NTNU

Norwegian University of
Science and Technology

Nonparametric estimation in trend-renewal processes

Per Erik Haugedal

Master of Science in Physics and Mathematics
Submission date: June 2017
Supervisor: Bo Henry Lindqvist, IMF

Norwegian University of Science and Technology
Department of Mathematical Sciences

Problem Description

- Give an introduction to stochastic modeling of repairable systems, in particular the nonhomogenous Poisson process (NHPP) and the trend-renewal process (TRP).
- Study kernel-based methods for nonparametric estimation of the trend function of TRPs.
- Apply the methods to real and simulated data.

Assignment given: January 27, 2017

Supervisor: Bo Henry Lindqvist (NTNU)

Preface

This thesis is written at the Department of Mathematical Sciences at the Norwegian University of Science and Technology (NTNU) in the period January to June 2017. I would like to thank my supervisor Bo Henry Lindqvist for productive discussions and excellent guidance through both my specialization project last term and this thesis. His continuous feedback has been a great help while working on this. I would also like to thank Kjartan Kloster Osmundsen for good advice and feedback.

Trondheim, June 2017

Per Erik Haugedal

Abstract

This thesis gives an introduction to stochastic modeling of repairable systems with failure and maintenance data, in particular the nonhomogeneous Poisson process and the trend-renewal process. It is studying kernel-based methods for nonparametric estimation of the trend function of trend-renewal processes and presents a method using weighted kernel estimation. These weights are found by maximization of the likelihood function that they are included in. The method is then tested on both real and simulated data sets.

Samandrag

Denne oppgåva gjer ein introduksjon til stokastisk modellering av reparerbare system med feil- og vedlikehaldsdata, spesielt ikkje-homogene Poisson prosessar og trend-renewal-prosessar. Den studerer kjernebaserte metodar for ikkje-parametrisk estimering av trend-funksjonen i trend-renewal-prosessar og presenterer ein metode som brukar vekta kjerneestimering. Desse vektene vert funne ved maksimering av likelihoodfunksjonen som dei inngår i. Metoden vert så testa både på verkelege og simulerte datasett.

Table of Contents

List of Tables

List of Figures

Abbreviations

HPP	=	Homogeneous Poisson process
NHPP	=	Non-homogeneous Poisson process
RP	=	Renewal process
TRP	=	Trend-renewal process

Chapter 1

Introduction

1.1 Modeling repairable systems

We assume that the data from the repairable systems are given as failure times $T_1, T_2, \ldots, T_n$. When doing modeling we are looking at the interfailure times, $X_i = T_i - T_{i-1}$ as illustrated in figure 1.1.

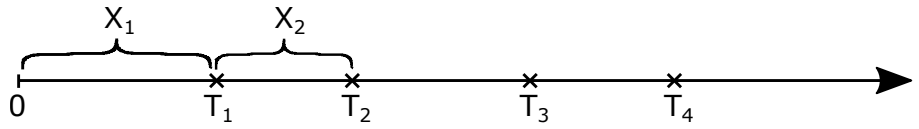

Figure 1.1: Illustrating the failure times $T_1, T_2, \ldots, T_n$ and interfailure times $X_1, X_2, \ldots, X_n$ on a timeline.

The most common models used to model the failure process of repairable systems are renewal processes (RP), homogeneous Poisson processes (HPP) and non-homogeneous Poisson processes (NHPP). If there seems to be a trend in the interfailure times, meaning that the frequency of failures are changing as time goes by, one can use a NHPP model with an intensity function $\lambda(t)$. If there does not seem to be a trend one can use a RP model. The RP model is what we call a perfect repair model, which means that after a failure the system is repaired to be as good as new. The NHPP model is what we call a minimal repair model, where the repair only restores the system to the state it was just before the failure occurred. The HPP model is a special case and corresponds to a NHPP with constant intensity λ, or it can also be seen as a RP with exponentially distributed interfailure times. The trend-renewal process (TRP), which we are working with in this thesis, is a supplement to these models that can be used for cases not satisfactory covered by the extreme cases with perfect or minimal repair in RP and NHPP. In this model we can have both a trend in the failure data, handled with the intensity function $\lambda(t)$, as well

as interfailure times other than the exponential one. All of these processes are described in [7].

In this master thesis we will be looking at nonparametric estimation of the intensity function $\lambda(t)$ of TRP's. A lot of the previous work on this topic has been done by Knut Heggland, Bo Henry Lindqvist and Maria Luz Gámiz in [3], [4], [5] and [8]. The method used in this thesis is based on the one from chapter 4 in [3], modifying it where some potential weaknesses were found. Using this method we will estimate $\lambda(t)$ of both simulated and real data sets.

Chapter 2

Theory

2.1 The Non-Homogeneous Poisson Process

The NHPP is similar to an ordinary Poisson process, with the difference that the rate of failures can change over time. This means that the NHPP has a varying intensity function $\lambda(t)$ where the ordinary Poisson process has a constant intensity λ. We denote an NHPP with intensity function $\lambda(t)$ as NHPP($\lambda(t)$). The number of failures in $(0, t]$ for a NHPP is Poisson-distributed with expectation $\int_0^t \lambda(u)du$. The NHPP can model a trend in the rate of failures. An intensity $\lambda(t)$ increasing over time corresponds to a deteriorating system, like a mechanical system aging and getting worse. An intensity $\lambda(t)$ decreasing over time corresponds to an improving system, like some software reliability getting better and better.

Let $\mathcal{F}_{t-}$ denote the history of events until time t. We then have the conditional intensity at t given the history until time t defined as

$$\gamma(t|\mathcal{F}_{t-}) = \lim_{h \to 0} \frac{\Pr(\text{failure in } [t, t+h)|\mathcal{F}_{t-})}{h}. \tag{2.1}$$

For the NHPP we have $\gamma(t|\mathcal{F}_{t-}) = \lambda(t)$, which means that the conditional intensity is independent of history. This is why the NHPP is a minimal repair model as stated in the previous chapter.

2.2 The Renewal Process

In a renewal process the interfailure times $X_1, X_2, \ldots, X_n$ are independent and identically distributed with a common distribution function F. We denote a process like this RP(F). If F is exponentially distributed then RP(F) is a Poisson process. For a renewal process RP(F) we have $\gamma(t|\mathcal{F}_{t-}) = z(t - T_{N(t-)})$, where $z(t)$ is the hazard rate of F. Here the conditional intensity only depends on the time since the previous failure, which is why we call this a perfect repair model.

2.3 The Trend-Renewal Process

The trend-renewal process is a generalization of the NHPP and RP, where we have an intensity given by $\lambda(t)$ and a cumulative intensity $\Lambda(t) = \int_0^t \lambda(u)du$. If we have an NHPP($\lambda(t)$) with failure times $T_1, T_2, \ldots, T_n$, then the time transformed process $\Lambda(T_1)$, $\Lambda(T_2)$, $\ldots$, $\Lambda(T_n)$ is a HPP(1). TRP extends this model by letting this time transformed process be any renewal process RP(F) as shown in figure 2.1. This means that the TRP has both an intensity $\lambda(t)$, also called trend function, and a distribution function F of the interfailure times of the time transformed process. This renewal distribution F is usually assumed to have expected value 1, to ensure uniqueness of the model.

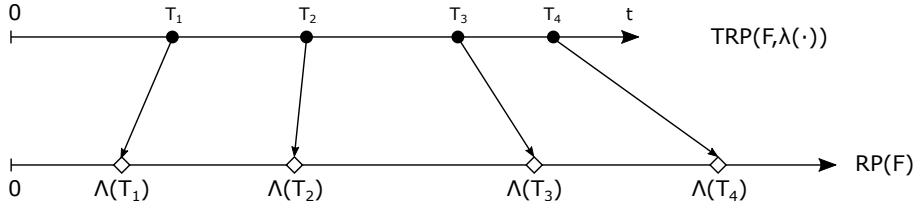

Figure 2.1: A figure illustrating the defining property of the TRP.

We are interested in the likelihood function for the TRP, and we start with a general counting process where the likelihood function is given as

$$L = \left(\prod_{i=1}^{N(\tau)} \gamma(T_i) \right) \exp\left(- \int_0^\tau \gamma(u)du \right), \tag{2.2}$$

where γ is the conditional intensity function. For the TRP we have

$$\gamma(t) = z\left(\Lambda(t) - \Lambda(T_{N(t-)})\right) \lambda(t), \tag{2.3}$$

where $z(t)$ is the hazard rate corresponding to the renewal distribution F and $T_{N(t-)}$ is the last failure before time t. If we now insert the conditional intensity function for the TRP (2.3) into the likelihood function for a counting process (2.2) we get

$$\begin{aligned} L = &\left(\prod_{i=1}^{N(\tau)} z\left(\Lambda(T_i) - \Lambda(T_{i-1})\right) \lambda(T_i) \right) \exp\left(- \sum_{i=1}^{N(\tau)} \int_{T_{i-1}}^{T_i} z(\Lambda(u) - \Lambda(T_{i-1}))\lambda(u)du \right) \\ &\times \exp\left(- \int_{T_{N(\tau)}}^{\tau} z(\Lambda(u) - \Lambda(T_{N(\tau)}))\lambda(u)du \right). \end{aligned} \tag{2.4}$$

If we now make the substitution $v = \Lambda(u) - \Lambda(T_{i-1})$, use the cumulative hazard $Z(t) = \int_0^t z(v)dv$ and take the log, we get the following log likelihood function

$$l = \log L = \sum_{i=1}^{N(\tau)} \{\log(z(\Lambda(T_i) - \Lambda(T_{i-1}))) + \log(\lambda(T_i)) - Z(\Lambda(T_i) - \Lambda(T_{i-1}))\} \\ - Z(\Lambda(\tau) - \Lambda(T_{N(\tau)})), \tag{2.5}$$

which will be the basis for the method presented later.

Chapter 3

Nonparametric estimation of $\lambda(t)$

3.1 Weighted kernel density estimation

We will now look at estimation of the intensity function $\lambda(t)$ of TRP's. First we present the method of chapter 4 in [3] and then we present a modification which turns out to work better in practice. Let the trend function $\lambda(t)$ be nonparametric and the renewal distribution $F = F(t;\beta)$ is given on parametric form with hazard rate $z(t;\beta)$ and expected value 1. The algorithm presented in [3] will maximize the log likelihood (2.5) with respect to the trend function $\lambda(t)$ and the parameter β of $z(t;\beta)$ and $Z(t;\beta)$. The idea is to iteratively maximizing with respect to $\lambda(t)$ with β fixed, and with respect to β with $\lambda(t)$ fixed alternately until convergence. The maximization with respect to β can be done by computing the time transformed interfailure times $Y_i = \Lambda(T_i) - \Lambda(T_{i-1})$ for $i = 1, 2, \ldots, N(\tau)+1$, where $T_0 = 0$ and $T_{N(\tau)+1} = \tau$, and then maximizing

$$\sum_{i=1}^{N(\tau)} \{\log z(Y_i;\beta) - Z(Y_i;\beta)\} - Z(Y_{N(\tau)+1};\beta), \tag{3.1}$$

which is just the ordinary log likelihood function for maximum likelihood estimation of β for the time transformed data.

To model the trend function $\lambda(t)$, weighted kernel density estimation will be used, and the estimator will be on the form

$$\lambda(t;a) = \frac{1}{h}\sum_{i=1}^{N(\tau)} w\left(\frac{t-T_i}{h}\right) a_i, \tag{3.2}$$

where w is a bounded density function symmetric around 0, h is a bandwidth to be chosen, and $a = (a_i; i = 1, 2, \ldots, N(\tau))$ is the weights. We will be using the Epanechnikov kernel, $w(u) = \frac{3}{4}\left(1-u^2\right)$ for $|u| \leq 1$ and $w(u) = 0$ otherwise. By substituting (3.2) into the log likelihood (2.5) and maximizing with respect to the weights a and β we can find

the optimal values for these parameters. In [3] the following approximation is suggested in order to simplify the computations

$$\Lambda(T_i; a) - \Lambda(T_{i-1}; a) \approx \lambda(T_i; a)(T_i - T_{i-1}) \equiv \lambda(T_i; a)X_i \tag{3.3}$$

for $i = 1, 2, \ldots, N(\tau) + 1$. By using this approximation and doing the substitution mentioned above the approximation of the log likelihood (2.5) now becomes

$$\begin{aligned} l_a(\beta) = \sum_{i=1}^{N(\tau)} &\{\log(z(\lambda(T_i; a)X_i; \beta)) + \log(\lambda(T_i; a)) - Z(\lambda(T_i; a)X_i; \beta)\} \\ &- Z(\lambda(\tau; a)X_{N(\tau)+1}; \beta), \end{aligned} \tag{3.4}$$

where $X_{N(\tau)+1} = \tau - T_{N(\tau)}$.

We will work with F being Weibull distributed in this thesis, and the hazard rate of a Weibull distribution with shape parameter β and expected value 1 is given by

$$z(t; \beta) = \beta[\Gamma(\beta^{-1} + 1)]^{\beta} t^{\beta-1}. \tag{3.5}$$

If we substitute (3.5) into (3.4) we can write the log likelihood as

$$\begin{aligned} l_a(\beta) =& N(\tau) \log \beta + N(\tau)\beta \log \Gamma(\beta^{-1} + 1) + \sum_{i=1}^{N(\tau)} \{\beta \log(\lambda(T_i; a)X_i) - \log X_i \\ &- [\Gamma(\beta^{-1} + 1)\lambda(T_i; a)X_i]^{\beta}\} - [\Gamma(\beta^{-1} + 1)\lambda(\tau; a)X_{N(\tau)+1}]^{\beta}. \end{aligned} \tag{3.6}$$

The algorithm starts out with all the weights $a_i = 1$ and then alternately and iteratively maximizes (3.1) with respect to β, and (3.6) with respect to the weights a_i for the given value of β.

3.2 Choosing bandwidth

The value of the bandwidth h decides how much smoothing is done in the estimation. From (3.2), with $w(u)$ being the Epanechnikov kernel, we can see we can see that at any time t the value of $\lambda(t)$ will only be affected by failure times T_i within $[t - h, t + h]$. This means that with a large bandwidth h the value of $\lambda(t)$ will be influenced by many failure times T_i at all times, and $\lambda(t)$ will be smoother since points close to each other will have most influencing failure times in common with each other. On the other hand, a too small value of h, will cause the value of $\lambda(t)$ to only be influenced by a few failure times T_i within $[t - h, t + h]$. In this case $\lambda(t)$ will be less smooth since points close to each other will have a smaller proportion of influencing failure times T_i in common. We want to choose a bandwidth h that gives a good result. If a too big h is chosen, any interesting trends will be smoothed out. With a too small value of h on the other hand the estimate of $\lambda(t)$ will become erratic, moving up and down all the time, and follow the failure times more exactly than is reasonable to assume. With that being said we find it better to choose

h a little too small rather than a little too big. The reason for this is that one can always do a little bit of smoothing by eye just from looking at the graph of $\lambda(t)$, however you can not see details that has been smoothed out by a large h.

Choosing bandwidths in this thesis were mostly done by trial and error, often guided by the automatic value of h found by the function *density* in R.

3.3 Final method

While working with the method of chapter 4 in [3] it was gradually modified either as problems occurred or to get more accurate estimations. These issues are described in the next chapter, here we present the final method we ended up using. One small change was just to not use the approximation (3.3) in order to get more accurate results. This means that the log likelihood (2.5), with the hazard rate (3.5), now becomes

$$\begin{aligned} l_a(\beta) =& N(\tau)\log\beta + N(\tau)\beta\log\Gamma(\beta^{-1}+1) + \sum_{i=1}^{N(\tau)}\{(\beta-1)\log(\Lambda(T_i;a)-\Lambda(T_{i-1};a)) \\ &+ \log(\lambda(T_i;a)) - [\Gamma(\beta^{-1}+1)(\Lambda(T_i;a)-\Lambda(T_{i-1};a))]^\beta\} \\ &- [\Gamma(\beta^{-1}+1)(\Lambda(\tau;a)-\Lambda(T_{N(\tau)};a))]^\beta. \end{aligned} \tag{3.7}$$

Here we have that

$$\Lambda(t;a) = \int_0^t \lambda(u;a)du = \frac{1}{h}\int_0^t \sum_{i=1}^{N(\tau)} w\left(\frac{u-T_i}{h}\right)a_i du, \tag{3.8}$$

and if we substitute $y = \frac{u-T_i}{h}$ and have $dy = \frac{du}{h}$ we get that

$$\Lambda(t;a) = \int_{\frac{-T_i}{h}}^{\frac{t-T_i}{h}} \sum_{i=1}^{N(\tau)} w(y)a_i dy = \sum_{i=1}^{N(\tau)}\left(W\left(\frac{t-T_i}{h}\right) - W\left(\frac{-T_i}{h}\right)\right)a_i, \tag{3.9}$$

where $W(t) = \int_{-\infty}^{u} w(u)du$. We are using the Epanechnikov kernel, which means that in this case we have

$$W(t) = \begin{cases} \int_{-1}^{t} \frac{3}{4}(1-u^2)du = \frac{-t^3+3t+2}{4} & \text{for } |t| \le 1, \\ 0 & \text{for } t < -1, \\ 1 & \text{for } t > 1. \end{cases} \tag{3.10}$$

The other modification to the method was to simply use (3.7) to find the optimal values for both β and the weights a_i at the same time, instead of the iterative method from [3] described above.

Chapter 4

Real data sets and simulation studies

4.1 Maximizing the likelihood function in R

4.1.1 First approach

All the functions used within the log likelihood functions (3.1) and (3.6), like $\lambda(t;a)$, $\Lambda(t;a)$, $w(t)$ and $W(t) = \int_{-\infty}^{t} w(u)du$ are implemented as functions in R [9], shown in appendix B. This makes it so that the Optim function [10] in R can be used to maximize these log likelihoods directly. The first approach was to follow the mentioned algorithm in chapter 4 in [3], and using Optim to iteratively maximize these two log likelihood functions. It would start out with all the weights $a_i = 1$, maximize (3.1) to get a first estimate of β and then use this value for β to maximize (3.6) and get new estimates of the weights a_i. Then it would run this alternately using the previous estimate until convergence.

4.1.2 Final approach

In the final approach there is only one call of Optim done, to maximize (3.7) with respect to both β and the weights a_i at the same time. Same as for the first approach all the weights a_i are given starting value 1, but now we also need an initial value for β. This was often just set to 1 as well, unless it was assumed that it would have a value in some other range. This choice turned out to not make any difference in practice.

4.1.3 Computation time

Since the maximization of (3.6) is optimizing a number of weights equal to the number of failures, in addition to the parameter β of the Weibull renewal distribution, the computation time obviously increases rapidly with increasing sizes of data sets. Some different approaches for the optimization were considered, and several of the different optimization

algorithms available in the Optim function in R were tried. The weights a_i, and the parameter β, needs to be non-negative, and the first solution to this was to use the method "L-BFGS-B" [10] in Optim, where you can give lower and upper bounds for the parameters. The cost you pay for this is that it is a slow method. Some other approaches were explored to allow for usage of quicker methods. For instance the idea of implementing a parameter substitution to take care of the non-negative restraint. However this brought up some other problems. Using the substitution $a_i = e^{u_i}$ would not allow for weights being equal to zero, which we will see later is important. Trying the substitution $a_i = u_i^2$ would not give unique solutions, which could cause problems. These other approaches were therefore discarded, and the slow method "L-BFGS-B" with lower and upper bounds in R were used.

4.2 Modifying the method

We started simple by simulating some data from a TRP with a Weibull renewal function and all parameters known. At some point during the process we came across a set of simulated failure times that did not converge using the first approach explained above. The estimate for β was oscillating between two different values, and hence the estimates for the weights a_i was oscillating between two sets of values. We calculated the profile log likelihood of β for some values of β and manually found the approximate optimal value for β, which was in between the two values it was oscillating between. We then tried feeding in a starting value for β to the algorithm that was close to this value, but that did not solve the problem. The next idea was to only use the log likelihood (3.6) and maximize it with respect to both β and the weights a_i at the same time, removing the need for the alternating iteration. This new method gave an estimate of β that was concurring with the one found by the profile likelihood test. This method will therefore be used in the following.

4.3 Data sets

In this chapter we present modeling done with both real and simulated data sets. We look closely at how $\lambda(t)$, β and the weights a_i behave for different data sets and different bandwidths h.

4.3.1 U.S.S. Halfbeak diesel engine

The first data set we look at is failure times in operating hours for the number 3 main propulsion engine of the submarine U.S.S. Halfbeak [2]. The failure times themselves are presented in table 4.1.

Figure 4.1 shows the estimated $\lambda(t)$ using the final method with optimizing both β and the weights a_i at the same time, and also not using the approximation (3.3). The time axis is here scaled down by a factor of 1000. In this figure we have presented $\lambda(t)$ estimated with three different bandwidths to illustrate the smoothing effect. We can see how the blue line with $h = 2$ is moving up and down quite a bit and the highest peak is pretty pointy.

Table 4.1: U.S.S. Halfbeak failure times

1382	2990	4124	6827	7472	7567	8845	9450	9794
10848	11993	12300	15413	16497	17352	17632	18122	19067
19172	19299	19360	19686	19940	19944	20121	20132	20431
20525	21057	21061	21309	21310	21378	21391	21456	21461
21603	21658	21688	21750	21815	21820	21822	21888	21930
21943	21946	22181	22311	22634	22635	22669	22691	22846
22947	23149	23305	23491	23526	23774	23791	23822	24006
24286	25000	25010	25048	25268	25400	25500	25518	

The green line with $h = 10$ on the other hand is very smooth, with less variance in the first part and much more blunt peak in the last part. We can see how this green line has lost some of the details due to the smoothing. The red line with $h = 5$ is naturally somewhere in between the two other lines. In this figure we can see the point made earlier that the graph of $\lambda(t)$ with a choice of a small bandwidth can be smoothed out by eye. If we look at the blue line, and just smooth out the small kinks by eye, it quickly becomes very similar to the red line. We can not however get from the smooth green line to the red or blue line just by changing it by eye, we would need to do the actual estimation with a smaller h.

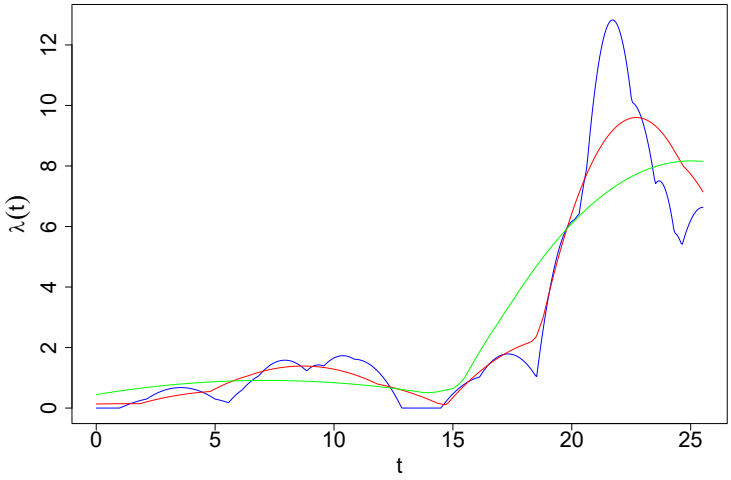

Figure 4.1: $\lambda(t)$ estimated from the U.S.S. Halfbeak data set with bandwidths $h = 2$, $h = 5$ and $h = 10$. The time axis is scaled down by a factor of 1000.

The estimated values of β can be found in table 4.2 and the values of the weights a_i are shown in table 4.3 for bandwidth $h = 2$ and in tables A.1 and A.2 in appendix A for bandwidths $h = 5$ and $h = 10$. A plot of the weights with bandwidth $h = 5$ is also shown in figure 4.2. We see that a large proportion of the weights have value 0. These results matches with what is noted by Jones and Henderson in [6] where they say only a

few weights will be nonzero and they will be clustered around common values of the T_i. It is possible that some of these weights that has a very low value would also become 0 if a lower tolerance was used in the maximization of the log likelihood.

Table 4.2: Estimated values of β for the U.S.S. Halfbeak data set.

	$h = 2$	$h = 5$	$h = 10$
β	0.959	0.908	0.868

Table 4.3: Values of the weights a_i for the U.S.S. Halfbeak data set with bandwidth $h = 2$.

0	0.98	0.98	0	0	3.28	1.33	0	0
4.30	0	0	0	2.83	0	0	2.88	0
0	0	0	0	0	0	0	0	0
17.30	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	13.29	6.63	6.49	0.74	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	7.57	10.12	

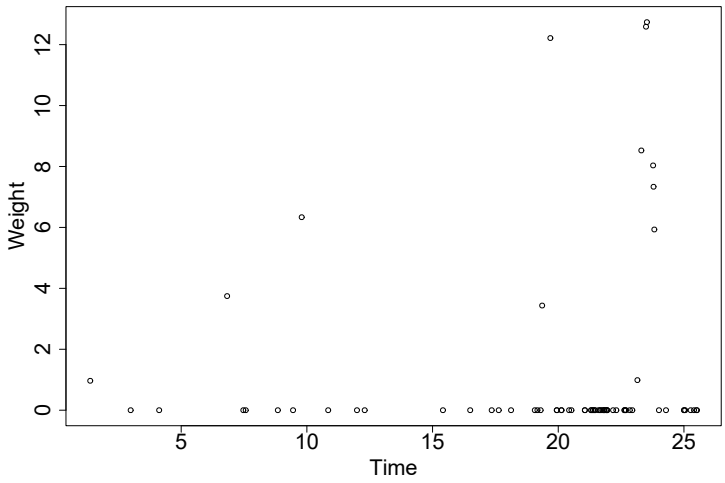

Figure 4.2: Plot of the weights with the U.S.S. Halfbeak data set using bandwidth $h = 5$.

4.3.2 U.S.S. Grampus diesel engine

Next we look at failure data from another submarine, called U.S.S. Grampus [2]. The failure times is in operating hours of unscheduled maintenance actions for one of its diesel

engines and can be found in table A.3 in appendix A. There is a pair of failures with the same failure time 14 173, and that does cause some problems for the method used here. These issues will be addressed more thoroughly for the next data set, and we just note that for this data set one of these failure times was simply changed to 14 174. Such a small change in one out of 56 failure times has no practical influence on the estimates done. We did a similar study of this data set as of the Halfbeak data set, comparing the effects of different choices of bandwidths. A plot of the different estimated $\lambda(t)$ is shown in figure 4.3, with the time axis scaled down by a factor of 1000.

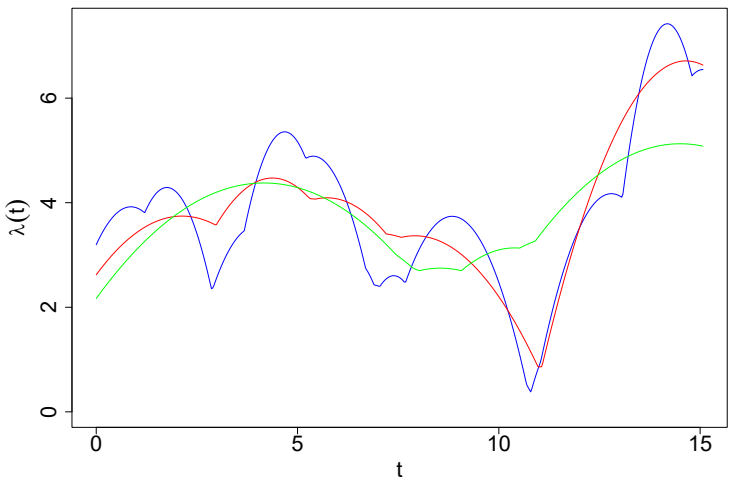

Figure 4.3: $\lambda(t)$ estimated from the U.S.S. Grampus data set with bandwidths $h = 2$, $h = 4$ and $h = 6$. The time axis is scaled down by a factor of 1000.

In this case with Grampus, $\lambda(t)$ is very different from the case with Hafbeak in figure 4.1. Here the estimate is swinging up and down around the same value over the whole area compared to the low start and high peak towards the end for the Halfbeak data set. The fact that the estimated values of β is approximately 1, see figure 4.4, and $\lambda(t)$ closer resembling a constant, especially for higher values for the bandwidth, one could possibly consider this to be a homogeneous Poisson process.

Table 4.4: Estimated values of β for the U.S.S. Grampus data set.

	$h = 2$	$h = 4$	$h = 6$
β	1.122	1.072	1.022

The values of the weights a_i are found in tables A.4, A.5 and A.6 in appendix A for bandwidths $h = 2$, $h = 4$ and $h = 6$ respectively. Again there are a lot of weights equal to 0, and the nonzero weights are clustered around common values of the T_i.

4.3.3 Photocopier

The next data set we look at is age in days of a photocopier at 92 successive failures shown in table A.7. This data set was gathered from [1]. One thing to note about this data set is that the failure times are only recorded in whole days, so there are several occurrences of two failures at the same time. This is is causing trouble with the modeling method used here as we will get $\log(0)$ in the log likelihood function (3.7). Two different workarounds were used for this problem. One workaround was to just push every second failure on a single day forward to the next day. The second workaround was to assume that two failures in one day would be the same failure twice, where it had not been repaired properly the first time, and therefore just count it as one single failure. The resulting estimates of $\lambda(t)$ with these two methods, both with bandwidth $h = 255$, is shown in figure 4.4. We can see that the estimate of $\lambda(t)$ with the method of counting double failures as a single failure is lower on average than the one moving the second failures forward in time. This makes sense because by combining failures it will have less failures overall in the same time period, and thus lower intensity. The estimated values for β were 0.99 for the method moving failures, and 1.06 for the method combining failures. The values of the weights a_i are found in table A.8 and A.9 in appendix A for the first and second methods respectively. In addition a graphical plot of the weights with the first method is shown in figure 4.5. Here we can visually see that many of the weights are 0 and the others clusters together.

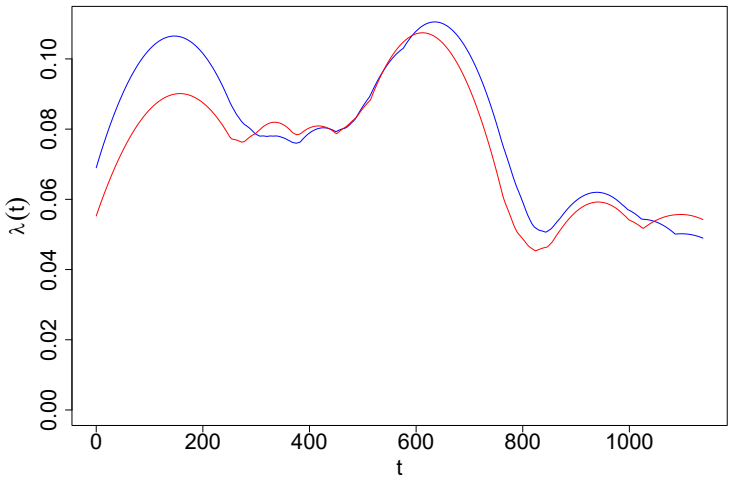

Figure 4.4: Photocopier data, $h = 255$. Blue line is with second failure in a single day is moved 1 day forward, red line with two failures in a single day counted as one. Blue estimated $\beta = 0.99$, red estimated $\beta = 1.06$.

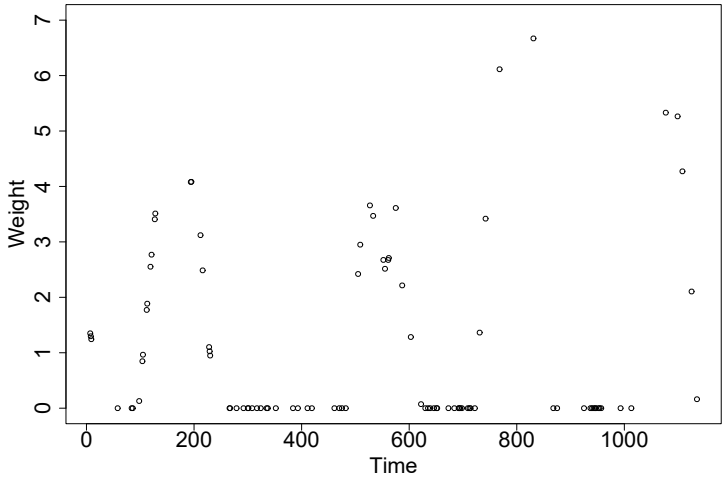

Figure 4.5: Plot of the weights with the photocopier data set using the method of pushing every second failure on a single day one day forward in time.

4.3.4 Simulated data sets

One big data set

In order to explore the edge effects of kernel density estimation a data set was simulated from a TRP with constant $\lambda = 1$, $\beta = 2$ and $\tau = 150$. $\tau = 150$ means that the simulation runs until a failure occurs after time $\tau = 150$. This last failure is not included in the data set. With a constant $\lambda = 1$ the expected number of failures is just τ itself, and in this particular simulation we ended up with 146 failures times, as shown in table A.10 in appendix A. This data set should be suited to see how the estimate of $\lambda(t)$ behaves close to the edges. What happens in regular kernel density estimation without weights (all weights equal to 1) is that the value is underestimated within the bandwidth of each of the edges. This happens because there are no failures before the start points or after the endpoint that can contribute to the estimate in these areas. One solution is to mirror all failure times within the bandwidth h of each edge illustrated in figure 4.6. This means that for any

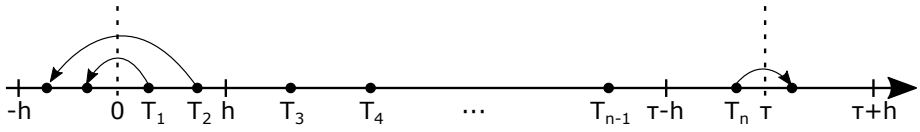

Figure 4.6: Illustration of the mirroring edge correction.

failure at T_i in $[0, h]$ a failure is added at $-T_i$ and for any failure T_i in $[\tau - h, \tau]$ a failure is added at $2\tau - T_i$. Then the estimate is calculated within the edges as if these new added

failures also had happened. The results of this for the simulated data set is shown in figure 4.7. Bandwidth $h = 20$ was used here.

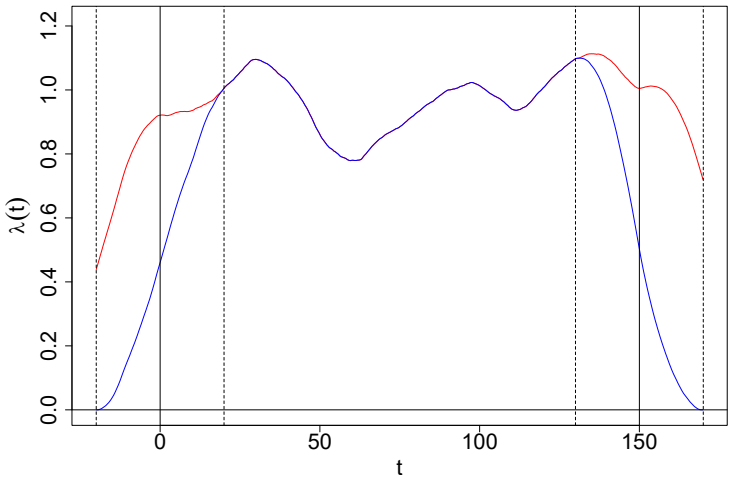

Figure 4.7: All weights $a_i = 1$. Red line is with failures within $h = 20$ from 0 or $\tau = 150$ mirrored for edge correction. The data set was simulated from a TRP with constant $\lambda = 1$.

As mentioned, this was simulated data from a TRP with constant $\lambda = 1$, and we can see how the blue line is clearly underestimated at the edges. We also see how the edge corrected red line is approximately 1 over the whole area 0 to 150 as it should be.

In figure 4.8 is the results of estimating $\lambda(t)$ with optimized weights shown, also with bandwidth $h = 20$. These weights can be found in table A.11 in appendix A. Although we here also see a lot of zero valued weights, there are more weights with a non-zero value here than what we have seen in the previous data sets. There are also quite a few values that are very close to zero. The reason for this is probably the size of this data set, and number of parameters being optimized at the same time, resulting in Optim not quite finding the optimal solution. If we had changed the tolerance and maximum iterations allowed in the Optim call, we might have seen results similar to the previous ones. But this method was already running very slow with this many parameters, so we left it like this. If we look at figure 4.8 we see that we do not get the same underestimation near the edges as for the blue line with all weights $a_i = 1$ in figure 4.7. It seems that in the process of optimizing the weights to maximize the log likelihood function (3.6) it somewhat counteracts this effect. We see that it would not make sense to use the mirroring method at the edges here.

Several smaller data sets

120 smaller data sets with $\tau = 50$ were simulated, also with constant $\lambda = 1$ and $\beta = 2$, to study the estimates of $\lambda(t)$ and β more closely. Bandwidth $h = 12$ were used here. A plot of the average estimated $\lambda(t)$ over the 120 data sets are shown in figure 4.9. Here

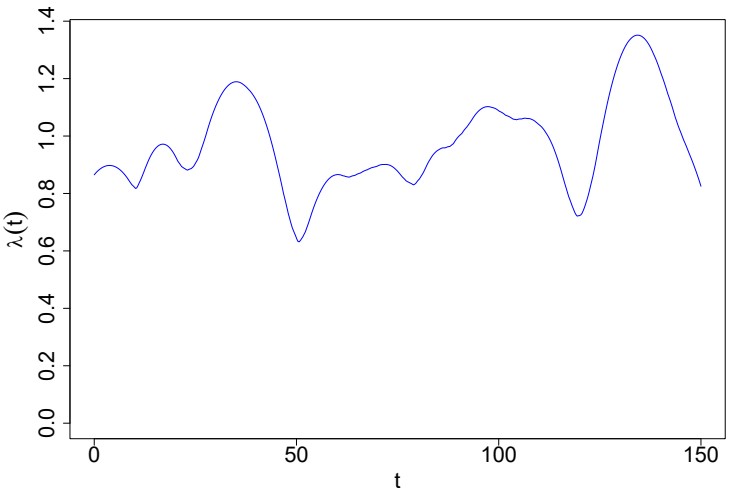

Figure 4.8: Same data set used as in figure 4.7, with optimal weights now applied. Mirroring edge correction does not make sense anymore.

the edge effect becomes even more clear than before. We see how the red line, estimates without weights, is clearly underestimating $\lambda(t)$ near the edges. The blue line, estimates with weights applied, does not have nearly as much of that underestimating trend near the edges, and swings up and down around $\lambda = 1$ over the whole interval. The average value is $\overline{\lambda} = 0.997$ of the blue line and $\overline{\lambda} = 0.907$ of the red line. This means that overall the estimates with weights applied are very close to the $\lambda = 1$ all the data were simulated from. However it seems this blue line gets some specific trend in the fluctuation around $\lambda = 1$ from the weights being applied, and not give as accurate estimate in the middle area.

In figure 4.10 is a histogram of all the estimated values of β in the 120 simulated data sets shown. Overall the values of β were a little overestimated compared with the value the data were simulated from with $\beta = 2$. Over all the 120 estimated $\beta's$, the mean value was 2.25 and median 2.20. This overestimation of β seemed to be bigger for smaller data sets, and smaller for bigger data sets.

To investigate how the choice of bandwidth influences these estimates we did the same again, simulated 120 data sets with $\tau = 50$, constant $\lambda = 1$ and $\beta = 2$, but we used bandwidth $h = 20$ for the estimation. The plot of this average estimated $\lambda(t)$ is shown in figure 4.11. Now with a larger bandwidth the areas near the edges where the red line, without weights applied, underestimates are obviously larger as well. And there seems to be an even bigger difference overall between the two lines. This is confirmed by looking at the mean values for $\lambda(t)$ in this plot, which is $\overline{\lambda} = 0.998$ for the blue line and $\overline{\lambda} = 0.850$ for the red line. Again, the overall average of the weighted line is nearly spot on the actual value $\lambda = 1$ used in the simulations. The larger bandwidth, and thus larger areas of underestimation for the red line, means that the overall average for this line gets even

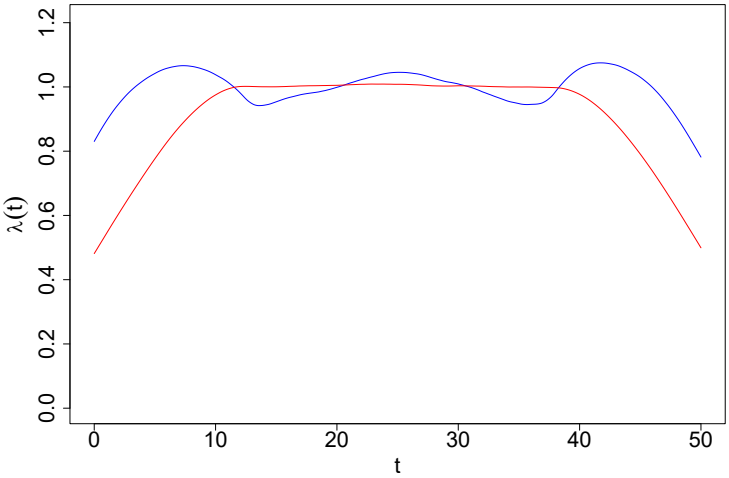

Figure 4.9: Average estimated $\lambda(t)$ over the 120 simulated data sets with $\lambda = 1$, $\beta = 2$ and $h = 12$. Blue line is with optimal weights applied, red line is with all weights $a_i = 1$.

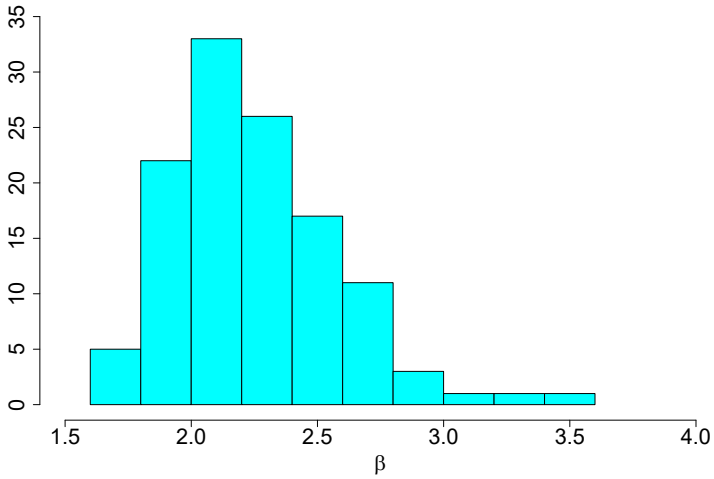

Figure 4.10: Histogram of the estimated values of β in the 120 simulated data sets with $\beta = 2$ and $h = 12$. Mean = 2.25, median = 2.20.

smaller now.

A histogram of the estimated values of β in these 120 simulated data sets is shown in figure 4.12. This looks very similar to the histogram in figure 4.10, but without any estimate exceeding 3. The mean value of all the estimated $\beta's$ were here 2.21, and median

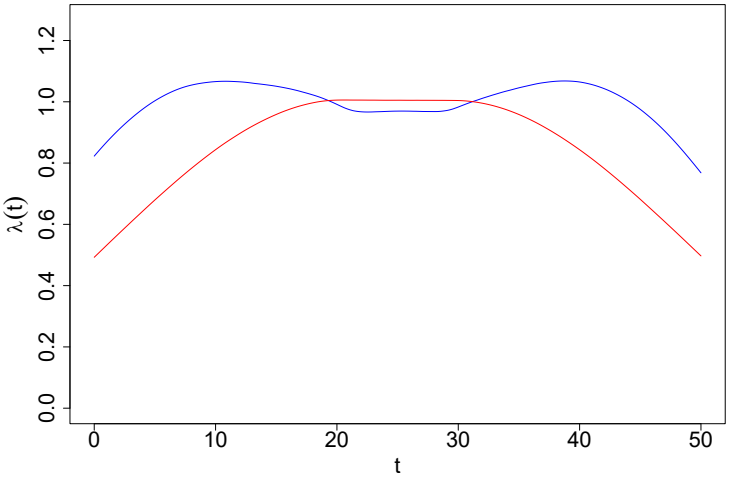

Figure 4.11: Average estimated $\lambda(t)$ over the 120 simulated data sets with $\lambda = 1$, $\beta = 2$ and $h = 20$. Blue line is with optimal weights applied, red line is with all weights $a_i = 1$.

2.18. The median is 0.02 less that of figure 4.10, and the mean is 0.04 less. This change is probably because of fewer really big estimates of β.

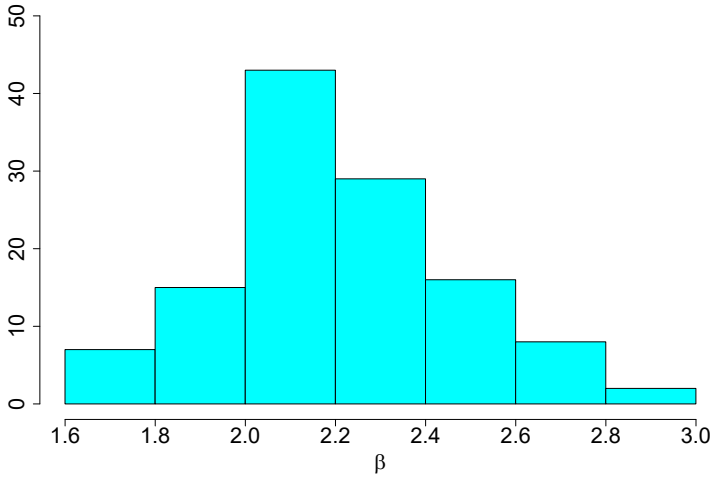

Figure 4.12: Histogram of the estimated values of β in the 120 simulated data sets with $\beta = 2$ and $h = 20$. Mean = 2.21, median = 2.18.

We also tried changing the value of β in the simulated data sets to see what effect this

would have, first we tried with $\beta = 0.8$. We still used $\tau = 50$ to have the same number of expected failures as before, but with bigger variance because of the smaller value of β. The number of failures in the time interval plays a big role when choosing the bandwidth, and it gets harder to generalize one value for the bandwidth when there now is bigger variance between the data sets, but for this test we will go back to $h = 12$. So again $\lambda(t)$ and β were estimated from 120 simulated data sets, now as said with the value $\beta = 0.8$ and the other values as before $\tau = 50, \lambda = 1$ and $h = 12$. The resulting average estimated $\lambda(t)$ can be seen in figure 4.13. It is similar to the results in figure 4.9, with $\beta = 2$ and $h = 12$, though the symmetries are not quite as clear. This probably has to do with the higher variance in the interfailure times with $\beta = 0.8$.

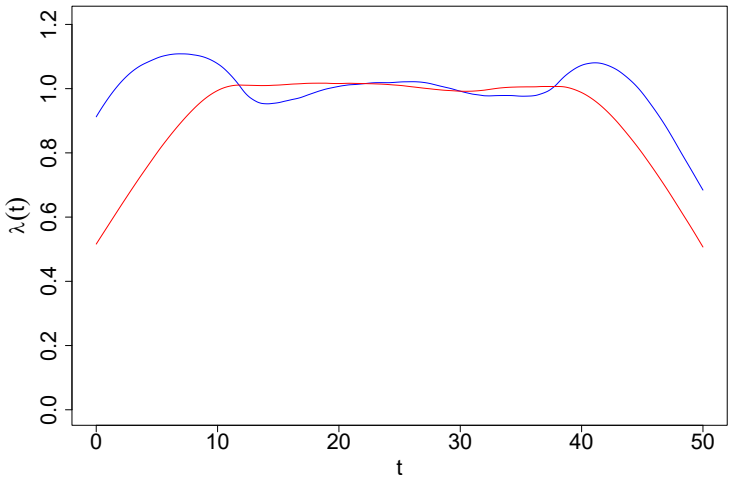

Figure 4.13: Average estimated $\lambda(t)$ over the 120 simulated data sets with $\lambda = 1$, $\beta = 0.8$ and $h = 12$. Blue line is with optimal weights applied, red line is with all weights $a_i = 1$.

The histogram of the estimated values of β is shown in figure 4.14. The trend of overestimation seems to be present also with a smaller value of $\beta = 0.8$ in the simulated data sets, with a mean value of 0.907 and median 0.902.

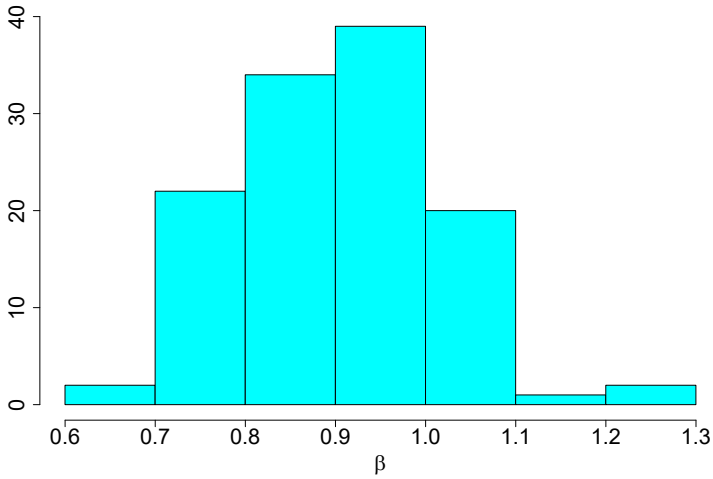

Figure 4.14: Histogram of the estimated values of β in the 120 simulated data sets with $\beta = 0.8$ and $h = 12$. Mean = 0.907, median = 0.902.

Chapter 5

Discussion

5.1 Further work

A clear thing to improve is the run time of code. The time it takes for the optimization to run increases rapidly with increasing size of data sets, as there is one parameter to be optimized for each single failure. Working on this thesis there was not spent too much time on finding the fastest way to do the optimization, though a few different approaches were tried. If quicker computation time was achieved one could do larger simulation studies more easily.

5.2 Conclusion

This thesis gives an introduction to stochastic modeling of repairable systems, with focus on the nonhomogeneous Poisson process and the trend-renewal process. It presents a kernel-based method for nonparametric estimation of the trend function of trend-renewal processes, a modified version of the method described in chapter 4 in [3]. This method is using weighted kernel estimation and is tested on several real and simulated data sets. These optimal weights are found to concur with what is said in [6], that a large proportion of the weights are zero, and the nonzero weights being clustered together. Simulation studies were done, and some characteristics were found in the methods estimates of the trend functions of the TRPs. It was also found that the method often overestimates the parameter β of the Weibull renewal distribution used in this thesis, but this overestimation got smaller the larger the data set was.

Bibliography

[1] R.D. Baker. Some new tests of the power law process. *Technometrics*, 38(3):256–265, 1996.

[2] M.J. Crowder, A.C. Kimber, R.L. Smith, and T.J. Sweeting. *Statistical Analysis of Reliability Data*. Chapman & Hall, 1994.

[3] M.L. Gámiz, K.B. Kulasekera, N. Limnios, and B.H. Lindqvist. *Applied Nonparametric Statistics in Reliability*. Springer Science & Business Media, 2011.

[4] M.L. Gámiz and B.H. Lindqvist. Nonparametric estimation in trend-renewal processes. *Reliability Engineering & System Safety*, 145:38–46, 2016.

[5] K. Heggland and B.H. Lindqvist. A non-parametric monotone maximum likelihood estimator of time trend for repairable system data. *Reliability Engineering & System Safety*, 92(5):575–584, 2007.

[6] M.C. Jones and D.A. Henderson. Maximum likelihood kernel density estimation. Technical report, Department of Statistics, The Open University, UK, 2005.

[7] B.H. Lindqvist. On the statistical modeling and analysis of repairable systems. *Statistical Science*, pages 532–551, 2006.

[8] B.H. Lindqvist. Nonparametric estimation of time trend for repairable systems data. In *Mathematical and Statistical Models and Methods in Reliability*, pages 277–288. Springer, 2010.

[9] R Core Team. R: A Language and Environment for Statistical Computing. `http://www.R-project.org/`.

[10] R Documentation. General-purpose Optimization. `https://stat.ethz.ch/R-manual/R-devel/library/stats/html/optim.html`. [Online; accessed 10-February-2017].

Appendix A

Tables

U.S.S. Halfbeak

Table A.1: Values of the weights a_i for the U.S.S. Halfbeak data set with bandwidth $h = 5$.

0.97	0	0	3.75	0	0	0	0	6.34
0	0	0	0	0	0	0	0	0
0	0	3.44	12.22	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0.99	8.53	12.59	12.74	8.03	7.33	5.93	0
0	0	0	0	0	0	0	0	

Table A.2: Values of the weights a_i for the U.S.S. Halfbeak data set with bandwidth $h = 10$.

0	0	0.34	4.83	3.62	3.41	0.02	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	2.61	2.78	3.08	4.82
7.19	11.67	11.72	11.89	12.74	13.14	13.74	13.85	

U.S.S. Grampus

Table A.3: U.S.S. Grampus failure times

860	1258	1317	1442	1897	2011	2122	2439
3203	3298	3902	3910	4000	4247	4411	4456
4517	4899	4910	5676	5755	6137	6221	6311
6613	6975	7335	8158	8498	8690	9042	9330
9394	9426	9872	10191	11511	11575	12100	12126
12368	12681	12795	13399	13668	13780	13877	14007
14028	14035	14173	14174	14449	14587	14610	15070

Table A.4: Values of the weights a_i for the U.S.S. Grampus data set with bandwidth $h = 2$.

10.46	0	0	0	0	0	0	0
6.47	0	0	0	0	0	0	0
0	3.54	1.46	8.51	0	0	0	0
0	0	0	0	0	5.74	4.31	0
0	0	0	0	0	0	0	0
0	0	11.13	0	0	0	0	0
0	0	0	0	0	0	0	17.46

Table A.5: Values of the weights a_i for the U.S.S. Grampus data set with bandwidth $h = 4$.

0	2.14	6.79	3.18	0	0	0	0
8.98	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	18.20	0	0	0	0	0	0
0	0	0	0	0	4.68	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	34.26

Table A.6: Values of the weights a_i for the U.S.S. Grampus data set with bandwidth $h = 6$.

0	0	0	6.54	0	0	0	0
0	0	0	0	0	0	0	0
11.62	10.25	8.27	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0.12	8.33	5.73
3.83	3.10	0	0	0	0	0	20.27

Photocopier

Table A.7: Photocopier data set

7	8	9	58	84	86	98	104	104	112
113	119	121	127	127	194	195	212	216	229
229	230	266	267	279	292	300	301	308	317
324	335	337	352	384	393	411	419	461	470
475	482	505	509	527	533	552	555	561	561
575	587	603	622	630	635	639	646	651	651
673	684	692	693	695	698	709	712	714	722
731	742	768	831	868	875	925	937	940	943
946	946	952	954	957	993	1013	1077	1099	1108
1125	1135								

Table A.8: Values of the weights a_i for the photocopier data set with bandwidth $h = 255$ and every second failure in a single day moved forward to the next day.

1.35	1.30	1.24	0	0	0	0.13	0.85	0.96	1.77
1.89	2.55	2.77	3.41	3.51	4.08	4.08	3.12	2.49	1.10
1.03	0.95	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	2.42	2.95	3.66	3.47	2.67	2.51	2.67	2.71
3.61	2.21	1.28	0.07	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
1.36	3.42	6.11	6.67	0	0	0	0	0	0
0	0	0	0	0	0	0	5.33	5.26	4.27
2.10	0.16								

Table A.9: Values of the weights a_i for the photocopier data set with bandwidth $h = 255$ and two failures on the same day counted as a single failure.

0	0	0	0	0	0	0	0.34	2.23	2.43
3.46	3.74	4.38	6.07	5.89	1.88	1.00	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	5.42
6.00	6.03	5.73	2.60	1.93	1.27	0	0.02	0.03	0.03
0.03	0.02	0.02	0	0	0	0	0	0	0
0	0	0	0	0	0.82	4.68	10.65	0.24	0
0	0	0	0	0	0	0	0	0	0
0	7.35	6.07	4.70	0.87	0				

Simulated data sets

Table A.10: Failure times simulated from a TRP with constant $\lambda = 1$, $\beta = 2$ and $\tau = 150$.

0.65	1.66	2.98	3.95	4.68	5.44	5.90	6.92	7.53	10.45
11.97	12.80	14.15	15.54	16.59	16.91	17.74	18.34	20.06	20.89
22.04	22.59	23.97	25.37	26.82	28.41	28.82	29.27	30.20	30.39
30.63	31.54	32.21	32.94	34.43	36.27	36.37	36.91	37.48	38.68
39.31	39.63	41.50	42.25	42.90	43.12	44.42	45.82	47.47	49.53
51.05	52.16	53.21	54.61	55.50	57.01	58.08	59.36	61.25	62.42
63.99	65.25	66.80	67.96	68.79	70.83	72.19	72.92	73.30	74.39
75.24	76.47	78.73	79.36	80.75	81.15	82.44	82.89	83.20	84.50
85.28	86.78	88.07	89.26	89.91	91.66	92.81	94.24	94.71	95.74
96.26	97.25	98.90	99.38	100.45	101.23	101.76	103.06	104.24	105.27
105.54	105.98	107.12	108.09	109.38	110.81	111.40	112.17	113.24	114.81
115.36	116.62	118.32	120.13	121.76	123.62	124.45	125.66	127.11	128.45
129.45	130.16	130.39	130.79	131.21	131.46	132.15	132.55	133.07	133.88
134.45	134.92	135.70	136.46	137.22	138.18	139.30	140.13	140.74	142.25
143.39	144.00	145.87	147.24	148.27	149.07				

Table A.11: Optimal weights for the simulated data set A.10 with bandwidth $h = 20$.

4.15	3.43	2.94	2.99	2.84	2.64	2.38	1.74	1.08	0
0	0	0.03	0.03	0.01	0	0	0	0	0
0	0	0	0	0	0	1.09	2.37	4.54	4.97
4.74	3.09	1.87	0.78	0	0	0	0	0	0
0	0.29	1.39	1.25	1.46	1.61	2.26	2.29	1.40	0
0	0	0	0	0.06	0.67	1.07	0.31	0	0
0	1.42	3.57	4.24	4.34	3.59	2.27	1.35	0.88	0
0	0	0	0	0.16	0.26	0.20	0.37	0.42	0.17
0	0.23	0.47	1.13	1.52	0.89	0	0.05	0.32	1.01
1.67	3.07	4.84	4.34	3.60	2.68	2.34	1.04	0.56	0.03
0.05	0.04	0	0	0	0	0	0	0	0
0	0	1.14	2.23	2.99	3.36	3.03	2.62	1.62	0.39
0.09	0	0	0	0	0	0	0	0	0
0	0	0.01	0.61	1.33	2.35	3.19	3.82	3.93	4.05
3.85	3.31	1.28	0	0	0				

Appendix B

R code

Listing 1: Making the $\lambda(t)$ function

```
lambda_est <- function(failures, k, h){
  # Makes the lambda function.
  #
  # Args:
  #   failures: A vector with failure times
  #   k: Kernel function
  #   h: Bandwidth
  #
  # Returns:
  #   A lambda function that takes in time t
  #   and vector a with weights

  function(t, a){
    l = 0
    for (i in 1:length(failures)){
      l = l + (k((t-failures[i])/h)/h)*a[i]
    }
    return(l)
  }
}
```

Listing 2: Making the $\Lambda(t)$ function

```
lambda_big_est <- function(failures, K, h){
  # Makes the big lambda function.
  #
  # Args:
  #   failures: A vector with failure times
  #   K: Integrated Kernel function
  #   h: Bandwidth
  #
  # Returns:
  #   A big lambda function that takes in time t
  #   and vector a with weights

    function(t, a){
    l = 0
    for (i in 1:length(failures)){
      l = l + (K((t-failures[i])/h)
            - K(-failures[i]/h))*a[i]
    }
    return(l)
  }
}
```

Listing 3: Epanechnikov kernel

```
epanech <- function(u){
  # Epanechnikov kernel
  if (abs(u) <= 1){
    return(3*(1-u^2)/4)
  }
  else{
    return(0)
  }
}
```

Listing 4: Integrated Epanechnikov kernel

```
epanech_int <- function(u){
  # Integrated Epanechnikov kernel
  if (u <= -1){
    return(0)
  }
  else if (u >=1){
    return(1)
  }
  else{
    return((3*u-u^3)/4 + 0.5)
  }
}
```

Listing 5: Log likelihood function (3.7)

```
likelihood <- function(failures, tau){
  # Makes the log likelihood function.
  #
  # Args:
  #   failures: A vector with failure times
  #   tau: Observation end point
  #
  # Returns:
  #   Log likelihood function that takes weights as input
  #
  # The parameter beta is stored as the
  # last element of the vector a
  #
    function(a){
    l = 0
    N = length(failures)

    l = l + N*(log(a[length(failures)+1])
        + a[length(failures)+1]
        *log(gamma(a[length(failures)+1]^(-1)+1)))

    failures2 = c(0, failures)
    for (i in 2:N+1){
    l = l + (a[length(failures)+1]-1)
        *log(lambda_big(failures2[i], a)
        - lambda_big(failures2[i-1], a))
        + log(lambda(failures2[i], a))
        - (gamma(a[length(failures)+1]^(-1)+1)
        *(lambda_big(failures2[i], a)
        - lambda_big(failures2[i-1], a)))
        ^a[length(failures)+1]
    }
    l = l - (gamma(a[length(failures)+1]^(-1)+1)
        *(lambda_big(tau, a) - lambda_big(failures2[
        length(failures2)], a)))^a[length(failures)+1]

    return(-l)
  }
}
```

Listing 6: Script to run the method

```
# Insert failure data, this is the start of
# the USS Halfbeak data shown

failures = c(1382,2990,4124,6827,7472,7567...)/1000
tau = 25.518

# Set initial values for the weights and beta = 1
a = rep(1, length(failures)+1)
# Choose bandwidth
h = 2
# Make lambda function
lambda = lambda_est(failures, epanech, h)
# Make big lambda function
lambda_big = lambda_big_est(failures, epanech_int, h)

# Make log likelihood function
lklh = likelihood(failures, tau)

# Optimize log likelihood function with respect
# to weights and beta
a = optim(a, lklh, method = "L-BFGS-B",
          lower = 0, upper = Inf)$par

# Plot estimate of lambda
x = 0
y = 0
x = seq(0, tau, tau/500)
for(i in 1:length(x)){
  y[i] = lambda(x[i], a)
}
plot(x, y, "l", col="blue", ylim=c(0, max(y)),
     ylab = expression(lambda (t)), xlab = "t")
```