Implementation and evaluation of two change detection-methods applied to multivariate Gaussian data-streams

Julia Petersson
Centre for Mathematical Sciences, Lund University

April 1, 2012
## Contents

1 Introduction ................................................. 2
   1.1 Aims and problem formulation .......................... 3
   1.2 Course of action .................................................. 4

2 Background .................................................. 5
   2.1 Overlapping Method ................................................ 5
      2.1.1 Main approach .................................................. 5
      2.1.2 The conjugate prior for a multivariate Gaussian distribution ........................................ 7
      2.1.3 Kullback-Leibler distance ....................................... 9
   2.2 The Kdq-Tree Method .............................................. 10
      2.2.1 Main approach .................................................. 10
      2.2.2 Chain Sampling .................................................. 11
      2.2.3 The Kdq-tree .................................................. 11
      2.2.4 The Kullback-Leibler distance .................................. 12
      2.2.5 Bootstrapping and Hypothesis testing ......................... 13

3 Experimental design ............................................ 14
   3.1 Experiments on a synthetic dataset ...................... 14
      3.1.1 Specific for the Overlapping Model ......................... 16
      3.1.2 Specific for the Kdq-tree method .......................... 16

4 Experiments and Results for a synthetic dataset .............. 17
   4.1 The Overlapping method ............................................. 18
      4.1.1 Performance for specific method-parameters .............. 18
      4.1.2 The window size $N$ vs. the decay rate $T$ ................... 21
      4.1.3 Different size of changes ...................................... 22
      4.1.4 Change of total number of added changes ..................... 25
   4.2 The Kdq-tree method .............................................. 26
      4.2.1 Performance for specific method-parameters .............. 26
      4.2.2 The window size $n$ vs. the node split constraint $\tau$ ......... 30
      4.2.3 Different size of changes ...................................... 32

5 Discussion .................................................... 36
   5.1 The Overlapping method ............................................. 36
      5.1.1 Performance for specific method-parameters .............. 36
      5.1.2 Different size of change ....................................... 38
      5.1.3 Change of total number of added changes ..................... 39
   5.2 The Kdq-tree Method .............................................. 39
      5.2.1 Performance for specific method-parameters .............. 39
      5.2.2 Different size of change ....................................... 42
      5.3 Comparison between the two methods ......................... 42

6 Conclusions ................................................... 44
   6.1 The Overlapping Method ............................................. 44
   6.2 The Kdq-tree Method .............................................. 44
7 Further work

A The Kalman filter method

A.0.1 Main approach ............................................. 46
A.0.2 Supervised learning - the Regression model ..................... 47
A.0.3 The discrete Kalman filter ..................................... 48
A.0.4 CUSUM ......................................................... 49

References ................................................................ 50
Abstract

Different types of data streams, time series, are generated in systems and collected in information bases all over the world. One may have data streams of temperatures, air pollution, share prices, traffic demand for networks or strength of the emitted radiation from a power plant. There are different statistical tools to analyse these data streams. An interesting tool to evaluate a data stream is to use a change detection method. A change detection method is used to learn if there are any changes in the stream or not. It is very useful in many areas, an example could be to analyse radiation level data collected from a power plant. The change detection method will tell if there are any changes in the radiation level and if there is an increase it could then be taken care of manually or automatically.

In this master’s thesis there are two change detection methods that are analysed and evaluated: the Overlapping method and the Kdq-tree method. The methods are of different types and have different approaches for finding a change/changes in a data stream. The Overlapping method is a parametric method and the Kdq-tree method is a non-parametric method.

The exploratory analyze in this report examines and evaluates the detection performance of the two methods in different ways: completed detection rate, detection delay, additional alarm rate with mean and standard deviation, false alarm rate with mean and standard deviation. The tests are made on a synthetic data set, with a multivariate Gaussian distribution, on both abrupt and linear changes.

From the results in this master’s thesis I draw the conclusions that the Kdq-tree method is a very useful method because one does not have to know the underlying distribution of the data stream. To be able to use the Overlapping method for change detection one have to know the underlying distribution of the data stream which is rarely known. The Overlapping method is pretty easy to understand and implement for a new user, while the Kdq-tree method requires more knowledge about the method. Both methods have parameters that have to be predetermined and which strongly affects the performance of the detection results, but the Overlapping method only has 4 parameters while the Kdq-tree method has 6 parameters. The parameters in the Kdq-tree method are also very dependent on each other which makes this method harder to work with. Both methods could be improved by studying approaches for other change detection methods.

The aim of this Master’s thesis is to obtain interesting research finding that can be used in future research.

Keywords: Change detection, sliding windows, Kullback-Leibler distance, data streams, Bayesian parameter estimation, Kdq-tree, Bootstrapping
Förord

Följande rapport är ett examensarbete som omfattar 30 hp och är avslutningen på mina studier till civilingenjör i Teknisk Fysik vid Lunds Tekniska Högskola. Examensarbetet är utfört på avdelningen Industrial Applications and Methods Lab (IAM) på Swedish Institute of Science (www.sics.se) i Kista fr.o.m. 1 April 2011 t.o.m 1 December 2011. Min handledare har varit Rebecca Steinert som är anställd vid SICS, Kista, sedan 2006 och doktorand vid KTH, Stockholm sedan början av 2010. Min examinator har varit Jimmy Olsson som är Assistant Professor sedan 2012 på Matematiska Institutionen (Matematikcentrum) vid Lunds Tekniska Högskola.

Läsarna av detta arbete förväntas ha en grundläggande kunskap inom matematisk statistik.

Ett tack riktas till min handledare Rebecca Steinert som har hjälp mig framåt under arbetets gång. Jag vill även tacka mina rumskambrater Sharenya och Alex för ert stöd och för att ni gjorde mina dagar så trevliga.

Ett stort tack till mina kära föräldrar, Ethel och Kjell, som stöttat och uppmuntrat mig under arbetets gång. Ett speciellt tack till pappa Kjell, som har hjälp mig med korrekturläsning av mitt arbete.

Slutligen vill jag tacka mina närmsta vänner som har stöttat och hjälp mig.

Stockholm 1 April 2012

Julia Petersson
Chapter 1

Introduction

Different types of data streams, time series\(^1\), are generated in systems and collected in information bases all over the world. One may have data streams of temperatures, air pollution, share prices, traffic demand for networks, strength of the emitted radiation from a power plant, quality values in a production, IP’s\(^2\), etc. Real world data like these often contain a concept drift\(^3\) or other more abrupt types of changes and therefore it is very interesting to evaluate data streams. An interesting tool to evaluate a data stream is to use a change detection method. A change detection method is used to learn if there are any changes in the stream or not. It has many interesting applications because it can give important information about the data stream and the underlying processes, that generate the data in a certain context. It can indicate changes in all kinds of contexts, like in the environment, improvement or not from a new manufacturing technique or an opportunity, or warning, for the financial analyst or problems in a power plant. An example could be to analyse radiation level data collected from a power plant. The change detection method will tell if there are any changes in the radiation level and if there is an increase it could then be taken care of manually or automatically. Research about change detection methods can be used for manual adjustment of the process or for self-regulation in complex systems. It is also of special interest in autonomous systems.

Change detection\(^1\)[2] and anomaly detection \(^3\)[4] are similar in that way that both methods tries to detect changes in a data streams, but the type of assumptions differs. In change detection the purpose is to find transitions from a stable state distribution to another stable state distribution and in anomality detection the goal could be to find anomalies from a base line distribution, see Figure 2.1 in \(^5\).

A data stream\(^6\) gives a sequence of data of unknown or unbounded size instead of a finite set of data as in a dataset (data batch), i.e. a change detection method for an offline time series, data batch, where one knows the hole dataset from the beginning are treated different from an online time series, a data stream\(^7\). A data stream poses special challenges for the change detection method. A change detection method that is used over a data stream has therefore special constraints on storage and duration of access to the data, which needs to be taken into account when constructing a change detection method\(^8\).

To detect changes in a data stream one can use many different approaches, which can be divided into parametric methods and non-parametric methods. A non-parametric method requires no prior knowledge about the underlying statistical properties of the data \(^9\)[10], whereas a parametric approach are based on parameter estimations from data according to a certain probability distribution\(^11\)[12].

A change detection method consists of different parts to achieve the goal of autonomously finding changes in a data stream from some kind of system in a special context. To find a change it is common to compare two datasets to each other. To store data sets for comparison one can use a sliding window approach, i.e. a window that contains a small part of the entire data set \(^13\)[14][15]. Some change detection approaches also take into account some kind of decay function \(^16\)[11]. A decay function will add memory of history to the model. The model will not only care about fresh data but it will also

\(^1\)time series - a sequence of data points obtained through successively repeated measurements over time, typically at uniform time intervals

\(^2\)Internet Protocol - the principal communication protocol that is used for transformation of information in the computer network Internet

\(^3\)concept drift - the statistical properties of the target variable changes over time in unforeseen ways, like seasonality
take previous data into account, i.e. it will gradually forget older data and this will bring robustness in adaption and detection performance.

To do the comparison of the data sets (windows) it is common to do some kind of modelling of the distributions in the windows. The modelling can be done with statistical analysis like Bayesian estimation \[17\] \[18\] \[1\], sketching \[19\], aggregation (also referred to as buckets) \[13\] \[20\], build a KS-structure \[10\] or a space partitioning scheme \[21\]. One can also build a prediction model, like a regression model, time series forecasting model \[22\] or a Kalman filter \[23\] \[24\] \[25\]. The comparison between the windows is generally based on similarity measure \[4\] \[26\]. Common similarity measurements are based on measuring the likelihood-ratio between models, or by comparing average measurements over time \[13\], measuring the Kullback-Leibler distance, \[27\], or Kolmogorov-Smirnov statistics \[10\]. The Kolmogorov-Smirnov test is hard to use on multidimensional data because it is a rank-based statistic test, i.e. the test relies commonly on data being ordered. Another common similarity measure is CUSUM \[24\] \[25\]. The similarity measure is compared with a predetermined threshold and a change is detected when the similarity measure reaches above the threshold. It will indicate that there is a significant difference between the distributions in the compared windows. The detection threshold can be set manually or in another way, as by using non-parametric Bootstrap and hypothesis testing \[28\].

The first method evaluated in this report is called the Overlapping Method \[1\] and is designed at the Industrial Applications and Methods Lab at Swedish Institute of Computer Science. This method is parametric and based on overlapping models (windows), where the data distribution is modeled based on statistical analysis, with Bayesian parameter estimation, and changes are detected by the specified symmetric Kullback-Leibler distance.

The second change detection method has been selected for the study, among other methods, based on special criterias, see chapter 1.1. It has been named the Kdq-tree method \[2\] \[21\]. This method is non-parametric and uses sliding windows to store data where the data distribution is modeled based on a space partitioning scheme, a Kdq-tree, and changes are detected by the general Kullback-Leibler distance.

This report evaluates the detection performance for the Overlapping method and the Kdq-tree method on synthetic data-streams generated by multivariate Gaussian processes.

### 1.1 Aims and problem formulation

There are a lot of change detection methods with different approaches. This work have been done at the Industrial Applications and Methods Lab at Swedish Institute of Computer Science (www.sics.se) in Kista. The main goal with this work is to do an exploratory analyze and evaluate SICS’s statistical parametric change detection method, the Overlapping Method, and another statistical change detection method that is considered interesting and with a different approach, preferably non-parametric.

The methods that were proposed to be studied are described in the following papers:

Paper 1: The Overlapping method \[1\].

Paper 2: The Sketch structure method \[29\].

Paper 3: The Kdq-tree method \[2\].

Paper 4: The KS-structure method \[10\].

Paper 5: The Adaptive-windowing method \[13\].

Paper 6: The Sketched based change detection method \[22\].

Paper 7: The Kalman method with regression \[30\].

Paper 8: The Decentralized CUSUM change detection \[31\].

Paper 9: The dualCUSUM method \[32\].

\(^{4}\)Similarity measures- a measure of distance between two distributions
Paper 10: The Adapted Gaussian mixture models method [18].

The selection of the method that would be treated in this report in addition to the Overlapping method, was based on following criterias. The method should be:

• of different approach from the Overlapping method
• of interest for the student and supervisor
• implementable, i.e. not too complicated
• somewhat time-efficient

The exploratory analyze examines and evaluates the detection performance of the methods in different ways: completed detection rate, detection delay, additional alarm rate with mean and standard deviation, false alarm rate with mean and standard deviation. The tests are made on a synthetic data set with both abrupt and linear changes. The two methods are implemented, evaluated, discussed and improvements are suggested.

The aim of this Master’s thesis is to obtain interesting research finding that can be used in future research.

1.2 Course of action

This Master’s thesis is built in four steps.

1. A study of existing methods: A short preliminary study of existing methods and approaches for change detection in data streams, see chapter [11]. An appropriate method that should be included in the exploratory study is chosen in discussions with the supervisor. Also an appropriate synthetic data distribution for the study is considered and chosen.

2. Implementation: The methods are implemented in MATLAB®[^matlab], real data is pretreated and methods for generating the synthetic data are implemented in MATLAB®. The computer that is used is a Lenovo ThinkPad X201.

3. Experimental tests: The experimental tests are designed and performed with guidance of the supervisor. I have an extra server to make the tests more time efficient and it is a Red Hat Enterprise Linux Server 5.7.

4. Finalization of the report: The results are evaluated, discussed and conclusions are made.

[^matlab]: matlab - is a computer program and programming language commonly used in mathematical and technical computing, developed by the company The MathWorks
Chapter 2

Background

In this chapter, the two methods that have been analyzed and evaluated in this work are described. The main goal with these methods is to find changes in the multivariate Gaussian distribution of a data set. It can be changes in only the mean vector, \( m \), or the covariance matrix, \( \Sigma \), or in both parameters for either linear or abrupt changes. See Figure 2.1 and 2.2 for a demonstration of abrupt changes in the mean, \( \mu \), and the variance, \( \sigma \), for a unidimensional Gaussian distribution.

![Figure 2.1: An abrupt change in the mean \( \mu \) of a dataset](image1)

![Figure 2.2: An abrupt change in the variance \( \sigma \) of a dataset](image2)

The two methods are tested on multivariate Gamma distributions and are therefore adapted to detect changes on a distribution like this.

2.1 Overlapping Method

2.1.1 Main approach

This method is parametric and uses overlapping models (windows) to compare data sets to each other by making a Bayesian estimation of the parameter \( \Theta = [m, Q] \) for each model\[8\]. The name of the method, Overlapping Method, indicates that the models evolve more or less in parallel as new data points arrive. Each model has room for \( N \) data items and has a decay rate \( T \). Both \( N \) and \( T \) are held constant for each model. The decay rate is a percentage of \( N \) and it determines the number of data items that are summarized and sent to the next model as a priori input. The summarization that forms the a priori is the mean, \( \mu^* \), and covariance matrix, \( \Sigma^* \), see (2.7) and (2.8) in chapter 2.1.2. The decay rate \( T \) can also be seen as a parameter which decides how much the models will overlap each other. By using a priori input the old data is always taken into account when the parameters for a model is estimated and will gradually be forgotten over time. There are always only two models that are compared to each other at each step, the current model and the previous model. The current
model is the model that is currently updated with new items and consists of \(n\) data items. The previous model has is the model that had already been filled with data items, i.e. it consists of \(N\) data items.

In Figure 2.3 the overlapping parts of the models are shown and the dashed line is the part of the previous model that is as \textit{a priori} input to the model. The first model, with estimated parameter \(\Theta_{i-2}\), is the first model made and therefore it does not have \textit{a priori} input (no dashed line).

![Figure 2.3: Overlapping models with model size \(N\), decay rate \(T\) and respectively estimated parameter \(\Theta\)]

The main approach of this method is to estimate the parameter set \(\Theta\) for the overlapping models with Bayesian parameter estimation and use the Kullback-Leibler distance to measure if a change has occurred, see chapter 2.1.3.

The Bayesian approach to parameter estimation aims to maximize the probability of the parameters given the data. The Bayesian parameter estimation of \(\Theta\) is:

\[
P(\Theta | X) = \frac{P(X | \Theta)P(\Theta)}{P(X)}
\]  

(2.1)

The factor \(P(X)\) is removed to make it solvable. The factor \(P(X)\) is considered to be a constant and it do not include the parameter \(\Theta\). The theorem is rewritten and becomes unnormalized:

\[
P(\Theta | X) \propto P(X | \Theta)P(\Theta)
\]  

(2.2)

In (2.2) the relation between the Bayesian approach, \(P(\Theta | X)\), and the classical way of parameter estimation, the maximum likelihood approach, \(P(X | \Theta)\), shows. The distribution \(P(\Theta)\) is usually referred to as the prior probability distribution, i.e. what is assumed before the data \(X\) is observed. The distribution \(P(\Theta | X)\) is referred to as the posterior distribution, i.e. what is assumed after the data \(X\) is observed.

The likelihood distribution, \(P(X | \Theta)\), is obtained by knowing the distribution of the data set and the data points \(X = [x_1, x_2, ..., x_n]\). The formula for a joint distribution is used:

\[
P(x_1, x_2, ..., x_n | \Theta) = \prod_{i} P(x_i | \Theta)
\]  

(2.3)

The prior distribution, \(P(\Theta)\), is chosen to be the \textit{conjugate prior} for the likelihood, see chapter 2.1.2. Some distributions has a conjugate prior and by choosing the conjugate prior for the likelihood as the prior distribution it ensures that the posterior distribution gets the same distribution as the prior distribution, i.e. a known distribution.

By using formula 2.2 with the conjugate prior, the posterior \(P(\Theta | X)\) will have the same distribution as the conjugate prior but with different hyper parameters \(\Theta^*\).

---

\(1\) Conjugate priors - analogous to eigenfunctions in operator theory, in that they are distributions on which the 'conditioning operator' (likelihood distribution) acts. However, the processes are only analogous not identical, because of the configuration of the space of distributions, i.e. the posterior is only of the same form as the prior distribution, not a scalar multiple.

\(2\) Hyper parameters - In Bayesian statistics, a hyper parameter is a parameter of a prior distribution. They arise particularly in the use of conjugate priors and is used to distinguish them from parameters of the model for the underlying system under analysis.
The expected value of the estimated parameter, \( E(\Theta^*) \), will be:

\[
E[m] = \mu \\
E[Q] = \Sigma^{-1}
\]  

(2.4)

where \( \mu \) and \( \Sigma \) are the hyperparameters for the posterior. This holds for a Wishart-Gaussian distribution[17][35].

In order to detect a change in the distribution, the current estimated model parameter, \( \Theta^*_i \), is frequently compared to the previous estimated model parameter, \( \Theta^*_{i-1} \), using the symmetric Kullback-Leibler divergence, see chapter 2.1.3. The expectation of the parameters for the previous model and the current model are compared for each new item in the current model. A change is detected if the Kullback-Leibler divergence is above some specified threshold, \( \eta_{div} \). To reduce the number of additional alarms a convergence threshold, \( \eta_{conv} \), can also be used in the method.

This method can be used for multidimensional data streams as well as unidimensional data streams.

Specific parameters for Overlapping Method can be seen in Table 2.1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detection threshold</td>
<td>( \eta_{div} )</td>
</tr>
<tr>
<td>Convergence threshold</td>
<td>( \eta_{conv} )</td>
</tr>
<tr>
<td>Model size</td>
<td>( N )</td>
</tr>
<tr>
<td>Decay Rate</td>
<td>( T )</td>
</tr>
</tbody>
</table>

Table 2.1: Parameters OM

2.1.2 The conjugate prior for a multivariate Gaussian distribution

The formulas in the derivation that follows are taken from [17]. The multivariate Gaussian distribution is a generalization of the one-dimensional Gaussian distribution to higher dimensions. The notation for a multivariate Gaussian distribution of an \( n \)-dimensional vector \( X \) is:

\[
X \sim N_n(m, Q)
\]

(2.5)

where the mean \( \in \mathbb{R}^n \) and the covariance \( \Sigma \in S^n_{++} \).

The probability density function is given by (2.6)[37] and the probability density function of a two dimensional distribution can be seen in Figure 2.4

\[
P(x|m, Q) = \frac{|Q|^{1/2}}{(2\pi)^{d/2}} exp\left(-\frac{1}{2}(x - m)^T Q(x - m)\right)
\]

(2.6)

The maximum likelihood estimate of the mean vector and the covariance matrix of a multivariate Gaussian distribution are:

\[
\mu^0 = \frac{1}{c} \sum_{j=1}^{c} x_j
\]

(2.7)

\[
\Sigma^0 = \frac{1}{c} \sum_{j=1}^{c} (x_j - \mu^0)(x_j - \mu^0)^T
\]

(2.8)

\(^3S^n_{++} \) is the space of a symmetric positive definite \( n \times n \) covariance matrix, defined as:

\( S^n_{++} = A \in \mathbb{R}^{n \times n} : A = A^T \) and \( x^T A x > 0 \) for all \( x \in \mathbb{R}^n \) such that \( x \neq 0 \).
Figure 2.4: The probability density function for a 2-dimensional Gaussian distribution [38]

where \( x_j \) is a column vector of size \( n \) containing samples from a data set of size \( c=n \). The formulas are also used over the data items that \( T \) determines in "the previous model", to calculate the \( a \ priori \), \( \mu^* \) and \( \Sigma^* \), with \( c=NT \) [35].

The conjugate prior distribution for a multivariate Gaussian distribution with unknown mean, \( \mu \), and covariance matrix, \( Q \), is the Wishart-Gaussian distribution with hyper parameters \( \mu^* \), \( \Sigma^* \) and \( \alpha \). The notation for the multidimensional parameter \( \Theta = [m,Q] \) that belongs to the Wishart-Gaussian distribution is:

\[
\Theta \sim W_n(\mu^*, \Sigma^*, \alpha^*)
\]

and the probability density function is given by (2.10):

\[
P(m,Q|\mu^*,\Sigma^*,\alpha) \propto |Q|^{(a-d-1)/2} \exp\left(-\frac{1}{2} tr(\alpha Q \Sigma^*)\right) \exp\left(-\frac{1}{2} tr(\alpha Q (\mu^* - m)(\mu^* - m)^T)\right)
\]

where the expected value of the mean, \( E[m] \), and the covariance matrix, \( E[Q] \), are:

\[
E[m] = \mu^*
\]
\[
E[Q] = \Sigma^*^{-1}
\]

The likelihood of the multivariate Gaussian distribution is calculated according to (2.3):

\[
P(X|m,Q) = \sum_{j=1}^{c} P(x_j|m,Q) = \sum_{j=1}^{c} \frac{|Q|^\frac{1}{2}}{(2\pi)^{\frac{d}{2}}} \exp\left(-\frac{1}{2} (x_j - m)^T Q (x_j - m)\right)
\]

\[
= \frac{|Q|^\frac{1}{2}}{(2\pi)^{\frac{d}{2}}} \exp\left(-\frac{1}{2} tr(Q \sum_{j=1}^{c} (x_j - m)(x_j - m)^T)\right)
\]

\[
= \frac{|Q|^\frac{1}{2}}{(2\pi)^{\frac{d}{2}}} \exp\left(-\frac{1}{2} tr(cQ \Sigma^0 + cQ(\mu^0 - m)(\mu^0 - m)^T)\right)
\]

By multiplying the conjugate prior distribution with the likelihood for the multivariate Gaussian, as in (2.2) we get:

\[
P(m,Q|X,\mu^*,\Sigma^*,\alpha) \propto P(X|m,Q)P(m,Q|\mu^*,\Sigma^*,\alpha)
\]

By inserting (2.10) and (2.12) into (2.13) it can be seen that the Bayesian posterior distribution over the mean and covariance also becomes a Wishart-Gaussian:
\[
P(m, Q | X, \mu^*, \Sigma^*, \alpha) =
\]

\[
|Q|^{n+\alpha+\frac{d-1}{2}} \exp \left( \frac{1}{2} tr(c+\alpha)Q \left( \frac{\alpha \Sigma^* + c\mu^0 + \alpha \mu^*}{c + \alpha} \right) \right) \exp \left( \frac{1}{2} tr(c+\alpha)Q \left( \frac{\alpha \Sigma^* + c\mu^0 + \alpha \mu^*}{c + \alpha} - m \right) \right)
\]

where the hyper parameters are \( \mu \) and \( \Sigma \):

\[
\mu = \frac{n\mu^0 + \alpha \mu^*}{n + \alpha} \tag{2.15}
\]

\[
\Sigma = \frac{n\Sigma^0 + \alpha \Sigma^* + \frac{n\alpha}{n+\alpha} (\mu^0 - \alpha \mu^*) (\mu^0 - \alpha \mu^*)^T}{n + \alpha} \tag{2.16}
\]

and:

\[
E[m] = \mu \tag{2.17}
\]

\[
E[Q] = \Sigma^{-1} \tag{2.17}
\]

The variables \( n \) and \( \alpha \) in (2.16) can be considered as weights for the maximum likelihood estimates of \( \mu^0 \) and \( \Sigma^0 \) respectively the prior inputs \( \mu^* \) and \( \Sigma^* \). The weight \( n \) is the number of items in one model and the a priori weight \( \alpha \) is set to 1. The expected value of the Bayesian estimates of the parameters \( m \) and \( Q \) are a weighted combination of the maximum likelihood estimates \( \mu^0 \) and \( \Sigma^0 \) and the a priori \( \mu^* \) and \( \Sigma^* \).

### 2.1.3 Kullback-Leibler distance

The Kullback-Leibler distance is also called the relative entropy\[39\] and measures the distance between two distributions. For two discrete distributions there are two probability functions \( B \) and \( P \) and the Kullback-Leibler distance of \( P \) with respect to \( B \) is\[27\]:

\[
D_{KL}(B||P) = \sum_n B_n \log \frac{B_n}{P_n} \tag{2.18}
\]

This measure is not symmetric and therefore not a true metric. The KL-distance from \( B \) to \( P \) is not necessarily the same as the KL-distance from \( P \) to \( B \).

\[
D_{KL}(B||P) \neq D_{KL}(P||B) \tag{2.19}
\]

The formula above is used when \( P \) is assumed to be the true distribution and \( Q \) typically represents a theory, model or approximation of \( P \). The symmetric Kullback-Leibler distance is\[35\]:

\[
D(B||P) = D_{KL}(B||P) + D_{KL}(P||B) \tag{2.20}
\]

The Kullback-Leibler divergence is specific defined for most distributions. In case of data of a multivariate Gaussian distribution the asymmetric measure is:

\[
D_{KL}(N(\mu_1, \Sigma_1)||N(\mu_2, \Sigma_2) = \frac{1}{2} \left( \log \frac{\text{det} \Sigma_2}{\text{det} \Sigma_1} + \text{tr} (\Sigma_2^{-1} \Sigma_1) + (\mu_2 - \mu_1)^T \Sigma_2^{-1} (\mu_2 - \mu_1) \right) \tag{2.21}
\]

where \( d \) is the number of dimensions in the multivariate Gaussian distribution\[35\].
2.2 The Kdq-Tree Method

2.2.1 Main approach

This method is described in [2] and the approach is summarized in this section. The method constructs distributions from two windows \( W_t \) and \( W_{t'} \) using a space partitioning tree\(^4\) a *kdq-tree*\(^2\), see chapter 2.2.3. A sliding window, \( W_{i,n} \) denotes a sequence of points from the data stream ending at \( x_i \) of size \( n \), \( W_{i,n} = (x_{i-n+1}, \ldots, x_i) \), where each \( x_i \in \mathbb{R}^n \). If the subscript \( n \) is dropped both windows in this model, \( W_t \) and \( W_{t'} \), are of size \( n \) and \( t = i \) and \( t' = i - n \), the so called *adjacent window model*.

A sliding window can be seen in Figure 2.5.

![Sliding window](image)

Figure 2.5: A sliding window of size \( n=4 \). [40]

To make the method more time efficient it uses *chain sampling* to obtain a sample \( S \) of size \( k \) from the window, \( W \), where \( k < n \), see chapter 2.2.2. When sample \( S \) is filled with data from the window the next samples are taken from the constantly updated *chain of samples*. Chain sampling is necessary when one has a data stream and it is done over both window \( W_t \) and \( W_{t'} \). Thereafter the Kdq-tree maps the sample of points \( S_t \) and \( S_{t'} \) from window \( W_t \) and \( W_{t'} \) into distributions, \( F_t \) and \( F_{t'} \). The distance between these two distributions is measured with the Kullback-Leibler distance, \( d_t = d(F_t, F_{t'}) \) see chapter 2.2.5. In the Kullback-Leibler distance the number of data in each leave of the two Kdq-trees is compared to each other for all leaves, see formula 2.25 and 2.26. Also the total amount of data in each sample that the Kdq-trees are made over, \( |W_t| \) and \( |W_{t'}| \) and the number of leaves for the Kdq-tree, \( L \), are included in the formula. Bootstrapping is done in advance and is used to get a significant threshold, \( d \), that the Kullback-Leibler distance can be compared to, see chapter 2.2.5.

Specific parameters for the Kdq-tree method are seen in Table 2.2.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Window size</td>
<td>( n )</td>
</tr>
<tr>
<td>Max. points in a cell</td>
<td>( \tau )</td>
</tr>
<tr>
<td>Min. cell side</td>
<td>( \delta )</td>
</tr>
<tr>
<td>Chain sample size</td>
<td>( k )</td>
</tr>
<tr>
<td>Significance level</td>
<td>( \alpha )</td>
</tr>
<tr>
<td>Bootstrap samples</td>
<td>( B )</td>
</tr>
</tbody>
</table>

\(^4\)Space partitioning tree - divides spaces recursively into subspaces. The division will have the structure of a tree.
2.2.2 Chain Sampling

Instead of working with the whole dataset of size \( n \) in a window, \( W \), a reservoir sample algorithm \[41\] can maintain a uniform sample of size \( k \) from the window. This is done to save memory and time. The sliding window \( W \) contains the most recent \( n \) elements from a data stream. When a new data item arrives an old one expires and is not included in the current window anymore. In streaming applications, when the data expires like this after a certain time, it is important to extend the reservoir sampling algorithm. *Chain sampling* is used to maintain a sample of size \( k \) over a sliding window\[42\].

In the chain sampling algorithm a vector \( S \) stores the sample of size \( k \). The probability that a data item, \( x_i \), at time \( i \) is chosen to be in the sample is:

\[
P(x_i \in S) = \frac{1}{\min(i, n)}
\]  

(2.22)

The first \( k \) items in vector \( S \) is chosen in this way. For a chosen item \( x_i \) a new index of an item, \( j \), is randomly chosen. The index \( j \) is between \((i+1)\) and \((i+n)\) and when this item arrives it is stored and the algorithm selects a new index of an item that will replace \( x_j \) when it expires, building a 'chain' of potential replacements. A data item in sample \( S \) expires when the index of the item is too small to be included in window, \( W \), i.e. it expires from window \( W \). For every item in sample \( S \) there is a chain of potential replacements and when an item expires in sample \( S \) a new data from its respective chain of potential replacements is added to \( S \)[13].

2.2.3 The Kdq-tree

The Kdq-tree is a combination of two space partitioning schemes\[5\], a Kd-tree\[6\] and a Quad-tree\[7\]. i.e. it is a data structure that emulates a hierarchial tree structure with a set of linked nodes\[44\]. The Kdq-tree is a binary tree where all the internal nodes have a parent and two children each, see Figure 2.6.

![Figure 2.6: A binary tree][8]

Each node is associated with a box and the box for the first node, i.e. node 1 which is the node without a parent, includes all the data samples from the start and is associated with the root \( v \). The box is then divided into two halves, node 2 and 3 which are associated with the two children of the root \( v, v_l \) and \( v_r \) respectively. The coordinates of the split for node 1 is root \( v \) and these coordinates are stored in vector \( V \), just as the "splitting-coordinates" \( v_l \) and \( v_r \) for node 2 and 3. Next step is to construct the trees rooted at \( v_l \) and \( v_r \), recursively, as we go down the tree.

In the 2-dimensional case that is shown in \[2.7\] the lines which divides the parent-nodes into two new children-nodes are shown and the structure of its Kdq-tree. As can be seen, the split is done by drawing a vertical cut passing the box’s center. For each level\[8\] of the tree the cuts alternate between

---

5Space partitioning scheme - in mathematics the process of dividing a region of space into two or more disjoint subsets
6Kd-tree - a binary tree where each node is associated with a k-dimensional point and each internal node has exactly two children
7Quad-tree - a tree data structure in which each internal node has exactly four children. Each node has a maximum capacity and when it is reached the node splits
8the set of all nodes at a given depth is sometimes called a level of the tree. The depth of a node \( n \) is the length of the path from the root to the node. The root node is at depth/level zero
the two dimensions \( x \) and \( y \). In the n-dimensional case the cuts alternate between the \( n \) dimensions.

![Diagram of a Kdq-tree and its subdivisions](image)

**Figure 2.7:** A Kdq-tree and its subdivisions

Each Kdq-tree has the specific parameters minimum cell side, \( \tau \), and maximum points in a node box, \( \tau \). The minimum cell side is the distance between the biggest and smallest value in a node box for each dimension. The split of the cells in the Kdq-tree continues until \( \tau \) or \( \tau \) has reached a specific predetermined value for a node. The last nodes in the kdq-tree, with no children, is called leaf-nodes. In picture ? node number 5, 6, 7, 8 and 9 are leaf-nodes.

When the Kdq-tree is completed it has organized the data into a multidimensional-histogram\(^9\). A 2-dimensional histogram can be seen in Figure 2.8 and 2.9.

![2-Dimensional histogram of two normally distributed variables](image)

**Figure 2.8:** A 2-dimensional histogram of two normally distributed variables

The first Kdq-tree is built over the samples \( S_t \) from window \( W_t \), and for this window the split-coordinates, root \( v \), for each node is stored in vector \( V \). The second Kdq-tree is built over samples from window \( W_t, S_t \), but according to the coordinates in vector \( V \). In this way the two Kdq-trees will have the same structure and the distributions can be compared to each other\(^2\).

### 2.2.4 The Kullback-Leibler distance

What follows is a summarization of what is mentioned in \([50] \) and \([51] \). The set \( w = a_1, a_2, \ldots, a_n \) is a multiset of letters from a finite alphabet \( A \). The Kdq-tree is used to store data and to form a distribution. For a \( d \)-dimensional data set, \( A \) will consist of a letter for each leaf of the kdq-tree. The type \( P_w \) of \( w \) is thus a vector representing the relative proposition of each element of \( A \) in \( w \), where:

\[
P_w(a) = \frac{N(a|w)}{k}
\]  

\(^9\)Histogram - a type of diagram that uses bars to represent the frequency (proportion or percentage) of cases associated with each outcome or interval of outcomes of a variable
and \( N(a|w) \) is the number of items in cell \( a \) and \( k \) is the total number of items stored in the tree. Thus \( P_w \) forms an empirical distribution that the sequence of points maps to. Given two windows \( W_1 \) and \( W_2 \) the Kullback-Leibler distance between them is:

\[
D(W_1||W_2) = \sum_{a \in A} P_{w_1}(a) \log \frac{P_{w_1}(a)}{P_{w_2}(a)}
\]  

(2.24)

\( P_v \) (resp. \( Q_v \)) is the number of points from the set \( W_1 \) (resp. \( W_2 \)) that are inside the node associated with the leaf \( v \) of the kdq-tree. The KL-distance between \( P_v \) and \( Q_v \) is:

\[
D(P_v||Q_v) = \sum_v \frac{P_v + L}{|W_1| + L/2} \log \frac{(P_v + L/2)/(|W_1| + L/2)}{(Q_v + L/2)/(|W_2| + L/2)}
\]  

\[= \log \frac{|W_2| + L/2}{|W_1| + L/2} + \sum_v \frac{(P_v + L)}{|W_1| + L/2} \log \frac{P_v + L/2}{Q_v + L/2}
\]  

(2.25)

In 2.25 we already know \(|W_1|\) and \(|W_2|\) which both are the sample size \( k \), and \( L \) which is the number of leaves in the Kdq-tree. We only need to maintain:

\[
D(P||Q) = \sum_v (P_v + L/2) \log \frac{P_v + L/2}{Q_v + L/2}
\]  

(2.26)

### 2.2.5 Bootstrapping and Hypothesis testing

To get a statistically significant measure of the threshold, \( d \), the theory of Bootstrapping is used. The null-hypothesis below is asserted:

\[ H_0 : F = G \]

A window of the first \( n \) data-items of the dataset is used and \( m \) sets, \( S^{*}_1, S^{*}_2, ..., S^{*}_m \), each of size \( 2n \), are sampled with replacement. The probability for a data-item to be in the sample, \( S^{*} \), is:

\[
P(x_i \in S^*) = \frac{1}{2n}
\]  

(2.27)

The \( m \) samples \( S^{*}_i \) are split into two samples, each of size \( n \). By treating the first \( n \) samples \( S^{*}_1 \) as coming from a distribution \( F \) and the remaining \( n \) samples \( S^{*}_2 \) as coming from another distribution \( G \), the Bootstrap estimate of \( d \) is calculated with \( \hat{d} = D(S^{*}_1||S^{*}_2) \). A critical region is computed from the Bootstrap estimates \([d_{H_0}(\alpha), \infty]\) and is the \((1 - \alpha)-\)percentile of the Bootstrap estimates.

If \( d_t \) falls into the critical region \( H_0 \) is considered not to be true, and a statistically significant change is detected.
Chapter 3

Experimental design

The detection performance of the two methods, the Overlapping Models and the Kdq-tree Method, are tested in the same way, on a synthetic datasets with a 3 dimensional Gaussian distribution.

3.1 Experiments on a synthetic dataset

For all the tests below a data stream of \( n = 10^5 \) data points was synthetically generated from multivariate Gaussian distributions. To construct a stream with changes, the distribution is split into \( P \) partitions with different parameter values, the mean vector \( \mu \) and covariance matrix \( \Sigma \). The variable \( P \) decides the amount of added changes (added changes= \( P-1 \)), i.e. for one abrupt change \( P=2 \) and for one linear change \( P=3 \).

The tests of the performance are the three following:

**Performance A. Detection delay:** Measures the detection delay that is the amount of items between the added change and the detected change.

**Performance B. Detected false alarms:** Measures the rate of false alarms that are detected. A false alarm is when the method detects a change even if there is no.

**Performance C. Detected additional alarms:** Measures the rate of detected additional alarms. An additional alarm occurs when the models are adapting from a change to a new stable state.

The three performance tests above have each been analyzed relative to the three different aspects below:

1. **Specific method parameters:** Two important parameters from the method are chosen and varied to see how the performance depends on these parameters. This experiment is done to analyze how sensitive the method is regarding to how the user selects the parameter values. The test is also used to know how these parameters should be set in the following tests. This test is done for a dataset with one abrupt change and a data set with one linear change in two separate tests:

   - **Abrupt change in \( \mu \) and \( \Sigma \):** The data set is made of two partitions where the multivariate parameters \( \mu \) and \( \Sigma \) are randomly drawn from an univariate Gaussian distribution with parameters \( \mu = 0 \) and \( \sigma^2 = 10 \) for each partition. The change occurs at sample 50000.
   - **Linear change in \( \mu \):** The data set is made of three data parts where the multivariate parameter \( \mu \) changes for the three parts and the covariance matrix \( \Sigma \) is held constant. In the first data part \( \mu \) and \( \Sigma \) are randomly drawn from an univariate Gaussian distribution with parameters \( \mu = 0 \) and \( \sigma^2 = 10 \). Data part one is the 25000 first samples. In data part two, which lasts for 50000 samples, there is a linear change in \( \mu \):
\[ \mu_{i+1} = \mu_i + k \quad \text{for } i = 1 \text{ to } i = 50000 \]  
\[ (3.1) \]

where \( \mu_1 \) is the same as in data part one and \( k \) is drawn from an univariate Gaussian distribution with parameters \( \mu=0 \) and \( \sigma^2=0.001 \). In data part three the distribution returns to a stable state distribution with \( \mu=\mu_{50000} \). To make it clear, there is a change when the linear change starts, at sample 25000, and when the linear change ends, at sample 75000.

- **The specific method parameters dependence for an abrupt change:** In this test the impact on some important performance measures of the specific method parameters dependence are tested. Two or three of the performance measures that are considered important are tested.

2. **Different sizes of changes:** In this test the size of the change is varied to see how the performance depend on the size of change. In test number 1 the change can vary a lot but in this test the variations are not random but controlled. The test is done for one abrupt change and one linear change with a value of the specific method parameters determined from test 1. For an abrupt change the test is done in two ways: one for a data set where the mean \( \mu \) is varied and the covariance \( \Sigma \) is held constant and one where the covariance matrix \( \Sigma \) is varied and the mean \( \mu \) is held constant. There is also one test for a data set with a linear change in \( \mu \).

- **Abrupt changes in \( \mu \):** There is one abrupt change just as in test 1 but instead of choosing a random mean \( \mu \) in partition two the parameter is set to:

\[ \mu_{\text{part}2} = \mu_{\text{part}1} + k\mu \]  
\[ (3.2) \]

where \( k\mu \) is set to \( \pm 0.6, \pm 0.8, \pm 1, \pm 1.3, \pm 1.5 \pm 3 \) and \( \pm 10 \). The covariance matrix \( \Sigma \) is held constant.

- **Abrupt changes in \( \Sigma \):** There is one abrupt change just as in test 1 but instead of choosing a random covariance matrix \( \Sigma \) in partition two the parameter is set to:

\[ \Sigma_{\text{part}2} = \Sigma_{\text{part}1} + k\Sigma \]  
\[ (3.3) \]

where \( k\sigma \) is set to \( \pm 0.6, \pm 0.8, \pm 1, \pm 1.2, \pm 1.5, \pm 2, \pm 2.5, \pm 5 \) and \( \pm 10 \). The mean vector \( \mu \) is held constant.

- **Linear changes in \( \mu \):** In this test the size of a linear change is varied, i.e. \( k \), and \[ (3.1) \] is used to make the linear change in the data set, which starts at sample 25000 and ends at sample 75000. The parameter \( k \) is not randomly chosen from a multivariate Gaussian distribution as in test 1 but instead the \( k \)-values is set to \( \pm 0.0001, \pm 0.0005, \pm 0.001, \pm 0.01 \) and \( \pm 0.1 \).

3. **The total number of added changes (extra test for the Overlapping method):**

- **The Overlapping method:** In this test the total number of added changes is varied for the Overlapping method, i.e. \( P \), in order to see how the performance depends on the number of changes. To study the number of total changes, the type of change and the value of the size of change are determined from test 2. The value of the size of change alternates randomly between an increase and a decrease. The number of partitions \( P \) that is tested is: 10000 (9 changes), 2040 (49 changes), 502 (199 changes), 401 (249 changes), 380 (263 changes), 380 (294 changes), 340 (294 changes), 300 (333 changes) and 280 (357 changes).

All together there are eleven different types of test for each of the two methods and all of them are made a repeated 20 times, to get significance in the experimental results.
3.1.1 Specific for the Overlapping Model

In test number 1 the number of items in a model, \(N\), and the decay rate, \(T\), are analysed. The test is done for: \(T= 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9\) and for \(N= 100, 200, 300, 500, 700, 1300, 2000\) and 5000 separately.

Unless otherwise is stated the parameters in Table 3.1 hold.

**Table 3.1:** Predetermined parameter values for the Overlapping method

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Detection threshold</td>
<td>(\eta_{div})</td>
<td>0.6</td>
</tr>
<tr>
<td>Convergence threshold</td>
<td>(\eta_{conv})</td>
<td>0.1</td>
</tr>
</tbody>
</table>

In test 1, ’The specific method parameters dependence for an abrupt change’ all window sizes \(N\) are tested for all decay rates \(T\).

3.1.2 Specific for the Kdq-tree method

In experiment 1 the sliding window size, \(n\), and maximum points in a cell, \(\tau\), are analysed. The test is done for: \(\tau= 10, 25, 50, 75\) and 125 and for \(n= 100, 200, 300, 500, 700, 1300, 2000\) and 5000 separately.

Parameters that are held constant are seen in 3.2

**Table 3.2:** Predetermined parameter values for the Kdq-tree method

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>Min.cell side</td>
<td>(\tau)</td>
<td>0.01</td>
</tr>
<tr>
<td>Chain sample size</td>
<td>(k)</td>
<td>0.4n</td>
</tr>
<tr>
<td>Significance level</td>
<td>(\alpha)</td>
<td>1</td>
</tr>
<tr>
<td>Bootstrap samples</td>
<td>(m)</td>
<td>1000</td>
</tr>
</tbody>
</table>

In test 1, ’The specific method parameters dependence for an abrupt change’ all window sizes \(n\) are tested for three values of the node split constraint. The node split constraint are: \((0.1n)/2, 0.1n\) and 3\((0.1n)/2\).
Chapter 4

Experiments and Results for a synthetic dataset

In this chapter the results of the experiments described in chapter 3 are reported. First the results for the Overlapping method are presented and thereafter the results for the Kdq-tree method. The results in the tables are based on the average of 20 tests and the figures shows the result of an example of one test.

In Table 4.1 the explanation of the performance measures in the tables that will follow are presented.

<table>
<thead>
<tr>
<th>Performance measure</th>
<th>Symbol</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completed detection</td>
<td>%</td>
<td>the ratio of how many of the 20 tests that actually found the added change</td>
</tr>
<tr>
<td>Detection delay, mean</td>
<td>samples</td>
<td>the mean of detection delay for the tests that found the added change</td>
</tr>
<tr>
<td>Detection delay, standard deviation</td>
<td>samples</td>
<td>the standard deviation of detection delay for the tests that found the added change</td>
</tr>
<tr>
<td>Additional alarm detection</td>
<td>%</td>
<td>the ratio of how many of the 20 tests that gave additional alarms</td>
</tr>
<tr>
<td>Detected additional alarms, mean</td>
<td>samples</td>
<td>the mean of additional alarms for the tests that found the added change</td>
</tr>
<tr>
<td>Detected additional alarms, standard deviation</td>
<td>samples</td>
<td>the standard deviation of additional alarms for the tests that found the added change</td>
</tr>
<tr>
<td>False alarm detection</td>
<td>%</td>
<td>the ratio of how many of the 20 tests that gave a false alarm detection</td>
</tr>
<tr>
<td>Detected false alarms, mean</td>
<td>samples</td>
<td>the mean number of detected false alarms for the tests that had a false alarm</td>
</tr>
<tr>
<td>Detected false alarms, standard deviation</td>
<td>samples</td>
<td>the standard deviation of number of detected false alarms for the tests that had a false alarm</td>
</tr>
<tr>
<td>Non-completed detection</td>
<td>numbers</td>
<td>used in &quot;Change of total number of added changes&quot; and gives the ratio of the 20 tests that could not find any of the changes</td>
</tr>
<tr>
<td>Bootstrap d, mean</td>
<td>KL-distance measure</td>
<td>the mean of the KL-distance estimated from the Bootstrap method</td>
</tr>
<tr>
<td>Bootstrap d, standard deviation</td>
<td>KL-distance measure</td>
<td>the standard deviation of the KL-distance estimated from the Bootstrap method</td>
</tr>
</tbody>
</table>

The Bootstrap estimated threshold, $d$, that is estimated by the Bootstrap method for the Kdq-tree method is not a performance measure but it is interesting to study because it is not set manually as in the Overlapping method. The test result is based on 20 tests and the Bootstrap estimate $d$ is computed for every test because it is based on the dataset which is made for every test.

In the tables in the section for "linear change" there are two main changes, i.e. when the linear change starts and when it ends. Therefore many of the performance measures mentioned in Table 4.1 is divided into "linear change starts" and "linear change ends".

In the chapter "Change of total number of added changes" the performance measure "completed detection" gives the ratio of the 20 tests that found all the added changes.

The tables can contain a space marked with "-" and it means that there is no result available. An example is if "completed detection" is 0, then one can not expect any "additional alarms" nor a "detection delay" and therefore these fields are marked with ".-".
4.1 The Overlapping method

Parameters that are set constant for all the tests for the Overlapping Method can be seen in Table 3.1. See chapter 3.1 for more details.

4.1.1 Performance for specific method-parameters

In the following tests the detection performance is analyzed when changing the window size $N$ and the decay rate $T$ for the Overlapping method. The tests are first done over a random abrupt change and thereafter over a random linear change.

**Abrupt change**

In the following two tests the parameters $N$ and $T$ are varied for a random abrupt change, to check their impact on the performance for a random abrupt change for the Overlapping method. In Table 4.2 the decay rate $T$'s impact on the change detection performance is analyzed for a random abrupt change for the Overlapping method with a window size $N=1000$.

**Table 4.2:** Performance of detection over a random abrupt change when varying the decay parameter $T$ with $N=1000$ for the Overlapping method

<table>
<thead>
<tr>
<th>$T$</th>
<th>$N=1000$</th>
<th>$T=0.1$</th>
<th>$T=0.2$</th>
<th>$T=0.3$</th>
<th>$T=0.4$</th>
<th>$T=0.5$</th>
<th>$T=0.6$</th>
<th>$T=0.7$</th>
<th>$T=0.8$</th>
<th>$T=0.9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completed detection</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Detection delay, mean</td>
<td>50.5</td>
<td>59.5</td>
<td>43.9</td>
<td>66.2</td>
<td>39.1</td>
<td>101.0</td>
<td>36.3</td>
<td>65.9</td>
<td>63.6</td>
<td></td>
</tr>
<tr>
<td>Detection delay, standard deviation</td>
<td>37.4</td>
<td>55.5</td>
<td>44.7</td>
<td>86.6</td>
<td>46.2</td>
<td>123.8</td>
<td>38.8</td>
<td>90.9</td>
<td>97.3</td>
<td></td>
</tr>
<tr>
<td>Detected additional alarms, mean</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0.05</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Detected additional alarms, standard deviation</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Detected false alarms, mean</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Detected false alarms, standard deviation</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

According to Table 4.2 the decay rate $T=0.7$, for a window of size $N=1000$, gives the lowest mean of the detection delay, 36.3 samples, with a relatively low standard deviation, 38.8 samples. Even so (despite that) $T=0.5$ is used in the following tests.

In Table 4.3 the window size $N$'s impact on the change detection performance is analyzed with a decay rate $T=0.5$ for a random abrupt change for the Overlapping method.

**Table 4.3:** Performance of detection over a random abrupt change when varying the model size $N$ with $T=0.5$ for the Overlapping method

<table>
<thead>
<tr>
<th>$N$</th>
<th>$T=0.5$</th>
<th>$N=100$</th>
<th>$N=200$</th>
<th>$N=300$</th>
<th>$N=500$</th>
<th>$N=700$</th>
<th>$N=1300$</th>
<th>$N=2000$</th>
<th>$N=5000$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completed detection</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Detection delay, mean</td>
<td>6.25</td>
<td>10.3</td>
<td>12.6</td>
<td>25.5</td>
<td>86.9</td>
<td>135.5</td>
<td>57.0</td>
<td>235.7</td>
<td></td>
</tr>
<tr>
<td>Detection delay, standard deviation</td>
<td>7.6</td>
<td>10.1</td>
<td>11.7</td>
<td>47.0</td>
<td>80.1</td>
<td>132.4</td>
<td>50.9</td>
<td>534.6</td>
<td></td>
</tr>
<tr>
<td>Detected additional alarms, mean</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Detected additional alarms, standard deviation</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>False alarm detection</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Detected false alarms, mean</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detected false alarms, standard deviation</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

According to Table 4.3 the window size could be set to a value between $N=200$ to $N=2000$ for a decay rate of $T=0.2$. Thees values give pretty low mean and standard deviation of the detection
delay and no false alarms are detected.

Figure 4.1 shows an example of a test where a data stream with a random abrupt change can be seen together with a line that indicates change detection at sample 500024. The change is detected with a delay of 24 samples, with no additional alarms. In Figure 5.1.1 the Kullback-Leibler distance is shown with a big measure around sample 500024. The window size is $N=300$ and the decay rate is $T=0.5$. The Kullback-Leibler measure is also high a little bit after sample 500025 which is hard to see in this figure because the interval is very small.

![Figure 4.1](image1.png)  
![Figure 4.2](image2.png)

**Figure 4.1:** A data stream with a random abrupt change and the change detection for the Overlapping method where $N=300$ and $T=0.5$

**Figure 4.2:** The Kullback-Leibler distance for a random abrupt change for the Overlapping method where $N=300$ and $T=0.5$

### Linear change

In the following two tests the parameters $N$ and $T$ are varied for a linear change, to check their impact on the performance for a random linear change for the Overlapping method.

In Table 4.4 the decay rate $T$’s impact on the change detection performance is shown for a linear change for the Overlapping method.

**Table 4.4:** Performance of detection over a random linear change when varying the decay parameter $T$ with $N=1000$ for the Overlapping method

<table>
<thead>
<tr>
<th></th>
<th>$T=0.1$</th>
<th>$T=0.2$</th>
<th>$T=0.3$</th>
<th>$T=0.4$</th>
<th>$T=0.5$</th>
<th>$T=0.6$</th>
<th>$T=0.7$</th>
<th>$T=0.8$</th>
<th>$T=0.9$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completed detection, linear change starts:</td>
<td>0.7</td>
<td>0.8</td>
<td>0.55</td>
<td>0.45</td>
<td>0.55</td>
<td>0.45</td>
<td>0.4</td>
<td>0.05</td>
<td>0.55</td>
</tr>
<tr>
<td>Completed detection, linear change ends:</td>
<td>0.35</td>
<td>0.15</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Detections with additional alarms, linear change starts:</td>
<td>0.35</td>
<td>0.15</td>
<td>0.45</td>
<td>0.25</td>
<td>0</td>
<td>0.05</td>
<td>0.05</td>
<td>0.35</td>
<td>0.2</td>
</tr>
<tr>
<td>Detections with additional alarms, linear change ends:</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Detection delay, linear change starts, mean:</td>
<td>1041.9</td>
<td>1544</td>
<td>1845</td>
<td>1815.5</td>
<td>871</td>
<td>1206</td>
<td>899</td>
<td>1961.0</td>
<td>1174.1</td>
</tr>
<tr>
<td>Detection delay, linear change starts, standard deviation:</td>
<td>462.7</td>
<td>893.4</td>
<td>1854.6</td>
<td>1752.0</td>
<td>342</td>
<td>451</td>
<td>1836.4</td>
<td>865.8</td>
<td></td>
</tr>
<tr>
<td>Detected additional alarms after linear change starts, mean:</td>
<td>44.4</td>
<td>38.6</td>
<td>30.4</td>
<td>10.5</td>
<td>-</td>
<td>1</td>
<td>2</td>
<td>2.3</td>
<td>1.5</td>
</tr>
<tr>
<td>Detected additional alarms after linear change starts, standard deviation:</td>
<td>12.0</td>
<td>18.0</td>
<td>15.6</td>
<td>12.6</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>1.4</td>
<td>1.0</td>
</tr>
<tr>
<td>Detection delay, linear change ends, mean:</td>
<td>109.8</td>
<td>201</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Detection delay, linear change ends, standard deviation:</td>
<td>19.0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Detected additional alarms after linear change ends, mean:</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Detected additional alarms after linear change ends, standard deviation:</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>False alarm detection:</td>
<td>0.05</td>
<td>0.05</td>
<td>0.15</td>
<td>0.005</td>
<td>0</td>
<td>0</td>
<td>0.05</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Detected false alarms, mean:</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>1</td>
<td>-</td>
</tr>
<tr>
<td>Detected false alarms, standard deviation:</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
</tr>
</tbody>
</table>
According to Table 4.4 the decay rate $T=0.2$, for a window size of $N=1000$, gives the highest completed detection rate, 0.8, and also a relatively low mean of the detection delay, 1544 samples, with a relatively low standard deviation, 894 samples. It only gives one false alarm in 1 of the 20 tests.

In Table 4.5 the window size $N$’s impact on the change detection performance is shown for a random linear change for the Overlapping method.

Table 4.5: Performance of detection over a random linear change when varying the model size $N$ with $T=0.2$ for the Overlapping method

<table>
<thead>
<tr>
<th>$T=0.2$</th>
<th>N=100</th>
<th>N=200</th>
<th>N=300</th>
<th>N=500</th>
<th>N=700</th>
<th>N=1300</th>
<th>N=2000</th>
<th>N=5000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completed detection, linear change starts:</td>
<td>1</td>
<td>0.85</td>
<td>0.35</td>
<td>0.2</td>
<td>0.4</td>
<td>0.95</td>
<td>0.95</td>
<td>1</td>
</tr>
<tr>
<td>Completed detection, linear change ends:</td>
<td>1</td>
<td>0.45</td>
<td>0</td>
<td>0</td>
<td>0.05</td>
<td>0.2</td>
<td>0.1</td>
<td>0.1</td>
</tr>
<tr>
<td>Detections with additional alarms, linear change starts:</td>
<td>1</td>
<td>0.65</td>
<td>0.2</td>
<td>0.2</td>
<td>0.35</td>
<td>0.95</td>
<td>0.85</td>
<td>0.25</td>
</tr>
<tr>
<td>Detections with additional alarms, linear change ends:</td>
<td>1</td>
<td>0.2</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Detection delay, linear change starts, mean:</td>
<td>332.7</td>
<td>11954.7</td>
<td>6598.3</td>
<td>1172.6</td>
<td>5025.5</td>
<td>1312</td>
<td>1558.2</td>
<td>2400.1</td>
</tr>
<tr>
<td>Detection delay, linear change ends, mean:</td>
<td>395.4</td>
<td>9409.4</td>
<td>10400.9</td>
<td>990.8</td>
<td>9790</td>
<td>476</td>
<td>418.9</td>
<td>1713.2</td>
</tr>
<tr>
<td>Detected additional alarms, linear change starts: mean:</td>
<td>101.2</td>
<td>35.8</td>
<td>129.5</td>
<td>75</td>
<td>49.9</td>
<td>31.7</td>
<td>23.5</td>
<td>9</td>
</tr>
<tr>
<td>Detected additional alarms, linear change ends, standard deviation:</td>
<td>26.1</td>
<td>84.0</td>
<td>61.3</td>
<td>46.0</td>
<td>24.2</td>
<td>12.4</td>
<td>1.5</td>
<td>0</td>
</tr>
<tr>
<td>Detection delay, linear change ends, mean:</td>
<td>547.3</td>
<td>11110</td>
<td>-</td>
<td>-</td>
<td>41</td>
<td>135.8</td>
<td>189.5</td>
<td>1001</td>
</tr>
<tr>
<td>Detection delay, linear change ends, standard deviation:</td>
<td>500.0</td>
<td>5397.0</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>96.9</td>
<td>180.3</td>
<td>3.260</td>
</tr>
<tr>
<td>Detected additional alarms, linear change ends: mean:</td>
<td>41.9</td>
<td>1.25</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Detected additional alarms, linear change ends, standard deviation:</td>
<td>5.2</td>
<td>0.5</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>False alarm detection:</td>
<td>1</td>
<td>0.45</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Detected false alarms, mean:</td>
<td>44.25</td>
<td>1.1</td>
<td>1</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Detected false alarms, standard deviation:</td>
<td>5.2</td>
<td>0.3</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
</tbody>
</table>

According to Table 4.5 the window size $N=1300$, with a decay rate $T=0.2$, gives the highest completed detection, 0.95, and the lowest mean of detection delay, 1312 samples, with a relatively low standard deviation, 476 samples, without any false alarms.

Figure 4.3 shows an example of a test where the data stream with a random linear change can be seen together with lines that indicate a change detection. The first change is detected when the linear change starts, a delay of 1267 samples, with 37 additional alarms. There is no detection when the linear change ends and there are no false alarms. The window size is $N=1300$ and the decay rate is $T=0.2$. The Kullback-Leibler distance can be seen in Figure 4.4 which clearly shows all the high values of the KL-measure.
4.1.2 The window size $N$ vs. the decay rate $T$

In the following figures the performance of detection’s impact of the dependence between the window size, $N$, and the decay rate, $T$, for the Overlapping method is shown. The tests are made over a random abrupt change for all the window sizes, $n$, and for all decay rates $T$. The first figure shows how the performance measure 'false alarm rate' is affected and the second figure how the performance measure 'detection delay' is affected.

A random abrupt change

In Figure 4.5 the impact of the performance measure 'False alarm rate' for varying the window size, $N$, and the decay rate $T$ is shown for the Overlapping method.

In Figure 4.5 the false alarm rate is 0 for all the window sizes with a decay rate $T$ between 0.4 and 0.5, except from the window size $N=100$ which needs a decay rate $T=0.5$ for a good false alarm rate. The best result in this test, i.e. the lowest false alarm rate, is given by $N=1000$, $N=2000$ and $N=5000$ because they have a false alarm rate of 0% for all decay rates $T$. The window sizes $N=500$, $N=700$ and $N=1300$ have a pretty high false alarm rate for some decay rates $T$ but the mean of false
alarms are only 1 alarm with a standard deviation of 0 alarms.

In Figure 4.6 the impact of the performance measure 'Detection delay, mean' for varying the window size, $N$, and decay rate $T$ is shown for the Overlapping method.

![Decay rate vs. window size](image)

**Figure 4.6**: The impact on the detection delay for varying the window size $N$ and the decay rate $T$ for the Overlapping method over a random abrupt change

In Figure 4.6 the detection delay is pretty low for all window sizes, except from $N=5000$. The best result in this test, i.e. the smallest detection delay is given by a small window size.

4.1.3 Different size of changes

In the following tests the performance of detection for different sizes of changes is analyzed. The first test changes the size of mean $\mu$, the second test changes the size of the covariance matrix $\Sigma$ and the third test changes the size of $\mu$ in a linear change.

Abrupt change in $\mu$

In Table 4.6 the size of $\mu$ in an abrupt change’s impact on the change detection performance is shown for the Overlapping method. The change in $\mu$ is varied with $k_\mu$ in (3.2). The parameter window size is $N=300$ and the decay rate $T=0.5$.

**Table 4.6**: Performance of detection when varying the size $\mu$ in an abrupt change for the Overlapping method where $T=0.5$ and $N=300$

<table>
<thead>
<tr>
<th>$k_\mu$</th>
<th>$-10$</th>
<th>$-3$</th>
<th>$-1.5$</th>
<th>$-1.3$</th>
<th>$-1$</th>
<th>$-0.8$</th>
<th>$-0.6$</th>
<th>$0$</th>
<th>$0.2$</th>
<th>$0.35$</th>
<th>$0.35$</th>
<th>$0.55$</th>
<th>$0.95$</th>
<th>$1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completed detection :</td>
<td>1</td>
<td>1</td>
<td>0.41</td>
<td>0.44</td>
<td>0.41</td>
<td>0.15</td>
<td>0.15</td>
<td>0.2</td>
<td>0</td>
<td>0.35</td>
<td>0.35</td>
<td>0.55</td>
<td>0.95</td>
<td>1</td>
</tr>
<tr>
<td>Detection delay, mean:</td>
<td>2.25</td>
<td>34.7</td>
<td>85.6</td>
<td>47.3</td>
<td>64.1</td>
<td>25.3</td>
<td>44.7</td>
<td>65.8</td>
<td>-</td>
<td>38.6</td>
<td>64.6</td>
<td>72.2</td>
<td>33.2</td>
<td>2.2</td>
</tr>
<tr>
<td>Detection delay, standard deviation:</td>
<td>1.4</td>
<td>42.9</td>
<td>84.6</td>
<td>24.2</td>
<td>69.3</td>
<td>35.3</td>
<td>25.7</td>
<td>74.1</td>
<td>-</td>
<td>17.6</td>
<td>30.6</td>
<td>71.4</td>
<td>24.6</td>
<td>1.6</td>
</tr>
<tr>
<td>Detected additional alarms, mean:</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Detected additional alarms, standard deviation:</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>False alarm detection:</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

In Table 4.6 is can be seen that for $k_\mu=\pm 10$ the method finds the change in all tests with a delay that is close to 2 samples and with a small standard deviation. The size of change in $\mu$ need to be over $k_\mu=3$ to be sure of detection.

Figure 4.7 shows an example of a test where a data set with an abrupt change in $\mu$, with $k_\mu=3$, can be seen together with the change detection line at sample 50002. The detection delay is 2 samples and there are no additional alarms or false alarms. The method parameters were set to $N=300$ and $T=0.5$ for the Overlapping method. The Kullback-Leibler distance is shown in Figure 4.8 and it can be seen that the KL-measure is high for some samples right after sample 50002.
Figure 4.7: A data stream with an abrupt change in the mean vector, \( k_\mu = 3 \), and the change detection for the Overlapping method where \( N = 300 \) and \( T = 0.5 \).

Figure 4.8: The Kullback-Leibler distance for an abrupt change in the mean vector, \( k_\mu = 3 \), for the Overlapping method where \( N = 300 \) and \( T = 0.5 \).

Abrupt change in \( \Sigma \)

In Table 4.7 the size of \( \Sigma \) in an abrupt change’s impact on the change detection performance is shown for the Overlapping method. The change in \( \mu \) is varied with \( k_\sigma \) in (3.3). The parameter window size is \( N = 300 \) and the decay rate \( T = 0.5 \).

Table 4.7: Performance of detection when varying the size \( \Sigma \) in an abrupt change for the Overlapping method where \( T = 0.5 \) and \( N = 300 \)

<table>
<thead>
<tr>
<th>( T = 0.5 ), ( N = 300 )</th>
<th>( k = 0.6 )</th>
<th>( k = 0.8 )</th>
<th>( k = 1 )</th>
<th>( k = 1.2 )</th>
<th>( k = 1.5 )</th>
<th>( k = 2 )</th>
<th>( k = 2.5 )</th>
<th>( k = 5 )</th>
<th>( k = 10 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completed detection :</td>
<td>0</td>
<td>0.15</td>
<td>0.2</td>
<td>0.25</td>
<td>0.25</td>
<td>0.55</td>
<td>0.4</td>
<td>0.55</td>
<td>0.95</td>
</tr>
<tr>
<td>Detection delay, mean:</td>
<td>-</td>
<td>50</td>
<td>47.8</td>
<td>55.0</td>
<td>44.2</td>
<td>34.6</td>
<td>34.5</td>
<td>36.7</td>
<td>44.7</td>
</tr>
<tr>
<td>Detection delay, standard deviation:</td>
<td>-</td>
<td>28.6</td>
<td>32.6</td>
<td>34.4</td>
<td>37.1</td>
<td>29.2</td>
<td>21.5</td>
<td>29.2</td>
<td>48.0</td>
</tr>
<tr>
<td>Detected additional alarms, mean:</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Detected additional alarms, standard deviation:</td>
<td>-</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>False alarm detection:</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

In Table 4.7 it can be seen that a change in the covariance matrix \( \Sigma \) must be over \( k_\sigma = 10 \) to be sure of a detection when the thresholds in Table 3.1 are used.

Figure 4.9 shows an example of a test where a data stream with a change of size \( k_\sigma = 10 \) in the covariance matrix can be seen together with a change detection line at sample 500032. The detection delay is 32 samples with no additional alarms or false alarms. In Figure 4.10 the Kullback-Leibler distance is shown.

23
Figure 4.9: A data stream with an abrupt change in the covariance matrix, $k_{cov} = 10$, and the change detection for the Overlapping method where $N=300$ and $T=0.5$.

Figure 4.10: The Kullback-Leibler distance for an abrupt change in the covariance matrix, $k_{cov} = 10$, for the Overlapping method where $N=300$ and $T=0.5$.

Linear change

In Table 4.8 the size of a linear change’s impact on the change detection performance is shown for the Overlapping method. The linear change in $\mu$ is varied with $k$ in (3.1). The parameter window size is $N=300$ and the decay rate $T=0.5$ for the Overlapping method.

Table 4.8: Performance of detection when varying the linear change size, $k$, for the Overlapping method where $T=0.2$ and $N=1300$

<table>
<thead>
<tr>
<th></th>
<th>$k=0.1$</th>
<th>$k=0.01$</th>
<th>$k=0.001$</th>
<th>$k=0.0005$</th>
<th>$k=0.0001$</th>
<th>$k=-0.0001$</th>
<th>$k=-0.0005$</th>
<th>$k=-0.001$</th>
<th>$k=-0.01$</th>
<th>$k=-0.1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completed detection, linear change starts:</td>
<td>1</td>
<td>1</td>
<td>0.95</td>
<td>0.45</td>
<td>0</td>
<td>0.35</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Completed detection, linear change ends:</td>
<td>0</td>
<td>0</td>
<td>0.05</td>
<td>0</td>
<td>0</td>
<td>0.05</td>
<td>0.1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Detections with additional alarms, linear change starts:</td>
<td>0</td>
<td>0.15</td>
<td>0.95</td>
<td>0.45</td>
<td>-</td>
<td>0.3</td>
<td>1</td>
<td>0.25</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Detections with additional alarms, linear change ends:</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Detection delay, linear change starts, mean:</td>
<td>49</td>
<td>234.3</td>
<td>1522.2</td>
<td>24656</td>
<td>-</td>
<td>4463.3</td>
<td>1968.6</td>
<td>228</td>
<td>33.8</td>
<td></td>
</tr>
<tr>
<td>Detection delay, linear change starts, standard deviation:</td>
<td>15.2</td>
<td>60.6</td>
<td>1115.0</td>
<td>3346.6</td>
<td>-</td>
<td>6156.9</td>
<td>2818.2</td>
<td>94.2</td>
<td>14.8</td>
<td></td>
</tr>
<tr>
<td>Detected additional alarms, linear change starts: mean:</td>
<td>-</td>
<td>1</td>
<td>32.4</td>
<td>26.2</td>
<td>-</td>
<td>25.2</td>
<td>30.1</td>
<td>1</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Detected additional alarms, linear change starts, standard deviation:</td>
<td>-</td>
<td>0</td>
<td>9.4</td>
<td>16.0</td>
<td>-</td>
<td>18.0</td>
<td>13.9</td>
<td>0</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Detection delay, linear change ends, mean:</td>
<td>-</td>
<td>-</td>
<td>11</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>84.5</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Detection delay, linear change ends, standard deviation:</td>
<td>-</td>
<td>-</td>
<td>0</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>40.3</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Detected additional alarms, linear change ends: mean:</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>Detected additional alarms, linear change ends, standard deviation:</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td>-</td>
<td></td>
</tr>
<tr>
<td>False alarm detection:</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

According to Table 4.8 the linear change must have a linear change in $\mu$ that is higher than $k=0.001$, otherwise the probability for not detecting the change is low (only 0.35 for $k=0.0005$).

Figure 4.11 shows an example of a test where a data stream with a linear change of $k=0.001$ can be seen together with lines that indicate a change detection. The first change is detected after the linear change starts, a delay of 1545 samples, with 37 additional alarms but no detection when the linear change ends. No false alarms are detected. The window size $N=1300$ and the decay rate is $T=0.2$. The Kullback-Leibler distance can be seen in Figure 4.12.
4.1.4 Change of total number of added changes

Abrupt change in $\mu$

In Table 4.9 the performance for different number of added abrupt changes can be seen for the Overlapping method. The abrupt change is in $\mu$, with $k_\mu = \pm 3$, where the sign of $k_\mu$ is randomly chosen, see (3.2). The parameters are set to $N=300$ and $T=0.5$.

Table 4.9: Performance of detection when varying the amount of abrupt changes in $\mu$ with $k_\mu = \pm 3$ for the Overlapping method where $T=0.5$ and $N=300$

<table>
<thead>
<tr>
<th></th>
<th>T=0.5, N=300</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>9 49 199 249 263 294</td>
</tr>
<tr>
<td>Completed detection, where all changes were found(in percent):</td>
<td>1 1 0.9 0.6 0 0</td>
</tr>
<tr>
<td>Non-completed detection:</td>
<td>- - - - - -</td>
</tr>
<tr>
<td>Completed detection, mean:</td>
<td>9 49 195.1 223.7 204.1 194.4</td>
</tr>
<tr>
<td>Completed detection, standard deviation:</td>
<td>0 0 17.4 55.1 34.6 40.7</td>
</tr>
<tr>
<td>Detected additional alarms, mean:</td>
<td>0 0 0 0 0 0</td>
</tr>
<tr>
<td>Detected additional alarms, standard deviation:</td>
<td>0 0 0 0 0 0</td>
</tr>
<tr>
<td>False alarm detection:</td>
<td>0 0 0 0 0 0</td>
</tr>
</tbody>
</table>

In Table 4.9 it can be seen that for abrupt changes in $\mu$ with $k_\mu = \pm 3$ the method finds almost all the number of changes if the number are below 199. There are no false alarms.

Figure 4.13 shows an example of a test where a data set with an abrupt change in $\mu$, with $k_\mu=3$, can be seen together with the change detection lines. All the 19 changes are detected when the window size is $N=300$ and the decay rate is $T=0.5$ in the Overlapping method. The Kullback-Leibler distance is shown in Figure 4.14.
4.2 The Kdq-tree method

Parameters that are set constant for all the tests for the Kdq-tree method are seen in Table 3.2. See chapter 3.1 for more details.

4.2.1 Performance for specific method-parameters

The first experiments test how the detection performance for the Kdq-tree method is affected by the window size $n$ and the node split constraint $\tau$. The tests are first done over a random abrupt change and thereafter over a random linear change.

**Abrupt change**

In the following two tests the parameters $n$ and $\tau$ are varied for a random abrupt change, to check their impact on the performance for a random abrupt change for the Kdq-tree method. In Table 4.2.3 the node split constraint $\tau$'s impact on the change detection performance is analyzed for a window size $n=1000$, for a random abrupt change for the Kdq-tree method.

**Table 4.10: Performance of detection over a random abrupt change when varying $\tau$ with $n=1000$ for the Kdq-tree method**

<table>
<thead>
<tr>
<th>$\tau$</th>
<th>n=1000</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\tau=10$</td>
<td>no tests</td>
</tr>
<tr>
<td>Detection delay, mean:</td>
<td>556</td>
</tr>
<tr>
<td>Detection delay, standard deviation:</td>
<td>71</td>
</tr>
<tr>
<td>Additional alarms detection:</td>
<td>1</td>
</tr>
<tr>
<td>Detected additional alarms, mean:</td>
<td>1324</td>
</tr>
<tr>
<td>Detected additional alarms, standard deviation:</td>
<td>101</td>
</tr>
<tr>
<td>False alarm detection:</td>
<td>0.65</td>
</tr>
<tr>
<td>Detected false alarms, mean:</td>
<td>200</td>
</tr>
<tr>
<td>Detected false alarms, standard deviation:</td>
<td>4</td>
</tr>
</tbody>
</table>
According to Table 4.2.3, the node split constraint $\tau = 50$, for a window of size $n = 1000$, gives minimum false alarms and the detection delay is 557 samples, with a relatively low standard deviation, 69 samples.

In Table 4.11, the node split constraint $\tau$’s impact on the change detection performance is analyzed for a random abrupt change for the Kdq-tree method. For $n = 5000$ the results are not made because it was too time-inefficient.

Table 4.11: Performance of detection over a random abrupt change when varying the model size $n$ with $\tau = 50$ for the Kdq-tree method

<table>
<thead>
<tr>
<th>$\tau = 50$</th>
<th>n=100</th>
<th>n=200</th>
<th>n=300</th>
<th>n=500</th>
<th>n=700</th>
<th>n=1000</th>
<th>n=1300</th>
<th>n=2000</th>
<th>n=5000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completed detection:</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>no tests</td>
</tr>
<tr>
<td>Detection delay, mean:</td>
<td>-</td>
<td>-</td>
<td>170</td>
<td>272</td>
<td>352</td>
<td>583</td>
<td>714</td>
<td>1535</td>
<td></td>
</tr>
<tr>
<td>Detection delay, standard deviation:</td>
<td>-</td>
<td>-</td>
<td>129</td>
<td>89</td>
<td>108</td>
<td>69</td>
<td>77</td>
<td>113</td>
<td></td>
</tr>
<tr>
<td>Additional alarm detection:</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td>Detected additional alarms, mean:</td>
<td>-</td>
<td>-</td>
<td>3272</td>
<td>1678</td>
<td>1769</td>
<td>1296</td>
<td>1747</td>
<td>3897</td>
<td></td>
</tr>
<tr>
<td>Detected additional alarms, standard deviation:</td>
<td>-</td>
<td>-</td>
<td>3753</td>
<td>2317</td>
<td>1544</td>
<td>123</td>
<td>106</td>
<td>168</td>
<td></td>
</tr>
<tr>
<td>False alarm detection:</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.7</td>
<td>0.75</td>
<td>0.2</td>
<td>0.35</td>
<td>0.3</td>
<td></td>
</tr>
<tr>
<td>Detected false alarms, mean:</td>
<td>-</td>
<td>-</td>
<td>3098</td>
<td>1743</td>
<td>1790</td>
<td>2012</td>
<td>2601</td>
<td>4001</td>
<td></td>
</tr>
<tr>
<td>Detected false alarms, standard deviation:</td>
<td>-</td>
<td>-</td>
<td>3572</td>
<td>2959</td>
<td>1869</td>
<td>21</td>
<td>0</td>
<td>0</td>
<td></td>
</tr>
</tbody>
</table>

According to Table 4.11, the window size could be set to $N = 1000$ or more for $\tau = 50$. $N = 1000$ is the value that gives lowest probability for false alarms with a low detection delay of 583 samples with a standard deviation of 69 samples.

Figure 4.15 shows an example of a test where a data stream with a random abrupt change can be seen together with the line that indicates a change detection at sample 50504. The change is detected with a delay of 504 samples and some additional alarms. There were two false alarms detected as can be seen in the figure. The Bootstrap threshold for this test was estimated to $d = 0.1003$. The window size is $n = 1000$ and the node split constraint $\tau = 50$. In Figure 4.16, the Kullback-Leibler distance is shown and the two false alarms shown as a little bit higher KL-measure and with an extremely higher KL-measure that starts at sample 50504.
**Linear change**

In the following two tests the parameters $n$ and $\tau$ are varied for a random linear change, to check their impact on the performance for a random linear change for the Kdq-tree method.

In Table 4.12 $\tau$’s impact on the linear change detection performance is shown for a random linear change for the Kdq-tree method.

**Table 4.12:** Performance of detection over a random linear change when varying $\tau$ with $n=1000$ for the Kdq-tree method

| $\tau$ | Completed detection, linear change starts: | Completed detection, linear change ends: | Detections with additional alarms, linear change starts: | Detections with additional alarms, linear change ends: | Detection delay, linear change starts, mean: | Detection delay, linear change starts, standard deviation: | Detected additional alarms after linear change starts, mean: | Detected additional alarms after linear change starts, standard deviation: | Detection delay, linear change ends, mean: | Detection delay, linear change ends, standard deviation: | Detected additional alarms after linear change ends, mean: | Detected additional alarms after linear change ends, standard deviation: | False alarm detection: | Detected false alarms, mean: | Detected false alarms, standard deviation: |
|--------|-------------------------------------------|------------------------------------------|--------------------------------------------------------|--------------------------------------------------------|---------------------------------------------|---------------------------------------------|-----------------------------------------------|-----------------------------------------------|---------------------------------------------|---------------------------------------------|-----------------------------------------------|-----------------------------------------------|---------------------------------------------|---------------------------------------------|
| 10     | no tests                                  | 0.95 0.75 0.45                           | 1 1 0.9 0.85                                           | 0.95 0.85 0.75 0.45                                     | 2341 1911 2167 9974                          | 3181 3246 13733                              | 33253 32954 34509 11989                        | 17499 18481 20566 18195                        | 34 21 14 2650                              | 76 61 32 6790                               | 705 853 1381 468                           | 505 339 769 483                             | 0.6 0.35 0.35 0.25                          | 2001 1715 2244 6309                        | 0.6 756 1842 866                           |
| 25     |                                           |                                          | 1 1 0.9 0.85                                           | 0.95 0.85 0.75 0.45                                     | 2341 1911 2167 9974                          | 3181 3246 13733                              | 33253 32954 34509 11989                        | 17499 18481 20566 18195                        | 34 21 14 2650                              | 76 61 32 6790                               | 705 853 1381 468                           | 505 339 769 483                             | 0.6 0.35 0.35 0.25                          | 2001 1715 2244 6309                        | 0.6 756 1842 866                           |
| 50     |                                           |                                          | 1 1 0.9 0.85                                           | 0.95 0.85 0.75 0.45                                     | 2341 1911 2167 9974                          | 3181 3246 13733                              | 33253 32954 34509 11989                        | 17499 18481 20566 18195                        | 34 21 14 2650                              | 76 61 32 6790                               | 705 853 1381 468                           | 505 339 769 483                             | 0.6 0.35 0.35 0.25                          | 2001 1715 2244 6309                        | 0.6 756 1842 866                           |
| 75     |                                           |                                          | 1 1 0.9 0.85                                           | 0.95 0.85 0.75 0.45                                     | 2341 1911 2167 9974                          | 3181 3246 13733                              | 33253 32954 34509 11989                        | 17499 18481 20566 18195                        | 34 21 14 2650                              | 76 61 32 6790                               | 705 853 1381 468                           | 505 339 769 483                             | 0.6 0.35 0.35 0.25                          | 2001 1715 2244 6309                        | 0.6 756 1842 866                           |
| 125    |                                           |                                          | 1 1 0.9 0.85                                           | 0.95 0.85 0.75 0.45                                     | 2341 1911 2167 9974                          | 3181 3246 13733                              | 33253 32954 34509 11989                        | 17499 18481 20566 18195                        | 34 21 14 2650                              | 76 61 32 6790                               | 705 853 1381 468                           | 505 339 769 483                             | 0.6 0.35 0.35 0.25                          | 2001 1715 2244 6309                        | 0.6 756 1842 866                           |

According to Table 4.12 the node split constraint $\tau=50$, for a window size of $N=1000$, gives a pretty low false rate, 0.35. The Kdq-tree method detects when the random linear change starts in all of the tests with a mean of the detection delay of 2090 samples and a standard deviation of 1911 samples. In 0.85 of the tests it also detects when the linear change ends.

In Table 4.13 the window size N’s impact on the change detection performance is analyzed for $\tau=50$.
for a random linear change. For \( n=5000 \) the results are not made because it was too time-inefficient.

**Table 4.13:** Performance of detection over a random linear change when varying \( n \) with \( \tau=50 \) for the Kdq-tree method

<table>
<thead>
<tr>
<th>( n )</th>
<th>100</th>
<th>200</th>
<th>300</th>
<th>500</th>
<th>1000</th>
<th>1300</th>
<th>1800</th>
<th>2000</th>
<th>5000</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completed detection, linear change starts:</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.95</td>
<td>0.9</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>no tests</td>
</tr>
<tr>
<td>Completed detection, linear change ends:</td>
<td>0</td>
<td>0</td>
<td>0.95</td>
<td>0.8</td>
<td>0.7</td>
<td>0.85</td>
<td>0.9</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Detections with additional alarms, linear change starts:</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0.95</td>
<td>0.85</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Detections with additional alarms, linear change ends:</td>
<td>0</td>
<td>0</td>
<td>0.95</td>
<td>0.8</td>
<td>0.7</td>
<td>0.85</td>
<td>0.9</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Detection delay, linear change starts, mean:</td>
<td>-</td>
<td>-</td>
<td>2332</td>
<td>3656</td>
<td>2038</td>
<td>2090</td>
<td>2161</td>
<td>2283</td>
<td></td>
</tr>
<tr>
<td>Detection delay, linear change starts, standard deviation:</td>
<td>-</td>
<td>-</td>
<td>4579</td>
<td>9173</td>
<td>2829</td>
<td>1911</td>
<td>2178</td>
<td>571</td>
<td></td>
</tr>
<tr>
<td>Detected additional alarms after linear change starts, mean:</td>
<td>-</td>
<td>-</td>
<td>6233</td>
<td>15520</td>
<td>34878</td>
<td>35146</td>
<td>39737</td>
<td>46993</td>
<td></td>
</tr>
<tr>
<td>Detected additional alarms after linear change starts, standard deviation:</td>
<td>-</td>
<td>-</td>
<td>5019</td>
<td>17985</td>
<td>59983</td>
<td>17551</td>
<td>17274</td>
<td>2570</td>
<td></td>
</tr>
<tr>
<td>Detection delay, linear change ends, mean:</td>
<td>-</td>
<td>-</td>
<td>2316</td>
<td>2275</td>
<td>150</td>
<td>46</td>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Detection delay, linear change ends, standard deviation:</td>
<td>-</td>
<td>-</td>
<td>5190</td>
<td>5187</td>
<td>229</td>
<td>61</td>
<td>176</td>
<td>0</td>
<td></td>
</tr>
<tr>
<td>Detected additional alarms after linear change ends, mean:</td>
<td>-</td>
<td>-</td>
<td>1266</td>
<td>761</td>
<td>952</td>
<td>853</td>
<td>118</td>
<td>2526</td>
<td></td>
</tr>
<tr>
<td>Detected additional alarms after linear change ends, standard deviation:</td>
<td>-</td>
<td>-</td>
<td>1264</td>
<td>1018</td>
<td>1588</td>
<td>340</td>
<td>498</td>
<td>683</td>
<td></td>
</tr>
<tr>
<td>False alarm detection:</td>
<td>0</td>
<td>0</td>
<td>0.9</td>
<td>0.7</td>
<td>0.6</td>
<td>0.35</td>
<td>0.4</td>
<td>0.35</td>
<td></td>
</tr>
<tr>
<td>Detected false alarms, mean:</td>
<td>-</td>
<td>-</td>
<td>1386</td>
<td>1222</td>
<td>1745</td>
<td>1716</td>
<td>2276</td>
<td>4001</td>
<td></td>
</tr>
<tr>
<td>Detected false alarms, standard deviation:</td>
<td>-</td>
<td>-</td>
<td>1417</td>
<td>1381</td>
<td>2144</td>
<td>756</td>
<td>919</td>
<td>2003</td>
<td></td>
</tr>
</tbody>
</table>

In Table 4.13 it can be seen that a bigger window size \( n \) gives a higher completed detection rate when the linear change starts. For \( n=1000 \) all the changes when the linear change starts are detected. The detection delay is above 2000 samples which is pretty high but has to do with the big window size. There are false alarms detected in 0.35 of the tests. When there is false alarms detected there are many of them, as can be seen under "detected false alarms: mean", which is 1716 false alarms. When the window size \( n \) is small there are no detections at all, see the result for \( n=100 \) and \( n=200 \).

Figure 4.17 shows an example of a test where a data stream with a random linear change in \( \mu \) can be seen together with the lines that indicates change detection. The window size is \( n=1000 \) and the node split constraint is \( \tau=50 \). The change which is detected when the linear change starts has a delay of 957 samples and of 1 sample when the linear change ends. The Bootstrap threshold for this test was estimated to \( d=0.0871 \). There were no false alarms detected as can be seen in the picture. During the linear change there are 21475 additional alarms and when the linear change ends it detects 248 additional alarms, which are clearly shown in the figure. In Figure 4.18 the Kullback-Leibler distance is shown and the higher KL-measure is clearly shown.

**Figure 4.17:** A data stream with a random linear change and the change detection for the Kdq-tree method \( n=1000 \) and \( \tau=50 \)

**Figure 4.18:** The Kullback-Leibler distance for a random linear change for the Kdq-tree method \( n=1000 \) and \( \tau=50 \)
4.2.2 The window size $n$ vs. the node split constraint $\tau$

In the following figures the dependence between the window size, $n$, and the node split constraint $\tau$ for the Kdq-tree method is shown. The tests are made over a random abrupt change for all the window sizes, $n$, with three different node split constraints for each window size. The three different node split constrains are $\tau=n\ast0.1-n/2$, $\tau=n\ast0.1$ and $\tau=n\ast0.1+n/2$. The figures shows how the performance measures: false alarm rate and detection delay are affected. It also shows how the average Bootstrap estimate of the threshold, $d$, is affected.

A random abrupt change

In Figure 4.19 the impact on the performance measure "False alarm rate" for varying the window size, $n$, and node split constraint $\tau$ and how it affects the performance of detection for the Kdq-tree method is shown. For $n=5000$ the result is not shown in the figure because of the scaling: $\tau=250$ gave 45% false alarms, $\tau=500$ gave 50% false alarms and $\tau=750$ gave 40% false alarms.

![The Kdq-tree method](image)

**Figure 4.19**: The impact on the false alarm rate for varying the window size $n$ and the node split constraint $\tau$ for the Kdq-tree method over a random abrupt change

In Figure 4.19 the false alarm rate is very high for the window sizes $n=100$, $n=200$, $n=300$ and $n=500$. The best result in this test, i.e. the lowest false alarm rate, is given by $n=1000$ and $\tau=50$, a false alarm rate of 20%.

In Figure 4.20 the impact on the performance measure 'False alarm rate' for varying the window size, $n$, and node split constraint $\tau$ is shown for the Kdq-tree method. For $n=5000$ the result is not shown because of the scaling: $\tau=250$ gave a detection delay of 2674 samples, $\tau=500$ gave a detection delay of 1395 samples and $\tau=750$ gave a detection delay of 1726 samples.
Figure 4.20: The impact on the detection delay rate for varying the window size $n$ and the node split constraint $\tau$ for the Kdq-tree method over a random abrupt change.

In Figure 4.20 the detection delay is low for the small window sizes $n=100$ and $n=200$. The best result in this test, i.e. the smallest detection delay is given by a window size of $n=100$ and $\tau=5$, a detection delay of 19 samples and a standard deviation of 21 samples.

In Figure 4.21 the impact on the estimated Bootstrap threshold, $d$, for varying the window size, $n$, and node split constraint $\tau$ is shown for the Kdq-tree method. For $n=5000$ the result is not shown in the figure because of the scaling: $\tau=250$ gave a Bootstrap estimate $d=0.11$, $\tau=500$ gave a Bootstrap estimate $d=0.016$ and $\tau=750$ gave a Bootstrap estimate $d=0.05$.

Figure 4.21: The impact on the Bootstrap estimate (Bootstrap threshold) for varying the window size $n$ and the node split constraint $\tau$ for the Kdq-tree method over a random abrupt change.

In Figure 4.21 the Bootstrap estimated threshold $d$ gets lower the bigger $\tau$ is for all the window sizes, where $n=100$ is an exception. The mean of the Bootstrap estimate means for the window sizes, $n$, with its lowest node split constraint value, $\tau$, is $d=0.1422$ with a mean of the standard deviation of 0.029. The Bootstrap estimate $d$ seems to depend on the relation between the window size $n$ and the node split constraint $\tau$. 

31
4.2.3 Different size of changes

In the following tests the performance of detection for different sizes of changes is analyzed for the Kdq-tree method. The first test changes the size of the mean $\mu$, the second test changes the size of the covariance matrix $\Sigma$ and the third test changes the size of a linear change $\mu$.

**Abrupt change in $\mu$**

In Table 4.14 the size of $\mu$ in an abrupt change’s impact on the change detection performance is shown for the Kdq-tree method. The change in $\mu$ is varied with $k_\mu$ in (3.2).

**Table 4.14:** Performance of detection when varying the size $\mu$ in an abrupt change for the Kdq-tree method where $\tau=50$ and $n=1000$

<table>
<thead>
<tr>
<th>$k_\mu$</th>
<th>Completed detection</th>
<th>Detection delay, mean</th>
<th>Detection delay, standard deviation</th>
<th>Additional alarm rate</th>
<th>Detected additional alarms, mean</th>
<th>Detected additional alarms, standard deviation</th>
<th>False alarm detection</th>
<th>False alarms, mean</th>
<th>False alarms, standard deviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$-10$</td>
<td>1</td>
<td>496</td>
<td>70</td>
<td>1</td>
<td>1443</td>
<td>798</td>
<td>0.35</td>
<td>1715</td>
<td>488</td>
</tr>
<tr>
<td>$-3$</td>
<td>1</td>
<td>543</td>
<td>71</td>
<td>1</td>
<td>1344</td>
<td>718</td>
<td>0.4</td>
<td>1576</td>
<td>534</td>
</tr>
<tr>
<td>$-1.5$</td>
<td>1</td>
<td>682</td>
<td>90</td>
<td>1</td>
<td>1085</td>
<td>630</td>
<td>0.35</td>
<td>1820</td>
<td>353</td>
</tr>
<tr>
<td>$-1.3$</td>
<td>1</td>
<td>765</td>
<td>99</td>
<td>1</td>
<td>1004</td>
<td>630</td>
<td>0.25</td>
<td>2001</td>
<td>408</td>
</tr>
<tr>
<td>$-1$</td>
<td>1</td>
<td>806</td>
<td>124</td>
<td>1</td>
<td>818</td>
<td>521</td>
<td>0.3</td>
<td>2001</td>
<td>408</td>
</tr>
<tr>
<td>$-0.8$</td>
<td>1</td>
<td>804</td>
<td>141</td>
<td>1</td>
<td>454</td>
<td>541</td>
<td>0.35</td>
<td>1834</td>
<td>517</td>
</tr>
<tr>
<td>$-0.6$</td>
<td>1</td>
<td>930</td>
<td>201</td>
<td>1</td>
<td>676</td>
<td>454</td>
<td>0.35</td>
<td>1949</td>
<td>408</td>
</tr>
<tr>
<td>$0.6$</td>
<td>1</td>
<td>842</td>
<td>134</td>
<td>1</td>
<td>811</td>
<td>521</td>
<td>0.25</td>
<td>2005</td>
<td>408</td>
</tr>
<tr>
<td>$0.8$</td>
<td>0.95</td>
<td>798</td>
<td>134</td>
<td>1</td>
<td>986</td>
<td>454</td>
<td>0.3</td>
<td>1721</td>
<td>517</td>
</tr>
<tr>
<td>$1$</td>
<td>1</td>
<td>725</td>
<td>130</td>
<td>1</td>
<td>1067</td>
<td>521</td>
<td>0.3</td>
<td>2001</td>
<td>408</td>
</tr>
<tr>
<td>$1.3$</td>
<td>1</td>
<td>683</td>
<td>125</td>
<td>1</td>
<td>1332</td>
<td>454</td>
<td>0.35</td>
<td>1370</td>
<td>517</td>
</tr>
<tr>
<td>$1.5$</td>
<td>1</td>
<td>557</td>
<td>125</td>
<td>1</td>
<td>1434</td>
<td>541</td>
<td>0.3</td>
<td>1332</td>
<td>517</td>
</tr>
<tr>
<td>$3$</td>
<td>1</td>
<td>505</td>
<td>80</td>
<td>1</td>
<td>1001</td>
<td>541</td>
<td>0.25</td>
<td>1262</td>
<td>517</td>
</tr>
<tr>
<td>$10$</td>
<td>1</td>
<td>505</td>
<td>80</td>
<td>1</td>
<td>1370</td>
<td>541</td>
<td>0.2</td>
<td>1370</td>
<td>517</td>
</tr>
</tbody>
</table>

In Table 4.14 is can be seen that when $k_\mu= \pm 1$ or over it the method finds the change in all tests. For $k= \pm 0.8$ the method finds the change in 19 test of 20 test. All the sizes of changes give rise to false alarms which is expected with the parameters $n=1000$ and $\tau=50$ according to previous tests, see Figure.

**Figure 4.22** shows an example of a test where a data set with an abrupt change in $\mu$, with $k_\mu=1$, can be seen together with the change detection line that starts at sample 50753 with additional alarms. The detection delay is 753 samples and there are 1001 additional alarms and no false alarms in this test. The Kullback-Leibler distance is shown in Figure 4.23 and in this figure it is shown and the higher KL-measure at sample 50753 and for some samples after is clearly shown. The Bootstrap estimate for this test is $d=0.1389$.

![Figure 4.22](image1.png)  ![Figure 4.23](image2.png)
**Abrupt change in \( \Sigma \)**

In Table 4.15 the size of \( \Sigma \) in an abrupt change’s impact on the change detection performance is shown for the Kdq-tree method. The method parameters are set to \( n=1000 \) and \( \tau=50 \).

**Table 4.15:** Performance of detection when varying the size \( \Sigma \) in an abrupt change for the Kdq-tree method where \( \tau=50 \) and \( n=1000 \)

<table>
<thead>
<tr>
<th>( k )</th>
<th>0.6</th>
<th>0.8</th>
<th>1.0</th>
<th>1.2</th>
<th>1.5</th>
<th>2.0</th>
<th>2.5</th>
<th>5.0</th>
<th>10.0</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completed detection :</td>
<td>0.15</td>
<td>0.25</td>
<td>0.45</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.75</td>
<td>0.95</td>
<td>1.0</td>
</tr>
<tr>
<td>Detection delay, mean:</td>
<td>6943</td>
<td>4487</td>
<td>4320</td>
<td>5332</td>
<td>3757</td>
<td>948</td>
<td>901</td>
<td>815</td>
<td>732</td>
</tr>
<tr>
<td>Detection delay, standard deviation:</td>
<td>5501</td>
<td>6983</td>
<td>6304</td>
<td>9379</td>
<td>845</td>
<td>197</td>
<td>118</td>
<td>116</td>
<td>117</td>
</tr>
<tr>
<td>Additional alarm rate:</td>
<td>0.1</td>
<td>0.25</td>
<td>0.45</td>
<td>0.5</td>
<td>0.5</td>
<td>0.5</td>
<td>0.75</td>
<td>0.95</td>
<td>1.0</td>
</tr>
<tr>
<td>Detected additional alarms, mean:</td>
<td>5</td>
<td>14</td>
<td>78</td>
<td>161</td>
<td>269</td>
<td>332</td>
<td>403</td>
<td>719</td>
<td>939</td>
</tr>
<tr>
<td>Detected additional alarms, standard deviation:</td>
<td>5</td>
<td>11</td>
<td>123</td>
<td>216</td>
<td>270</td>
<td>308</td>
<td>338</td>
<td>284</td>
<td>240</td>
</tr>
<tr>
<td>False alarm detection:</td>
<td>0.35</td>
<td>0.35</td>
<td>0.35</td>
<td>0.35</td>
<td>0.35</td>
<td>0.35</td>
<td>0.35</td>
<td>0.35</td>
<td>0.35</td>
</tr>
<tr>
<td>False alarms, mean:</td>
<td>1720</td>
<td>1344</td>
<td>1202</td>
<td>874</td>
<td>719</td>
<td>1915</td>
<td>1364</td>
<td>1287</td>
<td>1280</td>
</tr>
<tr>
<td>False alarms, standard deviation:</td>
<td>744</td>
<td>804</td>
<td>1001</td>
<td>992</td>
<td>943</td>
<td>959</td>
<td>913</td>
<td>448</td>
<td>474</td>
</tr>
</tbody>
</table>

In Table 4.15 it can be seen that a change in the covariance matrix \( \Sigma \) must be over 10 to be sure of detection. A change of \( k_{\sigma}=5 \) also gives a good value of completed detection (0.95).

**Figure 4.24** shows an example of a test where a data stream with an abrupt change of size 5 in the covariance matrix \( \Sigma \) can be seen together with a change detection line that starts at sample 792 with additional alarms. The detection delay is 792 samples with 1061 additional alarms and no false alarms in this test. In Figure 4.25 the Kullback-Leibler distance is shown and the higher KL-value at sample 50792 and afterwards are clearly shown. The Bootstrap estimate for this test is \( d=0.1281 \).

**Figure 4.24:** A data stream with an abrupt change in the covariance matrix, \( k_{\text{cov}}=5 \), and the change detection for the Kdq-tree method where \( n=1000 \) and \( \tau=50 \)

**Figure 4.25:** The Kullback-Leibler distance for an abrupt change in the covariance matrix, \( k_{\text{cov}}=5 \), for the Kdq-tree method where \( n=1000 \) and \( \tau=50 \)
Linear change

In Table 4.16 the size of $\mu$ in a linear change’s impact on the change detection performance is shown for the Kdq-tree method. The method parameters are set to $n=1000$ and $\tau=50$.

**Table 4.16:** Performance of detection when varying the linear change size, $k$, for the Kdq-tree method where $\tau=50$ and $n=1000$

<table>
<thead>
<tr>
<th>$k$</th>
<th>$-0.1$</th>
<th>$-0.01$</th>
<th>$-0.001$</th>
<th>$-0.0005$</th>
<th>$-0.0001$</th>
<th>$0.0001$</th>
<th>$0.0005$</th>
<th>$0.01$</th>
<th>$0.1$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Completed detection, linear change starts:</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.2</td>
<td>0.3</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Completed detection, linear change ends:</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.7</td>
<td>0</td>
<td>0</td>
<td>0.7</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Detections with additional alarms, linear change starts:</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.15</td>
<td>0.15</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Detections with additional alarms, linear change ends:</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>0.7</td>
<td>0</td>
<td>0</td>
<td>0.7</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Detection delay, linear change starts, mean:</td>
<td>608</td>
<td>662</td>
<td>1158</td>
<td>3021</td>
<td>19183</td>
<td>21895</td>
<td>2222</td>
<td>1195</td>
<td>635</td>
</tr>
<tr>
<td>Detection delay, linear change starts, standard deviation:</td>
<td>49</td>
<td>54</td>
<td>141</td>
<td>338</td>
<td>15203</td>
<td>14637</td>
<td>214</td>
<td>271</td>
<td>58</td>
</tr>
<tr>
<td>Detected additional alarms, linear change starts: mean:</td>
<td>49479</td>
<td>49333</td>
<td>47660</td>
<td>20790</td>
<td>12</td>
<td>90</td>
<td>23550</td>
<td>47938</td>
<td>49362</td>
</tr>
<tr>
<td>Detected additional alarms, linear change starts, standard deviation:</td>
<td>43</td>
<td>49</td>
<td>3599</td>
<td>17611</td>
<td>43</td>
<td>386</td>
<td>18442</td>
<td>2936</td>
<td>57</td>
</tr>
<tr>
<td>Detection delay, linear change ends, mean:</td>
<td>50</td>
<td>54</td>
<td>141</td>
<td>338</td>
<td>15203</td>
<td>14637</td>
<td>214</td>
<td>271</td>
<td>58</td>
</tr>
<tr>
<td>Detection delay, linear change ends, standard deviation:</td>
<td>49</td>
<td>54</td>
<td>141</td>
<td>338</td>
<td>15203</td>
<td>14637</td>
<td>214</td>
<td>271</td>
<td>58</td>
</tr>
<tr>
<td>Detected additional alarms, linear change ends: mean:</td>
<td>1919</td>
<td>1756</td>
<td>1122</td>
<td>491</td>
<td>-</td>
<td>26</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Detected additional alarms, linear change ends, standard deviation:</td>
<td>10</td>
<td>41</td>
<td>265</td>
<td>451</td>
<td>-</td>
<td>26</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>False alarm detection:</td>
<td>0.15</td>
<td>0.25</td>
<td>0.25</td>
<td>0.3</td>
<td>0.25</td>
<td>0.25</td>
<td>0.25</td>
<td>0.15</td>
<td></td>
</tr>
<tr>
<td>False alarms, mean:</td>
<td>1680</td>
<td>1755</td>
<td>1858</td>
<td>1608</td>
<td>2335</td>
<td>1875</td>
<td>1760</td>
<td>2001</td>
<td>2001</td>
</tr>
<tr>
<td>False alarms, standard deviation:</td>
<td>786</td>
<td>696</td>
<td>378</td>
<td>880</td>
<td>1</td>
<td>115</td>
<td>376</td>
<td>1</td>
<td>1</td>
</tr>
</tbody>
</table>

According to Table 4.16 the linear change in $\mu$ is detected, when it starts, if it is bigger than $k=0.0005$, otherwise the probability for not detecting the change is low (only 0.3 for $k=0.0001$). The mean of the change delay for $k=0.0005$ is 2222 samples with a standard deviation of 2224 samples. The detection when the linear change ends is 0.75. There are 0.25 false alarms but it seems to apply for all the size of linear change in $\mu$, because the mean of them are 0.24 with a standard deviation of 0.05.

Figure 4.26 shows an example of a test where a data stream with a linear change in $\mu$ of $k=0.0005$ can be seen together with lines that indicate a change detection. The first change is detected after the linear change starts, a delay of 1288 samples, with 46603 additional alarms and a detection when the linear change ends with a detection delay of 11 samples and 605 additional alarms. No false alarms are detected in this test. The Bootstrap threshold for this test is estimated to $d=0.1415$. The window size $n=1000$ and the node split constraint is $\tau=50$. The Kullback-Leibler distance can be seen in Figure 4.27 and gives a high KL-measure during the whole linear change.
Figure 4.26: A data stream with a linear change in the mean vector, \( k = 0.0005 \), and the change detection for the Kdq-tree method where \( n = 1000 \) and \( \tau = 50 \).

Figure 4.27: The Kullback-Leibler distance for a linear change in the mean vector, \( k = 0.0005 \), for the Kdq-tree method where \( n = 1000 \) and \( \tau = 50 \).
Chapter 5

Discussion

To make the results significant I chose to build them on 20 separate tests. The reason that I used 20 tests was that it gives significant result and sufficiently of time for the test runs.

The results showed in chapter 4.1.1 are discussed below.

5.1 The Overlapping method

5.1.1 Performance for specific method-parameters

In the first reported tests the parameters T (decay rate) and N (window size) are tested for an abrupt change for the "the Overlapping method".

The results when the decay rate $T$ is tested with a window size of $N=1000$ is shown in Table 4.2. There are neither false alarms nor detected additional alarms in the tests, except in one test where $T=0.5$ there are additional alarms. The detection delay is between 36 samples and 101 samples when $T$ is varied. The lowest detection delay is for $T=0.7$. When $T$ increases the model uses more data to estimate the a priori. It should have the best estimation for the a priori for $T=0.9$ but then the overlapping of the models are small so the detection delay should decrease. When $T$ decreases the estimate of the a priori will be less significant and the overlapping part of the models will be almost complete. However, in the results the decay rate does not make a big impact on neither the detection rate nor the detection delay. For a window size of $N=1000$ all the decay rates could be appropriate to use, cause the variation between the detection delays are pretty small, a mean of 59 samples and a standard deviation of 20 samples. The decay rate does not seem to have such a big impact on the result but you could say that a lower value of the decay rate $T$ gives a less sensitive model and a higher value of the decay rate $T$ makes the model more sensitive to give an indication for a change, true or false.

I chose $T=0.5$ as the most appropriate decay rate and the reason is that it gives a robust model according to the tests in this report, i.e. it gives a good estimate of the a priori and the models overlap by half. A decay rate below $T=0.5$ could also be chosen but it would make the method more sensitive to changes and therefore might cause false alarms and additional alarms (see for $T=0.4$).

The main reason that there are no additional alarms detected in these tests is that there is a convergence threshold, $\eta_{conv}$, set. This threshold ensures that no change is detected after a detection unless the Kullback-Leibler measure have been below $\tau_{low}$ again.

In Table 4.3 the results of a changing window size $N$, where $T=0.5$, is shown. It indicates that a small window size gives a small detection delay but it also makes the method more sensitive and could cause false alarms or additional alarms. For $N=100$ the detection delay is only 6.3 samples with a low standard deviation of 7.6 samples. For $N=5000$ the detection delay is 235.7 samples with a high standard deviation of 534.6 samples. The smaller a window size is, the fewer samples will be included in the Bayesian estimation of the parameters, and it makes the estimation less accurate and more
sensitive to changes. It can be seen in the results that there is a false alarm detected in two tests for \( N=100 \). The bigger the window size is the more samples from the previous model are already included in the model because of the decay rate of \( T=0.5 \), i.e. the model already consist of 2500 samples \( (T^*N \text{ samples}) \) for a window of size \( N=5000 \). This makes it harder for a new sample to affect the Bayesian estimation of the model parameters and therefore it takes more samples before a detection is found, i.e. the method gets less sensitive to changes and the detection delay increase.

For the window size \( N=2000 \) the result shows a low detection delay in both mean and standard deviation. According to the result for the other window sizes the detection delay mean should be higher for \( N=2000 \), if you compare to \( N=1300 \) which has a detection delay mean of 136 samples. The detection delay mean seems to stand in proportion to the window size. The detection delay should increase with an increasing value of the window size. The deviant result for \( N=2000 \) may be due to lack of tests that contributes to significance in the out coming result. The synthetic distribution which makes randomly abrupt changes for every new test might also give rise to 'abnormal' results. Some tests gets bigger abrupt changes to detect and some gets smaller changes to detect. It is much harder to detect small changes and this might impact the result. This is the reason we make the test 'Different size of changes'.

In this two tests the a priori input has the weight, \( \alpha \), of size 1 for all window sizes. It would be interesting to test to give the a priori a weight that stands in proportion to the window size, \( N \). For example, the proportion could be 0.01 so if \( N=100 \) it would give \( \alpha=1 \) and for \( N=1000 \) it would give \( \alpha=10 \). The weight affects the memory of the model and a higher value of \( \alpha \) would give the historical data more importance.

In Figure 4.1 a data stream with an abrupt change in both mean and variance is shown for a model of window size \( N=300 \) and \( T=0.5 \). These parameters are considered to be the best combination for an abrupt change based on the results in this report. There is a detection with a delay of 24 samples. Figure 5.1.1 shows the Kullback-Leibler distance. It is hard to see in the picture but the Kullback-Leibler distance gives some additional signals for change which the method removes because of the convergence threshold, \( \eta_{\text{conv}} \).

The test have been made for the detection threshold \( \eta=0.6 \). The test result would have been different if the detection threshold was set higher or lower than 0.6. As can be seen in Figure the KL-measure is significantly higher than 0.6 for a change and therefore the threshold could have been set higher. The higher the threshold is the less sensitive is the model for changes. If the detection threshold is set too high there is a high risk that the model does not detect smaller changes that only gives a small increase in the KL-measure. To be able to set an appropriate value of the threshold one need to know the KL-measures for the distribution with no changes in advance, or make tests to get to know it.

Linear change

In the next tests in chapter 4.1.1 the parameters \( T \) (decay rate) and \( N \) (window size) are tested for a linear change for the 'the Overlapping method'. In these two tests there are two big changes, one when the linear change starts and one when it ends. There are also continuous changes during the linear change. The additional alarms could be considered as not that important if they arise during the linear, because during the linear change there are changes throughout and by this method it should be detections.

According to Table 4.4 the decay rate should be set to \( T=0.2 \) for a window size \( N=1000 \). This is the decay rate that gives the highest percentage of completed detection, 0.8, when the linear change starts but only 0.15 when it end. No one of the other decay rates have a good rate of completed detection neither when the linear change starts nor when it ends. By choosing a low decay rate of \( T=0.2 \) it increases the memory in the model because the overlapping parts of the models increases. It makes the model more sensitive to linear changes compared to when the decay rate \( T \) is high. As can be seen in the results a low value of \( T \) increases the completed detection rate. It also detects some of the changes when the linear change ends (0.35).
In Table 4.5 the results when the window sizes \( N \) are varied for a decay rate \( T=0.2 \) over a linear change are shown. The smallest window sizes detect almost all the changes but there are false alarms in all tests for \( N=100 \) has a mean around 44 false alarms and in 0.45 of the test (for \( N=200 \)) has a mean around 1 false alarm. The smaller the window size is the more sensitive the method is and the more changes will be detected, false or not. It is better if the window size is pretty big, as can be seen for \( N=1300, 2000 \) and 5000, where all of these window sizes detect the change in 0.95 or more of the tests. In 4 of the tests for \( N=1300 \) it detects when the linear change ends. The difference between the bigger window sizes is the detection delay and additional alarms. The additional alarms after the linear change starts is inevitable for a linear change, because during the linear change there are throughout a constant change, so they are ignored. The detection delay is important and for \( N=1300 \) is 'only' 158.8 samples with a standard deviation of 96.9 samples. Therefore, in the future tests, \( N=1300 \) is chosen as the ultimate parameter when \( T=0.2 \). When the window size \( N \) is somewhere between small and big, \( N=300, 500 \) and 700, it is not good. The completed detection when the linear change ends is low for all these window sizes. The explanation could be that when the linear change starts the window size must be big enough otherwise the samples in the model will "cancel each other out" and the Bayesian estimation of the parameters will not change significant and give a change detection. There is a significant relationship between the size of linear change, i.e. how fast the linear change is, and the window size \( N \). If the linear change is very slow, i.e. small, it is better to choose a big window size \( N \) to make the model more sensitive to the small changes.

There is a high standard deviation for the detection delay for some of the window sizes. The reason could be that the linear change that is added to the synthetic distributions are randomly chosen from a Gaussian distribution with mean \( \mu=0 \) and \( \sigma^2=0.001 \). Some of the linear changes might be very small and therefore hard to detect and the detection delay mean and then also the standard deviation for the detection delay will decrease for the tests. The tests in 4.1.3 are made to test the size of change.

In Figure 4.3 a data stream with a linear change in both mean and variance is shown for a model of window size \( N=1300 \) and \( T=0.2 \). The detection delay is 1267 samples and all the additional alarms in the linear change are shown. It is hard to see in the picture but the Kullback-Leibler distance gives some additional signals for change that the method removes because of the convergence threshold, \( \eta_{low} \).

In Chapter 4.1.2 the dependence between the window size \( N \) and the decay rate \( T \) are analyzed for the false alarm rate and the detection delay mean. These two performance measures are important for this method where all the changes are detected. The tests are made over a random abrupt change with parameters \( N=300 \) and \( T=0.5 \). Figure 4.5 shows the impact of the two parameters on the false alarm rate. It can be seen that smaller window sizes are more affected of the decay rate \( T \) than bigger window sizes when it comes to false alarm rate.

Figure 4.6 shows the impact of the two parameters on the detection delay. It is clearly shown that a small window size gives a shorter detection delay mean than a big window size. The decay rate \( T \) does not seem to have a significant impact on the detection delay for the different window sizes.

### 5.1.2 Different size of change

#### Abrupt change

This test was made because in test 1 the size of change varies randomly and some of them might be really small and therefore not detected.

In Table 4.6 the results for different size of abrupt changes in \( \mu \) is shown, where \( k_\mu \) is varied. The change in \( k_\mu \) has to be pretty big. It has to be around \( \pm 3 \) to ensure detection for a window size \( N=300 \) and a decay rate \( T=0.5 \). There are no false alarms or additional alarms and the probability to detect a positive or a negative change seems to be equal. In 1 of the 20 tests the detection is not detected for \( k=\pm 3 \) and the detection delay is 33 samples with a standard deviation of 24 samples for
the tests where there is a detection. When \( k=10 \) the detection rate is 1 and the detection delay is really small, only 2 samples with a standard deviation of 2 samples.

In Figure 4.7 a data stream with an abrupt changes in the mean, \( \mu \), with \( k_{\mu}=3 \) is shown, for a model of window size \( N=300 \) and \( T=0.5 \). It is hard to see in the picture but the Kullback-Leibler distance gives some additional signals for change which the method removes because of the convergence threshold, \( \eta_{\text{conv}} \), see Figure 4.8.

In Table 4.7 an abrupt change in \( \Sigma \) is shown, \( k_{\Sigma} \) is varied, for a model with window size \( N=300 \) and a decay rate \( T=0.5 \). It seems to be harder to detect a change in the covariance matrix \( \Sigma \) and one can not be sure of a detection if the change is below \( k_{\mu}=10 \). If a detection is made then the detection delay is pretty much equal for all sizes of change. The threshold \( \eta \) have a big impact on the completed detection rate. The results would have been different if the threshold was set lower than 0.6 because the model would be more sensitive for changes and smaller changes would probably be detected. If the threshold is set too low the model will be too sensitive and the false alarm rate would increase.

In Figure 4.9 a data stream with an abrupt change in the covariance \( \Sigma \) of size 10 is shown, for a model of window size \( N=300 \) and \( T=0.5 \). The mean of the detection delay is 32 samples and there are no false alarms or additional alarms as expected from the results in Table 4.7. The Kullback-Leibler distance is shown in Figure 4.10.

### Linear change

In Table 4.8 the results for a linear change is shown, where \( k \) varies in \( \mu \), for a model with window size \( N=1300 \) and a decay rate \( T=0.2 \). The change in \( k \) must be more than 0.001 to be sure of detection. There does not seem to be any difference if the change is negative. The smaller the linear change is the bigger is the detection delay mean because it takes a longer time for the model to find the change. The results would have been different if the threshold \( \eta \) was set different.

In Figure 4.11 a data stream with a linear change in the covariance \( \Sigma \) of size 10 is shown, for a model of window size \( N=300 \) and \( T=0.5 \). The mean of the detection delay is 1545 samples. The Kullback-Leibler distance is shown in Figure 4.12.

### 5.1.3 Change of total number of added changes

In Table 4.9 the result when there are different numbers of added abrupt changes of size \( k_{\mu}=3 \), where the window size is \( N=300 \) and the decay rate \( T=0.5 \). When there are 199 changes the method detects almost all of them and it is because there is a change every 502 sample so the models have time to adapt to the change. When the total number of changes is above 249 then there is a change every 401 sample and it is more than the size of the window \( N \), which is \( N=300 \). The model miss some changes while it adapts to the changes it finds.

In Figure 4.13 and ?? the changing distribution with 19 respectively 199 changes can be seen. Both pictures shows that all the changed changes are detected and there are no false alarms nor additional alarms. In Figure 4.14 and ?? the Kullback-Leibler distance for respectively added amount of changes, 19 and 199, are shown. These pictures shows the increase in KL distance after a change when the model adapt to the new stable state. These additional alarms are not detected in the model because of the convergence threshold, \( \eta_{\text{conv}} \).

### 5.2 The Kdq-tree Method

#### 5.2.1 Performance for specific method-parameters

In Table 4.2.3 the result are shown for an abrupt change when \( n=1000 \) and \( \tau \) is varied for the Kdq-tree method. For \( \tau=50 \) the false alarm detection is pretty low, 0.2, and the detection rate is 1. The change
delay for this parameter is 557 samples and the standard deviation is 69 samples. There are also additional alarms in all of the 20 tests, with a mean in additional alarms of 1224 samples and a standard deviation of 123 samples which is expected when the model is adapting after a change detection. There is no test made for \( \tau = 10 \) because it was time-inefficient even if an extra server was used. The reason is that when the window size \( n \) is big in relation to the node split constraint \( \tau \) the Kdq-tree has to make many node splits before the constraint holds and it takes extra time for every new node in the tree. When the node split constraint is \( \tau = 25 \) the sensitivity in the model gets higher and therefore the detection delay increases but at the same time the false alarm rate increases.

The change detection method has many parameters that should be set. There is the window size \( n \), node split constraint (maximum points in a cell) \( \tau \), minimum cell side \( \delta \), chain sample size \( k \), significance level \( \alpha \) and Bootstrap samples \( B \). When the test for this method was made one could notice that many of the parameters affect each other. The window size is really important but you have to think of the node split constraint, when the node split should stop, \( \tau \). If \( \tau \) is too big and the size of the window, \( n \), is too small then there will be no detection cause the Kdq-tree can not split the sample in significant parts. This can be seen in the Table when \( \tau = 125 \) where the detection rate is only 0.85. The node split constraint \( \tau \) and the window size \( n \) have really big impact on each other.

The sample size \( k \) was always set to \( 0.4 \times k \) in the tests but if it was not it would be the parameter that had biggest impact on \( \tau \). The node split constraint \( \tau \) is also a big part of the Bootstrap. To get a good value for the Bootstrap there is the same problem as mentioned above. If the same window, \( n \), and sample size, \( k \), and \( \tau \) are used as in the rest of the method one has to think of the relation between these parameters, otherwise the threshold \( d \) from the Bootstrap will be really small and no detection will be made. The sample size could be set to some other value but it is used to make the method more time efficient so a pretty low value in relation to the window size would be most suitable.

If the window size is big and the \( \tau \) is small then the method will not be time efficient, which was what happened for \( \tau = 10 \).

In Table 4.11 the results when the window size \( n \) varies with a node split constraint \( \tau \) over an abrupt change is shown. In this test the relation between the two parameters \( n \) and \( \tau \) had a big impact on the time-efficiency. It is the same as in the test where \( n = 1000 \) and \( \tau = 10 \) but in this test it is a big window size \( n \) that makes the problem when \( \delta = 50 \). For \( n = 2000 \) and \( n = 5000 \) it take extremely long time to do the experiments and therefore no tests are made for \( n = 5000 \). The window size \( n = 1000 \) is chosen as the best parameter when \( \tau = 50 \), because it is less sensitive for changes and therefore has a low false alarm rate, 0.2. The detection delay is pretty high but it depends on the window size \( n = 1000 \), because the bigger window size the bigger detection delay will be, as can be seen in the results. The window size \( n = 100 \) and \( n = 200 \) did not give any change detection. The first reason is that when the Bootstrap estimate of the threshold, \( d \), is made it gets very small because the sample \( k \) will not split up in significant partitions. The second reason is that if the threshold was made in another way the samples \( k \) would still not be able to split up in significant parts to detect a change. When the window size increases the false alarm rate decrease because of this. The Bootstrap estimate of the threshold will be more significant and the model will be able to compare the windows to each other in a fair way.

In Figure 4.16 a data set with an abrupt change in both \( \mu \) and \( \Sigma \) is shown. You can clearly see three detection times, where one is at the correct change and two are false detection. The first two are the false alarms and each of them includes more than one false alarm. All together there are 95 false alarms. The third detection detects the real change with a delay of 504 samples and there are 1457 additional alarms afterward that are clearly shown in the picture. In Figure 4.16 the Kullback-Leibler distance is shown. In this figure the KL distance is shown and you can see the increase in the KL distance when the abrupt change is detected, the so called additional alarms. The two false alarms are also seen in the figure as two smaller increases of the KL distance. They occur because of the low threshold of \( d = 0.1003 \) that is estimated from the Bootstrap. The difference in size between the KL distance for the real change and the false alarms is very big. Therefore a better estimate of the threshold \( d \) would omit the small increases in the KL-distance that gives rise to false alarms.
In Table 4.12 the results when the window size \( n=1000 \) and the node split constraint \( \tau \) varies for a linear change in \( \mu \) is shown for the Kdq-tree method. There is the same problem here as in the test above. When \( \tau \) is too small the test will take long time. Therefore there are no results for \( \tau=10 \). The best result when \( n=1000 \) is when \( \tau=50 \). This detects most of the changes when the linear change starts, 0.9, and it has a pretty low false alarm rate, 0.35. It has additional alarms after the linear change starts but it would be expected when you look at the results from an abrupt change, which also has many additional alarms. The detection delay is very high though, a mean of 2090 samples and a standard deviation of 1911 samples. Probably the detection delay and the false alarm rate stands in proportion to each other here because of the threshold \( d \) made of the Bootstrap. If the threshold is set manually or with other parameters in the Bootstrap it will probably improve one of these detection measures at the expense of the other performance measure.

In Table 4.13 the result when the decay rate \( T=50 \) and the window size \( N \) varies for a linear change in \( \mu \) is shown for the Kdq-tree method. As for the abrupt change the relation between the window size and node split constraint \( \tau \) is shown. It is the same problem as for an abrupt change, that for \( n=2000 \) and \( n=5000 \) it takes long time to do the experiments and therefore no tests are made for \( n=5000 \). It can clearly be seen that the false alarm rate and the change delay is dependent on the window size \( n \) for \( \tau=50 \). The false alarm rate decreases as the window size increases but the window size seems to have lower impact when it comes to the detection delay. However, for parameters with high false alarm rate the detection delay for the real changes is not important.

In Figure 4.17 the data set with a linear change can be seen with the detection lines. As expected there are many additional alarms. One of the three dimensions has a bigger slope than the others. Maybe it is because of that the detection delay, 957 samples, is so good compared to the result in the table where the mean of detection delay was 2090 with a standard deviation 1911. The change when the linear change ends is also detected with a good detection delay of 1 sample, compared to the table where the mean of detection delay was 21 samples with a standard deviation of 61 samples.

In Chapter 4.2.2 the relation between the window size and the node split constraint is shown and its impact on the performance measure false alarm rate and detection delay mean. Also the impact on the Bootstrap estimated threshold is shown. All the random abrupt changes are detected. The parameters that are used in the model is \( n=1000 \) and \( \tau=50 \).

In Figure 4.19 the impact from varying the window size and \( \tau \) on the false alarm rate is shown. Small window sizes are not good to use in this method. As can be seen in the figure they all have a high false alarm for all respectively node split constraints. In the results in this report the window size \( n=1000 \) has the best false alarm rate of all the window sizes when \( \tau=50 \). Here the window sizes are tested but it could have been better to test the samples size \( k \) instead of the window size \( n \). It could be that a node split constraint \( \tau \) that is around 12% of the sample size \( k \) is the best. It is not that strange because in this method it is the samples of size \( k \) that is partitioned in the Kdq-tree and not the data items in the window of size \( n \).

In Figure 4.20 the relation between the window size \( n \) and the detection delay is obvious. The window sizes that give the best detection delay are the small ones. But according to what was mentioned above they also give the highest false alarm rate.

When you decide your parameters you have to decide what is most important to you. A low false alarm rate or a small detection delay, and thereafter decide your parameter set. I decided to choose the window size that gives the lowest false alarm rate, \( n=1000 \), together with \( \tau=50 \). The Kdq-tree method should use a big window size and a node split constraint \( \tau \) that stands in proportion to the window size, for example \( \tau=0.1k \) where \( k=0.4n \).

In Figure 4.21 the impact on the estimated Bootstrap threshold \( d \) when varying the window size \( n \) and node split constraint \( \tau \) is shown over an abrupt change. The highest Bootstrap estimates are given when the node split constraint is lowest and the lowest Bootstrap estimates are given when the node split constraint is highest.
5.2.2 Different size of change

Abrupt change

In Table 4.14 the size of an abrupt change in $\mu$ for the Kdq-tree is shown with the parameters $n=1000$ and $\tau=50$. This method has a good detection rate for all the size of changes and there are also additional alarms in all tests that found the change, as expected. You can see a small increase in the detection delay for the bigger changes, both positive and negative. The false alarm rate does not change significantly between the different sizes of changes.

In Figure 4.22 shows a data stream with an abrupt change of size $k=1$ and the line that indicates a change detection. The line that indicates change is thick because of the additional alarms that detects right after the real change. In Figure 4.23 the Kullback-Leibler distance is shown. The KL-distance has increased and reaches a top somewhere after the real change.

In Table 4.15 is the results from different changes in $\Sigma$ for an abrupt change. The detection rate increases as the changes gets bigger which is obvious. The detection delay decreases for bigger changes. The method is not as good in finding changes in the covariance $\Sigma$ as in the mean $\mu$.

In Figure 4.24 an abrupt change in $\Sigma$ of size $k=5$ is shown. It detects the change with a detection delay of 792 samples and 1061 additional alarms. The thicker line that shows a change detection indicate that there are some additional alarms detected as well.

Linear change

In Table 4.16 different sizes of $k$ in a linear change is shown. The method detects all changes that are bigger than $k=0.0005$. The false detection rate for all the different values of $k$ does not change significant, i.e. the standard deviation is low, but the detection delay when the linear change starts and ends does. There is a big gap from detecting all changes for $k=0.0005$ to only around 30% of the changes for $k=0.0001$. The results are affected of the Bootrap estimated threshold $d$. If the threshold was set in another way or manually the results would have been different. The values of the different size of linear change, $k$, was tested for the Overlapping method. Therefore the same values were tested here, but as can be seen in the results smaller values than $k=0.0005$ could have been tested.

In Figure 4.26 a data set with a linear change of size $k=0.0005$ is shown with the lines of detection. There are 46603 additional alarms after the linear change starts, which means that there are almost a change for every sample. The linear change reaches over 50000 samples. Therefore the detection lines covers almost the period for the linear change. In Figure 4.27 the Kullback-Leibler distance is shown.

5.3 Comparison between the two methods

For both methods the false alarm rate was considered to be the most important performance measure and I choosed the parameters for both methods regarding to it.

The Overlapping method uses a convergence threshold, $\eta_{\text{conv}}$, that decreases or removes the additional alarms completely. This is really good compared to The Kdq-tree method where they do not use this kind of convergence threshold. The Kdq-tree method could be improved by using a convergence threshold.

The Kdq method uses a significant method, the Bootstrap method, to estimate a detection threshold, $d$, which the Kullback-Leibler distance is compared to, to detect a change. It is a good way of finding a significant threshold but if the window size $n$ is too small and the node split constraint $\tau$ is too big,
there will be no significant threshold. The Bootstrap is also time-consuming, especially if you want many Bootstrap samples, \( B > 10000 \), to make \( d \) more significant. One could set the threshold manually just as in the Overlapping method. However, it can be seen in Figure 4.4, the threshold, \( \tau_{\text{div}} \), could be set too high and therefore not detect the small KL-distance increase in the figure when the linear change ends, that could indicate a change.

The Kdq-tree method is much better than the Overlapping method of finding small sizes of changes. For abrupt changes in mean \( \mu \) the Kdq-tree method found all the changes for \( k = \pm 1 \) and the Overlapping method found almost all the changes, 95\%, for \( k = 3 \). For an abrupt change in \( \Sigma \) the Kdq-tree method found almost all the changes, 0.95 for \( k = 5 \). The Overlapping method had the same rate for \( k = 10 \). The Kdq-tree method found all the changes when the linear change was \( k = 0.0005 \) and the detection rate for the Overlapping method for the same value was only 35\%. The Overlapping method had to detect a linear change of \( k = 0.001 \) to have a detection rate of 100\%. The main reason for the better results for the Kdq-tree method has to do with the threshold. The Kdq-tree method estimate the threshold \( d \) with the Bootstrap method, which takes into account the dataset. In the Overlapping method one has to have an idea about the Kullback-Leibler distance for the dataset without any change, to set the threshold \( \eta_{\text{div}} \). Though the Kdq-tree method has a higher change detection rate for small changes in abrupt and linear changes the consequences are that the Kdq-tree method have much higher false alarm rate in all the tests made. The Overlapping method only has false alarms detected for really small window sizes \( N \).

Overall the Kdq-tree method has much more parameters to predetermine and they also depend on each other. The sample size, \( k \), could be good to set as \( k = 0.4n \), just as in the tests. You could also set \( k \) to something else or another value that relates to \( n \). Even if you set \( k \) to a percentage of \( n \) or not you have to have in mind that it is these samples that are split up in the Kdq-tree. Therefore you need a small enough node split constraint \( \tau \) so you get an appropriate splitting of the samples in the Kdq-tree. If \( \tau \) is too big there will be no partitioning of the data and therefore no significant value of the Bootstrap estimated threshold \( d \). It could be a good choice to set \( \tau \) as a percentage of \( k \), for example \( \tau = 0.1k \), to be sure of an appropriate partitioning of the samples \( k \).

As could be seen in the test the parameters for the Kdq-tree method was much more dependent on each other and on the results. In the Overlapping method the window size \( N \) had the biggest impact on the result.

An important goal for both methods is to find the change and not any false alarms or additional alarms (for an abrupt change). The Overlapping method was best for both of these criterias and the convergence threshold, \( \eta_{\text{conv}} \), had a big part of increasing the additional alarms after an abrupt change. As mentioned above the Kdq-tree method could be approved by adding a convergence threshold. It would also be interesting to do test on the Kdq-tree method where the threshold is set manually.

The Overlapping method had the overall smallest detection delay in the test for abrupt changes compared to the Kdq-tree method. It is not really comparably because the window sizes for the both methods were different. The Overlapping method was tested for a window size \( N = 300 \) for abrupt changes and the Kdq-tree method had a window size \( n = 1000 \). The bigger the window size is the bigger will the detection delay become, nomatter which change detection method is used.

Many of the test in The Overlapping method for linear changes does not detect when the linear change ends. The Kdq-tree method was much better on it. However, in the Overlapping method you would know that the linear change ends because the additional alarms stop. It is not the same for the Kdq-tree method because it detects more false alarms and also more additional alarms in general. Even if it detects 0.85 of the change when the linear change ends it also detects additional alarms afterward and therefore you are uncertain for when the linear change ends.
Chapter 6

Conclusions

6.1 The Overlapping Method

From the results in this master’s thesis I draw the following conclusions for the Overlapping method: The Overlapping method is a change detection method that is pretty easy to understand and implement for a new user, in my opinion. The method is also easy to use because it only has a few parameters (4 parameters) that should be predetermined in advance and which affects the performance of the detection results. The Overlapping method only detects false alarms in a few tests and it uses a convergence threshold to prevent additional alarms, which seems to be very efficient according to the test results. The window size is the parameter which has the biggest impact on the results. A small window size gives a very small detection delay, but at the same time the risk of false alarms increases because of a poor estimate of the distributions in the windows. The method is not very good in finding small changes, but it depends largely on how the detection threshold is set (in this report 0.6). To be able to use the Overlapping method for change detection one has to know the underlying distribution of the data stream which is not always known. The detection threshold is also easier to set to a significant value if one have information about the underlying distribution of the data stream.

6.2 The Kdq-tree Method

From the results in this master’s thesis I draw the following conclusions for the Kdq-tree method: The Kdq-tree method is a very useful method because one does not have to know the underlying distribution of the data stream. The Bootstrap method is used to estimate the detection threshold, \(d\), which is compared to the calculated KL-distance to decide if there is a change or not. By using the Bootstrap method the detection threshold is adjusted to the underlying distribution of the data stream. Therefore, one does not have to know anything about the underlying distribution or make any predetermined suggestion about the detection threshold. The Kdq-tree method is very good in finding small changes in the data stream, but at the same time it detects false alarms in many of the tests. The reason is the methods sensitivity. The Kdq-tree method contains several parameters (6 parameters) that should be predetermined, which can be difficult for an untrained user as the parameters are largely dependent on each other. The more parameters that have to be predetermined in the method the more must be considered and taken into account to get the best results. The method is pretty complicated to implement for a new user, in my opinion.
Further work

Test could be made on the Kdq-tree method when the threshold is set manually. It would also be interesting to add a convergence threshold to the Kdq-tree method. For the Overlapping method test could be made when the threshold is set by using the Bootstrap method (or another method). Test should also be made over real world data streams. Further work could also be to study a third change detection method. A proposal is the Kalman filter method mentioned in Appendix A. It has a different approach from the two other methods and it is interesting and could be useful.
Appendix A

The Kalman filter method

This method is called the 'Kalman filter method'\cite{30}. The 'Kalman filter method' has a non-parametric approach that is different from the two methods described and evaluated in this report. This method also uses window to store data from the data set, the windows are cumulative but to compare the data sets in the windows it uses a Kalman filter and a regression model, to estimate a residual, and finally to detect a change it uses a CUSUM. A full description of the method follows below.

A.0.1 Main approach

This method is a modular detection system for regression problems and is evaluated in \cite{30}. The following description is taken from this paper.

The components in the detection system is: a learning algorithm, a Kalman filter and a CUSUM. The framework of the method can be seen in Figure A.1.

\begin{figure}[h]
\centering
\includegraphics[width=\textwidth]{framework.png}
\caption{The framework of the detection system}
\end{figure}

Regression analysis is used to obtain a regression model that serves as a learning algorithm\footnote{Learning algorithms - allowing computers to learn via inductive inference based on observation data, empirical data such as from databases, which represent incomplete information about statistical phenomenon} as the first component in the detection system, see chapter A.0.2. The regression model is trained upon data in a window W of size k, the so called \textit{training data}. The window, W, contains the first k items in the data set of n dimensions.

\begin{equation}
W = \begin{pmatrix}
x_{1,1} & x_{1,2} & \cdots & x_{1,n} \\
x_{2,1} & x_{2,2} & \cdots & x_{2,n} \\
\vdots & \vdots & \ddots & \vdots \\
x_{k,1} & x_{k,2} & \cdots & x_{k,n}
\end{pmatrix}
\end{equation}
We choose the first dimension to be the dependent variable, $y_i = x_{i,1}$ (desired value) and the rest will be the independent variable, $x_i = [x_{i,2} \ x_{i,3} \ldots \ x_{i,n}]$ (input object). The regression model is then constructed. Thereafter, at each iteration, the regression model receives new $x_i$ and predicts, $\hat{y}_i$. The model forecast value $\hat{y}_i$ and the input from the data set $y_i$ is compared and a residual $r_i$ is calculated:

$$r_i = |y_i - \hat{y}_i| \quad (A.1)$$

The Kalman filter, the system's second component, see A.0.3, makes an estimate of all the dimensions in the incoming data. We call the incoming vector $y_i$ and the Kalman estimate vector $\hat{y}_i^{kalman}$. The residual vector $\hat{r}_i$ is calculated:

$$\hat{r}_i = |y_i - \hat{y}_i^{kalman}| \quad (A.2)$$

To compute a residual for the dispersion, $rd_i$, the Kalman filter error estimate of the previous actual model, $\hat{r}_{i-1}$, is used:

$$rd_i = |r_i - \hat{r}_{i-1}| \quad (A.3)$$

The initial estimate of $\hat{r}_{i-1}$, $\hat{r}_0$, is a vector of zeros, of size $d$.

The residual for the dispersion of the Kalman estimate, $rd_i$, is:

$$rd_i = |r_i - \hat{r}_i| \quad (A.4)$$

The regression learning algorithm will be improved with the Kalman filter and the main assumption is that a change is reflected in the distribution of the data set, leading to an increase of the error of the actual regression model. The pair $(r_i, \hat{r}_i)$ is transmitted to the mean CUSUM and the pair $(rd_i, r\hat{d}_i)$ is transmitted to the dispersion CUSUM. See chapter A.0.4 for details of the CUSUM. The values in respectively pair are compared with each other. A change occurs if there is a significant difference between both pairs or a significant difference between consecutive residuals. If the change is an increase in the error mean or in the dispersion error the current learning regression model is erased and a new model is built upon the new data items. If the change is a decrease in the error mean or in the dispersion error then the system gives an order to the Kalman filter to weight the new residuals heavier, thus the filter can follow the mean error and dispersion error faster.

Specific parameters for the Kalman filter method are seen in table A.1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Window size</td>
<td>$k$</td>
</tr>
<tr>
<td>Proportions constant</td>
<td>$A$</td>
</tr>
<tr>
<td>Proportions constant</td>
<td>$H$</td>
</tr>
<tr>
<td>Process noise covariance</td>
<td>$Q$</td>
</tr>
<tr>
<td>Measurement noise covariance</td>
<td>$R$</td>
</tr>
<tr>
<td>Initial estimate of previous $x$</td>
<td>$x_0$</td>
</tr>
<tr>
<td>Initial estimate of previous error covariance</td>
<td>$P_0$</td>
</tr>
<tr>
<td>Initial estimate of previous residual</td>
<td>$\hat{x}_0$</td>
</tr>
</tbody>
</table>

A.0.2 Supervised learning - the Regression model

Supervised learning is the machine learning task of inferring a function from supervised training data. The training data consist of pairs of an input object, $x_i$, and a desired value, $y_i$. A supervised learning
algorithm analyzes the training data and produces an inferred function. If the output is continuous it is called a regression function.

Regression analysis create a regression function that approximate the observed data, \( y_i \). In simple linear regression it is assumed that the data can be adapted to a straight line, see (A.5).

\[
y = \beta_0 + \beta_1 x_1 + \cdots + \beta_n x_n
\]  

(A.5)

where \( y \) is the dependent variable and \( x_j \) are the independent variables, which affect \( y \), and \( \beta_j \) are the unknown parameters.

A window \( W \) of already observed data, so called training data, are used to calculate the unknown parameters \( \beta_j \). The regression model can now be used to forecast future values.

The simple linear regression line is plotted with the data points in window \( W \), in Figure A.2 and A.3.

\[
x_k = Ax_{k-1} + Bu_{k-1} + w_{k-1}
\]  

(A.6)

with a measured signal \( z \in \mathbb{R}^m \):

\[
z = Hx_k + v_k
\]  

(A.7)

where the random variables \( w_k \) and \( v_k \) represent the process and the measurement noise respectively. The noise is white\(^2\) assumed to be independent of each other and with normal probability distribution.

In the method used in this analysis, the process noise covariance \( Q \) and measurement noise covariance \( R \) matrices are assumed to be constant (though they could change for each time step). The distribution for the white noise is denoted as:

\[
p(w) \sim N(0, Q)
\]  

(A.8)

\[
p(v) \sim N(0, R)
\]  

(A.9)

The \( n \times n \) matrix \( A \) in (A.6) relates the state at the previous time step \( k-1 \) to the state at the current step \( k \), in the absence of either a driving function or process noise. The \( m \times n \) matrix \( H \) in the (A.7) relates the state to the measurement \( z_k \). We assume that both \( A \) and \( H \) are constants. The \( n \times l \) matrix \( B \) relates the optional input \( u \in \mathbb{R}^l \) to the state \( x \).

\[^2\text{White noise - the signal contains equal power within a fixed bandwidth at any center frequency.}\]
An equation that computes the *a posteriori* state estimate \( \hat{x}_k \) as a linear combination of *a priori* estimate, \( \hat{x}_{k-1} \), and a weighted difference between an actual measurement \( z_k \) and a measurement prediction \( H\hat{x}_{k-1} \) is shown in (A.10).

\[
\hat{x}_k = \hat{x}_{k-1} + K(z_k - H\hat{x}_{k-1}) \quad (A.10)
\]

The *gain* or *blending factor* \( K \) is a \( n \times m \) matrix that minimizes the *a posteriori* error covariance. The *a posteriori* estimate error covariance is:

\[
P_k = E[e_k e_k^T] \quad (A.11)
\]

where the *a posteriori* estimate error is:

\[
e_k = x_k - \hat{x}_k \quad (A.12)
\]

One form of \( K \) that minimizes (A.11) is:

\[
K_k = P^{-1}_k H^T (HP_k^{-1} H^T + R)^{-1} = \frac{P^{-1}_k H^T}{HP_k^{-1} H^T + R} \quad (A.13)
\]

An overview of the calculations in the Kalman filter can be seen in Figure A.4. There are initial values for \( \hat{x}_{k-1} \) and \( P_{k-1} \) and also for \( z_k (x_{k-1}) \).

![Diagram of Kalman filter cycle](image)

**Figure A.4:** The ongoing operations of the discrete Kalman filter cycle

### A.0.4 CUSUM

In statistical quality control CUSUM is used for change detection, see [24]. The shortening stands for *Cumulative Sum* and it summarizes distance measures \( s_i \) from a process, where each sample are assigned weights \( \omega_i \). In CUMSUM the test statistic, \( S_i \) is calculated according to:

\[
S_0 = 0 \\
S_i = \max(0, S_{i-1} + s_i - \omega_i) \\
S_i = 0 \text{ if } S_i < 0 \text{ or } S_i > h \quad (A.14)
\]

The distance measure \( s_i \) in this method (the Kalman Method with Regression) is the residuals, i.e. the CUSUM measures the cumulative sum of residuals:

\[
s_i = \begin{cases} 
  r - \hat{r} & \text{for mean CUSUM} \\
  rd - r\hat{d} & \text{for dispersion CUSUM} 
\end{cases} \quad (A.15)
\]
The weights, \( \omega_i \), is the mean of the residuals calculated from time-step 0 to \( i \). When the measure \( S_{i+1} \) reaches above some specified threshold, \( h \), a change is reported.
Bibliography


[33] R Steinert. Change detection applied to discrete and continuous data-streams. *Fig. 1*, page 4, 2010.


