Abstract

This project have mainly focussed on supervised adaptive filters. Different norm-based robust adaptive algorithms are introduced and discussed. Information theoretical methods are introduced by minimizing a KL-divergence between the true joint data distribution and the model joint distribution. This leads to the Shannon generalization error which shows to be a generalization of the norm criterium. A similar measure introduced is the Renyi generalization error, which is used as an information theoretical cost-function. The generalization error and its relation to regularization will be discussed. Two algorithms based on the Renyi generalization error are derived using a Gauss Newton approach. An existing information theoretical algorithm known as the Stochastic Information Gradient (SIG) is derived and discussed. Selected norm and information theoretical methods are tested in a real setup with data sets from an open loop measurement of a hearing aid placed in an artificial ear.

II

Synopsis

Dette projekt har hovedsageligt omhandlet robuste adaptive filtre. Nogle enkelte norm metoder er blevet introduceret og diskuteret. Ved at minimere Kullback Leibler divergensen mellem den sande simultane fordelingsfunktion og modellens simultane fordelingsfunktion, introduceres de informationsteoretiske metoder. Resultatet af denne minimering leder til en objekt funktion, som i projektet bliver kaldes for Shannons generalisations fejl. Shannons generalisations fejl er en generalisering af de klassiske norm metoder. Det vises at Shannons generalisations fejl kan omskrives til en Renyi generalisations fejl, som er et tilsvarende informationsmål. Renyi generalisations fejlen vil blive brugt som objekt funktion. Der eksisterer et sammenhæng mellem generalisations fejlen og regularisering, som vil blive vist. To algoritmer er udledt fra Renyi generalisations fejlen ved brug af en Gauss Newton metode. En eksisterende stokastisk metode (SIG) blev udledt fra Renyi generalisations fejlen. Udvalgte norm og informations teoretiske metoder testes med datasæt fra en åben sløjfe måling, fra et høre-apparat placeret i et kunstigt øre.

Nomenclature and mathematical conventions

Abbreviations

LMS	Least Mean Square
LMP	Least Mean p-norm
SIG	Stochastic Information Gradient
NLMS	Normalized LMS
NLMP	Normalized LMP
GNLMS	Gaussian Normalized LMS
RQEGN	Recursive Quadratic Entropy Gauss Newton
MSE	Mean Square Error
KL	Kullback Leibler
RT	Rule of Thumb
CV	Cross Validation
ANLL	Aveage negative log - likelihood
s.t.	Subject to
i.i.d.	Idependent identically distributed

Math conventions and symbols

Lower case bold letters refer to vectors eg.

$$\boldsymbol{x} = [x_1, x_2, x_3, \ldots, x_p]^T .$$

Upper case refers to matrices ($\boldsymbol{X}$). Scalar values are given as x.

Mathematical symbols

$D(x)$	The cumulated probability function of x
$\Gamma_{\boldsymbol{xx}}$	Autocorrelation matrix of $\boldsymbol{x}$
$\gamma_{y\boldsymbol{x}}$	Crosscorrelation vector between y and $\boldsymbol{x}$
$\boldsymbol{K}$	Gain vector (RLS/RLP)
$p(e)$	probability density function of the error.
$p(y\|\boldsymbol{x})$	conditional probability density function of y given $\boldsymbol{x}$
$p(y, \boldsymbol{x})$	joint probability density function
$f(\boldsymbol{w}, \boldsymbol{x})$	Mathematical function describing the model. In this project $f(\cdot)$ will be a linear function.
$G(\boldsymbol{w})$	Generalization error
$H(X)$	Entropy of the discrete or continuous variable X unless other specified

Operators

$\text{sgn}(x)$	Sign function
$\text{Tr}\{.\}$	Trace operator

Other symbols

k	time index unless other specified
x_k	input applied to an adaptive filter.
d_k	output of an adaptive filter.
y_k	desired response.
v_k	added signal / noise.
$e_k = d_k - y_k$	estimation error.
$\boldsymbol{w}$	The model weights.
L	Number of weights in the model.

Acknowledgements

I would like to thank Jan Larsen, Joaquin Quinonero and Lars Kai Hansen for help and inspiration during the project phase. A special thank to Preben Kidmose (*Widex*) for providing MATLAB material, answering questions and using time with producing the hearing aid data-sets for this project.

Also I would like to thank Niels Henrik Pontoppidan and Kaare Brandt Petersen for help with mathematical and LaTeX issues. A special thank to Mogens Dyrdal, Ulla Nørhave and all other of the students here at the floor for being nice persons and helpful during the whole project.

A last thank to my family and my girlfriend for putting up with me during the final part of this project. Thank you Sarah.

Anders Meng

March 3, 2003

Contents

Chapter 1

Introduction

This project will mainly focus on supervised adaptive filters. Robustness is defined in this context, as the ability of the algorithm to maintain convergence when applied to signals with properties that differ from the initial environment assumptions. Several techniques exist for robust filtering when the environment is non-Gaussian. Many adaptive algorithms are based on minimizing a specific norm, which is assumed to exist for the data. This assumption however, puts some limitation on fixed norm methods when applied in other environments than the intended. Some of the algorithms, which have been investigated, cope with this by either ensuring a certain norm or by minimizing a variable norm. Information theoretic methods will be introduced as to minimize an entropy based generalization error. The generalization error minimizes a Kullback Leibler divergence between the true joint density function and the model joint density function. The information theoretic methods will work directly on the unknown probability density function (working directly on the different moments of the distribution). The density can be approximated assuming a specific environment (parametric method) or one can use a density estimator (non-parametric method). In the project we have concentrated on non-parametric methods to determine the distribution from a finite sized data set.

In the following, a short explanation of the different chapters in the report is given.

Chapter 2 The additive noise model is introduced. A discussion on different kind of noise sources is also given in this chapter.

Chapter 3 Different robust adaptive filter algorithms are introduced. Both stochastic gradient and recursive methods are discussed in this chapter.

Chapter 4 The additive noise model together with the KL-divergence is used to define a generalization error. The generalization error is an information theoretic measure, and will be used as a cost function. Several properties of the cost function is discussed. The cost function, and its relation to regularization is investigated. Three different types of adaptive algorithms are derived from

the cost function.

Chapter 5 contains experiments performed on the algorithms based on the information theoretic method. In this section a practical example from the hearing aid industry is tested. Both the norm-algorithms and the information theoretic methods are tested in this context.

Chapter 6 A discussion on further work.

Chapter 2

The model

This section introduces the model, which is considered in this project. The second section of the chapter will discuss different kinds of noise, which are used later in the project to test the algorithms.

2.1 Additive noise model

The model under consideration in this project is known as an additive noise model given as

$$y_k = f(\boldsymbol{x}_k, \boldsymbol{w}) + e_k \tag{2.1}$$

where

1. $e_k \in \mathbb{R}$ is a noisy measurement (a stochastic process). The noise e_k is normally assumed to be of a specific shape.
2. $y_k \in \mathbb{R}$ is a noisy measurement which is linearly or nonlinearly related to $\boldsymbol{w}$ and $\boldsymbol{x}_k$.
3. $\boldsymbol{w} \in \mathbb{R}^L$ is the model parameters.
4. $\boldsymbol{x}_k \in \mathbb{R}^d$ could be a stochastic process and is given as the input to the model. In the adaption process the characteristic of this process can have quite big influence on the convergence of the different algorithms considered in this project.
5. $f(\cdot)$ is a linear or nonlinear function which maps the input $\boldsymbol{x}_k$ and the model parameters $\boldsymbol{w}$ into a scalar $\hat{y}_k$, so $f(\cdot) : \mathbb{R}^d \mapsto \mathbb{R}$. In this project only transversal filters are considered (FIR-filters) in which $f(\cdot)$ is a linear function of its inputs.

Other models than the additive noise model exists see [24] for examples of recurrent models.

Several examples of the practical use of an additive noise model can be found in [15], some which will be discussed next. First the *system identification setup* is discussed.

System identification setup
A system identification scheme is illustrated in figure (2.1) page (4), where the plant

is the unknown system to be identified using the model. In some cases the number of model parameters required is not known a priory. If one has a proper measure of the error, then one should select the number of parameters that minimize the error measure the most. In the case of linear adaptive filters, selecting a too big number of parameters (more than is necessary) will cause the model parameters with no importance to go to zero. The plant does not necessarily have to be stationary but could be dynamic in nature, which means that the unknown weights will be time dependent. With a time-varying plant the requirements to the adaptive algorithm will increase since the algorithm have to catch changes in the system parameters. Some different setups which would go under the category system identification are given in [8] [12] [20].

In the experimental part of this project, we will perform a system identification using a realistic setup with data from hearing aid setup. This is explained more in details in the experimental part of the project. Another setup, which is discussed, is the

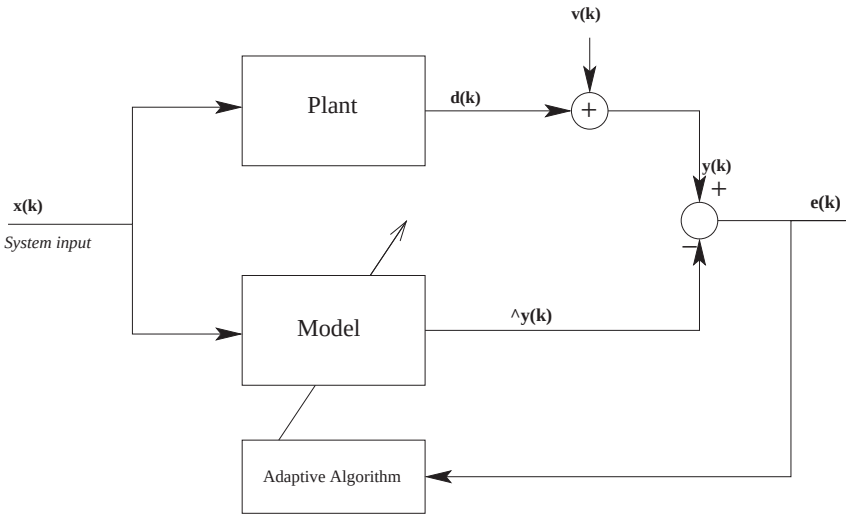


Figure 2.1: System Identification setup

Prediction setup.

Prediction setup

In this setup the adaptive filter's operation is to provide the best prediction of the present value of the signal applied (u_k). The predictor is shown in figure (2.2) page (5). The sample u_k acts as the desired signal, which the adaptive filter has to predict from earlier signal samples. The prediction can be one sample ahead or several samples ahead (depending on the delay). A typical example of a signal that can be used in a prediction setup is the autoregressive process (AR), which is discussed in more detail in section (2.3) page (12). The AR-process has been utilized

in [14] and [26] to test an adaptive algorithm (FIR) in a prediction setup. Also other signals may be applied in a prediction setup, such as deterministic signals, which are utilized in [6] to test a SIG-algorithm (Stochastic Information Gradient).

The prediction setup is applied to verify the various algorithms on how they perform in different statistical environments.

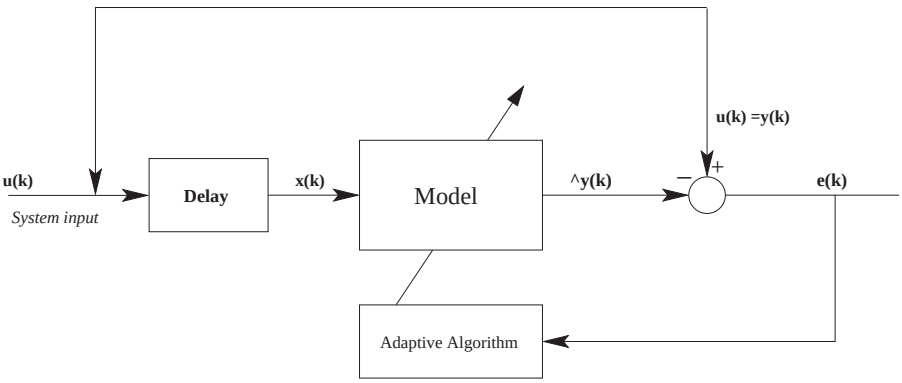


Figure 2.2: Prediction setup

The signals applied for the adaptive filter/system can take many different forms. Stationary and non-stationary signals are considered in this project. The non-stationary signals will mainly be the realistic setup, where speech signals are used.

2.2 Characterization of the noise

To test the different algorithms presented in this thesis some synthetic data is needed. Many of the algorithms presented in adaptive filter theory are optimized with respect to some specific norm. This normally puts a limitation on the flexibility of the algorithm, when used in other environments than the one specified for the algorithm. As an example the LMS algorithm is designed for handling Gaussian type of noise, but many signals are not well approximated by a Gaussian noise source.

Stationary and non-stationary signals have already been mentioned. A definition on a strictly sense stationary signal is found in [29]

> Suppose that we have n samples of the random process $X(t)$ at $t = t_i$, $i = 1, 2, ..., n$, and another set of n samples displaced in time from the first

set by an amount τ. Thus, the second set of samples are $X_{t_i+\tau} \equiv X(t_i + \tau),\ i = 1, 2, ..., n$. This second set of n random variables is characterized by the joint probability density function $p(x_{t_i+\tau}, ..., x_{t_n+\tau})$. The joint PDF's of the two sets of random variables may or may not be identical. When they are identical then

$$p(x_{t_1}, x_{t_2}, ..., x_{t_n}) = p(x_{t_1+\tau}, x_{t_2+\tau}, ..., x_{t_n+\tau}) \tag{2.2}$$

for all τ and n, then the random process is said to be stationary in the strict sense.

It is shown that if the joint PDF is not invariant to time, the process is non-stationary.

A weaker requirement to stationarity is the so-called wide-sense stationarity, which requires that the mean value of the process is constant and that the autocorrelation function is only dependent on the time-difference. Thus assuming that X(t) is a wide-sense stationary signal , then the mean value should be independent of time giving

$$E[X_{t_i}] = E[X(t_{i+\tau})] \tag{2.3}$$

for all i and τ. The autocorrelation function should only depend on a time difference between two random variables, defining $\tau = t_2 - t_1$ then

$$\gamma_{xx}(\tau) \equiv E[X_{t1} X_{t_2}] = E[X_{t1} X_{t_1+\tau}] \tag{2.4}$$

for all τ and t_1. Synthetic signals can be used for testing the different algorithms in their intended environment. The algorithms, when tested with real world signals, normally shows a worse performance than when used with synthetic signals. This is due to temporarily correlation and non-stationarity in the real world signals. Real world signals will have to be applied to the adaptive system as to really characterize the performance of an algorithm.

In the next section an introduction to stable processes are given. These are of interest primarily due to the fact that speech/ audio signals can be modelled using the stable processes [21]. Much of the information to this subject have been found in [30].

α stable random variables

In [21] several examples are given, which shows that stable signals (where $\alpha < 2$) models speech/audio signals better than Gaussian signals. Since in the hearing aid test of chosen algorithms speech signals are used, it makes sense to be able to generate alpha stable signals for testing. Some basic properties of a stable process are introduced in this section as well as a discussion of the parameters for alpha stable variables. Also fractional lower order moments (FLOM) are discussed in this section.

In [30] it is stated that stable processes share characteristics with Gaussian processes. The characteristics they share are the stability property and a generalized central

limit theorem. In [30] examples of noise sources, which are impulsive in nature, is given. They are not very well modelled by a Gaussian distribution since the tails of a Gaussian dies out as $exp(-x^2)$. The tails of an alpha-stable distribution is generally larger than for a Gaussian distribution. In addition, the hump is more peaked for an alpha-stable distribution. The stable distribution is very flexible, since the heaviness of the tails is controlled by a parameter called the characteristic exponent (α where $0 < \alpha \leq 2$). When $\alpha = 2$ the stable distribution is a Gaussian distribution. When $\alpha = 1$ the stable distribution is also known as a Cauchy distribution. This is used very often to model impulsive noise. The "Generalized Central Limit Theorem" and "Stability property" from [30]:

> *Generalized Central Limit Theorem*
> The generalized central limit theorem states that if a sum of infinitely many i.i.d. random variables with or without variance converges to a distribution when increasing the number of variables, the limit distribution will be stable[1].
> *Stability property*
> The sum of two independent stable random variables with the same characteristic exponent will again be a stable random variable with same characteristic exponent.

The above statements are a generalization of the central limit theorem and stability properties known from normal-distributed variables.

For the α-stable variables the moments are only finite for orders less than α. This is an important information since many algorithms are based on minimization of the two-norm of the error (LMS and RLS). A similar measure of variance for Gaussian variables is called dispersion for α-stable distributions. The larger the dispersion (γ) of a stable variable, the larger the spread around its median. If the error signal is an alpha stable signal, then algorithms minimizing the dispersion of the error signal are needed. This is, according to [30] equal to minimizing the fractional lower order moments of estimation errors (measure of the L_p-distance between a desired signal (y_k) and an estimated signal (predicted by model, $\hat{y}_k$), where $p < \alpha \leq 2$). Algorithms like the LMP (RLP), which are discussed in chapter (3.1.1) page (15) minimizes the dispersion.

The probability density function of stable random variables does not exists in closed form (Only for $\alpha = 2$ and $\alpha = 1$). Power series expansions exist for a standard stable distribution ($\gamma = 1$ and $a = 0$). Instead of specifying a PDF the characteristic function is given [30]

$$\varphi(t) = exp\{jat - \gamma|t|^\alpha[1 + j\beta \text{sgn}(t)\omega(t,\alpha)]\} \tag{2.5}$$

where

$$\omega(t,\alpha) = \left\{ \begin{array}{ll} tan\frac{\alpha\pi}{2} & \text{for } \alpha \neq 1 \\ \frac{2}{\pi}\log|t| & \text{for } \alpha = 2 \end{array} \right\}. \tag{2.6}$$

[1] This basically means that if a limit distribution exists - say $G(x, \theta_0)$, taking two or more random variables of this distribution and adding them, will again produce a variables with a distribution $G(x, \theta_1)$

and $-\infty < a < \infty$, $\gamma > 0$, $0 < \alpha \leq 2$, $-1 \leq \beta \leq 1$.

The stable distribution is completely specified by the location parameter a, the dispersion (scale parameter) γ, the skewness β and the characteristic exponent α. If $\beta = 0$ and $a = 0$ the process is also known as a $S\alpha S$-process (symmetric alpha stable process).

By setting $\beta = 0$ and $a = 0$ ($S\alpha S$ variable) the PDF of the stable process can be found by invoking an inverse Fourier transformation with a relevant scaling of the characteristic function.

The characteristic function of a $S\alpha S$ variable is then given by : $\varphi(\omega) = e^{-\gamma|\omega|^\alpha}$. Performing an inverse Fourier transformation of the characteristic functions gives

$$p(x,\alpha,\gamma) = \frac{1}{2\pi}\int_{-\infty}^{\infty} \varphi(\omega)e^{-j\omega x}d\omega$$

$$p(x,\alpha,\gamma) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-\gamma|\omega|^\alpha}e^{-j\omega x}d\omega$$

$$p(x,\alpha,\gamma) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{-\gamma|\omega|^\alpha}[cos(\omega x) - jsin(\omega x)]d\omega \tag{2.7}$$

$$p(x,\alpha,\gamma) = \frac{1}{\pi}\int_{0}^{\infty} e^{-\gamma\omega^\alpha}cos(\omega x)d\omega. \tag{2.8}$$

Since the sine function is an un-even function and the characteristic function an even the integral will equal zero. The integral from $-\infty$ to ∞ of two even functions can be changed to an integral from zero to infinity. For the case of $\alpha = 1$ then ([34] p.98 formula 15.68)

$$\begin{aligned} p(x,1,\gamma) &= \frac{1}{\pi}\int_{0}^{\infty} e^{-\gamma\omega}cos(\omega x)d\omega \\ &= \frac{1}{\pi}\frac{\gamma}{\gamma^2 + x^2} \end{aligned} \tag{2.9}$$

which is the Cauchy-distribution. For $\alpha = 2$ the Gaussian PDF is found ([34] p.98 formula 15.73)

$$\begin{aligned} p(x,2,\gamma) &= \frac{1}{\pi}\int_{0}^{\infty} e^{-\gamma\omega^2}cos(\omega x)d\omega \\ &= \frac{1}{2\pi}\sqrt{\frac{\pi}{\gamma}}\, e^{-\frac{x^2}{4\gamma}} \\ &= \frac{1}{\sqrt{2\pi\sigma}}\, e^{-\frac{x^2}{2\sigma^2}} \end{aligned} \tag{2.10}$$

if γ is selected to $\frac{\sigma^2}{2}$, which shows the relationship between the variance of a Gaussian variable and the dispersion. This is two special cases. Selecting $0 < \alpha \leq 2$ one have to use a series expansion of the exponential to give

$$p(x,\alpha,\gamma) = \sum_{k=0}^{\infty}\frac{-\gamma^k}{k!}\int \omega^{\alpha k}\cos(\omega x)d\omega. \tag{2.11}$$

From [30] the moments of an alpha stable variable are given as $(0 < \alpha < 2)$:

$$E\{|x|^p\} = \infty \quad \text{, if } p \geq \alpha \tag{2.12}$$
$$E\{|x|^p\} < \infty \quad \text{, if } 0 \leq p < \alpha \tag{2.13}$$
$$E\{|x|^p\} < \infty \quad \text{, for all } p \geq 0 \quad (\alpha = 2) \tag{2.14}$$

Only for $\alpha = 2$ all moments exists. Special for the symmetric alpha stable distributions the *Fractional lower order moment* also called FLOM are given as

$$E\{|x|^p\} = C(p,\alpha)\gamma^{\frac{p}{\alpha}} \quad \text{for } 0 < p < \alpha \tag{2.15}$$

where $C(p,\alpha)$ is given as

$$C(p,\alpha) = \frac{2^{p+1}\Gamma(\frac{p+1}{2})\Gamma(-p/\alpha)}{\alpha\sqrt{\pi}\Gamma(-p/2)} \tag{2.16}$$

and the $\Gamma(x)$- function

$$\Gamma(x) = \int_0^\infty t^{x-1}e^{-t}dt. \tag{2.17}$$

The FLOM shown in (2.15) and (2.16) is not dependent on the variable x, but only on the characteristic exponent α and on p. Inspired by [22] various p'th order FLOM is plotted for different values of α as a function of p. This plot is seen in figure (2.3) page (10). The p'th order moment of a $S\alpha S$-random variable increases dramatically when p is chosen higher than the corresponding α. The p'th order moment plotted for a random $S\alpha S$ variable is the case of an infinitely sample size. Since the sample size used for simulating is finite the p'th order moment of a sample size will generally not increase in value as fast as shown in figure (2.3) page (10).

How to generate α stable variables

To generate α-stable random variables, several methods exist. In this project a method described by [17] have been used. The code, for generation of alpha stable variables have been borrowed from *Preben Kidmose*[2]. The implementation was done during his PhD-thesis [22]. The method transforms two independent uniform variables non-linearly into a stable random variable.

[2] With his permission

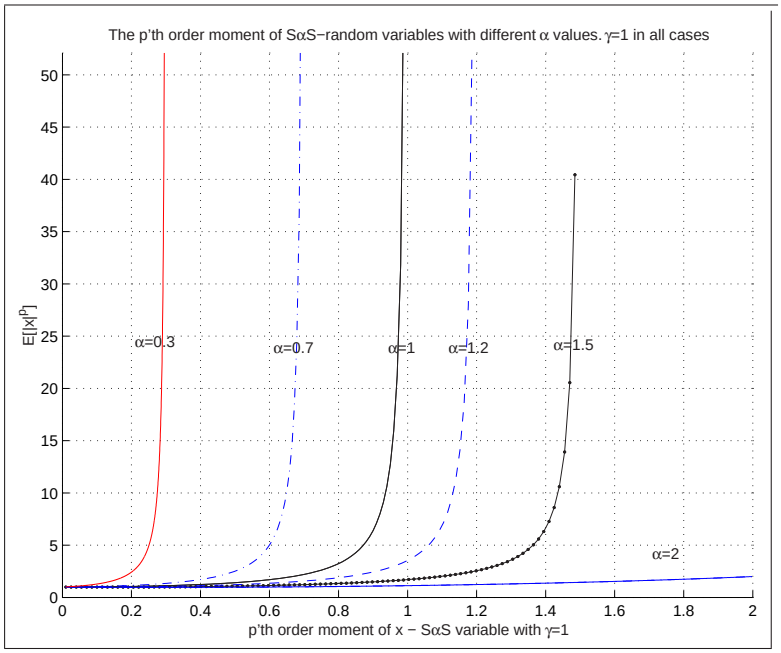


Figure 2.3: p'th order moment of a $S\alpha S$ random variable with dispersion $\gamma = 1$

How to generate an arbitrary distribution

Since MATLAB do not have built in functions for generation of random variables other than Gaussian and uniform distributed variables, some method for generation of arbitrary random variables is needed. Normally one transforms a uniform distributed random variable into the wanted distribution by transforming the uniform distribution in some nonlinear fashion.

To transform a random variable X which is uniform distributed into a variable Y having a distribution, say p(y), where $\int_{-\infty}^{\infty} p(y)dy = 1$ some transformation is needed [1]. Using the transformation rule of equaling the area under the density function of the uniform and arbitrary distribution, so:

$$\begin{aligned} \int p(y)dy &= \int p(x)dx \\ |p(y)dy| &= |p(x)dx| \\ p(y)|dy| &= p(x)|dx| \\ p(y) &= p(x)\left|\frac{dx}{dy}\right| \end{aligned} \tag{2.18}$$

and since

$$p(x) = \begin{cases} 1 & 0 \leq x \leq 1 \\ 0 & \text{Otherwise} \end{cases} \tag{2.19}$$

gives the following differential equation to be solved :

$$\frac{dx}{dy} = p(y).$$

The solution to this differential equation is the cumulated probability function of Y: $x = D(y)$. In order to generate random variables with a given PDF the inverse cumulated probability function of Y is needed to generate the wanted X-variables

$$y = D^{-1}(x).$$

So random variables, with an arbitrary PDF, can only be generated if the inverse cumulated probability function exist (can be found analytical or numerically).

2.3 Generation of data applying an AR-process

The autoregressive process AR(k) can be used in a prediction setup. The AR(k) process is given as

$$x_k + \sum_{i=1}^{k} a_i x_{k-i} = v_k \tag{2.20}$$

$$x_k = -\sum_{i=1}^{k} a_i x_{k-i} + v_k \tag{2.21}$$

where v_k could be a sequence of i.i.d. $S\alpha S$ random variables, Gaussian distributed samples or some sub-Gaussian process. When using i.i.d. $S\alpha S$ random variables for v_k then x_k is known as a linear $S\alpha S$ stationary process (when AR-parameters are selected properly).

One have to be careful when selecting the AR-parameters, since a bad selection will cause the signal x_k to become non-stationary.

Performing a z-transformation of the AR-process given in (2.20) gives

$$X(z) = -\sum_{i=1}^{k} a_i z^{-i} X(z) + V(z) \tag{2.22}$$

which after rearranging gives

$$\frac{X(z)}{V(z)} = \frac{1}{1 + \sum_{i=1}^{k} a_i z^{-i}} \equiv H(z), \tag{2.23}$$

a transfer function. In order for the AR-process to be stable the roots of the equation given in the nominator (solve $1 + \sum_{i=1}^{k} a_i z^{-i} = 0$) have to be within the unit circle [15]. From figure (2.4) page (13) an example of a AR-process, where the parameters are selected such that one of the poles are just outside the unit circle. The instability of the process can be seen. Selection of the parameters where one of the poles are at the unit-circle causes oscillation of the AR-process, which can be seen from figure (2.5) page (13). Figure (2.6) page (14) shows that a selection of the poles within the unit-circle causes the process to be stable as desired.

2.4 Discussion

In this chapter, the additive noise model have been introduced. Some scenarios in which the additive model can be used has been given. Different forms of additive noise, which will be used in the project for testing the various algorithms, have been discussed. The AR process has been introduced.

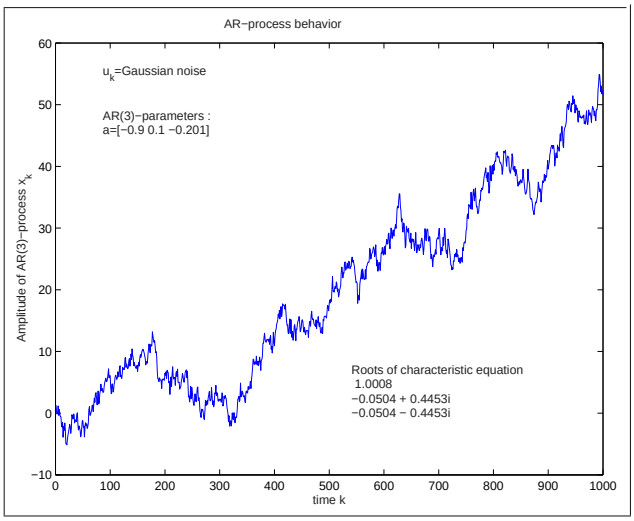


Figure 2.4: AR(3)-processes in unstable situation

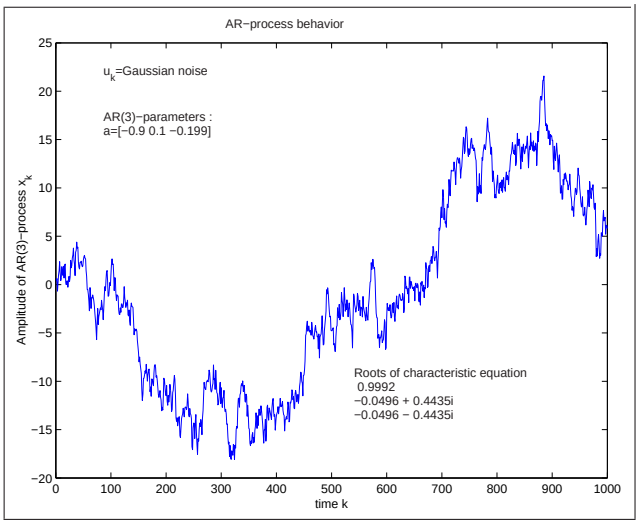


Figure 2.5: AR(3)-processes in potentially unstable situation

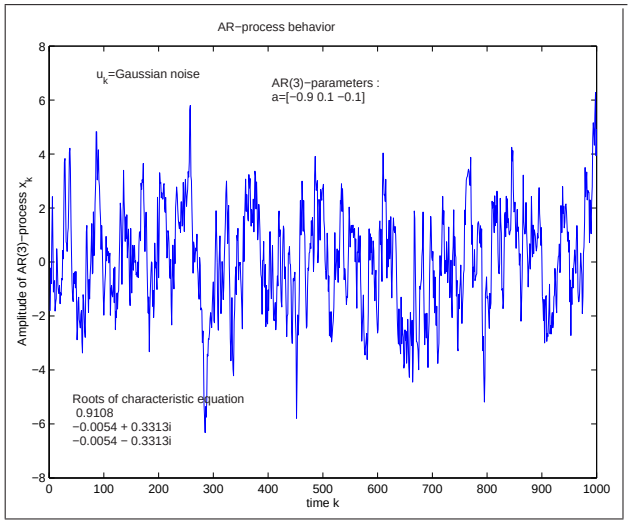


Figure 2.6: Stable AR(3) process

Chapter 3

Robust norm-related algorithms

This chapter will discuss a couple of well-known robust algorithms which are based on norm-minimization. The NLMP (Normalized Least p-norm, NLMS as a special case), RLP (Recursive Least p-norm) and GNLMS (Gaussian Normalized Least Square) will be discussed. In the section on the RLP algorithm a method to estimate the p-parameter when the environment is a $S(\alpha)S$ environment is given.

3.1 Stochastic gradient methods

3.1.1 NLMS/LMS

The least mean square algorithm (LMS) was originally presented in the sixties by Widrow and Hoff. The algorithm is a stochastic gradient algorithm and is finding practical use in many aspects. The LMS algorithm is also known to be a robust algorithm and is therefore used in many papers for performance comparison, mentioning just a few [21] [26] [6] and [30]. Most of the papers use a Normalized LMS algorithm which uses a normalization of the estimated gradient used in the LMS algorithm. In the literature [15] the LMS algorithm have been investigated analytically regarding convergence properties, transient behavior utilizing the independence assumption.

The independence assumption given in [15]

1. *The tap-input vectors* $\boldsymbol{x}_1$, $\boldsymbol{x}_2$, $\boldsymbol{x}_3, .., \boldsymbol{x}_k$ *constitute a sequence of statistically independent vectors.*
2. *At time k, the input vector* $\boldsymbol{x}_k$ *is statistically independent of all previous samples of the desired response, namely,* $y_1, y_2, ..., y_{k-1}$.
3. *At time k, the desired response* y_k *is dependent on the corresponding tap-input vector* $\boldsymbol{x}_k$*, but statistically independent of all previous samples of the desired response.*
4. *The tap-input vector* $\boldsymbol{x}_k$ *and the desired response consists of mutually Gaussian distributed random variables for all k.*

The LMS algorithm is a cost function based algorithm. The cost function is the mean square error which is given as

$$J_k = E\left[|e_k|^2\right]. \tag{3.1}$$

In an adaptive system setup the weights of the model are adjusted to minimize the mean square error. A simple method to achieve minimum MSE is to use a steepest descent method. This method steps in the direction of steepest descent (the negative gradient of the cost-function)

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \frac{\mu}{2}[-\nabla J_k]. \tag{3.2}$$

The gradient of the cost-function in (3.1) is found to be

$$\nabla J(\boldsymbol{w}) = -2\boldsymbol{\gamma}_{y\boldsymbol{x}} + 2\boldsymbol{\Gamma}_{\boldsymbol{x}\boldsymbol{x}}\boldsymbol{w}.$$

The minimum of the cost-function is found when the gradient equals zero. In order to be a minimum the Hessian matrix of the cost-function should be positive definite[1]. Solving for the model weights, when the Hessian is positive definite, produces the Wiener solution (Least Square solution). This solution is optimal in the mean square sense. The Hessian matrix is found to be proportional with the autocorrelation matrix $\nabla^2_{\boldsymbol{w}} = 2\boldsymbol{\Gamma}_{\boldsymbol{x}\boldsymbol{x}}$. It can be shown that the autocorrelation matrix usually is a positive definite matrix in which a minimum of the cost function exists and is found by the least squares solution. In [15] this property is discussed. So in order to find the Wiener solution one solves the system of linear equations

$$\boldsymbol{w} = {\boldsymbol{\Gamma}_{\boldsymbol{x}\boldsymbol{x}}}^{-1}\boldsymbol{\gamma}_{\boldsymbol{y}\boldsymbol{x}} \tag{3.3}$$

Normally we do not know the true auto/- and cross correlation function so the Wiener solution is approximated at some time instant k as

$$\boldsymbol{w}_k = \boldsymbol{R}^{-1}_{\boldsymbol{x}\boldsymbol{x}_k}\boldsymbol{r}_{y\boldsymbol{x}_k}. \tag{3.4}$$

where $\boldsymbol{R}_{\boldsymbol{x}\boldsymbol{x}_k}(\boldsymbol{w})$ is the autocorrelation matrix of the input and $\boldsymbol{r}_{y\boldsymbol{x}_k}(\boldsymbol{w})$ is the cross-correlation vector estimated at time instant k. Another way to minimize the cost-function is to use instantaneous estimates of the cross-correlation vector and the auto-correlation matrix. This leads to the stochastic gradient method. The stochastic gradient becomes

$$\hat{\nabla} J_k = -2\boldsymbol{x}_k y_k + 2\boldsymbol{x}_k\boldsymbol{x}_k^T\boldsymbol{w}_k.$$

Inserting this result into (3.2) using that $e_k = y_k - \boldsymbol{x}_k^T\boldsymbol{w}_k$ the following stochastic update is found, also known as the LMS algorithm:

[1] All eigenvalues of the Hessian matrix should be positive. It is also the same as requiring that $\boldsymbol{x}^T\boldsymbol{H}\boldsymbol{x} > 0$ for $\boldsymbol{x} : \mathbb{R}^L$.

$$\boldsymbol{w}_k = \boldsymbol{w}_k + \mu e_k \boldsymbol{x}_k. \tag{3.5}$$

It can be shown that in order for the LMS algorithm to converge in the mean square sense the step-size parameter should satisfy the following condition:

$$0 < \mu < \frac{2}{\lambda_{max}}$$

where λ_{max} is the largest eigenvalue of the autocorrelation matrix $\boldsymbol{\Gamma}_{\boldsymbol{xx}}(\boldsymbol{w})$.

The Normalized LMS algorithm (NLMS) can be derived from the previous observation of the convergence criteria or by minimizing another cost-function. The latter method is presented in [15] as a minimization problem of the form

$$||\delta \boldsymbol{w}_{k+1}||^2 \quad \text{s.t.} \quad \boldsymbol{w}_{k+1}^T \boldsymbol{x}_k = y_k$$

where $\delta \boldsymbol{w}_{k+1} = \boldsymbol{w}_{k+1} - \boldsymbol{w}_k$. This constrained optimization problem is solved using the method of Lagrange multipliers. The method gives insight, since the two norm of the weight update is minimized. A more direct method is to estimate the maximum eigenvalue from the fact that the autocorrelation matrix $\boldsymbol{R}_{\boldsymbol{xx}k}$ have nonnegative eigenvectors (assumed to be semi/ or -positive definite) in which an upper bound on λ_{max} can be found as

$$\lambda_{max} < \sum_{i=0}^{L-1} \lambda_i = tr\{\boldsymbol{R}_{\boldsymbol{xx}k}\}.$$

Using instantaneous estimates of the autocorrelation matrix the following expression for the step-size is found

$$\lambda_{max} < tr\{\boldsymbol{x}_k \boldsymbol{x}_k^T\} = ||\boldsymbol{x}_k||_2^2.$$

From where the step-size is no longer independent on k

$$\mu_k = \frac{\mu}{||\boldsymbol{x}_k||_2^2}.$$

Selecting $0 < \mu < 2$ will make the method convergent. The NLMS algorithm is written as

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \mu \frac{e_k \boldsymbol{x}_k}{\boldsymbol{x}_k^T \boldsymbol{x}_k + a} \tag{3.6}$$

where a^2 is a small constant which assures that the expression do not "blow up" in the case of very small input values.

[2]Typically chosen to be 1e-8. For most workstations the floating point relative accuracy, eps 1e-16, and sqrt(eps) 1e-8

The complexity of the LMS algorithms is $O(\mathrm{L})$, where L is the number of weights (L+1 multiplications and L additions). When the NLMS algorithm have been running for a while only 2 multiplications, 3 additions and 1 division is needed since the inner product can be updated recursively. Therefore, the low complexity of the LMS/NLMS algorithm makes it very useful in many simple setups.

3.1.2 NLMP/LMP

The Normalized Least Mean p-norm algorithm (NLMP) is based on the Least Mean p-norm algorithm. The LMP algorithm is presented in both [26] and [30] and is designed to handle non-Gaussian stable processes. Several authors have used this algorithm for comparison, see [20][21][14]. The algorithm is designed to minimize the dispersion of the estimation error (e_k) in $S(\alpha)S$ noise, which is as mentioned before more heavy-tailed than Gaussian noise. As discussed earlier $S(\alpha)S$ random processes do not have a second moment (for $\alpha < 2$), so minimizing a cost function based on the mean square error $J_k = E\left[|e_k|^2\right]$ do not provide any good results. Instead it makes sense to minimize the p'th order moment of the error, which can be expressed as the following cost-function

$$J_k = E\left[|e_k|^p\right]$$

where the p should be chosen small enough so the distribution under consideration, have information regarding the p'th moment(choose $p < \alpha$(in [21] $p = \alpha - 0.2$)). Minimizing the p'th order moment of the error corresponds to a minimization of the dispersion.

Using a stochastic update method, the gradient with respect to the weights of the cost function are determined. The gradient is found to be

$$\nabla J_k = -E\left[p|e_k|^{p-1}\mathrm{sgn}(e_k)\boldsymbol{x}_k\right].$$

Using instantaneous estimates of the gradient and inserting into the method of steepest descent one finds the following update formula also known as the LMP method

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \beta e_k^{\langle p-1\rangle}\boldsymbol{x}_k$$

where $\beta = \mu\frac{p}{2}$ and $a^{\langle p\rangle} = |a|^p\mathrm{sgn}(a)$

A special case of the LMP algorithm is when $p = 1$, in which the algorithm reduces to the sign algorithm. This is mentioned in [30] as the LMAD-algorithm (least mean absolute deviation).

Motivated by the increased convergence speed of the NLMS method with correlation in the input signal a normalized LMP algorithm is introduced. This algorithm should also work better in colored environments. The NLMP update algorithm is found in [26] as

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \frac{\beta}{||\boldsymbol{x}_k||_p^p + a} e_k^{\langle p-1 \rangle} \boldsymbol{x}_k \tag{3.7}$$

The p-norm of the input vector is used as normalization. From [8] it is shown that the following minimization problem

$$\min ||\boldsymbol{w}_{k+1} - \boldsymbol{w}_k||_p \quad \text{s.t.} \quad y_k - \boldsymbol{w}_{k+1}^T \boldsymbol{x}_k = 0 \tag{3.8}$$

leads to the normalized LMP algorithm. a is again a constant assuring that the weights will not diverge in case of very small input value.

The complexity of the LMP algorithm is given as (2+L) multiplications, L additions and 1 nonlinear operation. So the complexity of the NLMP algorithm is $\mathcal{O}(L)$ operations. The NLMP-algorithm when running only requires 3 additions, 2 nonlinear operations and 1 division additionally compared with the LMP algorithm.

For the algorithm to work in a changing environment some determination of the p-value is needed. In the section on the RLP algorithm a method to estimate p from the N prior samples is given. In [21] a variable p-norm method was tested in a double - talk situation with positive results.

3.1.3 GNLMS

The Gaussian NLMS algorithm was introduced in [20] as a method for working with non-Gaussian environments.

Median Orthogonality

The method is based on median orthogonality which has been introduced in [4]. Instead of minimizing a traditional cost-function the algorithm is based on median orthogonality, which is required at the solution. Definition of median orthogonality:

> Denote x_1 and x_2 as two random variables and let $M\{\ \}$ denote the median operator, which is similar to the normal expectation $E\{\ \}$. If the median product is zero between the two random variables $M\{x_1 x_2\} = 0$ then x_1 and x_2 are said to be median orthogonal. The median orthogonality can also be expressed as
>
> $$x_1 \perp_M x_2.$$

In the Gaussian case (x_1 and $x_2 \in N(\mu, \sigma^2)$) the MO criterion reduces to the normal orthogonality criterion. An interesting point is that median orthogonality do not restrict the distribution of x_1 and x_2 [4]. In an adaptive setup we require that $e \perp_M \boldsymbol{x}$ which basically means that no requirement on the distribution of the error and input is needed.

Properties of median orthogonality (MO) for random variables with symmetric distributions:

- Independence is necessary and sufficient for MO
- MO is necessary but not sufficient for independence

The last statement means that variables x_1 and x_2 can be median orthogonal even though the variables are dependent.

The following statement is proven in [4]. If x_1 and x_2 is two random variables (not necessarily independent variables) and defining a new random variable $z = x_1 x_2$, then the median orthogonality of x_1 and x_2 is achieved if the distribution of z is symmetric:

$z = x_1 x_2,\ \mathcal{M}\{x_1 x_2\} = \mathcal{M}\{z\} = 0 \Longleftrightarrow p(z) = p(-z).$

The GNLMS algorithm

The GNLMS perform a nonlinear transformation of the input variable $\boldsymbol{x}_k$ and the error variable e_k. The variables after the nonlinear transformation is Gaussian variables. This is to ensure a second order moment, since the NLMS algorithm is used in the minimization. The input signal and error signal may follow an arbitrary distribution. The algorithm as proposed in [20] can be divided into 4 stages:

1. *Perform a empirical density transformation to obtain a uniform distribution.* Let x be some arbitrary random variable from which $x_1, x_2, ..., x_N$ are N observations of this random variable (arbitrary distribution). Sort the N observations such that $x_{.,1} \leq x_{.,2} \leq .. \leq x_{n,m} \leq .. \leq x_{.,N}$, where m is the index of the sorted data and n corresponds to the n'th observation in the original data of x. Then the variable v (the sorted data) is forced to follow a uniform density with the PDF:
$$p(v) = \left\{ \begin{array}{ll} \frac{1}{2r} & \text{if } |x-a| \leq r \\ 0 & \text{if } |x-a| > r \end{array} \right\}.$$
if v_k is assigned the value:
$$v_n = \frac{2m - Nr}{N} + a$$
2. *Transform the uniform distributed variables into Gaussian distributed variables.* As mentioned earlier this is done using the inverse cumulated distribution function for the Gaussian.
3. *Scaling the transformed variables.* Scaling is needed since all information about amplitude is lost in the empirical transformation to the uniform data. The scaling should be performed in a norm, which exist for the arbitrary random variable x. Define $z = scaling(y)$, where $y = D^{-1}(v)$. The scaling should be so that $||x||_p = ||y||_p$ for some norm p. In the algorithm implementation p is chosen to be 0.5.
4. *Update weight coefficients* Denoting the last three steps as a nonlinear transformation function $g(\cdot)$, this function operates on the error e_k and the input signal vector $\boldsymbol{x}_k$. The transformed error and transformed input vector can be

entered in the NLMS-algorithm such that the following update applies for tap weights in an adaptive filter

$$\boldsymbol{w}_{k+1} = \boldsymbol{w}_k + \eta \frac{g(\boldsymbol{x}_k)g(e_k)}{||g(\boldsymbol{x}_k)||^2 + 1e-6}. \tag{3.9}$$

The normalization $||g(\boldsymbol{x}_k)||^2$ is based on the N-number of samples used in the empirical density transformation

The algorithm when viewed in an adaptive filter-setup can be seen from figure (3.1) page (21). As mentioned earlier no cost function is specified for this method. Instead,

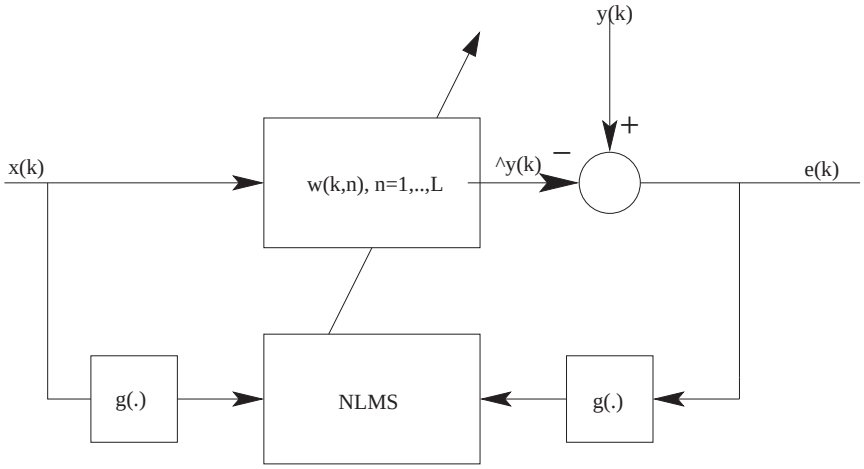


Figure 3.1: Adaptive filter setup using the Gaussian NLMS algorithm

a solution criterion is defined (such as the MO at the solution) from where the tap-updates are found. The tap-weight update equation given in (3.9) will have a stability point at the solution since $M[g(\boldsymbol{x}_k)g(e_k)] = 0$, so $g(\boldsymbol{x}_k) \perp_M g(e_k)$.

It is possible to determine the actual cost function that is minimized [4]. The cost function can be found by integration of the expectation of the stepping in the stochastic gradient algorithm in the limit as the step-size goes to zero. In [20] the cost function is found to be

$$J = E[|g(e)|^2], \tag{3.10}$$

which is minimized when using the Gaussian NLMS algorithm.

The computational workload of the Gaussian NLMS algorithm depends on the number of samples used in the window (N). Basically the larger the window used the better the transformation. This means that the variance of the tap-weights will be smaller when the coefficients have converged.

In the following a discussion on the complexity of the algorithm.

1. *Sorting of N-samples.* N is the number of samples used in the empirical density transformation The first sorting is the most time-consuming one, since all values have to be sorted. For full sorting each time several methods exists. Simple implementations normally requires $\mathcal{O}(N^2)$ operations while more complex algorithms require $\mathcal{O}(Nlog(N))$ operations. However, only the first time a full sorting is necessary since only one sample arrives at each tap-weight update.
2. *Transforming the N sorted samples with the empirical density transformation.* When the algorithm is running, only the new sample which arrives, needs to be transformed. Each transformation require around 5 operations (2 multiplications, 1 division and 2 additions).
3. *Transforming the uniform variables to a Gaussian variables.* Requires a non-linear operation.
4. *Scaling.* Can be done iteratively by removing the oldest sample and adding the transformed new sample.
5. *Update filter coefficients with the NLMS method*
   The complexity of the NLMS method is as discussed earlier in the order of $\mathcal{O}(L)$ operations.

So basically the sorting, and updating are controlling the computational complexity. In the implementation a new sorting is performed with each new sample which means that the computational complexity is $\mathcal{O}(NlogN)$ operations.

3.2 Recursive methods

3.2.1 RLP

The Recursive Least p-norm algorithm is closely related to the recursive Least Square algorithm (RLS). From [12] a family of recursive algorithms developed for alpha stable noise is presented. The RLP algorithm minimizes a cost function similar to the NLMP algorithm. From [12] the cost function associated with the RLP is given as

$$J_k = \sum_{i=0}^{k} \lambda^{k-i} \frac{|e(i)|^p}{p}. \tag{3.11}$$

In the derivations to follow the system model is assumed to be a linear system (transversal filter). λ in equation (3.11) is a forgetting factor in the range $0 < \lambda \leq 1$. If $\lambda = 1$ this corresponds to infinite memory. The memory depth (md) can be calculated as $md = \frac{1}{1-\lambda}$. In a non-stationary environment the forgetting factor should be chosen smaller than one.

To minimize (3.11) with respect to the unknown weights a linear system of equations appear. This linear system takes the form [12]

$$\boldsymbol{Q}_{\boldsymbol{x}\boldsymbol{x}_k}\boldsymbol{w}_k = \boldsymbol{q}_{\boldsymbol{x}\boldsymbol{y}_k}, \tag{3.12}$$

which is solved as

$$\boldsymbol{w}_k = \boldsymbol{Q}_{\boldsymbol{x}\boldsymbol{x}_k}^{-1}\boldsymbol{q}_{\boldsymbol{x}\boldsymbol{y}_k}, \tag{3.13}$$

where $\boldsymbol{Q}_{\boldsymbol{x}\boldsymbol{x}_k}$ is the weighted autocorrelation matrix and $\boldsymbol{q}_{\boldsymbol{x}\boldsymbol{y}_k}$ is the weighted cross-correlation vector at time k. To find the model weights one have to solve the linear system of equations, which using normal methods will be computational expensive. The weighted autocorrelation matrix and weighted cross correlation vector can be updated recursively, which makes use of the matrix inversion lemma[3] ideal. The matrix inversion lemma let us approximate $\boldsymbol{P}_k = \boldsymbol{Q}_{\boldsymbol{x}\boldsymbol{x}_k}^{-1}$ recursively. Some initialization of the matrix is needed. The initialization used is known as "soft constrained initialization" [15]. In [12] the following initialization is given $\boldsymbol{P}_{-1} = \boldsymbol{Q}_{\boldsymbol{x}\boldsymbol{x}_{-1}}^{-1} = \delta^{-1}\boldsymbol{I}$. Using the initial value for the inverse autocorrelation matrix changes the cost function defined in (3.11) to the following cost-function [4]

$$J_k = \sum_{i=0}^{k} \lambda^{k-i}\frac{|e_i|^p}{p} + \delta\lambda^{k+1}||\boldsymbol{w}_k||_2^2 \tag{3.14}$$

The cost function in (3.14) can be verified by finding the gradient with respect to the model weights, and see if this corresponds to the actual initialization of the inverse autocorrelation matrix. So

$$\begin{aligned}
&\tfrac{\partial J_k}{\partial \boldsymbol{w}} = -\textstyle\sum_{i=0}^{k}\lambda^{k-i}v_i e_i \boldsymbol{x}_i + \delta\lambda^{k+1}\boldsymbol{w}_k = 0 \\
&\text{where from [12] the following definition is made} \quad v_i = |e_i|^{p-2} \\
&\text{using that} \quad e_i = y_i - \boldsymbol{w}_k^T\boldsymbol{x}_i \\
&-\textstyle\sum_{i=0}^{k}\lambda^{k-i}v_i y_i \boldsymbol{x}_i + \sum_{i=0}^{k}\lambda^{k-i}v_i \boldsymbol{w}_k^T\boldsymbol{x}_i\boldsymbol{x}_i^T + \delta\lambda^{k+1}\boldsymbol{w}_k = 0 \\
&\text{re-arranging gives} \\
&\textstyle\sum_{i=0}^{k}\lambda^{k-i}v_i y_i \boldsymbol{x}_i = \left[\sum_{i=0}^{k}\lambda^{k-i}v_i \boldsymbol{x}_i\boldsymbol{x}_i^T + \delta\lambda^{k+1}\boldsymbol{I}\right]\boldsymbol{w}_k
\end{aligned} \tag{3.15}$$

In the above $v_i = \frac{\frac{\partial |e_i|^p}{\partial e_i}}{e_i}$. Since e_i requires information of the model weight vector $\boldsymbol{w}_k$ an approximation to e_i is made, the so-called instantaneous a priory estimation error $\xi_i = y_i - \boldsymbol{w}_{k-1}^T\boldsymbol{x}_i$. v_i is then estimated using this a priory estimate:

[3] The matrix inversion lemma as stated in [15]: Let $\boldsymbol{A}$ and $\boldsymbol{B}$ be two positive definite matrices related by $\boldsymbol{A} = \boldsymbol{B}^{-1} + \boldsymbol{C}\boldsymbol{D}^{-1}\boldsymbol{C}^{\boldsymbol{H}}$ where $\boldsymbol{D}$ is another positive definite $N \times M$ matrix and $\boldsymbol{C}$ is an $M \times N$ matrix. According to the matrix inversion lemma $\boldsymbol{A}^{-1}$ may be found as: $\boldsymbol{A}^{-1} = \boldsymbol{B} - \boldsymbol{B}\boldsymbol{C}(\boldsymbol{D} + \boldsymbol{C}^{\boldsymbol{H}}\boldsymbol{B}\boldsymbol{C})^{-1}\boldsymbol{C}^{\boldsymbol{H}}\boldsymbol{B}$.

[4] From [3] this is also known as a weight decay regularization.

$$v_i \approx \omega_i = \frac{\frac{\partial |\xi_i|^p}{\partial \xi_i}}{\xi_i}$$

Identifying the system of linear equations in (3.15) one can write the weighted autocorrelation and the weighted cross-correlation vector as:

$$\boldsymbol{Q}_{\boldsymbol{xx}k} = \sum_{i=0}^{k} \lambda^{k-i} \omega_i \boldsymbol{x}_i \boldsymbol{x}_i^T + \delta \lambda^{k+1} \boldsymbol{I} \tag{3.16}$$

$$\boldsymbol{q}_{\boldsymbol{xy}k} = \sum_{i=0}^{k} \lambda^{k-i} \omega_i y_i \boldsymbol{x}_i \tag{3.17}$$

The equations for the weighted autocorrelation matrix and weighted cross-correlation vector show that the cost function given by (3.14) is valid (insert $\boldsymbol{Q}_{\boldsymbol{xx}-1} = \delta \boldsymbol{I} = \boldsymbol{P}_{-1}^{-1}$).

Given the system to be solved in (3.12) the following steps are needed to solve the system [12] (also similar to the procedure described in [15] for the RLS algorithm):

1. Calculation of a gain vector (denoted with an uppercase letter $\boldsymbol{K}$, because k is selected as a time-index)

$$\boldsymbol{K}_k = \frac{\lambda^{-1} \boldsymbol{P}_{k-1} \boldsymbol{x}_k}{\omega_k^{-1} + \lambda^{-1} \boldsymbol{x}_k^T \boldsymbol{P}_{k-1} \boldsymbol{x}_k} \tag{3.18}$$

2. Tap weight update at time k

$$\boldsymbol{w}_k = \boldsymbol{w}_{k-1} + \xi_k \boldsymbol{K}_k \tag{3.19}$$

3. Update the inverse weighted autocorrelation matrix

$$\boldsymbol{P}_k = \lambda^{-1} \boldsymbol{P}_{k-1} - \lambda^{-1} \boldsymbol{K}_k \boldsymbol{x}_k^T \boldsymbol{P}_{k-1} \tag{3.20}$$

4. When the next sample arrives, repeat from step 1.

It should be mentioned that $\boldsymbol{P}_k = \boldsymbol{Q}_{\boldsymbol{xx}k}^{-1}$ and $\boldsymbol{K}_k = \omega_k \boldsymbol{P}_k \boldsymbol{x}_k$.

In [12] the following limitation is put on ω_i:

$$\omega_i = \begin{cases} |\frac{\xi_i}{\upsilon}|^{p-2} & \text{if } |\xi(i)| > \upsilon \\ 1 & \text{if } |\xi(i)| \leq \upsilon \end{cases} \tag{3.21}$$

The obvious reason for doing so is to avoid that the weighted autocorrelation matrix and cross-correlation vector explodes when small values of ξ_i is present. The recursive update of the autocorrelation matrix and the cross correlation vector can be found from equation (3.16) and equation (3.17) respectively and are given as [12]

$$\boldsymbol{Q}_{\boldsymbol{xx}k} = \lambda \boldsymbol{Q}_{\boldsymbol{xx}k-1} + \omega_k \boldsymbol{x}_k \boldsymbol{x}_k^T \tag{3.22}$$

$$\boldsymbol{q}_{\boldsymbol{xy}k} = \lambda \boldsymbol{q}_{\boldsymbol{xy}k-1} + \omega_k \boldsymbol{x}_k y_k \tag{3.23}$$

where the influence of ω_k is clear if the value "blow up".

The convergence speed of the RLS (RLP with p=2) algorithm is faster than both LMS and NLMS in most cases. Typical problems encountered using the RLS-algorithm have been analyzed of more than one author. In [15] the following problems with the algorithm are identified (quotation from [15]):

> The mean-squared error in the weight vector $\boldsymbol{w}_k$ is magnified by the inverse of the smallest eigenvalue λ_{min}. Hence, to a first order of approximation, the sensitivity of the RLS algorithm to eigenvalue spread is determined initially in proportion to the inverse of the smallest eigenvalue. Therefore, ill-conditioned least squares problems may lead to bad convergence properties.

The property will also exist for the RLP algorithm with $p < 2$ since the weighted autocorrelation matrix can become close to singular, in which the inverse matrix do not exists (some of the singular values will be zero).

The initial values for the recursive weighted autocorrelation matrix was given as the identity matrix multiplied with δ, so the eigenvalues equals δ at initialization. This puts some bounds on the choice of δ. Choosing the value to small, makes the eigenvalues very small (in the initialization phase), and hence makes the problem ill conditioned. Choosing to big a δ will make the weight decay regularization more dominant and the convergence speed will not be as fast (in the beginning). A compromise must be found. In [15] the following selection of δ is suggested $\delta < 0.01\sigma_x^2$ where σ_x^2 is the variance of the input signal.

The convergence speed of the RLP algorithm is also expected to be faster than when using the NLMP algorithm. Also the system mismatch is expected to be lower with the RLP than the tradionally NLMP.

The complexity of the RLP-algorithm is similar to the RLS algorithm $O(L^2)$.

Selection of p

The selection of p in the cost function should be based on the error-samples e. The value of p could either be selected to be constant during the adaption, or one could change p in accordance with the environment. To find the most optimal p value during the adaption phase a recursive α-estimator based on the error-samples are derived. It is assumed that the environment is a $S(\alpha)S$ environment. The reason for selecting p recursively comes from the fact that the RLP method is already a recursive method. This means that the largest weight will be on the new samples. This should in some sense influence the choice of p - value.

[21] is discussing issues on estimating the p-value from the given samples, and propose a NLMP-algorithm with adaptive changing p-value. In [25] various methods to estimate the α (in an $S(\alpha)S$ environment) from the samples are given. We want a method which is not to computationally expensive.

We will base the discussion on assuming that the environment that the algorithms will work in can be approximated with $S(\alpha)S$ distributions. As discussed earlier the *Fractional Lower Order moments* (FLOM) exist for symmetric alpha distributions and is given as

$$E\left[|X|^p\right] = C(p,\alpha)\gamma^{\frac{p}{\alpha}} \quad 0 < p < \alpha \tag{3.24}$$

which is used in determining the α estimator from the data. Assuming that X is $S(\alpha)S$ distributed then the estimator is based on taking the logarithm to the data. Let us denote $Y = \log_e(|X|)$ then it can be shown [25] that

$$var(Y) = \frac{\pi^2}{6}\left(\frac{1}{\alpha^2} + \frac{1}{2}\right) \tag{3.25}$$

from where the α value can be determined.

The variance of Y will be determined in a recursive manner. Defined here at time-sample k

$$\widehat{var(Y)}_k = \mu_k \sum_{n=1}^{k} \lambda^{k-n}\left(y_n - E\left[Y\right]\right)^2 \tag{3.26}$$

$$= \mu_k\lambda \sum_{n=1}^{k-1} \lambda^{k-n-1}\left(y_n - E\left[Y\right]\right)^2 + \mu_k\left(y_k - E\left[Y\right]\right)^2 \tag{3.27}$$

$$= (1-\mu_k)\widehat{var(Y)}_k + \mu_k\left(y_k - E\left[Y\right]\right)^2 \tag{3.28}$$

where $\mu_k = \frac{1-\lambda}{1-\lambda^k}$ and λ (forgetting factor) is selected to be the same as in the RLP-algorithm. A discussion on this recursive estimator can be found in section (4.6.3) page (62) which will also show the step in coming from $\mu_k\lambda$ to $(1-\mu_k)$. To determine the variance we need a measure of the mean value. The mean value will also be determined recursively

$$\widehat{E\left[Y\right]}_k = \mu_k \sum_{n=1}^{k} \lambda^{k-n} y_n \tag{3.29}$$

$$= (1-\mu_k)\widehat{E\left[Y\right]}_{k-1} + \mu_k y_k. \tag{3.30}$$

The above mean-value will be used in the variance estimate of Y. The α estimate can be calculated as

$$\hat{\alpha}_k = \left(\frac{6}{\pi^2}\widehat{var(Y)}_k - \frac{1}{2}\right)^{-0.5}. \tag{3.31}$$

This α-estimate may be very noisy and to make it more smooth, an averaging is performed also using a recursive method. The recursive averaging operation is determined as

$$avg\left[\hat{\alpha}_k\right] = (1-\mu_{p_k})avg\left[\hat{\alpha}_{k-1}\right] + \mu_{p_k}\hat{\alpha}_k \tag{3.32}$$

where $\mu_{p_k} = \frac{1-\lambda_p}{1-\lambda_p^k}$ and λ_p is the forgetting factor for the averaging operation.

It has to be remembered that the proposed estimator only works for $S(\alpha)S$ variables. To test the recursive α estimator a sample X which is $S(\alpha)S$ i.i.d. distributed was generated using three different α values $(1.3, 1.8, 1.1)$. The sample size was $N = 100000$. The identification of the changing moments using the recursive α estimator can be seen from figure (3.2) page (27), where also the true α value is shown for each interval as a black line. From the figure it can be seen that using

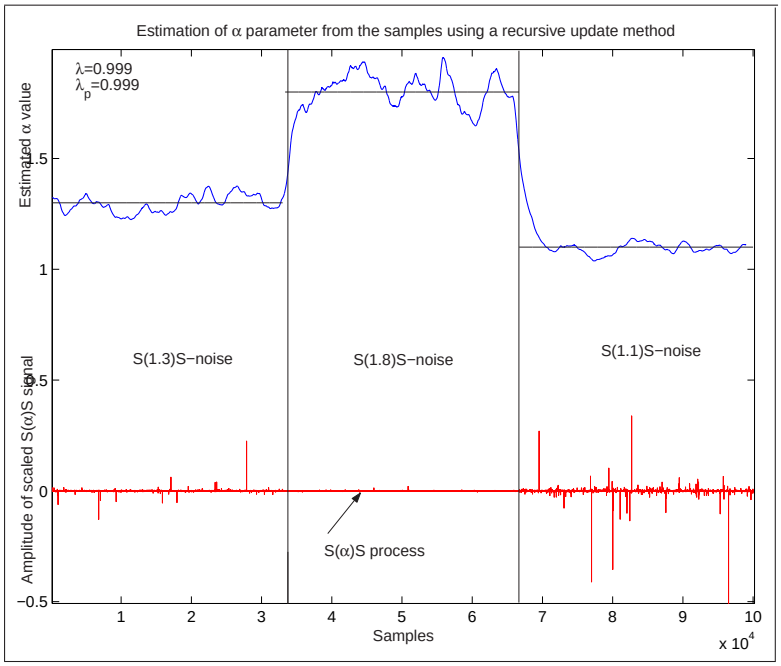


Figure 3.2: Estimation of α of a mixed $S(\alpha)S$ signal using the recursive method.

a forgetting factor of $\lambda = 0.999$ and $\lambda_p = 0.999$ still gives a bit noisy results, but generally the differences in the α values are found. When the system becomes sub-Gaussian an upper limit of $p = 2$ (RLS) will be applied, since the theory only applies to $S(\alpha)S$ variables. It is also known that the second order moment exists for uniform distributed data.

3.3 Discussion

In this section two different cost-functions, and algorithms for minimizing these cost-functions has been proposed. The theory which covers the stochastic gradient approaches have existed for many years, and new theory keeps on coming. The main difference between the methods presented in this section and the section to come (information theoretic methods) are the differences in the cost-function.

Chapter 4

Information Theoretical Methods

In this chapter we will start with a model of the conditional output density $p(y|\boldsymbol{x})$. By minimizing the KL-divergence between our model and the observed distribution will lead to the Shannon generalization error. This generalization error will be transformed to a Renyi generalization error. These two measures are information theoretic measures and gives rise to different investigations. The Parzen window density estimate will be discussed and methods for determining an optimal width parameter will be discussed. In the last section of the chapter three different algorithms: a Batch, stochastic information gradient and a recursive method will be developed / introduced.

4.1 Definition of generalization error for Information Theoretical Methods

The aim of this project will be to model the conditional output density[1]

$$p(y|\boldsymbol{x}) \tag{4.1}$$

where $\boldsymbol{x} \in \mathbb{R}^d$ is a d dimensional input to the system moreover $y \in \mathbb{R}$ is the output. We will denote the modelled conditional output density as

$$p_M(y|\boldsymbol{x}, \boldsymbol{w}) \tag{4.2}$$

where $\boldsymbol{w} \in \mathbb{R}^L$ is the weights of the model to be determined. The M is added to be able to differ between the model and the true conditional output density. The true joint data distribution will be denoted as $p(\boldsymbol{x}, y)$. It should be mentioned that we are not modelling the input distribution $p(\boldsymbol{x})$. A way to measure the difference between the two joint distributions $p(\boldsymbol{x}, y)$ and $p_M(\boldsymbol{x}, y)$, where dependence on the parameters $\boldsymbol{w}$ is understood, is to use the Kullback-Leibler divergence measure, also known as relative entropy. The Kullback-Leibler divergence measure is based on

[1] This subsection is primarily based on a note written by Jan Larsen (2002): "Nonparametric Adaptive Filtering".

Shannon's entropy, therefore share axioms with Shannon's entropy measure. Using the KL divergence [16] measure shows to be a good idea in the aim to determine the model parameters. Minimizing the KL-divergence minimizes the difference between the model joint distribution and the joint data distribution. So

$$
\begin{aligned}
KL\left(p(\boldsymbol{x},y),p_M(\boldsymbol{x},y)\right) &= \int \log\frac{p(\boldsymbol{x},y)}{p_M(\boldsymbol{x},y)}p(\boldsymbol{x},y)d\boldsymbol{x}dy && (4.3)\\
&= \int \log\frac{p(\boldsymbol{x},y)}{p_M(y|\boldsymbol{x},\boldsymbol{w})p(\boldsymbol{x})}p(\boldsymbol{x},y)d\boldsymbol{x}dy && (4.4)\\
&= -\int \log\left(p_M(y|\boldsymbol{x},\boldsymbol{w})\right)p(\boldsymbol{x},y)d\boldsymbol{x}dy \\
&\quad -\int \log\left(p(\boldsymbol{x})\right)p(\boldsymbol{x},y)d\boldsymbol{x}dy \\
&\quad +\int \log\left(p(\boldsymbol{x},y)\right)p(\boldsymbol{x},y)d\boldsymbol{x}dy && (4.5)\\
&= G + H(\boldsymbol{x}) - H(y,\boldsymbol{x}) && (4.6)\\
&= G - H(y|\boldsymbol{x}) && (4.7)
\end{aligned}
$$

Where G will is denoted as the generalization error and $H(y|\boldsymbol{x})$ is the conditional entropy[2]. In this project upper-case letters will be used to denote both the entropy of discrete and continuous random variables. The aim is as mentioned earlier to minimize the KL-divergence measure with respect to the model parameters, so

$$
\begin{aligned}
\min_{\boldsymbol{w}} KL(p(\boldsymbol{x},y),p_M(\boldsymbol{x},y) &= \min_{\boldsymbol{w}}\left(G - H(y|\boldsymbol{x})\right) && (4.8)\\
&= \min_{\boldsymbol{w}} G && (4.9)
\end{aligned}
$$

due to the fact that $H(y|\boldsymbol{x})$ is independent of the model parameters $\boldsymbol{w}$. Since we have assumed an additive noise model where the additive noise is assumed i.i.d.

$$
y = f(\boldsymbol{x},\boldsymbol{w}) + e \tag{4.10}
$$

then substituting this into $p_M(y|\boldsymbol{x},\boldsymbol{w})$ gives

$$
\begin{aligned}
p_M(y|\boldsymbol{x},\boldsymbol{w}) &= p(f(\boldsymbol{x},\boldsymbol{w}) + e|\boldsymbol{x},\boldsymbol{w}) && (4.11)\\
&= p_e(e|\boldsymbol{x},\boldsymbol{w}), && (4.12)
\end{aligned}
$$

where $p_e(e)$ is the noise distribution. The generalization error to be minimized can from (4.5) be written as

$$
G = -\int \log\left(p_e(e|\boldsymbol{x},\boldsymbol{w})\right)p(\boldsymbol{x},y)d\boldsymbol{x}dy \tag{4.13}
$$

The generalization error given in (4.13) have similarities with the maximum likelihood principle which will be shown in the following. In a maximum likelihood setup

[2] In [16] the term differential entropy is the word used for entropy of continuous variables which as well is denoted with a lowercase letter (h)

we consider the density function $p(y|\boldsymbol{x}, \boldsymbol{w})$ which depends on the model parameters and some given $\boldsymbol{x}$. Given a data-set $D = \{(\boldsymbol{x}_1, y_1), (\boldsymbol{x}_2, y_2), .., (\boldsymbol{x}_N, y_N)\}$ which is drawn independently from the distribution $p(y|\boldsymbol{x}, \boldsymbol{w})$ then the joint probability density of the data set will be given as [3]

$$p(D|\boldsymbol{w}) = \prod_{n=1}^{N} p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) \equiv L(\boldsymbol{w}) \tag{4.14}$$

where $L(\boldsymbol{w})$ is the likelihood of $\boldsymbol{w}$ given the data samples D. The technique of maximum likelihood is to maximize $L(\boldsymbol{w})$ with respect to the model parameters, hence make the model-parameters most likely to the given data set D.

Similarities between Maximum likelihood and Generalization error

Instead of maximizing the likelihood given in (4.14) one normally minimizes the negative logarithm of the likelihood, so

$$E = -\log\left(L(\boldsymbol{w})\right) = -\sum_{n=1}^{N} \log p(y_n|\boldsymbol{x}_n, \boldsymbol{w}) \tag{4.15}$$

Normally one assumes to know the density function (parametric).

From the generalization error given in (4.13) one can insert an empirical density estimate of $p(\boldsymbol{x}, y)$ (will be discussed in subsection (4.4.2) page (48))

$$\hat{p}(\boldsymbol{x}, y) = \frac{1}{N} \sum_{n=1}^{N} \delta(\boldsymbol{x} - \boldsymbol{x}_n)\delta(y - y_n) \tag{4.16}$$

of $p(\boldsymbol{x}, y)$ which would give

$$\widehat{G}_{ML}(\boldsymbol{w}) = -\frac{1}{N} \sum_{n=1}^{N} \int \log\left(p_e(e|\boldsymbol{x}, \boldsymbol{w})\right) \delta(\boldsymbol{x} - \boldsymbol{x}_n)\delta(y - y_n) d\boldsymbol{x} dy \tag{4.17}$$

$$= -\frac{1}{N} \sum_{n=1}^{N} \int \log\left(p(y|\boldsymbol{x}, \boldsymbol{w})\right) \delta(\boldsymbol{x} - \boldsymbol{x}_n)\delta(y - y_n) d\boldsymbol{x} dy \tag{4.18}$$

$$= -\frac{1}{N} \sum_{n=1}^{N} \log\left(p(y_n|\boldsymbol{x}_n, \boldsymbol{w})\right) = NE, \tag{4.19}$$

which is just a scaling of the maximum likelihood procedure.

4.2 Getting from Shannon generalization error to Renyi Generalization error

Motivated by the work of Jose C. Principe [18] [19] instead of using Shannon entropy measure the Renyi entropy measure is introduced. It is shown in this section that the Renyi generalization error can be written as

$$G_R = \frac{1}{1-\alpha} \log \int_{-\infty}^{\infty} p_e(e|\boldsymbol{x}, \boldsymbol{w})^{\alpha-1} p(\boldsymbol{x}, y) d\boldsymbol{x} dy \tag{4.20}$$

where in this content the generalization error will only be investigated for $\alpha \geq 1$. In the limit as α goes to one, the Shannon generalization error is equal to the Renyi generalization error. Introducing the Renyi generalization error instead of using Shannon generalization error is partly due to simplicity when $\alpha = 2$ but also due to a different weighting of the probability density function $p_e(e|\boldsymbol{x}, \boldsymbol{w})$. Shannon generalization error weights the low probability higher than Renyi entropy ($\alpha > 1$) due to the log. Renyi generalization error is weighting the more probable event higher, in which small probabilities have low weights. This is mentioned in [18] and [19].

Since Shannon and Renyi entropy share properties, which is discussed in section (4.3) page (37), the minimization of Renyi generalization error will also minimize Shannon generalization error.

To determine the Renyi generalization error one have to take a more general approach from the general theory of means [19]. In the following discrete probabilities are used but can be generalized to continuous distributions. In [19] a generalized mean of real numbers is given as

$$\bar{x} = \varphi^{-1} \left(\sum_{k=1}^{N} p_k \varphi(x_k) \right) \tag{4.21}$$

where $\varphi(x)$ is called the Kolmogorov-Nagumo function. This function is arbitrary continuous and a strictly monotonic function defined on the real numbers. Since Shannon entropy is an average of Hartley's information measure: $I(p_k) = -\log_2(p_k)$ a more general entropy measure can be written using (4.21) and Hartley's information measure

$$H(x) = \varphi^{-1} \left(\sum_{k=1}^{N} p_k \varphi(-\log_2(p_k)) \right). \tag{4.22}$$

Because information should be additive $\varphi(.)$ cannot be selected arbitrarily. A couple of functions which fulfills the additivity property is [19] $\varphi(x) = x$ and $\varphi(x) = 2^{(1-\alpha)x}$. Selecting $\varphi(x) = x$ results in Shannon's entropy since the mapping function is linear. Using $\varphi(x) = 2^{(1-\alpha)x}$ the mapping is nonlinear and leads to Renyi's entropy

$$\varphi(x) = 2^{(1-\alpha)x} \leftrightarrow \varphi(x)^{-1} = \frac{1}{1-\alpha} \log_2(\varphi(x)) \tag{4.23}$$

Inserting Hartley's information measure into (4.21) and using the above expression gives

$$\begin{aligned} H_R(x) &= \frac{1}{1-\alpha}\log_2\left[\sum_{k=1}^{N} p_k 2^{-(1-\alpha)\log_2(p_k)}\right] \\ &= \frac{1}{1-\alpha}\log_2\left[\sum_{k=1}^{N} p_k 2^{-(1-\alpha)\log_2(p_k)}\right] \\ &= \frac{1}{1-\alpha}\log_2\left[\sum_{k=1}^{N} p_k p_k^{\alpha-1}\right]. \end{aligned} \tag{4.24}$$

This result is the Renyi entropy, which can be generalized to continuous variables

$$H_R(x) = \frac{1}{1-\alpha}\log_2\int_{-\infty}^{\infty} p(\xi)p(\xi)^{\alpha-1}d\xi. \tag{4.25}$$

The above calculations show that the difference between Renyi's entropy and Shannon's entropy are the way in which the Hartley information measure is averaged.

The generalization error can be regarded as a kind of averaging with respect to $p(\boldsymbol{x}, y)$:

$$G = -\int \log p(e|\boldsymbol{x}, \boldsymbol{w})p(\boldsymbol{x}, y)d\boldsymbol{x}dy \tag{4.26}$$

$$G = -E_{\boldsymbol{x},y}\left[\log p(e|\boldsymbol{x}, \boldsymbol{w})\right]. \tag{4.27}$$

Instead of averaging using $\varphi(x) = x$ in (4.27) one can use the nonlinear mapper $\varphi(x) = 2^{(1-\alpha)x}$ which leads to the Renyi generalization error

$$G_R = \frac{1}{1-\alpha}\log_2 E_{\boldsymbol{x},y}\left[p(e|\boldsymbol{x}, \boldsymbol{w})^{\alpha-1}\right] \tag{4.28}$$

$$= \frac{1}{1-\alpha}\log_2\int_{-\infty}^{\infty} p_e(e|\boldsymbol{x}, \boldsymbol{w})^{\alpha-1}p(\boldsymbol{x}, y)d\boldsymbol{x}dy. \tag{4.29}$$

As in [11] the Renyi generalization error can be written as

$$G_R = \frac{1}{1-\alpha}\log_2 V_\alpha(\boldsymbol{w}) \tag{4.30}$$

where $V_\alpha(\boldsymbol{w})$ is the information potential given as

$$V_\alpha(\boldsymbol{w}) = \int_{-\infty}^{\infty} p_e(e|\boldsymbol{x}, \boldsymbol{w})^{\alpha-1}p(\boldsymbol{x}, y)d\boldsymbol{x}dy. \tag{4.31}$$

The generalization error do not have to be determined in the $\log_2(\cdot)$ base. In this project the natural logarithm is used as a base.

4.2.1 Similarities between measures using Jensen's Inequality and L'Hospitals rule

Using Jensen's Inequality it can be shown that the generalization error of Shannon and Renyi equals when the error distribution is uniform distributed. In the limiting case of $\alpha = 1$ it can be shown using L'Hospitals rule that Renyi's generalization error will equal Shannon's generalization error.

First a definition of *Jensen's Inequality*[9]

> *The inequality*
>
> $$f(\sum_i \lambda_i a_i) \leq \sum_i \lambda_i f(a_i) \tag{4.32}$$
>
> *whenever* $\sum \lambda_i = 1$ *and* $\lambda_i \leq 1$, *that is satisfied by all convex combinations of points in the domain of a convex function, and is equivalent to the convexity of the function f.*

Using an empirical density estimate for $p(\boldsymbol{x}, y)$[3] in the generalization error expression given by (4.28) and (4.13) (Renyi and Shannon) gives the following unbiased expressions of the generalization error

$$\hat{G}_S = -\frac{1}{N}\sum_k log\left(p(e_k)\right) \tag{4.33}$$

$$\hat{G}_R = \frac{1}{1-\alpha} log\left(\frac{1}{N}\sum_k p(e_k)^{\alpha-1}\right). \tag{4.34}$$

Using Jensen's inequality identifying that $\lambda_k = 1/N$ for all k and that $a_k = p(e_k)^{\alpha-1}$ then inserting $\hat{G}_R$ in (4.32) gives

$$\begin{aligned} &\frac{1}{1-\alpha} log\left(\frac{1}{N}\sum_k p(e_k)^{\alpha-1}\right) \leq \frac{1}{1-\alpha}\frac{1}{N}\sum_k log\left(p(e_k)^{\alpha-1}\right) \\ &\frac{1}{1-\alpha} log\left(\frac{1}{N}\sum_k p(e_k)^{\alpha-1}\right) \leq -\frac{1}{N}\sum_k log\left(p(e_k)\right) \quad \text{for } \alpha > 1 \end{aligned} \tag{4.35}$$

From (4.35) it can be seen that $\hat{G}_R \leq \hat{G}_S$. The equality is fulfilled in the limiting case $\alpha \to 1$, and when the probabilities equal each other, namely when $p(e_1) = p(e_2) = p(e_3) = \ .... \ = p(e_N)$. This requires that $p(e)$ is uniformly distributed. To show that this also applies in the continuous case we let the sample-size increase to infinity in the expression given by (4.35) since in the limiting case the empirical density estimate gives the true joint probability density $p(\boldsymbol{x}, y)$ and hence transforming the sums to integrals [4]

[3] $p(\boldsymbol{x}, y) = \frac{1}{N}\sum_i \delta(\boldsymbol{x} - \boldsymbol{x}_i)\delta(y - y_i)$

[4] $$p(\boldsymbol{x}, y) = \lim_{N\to\infty} \frac{1}{N}\sum_k \delta(\boldsymbol{x} - \boldsymbol{x}(k))\delta(y - y(k))$$

$$\lim_{N\to\infty} \frac{1}{1-\alpha} log\left(\frac{1}{N}\sum_k p(e_k)^{\alpha-1}\right) \leq \lim_{N\to\infty} -\frac{1}{N}\sum_k log\left(p(e_k)\right)$$
$$\frac{1}{1-\alpha}\log\int_{-\infty}^{\infty} p_e(e|\boldsymbol{x},\boldsymbol{w})^{\alpha-1}p(\boldsymbol{x},y)d\boldsymbol{x}dy \leq -\int_{-\infty}^{\infty} log\left(p_e(e|\boldsymbol{x},\boldsymbol{w})\right)p(\boldsymbol{x},y)d\boldsymbol{x}dy. \tag{4.36}$$

This means that $G_R \leq G_S$ for $\alpha > 1$ and the two expression are equal when $p(e|\boldsymbol{x}, y)$ is uniform distributed (or $\alpha = 1$). When the error distribution is not uniform distributed Renyi generalization error is smaller in value than Shannon generalization error.

In the limiting case when $\alpha \to 1$ both the numerator and denominator in (4.28) will go towards zero. Using L'Hospitals rule, we can determine the result in the limit:

Denote

$$t(\alpha) = \log\int p(e)^{\alpha-1}p(\boldsymbol{x},y)d\boldsymbol{x}dy \tag{4.37}$$
$$d(\alpha) = 1-\alpha \tag{4.38}$$

where the derivatives with respect to α are determined as

$$t'(\alpha) = \frac{1}{\int p(e)^{\alpha-1}p(\boldsymbol{x},y)d\boldsymbol{x}dy}\int \log p(e)p(e)^{\alpha-1}p(\boldsymbol{x},y)d\boldsymbol{x}dy \tag{4.39}$$
$$d'(\alpha) = -1. \tag{4.40}$$

Using L'Hospitals rule gives in the limit

$$\lim_{\alpha\to 1}\frac{t(\alpha)}{d(\alpha)} = \frac{t'(1)}{d'(1)} = -\int \log p(e)p(\boldsymbol{x},y)d\boldsymbol{x}dy = G_S \tag{4.41}$$

Which shows that $G_R = G_S$ in the limit as $\alpha \to 1$. This tells us that Renyi entropy contains Shannon entropy.

4.3 A discussion on properties of the entropy measures

Since we in this project focus on the Renyi generalization error instead of the Shannon generalization error, the properties of the two entropy measures are discussed here. A reasonable description of Renyi and Shannon entropy was found in [13] where the description is given using discrete random variables. For a random variable X taking K distinct values $X = \{x_1, x_2, x_3...., x_k\}$ with probability $p(X) = \{p_1, p_2..., p_K\}$ the Shannon and Renyi entropy are given respectively as

$$H(X) = -\sum_k p_k \log p_k, \tag{4.42}$$

$$H_R(X) = \frac{1}{1-\alpha} \sum_k p_k^\alpha \quad \text{where } \alpha > 0 \tag{4.43}$$

where the probability density functions should fulfill: $0 \leq p_k \leq 1$ and $\sum_{k=1}^{K} p_k = 1$.

With Renyi entropy using $\alpha > 1$ events with high probabilities are weighted more than events with low probability due to the power in the sum (or integral in terms of continuous variables).

The Shannon and Renyi entropy share the following properties[13]

1. The entropy measures are nonnegative for any arbitrary distribution $p(X)$.
2. The measures are strictly positive except for the certain event, in which both measures equals zero ($H = 0$).
3. Adding an event with zero probability do not give any new information, so $H(p_1, p_2, p_3, 0) = H(p_1, p_2, p_3)$.
4. Both measures achieves a maximum when events are equally likely to happen meaning that $p_1 = p_2 = ... = p_K = 1/K$. In the continuous case this corresponds to the uniform distribution.
5. Each measure is concave for an arbitrary probability function $p(X)$.

Further, the Renyi entropy is a monotonically decreasing function of α for any distribution function $p(X)$.

The Renyi and Shannon entropy measure differ in their additivity properties. The Shannon entropy of a composite event equals the sum of the conditional and marginal entropies. This is given as

$$H(X, Y) = H(Y) + H(X|Y) = H(X) + H(Y|X). \tag{4.44}$$

This is not fulfilled by Renyi entropy. Given that the events X and Y are independent the composite events gives the standard additivity

$$H(X, Y) = H(X) + H(Y) \tag{4.45}$$

which is fulfilled for Renyi entropy. So basically Renyi and Shannon entropy will differ when the events are dependent. In appendix (A.2) page (111) the additivity is shown for the continuous case.

Some properties, which holds for both Shannon and Renyi entropy regarding scaling and translation is summarized here

Translation property
$H(X + c) = H(X)$ where $c \in \mathcal{R}$ is an arbitrary constant.

Scaling property
$H(aX) = H(X) + \log |a|$ where $a \in \mathcal{R}$.

This is derived in appendix (A.2) page (111) for continuous variables.

4.4 Nonparametric density estimates

4.4.1 Parzen Windows

The Parzen estimator [27] is known as a non-parametric density estimate and is a kernel method. The primary reason for selecting a Parzen estimator is its nice analytical properties [10]. One of the problems using this kind of estimators though will be the selection of a proper kernel-width [3], which can be done in several ways.

Assume we have a given sequence of independent identically distributed random variables $x_1, x_2, x_3, .., x_N$ with common probability density function $p(x)$, then an estimate of the underlying distribution is given as [27]

$$\hat{p}(x) = \frac{1}{N\sigma} \sum_{k=1}^{N} \kappa \left(\frac{x - x_k}{\sigma} \right) \tag{4.46}$$

where σ is the width parameter and $\kappa(\cdot)$ is the chosen kernel (also called weighting function). The method is placing kernels on each data point with a weighting controlled by the chosen kernel and kernel width. This weighted sum gives an estimate which will be dependent on both the kernel-function and on the width parameter σ. The Parzen estimate given in (4.46) is asymptotically unbiased if σ is chosen as a function of k in the following way

$$\lim_{k \to \infty} \sigma(k) = 0 \tag{4.47}$$

then

$$\lim_{k \to \infty} E\left[\hat{p}(x)\right] = p(x). \tag{4.48}$$

When selecting a width parameter of zero, this corresponds to placing delta-function on each sample, similar to an empirical density estimate (will be discussed in the next subsection). To see what influence the kernel have on the PDF estimate the expectation of the PDF-estimate is performed

$$E\left[\hat{p}(x)\right] = E\left[\frac{1}{\sigma}\kappa\left(\frac{x-y}{\sigma}\right)\right] = \int_{-\infty}^{\infty} \frac{1}{\sigma}\kappa\left(\frac{x-y}{\sigma}\right) p(y)dy, \tag{4.49}$$

which turns out to be a convolution between the true density, here denoted by $p(y)$, and the kernel function.

Different kernels exist and Parzen suggested different kernels himself. A kernel have

Kernel Function	$\kappa(x)$
Uniform	$\begin{matrix} \frac{1}{2}, & \lvert x\rvert \leq 1 \\ 0, & \lvert x\rvert > 1 \end{matrix}$
Gaussian	$(2\pi)^{-1/2} e^{-\frac{1}{2}x^2}$
Cauchy	$\pi^{-1}(1+x^2)^{-1}$

Table 4.1: Different kernels

to be normalized in order to be a proper kernel[5]

$$\int_{-\infty}^{\infty} \kappa(x)dx = 1 \tag{4.50}$$

Table (4.1) page (38) show three different kernels. The uniform kernel is a so-called compact kernel. In this report the Gaussian kernel will be considered due to its nice mathematical properties.

Selection of width parameter for univariate data

The width parameter is more difficult to determine since we normally do not know the underlying distribution of the data that we are modelling. (non-parametric approach). The determination of the width parameter has to depend solely on the measurable data. When considering estimation at a single point a natural measure is given as the mean square error ([27] and [32])

$$MSE_x\{\hat{p}\} = E\left\{[\hat{p}(x) - p(x)]^2\right\} \tag{4.51}$$

This measure is denoted as a local measure. It is possible to rewrite the mean square error into an expression given as a bias term and a variance term

$$MSE\{\hat{p}(x)\} = E\left[(\hat{p}(x) - E\left[\hat{p}(x)\right])^2\right] + \left(E\left[E\left[\hat{p}(x)\right] - p(x)\right]\right)^2 \tag{4.52}$$

$$= var\left[\hat{p}(x)\right] + \left(bias\left[\hat{p}(x)\right]\right)^2. \tag{4.53}$$

Selecting a to small width parameter will increase the variance of the estimate while selection of a too big width parameter will give a big bias (smooth). In [32] a global measure of the accuracy of $\hat{p}(\cdot)$ on $p(\cdot)$ is denoted as the *mean integrated square error* (MISE) and is defined as

$$MISE\{\hat{p}\} = E\left[\int_{-\infty}^{\infty} \{\hat{p}(x) - p(x)\}^2 dx\right]. \tag{4.54}$$

This expression can be written as an integrated $bias^2$ and an integrated variance term.

To get an understanding of the influence of the kernel-size on the estimation of a unknown density three different test cases was put up. Gaussian distributed (N(0,1)), uniform distributed (width=1) and Cauchy distributed (unit dispersion) samples was generated (sample size N=100) and different width parameters was tested in the three cases using Gaussian kernels. The Gaussian distribution as well as the estimates is shown in figure (4.1) page (40). Figure (4.2) page (40) shows the uniform distributed estimates. The density estimates of the Cauchy distribution can be seen from figure (4.3) page (41). It is clear from all three examples that the selection of the width parameter in the Parzen-estimate will control the $MISE$ defined earlier. The density estimation when the width parameter is chosen to small will be very accurate but

[5] In addition, other functional properties should be fulfilled[27]

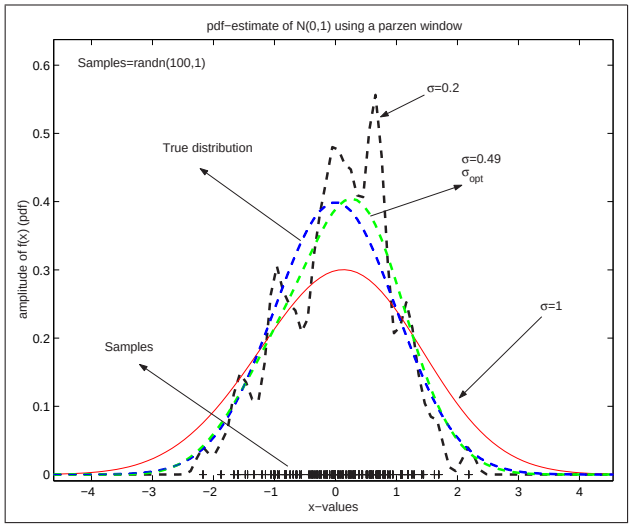


Figure 4.1: Estimation of a $N(0,1)$ distribution using various kernel sizes.

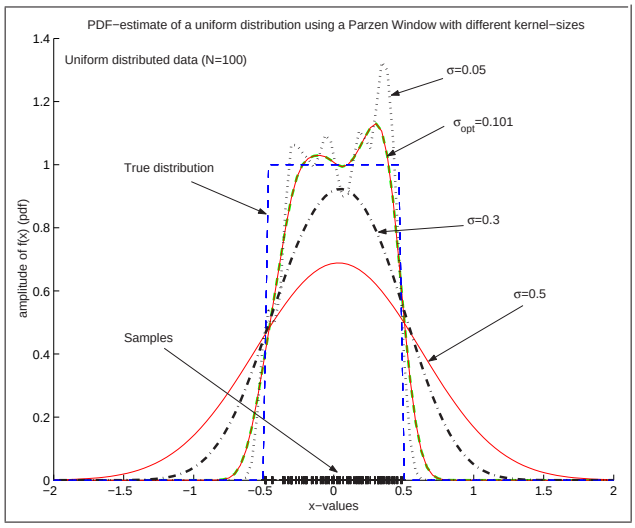


Figure 4.2: Estimation of a uniform distribution using various kernel sizes

very noisy, hence the variance is big. If the width parameter is chosen to large, the density is over-fitted, and the bias will increase in size. On the three figures an optimal value is given. This value is determined from the "Rule of Thumb" method, which will be discussed in the next subsection.

The Rule of Thumb method

By demanding that the kernel used in the Parzen density estimate is symmetric and that the unknown density $p(x)$ has continuous derivatives of all orders the integrated $bias^2$ and the integrated variance term can be approximated using Taylor expansion. Using Taylor expansion of the integrated $bias^2$ and integrated variance

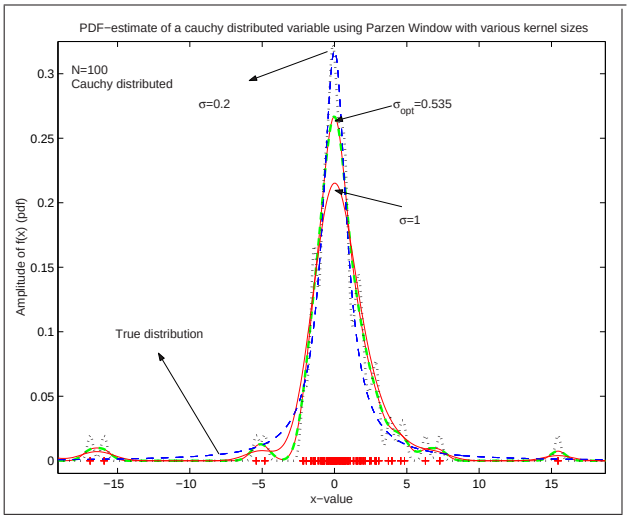


Figure 4.3: Estimation of a Cauchy distribution with unit dispersion using various kernel sizes.

an approximate expression for the width parameter can be found as [32]

$$\sigma_{opt} = \left[\frac{\int [\kappa(t)]^2 \, dt}{k_2^2 \int [p''(x)]^2 \, dx} \right]^{1/5} N^{-1/5} \tag{4.55}$$

where $k_2^2 = \int t^2 \kappa(t) dt$ and N is the number of samples used in the width estimate. In order to get an estimate of the optimal width parameter the unknown distribution should be known which of course put some limitations on this method. Selecting a Gaussian kernel and assuming that $p(x) : N(\mu, \sigma^2)$ gives an optimal width of the kernels of

$$\sigma_{opt} = 1.06 \sigma N^{-1/5}. \tag{4.56}$$

It should be noted that since this expression depends on N the expression given in (4.47) is fulfilled. Silverman [32] proposed a more robust measure where an adaptive estimate of the spread is used. The selection of kernel widths should be done according to

$$\sigma = 0.9 A N^{-1/5} \tag{4.57}$$

where

$$A = min\{standarddeviation, interquartilerange/1.34\} \tag{4.58}$$

where the interquartile range[6] is determined as $IQR = Q3 - Q1$, where $Q1$ is the 25 quantile of a distribution and $Q3$ is the 75 quantile. The measure of spread using the interquartile range is a robust measure. It is noted in [32] that: *"This will cope well with the unimodal densities and will not do too badly if the density is moderately bimodal"*.

From [35] the interquartile range can be determined as :

[6] Interquartile Range / 1.34 is a robust estimation of the spread.

1. Determine the statistical median of the data and denote a low and a high group (below and above the median)
2. Determine the median of the low and high group, and denote them as the Q1 and Q3 quartile respectively.
3. Calculate the interquartile range as $\widehat{IQR} = Q3 - Q1$

The complexity of this method will depend on the sorting algorithm used, since this will be needed to estimate the IQR from the data. A well-implemented sorting algorithm will have a complexity of $\mathcal{O}(NlogN)$ operations.

Likelihood cross-validation

Another method, which among others has been considered [32] is called the likelihood cross-validation method. The method relies on the fact that beside the data-set (which we want to determine the width parameter of) an additional measurement Y sampled from the unknown density is available. This is sampled independent from the other samples. The likelihood of the unknown density would then be $-log(p(Y|\sigma))$. Using a Parzen estimate of the unknown density would give a likelihood of $-log(\hat{p}(Y|\sigma)$. Since we do not have any independent observations of the samples, one of the samples (x_i) from the data-set can be used as an independent observation. This gives the log-likelihood $-log(\hat{p_{-i}}(x_i|\sigma))$ where the notation $\hat{p}_{-i}$ means that sample x_i is left out in the density estimate [7]. Since we do not know whether to leave sample i or $i+1$ out, an average of the samples is taken. The cost-function to be minimized with respect to the width parameter σ will be [32]

$$CV(\sigma) = -\frac{1}{N}\sum_{i=1}^{N} log\hat{p}_{-i}(x_i|\sigma) \tag{4.59}$$

where

$$\hat{p}_{-i}(x_i|\sigma) = \frac{1}{(N-1)\sigma}\sum_{k=1,k\neq i}^{N} \kappa\left(\frac{x_i - x_k}{\sigma}\right). \tag{4.60}$$

Noted in [32]: the method is very sensitive to outliers in the data. To give an explanation to this suppose that a sample x_i is an outlier. Using a Gaussian kernel in the estimate then $\hat{p}(x_i)$ will be very small (very close to zero). The "log" of this result will tend towards $-\infty$, hence the cost-function $CV(\sigma)$ will be big. This depends on the outlier and chosen kernel-function. So as to minimize $CV(\sigma)$ a rather large value of σ is needed which will result in an under-fit of the wanted distribution (big bias).

The complexity of the this method will be $\mathcal{O}(N^2)$ since the Parzen estimate requires $\mathcal{O}(N)$ evaluations and in the cross-validation formulation N kernel evaluations are needed.

[7] The method is also denoted as the Leave-One Out method

Comparison of the two methods

To test the two methods a sample size of 100 was drawn from three different unimodal distributions : a Gaussian, Uniform and a Cauchy distribution ($S(1)S$) as in the example before. Instead of comparing various width values only the optimal values given by the "Rule of Thumb" method and the "likelihood cross-validation" method were used in these plots. Only one realization of the data was used in this PDF-graph comparison. The estimated MISE is estimated for several runs in the end of this subsection. Figure (4.4) page (43) shows the two PDF-estimates of Gaussian distributed data. The $CV(\sigma)$ (log-likelihood) can be seen from figure (4.5) page (44). The minimum of the cross validation method do not seem to be very peaked (a flat minimum) and the solution given by the "Rule of Thumb" is not that far from the cross-validation method. Several runs of the test with different data showed quite similar performance of the two methods. The optimal value for Gaussian distributed data can be determined to $\sigma_{opt} \approx 0.42$.

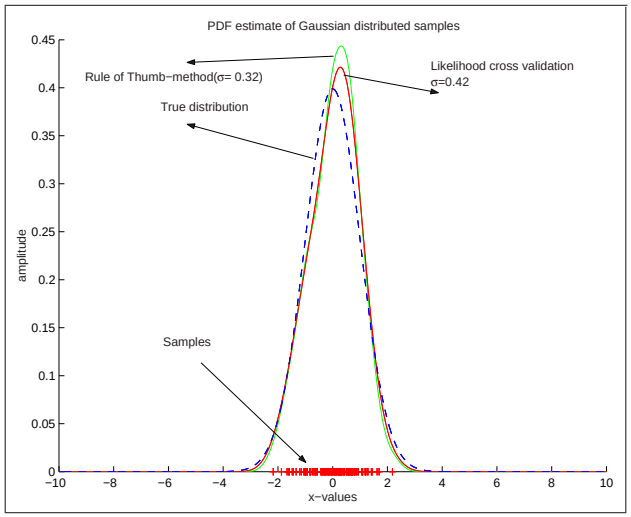


Figure 4.4: Gaussian distributed samples estimated using Parzen windows.

Identification of the underlying distribution using uniform distributed data can be seen from figure (4.6) page (44). The estimate using the width parameter from the cross validation log-likelihood method is more over-fitted than the estimate given by the "Rule of Thumb" method. The CV log-likelihood as a function of width parameter figure (4.7) page (45), show that the minimum of the two methods are not that far apart in absolute value. In the last example where Cauchy-distributed data is used, the problem of the CV log-likelihood method is apparent, see figure (4.8) page (45). The width parameter is chosen much higher than the width parameter using the "rule of thumb"-method. The CV log-likelihood estimate is highly biased, and do not follow the peak as well as the "Rule of Thumb" method. The CV log-likelihood as a function of width parameter, see figure (4.9) page (46), shows a not so well defined minimum. The reason for this high choice of width parameter was discussed earlier under the likelihood cross validation section. Since we only evaluated one run, the Mean Integrated Square Error (MISE) have been estimated for the different runs by

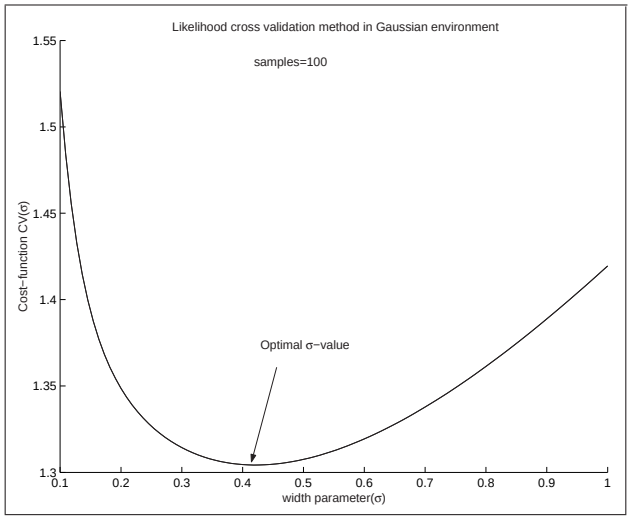


Figure 4.5: Cross validation log-likelihood as a function of width parameter.

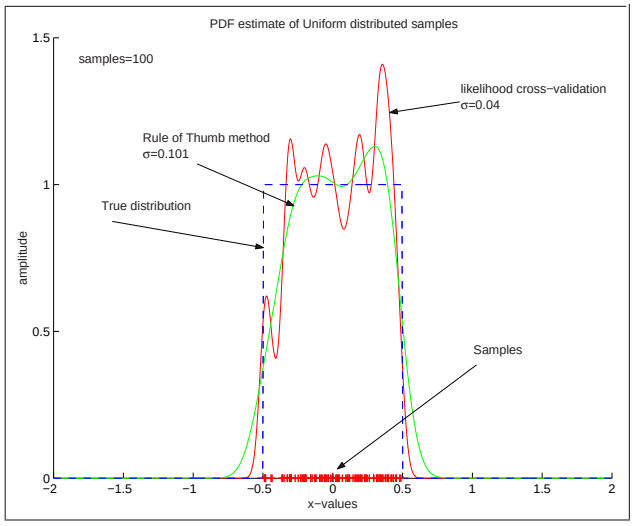


Figure 4.6: Uniform distributed samples estimated using Parzen windows.

splitting the expression into a $bias^2$ and a variance part. In the following the $bias^2$ and integrated variance have been estimated for the cross validation and "Rules of thumb" method. From [32]:

$$MISE(\hat{p}) = E\left[\int \left(\hat{p}(x) - p(x)\right)^2 dx\right] \tag{4.61}$$

$$= \int \left\{E\left[\hat{p}(x)\right] - p(x)\right\}^2 dx + \int var\{\hat{p}(x)\} dx \tag{4.62}$$

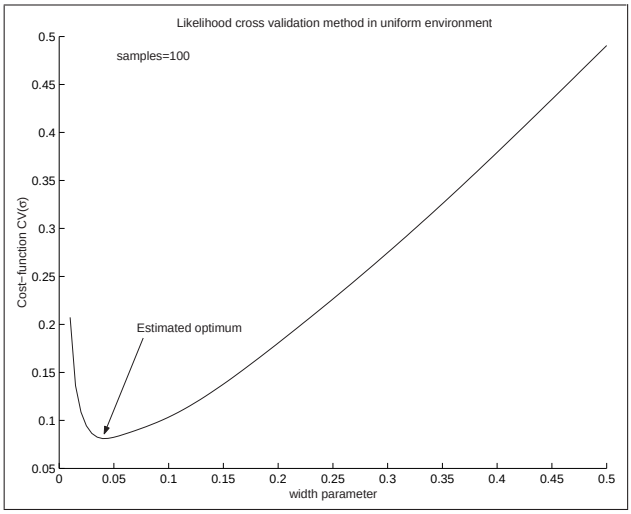


Figure 4.7: Log-likelihood as a function of width parameter

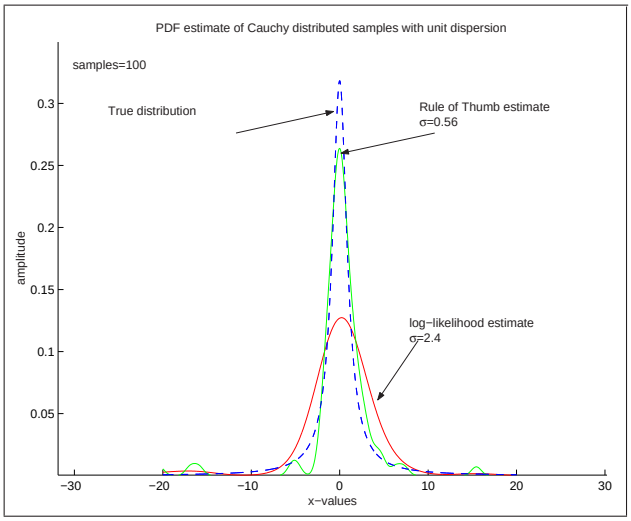


Figure 4.8: Cauchy distributed variables. PDF estimated using Parzen windows

where the integral is estimated using a simple numerical scheme

$$MISE(\hat{p}) \approx \Delta x \sum_{n=0}^{N} \left\{ \{E\left[\hat{p}(x_n)\right] - p(x_n)\}^2 + var\{\hat{p}(x_n)\} \right\} \tag{4.63}$$

$$\widehat{MISE}(\hat{p}) = \Delta x \sum_{n=0}^{N} \left\{ \left[\frac{1}{K} \sum_{j=0}^{K} \hat{p}_j(x_n) - p(x_n) \right]^2 + var\{\hat{p}(x_n)\} \right\} \tag{4.64}$$

$$\widehat{MISE}(\hat{p}) = I_{bias^2} + I_{var} \tag{4.65}$$

where $\Delta x = \frac{max_x - min_x}{N}$ and $min_x \in \mathcal{R}$ and $max_x \in \mathcal{R}$ is the window in which the density estimate is compared to the true density estimate.

In the following I_{bias^2} and I_{var} will be compared for the two methods. The results

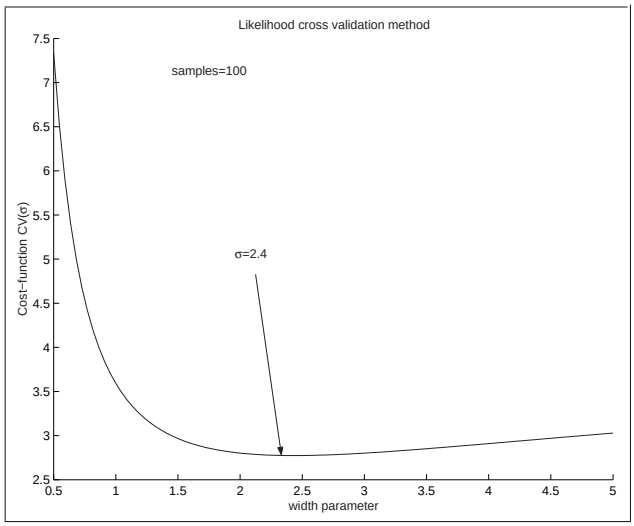


Figure 4.9: Cross validation log-likelihood as a function of the width parameter

of the different setups (uniform, Gaussian and Cauchy environment) can be seen from table (4.2) page (46), (4.3) page (47) and (4.4) page (47). The simulations was performed as follows

1. Determine optimal width parameter for Parzen windows using the "Rule of thumb" method and cross validation method. Out of 20 runs, the mean value was determined and used.
2. calculate I_{bias^2} and I_{var} for the two methods
3. Estimate the mean kernel-width value and the spread on the mean kernel-width value from these 20 runs.

The PDF-estimates of the Gaussian distributed variables showed quite similar performance for both RT and CV method. The RT method showed lower bias than the CV method, but the total $\widehat{MISE}$ can be seen to be approximately the same. In the

$min_x = -5$, $max_x = 5$, $N = 100$ $Avg = 20$		
Parameters	Rule of Thumb (RT)	Cross validation log-likelihood (CV)
I_{bias^2}	0.001	0.002
I_{var}	0.006	0.005
$\widehat{MISE}$	0.007	0.007
$E[\sigma]$	0.035	0.4323
$var[\sigma]$	0.001	0.011

Table 4.2: Mean and Variance decomposition of the $\widehat{MISE}$ for the two methods in a Gaussian environment.

uniform case it seems to be the other way around, since the RT method shows bigger bias and lower variance than the CV-method which have higher variance than bias in this setup. The $\widehat{MISE}$ however, is a little lower for the RT than the CV-method. In the last test setup, the $\widehat{MISE}$ is ten times lower for the RT-method than the CV-

$min_x = -2$, $max_x = 2$, $N = 100$ $Avg = 20$		
Parameters	Rule of Thumb	Cross validation log-likelihood method
I_{bias^2}	0.051	0.028
I_{var}	0.02	0.06
$\widehat{MISE}$	0.071	0.088
$E[\sigma]$	0.102	0.045
$var[\sigma]$	$3.3e-5$	$2.7e-4$

Table 4.3: Mean and Variance decomposition of the $\widehat{MISE}$ for the two methods in a uniform environment.

$min_x = -20$, $max_x = 20$, $N = 100$ $Avg = 20$		
Parameters	Rule of Thumb	Cross validation log-likelihood method
I_{bias^2}	0.0014	0.059
I_{var}	0.0044	$2.54e-4$
$\widehat{MISE}$	0.006	0.059
$E[\sigma]$	0.512	3.08
$var[\sigma]$	0.06	4.2

Table 4.4: Mean and Variance decomposition of the $\widehat{MISE}$ for the two methods in a Cauchy environment.

method. When looking at the variance of the estimated sigma value, the variance is extremely high with the CV-method, which again tells that the value of sigma is very dependent on outliers in the data.

Discussion

When outliers are present in the data the evaluation of the CV log-likelihood tend to give a much larger value for the width parameter σ, where the estimate given by the "Rule of Thumb" method was much more consistent with the true distribution. Since we need a method which is reliable for small sample sizes and efficient with respect to the computational burden makes the "Rule of Thumb" method the preferred method for the different algorithms later in the project.

Since the CV log-likelihood method over-fits the data in super-Gaussian environment some method, which is not dependent on the shape of the data is needed. A method which might be relevant is known as the *Least-squares cross-validation* technique [32]. The cost function is given as:

$$\int_{-\infty}^{\infty} [p(x) - \hat{p}(x)]^2 \, dx \tag{4.66}$$

from which the following cost-function will be found (only $\hat{p}(x)$ depend on the kernel width)

$$R(\hat{p}) = \int_{-\infty}^{\infty} \hat{p}(x)^2 dx - 2 \int_{-\infty}^{\infty} \hat{p}(x) p(x) dx. \tag{4.67}$$

This cost-function should also be minimized using cross-validation. This method does not have problems with outliers in the data as the CV log-likelihood approach. This method have not been implemented but a fruitful discussion on how to implement this method is given in [32].

4.4.2 Empirical density estimate

An empirical density estimate, which have already been used, is much like the Parzen estimate[8] defined as

$$\hat{p}(x) = \frac{1}{N}\sum_{k=1}^{N}\delta(x - x_k), \tag{4.68}$$

which is an unbiased estimate of the distribution function. The use of the empirical density estimate may come in handy when one have to determine a mean value

$$E[x] = \int_{-\infty}^{\infty} xp(x)dx, \tag{4.69}$$

inserting the empirical density estimate will give

$$\hat{E}[x] = \int_{-\infty}^{\infty} x\hat{p}(x)dx \tag{4.70}$$

$$= \frac{1}{N}\sum_{k=1}^{N}\int_{-\infty}^{\infty} x\delta(x - x_k)dx \tag{4.71}$$

$$= \frac{1}{N}\sum_{k=1}^{N} x_k. \tag{4.72}$$

Since the empirical density estimate is an unbiased estimator of the true distribution the following must hold

$$p(x) = \lim_{N\to\infty}\frac{1}{N}\sum_{k=1}^{N}\delta(x - x_k). \tag{4.73}$$

[8] In the limit as $\sigma \to 0$ the Parzen window will become equal to the empirical density estimate, since the kernel will approach a delta-function.

4.5 Generalization error and its relation to regularization

In this subsection, a Parzen window will be used to estimate $p(\boldsymbol{x}, y)$ instead of using an empirical density estimate. It is shown that this corresponds to a kind of regularization. A similar statement is found in [5].

The expression for the generalization error is repeated here (Using Renyi generalization and Shannon generalization error)

$$G_R = \frac{1}{1-\alpha} \log \int p(e(\boldsymbol{x}, \boldsymbol{w}, y))^{\alpha-1} p(\boldsymbol{x}, y) d\boldsymbol{x} dy \tag{4.74}$$

$$G_S = -\int \log p(e(\boldsymbol{x}, \boldsymbol{w}, y)) p(\boldsymbol{x}, y) d\boldsymbol{x} dy. \tag{4.75}$$

For simplicity $e(\boldsymbol{x}, \boldsymbol{w}, y)$ will be denoted as e where the dependency on the variables $\boldsymbol{x}, \boldsymbol{w}, y$ is understood. Applying a Parzen density estimate as in [33] to the joint density $p(\boldsymbol{x}, y)$ will simplify the expression. The Parzen estimate is given as

$$\hat{p}(\boldsymbol{x}, y) = \frac{1}{N} \sum_{n=1}^{N} \kappa_{\sigma_x}(\boldsymbol{x} - \boldsymbol{x}_n) \kappa_{\sigma_y}(y - y_n). \tag{4.76}$$

It was noted in [33] that this estimator is a good choice for estimation of the probability density function if it can be assumed that the underlying density function is continuous and that the first partial derivatives of any $\boldsymbol{x}$ is small[9]. Inserting the PDF-estimate into (4.74) and (4.75) gives

$$C_R = \hat{G}_\alpha = \frac{1}{1-\alpha} \log \frac{1}{N} \sum_{n=1}^{N} \int p(e)^{\alpha-1} \kappa_{\sigma_x}(\boldsymbol{x} - \boldsymbol{x}_n) \kappa_{\sigma_y}(y - y_n) d\boldsymbol{x} dy \tag{4.77}$$

$$C_S = \hat{G}_S = -\frac{1}{N} \sum_{n=1}^{N} \int \log p(e) \kappa_{\sigma_x}(\boldsymbol{x} - \boldsymbol{x}_n) \kappa_{\sigma_y}(y - y_n) d\boldsymbol{x} dy \tag{4.78}$$

The two integrals are rather complicated since we do not know the distribution of the error and further the error e is dependent on both y and $\boldsymbol{x}$ in a linear or nonlinear manner. It will be shown in the following derivations that the approximated Shannon generalization error (4.78) can be split into two parts as

$$C_S(\boldsymbol{w}) = \hat{C}_S(\boldsymbol{w}) + R(\boldsymbol{w}) \tag{4.79}$$

where $\hat{C}_S(\boldsymbol{w})$ is the result obtained when applying an empirical density estimate of $p(\boldsymbol{x}, y)$ to (4.78) plus a term which involves higher order derivatives of $\log p(e)$. The term $R(\boldsymbol{w})$ can be regarded as a regularization term controlling the curvature of

[9] This might not always be the case with the unknown data

$\log p(e)$ with respect to $\boldsymbol{x}$ and y. The modified Renyi generalization error given in (4.77) cannot be splitted directly. Only the information potential can be splitted, so

$$C_R(\boldsymbol{w}) = \frac{1}{1-\alpha} \log \left[\hat{V}_\alpha(\boldsymbol{w}) + V_\alpha R(\boldsymbol{w}) \right] \tag{4.80}$$

where $\hat{V}_\alpha(\boldsymbol{w})$ is the information potential that would be obtained using an empirical density estimate of $p(\boldsymbol{x}, y)$, and $V_\alpha R(\boldsymbol{w})$ is a regularization term. This term controls higher order derivatives of $p(e)^{\alpha-1}$ with respect to $\boldsymbol{x}$ and y.

A simple case

To get an understanding of the method a simple case is investigated at first. Consider the following integral in one variable (scalar)

$$G_{simple} = \int_{-\infty}^{\infty} f(x)p(x)dx \tag{4.81}$$

where both functions $f(x)$ and $p(x)$ is assumed to be probability functions. Further we assume that $f(x)$ has continuous derivatives. $p(x)$ is estimated using a Parzen window with width parameter σ using a Gaussian kernel

$$\hat{p}(x) = \frac{1}{N} \sum_i \kappa_\sigma(x - x_i). \tag{4.82}$$

This estimate is inserted into (4.81) to give

$$G_{simple} = \int_{-\infty}^{\infty} f(x) \frac{1}{N} \sum_i \kappa_\sigma(x - x_i)dx \tag{4.83}$$

$$= \frac{1}{N} \sum_i \int_{-\infty}^{\infty} f(x)\kappa_\sigma(x - x_i)dx. \tag{4.84}$$

Since we have assumed that $f(x)$ has continuous derivatives its Taylor expansion around x_i can be defined

$$f(x) = \sum_{q=0}^{\infty} \frac{1}{q!} f^{\langle q \rangle}(x_i)(x - x_i)^q, \tag{4.85}$$

where $f^{\langle q \rangle}(x_i)$ is the q'th derivative of $f(x_i)$. Inserting (4.85) into (4.84) and moving the dependency on x under the integral sign results in the following expression

$$G_{simple} = \frac{1}{N} \sum_i \sum_{q=0}^{\infty} \frac{1}{q!} f^{\langle q \rangle}(x_i) \int_{-\infty}^{\infty} (x - x_i)^q \kappa_\sigma(x - x_i)dx \tag{4.86}$$

Substituting variables such that $u = x - x_i$ in the integral above gives

$$I(q, x(i)) = \int_{-\infty}^{\infty} u^q \kappa_\sigma(u) du. \tag{4.87}$$

This integral is zero when q is not an even number, so

$$\int_{-\infty}^{\infty} u^q \kappa_\sigma(u) du = 0 \quad \text{for} \quad q = 1, 3, 5, ... \tag{4.88}$$

Using this observation the expression given by (4.86) can be written as

$$G_{simple} = \frac{1}{N} \sum_i \sum_{q=0}^{\infty} \frac{1}{2q!} f^{\langle 2q \rangle}(x_i) \int_{-\infty}^{\infty} (x - x_i)^{2q} \kappa_\sigma(x - x_i) dx. \tag{4.89}$$

The integral given in (4.89) can be solved in a closed form, since this is the central moments of x. The first couple of central moment of x given that the kernel is Gaussian can be seen from table (4.5) page (51). The q'th central moments for a

q	Central moments
0	1
1	σ^2
2	$3\sigma^4$

Table 4.5: The first central moments of a Gaussian distributed variable

Gaussian distributed variable can be expressed as [31]

$$\int_{-\infty}^{\infty} (x - x_i)^{2q} \kappa_\sigma(x - x_i) dx = \frac{(2q)! \sigma^{2q}}{q! 2^q} \quad q \in \mathcal{Z}. \tag{4.90}$$

Inserting the above observation into equation (4.87) gives the following

$$\begin{aligned} G_{simple} &= \frac{1}{N} \sum_i \sum_{q=0}^{\infty} \frac{1}{(2q)!} f^{\langle 2q \rangle}(x_i) \frac{(2q)! \sigma^{2q}}{q! 2^q} \\ &= \frac{1}{N} \sum_i \sum_{q=0}^{\infty} f^{\langle 2q \rangle}(x_i) \frac{\sigma^{2q}}{q! 2^q}. \end{aligned} \tag{4.91}$$

The solution given in (4.91) shows that the derivative of $f(x_i)$ should be bounded in order for the sum to converge.

It has been shown that the integral expression given in (4.81) can be splitted into an expression dependent on the samples alone and additional terms which will depend on the derivatives of $f(x)$. To select the number of terms to use, will depend on the application. Using $q = 0$ gives the following expression

$$G_{simple} = \frac{1}{N} \sum_i f(x_i), \tag{4.92}$$

which is the same as applying the empirical density estimate of $p(x)$ in (4.81). Using the first two terms would provide the following solution

$$G_{simple} = \frac{1}{N}\left[\sum_i f(x_i) + \frac{\sigma^2}{2!} f^{\langle 2 \rangle}(x_i)\right], \tag{4.93}$$

where the cost function will now contain second order information about $f(x)$[10] which will put a limit on the curvature for the unknown density $f(x)$. From the above formula it is seen that the width-parameter in the Parzen estimate will have an influence on how much regularization is performed.

[10] This is also known as Tikhonov regularization since we are modelling the density f(x)

Generalizing the simple case to several variables

Since our main purpose was to split the integral given in (4.78) or (4.77) we now have to generalize the result found in the simple case into several variables. Let us start with writing equation (4.78) using the parzen estimate for $p(\boldsymbol{x}, y)$

$$C_S = -\frac{1}{N}\sum_{n=1}^{N}\int f(\boldsymbol{x}, y)\kappa_{\sigma_x}(\boldsymbol{x}-\boldsymbol{x}_n)\kappa_{\sigma_y}(y-y_n)d\boldsymbol{x}dy \tag{4.94}$$

where $f(\boldsymbol{x}, y)$ is equal to $\log p(e|\boldsymbol{x}, y, \boldsymbol{w})$.

The next step as before, is to do a Taylor expansion of $f(\boldsymbol{x}, y)$, which from [35] and some additional calculation, which can be seen from appendix (A.1) page (110) can be found to the following (using following notation $f = f(\boldsymbol{x}', y')$) :

$$\begin{aligned} f(\boldsymbol{x}, y) = f(\boldsymbol{x}_n, y_n) + (\boldsymbol{x}-\boldsymbol{x}_n)^T \cdot \nabla_{\boldsymbol{x}'} f + (y-y_n)\nabla_{y'} f + \\ \frac{1}{2!}(\boldsymbol{x}-\boldsymbol{x}_n)^T \cdot \boldsymbol{\nabla}_{\boldsymbol{x}'\boldsymbol{x}'} f \cdot (\boldsymbol{x}-\boldsymbol{x}_n) + (y-y_n)(\boldsymbol{x}-\boldsymbol{x}_n)^T \cdot \nabla_{y'\boldsymbol{x}'} f + \\ \frac{1}{2!}(y-y_n)^2 \nabla_{y'y'} f + R_3\Big|_{\boldsymbol{x}'=\boldsymbol{x}_n, y'=y_n} \end{aligned} \tag{4.95}$$

where the term R_3 refers to the remainder after three terms. The Taylor expansion (4.95) is substituted into the expression for the generalization error in (4.94). This will be a very long expression, so each term is written out separately under the integral and calculated:

$$f(\boldsymbol{x_n}, y_n)\int \kappa_{\sigma_x}(\boldsymbol{x}-\boldsymbol{x}_n)\kappa_{\sigma_y}(y-y_n)d\boldsymbol{x}dy = f(\boldsymbol{x_n}, y_n). \tag{4.96}$$

$$\begin{aligned} \nabla_{\boldsymbol{x}'} f\Big|_{\boldsymbol{x}'=\boldsymbol{x}_n}^T \int (\boldsymbol{x}-\boldsymbol{x_n})\kappa_{\sigma_x}(\boldsymbol{x}-\boldsymbol{x}_n)\kappa_{\sigma_y}(y-y_n)d\boldsymbol{x}dy \quad &= \\ \nabla_{\boldsymbol{x}'} f\Big|_{\boldsymbol{x}'=\boldsymbol{x}_n}^T \int (\boldsymbol{x}-\boldsymbol{x_n})\kappa_{\sigma_x}(\boldsymbol{x}-\boldsymbol{x}_n)d\boldsymbol{x} \quad &= \quad \mathbf{0}. \end{aligned} \tag{4.97}$$

$$\begin{aligned} \nabla_{y'} f\Big|_{y'=y_n} \int (y-y_n)\kappa_{\sigma_x}(\boldsymbol{x}-\boldsymbol{x}_n)\kappa_{\sigma_y}(y-y_n)d\boldsymbol{x}dy = \\ \nabla_{y'} f\Big|_{y'=y_n} \int (y-y_n)\kappa_{\sigma_y}(y-y_n)dy \quad = \quad 0. \end{aligned} \tag{4.98}$$

$$\begin{aligned} \int \frac{1}{2!}(\boldsymbol{x}-\boldsymbol{x}_n)^T \nabla_{\boldsymbol{x}'\boldsymbol{x}'} f\Big|_{\boldsymbol{x}'=\boldsymbol{x}_n} (\boldsymbol{x}-\boldsymbol{x}_n)\kappa_{\sigma_x}(\boldsymbol{x}-\boldsymbol{x}_n)\kappa_{\sigma_y}(y-y_n)d\boldsymbol{x}dy = \\ \frac{1}{2!}\int (\boldsymbol{x}-\boldsymbol{x}_n)^T \nabla_{\boldsymbol{x}'\boldsymbol{x}'} f\Big|_{\boldsymbol{x}'=\boldsymbol{x}_n} (\boldsymbol{x}-\boldsymbol{x}_n)\kappa_{\sigma_x}(\boldsymbol{x}-\boldsymbol{x}_n)d\boldsymbol{x}. \end{aligned} \tag{4.99}$$

Since we know that, the result should be a scalar the trace operator $Tr\{\cdot\}$ is invoked which gives

$$\frac{1}{2!}\int Tr\left\{(\boldsymbol{x}-\boldsymbol{x}_n)^T\nabla_{\boldsymbol{x}'\boldsymbol{x}'}f\Big|_{\boldsymbol{x}'=\boldsymbol{x}_n}(\boldsymbol{x}-\boldsymbol{x}_n)\kappa_{\sigma_x}(\boldsymbol{x}-\boldsymbol{x}_n)d\boldsymbol{x}\right\}=$$
$$\frac{1}{2!}Tr\left\{\nabla_{\boldsymbol{x}'\boldsymbol{x}'}f\Big|_{\boldsymbol{x}'=\boldsymbol{x}_n}\int(\boldsymbol{x}-\boldsymbol{x}_n)(\boldsymbol{x}-\boldsymbol{x}_n)^T\kappa_{\sigma_x}(\boldsymbol{x}-\boldsymbol{x}_n)d\boldsymbol{x}\right\}=$$
$$\frac{1}{2!}Tr\left\{\nabla_{\boldsymbol{x}'\boldsymbol{x}'}f\Big|_{\boldsymbol{x}'=\boldsymbol{x}_n}\sigma_x^2\boldsymbol{I}\right\}=\frac{1}{2!}\sigma_x^2Tr\left\{\nabla_{\boldsymbol{x}'\boldsymbol{x}'}f\Big|_{\boldsymbol{x}'=\boldsymbol{x}_n}\right\}.$$

$$\int(y-y_n)(\boldsymbol{x}-\boldsymbol{x}_n)^T\nabla_{\boldsymbol{x}'\boldsymbol{x}'}f\Big|_{\boldsymbol{x}'=\boldsymbol{x}_n}\kappa_{\sigma_x}(\boldsymbol{x}-\boldsymbol{x}_n)\kappa_{\sigma_y}(y-y_n)d\boldsymbol{x}dy=$$
$$\int(y-y_n)\kappa_{\sigma_y}(y-y_n)dy\nabla_{\boldsymbol{x}'\boldsymbol{x}'}f^T\Big|_{\boldsymbol{x}'=\boldsymbol{x}_n}\cdot\int(\boldsymbol{x}-\boldsymbol{x}_n)\kappa_{\sigma_x}(\boldsymbol{x}-\boldsymbol{x}_n)d\boldsymbol{x}=0. \quad (4.100)$$

$$\frac{1}{2!}\int(y-y_n)^2\nabla_{y'y'}f\Big|_{y'=y_n}\kappa_{\sigma_y}(y-y_n)\kappa_{\sigma_x}(\boldsymbol{x}-\boldsymbol{x}_n)dyd\boldsymbol{x}=$$
$$\frac{1}{2!}\int(y-y_n)^2\nabla_{y'y'}f\Big|_{y'=y_n}\kappa_{\sigma_y}(y-y_n)dy=\frac{1}{2!}\sigma_y^2\nabla_{y'y'}f\Big|_{y'=y_n}. \quad (4.101)$$

The last many terms are collected in R_3 and is written as

$$C_{S3}=\int R_3\kappa_{\sigma_x}(\boldsymbol{x}-\boldsymbol{x}_n)\kappa_{\sigma_y}(y-y_n)d\boldsymbol{x}dy. \quad (4.102)$$

Using the results from the above calculations in (4.94), gives the following expression for C_S

$$C_S=-\frac{1}{N}\sum_n\left[f(\boldsymbol{x}_n,y_n)+\frac{1}{2!}\sigma_x^2Tr\left\{\nabla_{\boldsymbol{x}'\boldsymbol{x}'}f\Big|_{\boldsymbol{x}'=\boldsymbol{x}_n}\right\}+\right.$$
$$\left.\frac{1}{2!}\sigma_y^2\nabla_{y'y'}f\Big|_{y'=y_n}+C_{S3}\right]. \quad (4.103)$$

It is seen that the expression given in (4.94) has been splitted according to (4.79). It can be seen that C_R can be written in a similar manner:

$$C_R=\frac{1}{1-\alpha}\log\frac{1}{N}\sum_n\left[f(\boldsymbol{x}_n,y_n)+\frac{1}{2!}\sigma_x^2\,Tr\left\{\nabla_{\boldsymbol{x}'\boldsymbol{x}'}f\Big|_{\boldsymbol{x}'=\boldsymbol{x}_n}\right\}+\right.$$
$$\left.\frac{1}{2!}\sigma_y^2\nabla_{y'y'}f\Big|_{y'=y_n}+C_{R3}\right], \quad (4.104)$$

which corresponds to the splitting in (4.80). The terms C_{R3} and C_{S3} is the higher order terms of the Taylor expansion of $f(\boldsymbol{x}, y)$.

It is now clear that the Renyi generalization error cannot be splitted into an empirical estimate plus a regularization term, due to the log function outside the integral. Instead of minimizing Renyi generalization error normally the information potential is maximized. The information is exactly the integral so the curvature of $p(e)^{\alpha-1}$ is limited instead (regularized).

What have we shown

It have been shown that when using an Parzen estimate for the joint probability density function $p(\boldsymbol{x}, y)$ this corresponds to adding a regularization term to the generalization error, obtained with an empirical density estimate. In the case of Shannon generalization error this corresponds to higher derivatives of $\log p(e)$ while in the Renyi case (when maximizing the information potential) corresponds to an additional term controlling higher order derivatives of $p(e)^{\alpha-1}$. In this project though, I have only considered empirical density estimates.

4.6 Algorithms that can be derived from the generalization error

In this section, three algorithms are derived. One batch-algorithm, which updates the unknown weights of the model at each N'th time step. Another algorithm are the stochastic information gradient method, which updates the unknown weights of the model online. The last algorithm considered are a recursive algorithm, also an online version that is derived in a similar way to the RLS algorithm. In the first part of the section the cost-function that is considered is defined.

The Renyi generalization error will be considered as the cost-function to be minimized.

$$G_R(\boldsymbol{w}) = \frac{1}{1-\alpha} \log \int p(e|\boldsymbol{w}, \boldsymbol{x})^{\alpha-1} p(\boldsymbol{x}, y) d\boldsymbol{x} dy \quad \text{for} \quad \alpha \geq 1. \tag{4.105}$$

Instead of minimizing the Renyi generalization error, one can maximize the information potential [7]

$$V_\alpha(\boldsymbol{w}) = \int p(e|\boldsymbol{w}, \boldsymbol{x})^{\alpha-1} p(\boldsymbol{x}, y) d\boldsymbol{x} dy \tag{4.106}$$

since log is a monotonic function in the given interval. Using an empirical density estimate of $p(\boldsymbol{x}, y)$ in the information potential expression

$$\hat{p}(\boldsymbol{x}, y) = \frac{1}{N} \sum_{n=1}^{N} \delta(\boldsymbol{x} - \boldsymbol{x}_n) \delta(y - y_n) \tag{4.107}$$

gives

$$\boxed{\widehat{V_\alpha}(\boldsymbol{w}) = \frac{1}{N} \sum_{n=1}^{N} p(e_n|\boldsymbol{w}, \boldsymbol{x}_n)^{\alpha-1}.} \tag{4.108}$$

A special simple case happens when $\alpha = 2$ which will be denoted as the quadratic Renyi generalization error, similar to [7].

The cost-function given in (4.108) will be denoted as our general cost-function. From this cost-function several algorithms can be derived.

As mentioned earlier the error is determined as

$$e(\boldsymbol{w}) = y - f(\boldsymbol{x}, \boldsymbol{w}) \tag{4.109}$$

where $f(\cdot)$ might be a linear or non-linear function.

Throughout this chapter the following notation will be used:

$$e_{i,j}(\boldsymbol{w}) = e_i(\boldsymbol{w}) - e_j(\boldsymbol{w}) \tag{4.110}$$

$$\phi_{i,j}(\boldsymbol{w}) = \frac{\partial f(\boldsymbol{x}_i, \boldsymbol{w})}{\partial \boldsymbol{w}} - \frac{\partial f(\boldsymbol{x}_j, \boldsymbol{w})}{\partial \boldsymbol{w}} \tag{4.111}$$

where $\phi_{i,j}(\boldsymbol{w})$ is a vector of size $L \times 1$ where L is number weight in the model.

4.6.1 Batch Algorithms

In an online adaptive setup, the unknown system parameters need to be updated according to the ever-changing environment. In a batch-setup N- samples will be used to update the unknown weights of the system at time k. The cost function using the last N samples will look like

$$\widehat{V_{\alpha,k}}(\boldsymbol{w}) = \frac{1}{N}\sum_{n=k-N+1}^{k} p(e_n|\boldsymbol{w},\boldsymbol{x}_n)^{\alpha-1} \tag{4.112}$$

where k is the time instant where the weights of the model are updated. One way to maximize the cost-function is to update the unknown weight coefficients using a steepest accent method. This method have been used in several papers on information theoretic methods [28], [10] and [7]. The steepest accent updates the unknown weights at time k in the direction of the gradient

$$\boldsymbol{w}_k = \boldsymbol{w}_{k-1} - \eta_{k-1}\frac{\partial\widehat{V_{\alpha,k}}(\boldsymbol{w}')}{\partial\boldsymbol{w}'}\bigg|_{\boldsymbol{w}'=\boldsymbol{w}_{k-1}}, \tag{4.113}$$

where η_{k-1} is the step taken in the gradient direction. In steepest descent/accent approaches the step-size will control the convergence speed and the final error achieved (system mismatch). In the following, the notation $p(e) = p(e|\boldsymbol{x},\boldsymbol{w})$ is used. The gradient is calculated as

$$\frac{\partial\widehat{V_{\alpha,k}}(\boldsymbol{w})}{\partial\boldsymbol{w}} = \frac{1}{N}\sum_{n=k-N+1}^{k}(\alpha-1)p(e_n)^{\alpha-2}\frac{\partial p(e_n)}{\partial\boldsymbol{w}}. \tag{4.114}$$

As already mentioned the performance of the method will depend very much on the selection of η_{k-1} and normally a steepest descent algorithm will not converge very fast, since one should not step more than the smallest eigenvalue of the system. A method which contains the steepest descent/accent method, but can be modified to a Newton method is found in [23] and is given as

$$\boldsymbol{w}_k = \boldsymbol{w}_{k-1} - \eta_k\boldsymbol{R}_k^{-1}\frac{\partial\widehat{V_{\alpha,k}}(\boldsymbol{w}')}{\partial\boldsymbol{w}'}\bigg|_{\boldsymbol{w}'=\boldsymbol{w}_{k-1}} \tag{4.115}$$

where the steepest accent method for maximizing (4.112) can be identified when choosing $\boldsymbol{R} = -\boldsymbol{I}$, where $\boldsymbol{I}$ is the identity matrix. When selecting $\boldsymbol{R}$ as the Hessian of the cost-function Newton's method appears. The Hessian of the cost function given in (4.112) can be calculated to give (similar to [10])

$$\frac{\partial^2\widehat{V_{\alpha,k}}(\boldsymbol{w})}{\partial\boldsymbol{w}\partial\boldsymbol{w}} = \frac{1}{N}\sum_{n=k-N+1}^{k}(\alpha-1)(\alpha-2)p(e_n)^{\alpha-3}\frac{\partial p(e_n)}{\partial\boldsymbol{w}}+ \\ (\alpha-1)p(e_n)^{\alpha-2}\frac{\partial^2 p(e_n)}{\partial\boldsymbol{w}\partial\boldsymbol{w}}. \tag{4.116}$$

The above results will be used to derive a Gauss Newton algorithm in the quadratic case.

Deriving a Gauss Newton algorithm for $\alpha = 2$

It is clear that selecting $\alpha = 2$ will simplify things considerately. The unknown probability function ($p(e_n)$) will be estimated using a Parzen window using Gaussian kernel functions. In the following the PDF-estimates as well as its derivatives with respect to the unknown weights given as

$$p(e_n(\boldsymbol{w})) = \frac{1}{N}\sum_{i=k-N+1}^{k}\kappa_\sigma(e_{n,i}(\boldsymbol{w})) \tag{4.117}$$

$$\frac{\partial p(e_n(\boldsymbol{w}))}{\partial \boldsymbol{w}} = \frac{1}{N}\sum_{i=k-N+1}^{k}\frac{e_{n,i}(\boldsymbol{w})}{\sigma^2}\kappa_\sigma(e_{n,i}(\boldsymbol{w}))\boldsymbol{\phi}_{n,i}(\boldsymbol{w}) \tag{4.118}$$

where the derivative of a Gaussian kernel can be found in the footnote[11] . Differentiating once more using the chain rule gives (the result should be a matrix of dimension $L \times L$) the Hessian matrix of the non-parametric density estimate

$$\frac{\partial^2 p(e_n(\boldsymbol{w}))}{\partial \boldsymbol{w}\partial \boldsymbol{w}} = \frac{1}{N}\sum_{i=k-N+1}^{k}\Bigg[-\frac{\kappa_\sigma(e_{n,i}(\boldsymbol{w}))}{\sigma^2}\phi_{n,i}(\boldsymbol{w})\phi_{n,i}(\boldsymbol{w})^T + \frac{e_{n,i}^2}{\sigma^4}\kappa_\sigma(e_{n,i}(\boldsymbol{w}))\phi_{n,i}(\boldsymbol{w})\phi_{n,i}(\boldsymbol{w})^T + \frac{e_{n,i}(\boldsymbol{w})}{\sigma^2}\kappa_\sigma(e_{n,i}(\boldsymbol{w}))\frac{\partial \phi_{n,i}(\boldsymbol{w})}{\partial \boldsymbol{w}}\Bigg]. \tag{4.119}$$

So the gradient and the Hessian of the cost-function when $\alpha = 2$ is given as
The gradient:

$$\boldsymbol{b}_k(\boldsymbol{w}) = \frac{\partial \widehat{V_{2,k}}(\boldsymbol{w})}{\partial \boldsymbol{w}} = \frac{1}{N^2\sigma^2}\sum_{n=k-N+1}^{k}\sum_{i=k-N+1}^{k} e_{n,i}(\boldsymbol{w})\kappa_\sigma(e_{n,i}(\boldsymbol{w}))\boldsymbol{\phi}_{n,i}(\boldsymbol{w}) \tag{4.120}$$

and the Hessian:

$$\frac{\partial^2 \widehat{V_{2,k}}(\boldsymbol{w})}{\partial \boldsymbol{w}\partial \boldsymbol{w}} = \frac{1}{N^2\sigma^2}\sum_{n=k-N+1}^{k}\sum_{i=k-N+1}^{k}\Bigg[-\kappa_\sigma(e_{n,i}(\boldsymbol{w}))\phi_{n,i}(\boldsymbol{w})\phi_{n,i}(\boldsymbol{w})^T + \frac{e_{n,i}^2}{\sigma^2}\kappa_\sigma(e_{n,i}(\boldsymbol{w}))\phi_{n,i}(\boldsymbol{w})\phi_{n,i}(\boldsymbol{w})^T + e_{n,i}(\boldsymbol{w})\kappa_\sigma(e_{n,i}(\boldsymbol{w}))\frac{\partial \phi_{n,i}(\boldsymbol{w})}{\partial \boldsymbol{w}}\Bigg]. \tag{4.121}$$

Instead of using the full Hessian matrix a pseudo Hessian [23] can be used. In the vicinity of the maximum of the cost-function the Hessian matrix given by (4.121) can be approximated by

$$\frac{\partial^2 \widehat{V_{2,k}}(\boldsymbol{w})}{\partial \boldsymbol{w}\partial \boldsymbol{w}} = \frac{1}{N^2\sigma^2}\sum_{n=k-N+1}^{k}\sum_{i=k-N+1}^{k}\Bigg[-1+\frac{e_{n,i}^2}{\sigma^2}\Bigg] \kappa_\sigma(e_{n,i}(\boldsymbol{w}))\phi_{n,i}(\boldsymbol{w})\phi_{n,i}(\boldsymbol{w})^T \tag{4.122}$$

[11]
$$\frac{\partial \kappa_\sigma(e_{n,i}(\boldsymbol{w}))}{\partial \boldsymbol{w}} = \frac{-2e_{n,i}(\boldsymbol{w})}{2\sigma^2}\kappa_\sigma(\cdot)\frac{\partial e_{n,i}(\boldsymbol{w})}{\partial \boldsymbol{w}} = \frac{e_{n,i}(\boldsymbol{w})}{\sigma^2}\kappa_\sigma(e_{n,i}(\boldsymbol{w}))\phi_{n,i}(\boldsymbol{w})$$

since we assume independence of the error at the solution. The same method has been used in [23] to obtain the pseudo Hessian[12]. The pseudo Hessian in [23] is guaranteed positive definite (In that he minimizes the mean square error). The pseudo Hessian given in (4.122) should be negative definite (Since we are maximizing the cost-function), but this is not ensured due to the kernel-dependent part $\frac{e_{n,i}^2}{\sigma^2}$. Near the minimum $\boldsymbol{w} \approx \boldsymbol{w}_{opt}$ the kernel should be independent of the weights, so

$$\left.\frac{\partial \kappa_\sigma(e_{n,i}(\boldsymbol{w}'))}{\partial \boldsymbol{w}'}\right|_{\boldsymbol{w}'=\boldsymbol{w}_{opt}} \approx \boldsymbol{0} \tag{4.123}$$

which corresponds to the gradient. So the expression for the pseudo-Hessian can be approximated by

$$\widehat{\boldsymbol{H}}_k(\boldsymbol{w}) = \frac{\partial^2 \widehat{V_{2,k}}(\boldsymbol{w})}{\partial \boldsymbol{w} \partial \boldsymbol{w}} = \frac{-1}{N^2\sigma^2} \sum_{n=k-N+1}^{k} \sum_{i=k-N+1}^{k} \kappa_\sigma(e_{n,i}(\boldsymbol{w}))\phi_{n,i}(\boldsymbol{w})\phi_{n,i}(\boldsymbol{w})^T \tag{4.124}$$

which will also guarantee negative definiteness when the number of samples N is large enough ($N > L$ to ensure a non-singular Hessian matrix). The negative definiteness is required for the algorithm to converge and as can be seen from the expression the matrix will be symmetric. A small simulation study of including and excluding the term $\frac{e_{n,i}^2}{\sigma^2}$ in a simple AR(2) setup in different statistical environments, will be given in section (4.6.3) page (68).

Using the above pseudo Hessian gives a Gauss Newton update of the weights [23]. The weights will be updated as

$$\boldsymbol{w}_k = \boldsymbol{w}_{k-1} - \eta_{k-1}\widehat{\boldsymbol{H}}_k(\boldsymbol{w}_{k-1})^{-1}\boldsymbol{b}_k(\boldsymbol{w}_{k-1}). \tag{4.125}$$

In the simple case when considering a linear system such as a FIR-system setup then $\phi_{n,i}(\boldsymbol{w}) = \boldsymbol{x}_{n,i}$.

Minimizing the computational burden using symmetry

Due to the symmetry

$$\kappa_\sigma(e_{n,i}(\boldsymbol{w})) = \kappa_\sigma(e_{i,n}(\boldsymbol{w})) \tag{4.126}$$

$$e_{n,i}(\boldsymbol{w}) = -e_{i,n}(\boldsymbol{w}) \tag{4.127}$$

$$\phi_{n,i}(\boldsymbol{w}) = -\phi_{n,i}(\boldsymbol{w}). \tag{4.128}$$

the gradient and Hessian can be calculated computational more efficient[13]. The computation of the pseudo Hessian and the gradient can be calculated to be

[12] If the error is a linear function of the weights then the Hessian given by (4.122) is exact

[13] If thinking of e.g. $e_{i,j}$ as a matrix, then this is a symmetric matrix, and the summation of this symmetric matrix can be done by only summing the LU of the matrix.

Gradient

$$\boldsymbol{b}_k(\boldsymbol{w}) = \frac{1}{N^2\sigma^2} \sum_{n=k-N+1}^{k} \sum_{i=n}^{k} sc_{n,i} e_{n,i}(\boldsymbol{w}) \kappa_\sigma(e_{n,i}(\boldsymbol{w})) \phi_{n,i}(\boldsymbol{w}) \tag{4.129}$$

where $sc_{n,i} = 2$ when $n \neq i$ and $sc_{n,i} = 1$ for $n = i$.

pseudo Hessian

$$\widehat{\boldsymbol{H}}_k(\boldsymbol{w}) = \frac{-1}{N^2\sigma^2} \sum_{n=k-N+1}^{k} \sum_{i=n}^{k} sc_{n,i} \kappa_\sigma(e_{n,i}(\boldsymbol{w})) \phi_{n,i}(\boldsymbol{w}) \phi_{n,i}(\boldsymbol{w})^T \tag{4.130}$$

Thus only requiring around $N^2/2 + N$ summations instead of N^2 summations.

Algorithm layout and computational Requirements of the Algorithm

The general setup of the algorithm

- Initialization

  Select initial weights $\boldsymbol{w}_0$

  Select step-size η

- Update formula

  for $k = ninit..step(N)..Nsamp$

  calculate $\boldsymbol{b}(\boldsymbol{w}_{k-1})$ using (4.120).

  Calculate $\widehat{\boldsymbol{H}}(\boldsymbol{w}_{k-1})$ using (4.124).

  Update the unknown weights according to (4.125).

  end

A rough estimate of the calculation burden is $\mathcal{O}(L^3 + L^2(N/2)^2)$. The $L^2(N/2)^2$ term is the dominating part when building the gradient and the pseudo Hessian. The number of samples used in the PDF-estimate is controlled by N.

Instead of using the normal inversion in MATLAB, since we have to solve a linear system of equation, the MATLAB backslash operator is used. This solves the system of equations using Gaussian elimination, which also speed up the computations, and make the system more stable. But using Gaussian elimination still requires around $\mathcal{O}(L^3)$ operations.

The algorithm was implemented in MATLAB with the name "GNBatch.m".

4.6.2 Stochastic Gradient Algorithm

In [6] the stochastic information gradient (SIG) is introduced. This algorithm is minimizing the cost-function given by (4.112). The stochastic gradient can be determined using $N = 2$ in the cost-function, meaning that only the current sample k and previous sample $k-1$ is used in the evaluation of the gradient. Using $N = 2$ in equation (4.112) and inserting a Parzen estimate for $p(e_n)$ using Gaussian kernels gives

$$\widehat{V_\alpha, k}(\boldsymbol{w}) = \frac{1}{2^\alpha} \sum_{n=k-1}^{k} \left[\sum_{i=k-1}^{k} \kappa_\sigma(e_n - e_i) \right]^{\alpha-1}. \tag{4.131}$$

Deriving the gradient with respect to the unknown weights gives

$$\boldsymbol{b}_k(\boldsymbol{w}) = \frac{\partial \widehat{V_\alpha, k}(\boldsymbol{w})}{\partial \boldsymbol{w}} \tag{4.132}$$

$$= \frac{(\alpha-1)}{2^\alpha \sigma^2} \sum_{n=k-1}^{k} \left[\sum_{i=k-1}^{k} \kappa_\sigma(e_{n,i}(\boldsymbol{w})) \right]^{\alpha-2} \sum_{i=k-1}^{k} e_{n,i}(\boldsymbol{w}) \kappa_\sigma(e_{n,i}(\boldsymbol{w})) \phi_{n,i}(\boldsymbol{w}). \tag{4.133}$$

Only using the current sample k in the outer sum gives the following stochastic gradient [6]

$$\boldsymbol{b}_k(\boldsymbol{w}) = \frac{(\alpha-1)}{2^{\alpha-1} \sigma^2} \left[\sum_{i=k-1}^{k} \kappa_\sigma(e_{k,i}(\boldsymbol{w})) \right]^{\alpha-2} \sum_{i=k-1}^{k} e_{k,i}(\boldsymbol{w}) \kappa_\sigma(e_{k,i}(\boldsymbol{w})) \phi_{k,i}(\boldsymbol{w}). \tag{4.134}$$

This can be written as

$$\boldsymbol{b}_k(\boldsymbol{w}) = \frac{(\alpha-1)}{2^{\alpha-1} \sigma^2} \left[\kappa_\sigma(e_{k,k-1}(\boldsymbol{w})) + \kappa_\sigma(0) \right]^{\alpha-2} e_{k,k-1}(\boldsymbol{w}) \kappa_\sigma(e_{k,k-1}(\boldsymbol{w})) \phi_{k,k-1}(\boldsymbol{w}). \tag{4.135}$$

The expression is similar to the expression given in [6]. The unknown weights will be updated using a steepest accent approach

$$\boldsymbol{w}_k = \boldsymbol{w}_{k-1} + \eta \boldsymbol{b}_k(\boldsymbol{w}_{k-1}). \tag{4.136}$$

The window width still has to be determined. With this method only a few samples will be available with each update, and the width estimate would not be very good with only two samples. Either a constant width value is selected or else the width value is obtained using the last M samples (where M is a relevant number of samples for use in the estimate of the width parameter). The complexity of the algorithm is of $\mathcal{O}(L)$.

4.6.3 Recursive Quadratic Information Potential

To build a recursive algorithm, the empirical density estimate used in getting from (4.106) to (4.108) will be combined with a forgetting factor, in which the empirical density estimate of the joint probability density function $p(\boldsymbol{x}, y)$ will look like

$$\hat{p}(\boldsymbol{x}, y) = \mu_k \sum_{n=1}^{k} \beta_{k,n} \delta(\boldsymbol{x} - \boldsymbol{x}_n)\delta(y - y_n) \tag{4.137}$$

$$\tag{4.138}$$

where μ_k is a proper normalization

$$\mu_k = \frac{1}{\sum_{n=1}^{k} \beta_{k,n}} \tag{4.139}$$

ensuring that $\hat{p}(\boldsymbol{x}, y)$ is a probability function. Inspiration have been found in [23]. The weighting is selected to be of exponential form (as in the RLS) $\beta_{k,n} = \lambda^{k-n}$ where λ is real number between 0 and 1. If $\lambda = 1$ the system is said to have infinite memory, but normally one will choose $\lambda < 1$ if the algorithm should work in a non-stationary environment. μ_k can be determined in a closed form

$$\begin{aligned}
\mu_k &= \frac{1}{\sum_{n=1}^{k} \lambda^{k-n}} \\
&= \frac{1}{\sum_{n=1}^{k} \lambda^{n-1}} \\
&= \frac{1-\lambda}{1-\lambda^k} \quad \text{for} \quad \lambda < 1 && (4.140) \\
&= \frac{1}{k} \quad \text{for} \quad \lambda = 1. && (4.141)
\end{aligned}$$

Using the expression (4.137) in the expression for the information potential (4.106) gives the following estimate of the information potential

$$\boxed{\hat{V}_{\alpha,k}(\boldsymbol{w}) = \mu_k \sum_{n=1}^{k} \lambda^{k-n} p(e_n(\boldsymbol{w}))^{\alpha-1}.} \tag{4.142}$$

The expression given in (4.142) is regarded as the cost function to be maximized. The iterative algorithm applied to maximize this cost function is known as a recursive Gauss-Newton prediction error algorithm. The unknown model weights is updated according to

$$\boldsymbol{w}_k = \boldsymbol{w}_{k-1} - \eta_k \left[\hat{\boldsymbol{H}}_k(\boldsymbol{w}_{k-1})\right]^{-1} \hat{\boldsymbol{b}}_k(\boldsymbol{w}_{k-1}) \tag{4.143}$$

where $\widehat{\boldsymbol{H}}_k(\boldsymbol{w}_{k-1})$ is a pseudo Hessian of $\widehat{V}_{\alpha,k}(\boldsymbol{w})$ and $\hat{\boldsymbol{b}}_k(\boldsymbol{w}_{k-1})$ is a modified gradient of $\hat{V}_{\alpha,k}(\boldsymbol{w})$ [23].

The gradient of $\widehat{V}_{\alpha,k}(\boldsymbol{w})$ can be determined as

$$\begin{aligned}\boldsymbol{b}_k(\boldsymbol{w}) &= \frac{\partial \widehat{V}_{\alpha,k}(\boldsymbol{w})}{\partial \boldsymbol{w}} \\ &= \mu_k \sum_{n=1}^{k} \lambda^{k-n}(\alpha-1)p(e_n(\boldsymbol{w}))^{\alpha-2}\frac{\partial p(e_n(\boldsymbol{w}))}{\partial \boldsymbol{w}}.\end{aligned} \tag{4.144}$$

Rewriting the above expression

$$\begin{aligned}\boldsymbol{b}_k(\boldsymbol{w}) = \frac{\mu_k}{\mu_{k-1}}\lambda\mu_{k-1}\sum_{n=1}^{k-1}\lambda^{k-1-n}(\alpha-1)p(e_n(\boldsymbol{w}))^{\alpha-2}\frac{\partial p(e_n(\boldsymbol{w}))}{\partial \boldsymbol{w}} \\ + \mu_k(\alpha-1)p(e_k(\boldsymbol{w}))^{\alpha-2}\frac{\partial p(e_k(\boldsymbol{w}))}{\partial \boldsymbol{w}}.\end{aligned} \tag{4.145}$$

This can then be simplified using the recursive nature of the above expression. First $\frac{\mu_k}{\mu_{k-1}}\lambda$ is simplified

$$\begin{aligned}\frac{\mu_k}{\mu_{k-1}}\lambda &= \frac{1-\lambda}{1-\lambda^k}\frac{1-\lambda^{k-1}}{1-\lambda}\lambda \\ &= \frac{\lambda-\lambda^k}{1-\lambda^k} \\ &= \frac{1-1+\lambda-\lambda^k}{1-\lambda^k} \\ &= 1-\frac{1-\lambda}{1-\lambda^k} \\ &= 1-\mu_k\end{aligned} \tag{4.146}$$

from where the gradient can be expressed as

$$\boldsymbol{b}_k(\boldsymbol{w}) = (1-\mu_k)\boldsymbol{b}_{k-1}(\boldsymbol{w}) + \mu_k(\alpha-1)p(e_k(\boldsymbol{w}))^{\alpha-2}\frac{\partial p(e_k(\boldsymbol{w}))}{\partial \boldsymbol{w}}. \tag{4.147}$$

The recursive method is used in an online setup, which means that the gradient calculated with the weights from the last time step is used in updating the weights at time step k, so inserting $\boldsymbol{w}_{k-1}$ gives

$$\begin{aligned}\boldsymbol{b}_k(\boldsymbol{w}_{k-1}) = (1-\mu_k)\boldsymbol{b}_{k-1}(\boldsymbol{w}_{k-1}) + \\ \mu_k(\alpha-1)p(e_k(\boldsymbol{w}_{k-1}))^{\alpha-2}\frac{\partial p(e_k(\boldsymbol{w}'))}{\partial \boldsymbol{w}'}\bigg|_{\boldsymbol{w}'=\boldsymbol{w}_{k-1}}.\end{aligned} \tag{4.148}$$

Since we do not know $\boldsymbol{b}_{k-1}(\boldsymbol{w}_{k-1})$ we assume that $\boldsymbol{w}_{k-1}$ actually maximized $\hat{V}_{\alpha,k}(\boldsymbol{w}_{k-1})$ which means that $\boldsymbol{b}_{k-1}(\boldsymbol{w}_{k-1}) = \boldsymbol{0}$. The estimate of the gradient becomes

$$\hat{\boldsymbol{b}}_k(\boldsymbol{w}_{k-1}) = \mu_k(\alpha-1)p(e_k(\boldsymbol{w}_{k-1}))^{\alpha-2}\frac{\partial p(e_k(\boldsymbol{w}'))}{\partial \boldsymbol{w}'}\bigg|_{\boldsymbol{w}'=\boldsymbol{w}_{k-1}}. \tag{4.149}$$

Next, the Hessian has to be determined. The gradient (4.148) is differentiated once more with respect to the weights $\boldsymbol{w}$ to give

$$\boldsymbol{H}_k(\boldsymbol{w}) = \frac{\partial \boldsymbol{b}_k(\boldsymbol{w})}{\partial \boldsymbol{w}} \tag{4.150}$$

$$= (1-\mu_k)\frac{\partial \boldsymbol{b}_{k-1}(\boldsymbol{w})}{\partial \boldsymbol{w}} + \mu_k(\alpha-1)\left[\begin{array}{c}(\alpha-2)p(e_k(\boldsymbol{w}))^{\alpha-3}\left(\frac{\partial p(e_k(\boldsymbol{w}))}{\partial \boldsymbol{w}}\right)^2 \\ +(\alpha-1)p(e_k(\boldsymbol{w}))^{\alpha-2}\left(\frac{\partial^2 p(e_k(\boldsymbol{w}))}{\partial \boldsymbol{w}\partial \boldsymbol{w}}\right)\end{array}\right]$$

$$= (1-\mu_k)\boldsymbol{H}_{k-1}(\boldsymbol{w}) + \mu_k(\alpha-1)\left[\begin{array}{c}(\alpha-2)p(e_k(\boldsymbol{w}))^{\alpha-3}\left(\frac{\partial p(e_k(\boldsymbol{w}))}{\partial \boldsymbol{w}}\right)^2 \\ +(\alpha-1)p(e_k(\boldsymbol{w}))^{\alpha-2}\left(\frac{\partial^2 p(e_k(\boldsymbol{w}))}{\partial \boldsymbol{w}\partial \boldsymbol{w}}\right)\end{array}\right] \tag{4.151}$$

which is a rather complicated expression. The Hessian and the gradient simplify much when $\alpha = 2$, which was also seen with the batch algorithm.

Recursive algorithm for $\alpha = 2$

To fully determine the gradient and the Hessian when $\alpha = 2$, the gradient and Hessian of the unknown error distribution $p(e_k(\boldsymbol{w}))$ must be determined. A Parzen density estimate given as

$$\hat{p}(e_k(\boldsymbol{w})) = \frac{1}{N}\sum_{j=k-N}^{k}\kappa_\sigma(e_j(\boldsymbol{w}) - e_k(\boldsymbol{w})). \tag{4.152}$$

could be applied, but since the error-distribution will be dependent on the weights and the weights are updated at each time-step instead of in batches, a different Parzen estimate will be applied. The suggested Parzen estimate will weight the newest sample the most, so

$$\hat{p}(e_k(\boldsymbol{w})) = \gamma_N \sum_{j=k-N}^{k}\lambda_r^{j-k}\kappa_\sigma(e_{j,k}(\boldsymbol{w})) \tag{4.153}$$

where $\gamma_N = \frac{1-\lambda_r}{1-\lambda_r^N}$, which assures that the estimate is a true density estimate. So λ_r controls the weighting of the samples in the Parzen estimate.

Differentiation of the Parzen-estimate (using Gaussian kernels) gives

$$\frac{\partial \hat{p}(e_k(\boldsymbol{w}))}{\partial \boldsymbol{w}} = \gamma_N \sum_{j=k-N}^{k}\lambda_r^{k-j}\frac{\partial \kappa_\sigma(e_{j,k}(\boldsymbol{w}))}{\partial \boldsymbol{w}}$$

$$= \gamma_N \sum_{j=k-N}^{k}\lambda_r^{k-j}\frac{e_{j,k}(\boldsymbol{w})}{\sigma^2}\kappa_\sigma(e_{j,k}(\boldsymbol{w}))\boldsymbol{\phi}_{j,k}(\boldsymbol{w}). \tag{4.154}$$

The above found result inserted into equation (4.149) gives the gradient

$$\hat{\boldsymbol{b}}_k(\boldsymbol{w}_{k-1}) = \mu_k \frac{\gamma_N}{\sigma^2} \sum_{j=k-N}^{k} \lambda_r^{k-j} e_{j,k}(\boldsymbol{w}_{k-1}) \kappa_\sigma(e_{j,k}(\boldsymbol{w}_{k-1})) \boldsymbol{\phi}_{j,k}(\boldsymbol{w}_{k-1}). \tag{4.155}$$

Finding the Hessian of the Parzen estimate given in (4.153) using the chain-rule gives

$$\frac{\partial^2 \hat{p}(e_k(\boldsymbol{w}))}{\partial \boldsymbol{w} \partial \boldsymbol{w}} = \frac{\gamma_N}{\sigma^2} \sum_{j=k-N}^{k} \lambda_r^{k-j} \begin{bmatrix} -\kappa_\sigma(e_{j,k}(\boldsymbol{w})) \boldsymbol{\phi}_{j,k}(\boldsymbol{w}) \boldsymbol{\phi}_{j,k}(\boldsymbol{w})^T \\ + \frac{e_{j,k}(\boldsymbol{w})^2}{\sigma^2} \kappa_\sigma(e_{j,k}(\boldsymbol{w})) \boldsymbol{\phi}_{j,k}(\boldsymbol{w}) \boldsymbol{\phi}_{j,k}(\boldsymbol{w})^T \\ + e_{j,k}(\boldsymbol{w}) \kappa_\sigma(e_{j,k}, \boldsymbol{w})) \frac{\partial \boldsymbol{\phi}_{j,k}(\boldsymbol{w})}{\boldsymbol{w}} \end{bmatrix}. \tag{4.156}$$

Since we are maximizing the information potential, we wish to make sure that the Hessian matrix $\boldsymbol{H}_k(\boldsymbol{w})$ will be negative definite at all times. In [23] it is outlined that some part of the Hessian will be negligible near a point $\boldsymbol{w}_{opt}$ due to the fact that the error $e_{j,k}(\boldsymbol{w}_{opt})$ is assumed to be independent at $\boldsymbol{w}_{opt}$. In the above expression this corresponds to the last part of the expression

$$\frac{\gamma_N}{\sigma^2} \sum_{j=k-N}^{k} \lambda_r^{k-j} e_{j,k}(\boldsymbol{w}_{opt}) \kappa_\sigma(e_{j,k}(\boldsymbol{w}_{opt})) \frac{\partial \boldsymbol{\phi}_{j,k}(\boldsymbol{w}_{opt})}{\boldsymbol{w}} \approx \boldsymbol{0}. \tag{4.157}$$

This was also discussed in the subsection on the batch algorithm. This expression is zero if the error depends linearly on the weight parameters. We have also argued that the kernel-dependent part is assumed to be small at the solution. Using the above observations and expressions and inserting these into (4.150) gives the following pseudo Hessian

$$\hat{\boldsymbol{H}}_k(\boldsymbol{w}) = (1-\mu_k)\hat{\boldsymbol{H}}_{k-1}(\boldsymbol{w}) - \mu_k \frac{\gamma_N}{\sigma^2} \sum_{j=k-N}^{k} \lambda_r^{k-j} \left[\kappa_\sigma(e_{j,k}(\boldsymbol{w})) \boldsymbol{\phi}_{j,k}(\boldsymbol{w}) \boldsymbol{\phi}_{j,k}(\boldsymbol{w})^T\right]. \tag{4.158}$$

Inserting $\boldsymbol{w} = \boldsymbol{w}_{k-1}$ gives

$$\hat{\boldsymbol{H}}_k(\boldsymbol{w}_{k-1}) = (1-\mu_k)\hat{\boldsymbol{H}}_{k-1}(\boldsymbol{w}_{k-1}) - \mu_k \frac{\gamma_N}{\sigma^2} \sum_{j=k-N}^{k} \lambda_r^{k-j} \left[\kappa_\sigma\left[e_{j,k}(\boldsymbol{w}_{k-1})\right] \boldsymbol{\phi}_{j,k}(\boldsymbol{w}_{k-1}) \boldsymbol{\phi}_{j,k}(\boldsymbol{w}_{k-1})^T\right]. \tag{4.159}$$

Since we do not know $\hat{\boldsymbol{H}}_{k-1}(\boldsymbol{w}_{k-1})$ we approximate it with $\hat{\boldsymbol{H}}_{k-1}(\boldsymbol{w}_{k-2})$ which is a good approximation near the maximum of the cost-function.

The Hessian given in (4.159) have resemblance to the expression given for the weighted autocorrelation matrix of the RLP method given in (3.22).

The algorithm and computational complexity

The algorithm is called "RQEGN" for "Recursive Quadratic Entropy Gauss Newton". Below is given a layout on how the algorithm works.

- Step 1 Initialization
  Choose initial weights $\boldsymbol{w}(0)$

  Select the number of samples used for parzen estimate N.

  Select forgetting factors λ_r and λ.

  Select step size η

  Initialize $\hat{\boldsymbol{H}}(\boldsymbol{w}_0) = -\boldsymbol{I}\delta$ and $\hat{\boldsymbol{b}}_N(\boldsymbol{w}_0) = \boldsymbol{0}$ where the selection of δ will be discussed below.

- Step 2 Update formula

  for $k = N+1..Nsamp$

  calculate $\hat{\boldsymbol{b}}_k(\boldsymbol{w}_{k-1})$

  calculate $\hat{\boldsymbol{H}}_k(\boldsymbol{w}_{k-1})$

  update filter coefficients according to

$$\boldsymbol{w}_k = \boldsymbol{w}_{k-1} - \eta \left[\hat{\boldsymbol{H}}_k(\boldsymbol{w}_{k-1})\right]^{-1} \hat{\boldsymbol{b}}_k(\boldsymbol{w}_{k-1}) \tag{4.160}$$

  end

The step-size η is selected to be constant during the adaption. A more expensive selection of the step-size would be to select the step size such that $\widehat{V_{2,k}}(\boldsymbol{w_{k-1}}(\eta_{k-1})) < \widehat{V_{2,k}}(\boldsymbol{w_{k-1}}(\eta_k))$. This would require additional computations of the cost-function.

The initialization of the pseudo Hessian is called a *soft-constrained initialization* [15] and is done to ensure negative definiteness of the Hessian when $N < L$. L is the number of model weights and N is the number of points used in the Parzen window estimate. Experimentally I have found that a good selection of δ is $\approx \frac{\sigma^2}{10}$, where σ is the kernel width.

The algorithm was implemented in MATLAB and denoted as "RQEGN.m".

Computational complexity

When L is not that large, the number of samples used in the PDF-estimate will increase the computational complexity approximately as $\approx L^3 + (20N)L^2 + NL$. So

when $N < L$ the computational complexity will be $\mathcal{O}(L^3)$ (Gaussian elimination in solving the system of linear equations) and when $N > L$ then the computational complexity will be $\mathcal{O}(NL^2)$. In comparison the RLP-method uses around $\mathcal{O}(L^2)$ operations.

Investigation of the Hessian-matrix

The Hessian matrix was in section (4.6.1) page (57) and in the previous section simplified since we want the Hessian to be negative definite at all times. As to see what happens with the Hessian matrix in a simple AR(2)-prediction setup the eigenvalues are observed when including and excluding the term $\frac{e_{j,k}(\boldsymbol{w})^2}{\sigma^2}$ in the expression for the pseudo Hessian (given here for the RQEGN algorithm)

$$\hat{\boldsymbol{H}}_k(\boldsymbol{w}_{k-1}) = (1-\mu_k)\hat{\boldsymbol{H}}_{k-1}(\boldsymbol{w}_{k-1}) + \mu_k \frac{\gamma_N}{\sigma^2} \sum_{j=k-N}^{k} \lambda_r^{k-j} \left[\left[\frac{e_{j,k}(\boldsymbol{w}_{k-1})^2}{\sigma^2} - 1 \right] \kappa_\sigma \left[e_{j,k}(\boldsymbol{w}_{k-1}) \right] \boldsymbol{\phi}_{j,k}(\boldsymbol{w}_{k-1}) \boldsymbol{\phi}_{j,k}(\boldsymbol{w}_{k-1})^T \right]. \quad (4.161)$$

The main purpose of this investigation is to see how big the difference are in performance of the algorithm when including or excluding this term.

The AR(2)-parameters was selected to be $\boldsymbol{a} = [0.9 \quad -0.09]$. Three different noise environments was investigated:

- Sub-Gaussian (test 1 and Test 2)
- Gaussian (test 3 and test 4)
- Super - Gaussian (test 5 and test 6)

In the investigations the Recursive Quadratic Entropy Gauss Newton algorithm was used with the parameters: $N = 30$, λ_r and λ was both 0.999.

In test 1, 3 and 5 the term $\frac{e_{j,k}(\boldsymbol{w})^2}{\sigma^2}$ is excluded and in test 2,4 and 6 the term is included. The eigenvalues of the Hessian was plotted as a function of the time index in the different situations. Figure (4.10) page (68) show the eigenvalues of the

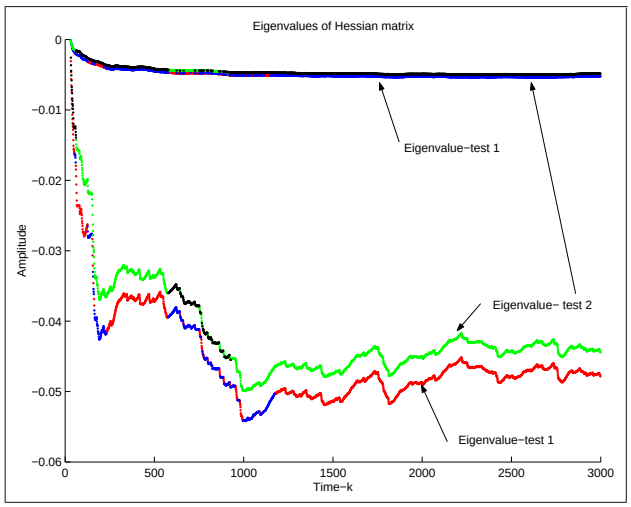


Figure 4.10: The eigenvalues in test 1 and test 2 as a function of time-k

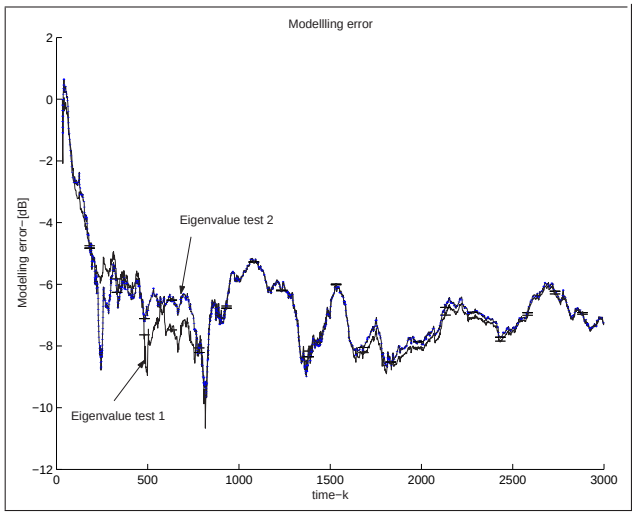


Figure 4.11: Modelling error in test 1 and test 2

Hessian matrix of test 1 and test 2. The similarities between the eigenvalues are observed, in that they follow the same trend. Figure (4.11) page (69) shows the corresponding modelling error[14] which shows similar result in both test cases. In a Gaussian environment the eigenvalues are behaving equally whether or not the term is included. This is shown in figure (4.12) page (69), where it is a bit difficult to see the difference. The modelling error, see figure (4.13) page (70) shows no difference between the two scenarios. It is noted that the Hessian is negative definite in both the sub-Gaussian and Gaussian environment. In the super Gaussian environment

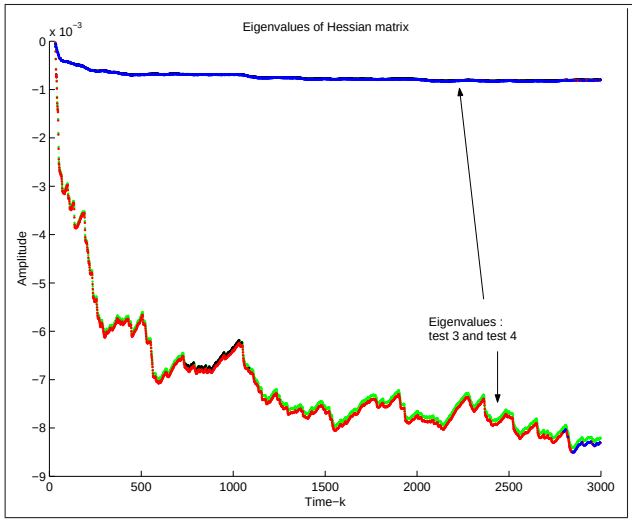


Figure 4.12: The eigenvalues of test 3 and test 4 as a function of time-k.

[14] which is defined as

$$ME = \log_e \frac{||\boldsymbol{w}_{opt} - \boldsymbol{w}||_2^2}{||\boldsymbol{w}_{opt}||_2^2}$$

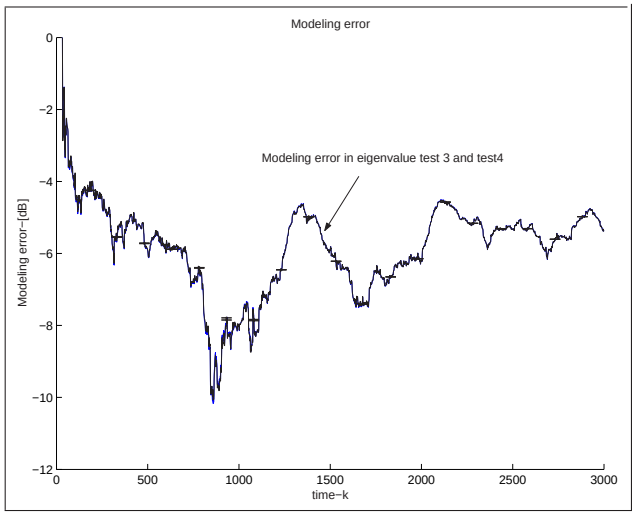


Figure 4.13: Modelling error from test 3 and test 4

the difference between excluding and including the term shows to have a big influence on the eigenvalues. The results from test 5 can be seen in figure (4.14) page (70) which shows that the eigenvalues is negative (It may be hard to see for the small valued eigenvalue, but it is below zero), while the eigenvalues is positive and in some cases close to singular in test 6 which can be seen from figure (4.15) page (71) in which the term is included. The modelling error of test 5 and test 6 can be seen from

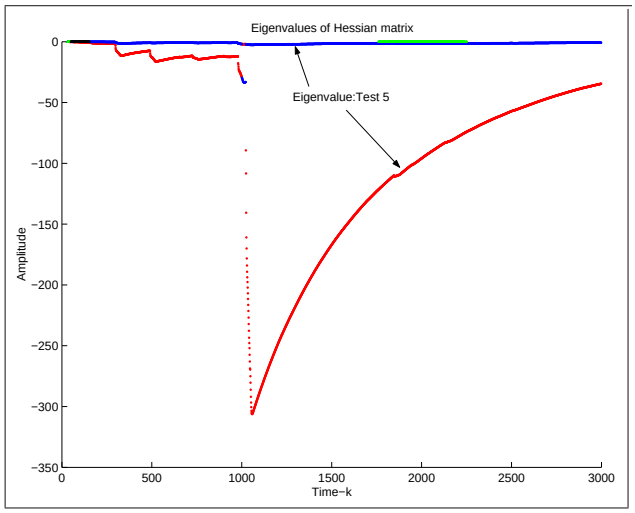


Figure 4.14: The eigenvalues in test 5

figure (4.16) page (71) clearly showing the nice convergence in the case of avoiding the term $\frac{e_{j,k}(\boldsymbol{w})^2}{\sigma^2}$ and what happens when the term is included.

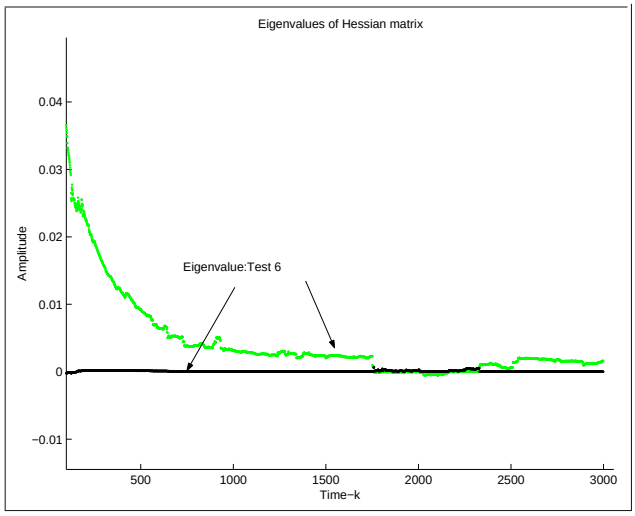


Figure 4.15: The eigenvalues in test 6

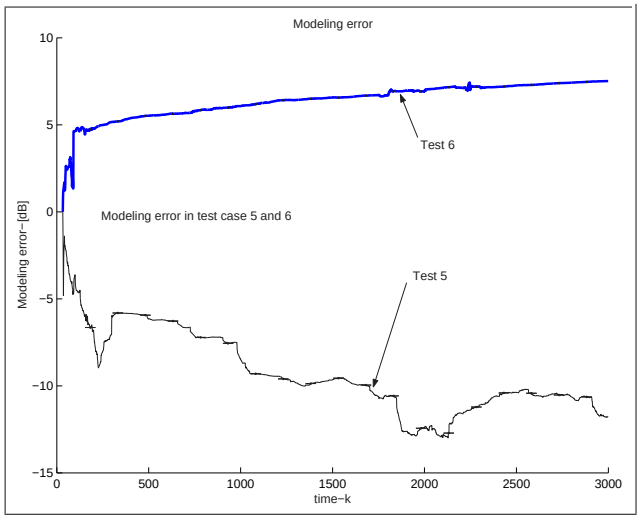


Figure 4.16: The modelling error of test 5 and test6

4.7 Discussion

This chapter has introduced the information theoretic method by introducing the Kullback Leibler divergence and minimizing the distance between the true/observed joint density function and model joint density function, which due to the additive noise model equals the error-distribution. It has been shown that the generalization error, which is found by minimization of the Kullback leibler divergence can be rewritten into a Renyi generalization error expression. Further the similarities between the two different measures have been discussed. The parzen window is introduced as an estimator of $p(\boldsymbol{x}, y)$ and $p(e)$ in the generalization error expression. Two different methods of determining the optimal width parameter in the case of unimodal data was investigated. It was further found that the generalization error

using Parzen estimates for $p(\boldsymbol{x}, y)$ corresponds to regularization. In the last section a stochastic, batch and recursive method was suggested for minimizing the Renyi generalization error.

Chapter 5

Experimental tests - simulations

In this section simple AR(3) processes will be used in an prediction setup to test the different information theoretic methods presented in the last chapter, for different environments. In the first section a discussion on how to evaluate performance is given. In preceding subsections an investigation of the parameters of the RQEGN and Batch algorithm will be given (shown in appendix). An investigation of the performance of the SIG algorithm in a prediction setup and system identification setup is given. A comparison of the RQEGN, RLP and GNLMS method is given in section (5.2.3) page (84) when mixed environments are investigated (bi/tri-modal distributions) in a AR(3)-setup. In the last section the SIG/RQEGN, NLMP/RLP and GNLMS algorithms will be tested in a hearing aid test setup.

5.1 How to evaluate performance

When selecting a measure of performance in evaluating the different algorithms, one have to be critical in the sense that the error measure might favor one algorithm over another. The choice of performance measure is divided into two groups: *Toy Examples* and *Realistic examples*.

Toy examples
By toy-examples we mean that the model-parameters, from which the data was generated are known. When this is the case some authors use the modelling error as a performance measure [20][14]. This measure is defined in this project to be

$$ME = \log_e \frac{||\boldsymbol{w} - \boldsymbol{h}||_2^2}{||\boldsymbol{h}||_2^2} \tag{5.1}$$

where $\boldsymbol{w}$ is the model weights, which are adjusted by the algorithm, and $\boldsymbol{h}$ is the optimal system weights which are used for the toy example. The natural logarithm have been used. The modelling error gives a good idea of the system mismatch when the model parameters have converged.

In the toy-examples, using stochastic data generated with MATLAB, normally several runs are performed using different signal realizations. From the different runs an

average value can be obtained as well as a spread on the average value. The spread of the mean value is given as

$$\sigma_{\bar{x}} = \frac{1}{\sqrt{N}}\sigma_x \tag{5.2}$$

where σ_x is the spread on the data.

Realistic examples
In many papers as well as books the mean square error (MSE)-measure is used as a performance measure. This measure is relevant to use, when the expected distribution of the error have a second order moment. In impulsive noise environments, the second order moment do not exist. For finite sample sizes a second order moment exists however, the value can be rather large, making it a non-robust measure. Another measure, which might be used, is the one-norm of the error $||e||_1$.

The goal of the adaptive filters are to model the error-distribution (as shown by minimization of KL-divergence). To measure the goodness of a distribution the average negative log-likelihood is used [3]

$$ANLL = -\frac{1}{M}\sum_{i=1}^{m} \log \hat{p}(e_i). \tag{5.3}$$

The error-distribution ($p(e_i)$) is modelled using a Parzen window with a fixed width parameter.

Algorithm selection in the simulations

The cost-functions, which the different algorithms are minimizing are shown in table (5.1) page (74). The different tests, except for the algorithm specific test, will include

method	cost function
NLMP/RLP	$E\{\|e\|^2\}$
GNLMS	$E\{\|g(e)\|^2\}$
SIG($\alpha = 2$)/RQEGN /Batch	$G_R = -\log \int_{-\infty}^{\infty} p_e(e)p(\boldsymbol{x}, y)d\boldsymbol{x}dy$

Table 5.1: Cost-function of the different algorithms

at least one of the algorithms from each of the different cost functions.

5.2 Simple Prediction setup-AR(3)

In the next three subsections, the main subject is identification of parameters in an AR(3) setup. As mentioned earlier the AR-process can be used in a prediction setup modelled as in figure (5.1) page (75). The parameters of the AR(3) process is

$$\boldsymbol{a} = [0.9 \quad -0.09 \quad 0.1]^T.$$

The poles of the characteristic equation are determined to be

$$p_{1,2,3} = \begin{cases} 0.9203 \\ -0.0101 + i0.3295 \\ -0.0101 - i0.3295 \end{cases}$$

which is within the unit-circle and hence the AR-process is a stable process. The different signals in accordance with figure ((5.1) page (75)) is

$$u_k = \sum_{i=1}^{3} a_i u_{k-i} + v_k \tag{5.4}$$

$$x_k = u_{k-1} \tag{5.5}$$

$$y_k = u_k \tag{5.6}$$

where it is apparent that the delay should be selected to 1. The process v_k is in most cases an i.i.d. process. In the tests of the information theoretic methods (RQEGN and the Batch algorithm) Gaussian, super-Gaussian and colored super-Gaussian noise are used for v_k. In the comparison of the norm-algorithms with the

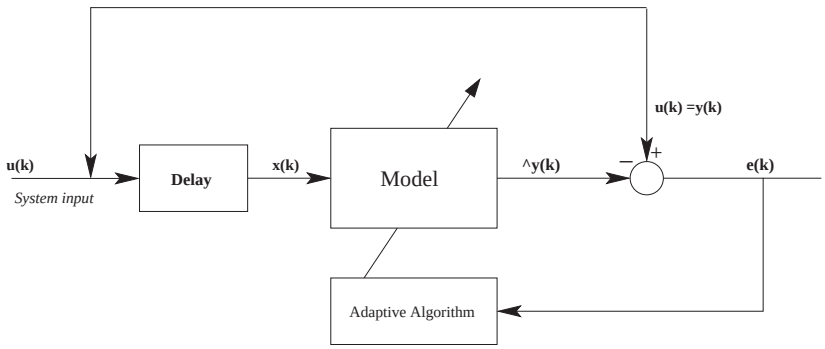


Figure 5.1: Prediction setup

information theoretic methods other distributions for v_k are tested.

5.2.1 Investigation of the RQEGN and Batch algorithm

Investigations of the recursive quadratic entropy Gauss Newton and Gauss Newton Batch method have been performed to get acquainted with the algorithms. The algorithms have been tested in the simple AR- test setup described in subsection (5.2)

page (74). The investigation of the RQEGN algorithm can be seen from appendix (B.1) page (113). The investigation of the Batch algorithm can be seen from appendix (B.2) page (123).

Main results from the investigation of the RQEGN algorithm:

As to summarize: the algorithm was tested in three different environments. The results was very similar whether the environment was a Gaussian or super Gaussian except for a better system mismatch in the super Gaussian case. The most important parameters seem to be λ, δ and the number of samples N which indirectly controls λ_r.

Main results from the investigation of the Batch algorithm:

The batch algorithm is minimizing an undisturbed error-distribution, which should make it more "correct". The algorithm is updated in Batches but requires that at least $N > L$ to be efficient. This algorithm has only been developed for comparison purposes with the recursive method. The Batch and recursive method show to have more or less similar performance, except for a more smooth convergence when using the RQEGN method.

5.2.2 Investigation of the SIG-algorithm

The Stochastic Information Gradient (SIG) algorithm was first introduced in [6] intended to be an online adaptive algorithm. The SIG-algorithm is derived from the batch-update selecting $N = 2$ (The number of samples to evaluate cost-function). The SIG-algorithm updates the unknown filter-weights according to

$$\boldsymbol{w}_k = \boldsymbol{w}_{k-1} + \eta \boldsymbol{b}_k(\boldsymbol{w}_{k-1}). \tag{5.7}$$

To verify the implementation of the SIG-algorithm, one of the test-examples given in [6] have been tried with this algorithm.

A time-prediction problem is used to evaluate the performance of the SIG-algorithm. The signal to be predicted is a signal of mixed sines :

$$x(t) = sin(20t) + 2sin(40t) + 3sin(60t). \tag{5.8}$$

The signal is sampled with a frequency of 100Hz, which gives at period of around 32 samples. The signals to the prediction filter is given as

$$u_k = sin(20t_k) + 2sin(40t_k) + 3sin(60t_k) \tag{5.9}$$

$$x_k = u_{k-1}. \tag{5.10}$$

where t_k is the current sample. The prediction scheme that is used is also referred to as a *One-step linear prediction in the forward direction*. In this setup a FIR-filter using two coefficients are used. In [6] a comparison is made to the traditionally LMS-algorithm, but here we compare the SIG algorithm with the NLMS algorithm

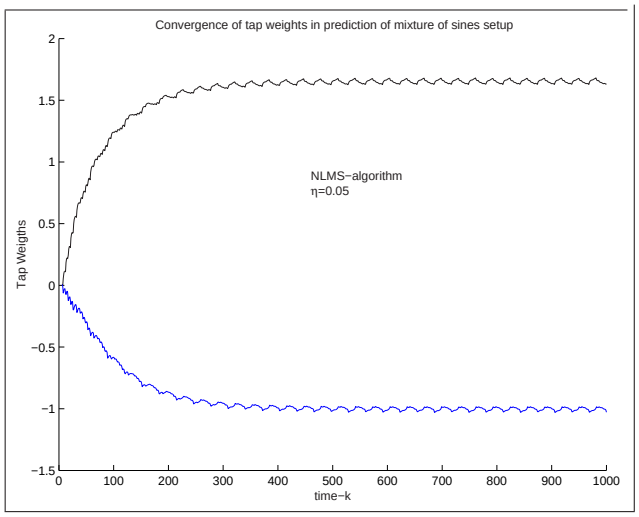


Figure 5.2: Convergence of the filter-taps using a NLMS-algorithm

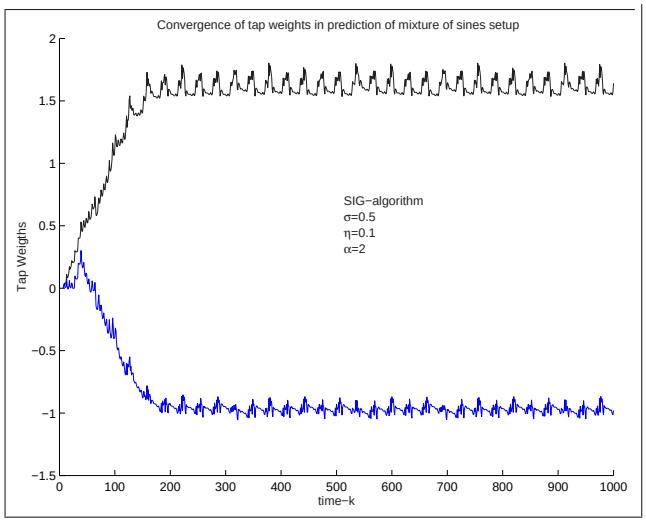


Figure 5.3: Convergence of the filter-taps using the SIG-algorithm

since this works better for environments with correlation in the input signal to the adaptive filter.

From [6] the optimal coefficients were determined to be $\boldsymbol{w} \approx [1.65 \quad -1]$. The convergence of the NLMS algorithm can be seen from figure (5.2) page (77) and the convergence of the SIG algorithm from figure (5.3) page (77) where it is clear that the solution given by the SIG algorithm is fluctuating more than the solution given by the NLMS algorithm. In [6] it is stated that the convergence of the filter coefficient of the SIG-algorithm is much smoother than the LMS algorithm with the same convergence. Using the NLMS algorithm ensures that the step size is adjusted according to the maximum eigenvalue of the autocorrelation matrix at the input.

To investigate how the SIG-algorithm works in a stochastic environment the simple AR(3)-process described in section (5.2) page (74) is used in a prediction setup. v_k

is in this test setup Gaussian distributed.

In the simulation the unknown model weights were initialized to the true values. Figure (5.4) page (78) shows the convergence of the filter taps and figure (5.5) page (78) shows the average modelling error, where error bars have been applied. Only five (5) realizations of v_k have been tested but as seen the spread on the modelling error is not that big. The surprising result is that the SIG-algorithm *do not* converge to the AR(3)-parameters as would be expected. Instead another maximum of the cost-function is found. To make the SIG-algorithm perform better in this kind

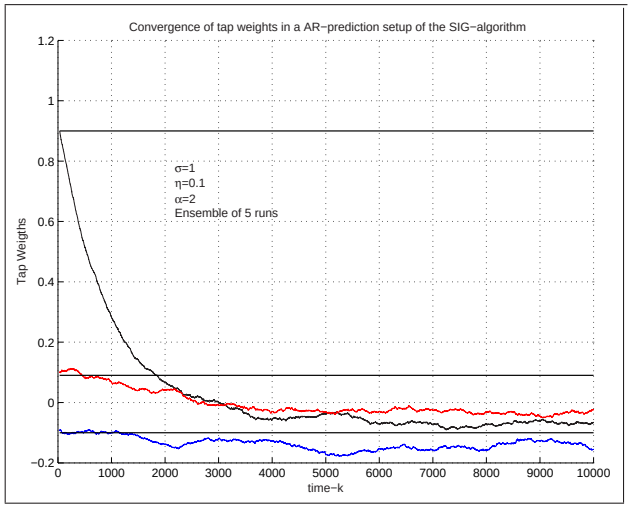


Figure 5.4: Convergence of the filter-taps to the AR-parameters using the SIG-algorithm

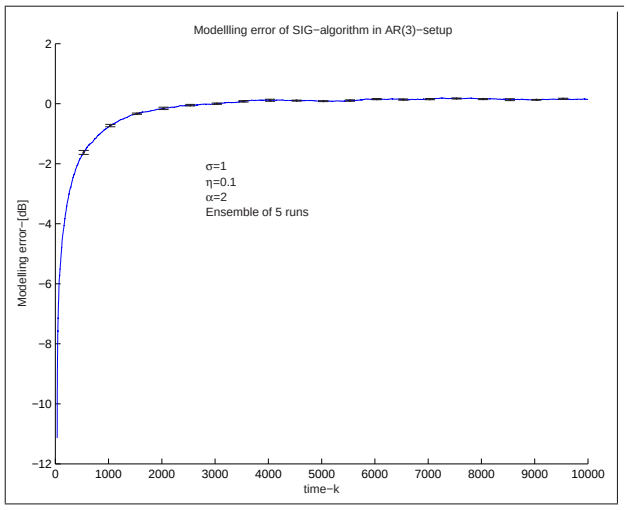


Figure 5.5: Model error of the SIG-algorithm with error-bars

of stochastic environment it was found that a delay-factor should to be introduced. Instead of using the prior sample for the density estimate, the d'th prior sample is

used. The gradient is then modified according to

$$\boldsymbol{b}_k(\boldsymbol{w}_{k-1}) = \frac{(\alpha-1)}{2^{\alpha-1}\sigma^2}\left[\kappa_\sigma(e_{k,k-d}(\boldsymbol{w}_{k-1})) + \kappa_\sigma(0)\right]^{\alpha-2} e_{k,k-d}(\boldsymbol{w}_{k-1})\kappa_\sigma(e_{k,k-d}(\boldsymbol{w}_{k-1}))\phi_{k,k-d}(\boldsymbol{w}_{k-1}). \quad (5.11)$$

When $d = 1$ then we have the original SIG algorithm. Increasing d will provide better convergence properties of the SIG-algorithm. As can be seen from figure (5.6) page (79) increasing d lead to an optimal limit in this setup around $d = 20$. The reason

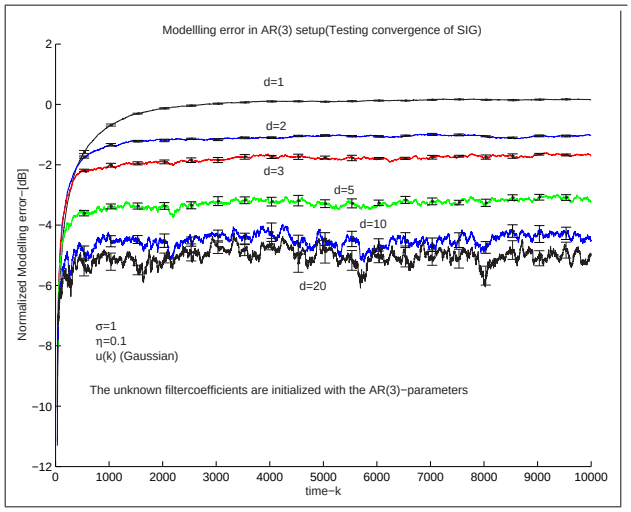


Figure 5.6: Convergence using different values of memory-depth value d

for the SIG algorithm do not converge to the true weights in the simple AR-setup is outlined in the following. For $\alpha = 2$ and $d = 1$ the stochastic gradient can be written as

$$\boldsymbol{b}_k(\boldsymbol{w}_{k-1}) = \frac{1}{2\sigma^2}e_{k,k-1}(\boldsymbol{w}_{k-1})\kappa_\sigma(e_{k,k-1}(\boldsymbol{w}_{k-1}))\phi_{k,k-1}(\boldsymbol{w}_{k-1}). \quad (5.12)$$

Given that we are at the optimum filter values, namely when $\boldsymbol{w_o} = \boldsymbol{a}$ then the expected value of the update should be zero, thus

$$E\left[\boldsymbol{b}_k(\boldsymbol{w_o})\right] = \mathbf{0}. \quad (5.13)$$

The expectation of the gradient is (the weights is left out)

$$E\left[\frac{1}{2\sigma^2}(e_{k,k-1})\kappa_\sigma(e_{k,k-1})\boldsymbol{x}_{k,k-1}\right], \quad (5.14)$$

which can also be written as

$$E\left[\kappa_\sigma(e_{k,k-1})\left[\boldsymbol{x}_k e_k - \boldsymbol{x}_k e_{k-1} - \boldsymbol{x}_{k-1}e_k + \boldsymbol{x}_{k-1}e_{k-1}\right]\right] \quad (5.15)$$

neglecting the constant term $\frac{1}{2\sigma^2}$. Since we are using a Gaussian kernel then

$$E\left[\boldsymbol{x}_k e_k - \boldsymbol{x}_k e_{k-1} - \boldsymbol{x}_{k-1} e_k + \boldsymbol{x}_{k-1} e_{k-1}\right] \geq \\ E\left[\kappa_\sigma(e_{k,k-1})\left[\boldsymbol{x}_k e_k - \boldsymbol{x}_k e_{k-1} - \boldsymbol{x}_{k-1} e_k + \boldsymbol{x}_{k-1} e_{k-1}\right]\right]. \quad (5.16)$$

The expression given by

$$E\left[\boldsymbol{x}_k e_k - \boldsymbol{x}_k e_{k-1} - \boldsymbol{x}_{k-1} e_k + \boldsymbol{x}_{k-1} e_{k-1}\right] \quad (5.17)$$

is investigated at the solution, where it should be zero. To see if the above expression is zero when $\boldsymbol{w_o} = \boldsymbol{a}$ then

$$e_k = \hat{y}_k - y_k = (\boldsymbol{w_o} - \boldsymbol{a})^T \boldsymbol{u}_{k-1} + v_k = v_k \quad (5.18)$$

and

$$\boldsymbol{x}_k = \boldsymbol{u}_{k-1} = \boldsymbol{a}^T \boldsymbol{u}_{k-2} + v_{k-1} \quad (5.19)$$

$$\boldsymbol{x}_{k-1} = \boldsymbol{u}_{k-2} = \boldsymbol{a}^T \boldsymbol{u}_{k-3} + v_{k-2} \quad (5.20)$$

is inserted into (5.17) and each term is evaluated:

$$E\left[\boldsymbol{x}_k e_k\right] = E\left[\boldsymbol{u}_{k-1} v_k\right] = 0 \quad (5.21)$$

$$E\left[\boldsymbol{x}_k e_{k-1}\right] = E\left[\boldsymbol{u}_{k-1} v_{k-1}\right] = \sigma_v^2 \quad (5.22)$$

$$E\left[\boldsymbol{x}_{k-1} e_k\right] = E\left[\boldsymbol{u}_{k-2} v_k\right] = 0 \quad (5.23)$$

$$E\left[\boldsymbol{x}_{k-1} e_{k-1}\right] = E\left[\boldsymbol{u}_{k-2} v_{k-1}\right] = 0. \quad (5.24)$$

Since we are using a Gaussian kernel ($\kappa_\sigma(e_{k,k-1}) > 0$) then

$$E\left[\kappa_\sigma(e_{k,k-1})\left[\boldsymbol{x}_k e_k - \boldsymbol{x}_k e_{k-1} - \boldsymbol{x}_{k-1} e_k + \boldsymbol{x}_{k-1} e_{k-1}\right]\right] > 0. \quad (5.25)$$

which means that $\boldsymbol{w_o} = \boldsymbol{a}$ is not the maximum found by the SIG algorithm in an AR setup.

As mentioned before using the d'th prior example gave better convergence. This means that the term, which did not give zero (5.22) now gives

$$E\left[\boldsymbol{x}_k e_{k-d}\right] = E\left[\boldsymbol{u}_{k-1} v_{k-d}\right] \approx 0 \quad (5.26)$$

when the d value is chosen properly. The AR process in some sense determines this depth factor. In the AR(3)-example it was shown that a value of d of around 20 would make the SIG-algorithm converge to values close to the minimum. Using a value of $d = 20$ with Gaussian distributed noise v_k starting from $\boldsymbol{w}_0 = \boldsymbol{0}$ gives a convergence of the filter-taps as shown in figure (5.7) page (81). The modelling error is seen from figure (5.8) page (81). From the filter-taps and modelling error it still looks as if there is a little bias in the system, which is due to the fact that the $d = 20$ only gives that $E\left[\boldsymbol{x}_k e_{k-20}\right] \approx 0$.

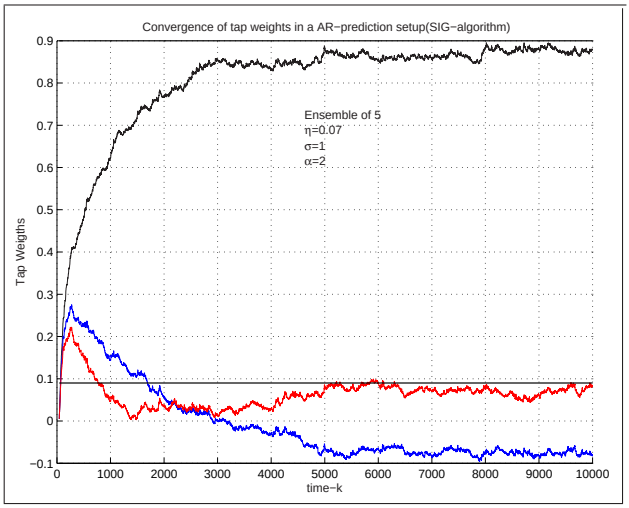


Figure 5.7: Convergence of the filter-taps to the AR-parameters using the SIG-algorithm with $d = 20$.

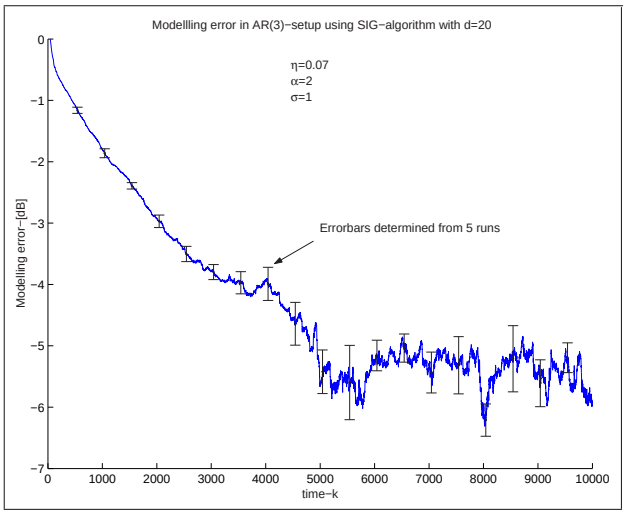


Figure 5.8: Model error of the SIG-algorithm with error-bars when using $d = 20$

Invoking the independence assumption

By looking at the requirements to the gradient at the optimal model parameters one actually finds that invoking the independence assumption, mentioned in the chapter on norm-algorithms (3.1) page (15), equation (5.17) is zero. The fourth requirement in the independence assumption does not necessarily have to be fulfilled when using the SIG algorithm, which is shown in the next couple of examples. The SIG-algorithm will now be tested in a system identification setup (figure (5.9) page (82)) where the plant is a FIR-filter of length $L = 5$ (the parameters were $\boldsymbol{h} = [0.9 \ -0.09 \ 0.1 \ 0.5 \ 0.20]^T$). The system input x_k will in all cases be Gaussian noise with unit variance but different characteristics of the added signal v_k are tested. The SIG algorithm is compared to its counterpart the NLMP algorithm. The following

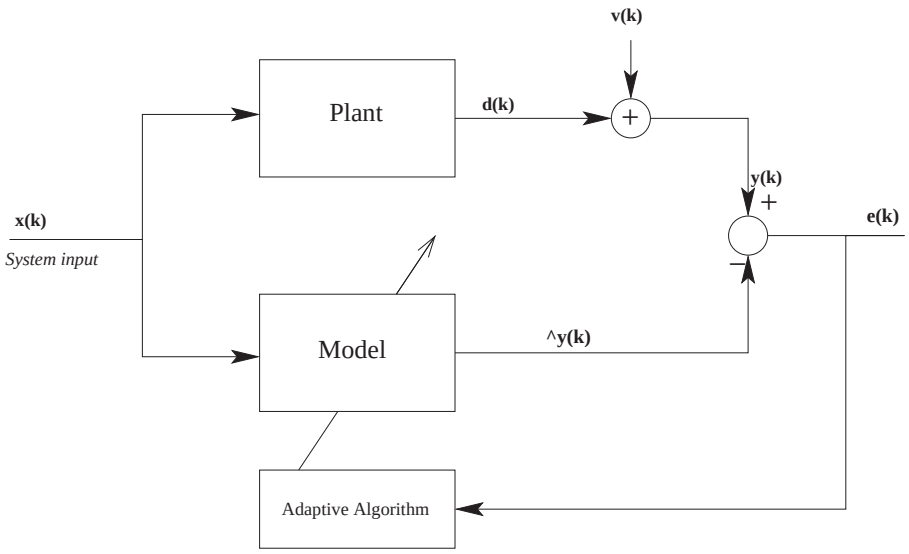


Figure 5.9: System Identification setup

tests are considered

- v_k is Gaussian noise with variance of 0.01.(Test SIG 1)
- v_k is $S(1.2)S$ distributed with a dispersion of ≈ 0.015. (Test SIG 2)
- v_k is uniform distributed with a variance of 0.01. (Test SIG 3)

where x_k in all cases are Gaussian distributed with variance one.

Test SIG 1
The modelling error from the first test can be seen from figure (5.10) page (83) using the NLMS (NLMP with $p = 2$) and the SIG algorithm with $\alpha = 2$ and $\sigma = 0.5$. The system mismatch of the SIG algorithm was around $-7dB$ which was about $2dB$ above the NLMS algorithm. The convergence speed of the two methods was selected to be more or less equal as to be able to compare the system mismatch. The SIG-algorithm seem to be a bit more noisy than the NLMS algorithm. Choosing another value for σ did not have big impact on the actual convergence or mismatch, except that one has to change the step-size.

Test SIG 2
In this test setup both the NLMP with p=2 and p=1.2 was tested. For the SIG algorithm $\sigma = 1$ and $\alpha = 2$. The modelling error of the algorithms can seen from figure (5.11) page (83). The system mismatch of the NLMP algorithm with $p = 1.2$ is the lowest, around $-7.2dB$, while the SIG algorithm achieves around $-5.9dB$ in system mismatch. Using a $p = 2$ shows that the algorithm is very sensitive to outliers, which is seen when the algorithm looses track of the filter coefficients. The

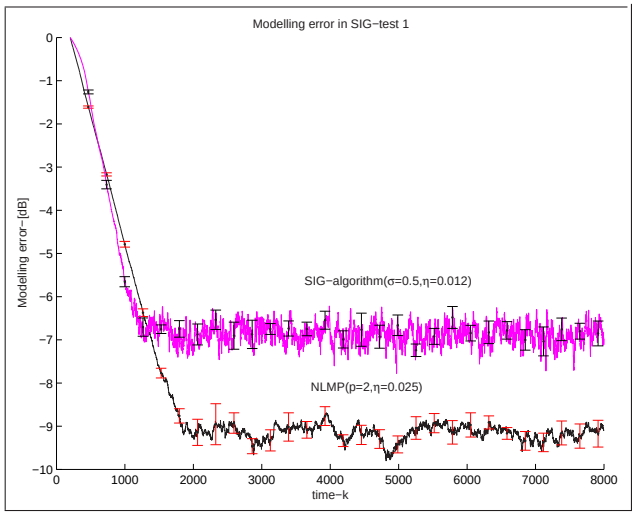


Figure 5.10: System Identification setup using Gaussian noise for v_k

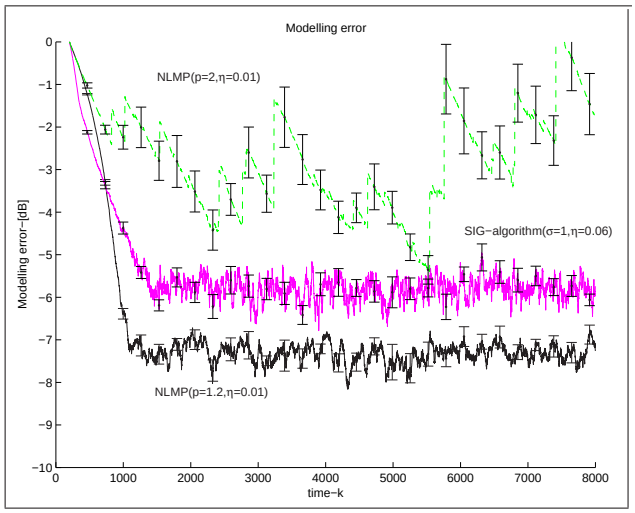


Figure 5.11: System Identification setup using $S(1.2)S$ noise for v_k

SIG algorithm actually seem to be a very robust to impulsive noise environments.

Test SIG 3
In the last test p=2 for the NLMP and $\sigma = 0.5$ for the SIG algorithm. The two methods are quite close in performance in this test, which is seen from the modelling error in figure (5.12) page (84). The system mismatch though was around a dB lower with the NLMP algorithm (around $-8dB$).

Discussion

The stochastic information gradient have been tested in a prediction setup and a system identification setup. The prediction setup showed that the SIG algorithm did not converge to the true model parameters, but converged to some other parameters.

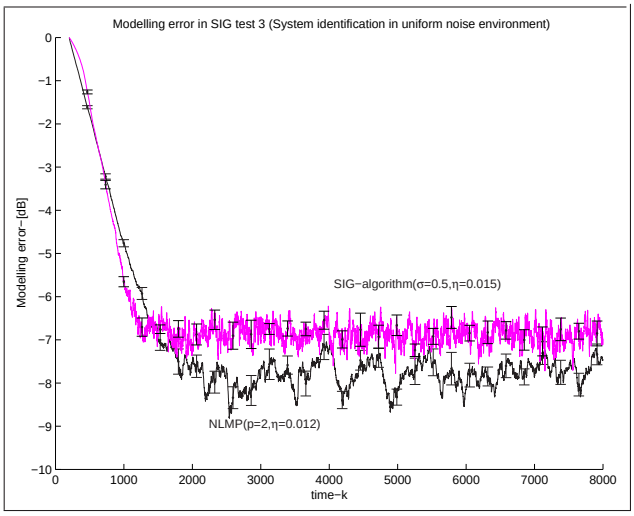


Figure 5.12: System Identification setup using uniform noise for v_k

The SIG algorithm converged in some sense if the d'th prior sample was used instead of the last sample in the PDF-estimate of $p_e(e)$. It was then mentioned that under independence assumption the algorithm would converge. This was tested using a system identification setup. The performance achieved with the SIG algorithm was quite good considering that the algorithm was converging nicely in both sub-, super- and Gaussian noise environments. For the NLMP algorithm we made the prior assumption that we knew the signal statistics, and also showed the drastic choice of $p = 2$ in an $S(1.2)S$ environment.

5.2.3 Comparison of Selected Robust Adaptive Algorithms in a mixed noise environment

In this subsection the GNLMS, RLP and RQEGN algorithm will be compared in other statistical environments than the previous environments investigated. The following different distributions for v_k in the AR(3)-prediction setup will be investigated

- Test Mixed 1: Bi-Gaussian process with parameters $\mu_1 = -5$,$\mu_2 = 5$, $\sigma_1^2 = 1/2$, $\sigma_2^2 = 3/4$.
- Test Mixed 2: Bi-Gaussian process with parameters $\mu_1 = 0$,$\mu_2 = 10$, $\sigma_1^2 = 1/2$, $\sigma_2^2 = 3/4$.
- Test Mixed 3: Mixed environment
  $v_k \in \left[N(\mu_1, \sigma_1^2) + Cauchy(\mu_2, \gamma) + Uniform(a, b)\right]$, where the parameters are $\mu_1 = -5, \sigma_1 = 1, \mu_2 = 5, \gamma = 1, a = 10, b = 13$. The way the stochastic signal is generated is to select with equal probability randomly between samples from the three distributions.

The modelling error have been produced as an average over 10 runs with different realizations of the noise to get an idea of the spread on the mean modelling error.

Test Mixed 1

In the first simulation the different investigated algorithms had the following values:

- RLP: $p = variable, \lambda_p = 0.95, \lambda = 0.999, \delta = 1e-3, \omega = 0.1$.
- RQEGN: $N = 50, \lambda_r = 0.98, \lambda = 0.999, \eta = 12, \delta = -1$
- GNLMS : $N = 300, \eta = 15$

The modelling error seen from figure (5.14) page (85) show that the lowest system mismatch is obtained by the RQEGN algorithm at $-8dB$ ($k = 1000$). The system mismatch obtained for the RLP and the GNLMS algorithm is very alike and are around $-4.5dB$. A typical error-signal (produced by the GNLMS algorithm) and

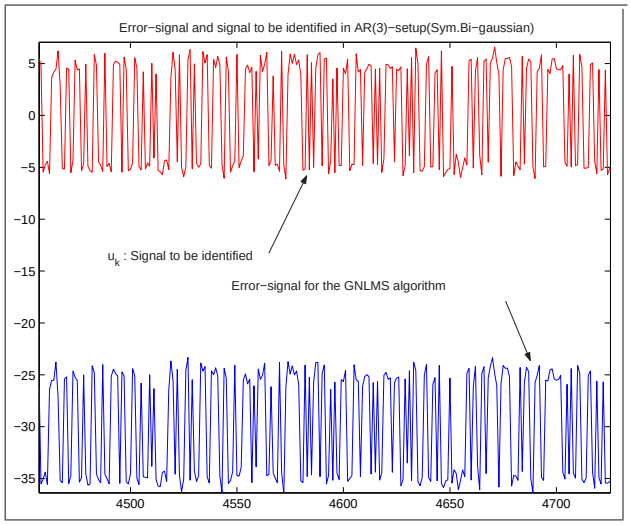


Figure 5.13: Error signal produced when using the GNLMS algorithm compared to the expected signal.

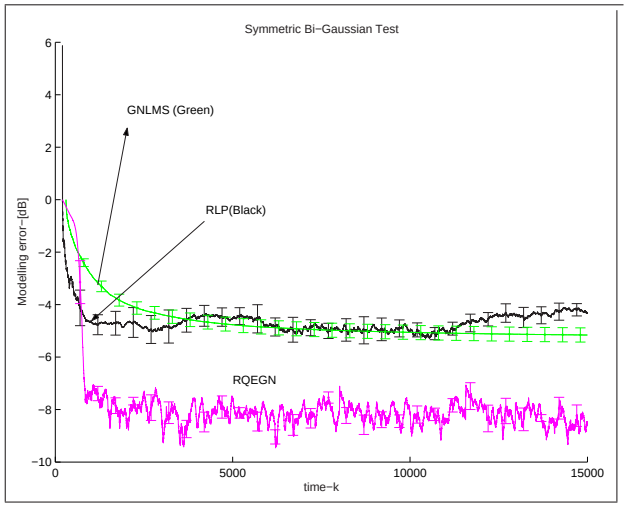


Figure 5.14: Modelling error of the three algorithms in Test mix 1.

its true value (v_k) is shown in figure (5.13) page (85). The different estimated PDF of the error can be seen from figure (5.15) page (86) showing best performance of the RLP-algorithm. It should be mentioned that the distributions shown, is generated from one realizations of the error and not as an average of all the runs. The difference

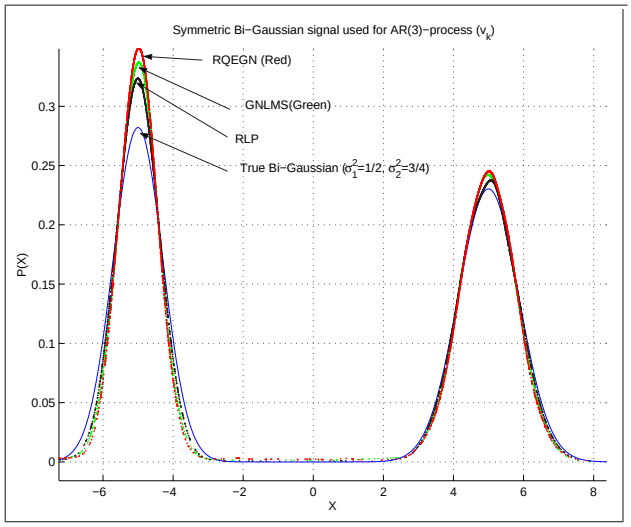


Figure 5.15: The Probability density function of one realization of the error for the different algorithms, compared to the true distribution (distribution of signal to be identified)

though are very small, and all the three algorithms identify the true distribution very nicely. In all cases a Parzen window estimator with a width of $\sigma = 0.2$ was used to plot the distributions and to estimate the ANLL (average negative log-likelihood). The ANLL was determined from each of the 10 realizations. For the GNLMS algorithm $ANLL = 1.908(0.019)$, RLP $ANLL = 1.880(0.014)$ and for the RQEGN $ANLL = 1.82(0.010)$. The number in parenthesis is the spread on the average value. The ANLL only shows a little difference between the three methods, which is expected from the similarities of the probability density functions.

Test Mixed 2

The different algorithms had the same values as in the simulation above. The signal to be identified was changed with respect to its mean-value. This change in mean value has a big influence on the norm-based algorithms. The modelling error of the three methods is seen from figure (5.16) page (87) showing the best overall performance of the RQEGN method. The nice thing about the information theoretical methods is that it works directly on the error-distribution and not on a specific norm. This makes the method more insensitive to changes (such as mean value) in the distribution. The system mismatch is around $-8dB$ as before and convergence is achieved around N=1000. The RLP algorithm still achieves a system mismatch as before, around $-4.5dB$ but the actual distribution is shifted in mean value into the symmetric case observed in "Test Mixed 1". The estimated error-distributions

from the different methods can be seen from figure (5.17) page (87). The algorithm

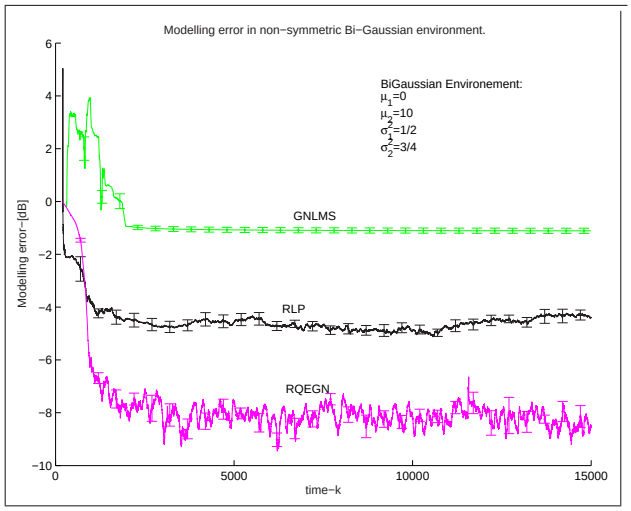


Figure 5.16: Modelling error produced using the three algorithms.

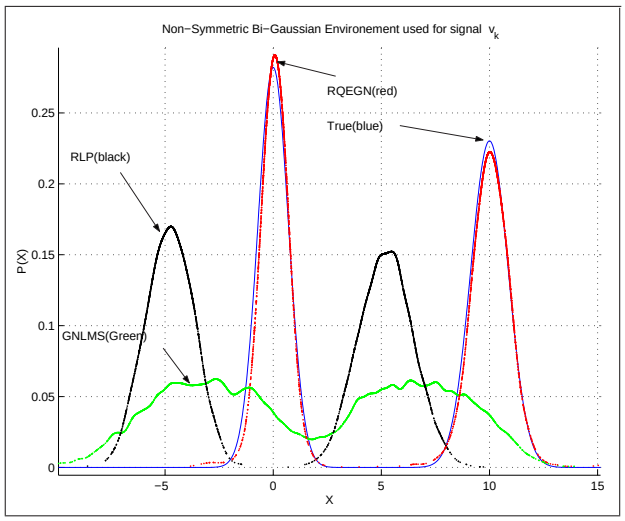


Figure 5.17: Estimated PDF for the three different methods.

which shows the biggest deviation from the simulation with a symmetric distribution was the GNLMS algorithm. The modelling error using this method was around $-1dB$. From the estimated PDF using one realization of the error with the GNLMS algorithm clearly shows that this distribution is far from the true distribution. The only algorithm which models the underlying distribution, is the RQEGN algorithm.

The ANLL have been calculated from the results of these three algorithms. The results are seen from table (5.2) page (88) obtained using 10 realizations of the noise. As expected, the RQEGN algorithm is having the lowest ANLL.

Method	ANLL
GNLMS	3.56(0.08)
RLP	2.39(0.01)
RQEGN	2.12(0.02)

Table 5.2: ANLL for the three methods in *test mixed 2*

Test Mixed 3

Again the same parameters was used for this setup as in the two other setups. The modelling error can be seen in figure (5.18) page (88). The GNLMS algorithm have a system mismatch at around $-1dB$. The RLP algorithm is quite close to the system mismatch obtained using the RQEGN algorithm, which is around $-8dB$. The small difference in modelling error apparently still have quite a influence on the distribution of the error. The error distribution

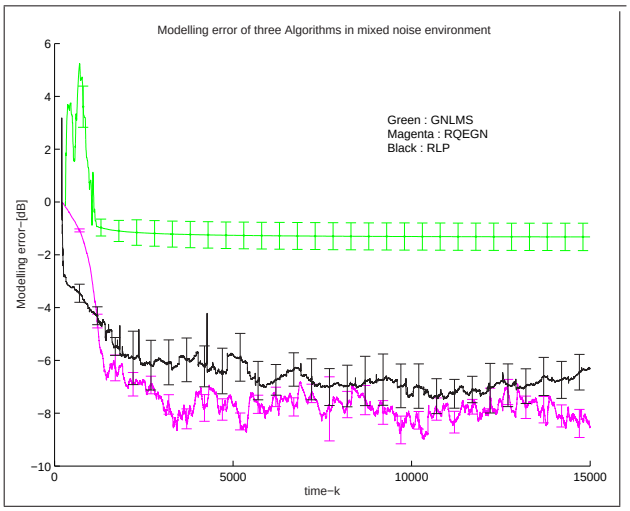


Figure 5.18: Average modelling error for the three different algorithms in the mixed environment.

found by using a Parzen window estimator on the error of the three different algorithms, show that the RQEGN is the one, which resembles the true distribution the most, see figure (5.19) page (89). Also in this test the GNLMS is far from the true distribution. The ANLL achieved by the three methods is seen from table (5.3) page

Method	ANLL
GNLMS	3.67(0.06)
RLP	3.20(0.02)
RQEGN	2.98(0.006)

Table 5.3: ANLL for the three methods in *test mixed 2*

(88), showing as expected lowest ANLL using the RQEGN method.

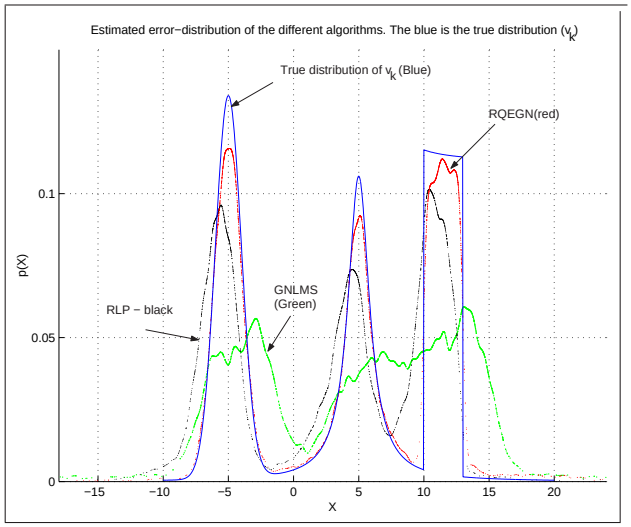


Figure 5.19: The Probability density function of one realization of the error determined using the different algorithms.

Discussion

What have been shown in this subsection on mixed noise environments is that the information theoretic methods are really efficient in modelling the underlying distribution when these are not just simple distributions like the Gaussian, Uniform or some $S(\alpha)S$ distribution. If the distribution is more special (non-symmetric) the information theoretical method (RQEGN in this case) really shows its efficiency. One drawback of the method is still the estimate of the width parameter. The width parameter was selected using the RT method. This showed to handle the bi-modal situation quite nicely.

5.3 Selected algorithms tested in an open loop hearing aid setup

To test the different proposed algorithms as well as some of the robust algorithms in a real setup, data has been obtained from a realistic hearing aid setup[1]. A hearing aid is placed in an artificial rubber ear. Noise or speech are played through the loudspeaker of the hearing aid and data is sampled synchronically at the microphone of the hearing aid. The measurement is a so-called open loop-measurement, and a setup would look "more or less" like figure (5.20) page (90) where the bar with microphone and speaker is the hearing aid. The open loop is not a true test setup, but functions to test whether or not the algorithms converges. The artificial rubber ear together with the hearing aid will in the following be denoted as the *test-setup*. Two situations have be set up for the two different signals applied to the loudspeaker

[1] With a great deal of help from Preben Kidmose which have performed the measurements.

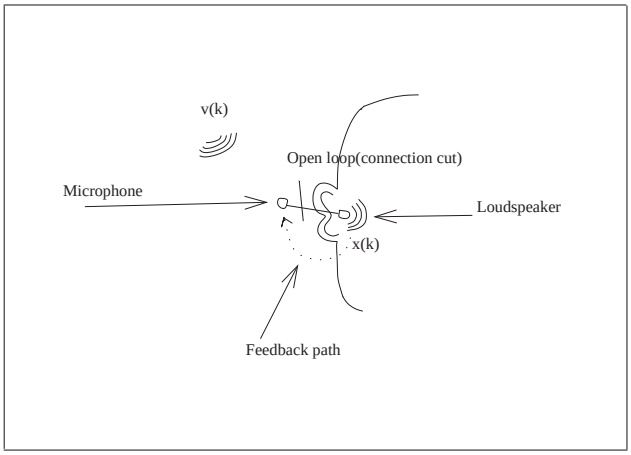


Figure 5.20: Model of test setup with the hearing aid.

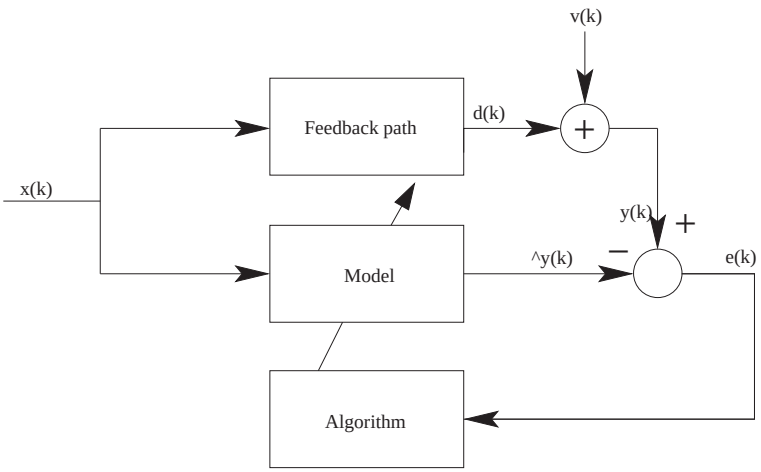


Figure 5.21: Model of the hearing aid setup

- The test-setup is put into a anechoic chamber (sound dead room). The setup is modelled as shown in figure (5.21) page (90). The ideal situation would be that v_k is zero which is not the case, since there is some inherent noise of the system[2].
- The test-setup is placed in a "simulated" open office environment which can also be modelled as shown in (5.21) page (90) but in this case the applied signal(v_k) will also contain a speech signal[3] and some background noise as well as the inherent noise.

[2]Thermical noise from the microphone and quantization noise from the converter
[3]A male news reader

The channel is well modelled by a FIR-filter since the tissue introduces delays and attenuation to the sound played in the loud-speaker.

The following data-sets have been recorded (the symbols is in accordance with the schematic sketch of the layout in figure (5.21) page (90)):

- *whitenoise.wav* Gaussian white noise signal which is played through the hearing aid loudspeaker ($\boldsymbol{x}$).
- *whitenoise_ anechoic_ mic.wav* Microphone signals recorded synchronized with *whitenoise.wav*. The test-setup have been placed in an anechoic chamber ($\boldsymbol{y}$).
- *whitenoise_ ambient_ mic.wav* Microphone signal sampled synchronically with *whitenoise.wav*. The test-setup have been placed in an open office environment($\boldsymbol{y}$).
- *Track13.wav* Speech signal ("DanTale" a standard speech signal - (Spor 13)) played through the hearing aid loudspeaker($\boldsymbol{x}$).
- *Track13_ ambient_ mic* Microphone signal recorded in an open office environment (Speech signals/noise etc.). The samples is synchronous with *Track13.wav*($\boldsymbol{y}$ in the adaptive filter setup).
- *Track13_ anechoic_ mic.wav* Microphone signals recorded where the test-setup is placed in an anechoic chamber($\boldsymbol{y}$). The samples were sampled synchronously with the samples in *Track13.wav*.

All of the data have been sampled at a sample-frequency of $F_s = 32kHz$. Each of the recordings were around 140 seconds. Only the first part of the signal is used for the investigation (The first 0.5-10 seconds). Due to the high sample-rate this gives a signal length of $nsamp = 3.04e5$.

During the measurements it has been assumed that the hearing aid have not been moved[4], meaning that the channel remains more or less fixed (same impulse-response) during the different tests.

The following test-cases have been set up

- *Test 1*
  $\boldsymbol{x}$: White noise signal. $\boldsymbol{y}$: Synchronized sampled signal at the microphone when the test-set is in an anechoic chamber (Microphone noise present).
- *Test 2*
  $\boldsymbol{x}$: White noise signal. $\boldsymbol{y}$: Synchronized sampled signal at the microphone when the test-setup is placed in an open office environment.
- *Test 3*
  $\boldsymbol{x}$: DanTale signal. $\boldsymbol{y}$: Synchronized sampled signal at the microphone when the test-setup is in an anechoic chamber (Microphone noise present).
- *Test 4*
  $\boldsymbol{x}$: DanTale signal. $\boldsymbol{y}$: Synchronized sampled signal at the microphone when the test-setup is placed in an open office environment (Speech and microphone noise present).

In *test 2 - test 4* the five algorithms which have been investigated are the RLP/NLMP, GNLMS and RQEGN/SIG.

[4]Preben Kidmose, which did the measurement, said that no practical changes to the setup have been made during the measurements.

Test 1 was made as to determine the optimal filter-length and the values of the filter coefficients. This was done using the Wiener-filter(see section (3.2.1) page (22)). The Wiener filter finds the optimal filter coefficients in a 2-norm sense.

To determine the optimal number of filter-coefficients 30000 samples was used in the estimate of the cross-correlation vector $\boldsymbol{r_{yx}}$ and autocorrelation matrix $\boldsymbol{R_{xx}}$. The linear system of equations was solved with the different filter lengths, and the filter-length was plotted against the training error and test-error. The test-error was determined using another 30000 samples of the signal. The filter length was varied in the range L=100 to L=300. The results from this simulation can be seen from figure (5.22) page (92). Selecting a filter-length of $L > 200$ will only give small improvement

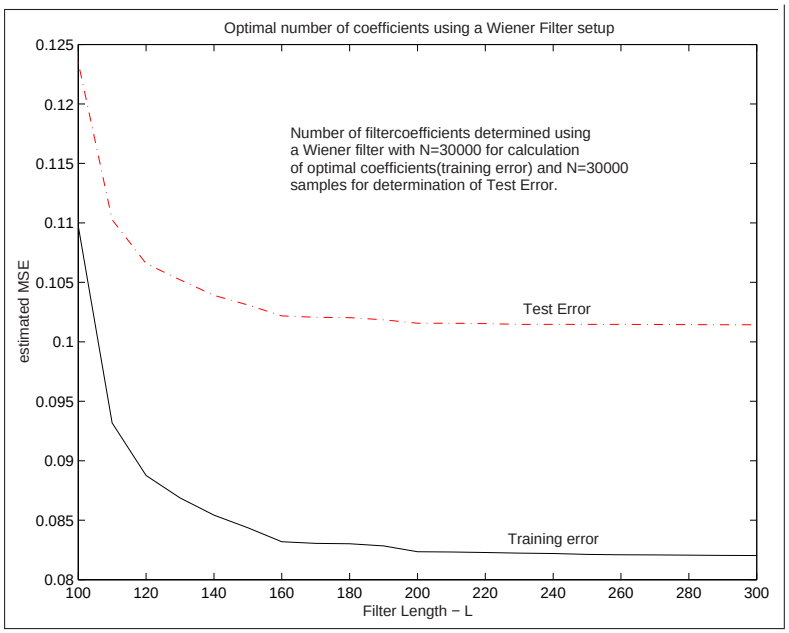


Figure 5.22: Determination of the optimal filter-length using the Wiener filter.

in the MSE both for the train- and test error so L was selected to 200. To check the result also the RLP algorithm (variable-p) was used in *Test 1*. Using the RLP method produced close to the same error as the Wiener filter solution. The RLP-algorithm was run on a sample-size of nsamp=5000. The obtained filter-coefficients was used to filter the 9.5 seconds of the input signal to produce an error signal. The one-norm and two-norm of the error-samples was compared for the optimal Wiener solution and the RLP-solution. This comparison is given in table (5.4) page (92) where one can see that the two measures are almost identical. In the following the

Measures	$\|\|e\|\|_1$	$\|\|e\|\|_2$
RLP (Variable p-norm)	340.06	0.9956
Wiener Solution	338.15	0.9922

Table 5.4: Comparison of 1, and 2-norm of the error in *Test 1* using RLP and Wiener-filter.

filter coefficients obtained from the Wiener filter solution are used as the optimal coefficients[5]. The obtained impulse response (filter coefficients) which were found can be seen from figure (5.23) page (93)(left figure) and the corresponding frequency response can be seen from the right figure. The frequency response shows that the

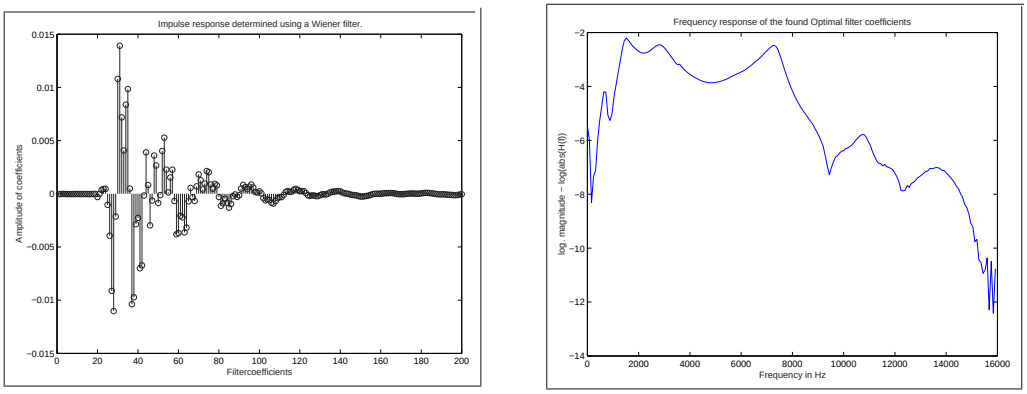


Figure 5.23: **Left:**Impulse response of the optimal filter coefficients. **Right:** Frequency response of the filter(amplitude)

coupling is largest in the frequency range $2000 - 7000Hz$ which is covering speech and music signals. Low frequencies are attenuated more as well as signals with higher frequency.

Since we know the "near optimal" filter coefficients it is possible to estimate the SNR in the different scenarios as a function of time. Even though the signal to noise ratio is a 2-norm measure it is used here since the signals applied have limited power. The signal to noise ratio will here be defined as

$$\widehat{SNR} = \frac{\hat{P}_v}{\hat{P}_d} = \frac{\sum_{k=1}^{N} |\hat{v}_k|^2}{\sum_{k=1}^{N} |\hat{d}_k|^2}. \tag{5.27}$$

where $\hat{d}_k = \boldsymbol{w}_{opt}^T \boldsymbol{x}_k$ and $\hat{v}_k$ is calculated as $\hat{v}_k = y_k - \hat{d}_k$ where y_k is the measurements from the microphone. Due to this definition a high SNR will normally give problems for the adaptive algorithms.

Test 2 The different parameters for the algorithm is given in the parenthesis below.

- RLP($\lambda = 0.999, \lambda_p = 0.95, \omega = 0.1, \delta = 1e-6$) and the RQEGN($\eta = 1, N = 100, \lambda = 0.999, \lambda_r = 0.999, \delta = -1e6$) algorithm
- NLMP($p = 1.2, \eta = 0.008$), SIG($\alpha = 2, \eta = 0.2, \sigma = 1$) and GNLMS($N = 1000, \eta = 0.1$)

The parameters were found trying different values and performing test-runs to determine parameters for the stochastic methods such that they had more or less the

[5]Since 300000 samples was available, I splitted the signal into 10 parts and determined the optimal filter coefficients for each window. The mean of the 2-norm of the filter-weights and the standard deviation of this mean was found to be $Mean(||\boldsymbol{w}||_2) = 0.0358$ and $std(Mean(||\boldsymbol{w}||_2)) \approx 2e-6$, showing that no big variations in the solution occurred over the data-set

same convergence speed. For the RQEGN a rather large δ value was needed, to ensure negative definiteness of the Hessian matrix. As mentioned in the investigation of the RQEGN algorithm a reasonable choice for δ is around $\frac{1}{10\sigma^2}$ to ensure negative definiteness when $L > N$. N is the number of samples used in the Parzen estimate of $p_e(e)$ and σ is the kernel width estimated from the error-data. The δ value for the RLP algorithm was chosen as mentioned in [15] to be $\delta < 0.01\sigma_x^2$, where σ_x^2 is the spread on the input signal.

The modelling error using the near optimal filter coefficients as comparison can be seen for the RLP and RQEGN algorithm in figure (5.24) page (94) and for the stochastic gradients in figure (5.25) page (95)

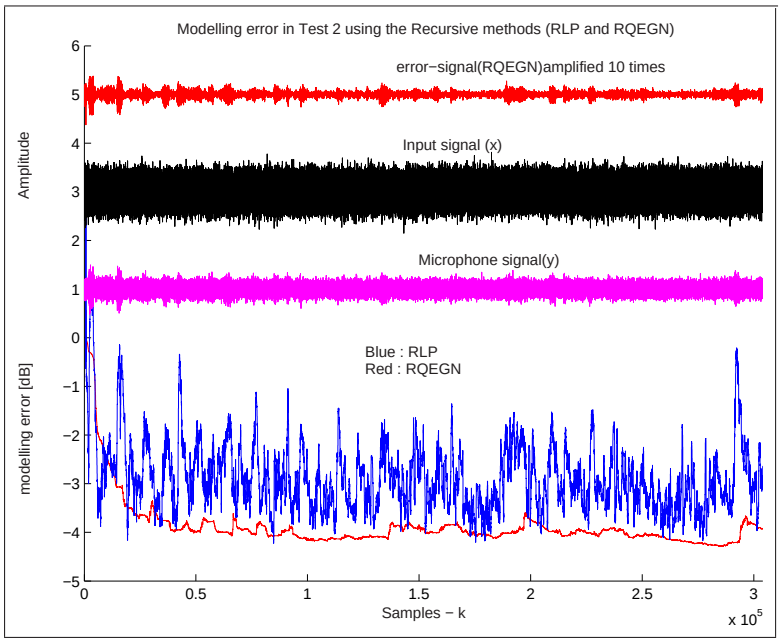


Figure 5.24: Modelling error in Test 2 for the RLP and RQEGN algorithm.

The modelling error shows that the performance of the stochastic methods is very equal to its recursive counterparts (NLMP, RLP) and (SIG/RQEGN). The SIG algorithm is really performing well in this setup which can be seen from the rather fast convergence rate and quite low system mismatch. It should be mentioned here that since we are using a quite large value for σ, compared to the spread of the data, the stochastic gradient at time instant k can be approximated by

$$\approx e_{k,k-1}\boldsymbol{x}_{k,k-1} \tag{5.28}$$

which is much like the LMS algorithm except that this algorithm works on differences [7]. The GNLMS algorithm was performing a little better than the NLMP algorithm. The variable p-norm RLP and the fixed norm NLMP did not perform bad, but it seems as these algorithms are very sensitive to the high SNR. The SNR calculated as an moving average is plotted for this scenario of the first 20000 samples, and is shown together with the modelling error of the GNLMS, NLMP and RLP-algorithm

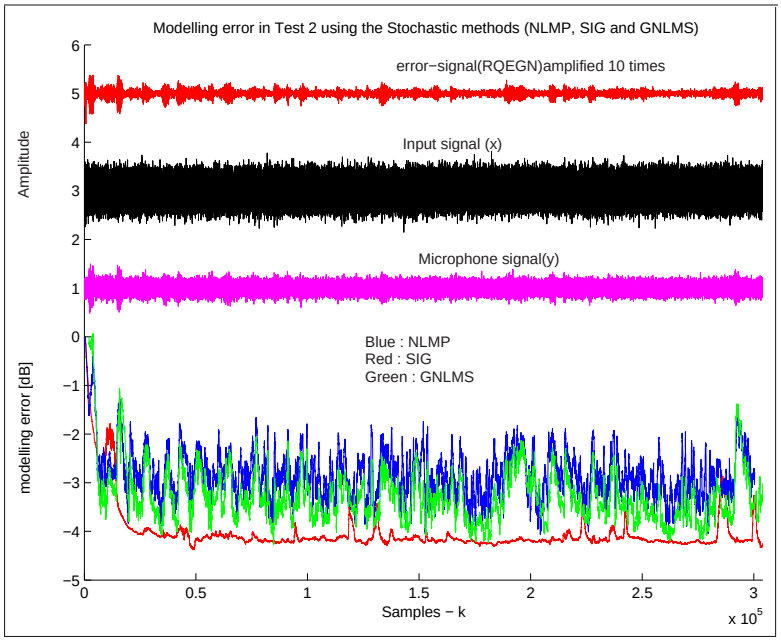


Figure 5.25: Modelling error in test 2 for the stochastic methods.

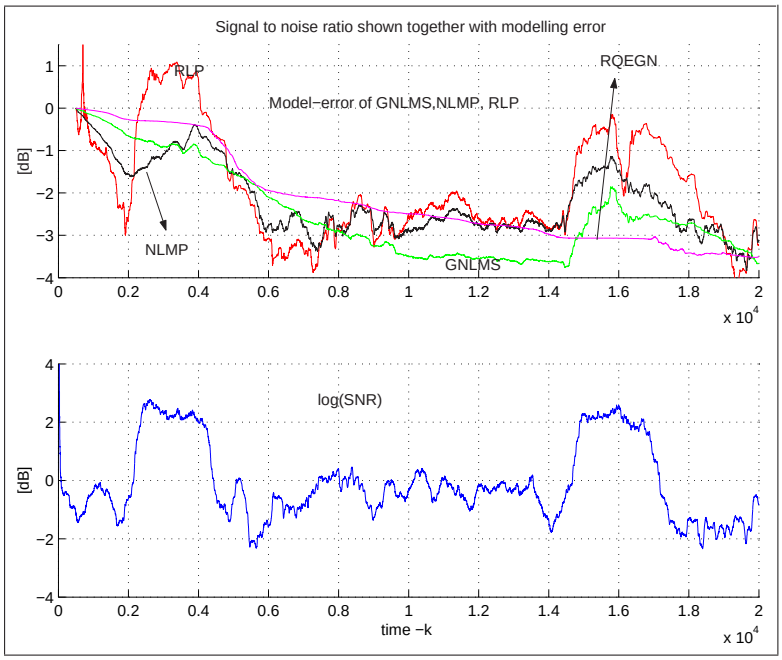


Figure 5.26: SNR plotted for test 2 scenario for the first 20000 samples(lower figure) upper figure shows the model-error of the GNLMS, NLMP and RLP algorithm.

(see figure (5.26) page (95)). It is clear that an increase in the SNR makes the GNLMS,NLMP and RLP algorithm lose tracking of the filter-coefficients. The RLP algorithm attains the highest system mismatch, which is due to its fast tracking. Increasing the number N (samples in transformation) in the GNLMS algorithm did

not really improve the system mismatch (N=2000 was tried). The RQEGN is also shown on the plot and the modelling error seem to become constant when the SNR is increasing.

Since we aim at identifying the office environment noise and speech ($\hat{\boldsymbol{v}}$) the error-signal is plotted against the $\hat{\boldsymbol{v}}$ signal. This plot is denoted as an EV-plot. If the model finds the optimal filter coefficients the plot should show a line with unit slope. Since we have only approximated the true coefficients with the Wiener solution, some inherent noise is present due to a non-perfect match. The different plots can be found in figure (5.27) page (96) for the stochastic methods, and in figure (5.28)

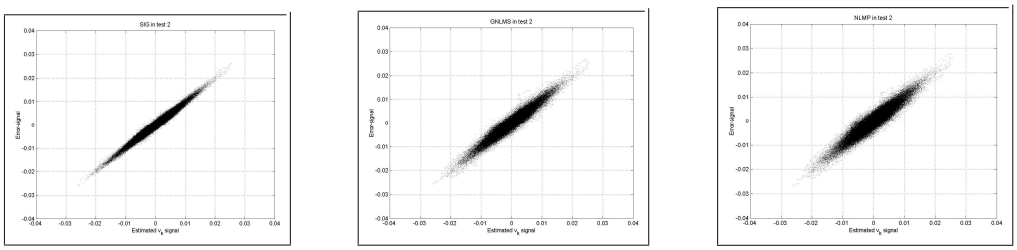


Figure 5.27: Error signal versus expected applied signal $\hat{\boldsymbol{v}}$ for the different stochastic algorithms. **Left:** SIG-algorithm **Middle:** GNLMS algorithm **Right:** NLMP algorithm

page (96) for the recursive methods. All the plots were generated from time index $k = 50000 - 304000$. As we have already seen from the modelling error, the information theoretic methods is performing the best. The spread around the optimal line is less for the information theoretic methods, than any of the other methods. This indicates

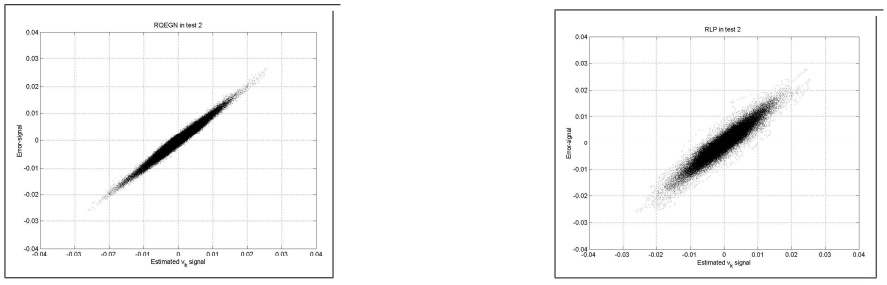


Figure 5.28: Error signal versus expected applied signal $\hat{\boldsymbol{v}}$ for the two recursive algorithms.**Left:** RQEGN algorithm **Right:** RLP algorithm

a good fit of the model-parameters to the coefficients determining the unknown path. Due to the unit slope of all the EV-plots the output distribution is more or less equal for the different methods.

Test 3

In *Test 3* and *Test 4* a Gaussian distributed signal with variance $1e-6$ and zero mean was added to the input signal ($\boldsymbol{x}$ - Dantale signal). It was shown in [2] that this corresponds to Tikhonov regularization when the cost function can be written as

$$E = \frac{1}{2}\int\int ||f(\boldsymbol{w},\boldsymbol{x}) - y||^2 p(\boldsymbol{x},y)d\boldsymbol{x}dy \tag{5.29}$$

where the added signals noise variance will control how much regularization is performed. Inserting an empirical density estimate in (5.29) actually gives the cost-function of the NLMS/RLS algorithm. As seen the variance should have been bigger in order to really act as regularization on the different algorithms.

The parameters, given below, for the algorithms was used in both *Test3* and *Test4*.

- RLP($\lambda = 0.999, \lambda_p = 0.95, \omega = 0.1, \delta = 1e-8$) and the RQEGN($\eta = 1, N = 100, \lambda = 0.999, \lambda_r = 0.999, \delta = -1e6$) algorithm
- NLMP($p = 1, \eta = 0.002$), SIG($\alpha = 2, \eta = 0.2, \sigma = 1$) and GNLMS($N = 1000, \eta = 0.1$)

The modelling error for this setup can for the recursive methods be seen in figure (5.29) page (97) and from figure (5.30) page (98) showing the results of the stochastic methods. The high SNR combined with the low input power, which makes the eigenvalues of the weighted autocorrelation matrix (as well as the eigenvalues of the Hessian) small, will make the recursive methods lose its tracking of the coefficients.

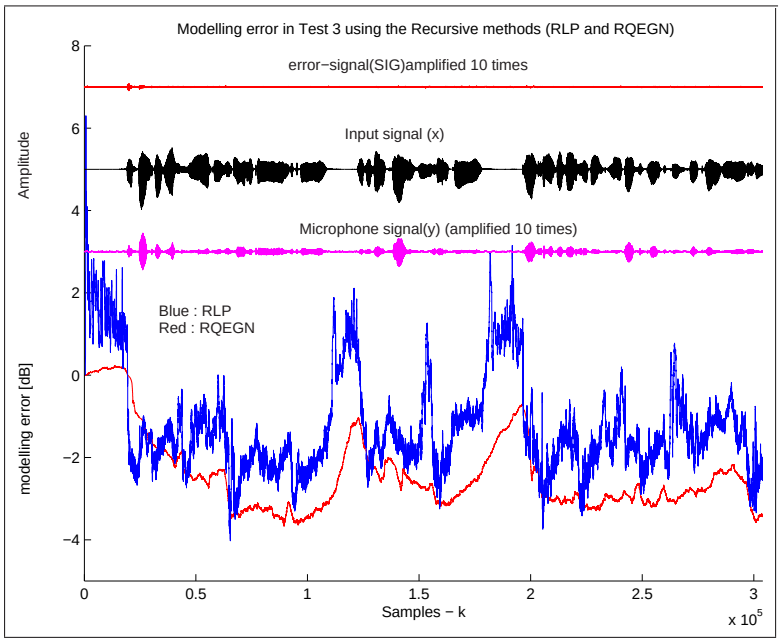


Figure 5.29: Modelling error in *Test 3* for the RLP and RQEGN algorithm.

The most successful algorithm in keeping the tracking of filter coefficients are the SIG-algorithm. The algorithm converges around $k = 70000$ with a "system mismatch" of $\approx -6dB$ which is very nice. The GNLMS method is losing tracking of the filter-coefficients drastically a couple of times. The reason has not been fully understood, but it normally happens when the environment have remained unchanged for a period and then suddenly changes. Figure (5.31) page (98) shows the SNR ratio in the time interval $k = 1e5 - 1.5e5$ as well as the model error of the RLP, RQEGN and GNLMS method. As mentioned before the effect of the low input signal and high SNR will cause the RLP and RQEGN algorithm to lose its tracking. In this time-interval the GNLMS managed alright. In *test 4* it is shown that the smallest eigenvalue of the

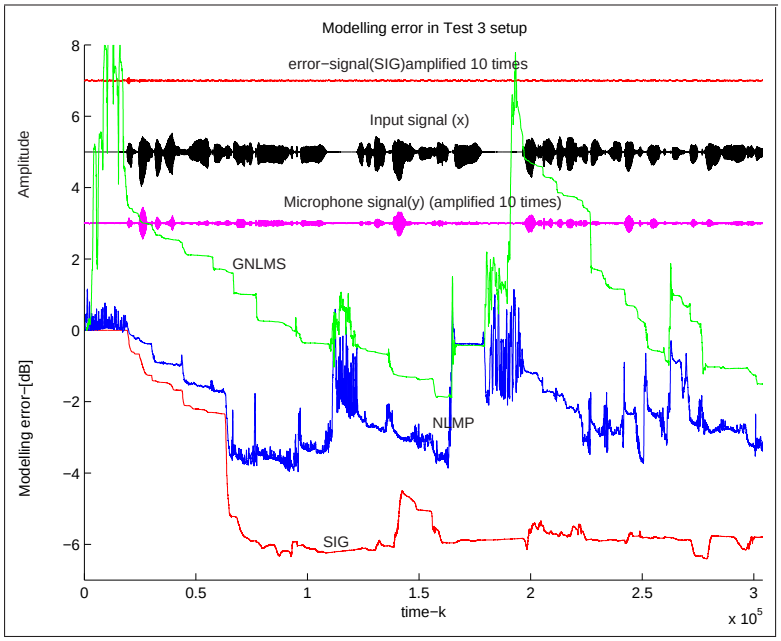


Figure 5.30: Modelling error in *test 3* for the stochastic methods.

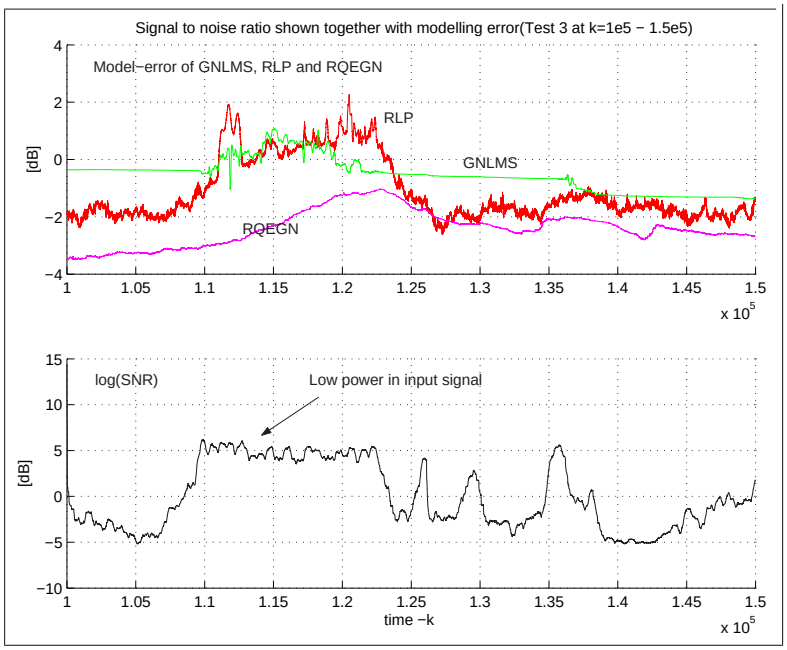


Figure 5.31: SNR plotted for test 3 scenario in the interval $k = 1e5 - 1.5e5$(lower figure). The upper figure shows the model-error of the GNLMS, NLMP and RLP algorithm.

Hessian is very close to zero, when the input signal gets small (as with the RLP - algorithm). It is shown that increasing the size of the eigenvalues by regularization improves the modelling error, in those situations. The EV-plots of the different

algorithms can be seen in figure (5.32) page (99) for the stochastic algorithms and

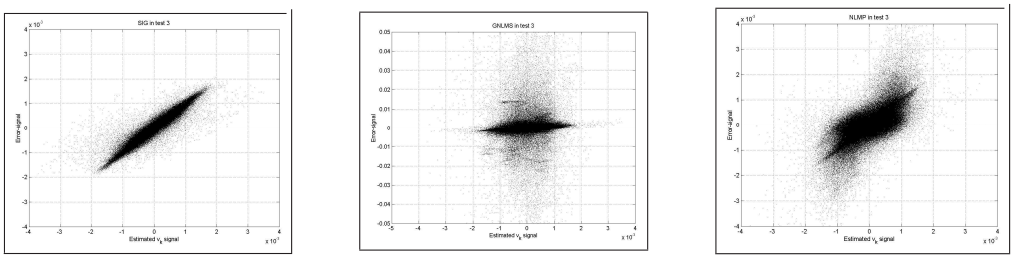


Figure 5.32: Test 3. Error signal versus expected applied signal $\hat{\boldsymbol{v}}$ for the different stochastic algorithms. **Left:** SIG-algorithm **Middle:** GNLMS algorithm **Right:** NLMP algorithm

in (5.33) page (99) for the recursive methods. As expected the SIG algorithm show to have the least spread, which means that it determines the microphone noise the best. The GNLMS algorithm is not really good, since the error-signal is much larger in amplitude than the signal ($\hat{\boldsymbol{v}}$) which means that it is dominated by an attenuated linear combination of the input signal. These figures is presented in order to give

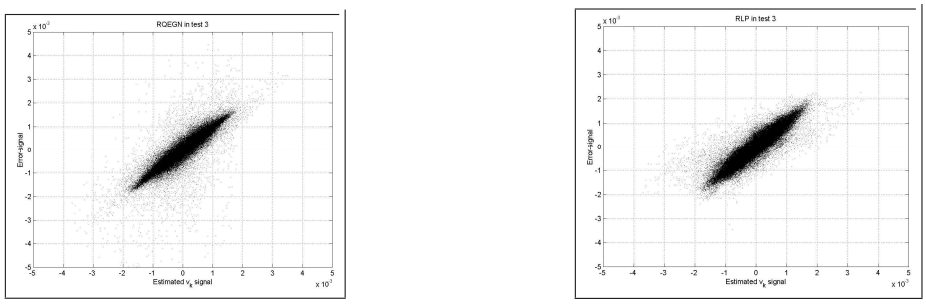


Figure 5.33: Test 3. Error signal versus expected applied signal $\hat{\boldsymbol{v}}$ for the two recursive algorithms.**Left:** RQEGN algorithm **Right:** RLP algorithm

a intuition of how the algorithms models the added signal(v). Since the modelling error has been shown, we more or less know the outcome, but as seen in the RLP-case one would apparently expect a bad results in the EV-plot but due to the fast convergence of the RLP algorithm the result is not that bad.

Test 4

The test is very similar to *test 3* except that the test setup is placed in an open office environment, making the situation a bit harder. When looking at the modelling error for the recursive (figure (5.34) page (100)) and for the stochastic methods (figure (5.35) page (100)) it is seen that the performance have been degraded in all cases, since the system mismatch is higher for the different algorithms. In this setup only the SIG algorithm seem to give nice performance with a system mismatch of around $-4dB$. The convergence rate of the SIG algorithm is the same as observed in *test 3* in that it converges around $k = 70000$. In this setup the smallest absolute eigenvalue as a function of time in the interval $k = 1e5 - 1.5e5$ was determined. This plot can be seen from figure (5.36) page (101) where the lower figure shows the estimated SNR (averaging 50 samples). The 3'th figure from above shows the smallest eigenvalue

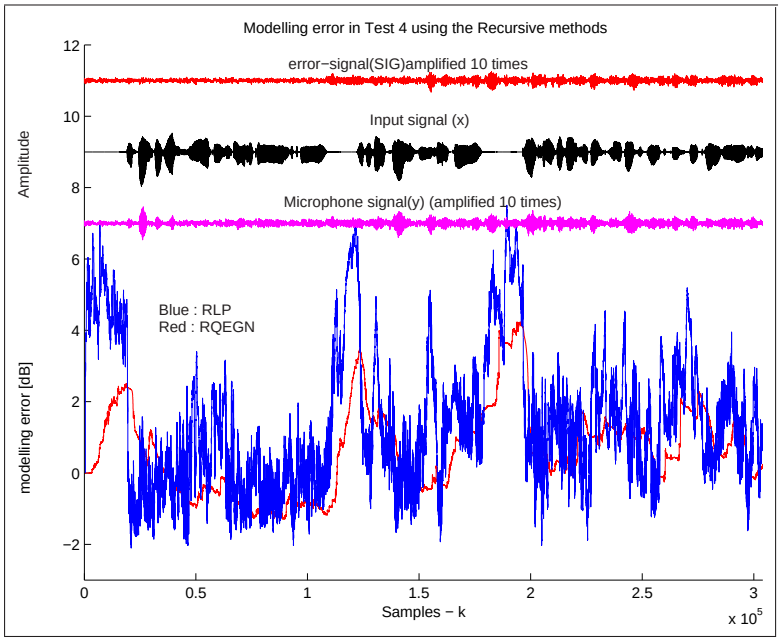


Figure 5.34: Modelling error in Test 4 for the RLP and RQEGN algorithm.

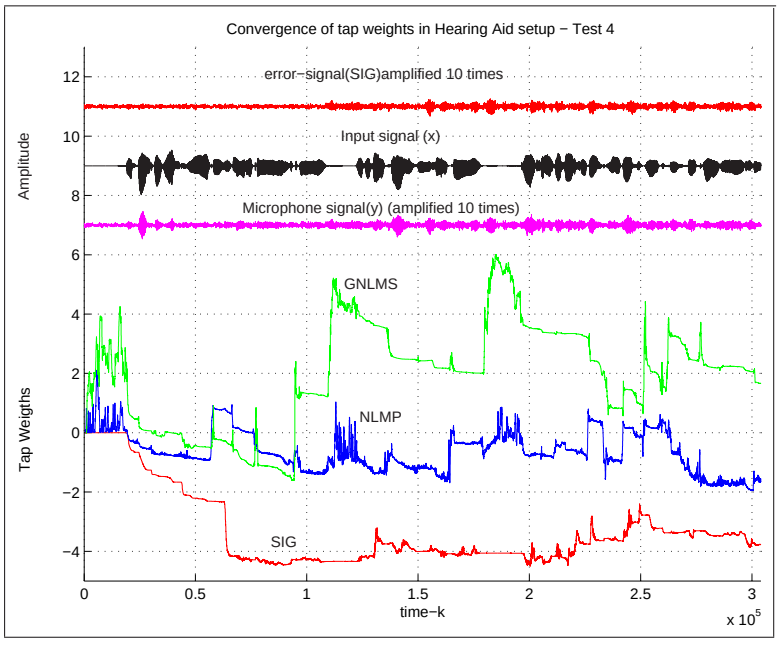


Figure 5.35: Modelling error in test 4 for the stochastic methods.

(in absolute sense) in the interval $k = 1e5 - 1.5e5$. It is seen that the eigenvalue gets very close to zero (near singular), which makes the algorithm lose its tracking. To improve the "ill-conditioned" problem a regularization (Levenberg-Marquardt) term was added to the Hessian [3]. The new Hessian is determined as

$$\hat{H}_k(\boldsymbol{w}) = H_k(\boldsymbol{w}) - a\boldsymbol{I}, \tag{5.30}$$

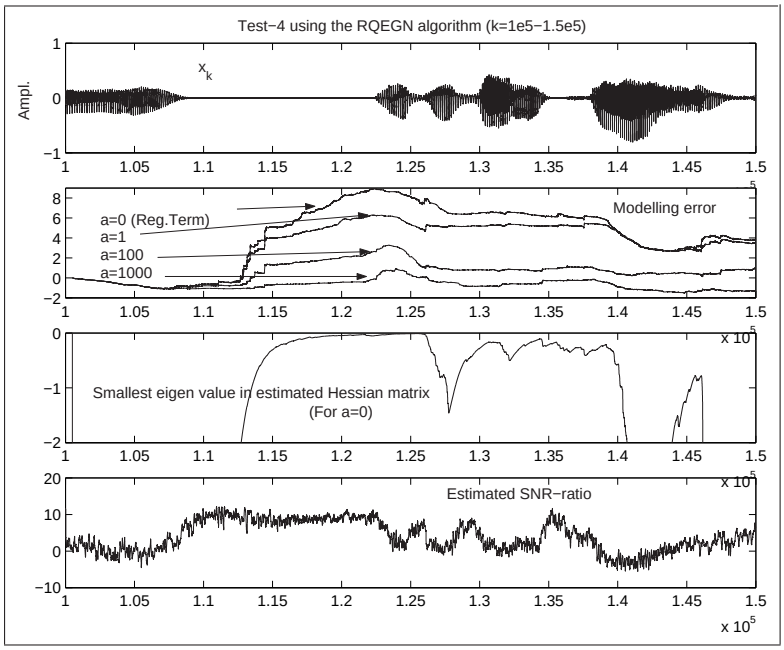


Figure 5.36: Model-error using different constants for regularization in time interval $k = 1e5 - 1.5e5$.

where $\boldsymbol{I}$ is the identity matrix with same dimension as the Hessian and if a is selected large the RQEGN algorithm will tend towards a steepest descent approach. Different values of a was tried which can be seen from the modelling error of the RQEGN method in the second plot. Selecting a value of $a = 1000$ shows to improve the problem with the higher SNR since the algorithm will not lose track of the coefficients as much as with $a = 0$. The EV-plots of the stochastic gradient approaches can be seen from figure (5.37) page (101) and for the recursive methods in figure (5.38) page (102). The good performance of the SIG algorithm is also seen in the EV-plot, showing that the office environment noise is catched quite well by this algorithm. Actually the second best algorithm seem to be the RLP-algorithm, which do not

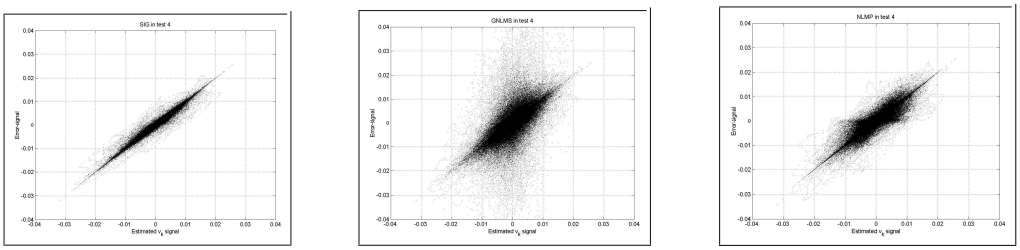
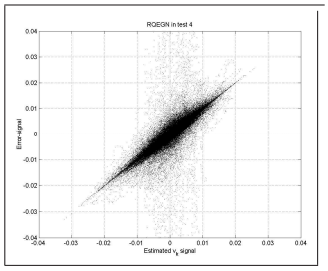


Figure 5.37: Test 4. Error signal versus expected applied signal $\hat{\boldsymbol{v}}$ for the different stochastic algorithms. **Left:** SIG-algorithm **Middle:** GNLMS algorithm **Right:** NLMP algorithm

show so big difference between the error signal and the office environment noise and speech($\hat{\boldsymbol{v}}$). Even though the error-estimate of the GNLMS algorithm is quite noisy, the result achieved looks better than in the last experiment.



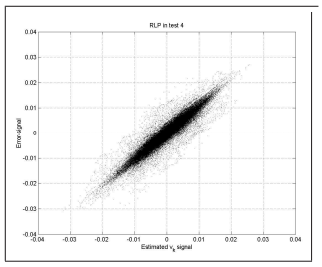


Figure 5.38: Test 4. Error signal versus expected applied signal $\hat{\boldsymbol{v}}$ for the two recursive algorithms.**Left:** RQEGN algorithm($a = 0$) **Right:** RLP algorithm

5.3.1 Discussion

The results obtained in *test 2*, *test 3* and *test 4* is summarized in table (5.5) page (102), using both the one norm and two norm as a measure. The ANLL have not been used for comparison since the different error distributions of the different algorithms in the different tests show very little deviation from each other, making the ANLL estimate show very similar performance. The one-norm and two-norm though seem to be adequate in this setup to describe the performance of the system. The results is merely a summation of what we have already observed, namely that the Stochastic Information Gradient (SIG) algorithm is the one performing the best in all three test cases. Considering the problem (L=200), this is a quite hard problem for the Recursive methods computational wise. For transversal adaptive filters with a rather large filter-length it seems as if stochastic methods are preferable partly due to its low

Method	*Test 2*		*Test 3*		*Test 4*	
	$\|\|e\|\|_2$	$\|\|e\|\|_1$	$\|\|e\|\|_2$	$\|\|e\|\|_1$	$\|\|e\|\|_2$	$\|\|e\|\|_1$
NLMP	2.82	1117	0.34	123	2.35	897
SIG	2.74	1085	0.30	132	2.34	960
GNLMS	2.80	1112	4.51	962	3.21	1203
RLP	2.85	1129	0.31	139	2.56	1049
RQEGN	2.76	1090	0.38	145	4.09	1251
Optimal	2.74	1092	0.31	134	2.37	971

Table 5.5: Norm-1 and Norm-2 of error of the different algorithms in *Test2*, *Test3* and *Test4*.

computational complexity but also, as seen, with respect to the observed problems with ill-conditioning of either the Hessian matrix or the cross-correlation matrix. I must say that I was impressed how the SIG algorithm performed in these different setups considering the problems in the AR- tests. Whether or not this algorithm will work in a closed loop system would need further investigation.

5.4 Discussion

In the simple simulations, it was shown that more complex distributions will be recovered well by using the information theoretic methods. One could say the these methods work well in unknown environments, since the methods works directly on the distributions. In the case of normal symmetric distributions the information theoretic methods did not really provide any efficiency compared to algorithms designed specifically to handle some norm in the distribution. In addition, the computational burden was seen to be bigger with the information theoretic methods (except for the SIG algorithm).

In the hearing aid example the stochastic information gradient really showed to be good when comparing it to other known algorithms. The recursive methods did not do very well in this setup due to bad conditioning of the Hessian and weighted autocorrelation matrix.

Chapter 6

Further work

During this project, there have been many interesting issues which have not been investigated fully. Some, which would be interesting are

- A theoretical investigation of the SIG algorithm similar to what exists on the LMS algorithm. How will the algorithm work in highly correlated environments? How is the tracking of this algorithm? etc.
- The RQEGN has the drawback that the Parzen estimate is using the last N samples in the PDF-estimate. Not all samples have equally influence on the PDF-estimate so it must be possible to determine the most important samples in this estimate. In that way the computational complexity will be reduced.
- Investigate different system setups using the Renyi generalization error with regularization as introduced in section (4.5) page (49).

Chapter 7

Conclusion

In chapter two the additive noise model was introduced and some scenarios were sketched in which this model can be used. A class of distributions were introduced, known as alpha stable distributions. These distributions were used when evaluating the different adaptive algorithms. A discussion on how to generate arbitrary distributed data was given. In chapter three a discussion on norm based robust adaptive algorithms were given. A couple of stochastic methods as well as some recursive methods were discussed. A method to determine the p-parameter for correct norm minimization using the RLP algorithm was introduced. The information theoretical methods were introduced using the KL-divergence measure between the true joint density function and the model joint density function. The minimization of the KL-divergence lead to the Shannon generalization error. It was argued that this measure could be substituted with a similar information measure, the Renyi generalization error, which in the limit as α goes to 1 equals the Shannon generalization error. The Renyi generalization error was especially simple for $\alpha = 2$. Similarities between Renyi and Shannon generalization error were discussed, and it was shown that the two measures equals when the error is uniform distributed. The non-parametric Parzen density estimate was introduced as to approximate the error distribution for the information theoretical methods. Two methods to determine an optimal kernel width were investigated, and it was found that the "Rule of Thumb" method was efficient in unimodal environments. Further investigation of the generalization error using a Parzen density estimate for the joint density $p(\boldsymbol{x}, y)$ showed to correspond to regularization. From the Renyi generalization error three algorithms were derived. One method based on batches of samples (Batch algorithm), one recursive method (RQEGN) and a stochastic method (SIG, [7]).

In the simple simulations, it was shown that more complex distributions are recovered well by using an information theoretical algorithm (RQEGN). It was basically found that these methods work well in unknown environments since the unknown error distribution is modelled. In "simple" environments (normal symmetric distributions) the information theoretical methods did not really provide any efficiency compared to simple norm algorithms. The proposed methods (RQEGN and Batch) have the drawback that they are computationally expensive $\mathcal{O}(L^3)$, except for the stochastic

information gradient which has a workload of $\mathcal{O}(L)$. It was shown that this algorithm do not converge to the true coefficient in an AR - setup. In the hearing aid example the stochastic information gradient performed really good, when compared to other robust algorithms. The recursive methods (RLP and RQEGN) did not do well in this setup due to bad conditioning of the Hessian and weighted autocorrelation matrix.

Appendix A

Derivations

A.1 Taylor expansion of a function with multiple variables

The aim of this appendix is to show how a function $f(\boldsymbol{x}, y)$, which is assumed continuous and infinitely differentiable can be Taylor expanded.
From [35] a Taylor expansion of a function $f : \mathbb{R}^n \mapsto \mathbb{R}$ is given as

$$f(\boldsymbol{z}) = \sum_{j=0}^{\infty} \frac{1}{j!} \left((\boldsymbol{z} - \boldsymbol{z}_n)^T \cdot \nabla_{z'} \right)^j f(\boldsymbol{z}') \Big|_{\boldsymbol{z}'=\boldsymbol{z}_n}. \tag{A.1}$$

So the first couple of derivatives of $f(\boldsymbol{z})$ would be

$$f(\boldsymbol{z}) = f(\boldsymbol{z}_n) + (\boldsymbol{z} - \boldsymbol{z}_n)^T \nabla_{z'} f(\boldsymbol{z}') \Big|_{\boldsymbol{z}'=\boldsymbol{z}_n} + \frac{1}{2!} (\boldsymbol{z} - \boldsymbol{z}_n)^T \nabla_{\boldsymbol{z}'\boldsymbol{z}'} f(\boldsymbol{z}') \Big|_{\boldsymbol{z}'=\boldsymbol{z}_n} (\boldsymbol{z} - \boldsymbol{z}_n) + R_3 \tag{A.2}$$

where R_3 accounts for the higher order terms. Next we assume that $\boldsymbol{z}$ consist of both $\boldsymbol{x}$ and y, so

$$\boldsymbol{z} = \begin{bmatrix} \boldsymbol{x} \\ y \end{bmatrix} \tag{A.3}$$

which is of dimension $p \times 1$ from which the terms in (A.2) can be substituted with $\boldsymbol{x}$ and y:

$$(\boldsymbol{z} - \boldsymbol{z}_n)^T \nabla_{\boldsymbol{z}'} f(\boldsymbol{z}') \Big|_{\boldsymbol{z}'=\boldsymbol{z}_n} = \sum_{i=1}^{p} (z_i - z_{ni}) \frac{\partial f}{\partial z_i'} \Big|_{z'=zn} \tag{A.4}$$

$$= \sum_{i=1}^{p-1} (x_i - x_{ni}) \frac{\partial f}{\partial x_i'} \Big|_{x'=x_n} + (y - y_n) \frac{\partial f}{\partial y'} \Big|_{y'=y_n} \tag{A.5}$$

$$= (\boldsymbol{x} - \boldsymbol{x}_n)^T \nabla_{\boldsymbol{x}'} f \Big|_{\boldsymbol{x}'=\boldsymbol{x}_n} + (y - y_n) \frac{\partial f}{\partial y'} \Big|_{y'=y_n} \tag{A.6}$$

$$(\boldsymbol{z}-\boldsymbol{z}_n)^T\nabla_{\boldsymbol{z}'\boldsymbol{z}'}f(\boldsymbol{z}')\Big|_{\boldsymbol{z}'=\boldsymbol{z}_n}(\boldsymbol{z}-\boldsymbol{z}_n)=\sum_{i=1}^{p}\sum_{j=1}^{p}(z_i-z_{ni})(z_j-z_{nj})\frac{\partial^2 f}{\partial z_i'\partial z_j'}\Big|_{z'=z_n} \tag{A.7}$$

$$\begin{aligned}\ldots=&\sum_{i=1}^{p-1}\sum_{j=1}^{p-1}(x_i-x_{ni})(x_j-x_{nj})\frac{\partial^2 f}{\partial x_i'\partial x_j'}\Big|_{x'=x_n}+\\&2(y-y_n)\sum_{j=1}^{p-1}(x_j-x_{nj})\frac{\partial^2 f}{\partial y'\partial x_j'}\Big|_{y'=y_n,x'=x_n}+(y-y_n)^2\frac{\partial^2 f}{\partial^2 y'}\Big|_{y'=y_n}\end{aligned} \tag{A.8}$$

which is then equal to :

$$\begin{aligned}\ldots=&(\boldsymbol{x}-\boldsymbol{x}_n)^T\nabla_{\boldsymbol{x}'\boldsymbol{x}'}f|_{\boldsymbol{x}'=\boldsymbol{x}_n}(\boldsymbol{x}-\boldsymbol{x}_n)+\\&2(y-y_n)(\boldsymbol{x}-\boldsymbol{x}_n)^T\nabla_{y'\boldsymbol{x}'}f|_{y'=y_n,\boldsymbol{x}'=\boldsymbol{x}_n}+(y-y_n)^2\frac{\partial^2 f}{\partial^2 y'}\Big|_{y'=y_n}\end{aligned} \tag{A.9}$$

From the above calculations it have now been shown that $f(\boldsymbol{x},y)$ can be written as

$$\begin{aligned}&f(\boldsymbol{x},y)=f(x_n,y_n)+(\boldsymbol{x}-\boldsymbol{x}_n)^T\nabla_{\boldsymbol{x}'}f\Big|_{\boldsymbol{x}'=\boldsymbol{x}_n}+(y-y_n)\frac{\partial f}{\partial y'}\Big|_{y'=y_n}+\frac{1}{2!}(\boldsymbol{x}-\boldsymbol{x}_n)^T\cdot\\&\nabla_{\boldsymbol{x}'\boldsymbol{x}'}f\Big|_{\boldsymbol{x}'=\boldsymbol{x}_n}(\boldsymbol{x}-\boldsymbol{x}_n)+(y-y_n)(\boldsymbol{x}-\boldsymbol{x}_n)^T\nabla_{y'\boldsymbol{x}'}f\Big|_{y'=y_n,\boldsymbol{x}'=\boldsymbol{x}_n}+\frac{1}{2!}(y-y_n)^2\frac{\partial^2 f}{\partial^2 y'}\Big|_{y'=y_n}+R_3\end{aligned} \tag{A.10}$$

using Taylor expansion.

A.2 Properties of Renyi and Shannon entropy

The term "entropy" is used to express differential entropy (continuous variables) and entropy determined from a finite sized data set.

A.2.1 Properties of Shannon's entropy

Translation property

$$H_S(X+c) = H_S(X+c) \quad \text{where } c \in \mathcal{R}. \tag{A.11}$$

Proof.

$$H_S(X+c) = -\int_{-\infty}^{\infty} p(\xi+c)\log p(\xi+c)d\xi = H_S(x) \tag{A.12}$$

Scaling property
In [16] the following property of Shannon's differential entropy is given $H(aX) = H(X)+\log|a|$, where $a \in \mathcal{R}$. The scaling property can be verified using the following identity $p(x) = 1/|a|p(x/a)$ (since the area of an probability density function is unity) in the expression for Shannon entropy.

Additivity property
A special property of Shannon entropy is the additivity property. Given two random variables X and Y, the additivity property states that $H(X,Y) = H(X|Y)+H(X)$. In the case that X and Y are independent the following result applies $H(X,Y) = H(X)+H(Y)$.

$$\begin{aligned} H(X,Y) &= -\int \log(p(x,y))p(x,y)dxdy \\ &= -\int \log(p(x|y)p(y))p(x,y)dxdy \\ &= -\int \log(p(x|y))p(x,y)dxdy - \int \log(p(y))p(x,y)dxdy \\ &= H(X|Y) - \int \log(p(y)) \int p(y|x)p(x)dxdy \\ &= H(X|Y) + H(Y) \end{aligned} \tag{A.13}$$

Which tells us that information is additive.

A.2.2 Renyi's differential entropy and its properties

Renyi's differential entropy defined here for the process X

$$H_R(X) = \frac{1}{1-\alpha}\log\int_{-\infty}^{\infty} p(x)^{\alpha}dx. \tag{A.14}$$

Translation property
In the following X is a random variable and $c \in \mathcal{R}$.

$$H_R(X+c) = \frac{1}{1-\alpha}\log\int p(x+c)^\alpha dx = H_R(X) \tag{A.15}$$

Scaling property
To show what happens with Renyi's entropy when X is scaled with a, $p(x) = \frac{1}{|a|}p(\frac{x}{a})$ is used.

$$\begin{aligned}
H_R(X) =& \frac{1}{1-\alpha}\log\int p(x)^\alpha dx \\
=& \frac{1}{1-\alpha}\log\int \frac{1}{|a|^\alpha}p(\frac{x}{a})^\alpha dx \\
=& -\frac{1}{1-\alpha}\log|a| + \log|a| + \frac{1}{1-\alpha}\log\int p(\frac{x}{a})^\alpha dx \\
& \text{inserting that aY=X} \\
=& -\frac{1}{1-\alpha}\log|a| + \log|a| + \frac{1}{1-\alpha}\log\int p(y)^\alpha d(|a|y) \\
=& \log|a| + H_R(Y)
\end{aligned} \tag{A.16}$$

where $a \in \mathcal{R}$. So the same scaling relative change in entropy is obtained using another scaling of the variable.

Additivity property
For standard additivity, this property can be shown for independent event using Renyi entropy :

$$\begin{aligned}
H_R(X,Y) &= \frac{1}{1-\alpha}\log\int\int p(x,y)^\alpha dxdy \\
H_R(X,Y) &= \frac{1}{1-\alpha}\log\int\int p(x|y)^\alpha p(y)^\alpha dxdy
\end{aligned}$$

If X and Y are independent then the following applies $p(x|y) = p(x)$

$$\begin{aligned}
H_R(X,Y) &= \frac{1}{1-\alpha}\log\int p(x)^\alpha dx\int p(y)^\alpha dy \\
H_R(X,Y) &= H_R(X) + H_R(Y)
\end{aligned} \tag{A.17}$$

Appendix B

Simulations

B.1 Investigation of the Recursive Quadratic Entropy Gauss Newton Algorithm

To test/investigate the Recursive Quadratic Entropy Gauss Newton algorithm with respect to the different adjustable parameters several simulation runs have been performed. Different environments (v_k) have been tested to see if the parameters behave equally, independent of the environment. The following environments are investigated

- Gaussian noise (Test A)
- Super-Gaussian noise ($S(1.2)S$)-distributed (Test B).
- Correlated super-Gaussian noise ($S(1.5)S$)-distributed using a moving average filter (Filter length 10) to generate process v_k. (Test C)

The RQEGN algorithm has five parameters, which can be adjusted

- λ controls the forgetting factor in the empirical density estimate of $p(\boldsymbol{x}, y)$. The closer to one, the more accurate density estimate (larger memory).
- λ_r controls the forgetting factor in the estimate of $p_e(e)$.
- N controls the number of samples used in the parzen estimate of $p_e(e)$. Is connected to the parameter λ_r.
- η is the step size which is used for the update equation. In this test (as well as other simulations), the step-size is fixed, but may be made adjustable as well as to improve the convergence of the algorithm.
- δ is controlling the initialization of the Hessian matrix.

Since we know the AR(3)-parameters we will concentrate on the modelling error as a quality measure, and plot the mean modelling error with corresponding error-bars ($\pm$ the standard deviation of the mean), since each simulation is run 10 times using different realizations of the noise.

The parameters investigated is seen in table (B.1) page (114)

parameter	red	blue	black	magenta	green	constant
	1	2	3	4	5	
λ	0.999	0.998	0.99	0.98	0.9	$\lambda_r = 0.999, N = 50$ $\eta = 4, \delta = -1$
	6	7	8	9	10	
λ_r	0.999	0.998	0.99	0.98	0.6	$\lambda = 0.999, N = 50$ $\eta = 4, \delta = -1$
	11	12	13	14	15	
N	10	30	50	70	100	$\lambda_r = 0.999, \lambda = 0.999$ $\eta = 4, \delta = -1$
	16	17	18	19	20	
η	1	4	6	10	20	$\lambda_r = 0.999, \lambda = 0.999$ $N = 50, \delta = -1$
	21	22	23	24	25	
δ	0	-0.1	-1	-5	-10	$\lambda_r = 0.999, \lambda = 0.999$ $N = 50, \eta = 4$

Table B.1: Parameters for the RQEGN algorithm in test A-B-C

Simulation results

The modelling error, in the investigation of λ, of the three environments can be seen from figure (B.1) page (115), (B.2) page (115) and (B.3) page (116). The first thing that should be observed is that the selection of λ seem to have a similar influence on the modelling error in the different environments. Another thing, which is seen from the figures

is that the system mismatch seem to be smaller when the system environment gets more super-Gaussian. In the case of i.i.d. S(1.2)S noise (Test B) and colored S(1.5)S noise (Test C) the system mismatch is at a lower level than in the Gaussian case (Test A). Selecting a λ value of $\lambda = 0.998$ seem to be a good compromise between system mismatch and initial convergence speed in this setup.

The forgetting factor (λ_r) controlling the effective number of samples used in the error-PDF estimate was investigated. This investigation can be seen from figure (B.4) page (116), (B.5) page (117) and (B.6) page (117) for the different environments. Again, similarities is seen between the different environments, where it is clear that this parameter do not seem to control the performance much, except when selected very small. It should be noted here, that the selection of λ_r is very much related to the number of points selected in the parzen estimate (N). It is known [15] that the forgetting factor is a kind of effective length of the window. Using N as selection of λ_r would give

$$\lambda_r = \frac{N-1}{N} \tag{B.1}$$

Selecting a value of $\lambda_r = 0.1$ or even lower corresponding to an effective length of around 1 will make the algorithm diverge in all cases. This issue is discussed in the experiment section on the SIG algorithm.

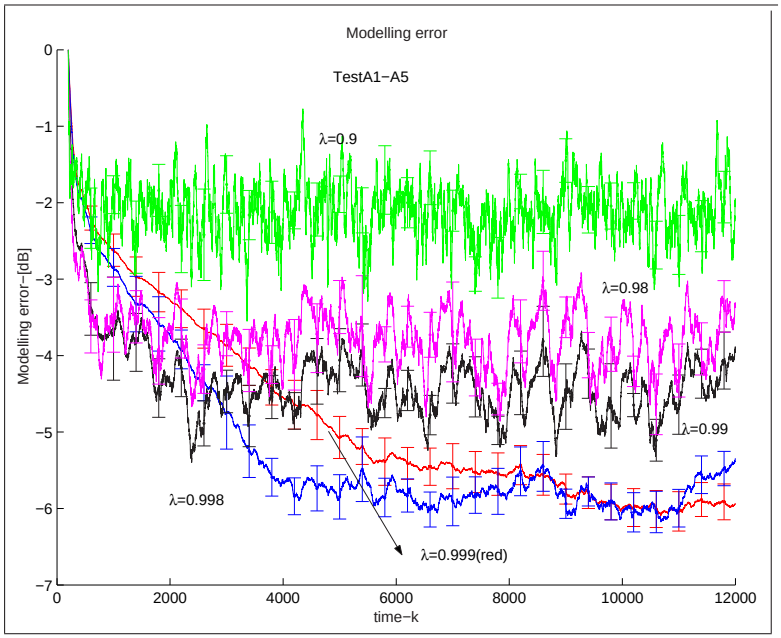


Figure B.1: Test A 1-5 (investigation of λ)

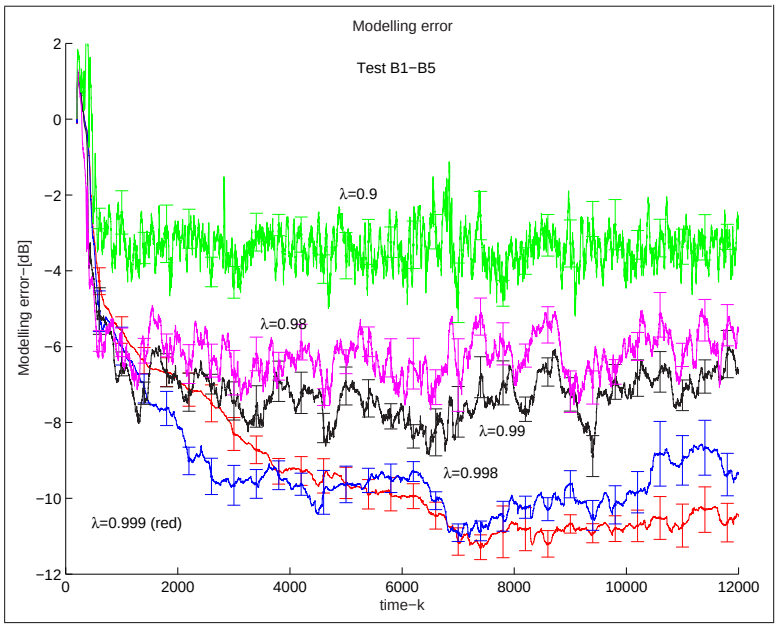


Figure B.2: Test B 1-5 (investigation of λ)

As already mentioned in the text a to low value of λ_r will give the model a larger system mismatch. The same is true when selecting a to small number of samples (N) in the PDF estimate of $p_e(e)$. N controls the effective number of samples which is used, where λ_r is a more soft weighting of samples. The test of different sample sizes N can be seen from figure (B.7) page (118), (B.8) page (118) and (B.9) page

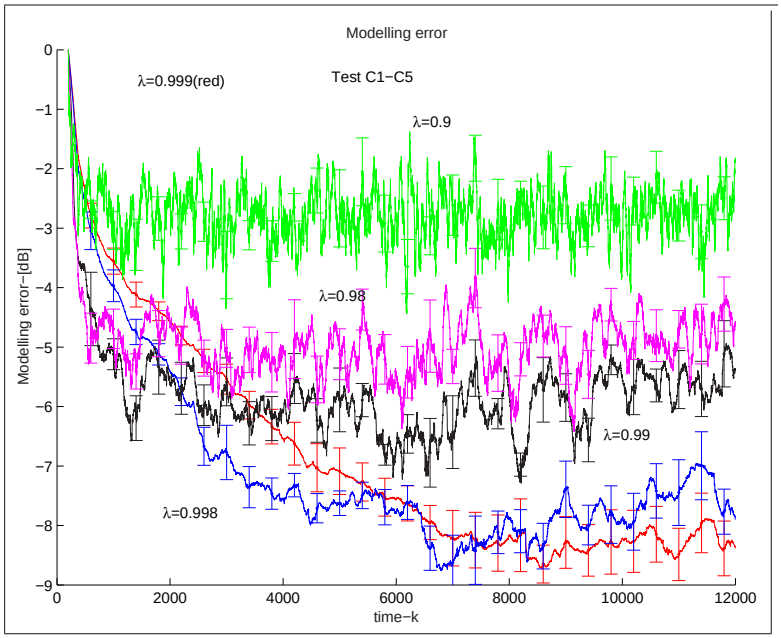


Figure B.3: Test C 1-5 (investigation of λ)

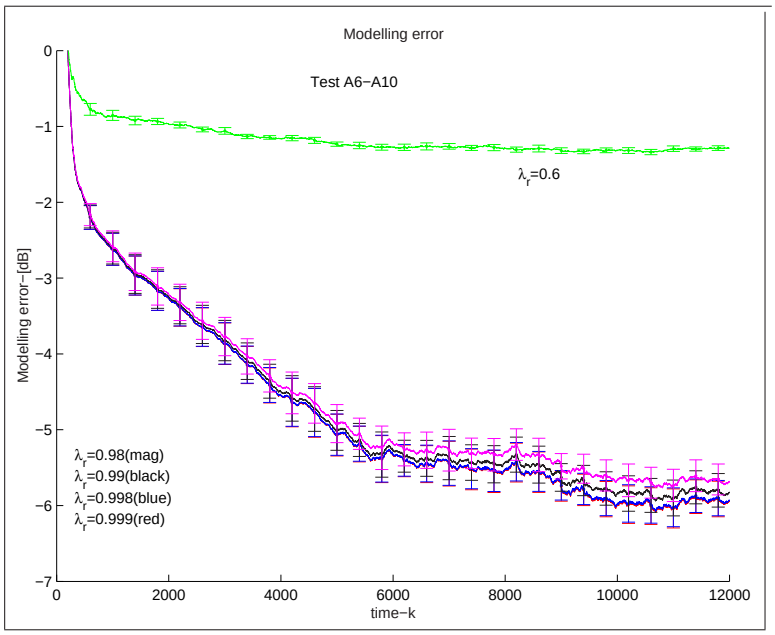


Figure B.4: Test A 6-10, investigation of λ_r

(119), for the different environments, where the system mismatch is biggest in the different environments for N=10. In test B the difference to the other sample sizes is not that big, this comes the fact that the cost-function is more well defined for super-Gaussian signals than for Gaussian signals, hence needs a smaller number of samples to obtain a good system mismatch. Selecting N very big will only give a

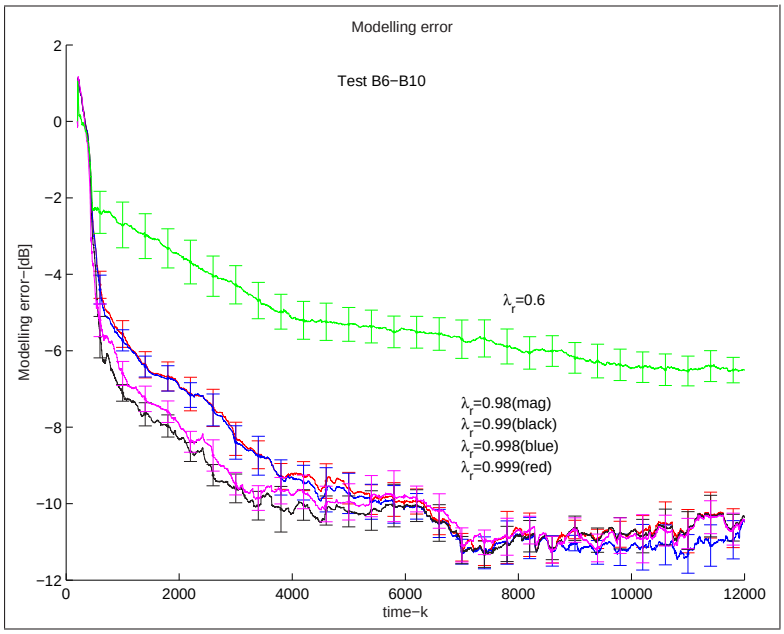


Figure B.5: Test B 6-10, investigation of λ_r

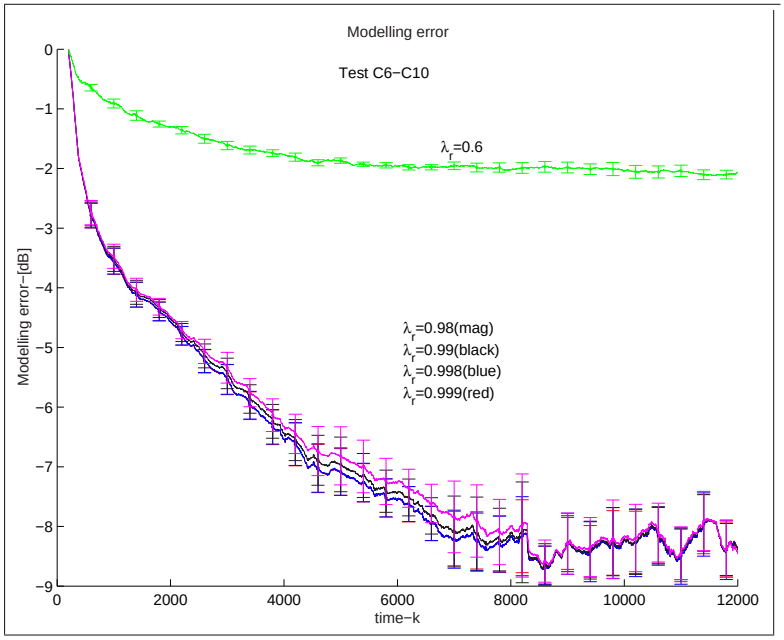


Figure B.6: Test C 6-10, investigation of λ_r

little system mismatch improvement.

In the next investigation the step-size η is investigated. As in other algorithms using a fixed step size, the step size here control the rate of convergence as well as system mismatch (variance of system mismatch). The different tests in the different environments is seen in figure (B.10) page (119), (B.11) page (120) and (B.12) page

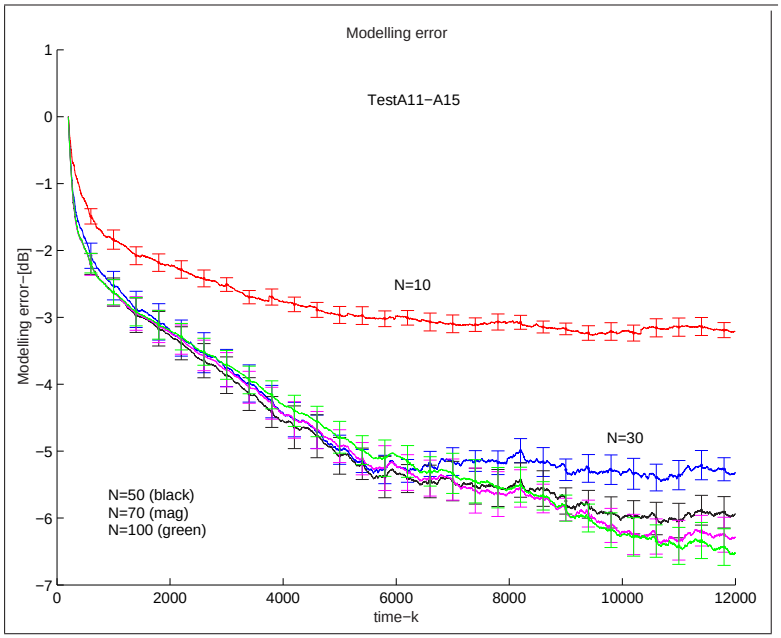


Figure B.7: Test A 11-15(Investigation of N)

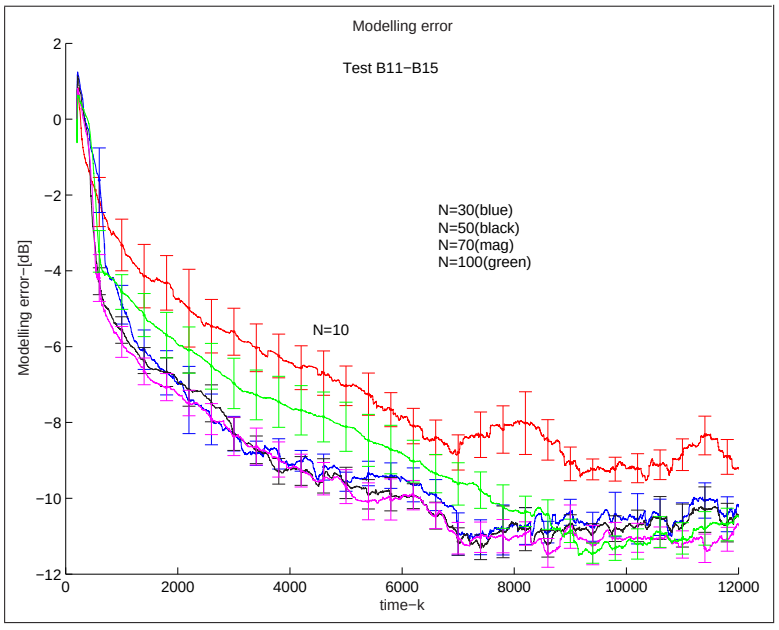


Figure B.8: Test B 11-15(Investigation of N)

(120). Selecting a to small value will cause the rate of convergence to be slow, but the system mismatch as well as the variance of the system mismatch is low. In the three different environments a selection of a to big η will make the convergence speed fast, but the system mismatch larger. This test should illustrate that even though the selection of η is crucial to the end performance, the algorithm is not so sensitive

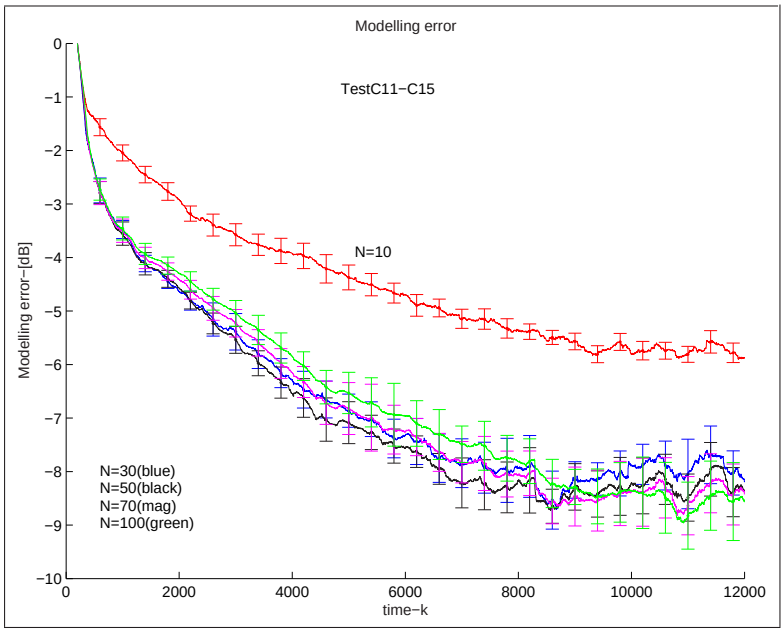


Figure B.9: Test C 11-15(Investigation of N)

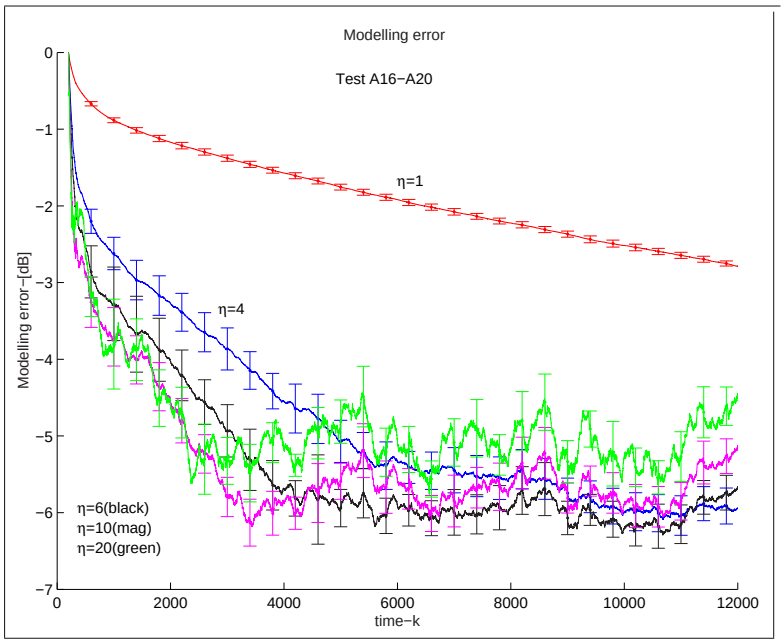


Figure B.10: Test A 16-20, investigation of η

on selection on this parameter.

In the last test the initialization factor δ is investigated. The modelling error can be seen from figure (B.13) page (121), (B.14) page (121) and (B.15) page (122) for the three different tests. In the simulations a positive modelling error have been observed during the first samples. The cost function (and its derivatives) need some

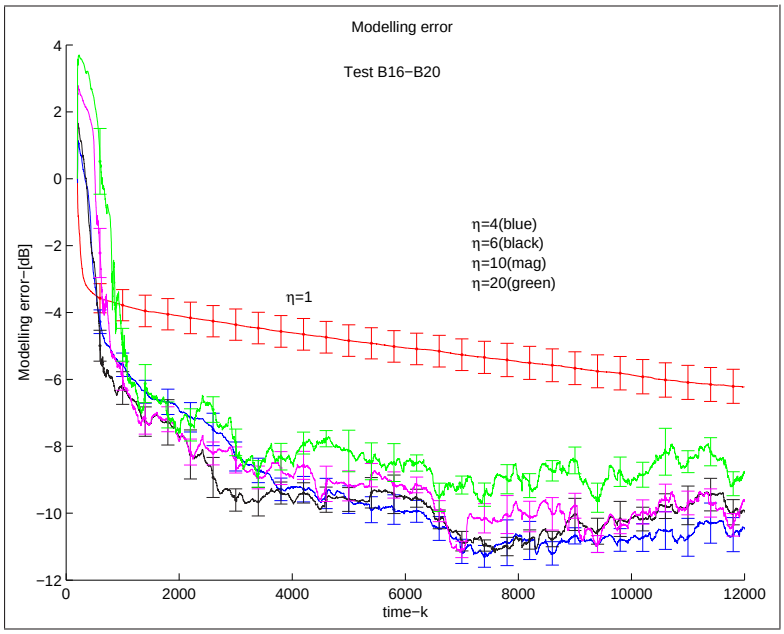


Figure B.11: Test B 16-20, investigation of η

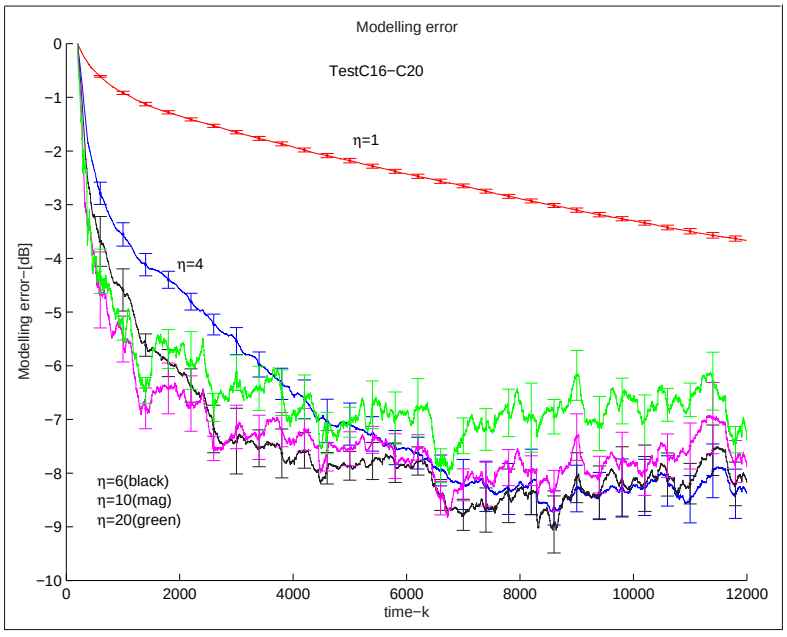


Figure B.12: Test C 16-20, investigation of η

samples to build up a good empirical density estimate of $p(\boldsymbol{x}, y)$. This means that local maximums/stationary points will exist, causing the weights to be different from the true weights. A very severe case where the weights jump into a local maximum is seen in Test case B ((B.14) page (121)) where the model parameters are far from the true ones until k=3500.

A selection of $\delta < 0$ introduces a kind of weight decay which was shown for the

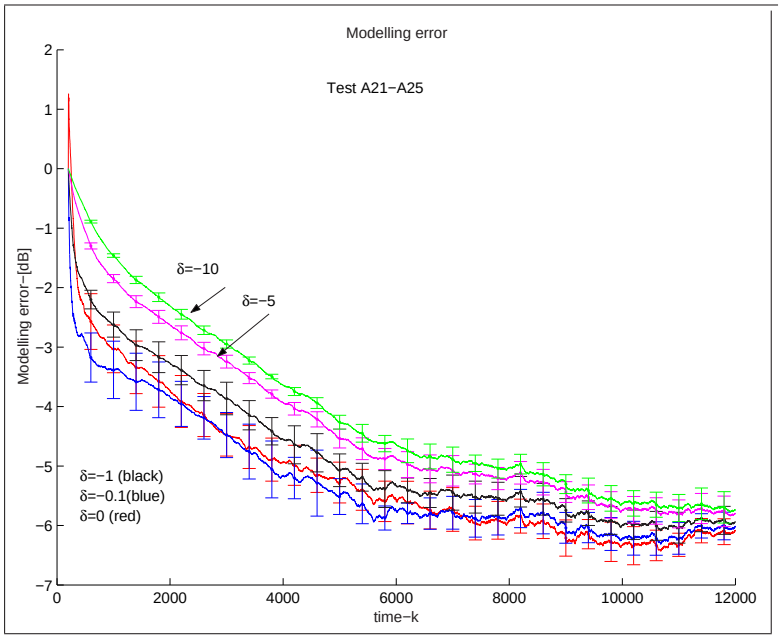


Figure B.13: Test A 21-25, investigation of initialization constant δ

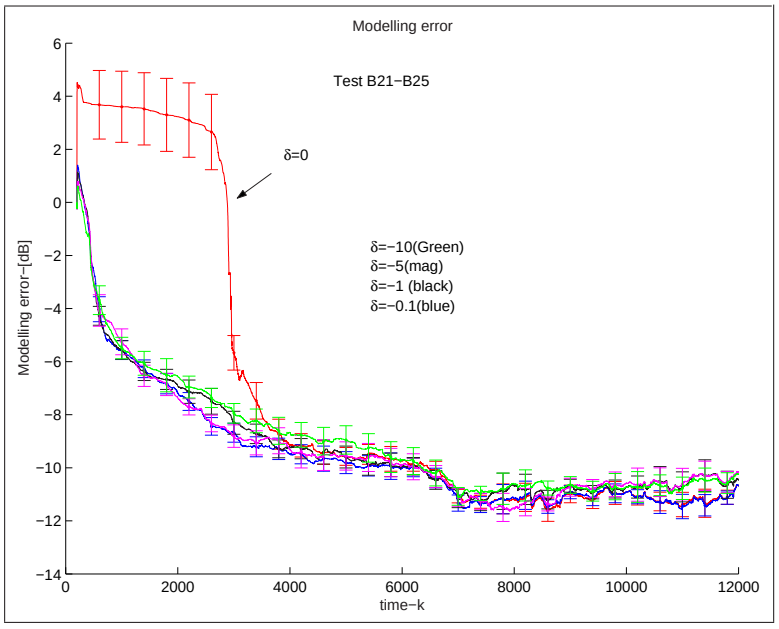


Figure B.14: Test B 21-25, investigation of initialization constant δ

RLS/RLP algorithm. This helps the algorithm not to fall into any local maximums. Selection of different values of δ can seem to have an influence on the convergence speed of the algorithm. I have found through different simulation that an appropriate value for δ can be determined as $\frac{-1}{10\sigma^2}$, where σ is the width of the kernel. In the Gaussian case that gave a estimate of $\delta = -0.4$ which seem to be a good compromise.

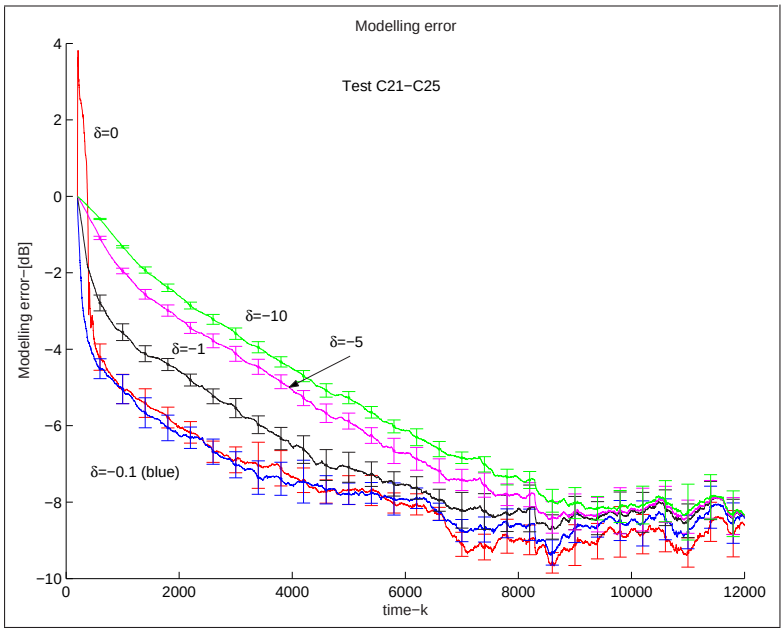


Figure B.15: Test C 21-25, investigation of initialization constant δ

Discussion

As to summarize the algorithm was tested in three different environments. The results was very similar whether the environment was a Gaussian or super Gaussian except for a better system mismatch in the super Gaussian case. The most important parameters seem to be λ, δ and the number of samples N which indirectly controls λ_r.

parameter	red	blue	black	magenta	green	constant
Test Nr.	1	2	3	4	5	
N	10	50	100	200	500	$\eta = 3$

Table B.2: Parameters for the GNBatch algorithm in test D-E-F

B.2 Investigation of the Gauss Newton Batch algorithm

The batch algorithm has two parameters to be controlled namely the step-size and the number of samples used in the PDF-estimate ($p_e(e)$) and in the empirical density estimate of $p(\boldsymbol{x}, y)$. In this section the three environments from the test on the RQEGN algorithm is used again. These environments where

- Gaussian noise (Test D)
- Super-Gaussian noise ($S(1.2)S$)-distributed (Test E).
- Correlated super-Gaussian noise ($S(1.5)S$)-distributed using a moving average filter (Filter length 10) to generate process v_k which is used for the AR(3)-process. (Test F)

The only parameter, which is investigated using this algorithm are the window length (N). The different values of window length investigated can be seen from table (B.2) page (123).

The simulations was performed with samples sizes of $nsamp = 30000$ moreover, the modelling error is determined from the mean of 10 realizations. In addition, the spread is shown on the mean value. The modelling error obtained in the three different environments is seen from figure (B.16) page (124), (B.17) page (124) and (B.18) page (125). As can be seen from the different tests the convergence properties are more or less equal in the three different environments. Again as with the RQEGN algorithm the system mismatch is lower with the $S(\alpha)S$ environment than the Gaussian environment. Using a low sample-size in the empirical density estimate of $p(\boldsymbol{x}, y)$ and parzen estimate of $p_e(e)$ gives a quite big variance on the model error, as well as big system mismatch. Increasing the number of samples in the estimates gives partly lower system mismatch but also decreases the variance of the estimate, which can be seen in all cases for N=200 and N=500. To see how the model parameters converges to the AR parameters in the three cases the adaptive weights have been plotted against the sample number in figure (B.19) page (125), (B.20) page (126) and (B.21) page (126) for the three different environments using N=100(The taps is the result of averaging 10 times).

From the taps a clear mismatch is seen in the Gaussian case, where one of the coefficients (0.9) is around 0.05 from the true value.

Discussion

The batch algorithm is minimizing an undisturbed error-distribution, which should make it more "correct". The algorithm is updated in batches and requires at least

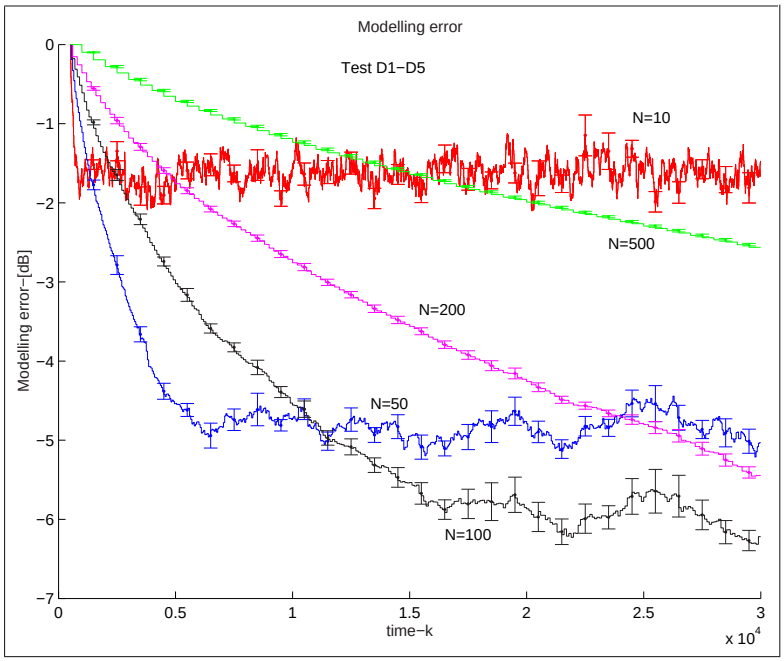


Figure B.16: Test D 1-5, testing different window lengths N in the empirical density estimate (and error density estimate)

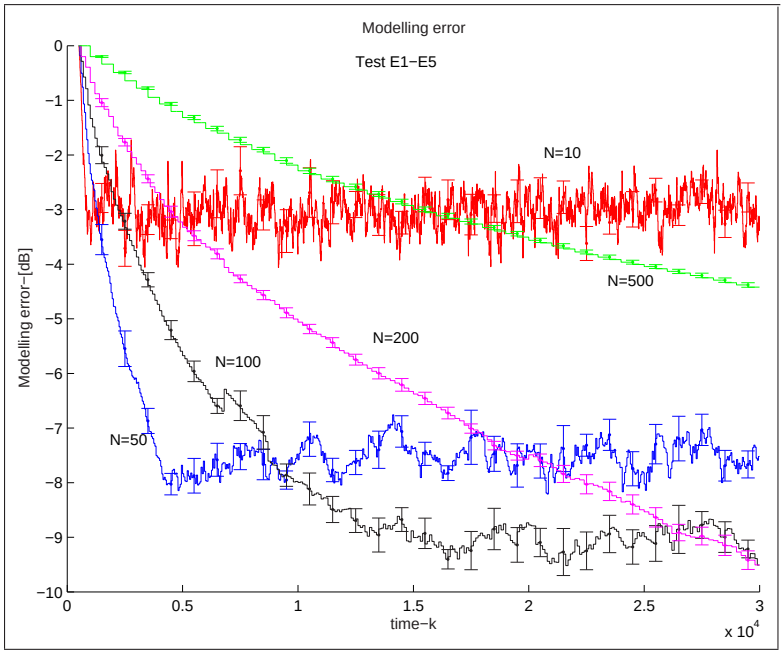


Figure B.17: Test E 1-5, testing different window lengths N in the empirical density estimate (and error density estimate)

$N > L$ to be efficient. This algorithm have only been developed for comparison purposes with the recursive method. The batch and recursive method show to have

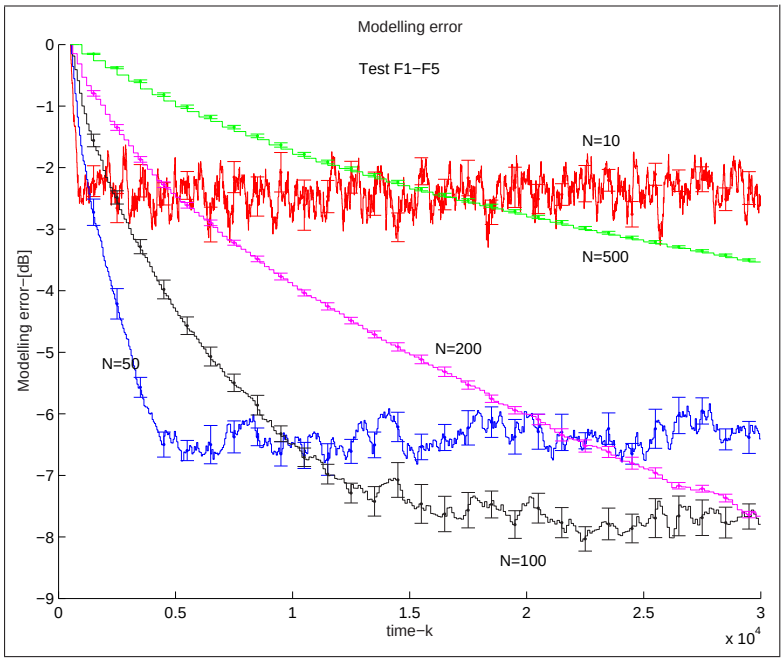


Figure B.18: Test F 1-5, testing different window lengths N in the empirical density estimate (and error density estimate)

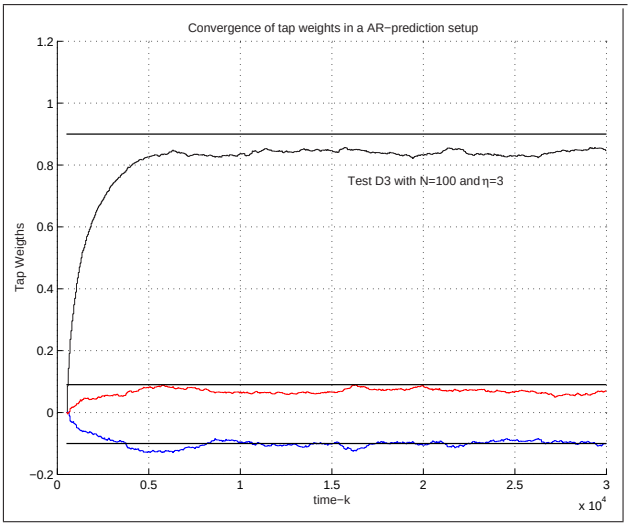


Figure B.19: Convergence of tap weights in test D - 3 (N=100) and $\eta = 3$.

more or less similar performance, except for a more smooth convergence when using the RQEGN method.

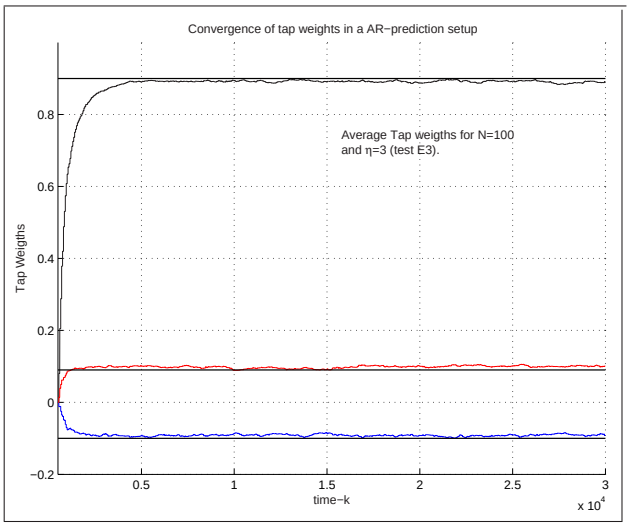


Figure B.20: Convergence of tap weights in test E - 3 (N=100) and $\eta = 3$.

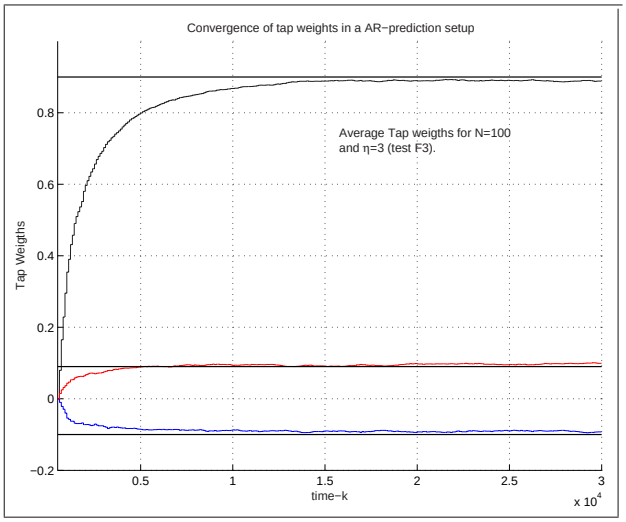


Figure B.21: Convergence of tap weights in test F - 3 (N=100) and $\eta = 3$.

Bibliography

[1] *Numerical Recipies in C: The art of scientific computing*, chapter 7.2. Cambridge University press, 1988-1992.

[2] Chris M. Bishop. Training with noise is equivalent to tikhonov regularization. *Neural Computing Research Group Report NCRG*, April 1994.

[3] Christopher M. Bishop. *Neural Networks for Pattern Recognition.* Oxford university press, 2000.

[4] John S. Bodenschatz and Chrysostomos L. Nikias. Symmetric alpha-stable filter theory. *IEEE Transactions on signal processing vol. 45, No.9*, pages 2301–2306, 1997.

[5] Olivier Chapelle, Jason Weston, Leon Bottou, and Vladimir Vapnik. Vicinal risk minimization. In *NIPS*, pages 416–422, 2000.

[6] Jose C. Principe Deniz Erdogmus. An on-line adaption algorithm for adaptive system training with minimum error entropy: Stochastic information gradient. 2000.

[7] Jose C. Principe Deniz Erdogmus and Kenneth E. Hild II. Beyond second-order statistics for learning: A pairwise interaction model for entropy estimation. *Natural Computing*, pages 85–108, 2002.

[8] Scott C. Douglas. A family of normalized lms algorithms. *IEEE signal processing letters*, pages 49–51, 1994.

[9] J.M. Borwein E.J. Borowski. *Collins Dictionary of Mathematics.* HarperCollins-Publishers, 1989(Reprint 109).

[10] Deniz Erdogmus and Jose C. Principe. Convergence analysis of the information potential criterion in adaline training. pages 123–132, 2001.

[11] Deniz Erdogmus and Jose C. Principe. Generalized information potential criterion for adaptive system training. *IEEE Transactions on Neural Networks*, feb. 2001.

[12] Apostolos Th. Georgiadis and Bernard Mulgrew. A family of recursive algorithms for channel identification in alpha-stable noise. *Fifth Bayona Workshop on Emerging Technologies in Telecommunications*, 1999.

[13] A. Golan and J. M. Perloff. Comparison of maximum entropy and higher-order entropy estimators. *J. Econometrics 107*, pages 195–211, 1-2 (2002).

[14] Orhan Arikan Gul Aydin and A.Enis Cetin. Robust adaptive filtering algorithms for α stable random processes. *IEEE Transactions on Circuits and Systems - II*, pages 198–202, 1999.

[15] Simon Haykin. *Adaptive Filter Theory.* Prentice-Hall Inc., 3 edition, 1996.

[16] Simon Haykin. *Neural Networks - A comprehensive foundation.* Prentice-Hall

International, Inc., 2 edition, 1999.

[17] C.L. Mallows J.M.Chambers and B.W. Stuck. A method for simulating stable random variables. *Journal of the American Statistical Association*, 71(354): 340–344, June 1976.

[18] John W. Fisher III Jose C. Principe, Dongxin Xu. Information-theoretic learning. A part of a book, 2000.

[19] Qun Zhao John W. Fisher III Jose C. Principe, Dongxin Xu. Learning from examples with information theoretic criteria. *No Journal*, 2000.

[20] Preben Kidmose. Adaptive filtering for non-gaussian processes. *Proceedings of International Conference on Acoustics, Speech and Signal Processing*, pages 424–427, 2000.

[21] Preben Kidmose. Alpha-stable distributions in signal processing of audio signals. *proceedings of SIMS2000*, pages 87–94, 2000.

[22] Preben Kidmose. *Blind Separation of Heavy Tail Signals.* PhD thesis, DTU - Technical university of Denmark, September 2001. www.imm.dtu.dk/documents/ftp/publphd.html.

[23] Lennart Ljung. *System Identification Theory for the user.* Prentice-Hall information and system sciences series, 1987.

[24] N. K. Poulsen L. K. Hansen M. Nørgaard, O. Ravn. *Neural Networks for Modelling and Control of Dynamic Systems.* Springer-Verlag, London, 2000.

[25] Xinyu Ma and Chrysostomos L. Nikias. Parameter estimation and blind channel identification in impulsive signal environments. *IEEE Transactions On Signal Processing*, 43(12):2884–2897, December 1995.

[26] A. Enis Cetin Orhan Arikan, Murat Belge and Engin Erzin. Adaptive filtering approaches for non-gaussian stable processes. *IEEE transactions on signal processing*, pages 1400–1403, 1995.

[27] Emanuel Parzen. On estimation of a probability density function and mode. *Annals of Mathematical Statistics*, 33:1065–1076, september 1962.

[28] Jose C. Principe and Deniz Erdogmus. From adaptive linear to information filtering. *IEEE transaction on signal processing*, pages 99–104, 2000.

[29] John G. Proakis and Dimitris G. Manolakis. *Digital Signal Processing.* Prentice-Hall International, INC., 3 edition, 1996.

[30] Min Shao and Chrysostomos L. Nikias. Signal processing with fractional lower order moments: Stable processes and their applications. 81:986–1010, July 1993.

[31] Kyle Siegrist. Virtual laboratories in probability and statistics. http://www.ds.unifi.it/VL, 1997.

[32] B.W. Silverman. *Density Estimation for Statistics and Data Analysis.* Chapman and Hall, 1986.

[33] Donald F. Specht. A general regression neural network. *IEEE Transactions on Neural Networks*, 2(6):568–576, November 1991.

[34] Murray R. Spiegel. *Mathematical Handbook of formulas and tables.* McGraw-Hill, inc., 1968.

[35] Eric Weisstein. Mathworld. http://mathworld.wolfram.com/InterquartileRange.html, 2000.