NTNU – Trondheim
Norwegian University of
Science and Technology

Estimation of Reliability by Monte Carlo Simulations

Combined with Optimized Parametric Models

Harald Svandal Bø

Master of Science in Physics and Mathematics
Submission date: June 2014
Supervisor: Arvid Næss, MATH
Co-supervisor: Bo Friis Nielsen, Technical University of Denmark

Norwegian University of Science and Technology
Department of Mathematical Sciences

Preface

This thesis is carried out at the Norwegian University of Science and Technology (NTNU) during the spring semester 2014, and concludes my Master of Science in Applied and Engineering Mathematics. The thesis is a part of the Nordic Five Tech in Applied and Engineering Mathematics (N5TeAM), with the Technical University of Denmark (DTU) as my home university and NTNU as host university.

I would like to thank my supervisors, Arvid Næss, Professor in Statistics at NTNU, and Bo Friis Nielsen, Associate Professor of Stochastic Modeling at DTU. I would especially like to thank Arvid Næss for valuable help and encoaraging meetings along the way.

I am grateful for the support and inspiration family and friends have provided me through the 5 years of studying. It has been a great journey, and I am excited to see what the future brings.

Abstract

A new method for efficient Monte Carlo simulations is developed, and used to estimate the reliability of dependent and independent systems. The method consists of Monte Carlo simulations on parametrized systems, where the robust qualities in the simulations are preserved. The parametrized system corresponds to the given system for set parameter values. By using the regularity of system reliability as a function of the parameter, the original system reliability can be predicted accurately with relative small samples. The estimate is obtained by using weighted least squares to fit the parametrized simulations. Probabilities in the order 10^{-6} to 10^{-8} can be estimated very accurate with $n = 10^5$ as the sample size. For some systems, even probabilities in the order 10^{-12} are estimated with 10% relative difference from the analytic solution.

Sammendrag

En ny metode for effektive Monte Carlo-simuleringer er utviklet, og brukt til å estimere påliteligheten til avhengige og uavhengige systemer. Metoden består av å gjøre Monte Carlo-simuleringer på parametriserte systemer, hvor de robuste egenskapene i simuleringene er bevart. Det parametriserte systemet svarer til det opprinnelige systemet for gitte parameterverdier. Ved å bruke regulariteten i systempålitelighet som funksjon av parameteren, kan det opprinnelige systemets pålitelighet forutsies nøyaktig med relativt få simuleringer. Estimatet er oppnådd ved å bruke vektet minste kvadraters metode for å tilpasse de parametriserte simuleringene. Selv med få genererte tall, $n = 10^5$, kan sannsynligheter i størrelsesorden 10^{-6} til 10^{-8} estimeres svært nøyaktig. For noen systemer, blir selv sannsynligheter i størrelsesorden 10^{-12} estimert med en prosentvis differanse på 10% fra den analytiske løsningen.

Problem Description

- Describe the statistical background in system reliability theory
- Analysis and further development of the Monte Carlo method
- Applying the method to models in system reliability

Assignment given: January 17, 2014
Supervisor: Arvid Næss, NTNU
Co-supervisor: Bo Friis Nielsen, DTU

Contents

Chapter 1

Introduction

This thesis is a continuance of the project assignment [2] completed in the fall semester 2013. The project assignment described a parametrized Monte Carlo method applied to problems in system reliability. The project assignment solely focused on independent systems, and the results from the estimations were promising. The method has been analysed further, both on independent and dependent systems. Some of the theory described and developed in the project assignment [2] is included in this thesis for completeness.

Standard Monte Carlo simulation forms a simple and robust alternative for estimating system reliability. One of the problems with the standard method is however its slow convergence, $\mathcal{O}(n^{-\frac{1}{2}})$. The standard method normally needs large samples to get accurate results, and this is a time and memory consuming operation. Husby, Naustdal and Vårli [6] used conditional Monte Carlo methods to make good estimates of system reliability. We will introduce a Monte Carlo method that allows us to investigate parametrized systems where failures occur more often than the original system. We can reduce the sample size for the parametrized simulations, and still achieve a precise estimation of these. To estimate the reliability for the original system, we minimize the least square errors between the parametrized results and three chosen families of equations. The result is an efficient way to determine system reliability, both for dependent and independent systems.

The four following chapters in the thesis presents the parametrized Monte Carlo method by the following set up:

Chapter 2 gives a statistical approach to, and describes standard concepts in system reliability. The two first sections are copied from the project assignment[2], and defines reliability block diagrams and structure functions. Cut and path sets are described further, and the use of Markov chains in system reliability is presented. The chapter presents construction of different independent and dependent systems, and how one can specify Markov chains and cut sets for the systems. The dependent systems consists of components with common cause failures or cascad-

ing failures.

Chapter 3 introduces Monte Carlo simulations, with some brief examples of how the method has been used in other applications. The chapter describes how we can simulate failure of components, and how we can use this to simulate the systems defined in Chapter 2. Calculations of mean, variance and confidence intervals for simulations are presented, with results that highlight the importance of a large sample size for original Monte Carlo simulation. The parametrized method is then introduced and described. The independent parametrization is the same as for the project assignment [2], but two parametrizations for the dependent systems are presented. The description of the least squares fit is the same as in the project assignment [2]. The chapter describes the optimization tools and weighting strategy used to fit the parametrized system. The weighting strategy from the project assignment [2] is presented, along with three other weighting strategies. The implementation of these weighting strategy is presented. Finally, the implementation of the different systems are presented. The implementation consists of different dependent and independent systems.

Chapter 4 presents the results from the parametrized Monte Carlo simulations. The independent systems are investigated with different number of components and failure rates, and for all the introduced weights. The estimate of independent systems are compared to the known analytic solution. The different dependent systems are investigated, and the result is compared with the limiting probability of Markov chain where this is possible. For the systems without Markov chains, the estimations are compared to large simulations of the original Monte Carlo method. The confidence intervals defined in chapter 3 are used to validate the accuracy of the estimates. Results are presented in tables and figures. The aim of the chapter is to show how the different systems affect the curve of the parametrized estimation. This is done by comparing the different families of equations and weights. The chapter also shows that the sample size can be reduced drastically when parametrizing the Monte Carlo simulations.

Chapter 5 concludes the master thesis with a discussion of the observed results and systems. The different weights and different families of functions are compares in order to find out how different systems should be estimated and parametrized. The chapter lists areas where the method can be analysed and improved even further.

Chapter 2

System Reliability

2.1 Reliability Block Diagram

From the project assignment [2], we have that the standard ISO 8402 defines reliability as

Definition 2.1.1. The ability of an item to perform a required function, under given environmental and operational conditions and for a stated period of time.

We follow the notation in Rausand and Høyland [8], and let the term "item" denote any component, subsystem, or system that can be considered as an entity. A function may be a single function or a combination of functions that is necessary to provide a specified service. By using a reliability block diagram, we are able to establish deterministic models of structural relationships. The reliability block diagram of s components in series can be seen in figure 2.1. When the components are in series, all of the components needs to function for the system to be functioning. When all the components are in parallel, however, it is sufficient that one component functions for the system to be functioning. The reliability block diagram of s components in parallel can be seen in figure 2.2. A way to combine combine components in series and parallel is to establish *koon systems*, described in Rausand and Høyland [8] page 124. For these systems, k out of the s components in the system needs to function for the system to be functioning. In figure 2.3, a structure with 9 components is given. This function has two *koon* sub-systems, both where $k = 2$ and $s = 3$. These are combined in series with three other components.

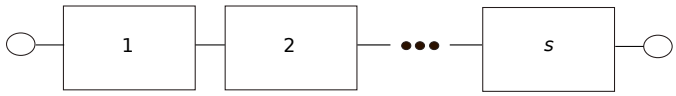


Figure 2.1: Series structure with s components.

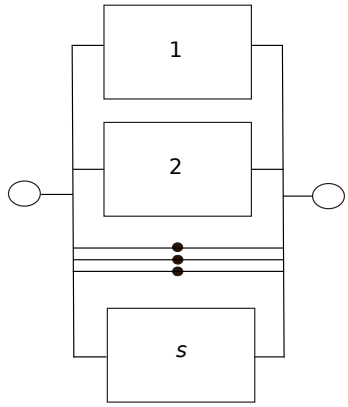


Figure 2.2: Parallel structure with s components.

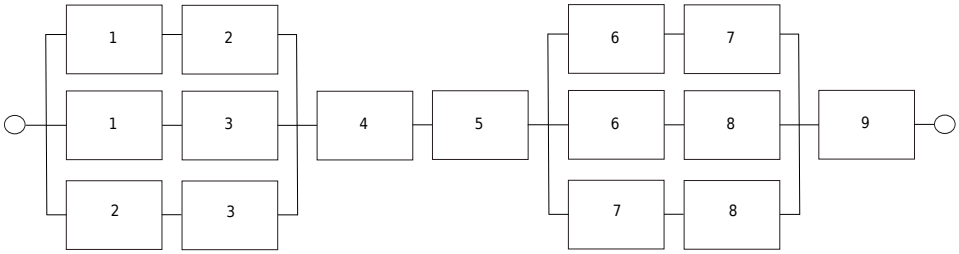


Figure 2.3: Structure with s components combined in parallel and series.

2.2 Structure Function

We also described the structure function in the project assignment [2], and it is included here for completeness. The establishment of a system in a reliability block diagram helps us to think of the system as a function, and to define the structure function. Given a system consisting of s components where each component has two distinguishable states, one functioning and one failed state. We can define the state of component i, $i = 1, 2, ..., s$ by

$$x_i = \begin{cases} 1 & \text{if component } i \text{ is functioning} \\ 0 & \text{if component } i \text{ is in a failed state} \end{cases}$$

The state of the system can be described by the function

$$\phi(\mathbf{x}) = \phi(x_1, x_2, ..., x_s),$$

where $\mathbf{x} = (x_1, x_2, ..., x_s)$ is called the *state vector* and

$$\phi(\mathbf{x}) = \begin{cases} 1 & \text{if the system is functioning} \\ 0 & \text{if the system is in a failed state} \end{cases} \tag{2.1}$$

$\phi(\mathbf{x})$ is called the *structure function* of the system. There are two main classes of combining components in a structure: series structure and parallel structure. In the series structure, the system is functioning if and only if all the n components in the system are functioning. For a parallel structure, it is sufficient that at least one of the n components in the system are functioning. It is shown in Rausand and Høyland [8] that the structure function for a series structure is

$$\phi(\mathbf{x}) = x_1 \cdot x_2 \cdots x_s = \prod_{i=1}^{s} x_i. \tag{2.2}$$

The structure function for a parallel structure is

$$\phi(\mathbf{x}) = 1 - (1 - x_1)(1 - x_2) \cdots (1 - x_s) = 1 - \prod_{i=1}^{s} (1 - x_i). \tag{2.3}$$

Since we can not predict with certainty whether or not a component will be in a failed state after t time units, we introduce random variables for the state vector by

$$X_1(t), X_2(t), \ldots, X_s(t).$$

The corresponding state vector will be denoted by

$$\mathbf{X}(t) = (X_1(t), X_2(t), \ldots, X_s(t), \tag{2.4}$$

and corresponding structure function

$$\phi(\mathbf{X}(t)).$$

With this state vector, we can define following probabilities:

$$p_i(t) = \Pr(X_i(t) = 1) \quad \text{for} \quad i = 1, 2, \ldots, s \quad \text{and} \tag{2.5}$$

$$p_S(t) = \Pr(\phi(\mathbf{X}(t)) = 1), \tag{2.6}$$

where (2.5) is the probability that component i will be functioning at time t and (2.6) is the probability that the system will be functioning at time t. The probability of a component to fail will be

$$q_i(t) = 1 - p_i(t). \tag{2.7}$$

2.3 Path and Cut Sets

We can arrange the different components in a system in cut and path sets. The definitions is given by Rausand and Høyland [8] page 129: Let the set of s components be denoted

$$C = 1, 2, \ldots, s.$$

Definition 2.3.1 (Path Sets, Minimal Path Sets)**.** A path set P is a set of components in C which by functioning ensures that the system is functioning. A path set is said to be minimal if it cannot be reduced without losing its status as a path set.

Definition 2.3.2 (Cut Sets, Minimal Cut Sets)**.** A Cut set K is a set of components in C which by failing causes the system to fail. A cut set is said to be minimal if it cannot be reduced without losing its status as a cut set.

Given the states of all the components, we can use the cut and path sets to find the condition of the system. To use the minimal cut and minimal path sets is the most efficient way to test calculate the state of the system. In our implementation, we use the minimal cut and path sets defined by the stochastic variables to have efficient algorithms. The minimal cut sets can also be used to define probability block diagrams for systems. All the components in a cut set can be represented by a parallel structure. The different sets are represented by a series structure.

The minimal path and cut sets for the system in figure 2.1 is given in table 2.1 For the system in figure 2.2 the minimal path and cuts sets are given in table 2.2.

Table 2.1: Minimal Path and Cut Sets in figure 2.1

Minimal path sets	Minimal cut sets
$\{1,2,\ldots,s\}$	$\{1\}$
	$\{2\}$
	$\vdots$
	$\{s\}$

The minimal cut ant path sets for the system in figure 2.3 is shown in figure 2.4.

Table 2.2: Minimal Path and Cut Sets in figure 2.2

Path sets	Cut sets
$\{1\}$	$\{1,2,\ldots,s\}$
$\{2\}$	
$\vdots$	
$\{s\}$	

Table 2.3: Minimal Path and Cut Sets in figure 2.3

Path sets	Cut sets
{1,2,4,5,6,7,9}	{4}
{1,3,4,5,6,7,9}	{5}
{2,3,4,5,6,7,9}	{9}
{1,2,4,5,6,8,9}	{1,2}
{1,3,4,5,6,8,9}	{1,3}
{2,3,4,5,6,8,9}	{2,3}
{1,2,4,5,7,8,9}	{6,7}
{1,3,4,5,7,8,9}	{6,8}
{2,3,4,5,7,8,9}	{7,8}

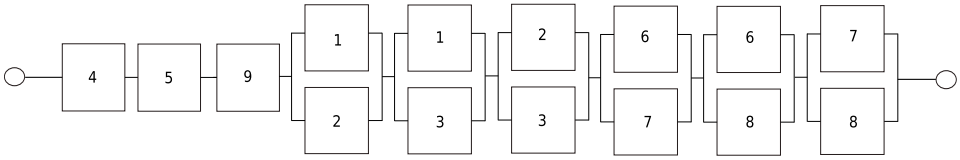


Figure 2.4: Series structure with s components.

2.4 Independent systems

In the project assignment [2], we investigated independent systems, and defined the stochastic variable

$$X_i : \begin{cases} \Pr(X_i = 1) &= p_i &= 1 - 10^{-z_i} \\ \Pr(X_i = 0) &= 1 - p_i &= 10^{-z_i}. \end{cases} \tag{2.8}$$

X_i represent the state of component i, and p_i is the probability that the ith component is functioning defined in equation (2.5). The failure probabilities are written on the form 10^{-z_i}, where $z_i > 0$

For the independent case, X_i is Bernoulli distributed, $X_i \rightsquigarrow Bern(p_i)$, with mean p_i and variance $p_i(1 - p_i)$.

The system reliability for a series of s components defined by the stochastic variable becomes:

$$p_S = \prod_{i=1}^{s} p_i = (1 - 10^{-z_1})(1 - 10^{-z_2}) \cdots (1 - 10^{-z_s}), \tag{2.9}$$

while the system reliability for a parallel of s components is

$$p_S = 1 - \prod_{i=1}^{s} (1 - p_i) = 1 - (10^{-z_1})(10^{-z_2}) \cdots (10^{-z_s}). \tag{2.10}$$

All independent systems can be divided into structures consisting of parallel and series. By using equation (2.9) and (2.10) we can get the analytical solution for the system reliability. This allows us to compare our simulation results with the expected analytical result.

2.5 Dependent Systems

In the project assignment [2] we defined *positive dependancy* and *negative dependancy*, but did not simulate any dependent systems. We described two examples of *negative dependencies*, common cause failures and cascading failures, which we will analyse further.

We will investigate systems with negative dependency, where one component fail may lead to a higher probability for other components in the system to fail. There are many ways to construct systems like this, and we will focus on the mentioned common cause failures and cascading failures. The constructed systems are intended to represent realistic systems and incidents.

2.5.1 Common Cause Failures

From the project assignment [2] we described common cause failures as: Common cause failures are according to NUREG/CR-628 [8] a "dependent failure in which two or more component fault states exist simultaneously, or within a short time interval, and are a direct result of a shared case." One example of a common cause failures is when a harsh environment makes multiple components fail simultaneously. Common cause failures may be modelled by a β-factor model, described in Rausand and Høyland [8] page 217. This is done by introducing two possible causes of component failure:

1. Circumstances that concern only the component (independent of the condition of the remaining components).
2. Occurrence of an external event (independent of the system) whereby *all* the components fail at the same time.

Rausand and Høyland [8] uses a failure rate λ for each component, where

$$\lambda = \lambda^{(i)} + \lambda^{(c)}. \tag{2.11}$$

$\lambda(i)$ represent the independent failures in case 1, and $\lambda(c)$ represent the common cause failures in case 2. By introducing the factor β as the "common cause factor"

$$\beta = \frac{\lambda^{(c)}}{\lambda} \tag{2.12}$$

$$\lambda^{(c)} = \beta\lambda \tag{2.13}$$

It is easy to calculate and construct β-factor models, but it may be difficult to determine the correct value for β for known systems. It is also a weakness that

the common cause failures make all components fail. When we implement systems that can fail due to common cause failures, we change Rausand and Høylands [8] notation of failure rate, λ, with the probability of failure, q, defined in equation (2.7). By doing this, we get the probability to fail for a given component, i,

$$q_i = 1 - (1 - q_i^{(i)})(1 - q_i^{(c)}). \tag{2.14}$$

$q_i^{(i)}$ describes the independent failure probability for component i, while $q_i^{(c)}$ is the probability for common cause failures for component i. When we construct the systems, we have the possibility to make the probability for common cause failures different for each of the components. This is not a possibility in the original β-factor model. We define β_{ij} to be

$$Pr(\text{Common cause failure that effect component } i \mid \text{Failure on component } j) = \beta_{ij}$$

The stochastic variable that represent the state of component i is represented by

$$X_i : \begin{cases} \Pr(X_i = 1) = (1 - q_i^{(i)})(1 - q_i^{(c)}) \\ \qquad\qquad = (1 - 10^{-z_i}) \prod_{j=1, j\neq i}^{s} (1 - \beta_{ij}(1 - 10^{-z_j})) \\ \Pr(X_i = 0) = 1 - (1 - q_i^{(i)})(1 - q_i^{(c)}) \\ \qquad\qquad = 1 - (1 - 10^{-z_i}) \prod_{j=1, j\neq i}^{s} (1 - \beta_{ij}(1 - 10^{-z_j})) \end{cases} \tag{2.15}$$

where s is the number of components in the system. β_{ij} is the probability that fail of component j affects failure on component i. The system reliability is given as

$$p_S = \mathrm{E}(\phi(\mathbf{X}),$$

and the probability of failure for the system is given as

$$p_F = 1 - p_S. \tag{2.16}$$

2.5.2 Cascading failures

In the project assignment [2] we described cascading failures as: The cascading failures are multiple failures initiated by a failure of one component, referred to as a "domino effect" in Rausand and Høyland [8]. These failures may occur when components share a common load, and that failure of one component increases the load on the remaining components.

When we implement the cascading failures, the probability of failure for the different components are dependent on the time, t. The stochastic variable that represent the state of component i is represented by

$$X_i(t, \mathbf{X_{-i}}) : \begin{cases} \Pr(X_i(t, \mathbf{X}_{-i}) = 1) &= p_i(t, \mathbf{X}_{-i}) &= 1 - 10^{-z_i(t, \mathbf{X}_{-i})} \\ \Pr(X_i(t, \mathbf{X}_{-i}) = 0) &= 1 - p_i(t, \mathbf{X}_{-i}) &= 10^{-z_i(t, \mathbf{X}_{-i})}. \end{cases} \tag{2.17}$$

The vector $\mathbf{X}_{-i}$ represent the state vector $X_1, X_2, \ldots, X_s$ without the $i'th$ entry.

The system reliability is given as

$$p_S(t) = \mathrm{E}(\phi(\mathbf{X}(t)),$$

and the probability of failure for the system is given as

$$p_F(t) = 1 - p_S(t). \tag{2.18}$$

We will investigate two ways to construct a realistic dependent probability of failure $p_i(t, \mathbf{X}_{-i})$. For the independent systems and common cause failures, we have automatically repaired the systems for each run. This implies that the system always is in its initial state for each run. By modelling cascading failures, previous behaviour will affect the probability to fail forward in time. To construct such systems in a good way, we need to have a repair interval or a condition that makes us repair the components back to their initial state. Otherwise the system would end up failing every time when it is run $n \to \infty$ times.

The different systems with cascading failures are described in section 3.4.2 to 3.4.5, and follow this procedure:

- If one component fails, it is removed from the system until the system fails or all components are repaired
- If one component fails, the probability of other components to fail increases

The two steps in the procedure are combined for the different components in a way that represent realistic systems.

2.6 Markov Chains

Some of the dependent systems may be represented by Markov chains. Let the stochastic process

$$Y_n, \quad n = 0, 1, 2, \ldots$$

represent the different states the system is in at different times, n. From Ross [9] page 185 we have that if $Y_n = i$, then the system is in state i. If the Markov chain should be valid, there must be a fixed probability P_{ij} that the system will go from state i to state j in the next time step. This is expressed in [9] page 185 as

$$Pr(Y_{n+1} = j | Y_n = i, Y_{n-1} = i_{n-1}, \ldots, Y_1 = i_1, Y_0 = i_0) = P_{ij} \tag{2.19}$$

for all states $i_0, i_1, \ldots, i_{n-1}, i, j$ and $n \geq 0$.

Equation (2.19) is visualised for a system with S states in figure 2.5

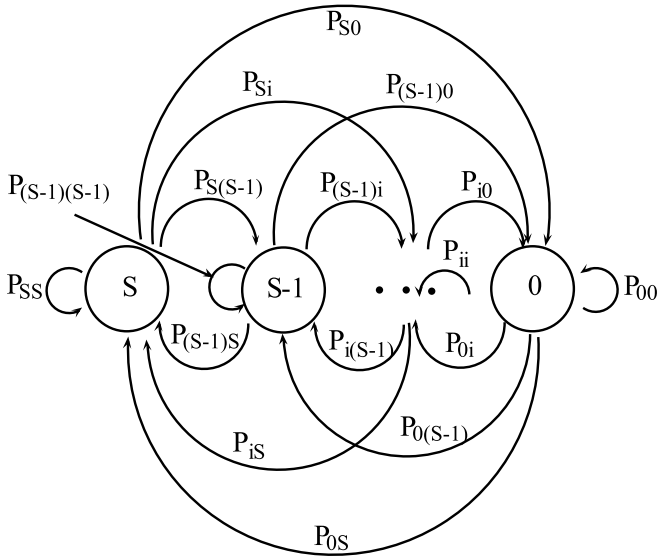


Figure 2.5: Markov chain of a system with S states. P_{ij} denotes the fixed probability defined in equation (2.19)

The transition probabilities in a Markov chain can also be visualised by a matrix. The matrix of one step transition probabilities for a Markov chain with S states is given in equation (2.20).

$$\mathbf{P} = \begin{bmatrix} P_{SS} & P_{S(S-1)} & \cdots & P_{S0} \\ P_{(S-1)S} & P_{(S-1)(S-1)} & \cdots & P_{(S-1)0} \\ \vdots & \vdots & \vdots & \vdots \\ P_{0S} & P_{0(S-1)} & \cdots & P_{00} \end{bmatrix} \tag{2.20}$$

The matrix in equation (2.20) can be used to calculate the limiting probability of the Markov chain. Theorem 4.1 in Ross [9] page 205 states

Theorem 2.6.1 (Limiting Probabilities)**.** *For an irreducible ergodic Markov chain* $\lim_{n\to\infty} P_{ij}^n$ *exists and is independent of* i*. Furthermore, letting*

$$\pi_j = \lim_{n\to\infty} P_{ij}^n, \quad j \geq 0$$

then π_j *is the unique nonnegative solution of*

$$\pi_j = \sum_{i=0}^{\infty} \pi_i P_{ij}, \quad j \geq 0,$$

$$\sum_{j=0}^{\infty} \pi_j = 1$$

If we have systems of components defined as a Markov chain, we can use the limiting probabilities to find the probability of failure for the system. This is done by adding π_j for the j states where the system is not functioning.

One example of an independent system represented by a Markov Chain is a parallel structure with three components, figure 2.2 with $s = 3$. The states can be formed as

3. Three components functioning
2. Two components functioning
1. One component functioning
0. None components functioning

To find the limiting probabilities of the states, the conditions in theorem 2.6.1 and equation (2.19) needs to be satisfied. This means that the Markov chain needs to be aperiodic, all states needs to communicate with each other with fixed transition probabilities, and if starting in state i, the expected time until the process returns to state i should be finite. For a standard parallel structure, state 0 is the only state where the system has failed. If the necessary conditions are satisfied, the probability of failure in the system would be $p_F = \pi_0$.

Chapter 3

Monte Carlo simulation

3.1 Monte Carlo method

3.1.1 Monte Carlo Definition

Monte Carlo methods are a class of mathematical models based on random sampling. The idea behind the Monte Carlo simulation is to evaluate an integral as an expected value. The integral we want to estimate is

$$I = E_\mu[\phi(X)] = \int_\Omega \phi(x) d\mu(x), \tag{3.1}$$

where ψ is a function on $\Omega \in \mathbb{R}^n$ over $\mathbb{R}$ and $X = (X_1, \ldots, X_n)$ an n-dimensional vector of random variables from the same distribution with law μ. The method is based on the *Strong Law of Large Numbers* and the *Central Limit Theorem*, given by Ross, [9] page 79, and provides an unbiased estimator:

Theorem 3.1.1 (Strong Law of Large Numbers)**.** *Let $\phi(X_1), \phi(X_2), \ldots$ be a sequence of independent random variables having a common distribution, and let $E[\phi(X_i)] = \mu < \infty$. Then, with probability 1,*

$$\frac{\phi(X_1) + \phi(X_2) + \cdots + \phi(X_n)}{n} \to \mu, \quad as \;\; n \to \infty \tag{3.2}$$

Theorem 3.1.2 (Central Limit Theorem)**.** *Let $\phi(X_1), \phi(X_2), \ldots$ be a sequence of independent, identically distributed random variables, each with mean μ and variance σ^2. Then the distribution of*

$$\frac{\phi(X_1) + \phi(X_2) + \cdots + \phi(X_n) - n\mu}{\sigma\sqrt{n}} \tag{3.3}$$

tends to the standard normal as $n \to \infty$. That is,

$$P\left\{ \frac{\phi(X_1) + \phi(X_2) + \cdots + \phi(X_n) - n\mu}{\sigma\sqrt{n}} \leq a \right\} \quad \to \quad \frac{1}{\sqrt{2\pi}} \int_{-\infty}^{a} e^{-x^2/2} dx \tag{3.4}$$

as $n \to \infty$.

Let $\mu = E[\phi(X)]$. With standard Monte Carlo, the unbiased estimator of I for sample size n is

$$\mu = \frac{1}{n} \sum_{i=1}^{n} \phi(x_i) \tag{3.5}$$

3.1.2 Buffon's needle

Experimental Monte Carlo methods have been used to approximate the mathematical constant π, where one of the experiments is Buffon's needle experiment stated in the 18th century. Buffon's needle [11] experiment consists of tossing a needle with length l, n times on a grid with equidistant parallel lines. The distance between the parallel lines should be $d \geq l$. The experiment is sketched in figure 3.1. Each needle toss can be described by two random variables:

Y : distance from the center of the needle to the closest line

Θ : angle of the needle with response to the lines

where $0 \leq Y \leq d/2$ and $0 \leq \Theta \leq \pi$.
Each toss where the needle crosses one of the parallel line is counted. By defining

h : number of needles that cross a line

n : number of needle tosses

we can use probability theory and geometry to define the approximation of π

$$\hat{\pi} = \frac{2ln}{dh}. \tag{3.6}$$

Mario Lazzarini performed the Buffon's needle experiment in 1901 [11], and got the approximation $\hat{\pi} \approx \frac{355}{113}$ by tossing a needle 3408 times, which differs from π with less than 3×10^{-7}. This result is remarkably accurate, and has a better accuracy than one can expect with only 3408 tosses.

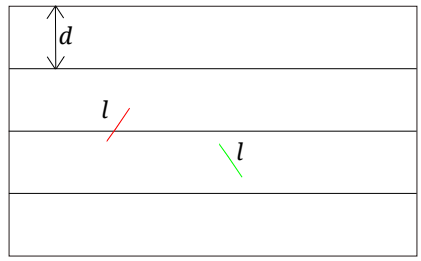


Figure 3.1: Visualisation of the Buffon's needle experiment. d is the distance between the parallel lines and l is the length of the needle. The red needle shows a needle that crosses a line, and the green shows a needle that does not.

3.1.3 Modern Monte Carlo

The modern version of Monte Carlo method was initiated by Stanislaw Ulam [7] in the late 1940s, and methods have been used to study many phenomena since. For instance may the mathematical constant π now be estimated by generating samples of two random independent uniform variables $(X, Y) \sim U(0, 1)$, and calculate the fraction of the samples where $x^2 + y^2 \leq 1$. By multiplying this fraction with 4, we obtain an estimate for π. The two mentioned examples of estimating π shows the idea behind the Monte Carlo methods, and we can establish a four step procedure that can be applied to most Monte Carlo methods.

1. Define input domain
2. Generate results from the random variables over the domain
3. Sort the different input into their respective outcomes
4. Calculate the result by using the coherence between the defined domain and the random variables

The first step is to define input domain. In Buffon's needle experiment the input is a grid with equidistant parallel lines. The distance, d, between the lines is larger than the length of the needle, l. For the other case, the domain is the square $[0, 1] \times [0, 1]$. The second step is to generate results from the random variables over the domain. This would be the tossing of a needle in Buffon's needle experiment. For the second case this is to generate x and y from the random variables $(X, Y) \sim U(0, 1)$. The third step is to sort different input into their respective outcomes. In the needle experiment we count the needles that cross one of the parallel lines. For the other case, we count the incidents where $x^2 + y^2 \leq 1$. The fourth step is to calculate the result by using the defined domain, random variables and outcomes. For the needle experiment, the combination of geometry of probability theory gives the result (3.6). For the second case we multiply the fraction of results where $x^2 + y^2 \leq 1$ by 4. By following the same procedure, we can define the four step procedure we will use in our Monte Carlo simulations:

1. Input domain is [0, 1]
2. Generate, n random variables, X_n uniformly distributed over the input domain, $X \sim U(0, 1)$
3. Count the incidents where the components fail, $X > p_i$, for all the components in the system. p_i is the i'th components reliability, defined in equation (2.5).
4. Calculate the system reliability from the results in step 3. The fraction of system failures divided by n provides the Monte Carlo estimate for system failure probability, $\hat{p_{F}}_n$

The random events we represent with the random variables is whether or not a component in our system is functioning. By following step 1 to 4 we can get an estimate of the probability of failure, $\hat{p_F}$, for a specified system. The simulations are based on the stochastic variable defined in 2.8, and we simulate n Bernoulli trials for each component in the system. The result for each components is combined to decide the state of the system.

3.1.4 Generation of pseudo random numbers

To implement the simulations, we need to generate random numbers. We assume that the components we want to investigate have a known probability to fail. The best way to simulate fails of these components is to generate random variables from the uniform distribution, and to compare each number with the probability to fail for the components. We generate the vector $\mathbf{u} \sim U(0,1)$, where the length of $\mathbf{u}$ is the size of the sample, n. For each of the n values in $\mathbf{u}$, we compare $u(j) > p_i$ with p_i from equation (2.5) and $j = 1, 2, \ldots, n$. This gives two outputs for a component, i

- $u > p_i \quad \rightarrow \quad$ component i fails
- $u \leq p_i \quad \rightarrow \quad$ component i is functioning

The procedure is repeated for all n random variables in the sample. The vector of uniform random variables is created with the function `rand(n)` in MATLAB. The call `rand(n)` generates a sequence of n pseudo random scalars between 0 and 1. The generated scalars are not truly random, but follow a given sequence defined in MATLAB. The sequence is a list of numbers that approximates the behaviour of random numbers. Since the generated numbers follow a given sequence, it is possible to generate vectors containing the same numbers many times. This allows us to test different methods with the same random variables without storing all the numbers. It is also possible to sample from the standard normal distribution and to use the Box-Muller transform to generate $\mathbf{u} \sim U(0,1)$, but this is not done in this thesis.

3.2 Convergence rate and confidence interval

3.2.1 Experimental

By applying the Monte Carlo method on the system reliability p_S from (2.6), we get an estimator of I for N trials,

$$\hat{p_{S}}_N = \frac{1}{N} \sum_{i=1}^{N} \phi(X_i), \tag{3.7}$$

where $\hat{p_{S}}_N$ is the estimator of p_S obtained with N trials. X_i are the independent, identically distributed random variables defined in (2.8), and ϕ the structure function of the system. By the Law of Large Numbers, theorem 3.1.1, the estimator $\hat{p_S}$

is unbiased. The variance of the estimator is

$$\sigma_N^2 = \frac{\sigma}{N} \tag{3.8}$$

which is estimated by

$$\hat{\sigma}_N^2 = \frac{1}{N-1}\left[\frac{1}{N}\sum_{i=1}^{N}\phi(X_i)^2 - \hat{p_S}_N^2\right]. \tag{3.9}$$

Since X_i is binary, (3.9) can be simplified to

$$\hat{\sigma^2}_N = \frac{1}{N-1}\left[\frac{1}{N}\sum_{i=1}^{N}\phi(X_i) - \hat{p_S}_N^2\right]. \tag{3.10}$$

Equation (3.10) can be simplified to

$$\hat{\sigma^2}_N = \frac{1}{N(N-1)}\big(S(1-\hat{p_S}_N^2) - F\hat{p_S}_N^2\big), \tag{3.11}$$

where F is the number of failures for the system and $S = (N - F)$ is the number of successes in N trials. From equation (3.11)

$$\lim_{N\to\infty} \hat{\sigma^2}_N = 0$$

We can define confidence intervals of the estimator by applying the Central Limit Theorem [5], which yields

$$CI = [\hat{p_S}_N - z_\alpha\sigma_N, \hat{p_S}_N + z_\alpha\sigma_N], \tag{3.12}$$

where z_α is found from the tables in [13]. For a given α, we get

$$Pr(p_S \in CI) = 1 - 2\alpha \tag{3.13}$$

Chosen $\alpha = 2.5\%$ provides a 95% confidence interval

$$CI_{95} = [\hat{p_S}_N - 1.96\sigma_N, \hat{p_S}_N + 1.96\sigma_N] \tag{3.14}$$

$$Pr(p_S \in CI_{95}) = 0.95 \tag{3.15}$$

With σ_N from equation (3.9), we see that the convergence rate of the estimator is $\mathcal{O}(N^{-\frac{1}{2}})$

3.2.2 Chebychev Confidence Interval

In the project work [2] we used Chebychev's inequality to etablish a crude confidence interval for the estimator. To define this confidence interval we do not need to know anything about the system. We had that for any $\lambda > 0$,

$$Pr\left(|\widehat{p_S} - p_S| \le \lambda\sqrt{\frac{p_S(1-p_S)}{n}}\right) = 1 - Pr\left(|\widehat{p_S} - p_S|) > \lambda\sqrt{\frac{p_S(1-p_S)}{n}}\right),$$
$$\ge 1 - \frac{1}{\lambda^2}$$

To make a 95% confidence interval, let $\lambda^2 = 20$.

$$Pr\left(|\widehat{p_S} - p_S| \leq \sqrt{20}\sqrt{\frac{p_S(1-p_S)}{n}}\right) \geq 1 - \frac{1}{20}$$
$$Pr\left(|\widehat{p_S} - p_S| \leq \sqrt{20}\sqrt{\frac{p_S(1-p_S)}{n}}\right) \geq 0.95.$$

The standard deviation used in the interval is maximized by setting $p_S = \frac{1}{2}$. By doing this simplification the specified confidence interval is maximized, and it only depends on the size of n. We then get with probability 95%

$$|\widehat{p_S} - p_S| \leq \sqrt{\frac{20(\frac{1}{4})}{n}}$$
$$|\widehat{p_S} - p_S| \leq \sqrt{\frac{5}{n}},$$

and we can define confidence intervals

$$Pr\left(\widehat{p_S} - \sqrt{\frac{5}{n}} \leq p_S \leq \widehat{p_S} + \sqrt{\frac{5}{n}}\right) > 0.95, \tag{3.16}$$

3.2.3 Precision Results

The table 3.1 from the project assignment [2] shows precision results of Monte Carlo simulations of a series structure for different values n. The series structure consists of three independent components, each with probability to fail, $q_1 = q_2 = q_3 = 10^{-7}$. This will give a theoretical probability to fail from equation (2.9) $p_F \approx 3 \cdot 10^{-7}$.

Table 3.1: Comparison of Monte Carlo simulation for different n

n	Fails	$p_{F\,n}$	Difference
10^4	0	0	+ 100 %
10^6	0	0	+ 100 %
10^7	4	$4 \cdot 10^{-7}$	+ 33.3 %
10^8	27	$2.7 \cdot 10^{-7}$	- 10.0 %
10^9	314	$3.14 \cdot 10^{-7}$	+ 4.67 %
10^{10}	2945	$2.945 \cdot 10^{-7}$	- 1.83 %
10^{11}	30195	$3.02 \cdot 10^{-7}$	+0.65 %
10^{12}	299823	$2.998 \cdot 10^{-7}$	-0.059 %

3.3 Curve fitting

3.3.1 Parametrization

Since Monte Carlo simulation has a slow convergence rate, we would like to parametrize the stochastic variables defined in equation (2.8), (2.15) and (2.17). Table 3.1 shows ut that increasing the sample size, n gives good precision results. The simulation results where $n > q = 10^a, a > 2$ have relative difference less than 5% from the analytical solution. The idea behind the parametrization is to investigate the system for different failure probabilities. We want to increase the failure probabilities for each component in order to take advantage of the robustness of the method. When the failure rate increases, we need fewer simulations to get a descent result from Monte Carlo simulations. The goal is that it should be possible to fit a curve from the simulation results when $0 < \lambda \leq 1$, and draw a conclusion of the original system reliability, which is obtained for $\lambda = 1$.

Independent Systems

The parametrization for independent systems is copied from the project work [2].

Define a parametrized stochastic variable, $X_{i,\lambda}$ as

$$X_{i,\lambda} : \begin{cases} \Pr(X_{i,\lambda} = 1) &= p_{i,\lambda} &= 1 - (1 - p_i)10^{z_i(1-\lambda)} \\ \Pr(X_{i,\lambda} = 0) &= 1 - p_{i,\lambda} &= (1 - p_i)10^{z_i(1-\lambda)} \end{cases} \tag{3.17}$$

where

$$0 < \lambda \leq 1. \tag{3.18}$$

By inserting the expression for p_i and $1 - p_i$ from equation (2.8), this can be simplified to

$$X_{i,\lambda} : \begin{cases} \Pr(X_{i,\lambda} = 1) &= p_{i\lambda} &= 1 - 10^{-z_i\lambda} \\ \Pr(X_{i,\lambda} = 0) &= 1 - p_{i,\lambda} &= 10^{-z_i\lambda} \end{cases} \quad 0 < \lambda \leq 1. \tag{3.19}$$

This enables us to investigate how the system responds to different failure probabilities. By inserting $\lambda = 1$ to equation (3.19), we get

$$X_{i,\lambda=1} : \begin{cases} \Pr(X_{i,\lambda=1} = 1) &= 1 - 10^{-z_i} = p_i \\ \Pr(X_{i,\lambda=1} = 0) &= 10^{-z_i} = 1 - p_i \end{cases} \tag{3.20}$$

which is the same stochastic variable as we defined in equation (2.8). When λ goes to zero we get the limit

$$X_{i,\lambda\to 0} : \begin{cases} \Pr(X_{i,\lambda\to 0} = 1) &= 1 - 10^{-0z_i} = 0 \\ \Pr(X_{i,\lambda\to 0} = 0) &= 10^{-0z_i} = 1 \end{cases} \tag{3.21}$$

Common cause failures

The parametrized stochastic variable, $X_{i,\lambda}$ for common cause failures becomes

$$X_{i,\lambda}: \begin{cases} \Pr(X_{i,\lambda}=1) = (1-q_{i,\lambda}^{(i)})(1-q_{i,\lambda}^{(c)}) \\ \qquad\qquad = (1-10^{-z_i\lambda}) \prod_{j=1,j\neq i}^{s} (1-\beta_{ij}(1-10^{-z_j\lambda})) \\ \Pr(X_{i,\lambda}=0) = 1-(1-q_{i,\lambda}^{(i)})(1-q_{i,\lambda}^{(c)}) \\ \qquad\qquad = 1-(1-10^{-z_i\lambda}) \prod_{j=1,j\neq i}^{s} (1-\beta_{ij}(1-10^{-z_j\lambda})) \end{cases} \tag{3.22}$$

where

$$0 < \lambda \leq 1. \tag{3.23}$$

By inserting $\lambda = 1$ in (3.22), we get

$$X_{i,\lambda=1}: \begin{cases} \Pr(X_{i,\lambda=1}=1) = (1-q_{i,\lambda=1}^{(i)})(1-q_{i,\lambda=1}^{(c)}) \\ \qquad\qquad = (1-10^{-z_i}) \prod_{j=1,j\neq i}^{s} (1-\beta_{ij}(1-10^{-z_j})) \\ \Pr(X_{i,\lambda=1}=0) = 1-(1-q_{i,\lambda=1}^{(i)})(1-q_{i,\lambda=1}^{(c)}) \\ \qquad\qquad = 1-(1-10^{-z_i}) \prod_{j=1,j\neq i}^{s} (1-\beta_{ij}(1-10^{-z_j})) \end{cases} \tag{3.24}$$

which is the same stochastic variable as we defined in (2.15). When λ goes to zero we get the limit

$$X_{i,\lambda\to 0}: \begin{cases} \Pr(X_{\lambda\to 0}=1) = (1-q_{i,\lambda\to 0}^{(i)})(1-q_{i,\lambda\to 0}^{(c)}) \\ \qquad\qquad = (1-10^{-0\cdot z_i}) \prod_{j=1,j\neq i}^{s} (1-\beta_{ij}(1-10^{-0\cdot z_j})) = 0 \\ \Pr(X_{\lambda\to 0}=0) = 1-(1-q_{i,\lambda\to 0}^{(i)})(1-q_{i,\lambda=1}^{(c)}) \\ \qquad\qquad = 1-(1-10^{-0\cdot z_i}) \prod_{j=1,j\neq i}^{s} (1-\beta_{ij}(1-10^{-0\cdot z_j})) = 1 \end{cases} \tag{3.25}$$

Cascading failures

The parametrization of the stochastic variable, $X_{i,\lambda}(t,\mathbf{X_{-i}})$, for cascading failures becomes

$$X_{i,\lambda}(t,\mathbf{X_{-i}}): \begin{cases} \Pr(X_{i,\lambda}(t,\mathbf{X}_{-i})=1) &=& p_{i,\lambda}(t,\mathbf{X}_{-i}) &= 1-10^{-z_i\lambda(t,\mathbf{X}_{-i})} \\ \Pr(X_{i,\lambda}(t,\mathbf{X}_{-i})=0) &=& 1-p_{i,\lambda}(t,\mathbf{X}_{-i}) &= 10^{-z_i\lambda(t,\mathbf{X}_{-i})}. \end{cases} \tag{3.26}$$

where

$$0 < \lambda \leq 1. \tag{3.27}$$

By inserting $\lambda = 1$ in (3.26), we get

$$X_{i,\lambda=1}(t, \mathbf{X_{-i}}) : \begin{cases} \Pr(X_{i,\lambda=1}(t, \mathbf{X}_{-i}) = 1) = 1 - 10^{-z_i(t,\mathbf{X}_{-i})} = p_i(t, \mathbf{X}_{-i}) \\ \Pr(X_{i,\lambda=1}(t, \mathbf{X}_{-i}) = 0) = 10^{-z_i\lambda(t,\mathbf{X}_{-i})} \quad = 1 - p_i(t, \mathbf{X}_{-i}) \end{cases} \tag{3.28}$$

which is the same stochastic variable as we defined in equation (2.17). When λ goes to zero we get the limit

$$X_{i,\lambda\to 0}(t, \mathbf{X_{-i}}) : \begin{cases} \Pr(X_{i,\lambda\to 0}(t, \mathbf{X}_{-i}) = 1) = 1 - 10^{-0\cdot z_i(t,\mathbf{X}_{-i})} = 0 \\ \Pr(X_{i,\lambda\to 0}(t, \mathbf{X}_{-i}) = 0) = 10^{-0\cdot z_i(t,\mathbf{X}_{-i})} \quad = 1 \end{cases} \tag{3.29}$$

The results from simulations of a parametrized system is shown in figure 3.2. The system is the dependent system with cascading failures, defined in section 3.4.2. Since the range of the estimated probability of failure, $\hat{p_F}$, is from 0.14 to 10^{-7}, we use logarithmic y-axis to present the results. The original system is obtained for $\lambda = 1$, and the behaviour of the $\log(\hat{p_F}(\lambda))$ seems to be close to linear. The estimations of $\hat{p_F}(\lambda)$ are calculated by $n = 10^8$ samples for each λ. The number of fails in the different simulations is given in equation (3.30),

$$\begin{bmatrix} 0.2 \\ 0.289 \\ 0.378 \\ 0.467 \\ 0.556 \\ 0.644 \\ 0.733 \\ 0.822 \\ 0.911 \\ 1 \end{bmatrix} \quad \text{fails}(\lambda) = \quad \begin{bmatrix} 14101098 \\ 5270988 \\ 1921357 \\ 694723 \\ 249901 \\ 89568 \\ 32338 \\ 11677 \\ 4122 \\ 1489 \end{bmatrix} \tag{3.30}$$

This parametrization was done by a relatively large sample size, $n = 10^8$. By decreasing the sample size to $n = 10^5$, the number of fails when $\lambda \to 1$ will be 0, but we will have good estimates for $\hat{p_F}$ for the small values of λ. We want to use these good estimates to predict how the system will behave for the values of λ with no fails.

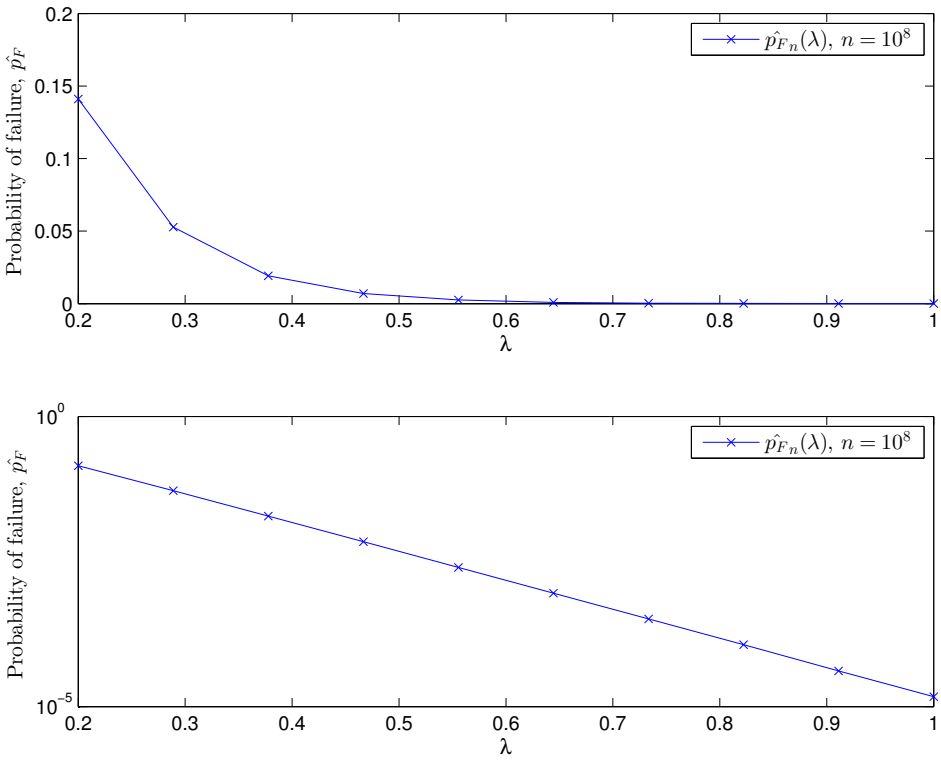


Figure 3.2: Simulated probability failure, $\hat{p_{F_n}}$, as a function of λ. Simulations are done with $n = 10^8$ for the model described in section 3.4.2. and $p = 1 - 10^{-7}$ for each component. Parametrized system, so the original system is obtained for $\lambda = 1$. The logarithmic plot shows a behaviour that looks quite linear.

3.3.2 Least squares fit

This section is copied from the project assignment [2] for completeness. When we obtain results for a given system from the different values of λ in the parametrization, we want to fit a curve to these results in order to obtain the probability of failure for the non-parametrized system. To do this curve fitting, we do $m = 10$ simulations of size n for each value of λ. From the mean of each simulation we fit a curve by using minimization of least squares. We investigate different families of functions from the form

$$\tilde{p_F} = 10^{a(b+\lambda)^c+d}, \tag{3.31}$$

where $\tilde{p_F}$ denotes the probability of failure defined in (2.16) and a, b, c and d are parameters in $\mathbb{R}$. The standard minimization of least squares can be written as

$$\min_{a,b,c,d} \frac{1}{2} \sum_{i}^{l} (f(x_i) - y_i)^2,$$

where y_i is the observation from the simulations for the different l values of λ, and $f(x_i)$ is the family of functions we would like to fit the model for. For our family of functions (3.31), we would get the following minimization problem

$$\min_{a,b,c,d} \frac{1}{2} \sum_{i}^{l} (10^{a(b+\lambda_i)^c+d} - y_i)^2.$$

To simplify the calculations and make it easier to visualize the results, we choose to do the minimization of the logarithmic model. The transformed family of functions is

$$\log_{10}(10^{a(b+\lambda)^c+d}) = a(b+\lambda)^c + d,$$

and the minimization becomes

$$\min_{a,b,c,d} \frac{1}{2} \sum_{i}^{l} (a(b+\lambda_i)^c + d - \log_{10}(y_i))^2. \tag{3.32}$$

We want to investigate three cases of the family of functions (3.31). If $c = 1$, the family of functions can be written as a family of linear functions,

$$\log_{10}(\tilde{p_{F\,L}}(\lambda)) = a\lambda + b. \tag{3.33}$$

If $c = 2$, the family of functions can be written as a family of quadratic functions,

$$\log_{10}(\tilde{p_{F\,Q}}(\lambda)) = a\lambda^2 + b\lambda + c. \tag{3.34}$$

The last case is where c is not fixed, and we get four parameters

$$\log_{10}(\tilde{p_{F\,G}}(\lambda)) = a(b+\lambda)^c + d. \tag{3.35}$$

3.3.3 Weighted fit

The intention with the weighting is as described in the project assignment [2], but three additional weighting strategies are added and investigated. To make a more accurate result, the different values for λ are given different weights in the minimization. We assign large weights to the most describing values of λ and small weights to the least describing values of λ. With an unweighted fit the simulations with $\lambda > 0.5$ often force an unnatural behaviour of the fitted curve at the tail. When weights are added to our problem, we want to minimize the following:

$$\min_{a,b,c,d} \frac{1}{2} \sum_{i}^{l} w_i(a(b+\lambda_i)^c + d - \log_{10}(y_i))^2, \tag{3.36}$$

where w is the proportion each value of λ should be weighted. We want to assign large weights to some λs so the most significant results have larger weights.

To make the minimization, the MATLAB function `lsqcurvefit` [12] is used. This function find coefficients x that for given input data $xdata$ and observations $ydata$ best fit the equation

$$\min_x \frac{1}{2}\sum_i (F(x, xdata_i) - ydata_i)^2. \tag{3.37}$$

If we define the inline function in MATLAB to be

$$F \;=\; @(x, xdata_i) \quad \sqrt{w_i}(x(1)(x(2) + xdata_i)^{x(3)} + x(4)), \tag{3.38}$$

where

$$\mathbf{x} = \begin{bmatrix} a & b & c & d \end{bmatrix}$$

To get the minimization on the same form as (3.36), we need to send in the vector $ydata = \sqrt{w}\log_{10}(y)$ to the MATLAB call in (3.37) The minimization procedure `lsqcurvefit` uses one of two optimization methods: Trust region method or Levenberg-Marquardt. We will only use the Trust region method as this gives the most accurate results. When the initial point is far away from the solution, the trust region method will move faster to the desired area than Levenberg-Marquardt [1]. We need to define the initial point [a,b,c,d], which is set to [0,0] for the linear fit, [0,0,0] for the quadratic fit and [0,0,1,0] for the general fit.

Coefficient of variance, weightN

One way to represent the weights is by the inverse logarithmic difference of the coefficient of variance for the different λs. The coefficient of variance of our Bernoulli trials may be written as

$$CV(\lambda) = \frac{1 - \overline{\hat{p_F}}_n(\lambda)}{n\overline{\hat{p_F}}_n(\lambda)}, \tag{3.39}$$

where $\overline{\hat{p_F}}_n(\lambda)$ is the mean over the $m = 10$ samples of the simulated probability to fail, $\hat{p_F}_n(\lambda)$ and n is the sample size. The coefficient of variance describes the proportion of variation relative to the sample mean. By constructing a 95 % confidence interval for the coefficient of variance, we get the following representation for the different λs.

$$\mathrm{CI}(\lambda) = \overline{\hat{p_F}}_n(\lambda)(1 \pm 1.96\sqrt{CV(\lambda)}) \tag{3.40}$$

Then the weights can be defined as

$$w(\lambda) = \frac{1}{(\log_{10}(\mathrm{CI}_+(\lambda)) - \log_{10}(\mathrm{CI}_-(\lambda)))^2}, \tag{3.41}$$

where

$$\mathrm{CI}_+(\lambda) = \overline{\hat{p_F}}_n(1 + 1.96CV(\lambda))$$
$$\mathrm{CI}_-(\lambda) = \overline{\hat{p_F}}_n(1 - 1.96CV(\lambda))$$

For some of the weights, $w_i(\lambda)$, $\mathrm{CI}_-(\lambda)$ is negative. This happens near the tail, where the value of $\hat{p_F}(\lambda)$ is small compared to the variance coefficient. For negative values of $\mathrm{CI}_-(\lambda)$, (3.41) is not a good measurement of the weights. To solve this, the weights are calculated, for each of the l values of λ by algorithm 1:

Algorithm 1 Assigning weights coefficient of variance

Input: $\overline{P_f}(\lambda)$, n
  $CV(\lambda) = \frac{1-\overline{P_f}(\lambda)}{nP_f}$
  $CI_+(\lambda) = \overline{P_f}(\lambda)(1 + 1.96\sqrt{CV(\lambda)})$
  $CI_-(\lambda) = \overline{P_f}(\lambda)(1 - 1.96\sqrt{CV(\lambda)})$
  k=0
  for i=1 to length of $\overline{P_f(\lambda)}$ **do**
    if $\overline{P_f}(i) > 0$ **then**
      k = k+1
      if $CI_-(i) > 0$ **then**
        $w_i(k) = \frac{1}{(\log_{10}(CI_+(i))-\log_{10}(CI_-(i)))^2}$
      else
        $w_i(k) = \frac{w_i(k-1)}{(\overline{P_f}(k-1)/\overline{P_f}(k))}$
      end if
    end if
  end for
  Output: $w_i = \frac{w_i}{sum(w_i)}$

By using this weighting algorithm, we adjust the weights in the tails, where often $CI_- < 0$. The factor $(\overline{P_f}(k-1)/\overline{P_f}(k))$ will in most cases be between 1.5 and 6, and ensures that the simulations with few fails have less to say for the fitted curve.

Mean times inverse empirical standard deviation, weightS

Another way to assign weights is to directly divide the mean of the simulations by the empirical standard deviation,

$$w(\lambda) = \frac{\overline{fails(\lambda)}}{std(fails(\lambda))}. \tag{3.42}$$

$\overline{fails(\lambda)}$ is the mean of the vector with number of fails, $fails(\lambda)$, and $std(fails(\lambda)$ is the empirical standard deviation of the same vector.

For this weighting strategy, results with many fails will be given high weights, but will be adjusted by the empirical standard deviation from the simulations. The standard deviation is usually larger for the large values of λ. The weighting strategy must a contain a special condition for the cases where the empirical standard deviation is 0. This may be a possibility, especially for small values of λ. The weight for these values of λ is set to be the sum of the fails, $sFails(\lambda)$.

Algorithm 2 Mean times inverse

Input: $fails(\lambda)$
  $sFails(\lambda) = sum(fails(\lambda))$
  k=0
  for i=1 to length of $sFails(\lambda)$ **do**
    if $sFails(i) > 0$ **then**
      k = k+1
      if $var(fails(i)) = 0$ **then**
        $w_i(k) = sFails(i)$
      else
        $w_i(k) = \frac{\overline{fails}(i)}{std(fails(i))}$
      end if
    end if
  end for
  Output: $w_i = \frac{w_i}{sum(w_i)}$

Empirical coefficient of variance, weightN2

It is also possible to calculate the empirical coefficient of variance by dividing the empirical variance by the number of fails for the given values of λ This is the same procedure as 1, but the coefficient of variance is calculated by empirical variance and mean,

$$CV(\lambda) = \frac{var(fails(\lambda))}{sFails(\lambda)}. \tag{3.43}$$

$fails(\lambda)$ is the vector with the m fails for a given λ and $sFails(\lambda)$ is the sum of the this vector.

This weighting strategy will be more affected by variations in the m simulations for each λ than the weights in 1. The algorithm contains special cases for negative values of CI_- and when the variance of $fails(\lambda) = 0$, constructed the same way as for 1 and 2.

Algorithm 3 Assigning weights relative empirical variance

Input: $\overline{P_f}(\lambda)$, n, $fails(\lambda)$
$sFails(\lambda) = sum(fails(\lambda))$
$CV(\lambda) = \frac{var(fails(\lambda))}{sFails(\lambda)}$
$CI_+(\lambda) = \overline{P_f}(\lambda)(1 + 1.96\sqrt{CV(\lambda)})$
$CI_-(\lambda) = \overline{P_f}(\lambda)(1 - 1.96\sqrt{CV(\lambda)})$
k=0
for i=1 to length of $\overline{P_f(\lambda)}$ **do**
  if $\overline{P_f}(i) > 0$ **then**
    k = k+1
    if $CI_-(i) > 0$ **then**
      $w_i(k) = \frac{1}{(\log_{10}(CI_+(i)) - \log_{10}(CI_-(i)))^2}$
    else
      $w_i(k) = \frac{w_i(k-1)}{(\overline{P_f}(k-1)/\overline{P_f}(k))}$
    end if
  end if
  if Var(fails(i)=0 **then**
    $w_i(k) = sFails(i)$
  end if
end for
Output: $w_i = \frac{w_i}{sum(w_i)}$

Inverse empirical variance, weightQ

The inverse empirical variance is the most common weighting strategy in least squares [3] It is common since it makes the uncertain results less important to the fit. The most common use is however when there is little variation in the order of the mean. The weights are calculated by

$$w(\lambda) = \frac{1}{var(fails(\lambda))}, \tag{3.44}$$

with the vector $fails(\lambda)$ as before.
This weighting strategy does not take the mean into consideration, so we expect that that the fit will do the opposite of our intentions. It may however be interesting to see how the results are affected by this.

Algorithm 4 Assigning weights inverse empirical variance

Input: $fails(\lambda)$
  $sFails(\lambda) = sum(fails(\lambda))$
  k=0
  for i=1 to length of $sFails(\lambda)$ **do**
    if $sFails(i) > 0$ **then**
      k = k+1
      if $var(sFails(\lambda)) = 0$ **then**
        $w_i(k) = \overline{fails}(i)$
      else
        $w_i(k) = \frac{1}{var(fails(i))}$
      end if
    end if
  end for
  Output: $w_i = \frac{w_i}{sum(w_i)}$

No weights

For some of the systems, we will investigate how the least squares fit is without weighting. This did not give good results in the project [2], but it will be interesting to analyse the behaviour of different systems without weighting as well.

3.4 Implementation of systems

Implementation of the different systems simulated

3.4.1 Independent components in series

A direct algorithm for simulating a series structure with s components is given in algorithm 5. The reliability block diagram of the system can be seen in figure 2.1

Algorithm 5 Calculation of system failure for s components in series

Input: n,s,**p**
Require: n,$s \in \mathbb{N}^+$, $0 < p(j) < 1, \quad j = 1, 2, \ldots, s$
  A = Generate matrix of $(n \times s)$ random variables $\rightsquigarrow \mathcal{U}(0,1)$
  fails = 0
  for $i = 1$ to n **do**
    for $j = 1$ to s **do**
      if $\mathbf{A}(i,j) > p(j)$ **then**
        $fails = fails + 1$
        break
      end if
    end for
  end for
  Output: $\hat{p_F} = \frac{fails}{n}$

n is the sample size, s is the number of components in series and **p** is a vector of length s with the probability to fail for each component $1, 2, \ldots s$.

This system was used to generate all the independent results given in section 4.1

3.4.2 Cascading failure on 2003-system

Consider the 2003-system in figure 3.3. Let the components be defined by the stochastic variable in equation (2.17). The system can represent a case where the components each share a common load. When one of the components fail, the other components need to take a larger share of the load. The implementation of the system from figure 3.3 is given in algorithm 6. The system is functioning when 2 components are functioning. When the first components in the system fail, the probability to fail for the two other components increase with 50%. The component that failed remains failed until it gets repaired. In the implemented system, the components only get repaired when the system has failed. That is, when 2 or 3 of the components are not functioning.

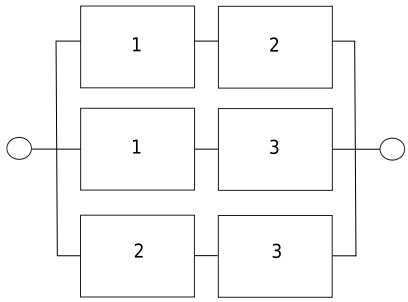


Figure 3.3: Logarithmic plot of the simulated probability failure, P_f, as a function of λ.

Consider the following four states of the system:

3. Three components functioning - System functioning
2. Two components functioning - System functioning
1. One component functioning - System failed
0. None components functioning - System failed

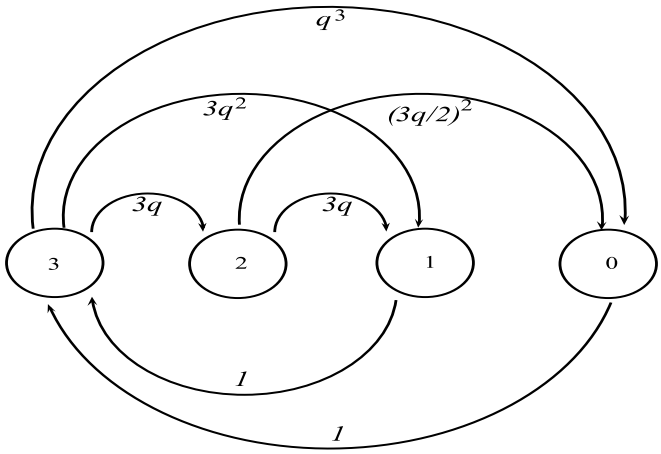


Figure 3.4: Logarithmic plot of the simulated probability failure, P_f, as a function of λ. Simulations are done in 10 samples with $n = 10^4$ for a structure with 1 component and $p_1 = 1 - 10^{-7}$. Original system is obtained for $\lambda = 1$.

Since the transition probabilities are fixed, we can define the Markov chain of the system, given in figure 3.4. From the Markov Chain, we see that all the components gets repaired at the moment the system fails. We have included the probabilities that two components fail simultaneously, but when we have $q \approx 10^{-7}$,

Algorithm 6 Calculation of system failure of the described 2003-system

Input: n,**p**
Require: $n, \in \mathbb{N}^+$, $0 < p(j) < 1, \quad j = 1, 2, 3$
  $\mathbf{A}$ = Generate matrix of $(n \times 3)$ random variables $\rightsquigarrow \mathcal{U}(0, 1)$
  $fails = 0$
  $pfailed = zeros(3, 1)$
  $\mathbf{p_s}$=**p**
  for $i = 1$ to n **do**
    if $(\mathbf{A}(i, 1) > p_s(1)$ **and** $A(i, 2) > p_s(2))$ **or**
    $(\mathbf{A}(i, 1) > p_s(1)$ **and** $A(i, 3) > p_s(3))$ **or**
    $(\mathbf{A}(i, 2) > p_s(2)$ **and** $A(i, 3) > p_s(3))$ **then**
      $fails = fails + 1$
    end if
    if $\mathbf{A}(i, 1) > p_s(1)$ **and** $pfailed(1) == 0$ **then**
      $p_s(2) = 1 - (1 - p_s(2)) \cdot 3/2$
      $p_s(3) = 1 - (1 - p_s(3)) \cdot 3/2$
      $p_s(1) = 0$
      $pfailed(1) = 1$
    end if
    if $\mathbf{A}(i, 1) > p_s(2)$ **and** $pfailed(2) == 0$ **then**
      $p_s(1) = 1 - (1 - p_s(1)) \cdot 3/2$
      $p_s(3) = 1 - (1 - p_s(3)) \cdot 3/2$
      $p_s(2) = 0$
      $pfailed(2) = 1$
    end if
    if $\mathbf{A}(i, 1) > p_s(3)$ **and** $pfailed(3) == 0$ **then**
      $p_s(1) = 1 - (1 - p_s(1)) \cdot 3/2$
      $p_s(2) = 1 - (1 - p_s(2)) \cdot 3/2$
      $p_s(3) = 0$
      $pfailed(3) = 1$
    end if
  end for
  Output: $\hat{p_F} = \frac{fails}{n}$

q^2 and q^3 is negligible. When we do this simplification the last state is removed. This allows us to define a new, shorter Markov chain with following states:

2. Three components functioning - System functioning

1. Two components functioning - System functioning

0. One component functioning - System failed

This Markov chain is shown in figure 3.5. This Markov chain is irreducible and ergodic, and we can use theorem 2.6.1 to find the limiting probabilities.

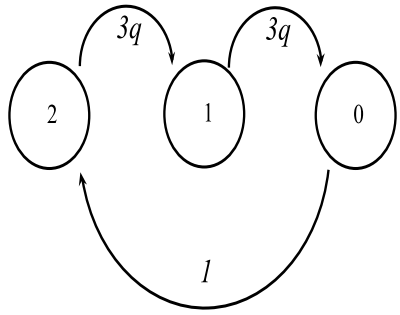


Figure 3.5: Logarithmic plot of the simulated probability failure, P_f, as a function of λ. Simulations are done in 10 samples with $n = 10^4$ for a structure with 1 component and $p_1 = 1 - 10^{-7}$. Original system is obtained for $\lambda = 1$.

Let $\mathbf{P}$ be the one step transition probability matrix introduced in equation (2.20), The original 4 state matrix from figure 3.4 is

$$\mathbf{P} = \begin{bmatrix} 1-3q-3q^2-q^3 & 3q & 3q^2 & q^3 \\ 0 & 1-3q-(\frac{3q}{2})^2 & 3q & (\frac{3q}{2})^2 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}, \tag{3.45}$$

which is simplified to

$$\mathbf{P} = \begin{bmatrix} 1-3q & 3q & 0 \\ 0 & 1-3q & 3q \\ 1 & 0 & 0 \end{bmatrix} \tag{3.46}$$

when we neglect q^2 and q^3. The system of equations for the limiting probabilities is given in equations (3.47) to (3.50).

$$\pi_2 = (1-3q)\pi_2 + \pi_0 \tag{3.47}$$
$$\pi_1 = 3q\pi_2 + (1-3q)\pi_1 \tag{3.48}$$
$$\pi_0 = 3q\pi_1 \tag{3.49}$$
$$\pi_2 + \pi_1 + \pi_0 = 1 \tag{3.50}$$

Equation (3.48) gives

$$\Pi_1 = \Pi_2,$$

and by combining equation (3.49) and (3.50)

$$2\pi_1 + \pi_0 = 1$$
$$\Big(\frac{2}{3q} + 1\Big)\pi_0 = 1$$
$$\pi_0 = \frac{1}{\frac{2}{3q} + 1}$$
$$\pi_0 = \frac{q}{q + \frac{2}{3}}$$

Since 0 was the only state, where the system is not functioning, the probability to fail for the system is

$$p_F = \pi_0 = \frac{q}{q + \frac{2}{3}} \tag{3.51}$$

3.4.3 Cascading failure on two 2003-systems and three independent components in series.

This system is on the same form as figure 2.3, where the 2003-sub systems are identical to the 2003 system defined in figure 3.3. The other three components in the system act independently. This system is also possible to monitor by Markov chains, to get an analytical solution for the probability to fail, p_F. Let p_4, p_5 and p_9 be the reliability for the three independent components in series, 4,5 and 9. The probability of system failure, p_F, for this system can be expressed by

$$p_F = 1 - (1 - \pi_0)_1(1 - \pi_0)_2(p_4)(p_5)(p_6), \tag{3.52}$$

where $(1 - \pi_0)_1$ is the reliability of the first 2003-sub system and $(1 - \pi_0)_2$ the reliability of the second.

3.4.4 Cascading failures with repair interval combined in series

The reliability block diagram for this system is shown in figure 3.6. The single components, 3 and 6 are independent, but the other four components are implemented with dependencies. When one of the dependent components fail, it is taken out of the system until it is repaired. The dependent components 1 and 2 are repaired simultaneously when both fail, and when at least one of the two components have been functioning for $n = \frac{1}{1-p}$ runs. The dependent components 4 and 5 are only repaired when both of them have failed. The implementation is given in algorithm 7.

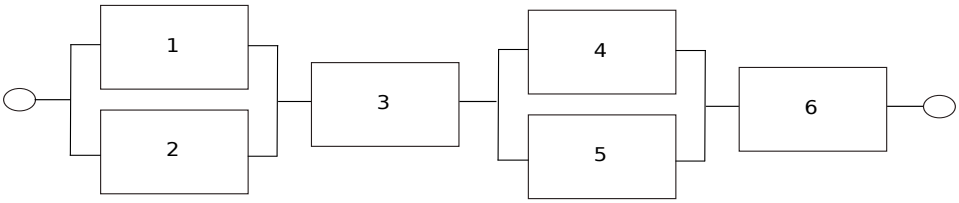


Figure 3.6: Logarithmic plot of the simulated probability failure, P_f, as a function of λ. Simulations are done in 10 samples with $n = 10^4$ for a structure with 1 component and $p_1 = 1 - 10^{-7}$. Original system is obtained for $\lambda = 1$.

Algorithm 7 Cascading failures with repair interval combined in series

Input: n,**p**
Require: $n, \in \mathbb{N}^+$, $0 < p(j) < 1, \quad j = 1, 2, \ldots, 6$
  A = Generate matrix of $(n \times 6)$ random variables $\rightsquigarrow \mathcal{U}(0,1)$
  $fails = 0$
  $j = 0$
  $\mathbf{p_s}$=**p**
  for $i = 1$ to n **do**
    j = j+1
    if $\mathbf{A}(i,3) > p_s(3)$ **or** $A(i,6) > p_s(6)$ **then**
      $fails = fails + 1$
    else
      if $\mathbf{A}(i,1) > p_s(1)$ **and** $A(i,2) > p_s(2)$ **then**
        $fails = fails + 1$
        $j = 0$
      else
        if $\mathbf{A}(i,4) > p_s(4)$ **and** $A(i,5) > p_s(5)$ **then**
          $fails = fails + 1$
        end if
      end if
    end if
    for $q = 1, 2, 4, 5$ **do**
      if $\mathbf{A}(i,q) > p_s(q)$ **then**
        $p_s(q) = 0$
      end if
      if $\mathrm{mod}(\mathrm{j}, \frac{1}{1-p(1)}) == 0$ **then**
        $p_s(1) = p(1)$
        $p_s(2) = p(2)$
      end if
      if $p_s(4) == 0$ **and** $p_s(5) == 0$ **then**
        $p_s(4) = p(4)$
        $p_s(5) = p(5)$
      end if
    end for
  end for
  Output: $\hat{p_F} = \frac{fails}{n}$

3.4.5 Cascading failures in parallel structure with repair interval

The reliability block diagram for this system is shown in figure 3.7, where all the components needs to fail for the system to fail. The model is implemented by removing the components that fails. The repair interval is unique for all the components, $\frac{1}{1-p(i)}$ and corresponds to the components expected time to fail. If the system fails, all components are repaired simultaneously. The algorithm is given in 8.

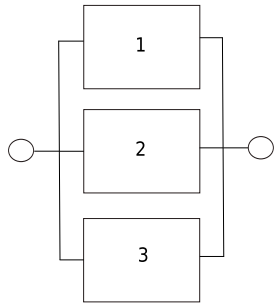


Figure 3.7: Logarithmic plot of the simulated probability failure, P_f, as a function of λ. Simulations are done in 10 samples with $n = 10^4$ for a structure with 1 component and $p_1 = 1 - 10^{-7}$. Original system is obtained for $\lambda = 1$.

3.4.6 Common cause failure

The common cause failures are introduced by the stochastic variable in equation (2.15). We introduced the factor β_{ij} that describes the probability for component i to fail if component j fails. The model consists of two parallel structures in series. Failure on a component in one of the parallels affect the probability that the two components in the other parallel fail. The components are "repaired" back to the initial state after every run. The implementation is shown in algorithm 9

Algorithm 8 Cascading failures in parallel structure with repair interval

Input: n,**p**
Require: $n, \in \mathbb{N}^+$, $0 < p(j) < 1, \quad j = 1, 2, 3$
  $\mathbf{A}$ = Generate matrix of $(n \times 3)$ random variables $\rightsquigarrow \mathcal{U}(0, 1)$
  $fails = 0$
  $j = 0$
  $\mathbf{p_s}$=**p**
  for $i = 1$ to n **do**
    $j = j + 1$
    if $(\mathbf{A}(i, 1) > p_s(1)$ **and** $A(i, 2) > p_s(2))$ **and** $A(i, 3) > p_s(3))$ **then**
      $fails = fails + 1$
      $j = 0$
    end if
    if $\mathbf{A}(i, 1) > p_s(1)$ **then**
      $p_s(1) = 0$
    end if
    if $\mathbf{A}(i, 2) > p_s(2)$ **then**
      $p_s(2) = 0$
    end if
    if $\mathbf{A}(i, 3) > p_s(3)$ **then**
      $p_s(3) = 0$
    end if
    if $\mathrm{mod}(j, \frac{1}{1-p(1)}) == 0$ **then**
      $p_s(1) = p(1)$
    end if
    if $\mathrm{mod}(j, \frac{1}{1-p(2)}) == 0$ **then**
      $p_s(2) = p(2)$
    end if
    if $\mathrm{mod}(j, \frac{1}{1-p(3)}) == 0$ **then**
      $p_s(3) = p(3)$
    end if
  end for
  Output: $\hat{p_F} = \frac{fails}{n}$

Algorithm 9 Calculation of system failure in figure

Input: n,$\mathbf{p}$,β
Require: $n, \in \mathbb{N}^+$, $0 < p(j) < 1$, $0 \leq \beta(i,j) \leq 1 \quad i,j = 1,2,3,4$
  $\mathbf{A}$ = Generate matrix of $(n \times 4)$ random variables $\rightsquigarrow \mathcal{U}(0,1)$
  $fails = 0$
  for $i = 1$ to n **do**
    $\mathbf{p_s}$=$\mathbf{p}$
    if $\mathbf{A}(i,1) > p(1)$ **then**
      if `rand(1)` $< \beta(3,1)$ **then**
        $p_s(3) = 0$
      end if
      if `rand(1)` $< \beta(4,1)$ **then**
        $p_s(4) = 0$
      end if
    end if
    if $\mathbf{A}(i,2) > p(2)$ **then**
      if `rand(1)` $< \beta(3,2)$ **then**
        $p_s(3) = 0$
      end if
      if `rand(1)` $< \beta(4,2)$ **then**
        $p_s(4) = 0$
      end if
    end if
    if $\mathbf{A}(i,3) > p(3)$ **then**
      if `rand(1)` $< \beta(1,3)$ **then**
        $p_s(1) = 0$
      end if
      if `rand(1)` $< \beta(2,3)$ **then**
        $p_s(2) = 0$
      end if
    end if
    if $\mathbf{A}(i,4) > p(4)$ **then**
      if `rand(1)` $< \beta(1,4)$ **then**
        $p_s(1) = 0$
      end if
      if `rand(1)` $< \beta(2,4)$ **then**
        $p_s(2) = 0$
      end if
    end if
    if $(A(i,1) > p_s(1)$ **and** $A(i,2) > p_s(2))$ **or**
    $(A(i,3) > p_s(3)$ or $A(i,4) > p_s(4))$ **then**
      $fails = fails + 1$
    end if
  end for
  Output: $\hat{p_F} = \frac{fails}{n}$

Chapter 4

Results

The results from all the tests is presented in this chapter. The result highlights the different precisions achieved by the different weights and fits. The results are presented in tables and figures, with the aim to describe the overall behaviour for each variant for the different models. The notation linspace(a,b,c) is used, and it describes a vector which starts at a, has b equidistant entries, and ends at c.

4.1 Independent Systems

For the independent systems, Monte Carlo simulations of the parametrized models are compared with the known analytical solution of the different systems. The simulations where we compare different properties of a given system use the same generated random variables. This makes it easier to find out how the different methods behave for similar conditions. Section 4.1.1 shows how changing the parameter λ changes the estimate for a single component system. The weighting strategy is the original, weightN. In section 4.1.2, the different weights are presented, with fixed parameters λ. The different weights are tested on a single component model. The last single component model is presented in section 4.1.3. This model is tested with a very a low probability to fail (10^{-12}). In section 4.1.4, we increase the number of components to 6, and analyse the effect of different parameters λ for two of the most interesting weights.

4.1.1 One component with different values of λ

The results presented in this section shows us that the best parameter λ for this problem is λ=linspace(0.05,10,1). The linear fit is the fit that is least affected by changing the parameter λ, which can be desrcribed with the linearity in this fit. For the quadratic fit, the starting point is more crucial for the fit, and this fit is very affected by increasing λ. The general fit is also quite stable with different λs, but the best result is achieved for λ=linspace(0.05,10,1). See figures 4.1 to 4.3 and tables 4.1 to 4.3.

WeightN, λ =linspace(0.05,10,1)

Table 4.1: Comparison of the three fits when $s = 1$, $p_1 = 1 - 10^{-7}$, $\lambda = \text{linspace}(0.05, 10, 1)$ and simulation size is 10 samples with $n = 10^4$. The least squares fit is weighted with weightN, described in algorithm 1. $\delta_{\tilde{p_F}(1)} = \frac{\tilde{p_F}(1) - p_F(1)}{p_F(1)}$, where $p_F(1) = 10^{-7}$ is the analytically calculated probability to fail.

$\log_{10}(\tilde{p_F}(\lambda))$	a	b	c	d	$\tilde{p_F}(1)$	$\delta_{\tilde{p_F}(1)}$
$a\lambda + b$	-7.0328	0.0030			$9.337 \cdot 10^{-8}$	-0.066
$a\lambda^2 + b\lambda + c$	-0.1893	-6.9773	0.0004		$6.820 \cdot 10^{-8}$	-0.310
$a(b + \lambda)^c + d$	-7.0329	0.0002	1.0011	0.0030	$9.302 \cdot 10^{-8}$	-0.070

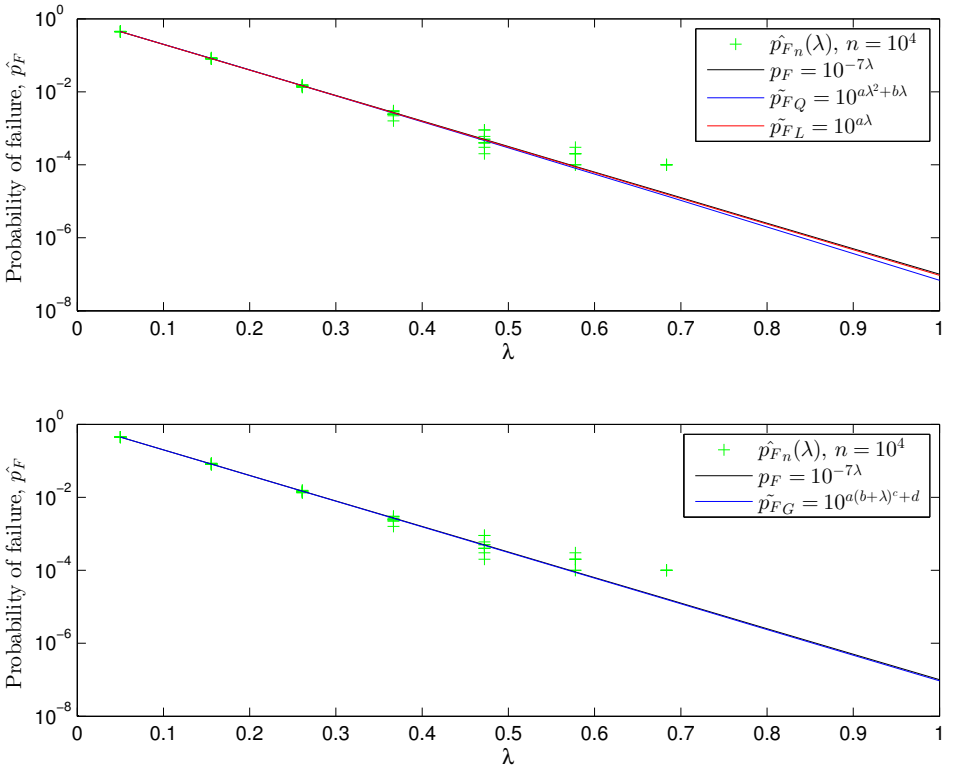


Figure 4.1: Logarithmic plot of the different fits of the simulated probability failure, $\tilde{p_F}$, as a function of λ. Simulations are done in 10 samples with $n = 10^4$ for a structure with 1 component and $p_1 = 1 - 10^{-7}$. w_i = weightN and $\lambda(1) = 0.05$. Original model is obtained for $\lambda = 1$.

WeightN, λ =linspace(0.1,10,1)

Table 4.2: Comparison of the three fits when $s = 1$, $p_1 = 1 - 10^{-7}$, $\lambda = \text{linspace}(0.1, 10, 1)$ and simulation size is 10 samples with $n = 10^4$. The least squares fit is weighted with weightN, described in algorithm 1. $\delta_{\tilde{p_F}(1)} = \frac{\tilde{p_F}(1) - p_F(1)}{p_F(1)}$, where $p_F(1) = 10^{-7}$ is the analytically calculated probability to fail.

$\log_{10}(\tilde{p_F}(\lambda))$	a	b	c	d	$\tilde{p_F}(1)$	$\delta_{\tilde{p_F}(1)}$
$a\lambda + b$	-7.0520	0.0094			$9.066 \cdot 10^{-8}$	-0.093
$a\lambda^2 + b\lambda + c$	-0.5750	-7.2818	0.0274		$2.092 \cdot 10^{-7}$	1.092
$a(b + \lambda)^c + d$	-7.0518	-0.0008	0.9966	0.0095	$9.194 \cdot 10^{-8}$	-0.081

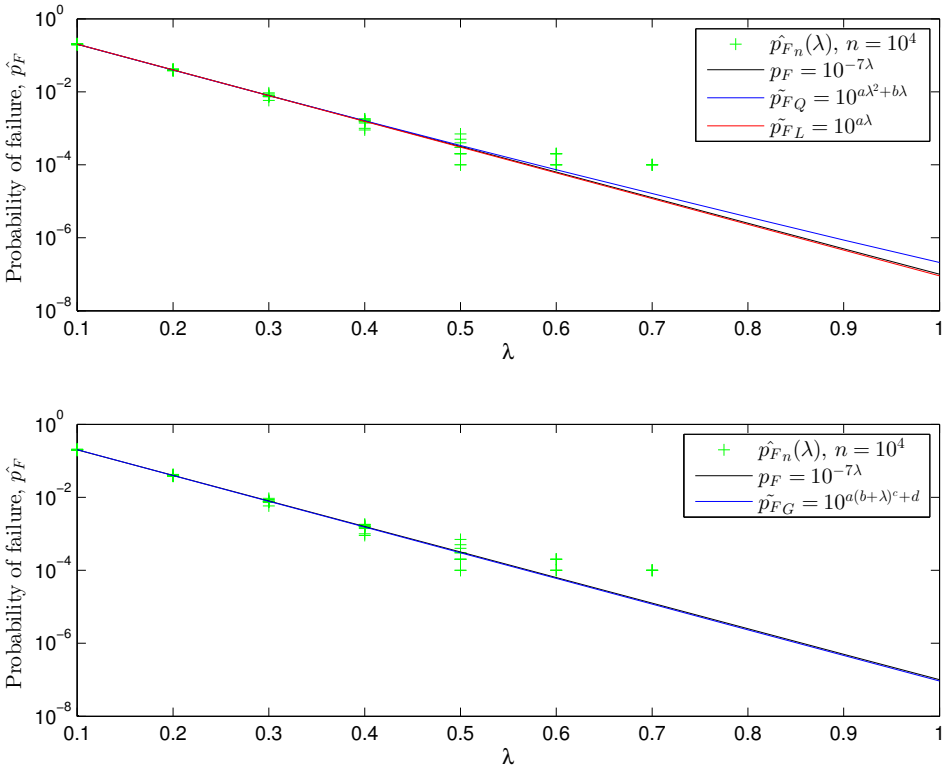


Figure 4.2: Logarithmic plot of the different fits of the simulated probability failure, $\tilde{p_F}$, as a function of λ. Simulations are done in 10 samples with $n = 10^4$ for a structure with 1 component and $p_1 = 1 - 10^{-7}$. w_i = weightN and $\lambda(1) = 0.1$. Original model is obtained for $\lambda = 1$.

WeightN, λ =linspace(0.2,10,1)

Table 4.3: Comparison of the three fits when $s = 1$, $p_1 = 1 - 10^{-7}$, $\lambda = \text{linspace}(0.2, 10, 1)$ and simulation size is 10 samples with $n = 10^4$. The least squares fit is weighted with weightN, described in algorithm 1. $\delta_{\tilde{p_F}(1)} = \frac{\tilde{p_F}(1) - p_F(1)}{p_F(1)}$, where $p_F(1) = 10^{-7}$ is the analytically calculated probability to fail.

$\log_{10}(\tilde{p_F}(\lambda))$	a	b	c	d	$\tilde{p_F}(1)$	$\delta_{\tilde{p_F}(1)}$
$a\lambda + b$	-7.0007	-0.0019			$9.805 \cdot 10^{-8}$	-0.020
$a\lambda^2 + b\lambda + c$	1.9206	-8.1389	0.1504		$8.551 \cdot 10^{-7}$	7.551
$a(b + \lambda)^c + d$	-7.0056	-0.0176	0.9457	0.0006	$1.293 \cdot 10^{-7}$	0.293

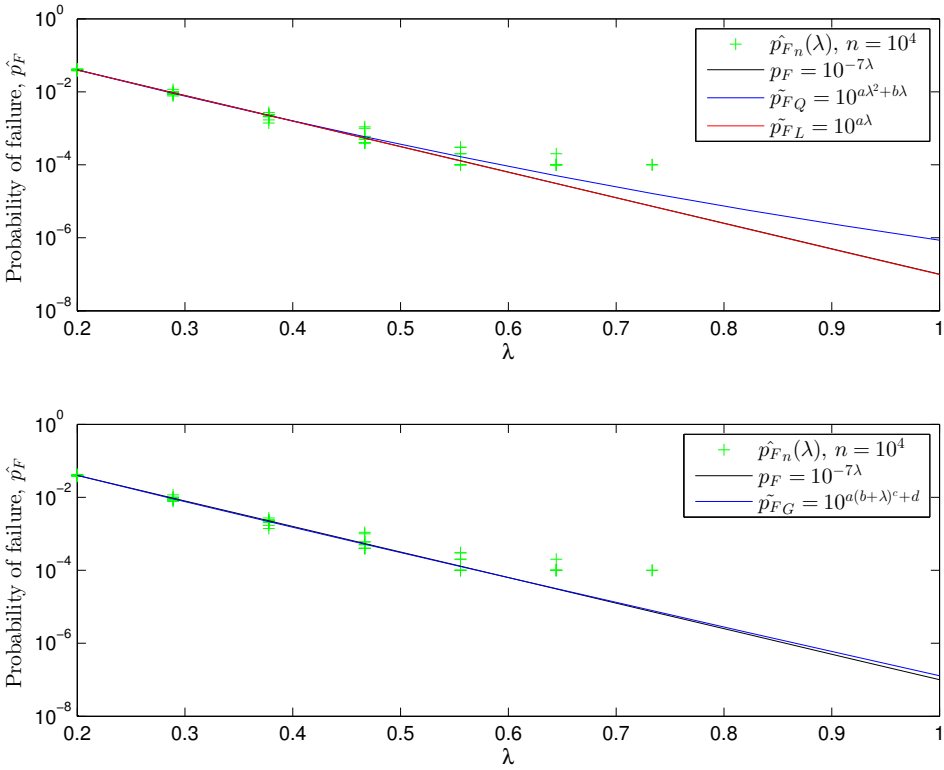


Figure 4.3: Logarithmic plot of the different fits of the simulated probability failure, $\tilde{p_F}$, as a function of λ. Simulations are done in 10 samples with $n = 10^4$ for a structure with 1 component and $p_1 = 1 - 10^{-7}$. w_i = weightN and $\lambda(1) = 0.2$. Original model is obtained for $\lambda = 1$.

4.1.2 One component with different weights

The results presented in this section shows that the linear fit with weighting weightN is the most accurate solution. From table 4.4, we see that this fit was the most accurate in 288 of 1000 cases. The weighting strategy weightN2 is also quite good, especially for the general fit. By looking at table 4.5, we see the sum of squared errors for the 1000 simulations. The linear and general fits with weighting weightN and weightN2 are best in this measurement as well. The traditional least squares weighting, weightQ gives as we expected, bad results. These weights are not relative to the mean of the simulations, and this is not beneficial for our use. Figures and tables from the tests are given in figure 4.4 to 4.7 and table 4.6 to 4.9.

Table 4.4: Sum of the combinations of fit and weighting that give an estimate, $\tilde{p_F}$, closest to the analytical solution, $p_F = 10^-7$, for 1000 realisations.

	WeightN	WeightS	WeightN2	WeightQ
Linear fit	288	72	133	9
Quadratic fit	45	33	37	4
General fit	120	88	159	12

Table 4.5: Sum of squares for each combination of fit and weighting for 1000 realisations. The sum of squares are calculated by $(\tilde{p_F} - p_F)^2$, where $p_F = 10^-7$ is the analytical solution.

	WeightN	WeightS	WeightN2	WeightQ
Linear fit	$8.10{\cdot}10^{-14}$	$1.18{\cdot}10^{-12}$	$8.26{\cdot}10^{-14}$	$2.33{\cdot}10^{-9}$
Quadratic fit	$8.70{\cdot}10^{-12}$	$2.26{\cdot}10^{-10}$	$1.20{\cdot}10^{-11}$	$1.36{\cdot}10^{-6}$
General fit	$8.41{\cdot}10^{-14}$	$6.70{\cdot}10^{-12}$	$8.59{\cdot}10^{-14}$	$3.32{\cdot}10^{-9}$

WeightS, λ =linspace(0.05,10,1)

Table 4.6: Comparison of the three fits when $s = 1$, $p_1 = 1 - 10^{-7}$, $\lambda = \text{linspace}(0.05, 10, 1)$ and simulation size is 10 samples with $n = 10^4$. The least squares fit is weighted with weightS, described in algorithm 2. $\delta_{\tilde{p_F}(1)} = \frac{\tilde{p_F}(1) - p_F(1)}{p_F(1)}$, where $p_F(1) = 10^{-7}$ is the analytically calculated probability to fail.

$\log_{10}(\tilde{p_F}(\lambda))$	a	b	c	d	$\tilde{p_F}(1)$	$\delta_{\tilde{p_F}(1)}$
$a\lambda + b$	-6.9327	-0.0078			$1.147 \cdot 10^{-7}$	0.147
$a\lambda^2 + b\lambda + c$	1.055	-7.4248	0.0239		$4.512 \cdot 10^{-7}$	3.512
$a(b + \lambda)^c + d$	-6.9277	-0.0038	0.9823	-0.0073	$1.232 \cdot 10^{-7}$	0.232

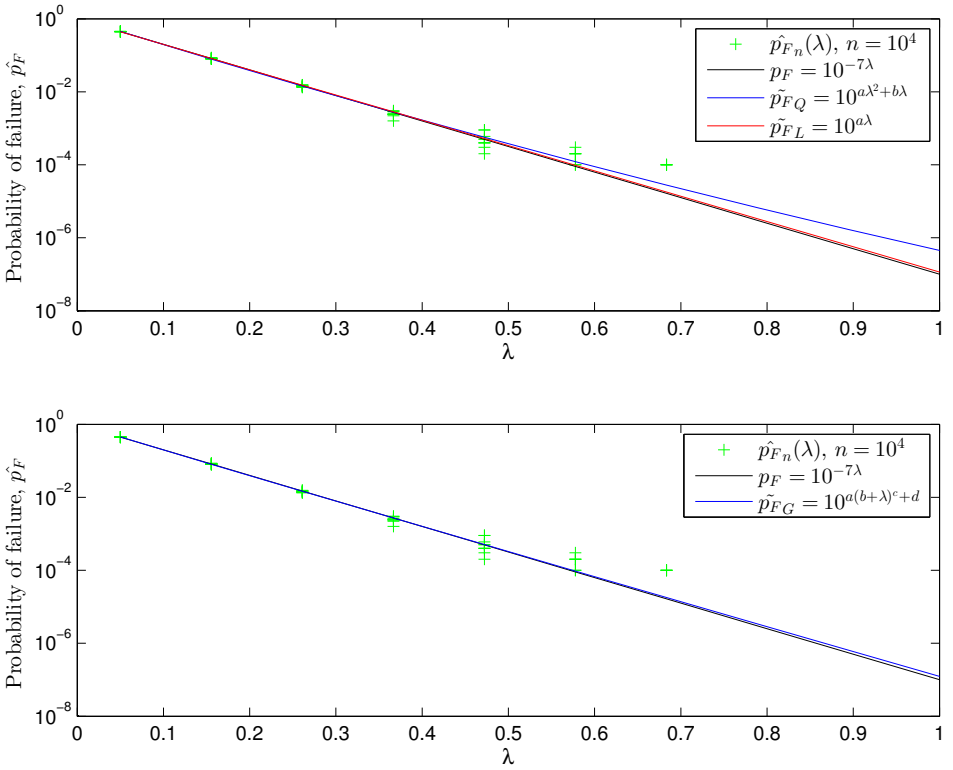


Figure 4.4: Logarithmic plot of the different fits of the simulated probability failure, $\tilde{p_F}$, as a function of λ. Simulations are done in 10 samples with $n = 10^4$ for a structure with 1 component and $p_1 = 1 - 10^{-7}$. w_i = weightS and $\lambda(1) = 0.05$. Original model is obtained for $\lambda = 1$.

WeightN2, λ =linspace(0.05,10,1)

Table 4.7: Comparison of the three fits when $s = 1$, $p_1 = 1 - 10^{-7}$, $\lambda = \text{linspace}(0.05, 10, 1)$ and simulation size is 10 samples with $n = 10^4$. The least squares fit is weighted with weightN2, described in algorithm 3. $\delta_{\tilde{p_F}(1)} = \frac{\tilde{p_F}(1) - p_F(1)}{p_F(1)}$, where $p_F(1) = 10^{-7}$ is the analytically calculated probability to fail.

$\log_{10}(\hat{p_F}(\lambda))$	a	b	c	d	$\tilde{p_F}(1)$	$\delta_{\tilde{p_F}(1)}$
$a\lambda + b$	-7.0333	0.0028			$9.3202 \cdot 10^{-8}$	-0.068
$a\lambda^2 + b\lambda + c$	-0.1166	-6.9990	0.0012		$7.682 \cdot 10^{-8}$	-0.232
$a(b + \lambda)^c + d$	-7.0334	-0.0001	1.0008	0.0027	$9.297 \cdot 10^{-8}$	-0.070

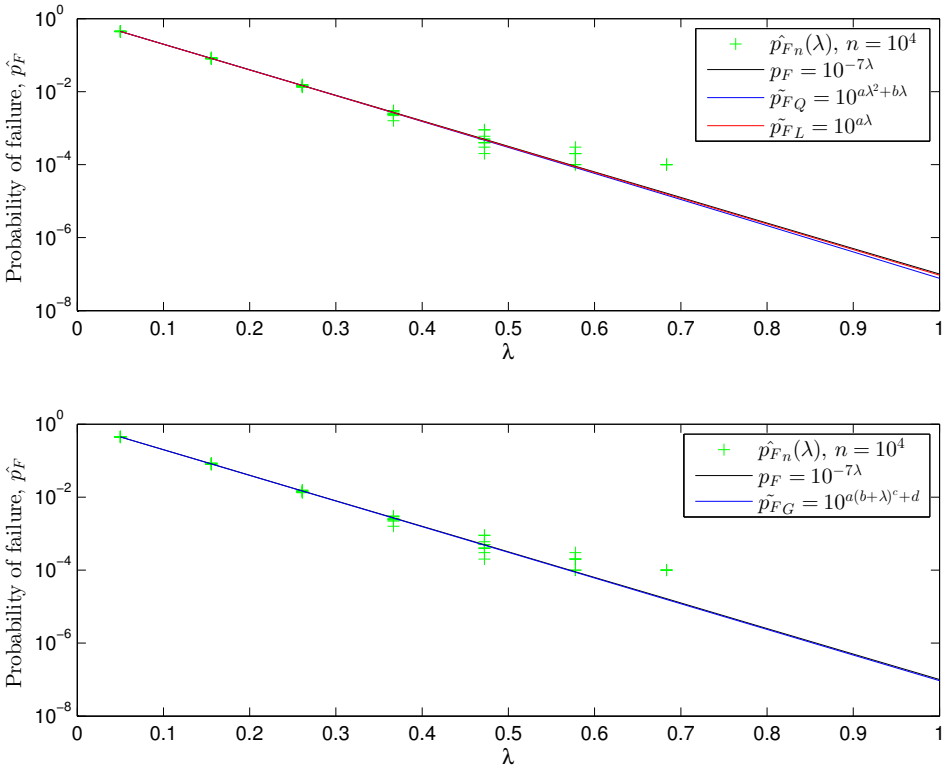


Figure 4.5: Logarithmic plot of the different fits of the simulated probability failure, $\tilde{p_F}$, as a function of λ. Simulations are done in 10 samples with $n = 10^4$ for a structure with 1 component and $p_1 = 1 - 10^{-7}$. w_i = weightN2 and $\lambda(1) = 0.05$. Original model is obtained for $\lambda = 1$.

WeightQ, λ =linspace(0.05,10,1)

Table 4.8: Comparison of the three fits when $s = 1$, $p_1 = 1 - 10^{-7}$, $\lambda = \text{linspace}(0.05, 10, 1)$ and simulation size is 10 samples with $n = 10^4$. The least squares fit is weighted with weightQ, described in algorithm 4. $\delta_{\tilde{p_F}(1)} = \frac{\tilde{p_F}(1) - p_F(1)}{p_F(1)}$, where $p_F(1) = 10^{-7}$ is the analytically calculated probability to fail.

$\log_{10}(\tilde{p_F}(\lambda))$	a	b	c	d	$\tilde{p_F}(1)$	$\delta_{\tilde{p_F}(1)}$
$a\lambda + b$	-5.1410	-0.8984			$9.133 \cdot 10^{-7}$	8.133
$a\lambda^2 + b\lambda + c$	5.7876	-11.6718	96.551		$9.755 \cdot 10^{-6}$	-0.232
$a(b+\lambda)^c + d$	-5.3780	0.0749	0.8548	-0.1581	$1.323 \cdot 10^{-6}$	12.225

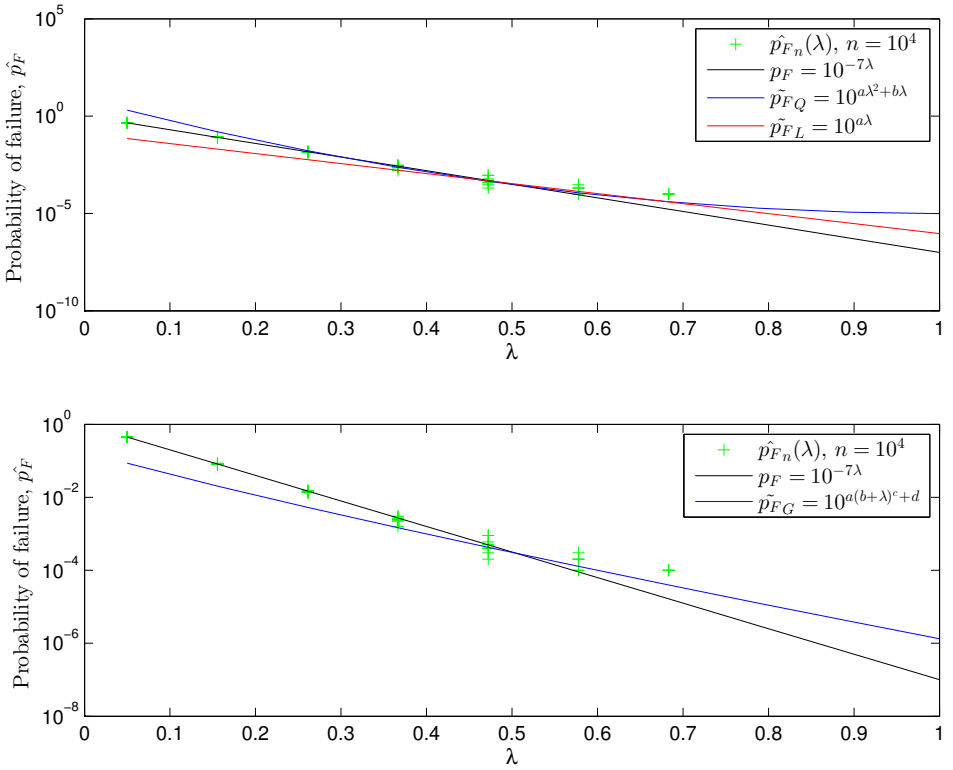
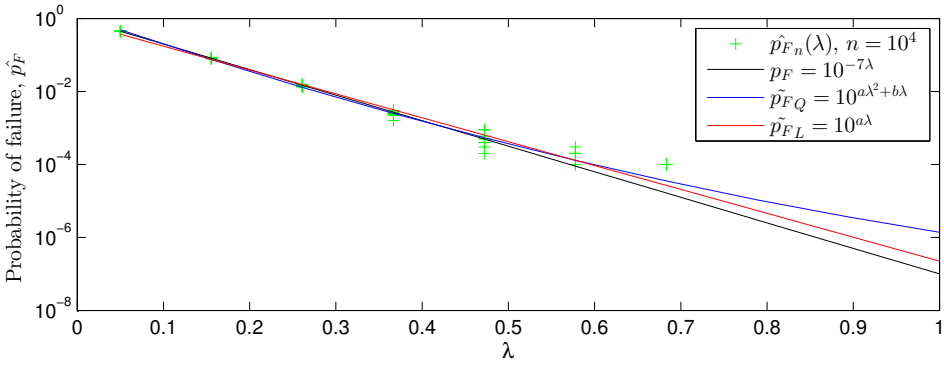


Figure 4.6: Logarithmic plot of the different fits of the simulated probability failure, $\tilde{p_F}$, as a function of λ. Simulations are done in 10 samples with $n = 10^4$ for a structure with 1 component and $p_1 = 1 - 10^{-7}$. w_i = weightQ and $\lambda(1) = 0.05$. Original model is obtained for $\lambda = 1$.

Unweighted fit, λ =linspace(0.05,10,1)

Table 4.9: Comparison of the three fits when $s = 1$, $p_1 = 1 - 10^{-7}$, $\lambda = \text{linspace}(0.05, 10, 1)$ and simulation size is 10 samples with $n = 10^4$. The least squares fit is unweighted. $\delta_{\tilde{p_F}(1)} = \frac{\tilde{p_F}(1) - p_F(1)}{p_F(1)}$, where $p_F(1) = 10^{-7}$ is the analytically calculated probability to fail.

$\log_{10}(\hat{p_F}(\lambda))$	a	b	c	d	$\tilde{p_F}(1)$	$\delta_{\tilde{p_F}(1)}$
$a\lambda + b$	-6.5437	-0.1048			$2.246 \cdot 10^{-7}$	8.133
$a\lambda^2 + b\lambda + c$	2.2191	-8.1710	0.0946		$1.389 \cdot 10^{-6}$	-0.232
$a(b + \lambda)^c + d$	-7.0567	0.2951	0.8725	2.3880	$3.509 \cdot 10^{-7}$	12.225



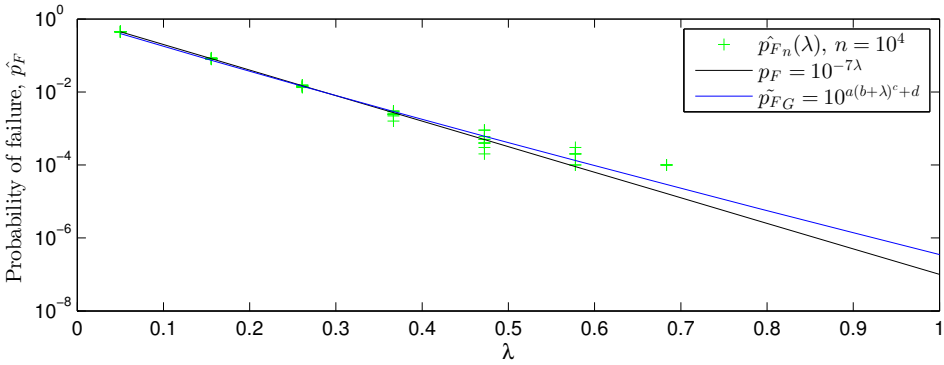


Figure 4.7: Logarithmic plot of the different fits of the simulated probability failure, $\tilde{p_F}$, as a function of λ. Simulations are done in 10 samples with $n = 10^4$ for a structure with 1 component and $p_1 = 1 - 10^{-7}$. Unweighted fit and $\lambda(1) = 0.05$. Original model is obtained for $\lambda = 1$.

4.1.3 Single component with reliability $p_1 = 1 - 10^{-12}$

This section presents the method applied on a system with a very small probability to fail. This is done to investigate how accurate simple systems like the one component system can be. The model has probability to fail, $p_F = 10^{-7}$, and results from the simulations are quite good. For the linear fit, we have obtained estimates that only differs 10% and 20% from the analytical solution. This is achieved by 10 samples with sample size $n = 10^4$ each. The results for the quadratic and general fit were not as stable as the linear, but for some of the runs, they were quite accurate as well. The results presented in figure 4.8 and table 4.10 shows a fit where the general fit was very bad. This shows that the general fit is not the most stable method when we have as extreme probabilities as 10^{12}. The quadratic fit differs with more than 100%. In figure 4.9 and table 4.11 we see that the linear fit is very good once more, and that all the fits are very accurate. It is worth noticing that the general fit in this case approximated a fit identical to the linear, with $c = 1$ and $b = 0$. All the fits are done with λ=linspace(0.05,10,1) and w_i = weightN, which gave the best results.

Realisation with a bad general fit, $\tilde{p_{FG}}$

Table 4.10: Comparison of the three fits when $s = 1$, $p_1 = 1 - 10^{-12}$, $\lambda = \text{linspace}(0.05, 10, 1)$ and simulation size is 10 samples with $n = 10^4$.The least squares fit is weighted with weightN, described in algorithm 1. $\delta_{\tilde{p_F}(1)} = \frac{\tilde{p_F}(1) - p_F(1)}{p_F(1)}$, where $p_F(1) = 10^{-12}$ is the analytically calculated probability to fail.

$\log_{10}(\tilde{p_F}(\lambda))$	a	b	c	d	$\tilde{p_F}(1)$	$\delta_{\tilde{p_F}(1)}$
$a\lambda + b$	-12.0529	0.0048			$8.952 \cdot 10^{-13}$	-0.105
$a\lambda^2 + b\lambda + c$	0.5310	-12.1707	0.0094		$2.343 \cdot 10^{-12}$	1.343
$a(b + \lambda)^c + d$	-10.0637	-0.04037	0.8740	-0.4206	$7.443 \cdot 10^{-11}$	73.428

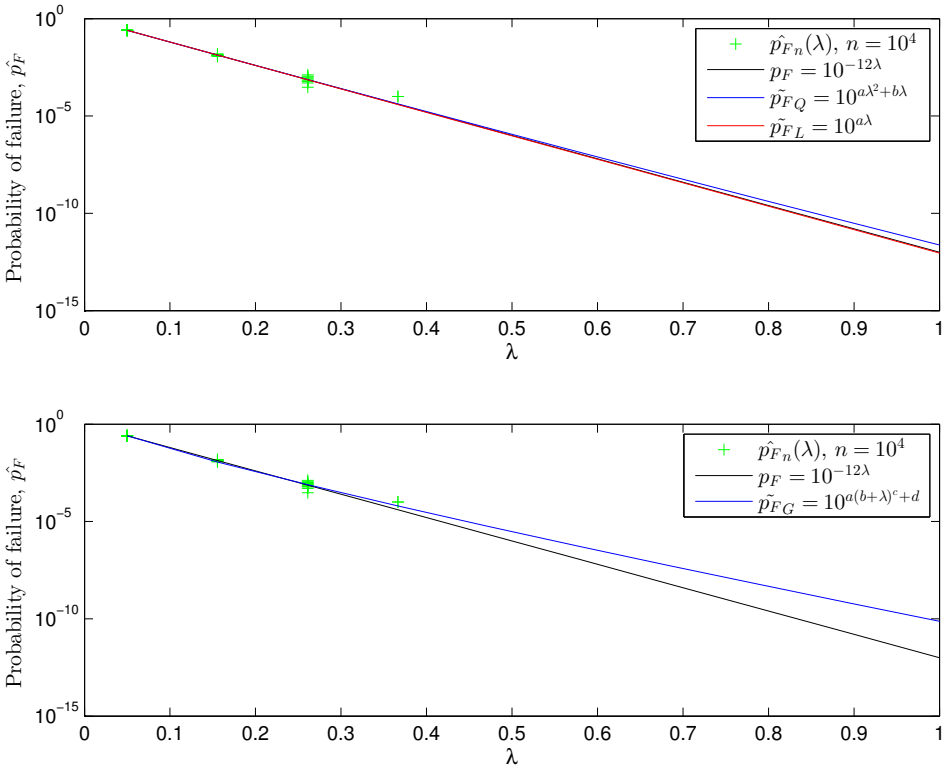


Figure 4.8: Logarithmic plot of the different fits of the simulated probability failure, $\tilde{p_F}$, as a function of λ. Simulations are done in 10 samples with $n = 10^4$ for a structure with 1 component and $p_1 = 1 - 10^{-12}$. $w_i = \text{weightN}$ and $\lambda(1) = 0.05$. Original model is obtained for $\lambda = 1$.

Realisation with a good general fit, $\tilde{p_{FG}}$

Table 4.11: Comparison of the three fits when $s = 1$, $p_1 = 1 - 10^{-12}$, $\lambda = \text{linspace}(0.05, 10, 1)$ and simulation size is 10 samples with $n = 10^4$.The least squares fit is weighted with weightN, described in algorithm 1. $\delta_{\tilde{p_F}(1)} = \frac{\tilde{p_F}(1) - p_F(1)}{p_F(1)}$, where $p_F(1) = 10^{-12}$ is the analytically calculated probability to fail.

$\log_{10}(\tilde{p_F}(\lambda))$	a	b	c	d	$\tilde{p_F}(1)$	$\delta_{\tilde{p_F}(1)}$
$a\lambda + b$	-11.9136	-0.0036			$1.210 \cdot 10^{-12}$	0.210
$a\lambda^2 + b\lambda + c$	-0.3115	-11.8444	-0.0063		$6.884 \cdot 10^{-13}$	-0.311
$a(b + \lambda)^c + d$	-11.9136	0	1	-0.0036	$1.210 \cdot 10^{-12}$	0.210

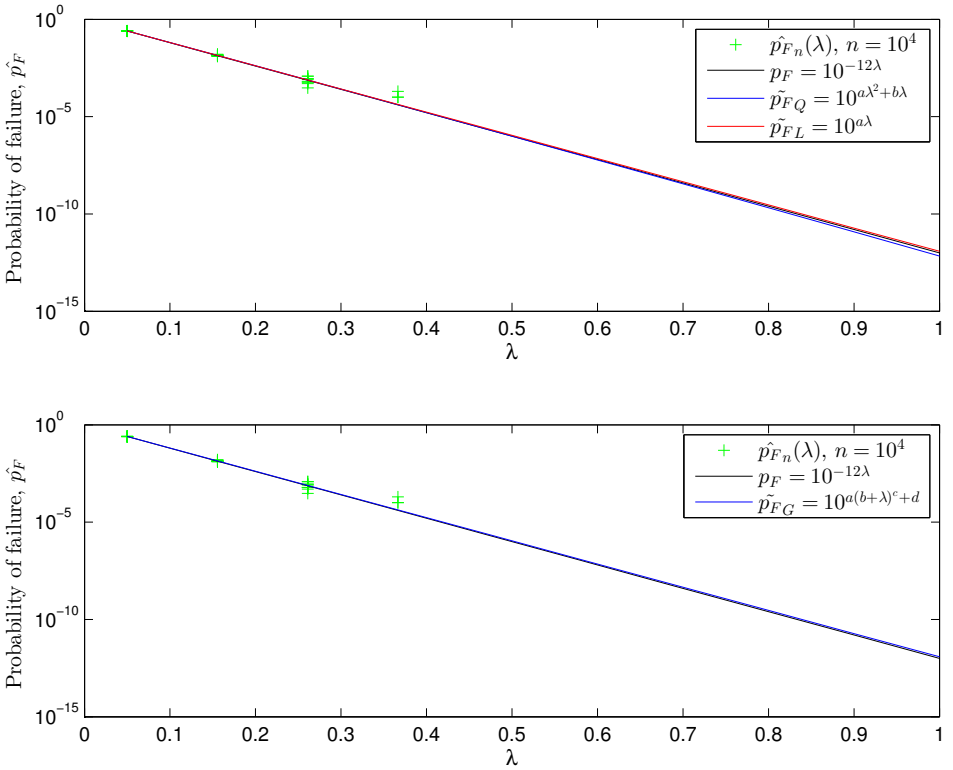


Figure 4.9: Logarithmic plot of the different fits of the simulated probability failure, $\tilde{p_F}$, as a function of λ. Simulations are done in 10 samples with $n = 10^4$ for a structure with 1 component and $p_1 = 1 - 10^{-12}$. $w_i = \text{weightN}$ and $\lambda(1) = 0.05$. Original model is obtained for $\lambda = 1$.

4.1.4 Six components in series

The result presented in this section shows that the linear fit with weighting weightN has the most accurate solutions, shown in tables 4.12 and 4.13. This combination has the estimate closest to the analytical solution in 216 of the 1000 realisations tested. The sum of squares for the linear fits with weighting weightN and weightN2 are similar, and the smallest of the combinations. The results are obtained by letting the parameter λ start at 0.25. This differs from the single component models, where $\lambda = 0.05$ clearly was the best option. The section will present different realisations of the weights weightN and weightS, where the parameter λ starts at $\lambda(1) = 0.1$ and $\lambda(1) = 0.25$. The results of the realisations are given in figures 4.10 to 4.11 and tables 4.14 to 4.15. We see that all the weights vary when changing the parameter λ. The weighting strategy weightN has good estimates for all the fits.

Table 4.12: Sum of the combinations of fit and weighting that give an estimate, $\tilde{p_F}$, closest to the analytical solution, $p_F = 6 \cdot 10^{-7}$, for 1000 realisations.

	WeightN	WeightS	WeightN2	WeightQ
Linear fit	216	140	105	37
Quadratic fit	90	37	51	16
General fit	122	40	109	37

Table 4.13: Sum of squares for each combination of fit and weighting for 1000 realisations. The sum of squares are calculated by $(\tilde{p_F} - p_F)^2$, where $p_F = 6 \cdot 10^{-7}$ is the analytical solution.

	WeightN	WeightS	WeightN2	WeightQ
Linear fit	$2.11{\cdot}10^{-11}$	$7.53{\cdot}10^{-11}$	$2.11{\cdot}10^{-11}$	$6.73{\cdot}10^{-9}$
Quadratic fit	$2.41{\cdot}10^{-10}$	$1.97{\cdot}10^{-9}$	$3.30{\cdot}10^{-10}$	$9.19{\cdot}10^{-8}$
General fit	$3.38{\cdot}10^{-11}$	$7.51{\cdot}10^{-10}$	$3.82{\cdot}10^{-11}$	$7.66{\cdot}10^{-9}$

WeightN, $\lambda = 0.25$

Table 4.14: Comparison of the three fits when $s = 6$, $p_i = 1 - 10^{-7}, i = 1, 2, \ldots, 6$, $\lambda = \text{linspace}(0.25, 10, 1)$ and simulation size is 10 samples with $n = 10^4$.The least squares fit is weighted with weightN, described in algorithm 1. $\delta_{\tilde{p_F}(1)} = \frac{\tilde{p_F}(1) - p_F(1)}{p_F(1)}$, where $p_F(1) = 6 \cdot 10^{-7}$ is the analytically calculated probability to fail.

$\log_{10}(\tilde{p_F}(\lambda))$	a	b	c	d	$\tilde{p_F}(1)$	$\delta_{\tilde{p_F}(1)}$
$a\lambda + b$	-6.8861	0.7330			$7.028 \cdot 10^{-7}$	0.171
$a\lambda^2 + b\lambda + c$	0.0734	-6.9362	0.7410		$7.555 \cdot 10^{-7}$	0.259
$a(b + \lambda)^c + d$	-6.8861	0	1	0.7330	$7.028 \cdot 10^{-7}$	0.171

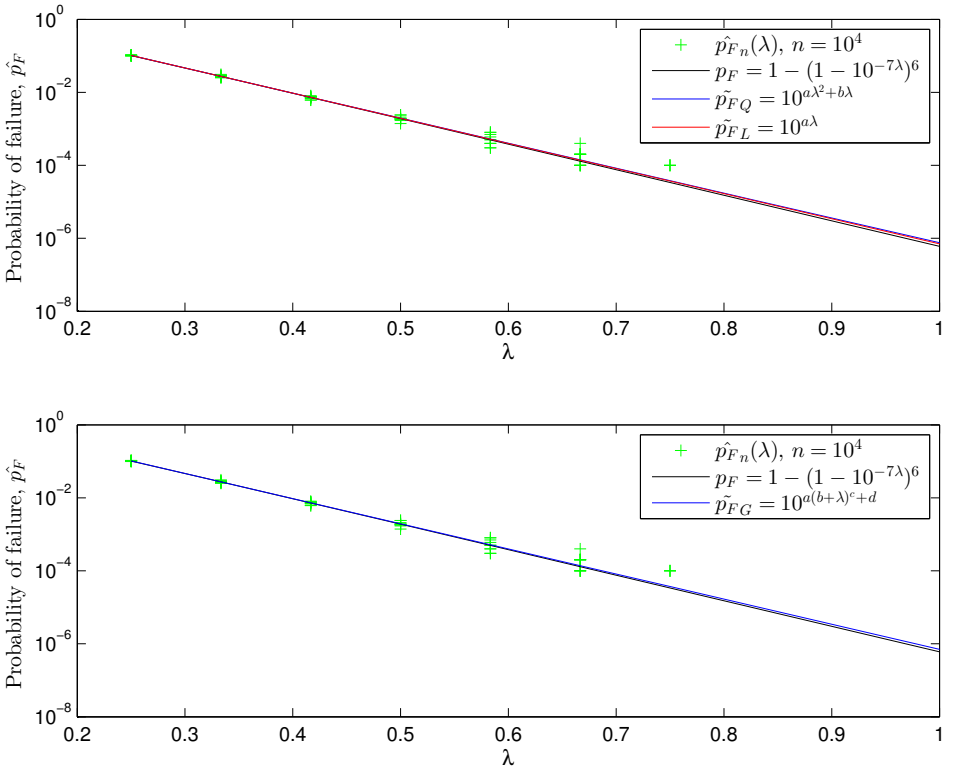


Figure 4.10: Logarithmic plot of the different fits of the simulated probability failure, $\tilde{p_F}$, as a function of λ. Simulations are done in 10 samples with $n = 10^4$ for a structure with 6 component and $p_i = 1 - 10^{-7}$ $\forall i$. $w_i = \text{weightN}$ and $\lambda(1) = 0.25$. Original model is obtained for $\lambda = 1$.

WeightS, $\lambda = 0.25$

Table 4.15: Comparison of the three fits when $s = 6$, $p_i = 1 - 10^{-7}, i = 1, 2, \ldots, 6$, $\lambda = \text{linspace}(0.25, 10, 1)$ and simulation size is 10 samples with $n = 10^4$.The least squares fit is weighted with weightS, described in algorithm 2. $\delta_{\tilde{p_F}(1)} = \frac{\tilde{p_F}(1) - p_F(1)}{p_F(1)}$, where $p_F(1) = 6 \cdot 10^{-7}$ is the analytically calculated probability to fail.

$\log_{10}(\tilde{p_F}(\lambda))$	a	b	c	d	$\tilde{p_F}(1)$	$\delta_{\tilde{p_F}(1)}$
$a\lambda + b$	-6.8423	0.7199			$7.544 \cdot 10^{-7}$	0.257
$a\lambda^2 + b\lambda + c$	0.2972	-7.0887	0.7650		$9.409 \cdot 10^{-7}$	0.568
$a(b + \lambda)^c + d$	-6.8422	-0.0105	0.9693	0.7214	$8.884 \cdot 10^{-7}$	0.481

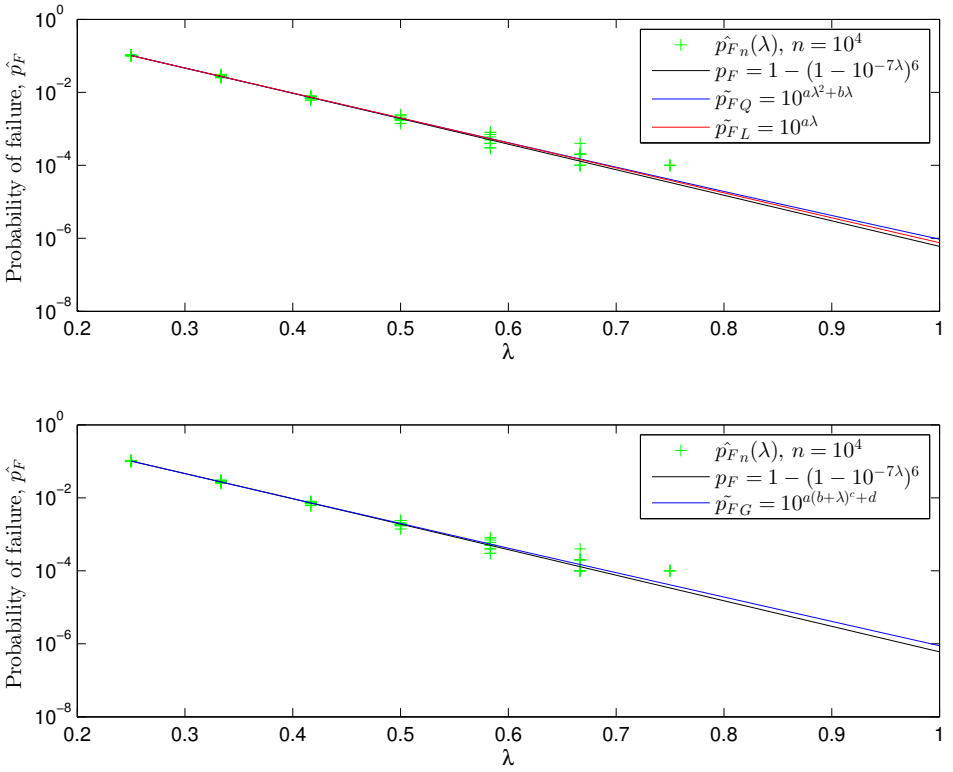


Figure 4.11: Logarithmic plot of the different fits of the simulated probability failure, $\tilde{p_F}$, as a function of λ. Simulations are done in 10 samples with $n = 10^4$ for a structure with 6 component and $p_i = 1 - 10^{-7}$ $\forall i$. w_i = weightS and $\lambda(1) = 0.25$. Original model is obtained for $\lambda = 1$.

WeightN, $\lambda = 0.1$

Table 4.16: Comparison of the three fits when $s = 6$, $p_i = 1 - 10^{-7}, i = 1, 2, \ldots, 6$, $\lambda = \text{linspace}(0.1, 10, 1)$ and simulation size is 10 samples with $n = 10^4$.The least squares fit is weighted with weightN, described in algorithm 1. $\delta_{\tilde{p_F}(1)} = \frac{\tilde{p_F}(1) - p_F(1)}{p_F(1)}$, where $p_F(1) = 6 \cdot 10^{-7}$ is the analytically calculated probability to fail.

$\log_{10}(\tilde{p_F}(\lambda))$	a	b	c	d	$\tilde{p_F}(1)$	$\delta_{\tilde{p_F}(1)}$
$a\lambda + b$	-5.9805	0.4876			$3.214 \cdot 10^{-6}$	4.357
$a\lambda^2 + b\lambda + c$	0.2972	-7.0887	0.7650		$9.642 \cdot 10^{-9}$	-0.984
$a(b + \lambda)^c + d$	-6.8422	-0.0105	0.9693	0.7214	$1.346 \cdot 10^{-6}$	1.243

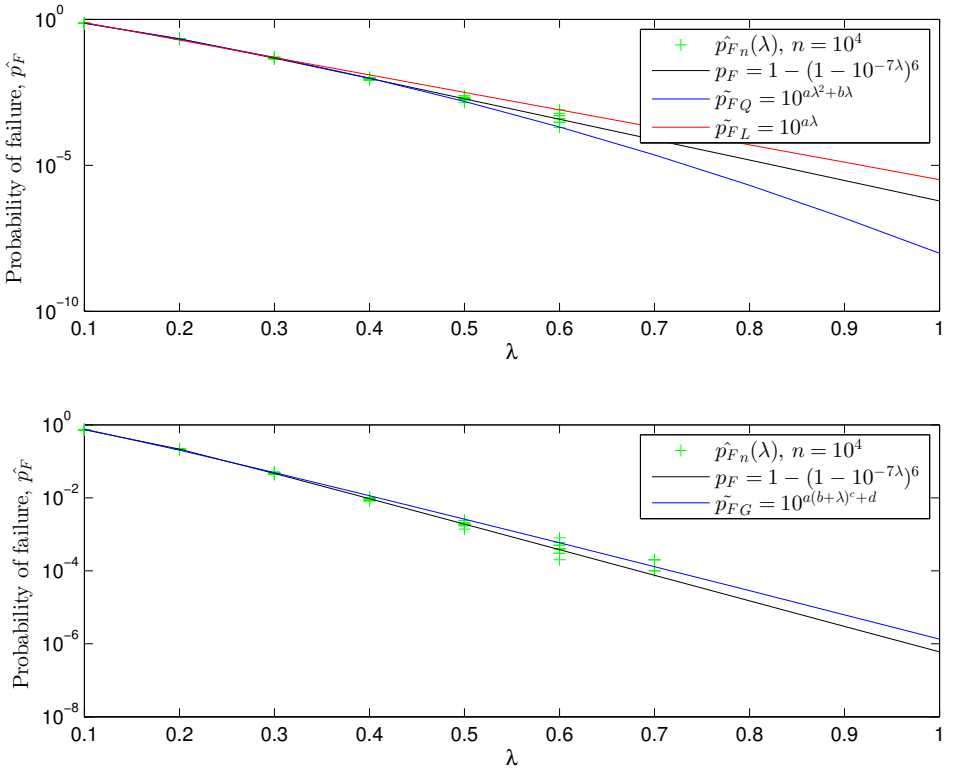


Figure 4.12: Logarithmic plot of the different fits of the simulated probability failure, $\tilde{p_F}$, as a function of λ. Simulations are done in 10 samples with $n = 10^4$ for a structure with 6 component and $p_i = 1 - 10^{-7}$ $\forall i$. w_i = weightS and $\lambda(1) = 0.1$. Original model is obtained for $\lambda = 1$.

WeightS, $\lambda = 0.1$

Table 4.17: Comparison of the three fits when $s = 6$, $p_i = 1 - 10^{-7}, i = 1, 2, \ldots, 6$, $\lambda = \text{linspace}(0.1, 10, 1)$ and simulation size is 10 samples with $n = 10^4$.The least squares fit is weighted with weightS, described in algorithm 2. $\delta_{\tilde{p_F}(1)} = \frac{\tilde{p_F}(1) - p_F(1)}{p_F(1)}$, where $p_F(1) = 6 \cdot 10^{-7}$ is the analytically calculated probability to fail.

$\log_{10}(\tilde{p_F}(\lambda))$	a	b	c	d	$\tilde{p_F}(1)$	$\delta_{\tilde{p_F}(1)}$
$a\lambda + b$	-6.2870	0.5213			$3.214 \cdot 10^{-6}$	1.858
$a\lambda^2 + b\lambda + c$	-2.345	-4.9774	0.3940		$9.642 \cdot 10^{-9}$	-0.804
$a(b + \lambda)^c + d$	-6.5283	-0.1067	1.0475	-0.1629	$1.346 \cdot 10^{-6}$	0.812

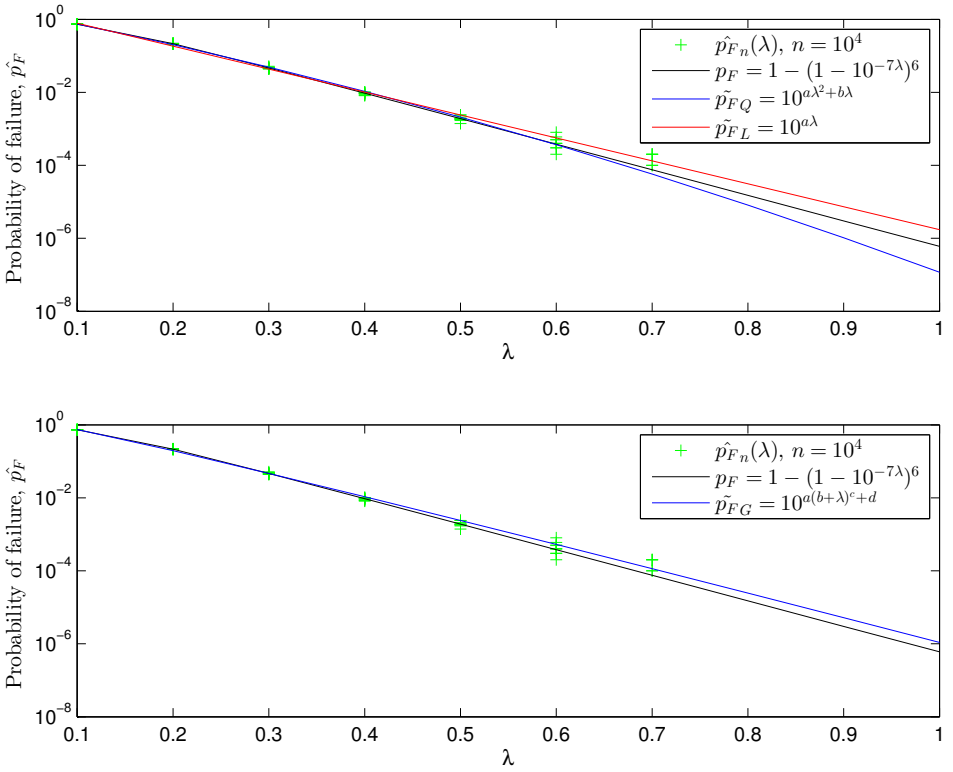


Figure 4.13: Logarithmic plot of the different fits of the simulated probability failure, $\tilde{p_F}$, as a function of λ. Simulations are done in 10 samples with $n = 10^4$ for a structure with 6 component and $p_i = 1 - 10^{-7}$ $\forall i$. w_i = weightS and $\lambda(1) = 0.1$. Original model is obtained for $\lambda = 1$.

4.2 Dependent Systems

The dependent systems, the Monte Carlo simulations of the parametrized models are compared with standard Monte Carlo simulations with large sample sizes. We introduce the confidence intervals of the standard Monte Carlo simulations, and compare our estimates to this. For the cases where we were able to formulate the models by Markov chains, the analytical solution was found by limiting probabilities.

4.2.1 Cascading failures on 2003-system

In this section, we construct the model described in section 3.4.2, and compare the results with the analytical solution obtained by limiting probabilities from equation (3.51). The probability to fail, q_i for the components $i = 1, 2, 3$ were set to $q = 10^{-7}$. Equation (3.51) gives

$$p_F \approx 1.50 \cdot 10^{-7} \tag{4.1}$$

In table 4.18, the combination of weights and fits are compared for 1000 realisations. We see that the linear fit with weighting strategy weightN is best, with more than 50% of the best rest results. Table 4.19 tells the same story, where the linear fits with weightN and weightN2 have the smallest squared error. The general fit with weightN is also much better than many of the other fits. Figure 4.14 and table 4.20 shows a realisation where all the fits are very accurate, and the general fit obtained a precision only 1.2% from the analytic solution. Notice that the general fit is close to being linear.

Table 4.18: Sum of the combinations of fit and weighting that give an estimate, $\tilde{p_F}$, closest to the analytical solution, $p_F = 1.50 \cdot 10^{-7}$, for 1000 realisations.

	WeightN	WeightS	WeightN2	WeightQ
Linear fit	507	59	114	23
Quadratic fit	26	7	39	14
General fit	103	23	60	25

Table 4.19: Sum of squares for each combination of fit and weighting for 1000 realisations. The sum of squares are calculated by $(\tilde{p_F} - p_F)^2$, where $p_F = 1.50 \cdot 10^{-7}$ is the analytical solution.

	WeightN	WeightS	WeightN2	WeightQ
Linear fit	$4.71 \cdot 10^{-13}$	$2.00 \cdot 10^{-12}$	$5.32 \cdot 10^{-13}$	$1.30 \cdot 10^{-10}$
Quadratic fit	$1.49 \cdot 10^{-11}$	$3.79 \cdot 10^{-11}$	$1.57 \cdot 10^{-11}$	$1.33 \cdot 10^{-5}$
General fit	$7.87 \cdot 10^{-13}$	$8.69 \cdot 10^{-12}$	$1.03 \cdot 10^{-12}$	$1.48 \cdot 10^{-10}$

Table 4.20: Comparison of the three fits for the model described in section 4.2.1 with $p_1 = p_2 = p_3 = 1 - 10^{-7}$ and $\lambda = \text{linspace}(0.2, 10, 1)$. The system is simulated by 10 samples with $n = 10^4$. The least squares fit is weighted with weightN, described in algorithm 1. $\delta_{\tilde{p_F}(1)} = \frac{\tilde{p_F}(1) - p_{FMC}(1)}{p_{FMC}(1)}$, where $p_{FMC}(1) = 1.500 \cdot 10^{-7}$ is the probability to fail estimated by Markov chain limiting probabilities.

$\log_{10}(\tilde{p_F}(\lambda))$	a	b	c	d	$\tilde{p_F}(1)$	$\delta_{\tilde{p_F}(1)}$
$a\lambda + b$	-6.9569	0.1576			$1.587 \cdot 10^{-7}$	0.058
$a\lambda^2 + b\lambda + c$	-0.4888	-6.6756	0.1203		$9.035 \cdot 10^{-8}$	-0.398
$a(b + \lambda)^c + d$	-6.9571	0.0027	1.0083	0.1572	$1.518 \cdot 10^{-7}$	0.012

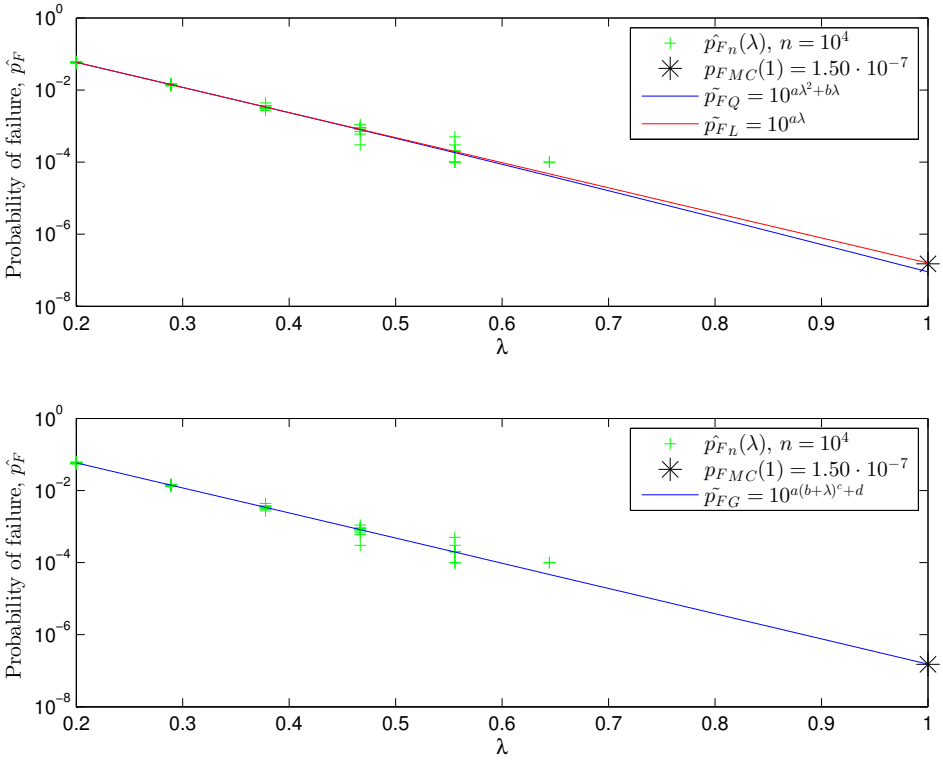


Figure 4.14: Logarithmic plot of the different fits of the simulated probability failure, $\tilde{p_F}$, as a function of λ. Simulations are done in 10 samples with $n = 10^4$ for the model described in section 4.2.1 $w_i = \text{weightN}$ and $\lambda(1) = 0.2$. Original model is obtained for $\lambda = 1$.

4.2.2 Cascading failure on two 2003-systems and three independent components in series.

In this section, we construct the model described in section 3.4.3, with a probability to fail, $q_i = 10^{-7}$ for the dependent components $1, 2, 3, 6, 7$ and 8 and $q_i = 10^{-8}$ for the other components, $4, 5$ and 9. The analytical solution of the probability of failure, p_F, is from equation (3.52)

$$p_F \approx 3.30 \cdot 10^{-7}. \tag{4.2}$$

The table 4.21 shows that once again, the weighting strategy, weightN has best estimate most times. The general fit is the best for 281 of the 1000 realisations. In table 4.22, we see that the linear fit with weighting strategy weightN2 has the least sum of squared error. A realisation of simulating the parametrized model is presented in figure 4.15 and table 4.23. This is generated with weighting strategy, weightN and $\lambda(1) = 0.2$. The linear and general fit are accurate, but the quadratic fit is as good as it was for 4.20.

Table 4.21: Sum of the combinations of fit and weighting that give an estimate, $\tilde{p_F}$, closest to the analytical solution, $p_F = 3.30 \cdot 10^{-7}$, for 1000 realisations.

	WeightN	WeightS	WeightN2	WeightQ
Linear fit	207	84	220	4
Quadratic fit	5	3	8	1
General fit	281	5	176	6

Table 4.22: Sum of squares for each combination of fit and weighting for 1000 realisations. The sum of squares are calculated by $(\tilde{p_F} - p_F)^2$, where $p_F = 1.50 \cdot 10^{-7}$ is the analytical solution.

	WeightN	WeightS	WeightN2	WeightQ
Linear fit	$1.66 \cdot 10^{-12}$	$5.60 \cdot 10^{-12}$	$1.24 \cdot 10^{-12}$	$6.35 \cdot 10^{-10}$
Quadratic fit	$5.13 \cdot 10^{-11}$	$8.44 \cdot 10^{-11}$	$5.34 \cdot 10^{-11}$	$1.94 \cdot 10^{-6}$
General fit	$1.29 \cdot 10^{-12}$	$4.35 \cdot 10^{-11}$	$1.75 \cdot 10^{-12}$	$8.82 \cdot 10^{-10}$

Table 4.23: Comparison of the three fits for the model described in 4.2.2 with $p_1 = p_2 = p_3 = 1 - 10^{-7}$ and $\lambda = \text{linspace}(0.2, 10, 1)$. The system is simulated by 10 samples with $n = 10^4$. The least squares fit is weighted with weightN, described in algorithm 1. $\delta_{p_{\tilde{F}}(1)} = \frac{p_{\tilde{F}}(1) - p_{FMC}(1)}{p_{FMC}(1)}$, where $p_{FMC}(1) = 3.300 \cdot 10^{-7}$ is the probability to fail estimated by Markov chain limiting probabilities in series.

$\log_{10}(p_{\tilde{F}}(\lambda))$	a	b	c	d	$p_{\tilde{F}}(1)$	$\delta_{p_{\tilde{F}}(1)}$
$a\lambda + b$	-7.1395	0.6778			$3.454 \cdot 10^{-7}$	0.047
$a\lambda^2 + b\lambda + c$	-1.2438	-6.4342	0.5852		$8.076 \cdot 10^{-8}$	-0.755
$a(b+\lambda)^c + d$	-7.1399	0.0058	1.0177	0.6770	$3.129 \cdot 10^{-7}$	-0.052

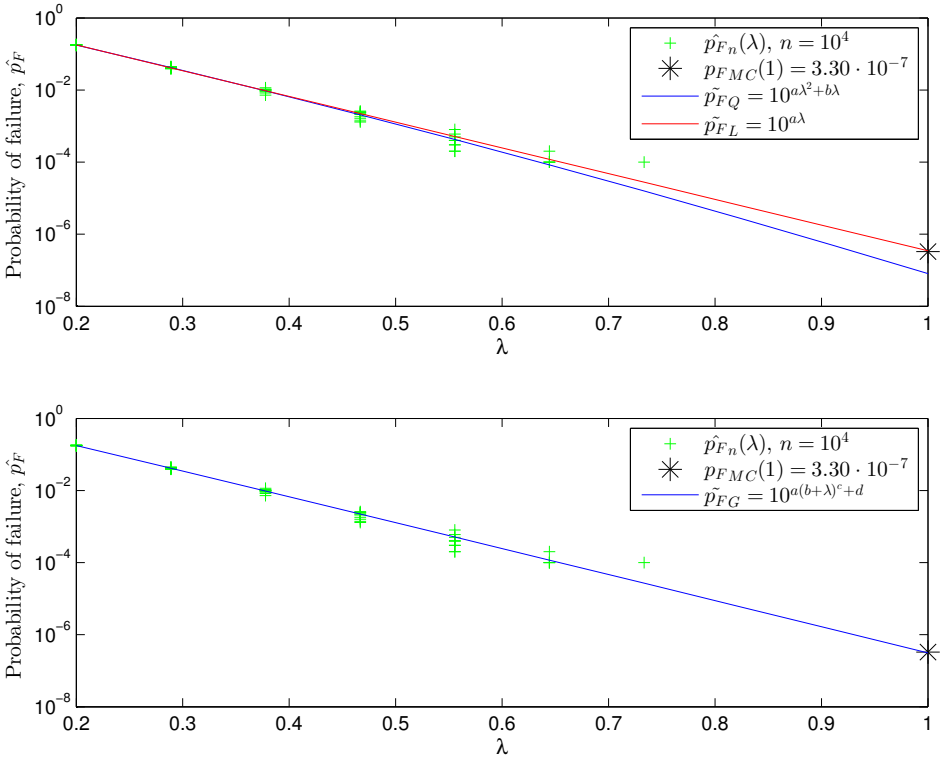


Figure 4.15: Logarithmic plot of the different fits of the simulated probability failure, $\tilde{p_F}$, as a function of λ. Simulations are done in 10 samples with $n = 10^4$ for the model described in section 4.2.2, w_i = weightN and $\lambda(1) = 0.2$. Original model is obtained for $\lambda = 1$.

4.2.3 Cascading failures with repair interval combined in series

In this section, we construct the model described in section 7, with a probability to fail, $q_i = 10^{-7}$ for the dependent components, $i = 1, 2, 4, 5$ and $q_i = 10^{-6}$ for the independent components,$i = 3, 6$. The estimation of the parametrized model is compared to estimates from standard Monte Carlo simulations with large sample sizes. The estimates with standard Monte Carlo have been done with both $n = 8$ and $n = 11$ sample sizes, and we have estimated the corresponding confidence to these estimates. The parametrized method still uses 10 samples with $n = 10^4$ random variables in each sample. The tables 4.24 and 4.25 and figures 4.17 and 4.25 that the different fits are accurate. The estimates are done with weighting strategy weightN2 and $\lambda(1) = 0.2$, and compared to the estimations with larger sample size, $\hat{p_{F}}_n$, where $n = 10^8$ and $n = 10^{11}$. The confidence interval of these estimates are added to the figure. The experimental confidence interval is calculated by equation 3.15 and Chebychev confidence interval is calculated by equation (3.16). Notice that the estimate with the general fit is inside the 95% confidence interval obtained for $\hat{p_{F}}_n$, $n = 10^8$.

Chebychev Confindence Interval, $n = 10^{11}$

A plot with the estimates compared to the Chebychev confidence interval is presented in figure 4.16. We see that our estimates are inside the confidence interval, even though the confidence interval is calculated with $n = 11$. This is since the interval is a crude and maximized confidence interval.

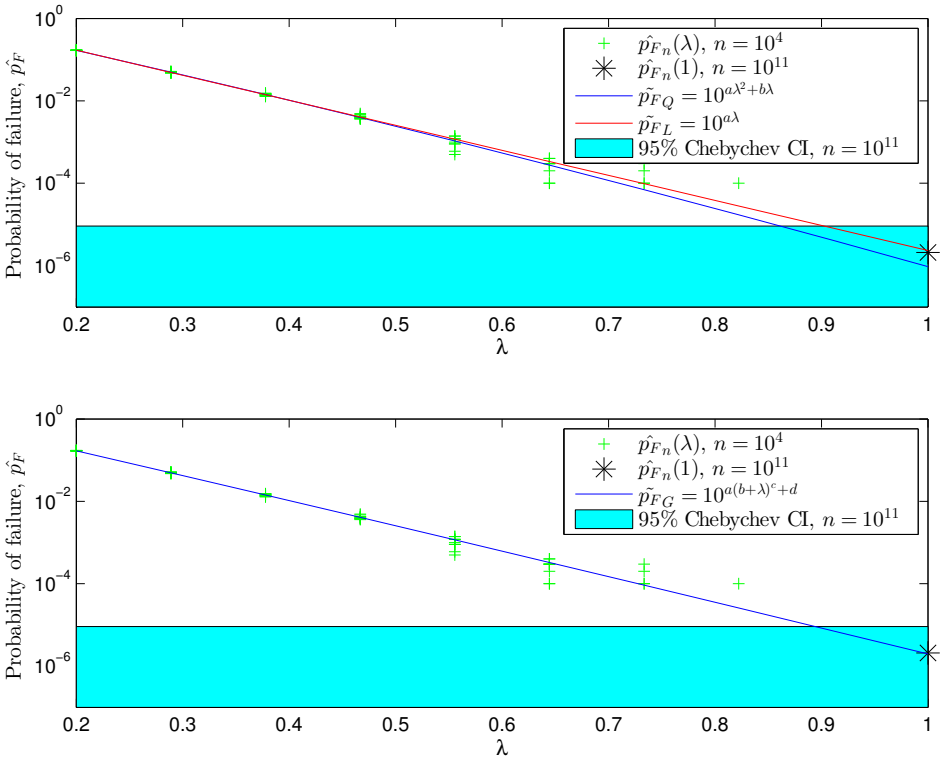


Figure 4.16: Logarithmic plot of the different fits of the simulated probability failure, $\tilde{p_F}$, as a function of λ. Simulations are done in 10 samples with $n = 10^4$ for the model described in section 4.2.3, w_i = weightN and $\lambda(1) = 0.2$. Original model is obtained for $\lambda = 1$.

Confidence Interval, $n = 10^{11}$

Table 4.24: Comparison of the three fits for the system described in section 4.2.3. $p_1 = p_2 = p_4 = p_5 = 1 - 10^{-7}$, $p_3 = p_6 = 1 - 10^{-6}$ and $\lambda = \text{linspace}(0.2, 10, 1)$. The system is simulated by 10 samples with $n = 10^4$. The least squares fit is weighted with weightN, described in algorithm 1. $\delta_{\tilde{p_F}(1)} = \frac{\tilde{p_F}(1) - \hat{p_F}_N(1)}{\hat{p_F}_N(1)}$, where $\hat{p_F}_N(1) = 2.085 \cdot 10^{-6}$ is the probability to fail estimated by original Monte Carlo simulation with sample size $N = 10^{11}$.

$\log_{10}(\tilde{p_F}(\lambda))$	a	b	c	d	$\tilde{p_F}(1)$	$\delta_{\tilde{p_F}(1)}$
$a\lambda + b$	-6.0764	0.4448			$2.335 \cdot 10^{-6}$	0.120
$a\lambda^2 + b\lambda + c$	-0.8323	-5.5663	-0.0063		$9.450 \cdot 10^{-7}$	-0.547
$a(b + \lambda)^c + d$	-6.0772	0.0109	1.0335	0.4430	$1.984 \cdot 10^{-6}$	-0.048

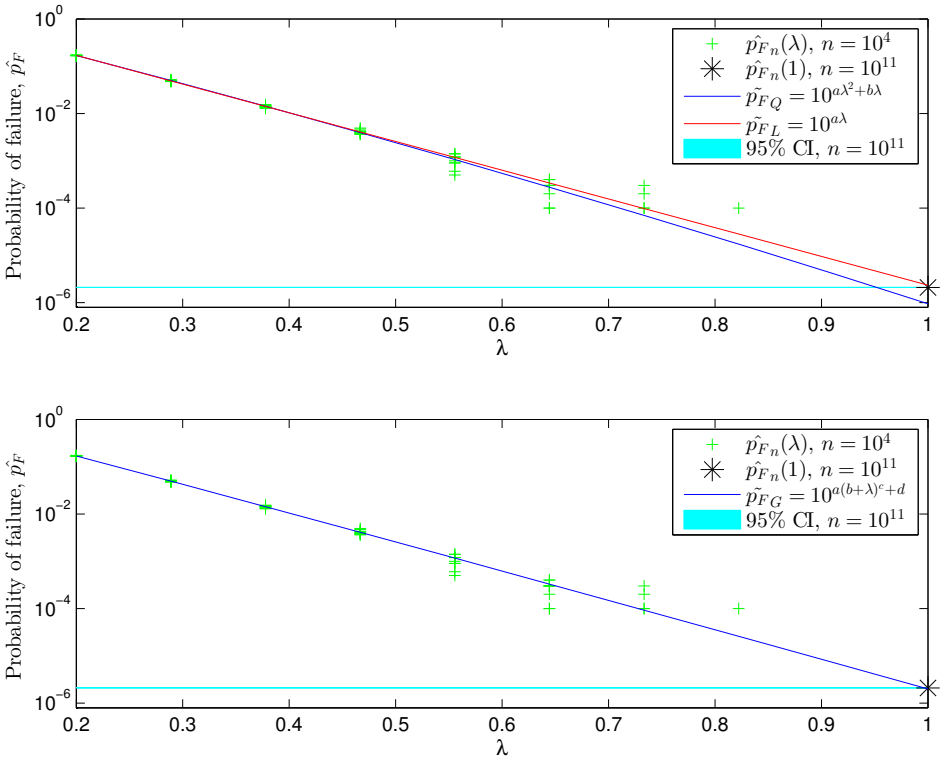


Figure 4.17: Logarithmic plot of the different fits of the simulated probability failure, $\tilde{p_F}$, as a function of λ. Simulations are done in 10 samples with $n = 10^4$ for the model described in section 4.2.3, $w_i = \text{weightN}$ and $\lambda(1) = 0.2$. Original model is obtained for $\lambda = 1$..

Confidence Interval, $n = 10^8$

Table 4.25: Comparison of the three fits for the system described in section 4.2.3. $p_1 = p_2 = p_4 = p_5 = 1 - 10^{-7}$, $p_3 = p_6 = 1 - 10^{-6}$ and $\lambda = \text{linspace}(0.2, 10, 1)$. The system is simulated by 10 samples with $n = 10^4$. The least squares fit is weighted with weightN, described in algorithm 1. $\delta_{\tilde{p_F}(1)} = \frac{\tilde{p_F}(1) - \hat{p_F}_N(1)}{\hat{p_F}_N(1)}$, where $\hat{p_F}_N(1) = 2.130 \cdot 10^{-6}$ is the probability to fail estimated by original Monte Carlo simulation with sample size $N = 10^8$.

$\log_{10}(\tilde{p_F}(\lambda))$	a	b	c	d	$\tilde{p_F}(1)$	$\delta_{\tilde{p_F}(1)}$
$a\lambda + b$	-6.0764	0.4448			$2.335 \cdot 10^{-6}$	0.096
$a\lambda^2 + b\lambda + c$	-0.8323	-5.5663	-0.0063		$9.450 \cdot 10^{-7}$	-0.556
$a(b + \lambda)^c + d$	-6.0772	0.0109	1.0335	0.4430	$1.984 \cdot 10^{-6}$	-0.069

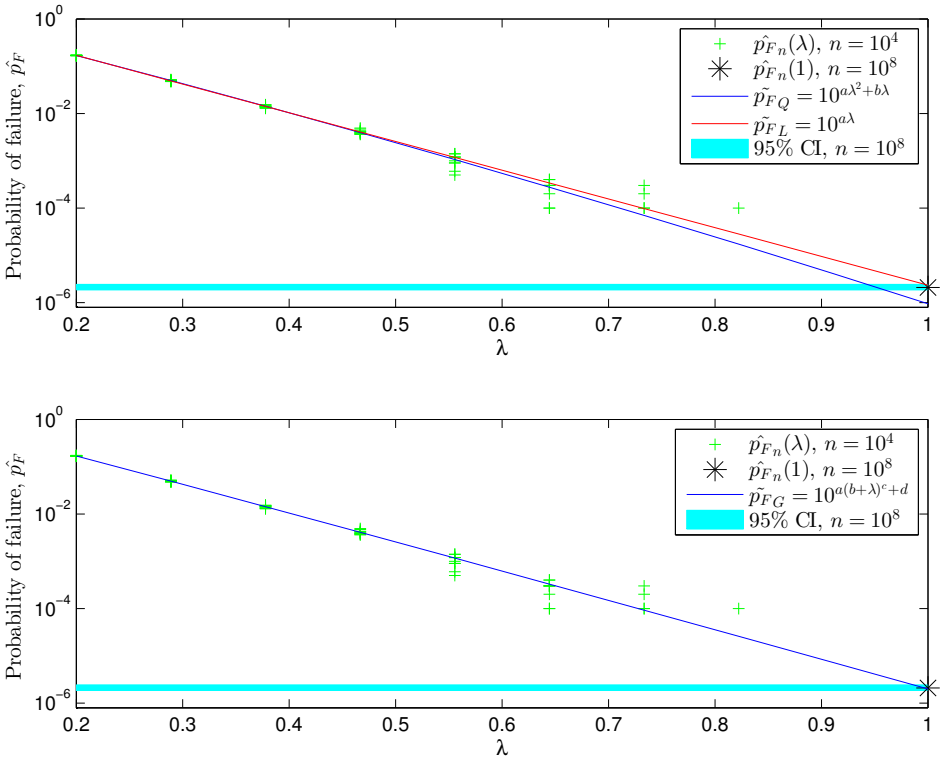


Figure 4.18: Logarithmic plot of the different fits of the simulated probability failure, $\tilde{p_F}$, as a function of λ. Simulations are done in 10 samples with $n = 10^4$ for the model described in section 4.2.3, $w_i = \text{weightN}$ and $\lambda(1) = 0.2$. Original model is obtained for $\lambda = 1$.

4.2.4 Cascading failures in parallel structure with repair interval

In this section, we construct the model described in section 3.4.5, with probability to fail, $q = 10^{-7}$ for all the three components. The parametrized method is compared to simulations of original Monte Carlo simulations, with larger sample size. The result in table 4.26 shows that the combination with weightS and linear fit had the most accurate estimates. The sum of squared error in 4.27 shows that the linear and the general fit with weighting strategy weightN have a smaller squared error. Results from a realisation of the parametrized system is given in figures 4.19, 4.20 and 4.21 and tables 4.28 and 4.29. The confidence intervals are calculated and included as for section 4.2.2. We see that our general and linear fit is inside the 95% confidence interval for $n = 10^8$, and that all the estimates are in the Chebychev confidence interval with $n = 10^{11}$.

Table 4.26: Sum of the combinations of fit and weighting that give an estimate, $\tilde{p_F}$, closest to the simulated estimate, $\hat{p_{F_n}} = 3.158 \cdot 10^{-8}$, $n = 10^{11}$, for 1000 realisations.

	WeightN	WeightS	WeightN2	WeightQ
Linear fit	47	289	71	14
Quadratic fit	1	42	5	4
General fit	272	159	89	7

Table 4.27: Sum of squares for each combination of fit and weighting for 1000 realisations. The sum of squares are calculated by $(\tilde{p_F} - \hat{p_{F_n}})^2$, where $\hat{p_{F_n}} = 3.158 \cdot 10^{-8}$, $n = 10^{11}$, is the simulated estimate.

	WeightN	WeightS	WeightN2	WeightQ
Linear fit	$1.11{\cdot}10^{-14}$	$1.79{\cdot}10^{-14}$	$1.18{\cdot}10^{-14}$	$2.10{\cdot}10^{-12}$
Quadratic fit	$2.43{\cdot}10^{-11}$	$1.53{\cdot}10^{-12}$	$2.18{\cdot}10^{-12}$	$3.60{\cdot}10^{-4}$
General fit	$9.79{\cdot}10^{-15}$	$2.94{\cdot}10^{-14}$	$1.06{\cdot}10^{-14}$	$3.93{\cdot}10^{-12}$

Chebychev Confidence Interval, $n = 10^{11}$

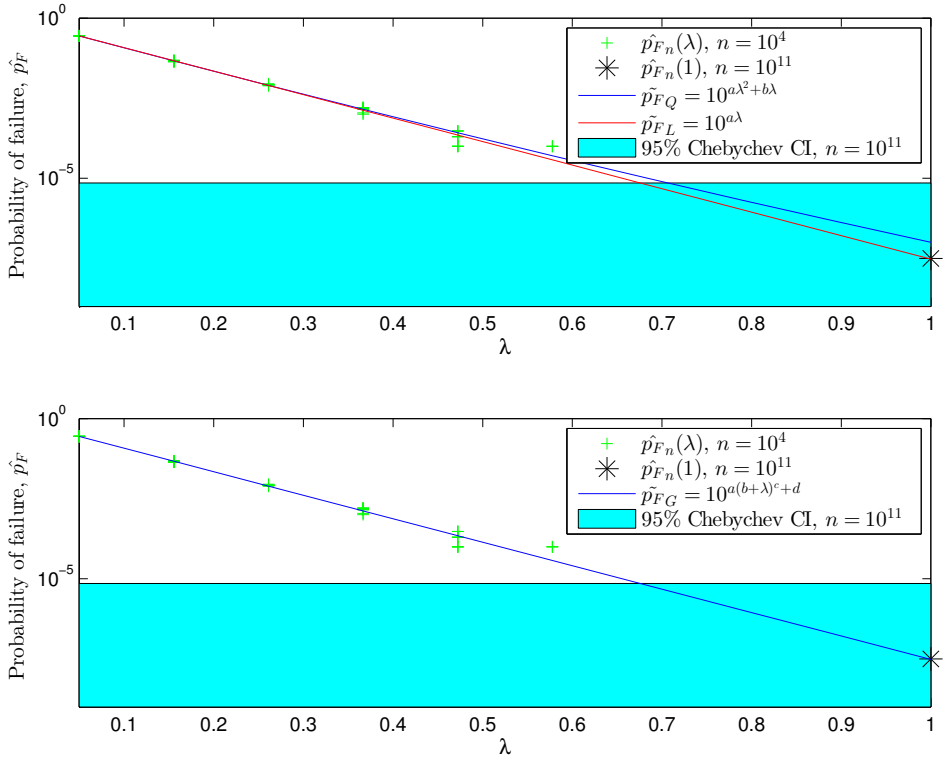


Figure 4.19: Logarithmic plot of the different fits of the simulated probability failure, $\tilde{p_F}$, as a function of λ. Simulations are done in 10 samples with $n = 10^4$ for the model described in section 4.2.3, w_i = weightN and $\lambda(1) = 0.05$. Original model is obtained for $\lambda = 1$.

Confidence Interval, $n = 10^{11}$

Table 4.28: Comparison of the three fits for the system described in section 4.2.4. $p_1 = p_2 = p_3 = 1 - 10^{-7}$ and $\lambda = \text{linspace}(0.05, 10, 1)$. The system is simulated by 10 samples with $n = 10^4$. The least squares fit is weighted with weightN, described in algorithm 1. $\delta_{\tilde{p_F}(1)} = \frac{\tilde{p_F}(1) - \hat{p_{F\,N}}(1)}{\hat{p_{F\,N}}(1)}$, where $\hat{p_{F\,N}}(1) = 3.158 \cdot 10^{-8}$ is the probability to fail estimated by original Monte Carlo simulation with sample size $N = 10^{11}$.

$\log_{10}(\tilde{p_F}(\lambda))$	a	b	c	d	$\tilde{p_F}(1)$	$\delta_{\tilde{p_F}(1)}$
$a\lambda + b$	-7.3917	-0.1858			$2.645 \cdot 10^{-8}$	-0.162
$a\lambda^2 + b\lambda + c$	0.5967	-7.5619	-0.1782		$7.188 \cdot 10^{-8}$	1.276
$a(b + \lambda)^c + d$	-7.3915	0.0004	0.9979	-0.1858	$2.664 \cdot 10^{-8}$	-0.157

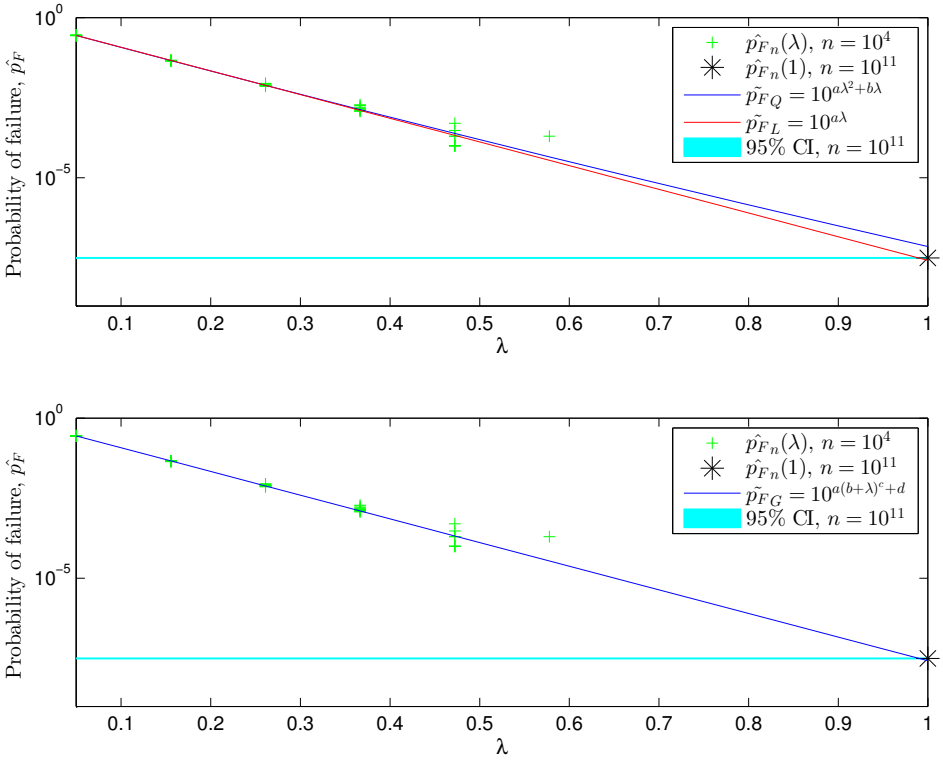


Figure 4.20: Logarithmic plot of the different fits of the simulated probability failure, $\tilde{p_F}$, as a function of λ. Simulations are done in 10 samples with $n = 10^4$ for the model described in section 4.2.3, $w_i = \text{weightN}$ and $\lambda(1) = 0.05$. Original model is obtained for $\lambda = 1$.

Confidence Interval, $n = 10^8$

Table 4.29: Comparison of the three fits for the system described in section 4.2.4. $p_1 = p_2 = p_3 = 1 - 10^{-7}$ and $\lambda = \text{linspace}(0.05, 10, 1)$. The system is simulated by 10 samples with $n = 10^4$. The least squares fit is weighted with weightN, described in algorithm 1. $\delta_{\tilde{p_F}(1)} = \frac{\tilde{p_F}(1) - \hat{p_F}_N(1)}{\hat{p_F}_N(1)}$, where $\hat{p_F}_N(1) = 3.000 \cdot 10^{-8}$ is the probability to fail estimated by original Monte Carlo simulation with sample size $N = 10^8$.

$\log_{10}(\tilde{p_F}(\lambda))$	a	b	c	d	$\tilde{p_F}(1)$	$\delta_{\tilde{p_F}(1)}$
$a\lambda + b$	-7.3917	-0.1858			$2.645 \cdot 10^{-8}$	-0.118
$a\lambda^2 + b\lambda + c$	0.5967	-7.5619	-0.1782		$7.188 \cdot 10^{-8}$	1.396
$a(b + \lambda)^c + d$	-7.3915	0.0004	0.9979	-0.1858	$2.664 \cdot 10^{-8}$	-0.112

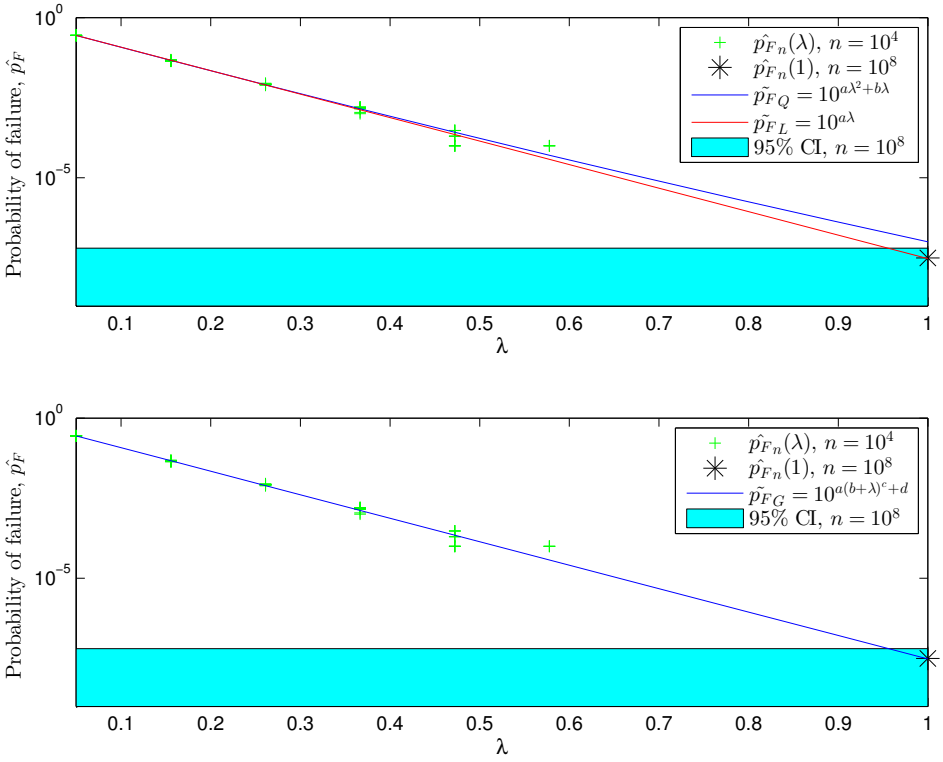


Figure 4.21: Logarithmic plot of the different fits of the simulated probability failure, $\tilde{p_F}$, as a function of λ. Simulations are done in 10 samples with $n = 10^4$ for the model described in section 4.2.3, $w_i = \text{weightN}$ and $\lambda(1) = 0.05$. Original model is obtained for $\lambda = 1$.

4.3 Common cause failure

In this section, we construct the model described in section 3.4.6, with a probability to fail, $q = 10^{-7}$ for all the four components in the model. The parametrized model is compared to the estimation of standard Monte Carlo simulation with a large sample size. The table 4.30 is quite different from the previous tables of the best weights. The general fit combined with weightS was the best fit for 227 of 1000 realisations. The linear fit with weightN was never the best combination. Table 4.31 gives the relative error of each simulation, and we see that all the weights except weightQ differ a lot from the simulated estimate, $\hat{p_{F}}_n$, $n = 10^{11}$. The weights weightQ was however quite good for this realisation. The realisation with weightN is presented in figure 4.22 and for weightQ in figure 4.23. The confidence interval obtained with $n = 10^{11}$ is added to the figures. The unweighted fit was also tested on this problem, and the result is presented in 4.24 and 4.34.

Table 4.30: Sum of the combinations of fit and weighting that give an estimate, $\tilde{p_F}$, closest to the simulated estimate, $\hat{p_{F}}_n = 1.101 \cdot 10^{-7}$, $n = 10^{11}$, for 1000 realisations.

	WeightN	WeightS	WeightN2	WeightQ
Linear fit	0	12	3	124
Quadratic fit	159	167	52	37
General fit	7	227	32	180

Table 4.31: Relative difference, $\delta_{\tilde{p_F}(1)}$, for each of the combinations, where $\delta_{\tilde{p_F}(1)} = \frac{\tilde{p_F}(1) - \hat{p_{F}}_n}{\hat{p_{F}}_n}$. $\hat{p_{F}}_n = 1.012 \cdot 10^{-7}$ is simulated by $n = 10^{11}$

	WeightN	WeightS	WeightN2	WeightQ
Linear fit	-0.986	-0.959	-0.983	-0.266
Quadratic fit	2.419	1.887	1.951	18.30
General fit	-0.978	-0.973	-0.96	-0.043

Table 4.32: Comparison of the three fits for the system described in section 4.3. $p_1 = p_2 = p_3 = p_4 = 1 - 10^{-7}$ and $\lambda = \text{linspace}(0.2, 10, 1)$. The system is simulated by 10 samples with $n = 10^4$. The least squares fit is weighted with weightN, described in algorithm 1. $\delta_{\tilde{p_F}(1)} = \frac{\tilde{p_F}(1) - \hat{p_F}_N(1)}{\hat{p_F}_N(1)}$, where $\hat{p_F}_N(1) = 1.012 \cdot 10^{-7}$ is the probability to fail estimated by original Monte Carlo simulation with sample size $N = 10^{11}$.

$\log_{10}(\tilde{p_F}(\lambda))$	a	b	c	d	$\tilde{p_F}(1)$	$\delta_{\tilde{p_F}(1)}$
$a\lambda + b$	-9.0077	0.1752			$1.471 \cdot 10^{-9}$	-0.985
$a\lambda^2 + b\lambda + c$	4.6061	-11.574	0.5066		$3.460 \cdot 10^{-7}$	2.42
$a(b + \lambda)^c + d$	-9.0067	-0.0199	0.9383	0.1774	$2.183 \cdot 10^{-9}$	-0.978

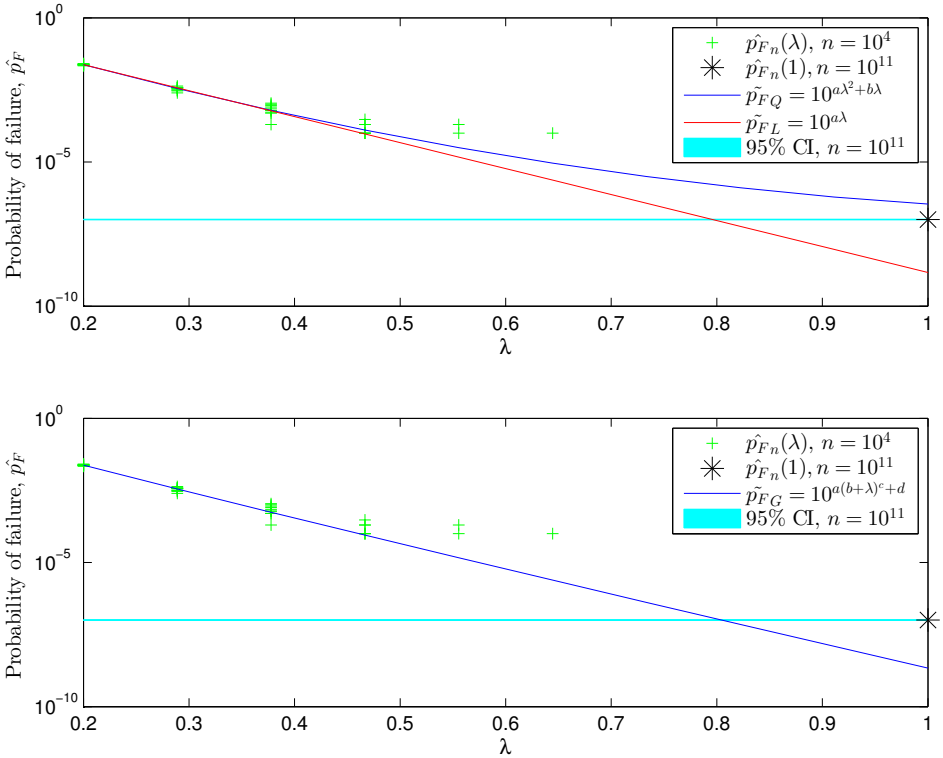


Figure 4.22: Logarithmic plot of the different fits of the simulated probability failure, $\tilde{p_F}$, as a function of λ. Simulations are done in 10 samples with $n = 10^4$ for the model described in section 4.3, $w_i = \text{weightN}$ and $\lambda(1) = 0.2$. Original model is obtained for $\lambda = 1$.

Table 4.33: Comparison of the three fits for the system described in section 4.3. $p_1 = p_2 = p_3 = p_4 = 1 - 10^{-7}$ and $\lambda = \text{linspace}(0.2, 10, 1)$. The system is simulated by 10 samples with $n = 10^4$. The least squares fit is weighted with weightQ, described in algorithm 4. $\delta_{\tilde{p_F}(1)} = \frac{\tilde{p_F}(1) - \hat{p_F}_N(1)}{\hat{p_F}_N(1)}$, where $\hat{p_F}_N(1) = 1.012 \cdot 10^{-7}$ is the probability to fail estimated by original Monte Carlo simulation with sample size $N = 10^{11}$.

$\log_{10}(\tilde{p_F}(\lambda))$	a	b	c	d	$\tilde{p_F}(1)$	$\delta_{\tilde{p_F}(1)}$
$a\lambda + b$	-5.9631	-1.1659			$7.430 \cdot 10^{-8}$	-0.266
$a\lambda^2 + b\lambda + c$	7.2010	-13.840	0.9292		$1.953 \cdot 10^{-6}$	18.3
$a(b + \lambda)^c + d$	-5.9623	-0.0199	0.9306	-1.1626	$9.678 \cdot 10^{-8}$	-0.043

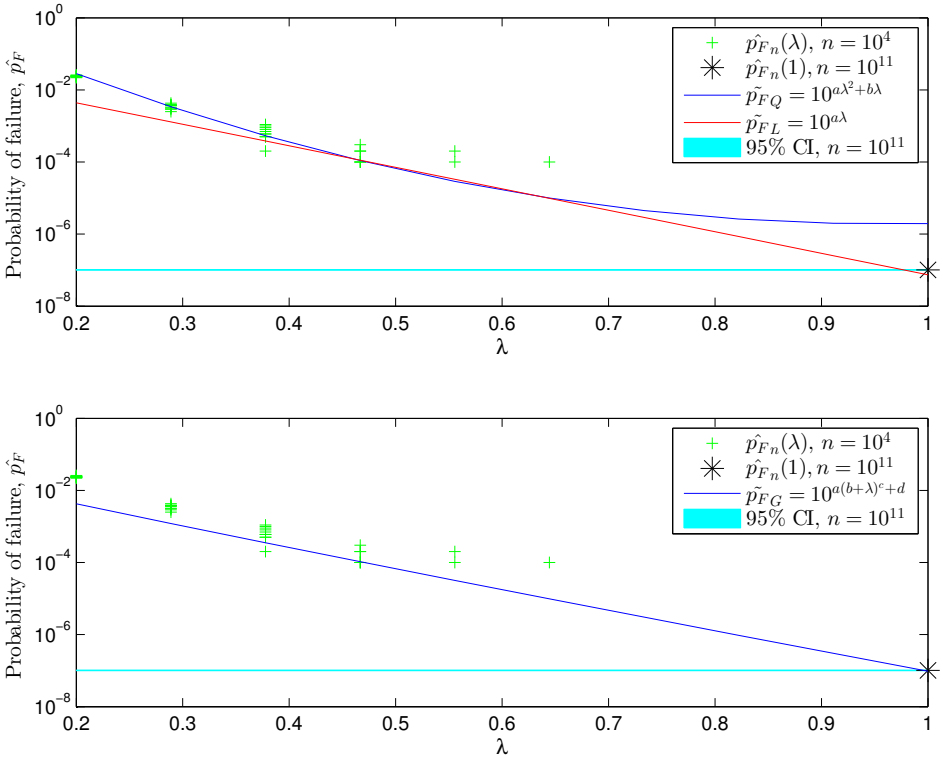


Figure 4.23: Logarithmic plot of the different fits of the simulated probability failure, $\tilde{p_F}$, as a function of λ. Simulations are done in 10 samples with $n = 10^4$ for the model described in section 4.3, $w_i = \text{weightQ}$ and $\lambda(1) = 0.2$. Original model is obtained for $\lambda = 1$.

Table 4.34: Comparison of the three fits for the system described in section 4.3. $p_1 = p_2 = p_3 = p_4 = 1 - 10^{-7}$ and $\lambda = \text{linspace}(0.2, 10, 1)$. The system is simulated by 10 samples with $n = 10^4$. The least squares fit is unweighted. $\delta_{\tilde{p_F}(1)} = \frac{\tilde{p_F}(1) - \hat{p_{F\,N}}(1)}{\hat{p_{F\,N}}(1)}$, where $\hat{p_{F\,N}}(1) = 1.012 \cdot 10^{-7}$ is the probability to fail estimated by original Monte Carlo simulation with sample size $N = 10^{11}$.

$\log_{10}(\tilde{p_F}(\lambda))$	a	b	c	d	$\tilde{p_F}(1)$	$\delta_{\tilde{p_F}(1)}$
$a\lambda + b$	-6.8406	-0.5424			$4.140 \cdot 10^{-8}$	-0.591
$a\lambda^2 + b\lambda + c$	10.968	-17.078	1.4496		$2.456 \cdot 10^{-5}$	242
$a(b + \lambda)^c + d$	-12.8815	0.4242	0.4904	8.3860	$1.163 \cdot 10^{-7}$	0.149

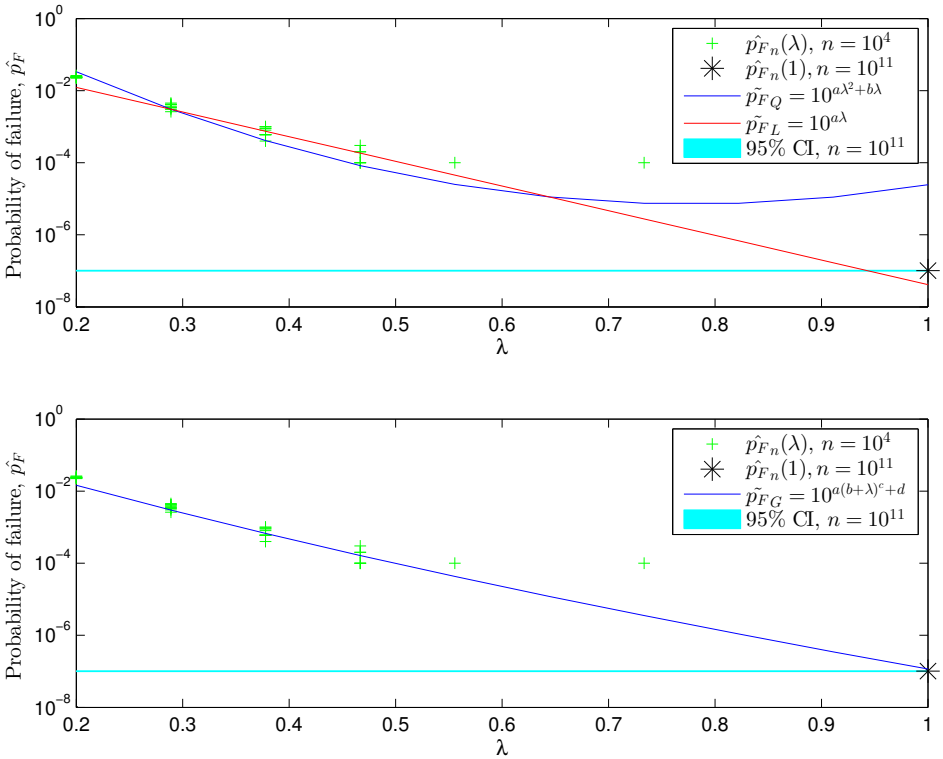


Figure 4.24: Logarithmic plot of the different fits of the simulated probability failure, $\tilde{p_F}$, as a function of λ. Simulations are done in 10 samples with $n = 10^4$ for the model described in section 4.3, The fit is unweighted and $\lambda(1) = 0.2$. Original model is obtained for $\lambda = 1$.

Chapter 5

Concludinig remarks

The result shows that the we can estimate the probability of failure efficiently and accurately by using Monte Carlo simulations on parametrized systems. The sample size can be reduced from 10^8 in standard Monte Carlo simulation to 10^5, and still achieve results with the same precision. The parametrization seems to work good for a wide range of model types.

The results from the independent system shows us that the weighting strategy weightN has the best results, and that the linear and "general fit" defined in (3.35) are the most accurate. The linear fit seems to be the most accurate for many models, and especially for the system with probability to fail, $p_F = 10^{-12}$. The linear fit is the easiest of the three to implement and optimize. There has been made two changes in the implementation of the general fit from the project assignment [2]. The first change is the starting point in the optimization. This has been changed to be the initial condition [0,0,1,0]. In the project assignment, we used knowledge about the systems to define a more describing initial condition. The other change that has been done is to change the tolerance function 'TolFun' in MATLAB. This denotes how much the parameters estimated [a, b, c, d] should differ from the optimum. We reduced this tolerance to 10^{-4}, which gave very good results for the general fit. These two changes results in a general fit that is more stable and more accurate than it was in the project assignment. The general fit is however the hardest fit to optimize.

In the results from the dependent systems, there was a difference in the achieved accuracy for systems with cascading failures and systems with common cause failures. Estimation of the parametrized cascading failures had the same precision as the independent systems, but the system with common cause failures did not get good estimations. The best fits for the common cause systems were the ones with a weighting strategy we did not expect: The unweighted fit, and the fit with weightQ from algorithm 4. This indicates that the parametrization may not be suitable as it was defined. It seems like the parametrized system had to little fails, and that the linear and general fits were estimated to be too small in every run. The general

fit seemed to be the best for many of the systems with cascading failures. This might be because this fit has more flexibility than the linear fit, and that it is able to make a better extrapolation because of this.

The different weights introduced in this thesis shows that our strategy, weightN is a stable and accurate weighting to choose. The other weights have been varying a lot in accuracy and stability. The weighting strategy weightQ should especially be avoided.

5.1 Further Work

It would be interesting to parametrize more dependent systems, systems that cover a wide range of practical applications. The systems with common cause failures may be improved, perhaps by letting the parameter λ affect the β parameter introduced for the systems. Different conditions in the systems with cascading failures can be further investigated.

The estimation of some of the systems were dependent of the starting point of λ. It would be interesting to do a more thorough analysis of the effects of changing λ. This could be done together with an analyse of the optimal sample size for different systems.

The accurate result of the system with probability to fail, $p_F = 10^{-12}$, was very surprising, and it would be interesting to investigate which type of systems that can be estimated so accurately despite the low sample size used 10^5.

A further task may be to establish an understanding of how to determine the reliability of a component, based on *mean residual life, failure rate* and other concepts from Rausand and Høyland [8]. It is also possible to include human interaction to the models we have. That there can be human errors, either in repairing or maintaining systems.

Further work may also consist of a framework to ease the implementation of systems. A framework that can handle dependencies and different structures, and automatically generates how the different components affects the system.

Bibliography

[1] Berghen F., V., Levenberg-Marquardt algorithms vs Trust Region Algorithms, 2004

[2] Bø, H. S., Estimation of Reliability by Monte Carlo Mathods, 2014

[3] Canty, A.J., Weighted Least Squares, http://ms.mcmaster.ca/canty/teaching/stat3a03/Lectures7.pdf

[4] Casella, G. & Berger, R.L., *Statistical Inference*, Brooks/Cole, 2nd Edition, 2002.

[5] Gille-Genest A., Introduction to the Monte Carlo Methods, 2012 https://quanto.inria.fr/pdf_html/mc_standard_doc/

[6] Huseby, B., Naustdal, M., Vårli, I.D., System Reliability Evaluation using Conditional Monte Carlo Methods

[7] Kalos, M.H. & Whitlock P.A., *Monte Carlo Methods*, Wiley, 2nd Edition, 2008.

[8] Rausand, M. & Høyland, A., *System Reliability Theory*, Wiley, 2nd Edition, 2004.

[9] Ross, S. M., *Introduction to Probability Models*, Elslevier, 9th Edtition, 2007

[10] Sigman, K., Introduction to reducing variance in Monte Carlo simulations, http://www.columbia.edu/ ks20/4703-Sigman/4703-07-Notes-ATV.pdf

[11] Yu, J. Buffon's Needle and The Monte Carlo Method, 2012, http://www.slideshare.net/ihatetheses/buffon-needle-and-the-monte-carlo-method

[12] Optimization Toolbox, MATLAB, http://nf.nci.org.au/facilities/software/Matlab/toolbox/optim/lsqcurvefit.html

[13] Tabeller og formler i statistikk, Institutt for matematiske fag, NTNU

Appendix A

Matlab Code

This example of simulating the model defined in section 3.4.5 is included to make the implementation more understandable.

The function `parasysrem.m` is the standard Monte Carlo simulation of the model defined in section 3.4.5.

Listing A.1: Parasysrem

```
%%This function counts the fails and the estimated probability to fail ↩
    for
%%the model defined in section 3.4.5.
function [fails, estimated] = parasysrem(n,p,A)
% Input:
% n: Sample size
% p: Reliability of components
% A: Generated matrix of (n,3) random variables
%Output:
% fails: number of fails counted simulated
% estimated: estimated probability to fail

% Initializing

fails = 0;
p1=p;
p2=p;
p3=p;
j = 0;

% For loop that counts the fails
for i = 1:n
    j = j+1;
    % All components in failed state
    if A(i,1)>p1&& A(i,2)>p2 && A(i,3)>p3
        fails = fails + 1;
        j = 0;
    end
    % Component 1 failed and is removed
    if A(i,1)>p1
        p1 = 0;
    end
    % Component 2 failed and is removed
    if A(i,2)>p2
        p2 = 0;
    end
```

```
    % Component 3 failed and is removed
    if A(i,3)>p3
        p3 = 0;
    end
    % This will be true if j = 0 (fail in system) or j = a*10^z, where z ↩
        is
    % the exponent in the reliability p = 1 - 10^-z of the component and ↩
        a
    % is an integer. It is rounded to the nearest integer.
    if ~mod(j,round(1/(1-p)))
        p1=p;
        p2=p;
        p3=p;
    end

end

estimated = fails/n;

end
```

This script utilizes the function `parasysrem.m` to get simulations for the parametrized model. The different weights are calculated and the different fits are estimated.

Listing A.2: Compareweights in parasysrem

```
m = 10;              % Number of simulation samples
Ls = 10;             % Number of entries in the parameter lambda
z = 7;               % z in p= 1-10^-z, where p is the component ↩
    reliability
n = 10000;           % Sample size in each of the m samples
lambda = linspace(0.05,1,Ls);  % Vectorize the parameter lambda
plambda = 1-10.^-(z*lambda);   % Calculate the parametrized reliability
L = length(lambda);
A=rand(s*n,3);                 % Generate random s*n variables U(0,1) for↩
    3
                               % components

% Initializing matrices that counts number of fails and estimated
% probability to fail for the tested system
fails = zeros(s,L);
estimated = zeros(s,L);

% Double loop that testes all random variables, and returns two matrices:
% The number of fails in every sample for every lambda
% The estimated probability to fail in every sample for every lambda
% The tested system is now parasysrem
    for j = 1:m
        for i = 1:L
        [fails(j,i), estimated(j,i)] = ...
            parasysrem(n,plambda(i),A((j-1)*n+1:j*n,:));
        end
    end


% Calculate the mean of the simulation results and coefficient of ↩
    variance
% for weightN
meanestimated = sum(estimated)/s;
covi = sqrt((1./(meanestimated.*(1-meanestimated)))./n);

% Calculate the sum of the fails for each lambda and the coefficient og
% variance for weightN2
sumfails = sum(fails);
covi2 = std(fails)./sqrt(sumfails);
```

```
% Initializing weights
k = 1;
NZ = nnz(meanestimated);
weightN = zeros(1,NZ);
weightN2 = zeros(1,NZ);

% Initialize the vector of simulated fails with correct length
posestimated = zeros(1,NZ);
poslam = zeros(1,NZ);


% Confidence interval for weightN
CINplus = meanestimated.*(1 + 1.96*covi);
CINminus = meanestimated.*(1 - 1.96*covi);

% Confidence interval for weightN2
CINplus2 = meanestimated.*(1+1.96*covi2);
CINminus2 = meanestimated.*(1-1.96*covi2);


% Calculate the weights weightN and weightN2 for the values of lambda
% where the weights should be defined. The implementation has special
% cases when CI- is negative

for i = 1:L
    if meanestimated(i) ~= 0
        posestimated(k) = log10(meanestimated(i));
        poslam(k) = lambda(i);
        if i == 1 && CINminus(1) < 0
            weightN(k) = 1/(log10(CINplus(i)) -...
                real((log10(CINminus(i)))))^2;
        elseif i == 1 && CINminus(2) < 0
            weightN(k) = 1/(log10(CINplus(i)) -...
                real((log10(CINminus2(i)))))^2;
        else
            if CINminus(i)<0
                weightN(k) = weightN(k-1)/(sumfails(k-1)/sumfails(k));
            else
                weightN(k) = 1/(log(CINplus(i))-((log(CINminus(i))))^2;
            end
            if CINminus2(i)<0
                weightN2(k) = weightN2(k-1)/(sumfails(k-1)/sumfails(k));
            else
                weightN2(k) = 1/(log(CINplus2(i))-((log(CINminus2(i))))↩
                    ^2;
            end

        end
        k = k+1;
    end
end

%Calculate the weights weightQ and weightS
weightQ = 1./var(fails(:,1:NZ));
weightS = mean(fails(:,1:NZ))./std(fails(:,1:NZ));

%Special conditions if std(fails(:,i)) is zero
for i = 1:NZ
    if std(fails(:,i)) == 0
        weightS(i) = mean(fails(:,i));
        weightQ(i) = mean(fails(:,i));
    end
end

%Normalizeing the weights
weightS = weightS./sum(weightS);
weightN = weightN./sum(weightN);
```

```
weightN2 = weightN2./sum(weightN2);
weightQ = weightQ./sum(weightQ);


%Sets the wanted options for the lsqcurvefit-algorithm
opts = optimset('Algorithm', 'trust-region-reflective',...
    'TolFun',1e-4,'MaxFunEvals',500);


%Optimizes the linear, quadratic and general curve for weightN
x02 = [0,0];
F2x = @(x2,xdata)sqrt(weightN).*(x2(1)*xdata+x2(2));
[x2,resnorm2,~,exitflag2,output2] = lsqcurvefit(F2x,x02,poslam,...
    posestimated.*sqrt(weightN),[],[],opts);

x03 = [0,0,0];
F3x = @(x3,xdata)sqrt(weightN).*(x3(1)*xdata.^2+x3(2)*xdata+x3(3));
[x3,resnorm3,~,exitflag3,output3] = lsqcurvefit(F3x,x03,poslam,...
    posestimated.*sqrt(weightN),[],[],opts);

x04 = [0,0,1,0];
F4x = @(x4,xdata)sqrt(weightN).*(x4(1)*(x4(2)+xdata).^x4(3)+x4(4));
[x4,resnorm4,~,exitflag4,output4] = lsqcurvefit(F4x,x04,poslam,...
    posestimated.*sqrt(weightN),[],[],opts);
x4 = real(x4);

%Optimizes the linear, quadratic and general curve for weightS
y02 = [0,0];
F2y = @(y2,xdata)sqrt(weightS).*(y2(1)*xdata+y2(2));
[y2,resnorm2y,~,exitflag2y,output2y] = lsqcurvefit(F2y,y02,poslam,...
    posestimated.*sqrt(weightS),[],[],opts);

y03 = [0,0,0];
F3y = @(y3,xdata)sqrt(weightS).*(y3(1)*xdata.^2+y3(2)*xdata+y3(3));
[y3,resnorm3y,~,exitflag3y,output3y] = lsqcurvefit(F3y,y03,poslam,...
    posestimated.*sqrt(weightS),[],[],opts);

y04 = [0,0,1,0];
F4y = @(y4,xdata)sqrt(weightS).*(y4(1)*(y4(2)+xdata).^y4(3)+y4(4));
[y4,resnorm4y,~,exitflag4y,output4y] = lsqcurvefit(F4y,y04,poslam,...
    posestimated.*sqrt(weightS),[],[],opts);
y4 = real(y4);

%Optimizes the linear, quadratic and general curve for weightN2
z02 = [0,0];
F2z = @(z2,xdata)sqrt(weightN2).*(z2(1)*xdata+z2(2));
[z2,resnorm2z,~,exitflag2z,output2z] = lsqcurvefit(F2z,z02,poslam,...
    posestimated.*sqrt(weightN2),[],[],opts);

z03 = [0,0,0];
F3z = @(z3,xdata)sqrt(weightN2).*(z3(1)*xdata.^2+z3(2)*xdata+z3(3));
[z3,resnorm3z,~,exitflag3z,output3z] = lsqcurvefit(F3z,z03,poslam,...
    posestimated.*sqrt(weightN2),[],[],opts);

z04 = [0,0,1,0];
F4z = @(z4,xdata)sqrt(weightN2).*(z4(1)*(z4(2)+xdata).^z4(3)+z4(4));
[z4,resnormzy,~,exitflagzy,outputzy] = lsqcurvefit(F4z,z04,poslam,...
    posestimated.*sqrt(weightN2),[],[],opts);
z4 = real(z4);

%Optimizes the linear, quadratic and general curve for weightQ

w02 = [0,0];
F2w = @(w2,xdata)sqrt(weightQ).*(w2(1)*xdata+w2(2));
[w2,resnorm2w,~,exitflag2w,output2w] = lsqcurvefit(F2w,w02,poslam,...
    posestimated.*sqrt(weightQ),[],[],opts);

w03 = [0,0,0];
```

```
F3w = @(w3,xdata)sqrt(weightQ).*(w3(1)*xdata.^2+w3(2)*xdata+w3(3));
[w3,resnorm3w,~,exitflag3w,output3w] = lsqcurvefit(F3w,w03,poslam,...
    posestimated.*sqrt(weightQ),[],[],opts);
w04 = [0,0,1,0];

F4 = @(w4,xdata)sqrt(weightQ).*(w4(1)*(w4(2)+xdata).^w4(3)+w4(4));
[w4,resnorm4w,~,exitflag4w,output4w] = lsqcurvefit(F4,w04,poslam,...
    posestimated.*sqrt(weightQ),[],[],opts);
w4 = real(w4);
```