MASTER

A Comparison of the Likelihood Estimation of Crude Monte Carlo Against Sequential Monte Carlo for Gamma Process-Based Degradation Models With Noisy Measurement Data

Buist, Merel B.C.

Award date:
2023

Awarding institution:
University of Queensland

Link to publication

Disclaimer
This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights
Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

• Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
• You may not further distribute the material or use it for any profit-making activity or commercial gain
A Comparison of the Likelihood Estimation of Crude Monte Carlo Against Sequential Monte Carlo for Gamma Process-Based Degradation Models With Noisy Measurement Data

Master Thesis

Merel B.C. Buist
1253409

Supervisors
Dr. Slava Vaisman
Prof. Dr. Maria Vlasiou

Brisbane
May 10, 2023

TU/e EINDHOVEN UNIVERSITY OF TECHNOLOGY
THE UNIVERSITY OF QUEENSLAND
AUSTRALIA

Department of Mathematics
Faculty of Science
Acknowledgements

This thesis marks the completion of my Master’s Program Industrial and Applied Mathematics at the Eindhoven University of Technology. The project was partially conducted at Eindhoven University of Technology and partially at the University of Queensland. I am truly grateful for the opportunity to undertake such a significant project in the foreign country Australia. This experience has been one of the highlights of my academic journey and has broaden my horizons greatly.

I would like to start by expressing my sincere gratitude to my two supervisors, Slava Vaisman and Maria Vlasiou, for their invaluable support and guidance throughout this entire process. Their expertise and engagement have been crucial in helping me to arrive at this stage. This thesis is greatly inspired by Slava, his inspiration and guidance have been invaluable to the success of this project. I am highly appreciative of his willingness to take on the challenge of supervising a project from a foreign student and dedicating a substantial part of his time in the project and in my progress. Throughout the entirety of the project, he has demonstrated an immense commitment to support, always providing his assistance whenever needed. In addition, Maria agreed to oversee this project despite it not being within her area of expertise, simply to provide me the opportunity to execute this project, which I am deeply thankful for. Along the way, she has demonstrated great effort in understanding the project, posed me critical questions to help assess my comprehension of the subject matter, supported me in the process of writing a scientific thesis and encouraged me to view the problem from alternate perspectives.

I would also like to express my gratitude to Rui Castro for taking the time to serve on the assessment committee and for offering valuable feedback towards the end of the project. In addition, I would like to extend my sincere thanks to Marko Boon for his assistance in resolving computer errors and running my code. His expertise has made a considerable difference in obtaining results.

Finally, I am thankful to everyone whose support and help I have received along the way. This master’s thesis marks a meaningful milestone in my academic journey.
Abstract

The usage of the stochastic gamma process for modeling various degradation phenomena has recently gained extensive popularity. Nevertheless, in many cases, the degradation data contains measurement errors and an intractable likelihood phenomenon is coming into sight. Therefore, with the view to performing efficient statistical inference, one must obtain high-quality estimates of the corresponding likelihood. Our findings indicate that the Monte Carlo method which is a de facto state of the art used today is not adequate in practice for efficient likelihood estimation. To cope with this problem, we propose to employ the sequential Monte Carlo approach, which compares favorably with the current state of the art, leads to variance minimization, and opens the way for effective and scalable inference procedures.
Contents

1 Introduction and Literature 4
  1.1 The Degradation Model 6
  1.2 Particle Filtering / SMC 7
  1.3 Thesis Outline 8

2 Problem Description 10

3 Methods 12
  3.1 Crude Monte Carlo 12
  3.2 The failure of the CMC method 17
  3.3 Sequential Monte Carlo 20
    3.3.1 The Bootstrap Filter 22

4 Numerical Study 26
  4.1 Synthetic Data 28
    4.1.1 1 Unknown Parameter 29
    4.1.2 2 Unknown Parameters 33
    4.1.3 6 Unknown Parameters 36
  4.2 Real Data 39
    4.2.1 Extracted Real Data 39
    4.2.2 A Brief Data Analysis 40
    4.2.3 Numerical Results 44

5 Conclusions 49

6 Discussion and Recommendations for Future Research 51

References 54

Appendices 57
  A Real Degradation Data 57
  B Synthetic Data Results Code 57
    B.1 Data Generation 57
    B.2 CMC - 1 unknown 58
    B.3 CMC - 2 unknowns 61
    B.4 CMC - 6 unknowns 63
    B.5 Bootstrap - 1 unknown 66
    B.6 Bootstrap - 2 unknowns 68
    B.7 Bootstrap - 6 unknowns 69
    B.8 Figures 71
  C Real Data Results Code 74
    C.1 Data Transformation 74
    C.2 CMC and Bootstrap Result Generation 75

M.B.C. Buist 2
<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>C.3</td>
<td>Figures</td>
<td>78</td>
</tr>
<tr>
<td>D</td>
<td>Bootstrap Functions</td>
<td>80</td>
</tr>
<tr>
<td>D.1</td>
<td>Execution</td>
<td>80</td>
</tr>
<tr>
<td>D.2</td>
<td>Functions</td>
<td>82</td>
</tr>
<tr>
<td>D.3</td>
<td>Model</td>
<td>84</td>
</tr>
<tr>
<td>E</td>
<td>Data Analysis Code</td>
<td>85</td>
</tr>
</tbody>
</table>
1 Introduction and Literature

Many complex systems and products that empower our daily activities are subject to degradation. The latter affects the system’s lifetime, the availability of services, and, the corresponding safety of usage. Nature, the human body, medicine and hardware are a few of the numerous examples of critical systems that undergo degradation. Everywhere where you look around you, you can find systems that deteriorate over time. These examples include for instance the decay in the fish population, the decay of organs over time, vaccines decaying in the time they are shelved or a nuclear reactor deteriorating. Therefore, the reliable buildup of reliability management and prognostic plans is of crucial importance. This work aims to provide an important step forward in understanding and managing general degradation processes. Specifically, we concentrate on the problem of the analysis of degradation data. Such data usually contains measurement errors and as a consequence, reliability scientists and practitioners need to deal with intractable likelihood phenomena when performing statistical inference tasks. In this thesis, the objective is to develop and test a particle-based method for a reliable estimation of intractable likelihood functions.

Having in mind that it is of great importance to estimate degradation processes as precisely as possible, we aim to implement and compare the computation of the likelihood through Crude Monte Carlo (CMC) with Sequential Monte Carlo (SMC). We will show both numerically and theoretically, that SMC outperforms CMC. The result is of great theoretical and practical importance since today, various estimation algorithms rely on CMC for the corresponding likelihood function estimation.

Degradation processes have an impact on our lives and there exist numerous examples of degradation processes in various domains. For instance, degradation processes appear in ecology (water quality, forest degradation, deterioration of air quality due to pollution, and fish population decay), solar panels output degradation, plane engines lifetime, corrosion, medicine, progression of chemical reactions, and many more. One important application which affects our life is the deterioration of drugs in medicine. For example, the United States Food and Drug Administration requires the corresponding shelf-life to be indicated on the immediate container label. This requirement is based on the fact that active drug components are subject to deterioration. A more recent example is the mRNA-based vaccination. While mRNA vaccines are well positioned to take the lead role as the fastest vaccine candidates for COVID-19, mRNA molecules were shown to spontaneously degrade even when refrigerated. Hence, accurate modeling of the mRNA degradation process is of urgent importance.

System lifetime data has been used extensively by reliability analysts for a long time. Nevertheless, the advantage of considering degradation processes was recognized in the 1990s. This is due to the fact that many systems of interest are extremely reliable (for example, nuclear reactors), and the time to failure can be prohibitively large. The latter is problematic from the point of view of an adequate reliability analysis.

In addition, the study of degradation data as compared to lifetime data has the following advantage: the degradation data is more informative for highly reliable components that
have a long lifetime. During the study of system degradation data, the data analyst can assess many important properties such as the degradation rate, and the remaining useful life, without waiting for an actual failure to occur. Moreover, a reliability engineer can avoid the need to design complex and expensive accelerated life testing experiments. Finally, if the available data is censored, the degradation analysis can be our only hope to assess the system’s reliability (Hamada, Wilson, Reese, & Martz, 2008).

Figure 1: Three systems (components) in the figure are assumed to be in proper working condition before their initial usage (at time zero). Then, each system starts to operate and a deterioration of its initial health level occurs. This process continues until the system reaches a predetermined health threshold (called the critical health value), for which, the system is assumed to enter a down state. This figure also shows a component with abnormal degradation behavior (denoted by dashed line). Such an atypical behavior is generally assumed to be rare, but for mission-critical and highly reliable systems, extreme scenarios and their corresponding distribution are of critical importance. This extreme behavior corresponds to the rare event — the so-called black swan phenomenon.

By rigorously analyzing degradation mechanisms, it is possible to design good prognostic plans and to gain important information that will eventually help to improve the system’s reliability, minimize down times, and improve the corresponding safety of usage. Figure 1 shows a simplified setting with four degradation trajectories that correspond to four identical systems or components. Three typical degradation scenarios are plotted using a solid line. In addition, there is one atypical degradation scenario depicted by a dashed line. The horizontal dotted line stands for the system’s critical health threshold beyond which, a replacement or a maintenance procedure is required. In practice, people are interested in knowing the probability of the atypical degradation scenario. Therefore, it is of great importance to find the true model parameters, as then one can determine this bad scenario probability and take appropriate preventative measures.
1.1 The Degradation Model

This thesis studies a stochastic model for degradation processes that is based on the gamma process, inspired by (Hazra, Pandey, & Manzana, 2020). There are various approaches to modelling degradation. The study of (Yea & Xie, 2015) emphasized that the random nature of unobserved external factors results in unexplained randomness in the measured degradation. Modeling the variability in degradation processes, arising from both innate randomness and unexplained randomness caused by environmental factors, naturally lends itself to the utilization of stochastic processes. In the past, next to gamma processes, random effect models and Wiener processes have been applied as well for stochastic degradation modeling. Wiener processes are widely applied in degradation modeling as the differences in the values over time are normally distributed and, due to the central limit theorem, the increase in degradation of a system over time should approximately follow a normal distribution as well. In 1975, the use of the gamma process in reliability modeling was introduced. Gamma processes seem to be more suitable for systems that show degradation through fatigue, as it displays monotone and gradual degradation. For Wiener processes on the other hand, monotone degradation breaches the fundamental premises. Moreover, usually the influence from the outside environment is the main cause of the degradation of a process. While it is a main cause, the influence of the environment however is often quite little, as one can imagine that in normal circumstances there will not be a humongous effect over a short time. To be more specific, the impact time of the outside environment follows a compound Poisson distribution, which is exactly what a gamma process follows. Hence, making the gamma distribution a great fit (Chen, Lio, Ng, & Tsai, 2017).

By noting that the probability density function of the gamma process converges to zero for large values, one can arrive at the conclusion that relative to the parameters, the degradation over time will not go to extremely large values. Due to the setup of the model applied in this thesis, the larger the time difference, the larger the shape parameter of the gamma process. This makes higher degradation rates to be more likely. Intuitively, it is sensible that an increased time difference will result in larger degradation.

The above discussion implies that a wide usage of the gamma process for degradation modeling is not surprising. Examples of such applications are inspection analysis, such that a profound estimation on when maintenance is needed can be made (Bakker & van Noortwijk, 2004). Moreover, (Lawless & Crowder, 2004) implements a random effect in a gamma process to model crack expansion. The random effect is incorporated as even though the objects should be the same, in real life one still observes different degradation, some phone batteries might degrade quicker than others for the same phone with the same usage for instance. The extension of a random effect could comprehend this difference. In addition, to optimize the policy for maintenance with noisy data observations, (Bousquet, Fouladirad, Grall, & Paroissin, 2015) applies the gamma process to model crack expansion. Furthermore, the gamma process has been applied to model corrosion damage, (Kallen & Van Noortwijk, 2005) applies the model on data of a pressurized steel vessel, in order to determine a guideline for check ups and maintenance. The guideline that is determined is one which will minimize the yearly mean expenses. Additionally, (Pandey, Lu, & Komljenovic, 2011) models corrosion of a nuclear reactor through both a gamma process model and a random variable rate model.
A Comparison of CMC Against SMC

While the study cannot conclude which model performs better, it does state that the RV model has a fixed rate of degradation throughout the entire lifetime, which hinders it from capturing the variance of degradation over time. The gamma model on the other hand is able to capture this. As such, the variance of the degradation over time rises linearly for the gamma process model while it rises quadratically for the RV model. Two of the same authors as the previous study published another paper (Lu, Pandey, & Xie, 2013) dedicated to the exploration of the gamma process in noisy degradation data two years later, where the process is applied anew for corrosion in a nuclear reactor. (Qin, Zhou, & Zhang, 2015) employs a gamma process to model the occurrence and development of corrosion in steel gas and oil pipelines. Next to the multiple examples mentioned already, the degradation of infrastructure is another example where the gamma process has been applied (Frangopol, Kallen, & Van Noortwijk, 2004), just as the modeling of concrete creep (Cinlar, Bazant, & Osman, 1977). To conclude, the application of the gamma process in noisy degradation data is quite extensive, as it is applicable in numerous different types of fields. It allows for a promising model which deserves to be explored even more.

In this thesis, we focus on the effect of measurement errors. This implies that the corresponding likelihood function is noisy, or, in other words, the likelihood function is intractable. The latter imposes a serious theoretical and practical challenge. Following (Hazra et al., 2020), our main focus is on normally distributed measurement errors. While the setting in normally distributed measurement errors is common, it is instructive to note that other studies showed that in some cases, the noise might follow a different law. For (Corbetta, Sbarufatti, Giglio, & Todd, 2018) considers an unbiased log-normal process noise. Their study suggests a similar performance for both the additive Gaussian noise and their presented noise term. However, the normally distributed noise might fail to perform well as compared to the log-normal process noise, when the corresponding process variance is large. As discussed above, we focus on the Gaussian noise.

In particular, in this thesis, we apply the same model as (Hazra et al., 2020), which considers the gamma process to noisy measurement degradation data for both synthetic data as well as real data. The study employs the additive Gaussian noise. It is important to note that the Gaussian setting does not impose a serious limitation on the methods developed in our study, since the proposed algorithms are not sensitive to the error model choice.

This work concentrates on the SMC algorithm. We provide a brief introduction in Section 1.2.

1.2 Particle Filtering / SMC

Particle filtering, or Sequential Monte Carlo (SMC), is a computational technique that efficiently tracks and estimates hidden states in a dynamic system by using a set of weighted particles to approximate the posterior distribution of such states. The first research that introduces particle filtering in degradation modeling is the one of (Gordon, Salmond, & Smith, 1993), where the authors intended to combat the restrictions of linearity and Gaussian noise in the corresponding model. Particle filtering namely allows for nonlinear processes that are defined by non-Gaussian random variables (Corbetta et al., 2018). Thus, regarding the gamma process intertwined in the model which is non-Gaussian, this technique sounds...
promising. Due to particle filtering being a less restrictive algorithm, the method is quickly gaining popularity for estimation, with applications ranging in a wide spread of engineering systems. Applications have occurred in various scientific domains. Several times for instance in the parameter estimation in nonlinear structural systems (Chatzi & Smyth, 2014; Corbetta et al., 2018), in specific in (Xue, Tang, & Xie, 2009) particle filtering has been applied for structural damage detection, where one tracks whether there are sudden parameter changes which could indicate damage to the foundation of a building. More applications are in the identification of the development of cracks (Compare, Baraldi, Turati, & Zio, 2015), in remaining useful life forecasting of turbine blades (Baraldi, Cadini, Mangili, & Zio, 2013) and in battery health forecasting (Goebel, Saha, Saxena, Celaya, & Christophersen, 2008; Saha, Poll, Goebel, & Christophersen, 2007). Furthermore, particle filters have additionally been applied to models relying on gamma processes before, (Schirru, Pampuri, & De Nicolao, 2010) for instance applies this method for predictive maintenance.

The SMC algorithm is very promising for this setting, due to the fact that it can comprehend the gamma distributed degradation increases. There are different implementations of particle filtering. The most common and straightforward one, is the Bootstrap particle filter. More general and advanced filtering algorithms include the guided particle filter and the auxiliary particle filter (Chopin & Papaspiliopoulos, 2020). While Bootstrap is a very prevailing filter (Corbetta et al., 2018), the auxiliary particle filter is gaining popularity and has shown better performances than the well-known Bootstrap filter in several research studies. (Lopes & Tsay, 2011) shows promising results in the possible use in econometrics and financial time series analysis. (Jinan & Raveendran, 2016) presents its application in target tracking, where the observations contain measurement noise as well. (Xi, Peng, Kitagawa, & Chen, 2015) implements an auxiliary iterated extended filter for processes with gamma noise and with Gaussian noise, and shows it outperforms multiple other particle filters. Lastly, (Douc, Moulines, & Olsson, 2009) explores the auxiliary particle filter and some asymptotic properties of the guided particles, which also shows better results than the trivial Bootstrap filter.

1.3 Thesis Outline

The major contribution of this study is as follows.

1. We show that the practice of relying on CMC for the noisy likelihood evaluation is not adequate. Specifically, we rigorously show that CMC will most likely fail to estimate the intractable likelihood function that arises from the gamma process for degradation data in an adequate fashion. This problem occurs because the CMC algorithm enters the rare-event setting. Therefore, state-of-the-art estimation algorithms should not rely on CMC.

2. We propose to apply the particle filter SMC algorithm that avoids the rare-event trap and thus becomes a viable alternative to the CMC method.

3. We show that the SMC algorithm outperforms the CMC method on several synthetic and real-life data examples. The SMC algorithm demonstrates to be more reliable, due to its smaller relative error and larger differences in likelihood.
4. Finally, we provide a research software package that can be used for intractable likelihood function estimation.

The rest of the study is organized as follows. In Section 2, we present the Problem Description in greater detail. Section 3 gives an introduction to CMC and SMC methods used in this study. In Section 4 we perform a comprehensive numerical evaluation of the proposed methodology. Finally, Section 5 summarizes our findings whereas Section 6 discusses the findings and proposes some ideas for future research.
2 Problem Description

Following (Hazra et al., 2020), the degradation model under consideration relies on the gamma process. Specifically, we consider a system with \( m \) components that are subject to degradation. The corresponding degradation for component \( i \in \{1, \ldots, m\} \) will be measured at each time \( t(l), l \in \{1, \ldots, n\} \). All components are independent realizations of a stochastic gamma process; for example, one can consider the degradation of \( m \) different batteries. As exact measurement usually requires a destruction or dismantling of the system, the observed data contains measurement errors. Thus, the stochastic process of degradation will contain a term representing the measurement error that varies per component \( i \) and time \( t(l) \), which will be referred to as \( Z_i(t(l)) \). Moreover, each component, i.e. replication of the system, starts with an initial degradation \( A_i \). For instance when observing a battery, it might already have a degradation of 2 percent; Hence, it can only charge until 98 percent of the original battery life. Last, for each component there is the degradation that occurs over time, which will be denoted by \( X_i(t(l)) \). The battery for example deteriorates over time, the amount it has deteriorated from time zero to time \( t(l) \) equals \( X_i(t(l)) \). Together, these random variables form the entire degradation process:

\[
Y_i(t(l)) = A_i + X_i(t(l)) + Z_i(t(l)).
\] (1)

The initial degradation \( A_i \) is independent and normally distributed with mean \( \mu_A \) and variance \( \sigma_A^2 \) for all \( i \in \{1, \ldots, m\} \); That is, \( A_i \sim i.i.d. N(\mu_A, \sigma_A^2) \). The measurement error \( Z_i(t(l)) \) is also independent and normally distributed with mean 0 and variance \( \sigma_Z^2 \) for all \( i \in \{1, \ldots, m\} \); That is, \( Z_i(t(l)) \sim i.i.d. N(0, \sigma_Z^2) \). Lastly, the degradation over time has an increase in each time step, this increase is modeled by a gamma distribution. Hence, by defining

\[
X_i(t(l)) - X_i(t(l - 1)) = \Delta X_i(t(l)),
\]

we say that \( \Delta X_i(t(l)) \) is gamma distributed, which is also independent for all \( i \in \{1, \ldots, m\} \). Specifically, similar to (Hazra et al., 2020), we define \( \Delta X_i(t(l)) \sim i.i.d. Gamma(\alpha t(l)^n - \alpha t(l - 1)^n, \beta) \) for all \( l \in \{1, \ldots, N\} \). Here, \( \alpha t(l)^n - \alpha t(l - 1)^n \) is the shape parameter and \( \beta \) is the scale parameter of the gamma distribution.

The above choice of shape and scale parameters allows for a lot of freedom in the model. Note that \( t(0) = 0 \), as this is the starting point. \( X_i(0) \) is also 0, as no time has passed yet, and as there is no measurement at time 0, \( Z_i(0) \) equals 0 as well. Therefore, \( Y_i(0) = A_i \). Given the gamma distribution, the degradation over a positive time frame will always be positive, as the probability of \( \Delta X_i \) being smaller than or equal to 0 equals 0. Throughout this thesis, we refer to (1) including the distributions linked to the random variables in that equation as the model used for this research. As stated above, this model has recently gained extensive popularity in literature (for additional details, please see (Hazra et al., 2020)). Key limitations of the model however are that the mean and variance are linked to one another, namely, \( Var(\Delta X_i(t(l))) = E[\Delta X_i(t(l))] \cdot \beta \). The scale parameter \( \beta \) is time independent, hence, the mean and variance must move with each other, the ratio between the two remains the same over time. For some systems however, it might be that over time the ratio changes. One can imagine the variance in degradation over time being larger after a while compared to
A Comparison of CMC Against SMC

the mean as the probability of a sudden shock might increase. Including a time component in the scale parameter could address such a probability. However, this complicates the model and for short-term processes like machine maintenance or medicine efficacy prediction, this complication is not required. Moreover, the gamma process is not suitable for systems that degrade due to sporadic shocks. Heaps of systems show monotone and gradual degradation however, which the gamma process does model well (Gorjian, Ma, Mittinty, Yarlagadda, & Sun, 2009).

With the above model definition, the “better performing method” for the likelihood estimation can be further elaborated. There are namely various parameters in the model defined in (1), which are \( \theta = (\alpha, \eta, \beta, \mu_\Lambda, \sigma_\Lambda, \sigma_\Sigma) \). Eventually, the main goal is to estimate these parameters of the degradation process based on available data in the “best possible way”, with the view to inferring both the system deterioration properties and providing the most plausible prediction of the future degradation of this system.

In the data collection process, degradation data \((y_1, \ldots, y_m)\) is observed for all components \(i \in \{1, \ldots, m\}\). To perform parameter inference, the likelihood of the observed data given a parameter set \(\theta\), denoted as \(L(\theta|y_1, \ldots, y_m)\), needs to be evaluated. However, due to introduced measurement errors in the model, the likelihood function becomes intractable, see Section 3 for details. The intractability of the likelihood function complicates the task of finding the maximum likelihood. Therefore, one can only aim to obtain an estimator for the true value of \(L(\theta|y_1, \ldots, y_m)\).

The method used for estimating the likelihood value \(L(\theta|y_1, \ldots, y_m)\) is of key importance to the performance of the maximum likelihood estimation, which is why we aim to find the “best performing method”. Unreliable likelihood estimates, i.e. estimates with high variance, can harm any reasonable optimization procedure. Therefore, in this work, we aim to explore two estimators of \(L(\theta|y_1, \ldots, y_m)\); the CMC and the SMC. The detailed theoretical setup is provided in Section 3.
3 Methods

The main goal is to estimate the parameters of the degradation model as precisely and correctly as possible (i.e. with the least distance to the true values) assuming that the data indeed follows the gamma process model.

Regardless of the optimization algorithm that is going to be used for obtaining maximum likelihood estimators, for a given parameter vector \( \theta \), we aim to examine two estimation methods of \( \mathcal{L}(\theta | y_1, \ldots, y_m) \). In particular, the two different methods performed are both implementations of Monte Carlo simulation, namely Crude Monte Carlo and Sequential Monte Carlo. Since both CMC and SMC are randomized algorithms, we would like to work with one that has the lowest variance, if possible, for any parameter set \( \theta \). There is the, relatively, simple CMC algorithm, which we compare to the more advanced SMC method. Both methods were implemented and tested. These will be elaborated in the upcoming subsections. Specifically, we consider the CMC and the SMC algorithm in Section 3.1 and Section 3.3, respectively.

3.1 Crude Monte Carlo

To find the likelihood value estimate of a certain parameter set we must always have observed data. Hence for this estimation we suppose we have the degradation data \( y_i = (y_{i,1}, \ldots, y_{i,n}) \) for each component \( i \in \{1, \ldots, m\} \), which is observed through measurements. The method of estimating the likelihood through Crude Monte Carlo is a method where one simply computes the mean over all \( N \) runs. Suppose we have the degradation data \( y_i = (y_{i,1}, \ldots, y_{i,n}) \) for each component \( i \in \{1, \ldots, m\} \) and define the parameter vector \( \theta = (\alpha, \eta, \beta, \mu_A, \sigma_A, \sigma_Z) \). Then, the likelihood function of component \( i \) is the function \( f(y_i | \theta) = \mathcal{L}(\theta | y_i) \) over \( \theta \). Since all components are assumed to be independent of one another, the likelihood of all observed data given the parameter set is

\[
\mathcal{L}(\theta | y_1, \ldots, y_m) = \prod_{i=1}^{m} \mathcal{L}(\theta | y_i).
\]

Given the model in 1, we can derive the likelihood \( \mathcal{L}(\theta | y_i) = f(y_i | \theta) \). Note that \( \Delta y_{i,t} = \Delta x_{i,t} + \Delta z_{i,t} \), then the likelihood equals:

\[
\begin{align*}
f(y_i | \theta) &= f(y_{i,1}, \ldots, y_{i,n} | \theta) \\
&= \int_{\mathbb{R}^n} \int_{\mathbb{R}^n} f(y_{i,1}, \ldots, y_{i,n} | A_i = a_i, Z_i = z_i) f(a_i) f(z_i) da_i dz_i \\
&= \int_{\mathbb{R}^n} \int_{\mathbb{R}^n} f(y_{i,1} - y_{i,0}, \ldots, y_{i,n} - y_{i,n-1} | A_i = a_i, Z_i = z_i) f(a_i) f(z_i) da_i dz_i \\
&= \int_{\mathbb{R}^n} \int_{\mathbb{R}^n} f(\Delta x_{i,1} + \Delta z_{i,1}, \ldots, \Delta x_{i,n} + \Delta z_{i,n} | A_i = a_i, Z_i = z_i) f(a_i) f(z_i) da_i dz_i \\
&= \int_{\mathbb{R}^n} \int_{\mathbb{R}^n} f(\Delta x_{i,1}, \ldots, \Delta x_{i,n} | A_i = a_i, Z_i = z_i) f(a_i) f(z_i) da_i dz_i \\
&= \int_{\mathbb{R}^n} \int_{\mathbb{R}^n} \prod_{i=1}^{n} f(\Delta x_{i,i} | A_i = a_i, Z_i = z_i) f(a_i) f(z_i) da_i dz_i
\end{align*}
\]
where $\Gamma(\cdot)$ and $\mathbb{1}$ stand for the gamma function and the indicator function, respectively. As $\Delta x_{i,t} = \Delta y_{i,t} - \Delta z_{i,t}$, $y_{i,0} = a_i$, and $(y_{i,1}, \ldots, y_{i,n})$ is observed, knowing the initial degradation $a_i$ and the measurement errors $z_i$, the degradation $\Delta x_{i,t}$ can be computed. Thus the expectation is taken over $a_i$ and $z_i$. Given this expression for $f(y_i|\theta)$, the likelihood of the degradation data $(y_1, \ldots, y_m)$ given the parameter set $\theta$ equals

$$
\mathcal{L}(\theta|y_1, \ldots, y_m) = \prod_{i=1}^{m} f(y_i|\theta) = \prod_{i=1}^{m} \mathbb{E}_{a_i, z_i} \left[ \prod_{t=1}^{n} \frac{\Gamma(\alpha(t))^n - \alpha(t - 1)^n}{\Gamma(\alpha(t)^n - \alpha(t - 1)^n)^n} \exp\left(\frac{-\Delta x_{i,t}}{\beta}\right) \mathbb{1}\{\Delta x_{i,t} \geq 0\} \right]
$$

where (*) is due to independence. This likelihood can subsequently be estimated via simulation. The procedure is as follows. First, draw a value for $A_i$ (the initial degradation at time 0), for each $i \in \{1, \ldots, m\}$ given the distribution defined in model (1), and draw values $z_i$ for each $i \in \{1, \ldots, m\}$ given their distribution. With these values, one can compute $\Delta x_{i,t}$ for all $l$ and all $i$ and thereby compute the product inside the expectation. Performing this computation for $N$ runs and computing the mean of the outcome leads to an approximation of $\mathcal{L}(\theta|y_1, \ldots, y_m)$:

$$
\hat{\mathcal{L}}(\theta|y_1, \ldots, y_m) = \frac{1}{N} \sum_{j=1}^{N} \prod_{i=1}^{m} \prod_{t=1}^{n} \frac{\Gamma(\alpha(t))^n - \alpha(t - 1)^n}{\Gamma(\alpha(t)^n - \alpha(t - 1)^n)^n} \exp\left(\frac{-\Delta x_{i,t}}{\beta}\right) \mathbb{1}\{\Delta x_{i,t} \geq 0\}
$$

The estimator $\hat{\mathcal{L}}(\theta|y_1, \ldots, y_m)$ is unbiased. To see this, note that it holds that

$$
\mathbb{E} \left[ \hat{\mathcal{L}}(\theta|y_1, \ldots, y_m) \right] = \mathbb{E} \left[ \prod_{j=1}^{N} \prod_{i=1}^{m} \frac{\Gamma(\alpha(t))^n - \alpha(t - 1)^n}{\Gamma(\alpha(t)^n - \alpha(t - 1)^n)^n} \exp\left(\frac{-\Delta x_{i,t}}{\beta}\right) \mathbb{1}\{\Delta x_{i,t} \geq 0\} \right]
$$

$$
= \frac{1}{N} \sum_{j=1}^{N} \mathbb{E} \left[ \prod_{i=1}^{m} \prod_{t=1}^{n} \frac{\Gamma(\alpha(t))^n - \alpha(t - 1)^n}{\Gamma(\alpha(t)^n - \alpha(t - 1)^n)^n} \exp\left(\frac{-\Delta x_{i,t}}{\beta}\right) \mathbb{1}\{\Delta x_{i,t} \geq 0\} \right]
$$

$$
= \frac{1}{N} \sum_{j=1}^{N} \prod_{i=1}^{m} \prod_{t=1}^{n} \mathbb{E} \left[ \frac{\Gamma(\alpha(t))^n - \alpha(t - 1)^n}{\Gamma(\alpha(t)^n - \alpha(t - 1)^n)^n} \exp\left(\frac{-\Delta x_{i,t}}{\beta}\right) \mathbb{1}\{\Delta x_{i,t} \geq 0\} \right]
$$

$$
= \frac{1}{N} \sum_{j=1}^{N} \prod_{i=1}^{m} \prod_{t=1}^{n} f(y_{i,t}|\theta) = \frac{1}{N} \sum_{j=1}^{N} \prod_{t=1}^{n} f(y_{i,t}|\theta) = \frac{1}{N} \sum_{j=1}^{N} \prod_{t=1}^{n} \mathcal{L}(\theta|y_t)
$$
\[ = \frac{1}{N} \sum_{j=1}^{N} \mathcal{L}(\theta | y_1, \ldots, y_m) = \mathcal{L}(\theta | y_1, \ldots, y_m) \]

where (*) is due to the independence of the terms inside the expectation.

**Algorithm 1** Crude Monte Carlo

```plaintext
for all runs j from 1 to N do
  for all components i from 1 to m do
    Draw \( a_i \) from \( N(\mu_A, \sigma_A^2) \)
    Set \( y_{i,0} = a_i \)
    Set \( z_{i,0} = 0 \)
    for time periods \( l \) from 1 until n do
      Draw \( z_{i,l} \) from \( N(0, \sigma_Z^2) \)
      \( \Delta x_{i,l}^j = y_{i,l} - y_{i,l-1} - z_{i,l} + z_{i,l-1} \)
    end
  end
end
```

\( \hat{\mathcal{L}}(\theta | y_1, \ldots, y_m) = \frac{1}{N} \sum_{j=1}^{N} \prod_{i=1}^{m} \prod_{l=1}^{n} \frac{(\Delta x_{i,l}^j)^{(\alpha t(l)^\eta - \alpha t(l-1)^\eta - 1)} \exp(-\frac{\Delta x_{i,l}^j}{\beta}) \mathbb{I}\{\Delta x_{i,l}^j \geq 0\}}{\Gamma(\alpha t(l)^\eta - \alpha t(l-1)^\eta) \beta^\alpha t(l)^\eta - \alpha t(l-1)^\eta} \]

This approximation of the likelihood value will be executed for \( N_{\text{part}} \) particles, and it will be executed \( N_{\text{sim}} \) times. This is done through the programming language Matlab. One run of \( N_{\text{sim}} \), i.e. one approximation of the likelihood given parameter set \( \theta \) through the method of CMC that has been explained above, is described in pseudocode in Algorithm 1.

Due to small values involved, a numerical stability problem arises when implementing this in computer software. To ease the explanation, let us define

\[ W_{i,l}^j := \frac{(\Delta x_{i,l}^j)^{(\alpha t(l)^\eta - \alpha t(l-1)^\eta - 1)} \exp(-\frac{\Delta x_{i,l}^j}{\beta}) \mathbb{I}\{\Delta x_{i,l}^j \geq 0\}}{\Gamma(\alpha t(l)^\eta - \alpha t(l-1)^\eta) \beta^\alpha t(l)^\eta - \alpha t(l-1)^\eta} \]

If for a certain \( j \) at least one indicator \( \mathbb{I}\{\Delta x_{i,l}^j \geq 0\} \) over all \( i \) and \( l \) is false, then the entire term \( \prod_{i=1}^{m} \prod_{l=1}^{n} W_{i,l}^j \) equals zero. However, if all indicators are satisfied, then \( \prod_{i=1}^{m} \prod_{l=1}^{n} W_{i,l}^j \) as well as \( W_{i,l}^j \) will probably take a very small value, and thus possibly causing numerical instability in the program when computing the estimator of the likelihood. Therefore, a diversion using the logarithm is applied, as this results in larger numbers which can better be processed by Matlab. The diversion works as follows. First, note that

\[ \hat{\mathcal{L}}(\theta | y_1, \ldots, y_m) = \frac{1}{N} \sum_{j=1}^{N} \prod_{i=1}^{m} \prod_{l=1}^{n} W_{i,l}^j = \frac{1}{N} \sum_{j=1}^{N} \exp \left\{ \log \left( \prod_{i=1}^{m} \prod_{l=1}^{n} W_{i,l}^j \right) \right\} \]

\[ = \frac{1}{N} \sum_{j=1}^{N} \exp \left\{ \sum_{i=1}^{m} \sum_{l=1}^{n} \log (W_{i,l}^j) \right\} \]
A Comparison of CMC Against SMC

holds. If at least one indicator in \( W_{i,l}^j \) over all \( i \) and \( l \) is false, the term \( \prod_{i=1}^{n} \prod_{l=1}^{n} W_{i,l}^j \) equals zero; therefore these CMC runs \( (j) \), do not need any further derivation. Considering the runs \( j \) where all indicators are satisfied, \( \log W_{i,l}^j \) equals to:

\[
\log W_{i,l}^j = \log \left( \left( \Delta x_{i,l}^j \right)^{(\alpha(t) - \alpha(t-1))} \exp \left\{ \frac{-\Delta x_{i,l}^j}{\beta} \right\} \right) 
- \log \left( \Gamma(\alpha(t) - \alpha(t-1)) \right)

= (\alpha(t) - \alpha(t-1) - 1) \log \left( \Delta x_{i,l}^j \right) + \frac{-\Delta x_{i,l}^j}{\beta} - \log \left( \Gamma(\alpha(t) - \alpha(t-1)) \right)

+ (\alpha(t) - \alpha(t-1)) \log \left( \beta \right). 
\]

In addition, the manner in which this diversification is used to derive the likelihood estimator is of importance. The terms \( \exp \left( \sum_{i=1}^{n} \sum_{l=1}^{n} \log W_{i,l}^j \right) \) used in the estimation of the likelihood can be very small values, leading to a numerical stability problem. A common solution is to take the logarithm of the estimator and apply the following diversion, which is also called the exponential trick. Now, define \( U_j = \sum_{i=1}^{n} \sum_{l=1}^{n} \log W_{i,l}^j \) and suppose that \( c = \max\{U_1, \ldots, U_N\} \), since this choice works well usually. The first sum in the likelihood estimator includes all \( j \)’s, hence possibly also \( j \)’s where at least one indicator \( 1 \{ \Delta x_{i,l}^j \geq 0 \} \) over all \( i \) and \( l \) is false. For those \( j \)’s in our implementation, we set \( U_j \) equal to \(-10000\); this is done for the numerical stability reason and the \(-10000\) value signifies the minus infinity.

The log likelihood estimator is subsequently derived as

\[
\log \left( \hat{L}(\theta|\mathbf{y}_1, \ldots, \mathbf{y}_m) \right) = \log \frac{1}{N} \sum_{j=1}^{N} \exp(U_j) = - \log(N) + \log \sum_{j=1}^{N} \exp(U_j - c + c)

= - \log(N) + \log \left( \exp(c) \sum_{j=1}^{N} \exp(U_j - c) \right)

= - \log(N) + c + \log \sum_{j=1}^{N} \exp(U_j - c). 
\]

With these diversifications, the implemented code is described in pseudocode in Algorithm 2. The \( \log \left( \hat{L}(\theta|\mathbf{y}_1, \ldots, \mathbf{y}_m) \right) \) that is returned from the program can be used for maximum likelihood estimation of the parameters. To compare the performance and hence the variance of the estimator with that of SMC methods, the code in Algorithm 2 is run \( N_{\text{sim}} \) number of times.

While the CMC method is both numerically stable and relatively simple to implement, it unfortunately can be very unreliable. Namely, the CMC algorithm can have a prohibitively large variance. This happens since CMC falls into the so-called rare-event trap. We discuss this issue in Section 3.2.
Algorithm 2 Diversified Crude Monte Carlo

Create array $\bar{\alpha}$ with length the number of time steps $n$

$\bar{\alpha}(1) = \alpha \cdot (t(1)^n - 0^n)$

for number of time periods $l$ from 2 until $n$ do

$\bar{\alpha}(l) = \alpha \cdot (t(l)^n - t(l-1)^n)$

end

$\ln \text{Gamma} = \sum_{j=1}^{m} \sum_{l=1}^{n} \log \left( \Gamma \left( \alpha \cdot (t(l)^n - t(l-1)^n) \right) \right) = m \sum_{l=1}^{n} \log \left( \Gamma \left( \alpha \cdot (t(l)^n - t(l-1)^n) \right) \right)$

$\ln \text{Beta} = \sum_{j=1}^{m} \sum_{l=1}^{n} (\alpha \cdot (t(l)^n - t(l-1)^n)) \log(\beta) = m \sum_{l=1}^{n} (\alpha \cdot (t(l)^n - t(l-1)^n)) \log(\beta)$

Set vector, of length $N$, $U = -10000$

for all runs $j$ from 1 to $N$ do

Draw vector $a$ of length $m$, each element with distribution $N(\mu_A, \sigma_A^2)$

Set $y_0 = a$

Set $Y = [y_0 \ y_1 \ \ldots \ \ y_n]$

Set $z_0 = 0$

Draw matrix $Z'$ of size $m \times (n-1)$, each element with distribution $N(0, \sigma_Z^2)$

Set $Z = [z_0 \ Z']$

$\Delta X = Y(:, 2 : end) - Y(:, 1 : end - 1) - Z(:, 2 : end) + Z(:, 1 : end - 1)$

if $\Delta X > 0$ then

$U_j = \sum_{i=1}^{m} \sum_{l=1}^{n} ((\bar{\alpha}(l) - 1) \log(\Delta X_{i,l})) - \sum_{i=1}^{m} \sum_{l=1}^{n} (\Delta X_{i,l}) / \beta - \ln \text{Gamma} + \ln \text{Beta}$

end

$c = \max\{U_1, \ldots, U_N\}$

$\log \left( \hat{L}(\theta | y_1, \ldots, y_m) \right) = - \log(N) + c + \log \left( \sum_{j=1}^{N} \exp(U_j - c) \right)$
3.2 The failure of the CMC method

In order to get a better understanding of the rare-event phenomenon, consider the following general setting. Let $\mathcal{X}$ and $\mathcal{X}^* \subseteq \mathcal{X}$ be sets and suppose that we would like to estimate the probability $\ell = |\mathcal{X}^*|/|\mathcal{X}|$, where $\mathcal{X}^*$ are rare-events; hence $\ell$ is the probability of a rare-event. Let $X$ be a random variable distributed uniformly over the $\mathcal{X}$ set. Then, it holds that $E[\mathbb{1}\{X \in \mathcal{X}^*\}] = \ell$. The CMC estimator then equals

$$\hat{\ell}_{\text{CMC}} = \frac{1}{N} \sum_{i=1}^{N} \mathbb{1}\{X_i \in \mathcal{X}^*\}.$$  

Note that $\mathbb{1}\{X_i \in \mathcal{X}^*\}$ are independent and identically distributed (i.i.d.) Bernoulli random variables. Hence, $\hat{\ell}_{\text{CMC}}$ is an unbiased estimator of $\ell$, with known variance

$$\text{Var} \left( \hat{\ell}_{\text{CMC}} \right) = \frac{\ell(1 - \ell)}{N}. \tag{2}$$

We now consider the accuracy, i.e. efficiency, of the CMC estimator under this rare-event setting. A standard way of assessing Monte Carlo estimators is through their relative error (RE), which can be used for creating confidence intervals (Rubinstein & Kroese, 2016). The RE of $\hat{\ell}_{\text{CMC}}$ is

$$\text{RE} \left( \hat{\ell}_{\text{CMC}} \right) = \frac{\sqrt{\text{Var} \left( \hat{\ell}_{\text{CMC}} \right)}}{\hat{\ell}_{\text{CMC}}} \approx \frac{\sqrt{\ell(1 - \ell)/N}}{\ell}. \tag{2}$$

Under our rare-event setting, we have that $\ell \ll 1$, so

$$\text{RE} \left( \hat{\ell}_{\text{CMC}} \right) \approx 1/\sqrt{N\ell}, \tag{3}$$

which imposes a serious problem. To see this, consider the rare-event probability $\ell \approx 10^{-12}$, and suppose that we are interested in a very modest 10% RE. It is not very hard to verify from (3) that the required number of experiments $N$ is about $10^{14}$.

We return to the thesis setup now. If the likelihood value is estimated to be exactly zero for different parameter values, then one has no indication of which parameter values are closer to the true ones, hence there is no indication of the direction to go to. We define the method to be failing once it estimates a likelihood of exactly zero, as it does not inform one any more than those exact parameters being not the true ones. CMC draws an initial degradation and the noise variables and subsequently computes the degradation over time. If due to the draws the degradation is negative, the likelihood will be zero, as degradation over positive time cannot be negative by the model assumptions. In the remaining part of this section, we explore the probability of CMC failing, i.e. the probability of the likelihood being zero. We find that for the number of simulations fixed and the number of components or the number of time periods going to infinity, the probability of CMC failing will go to one. Moreover, generally, for higher $\sigma_Z$ the probability of CMC failing will converge quicker to one.
Recall that the computation of the likelihood for CMC was
\[
\hat{L} (\theta | y_1, \ldots, y_m) = \frac{1}{N} \sum_{j=1}^{N} \prod_{i=1}^{m} \prod_{t=1}^{n} \left( \frac{\Delta x_{i,t}^j (\alpha t (l - 1)^{n-1} - \alpha t (l-1)^n - 1)}{\Gamma (\alpha t (l-1)^n - \alpha t (l-1)^n - 1)} \right) \text{exp} \left( \frac{-\Delta x_{i,t}^j}{\alpha t} \right) \mathbb{I} \{ \Delta x_{i,t}^j \geq 0 \}. 
\]

Hence, if for all \( j \in \{1, \ldots, N \} \) there exists a \( \Delta x_{i,t}^j \leq 0 \), then the estimated likelihood value will be zero. Thus, the probability of CMC failing (hence, the likelihood equaling zero) is the following
\[
\mathbb{P} (\forall_j \exists_{i,t} \text{ s.t. } \Delta x_{i,t}^j \leq 0) = \prod_{j=1}^{N} \mathbb{P} (\exists_{i,t} \text{ s.t. } \Delta x_{i,t}^j \leq 0) = \prod_{j=1}^{N} \left( 1 - \mathbb{P} (\forall_{i,t} \text{ s.t. } \Delta x_{i,t}^j \geq 0) \right) = \left( 1 - \mathbb{P} (\forall_{i,t} \text{ s.t. } \Delta x_{i,t}^j \geq 0) \right)^N.
\]

We know that the data \( y_{i,t} \) is observed for \( i \in \{1, \ldots, m\} \) and \( l \in \{1, \ldots, n\} \). For time zero, i.e. \( l = 0 \), \( y_{i,0} = A_i \), \( x_{i,0} = 0 \) and \( z_{i,0} = 0 \). Hence, we have two cases:

- \( l = 0 \):
  \( \Delta x_{i,t}^j = \Delta y_{i,l} - \Delta z_{i,t}^j = y_{i,1} - A_i - z_{i,1} \sim \mathcal{N}(y_{i,1} - \mu_A, \sigma_A^2 + \sigma_Z^2) \)

- \( l \neq 0 \):
  \( \Delta x_{i,t}^j = \Delta y_{i,l} - \Delta z_{i,t}^j \sim \mathcal{N}(\Delta y_{i,l}, 2\sigma_Z^2) \)

In both aforementioned cases \( \Delta x_{i,t}^j \) is normally distributed.

First, we explore how this probability behaves when the number of time periods, \( n \), goes to infinity. In this case, we apply the following derivation.
\[
\mathbb{P} (\forall_t \text{ s.t. } \Delta x_{i,t}^1 \geq 0) = \mathbb{P} (\Delta x_{i,1}^1 \geq 0) \mathbb{P} (\Delta x_{i,2}^1 \geq 0 | \Delta x_{i,1}^1 \geq 0) \ldots \\
\ldots \mathbb{P} (\Delta x_{i,n}^1 \geq 0 | \Delta x_{i,1}^1 \geq 0, \ldots, \Delta x_{i,n-1}^1 \geq 0).
\]

Now choose
\[
a = \max_{i,j} \{ \mathbb{P} (\Delta x_{i,t}^1 \geq 0 | \Delta x_{i,1}^1 \geq 0, \ldots, \Delta x_{i,j-1}^1 \geq 0) \},
\]
where for \( j = 1 \) we define
\[
\mathbb{P} (\Delta x_{i,1}^1 \geq 0 | \Delta x_{i,1}^1 \geq 0, \ldots, \Delta x_{i,0}^1 \geq 0) = \mathbb{P} (\Delta x_{i,1}^1 \geq 0).
\]

Because \( \Delta x_{i,t}^1 \) is normally distributed, in other words, it has a continuous distribution, we know that \( 0 \leq a < 1 \). Therefore, by choosing \( a \) in the above manner, as \( n \) goes to infinity, we can subsequently bound the probability in the following manner:
\[
\mathbb{P} (\forall_t \text{ s.t. } \Delta x_{i,t}^1 \geq 0) = \mathbb{P} (\Delta x_{i,1}^1 \geq 0) \mathbb{P} (\Delta x_{i,2}^1 \geq 0 | \Delta x_{i,1}^1 \geq 0) \ldots \\
\ldots \mathbb{P} (\Delta x_{i,n}^1 \geq 0 | \Delta x_{i,1}^1 \geq 0, \ldots, \Delta x_{i,n-1}^1 \geq 0) \leq a^n \xrightarrow{n \rightarrow \infty} 0
\]
Therefore, as \( n \) goes to infinity and for a fixed sample size \( N \), the probability of CMC failure will approach one, namely, it holds that

\[
P(\forall j \exists i, l \text{ s.t. } \Delta x_{i,l}^j \leq 0) = \left( 1 - \prod_{i=1}^{m} P(\forall l \text{ s.t. } \Delta x_{i,l}^1 \geq 0) \right)^N
\]

\[
> \left( 1 - \prod_{i=1}^{m} a^n \right)^N = \left( 1 - a^{mn} \right)^N \stackrel{n \to \infty}{\to} \left( 1 - 0^m \right)^N = 1
\]

Now, for the case that \( n \) does not go to infinity, one can show that the probability of CMC failing will still go to 1 as \( \sigma_Z \) and \( m \) increase. To see this, take the index of the largest degradation observation at time 1, i.e.

\[
k = \text{argmax}_i \{ y_{k,1} \}.
\]

Then, we can bound the probability \( \prod_{i=1}^{m} P(\forall l \text{ s.t. } \Delta x_{i,l}^1 \geq 0) \) in the following manner

\[
\prod_{i=1}^{m} P(\forall l \Delta x_{i,l}^1 \geq 0) < \prod_{i=1}^{m} P(\Delta x_{i,1}^1 \geq 0) < P(\Delta x_{k,1}^1 \geq 0)^m
\]

\[
= \left( 1 - \Phi \left( \frac{-\left( y_{k,1} - \mu_A \right)}{\sqrt{\sigma_A^2 + \sigma_Z^2}} \right) \right)^m = \Phi \left( \frac{y_{k,1} - \mu_A}{\sqrt{\sigma_A^2 + \sigma_Z^2}} \right)^m.
\]

For \( m \) going to infinity, one can immediately see that \( \Phi \left( \frac{y_{k,1} - \mu_A}{\sqrt{\sigma_A^2 + \sigma_Z^2}} \right)^m \to 0 \) and hence the probability of CMC failure goes to 1. However, one does not know how quick this convergence will happen. For a positive \( y_{k,1} - \mu_A \), the larger \( \sigma_Z \), the quicker the convergence. When \( \sigma_Z \) goes to infinity, \( \Phi \left( \frac{y_{k,1} - \mu_A}{\sqrt{\sigma_A^2 + \sigma_Z^2}} \right)^m \to 0.5 \). Thus, for a fixed \( N \)

\[
P(\forall j \exists i, l \text{ s.t. } \Delta x_{i,l}^j \leq 0) = \left( 1 - \prod_{i=1}^{m} P(\forall l \text{ s.t. } \Delta x_{i,l}^1 \geq 0) \right)^N
\]

\[
> \left( 1 - \Phi \left( \frac{y_{k,1} - \mu_A}{\sqrt{\sigma_A^2 + \sigma_Z^2}} \right)^m \right)^N \stackrel{\sigma_Z \to \infty}{\to} \left( 1 - 0.5^m \right)^N \stackrel{m \to \infty}{\to} \left( 1 - 0 \right)^N = 1.
\]

Hence, for both \( m \) and \( n \) going to infinity, the probability of CMC failing goes to 1. Moreover, for positive \( y_{k,1} - \mu_A \) the lower bound of the failing probability becomes higher for a larger \( \sigma_Z \). Note that because \( y_{k,1} \) is the largest observation, it is very likely that \( y_{k,1} - \mu_A \) will be positive for \( m \) even equal to a small integer. For positive \( y_{k,1} - \mu_A \), the larger \( \sigma_Z \), the quicker the failing probability will converge to 1 with \( m \).

To conclude, for \( n \) going to infinity the probability of the likelihood value equalling 0 will go to 1, as well as for \( m \) going to infinity. Moreover, it is very likely that the larger \( \sigma_Z \), the quicker this probability will converge to 1 with \( m \) increasing.
We can now see the connection between the CMC failure and the rare-event setting. That is, as shown above, the CMC failure happens due to the rare-event trap, as for a large enough data set the event \( \{ \forall_{i,l} \Delta x_{i,l}^j \geq 0 \} \) becomes a rare-event. In order to avoid the rare-event problem, we propose to use a different Monte Carlo algorithm, namely the SMC. The SMC method is discussed in Section 3.3.

### 3.3 Sequential Monte Carlo

Crude Monte Carlo executes its \( N \) runs separately, where in each run, for each time step, a new \( z_{i,l} \) is drawn. However, this approach may draw a very unlikely value given the degradation data \( (y_1, \ldots, y_m) \), which will be used throughout that entire simulation even though it may not accurately reflect the data. Sequential Monte Carlo addresses this problem, the new particles \( x_{i,l}^{j+1} \) are namely based on the likeliness of the old particles \( x_{i,l}^j \) for \( j \in \{1, \ldots, N\} \).

All \( N \) runs are executed simultaneously. The idea of SMC is to at each time step \( l \) compute weights \( w_l^j \) for all \( x_{i,l}^j + a_l^j \) based on the likeliness of the values \( z_l^j = y_l - x_{i,l}^j - a_l^j \). In the next time step, a resampling of the indexes occurs first, where index \( j \) becomes \( k(j) \) based on the weights \( w_l^j \). The new values \( x_{i,l+1}^{j+1} + a_l^{j+1} \) subsequently equal \( x_{i,l}^{k(j)} + a_l^{k(j)} + \Delta x_{i,l+1}^j \), where \( \Delta x_{i,l+1}^j \) is number drawn from the gamma distribution described in the model in (1). Hence, the algorithm continues with values that are more likely to occur (Chopin & Papaspiliopoulos, 2020).

To make the algorithm more efficient, the resampling step is not performed at each time step, though only at certain time steps. This is called adaptive resampling. Adaptive resampling only resamples once the Effective Sample Size (ESS) is below a certain threshold. The ESS is defined as:

\[
ESS_l(w_l^j) = \frac{\left( \sum_{j=1}^{N} w_l^j \right)^2}{\sum_{j=1}^{N} (w_l^j)^2} = \frac{1}{\sum_{j=1}^{N} (W_l^j)^2}
\]

where \( W_l^j = \frac{w_l^j}{\sum_{j=1}^{N} w_l^j} \).

\( W_l^j \) are the standardized weights, hence they sum up to 1 over all \( j \in \{1, \ldots, N\} \). The interpretation of the ESS per time period is as follows, when \( k \) weights \( w_l^j \) are equal to 1 and \( N - k \) weights are equal to \( 0 \), then the ESS equals \( k \). Hence, if effectively there are \( k \) particles that completely count and all others do not, than the ESS is \( k \). A common choice of the threshold is \( ESS_{\text{min}} = \frac{N}{2} \), if \( ESS < ESS_{\text{min}} \), then the algorithm resamples (Chopin & Papaspiliopoulos, 2020).

A generic algorithm for SMC with adaptive resampling can be observed in Algorithm 3 (Chopin & Papaspiliopoulos, 2020). The choice for the distributions \( M_l \) and \( G_l \) depends on the method of Sequential Monte Carlo one applies. We implement the most basic and most common method, namely the Bootstrap filter.
Algorithm 3 Sequential Monte Carlo, Generic

for all runs \( j \) do
  for all components \( i \) do
    Draw \( m \) from a distribution \( M_0(dx_{i,0}) \)
    Initialize \( X_i^j(0) = m \)
  end
  Initialize \( w_i^j = G_0(X_i^j) \)
end

for all runs \( j \) do
  Initialize \( W_i^j = \frac{w_i^j}{\sum_{t=1}^{N} w_i^t} \)
end

for all time periods \( l \) do
  if \( \text{ESS}_{l-1}(W_{l-1}) = \frac{1}{\sum_{j=1}^{N} (w_{l-1}^j)^2} < N/2 = \text{ESS}_{\text{min}} \) then
    Draw a vector \( k \) with \( N \) elements drawn from \( \{1, \ldots, N\} \) where number \( j \) has probability \( W_i^{j-1} \) of being drawn
    \( \hat{w}_{l-1} = 1 \)
  else
    \( k = [1, \ldots, N] \)
    \( \hat{w}_{l-1} = w_{l-1} \)
  end

for all runs \( j \) do
  for all components \( i \) do
    Draw \( m \) from distribution \( M_t(X_i^{k(j)}(t(l-1)), dx_{i,t}) \)
    \( X_i^j(t(l)) = m \)
  end
  \( w_i^j = \hat{w}_{l-1} G_t(X^{k(j)}(t(l-1)), X^j(t(l))) \)
end

\( W_l = \frac{\sum_{i=1}^{N} w_i^l}{\sum_{h=1}^{N} w_h^l} \)

end
3.3.1 The Bootstrap Filter

As mentioned above, the Bootstrap filter is one that is quite commonly used in Sequential Monte Carlo literature. The type of filter defines the choice of $M_t$ and $G_t$ in Algorithm 3. For Bootstrap and our model defined in (1), the choice is as follows:

\[
M_0(dx_0) = N(\mu_A, \sigma_A^2) \\
G_0(X^0_t) = 1 \\
M_t(X_i^{k(j)}(t(l - 1)), dx_{i,t}) = \text{Gamma}(at(l)^n - at(l - 1)^n, X_i^{k(j)}(t(l - 1)), \beta) \\
G_t(X^{k(j)}(t(l - 1)), X^j(t(l))) = \prod_{i=1}^{m} \frac{1}{\sigma Z \sqrt{2\pi}} \exp\left(-\frac{(y_{i,t} - X_i^j(t(l)))^2}{2\sigma_Z^2}\right)
\]

The initial value $X_i^j(0)$ represents the initial degradation, $A_i$, therefore we draw from a normal distribution with the parameters $\mu_A$ and $\sigma_A^2$. Note that this notation is different from the model defined in (1), $A_i$ is here included in $X_i^j(t(l))$ for simplicity of further notation. The initial weights are all equal to 1, hence initially all particles are equally counted, i.e. they have an equal chance of being picked during resampling. The $X_i^j(t(l))$ in a new time step are a sum of $X_i^{k(j)}(t(l - 1))$, i.e. the previously occurred degradation, and the degradation that occurred in the last time step which is a gamma distributed random variable. Lastly, the weights $\tilde{w}_{i,t-1}$ are multiplied by the probability of $X^j(t(l))$ given the degradation data $y_t$ to compute the new weights $\tilde{w}_{i,t}$. This probability is the product of the probability density function of the normal distribution $N(0, \sigma_Z^2)$ for all $i \in \{1, \ldots, m\}$. Each $(y_{i,t} - X_i^j(t(l))) = z_{i,t}$ is namely normally distributed with mean 0 and variance $\sigma_Z^2$, hence the probability of the particles $X_i^j(t(l))$ being equal to their true values given the degradation data equals the probability of $z_{i,t}$ being equal to their true value, which is the product defined above for $G_t$.

With these choices of $M_t$ and $G_t$, the algorithm for Bootstrap is defined, which can be observed in Algorithm 4.

Bootstrap is intuitively quite easy to understand, which might also be a reason why it is a commonly used filter. Intuitively namely, the particles are a Markov Chain $\{X^j\}$ where if resampling occurs, one draws a chain from all runs $j \in \{1, \ldots, N\}$ based on their probabilities of being given the observed degradation data. Subsequently, one continues with this drawn chain in run $j$ and extends it with a new value $X_i^{j+1}$ by drawing an increase in degradation from $X_i^j$ from the gamma distribution. Hence, one keeps continuing with the most likely chains $\{X^j\}$ given the observed degradation data $(y_1, \ldots, y_m)$.

To estimate the likelihood value through the Bootstrap algorithm, the output $w_i^j$ for all $l \in \{1, \ldots, n\}$ and $j \in \{1, \ldots, N\}$ is used. The estimator of the likelihood namely is

\[
\hat{L}(\theta | y_1, \ldots, y_m) = \prod_{l=1}^{n} \hat{L}_l(\theta | y_1, \ldots, y_m),
\]

i.e. the product of the estimated likelihood values for each time step. Furthermore, one can estimate the likelihood values for each time step as follows (Chopin & Papaspiliopoulos, 2020)
Algorithm 4 Bootstrap Filter

for all runs \( j \) do
  for all components \( i \) do
    Draw \( m \) from \( N(\mu_A, \sigma_A^2) \)
    Initialize \( X_j^i(0) = m \)
  end
  Initialize \( w_j^0 = 1 \)
  Initialize \( W_j^0 = \frac{1}{N} \)
end

for all time periods \( l \) do
  if \( ESS_{l-1}(W_{l-1}) = \frac{1}{\sum_{j=1}^{N} (W_{l-1}^j)^2} < N/2 = ESS_{min} \) then
    Draw a vector \( k \) with \( N \) elements drawn from \( \{1, \ldots, N\} \) where number \( j \) has probability \( W_{l-1}^j \) of being drawn
    \( \hat{w}_{l-1} = 1 \)
  else
    \( k = [1, \ldots, N] \)
    \( \hat{w}_{l-1} = w_{l-1} \)
  end
  \( w_l = \hat{w}_{l-1} \)
  for all runs \( j \) do
    for all components \( i \) do
      Draw \( \Delta x \) from \( \text{Gamma}(\alpha t(l)^{\eta} - \alpha t(l-1)^{\eta}, \beta) \)
      \( X_j^i(t(l)) = \Delta x + X_j^i(t(l-1)) \)
      \( w_l^j = w_l^j \frac{1}{\sigma_Z \sqrt{2\pi}} \exp\left(-\frac{(y_{l,1}-X_j^i(t(l)))^2}{2\sigma_Z^2}\right) \)
    end
  end
  \( W_l = \sum_{h=1}^{N} w_l^h \)
end

\[
\hat{L}_l(\theta|y_1, \ldots, y_m) = \begin{cases} 
\frac{1}{N} \sum_{j=1}^{N} w_l^j & \text{if resampling occurred at time } t(l) \\
\frac{\sum_{j=1}^{N} w_l^j}{\sum_{j=1}^{N} w_{l-1}^j} & \text{otherwise.}
\end{cases}
\]

Similar to the diversification applied in the CMC method, this likelihood estimation is also diversified in order to avoid numerical instability. This implies that the logarithm is taken and the same trick in the exponential is applied, as shown below. Let \( \log\left(\hat{L}(\theta|y_1, \ldots, y_m)\right) = \log \hat{L} \); then it holds that

\[
\log \hat{L} = \sum_{l=1}^{n} \log \left(\hat{L}_l\right) \tag{4}
\]
where

\[ c_l = \max\{\log(w_l^1), \ldots, \log(w_l^N)\} \] and

\[
\log(w_l^j) = \log\left(\frac{1}{w_l^j} \prod_{i=1}^{m} \frac{1}{\sigma_Z \sqrt{2\pi}} \exp\left(-\frac{(y_{i,t} - X_i^j(t))^2}{2\sigma_Z^2}\right)\right)
\]
\[
= \log(\hat{w}_{l,-1}^j) + \sum_{i=1}^{m} \log\left(\frac{1}{\sigma_Z \sqrt{2\pi}} \exp\left(-\frac{(y_{i,t} - X_i^j(t))^2}{2\sigma_Z^2}\right)\right)
\]
\[
= \log(\hat{w}_{l,-1}^j) + \sum_{i=1}^{m} -\log(\sigma_Z \sqrt{2\pi}) - \frac{(y_{i,t} - X_i^j(t))^2}{2\sigma_Z^2}.
\]

\(W_l^j\) and \(ESS_l(W_i)\) are terms that depend on the values \(w_l^j\) over all \(j\), hence these terms are transformed to the expressions shown next.

\[
\log(W_l^j) = \log\left(\frac{w_l^j}{\sum_{h=1}^{N} w_l^h}\right) = \log(w_l^j) - \log\left(\sum_{h=1}^{N} w_l^h\right)
\]
\[
= \log(w_l^j) - \log\left(\sum_{h=1}^{N} \exp(\log(w_l^h))\right) = \log(w_l^j) - c - \log\left(\sum_{h=1}^{N} \exp(\log(w_l^h) - c)\right)
\]

where

\[ c = \max\{\log(w_l^1), \ldots, \log(w_l^N)\}, \]

and in addition,

\[
\log(ESS_l(W_i)) = \log\left(\frac{1}{\sum_{j=1}^{N} (W_l^j)^2}\right) = \log\left(\frac{\sum_{j=1}^{N} w_l^j}{\sum_{j=1}^{N} (w_l^j)^2}\right)
\]
\[
= 2 \log\left(\sum_{j=1}^{N} w_l^j\right) - \log\left(\sum_{j=1}^{N} (w_l^j)^2\right)
\]
\[
= 2c + 2 \log\left(\sum_{j=1}^{N} \exp(\log(w_l^j) - c)\right) - \log\left(\sum_{j=1}^{N} \exp(\log((w_l^j)^2))\right)
\]
\[
= 2c + 2 \log\left(\sum_{j=1}^{N} \exp(\log(w_l^j) - c)\right) - 2c - \log\left(\sum_{j=1}^{N} \exp(2 \log(w_l^j) - 2c)\right)
\]
\[
= 2 \log\left(\sum_{j=1}^{N} \exp(\log(w_l^j) - c)\right) - \log\left(\sum_{j=1}^{N} \exp(2 \log(w_l^j) - 2c)\right).
\]

The implemented code, hence including these diversifications, is described in pseudocode in Algorithm 5.

M.B.C. Buist 24
Algorithm 5 Diversified Bootstrap Filter

for all runs \( j \) do
  for all components \( i \) do
    Draw \( m \) from \( N(\mu_A, \sigma_A^2) \)
    Initialize \( X^j_i(0) = m \)
  end
  Initialize \( \log(w^j_0) = 0 \)
  Initialize \( \log(W^j_0) = -\log(N) \)
end

for time steps \( l \) from 1 to \( n \) do
  \( c = \max\{\log(w^1_l), \ldots, \log(w^N_l)\} \)
  \( \log\left(\text{ESS}_{l-1}(W_{l-1})\right) = 2\log\left(\sum_{j=1}^{N} \exp(\log(w^j_{l-1}) - c)\right) - \log\left(\sum_{j=1}^{N} \exp(2\log(w^j_{l-1}) - 2c)\right) \)
  if \( \exp(\log(\text{ESS}_{l-1}(W_{l-1}))) < \text{ESS}_{\text{min}} \) then
    Draw a vector \( k \) with \( N \) elements drawn from \( \{1, \ldots, N\} \) where number \( j \) has probability \( \exp(\log(W^j_{l-1})) \) of being drawn
    \( \log(\hat{w}_{l-1}) = 0 \)
  else
    \( k = [1, \ldots, N] \)
    \( \log(\hat{w}_{l-1}) = \log(w_{l-1}) \)
  end
  for all runs \( j \) do
    Draw vector \( \Delta x \) of length \( m \), each element drawn from \( \text{Gamma}(\alpha t^n - \alpha (t - 1)^n, \beta) \)
    \( X^j(t(l)) = \Delta x + X^{k(j)}(t(l - 1)) \)
    \( \log(w^j_l) = \log(\hat{w}^j_{l-1}) + \sum_{i=1}^{m} -\log(\sigma Z \sqrt{2\pi}) - \frac{(y_{i,t} - X^j_i(t))^2}{2\sigma_Z^2} \)
  end
  \( c = \max\{\log(w^1_l), \ldots, \log(w^N_l)\} \)
  \( \log(W_l) = \log(w_l) - c - \log\left(\sum_{j=1}^{N} \exp(\log(w^j_l) - c)\right) \)
end

To obtain the estimated log likelihood, the outputs \( \log(w_l) \) for all time steps \( l \) are inputted in the formula for the log likelihood in the equations (4). Again, this estimator can be used for maximum likelihood estimation of the parameters. Moreover, this log likelihood estimation is executed \( N_{\text{sim}} \) number of times in order to compare the performance and thus the variance of the estimator with Crude Monte Carlo.
4 Numerical Study

This section involves the numerical study of the results, which mainly focuses on comparing the estimated likelihood values of the two different methods with one another. The comparison includes a comparison of the estimators variance. We expect the variance of CMC to be significantly larger than the variance of SMC. If so, CMC will be a less reliant method of estimating the likelihood: the estimator is less reliable and thus the maximal likelihood over different parameter settings may be less likely to approach the true parameter setting. For the comparison, we consider the following data settings.

1. Synthetic Data
   In the first case study, we examine the synthetic data from (Hazra et al., 2020), which was introduced in Section 1. Using the model explained in (1) and choosing the parameter set \( \theta = (\alpha, \eta, \beta, \mu_A, \sigma_A, \sigma_Z) = (2, 2.5, 0.01, 0.5, 0.01, 0.01) \), time periods \( \{2, 4, 6\} \) and number of components \( m = 10 \), synthetic data is created. We start with a synthetic data study as to know the real parameter values can be very convenient; the estimated likelihood value for the true parameter values can be computed namely. Moreover, one can see what the estimated likelihood value does the further it goes away from the true parameter values, or in general for other parameter values than the true ones. Even more, one can leave certain parameter values to be known, thus set them on their true value, and hence vary only a few parameter values. We will compare the two methods for three different number of unknowns, namely one unknown, two unknowns and six unknowns, which is based on (Hazra et al., 2020), which attempts to estimate parameters under the same circumstances. The prior distributions used for the unknown parameters will be applied as well in this research. For this study, we define by the prior distribution the distribution from which a number of parameter values for which the likelihood is estimated are drawn.

2. Real Data
   Our second case study is concerned with the application of the degradation model to real data. The synthetic data is based on the model 1, however, if real data does not comply with the model in (1), then results will likely be significantly worse for this case. Hence, to test that the methods are effective for real data as well, this case is considered. The data is extracted from (Chow & Shao, 1991), which contains drug shelf-life data. The longer drugs are shelved, the more their effectiveness deteriorates. Concretely, they consider drug potency in percentage of claimed potency from drugs over the time they are shelved. Their data consists of 24 drug batches, measured at 0, 1, 2 and 3 years after the drugs were produced, which can be observed in the Appendix in Table 11.

   Due to the model definition in (1), the data is transformed to fit the model input, hence the data is transformed to represent the degradation that has occurred. Therefore, a starting value needs to be set. The data starts with a percentage higher than 100 for all drug batches, hence most drugs are produced with more substance than needed for effectiveness. We assume the start potency to be 105 percent of the claimed potency. The only parameter that this assumption influences is \( \mu_A \), which will be higher or lower
A Comparison of CMC Against SMC

depending on which starting value is chosen. The observed data \( \{y_1, \ldots, y_{10}\} \) used to apply CMC and SMC to is the following:

\[
y_i(t) = 105 - \text{drug potency of batch } i \text{ at time } t
\]

where time 0 is left out due to the model assumptions.

In the upcoming section the primary results of the numerical analysis are the following:

1. Synthetic Data
   - The Bootstrap estimation of the likelihood clearly outperforms the Crude Monte Carlo estimation. It shows significantly smaller relative variance and thus is essentially more reliable in terms of the correct parameter settings having the largest likelihood value. As such, Bootstrap is truly more suitable for maximum likelihood estimation of the parameters than Crude Monte Carlo.
   - The relative errors associated with the Crude Monte Carlo method are in fact so substantial, that the likelihood values for all different parameter settings, within 95% certainty, could as well be estimated to be zero.
   - The Bootstrap method demonstrates increasing likelihoods towards the true parameter setting accompanied by distinct confidence intervals for parameter settings sufficiently apart from each other.

2. Real Data
   - A data analysis on the real data conveys the impression that the data could have been from the same distribution as the model assumptions. Random density plots have similar shape to the real data density plot. It does reject the null hypothesis of the data being either from a normal or gamma distribution, though not by extremely low p-values. Hence a gamma plus a normal distribution for the real data seems plausible.
   - Bootstrap outperforms CMC in terms of reliability. The Bootstrap method has significantly lower relative variance, making the estimates more accurate and the method more suitable for maximum likelihood estimation of the parameters than Crude Monte Carlo.
   - Both methods have a high relative error causing confidence intervals to overlap with each other and with zero, even though Bootstrap’s relative error is smaller than the one of Crude Monte Carlo.
   - The likelihoods are extremely small, presumably partly because of the parameter search and sensitivity of the likelihood estimates.

Note that the results obtained in the following section are regarding the model used in (1) and regarding the data, either real or synthetic, employed in this research. Another model or other data may result in different conclusions. The model applied in the thesis is however a common one for degradation processes as explained in the Introduction and Literature
1. Moreover, the synthetic data should not influence the conclusions of the study, as other parameter values should give the same comparison between the likelihoods. The real data could have an influence however, the conclusions for other real degradation data could be different, this however relies on the correctness of the model definition just discussed, which will be evaluated in a brief data analysis in this section. The code generating all results is included Appendices B, C, D and E.

4.1 Synthetic Data

For each component \( i \in \{1, \ldots, 10\} \) we draw an \( a_i \) from \( N(\mu_A, \sigma_A^2) \) and for each time period \( l \in \{1, 2, 3\} \) as well we draw a \( \Delta x_{i,l} \) from \( \text{Gamma}(\alpha t(l)^{\eta} - \alpha t(l-1)^{\eta}, \beta) \), where \( t = [0, 2, 4, 6] \), and a \( z_{i,l} \) from \( N(0, \sigma_Z^2) \). We set \( \Delta X_i(t(l)) = \Delta x_{i,l} \), \( X_i(0) = a_i \), \( Z_i(t(l)) = z_{i,l} \) and \( Z_i(0) = 0 \). From this data, the observed data \( \{y_1, \ldots, y_{10}\} \) can be composed. We save the measured times vector \([2, 4, 6]\) and the observed data for those times \( \{y_1, \ldots, y_{10}\} \). As such, we will consider three different cases, all with another number of unknown parameters. The cases that will be considered are:

1. 1 Unknown Parameter
   The first considered case will be that of leaving only one parameter unknown, which will be \( \alpha \). Equivalent to (Hazra et al., 2020), the prior distribution applied in this research is:
   - \( U[0, 10] \) for \( \alpha \).

2. 2 Unknown Parameters
   Second, the case with two unknown parameters will be considered, leaving \( \eta \) and \( \mu_A \) unknown and all other parameters equal to their true value. Again, the prior distributions for the unknown parameters are chosen to be the ones used in (Hazra et al., 2020). This means that the original distributions applied are:
   - \( U[0, 5] \) for \( \eta \),
   - \( U[0, 1] \) for \( \mu_A \).

3. 6 Unknown Parameters
   Lastly, the case with all parameters left unknown. The original distributions for the parameters are:
   - \( U[0, 5] \) for \( \alpha \) and \( \eta \),
   - \( U[0, 1] \) for \( \mu_A, \beta, \sigma_A \) and \( \sigma_Z \),
   which are equivalent to (Hazra et al., 2020).

The first case study considers the synthetic data as described above. For all different number of unknowns we visualize the mean estimated likelihood value with the corresponding 95% confidence interval (CI). As the highest values for Bootstrap are significantly larger than of those for CMC in most instances, both the original plot and a zoomed in plot is shown, in
A Comparison of CMC Against SMC

order to visualize the CMC values better. The y-axis of the graphs ranges over the likelihood value, and the x-axis over the different parameter settings that are used to estimate the likelihood. These different settings include as first setting the true parameter values, and the further along the axis the further away the parameters are from the true values. They are drifted by a ratio $r_c$ of the total interval of the distribution drawn from. Thus, suppose $\alpha$ is originally drawn from $U[w, y]$ and the true value for $\alpha$ is $\alpha_{\text{true}}$, then for ratio of change $r_c$ the value used for $\alpha$ is $\alpha_{\text{true}} + r_c \cdot (y - w) \cdot \text{random}(-1, 1)$, where $\text{random}(-1, 1)$ is a random integer of the set $\{1, -1\}$. Hence the drift is randomly added or subtracted to the true value. If the value exceeds the bounds $w$ or $y$ then the drift is reversed, i.e. multiplied by $-1$.

All results are obtained with the number of particles $N_{\text{part}} = 10000$ and the number of times the likelihood is estimated $N_{\text{sim}} = 1000$.

4.1.1 1 Unknown Parameter

The results for the two methods CMC and Bootstrap for leaving only $\alpha$ unknown can be observed in Figure 2. From the figure one can observe that the likelihood estimates of Bootstrap are generally much higher than those of CMC for parameters close to the true values. From a certain change in ratio, the likelihood decays very rapid. As the Bootstrap values are much higher for parameters with a maximum change less than 0.02 of the total interval than for parameters further away, and have no overlapping CI’s, an appropriate maximum likelihood estimation will very likely lead to parameters with no more deviation than a ratio of 0.02 from the true values. The mean likelihood of the true parameter setting deviates approximately 21 times its 95% half-width from 0, where likelihood values from parameters with a deviation of 0.05 of the true parameter setting lie. For the first three parameter settings the 95% confidence intervals are overlapping however, thus the estimation is still subjective to some uncertainty. It might not predict the true parameter value but a value with a ratio change of 0.01 for instance. Given the size of the true parameter, which is 2, a 0.02 deviation would be very reasonable. In fact, if the likelihood goes to zero for considerably smaller deviation, this might complicate maximum likelihood estimation.

Examining the zoomed in graph, one can also see confidence intervals for CMC for the first four parameter settings. First note that some 95% confidence intervals show negative values, however it is common knowledge that likelihood values cannot be negative, the reason for these intervals is simply due to higher valued outliers which increase the variance. That said, the intervals for the true parameter setting as well as for close to the true values are all overlapping with the results for parameter settings further away. Thus, executing a maximum likelihood estimation using CMC to estimate the likelihood values is much less reliable than Bootstrap. The variance for the likelihood estimates is relatively substantially higher in comparison, hence the estimate using the true parameter value, or a value close by, could be lower than the estimate using a parameter value far away, for instance with a 0.5 ratio change. In such a case, maximum likelihood estimation would return the parameter far from the true value as estimated parameter. The method is very uncertain due to the high relative variance and hence relatively large confidence intervals. Moreover, surprisingly,
A Comparison of CMC Against SMC

Figure 2: Likelihood Comparison for 1 Unknown Parameter

(a) Including the mean estimated likelihood value and confidence interval for different parameter settings of the 1 unknown

(b) Including the mean estimated likelihood value and confidence interval for different parameter settings of the 1 unknown with a zoomed in y-axis

Figure 2: Likelihood Comparison for 1 Unknown Parameter
the estimated mean for the parameter with a 0.001 and 0.01 ratio change is substantially higher than for the true parameter value. This is likely due to a few outliers that rocket these values, as CMC has shown to have more significant outliers.

Hence, we have observed that in this case CMC is immensely less reliable than Bootstrap due to the high relative error (RE). To take a closer look, the relative errors are noted beneath in Table 1. As some values are extremely small, the relative error could not always be computed, as MatLab considered them to be zero due to their size. Due to machine error, the relative error of small likelihood estimates can also be inaccurate. Hence, we only examine the relative errors in which the mean and standard error were deemed large enough to evaluate the relative error correctly. For this case therefore, we leave out the parameter setting with a 0.05, 0.1 and 0.5 change.

<table>
<thead>
<tr>
<th>Method</th>
<th>$r_e$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>0.000</td>
</tr>
<tr>
<td>CMC</td>
<td>0.9708</td>
</tr>
<tr>
<td>Bootstrap</td>
<td>0.0240</td>
</tr>
</tbody>
</table>

Table 1: The relative error for both estimation methods over the different parameter inputs for one unknown.

From this table one can clearly see that the relative errors of Bootstrap are incredibly smaller than the relative errors of CMC. In fact, the values for the CMC estimation are extremely high and hence unfavorable. The smaller relative errors of Bootstrap indicate less deviation within the estimation of the likelihood, and hence result in more accurate maximum likelihood estimation of the parameters. It is clear from this table that Bootstrap is by far outperforming CMC.

Next to the relative error which indicates reliability of the method, we examine the sensitivity of the likelihood function around the true parameter value for both methods. The sensitivity is crucial for maximum likelihood estimation. Namely, if the likelihood goes to zero very quickly around the true parameters, it will be harder to find the direction to the true parameters, while if it remains high in a large area around the true parameters, the estimation will likely not be very accurate. Table 2 shows the relative change in likelihood compared to the estimated likelihood for the true parameters. As what could have been observed in the figure, the likelihood of CMC is very elastic for only small random parameter changes. However, the value increases, and the likelihood is still relatively high at a 0.02 parameter change. Looking at the confidence intervals though, the relative change between the likelihoods could be equal any value, which is due to the huge variance of the likelihood making them still very probable to equal zero. This presents the critical unreliability of CMC once again. This is different for the random changes in Bootstrap, which is significantly less elastic around the true parameter values and decreases a bit quicker. As the variance is significantly smaller, the confidence intervals are a lot smaller as well. The negative change in parameters seems to decrease the likelihood quicker for the CMC method, however, due to the confidence intervals this is quite uncertain. For the Bootstrap method the negative
change seems to decrease the likelihood quicker as well, with more certainty.

<table>
<thead>
<tr>
<th>Method</th>
<th>$r_c$</th>
<th>0.000</th>
<th>0.001</th>
<th>0.010</th>
<th>0.020</th>
<th>0.050</th>
<th>0.100</th>
<th>0.500</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMC</td>
<td>Random</td>
<td>100.0%</td>
<td>2158.4%</td>
<td>830.9%</td>
<td>92.4%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td></td>
<td>Positive</td>
<td>100.0%</td>
<td>3418.7%</td>
<td>830.9%</td>
<td>92.4%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td></td>
<td>Negative</td>
<td>100.0%</td>
<td>2158.4%</td>
<td>1.7%</td>
<td>29.3%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td>Bootstrap</td>
<td>Random</td>
<td>100.0%</td>
<td>90.0%</td>
<td>94.6%</td>
<td>8.9%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td></td>
<td>Positive</td>
<td>100.0%</td>
<td>109.0%</td>
<td>91.2%</td>
<td>9.5%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td></td>
<td>Negative</td>
<td>100.0%</td>
<td>90.0%</td>
<td>10.2%</td>
<td>0.1%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
</tbody>
</table>

Table 2: Relative change in likelihood over a certain change in parameter values for one unknown. Random indicates the random change as displayed in the graphs, positive indicates all changes are positive and vice versa. The lower bound of the confidence interval for the value of $r_c$ indicates the lowest value in the 95% likelihood confidence interval for that parameter divided by the highest value in the 95% likelihood interval of the true parameter. The upper bound is the highest value within the 95% interval for that parameter divided by the lowest value in the interval of the true parameter. This leads to an approximately 90% confidence interval.

To review the sensitivity in numbers, we inspect the log transformed mean likelihoods of both methods. These can be observed in Table 3. We notice that even though in the sensitivity we saw multiple likelihoods of parameters for higher $r_c$ being zero percent of the original likelihood, they are descending in size. i.e. for parameter settings closer to the true one, the likelihoods are generally larger.

<table>
<thead>
<tr>
<th>Method</th>
<th>$r_c$</th>
<th>0.000</th>
<th>0.001</th>
<th>0.010</th>
<th>0.020</th>
<th>0.050</th>
<th>0.100</th>
<th>0.500</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMC</td>
<td></td>
<td>4.9</td>
<td>7.9</td>
<td>7.0</td>
<td>4.8</td>
<td>-3.5 · 10</td>
<td>-2.4 · 10²</td>
<td>-3.0 · 10³</td>
</tr>
<tr>
<td>Bootstrap</td>
<td></td>
<td>1.6 · 10</td>
<td>1.6 · 10</td>
<td>1.6 · 10</td>
<td>1.3 · 10</td>
<td>-7.0</td>
<td>-2.4 · 10²</td>
<td>-7.7 · 10³</td>
</tr>
</tbody>
</table>

Table 3: The log transformed likelihood estimate for 1 unknown parameter.

In order to assess the performance of each technique, we are combining the sensitivity with the relative error. Our objective is to determine if the 95% confidence intervals of the likelihood estimates overlap for various parameter values. If the intervals overlap, it is more challenging to determine the direction of the true parameters. On the other hand, if the intervals do not overlap and are increasing for parameters that are closer to the true ones,
it is significantly easier to identify the direction of the true parameters. Hence, returning to
the relative error, we take on educated guess of the relative error of before, which is a 0.8
relative error for CMC and 0.05 for Bootstrap These estimates which are quite conservative.
We deduce the lower bound and upper bound of the confidence interval with the following
equations, noting the likelihood estimate as \( l \).

\[
\text{Upper bound} = \log(l) + \log(1 + 1.96 \cdot RE) \\
\text{Lower bound} = \log(l) + \log(1 - 1.96 \cdot RE)
\]

Where if the value inside the lower bound becomes negative we simply regard it as 0, as the
likelihood will not go beneath zero.
For CMC with a relative error of 0.8, all 95% confidence intervals overlap, as they all contain
zero. For Bootstrap with a relative error of 0.05, after a ratio of change of 0.01 none of the
confidence intervals overlap. Hence, in contrast to CMC, the estimates are expected to be
more likely to give a correct indication of the direction towards the true parameters for the
parameter search for a change larger than 0.02, as the likelihoods for these ratio changes are
with at least 95% probability estimated larger if closer to the true parameters. For changes
smaller than 0.02, Bootstrap might be unreliable as well, hence an error of up to 0.02 is
possible. One could consider such an size of error acceptable though, and possibly even
favorable for the ease of MLE.

### 4.1.2 2 Unknown Parameters

One can observe the likelihood estimate results for different parameter settings of \( \eta \) and \( \mu_A \)
in Figure 3. Here, no zoomed plot is included as the likelihood results for CMC also reach
very high values for the true parameter setting.
Similar to the results for one unknown parameter for CMC, these results show that the like-
lihood’s estimate 95% confidence interval of the true parameter values and also of parameter
settings close by the true one is overlapping with the likelihood results of all other parameter
settings. Therefore, applying CMC in maximum likelihood estimation is again unreliable,
due to same the reasoning as before. The Bootstrap does not have overlapping confidence
intervals, and for the first few parameter settings the results are much higher than for others.
In fact, the parameter setting with 0.001 ratio change of the true value results in the highest
mean likelihood estimate, with a 95% confidence interval not overlapping the true param-
eter setting or any other setting. Thus, for the synthetic data used, maximum likelihood
estimation using Bootstrap will likely estimate parameters with a 0.001 ratio change to the
true parameters for a large enough parameter search. Otherwise, the likelihood within 0.01
change is nonetheless distinctly higher than for parameters further away as well, hence a less
extensive parameter search will likely even so lead to a estimation within a 0.01 change. One
could consider this quite close by, and hence a well performing method. The mean likelihood
of the true parameter values deviate approximately 22 times its 95% half-width from zero,
where the likelihood lies from a deviation of 0.02 of the true parameter values. Hence, it is
extremely unlikely that for a sufficient parameter search, parameter values with a deviation
of 0.02 or more will be returned as an outcome of maximum likelihood estimation.
Note that for a ratio change of 0.01 the estimates of Bootstrap are significantly lower in the

M.B.C. Buist
A Comparison of CMC Against SMC

Figure 3: Likelihood comparison, including the mean estimated likelihood value and confidence interval for different parameter settings of the 2 unknowns.

Two unknowns case compared to the one unknown case, which is likely as two parameters are adjusted, hence the setting is further away from the true one than when only one parameter is adjusted.

In addition, as we have observed once again CMC is significantly less reliable than Bootstrap due to the high relative error, we take a closer look at the relative errors. These values are listed beneath in Table 4. Same as before, we leave out some parameter cases due to an inaccurate estimation of the relative error.

<table>
<thead>
<tr>
<th>Method</th>
<th>$r_c$</th>
<th>0.000</th>
<th>0.001</th>
<th>0.010</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMC</td>
<td>0.9982</td>
<td>0.7752</td>
<td>0.9974</td>
<td></td>
</tr>
<tr>
<td>Bootstrap</td>
<td>0.0229</td>
<td>0.0228</td>
<td>0.0367</td>
<td></td>
</tr>
</tbody>
</table>

Table 4: The relative error for both estimation methods over the different parameter inputs for two unknowns.

Similar to the case of one unknown, there is a humongous difference in the relative errors, with Bootstrap performing outstandingly better. Therefore, confirming Bootstrap to be a more suitable method of likelihood estimation for MLE.

Anew, we examine the sensitivity of the likelihood estimates around the true parameter values in Table 5 for evaluation of the suitability to MLE. CMC seems to be a lot more...
A Comparison of CMC Against SMC

sensitive than Bootstrap again, with the likelihood decreasing very rapidly with only a 0.001 change in parameter values. Bootstrap on the other hand has a slightly slower decay, which would make it easier to find the true parameters using MLE. Once again, CMC has a confidence interval of every possible value, due to its devastating variance. Bootstrap is a lot more precise and hence reliable. The likelihood seems to be slightly more sensitive again with a negative change compared to a positive change in parameters for both methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>$r_c$</th>
<th>0.000</th>
<th>0.001</th>
<th>0.010</th>
<th>0.020</th>
<th>0.050</th>
<th>0.100</th>
<th>0.500</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMC</td>
<td>Random</td>
<td>100.0%</td>
<td>1.3%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[0; Inf]</td>
<td>[0; Inf]</td>
<td>[0; Inf]</td>
<td>[0; Inf]</td>
<td>[0; Inf]</td>
<td>[0; Inf]</td>
<td>[0; Inf]</td>
</tr>
<tr>
<td></td>
<td>Positive</td>
<td>100.0%</td>
<td>1.3%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[0; Inf]</td>
<td>[0; Inf]</td>
<td>[0; Inf]</td>
<td>[0; Inf]</td>
<td>[0; Inf]</td>
<td>[0; Inf]</td>
<td>[0; Inf]</td>
</tr>
<tr>
<td></td>
<td>Negative</td>
<td>100.0%</td>
<td>0.3%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[0; Inf]</td>
<td>[0; Inf]</td>
<td>[0; Inf]</td>
<td>[0; Inf]</td>
<td>[0; Inf]</td>
<td>[0; Inf]</td>
<td>[0; Inf]</td>
</tr>
<tr>
<td>Bootstrap</td>
<td>Random</td>
<td>100.0%</td>
<td>121.7%</td>
<td>25.7%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[111.2; 133.1]</td>
<td>[22.8; 28.8]</td>
<td>[0; 0]</td>
<td>[0; 0]</td>
<td>[0; 0]</td>
<td>[0; 0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Positive</td>
<td>100.0%</td>
<td>121.7%</td>
<td>13.3%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[111.2; 133.1]</td>
<td>[11.5; 15.2]</td>
<td>[0.0; 0.0]</td>
<td>[0.0; 0.0]</td>
<td>[0.0; 0.0]</td>
<td>[0.0; 0.0]</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Negative</td>
<td>100.0%</td>
<td>77.2%</td>
<td>0.2%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
<td>0.0%</td>
</tr>
<tr>
<td></td>
<td></td>
<td>[70.2; 84.9]</td>
<td>[0.2; 0.2]</td>
<td>[0.0; 0.0]</td>
<td>[0.0; 0.0]</td>
<td>[0.0; 0.0]</td>
<td>[0.0; 0.0]</td>
<td></td>
</tr>
</tbody>
</table>

Table 5: Relative change in likelihood over a certain change in parameter values for two unknowns. Random indicates the random changed as displayed in the graphs, positive indicates all changes are positive and vice versa. The lower bound of the confidence interval for the value of $r_c$ indicates the lowest value in the 95% likelihood confidence interval for those parameters divided by the highest value in the 95% likelihood interval of the true parameters. The upper bound is the highest value within the 95% interval for those parameters divided by the lowest value in the interval of the true parameters. This leads to an approximately 90% confidence interval.

Next, even though the likelihoods are zero percent of the original for further away parameter values, in Table 6 the log transformed values show that they do differ over the various parameter values, and in fact decrease for values with a larger deviation from the true setting.

<table>
<thead>
<tr>
<th>Method</th>
<th>$r_c$</th>
<th>0.000</th>
<th>0.001</th>
<th>0.010</th>
<th>0.020</th>
<th>0.050</th>
<th>0.100</th>
<th>0.500</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMC</td>
<td></td>
<td>1.5 \cdot 10</td>
<td>1.1 \cdot 10</td>
<td>6.9</td>
<td>-9.3</td>
<td>-1.7 \cdot 10^2</td>
<td>-4.0 \cdot 10^2</td>
<td>-1.0 \cdot 10^3*</td>
</tr>
<tr>
<td>Bootstrap</td>
<td></td>
<td>1.6 \cdot 10</td>
<td>1.6 \cdot 10</td>
<td>1.4 \cdot 10</td>
<td>4.0</td>
<td>-1.8 \cdot 10^2</td>
<td>-4.3 \cdot 10^2</td>
<td>-1.2 \cdot 10^7</td>
</tr>
</tbody>
</table>

Table 6: The log transformed likelihood estimate for 2 unknown parameters.
* This is the lower bound for CMC, considered zero.
Returning to the relative error and assuming a 0.8 RE for CMC and 0.04 RE for Bootstrap, we compute the upper and lower bounds of the 95% confidence intervals. Again, for CMC they all contain the value zero, hence they all overlap. For Bootstrap on the other hand, none overlap. The second value is larger than the one of the true parameter setting however. As the deviation is merely 0.001 which is assumed to be an acceptable error, this is no problem. For the other values, it holds that the closer to the true parameters the larger the likelihood. Thus, with a relative error of 0.04 Bootstrap would presumably correctly indicate the direction of the true parameters for these changes.

4.1.3 6 Unknown Parameters

Lastly, the results for leaving all parameters unknown can be seen in Figure 4. In this figure, one can observe an even more rapid decay in likelihood estimates for Bootstrap than in the previous two cases. Once again, this is likely due the fact that more parameters, six to be precise, are diverted compared to only one or two parameters. The likelihood mean and 95% confidence interval are clearly higher for the true parameter setting than any other setting for Bootstrap. However, as the decay in likelihood is very rapid over a parameter change, maximum likelihood estimation may be required to go over a rather extensive parameter search to find parameters producing a high likelihood, hence complicating the estimation. For a sufficient parameter search, estimation through maximum likelihood applying Bootstrap is likely to lead to estimates approaching the true values extremely well, at least with a lower deviance than 0.001 of the total interval of all distributions, nonetheless do note that the parameter search requires an efficient method to reach these precise estimates. The mean likelihood lies at a distance of approximately 22 half-widths away from zero, which is where likelihoods from a deviation of 0.01 from the true parameter settings lie close to. CMC has a rapid decay in likelihood value as well, thus may also require an extensive parameter search for MLE. Moreover, CMC shows again results in which (values close to) the true parameter settings have an overlapping 95% confidence interval with the results of all other parameter settings. Thus similar as before, CMC is not reliable in maximum likelihood estimation for this data and model and Bootstrap is a significantly more reliable algorithm.

Once more, we analyze the relative errors of the method to bring to light the exact difference between the reliability of Bootstrap and CMC. The relative errors for the true parameter setting and a 0.001 change can be observed in Table 7.

<table>
<thead>
<tr>
<th>Method</th>
<th>$r_e$</th>
<th>0.000</th>
<th>0.001</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMC</td>
<td>0.7581</td>
<td>0.9577</td>
<td></td>
</tr>
<tr>
<td>Bootstrap</td>
<td>0.0234</td>
<td>0.0386</td>
<td></td>
</tr>
</tbody>
</table>

Table 7: The relative error for both estimation methods over the different parameter inputs for six unknowns.

Not surprisingly, the relative errors of Bootstrap are tremendously smaller than the ones of
A Comparison of CMC Against SMC

Figure 4: Likelihood Comparison for 6 Unknown Parameters

(a) Including the mean estimated likelihood value and confidence interval for different parameter settings of the 6 unknowns

(b) Including the mean estimated likelihood value and confidence interval for different parameter settings of the 6 unknowns with a zoomed in y-axis

M.B.C. Buist
CMC, indicating the method is extremely more reliable and hence more suitable for MLE.

As can be observed from Table 8, the likelihood seems to be extremely sensitive to only a slight change in parameters for both methods. CMC again has the intervals covering the entire space of possible values, while Bootstrap seems quite precise. For a change of only 0.01 in parameter values we observe that the likelihood has declined to approximately zero percent of the likelihood of the true parameters. Anew, the likelihood seems to be more sensitive to negative change than to positive change.

Investigating the true values in log transform for the random change, we do observe that the values are still different from one another, with larger likelihoods for parameters closer to the true ones, which could give a sense of direction for estimating the true parameters.
A Comparison of CMC Against SMC

<table>
<thead>
<tr>
<th>Method</th>
<th>$r_c$</th>
<th>0.000</th>
<th>0.001</th>
<th>0.010</th>
<th>0.020</th>
<th>0.050</th>
<th>0.100</th>
<th>0.500</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMC</td>
<td>4.2</td>
<td>5.3</td>
<td>-3.6·10^2</td>
<td>-4.7·10^2</td>
<td>-4.1·10^2</td>
<td>-6.1·10^3</td>
<td>-1.0·10^4*</td>
<td></td>
</tr>
<tr>
<td>Bootstrap</td>
<td>1.6·10</td>
<td>1.4·10</td>
<td>-1.3·10^3</td>
<td>-1.6·10^3</td>
<td>-2.0·10^3</td>
<td>-1.2·10^5</td>
<td>-4.4·10^9</td>
<td></td>
</tr>
</tbody>
</table>

Table 9: The log transformed likelihood estimate for 6 unknown parameters.  
* This is the lower bound for CMC, considered zero.

To actually check if the values are reliable to give a sense of direction of the true parameters, we combine the values in Table 9 with the relative error. Taking conservative estimates, we assume CMC to have a relative error of 0.8 and Bootstrap of 0.04. Without surprise, all 95% confidence intervals of CMC overlap, as they all contain zero. For Bootstrap with a relative error of 0.04, none of the confidence intervals overlap. Thus again, Bootstrap is significantly more reliable to indicate the right direction of the parameter search than CMC, as the likelihoods for the different $r_c$ are with at least 95% probability estimated to be larger if closer to the true parameters.

As has been repeatedly stated throughout the entire Synthetic Data subsection, Bootstrap in general estimates likelihoods with significantly smaller relative errors leading its estimation to be heavily more accurate with increasing likelihoods towards the true parameter setting. Therefore, making the method essentially more suitable for MLE of the parameters than the state of art Crude Monte Carlo method.

4.2 Real Data

4.2.1 Extracted Real Data

The real data used in this research is extracted from (Chow & Shao, 1991). The paper aims to find a better estimation of the shelf-life of drugs than is now stated on the label. It firstly assumes the potency data of drugs to behave as $y_{ij} = \alpha_i + \beta_j x_j + \epsilon_{ij}$ where $i \in \{1, \ldots, k\}$ is the component and $j \in \{1, \ldots, n\}$ indicates the time period. Here $\alpha_i$ are i.i.d. with mean $a$ and variance $\sigma^2_{\alpha_i}$; $\beta_j$ are i.i.d. with mean $b$ and variance $\sigma^2_{\beta_j}$ and lastly $\epsilon_{ij}$ are i.i.d. with mean 0 and variance $\sigma^2_{\epsilon}$ for all $i \in \{1, \ldots, k\}$ and $j \in \{1, \ldots, n\}$. The paper believes that as most drug producing processes will not allow for a large variance $\sigma^2_{\alpha}$ and $\sigma^2_{\beta}$ for conformity, and therefore they do an estimation assuming the variances equal zero, hence without batch-to-batch variation. This results in a simple linear regression problem. A widely used method to estimate parameters in such problems is OLS, though as the covariance matrix can be consistently estimated, WLS is more efficient and hence applied in this paper. The paper finds the shelf-life to be higher than the conservatively established shelf-lives. The shelf-life is estimated to be on average 64 months, though varying the parameters within their 95% confidence interval could differ the potency results of the drugs after 64 months still with a few percentage points. If the time to approximate over is short this can be considered redundant, however if one needs to estimate in the far future, for instance 10 years, this standard error could be considered a problem. We expect the general degradation model
not to be in the linear form that the paper states, but to be represented better with a gamma process degradation. Therefore, this research considers another method. In the next subsection we analyze the fit of the gamma process degradation for the data.

### 4.2.2 A Brief Data Analysis

As mentioned in the beginning of this section, the results of this thesis depend on real data applied and the correctness of the model for this data. Because of this, we will execute a short data analysis on the real data extracted from (Chow & Shao, 1991). We will evaluate the distribution of this data and how likely it is that this data conforms to the model assumptions in 1.

According to the model assumptions, we have that

\[
\Delta Y_i(t(l)) = \Delta X_i(t(l)) + \Delta Z_i(t(l))
\]

where

\[
\Delta X_i(t(l)) \sim \text{Gamma}(\alpha t(l)^\eta - \alpha t(l-1)^\eta, \beta)
\]

\[
\Delta Z_i(t(l)) \sim \mathcal{N}(0, 2\sigma_Z^2)
\]

Hence, the difference in the observation data should be distributed as a gamma plus a normal distribution.

To check this assumption, we start by plotting the density of the differences in observation data. In other words, as the measurement times are 1, 2 and 3 years, we plot \(y(2) - y(1)\) and \(y(3) - y(2)\). The smoothed densities are visualized in Figure 5. As there are only 24 components, so 24 observations per time difference, the data is not at all smooth, which is likely the reason why there are bumps in the density functions. This small number of components will also cause the comparison with a distribution to be harder. In both figures we see a distribution that for the largest part seems to resemble a normal, except a slight skew to the right for the first plot and a skew to the left for the second plot. These different bumps in the figure however also remind us of the convolution of two normally distributed random variables. Nevertheless, as they are both to another side of the maximum, it tends us to believe that they are simply irregularities which are a consequence of the small number of components. A skew to the left would not suit the sum of a gamma and normal random variable, as the gamma is skewed to the right and a normal has no skew.

To very informally compare this distribution with a sum of a gamma and normal distributed random variable, we plot such sums for different parameter settings. Consider \(k\) to be the shape parameter and \(\beta\) the scale parameter of the gamma distribution and consider \(\sigma\) to be the standard deviation of the normal distribution. Hence, for this model, we would have

- \(k = \alpha t(l)^\eta - \alpha t(l-1)^\eta\),
- \(\beta = \beta\),
- \(\sigma = \sqrt{2\sigma_Z^2}\)
A Comparison of CMC Against SMC

Figure 5: Densities of the degradation plus difference in measurement error observed

(a) A density plot of the difference in observed degradation between the second and the first measurement

(b) A density plot of the difference in observed degradation between the third and the second measurement

M.B.C. Buist
Figure 6: Probability densities for different parameter settings of a sum of a gamma distributed RV and a normal distributed RV

Then, for different values of these parameters the probability density functions (pdf’s) can be observed in Figure 6. The chosen parameter values are based on the prior distributions applied in the next section and values we suspect to be reasonable. While the gamma distribution can have quite a skew to the right, the pdf’s of these sums of gamma and normal random variables resemble a normal distribution quite well, which is presumably due to the fact that the gamma distribution converges to a normal distribution for large $k$. Hence for the prior distribution boundaries of the parameters, in particular the scale factor not exceeding 1 and $\sigma$ being relatively large, these distributions will fairly mimic a normal distribution. Comparing the densities of the observed data in Figure 5 with the sample densities in Figure 6 and recognizing that some irregularities in the density of the observed data are likely present due to the small number of components, it seems plausible that the distribution of the data could be from the sum of a gamma and a normal distributed random variable. The distributions of the observed data without the small increments on either side namely appear rather similar to the purple line in Figure 6.

Next to this informal comparison, ultimately we would like to perform a statistical test to test the model assumption. A common method is to do a Goodness-of-Fit test, using the Kolmogorov-Smirnov statistic for instance. Say $F$ is the distribution of the observation data, thus either of $y(2) - y(1)$ or of $y(3) - y(1)$, and $F_0$ is the distribution we want to test it for. A Goodness-of-Fit test would then test the following hypothesis

$$H_0 : F = F_0 \text{ against } H_1 : F \neq F_0.$$
For this test one needs to have a specified distribution $F_0$ to test for. In other words, the value of the parameters of the distribution are required as well. In this thesis however, the model assumption is simply that the data comes from a gamma plus a normal distributed random variable, not specifying the parameter values. Hence, say $\mathcal{F}$ is the class of distributions of a gamma plus a normal distributed random variable, then ultimately, the test that one wants to execute is

$$H_0 : F \in \mathcal{F} \text{ against } H_1 : F \notin \mathcal{F}.$$ 

That is, not testing for a specific parameter set, though testing for the whole class of distributions. This form of test, where one tests for the entire class of distributions, is called a composite Goodness-of-Fit test (Bucchianico & Castro, 2022). Due to ease of execution we simply test on the class of gamma distributions as well as the class of normal distributions separately in the upcoming part. Do note however that testing on the sum is more favorable. The test is executed in Rstudio. Another note is that due to the small number of components as well as the rounding off in observed data, the data will be less in accordance with the continuous distribution it is tested for.

To start, the hypothesis test of the data being gamma distributed is performed. The test statistic we apply is the Anderson-Darling for the gamma statistic. For the two data sets the following $p$-values are returned

- $y(2) - y(1) : 0.01 \leq p < 0.05$
- $y(3) - y(2) : p < 0.01$

Hence, for a certainty of 99%, the null hypothesis of the first time difference is not rejected. It is however still quite low giving the impression that the data is unlikely to be from a gamma distribution. The density of the first time difference, seen in 5(a) is skewed to the right, just as a gamma can be, which makes the higher $p$-value for this data compared to the second time difference not very surprising. For the second time difference, the $p$-value is lower than 0.01, hence the null hypothesis of the true distribution being a gamma distribution would be rejected with 99% certainty.

Next, the hypothesis of the data being normal distributed is tested. For this the Lilliefors Goodness-of-Fit test is applied, which is based on the Kolmogorov-Smirnov test statistic. While this is a relatively low-power test, other composite hypothesis tests like Anderson-Darling and Cramer-von Mises are not applicable for the second time difference as the data does not contain at least seven distinct values. Performing the Lilliefors test results in the following $p$-values

- $y(2) - y(1) : 0.004073158$
- $y(3) - y(2) : 0.001465543$

One can deduce that for both time differences the $p$-value is lower than 0.01, hence with a 99% certainty level one would reject the null hypothesis of the data coming from a normal distribution.
Summarizing both results, for both tests individually and both time differences the null hypotheses were rejected with a 95% certainty level. However, even though the \( p \)-values are low, they are not extremely low, hence not too distant from 0.01 or 0.05. One could envision that for more data points, factoring out irregularities, and for the test on the class of distributions of a gamma plus a normal, that the test would not be rejected. Hence, considering the similarity in densities between 5 without the bumps and 6 and observing the \( p \)-values above are not extremely low, it is not unreasonable to believe that the data is originally distributed as a sum of a gamma and a normal random variable. However, this sample, especially the second time difference with a skew to the left, might not be most representative of the distribution, which could cause for smaller likelihood values. Having concluded that the data may be modeled as the model in 1, we continue to apply the CMC and Bootstrap likelihood estimation on the data using this particular model.

### 4.2.3 Numerical Results

This is different data from (Hazra et al., 2020), however for the most part we apply the same prior distributions. The only prior distribution that is adjusted is the one of \( \sigma_Z \), (Chow & Shao, 1991) namely indicates that this parameter might have a larger value than the upper bound of the prior distribution in (Hazra et al., 2020). Differing the other prior distributions does not lead to significantly better results. Hence, the following prior distributions are integrated:

- \( U[0, 50] \) for \( \alpha \),
- \( U[0, 5] \) for \( \eta \) and \( \mu_A \),
- \( U[0, 1] \) for \( \beta \) and \( \sigma_A \),
- \( U[0, 5] \) for \( \sigma_Z \).

As the true parameters are unknown, we cannot create graphs showing the estimates for the true parameter values and with an increasing drift. As could be seen in the previous graphs, the likelihood could decrease very rapidly for an increasing drift, hence many parameter settings will result in a likelihood close to zero. As such, for many different parameter settings we will randomly draw parameters from the prior distributions stated above, and plot the mean likelihood estimator with its 95% confidence interval. In particular, the number of different parameter settings used to obtain the results is \( N_\theta = 18000 \). Next to that, the number of particles is \( N_{\text{part}} = 10000 \) and the number of simulations \( N_{\text{sim}} = 100 \).

**Crude Monte Carlo**

First, we start by examining the ten highest estimated likelihoods of CMC depicted in Figure 7. An observation that is very apparent, is the magnitude of the likelihood values. They are extremely low, especially compared to the previous likelihoods for the synthetic data. A cause for this could be that the model simply does not fit the data, causing the low likelihoods. However, the explanation we deem most plausible, is that the true parameter set has not been approached sufficiently. As stated before, the likelihood decreased rapidly.
for an increasing drift in the synthetic data setting. There are six unknown parameters, hence it is very unlikely to have the exact combination of parameters that approaches the true values substantially close. Suppose we would want to have all the parameters within a 0.01 ratio of change, as this was a change that had a likelihood of order $10^{-157}$ for CMC and even smaller for Bootstrap in the six unknowns case of the synthetic data, then the probability of drawing that parameter setting is

$$(2 \cdot 0.001)^6 = 6.4 \cdot 10^{-11}$$

which would imply that in order to draw such a value applying uniform draws, one should draw over $N_\theta = 10^{11}$ different parameter settings, which in the synthetic data would yet cause a likelihood of order $10^{-157}$, hence even lower than the current outcomes. Considering the program speed, running over this many parameters is practically impossible. Due to this reasoning however, this is a quite probable explanation of the low likelihood values. As the small likelihoods can still cause for an indication of direction however, small likelihoods need not to be a problem. To approach the eventual true parameters nevertheless, this direction should be exploited for a more efficient parameter search. In such a manner, larger likelihoods can be reached as well.

Next to this observation, one can see that the 95% confidence interval of the highest likelihood value is fairly wide, covering zero as well. As discussed in the synthetic data subsection, this causes for unreliable results in maximum likelihood estimation. For the second highest likelihood value, the 95% confidence interval does not cover zero by just a fraction. A slightly higher percentage confidence interval would hence cover zero. This indicates its high variance and hence unreliability as well.

Looking at the parameter values on the x-axis, while there might be some similar values for the largest likelihoods, there does not seem to be a definite choice for the parameters. Comparing these values to the study where the data originated from, it would roughly estimate $\mu_A = 1.032$, $\sigma_A = 0$ and $\beta = 0.418$. For the highest likelihood setting, the estimations for $\sigma_A$ and $\beta$ are not too different, though the estimation of $\mu_A$ is rather different. There is still a reasonable wide variance between the parameter values in Figure 7, this in specific indicates a problem for the overlapping confidence intervals. It is not certain which parameter setting would be chosen due to the high variance and hence overlapping confidence interval, thus outcomes can vary a lot depending on random draws. The number of simulations $N_{\text{sim}}$ was ten times smaller for the real data case than for the synthetic data, which could cause wider confidence intervals. Nonetheless, the synthetic data case showed too wide confidence intervals as well, hence the method performs poorly in both settings.

**Sequential Monte Carlo**

Similar to the CMC graph, Figure 8 shows rather small likelihood values. Surprisingly, they are even (slightly) smaller than the CMC values. We observed however with the synthetic data, that the decline in likelihood value was more steep for Bootstrap than for CMC, accompanied by lower likelihoods for parameters with a larger drift from the true setting. This could explain the slightly smaller likelihoods.

However, while in the synthetic data case the confidence intervals did not overlap each other or zero, the 95% confidence interval of the highest likelihood is very wide and covers zero as
Figure 7: The 10 highest likelihood values of 18000 different parameter settings of CMC for the real data set, including the line indicating $y=0$

Figure 8: The 10 highest likelihood values of 18000 different parameter settings of Bootstrap for the real data set, including the line indicating $y=0$
A Comparison of CMC Against SMC

well, which indicates unreliability. The second and third highest do not have their confidence interval overlapping zero however, though still reasonably close, which is undesirable. It is to note however that they are relatively further away from zero than the second value from CMC.

Moreover, we observe the parameter values on the x-axis with overlapping confidence intervals varying a lot from each other. Again, (Chow & Shao, 1991). would roughly estimate $\mu_A = 1.032$, $\sigma_A = 0$ and $\beta = 0.418$. Compared to CMC, $\mu_A$ is estimated closer to this value, $\beta$ a bit further away and $\sigma_A$ practically the same for the highest likelihood setting. However, as the confidence intervals are overlapping, we are not sure whether the direction that the results indicate for the parameters is correct or not. All in all, the results seem as if the method is reasonably unreliable. Increasing $N_{\text{sim}}$ could decrease the interval widths, as this parameter is a factor of 10 smaller in the real data case.

As we have seen large confidence intervals, which cause unreliability of a method, we analyze the relative errors of both methods.

<table>
<thead>
<tr>
<th>Method</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
<th>9</th>
<th>10</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>CMC</td>
<td>0.9575</td>
<td>0.4554</td>
<td>0.7063</td>
<td>0.9612</td>
<td>1.0000</td>
<td>0.9744</td>
<td>0.6981</td>
<td>0.1408</td>
<td>0.6556</td>
<td>0.8473</td>
<td></td>
</tr>
<tr>
<td>Bootstrap</td>
<td>0.9177</td>
<td>0.3367</td>
<td>0.4040</td>
<td>0.9740</td>
<td>0.7304</td>
<td>0.5741</td>
<td>0.4412</td>
<td>0.1462</td>
<td>0.3872</td>
<td>0.4092</td>
<td>0.3220</td>
</tr>
</tbody>
</table>

Table 10: The relative error for both estimation methods over the different parameter inputs for the ten highest likelihoods and the overall mean of all relative errors excluding 0 and NaN values. Values are at the x highest likelihood or the mean.

As can be observed in Table 10, even though Bootstrap seems unreliable as well, the relative errors remain smaller in general than the relative errors of CMC. This implies that the Bootstrap method remains more reliable than the CMC method, considering the likelihood estimates of Bootstrap vary more as well. Compared to the relative errors in the synthetic data however, the performance of Bootstrap is significantly inferior in this case.

Possible explanations for the high variance could include the model not fitting correctly, for instance through the presence of a hidden stochastic in the true model. This could be an explanation for the small likelihood estimates as well. A composite hypothesis test on a normal plus gamma distributed random variable could make the presence of a hidden stochastic more clear, as well as availability to more data. Moreover, the process could be chaotic as well, random outliers would increase the variance. Next to that, due to the small values that MatLab has to deal with, there exists the probability that machine error is the reason of the larger relative errors. Lastly, increasing the number of simulations $N_{\text{sim}}$ could decrease the relative error, as it is a factor 10 smaller than for the synthetic data due to computer efficiency. We do not suspect the scale of the process to cause any harm as different parameter ranges did not drastically change the likelihood and if that were the case the results would exhibit at least one parameter value to be estimated close to the boundary of the range.
To conclude, the estimated likelihood values are rather low and relative variances are of substantial size. We assume that the low likelihoods are at least partly attributed to the method of choosing parameter settings to test over in combination with the number of parameter settings tested over. A more sophisticated method is discussed in Section 6, which will approach the true parameter settings sooner. If the true parameters were approached better, higher likelihoods would be estimated presumably. Increasing the number of simulations $N_{\text{sim}}$ could decrease the relative error. Even though both methods have a high relative error which is destructive for the reliability, Bootstrap has a significantly smaller relative error than CMC in general. This in combination with the fact that Bootstrap in general has a larger difference in likelihood values leads to the conclusion that the Bootstrap method is more suitable for MLE.
5 Conclusions

**Crude Monte Carlo will fail for a too large data set and high noise variance**

First of all, the analytical study revealed that Crude Monte Carlo is very vulnerable to failure. It requires a large number of particles in order to have a reasonable relative error, due to the rare-event setting. Moreover, if $N_{\text{part}}$ is not large enough, then CMC can fail because of the likelihood being estimated zero, which happens if for all $N_{\text{part}}$ runs the degradation is estimated to be negative at least once over all components and time periods. CMC is vulnerable to this as the degradation can be estimated to be negative, which breaches the model assumptions, due to the drawn measurement errors and initial degradation. The larger the data set and the higher the noise variance $\sigma_Z^2$, the larger this probability of failure, which eventually converges to 1. Due to the set up of Sequential Monte Carlo, it is not vulnerable to this failure.

*For synthetic data, Bootstrap is significantly more reliable than CMC, estimating the likelihoods accurately and not prone to having overlapping confidence intervals for reasonably distant parameters.*

The comparison of both methods using the synthetic data shows an obvious outperformance of SMC as compared to CMC. While CMC includes zero in the 95% interval at the true parameter setting for all three cases, Bootstrap differs approximately 22 times its 95% half-width from zero for the true parameter setting, making an estimation through maximum likelihood estimation of more than 0.05 deviation from the true parameter setting extremely unlikely for Bootstrap given an adequate parameter search. For CMC, on the other hand, as the confidence interval overlaps with zero, an estimation of any other parameter setting, regardless of the deviation to the true parameter setting, is not exceptional. The relative error is significantly lower for the Bootstrap method than for the Crude Monte Carlo method, hence, estimates are in general expected to be substantially more accurate and less overlapping. Because there is a larger difference in likelihood estimates for Bootstrap, the chances of overlapping confidence intervals decrease as well. As such, this leads Bootstrap to have an essentially higher probability of indicating the true direction of the parameters and truly estimating the parameters nearest to the true values through maximum likelihood estimation. Thus, we conclude that this case study shows that Bootstrap is a significantly more reliable method for estimation of the likelihood: the probability of the likelihood for the true parameters, or parameters close to the true values, being larger than that of parameters further away is substantially larger for the Bootstrap method than for the CMC method. In fact, considering the relative errors of merely 0.05 maximal, Bootstrap is not only better than CMC though in general a well performing method.
A Comparison of CMC Against SMC

For the real data, both methods have a high relative error which is unfavorable. Nonetheless, Bootstrap’s relative error is generally still substantially lower than CMC’s relative error, making the method more reliable.

The results of the real data present a significantly inferior performance of Bootstrap compared to its performance for synthetic data. The relative error increased critically, leading to less reliable results. Several 95% confidence intervals overlap with zero, hence overlap with all other lower likelihood values. Various reasons for the high relative error could be the model not fitting the data correctly, for instance by the presence of a hidden stochastic. While the data did reasonably resemble a normal plus gamma distributed random variable, there were some irregularities, and, no composite hypothesis test has been executed for this combined distribution. Moreover, the data set is fairly small for a decent data analysis. A more thorough data analysis accompanied by a larger data set could clarify the presence of a hidden stochastic and the model fit. In addition, even though it is unlikely considering the data, the process may as well be chaotic, leading to larger errors. Anew, a more sophisticated data analysis could clarify. Moreover, as values that are dealt with are rather small, machine error could possible cause the estimation of the relative error to be incorrect. Finally, the number of simulations $N_{\text{sim}}$ is set notably smaller for the real data case than for the synthetic data. For the synthetic data, $N_{\text{sim}} = 1000$, for the real data $N_{\text{sim}} = 100$. This might very well be the explanation for the larger relative errors. In (Lu et al., 2013), the error was 0.3 for running Crude Monte Carlo approximately 5500 times, and showed a decrease to about 0.02 for $10^6$ simulations.

Next to the large relative errors, it was noticeable that both methods presented low likelihood estimates, that is, their magnitude might practically be considered zero. Presumably, this is at least partly due to the parameter search, which is relatively low considering the substantial decrease in likelihood value over a parameter change in the synthetic data case. As well as for the relative error, an incorrect model fit could explain the lower likelihoods too.

Despite the overall small likelihoods and large relative error, the relative error of the Bootstrap method compared to the Crude Monte Carlo method remains considerably better. Combining that observation with the knowledge of Bootstrap varying more in likelihood values, Bootstrap is deemed to be better suitable for MLE than Crude Monte Carlo.

All in all, Bootstrap shows to perform significantly better given the assumed model, being a more reliable method to apply in maximum likelihood estimation. For the real data, we observed a more reliable performance from the Bootstrap method as well. The only benefit CMC has against Bootstrap is that the algorithm is more computationally efficient. For the synthetic data case, the algorithm of CMC was approximately four times faster than the algorithm of Bootstrap. The reliability of Bootstrap for the real data may however still not be sufficient, and therefore, it would be beneficial to conduct further investigation of the cause of the high relative errors. Development of the likelihood estimation, for instance the implementation of the auxiliary particle filter, may as well result in an even more reliable method of estimation.

M.B.C. Buist
6 Discussion and Recommendations for Future Research

This thesis shows a clear outperformance of Sequential Monte Carlo against Crude Monte Carlo. However, for SMC the only particle filter implemented was the common Bootstrap filter, while the guided particle filter or the auxiliary particle filter are particle filters that allow for more freedom hence potentially for a better approximation. As described in the Introduction and Literature 1, the auxiliary particle filter shows a better performance than the well-known Bootstrap particle filter. Regarding the high relative errors and low likelihood estimates for the real data, these could be improved as well by an auxiliary particle filter. The particles can be guided even better, leading to higher likelihoods and smaller errors. Hence, we recommend to investigate other particle filters as well, in specific the auxiliary particle filter, to test for an even better approximation of the likelihood value.

The ESS threshold used for the resampling scheme in Sequential Monte Carlo was $N/2$, which is a common one. However, applying another threshold will result in different outcomes and efficiency. If the threshold is increased, the algorithm will resample more often, resulting in more freedom in the algorithm and hence presumably a more accurate and reliable approximation. However, as resampling is costly, the algorithm will be less efficient as well. Decreasing the threshold leads to the exact opposite, so presumably a less accurate estimate though a more efficient algorithm. Investigating the optimal choice of the ESS threshold for this model could lead to promising results for Sequential Monte Carlo.

The synthetic data results were obtained for $N_{\text{part}} = 10000$ particles and $N_{\text{sim}} = 1000$ independent simulations. More particles allows SMC to have a larger choice of particles in the resampling step, and less particles the opposite. Hence, the influence of this parameter when being increased or decreased is interesting. Increasing the number of simulations could decrease the confidence interval as well. For the real data we used $N_\theta = 18000$, $N_{\text{part}} = 10000$ and $N_{\text{sim}} = 100$. As stated earlier in this thesis, the parameter search was not adequate. More time or a more efficient algorithm would allow for larger likelihoods and possibly smaller errors. A suggestion would be to increase the number of parameters tested. The parameter space is of dimension six, thus the space to test over must be humongous to find likelihoods that are not close to zero. That is, as the exact combination of the six parameters that is close to the true setting must be found. In the synthetic data we observed that only a slight deviation of the true parameters caused a drastic decrease in likelihood, hence we suspect that for the real data a slight deviation also causes the likelihood to be substantially lower than the likelihood of the true parameters. For that reason, a lot of parameter combinations need to be tested to find some close to the true parameters. Increasing $N_\theta$ will likely lead to higher likelihood values, and in return a more accurate estimation through maximum likelihood estimation, though the next paragraph discusses more sophisticated methods.

The computations were already fairly costly, and a humongous increase in $N_\theta$ is necessary for adequate results, which seems practically impossible. Therefore, next to increasing the number of parameters to run over to find higher likelihoods, a more thoughtful method of selecting the parameter settings to run over could be tested for efficiency increase. A way
of addressing this problem could be by applying Principal Component Analysis, to reduce the dimensionality of the parameter space. Currently, the method of parameter selection is simply uniformly picking values over the range of values. Next to the Principal Component Analysis, instead of simply uniformly choosing parameter settings each run, one could exploit the likelihood values for a sense of direction of the true parameter values in the next run. If increasing a parameter causes a higher likelihood for instance, one could increase the parameter value based on a certain policy. For instance, based on the increase in estimated likelihood $y$ over a change in parameter value $x$, i.e. the slope $\frac{dy}{dx}$, the next change in parameter value is selected. The steeper the slope, the larger the change in parameter value should be. If the slope is positive, the parameter should be increased and vice versa. This type of more thoughtful methods are in fact applied in optimization algorithms for maximum likelihood estimation, which we will address in the next paragraph.

This thesis focused on the first step towards estimating the parameters of a degradation process, namely the likelihood estimation. Nevertheless, eventually one is interested in the true parameters, as this informs one on the course of the degradation system. Therefore, a recommendation for future research would be to apply the Bootstrap particle filter in combination with an optimization method to obtain the maximum likelihood estimator. An optimization method includes a policy for new parameter selection in each run based on the previous runs, and a stopping criteria of the algorithm, i.e. when the parameter estimation is deemed to be sufficient. There are different types of optimization algorithms, the one mentioned earlier depending on the slope is in the class of gradient-based algorithms. However, for complex likelihoods, such as the one considered in this thesis, genetic algorithms are a more suitable optimization method. They have shown to be eminently efficient in the maximization of complex, multimodal and multivariable functions. The algorithm is inspired by the ideas of natural genetic evolution. It starts with an initial selection of parameters, and repeatedly applies procedures such as mutations, crossovers or reproductions based on a certain policy to continue with new, more promising parameter values (Sharman & McClurkin, 1989). Applying maximum likelihood estimation using particle filtering and a genetic algorithm will likely lead to an efficient, accurate and more reliable estimation of the parameters, as the weights of the particles in the particle filter can be optimized by genetic algorithms. As is apparent, the accurate estimation of the parameters is the ultimate goal of this modeling problem. Accordingly, a future research in the combination of particle filtering with different optimization methods is recommended.

In order to evaluate the performance of the estimation through particle filtering and an optimization method, one would among others like to know the variance of the parameter estimates. Instead of using traditional methods to develop confidence intervals, one could apply a bootstrap method, as the sample size is relatively low and the underlying distribution is complex. Another method to evaluate the performance of the parameter estimates in a future study could be to apply the train-test split method. This method involves taking, for instance, 80 percent of the data to estimate the parameters on, i.e. ’train’ the model, and leaving the remaining data to check the parameters, i.e. ’test’ the model. One uses the estimated parameters to predict the remaining data, these predictions are compared with the remaining 20 percent of the data that was not used to train the model on. The error
A Comparison of CMC Against SMC

between the two sets estimates the performance of the model.

The relative error of the likelihood estimation through Bootstrap remained fairly high with overlapping confidence intervals for the real data case. As mentioned before, through increasing $N_{\text{sim}}$ and checking the data truly fits the model assumptions, the relative error might be decreased. To possibly improve the model fit, a log-normal noise instead of a normal noise could be implemented, as discussed in the Section 1. In addition, assuming the model fits, we mentioned implementing an improved parameter search which obtains higher estimates or eliminating machine error in a different manner could decrease the relative error. Next to that, the study of the ESS threshold was suggested, which could lower the errors as well. If these actions do not decrease the error while the data does fit the model, depending on the criticality of the process forecasts, the Bootstrap method might not be reliable enough, and hence as stated before we recommend the implementation of another particle filter.

Lastly, an interesting study would be to evaluate the performance of Markov Chain Monte Carlo (MCMC). While SMC might be more robust and flexible given complex likelihood functions, MCMC might allow for more efficiency. Applying MCMC can namely improve the estimation of the state of the degradation system.
References


A Comparison of CMC Against SMC

Engineering & System Safety, 144, 334–342. doi: 10.1016/J.RESS.2015.08.007
Appendices

A  Real Degradation Data

<table>
<thead>
<tr>
<th>Batch</th>
<th>0 years</th>
<th>1 year</th>
<th>2 years</th>
<th>3 years</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>105</td>
<td>104</td>
<td>101</td>
<td>98</td>
</tr>
<tr>
<td>2</td>
<td>106</td>
<td>102</td>
<td>99</td>
<td>96</td>
</tr>
<tr>
<td>3</td>
<td>103</td>
<td>101</td>
<td>98</td>
<td>95</td>
</tr>
<tr>
<td>4</td>
<td>105</td>
<td>101</td>
<td>99</td>
<td>95</td>
</tr>
<tr>
<td>5</td>
<td>104</td>
<td>102</td>
<td>100</td>
<td>96</td>
</tr>
<tr>
<td>6</td>
<td>102</td>
<td>100</td>
<td>100</td>
<td>97</td>
</tr>
<tr>
<td>7</td>
<td>104</td>
<td>103</td>
<td>101</td>
<td>97</td>
</tr>
<tr>
<td>8</td>
<td>105</td>
<td>104</td>
<td>101</td>
<td>100</td>
</tr>
<tr>
<td>9</td>
<td>103</td>
<td>101</td>
<td>99</td>
<td>99</td>
</tr>
<tr>
<td>10</td>
<td>103</td>
<td>102</td>
<td>97</td>
<td>96</td>
</tr>
<tr>
<td>11</td>
<td>101</td>
<td>98</td>
<td>93</td>
<td>91</td>
</tr>
<tr>
<td>12</td>
<td>105</td>
<td>102</td>
<td>100</td>
<td>98</td>
</tr>
<tr>
<td>13</td>
<td>105</td>
<td>104</td>
<td>99</td>
<td>95</td>
</tr>
<tr>
<td>14</td>
<td>104</td>
<td>103</td>
<td>97</td>
<td>94</td>
</tr>
<tr>
<td>15</td>
<td>105</td>
<td>103</td>
<td>98</td>
<td>96</td>
</tr>
<tr>
<td>16</td>
<td>103</td>
<td>101</td>
<td>99</td>
<td>96</td>
</tr>
<tr>
<td>17</td>
<td>104</td>
<td>102</td>
<td>101</td>
<td>98</td>
</tr>
<tr>
<td>18</td>
<td>106</td>
<td>104</td>
<td>102</td>
<td>97</td>
</tr>
<tr>
<td>19</td>
<td>105</td>
<td>103</td>
<td>100</td>
<td>99</td>
</tr>
<tr>
<td>20</td>
<td>103</td>
<td>101</td>
<td>99</td>
<td>95</td>
</tr>
<tr>
<td>21</td>
<td>101</td>
<td>101</td>
<td>94</td>
<td>90</td>
</tr>
<tr>
<td>22</td>
<td>102</td>
<td>100</td>
<td>99</td>
<td>96</td>
</tr>
<tr>
<td>23</td>
<td>103</td>
<td>101</td>
<td>99</td>
<td>94</td>
</tr>
<tr>
<td>24</td>
<td>105</td>
<td>104</td>
<td>100</td>
<td>97</td>
</tr>
</tbody>
</table>

Table 11: Drug Potency Degradation Data, percentage of claimed potency

B  Synthetic Data Results Code

B.1  Data Generation

```
function [t, Y] = GpDataGeneration()
rng (1234)

% Initialization parameters
m = 10;
t = [2, 4, 6];
alpha = 2;
eta = 2.5;
```
A Comparison of CMC Against SMC

beta = 0.01;
mu_A = 0.5;
sigma_A = 0.1;
sigma_Z = 0.1;

Y = [ ];
A = [ ];
for comp=1:m
    % for single component
    Y_j = [ ];
    A_j = normrnd(mu_A, sigma_A);
    A = [A; A_j ];
    X_prev = 0 ;
    for i=1:length(t) %for each timestep
        if( i==1)
            alpha_t1 = 0 ;
            alpha_t2 = t(1) ;
        else
            alpha_t1 = t(i-1) ;
            alpha_t2 = t(i) ;
        end
        alpha_tmp = alpha * alpha_t2^eta - alpha * alpha_t1^eta ;
        X_t = gamrnd(alpha_tmp, beta) + X_prev ;
        X_prev = X_t ;
        Z_t = normrnd(0 , sigma_Z) ;
        Y_t = A_j + X_t + Z_t ;
        Y_j = [Y_j, Y_t] ;
    end
    Y = [Y; Y_j];
end

% plot
x = [0,t];
tmp = [A, Y];
figure
for comp=1:m
    hold on
    stairs(x , tmp(comp , : ) )
end
hold off
save('example2.mat' , 't' , 'Y');
end

The next subsections are structured as follows. The code starts with creating the parameter values, and executes the real simulation after.

B.2 CMC - 1 unknown

tic
%%%%%%% INPUT OWN TRAJECTORY HERE %%%%%%%
data = load('C:\Users\20173793\OneDrive - TU Eindhoven\Final ... 
Project\Matlab Code\example2.mat');
t = data.t;
y = data.Y;

% *PARAMETER VALUES*
rng(555)

%lowerbound distributions
min_alpha = 0;

%Upperbound distributions
max_alpha = 10;

%True parameters
true_alpha = 2;
true_eta = 2.5;
true_beta = 0.01;
true_mu = 0.5;
true_sigma_A = 0.1;
true_sigma_Z = 0.1;

steps = [0; 0.001; 0.01; 0.02; 0.05; 0.1; 0.5]; %Percentage of total ... interval length, that amount will differ from true parameter

NumParSet = length(steps);

alpha_vals_cmos_1 = zeros(NumParSet,1);

for i=1:(NumParSet)
    randnum = (2*randi([0 ,1],1,1)-1);
    alpha_vals_cmos_1(i) = true_alpha + ... 
    randnum(i)*steps(i)*(max_alpha-min_alpha);
    if (alpha_vals_cmos_1(i)>max_alpha||alpha_vals_cmos_1(i)<min_alpha)
        alpha_vals_cmos_1(i) = true_alpha - ... 
        1*randnum(i)*steps(i)*(max_alpha-min_alpha);
    end
end

eta = true_eta;
beta = true_beta;
mu_A = true_mu;
sigma_A = true_sigma_A;
sigma_Z = true_sigma_Z;

% *SIMULATION*
NumSim = 1000; %num times compute estimate of expectation (=likelihood)
N = 10000; %num times calculating product, so mean approximates ... 
    expectation (=likelihood=estimate)
sz_comps = size(y,1) ;
sz = length(t) ;

Results_cmos = zeros(NumParSet,6);
AllValues = zeros(N,NumParSet);
for m=1:NumParSet
alpha = alpha.vals_cmc_1(m);

% precalculate alphas for gamma distribution
alpha_array = zeros(sz,1);
for i=1:sz
    if(i==1)
        alpha_array(i) = alpha*(t(1).^eta-0.^eta);
    else
        alpha_array(i) = alpha*(t(i).^eta-t(i-1).^eta);
    end
end

% precalculate
lnGamma = sz.comps *sum(gammaln(alpha_array));
lnBeta = sz.comps *sum(alpha_array .* log(beta));

loglikeli = zeros(NumSim,1);
for k=1:NumSim
    ell = zeros(N, 1);
    for i =1:N
        A = normrnd(mu_A, sigma_A, sz.comps, 1);
        y_tmp = [A, y tmp; , 1:end-1] ;
        Z = [ zeros(sz.comps, 1), normrnd(0, sigma_Z, sz.comps, sz) ] ;
        DeltaZ = Z(:, 2:end) - Z(:, 1:end-1) ;
        DeltaX = DeltaY - DeltaZ;
        ell(i) = -10000;
    end
    if(isempty(find(DeltaX < 0, 1)))
        lnDeltaX = (alpha_array'-1).* log(DeltaX);
        DeltaXDivBeta = (1/beta) *DeltaX ;
        logU = sum( lnDeltaX , 'all' ) - sum(DeltaXDivBeta , 'all' ...
            ) - lnGamma - lnBeta;
        ell(i) = logU;
    end
end
xstar = max(ell);
loglikelihood(k) = -log(N) + xstar + log(sum(exp(ell-xstar))); %Estimate
end
xstar2 = max(loglikelihood);
Results_CMC1(m,1) = -log(NumSim) + xstar2 + ...
        log(sum(exp(loglikelihood-xstar2)));%log mean of estimates
Results_CMC1(m,2) = ... 0.5*(-log(NumSim)+log(sum((exp(loglikelihood)-exp(Results_CMC1(m,1)))^2)));
Results_CMC1(m,3) = mean(exp(loglikelihood));%mean of estimates
Results_CMC1(m,4) = std(exp(loglikelihood));%std of estimates
end
time_CMC1 = toc
Results_CMC1(:,5) = Results_CMC1(:,10)/sqrt(NumSim);... %Standard error
101 Results.CMC1(:,6) = norminv(1-0.05/2)*Results.CMC1(:,11); ...
  %halfwidth CI
102 save('Results_CMC1.mat', 'Results_CMC1', 'time_CMC1', 'steps', ...
  'alpha_vals_cmc1');

B.3 CMC - 2 unknowns

tic
  %%%% INPUT OWN TRAJECTORY HERE %%%%%%
3  data = load('C:\Users\20173793\OneDrive - TU Eindhoven\Final ...
  Project\Matlab Code\example2.mat');
4  t = data.t;
5  y = data.Y;
6
7  % *PARAMETER VALUES*
8  rng(555)
9  %lowerbound distributions
10  min_\eta = 0;
11  min_\mu = 0;
12
13  %Upperbound distributions
14  max_\eta = 5;
15  max_\mu = 1;
16
17  %True parameters
18  true_\alpha = 2;
19  true_\eta = 2.5;
20  true_\beta = 0.01;
21  true_\mu = 0.5;
22  true_\sigma_A = 0.1;
23  true_\sigma_Z = 0.1;
24
25  steps = [0; 0.001; 0.01; 0.02; 0.05; 0.1; 0.5]; %Percentage of total ... 
  interval length, that amount will differ from true parameter
26
27  NumParSet = length(steps);
28
29  eta_vals_cmc2 = zeros(NumParSet,1);
30  mu_A_vals_cmc2 = zeros(NumParSet,1);
31
32  for i=1:(NumParSet)
33    randnum = (2*randi([0 ,1],2,1)-1);
34    eta_vals_cmc2(i) = true_\eta + randnum(i)*steps(i)*(max_\eta-min_\eta);
35    mu_A_vals_cmc2(i) = true_\mu + randnum(2)*steps(i)*(max_\mu-min_\mu);
36    if (eta_vals_cmc2(i)>max_\eta || eta_vals_cmc2(i)<min_\eta)
37      eta_vals_cmc2(i) = true_\eta - ...
38        1*randnum(i)*steps(i)*(max_\eta-min_\eta);
39    end
40    if (mu_A_vals_cmc2(i)>max_\mu || mu_A_vals_cmc2(i)<min_\mu)
41      mu_A_vals_cmc2(i) = true_\mu - ...
42        1*randnum(2)*steps(i)*(max_\mu-min_\mu);
A Comparison of CMC Against SMC

alpha = true_alpha;
beta = true_beta;
sigma_A = true_sigma_A;
sigma_Z = true_sigma_Z;

alpha = true;
beta = true;
sigma_A = true;
sigma_Z = true;

% *SIMULATION*
NumSim = 1000; %num times compute estimate of expectation (=likelihood)
N = 10000; %num times calculating product, so mean approximates ... 
expectation (=likelihood=estimate)
sz_comps = size(y,1);
sz = length(t);

Results_CMC2 = zeros(NumParSet,6);
AllValues = zeros(N,NumParSet);
for m=1:NumParSet
  mu_A = mu_A.vals_cmc_2(m);
  eta = eta.vals_cmc_2(m);

  % precalculate alphas for gamma distribution
  alpha_array = zeros(sz,1);
  for i=1:sz
    if(i==1)
      alpha_array(i) = alpha*(t(1)^eta-0^eta);
    else
      alpha_array(i) = alpha*(t(i)^eta-t(i-1)^eta);
    end
  end

  % precalculate
  lnGamma = sz_comps *sum(gammaln(alpha_array));
  lnBeta = sz_comps *sum(alpha_array .* log(beta));

  loglikeli = zeros(NumSim,1);
  for k=1:NumSim
    ell = zeros(N, 1);
    for i =1:N
      A = normrnd(mu_A, sigma_A, sz_comps, 1);
      y_tmp = [A, y];
      DeltaY = y - y_tmp(:, 1:end-1) ;
      Z = [ zeros(sz_comps, 1), normrnd(0, sigma_Z, sz_comps, sz) ] ;
      DeltaZ = Z(:, 2:end) - Z(:, 1:end-1) ;
      DeltaX = DeltaY - DeltaZ;
      ell(i) = -10000;

      % DeltaX>0 for all
      if(isempty(find(DeltaX < 0, 1)))
        lnDeltaX = (alpha_array'-1).* log(DeltaX);
        DeltaXDivBeta = (1/beta) *DeltaX ;
        logU = sum( lnDeltaX , 'all' ) - sum(DeltaXDivBeta , 'all' ... ) - lnGamma - lnBeta;
        ell(i) = logU;
      end

M.B.C. Buist
A Comparison of CMC Against SMC

```matlab
end
xstar = max(ell);
loglikeli(k) = -log(N) + xstar + log(sum(exp(ell-xstar))); %Estimate
end
xstar2 = max(loglikeli);
Results_CMC2(m,1) = -log(NumSim) + xstar2 + ...
log(sum(exp(loglikeli-xstar2)));
Results_CMC2(m,2) = ...
0.5*(-log(NumSim)+log(sum((exp(loglikeli)-exp(Results_CMC2(m,1))).^2)));
Results_CMC2(m,3) = mean(exp(loglikeli));
Results_CMC2(m,4) = std(exp(loglikeli));
end
time_CMC2 = toc
Results_CMC2(:,5) = Results_CMC2(:,10)/sqrt(NumSim); ... %Standard error
Results_CMC2(:,6) = norminv(1-0.05/2)*Results_CMC2(:,11); ... %halfwidth CI
save('Results_CMC2.mat', 'Results_CMC2', 'time_CMC2', 'steps', ...
'eta_valscmc2', 'mu_A_valscmc2');
```

### B.4 CMC - 6 unknowns

```matlab
tic
%%% INPUT OWN TRAJECTORY HERE %%%%
data = load ("C:\Users\ugmbuist\OneDrive - TU Eindhoven\Final ...
    Project\Matlab Code\example2.mat") ;
data = load ('C:\Users\20173793\OneDrive - TU Eindhoven\Final ...
    Project\Matlab Code\example3.mat');
t = data.t;
y = data.Y;
%
% *PARAMETER VALUES*
rng(555)
%lowerbound distributions
min_alpha = 0;
min_eta = 0;
min_beta = 0;
min_mu = 0;
min_sigma_A = 0;
min_sigma_Z = 0;
%Upperbound distributions
max_alpha = 5;
max_eta = 5;
max_beta = 1;
max_mu = 1;
max_sigma_A = 1;
max_sigma_Z = 1;
```

M.B.C. Buist
A Comparison of CMC Against SMC

26 %True parameters
27 true.alpha = 2;
28 true.eta = 2.5;
29 true.beta = 0.01;
30 true.mu = 0.5;
31 true.sigma_A = 0.1;
32 true.sigma_Z = 0.1;
33
34 steps = [0; 0.001; 0.01; 0.02; 0.05; 0.1; 0.5]; %Percentage of total ... 
interval length, that amount will differ from true parameter
35 NumParSet = length(steps);
36
37 alpha_vals_cmc_6 = zeros(NumParSet,1);
38 eta_vals_cmc_6 = zeros(NumParSet,1);
39 beta_vals_cmc_6 = zeros(NumParSet,1);
40 mu_A_vals_cmc_6 = zeros(NumParSet,1);
41 sigma_A_vals_cmc_6 = zeros(NumParSet,1);
42 sigma_Z_vals_cmc_6 = zeros(NumParSet,1);
43
44 for i=1:(NumParSet)
45    randnum = (2*randi([0 ,1],6,1)-1);
46    alpha_vals_cmc_6(i) = true.alpha + ... 
47        randnum(1)*steps(i)*(max.alpha-min.alpha);
48    eta_vals_cmc_6(i) = true.eta + randnum(2)*steps(i)*(max.eta-min.eta);
49    beta_vals_cmc_6(i) = true.beta + ... 
50        randnum(3)*steps(i)*(max.beta-min.beta);
51    mu_A_vals_cmc_6(i) = true.mu + randnum(4)*steps(i)*(max.mu-min.mu);
52    sigma_A_vals_cmc_6(i) = true.sigma_A + ... 
53        randnum(5)*steps(i)*(max.sigma_A-min.sigma_A);
54    sigma_Z_vals_cmc_6(i) = true.sigma_Z + ... 
55        randnum(6)*steps(i)*(max.sigma_Z-min.sigma_Z);
56    if (alpha_vals_cmc_6(i)>max.alpha||alpha_vals_cmc_6(i)<min.alpha)
57        alpha_vals_cmc_6(i) = true.alpha - ... 
58            1*randnum(1)*steps(i)*(max.alpha-min.alpha);
59    end
60    if (eta_vals_cmc_6(i)>max.eta||eta_vals_cmc_6(i)<min.eta)
61        eta_vals_cmc_6(i) = true.eta - ... 
62            1*randnum(2)*steps(i)*(max.eta-min.eta);
63    end
64    if (beta_vals_cmc_6(i)>max.beta||beta_vals_cmc_6(i)<min.beta)
65        beta_vals_cmc_6(i) = true.beta - ... 
66            1*randnum(3)*steps(i)*(max.beta-min.beta);
67    end
68    if (mu_A_vals_cmc_6(i)>max.mu||mu_A_vals_cmc_6(i)<min.mu)
69        mu_A_vals_cmc_6(i) = true.mu - ... 
70            1*randnum(4)*steps(i)*(max.mu-min.mu);
71    end
72    if ...
73        (sigma_A_vals_cmc_6(i)>max.sigma_A||sigma_A_vals_cmc_6(i)<min.sigma_A)
74            sigma_A_vals_cmc_6(i) = true.sigma_A - ... 
75                1*randnum(5)*steps(i)*(max.sigma_A-min.sigma_A);
76    end
77}
68 if ...
69 (sigma_Z.vals.cmc.6(i)>max.sigma_Z||sigma_Z.vals.cmc.6(i)<min.sigma_Z)
70 sigma_Z.vals.cmc.6(i) = true.sigma_Z - ... 
71 randnum(6)*steps(i)*(max.sigma_Z-min.sigma_Z);
72 end
73 end
74
75 % *SIMULATION*
76 NumSim = 1000; %num times compute estimate of expectation (=likelihood)
77 N = 10000; %num times calculating product, so mean approximates ... 
78 sz_comps = size(y,1) ;
79 sz = length(t) ;
80
81 Results = zeros(NumParSet,6);
82 AllValues = zeros(N,NumParSet);
83 for m=1:NumParSet
84 mu_A = mu_A.vals.cmc.6(m);
85 beta = beta.vals.cmc.6(m);
86 eta = eta.vals.cmc.6(m);
87 alpha = alpha.vals.cmc.6(m);
88 sigma_A = sigma_A.vals.cmc.6(m);
89 sigma_Z = sigma_Z.vals.cmc.6(m);
90 % precalculate alphas for gamma distribution
91 alpha_array = zeros(sz,1) ;
92 for i=1:sz
93 if(i==1)
94 alpha_array(i) = alpha*(t(1)^eta-0^eta);
95 else
96 alpha_array(i) = alpha*(t(i)^eta-t(i-1)^eta);
97 end
98 end
99 % precalculate
100 lnGamma = sz_comps *sum(gammaln(alpha_array));
101 lnBeta = sz_comps *sum(alpha_array .* log(beta));
102
103 loglikeli = zeros(NumSim,1);
104 for k=1:NumSim
105 ell = zeros(N, 1);
106 for i =1:N
107 A = normrnd(mu_A, sigma_A, sz_comps, 1);
108 y_tmp = [A, y];
109 DeltaY = y - y_tmp(:, 1:end-1) ;
110 Z = [ zeros(sz_comps, 1), normrnd(0, sigma_Z, sz_comps, sz) ] ;
111 DeltaZ = Z(:, 2:end) - Z(:, 1:end-1) ;
112 DeltaX = DeltaY - DeltaZ;
113 ell(i) = -10000;
114 % DeltaX>0 for all
115 if(isempty(find(DeltaX < 0, 1)))
116 InDeltaX = (alpha_array'-1).* log(DeltaX);
117 DeltaXDivBeta = (1/beta) *DeltaX ;
118
119 %
A Comparison of CMC Against SMC

\[
\log U = \text{sum( lnDeltaX , 'all' )} - \text{sum(DeltaXDivBeta , 'all' ... )} - \ln Gamma - \ln Beta;
\]

\[
e(i) = \log U;
\]

\[
xstar = \text{max(ell)};
\]

\[
\loglikeli(k) = -\log(N) + xstar + \log(\text{sum(exp(ell-xstar)})); \quad \text{%Estimate}
\]

\[
xstar2 = \text{max(loglikeli)};
\]

\[
\text{Results}(m,1) = -\log(\text{NumSim}) + xstar2 + \log(\text{sum(exp(loglikeli-xstar2)}));
\]

\[
\text{Results}(m,2) = -0.5*(-\log(\text{NumSim})+\log(\text{sum((exp(loglikeli)-exp(Results(m,1)))).^2)})
\]

\[
\text{Results}(m,3) = \text{mean(exp(loglikeli))};
\]

\[
\text{Results}(m,4) = \text{std(exp(loglikeli))};
\]

\[
time_CMC = \text{toc}
\]

\[
\text{Results}(:,5) = \text{Results}(:,10)/\sqrt{\text{NumSim}}; \quad \text{... Standard error}
\]

\[
\text{Results}(:,6) = \text{norminv}(1-0.05/2)*\text{Results}(:,11); \quad \text{... halfwidth CI}
\]

\[
\text{save('Results.mat', 'Results', 'time_CMC', 'steps', ... 'alpha_valscmc6', 'eta_valscmc6', 'beta_valscmc6', ... 'muA_valscmc6', 'sigmaA_valscmc6', 'sigmaZ_valscmc6');}
\]

B.5 Bootstrap - 1 unknown

\[
tic \quad \% \text{*PARAMETERS*}
\]

\[
\text{rng}(555)
\]

\[
\text{%lowerbound distributions}
\]

\[
\text{min_alpha} = 0;
\]

\[
\text{%Upperbound distributions}
\]

\[
\text{max_alpha} = 10;
\]

\[
\text{%True parameters}
\]

\[
\text{true_alpha} = 2;
\]

\[
\text{true_eta} = 2.5;
\]

\[
\text{true_beta} = 0.01;
\]

\[
\text{true_mu} = 0.5;
\]

\[
\text{true_sigma_A} = 0.1;
\]

\[
\text{true_sigma_Z} = 0.1;
\]

\[
\text{steps} = [0; 0.001; 0.01; 0.02; 0.05; 0.1; 0.5]; \quad \text{%Percentage of total ... interval length, that amount will differ from true parameter}
\]

\[
\text{NumParSet} = \text{length(steps)};
\]

\[
\text{alpha_vals_cmc1} = \text{zeros(NumParSet,1)};
\]
A Comparison of CMC Against SMC

for i=1:(NumParSet)
    randnum = (2*randi([0 ,1],1,1)-1);
    alpha_vals_cmc_1(i) = true_alpha + ...
    randnum(1)*steps(i)*(max_alpha-min_alpha);
    if (alpha_vals_cmc_1(i)>max_alpha||alpha_vals_cmc_1(i)<min_alpha)
        alpha_vals_cmc_1(i) = true_alpha - ...
        1*randnum(1)*steps(i)*(max_alpha-min_alpha);
end
eta = true_eta;
beta = true_beta;
mu_A = true_mu;
sigma_A = true_sigma_A;
sigma_Z = true_sigma_Z;

% *SIMULATION*
NumSim = 1000;%num times compute estimate of expectation (=likelihood)
N = 10000; %num times calculating product, so mean approximates ...
    expectation (=likelihood=estimate)
ESS_min = N/2;

Results Boot1 = zeros(NumParSet,6);
for m=1:NumParSet
    alpha = alpha_vals_cmc_1(m);
    theta = [alpha,eta,beta,mu_A,sigma_A,sigma_Z]';
    loglikeli = zeros(NumSim,1);
    for i=1:NumSim
        model = ReliabilityModelBootstrap(theta);
        smc = BootstrapSMC(model,N,ESS_min);
        smc.RunSMC();
        [RE, logLt] = smc.GetLogLiklihood();
        loglikeli(i) = logLt;
    end
    xstar2 = max(loglikeli);
    Results_Boot1(m,1) = -log(NumSim) + xstar2 + ...
        log(sum(exp(loglikeli-xstar2)));
    Results_Boot1(m,2) = ...
        0.5*(-log(NumSim)+log(sum((exp(loglikeli)-exp(Results_Boot1(m,1))).^2)));
    Results_Boot1(m,3) = mean(exp(loglikeli));
    Results_Boot1(m,4) = std(exp(loglikeli));
end
time_Boot1 = toc
Results_Boot1(:,5) = Results_Boot1(:,10)/sqrt(NumSim); ... %Standard error
Results_Boot1(:,6) = norminv(1-0.05/2)*Results_Boot1(:,11); ... %halfwidth CI
save('Results_Boot1.mat', 'Results_Boot1', 'time_Boot1', 'steps', ...
    'alpha_vals_cmc_1');
A Comparison of CMC Against SMC

B.6 Bootstrap - 2 unknowns

cic
% PARAMETERS
rng(555)
% lower bound distributions
min_eta = 0;
min_mu = 0;

% upper bound distributions
max_eta = 5;
max_mu = 1;

% True parameters
true_alpha = 2;
true_eta = 2.5;
true_beta = 0.01;
true_mu = 0.5;
true_sigma_A = 0.1;
true_sigma_Z = 0.1;

steps = [0; 0.001; 0.01; 0.02; 0.05; 0.1; 0.5]; % Percentage of total ... interval length, that amount will differ from true parameter

NumParSet = length(steps);

eta_vals_cmc_2 = zeros(NumParSet,1);
mu_A_vals_cmc_2 = zeros(NumParSet,1);

for i=1:(NumParSet)
    randnum = (2*randi([0 ,1],2,1)-1);
    eta_vals_cmc_2(i) = true_eta + randnum(1)*steps(i)*(max_eta-min_eta);
    mu_A_vals_cmc_2(i) = true_mu + randnum(2)*steps(i)*(max_mu-min_mu);
    if (eta_vals_cmc_2(i)>max_eta || eta_vals_cmc_2(i)<min_eta)
        eta_vals_cmc_2(i) = true_eta - ... 1*randnum(1)*steps(i)*(max_eta-min_eta);
    end
    if (mu_A_vals_cmc_2(i)>max_mu || mu_A_vals_cmc_2(i)<min_mu)
        mu_A_vals_cmc_2(i) = true_mu - ... 1*randnum(2)*steps(i)*(max_mu-min_mu);
    end
end
alpha = true_alpha;
beta = true_beta;
sigma_A = true_sigma_A;
sigma_Z = true_sigma_Z;

% SIMULATION
NumSim = 1000; % num times compute estimate of expectation (= likelihood)
N = 10000; % num times calculating product, so mean approximates ... expectation (= likelihood = estimate)
ESS_min = N/2;
A Comparison of CMC Against SMC

```matlab
48 Results_Boot2 = zeros(NumParSet,6);
49 for m=1:NumParSet
50    mu_A = mu_A_vals_cmc_2(m);
51    eta = eta_vals_cmc_2(m);
52    theta = [alpha,eta,beta,mu_A,sigma_A,sigma_Z]';
53    loglikeli = zeros(NumSim,1);
54    for i=1:NumSim
55        model = ReliabilityModelBootstrap(theta);
56        smc = BootstrapSMC(model,N,ESS_min);
57        smc.RunSMC();
58        [RE, logL_t] = smc.GetLogLiklihood();
59        loglikeli(i) = logL_t;
60    end
61    xstar2 = max(loglikeli);
62    Results_Boot2(m,1) = -log(NumSim) + xstar2 + ...
63        log(sum(exp(loglikeli-xstar2))));
64    Results_Boot2(m,2) = ...
65        0.5*(-log(NumSim)+log(sum((exp(loglikeli)-exp(Results_Boot2(m,1))).^2))));
66    Results_Boot2(m,3) = mean(exp(loglikeli));
67    Results_Boot2(m,4) = std(exp(loglikeli));
68 end
69 time_Boot2 = toc
70 Results_Boot2(:,5) = Results_Boot2(:,10)/sqrt(NumSim); ... %Standard error
71 Results_Boot2(:,6) = norminv(1-0.05/2)*Results_Boot2(:,11); ... %halfwidth CI
72 save('Results_Boot2.mat', 'Results_Boot2', 'time_Boot2', 'steps', ...
73     'eta_vals_cmc_2', 'mu_A_vals_cmc_2');
```

B.7 Bootstrap - 6 unknowns

```matlab
tic
% *PARAMETERS*
rng(555)

%lowerbound distributions
min_alpha = 0;
min_eta = 0;
min_beta = 0;
min_mu = 0;
min_sigma_A = 0;
min_sigma_Z = 0;

%Upperbound distributions
max_alpha = 5;
max_eta = 5;
max_beta = 1;
max_mu = 1;
max_sigma_A = 1;
max_sigma_Z = 1;
```
%True parameters
true.alpha = 2;
true.eta = 2.5;
true.beta = 0.01;
true.mu = 0.5;
true.sigma_A = 0.1;
true.sigma_Z = 0.1;

steps = [0; 0.001; 0.01; 0.02; 0.05; 0.1; 0.5]; %Percentage of total ... interval length, that amount will differ from true parameter

NumParSet = length(steps);

alpha_vals_cmos_6 = zeros(NumParSet,1);
eta_vals_cmos_6 = zeros(NumParSet,1);
beta_vals_cmos_6 = zeros(NumParSet,1);
mu_A_vals_cmos_6 = zeros(NumParSet,1);
sigma_A_vals_cmos_6 = zeros(NumParSet,1);
sigma_Z_vals_cmos_6 = zeros(NumParSet,1);

for i=1:(NumParSet)
    randnum = (2*randi([0,1],6,1)-1);
    alpha_vals_cmos_6(i) = true.alpha + ... 
        randnum(1)*steps(i)*(max.alpha-min.alpha);
    eta_vals_cmos_6(i) = true.eta + randnum(2)*steps(i)*(max.eta-min.eta);
    beta_vals_cmos_6(i) = true.beta + ... 
        randnum(3)*steps(i)*(max.beta-min.beta);
    mu_A_vals_cmos_6(i) = true.mu + randnum(4)*steps(i)*(max.mu-min.mu);
    sigma_A_vals_cmos_6(i) = true.sigma_A + ... 
        randnum(5)*steps(i)*(max.sigma_A-min.sigma_A);
    sigma_Z_vals_cmos_6(i) = true.sigma_Z + ... 
        randnum(6)*steps(i)*(max.sigma_Z-min.sigma_Z);
    if (alpha_vals_cmos_6(i)>max.alpha || alpha_vals_cmos_6(i)<min.alpha)
        alpha_vals_cmos_6(i) = true.alpha - ... 
            1*randnum(1)*steps(i)*(max.alpha-min.alpha);
    end
    if (eta_vals_cmos_6(i)>max.eta || eta_vals_cmos_6(i)<min.eta)
        eta_vals_cmos_6(i) = true.eta - ... 
            1*randnum(2)*steps(i)*(max.eta-min.eta);
    end
    if (beta_vals_cmos_6(i)>max.beta || beta_vals_cmos_6(i)<min.beta)
        beta_vals_cmos_6(i) = true.beta - ... 
            1*randnum(3)*steps(i)*(max.beta-min.beta);
    end
    if (mu_A_vals_cmos_6(i)>max.mu || mu_A_vals_cmos_6(i)<min.mu)
        mu_A_vals_cmos_6(i) = true.mu - ... 
            1*randnum(4)*steps(i)*(max.mu-min.mu);
    end
    if ...
        (sigma_A_vals_cmos_6(i)>max.sigma_A || sigma_A_vals_cmos_6(i)<min.sigma_A)
        sigma_A_vals_cmos_6(i) = true.sigma_A - ... 
            1*randnum(5)*steps(i)*(max.sigma_A-min.sigma_A);
    end
end
A Comparison of CMC Against SMC

if ...
    (sigma_Z.vals_cmc_6(i)>max.sigma_Z || sigma_Z.vals_cmc_6(i)<min.sigma_Z)
    sigma_Z.vals_cmc_6(i) = true.sigma_Z - ...
    1*randnum(6)*steps(i)*(max.sigma_Z-min.sigma_Z);
end
end

% *SIMULATION*
NumSim = 1000; % num times compute estimate of expectation (=likelihood)
N = 10000; % num times calculating product, so mean approximates ...
    expectation (=likelihood=estimate)
ESS_min = N/2;

Results_Boot = zeros(NumParSet,6);
for m=1:NumParSet
    mu_A = mu_A.vals_cmc_6(m);
    beta = beta.vals_cmc_6(m);
    eta = eta.vals_cmc_6(m);
    alpha = alpha.vals_cmc_6(m);
    sigma_A = sigma_A.vals_cmc_6(m);
    sigma_Z = sigma_Z.vals_cmc_6(m);
    theta = [alpha,eta,beta,mu_A,sigma_A,sigma_Z]';
    loglikeli = zeros(NumSim,1);
    for i=1:NumSim
        model = ReliabilityModelBootstrap(theta);
        smc = BootstrapSMC(model,N,ESS_min);
        smc.RunSMC();
        [RE, logL_t] = smc.GetLogLiklihood();
        loglikeli(i) = logL_t;
    end
    xstar2 = max(loglikeli);
    Results_Boot(m,1) = -log(NumSim) + xstar2 + ...
        log(sum(exp(loglikeli-xstar2)));
    Results_Boot(m,2) = ...
        0.5*(-log(NumSim)+log(sum((exp(loglikeli)-exp(Results_Boot(m,1))).^2)));
    Results_Boot(m,3) = mean(exp(loglikeli));
    Results_Boot(m,4) = std(exp(loglikeli));
end
time_Boot = toc
Results_Boot(:,5) = Results_Boot(:,10)/sqrt(NumSim); ...
    % Standard error
Results_Boot(:,6) = norminv(1-0.05/2)*Results_Boot(:,11); ...
    % halfwidth CI
save('Results_Boot.mat', 'Results_Boot', 'time_Boot', 'steps', ...
    'alpha.vals_cmc_6', 'eta.vals_cmc_6', 'beta.vals_cmc_6', ...
    'mu_A.vals_cmc_6', 'sigma_A.vals_cmc_6', 'sigma_Z.vals_cmc_6');

B.8 Figures
data CMC1 = load ('C:\Users\20173793\OneDrive - TU Eindhoven\Final ...\Matlab Code\Results_CMC1.mat');
data CMC2 = load ('C:\Users\20173793\OneDrive - TU Eindhoven\Final ...\Matlab Code\Results_CMC2.mat');
data CMC6 = load ('C:\Users\20173793\OneDrive - TU Eindhoven\Final ...\Matlab Code\Results.mat');
data BOOT1 = load ('C:\Users\20173793\OneDrive - TU Eindhoven\Final ...\Matlab Code\smc\Results_Boot1.mat');
data BOOT2 = load ('C:\Users\20173793\OneDrive - TU Eindhoven\Final ...\Matlab Code\smc\Results_Boot2.mat');
data BOOT6 = load ('C:\Users\20173793\OneDrive - TU Eindhoven\Final ...\Matlab Code\smc\Results_Boot.mat');

Results_CMC1 = data_CMC1.Results_CMC1;
Results_CMC2 = data_CMC2.Results_CMC2;
Results = data_CMC6.Results;
steps = data_CMC6.steps;
alpha_valscmc1 = data_CMC6.alpha_vals_cmc1;
etasmc2 = data_CMC6.eta_vals_cmc2;
munmc2 = data_CMC6.mu_A_vals_cmc2;
alpha_vals_cmc6 = data_CMC6.alpha_vals_cmc6;
etasmc6 = data_CMC6.eta_vals_cmc6;
betasmc6 = data_CMC6.beta_vals_cmc6;
munmc6 = data_CMC6.mu_A_vals_cmc6;
sigmamc6 = data_CMC6.sigma_A_vals_cmc6;
sigmazmc6 = data_CMC6.sigma_Z_vals_cmc6;

Results_Boot1 = data_BOOT1.Results_Boot1;
Results_Boot2 = data_BOOT2.Results_Boot2;
Results_Boot = data_BOOT6.Results_Boot;

NumParSet = size(Results_Boot,1);

%% Figures

%% 1 unknown
x = [1:NumParSet; 1:NumParSet]';
h = bar(x, [Results_CMC1(:,3),Results_Boot1(:,3)]);
xCnt = h(1).XData.' + [h.XOffset];

figure
hold on
e1 = errorbar(xCnt(:,1), Results_CMC1(:,3), Results_CMC1(:,6), '*');
set(e1,'Color', 'red');
e2 = errorbar(xCnt(:,2), Results_Boot1(:,3), Results_Boot1(:,6), '*');
set(e2,'Color', 'blue');
ylim([-3000 8000])
legend('CMC','Bootstrap','Location','southeast')
labelArray = [{'r_c' '\alpha'}; strcat({''}, ...
num2str(steps(:),'%.3f')); strcat({''}, ...
num2str(alpha_vals_cmc1(:,),'%.3f'))]
labelArray = transpose(labelArray);
tickLabels = strtrim(sprintf('%s\newline%s', labelArray{:}));
A Comparison of CMC Against SMC

```matlab
ax = gca();
ax.XTick = 0:NumParSet;
ax.XLim = [0,NumParSet+1];
ax.XTickLabel = tickLabels;
%xlabel({'Ratio of Change', '	heta'})
ylabel('Estimated Likelihood Value')
[t,s] = title('Likelihood Comparison of CMC and Bootstrap for 1 Unknown');
set(gca,'fontsize',22);
hold off

%% 2 unknowns
x = [1:NumParSet; 1:NumParSet];
h = bar(x, [Results_CMC2(:,3),Results_Boot2(:,3)]);
xCnt = h(1).XData.' + [h.XOffset];
figure
hold on
e1 = errorbar(xCnt(:,1), Results_CMC2(:,3), Results_CMC2(:,6), '*');
set(e1,'Color', 'red');
e2 = errorbar(xCnt(:,2), Results_Boot2(:,3), Results_Boot2(:,6), '*');
set(e2,'Color', 'blue');
ylim([-2000 5000])
legend('CMC','Bootstrap','Location','southeast')
labelArray = [{'r,c' ' \eta' ' \mu_A'}; strcat({' ' }, ...
num2str(steps(:,),'%.3f')), strcat({' ' }, ...
num2str(eta_vals_cmc2(:,),'%.3f')),strcat({' ' }, ...
num2str(mu_A_vals_cmc2(:,),'%.3f'))];
labelArray = transpos
tickLabels = strtrim(sprintf('%s
\newline%s
\newline%s
\n', labelArray{1}));
ax = gca();
ax.XTick = 0:NumParSet;
ax.XLim = [0,NumParSet+1];
ax.XTickLabel = tickLabels;
%xlabel({'Ratio of Change', '	heta'})
ylabel('Estimated Likelihood Value')
[t,s] = title('Likelihood Comparison of CMC and Bootstrap for 2 Unknowns');
set(gca,'fontsize',22);
hold off

%% 6 unknowns
x = [1:NumParSet; 1:NumParSet];
h = bar(x, [Results(:,3),Results_Boot(:,3)]);
xCnt = h(1).XData.' + [h.XOffset];
figure
hold on
e1 = errorbar(xCnt(:,1), Results(:,3), Results(:,6), '*');
set(e1,'Color', 'red');
e2 = errorbar(xCnt(:,2), Results_Boot(:,3), Results_Boot(:,6), '*');
set(e2,'Color', 'blue');
ylim([-200 600])
legend('CMC','Bootstrap','Location','southeast')
```

M.B.C. Buist
A Comparison of CMC Against SMC

```matlab
labelArray = [{'\textsubscript{r-c}' ' \textsubscript{alpha}' ' \textsubscript{beta}' ' \textsubscript{mu}\_A' ' \textsubscript{sigma}\_A'} ...
' \textsubscript{sigma}\_Z'}; strcat( {' ' }, num2str(steps(:,'%3f'))), strcat( {' ' }, ...
num2str(alpha\_vals\_cmc\_6(:,'%3f'))), strcat( {' ' }, ...
num2str(beta\_vals\_cmc\_6(:,'%3f'))), strcat( {' ' }, ...
num2str(mu\_A\_vals\_cmc\_6(:,'%3f'))), strcat( {' ' }, ...
num2str(sigma\_A\_vals\_cmc\_6(:,'%3f'))), strcat( {' ' }, ...
num2str(sigma\_Z\_vals\_cmc\_6(:,'%3f'))));
labelArray = transpose(labelArray);
tickLabels = ...
strtrim(sprintf('%s\n\newline%s\n\newline%s\n\newline%s\n\newline%s\n\newline%s\n\newline%s
\newline%s', ...
labelArray{:}));
ax = gca();
ax.XTick = 0:NumParSet;
ax.XLim = [0,NumParSet+1];
ax.XTickLabel = tickLabels;
xlabel('Ratio of Change', '\theta')
ylabel('Estimated Likelihood Value')
[t,s] = title('Likelihood Comparison of CMC and Bootstrap for 6 ...
Unknowns, Zoomed');
set(gca,'fontsize',22);
hold off

C Real Data Results Code

C.1 Data Transformation

```matlab
originaldata = [1 105 104 101 98; 2 106 102 99 96; 3 103 101 98 95;4 ...
105 101 99 95;
2 5 104 102 100 96;
3 6 102 100 100 97;
4 7 104 103 101 97;
5 8 105 104 101 100;
6 9 103 101 99 99;
7 10 103 102 97 96;
8 11 101 98 93 91;
9 12 105 102 100 98;
10 13 105 104 99 95;
11 14 104 103 97 94;
12 15 105 103 98 96;
13 16 103 101 99 96;
14 17 104 102 101 98;
15 18 106 104 102 97;
16 19 105 103 100 99;
17 20 103 101 99 95;
18 21 101 101 94 90;
19 22 102 100 99 96;
20 23 103 101 99 94;
21 24 105 104 100 97];
t = [ 1 , 2 , 3 ] ;
m = size(originaldata, 1 );
```
n = size(originaldata, 2) - 1;

Y = zeros(m, n);
Y(:, 1) = 105 - originaldata(:, 2);
Y(:, 2) = Y(:, 1) + originaldata(:, 2) - originaldata(:, 3);
Y(:, 3) = Y(:, 2) + originaldata(:, 3) - originaldata(:, 4);
Y(:, 4) = Y(:, 3) + originaldata(:, 4) - originaldata(:, 5);

% plot
x = [0, t];
tmp = Y;
for comp = 1:m
    hold on
    stairs(x, tmp(comp, :))
end
hold off
Y = Y(:, 2:4);
save('drugexampleV3.mat', 't', 'Y');

C.2 CMC and Bootstrap Result Generation

rng(666)

% lowerbound distributions
min_alpha = 0;
min_eta = 0;
min_beta = 0;
min_mu = 0;
min_sigma_A = 0;
min_sigma_Z = 0;

% upperbound distributions
max_alpha = 50;
max_eta = 5;
max_beta = 1;
max_mu = 5;
max_sigma_A = 1;
max_sigma_Z = 5;

NumParSet = 18000;
alpha_vals_real = min_alpha + (max_alpha - min_alpha)*rand(NumParSet, 1);
eta_vals_real = min_eta + (max_eta - min_eta)*rand(NumParSet, 1);
beta_vals_real = min_beta + (max_beta - min_beta)*rand(NumParSet, 1);
mu_A_vals_real = min_mu + (max_mu - min_mu)*rand(NumParSet, 1);
sigma_A_vals_real = min_sigma_A + ... 
    (max_sigma_A - min_sigma_A)*rand(NumParSet, 1);
sigma_Z_vals_real = min_sigma_Z + ... 
    (max_sigma_Z - min_sigma_Z)*rand(NumParSet, 1);

% *SIMULATION*
NumSim = 100;% num times compute estimate of expectation (=likelihood)
N = 10000; % num times calculating product, so mean approximates ... 
    expectation (=likelihood=estimate) (and num of particles)
ESS_{\text{min}} = N/2;

% *SMC*
tic

Results_RealBoot = zeros(NumParSet,6);
for m=1:NumParSet
    mu_A = mu_A_vals_real(m);
    beta = beta_vals_real(m);
    eta = eta_vals_real(m);
    alpha = alpha_vals_real(m);
    sigma_A = sigma_A_vals_real(m);
    sigma_Z = sigma_Z_vals_real(m);
    theta = [alpha,eta,beta,mu_A,sigma_A,sigma_Z]';
    loglikeli = zeros(NumSim,1);
    for i=1:NumSim
        model = ReliabilityModelBootstrap(theta);
        smc = BootstrapSMC(model,N,ESS_{\text{min}});
        smc.RunSMC();
        [RE, logL_t] = smc.GetLogLiklihood();
        loglikeli(i) = logL_t;
    end
    xstar2 = max(loglikeli);
    Results_RealBoot(m,1) = -log(NumSim) + xstar2 + ... 
                         log(sum(exp(loglikeli-xstar2)));
    Results_RealBoot(m,2) = ... 
                        0.5*(-log(NumSim)+log(sum((exp(loglikeli)-exp(Results_RealBoot(m,1)))).^2));
    Results_RealBoot(m,3) = mean(exp(loglikeli));
    Results_RealBoot(m,4) = std(exp(loglikeli));
%INBETWEEN RESULTS
    if(mod(m,1000)==0)
        m
        Inbetween_time_RealBoot = toc;
        save('Inbetween_Results_RealBoot_test.mat', ...
             'Results_RealBoot', 'Inbetween_time_RealBoot', 'm', ...
             'NumSim', 'N');
    end
end
time_RealBoot = toc;
Results_RealBoot(:,5) = Results_RealBoot(:,10)/sqrt(NumSim); ... 
% Standard error
Results_RealBoot(:,6) = norminv(1-0.05/2)*Results_RealBoot(:,11); ... 
% halfwidth CI

Results_RealBoot = [Results_RealBoot, alpha_vals_real, eta_vals_real, ... 
                    beta_vals_real, mu_A_vals_real, sigma_A_vals_real, sigma_Z_vals_real];
save('Results_RealBoot.mat', 'Results_RealBoot', 'time_RealBoot', ...
     'NumParSet', 'NumSim', 'N');

% *CMC*
tic
data = load ('C:\Users\20173793\OneDrive - TU Eindhoven\Final ... 
Project\Matlab Code\drugexampleV3.mat');
t = data.t;
y = data.Y;

% *SIMULATION*
sz_comps = size(y,1);
sz = length(t);

Results_RealCMC = zeros(NumParSet,6); % Estimate, STD, RE, SE,
for m=1:NumParSet
    mu_A = mu_A_vals_real(m);
    beta = beta_vals_real(m);
    eta = eta_vals_real(m);
    alpha = alpha_vals_real(m);
    sigma_A = sigma_A_vals_real(m);
    sigma_Z = sigma_Z_vals_real(m);

    % precalculate alphas for gamma distribution
    alpha_array = zeros(sz,1);
    for i=1:sz
        if(i==1)
            alpha_array(i) = alpha*(t(1)^eta-0^eta);
        else
            alpha_array(i) = alpha*(t(i)^eta-t(i-1)^eta);
        end
    end

    % precalculate
    lnGamma = sz_comps *sum(gammaln(alpha_array));
    lnBeta = sz_comps *sum(alpha_array .* log(beta));

    loglikeli = zeros(NumSim,1);
    for k=1:NumSim
        ell = zeros(N, 1);
        for i =1:N
            A = normrnd(mu_A, sigma_A, sz_comps, 1);
            y_tmp = [A, y];
            DeltaY = y - y_tmp(:, 1:end-1);
            Z = [ zeros(sz_comps, 1), normrnd(0, sigma_Z, sz_comps, sz) ];
            DeltaZ = Z(:, 2:end) - Z(:, 1:end-1);
            DeltaX = DeltaY - DeltaZ;
            ell(i) = -10000;
        end
        if isempty(find(DeltaX < 0, 1))
            lnDeltaX = (alpha_array'-1).* log(DeltaX);
            DeltaXDivBeta = (1/beta) *DeltaX;
            logU = sum( lnDeltaX , 'all' ) - sum(DeltaXDivBeta , 'all' ... 
            ) - lnGamma - lnBeta;
            ell(i) = logU;
        end
    end
end
xstar = max(ell);

loglikeli(k) = -log(N) + xstar + log(sum(exp(ell-xstar))); %Estimate

end

xstar2 = max(loglikeli);

Results_RealCMC(m,1) = -log(NumSim) + xstar2 + ...
  log (sum (exp (loglikeli-xstar2)));

Results_RealCMC(m,2) = ...
  0.5*(-log(NumSim)+log(sum((exp(loglikeli)-exp(Results_RealCMC(m,1))).^2)));

Results_RealCMC(m,3) = mean(exp(loglikeli));

Results_RealCMC(m,4) = std(exp(loglikeli));

%INBETWEEN RESULTS
if(mod(m,1000)==0)
  m
  Inbetween_time_RealCMC = toc;
  save('Inbetween_Results_RealCMC_test.mat', 'Results_RealCMC', ...
     'Inbetween_time_RealCMC', 'm', 'NumSim', 'N');
end

time_RealCMC = toc;

Results_RealCMC(:,5) = Results_RealCMC(:,10)/sqrt(NumSim); ... %Standard error

Results_RealCMC(:,6) = norminv(1-0.05/2)*Results_RealCMC(:,11); ... %halfwidth CI

Results_RealCMC = [Results_RealCMC, alpha_vals_real, eta_vals_real, ...
  beta_vals_real, muA_vals_real, sigmaA_vals_real, sigma2_vals_real];

save('Results_RealCMC.mat', 'Results_RealCMC', 'time_RealCMC', ...
'NumParSet', 'NumSim', 'N');

C.3 Figures

dataBoot = load ("C:\Users\20173793\OneDrive - TU Eindhoven\Final ...
  Project\Matlab Code\smc\Results_RealBoot_test.mat");
dataCMC = load ("C:\Users\20173793\OneDrive - TU Eindhoven\Final ...
  Project\Matlab Code\smc\Results_RealCMC_test.mat");

Results_RealBoot = dataBoot.Results_RealBoot;
time_RealBoot = dataBoot.time_RealBoot;

Results_RealCMC = dataCMC.Results_RealCMC;
time_RealCMC = dataCMC.time_RealCMC;

NumSim = dataCMC.NumSim;
N = dataCMC.N;

% REAL DATA BOOTSTRAP 10 MAX LIKELIHOOD
Sorted_Results_RealBoot = sortrows(Results_RealBoot, -9);

maxres = 10;
x = [1:maxres; 1:maxres]';
h = bar(x, [Results_RealBoot(1:maxres,1)]);
xCnt = h(1).XData.' + [h.XOffset];
A Comparison of CMC Against SMC

```matlab
% EMPIRICAL DISTRIBUTIONS
figure
hold on
e1 = errorbar(xCnt(:,1), Sorted_Results_RealBoot(1:maxres,9), ...
    Sorted_Results_RealBoot(1:maxres,12), '*
set(e1,'Color', 'red');
labelArray = {'alpha' '\eta' '\beta' '\mu_A' '\sigma_A' '\sigma_Z'}; ...
strcat({' '}, ...
    num2str(Sorted_Results_RealBoot(1:maxres,16),'%.3f')),strcat({' '}, ...
    num2str(Sorted_Results_RealBoot(1:maxres,17),'%.3f')),strcat({' '}, ...
    num2str(Sorted_Results_RealBoot(1:maxres,18),'%.3f')),strcat({' '}, ...
    num2str(Sorted_Results_RealBoot(1:maxres,19),'%.3f')),strcat({' '}, ...
    num2str(Sorted_Results_RealBoot(1:maxres,20),'%.3f')),strcat({' '}, ...
    num2str(Sorted_Results_RealBoot(1:maxres,21),'%.3f')));
labelArray = transpose(labelArray);
tickLabels = ...
    strtrim(sprintf('%s
\newline%s
\newline%s
\newline%s ...
\newline%s
', labelArray{:}));
ax = gca();
ax.XTick = 0:maxres;
ax.XLim = [0,maxres+1];
ax.XTickLabel = tickLabels;
ylabel('Estimated Likelihood Value');
[t,s] = title('10 Highest Mean Likelihood Values for Bootstrap');
hold off

%% REAL DATA CMC 10 MAX LIKELIHOOD

Sorted_Results_RealCMC = sortrows(Results_RealCMC, -9);

maxres = 10;
x = [1:maxres; 1:maxres];
h = bar(x, [Results_RealCMC(1:maxres,1)]);
xCnt = h(1).XData.' + [h.XOffset];
figure
hold on
e1 = errorbar(xCnt(:,1), Sorted_Results_RealCMC(1:maxres,9), ...
    Sorted_Results_RealCMC(1:maxres,12), '*
set(e1,'Color', 'red');
labelArray = {'alpha' '\eta' '\beta' '\mu_A' '\sigma_A' '\sigma_Z'}; ...
strcat({' '}, ...
    num2str(Sorted_Results_RealCMC(1:maxres,16),'%.3f')),strcat({' '}, ...
    num2str(Sorted_Results_RealCMC(1:maxres,17),'%.3f')),strcat({' '}, ...
    num2str(Sorted_Results_RealCMC(1:maxres,18),'%.3f')),strcat({' '}, ...
    num2str(Sorted_Results_RealCMC(1:maxres,19),'%.3f')),strcat({' '}, ...
    num2str(Sorted_Results_RealCMC(1:maxres,20),'%.3f')),strcat({' '}, ...
    num2str(Sorted_Results_RealCMC(1:maxres,21),'%.3f')));
labelArray = transpose(labelArray);
tickLabels = ...
    strtrim(sprintf('%s
\newline%s
\newline%s
\newline%s ...
\newline%s
', labelArray{:}));
```

M.B.C. Buist

79
A Comparison of CMC Against SMC

:\\newline$n'$, labelArray{:})

ax = gca();
ax.XTick = 0:maxres;
ax.XLim = [0,maxres+1];
ax.XTickLabel = tickLabels);
%xlabel({'Ratio of Change', '\theta'})
ylabel('Estimated Likelihood Value')
[t,s] = title('10 Highest Mean Likelihood Values for CMC');
set(gca,'fontsize',22);
hold off

\%% Relative errors

RE_SortedCMC = ...
Sorted_Results_RealCMC(1:maxres,10)./(Sorted_Results_RealCMC(1:maxres,9)*sqrt(NumSim));
RE_RealCMC = Results_RealCMC(:,10)./(Results_RealCMC(:,9)*sqrt(NumSim));
sum( isnan(RE_RealCMC)); % number of results that have zero mean
RE_RealCMC = RE_RealCMC( ~any( isnan(RE_RealCMC ), 2),: );
mean(RE_RealCMC);

RE_SortedBoot = ...
Sorted_Results_RealBoot(1:maxres,10)./(Sorted_Results_RealBoot(1:maxres,9)*sqrt(NumSim));
RE_RealBoot = Results_RealBoot(:,10)./(Results_RealBoot(:,9)*sqrt(NumSim));
sum(isnan(RE_RealBoot));
RE_RealBoot = RE_RealBoot( ~any( isnan(RE_RealBoot ), 2),: );
mean(RE_RealBoot);

D Bootstrap Functions

D.1 Execution

\classdef BootstrapSMC < handle
\properties (Access = public)
  m_model;
  m_N;
  m_ESS_min;
\end

\properties (Access = private)
  m_ds0;
  m_IterationObjectArray;
\end

\methods
  \function obj = BootstrapSMC(model,N,ESS_min)
  obj.m_model = model;
  obj.m_N = N;
  obj.m_ESS_min = ESS_min;
\end
function RunSMC(obj)
% initialize
obj.m_ds0 = obj.GetNewIterationObject();
for j=1:obj.m_N
    X_j = obj.m_model.M0();
    obj.m_ds0.X{1} = X_j;
    obj.m_ds0.lw(j) = obj.m_model.logG0(X_j);
end
c = max(obj.m_ds0.lw);
obj.m_ds0.lW = obj.m_ds0.lw - c - ... 
    log(sum(exp(obj.m_ds0.lw - c)));

obj.m_IterationObjectArray = cell(1, obj.m_model.m_T);
for t=1:obj.m_model.m_T
    if(t == 1)
        ds_prev = obj.m_ds0;
    else
        ds_prev = obj.m_IterationObjectArray{t-1};
    end
    LogESS = obj.GetLogESS(ds_prev.lw);
    ds = obj.GetNewIterationObject();
    if(exp(LogESS) < obj.m_ESS_min)
        resample_ids = ...
            randsample(1:obj.m_N, obj.m_N, 'true', exp(ds_prev.lW));
        log_what_arr = zeros(obj.m_N, 1);
        ds.resampling = 1;
    else
        resample_ids = 1:obj.m_N;
        log_what_arr = ds_prev.lw;
        ds.resampling = 0;
    end
    % Forward step
    ds.lw = log_what_arr;
    for j=1:obj.m_N
        X_j_prev = ds_prev.X{resample_ids(j)};
        X_j = obj.m_model.Mt(X_j_prev, t);
        ds.X{1} = X_j;
        ds.lw(j) = ds.lw(j) + ... 
            obj.m_model.logGt(X_j_prev, X_j, t);
    end
c = max(ds.lw);
    ds.lW = ds.lw - c - log(sum(exp(ds.lw - c)));
    obj.m_IterationObjectArray{t} = ds;
end

function [RE, logL_t] = GetLogLiklihood(obj)
A Comparison of CMC Against SMC

70 \log L_t = 0;
71 \log W = \text{zeros}(\text{obj.m}_N,1);
72 \text{RE} = \text{zeros}(\text{obj.m}_\text{model.m}_T,1);
73 \text{for} \ t=1:\text{obj.m}_\text{model.m}_T
74 \quad \text{ds} = \text{obj.m}_\text{IterationObjectArray{t}};
75 \quad \text{if}(t == 1)
76 \quad \quad \text{ds}_\text{prev} = \text{obj.m}_\text{ds0};
77 \quad \text{else}
78 \quad \quad \text{ds}_\text{prev} = \text{obj.m}_\text{IterationObjectArray{t-1}};
79 \quad \text{end}
80
81 \% L_t = \text{loglikelihood for one time period} \ t
82 \text{if}(\text{ds}.\text{resampling}==1)
83 \quad \text{c} = \text{max}(\text{ds}.\text{lw});
84 \quad L_t = -\log(\text{obj.m}_N) + \text{c} + \log(\text{sum}(\text{exp}(\text{ds}.\text{lw}-\text{c})));\n85 \quad \text{RE}(t) = \ldots
86 \quad \text{std}(\text{exp}(\text{ds}.\text{lw}))/(\text{mean}(\text{exp}(\text{ds}.\text{lw})))\times\sqrt{\text{obj.m}_N});
87 \text{else}
88 \quad c_t = \text{max}(\text{ds}.\text{lw});
89 \quad c_\text{prev} = \text{max}(\text{ds}_\text{prev}.\text{lw});
90 \quad L_t = c_t + \log(\text{sum}(\text{exp}(\text{ds}.\text{lw}-c_t))) - c_\text{prev} \ldots
91 \quad \ldots \log(\text{sum}(\text{exp}(\text{ds}_\text{prev}.\text{lw}-c_\text{prev})))\n92 \text{end}
93 \% \text{logL}_t = \text{loglikelihood for all} \ t \ \text{eventually}
94 \text{logL}_t = \text{logL}_t + L_t;
95 \text{logW} = \text{logW} + \text{ds}.\text{lW};
96 \text{end}
97
98 \text{methods (Access = private)}
99
100 \text{function} \text{ds} = \text{GetNewIterationObject(obj)}
101 \quad \text{ds.X} = \text{cell}(\text{obj.m}_N,1);
102 \quad \text{ds.lw} = \text{zeros}(\text{obj.m}_N,1);
103 \quad \text{ds.lW} = \text{zeros}(\text{obj.m}_N,1);
104 \quad \text{ds.resampling} = 0;
105 \text{end}
106
107 \text{function} \text{LESS} = \text{GetLogESS(obj,lw)}
108 \quad \text{c} = \text{max}(\text{lw});
109 \quad \text{LESS} = 2\times\log(\text{sum}(\text{exp}(\text{lw}-\text{c}))) - \log(\text{sum}(2\times(\text{lw}-\text{c})))\n110 \text{end}
111
112 \text{end}
113
114 \text{D.2 Functions}

115 \text{classdef ReliabilityModelBootstrap < FeynmanKac}
A Comparison of CMC Against SMC

properties
% theta parameters
m_alpha;
m_eta;
m_beta;
m_mu_A;
m_sigma_A;
m_sigma_Z;

% data
m_dataY;
m_t_arr;
m_m;
m_T;

m_alpha_t_arr;
end

methods

function SetParameters(obj,theta)
    obj.m_alpha = theta(1);
    obj.m_eta = theta(2);
    obj.m_beta = theta(3);
    obj.m_mu_A = theta(4);
    obj.m_sigma_A = theta(5);
    obj.m_sigma_Z = theta(6);

    for t=1:obj.m_T
        if(t==1)
            alpha_t1 = 0;
            alpha_t2 = obj.m_t_arr(1);
        else
            alpha_t1 = obj.m_t_arr(t-1);
            alpha_t2 = obj.m_t_arr(t);
        end
        obj.m_alpha_t_arr(t) = obj.m_alpha * ...
            alpha_t2 * obj.m_eta - obj.m_alpha * ...
            alpha_t1 * obj.m_eta;
    end
end

function obj = ReliabilityModelBootstrap(theta)

   ;%%%%%%%% INPUT OWN TRAJECTORY HERE %%%%%%%%
    tmp = load('C:\Users\20173793\OneDrive - TU ... Eindhoven\Final Project\Matlab Code\example2.mat');
    obj.m_dataY = tmp.Y;
    obj.m_t_arr = tmp.t;
    obj.m_m = size(obj.m_dataY,1);
    obj.m_T = size(obj.m_dataY,2);
    obj.m_alpha = theta(1);
A Comparison of CMC Against SMC

```matlab
obj.m_eta = theta(2);
obj.m_beta = theta(3);
obj.m_mu_A = theta(4);
obj.m_sigma_A = theta(5);
obj.m_sigma_Z = theta(6);

obj.m_alpha_t_arr = zeros(obj.m_T,1);
for t=1:obj.m_T
    if(t==1)
        alpha_t1 = 0;
        alpha_t2 = obj.m_t_arr(1);
    else
        alpha_t1 = obj.m_t_arr(t-1);
        alpha_t2 = obj.m_t_arr(t);
    end
    obj.m_alpha_t_arr(t) = obj.m_alpha * ...
                          alpha_t2ˆobj.m_eta - obj.m_alpha * ...
                          alpha_t1ˆobj.m_eta;
end

function res = M0(obj)
    res=normrnd(obj.m_mu_A,obj.m_sigma_A,obj.m_m,1);
end

function res = logG0(obj,x)
    res=0;
end

function res = Mt(obj,x_prev,t)
    DeltaX = ...
    gamrnd(obj.m_alpha_t_arr(t),obj.m_beta,obj.m_m,1) ;
    res= x_prev + DeltaX;
end

function res = logGt(obj,~,x_t,t)
    z = obj.m_dataY(:,t) - x_t;
    mean(- 0.5 * z.^2 ./obj.m_sigma.Z^2);
    logpdf = -0.5*log(2*pi*obj.m_sigma.Z^2) - 0.5 * z.^2 ... 
             ./obj.m_sigma.Z^2;
    res=sum(logpdf);
end
end
end
end

D.3 Model

```
E Data Analysis Code

This code was written in RStudio.

```r
set.seed(1234)
firstdeg <- c(3, 3, 3, 2, 2, 0, 2, 3, 2, 5, 5, 2, 5, 6, 5, 2, 1, 2, 3, 2, 7, 1, 2, 4)
seconddeg <- c(3, 3, 3, 4, 4, 1, 0, 1, 2, 4, 3, 2, 4, 1, 1, 2, 4, 3, 2, 3, 5, 1, 4, 4, 3, 5, 3)
adjustfirstdeg <- c(3, 3, 3, 2, 2, 2, 3, 2, 5, 5, 2, 5, 6, 5, 2, 1, 2, 3, 2, 7, 1, 2, 4)
adjustseconddeg <- c(3, 3, 4, 4, 1, 1, 2, 4, 3, 2, 3, 5, 1, 4, 3, 5, 3)
gofTest(adjustfirstdeg, test="proucl.ad.gamma", distribution="gamma")
gofTest(adjustseconddeg, test="proucl.ad.gamma", distribution="gamma")
gofTest(firstdeg, test="ad", distribution="norm")
gofTest(firstdeg, test="cvm", distribution="norm")
gofTest(firstdeg, test="lillie", distribution="norm")
gofTest(seconddeg, test="ad", distribution="norm")
gofTest(seconddeg, test="cvm", distribution="norm")
gofTest(seconddeg, test="lillie", distribution="norm")

dat <- rgamma(20, shape = 2, scale = 3)
gof.list <- gofTest(dat, distribution = "gamma")
gof.list
hist(firstdeg, breaks=-1:7, col="lightblue", border="#000033", main="Difference Between Second and First Time Period Observation", xlab = "y(2)-y(1)", xaxp = c(-1, 7, 8), plot=FALSE)
hist(seconddeg, breaks=-1:5, col="lightblue", border="#000033", main="Difference Between Third and Second Time Period Observation", xlab="y(3)-y(2)")
plot(density(firstdeg), main= "Difference Between Second and First Time Period Observation", xlab = "Degradation + measurement error difference = y(2)-y(1)"
plot(density(seconddeg), main= "Difference Between Third and Second Time Period Observation", xlab = "Degradation + measurement error difference = y(3)-y(2)"
```

M.B.C. Buist
A Comparison of CMC Against SMC

```r
n=1000000

g1 = rgamma(n, shape = 0.5, scale = 0.2)
g2 = rgamma(n, shape = 5, scale = 0.2)
g3 = rgamma(n, shape = 50, scale = 0.2)
g4 = rgamma(n, shape = 125, scale = 0.2)
g5 = rgamma(n, shape = 0.5, scale = 0.8)
g6 = rgamma(n, shape = 5, scale = 0.8)
g7 = rgamma(n, shape = 50, scale = 0.8)
g8 = rgamma(n, shape = 20, scale = 0.99)

# mean = 0
# sd between 0 and 5 (1 and 3 potential values)
# sd = sqrt(2*sd^2) = sqrt(2) en sqrt(18)

n1 = rnorm(n, mean = 0, sd = sqrt(2)) #sigma = sqrt(2)

n2 = rnorm(n, mean = 0, sd = sqrt(18)) #sigma = sqrt(18)

plot(density(g1+n1), xlim=c(-5,35), ylim=c(0,0.3), main= "Probability Function of Sums of Gamma and Normal RV's", xlab = "Degradation + measurement error difference")
lines(density(g2+n1), col="red")
lines(density(g5+n1), col="green")
lines(density(g6+n1), col="purple")
lines(density(g3+n2), col="blue")
lines(density(g6+n2), col="orange")
lines(density(g8+n2), col="deeppink")

legend(x = c(20, 34), y = c(0.3, 0.09), legend=c("k = 0.5, beta = 0.2, sigma = \sqrt{2}", "k = 5, beta = 0.2, sigma = \sqrt{2}", "k = 50, beta = 0.2, sigma = \sqrt{2}", "k = 5, beta = 0.8, sigma = \sqrt{2}", "k = 20, beta = 0.99, sigma = \sqrt{18}", "k = 5, beta = 0.8, sigma = \sqrt{18}", "k = 50, beta = 0.2, sigma = \sqrt{18}"), col=c("black", "red", "green", "purple", "blue", "orange", "deeppink"), lty=1, cex=0.8)
```

M.B.C. Buist 86