type representation.

The underlying Markov chain of an Erlang k-distribution consists of k phases which we process sequentially through. And since the shape parameters in the underlying distributions increase sequentially up to m, the minimal phase-type representation of this m mixture distribution requires $m(m-1)/2$ phases. The initial probability vector is given by

$$\begin{aligned}\boldsymbol{\pi} &= \frac{1}{F(\frac{m}{n})}\begin{pmatrix} F\left(\frac{1}{n}\right) & F\left(\frac{2}{n}\right) - F\left(\frac{1}{n}\right) & 0 & \dots & F\left(\frac{m}{n}\right) - F\left(\frac{m-1}{n}\right) & \mathbf{0}_{1\times(m-1)} \end{pmatrix} \\ &= \begin{pmatrix} \frac{F\left(\frac{1}{n}\right)}{F(\frac{m}{n})} & \frac{F\left(\frac{2}{n}\right)-F\left(\frac{1}{n}\right)}{F(\frac{m}{n})} & 0 & \dots & \frac{F\left(\frac{m}{n}\right)-F\left(\frac{m-1}{n}\right)}{F(\frac{m}{n})} & \mathbf{0}_{1\times(m-1)} \end{pmatrix} \\ &= \begin{pmatrix} \frac{F\left(\frac{1}{n}\right)}{F(\frac{m}{n})} & \frac{F\left(\frac{2}{n}\right)-F\left(\frac{1}{n}\right)}{F(\frac{m}{n})} & 0 & \dots & 1 - \frac{F\left(\frac{m-1}{n}\right)}{F(\frac{m}{n})} & \mathbf{0}_{1\times(m-1)} \end{pmatrix},\end{aligned} \tag{5.28}$$

which is a $m(m-1)/2$ dimensional row vector. Notice that

$$\begin{aligned}\sum_{k=1}^{m(m-1)/2} \pi_k &= \sum_{k=1}^{m} \frac{F\left(\frac{k}{n}\right) - F\left(\frac{k-1}{n}\right)}{F(\frac{m}{n})} \\ &= \frac{1}{F(\frac{m}{n})} \sum_{k=1}^{m} \left(F\left(\frac{k}{n}\right) - F\left(\frac{k-1}{n}\right)\right) \\ &= \frac{1}{F(\frac{m}{n})} \left(F\left(\frac{m}{n}\right) - F\left(\frac{0}{n}\right)\right) \\ &= 1,\end{aligned} \tag{5.29}$$

since the sum is telescoping. Hence, this initial probability vector is well defined. The sub-intensity matrix is given by

$$\boldsymbol{T} = \begin{pmatrix} \boldsymbol{T}_{1\times1} & \mathbf{0}_{1\times2} & \dots & \mathbf{0}_{1\times m} \\ \mathbf{0}_{2\times1} & \boldsymbol{T}_{2\times2} & \dots & \mathbf{0}_{2\times m} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{0}_{m\times1} & \mathbf{0}_{m\times2} & \dots & \boldsymbol{T}_{m\times m} \end{pmatrix}, \tag{5.30}$$

which is a $(m(m-1)/2) \times (m(m-1)/2)$ matrix, where the $k \times k$ sub-matrix $\boldsymbol{T}_k$ is given by

$$\boldsymbol{T}_k = \begin{pmatrix} -n & n & 0 & \dots & 0 \\ 0 & -n & n & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & -n \end{pmatrix}. \tag{5.31}$$

Consequently, the exit vector is given by

$$\boldsymbol{t} = \begin{pmatrix} n \\ 0 \\ n \\ 0 \\ 0 \\ n \\ \vdots \\ \mathbf{0}_{(m-1)\times 1} \\ n \end{pmatrix}. \tag{5.32}$$

Since computational time increases with the number of phases, the goal should be to make as accurate approximations as possible, and concurrently economize with the number of phases, used in the approximation. We will now examine how the level of m and n influence the approximation. From the formula we see that the approximation truncates the distribution, such that, probability mass above the support m/n is ignored. As a result of that, $F(k) = 1$ for $k \geq \frac{m}{n}$. Hence m should be set such that sufficiently probability mass is included in the approximation. Since the absolute value of m is not important, but rather the truncation or cut-off level of the distribution, this "truncation level" will instead be used as an input variable for the approximation. For now on, this input variable will be referred to as the cut-off level. To clarify, a cut-off level of $x\%$ means that $(100 - x)\%$ of the probability mass in the tail will be ignored. m is then an endogenous variable, related to the cut-off level by

$$m = [F^{\leftarrow}(\textit{cut-off level})n] + 1. \tag{5.33}$$

The reasoning for this is the following: If we want to use the approximation for tail probabilities of at least a percent, we need the cut-off level to be at least $1 - a$ percent such that m needs to be $[F^{\leftarrow}(\textit{cut-off level})n] + 1$ at least. Expressing 5.27 in terms of 5.33 gives

$$\begin{aligned} &F_{cl,n}(t) \\ &= F\left(\frac{[F^{\leftarrow}(cl)n] + 1}{n}\right)^{-1} \sum_{k=1}^{[F^{\leftarrow}(cl)n]+1} \left(F\left(\frac{k}{n}\right) - F\left(\frac{k-1}{n}\right)\right)\left(1 - \sum_{i=0}^{k-1} \frac{e^{-nt}(nt)^i}{i!}\right). \end{aligned} \tag{5.34}$$

Given the chosen value of m, we approximate the distribution over the interval $[0, m/n]$. Given that the aforementioned cut-off level is fixed, n is then the number of sub intervals we partition the interval into. Hence n is a parameter which adjust the smoothness of the approximation.

5.2.1 Phase-type Approximation of log-normal distribution

We will start out by approximating a log-normal distribution. All the code for visualizing the approximations are included in appendix A.

The log-normal distribution is interesting since it has a non-negative support and cannot be written as any finite combination of Erlang distributions. Consider $X \sim LN(0.5, 1.5)$ such that the density is given by

$$f(x) = \frac{1}{1.5x\sqrt{2\pi}} e^{-\frac{(\log x - 0.5)^2}{3}}. \tag{5.35}$$

On figure 5.1(a) on the left and 5.1(b) on the right, n is fixed at 5. The cut-off level, defined in 5.33, increases with 1.5 percent points from 93.5% (the brightest line) to 99.5% (the darkest line which is not dashed). The black dashed line is the lognormal distribution, defined above.

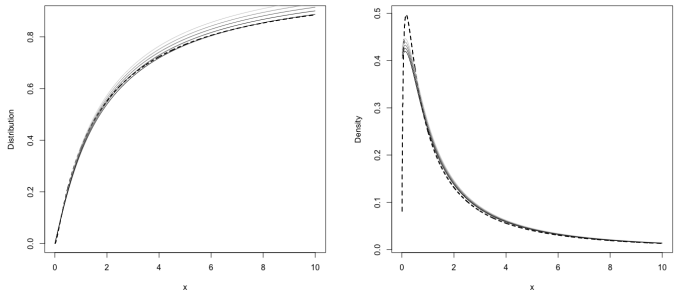


Figure 5.1: Adjustment of cut off level. n fixed at 5

The corresponding number of phases used for each approximation appears on table 5.1. The number of phases are very high compared to the number of phases typically used when for example phase-type approximations using the EM algorithm. We will comment on this issue leader in this section.

Table 5.1: No. of phases used in 5.1(a), 5.1(b) and 5.3(a). n fixed at 5.

Cut-off level	**Phases**
93.5%	3160
95.0%	4753
96.5%	7750
98.0%	16110
99.5%	77028

The number of phases increases rapidly when we increase the cut-off level, which can be explained by the quantile, which diverges to infinity as the cut-off level approaches 1 in 5.33. Since the number of phases increases quadraticly with m, this rapid increase in phases amplifies further when the cut-off level is raised. This is easily seen

from the formula below:

$$\begin{aligned} phases &= \frac{m(m-1)}{2} \\ &= \frac{([F^{\leftarrow}(\textit{cut-off level})n] + 1)[F^{\leftarrow}(\textit{cut-off level})n]}{2} \\ &= \frac{[F^{\leftarrow}(\textit{cut-off level})n]^2 + [F^{\leftarrow}(\textit{cut-off level})n]}{2} \end{aligned} \tag{5.36}$$

For example, the number of phases increases from 3160 to 4753 when the cut-off level increases from 93.5% to 94%, while the number of phases increases from 16110 to 77028 when the cut-off level increases from 98% to 99.5%. This increase is even more rapid when the distribution is heavy tailed.

As expected, the tail of the approximation concourses with the tail of the theoretical distribution, when the cut-off level increases. On the flip side, the approximation of the mode worsens, when the cut off level increases. The reason for that is that n is fixed at 5. Increasing the cut-off level increases the upper truncation point for the approximation which, furthermore, increases the length of the aforementioned sub-intervals, since the number of sub-intervals is fixed. Consequently, the smoothness of the approximation deteriorates.

If the cut-off level are fixed, this smoothness of the approximation increases with n, as shown on 5.2(a) and figure 5.2(b). On these two figures the cut-off level is fixed at 99% and n increases from 1 (the brightest line) to 5 (the darkest line). The corresponding number of phases appears on 5.2.

Table 5.2: No. of phases used in 5.2(a) and 5.2(b). Cut-off level fixed at 99%.

n	**Phases**
1%	1485
2%	5886
3%	13203
4%	23436
5%	36585

As the figures show when n, and thereby the number of phases is low, the curvature gets flatten out, meaning that the accuracy of the approximation is pour at places where the theoretical distribution exhibits volatile movements.

In the QQ plot 5.3(a) the approximating distributions are plotted against the theoretical distribution. The graphs in the QQ plot are based on the same level of n and m as in 5.1(a) and 5.1(b). The QQ plot shows that the theoretical distribution has a more right-skewed tail compared to the approximations. This deviation in the tail decreases as the cut-off level, and thereby the point of truncation, increases.

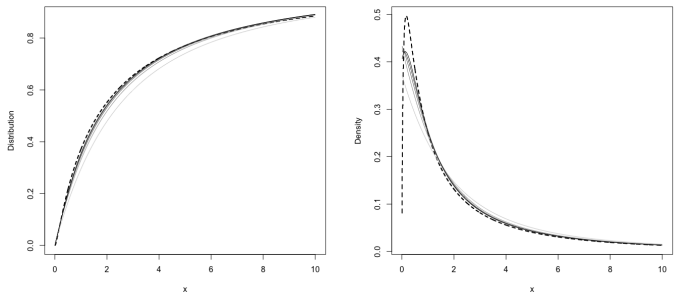
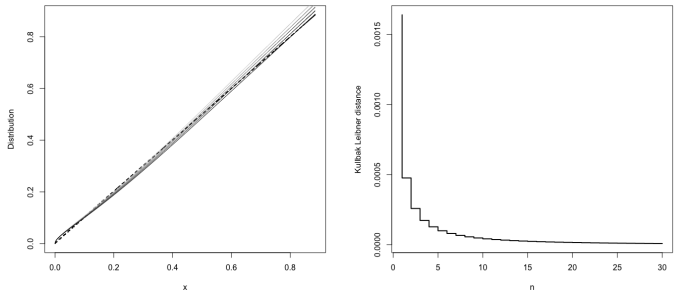


Figure 5.2: Adjustment of phases. Cutoff level fixed at 99 percent

On figure 5.3(b) the Kulbak Leibner distance between the theoretical distribution and the approximating distribution has been calculated using the package "Laplaces-Demon" in R. As in 5.2(a) and figure 5.2(b) the cut-off level is fixed at 99%. We also see that the Kulbak Leibner distance decreases with n, but at a decreasing rate, such that the gain in the fitness of the approximation decreases with n.



(a) QQ Plot. Adjustment of cut off level. n fixed at 5

(b) Kulbak Leibner distance between theoretical and aproximating distribution when n increases

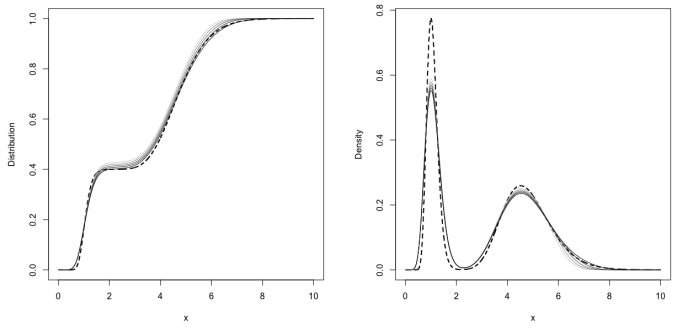
Figure 5.3: QQ plot on the left and Kulbak Leibner distance on the right

5.2.2 Phase-type Approximation of log-normal mixture

We will now test the flexibility of this approximation by approximating a multimodal distribution. Therefore, consider a mixture distribution with mixture weights 0.4 and 0.6 where the underlying distributions are given by $X \sim LN(0.05, 0.2)$ and $X \sim LN(1.55, 0.2)$ respectively. The density is then given by

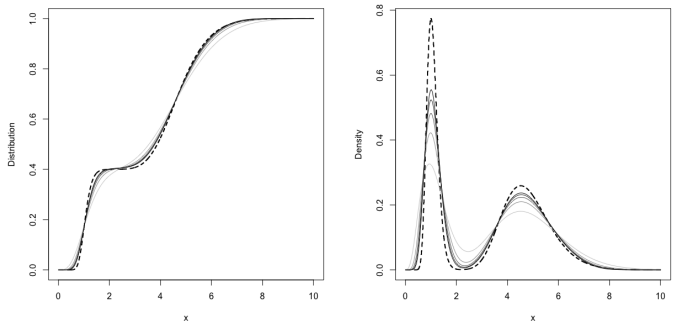
$$f(x) = 0.4 \frac{1}{0.2x\sqrt{2\pi}} e^{-\frac{(\log x - 0.05)^2}{0.4}} + 0.6 \frac{1}{0.2x\sqrt{2\pi}} e^{-\frac{(\log x - 1.55)^2}{0.4}} \tag{5.37}$$

On figure 5.4 same procedure as in last subsection has been carried out. On figure 5.4(a) and figure 5.4(b), n is fixed at 25 and on figure 5.4(c) and figure 5.4(d) the cut-off level is fixed at 99%. The corresponding phases used appear on 5.3 and on 5.4



(a) n fixed at 25

(b) n fixed at 25



(c) Cut off level fixed at 99%

(d) Cut off level fixed at 99%

Figure 5.4: Approximation of mixture of log-normal distributions

Even though the density can be considered more volatile compared to the distribution in the last subsection, it seems that we are able to get a better fit in less phases. The reason for that is that the previous distribution was more heavy tailed, which made the quantiles increasing more rapidly, and thereby the total number of phases increasing more rapidly, as the cut-off level increases. As expected, the first mode require a higher n than the second mode, since more volatile fluctuations in the theoretical distribution require a more smooth approximation. The downside of this is that we use an unnecessary fine smoothness (and as a result of the more phases) on the other mode.

Table 5.3: No. of phases used in 5.4(a) and 5.4(b). n fixed at 25.

Cut-off level	**Phases**
93.5%	11325
95.0%	12090
96.5%	13041
98.0%	14365
99.5%	18145

Table 5.4: No. of phases used in 5.4(c) and 5.4(d). Cut-off level fixed at 99%.

n	**Phases**
5%	666
10%	2628
15%	5866
20%	10440
25%	16290

5.3 Evaluation of the phase-type approximation and outlook

As seen, the denseness formula gives a good approximation but may be conceived as a tool of limited use, due to the high amount of phases needed for a good fit. But even though a high number of phases is needed, there is no issues regarding the execution time of the approximations. This is partly because, unlike the formula for general phase-type distributions in 3.20, no evaluation of matrix exponentials is necessary on this form. The phase-type approximations in the examples in last section were calculated in a split second, even when the number of phases increased to a million. Possible applications of the phase-type approximation will now be discussed.

Directly calculations of moments of arbitrary order can in some cases involve analytically tedious calculations of the integral. By inserting the corresponding initial probability vector and sub-intensity matrix in 5.28 and 5.30 in 3.24, we also get an approximation lth order moment numerically. Due to the many 0 entries in the initial probability vector and in the sub-intensity matrix, numerical calculation may be fast, even though the dimension is high.

As mentioned in section 4.6 integrals can also be calculated numerically by conversion into a weighted sum. A high upper limit and a high number of discretization points increases the precision of the weighted sum, but at the cost of computation time.

Analogous to the weighted sum, the phase-type approximation is based on some upper truncation point m/n and a given number of partitions n. When calculating moments, this phase-type approximation may be better in less partitions compared to the simple discretization approach. Same method may be applied to other functionals. Likewise, the Laplace transform can be approximated arbitrary close by inserting 5.28, 5.30 and 5.32 into 3.25.

The denseness formula may also be used to accelerate convergence in the Metropolis-Hastings algorithm. The Metropolis–Hastings algorithm is a Markov chain Monte Carlo method which makes it possible to obtain simulations from a distribution, referred to as the target distribution, from which simulating directly is difficult. After sufficiently many iterations, the Metropolis–Hastings algorithm will simulate from the target distribution. However, the Metropolis–Hastings algorithm needs simulations from a proposal distribution, whose distribution must be close to the target distribution, in order for the Metropolis–Hastings algorithm to converge fast. Simulating from the denseness formula can be done by simulating from a standard uniform distribution and then taking the generalized inverse to the simulations. Since the denseness formula gives good approximations, it can be a qualified candidate for a proposal distribution.

Under some circumstances the phase-type approximation the denseness formula may be used as an alternative to approximating continuous distributions with the EM algorithm, which appeared in section 4.6. Approximation by using the EM algorithm has shown to be specially slow for theoretical distributions, where the mode is located far from zero. An example of this issue is given in [32], where an approximation of a Weibull distribution with a huge delay is attempted. In this approximation, 25 phases where used. This phase-type approximation in the denseness formula may be used, when the structure of the theoretical distribution makes it too computationally demanding for using the EM algorithm.

5.4 Bibliographic references

The proof of denseness of phase-type distribution was first conducted in [35]. The proof in section 5.1 has also been inspired by [8]. Proof of Chebyshev's inequality was first carried out in 1867 in [37]. The pioneering work for the Metropolis-Hastings algorithm was first introduced in [26] and later extended in [21].

Chapter 6

Confidence regions for phase-type estimates

This section focuses on how to obtain confidence regions for estimates of parameters in phase-type distributions. The non-uniqueness of phase-type distributions is the main issue in this research field. Consequently, proposed approaches for obtaining confidence regions have been restricted to unique phase-type distributions. In this chapter a method for calculating confidence intervals which in most cases also can be applied to non-unique phase-type distributions will be proposed. This alternative approach involves combining bootstrap, introduced by Efron in [13], with the EM algorithm.

Before elaborating further on this method an introduction to bootstrap in section 6.1 is given. In section 6.2 a method for obtaining sample distributions of the estimates is introduced and summarized in an algorithm. Section 6.3 concerns the implementation of this method which is carried out in C by modification of an already existing program. Section 6.4 includes an analysis of the asymptotic results of the confidence bounds achieved by this algorithm. We end this chapter with and evaluation of this approach and compare it with existing methods.

6.1 Bootstrap

Suppose we have a sample $x_1, x_2, \ldots, x_n$ consisting of n observations from independent and identically distributed random variables $X_1, X_2, \ldots, X_n$. This sample distribution is used for estimation of some parameter $\boldsymbol{\theta}$. This could for example be $\boldsymbol{\theta} = (\boldsymbol{\pi}, \boldsymbol{T})$. Let λ denote one of the entries in $\boldsymbol{\theta}$. We want to obtain a confidence interval for λ. Letting F denote the true distribution of λ, this can be achieved by taking the $p/2$ and the $1 - p/2$ quantile from the distribution of λ to obtain a $1 - p$ two-sided centered p confidence interval. Suppose that the true distribution is continuous and $F^{-1}(b) > 0$, for $0 < b < 1$, where $F^{-1}(b)$ is the population quantile. Then for n large enough, the sample quantiles

q_b for an arbitrary $b \in (0, 1)$ are asymptotic normally distributed with

$$q_b \sim \mathcal{N}\left(F^{-1}(b), \frac{b(1-b)}{n\left[f\left(F^{-1}(b)\right)\right]^2}\right). \tag{6.1}$$

As seen, the variance converges to 0 and the mean equals the population quantile. Consequently, the sample quantile q_b is a consistent estimator of the population quantile $F^{-1}(b)$. Since the distribution of λ is not known, this approach is not a possibility. Applying Monte Carlo simulation to obtain a sample distribution of λ is neither an option, since that also require knowledge of the true distribution.

The bootstrap approach make it possible to obtain a sample distribution, even though the true distribution is unknown. Suppose we resample the observations, which means, drawing n observations with replacement from the n sized sample $x_1, x_2, \ldots, x_n$, and thereby obtaining $x_1^{(1)}, x_2^{(1)}, \ldots, x_n^{(1)}$. In other words we simulate n times from the empirical distribution of $x_1, x_2, \ldots, x_n$. This corresponds to simulating from a multinomial distribution with n trials and success parameters $1/n$ for each of the n categories $x_1, x_2, \ldots, x_n$. Repeating this procedure N times gives $x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}$ for $i = 1, 2, \ldots, N$. From these resamples we estimate the parameters $\lambda_1, \lambda_2, \ldots, \lambda_N$. As a result, we obtain a sample distribution of λ, denoted by F_N^λ, with N observations. The sample distribution of λ obtained is then the empirical distribution which put $1/n$ probability mass on each observation:

$$F_N^\lambda(x) = \frac{1}{N}\sum_{i=1}^{N} 1_{[\lambda_i, \infty)}(x) \tag{6.2}$$

The bootstrap principle then says the following:

- This true unknown distribution of λ is well-approximated by the empirical distribution F_N^λ.
- The variation of λ is well-approximated by the variation of $\lambda_1, \lambda_2, \ldots, \lambda_N$.

Based on this bootstrap principle we can get valid estimates of $F^{-1}(p/2)$ and $F^{-1}(1 - p/2)$. By taking the $p/2$ and the $1 - p/2$ quantiles, and thereby obtaining a $1 - p$ two-sided centered confidence interval, valid estimates will be achieved if they are based on sufficiently many simulations. These quantiles are approximately given by $[\lambda_{[N(1+p)/2]+1,N}, \lambda_{[Np/2]+1,N}]$ where $\lambda_{1,N} \geq \lambda_{2,N} \geq \cdots \geq \lambda_{N,N}$ is the ordered sample of $\lambda_1, \lambda_2, \ldots, \lambda_N$. Notice that the size of each resample $x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}$ is the same as the size of the original sample. The reason for that is, the variation of the statistic λ depends on the size of the sample. Resamples of the same size as the original sample is therefore necessary to approximate this variation.

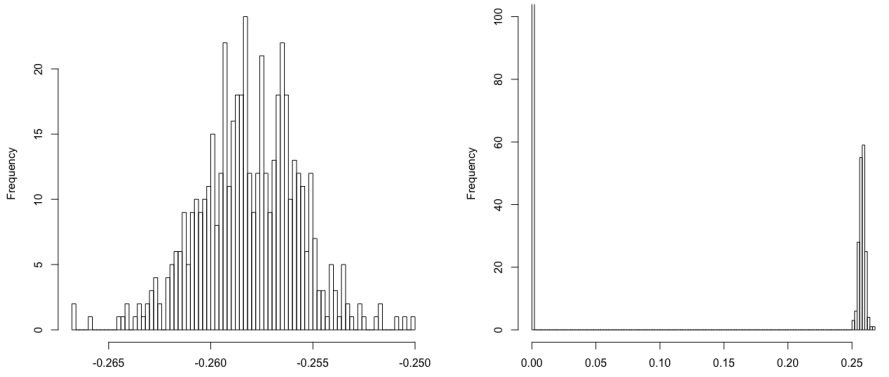
Usually we only simulate from the empirical distribution if the data comprise all the available information about the parameter. This is referred to as non-parametric bootstrap. If we know that the distribution of the parameter belongs to a family of parametric distributions we can use parametric bootstrap where we substitute the empirical distribution with the parametric distribution.

6.2 Combining bootstrap with the EM algorithm

In this section the bootstrap approach will be combined with the EM algorithm to obtain confidence regions for the estimated parameter values in phase-type distributions. A detailed description of the implementation will be given below.

Since we in the general case do not have knowledge about the underlying distribution, non-parametric bootstrap will be applied.

The question is how to implement this combination correctly. Consider the case where some arbitrary initialized parameter values $(\boldsymbol{\pi}_0, \boldsymbol{T}_0)$ are used, such that we alternate between resampling and running the EM algorithm with these initialized parameter values $(\boldsymbol{\pi}_0, \boldsymbol{T}_0)$ until convergence. As mentioned, one shortfall of the phase-type distributions is the issues concerning non-uniqueness. If we start out with this initialization for each simulation, we will most likely get different representations of the phase-type distribution. As a result of that the distribution of the estimates derived will not be well-defined. 6.1(a) and 6.1(b) illustrate this issue perfectly where this method has been tried out with 3 phases.



(a) Distribution of t_{11}

(b) Distribution of t_{12}. The second axis has been truncated at 100.

Figure 6.1: Distribution of entries in sub-intensity matrix using bootstrap approach with 500 simulations and random initialization

In the histograms the sample distribution of t_{11} and t_{12} in the sub-intensity matrix are depicted. Even though the sample distribution of 6.1(a) seems well-defined the histogram of 6.1(b) indicates that there exists a parameterization with $t_{12} = 0$, $t_{13} \neq 0$ and $t_1 \neq 0$ and another parameterization with $t_{12} \neq 0$ and $t_{13} = t_1 = 0$, since $\sum_{j=1}^{3} t_{ij} + t_i = 0$ for $i = 1, 2, 3$. Furthermore, since the EM algorithm requires quite many iterations for convergence, running each simulation with some randomly initialized parameter values $(\boldsymbol{\pi}_0, \boldsymbol{T}_0)$ until convergence will be very time-consuming. Finally running the EM algorithm on different resamples may result in global maxima reached in some of the

simulations and local maxima or saddle points reached in other simulations. Therefore random initialization between each simulation is not even an optimal approach for unique distributions and confidence regions may not be well-defined.

Suppose instead that we have some maximum likelihood estimated parameter values. Let $(\boldsymbol{\pi}_M, \boldsymbol{T}_M)$ denote these parameter values. These parameter values will be used as starting values for each simulation instead of $(\boldsymbol{\pi}_0, \boldsymbol{T}_0)$.

The estimated sub-intensity matrix and initial probability vector $(\boldsymbol{\pi}_M, \boldsymbol{T}_M)$ will most likely contain a structure with zeros in some entries. These zero entries will be organized in a way that makes alternative representations of the simulated phase-type distributions unlikely, since entries with 0s will be preserved in the subsequent iterations when running the EM algorithm. Hence, utilizing this property of the EM algorithm is exactly what makes it possible to achieve these confidence bounds, even though the distributions considered are non-unique. For the same reason implementing this bootstrap approach into Markov chain Monte Carlo estimation of phase-type distributions is not achievable since 0 entries in this estimation technique are not preserved.

Convergence should furthermore be possible in significantly fewer iterations with these starting values $(\boldsymbol{\pi}_M, \boldsymbol{T}_M)$ for two reasons. The starting values $(\boldsymbol{\pi}_M, \boldsymbol{T}_M)$ used in the EM algorithm maximizes the incomplete log likelihood function of the original sample. Since there exists some similarity between the original sample and the resamples we most likely initialize in a point close to the global maximum. The structure of the converged estimates $(\boldsymbol{\pi}_M, \boldsymbol{T}_M)$ will most likely consist of many 0 entries which will speed up the convergence. We will therefore presume that these starting values $(\boldsymbol{\pi}_M, \boldsymbol{T}_M)$ will "catalyze" the convergence. The three issues regarding non-uniqueness, computational time and local maxima or saddle points can therefore be improved by using these starting values.

The algorithm with m_i iterations for simulation i is summarized below:

0. Run the EM algorithm on the original sample $x_1, x_2, \ldots, x_n$ for different initialized parameter values $(\boldsymbol{\pi}_0, \boldsymbol{T}_0)$ to identify the global maximum of the incomplete log-likelihood function ℓ. Let $(\boldsymbol{\pi}_M, \boldsymbol{T}_M)$ denote the parameter values which maximizes the incomplete log-likelihood function. Set $i := 1$.

1. Resample the observations. Let $x_1^{(i)}, x_2^{(i)}, \ldots, x_n^{(i)}$ denote the resample. Run the EM algorithm on the resample. Set $(\boldsymbol{\pi}^{(i)}, \boldsymbol{T}^{(i)}) := (\boldsymbol{\pi}_M, \boldsymbol{T}_M)$

2. (Expectations step) Calculate $\ell(\boldsymbol{\pi}^{(i)}, \boldsymbol{T}^{(i)}) = \mathbb{E}_{(\hat{\boldsymbol{\pi}}^{(i)}, \hat{\boldsymbol{T}}^{(i)})}\left(\ell_c\left((\boldsymbol{\pi}, \boldsymbol{T}); X\right) \mid \boldsymbol{Y} = \boldsymbol{y}\right)$, which reduces to the conditional sufficient statistics in section 4.2 and 4.3.

3. (Maximization step) Let $(\hat{\boldsymbol{\pi}}^{(i)}, \hat{\boldsymbol{T}}^{(i)}) := \operatorname{argmax}_{(\boldsymbol{\pi}^{(i)}, \boldsymbol{T}^{(i)})} L(\boldsymbol{\pi}^{(i)}, \boldsymbol{T}^{(i)})$

4. If $k = m_i$ then set $i := i+1$ and GOTO 1. Else set $k := k+1$, $(\boldsymbol{\pi}^{(i)}, \boldsymbol{T}^{(i)}) := (\hat{\boldsymbol{\pi}}^{(i)}, \hat{\boldsymbol{T}}^{(i)})$ and GOTO 2.

We will refer to this algorithm as the bootstrap algorithm for the rest of this chapter.

6.3 Implementation

The bootstrap algorithm listed in the end of last section will now be implemented. Important results in some of the coming sections are achieved by modification and extension of an existing program, called the EMpht program. The code of the modified program is included in appendix B, together with R-code used for visualization of the results. All non-used subroutines and functions from the original EMpht program has been removed.

The modified program consist of two components - one which concerns the model selection, corresponding to the initialization step in the bootstrap algorithm, and another which concerns the simulations. By model selection we mean the process of finding the optimal estimates, which subsequently will be used as starting values for the simulation process. The motive behind this setup is to separate the issue of finding a global maximum, with the issue of deriving sample distributions of the estimated parameter values. Better understanding on how these results are achieved necessitates an introduction to the original EMpht program. A description of the EMpht program is contained in this section together, with a description of some of the modifications made in order to implement the algorithm.

First subsection describes the overall structure of the original program, its capabilities and its limitations. The subsequent subsections 6.3.2 and 6.3.3 are denoted these two components. This section ends with a demonstration of the modified program in subsection 6.3.4.

6.3.1 The original EMpht program

The EMpht program is capable of fitting phase-type distributions to both sample data and theoretical distributions by using the EM algorithm. Each scenario is treated separately below.

Sample data

The program allows the sample data to be censored and weighted. Some restrictions are put on the structure of the sample data in order to execute the program successfully. First of all, the input data needs to be stored in a text file located in the same folder as the program. The text file must end on a negative value (convention is -1), indicating the end of the dataset, since the program will continue to read records until it finds a negative valued record. Furthermore the variables in each records have to be separated by spaces. The requirements for the structure of the sample data also depend on whether the data is complete and whether the data is weighted.

Assume that the data is unweighted. The input file must then be named "unweighted". If the data is furthermore uncensored, the file must only consist of one column with the observed absorption times. For censored and unweighted data the first column is denoted to a logical variable with the values 0, 1 or 2, indicating whether the data is uncensored, right censored or interval censored respectively. Next column is then denoted to the absorption times, or minimal absorption times, depending on whether the data is uncensored or right censored respectively. If the data is interval

censored, the next two columns are denoted to the lower and upper limit of the absorption times. The program will read the input file “unweighted” and create a new file named "sample", almost identical to the original file with the exception that there has been added another column consisting of only 1s. The reason for that is that the program treats the scenario with unweighted data as a special case of the scenario with weighted data, where each record has been assigned the weight 1.

Now assume that the data is weighted. Then the input file must be named "sample". The requirements of the structure of the input data is almost identical to the scenario with unweighted data, with the exception that there has to be added a field in the end of each record, consisting of the weight which belongs to the corresponding absorption time.

Approximation of distributions

As mentioned above, the program is also capable of fitting phase-type distributions to theoretical distributions. However, the range of theoretical distributions available is limited to a few pre-programmed theoretical distributions. The procedure of user-defined input distributions depends on the type. If the user-defined distribution is of phase-type, an initial probability vector and sub-intensity matrix need to be specified, but coding is avoided. If we want to fit other non-phase-type distributions, we have to program them ourselves.

The program uses an algorithm, equal to the one presented in section 4.6, and calculates the integrals numerically by simple discretization. Consequently, the length between the discretization points have to be specified. The preciseness of the numerically approximations of the integrals increases when the length between the discretization points narrows down but at the cost of extra computational time. Since conversion to a weighted sum requires and upper truncation point, this have to be specified as well.

Fitting procedure

We can choose to fit the sample data or the theoretical distribution with a general phase-type distribution or some special case of a phase-type distribution. These special cases are handled by initiating the sub-intensity matrix with some specific entries, preset to 0. It is also possible to make a user-defined initiation of the initial probability vector and the sub-intensity matrix just by entering the values we want in the given entries or by loading an input file. The structure must be saved in an text-file named "distrtype”. First column in the input file are denoted to the initial probability vector, and the remaining columns are denoted to the sub-intensity matrix. The file must only consist of 0s and 1s. However a diagonal entry in the sub-intensity matrix set to 0 means that we want a 0 in the corresponding entry in the exit vector. For example in case we want to fit an Erlang distribution we will preset the entries to:

$$\begin{pmatrix} 0 & 1 & 0 & \dots & 0 \\ 0 & 0 & 1 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{6.3}$$

This specification puts restrictions on the overall structure of the sub intensity matrix and the initial probability vector since entries preset to 0, will remain 0 when the EM algorithm is running. Given that a structure consisting of 0s and 1s have been chosen, the program will then ask for a random integer. This is a seed, specifying random starting values in the initial probability vector and the sub-intensity matrix.

It is also possible to load some pre-specified starting values of the parameters. User-defined starting values can be done by loading a text file named "phases" or by manually entering them on the keyboard. The set up and the location of the input file "phases" are analogous to the set up of "distrtype". However, values different from 0 and 1 are obviously allowed in this file.

The matrix exponentials are handled numerically by the Runge Kutta method where the differential equations are solved in fourth order. It is possible to choose between a default or user-defined step length in the Runge Kutta procedure. Default step length is 0.1 divided by the highest numeric value of the diagonals in the sub-intensity matrix. A higher step length decreases computational time, but at the cost of numeric precision.

During the process of fitting, the program will display the log-likelihood value and the estimated parameter values for the first five iterations, for each 25th iteration and for the last iteration. The program will export an output file named “phases”, containing the estimated parameters. Notice from the options specified above that these parameter values can be used directly as starting values in a new fit. Furthermore, if the observations were uncensored then a file named “inputdistr” containing either a sample or a discretized density is exported. The two exported files are used as input for the program “PHplot.m” in Matlab which visualizes the results in graphs.

6.3.2 Model selection

Recall that first component of the modified program deals with the model selection which can be separated into two distinct issues; 1. Finding the optimal number of phases and 2. ensuring that the maximum found, within the given number of phases, is a global maximum. Some theoretical background regarding these two issues are introduced below, together with a description of the setup in this part of the program.

Information criteria

As part of the initialization step, the optimal number of phases used to fit the model will be identified. To compare the different models a measure, referred to as an information criterion, that rewards for the preciseness of the fit and penalizes for the number of free parameters is used. A relatively low information criterion will indicate a better model.

That necessitates some clarification of the definition of free parameters. One could argue that if the estimated parameters of the phase-type distribution represents for example an Erlang distribution then the number of free parameters comprise 2, one for the form parameter and one for the scale parameter in the Erlang distribution. However, comparison of the models will not be within this subclass of phase-type distributions, but in the general framework. For that reason, all free parameters of the general phase-type distribution will be included in the penalization term even though the fitting distribution considered constitute a subclass. More specifically the number of free parameters amounts to:

- $p - 1$ for the initial probability vector. One entry in $\boldsymbol{\pi}$ is fixed since the probabilities have to sum up to 1.
- $p^2 - p$, corresponding to the number off off-diagonal in the sub-intensity matrix.
- p, corresponding to the number of entries in the exit vector.

This amounts to $p^2 + p - 1$ free parameters in total.

We will, among other, use the Akaike information criterion as an assessment tool for the models. The AIC is given by

$$\text{AIC} = 2k - 2l = 2(p^2 + p - 1) - 2l, \tag{6.4}$$

where k is the number of free parameters and l is the log-likelihood function. We will include the Bayesian information criterion as a complementary tool in the model selection which is given by

$$\text{BIC} = \ln(n)k - 2l = \ln(n)(p^2 + p - 1) - 2l, \tag{6.5}$$

where n is the number of observations.

Identification of global maximum

Different random initializations in the EM algorithm are necessary to confirm that the stationary point found is in fact global. This procedure should be carried out for each model.

However in some scenarios different random initializations may be less important than others. When the number of phases is low the log likelihood function will as a function of the parameters $\boldsymbol{\theta}$ most likely be baldy and unimodal. Strassen showed that an Erlang density is a lower bound for convex functions such that for all convex functions g it holds that

$$\mathbb{E}(g(X)) \leq \mathbb{E}(g(Y)) \tag{6.6}$$

where X is an Erlang distributed random variable. If the log likelihood function is unimodal in the parameters, $\boldsymbol{\theta}$ then the log likelihood function is concave such that $\mathbb{E}(-l(\boldsymbol{\theta}))$ is convex. Minimizing should therefore result in an Erlang distribution. Bottomline is that if fitting with a low amount of phases results in an Erlang distribution this indicates that there might only be one stationary point, which also is the global

maximum. Redundant search for local maxima can therefore be avoided in some occasions. However notice that a multimodal log likelihood function does not exclude that the fitting distribution converges to an Erlang distribution. Therefore cautions should be made before drawing any conclusion from this reverse causality.

Setup

Before using the modified program, some preparatory work is necessary. An extension of the program to weighted and censored data is not relevant for the issues presented in this chapter. The applicability of the modified program has therefore been restricted to unweighted and uncensored data. Consequently, the input data file must be a text-file named "unweighted" which must be located in the same folder as the program. The input file must only contain one column, consisting of the absorption times and a negative value at the end of the dataset. This also holds for the simulation part of the program.

The model selection component of the modified EMpht program is build up the following way. When the program is run, the user is presented with the choice of either pressing 1 for initializations or 2 for simulations. When pressing 1 for initializations, the user is asked whether he or she wants to overwrite a file named "loglik", whose contents will be described in a bit. The user is then requested to specify the number of phases and a seed, just as in the original EMpht program. To ease the implementation, step-length and structure are restricted to the default length and the general phase-type structure. Each time this initialization step is run, a record will be written to the file "loglik", which contains information about the log likelihood function, the number of phases, the number of iterations, the seed and the information criteria AIC and BIC. Specifying yes to overwrite the "loglik"-file, will clear the file before a new record is written.

The "loglik"-file gives the user the opportunity to compare models of different choices of phases and random initialization, specified by the seed. The issues regarding the optimal number of phases can be handled by considering the information criteria in the file. Within a given number of phases, the user can identify local maxima by comparing log likelihood values for different seeds.

6.3.3 Simulation procedure

Below the resampling procedure is described which is an essential part of the simulation mechanism. Some issues regarding optimization of the execution time necessitates some clarification of convergence. These issues are also introduced and dealt with below. Finally, a description of the setup of this part of the program is included in the end of this subsection.

Resampling procedure

Recall that the EMpht program assigns the weight 1 to each record and treats the unweighted data case as a special case of the weighted case where each weight equals 1. This setup has been utilized in the resampling algorithm, programmed in the modified

version. Instead of drawing observations randomly from the observation vector in the beginning of each simulations, the modified program manipulates the weight vector by first initializing the weight vector with 0s in each entry and then adding 1s to random entries n times, where n is the number of observations in the original sample. Notice that this procedure is equivalent to simulating n times from the empirical distribution, as intended. An example of this process is illustrated below.

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ \vdots \end{pmatrix} \rightarrow \begin{pmatrix} 4 \\ 0 \\ 1 \\ 2 \\ \vdots \end{pmatrix} \tag{6.7}$$

This setup is optimal for implementation, due to some technical prerequisites in the Runge Kutta fitting procedure in the original program. Different simulations are achieved by taking a random seed before each resample which is necessary to avoid identical simulations. It is still possible to recreate exactly the same simulations by specifying a seed in the beginning, since this seed will determine how the remaining seeds are randomized.

Precision level of convergence

The next issue regards how many iterations are necessary for convergence. Assume that the algorithm has been implemented on some smaller number of simulations, in order to determine the number of iterations necessary for each simulation to converge. The simulation which were slowest in convergence will then give an upper bound for the number of iterations. Using this upper bound as the number of iterations for each simulation is easier to implement, but not time optimizing. Many redundant iterations will be used on the remaining simulations to ensure convergence which gives a dilemma regarding the execution time, analogous to the proverb that "a chain is only as strong as its weakest link". Since issues regarding execution time is crucial in this implementation this aforementioned setup will not be used.

Instead a specification of the preciseness of convergence is needed. The user is asked to specify a precision level. A precision level of x means that the EM algorithm will keep iterating until the absolute change in any of the estimated parameter values from one iteration to another is less than 10^{-x}. The number of EM iterations must also be specified which will serve as an upper bound. If the given simulation does not reach that level of precision within this specified number of simulations resampling will occur.

One shortfall of this approach is that a temporarily stationary mode can be mistaken with convergence. These scenarios may be reduced by requiring that this precision level have to be fulfilled for multiple consecutive iterations.

Normally, convergence would be specified in terms of the log-likelihood function. However, it is still possible for the parameter values to exhibit volatile movements for some given precision of the log-likelihood function. Specifying the precision in terms of the parameter values serves the purpose of ensuring, that stable parameter values are used in the resulting sample distribution.

Setup

Recall that the EMpht program exports the estimated parameter values in a file named "phases". The modified program exploits this feature and uses this file as input for each initialization. The two components can therefore be run in continuation of each other without intermediate preparation if they are located in the same folder.

When running the program, the user must press 2 for simulations when the start menu appears. Subsequently, the program asks for specification of the number of simulations and the aforementioned precision level, in the given order. The program will then ask for specification of the number of phases used in the imported file, a seed and an upper bound for the number of EM iterations. While the program is running the temporarily estimated parameter values will be displayed together with the number of simulation reached, the number of iterations reached for the given simulation and finally the temporarily precision level of convergence reached for the given simulation. The aforementioned "loglik"-file will not be created when running simulations.

The modified program exports 2 files named "phasesPi" and "phasesT", containing the fitted values of the initial probability vector and the sub-intensity matrix respectively. When each simulation has ended, the program writes a new record in the end of the two files, containing the fitted values. Assuming that the program has been run on 4 phases, row 4000 and column 7 then contain the estimated value of $t_{2,3}$ from the 4000th simulation. The program will export a file named "iterations" which contains the number of iterations used for each simulation. This gives the possibility of discarding simulations for which the number of iterations reached the upper bound.

6.3.4 Demonstration of program

In order to demonstrate the program, it has been run on a dataset consisting of fire insurance claims in millions. The data has a lower truncation of 1 million, probably due to the deductibles. For that reason the data values have been subtracted by 1, and 0 values have subsequently been removed. The data is quite heavy-tailed which will slow down the convergence. For that reason any claim above 7 million, which corresponds to observation values larger than 6, have been removed. A histogram of the manipulated data is shown below.

Demonstration of model selection

Phase-type distributions of 1, 2, 3 and 4 phases have been fitted to the data. By the use of the exported "loglik"-file, the information criteria for each model appear next to each other on figure 6.3(b). It is seen that AIC is minimized for 3 phases while BIC is minimized for 1 phase, corresponding to an exponential distribution. The discrepancy of the two measures follows from the fact that BIC penalizes harder for free parameters. While AIC penalizes with a factor of 2 for each free parameter, the corresponding factor of BIC is ln(1999) = 7.6 (data consists of 1999 observations).

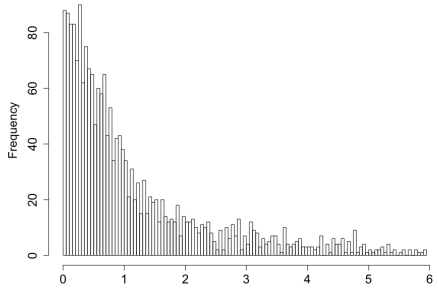


Figure 6.2: Histogram of sample data

On figure 6.3(a) the empirical distribution of the sample data (black line) and the fitting distributions (grey lines) have been plotted for 2 and 3 phases. The two graphs for 2 and 3 phases are however almost coinciding.

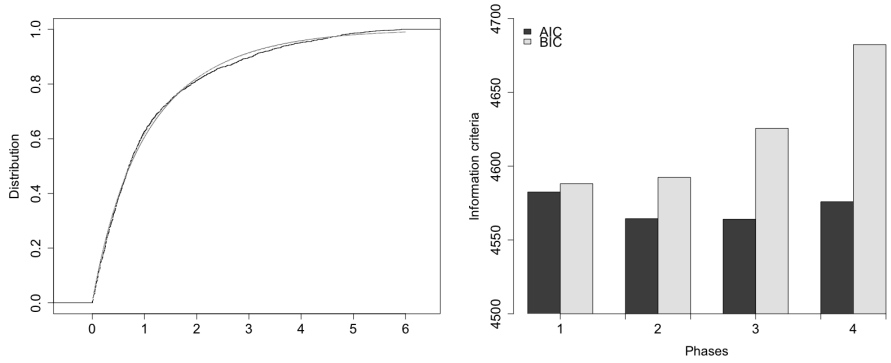


Figure 6.3: Empirical and fitting distributions on the left. Information criteria for each model on the right.

One can argue that using 2 phases is optimal since the average of these two information criteria then will be minimized. We will however continue with 3 phases since this model suits better for illustration of the simulation mechanism. This model have the following phase-type representation:

$$(\boldsymbol{\pi}_M, \boldsymbol{T}_M) = \left(\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} -2.68573 & 0 & 1.15637 \\ 1.83709 & -2.68573 & 0 \\ 0 & 1.55736 & -1.55736 \end{pmatrix} \right) \tag{6.8}$$

Demonstration of simulation procedure

We will now use the modified program to obtain sample distributions of the estimated parameter values in 6.8. 500 simulations have been run with these starting values. A precision level of 8 digits, together with an upper bound of 50,000 iterations, have been used. The histogram in 6.5(a) shows how many iterations were used for convergence. 3 simulations did not reach convergence in 50,000 iterations and have therefore been discarded.

The sample distribution of each entry in the sub-intensity matrix are depicted on figure 6.4(a). The vertical dashed line and the two dotted lines correspond to the estimated parameter value and the bounds in a 95% two-sided centered confidence interval, respectively. On figure 6.4(a) a histogram in the ith row and the jth column corresponds to the sample distribution of entry t_{ij}. It is clearly seen that the sample distribution of each entry is well-defined, although, some graphs may look strange due to a deterministic distribution. Since the estimated values are used as starting values, each simulation will keep converging towards 0 in these entries, which explains why starting values in the "0" entries are a little higher that the simulated values.

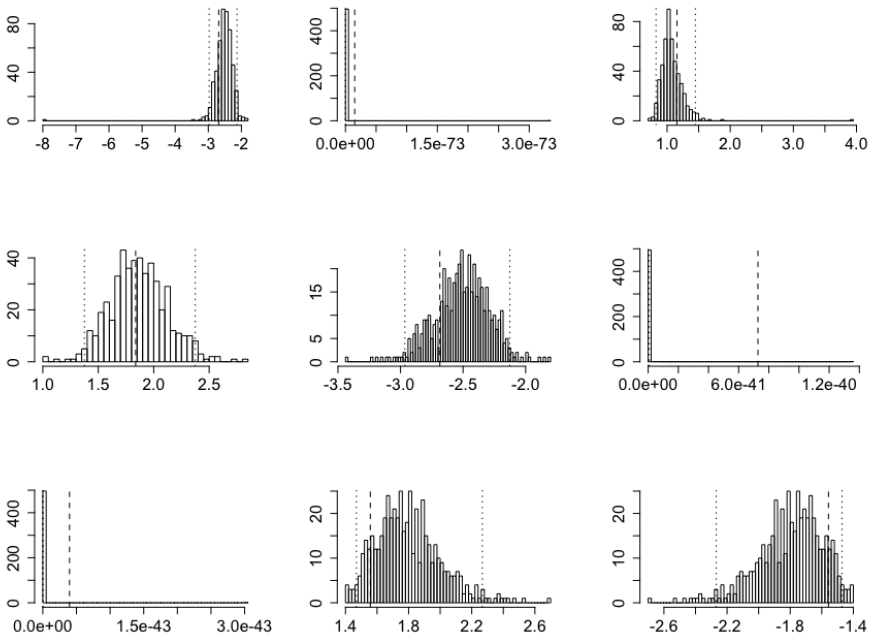


Figure 6.4: Sample distribution of each entry in sub-intensity matrix

Since all the entries in the initial probability vector are deterministic, these sample distributions will not be shown.

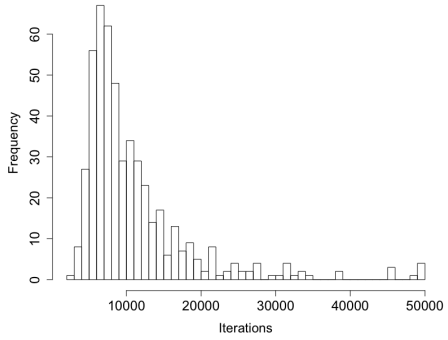


Figure 6.5: Number of iterations used for convergence for each simulation

On figure 6.6(a) the sample quantiles are plotted for each entry. Analogous to 6.4(a) the dashed line and the dotted lines shows the estimated parameter values and the 2.5% and 97.5% quantiles.

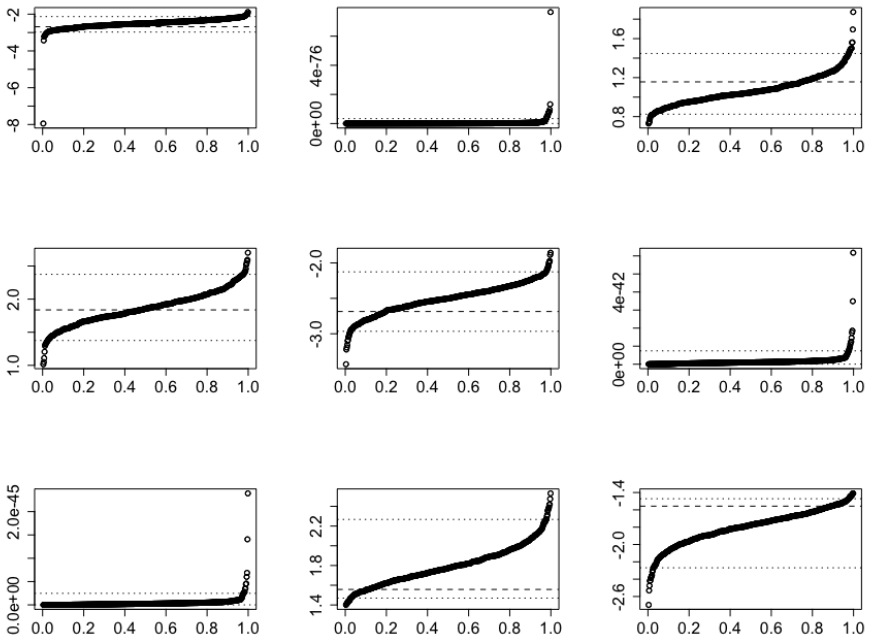


Figure 6.6: Sample quantiles of each entry in sub-intensity matrix

6.4 Analysis of asymptotic properties

Using the modified EMpht program enable us to gain insight in the asymptotic properties of the estimates.

For this analysis a dataset, where convergence happens in few iterations, is necessary. We will consider 1000 observations, simulated from an Erlang distribution. The original EMpht program has been used to fit a phase-type distribution with 3 phases, and the following estimated parameter values where obtained:

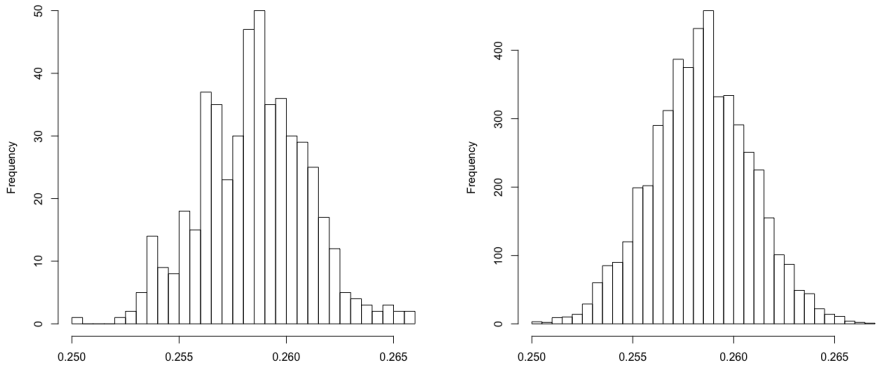
$$(\boldsymbol{\pi}_M, \boldsymbol{T}_M) = \left(\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}, \begin{pmatrix} -0.25578 & 0 & 0 \\ 0 & -0.25578 & 0.25578 \\ 0.25578 & 0 & -0.25578 \end{pmatrix} \right) \tag{6.9}$$

The simulations from these starting values were able to converge within 1000 iterations which made them ideal for this implementation.

The asymptotic normality of the estimators, and the consistence of the sample quantiles, are dealt with distinctly below.

Asymptotic normality

An example of entry $t_{3,1}$ of the sub-intensity matrix is given at figure 6.7(a) and 6.7(b). According to 6.1, the sample distribution converges to a normal distribution when the number of simulations goes to infinity. Figure 6.7(a) and figure 6.7(b) give an indication that this might be true, by the change of the shape of the densities when the number of simulations increases from 500 to 5000.



(a) 500 simulations

(b) 5000 simulations

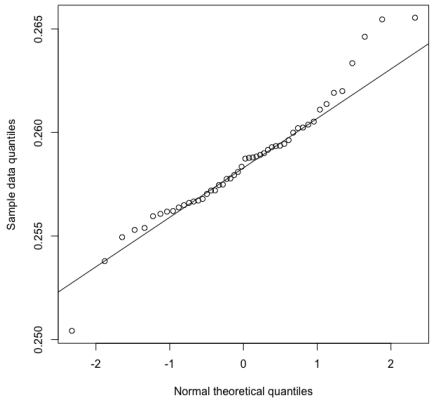
Figure 6.7: Distribution of t_{31} without initialization between simulations

This is also confirmed by the QQ plots on figure 6.8 where the theoretical quantiles of the standard normal distribution has been plotted against the sample quantiles. For

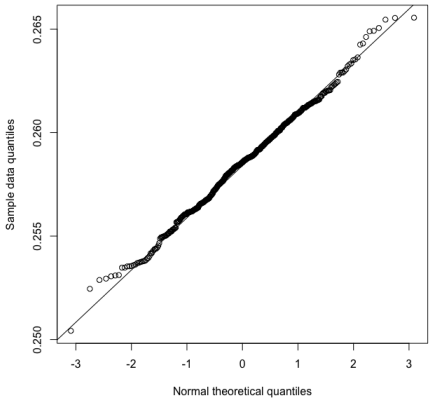
a few simulations the sample distribution seems to be more heavy-tailed, compared to the normal distribution. However, this tendency decreases as the number of simulations increases. The QQ plots also show that quantiles in the tail, meaning confidence bounds close to 0 or 1, require many simulations, compared to confidence bounds in the middle.

Consistency

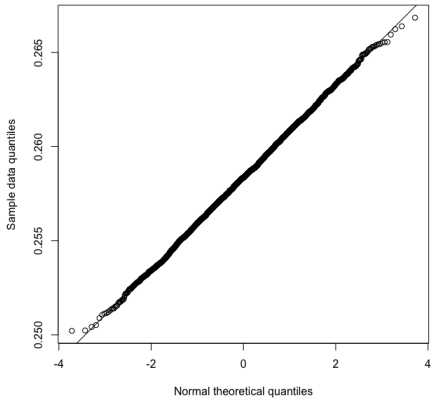
Next, consider a two-sided centered 95% confidence interval t_{31}. The lower and upper confidence bound, denoted by $q_{2.5\%}$ and $q_{97.5\%}$, are plotted on figure 6.9(a) and figure 6.9(b), as a function of the number of simulations.



(a) 50 simulations



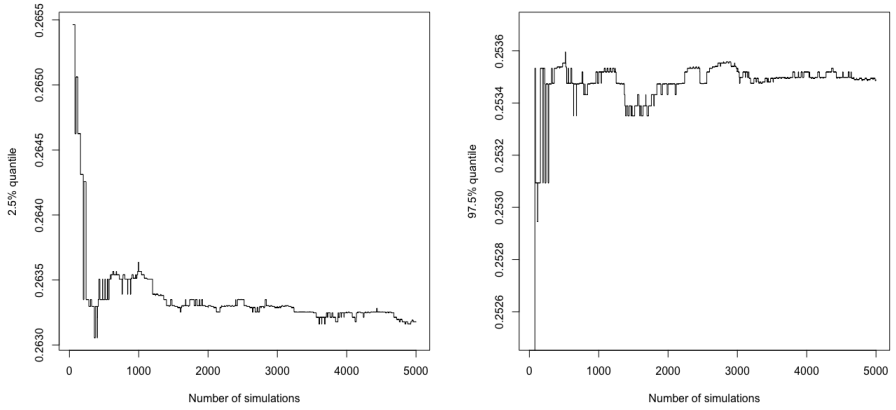
(b) 500 simulations



(c) 5000 simulations

Figure 6.8: QQ plot of sample data with standard normal distribution as reference distribution

As seen on the figures, the graphs exhibit fluctuations for a low number of simulations, but seems to stabilize as the number of simulations increase. The same holds for the bandwidth of the confidence interval, as seen on figure 6.10(a) and figure 6.10(b) respectively.

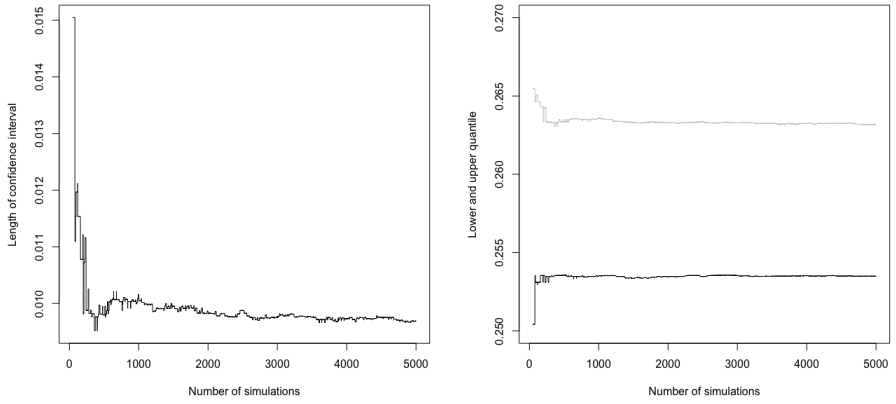


(a) 2.5% quantile

(b) 97.5% quantile

Figure 6.9: Lower and upper confidence bounds

The estimated sample quantiles $q_{2.5\%}$ and $q_{97.5\%}$ of t_{31} seems to be in accordance with the results in 6.1.



(a) Absolute difference between upper and lower confidence bound

(b) Upper (bright line) and lower confidence bound respectively

Figure 6.10: Size of confidence interval

The figures confirm indisputably that the estimates of the parameter values in this given example are asymptotic normally distributed, and that the quantiles are consistent. Given the assumptions about the bootstrap principle hold, we are 95% confident that the estimated value of t_{31} lies in the interval

$$[\{t_{31}\}_{[5000*95\%/2]+1,5000}, \{t_{31}\}_{[5000*(1-95\%)/2]+1,5000}] = [0.2535, 0.2632]. \tag{6.10}$$

6.5 Evaluation of model and outlook

This section accommodates an introduction to the common method for calculating confidence regions for phase-type estimates. A comparison of the bootstrap approach with the existing method is included afterwards.

Statistical inference using the asymptotic covariance matrix

The most frequently used methods are based on the asymptotic property of maximum likelihood estimates. This method also includes estimates, obtained by the EM algorithm, since the EM algorithm is an iterative procedure for obtaining the maximum likelihood estimates. In maximum likelihood theory the Fisher information matrix is obtained by differentiating the log likelihood function $l(\boldsymbol{\theta})$ twice with respect to the parameter vector $\boldsymbol{\theta}$ and then taking the negative expected value.

$$\mathcal{I}(\boldsymbol{\theta}) = -\mathbb{E}\left(\frac{\partial^2}{\partial^2 \boldsymbol{\theta}} l(\boldsymbol{\theta})\right) \tag{6.11}$$

The Fisher information matrix can be used to derive the asymptotic distributions of the estimates, by using the property of asymptotic normality of maximum likelihood estimates.

$$\boldsymbol{\theta} \sim \mathcal{N}\left(\boldsymbol{\theta}_T, \mathcal{I}(\boldsymbol{\theta})^{-1}\right) \tag{6.12}$$

From the asymptotic normality, the confidence region can then be derived.

Another possible way of obtaining the Fisher information matrix analytically is by the Newton-Raphson approach where the log likelihood function is approximated by a Taylor expansion of first or second order. Calculation of the gradient vector of the log likelihood function can be computationally demanding, but convergence may happen in fewer iterations. The Newton-Raphson is not designed to work with constraints, but there is a way around that.

The Fisher information matrix of phase-type distributions have so far been derived, and expressed in terms of the derivatives, when the sample only consists of uncensored observations. The Fisher information matrix have both been calculated directly, by the approach in 6.11, and by the Newton-Raphson method. The Fisher information matrix can then be used in connection with the EM algorithm.

Non-uniqueness

Due to non-uniqueness of the phase-type distributions the approach, by using the asymptotic covariance matrix, is restricted to a subclass, for which the representation is an acyclic graph. It has been shown in [11] that these types have an unique representation, in the form of a Coxian model, see 3.47 and 3.48 for more.

The bootstrap approach will therefore be an obvious alternative for phase-type distributions that does not constitute this subclass. However, even though well-defined confidence intervals are most likely to occur when using this bootstrap method, the possibility that there exist different parameterizations within the converged starting values cannot be completely excluded. To illustrate this issue, a phase-type distribution have been fitted on 6.11 with 10 phases. The converged sub-intensity matrix contains an entry with the value $t_{2,9} = 0.0045$. Under some simulations this entry will converge to 0 which gives potentially two parameterizations.

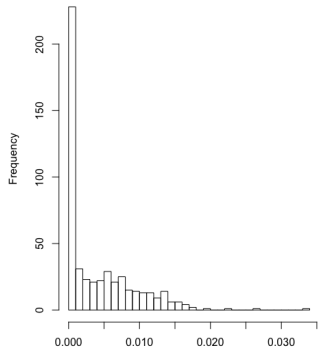


Figure 6.11: Sample distribution of $t_{2,9}$

Issues regarding few observations

Using statistical inference based on the Fisher information matrix requires sufficiently many observations, since this approach is based on asymptotic results. Some normalization techniques may however be used, in order to accelerate the process of convergence. The bootstrap method can be used as an alternative, when the number of observations are insufficient for the maximum likelihood estimates to be normally distributed, since the asymptotic results in the bootstrap approach is based on the number of simulations, and not the number of observations. However, the assumptions of the bootstrap principle, which say that the true distribution can be approximated by the empirical distribution, become less realistic when the number of observations is low.

Evaluation of derivatives

Using the asymptotic covariance matrix for deriving confidence regions require calculation of derivatives. One shortfall of the EM algorithm is that the asymptotic covariance matrix can be quite tedious to calculate, since the EM algorithm does not require evaluation of neither the observed log-likelihood nor the derivatives. Bootstrapping makes a way around direct calculations of the derivatives, and can therefore be applied, when the derivatives are too analytically intractable.

Uncensored data

As mentioned, the Fisher information matrix has been derived for uncensored observations. Implementing this bootstrap algorithm in the EMpht program for censored observations can most likely be done analogous to the approach presented in this chapter, and thereby making it possible to do statistical inference on censored observations as well.

The flip side of using the bootstrap algorithm with censored observations, is that censored observations decreases the rate of convergence and thereby increases computation time. The latter is easily seen by comparing the expectation step in the EM algorithm for uncensored and censored observations respectively.

Given that the censored observations to not comprise a too large part of the sample, a completely different approach is using a modified version of the stochastic EM algorithm introduced in [30]. Instead of the classical approach in the E-step we can simulate observations until the data is complete. Hence, we will for each iteration replace each incomplete data with a simulation from the distribution of the observed data. This reduces the problem to a complete maximum likelihood which makes the M-step easier to calculate. By the use of stochastic EM algorithm the problem reduces to a complete maximum likelihood, from which the formula of the Fisher information matrix can be applied.

Execution time

The last issue regards execution time. If the average number of iterations, necessary for convergence, increases by, say x, the total number of iterations increases by x times the numbers of simulations.

Even though computation time for each iteration increases quadratic with the number of dimensions, convergence is also slower. Therefore, a higher number of iterations are required such that total execution time until convergence increases more than quadratic with the dimension. The execution time of the bootstrap algorithm is therefore specially susceptible towards high dimensions, when taking the number of simulations into perspective.

As we saw in 6.8, quantiles near the tails may require many simulations to get valid estimates. We may be able to reduce the number of simulations by combining importance sampling with the bootstrap algorithm. By introducing a shifted measure, we may be able to draw outlying observations more frequently in the resampling process, and thereby achieving more simulations near the tail. This problem, is however nontrivial since we have to find a Radon-Nikodym derivative which suits for this purpose.

From the perspective of a computer scientist, the issue regarding execution time can be reduced significantly by implementing multithreading into the program. Multithreading allows different procedures to be run concurrently rather than sequentially. Since the maximization step depends on the results of the expectation step, concurrency of the E step and M step is not possible, so multithreading will not speed up the process of the EM algorithm. However the time issues concerning this bootstrap algorithm implemented in this chapter can obviously be improved by multithreading, since each simulation can run concurrently. From a theoretical point of view, execution time can therefore be reduced by k times if the machine contains k processors. Implementing multithreading into the bootstrap algorithm is therefore particularly advantageous when the number of processors is large.

6.6 Bibliographic references

The introduction to bootstrap has been inspired by [14], [15], [23] and [5]. The central limit theorem for sample quantiles in equation 6.1 follows from [34].

The EMpht program is originally developed by Asmussen et.al [20] and modified by later on by the same authors as an implementation of [4] and [31]. Olsson also gives an introduction to the EMpht program in [32].

See [7] for estimation of phase-type distributions using Markov chain Monte Carlo. Alkaike's information criterion was first introduced in [1]. Bayesian information criterion was first introduced in [36]. In [6] Bladt et al. suggested using Alkaike's information criterion for model selection when fitting phase-type distributions. In [6] and [17] the Fisher information matrix is calculated for both discrete and continuous phase-type distributions.

Appendix A: Code for phase-type approximations

Visualization of log-normal distribution in R

```
#The package LaplacesDemon and moments is required

#Defining time intervals
ti_temp <- 0.01
ti <- matrix(seq(ti_temp,10,ti_temp),ncol=1)

#Indicator function
In <- function(x){ifelse(x,1,0)}

#Theoretical function
p1 <- 0.5
p2 <- 1.5

F <- function(x){plnorm(x,p1,p2)}      # theoretical cumulative function
D <- function(x){dlnorm(x,p1,p2)}      # theoretical density

inverse <- function (f, lower = 0, upper = 100) {
  function (y) uniroot((function (x) f(x) - y),
                       lower = lower, upper = upper)[1]
}
Fi <- inverse(F, 0.001, 10000)              # theoretical inverse function

#Plot of theoretical function
par(mfrow=c(1,2))
plot(ti,F(ti), type="l",xlab="x",ylab=" Distribution ")
plot(ti,mapply(D,ti), xlab="x",type="l",ylab="Density")
labels = c(" Theoretical distribution "," Theoretical dentisity ")

#Theoretical approximating distribution
```

```
app <- function(cutoff,n,x){
    m <- floor(n*(Fi(cutoff))$ root)+1
    k <- seq(1:m)
    ks <- k-rep(1,m)
    return((1/F(m/n))*t(F(k/n)-F(ks/n))%*%pgamma(x,k,n))
}

#Theoretical approximating density
appd <- function(cutoff,n,x){
  m <- floor(n*(Fi(cutoff))$ root)+1
  k <- seq(1:m)
  ks <- k-rep(1,m)
  return((1/F(m/n))*t(F(k/n)-F(ks/n))%*%dgamma(x,k,n))
}

# Initialisation to the plots
par(mfrow=c(1,1))
gr <- 5
collabels <- labels <- rep("Na",gr+1)
phlabels <- rep("Na",gr)
labels[1] <- "Theoretical distribution "
collabels[1] <- "black"

#Adjustment of cutoff level distribution
plot(ti,F(ti), col="black",type="S",lwd=2,
     lty=2,main="",xlab="x",ylab=" Distribution ")
for (i in 1:gr){
  lines(ti,mapply(app,0.92+i*0.015,5,ti),
        col=paste("gray",100-i*19,sep=""))
  mtemp <- floor(5*(Fi(0.92+i*0.015))$ root)+1 #Calculating number of phases
  labels[i+1] <- paste((0.92+i*0.015)*100,"% cutoff level. ")
  phlabels[i] <- paste(mtemp*(mtemp-1)/2," phases")
  collabels[i+1] <- paste("gray",100-i*19,sep="")
}
legend("right", inset=.26, labels,lwd=2,
       col=collabels, lty = c(2,rep(1,gr)), bty = "n",cex=0.8)
legend("bottom", inset=.15, phlabels,lwd=2,
       col=collabels[1:gr+1], lty = rep(1,gr), bty = "n",cex=0.8)

#Adjustment of cutoff level density
plot(ti,D(ti), col="black",type="S",
     lwd=2,lty=2,main="",xlab="x",ylab="Density")
for (i in 1:gr){
  lines(ti,mapply(appd,0.92+i*0.015,5,ti),
        col=paste("gray",100-i*19,sep=""))
  mtemp <- floor(5*(Fi(0.92+i*0.015))$ root)+1
```

```
   labels[i+1] <- paste((0.92+i*0.015)*100,"% cutoff level. ")
   phlabels[i] <- paste(mtemp*(mtemp-1)/2," phases")
   collabels[i+1] <- paste("gray",100-i*19,sep="")
}
legend("right", inset =.26, labels, lwd=2,
        col=collabels, lty = c(2,rep(1,gr)), bty = "n",cex=0.8)
legend("bottom", inset =.15, phlabels, lwd=2,
        col=collabels[1:gr+1], lty = rep(1,gr), bty = "n",cex=0.8)

#Adjustment of cutoff level QQ plot
qqplot(F(ti), F(ti), col="black", type="S",
        lwd=2,lty=2,xlab="x",ylab="Distribution")
for (i in 1:gr){
   lines(mapply(app,0.92+i*0.015,5, ti), F(ti),
          col=paste("gray",100-i*19,sep=""))
}
legend("topleft", inset =.05, labels, lwd=2,
        col=collabels, lty = c(2,rep(1,gr)), bty = "n")
legend("bottomright", inset =.10, phlabels, lwd=2,
        col=collabels[1:gr+1], lty = rep(1,gr), bty = "n")

#Adjustment of phases initialization
collabels[1:gr+1] <- labels[1:gr+1] <- rep("Na",gr+1)
phlabels <- rep("Na",gr)

#Adjustment of phases distribution
plot(ti, mapply(F,ti), col="black", type="S",
      lwd=2,lty=2,xlab="x",ylab="Distribution")
for (i in 1:5){
   mtemp <- floor(i*(Fi(0.99))$root)+1
   lines(ti, mapply(app,0.99,i, ti),
          col=paste("gray",100-i*19,sep=""))
   labels[i+1] <- paste("n=",i)
   phlabels[i] <- paste(mtemp*(mtemp-1)/2," phases")
   collabels[i+1] <- paste("gray",100-i*19,sep="")
}
legend("right", inset =.26, labels, lwd=2,
        col=collabels, lty = c(2,rep(1,gr)), bty = "n",cex=0.8)
legend("bottom", inset =.15, phlabels, lwd=2,
        col=collabels[1:gr+1], lty = rep(1,gr), bty = "n",cex=0.8)

#Adjustment of phases density
plot(ti, mapply(D,ti), col="black",
      type="S",lwd=2,lty=2,xlab="x",ylab="Density")
for (i in 1:5){
   lines(ti, mapply(appd,0.99,i, ti),
```

```
          col=paste("gray",100-i*19,sep=""))
}
legend(" right ", inset =.26, labels ,lwd=2,
       col=collabels , lty = c(2,rep(1, gr )), bty = "n",cex=0.8)
legend("bottom", inset =.15, phlabels ,lwd=2,
       col=collabels [1: gr+1], lty = rep(1,gr ), bty = "n",cex=0.8)

#Adjustment of phases QQ plot
qqplot(F( ti ), F( ti ), col="black", type="S",lwd=2,lty=2,
       xlab="Approximating distributions",
       ylab="Theoretical distribution ")
for (i in 1: gr){
  lines (mapply(app,0.99, i , ti ), F( ti ),
          col=paste("gray",100-i*19,sep=""))
}
legend(" topleft ", inset =.10, labels [1: gr+1],lwd=2,
       col=collabels [1: gr+1], lty = rep(1,gr ), bty = "n")
legend("bottomright", inset =.10, phlabels ,lwd=2,
       col=collabels [1: gr+1], lty = rep(1,gr ), bty = "n")

#Kullbak Leibner distance
kld_udf <- function(cutoff, n){
  return(KLD(F(ti),mapply(app, cutoff , n, ti ))$ sum.KLD.px.py)
}

#Plot of Kulbak Leibner distance - cut-off level is fixed
n <- seq(1,30)
plot (seq (1,30), mapply(kld_udf ,0.99, seq (1,30)),
      col="black",type="S",lwd=2,lty=1,
      xlab="n",ylab="Kullbak Leibner distance")

#Calculating moments. package moments is required
mom2 <- function(cutoff,n,i){
  return( all .moments(mapply(app,cutoff,n, ti ))[ i ])
}

n <- seq(2:30)
plot (n,mapply(mom2,0.99,n,3))
plot (n,mapply(mom2,0.99,n,2))
```

Visualization of mixture of log-normal distributions in R

```
# defining time intervals
ti_temp <- 0.01
ti <- matrix(seq(ti_temp,10, ti_temp),ncol=1)
```

```
#  indicator  function
In <-  function(x){ifelse(x,1,0)}

#  theoretical  function
p1 <- 0.05
p2 <- 0.2
F <-  function(x){0.4*plnorm(x,p1,p2)+0.6*plnorm(x,p1+1.5,p2)}
D <-  function(x){0.4*dlnorm(x,p1,p2)+0.6*dlnorm(x,p1+1.5,p2)}
inverse <- function (f, lower = -100, upper = 100) {
  function (y) uniroot((function (x) f(x) - y),
    lower = lower, upper = upper)[1]
}
Fi <-  inverse(F, 0, 100)
par(mfrow=c(1,2))
plot(ti,F(ti), type="l",xlab="x",ylab=" Distribution ")
plot(ti,mapply(D,ti), xlab="x",type="l",ylab="Density")
labels = c(" Theoretical  distribution "," Theoretical  dentisity ")

#  theoretical  approximating  distribution
app <- function(cutoff,n,x){
  #m <- floor(n*Fi(cutoff))+1
  m <- floor(n*(Fi(cutoff))$ root)+1
  k <- seq(1:m)
  ks <- k-rep(1,m)
  return((1/F(m/n))*t(F(k/n)-F(ks/n))%*%pgamma(x,k,n))
}

#  theoretical  approximating  density
appd <- function(cutoff,n,x){
  m <- floor(n*(Fi(cutoff))$ root)+1
  k <- seq(1:m)
  ks <- k-rep(1,m)
  return((1/F(m/n))*t(F(k/n)-F(ks/n))%*%dgamma(x,k,n))
}

#  initialisation  to  the  plots
par(mfrow=c(1,1))
gr <- 5
 collabels <- labels <- rep("Na",gr+1)
phlabels <-  rep("Na",gr)
labels[1] <- "Theoretical  distribution "
 collabels[1] <- "black"

# adjustment  of  cutoff  level  distribution
plot(ti,F(ti), col="black",type="S",lwd=2,
```

```
lty=2,main="",xlab="x",ylab=" Distribution ")
for (i in 1:gr){
  lines(ti,mapply(app,0.92+i*0.015,25,ti),
  col=paste("gray",100-i*19,sep=""))
  mtemp <- floor(25*(Fi(0.92+i*0.015))$root)+1
  labels[i+1] <- paste((0.92+i*0.015)*100,"% cutoff level. ")
  phlabels[i] <- paste(mtemp*(mtemp-1)/2," phases")
  collabels[i+1] <- paste("gray",100-i*19,sep="")
}
#legend(" right ", inset =.16, labels, lwd=2,
col=collabels, lty = c(2,rep(1,gr)), bty = "n",cex=0.8)
#legend("bottom", inset =.15, phlabels, lwd=2,
col=collabels[1:gr+1], lty = rep(1,gr), bty = "n",cex=0.8)

# adjustment of cutoff level density
plot(ti,D(ti), col="black",type="S",
lwd=2,lty=2,main="",xlab="x",ylab="Density")
for (i in 1:gr){
  lines(ti,mapply(appd,0.92+i*0.015,25,ti),
  col=paste("gray",100-i*19,sep=""))
  mtemp <- floor(25*(Fi(0.92+i*0.015))$root)+1
  labels[i+1] <- paste((0.92+i*0.015)*100,"% cutoff level. ")
  phlabels[i] <- paste(mtemp*(mtemp-1)/2," phases")
  collabels[i+1] <- paste("gray",100-i*19,sep="")
}
#legend(" topright ", inset =0.06, labels, lwd=2,
col=collabels, lty = c(2,rep(1,gr)), bty = "n",cex=0.8)
#legend(" right ", inset =.15, phlabels, lwd=2,
col=collabels[1:gr+1], lty = rep(1,gr), bty = "n",cex=0.8)

# adjustment of phases distribution
collabels[1:gr+1] <- labels[1:gr+1] <- rep("Na",gr+1)
phlabels <- rep("Na",gr)
plot(ti,mapply(F,ti), col="black",type="S",
lwd=2,lty=2,xlab="x",ylab=" Distribution ")

for (i in 1:5){
  mtemp <- floor(5*i*(Fi(0.99))$root)+1
  lines(ti,mapply(app,0.99,5*i,ti),
  col=paste("gray",100-i*19,sep=""))
  labels[i+1] <- paste("n=", 5*i)
  phlabels[i] <- paste(mtemp*(mtemp-1)/2," phases")
  collabels[i+1] <- paste("gray",100-i*19,sep="")
}
#legend(" right ", inset =.16, labels, lwd=2,
col=collabels, lty = c(2,rep(1,gr)), bty = "n",cex=0.8)
```

```
#legend("bottom", inset =.15, phlabels, lwd=2,
col= collabels [1: gr+1], lty = rep(1,gr), bty = "n",cex=0.8)

# adjustment of phases density
 collabels [1: gr+1] <- labels[1:gr+1] <- rep("Na",gr+1)
 phlabels <- rep("Na",gr)
plot( ti ,mapply(D,ti ), col="black", type="S",
lwd=2,lty=2,xlab="x",ylab="Density")

for (i in 1:5){
   lines( ti ,mapply(appd,0.99,5*i , ti ),
   col=paste("gray",100-i*19,sep=""))
}
#legend(" right ", inset =.26, labels , lwd=2,
col= collabels , lty = c(2,rep(1, gr )), bty = "n",cex=0.8)
#legend("bottom", inset =.15, phlabels, lwd=2,
  col= collabels [1: gr+1], lty = rep(1,gr), bty = "n",cex=0.8)
```

Appendix B: Code for bootstrap implementation in the EM algorithm

Modification of the EMpht program

```
/* importing libraries */
#include<stdio.h>
#include< stdlib .h>
#include<math.h>

/* declaration of global variables */
double *obs, *obsS, *weight;
double  SumOfWeights, SumOfWeightsS, cv, prec;
int   NoOfObs, Ra, sim_k, iT,*pilegal, **Tlegal, IV, OW, RI;

/*  functions for allocation of memory */
double *v_alloc(int n)
{
  double *v;
  v=(double *) calloc(n, sizeof(double));
  if(v==NULL) {
     fprintf( stderr ,"could not allocate memory");
     exit (1);
  }
  return v;
}

int * int_v_alloc (int n)
{
  int *v;
  v=(int *) calloc (n, sizeof(int ));
```

```
  if(v==NULL) {
    fprintf( stderr ,"could not allocate memory");
    exit(1);
  }
  return v;
}

double **m_alloc(int n, int k)
{
  int i;
  double **mat;
  mat=(double **) calloc(n,sizeof(double *));
  if(mat == NULL) {
    fprintf( stderr ,"could not allocate memory");
    exit(1);
  }
  for(i=0; i<n; i++) {
    mat[i]=(double *) calloc(k,sizeof(double));
    if(mat[i] == NULL) {
      fprintf( stderr ,"could not allocate memory");
      exit(1);
    }
  }
  return mat;
}

int **int_m_alloc(int n, int k)
{
  int i, **mat;
  mat=(int **) calloc(n,sizeof(int *));
  if(mat == NULL) {
    fprintf( stderr ,"could not allocate memory");
    exit(1);
  }
  for(i=0; i<n; i++) {
    mat[i]=(int *) calloc(k,sizeof(int));
    if(mat[i] == NULL) {
      fprintf( stderr ,"could not allocate memory");
      exit(1);
    }
  }
  return mat;
}

double ***m3_alloc(int n, int k, int v)
{
```

```
  int i,j;
  double ***mat;
  mat=(double ***) calloc(n,sizeof(double **));
  if(mat == NULL) {
     fprintf( stderr ,"could not allocate memory");
     exit(1);
  }
  for(i=0; i<n; i++) {
     mat[i]=(double **) calloc(k, sizeof(double *));
     if(mat[i] == NULL) {
        fprintf( stderr ,"could not allocate memory");
        exit(1);
     }
     for(j=0; j<k; j++) {
        mat[i][j]=(double *) calloc(v, sizeof(double));
        if(mat[i][j] == NULL) {
           fprintf( stderr ,"could not allocate memory");
           exit(1);
        }
     }
  }
  return mat;
}

/* subroutines for initializations */
void init_vector (double *vector, int NoOfElements)
{
  int i;
  for(i=0; i < NoOfElements; i++)
     vector[i] = 0.0;
}

void init_matrix (double **matrix, int dim1, int dim2)
{
  int i, j;

  for (i=0; i<dim1; i++)
     for(j=0; j<dim2; j++)
        matrix[i][j] = 0.0;
}

void init_3dimmatrix(double ***matrix, int dim1, int dim2, int dim3)
{
  int i, j, k;

  for (i=0; i<dim1; i++)
```

```
      for(j=0; j<dim2; j++)
        for(k=0; k<dim3; k++)
          matrix[i][j][k] = 0;
}

void free_matrix(double **matrix, int dim1)
{
  int i;

  for (i=0; i<dim1; i++)
    free(matrix[i]);
  free(matrix);
}

void free_integermatrix(int **matrix, int dim1)
{
  int i;

  for (i=0; i<dim1; i++)
    free(matrix[i]);
  free(matrix);
}

void free_3dimmatrix(double ***matrix, int dim1, int dim2)
{
  int i;

  for (i=0; i<dim1; i++)
    free_matrix(matrix[i],dim2);
  free(matrix);
}

/* Runge Kutta procedure for calculation of matrix exponentials */
void a_rungekutta(int p, double *avector, double **ka, double dt, double h,
                  double **T)
{
  int i,j;
  double eps, h2, sum;

  i=dt/h;
  h2=dt/(i+1);
  init_matrix(ka, 4, p);

  for (eps=0; eps<=dt-h2/2; eps += h2) {
    for (i=0; i < p; i++) {
      sum=0;
```

```
        for (j=0; j < p; j++)
          sum += T[j][i]*avector[j];
        ka[0][i] = h2*sum;
      }
      for (i=0; i < p; i++) {
        sum=0;
        for (j=0; j < p; j++)
          sum += T[j][i]*(avector[j]+ka[0][j]/2);
        ka[1][i] = h2*sum;
      }
      for (i=0; i < p; i++) {
        sum=0;
        for (j=0; j < p; j++)
          sum += T[j][i]*(avector[j]+ka[1][j]/2);
        ka[2][i] = h2*sum;
      }
      for (i=0; i < p; i++) {
        sum=0;
        for (j=0; j < p; j++)
          sum += T[j][i]*(avector[j]+ka[2][j]);
        ka[3][i] = h2*sum;
      }

      for (i=0; i < p; i++)
        avector[i] += (ka[0][i]+2*ka[1][i]+2*ka[2][i]+ka[3][i])/6;
    }
}

void rungekutta(int p, double *avector, double *gvector, double *bvector,
                double **cmatrix, double dt, double h, double **T, double *t,
                double **ka, double **kg, double **kb, double ***kc)
{
  int i, j, k, m;
  double eps, h2, sum;

  i = dt/h;
  h2 = dt/(i+1);
  init_matrix(ka, 4, p);
  init_matrix(kb, 4, p);
  init_3dimmatrix(kc, 4, p, p);
  if (kg != NULL)
    init_matrix(kg, 4, p);

  for (eps = 0; eps <= dt-h2/2; eps += h2) {
    for (i=0; i < p; i++) {
      sum=0;
```

```
  for (j=0; j < p; j++)
    sum += T[j][i]*avector[j];
  ka[0][i] = h2*sum;
}
for (i=0; i < p; i++) {
  sum=0;
  for (j=0; j < p; j++)
    sum += T[j][i]*(avector[j]+ka[0][j]/2);
  ka[1][i] = h2*sum;
}
for (i=0; i < p; i++) {
  sum=0;
  for (j=0; j < p; j++)
    sum += T[j][i]*(avector[j]+ka[1][j]/2);
  ka[2][i] = h2*sum;
}
for (i=0; i < p; i++) {
  sum=0;
  for (j=0; j < p; j++)
    sum += T[j][i]*(avector[j]+ka[2][j]);
  ka[3][i] = h2*sum;
}

if (gvector != NULL) {
  for (i=0; i < p; i++)
    kg[0][i] = h2*avector[i];
  for (i=0; i < p; i++)
    kg[1][i] = h2*(avector[i]+ka[0][i]/2);
  for (i=0; i < p; i++)
    kg[2][i] = h2*(avector[i]+ka[1][i]/2);
  for (i=0; i < p; i++)
    kg[3][i] = h2*(avector[i]+ka[2][i]);
  for (i=0; i < p; i++)
  gvector[i] += (kg[0][i]+2*kg[1][i]+2*kg[2][i]+kg[3][i])/6;
}

for (i=0; i < p; i++) {
  sum=0;
  for (j=0; j < p; j++)
    sum += T[i][j]*bvector[j];
  kb[0][i] = h2*sum;
}
for (i=0; i < p; i++) {
  sum=0;
  for (j=0; j < p; j++)
    sum += T[i][j]*(bvector[j]+kb[0][j]/2);
```

```
  kb[1][i] = h2*sum;
}
for (i=0; i < p; i++) {
  sum=0;
  for (j=0; j < p; j++)
    sum += T[i][j]*(bvector[j]+kb[1][j ]/2);
  kb[2][i] = h2*sum;
}
for (i=0; i < p; i++) {
  sum=0;
  for (j=0; j < p; j++)
    sum += T[i][j]*(bvector[j]+kb[2][j ]);
  kb[3][i] = h2*sum;
}

for (m=0; m < p; m++)
  for (i=0; i < p; i++) {
    sum=t[m]*avector[i];
    for (j=0; j < p; j++)
      sum += T[m][j]*cmatrix[j][i];
    kc[0][m][i] = h2*sum;
  }
for (m=0; m < p; m++)
  for (i=0; i < p; i++) {
    sum=t[m]*(avector[i]+ka[0][i ]/2);
    for (j=0; j < p; j++)
      sum += T[m][j]*(cmatrix[j][i]+kc[0][j ][ i ]/2);
    kc[1][m][i] = h2*sum;
  }
for (m=0; m < p; m++)
  for (i=0; i < p; i++) {
    sum=t[m]*(avector[i]+ka[1][i ]/2);
    for (j=0; j < p; j++)
      sum += T[m][j]*(cmatrix[j][i]+kc[1][j ][ i ]/2);
    kc[2][m][i] = h2*sum;
  }
for (m=0; m < p; m++)
  for (i=0; i < p; i++) {
    sum=t[m]*(avector[i]+ka[2][i ]);
    for (j=0; j < p; j++)
      sum += T[m][j]*(cmatrix[j][i]+kc[2][j ][ i ]);
    kc[3][m][i] = h2*sum;
  }

for (i=0; i < p; i++) {
  avector[i] += (ka[0][i]+2*ka[1][i]+2*ka[2][i]+ka[3][ i ])/6;
```

```
        bvector[i] += (kb[0][i]+2*kb[1][i]+2*kb[2][i]+kb[3][i])/6;
        for (j=0; j < p; j++)
          cmatrix[i][j] +=(kc[0][i][j]+2*kc[1][i][j]+2*kc[2][i][j]+kc[3][i][j])/6;
      }
    }
}

/* subroutine for displaying estimated parameter values */
void show_pi_T(int p, double *pi, double **T, double *t)
{
  int i, j;

  for (i=0; i < p; i++) {
     printf("\n%lf      ", pi[i]);
     for (j=0; j < p; j++)
        printf("%lf ", T[i][j]);
     printf("     %lf",t[i]);
  }
  printf("\n");
}

/* default step length */
double set_steplength(int p, double **T)
{
  int i;
  double h;

  h= -0.1/T[0][0];
  for (i=1; i < p; i++)
    if (h > -0.1/T[i][i])
      h = -0.1/T[i][i];
  return(h);
}

/* subroutine for importing and observations */
void input_sample(int NoOfInput[3])
{
  double Obs, Weight;
  FILE *utfil, *infil;

     Weight=1;
     infil=fopen("unweighted", "r");
     utfil=fopen("sample", "w");
     fscanf(infil, "%le", &Obs);
     while (Obs >= 0) {
       NoOfInput[1]++;
```

```
        fprintf( utfil , "%e %e \n", Obs, Weight);
        fscanf( infil , "%le", &Obs);
      }
      fprintf( utfil , "-1");
      fclose( infil );
      fclose( utfil );

}

/* assigning observations to vectors */
void assign_vectors (int No)
{
  int i;
  FILE *infil;

   infil =fopen("sample", "r");
   for(i=0; i < No; i++) {
     fscanf( infil , "%le %le", &obs[i], &weight[i]);
     SumOfWeights += weight[i];
  }
  fclose( infil );
}

/* sorting observations and identification of
 * unique observations */
int sort_observations (int size, double *vec1, double *vec2)
{
  int i,j, tempweight, newsize;
  double tempobs;

  newsize=size;
  for (i=0; i < size-1; i++)
    for (j=i+1; j < size; j++)
      if (vec1[i] > vec1[j]) {
        tempobs = vec1[i];
        vec1[i] = vec1[j];
        vec1[j] = tempobs;
        tempweight = vec2[i];
        vec2[i] = vec2[j];
        vec2[j] = tempweight;
      }
  for (i=size-2; i >= 0; i--)
    if (vec1[i] == vec1[i+1]) {
      vec2[i] += vec2[i+1];
      newsize--;
      for(j=i+1; j < newsize; j++) {
```

```
            vec1[j] = vec1[j+1];
            vec2[j] = vec2[j+1];
          }
        }
    return (newsize);
}

/* preparatory procedure */
int AskForInput(int NoOfInput[3])
{
    int input, sampletype;

        sampletype = 1;
        input_sample(NoOfInput);
        obs = v_alloc(NoOfInput[1]);
        obsS = v_alloc(NoOfInput[1]);
        weight = v_alloc(NoOfInput[1]);
        assign_vectors(NoOfInput[1]);
        NoOfObs = sort_observations(NoOfInput[1], obs, weight);
        printf("Total number of observations = %d\n", (int) SumOfWeights);

    return(sampletype);
}

/* random initializations - not used
 * when simulating */
double random1()
{
    double r;

    r=rand();
    r /= ((double) RAND_MAX);
    r *= 0.9;
    r += 0.1;
    return(r);
}

void randomphase(int p, double *pi, double **T, double *exitvector,
                 int *pilegal, int **Tlegal)
{
    int i, j;
    double r, sum, scalefactor;

      scalefactor =obs[(NoOfObs-1)/2];

    sum=0;
```

```
  for (i=0; i < p; i++)
    if ( pilegal[i] == 1) {
      pi[i]=random1();
      sum += pi[i];
    }
  for (i=0; i < p; i++)
    pi[i]=pi[i]/sum;
  for (i=0; i < p; i++)
    for (j=0; j < p; j++)
      if ((i != j) && (Tlegal[i][j] == 1)) {
        T[i][j]=random1();
        T[i][i] -= T[i][j];
      }
  for (i=0; i < p; i++)
    if (Tlegal[i][i] == 1) {
      r=random1();
      T[i][i] -= r;
      exitvector[i]=r;
    }
  for (i=0; i < p; i++) {
    exitvector[i] = exitvector[i] * p/ scalefactor ;
    for (j=0; j < p; j++)
      T[i][j] = T[i][j] * p / scalefactor ;
  }
}

/* selection of structure :
 * Random initiation used in modern selection
 * When simulating saved parameter values are imported */
void selectstructure (int p, double *pi, double **T, double *t, int *pilegal,
                      int **Tlegal)
{
  int i, j, structure ;
  FILE *infil;

    if (IV == 2) {
        structure = 7;
    } else {
        structure = 1;
    }

  switch( structure ) {
  case 1:
    for (i=0; i < p; i++) {
      pilegal[i]=1;
      for (j=0; j < p; j++)
```

```
            Tlegal[i][j]=1;
        }
        break;
    case 7:
        infil=fopen("phases", "r");
        for (i=0; i < p; i++) {
            fscanf( infil , "%le", &pi[i]);
            for (j=0; j < p; j++)
                fscanf( infil , "%le", &T[i][j]);
        }
        fclose( infil );
        break;
    }

    if ( structure < 7) {
        printf("\n Phase-type structure:");
        for (i=0; i < p; i++) {
            printf("\n%d ", pilegal[i]);
            for (j=0; j < p; j++)
                printf("%d ", Tlegal[i][j]);
        }
        printf("\nRandom initiation - enter an integer (at random):");
        scanf("%d",&RI);
        srand(RI);
        printf("\n");
        randomphase(p, pi, T, t, pilegal , Tlegal);
    }
    else
        for (i=0; i < p; i++)
            for (j=0; j < p; j++)
                t[i] -= T[i][j];
}

/* EM algorithm */
void EMstep(int p, double h, double *pi, double **T, double *t, double
                *gvector, double *avector, double *bvector, double **cmatrix,
                double *Bmean, double *Zmean, double **Nmean, double **kg,
                double **ka, double **kb, double ***kc)
{
    int i, j, k;
    double pitimesb, dt;
    init_vector (Bmean, p);
    init_vector (Zmean, p);
    init_matrix (Nmean, p, p+1);

    if (NoOfObs > 0) {
```

```
  for (i=0; i < p; i++) {
    avector[i]=pi[i];
    bvector[i]=t[i];
  }
  init_matrix(cmatrix, p, p);
  dt = obs[0];
  for (k=0; k < NoOfObs; k++) {
    if (dt > 0)
      rungekutta(p, avector, NULL, bvector, cmatrix, dt, h, T, t, ka, NULL,
                 kb, kc);
    pitimesb=0;
    for (i=0; i < p; i++)
      pitimesb += pi[i]*bvector[i];
    for (i=0; i < p; i++) {
      Bmean[i] += pi[i]*bvector[i]*weight[k]/pitimesb;
      Nmean[i][p] += avector[i]*t[i]*weight[k]/pitimesb;
      Zmean[i] += cmatrix[i][i]*weight[k]/pitimesb;
      for (j=0; j < p; j++)
        Nmean[i][j] += T[i][j]*cmatrix[j][i]*weight[k]/pitimesb;
    }
    dt=obs[k+1]-obs[k];
  }
}

  /* identification of change in estimated
   * parameter values since last iteration */
  cv = 0;
  for (i=0; i < p; i++) {
      if (cv < fabs(pi[i]-Bmean[i] / SumOfWeights))
      cv = fabs(pi[i]-Bmean[i] / SumOfWeights);

      if (cv < fabs(t[i] - Nmean[i][p] / Zmean[i]))
          t[i] = fabs(Nmean[i][p] / Zmean[i]);

      for (j=0; j < p; j++){
          if (cv < fabs(T[i][j] - Nmean[i][j] / Zmean[i]))
              cv = fabs(T[i][j] - Nmean[i][j] / Zmean[i]);
          }
  }

  for (i=0; i < p; i++) {
  pi[i] = Bmean[i] / (SumOfWeights);
  if (pi[i] < 0)
    pi[i] = 0;
  t[i] = Nmean[i][p] / Zmean[i];
  if (t[i] < 0)
```

```
      t[i] = 0;
    T[i][i] = -t[i];
    for (j=0; j < p; j++)
      if (i!=j) {
        T[i][j] = Nmean[i][j] / Zmean[i];
        if (T[i][j] < 0)
          T[i][j] = 0;
        T[i][i] -= T[i][j];
      }
  }

}

/* computation of log-likelihood */
void compute_loglikelihood(double h, int p, double *pi, double **T, double *t,
                           int stepindicator, double *avector, double **ka,
                           double **a_left, int k, int NoOfEMsteps)
{
  int i, j;
  double loglikelihood, atimesexit, dt;

  if ( stepindicator == 1)
    h = set_steplength(p, T);
  loglikelihood = 0;

  if (NoOfObs > 0) {
    for (i=0; i < p; i++)
      avector[i] = pi[i];
    dt = obs[0];
    for (i=0; i < NoOfObs; i++) {
      atimesexit = 0.0;
      if (dt > 0)
        a_rungekutta(p, avector, ka, dt, h, T);
      for (j=0; j < p; j++)
        atimesexit += avector[j] * t[j];
      loglikelihood += weight[i] * log( atimesexit );
      dt = obs[i+1]-obs[i];
    }
    /* calculation and exportation of information criterias
     * for each simulation */
      if (k==NoOfEMsteps) {
          FILE *utfil;
        if (OW == 1){
            utfil = fopen("loglik", "w");
        } else {
            utfil = fopen("loglik", "a");
```

```
            }
                fprintf( utfil , "%lf phases: %d iterations: %d seed: %d AIC: %f BIC: %f\n",
                                loglikelihood,p,NoOfEMsteps,RI,
                                2*(p*p+p-1)-2*loglikelihood,
                                log(SumOfWeights)*(p*p+p-1)-2*loglikelihood);
                fclose( utfil );
        }
    }

    printf ("Log-likelihood = %lf \n", loglikelihood );
}

/* exportation of estimated parameter values
 * when using model selection */
void SavePhases(int p, double *pi, double **T)
{
    int i,j;
    FILE *utfil;

    utfil =fopen("phases", "w");
    for (i=0; i < p; i++) {
        fprintf ( utfil , "\n%e     ", pi[i ]);
        for (j=0; j < p; j++)
            fprintf ( utfil , "%e ", T[i][j ]);
    }
    fclose ( utfil );
}

/* exportation of the estimated parameter values
 * for each simulation */
void SavePhasesT(int p, double **T) {
        int i,j;
        FILE *utfil;


        utfil =fopen("phasesT", "a" );
        for (i=0; i < p; i++) {
                for (j=0; j< p; j++) {
                        fprintf( utfil , "%e ", T[i][j ]);
                }
        }
        fprintf ( utfil , "\n");
        fclose( utfil );
}
```

```
void SavePhasesPi(int p, double *pi) {
     int i;
     FILE *utfil;

      utfil=fopen("phasesPi", "a");
     for (i=0; i < p; i++) {
          fprintf( utfil , "%e ", pi[i ]);

     }
      fprintf ( utfil , "\n");
     fclose( utfil );
}

/* main subroutine for the EM algorithm */
void EMiterate(int NoOfEMsteps, int p, double *pi, double **T, double *t,
                    double *gvector, double *avector, double *bvector,
                    double **cmatrix, double *Bmean, double *Zmean, double **Nmean,
                    double **kg, double **ka, double **kb, double ***kc,
                    double *ett)
{
  int i, j, k,l, stepindicator , stepchoice , tempSim, tempEM;
  double RKstep;

      stepindicator = 1;


     if (IV == 2){
          tempSim = 1;
          NoOfEMsteps *= sim_k;

          /* Random seed in order to recreate the simulations */
          printf("\nRandom initiation - enter an integer (at random):");
          scanf("%d",&iT);
          srand(iT);
          printf("\n");

          /* Resample procedure */
          init_vector(weight,NoOfObs);
          for(i=0; i < SumOfWeights; i++) {
               Ra=rand()%NoOfObs;
                    weight[Ra] += 1;
          }

          FILE *utfil;
          utfil=fopen("phasesT", "w");
          fclose( utfil );
```

```
    utfil=fopen("phasesPi", "w");
    fclose( utfil );
    utfil=fopen(" Iterations ", "w");
    fclose( utfil );
}

for (k=1; k <= NoOfEMsteps; k++) {
    RKstep = set_steplength(p, T);
    EMstep(p, RKstep, pi, T, t, NULL, avector, bvector, cmatrix, Bmean,
           Zmean, Nmean, NULL, ka, kb, kc);
    if (IV == 2) {
        /* displaying Number of iterations, simulations
         * estimated parameter values and the biggest
         * absolute change since last iteration */
        tempEM = (k % ((NoOfEMsteps / sim_k)+1)+sim_k-1);
        printf("\nEM(%d)",tempEM);
        printf("\nSim(%d)",tempSim);
        printf("\nPrecision(%f)",cv);
        show_pi_T(p, pi, T, t);

        /* procedure for when the preciseness
         * in the convergence is achieved */
        if (cv <  prec){
          FILE *utfil;
          utfil=fopen(" Iterations ", "a");

          /* exporting the number of iterations
           * used for convergence for each simulation */
          fprintf( utfil , "%d\n", tempEM);
          fclose( utfil );
          k = tempSim*(NoOfEMsteps / sim_k);
        }

        /*  exporting converged values of the simulation */
        if ((k % (NoOfEMsteps / sim_k)) == 0) {
          if (cv >  prec){
            FILE *utfil;
            utfil=fopen(" Iterations ", "a");

            /* exporting the number of iterations
             * used for convergence if upper bound reached */

            fprintf( utfil , "%d\n", (NoOfEMsteps / sim_k));
            fclose( utfil );
          }
            SavePhasesPi(p, pi);
```

```
                    SavePhasesT(p, T);

                    /* Randomization */
                    iT = rand();
                    srand(iT);

                    /* reload the parameter values
                     * for initializations of the next simulation */
                    selectstructure(p, pi, T, t, pilegal, Tlegal);
                    init_vector(weight, NoOfObs);

                    /* resample algorithm */
                    for (i = 0; i < SumOfWeights; i++) {
                        Ra = rand() % NoOfObs;
                        weight[Ra] += 1;
                    }

                }
            }

    /* exportation. only used in modern selection */
    if ((k < 6) || ( (k % 25)==0 || k==NoOfEMsteps )) {
                if (IV == 1) {
                    SavePhases(p, pi, T);
                    printf("\nEM(%d)",k);
                    show_pi_T(p, pi, T, t);
                    compute_loglikelihood(RKstep, p, pi, T, t, stepindicator, avector, ka,
                                          NULL,k,NoOfEMsteps);


                } else {
                    if ((k % (NoOfEMsteps / sim_k)) == 0) {
                        tempSim += 1;
                    }


                }
    }
  }
}

main()
{
  int i, j, sampletype, fitting, p, NoOfEMsteps, choice, newp;
  int *pilegal, **Tlegal, NoOfInput[3]={0, 0, 0}, NoToSave;
  double *pi, **T, *t, *avector, *gvector, *bvector, **cmatrix;
```

```
  double *Bmean, *Zmean, **Nmean, **ka, **kg, **kb, ***kc, dig;
  double **a_left, **g_left, **b_left, ***c_left, *ett;

  sampletype=0;
   fitting =1;
  NoOfObs=0;
  SumOfWeights=0;

  /* start menu */
     printf("Press 1 for  initialization \n");
     printf("Press 2 for simulations\n");
     scanf("%d", &IV);
     if (IV == 2) {
          printf("Enter number of simulations\n");
          scanf("%d", &sim_k);

        printf("Enter level of precision\n");
        scanf("%lf", &dig);
        prec = pow(10,-dig);
     }
if (IV == 1){
     printf("Overwrite loglik-file ?\n");
     printf("Press 1 for yes\n");
     printf("Press 2 for no\n");
     scanf("%d", &OW);
}
  sampletype = AskForInput(NoOfInput);
   printf("\nNumber of phases of the PH-distribution to be fitted, (p): ");
  scanf("%d", &p);

  pi=v_alloc(p); T=m_alloc(p, p); t=v_alloc(p);
   pilegal=int_v_alloc(p); Tlegal=int_m_alloc(p, p);
  avector=v_alloc(p);  bvector=v_alloc(p); cmatrix=m_alloc(p, p);
  Bmean=v_alloc(p); Zmean=v_alloc(p); Nmean=m_alloc(p, p+1);
  ka=m_alloc(4, p);  kb=m_alloc(4, p); kc=m3_alloc(4, p, p);
  if (sampletype==2) {
     ett = v_alloc(p);
     for(i=0; i < p; i++)
        ett[i] = 1;
  }

   selectstructure (p, pi, T, t, pilegal, Tlegal);
  show_pi_T(p, pi, T, t);

  while ( fitting == 1) {
     printf("\nNumber of EM-iterations: ");
```

```
    scanf("%d", &NoOfEMsteps);
    EMiterate(NoOfEMsteps, p, pi, T, t, gvector, avector, bvector, cmatrix,
      Bmean, Zmean, Nmean, kg, ka, kb, kc, ett);

      if (IV == 1) {
          SavePhases(p, pi, T);
      }

      fitting = 0;
      }

  free(pi); free_matrix(T, p); free(t); free(pilegal);
   free_integermatrix(Tlegal,p);
  free(avector); free(bvector); free_matrix(cmatrix,p);
  free_matrix(ka, 4); free_matrix(kb, 4);
  free_3dimmatrix(kc, 4, p); free(Bmean); free(Zmean);
  free_matrix(Nmean, p);

  if(sampletype == 2)
    free(ett);
}
```

Visualization in R

Visualization of demonstration of modified EMpht program in R

```
library(expm)

setwd("/Users/maltelindholmstoltzerasmussen/CLionProjects/untitled/fire")
x = as.matrix(read.table("unweighted.txt"))
x <- x-1
x <- subset(x,x[,1]>0 & x[,1]<6)
x
y <- sort(x)
y[1494:1496]
write.table(x)
write.table(x, file="mymatrix.txt", row.names=FALSE, col.names=FALSE)

# histogram of sample data
```

```
par(mfrow=c(1,1))
hist(x[,1], breaks = 100,main ="",xlab="",cex.lab=1.45,cex.axis=1.5)

# importing estimated parameter values
x1 = as.matrix(read.table("phases"))
x2 = as.matrix(read.table("phases2"))
x3 = as.matrix(read.table("phases3"))
x4 = as.matrix(read.table("phases4"))
x5 = as.matrix(read.table("phases5"))
x6 = as.matrix(read.table("phases6"))

# phase-type density
dens <- function(t,d){
  r <- d[,-1]
  ipv <- d[,1]
  ev <- -r%*%(as.vector(rep(1,length(d[,1]))))
  return(ipv%*%expm(r*t)%*%ev)
  }

# phase-type distribution
dis <- function(t,d){
  r <- d[,-1]
  ipv <- d[,1]
  e <- as.vector(rep(1,length(d[,1])))
  return(1-ipv%*%expm(r*t)%*%e)
}

# plot of density
ti <- seq(0.01,6,0.01)
plot(ti,mapply(function(t){dens(t,x2)}, ti), col="black",type="S",
lwd=1,lty=1,main="",xlab="",ylab="Density")
densp <- function(d,j,i){
lines(ti,mapply(function(t){dens(t,d)}, ti), col=paste("grey",j),
type="S",lwd=1,lty=i,main="",xlab="",ylab="Density")
}
densp(x3,70,1)
densp(x4,20,2)

# plot of distribution
disp <- function(d,j,f){
  lines(ti,mapply(function(t){dis(t,d)}, ti), col=paste("grey",j),
  type="S",lwd=1,lty=f,main="",xlab="x",ylab=" Distribution ")
}

# difference in fit between 2 and 3 phases
plot(ti,mapply(function(t){dis(t,x2)}, ti)-
```

```
mapply(function(t){ dis(t,x3)}, ti ),lwd=1,lty=1,type="S")

# empirical distribution function
plot(ecdf(x),main="",xlab="",ylab=" Distribution ",
 verticals =T, do.points=F,col.01 line = NULL,cex.lab=1.45,cex.axis=1.5)
disp(x2,50,1)
disp(x3,80,2)

# plot of information criteria
ll = as.matrix(read.table(" loglik "),sep=" ")[c (2,1,3,4),]
as.numeric(ll [,9])
barplot(t(cbind(as.numeric(ll [,9]), as.numeric(ll [,11]))),
          legend = c("AIC","BIC"),beside = TRUE,ylim = range(4500,4700),
          xpd = FALSE,cex.names=1.5,names.arg=seq(1:4),
          xlab = "Phases", ylab = "Information criteria ",
          args.legend = list(x=2,bty="n",cex=1.5),
          cex.lab=1.45,cex.axis=1.5)
axis (1, labels =FALSE)

# histogram of number of iterations necessary
# for convergence for each simulation
setwd("/Users/ maltelindholmstoltzerasmussen /CLionProjects/ original / Lastanalyses ")
 it = as.matrix(read.table(" iterations "))
hist( it ,breaks = 60,xlab = " Iterations ",main ="",
cex.lab=1.45,cex.axis=1.5)

# discarding non-converged estimated
# parameter values
S <- as.matrix(read.table("phasesT"))
temp <- cbind(S,it)
Su <- subset(temp,temp[,10]<50000)

# estimated parameter values
Se <- matrix(t(as.matrix(read.table("phases" ))[,-1]),
ncol=9,byrow = T)

# calculating confidence bounds
qlv <- quv <- rep(0,9)
for (i in 1:9) {
  Le <- sort(Su[,i])
  qlv[i] <- Le[floor(length(Le)*0.025)]
  quv[i] <- Le[floor(length(Le)*(1-0.025))+1]
}

# plotting sample distributions
par(mfrow=c(3,3))
```

```
  for (i in 1:9) {
    hist(Su[,i], breaks = 60,xlab = "",main ="",
    cex.axis = 1.5,ylab = "")
    abline(v=Se[,i], lty=2)
    abline(v=qlv[i], lty=3)
    abline(v=quv[i], lty=3)
  }

# function for finding lower and upper confidence bound
conreg <- function(x,alpha){
  Ld <- sort(x)
  floor(length(Ld)*alpha)
  floor(length(Ld)*(1-alpha))+1
  return(Ld[floor(length(Ld)*alpha)])
}

# plotting quantiles
cr <- seq(0.001,0.999,0.001)
for (i in 1:9) {
plot(cr,mapply(function(t){conreg(Su[,i], t)}, cr),
xlab = "",main ="",cex.axis = 1.5,ylab = "")
  abline(h=Se[,i], lty=2)
  abline(h=qlv[i], lty=3)
  abline(h=quv[i], lty=3)
}
quv
conreg(Su [,1],0.5)
```

Visualization of asymptotic results in R

```
#Example with non-unique distribution and no initialization
setwd(paste("/Users/ maltelindholmstoltzerasmussen ",
                "/CLionProjects/ untitled /",
       sep=""))
x = as.matrix(read.table("phasesT"))
i <- 7
xt <- x[1:500,i]

#Histograms
```

```
hist(x[,i], breaks = 30,main ="",xlab="")
hist(xt,breaks = 30,main ="",xlab="")
hist(x[1:50,i] ,breaks = 10,main ="",xlab="")

#Example with non-unique distribution
setwd(paste("/Users/ maltelindholmstoltzerasmussen ",
                "/CLionProjects/ untitled / exportbootstrap /",
        sep=""))
x = as.matrix(read.table("phasesT"))

hist(x [,1], breaks = 100,main ="",xlab="")
hist(x [,2], breaks = 100,main ="",xlab="",ylim=c(0,100))

#QQ plots
qqtest <- function(x){
  qqplot(qnorm(ppoints(x)),x,
         xlab = "Normal theoretical quantiles",
         ylab = "Sample data quantiles")
  qqline(x)
}

qqtest(x [1:50,7])
qqtest(x [1:500,7])
qqtest(x [,7])

#Function for finding lower and upper confidence bound
conreg <- function(x,alpha){
  Ld <- sort(x)
  Ld
  floor(length(Ld)*alpha)
  floor(length(Ld)*(1-alpha))+1
  qu <- Ld[floor(length(Ld)*(1-alpha))+1]
  ql <- Ld[floor(length(Ld)*alpha)]
  c(ql,qu)
}

#Calculating confidence bounds as a function of the number
#of simulations
rm(quv)
quv <- rep(0,4950)
qlv <- rep(0,4950)
for (i in 50:5000) {
  quv[i-49] <- conreg(x[1:i ,7],0.025)[2]
  qlv[i-49] <- conreg(x[1:i ,7],0.025)[1]
}
cr <- cbind(qlv,quv,seq(50,5000))
```

```
plot(cr[,3], qlv, col="black", type="S", lwd=1, lty=1, main="",
     xlab="Number of simulations", ylab="97.5% quantile",
     ylim = c(0.2525,0.2537))
plot(cr[,3], quv, col="black", type="S", lwd=1, lty=1, main="",
     xlab="Number of simulations", ylab="2.5% quantile")
plot(cr[,3], qld, col="black", type="S", lwd=1, lty=1, main="",
     xlab="Number of simulations",
     ylab="Length of confidence interval")

plot(cr[,3], qlv, col="black", type="S", lwd=1, lty=1, main="",
     xlab="Number of simulations",
     ylab="Lower and upper quantile",
     ylim=c(0.25,0.27))
lines(cr[,3], quv, col="grey80", type="S", lwd=1, lty=1, main="",
      xlab="Number of simulations", ylab="0.995 quantile",
      ylim=c(0.245,0.275))

# example with multiple parameterization
# within converged starting values
setwd("/Users/maltelindholmstoltzerasmussen/CLionProjects/untitled/10p")
hist(x[,29], breaks = 30, main ="", xlab="")
hist(x[,26], breaks = 60, main ="", xlab="")
qqtest(x[,26])
qqtest(x[,23])
```

Bibliography

[1] Hirotugu Akaike. A new look at the statistical model identification. *IEEE transactions on automatic control*, 19(6):716–723, 1974.

[2] Nicolas (http://math.stackexchange.com/users/6312/andrAndrI&233. Reliability function, proving exponential distribution. Mathematics Stack Exchange. URL:http://math.stackexchange.com/q/107146 (version: 2012-02-08).

[3] Søren Asmussen and Hansjörg Albrecher. *Ruin probabilities*, volume 14. World scientific, 2010.

[4] Søren Asmussen, Olle Nerman, and Marita Olsson. Fitting phase-type distributions via the em algorithm. *Scandinavian Journal of Statistics*, pages 419–441, 1996.

[5] Peter J Bickel and David A Freedman. Some asymptotic theory for the bootstrap. *The Annals of Statistics*, pages 1196–1217, 1981.

[6] Mogens Bladt, Luz Judith R Esparza, and Bo Friis Nielsen. Fisher information and statistical inference for phase-type distributions. *Journal of Applied Probability*, 48(A):277–293, 2011.

[7] Mogens Bladt, Antonio Gonzalez, and Steffen L Lauritzen. The estimation of phase-type related functionals using markov chain monte carlo methods. *Scandinavian Actuarial Journal*, 2003(4):280–300, 2003.

[8] Mogens Bladt and Bo Friis Nielsen. Matrix-exponential distributions in applied probability, 2017.

[9] Andrea Bobbio, András Horváth, Marco Scarpa, and Miklós Telek. Acyclic discrete phase type distributions: Properties and a parameter estimation algorithm. *Performance evaluation*, 54(1):1–32, 2003.

[10] Gunter Bolch, Stefan Greiner, Hermann de Meer, and Kishor S Trivedi. *Queueing networks and Markov chains: modeling and performance evaluation with computer science applications*. John Wiley & Sons, 2006.

[11] Aldo Cumani. On the canonical representation of homogeneous markov processes modelling failure-time distributions. *Microelectronics Reliability*, 22(3):583–602, 1982.

[12] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the royal statistical society. Series B (methodological)*, pages 1–38, 1977.

[13] Bradley Efron. Bootstrap methods: another look at the jackknife. *The annals of Statistics*, pages 1–26, 1979.

[14] Bradley Efron. Bootstrap methods: another look at the jackknife. In *Breakthroughs in Statistics*, pages 569–593. Springer, 1992.

[15] Bradley Efron and Robert J Tibshirani. An introduction to the bootstrap chapman & hall. *New York*, 436, 1993.

[16] Agner Krarup Erlang. Sandsynlighedsregning og telefonsamtaler. *Nyt tidsskrift for Matematik*, 20:33–39, 1909.

[17] Luz Judith R Esparza, Bo Friis Nielsen, and Mogens Bladt. *Maximum likelihood estimation of phase-type distributions*. PhD thesis, Technical University of DenmarkDanmarks Tekniske Universitet, Department of Applied Mathematics and Computer ScienceInstitut for Matematik og Computer Science, 2010.

[18] W Feller. Analítico: introduction to probability theory and its applications. 1971.

[19] Math fun (http://math.stackexchange.com/users/195344/math fun). Memoryless property and geometric distribution. Mathematics Stack Exchange. URL:http://math.stackexchange.com/q/1074323 (version: 2014-12-19).

[20] Olle Häggström, Olle Nerman, and Søren Asmussen. *EMPHT: A Program for Fitting Phase-type Distributions*. Chalmers tekniska högskola, 1992.

[21] W Keith Hastings. Monte carlo sampling methods using markov chains and their applications. *Biometrika*, 57(1):97–109, 1970.

[22] András Horváth and Miklós Telek. Phfit: A general phase-type fitting tool. In *International Conference on Modelling Techniques and Tools for Computer Performance Evaluation*, pages 82–91. Springer, 2002.

[23] Henrik Hult and Filip Lindskog. Mathematical modeling and statistical methods for risk management, lecture notes, 1995.

[24] Arne Jensen. *A distribution model, applicable to economics*. Munksgaard, 1954.

[25] Wilhelm Kutta. Beitrag zur näherungweisen integration totaler differentialgleichungen. 1901.

[26] Nicholas Metropolis, Arianna W Rosenbluth, Marshall N Rosenbluth, Augusta H Teller, and Edward Teller. Equation of state calculations by fast computing machines. *The journal of chemical physics*, 21(6):1087–1092, 1953.

[27] Marcel F Neuts. *Matrix-geometric solutions in stochastic models: an algorithmic approach*. Courier Corporation, 1981.

[28] Marcel F Neuts. *Structured stochastic matrices of MG-1 type and their applications*. Dekker, 1989.

[29] MF Neuts. Algorithmic probability. stochastic modeling series, 1995.

[30] Søren Feodor Nielsen. The stochastic em algorithm: estimation and asymptotic results. *Bernoulli*, pages 457–489, 2000.

[31] Marita Olsson. Estimation of phase-type distributions from censored data. *Scandinavian journal of statistics*, pages 443–460, 1996.

[32] Marita Olsson. The empht-programme. *Relatório técnico, Department of Mathematics, Chalmers University of Technology, and Göteborg University, Sweden*, 1998.

[33] Philipp Reinecke, Levente Bodrog, and Alexandra Danilkina. Phase-type distributions. In *Resilience Assessment and Evaluation of Computing Systems*, pages 85–113. Springer, 2012.

[34] David Ruppert. *Statistics and data analysis for financial engineering*. Springer, 2011.

[35] Rolf Schassberger. Warteschlangen. 1973.

[36] Gideon Schwarz et al. Estimating the dimension of a model. *The annals of statistics*, 6(2):461–464, 1978.

[37] P. TCHEBYCHEF. Des valeurs moyennes (traduction du russe, n. de khanikof. *Journal de mathematiques pures et appliquees 2e serie*, pages 177–184, 1867.

[38] Roel Verbelen. *Phase-type distributions & mixtures of Erlangs*. PhD thesis, UNIVERSITY OF LEUVEN, 2013.