NTNU – Trondheim
Norwegian University of
Science and Technology

Calibration of mismatch effects in time-interleaved analog-to-digital converters

Anastasia A Kabaeva

Master of Science in Communication Technology
Submission date: June 2013
Supervisor: Kimmo Kansanen, IET

Norwegian University of Science and Technology
Department of Electronics and Telecommunications

Norwegian University of Science and Technology

Master thesis

Calibration of mismatch effects in time-interleaved analog-to-digital converters

Author:
Anastasia Kabaeva

Supervisor:
Associate Prof. Kimmo Kansanen

June 10, 2013

Abstract

The performance of time-interleaved analogue-to-digital (A/D) converters is often significantly degraded by timing mismatch errors. In this project we explore methods for performing reconstruction of signals from recurrent non-uniform samples. We analyse how time-skews affect the spectrum of the output signal. Several methods for estimation of the unknown parameters (time-skew and gain), using techniques required only the outputs of the A/D converters, are represented in this paper. We also present a filter-bank which perform signal reconstruction from these estimates. It is assumed that the input signal is analog band-limited and that the time-skews, which are introduced by the system before each A/D, are smaller than the sampling interval. The system is described in time-domain.

Contents

1 Introduction

Requirements for modern electronic systems as a rule include a reduction in cost, power consumption, and size, along with providing an increase in speed, accuracy, testability, and system robustness. These benefits can be achieved by digital methods which allow to compensate for inadequate analog circuit performance and by allowing system designers to relax analog system constraints.

Converting an analog signal to the digital form includes two main processes: sampling and quantization. A single analog-to-digital (A/D) converter has to provide sampling with very fast sampling rate (F_s) in order to handle modern system requirements. Because the cost of A/D converters and power consumption grow faster than linearly with sampling rate, the method which distributes the load across many converters becomes very attractive for systems with high sampling rates. In other words, we can benefit from converting a continuous-time band-limited signal to a discrete-time signal using a number of A/D converters equal to M, each operating at $1/M$ of the Nyquist rate.

Since the converters are typically not synchronized, the resulting time-discrete signal is a combination of uniform samples, where each sequence corresponds to samples at $1/M$ of the Nyquist rate of a time delayed version of the continuous time signal. Thus, the resulting discrete-time signal corresponds to recurrent non-uniform samples of the continuous-time signal.

It is well known that a band-limited signal is uniquely determined from its non-uniform samples, provided that the average sampling rate exceeds the Nyquist rate.The article "Filter-bank reconstruction of band-limited signals from non-uniform and generalized samples" [1] shows that we can reconstruct such a signal by dividing the time axes into non-overlapping intervals of length MT_s [1] , and apply a reconstruction filter-bank by assuming that time-skews are constant and known to the filter-bank.

Because it is impractical to manufacture totally identical hardware, some other parameters, such as gains introduced by the branches, have to be estimated for improving signal reconstruction. Some systems (for example, systems operating in a big range of temperature) may require estimation to be done periodically or adaptively.

[1] T_s here is the sampling interval

There are two general approaches to system calibration. The first one is to pilot a signal (the signal with is known for the receiver) in the input in order to facilitate direct estimation of the unknown parameters. The various implementation of such a method can have disadvantages such as requiring extra hardware, decreasing the sampling resolution and cause system delays by requiring operation of the converters to be interrupted. The second approach is to perform blind recovery using only the output of A/D converters. In this project the last approach is used.

Thus, our problem is to determine unknown parameters and design a reconstruction filterbank which will be used in order to reconstruct signal efficiently. We will describe the system in the time domain.

This report is divided into 7 chapters. Chapter 2 presents the problem formulation, stating the challenges which relate to the processes of sampling and quantization a designer is faced with. The reconstruction of a signal represented by non-uniform samples is represented in Chapter 3. Chapters 4 and 5 discuss the estimation of the unknown parameters and evaluate performance for the estimates. The system evaluation, including quantization, is described in Chapter 7. In Chapter 8 the conclusion sums up the most important results, and finally an appendix contains the main important scripts and function that have been used for implementation of the system using Matlab.

2 Problem formulation

As applications become more advanced, a single state-of-the-art analogue-to-digital converter (A/D converter) may be insufficient to handle system sampling requirements since it may not able to sample fast enough or may consume a large amount of power to do so. At high sampling rates, time-interleaved A/D converters offer an efficient method of sampling by distributing the load across many A/D converters.

As shown by figure 2.1, time-interleaved A/D converters operate in a round-robin manner. If the system contains M converters, and the sampling period is T_s, then each converter operates with period $T = MT_s$ for each converter and a spacing T_s between consecutive converters. This means that we have reduced the requirement for sampling rate for each converter of the system, but a new problem arises: inaccurate (or non-uniform) sampling. This problem may be caused by differences in a signal path length or other kind of restrictions in the physical layout.

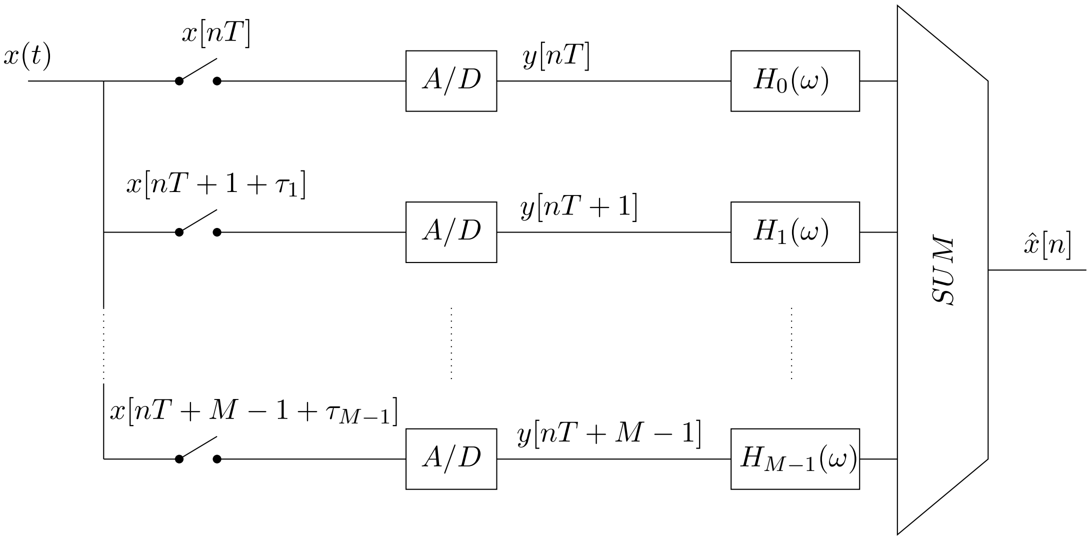

Figure 2.1: Time-interleaved ADC system with M converters

The block diagram in figure 2.1 shows that the input analog signal $x(t)$ is split into M branches, sampled with period T_s, and converted in to the digital signal $\hat{x}[n]$. The role of a filter-bank is to compensate for the time-skews τ, introduced by each branch.

We assume that $x(t)$ is band-limited with cut-off frequency Ω_c, i.e. the Fourier transform is zero for $\Omega_c < |\Omega|$. The overall sampling period T_s is chosen such that the sampling procedure is able to fulfil the Nyquist criterion: the Nyquist rate is the minimum sampling rate required to avoid aliasing, equal to two times the highest frequency component contained within the signal. Restrictions for the input signal are described in more details in Chapter 3.

The time-skew τ_0 introduced by the first branch is taken as reference, i.e. it is equal to zero. Time-skews introduced by the other branches can take any value between $-T_s < \tau_i < T_s$. So, the time-skews are defined by following interval vector, where the first element is equal to zero:

$$\tau = [0 \quad \tau_1 \quad \tau_2 \quad \dots \quad \tau_{M-1}] \tag{2.1}$$

The filter-bank uses information about time-skews to compensate distortions of the output signal. To estimate τ in this project, the method of least squares (LS) and some adaptive methods (recursive least squares (RLS) and least mean squares (LMS)) are used. Only outputs from A/D converters are used to perform estimation of time-skew parameter. These three methods described in more details in Chapter 5.

The functions of the A/D converters and the filter-bank for the system presented in figure 2.1 are discussed in the next two sections.

2.1 Analog-to-Digital Conversion

A/D conversion in our system (figure 2.1) is realised by quantizing the sampled signal, i.e. the compression of a discrete signal to a lesser set of representative levels. We assume that the A/D converters in each branch are identical.

Quantization is a many-to-one mapping, as such, there will be loss of data when the quantization is performed on an arbitrary analog signal. The number of quantization levels is determined by the number of bits used to quantize. A quantizer size of n bits will be able to represent 2^n different quantization levels.

In this project, uniform, scalar, mid-rise quantization is used because of its simplicity. It means that the process of quantizing consists of mapping a set of one-dimensional (scalar) signal values with uniform quantization steps, without being able to represent 0 (mid-rise). We assume that this quantization is suitable because the only restriction for input signal is that it has to be band-limited. If we have some other (statistical) information about the input signal, some other type of quantization can be used in order to improve efficiency (or reduce quantization noise). The uniform, scalar, mid-rise quantization process is given by:

$$Quantized = \left\lfloor \frac{S}{Q_{step}} + 0.5 \right\rfloor \Delta \tag{2.2}$$

Where S is the signal sample value, Δ is the quantization step and $\lfloor \cdot \rfloor$ is rounding downwards to the nearest integer.

We first need to find the representative values the quantizer can take. As such we define the dynamic range of the input as $2V$, the number of quantization levels as L, and the step-size as Δ. L is defined by the number of bits (n) used by the quantizer, such that $L = 2^n$. Dynamic range can vary slightly for different A/D converters in the system.

The step size of a uniform scalar quantizer is a function of both the dynamic range of the input and the number of quantization levels, such that:

$$\Delta = \tfrac{2V}{L} \tag{2.3}$$

The quantizer is described by a set of quantized representative levels r_k, where $k = [1 \quad 2 \ldots L]$ and decision boundaries q_i, with $i = [1 \quad 2 \ldots L-1]$. This can be seen in figure 2.2.

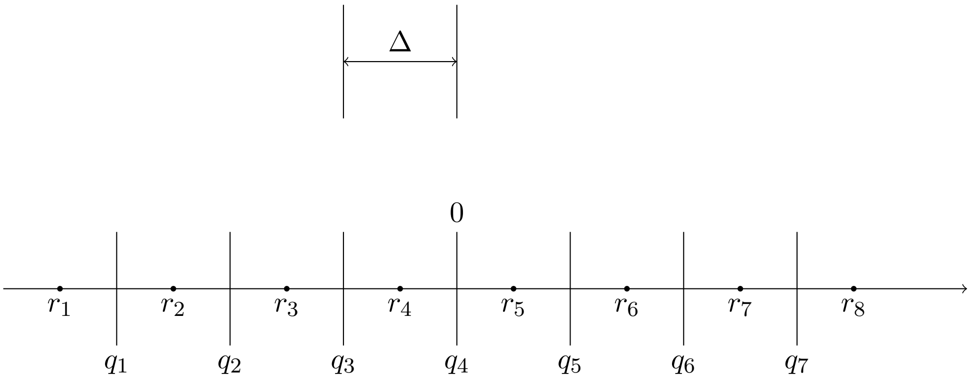

Figure 2.2: Depiction of uniform, scalar, mid-rise quantizer by representative levels and decision boundaries.

Figure 2.2 shows that if an input value is between two successive q's: q_i and q_{i+1}, then the output value from the quantizer is the r_k lying in between of these two decision levels. If the input value is either above the highest, or below the lowest quantizer representative level, the signal value will be represent by the nearest quantization representative level, or simply be "clipped".

If we define the quantization error ϵ as the difference between the quantized sample and the original signal sample, we can calculate the mean squared quantization error and relate it to the quantizer output resolution. For uniform quantizer and signal values distributed uniformly over the given range, the mean squared quantization error is defined as $\frac{\Delta^2}{12}$ [?]. Thus the selection of the dynamic range of the quantizer ($r_1 \to r_k$) results in a trade-off between quantizer resolution and signal clipping. If the r's are chosen closer to each other, with a lower maximum and higher minimum value, the resolution will increase and the mean squared quantization error will be reduced. In return all the signal values above or below r's maximum and minimum will be truncated to the nearest representation level.

If we assume that the input signal is highly correlated, we can improve our system by using linear prediction (or DPCM[1]), that means we can modulate the input signal such that distortions[2] introduced by quantization are reduced.

2.2 Filter-bank reconstruction

The function of the filter-bank is to recover the transmitted signal (from a continuous-time signal that has been distorted by a noisy channel) and the time-skews between samples. According to the Nyquist criterion, the minimum bandwidth required to avoid ISI [3] is half of the symbol rate ($\frac{1}{2T}$). There are an infinite number of pulses that satisfy the Nyquist criterion, and in practical systems *Raised cosine pulses* is commonly used [3]. The bandwidth W of such a pulse is larger than its minimum value (which corresponds to ideal band-limited pulses) by a factor of $1+\alpha$:

$$W = \frac{1+\alpha}{2T} \tag{2.4}$$

and the pulse shape of the *raised cosine pulses* is given by:

$$g(t) = \left(\frac{\sin(\pi t/T)}{\pi t/T}\right)\left(\frac{\cos(\alpha\pi t/T)}{1-(2\alpha t/T)^2}\right) \tag{2.5}$$

This gives us pulses with zero-crossing at multiples of T (here we have $T = T_s$). This is illustrated by figure 2.3. There are two parameters which can be optimised in filter design: roll-off factor α and filter length K. The filter length K assumes that we have duration of the filter impulse response equals $(2K+1)T$ or $t \in (-KT, KT)$ for $K = 1, 2, \ldots$). With increasing α follows an increase in bandwidth, but the pulse tail decays more rapidly in the time domain. The same with the length of the filter: as we increase the length, characteristics of the filter becomes more accurate. In this project we use $\alpha = 0.5$. Choosing the value of K is discussed in chapter 7.

[1] Differential Pulse Code Modulation

[2] Distortions here is measured as $D = \sum_i \frac{(\hat{x}[i]-x[i])^2}{x[i]^2}$

[3] Inter symbol interference

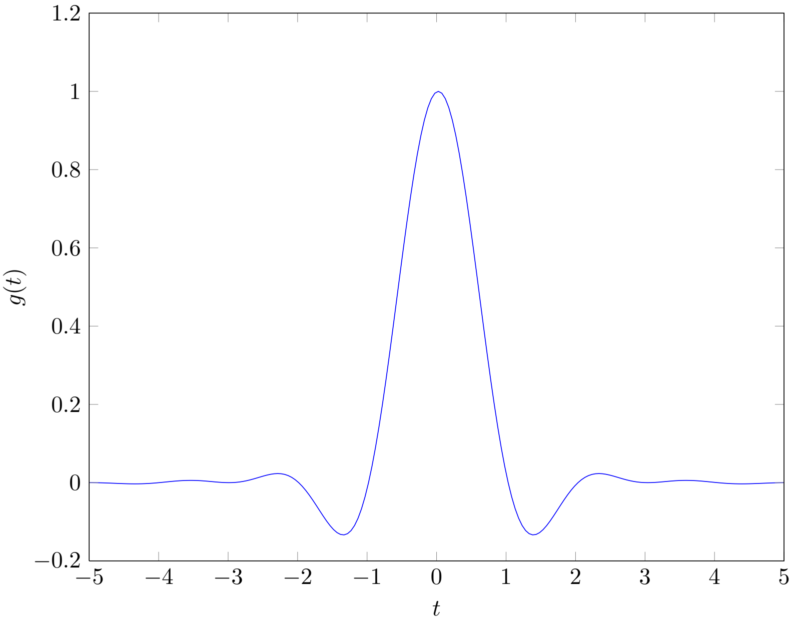

Figure 2.3: Cosine roll-off filter in the time domain. Parameters: $\alpha = 0.5$, $K = 5$

As we can see from figure 2.3, the pulse has its maximum value (which is equal to 1) at $t = 0$ and zero-crossing approximately at $t = nT, \quad \text{for} \quad n = \pm 1, 2, \ldots$. So this filter can reconstruct the signal, which is sampled with sampling rate $T_s = T$, perfectly, without ISI.

On the other hand, if we have some time-skew τ (as was defined earlier) in the input pulse-sequence, we no longer have zero-crossing at $t = nT, \quad \text{for} \quad n = \pm 1, 2, \ldots$ and maximum is no longer equal to 1 at $t = 0$. This is illustrated by figure 2.4. As a consequence, we have distortions of the reconstructed signal, caused by ISI.

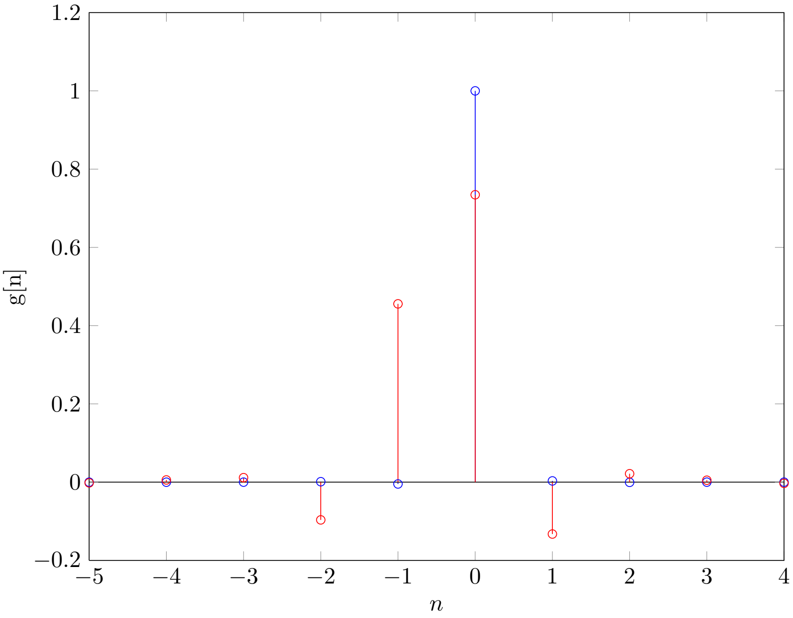

Figure 2.4: Inaccuracy in cosine roll-off filter coefficients caused by the time-skew. (Blue - coefficients without any time-skews, red - time-skewed coefficients)

The filter-bank have to use information about τ in order to compensate for distortions, which are caused by time-skews (or ISI-distortions) of the output signal. As was mentioned above, to find τ, we use oversampling and take advantage of the excess bandwidth in the system. The filter-bank treats the signal in each branch separately, and adds them together at the output. The main principles of designing the filter-bank are described in detail in Chapter 4.

3 Signal modelling

As was already mentioned in Chapter 2, the input signal is restricted to be band-limited with cut-off frequency Ω_c, i.e. the Fourier transform is zero for $\Omega_c < |\Omega|$. In this chapter we discuss modelling of the signals which fulfil these restrictions.

There are a lot of functions which fulfils the restrictions for input signal. First of all we model our input signal in the time domain as sinc-shaped pulse because it is strictly band-limited in the frequency domain. And then we model input signal as band-limited white Gaussian noise.

3.1 Input signal modelled as sinc-function

In order to make an adjustment of the input signal we model our signal such that the bandwidth of the input signal can be changed. Parameters which define the signal bandwidth in the frequency domain are time interval, sampling interval and length of the signal. In the Matlab implementation the function which is used for signal modelling in this project is $x = gensignal(tau, l, time, w_c)$, where

- τ is a vector of M elements representing time-skews between the first and other branches;
- l is a scalar representing the length of the signal as number of samples (signal is represented as vector, l is length of this vector);
- $time$ is the length of the signal measured in seconds (sec) such that we can represent sampling interval $T_s = \dfrac{time}{l}$ measured in sec; and $F_s = \dfrac{1}{T_s} = \dfrac{l}{time}$ measured in Hz.
- w_c represents the desired bandwidth of the sampled signal.

The next sections demonstrate more detailed examples of modelling the input signal.

3.1.1 Example for the case with two branches

As example for modelling band-limited signal, we first use a determined (sinc-shaped) function. In Matlab implementation an analog signal is always represented as discrete signal with a small sampling interval. Analog version of the signal $x(t) = \dfrac{sin(w_c t)}{\pi t}$ with parameters $M = 2$, $l = 2000$; $time = 200$; $w_c = 0.1 * 2\pi$ in the time domain is represented below:

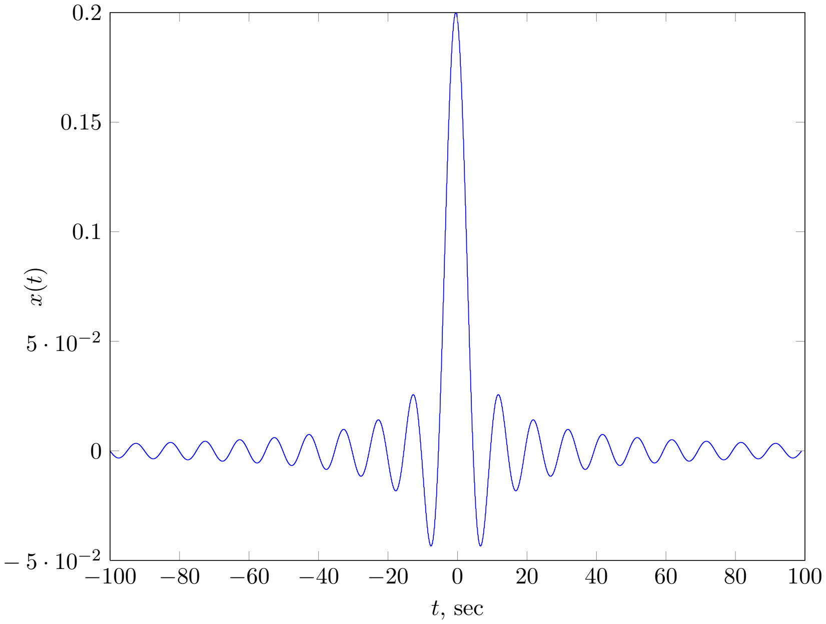

Figure 3.1: Analog signal $x(t)$

The highest (or cut-off) frequency for a given signal is $w_c = 0.2\pi$. According to the Nyquist criteria, to avoid aliasing, sampling rate F_s ($w_s = 2\pi F_s$) for this signal has to be equal to or exceed $2F_c$ ($w_c = 2\pi F_c$). We can then sample the signal using sampling frequency $F_s \geq 0.2\,Hz$. The signal in figure 3.1 is represented as the sampled signal with $F_s = 10\,Hz$. Because sampling rate is 50 times more then we should take to fulfil Nyquist criteria, it looks like an analog signal. The signal sampled with sampling rate $F_s = 1\,Hz$ (or five times bigger than the Nyquist rate for the signal) is represented in 3.2. We can see that the length of the sampled signal $y[n]$ is ten times smaller than the length of the analog signal $x(t)$.

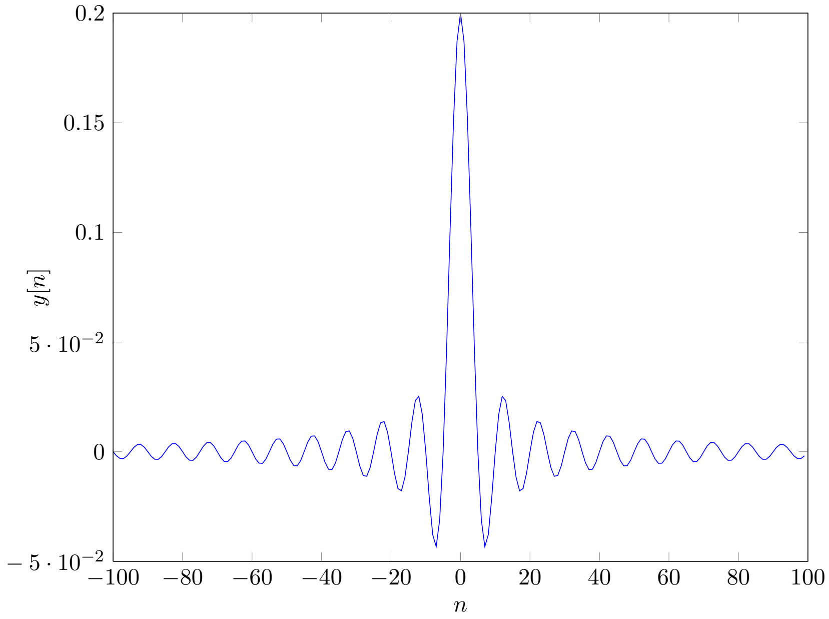

Figure 3.2: Signal $y[n]$ modelled as a sampled version of the analoge signal with sampling interval ten times bigger than for signal $x(t)$. Parameters: $l = 200, \tau = [0\,0]$

As we can see from figures 3.1 and 3.2, although the form of the sampled signal is not so smooth as for analog signal, it is almost the same. It means there would not be any problem to reconstruct the analog signal from its samplings if we choose a sampling rate fulfilling the Nyquist criteria.

In order to analyse the signal in the frequency domain, we take the Fourier transformation for a discrete time aperiodic signal [4]. Analysis equation for Discrete Time Fourier Transform (DTFT) is:

$$X(w) = \sum_{n=-\infty}^{\infty} x[n]e^{-jwn} \tag{3.1}$$

In our example signal $y[n]$ is a sampled version of $x(t)$ consisting of signals from M branches. Than the DTFT of $y[n]$, assuming no time-skews between branches, is:

$$\begin{aligned} Y(w) &= \sum_{n=-\infty}^{\infty} y[n]e^{-jwn} && (3.2)\\ &= \sum_{n=-\infty}^{\infty} \frac{\sin(w_c n)}{\pi n} e^{-jwn} && (3.3) \end{aligned}$$

For $M = 2$ we can represent odd samples as $y_{odd}[n] = x(t)|_{t=(2k+1+\tau_1)T_s}$ and even samples as $y[n]_{even} = x(t)|_{t=2kT_s}$ for $k = 0, \pm 1, \pm 2, \dots \pm \frac{l}{2}$. Without loss of generality, we assume $T_s = 1$.

$$Y(w) = \frac{-j}{2\pi} \sum_{k=-l/4}^{l/4} \left[\frac{(e^{jw_c 2k} - e^{-jw_c 2k})e^{-jw2k}}{2k} + \frac{(e^{jw_c(2k+1+\tau_1)} - e^{-jw_c(2k+1+\tau_1)})e^{-jw(2k+1)}}{2k+1+\tau_1} \right] \tag{3.4}$$

$$= \frac{-j}{2\pi} \sum_{k=-l/4}^{l/4} \left[\frac{e^{j2k(w_c-w)} - e^{-j2k(w_c+w)}}{2k} + \frac{e^{j(2k+1)(w_c-w)+jw_c\tau_1} - e^{-j(2k+1)(w_c+w)-jw_c\tau_1}}{2k+1+\tau_1} \right] \tag{3.5}$$

Now we assume τ to be small and apply first order Taylor series expansion for the exponential function e^x with complex x.

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} \tag{3.6}$$

$$= 1 + x + O(x^2) \tag{3.7}$$

For our example now we have $e^{jw_c\tau_1} \approx 1 + jw_c\tau_1$ and $e^{-jw_c\tau_1} \approx 1 - jw_c\tau_1$. We rewrite expression 3.5 as:

$$\begin{aligned} Y(w) = &\frac{-j}{2\pi} \sum_{k=-l/4}^{l/4} \left[\frac{e^{j2k(w_c-w)} - e^{-j2k(w_c+w)}}{2k} \right] \\ &+ \frac{-j}{2\pi} \sum_{k=-l/4}^{l/4} \left[\frac{e^{j(2k+1)(w_c-w)} - e^{-j(2k+1)(w_c+w)}}{2k+1+\tau_1} \right] \\ &+ \frac{w_c\tau_1}{2\pi} \sum_{k=-l/4}^{l/4} \left[\frac{e^{j(2k+1)(w_c-w)} - e^{-j(2k+1)(w_c+w)}}{2k+1+\tau_1} \right] \end{aligned} \tag{3.8}$$

If we deal with uniform sampling (or $\tau_1 = 0$), the last term disappears, and 3.8 can be rewritten as:

$$Y(w)_{uniform} = \frac{-j}{2\pi}\sum_{k=-l/4}^{l/4}\left[\frac{e^{j2k(w_c-w)}-e^{-j2k(w_c+w)}}{2k}+\frac{e^{j(2k+1)(w_c-w)}-e^{-j(2k+1)(w_c+w)}}{2k+1}\right] \tag{3.9}$$

$$= \frac{-j}{2\pi}\sum_{n=-l/4}^{l/4}\frac{\left(e^{jn(w_c-w)}-e^{-jn(w_c+w)}\right)}{n} \tag{3.10}$$

$$= \sum_{n=-l/2}^{l/2}\frac{\sin(w_c n)}{\pi n}e^{-jwn} \tag{3.11}$$

$$\approx \begin{cases}1, & |w| \leq w_c\\ 0, & w_c < |w| \leq \pi\end{cases} \tag{3.12}$$

The spectrum $Y(f)$[1] for the uniformly sampled signal $y[n]$ is shown in figure 3.3.

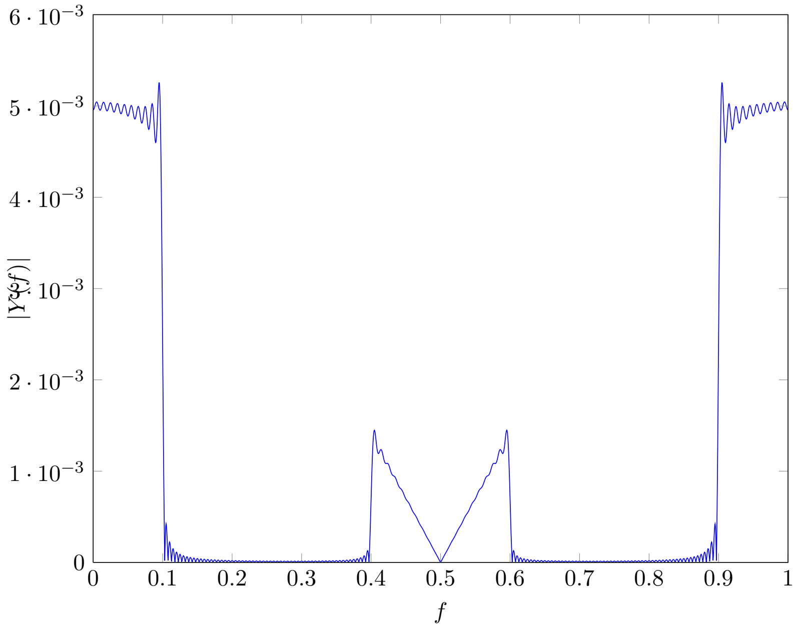

Figure 3.3: Spectrum of the uniformly sampled signal $y[n]$

The oscillatory behavior (Gibbs phenomenon) which we can see near the band edge is due to the truncation of the input sinc-function in the time domain. Truncation of the length (or multiplication of $y[n]$ with rectangular window) is known to introduce ripples in the frequency response characteristic due to the non-uniform convergence of the Fourier series at a

[1] the digital frequency $f = \frac{F}{F_c}$; $w = 2\pi f$

discontinuity [3], Chapter 10.2.2.

If $\tau_1 \neq 0$, the first two terms of equation 3.8 does not give the same spectrum as shown in figure 3.3 because of the second term is divided by τ_1 The spectrum consisting of the first two terms for the case with $\tau = [0\,0.9]$ is represented in figure 3.4.

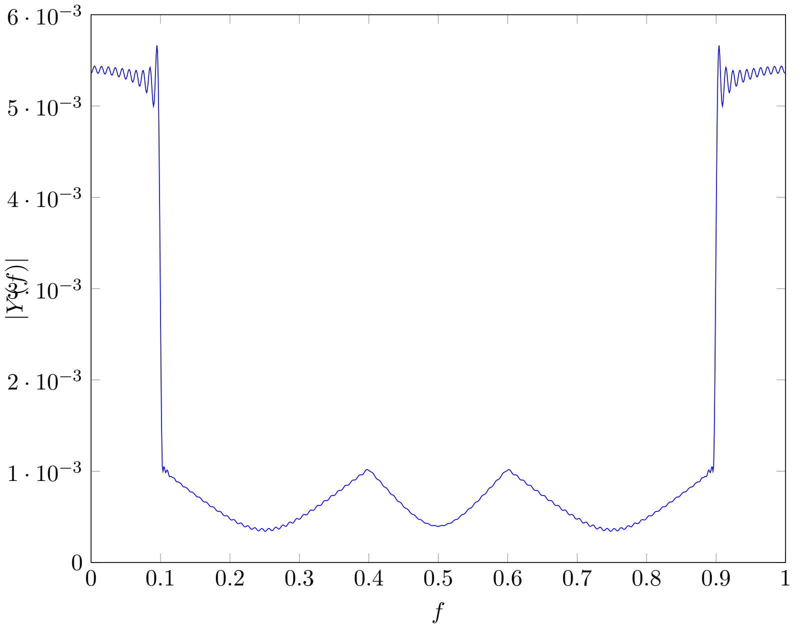

Figure 3.4: Spectrum for the non-uniformly sampled signal $y[n]$, consisting of the first two terms of equation 3.8

The last term in 3.8 can be rewritten as

$$\frac{w_c\tau_1}{2\pi}\sum_{k=0}^{l/2}\frac{e^{j(2k+1)(w_c-w)}-e^{-j(2k+1)(w_c+w)}}{2k+1+\tau_1} = w_c\tau_1\sum_{k=0}^{l/2}\frac{\cos((2k+1)w_c)}{\pi(2k+1+\tau_1)}e^{-j(2k+1)w} \tag{3.13}$$

The spectrum given by the third term in equation 3.8 is shown in figure 3.5.

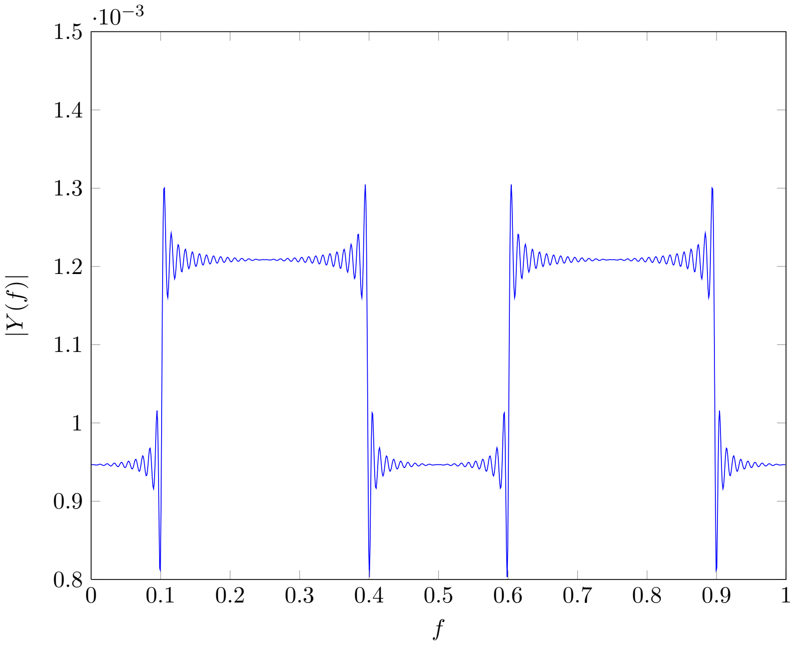

Figure 3.5: Spectrum for the non-uniformly sampled signal $y[n]$, consisting of the third terms of equation 3.8

The figure 3.6 shows the spectrum for the non-uniformly sampled signal given by equation 3.8.

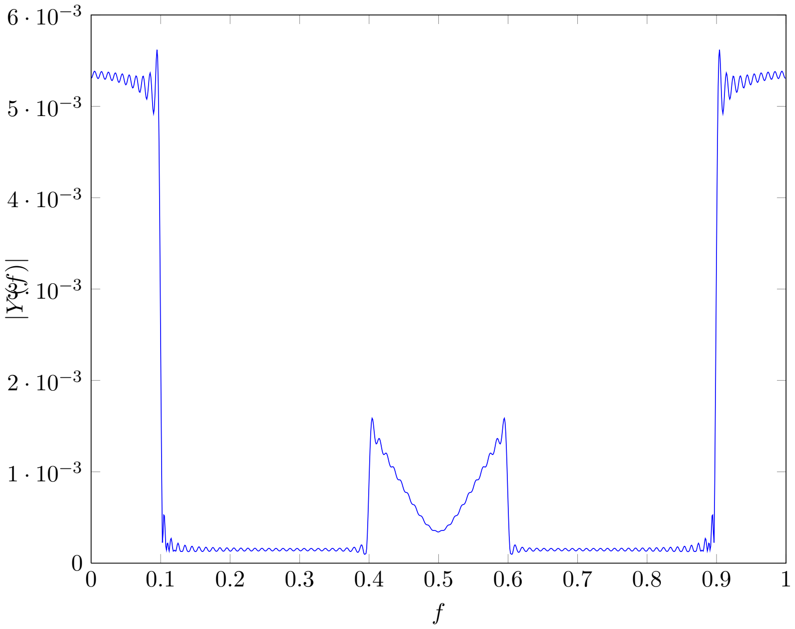

Figure 3.6: Spectrum for the non-uniformly sampled signal $y[n]$, given by equation 3.8

The approximation used in 3.7 can be improved by using second and higher orders of Taylor approximation. The expression for second order Taylor approximation is represented below:

$$e^x = \sum_{n=0}^{\infty} \frac{x^n}{n!} \tag{3.14}$$

$$= 1 + x + \frac{x^2}{2!} + O(x^3) \tag{3.15}$$

If we use second order Taylor approximation for expression 3.5, equation 3.8 will have one more term:

$$\frac{-j}{2\pi} \sum_{k=0}^{l/2} \frac{-w_c^2 \tau_1^2 \left(e^{j(2k+1)(w_c - w)} - e^{-j(2k+1)(w_c + w)}\right)}{2(2k+1+\tau_1)} = \frac{-w_c^2 \tau_1^2}{2} \sum_{k=0}^{l/2} \frac{\sin((2k+1)w_c)}{\pi(2k+1+\tau_1)} e^{-j(2k+1)w} \tag{3.16}$$

Figure 3.7 shows the spectrum given by the equation 3.16. The spectrum for the non-uniformly sampled signal consisting of four therms is represented in figure 3.8.

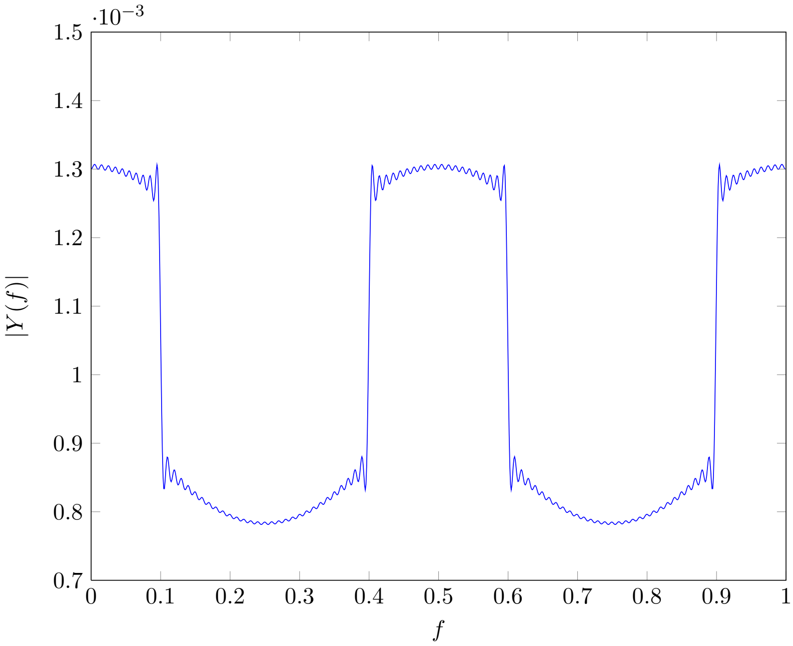

Figure 3.7: Spectrum for the non-uniformly sampled signal $y[n]$, consisting of the fourth term given by equation 3.16

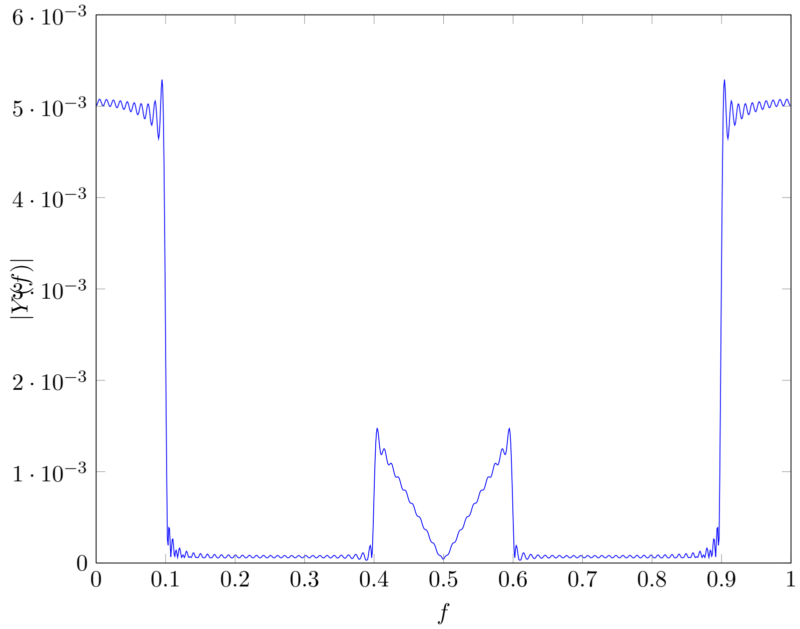

Figure 3.8: Spectrum for the non-uniformly sampled signal $y[n]$, including the fourth term given by equation 3.16

If we continue to increase the order of Taylor approximation, the resulting spectrum for the non-uniformly sampled signal $y[n]$ will have a shape as represented by figure 3.9.

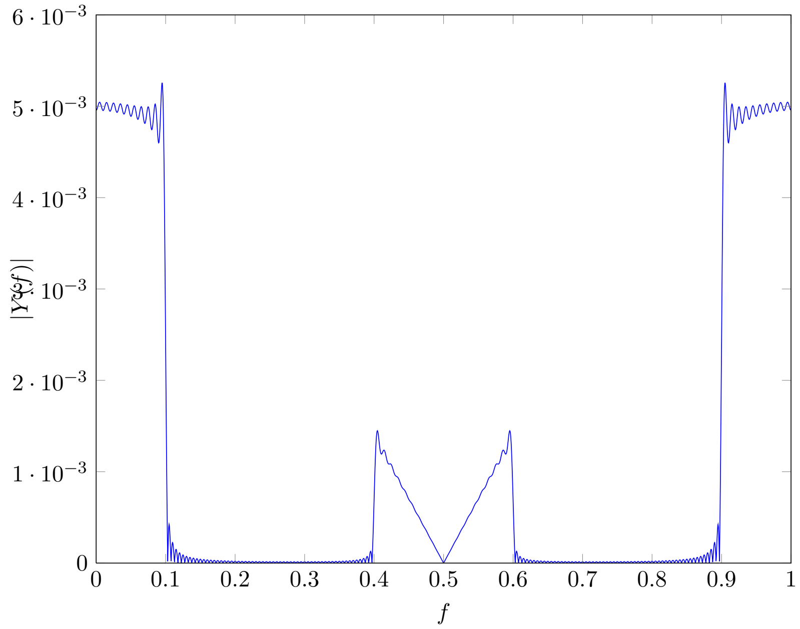

Figure 3.9: Spectrum of the sampled signal $y[n]$, $\tau = [0\,0.9]$

As we see from figure 3.9 with $\tau_1 \neq 0$ we have the frequency components which are symmetric around $f = \frac{n}{2}$ for $n = 0, \pm 1, \pm 2 \ldots$ and have the width of 0.1 (the same as band for the signal). These lobes increase in magnitude with τ_1. With small oversampling (in our example we have sampling rate five times bigger than we need to fulfil the Nyquist criteria), it can result in aliasing and distortions in reconstructed signal. We can also see that the shape of the spectrum in the band $|f| \leq 0.1$ is not rectangular as in figure 3.3 but it is slightly affected by the magnitude of τ_1.

If we have a system with $M > 2$, then we can rewrite the samples from p-branch as $y[n]_p = x(t)|_{t=(Mk-p+\tau[p])T_s}$ for $k = 0, \pm 1, , \pm 2 \ldots \frac{l}{2M}$. In the case with M branches, equation 3.8 can be rewritten as

$$Y(w) = \frac{j}{2\pi} \sum_{p=0}^{M-1} \sum_{k=\frac{-l}{2M}}^{\frac{l}{2M}} \left[\frac{e^{j(kM+p)(w_c-w)} e^{j\tau_p(w_c-w)} - e^{-j(kM+p)(w_c+w)} e^{-j\tau_p(w_c+w)}}{kM+p+\tau_p} \right] \tag{3.17}$$

The spectrum for the non-uniform sampled signal for the case with $M = 4$ and $\tau = [00.90.90.9]$ is shown in figure 3.10.

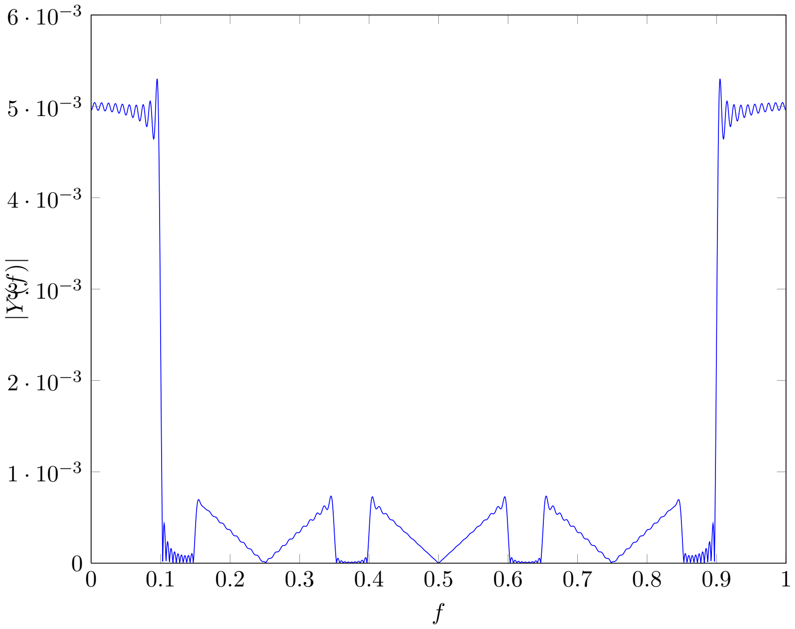

Figure 3.10: Spectrum of the sampled signal $y[n]$, $\tau = [0\,0.9\,0.9\,0.9]$

As we can see from figure 3.10, for the case with $M = 4$ we have the frequency components which are symmetric around $f = \frac{n}{M}$ for $n = 0, \pm 1, \pm 2 \ldots$ and have the width of 0.1. For the case with $M \geq 5$ these lobes will cause aliasing as shown by figure 3.11.

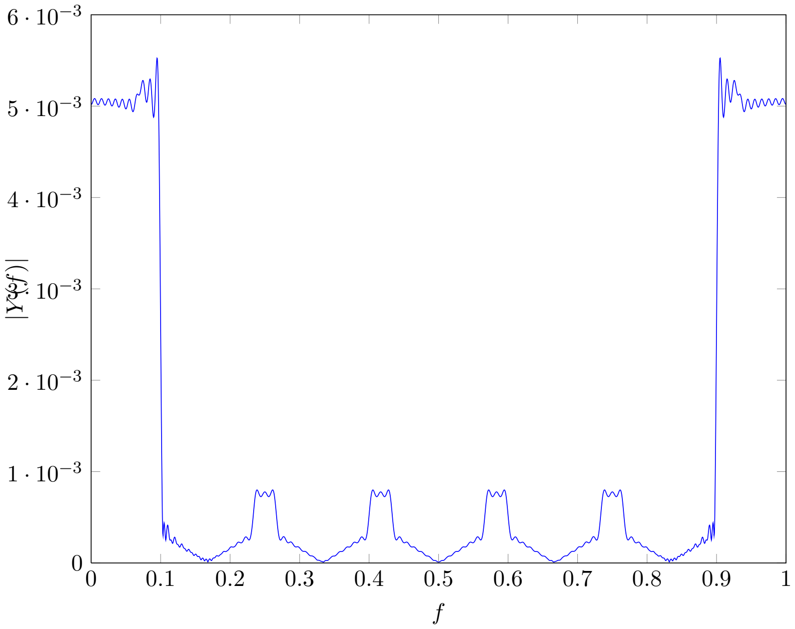

Figure 3.11: Spectrum of the sampled signal $y[n]$, $\tau = [0\ 0.9\ 0.9\ 0.9\ 0.9\ 0.9]$

Thus far we discussed the subclass of band-limited signals which consists of spectrally "full" signals. We see that aliasing for this type of signals occurs in the band $w_c < |w| \leq \pi$ and the algorithm described in Chapters 5 and 6 can be used to estimate the unknown timing skews. We can also use passband signals with aliasing in other band, i.e. the Fourier transform is non-zero for $\frac{\pi}{2} \leq |w| \leq \frac{2\pi}{3}$. If we use this kind of signals for the case with $M = 2$ converters, then aliasing in spectrum for $y[n]$ appears in the band $\frac{\pi}{3} \leq |w| < \frac{\pi}{2}$. The calibration algorithm described in this report can be modified to handle such a signals as long as the input signal band is known the the system. In this report we discuss only the first subclass of signals.

3.2 Input signals generated as band-limited white Gaussian noise

To justify why we use band-limited white Gaussian noise as input signal, we will see how signals used in practical systems such as speech, music and pictures can be modulated in order to improve system efficiency. These signals are highly correlated. In this section we describe how removal of the input signal redundancy affect distortions[2] introduced by quantization.

Assuming the original signal is redundant, there are a lot of techniques which allows us to reduce range of the signal measured as difference between the biggest and smallest values. The

[2] or Signal-to-Quantization noise Ration (SQNR), see Chapter 7

redundancy can be reduced for example by sending the difference between two neighboring values instead of sending signal values directly. As was mentioned in Chapter 2, the selection of the dynamic range for quantizer (which depends on signal range) results in a trade-off between quantizer resolution and signal clipping.

We will illustrate how this technique works in the next example. In our example we are going to use an audio signal. Audio quality speech has a bandwidth of $8\,kHz$ and is thus, according to the Nyquist criteria, sampled at $16\,kHz$. A 16 bit A/D converter is used, thus resulting in a bitrate of $16 * 16 = 256\,kbits/sec$. A resolution of $16\,bits$ corresponds to 65536 different quantization representation levels (the corresponding range is $(32678, 32767)$). Assume we manage to find a representation of the input signal with a range two times smaller than the range for the original signal, this corresponds to reducing quantization noise by four (see Chapter 2). On the other hand, we can reduce bitrate while keeping quantization noise the same.

There are a lot of techniques allowing us to reduce range for redundant signals. Some of them quantize each successive signal sample separately and then transmit the difference between two quantized values obtained. It is also possible to derive the difference of two successive signal samples and then transmit the quantized value of this difference. In these report we describe a more advanced type of modulation of the input signal - DPCM[3]) based on linear prediction. DPCM builds upon the above discussed "difference" principle, but instead of using the difference signal directly, a linear predictor is used. Also, the difference signal is quantized and hence DPCM is a lossy type of modulation. Before discussing the main principles of DPCM, we will take a look on linear prediction.

3.2.1 Linear Prediction

A Pth order linear prediction is basically a FIR-filter structure with P unit delay elements and P weights described by [4]:

$$\hat{x} = \sum_{k=1}^{P} a_k x[n-k] \tag{3.18}$$

Where x is the input signal. We want to make the estimation error $e(n) = x(n) - \hat{x}(n)$ as small as possible, and to do so we need to minimize the mean square error with respect to the prediction coefficient a_k. To do so, it is assumed that input signals are correlated and it is a stationary process.

For a first order linear prediction estimator, the resulting equation yields [4]:

[3] Differential Pulse Code Modulation

$$\gamma_{ee}(0) = E[e^2(n)] = (1 + a^2)\gamma_{xx}(0) - 2a\gamma_{xx}(1) \tag{3.19}$$

where γ_{ee} and γ_{xx} represent autocorrelation coefficients for x and e respectively. Equation 3.19, then minimized with respect to a, results in:

$$a = \frac{\gamma_{xx}(1)}{\gamma_{xx}(0)} \tag{3.20}$$

Thus we have an expression for the coefficient a which minimise the estimation error.

3.2.2 Differential Pulse Code Modulation (DPCM)

In an ideal setting, where the bandwidth (and energy) is unlimited and the channel noise is not an issue, we could have transmitted the prediction error sequence directly and used a decoder on the receiver side to regain the original signal. But since we don't have unlimited energy or bandwidth, and the goal of DPCM is to lower the bit-rate needed for transmission, we need to quantize the prediction error sequence.

The quantizer will introduce noise to the prediction error sequence e. With a reasonable broadband signal, and small quantization steps, it's safe to assume that this noise, q, will be close to additive white noise. Thus the prediction error sequence after quantization e_q is:

$$e_q(n) = e(n) + q(n) \tag{3.21}$$

This results in a lowered SNR^4 at the decoder output due to the introduction of noise in the system.

A smart way to reduce the noise caused by quantization is to not quantize and transmit the prediction error sequence directly, but rather "encode-quantize-and-decode" the signal in the encoder as shown in figure 3.12. This enables us to lower the distortion the quantizer introduces before transmission, and if we subtract the reconstructed signal from the original signal, and use this as a new input for the encoder, we have efficiently reduced the noise generated by the quantization. If we denote the reconstructed signal in the encoder as x_r, and $\hat{x}$ as $a \cdot x_r$, where a is the prediction coefficient of a first order linear predictor, the encoder can be described as following:

$$e_q = e + q = x - a \cdot x_r + q \tag{3.22}$$

[4] Signal to Noise Ratio

DPCM utilizes a linear predictor along with a quantizer in its encoder, in a method called analysis-by-synthesis, to encode and quantize the prediction error sequence. This method will reduce the noise power, and thus reduce the loss of SNR at the decoder output [?].

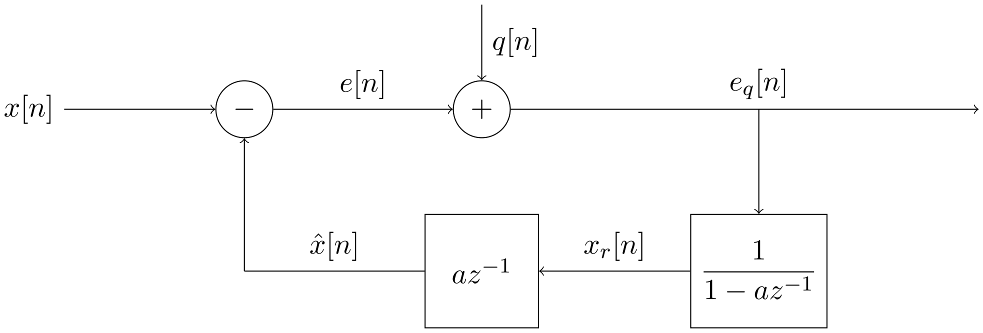

Figure 3.12: Analysis by synthesis encoder structure

Applying DPCM for the input signals gives a stationary process (white Gaussian noise with zero mean and variance σ^2 limited to the bandwidth Ω_c (by using filtering operation). It means that input signal for our system can be modelled as band-limited white Gaussian noise (such a signal in time- and frequency domain represented by figures 3.13 and 3.14). The Matlab-function for generating band-limited white Gaussian noise is $x = gensignalasnoise(tau, l, time, w_c)$ with the same parameters as for generating signal as sinc-function (see section 3.1).

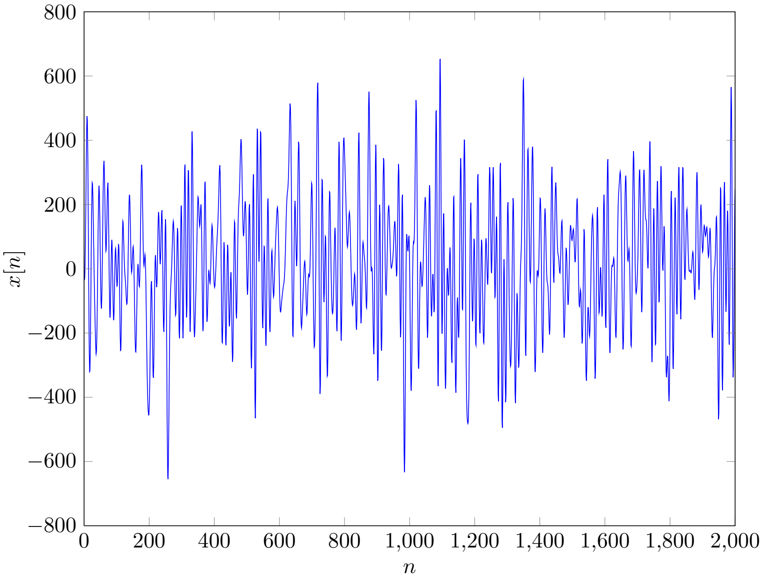

Figure 3.13: Sampled signal $y[n]$ generated as filtered white Gaussian noise

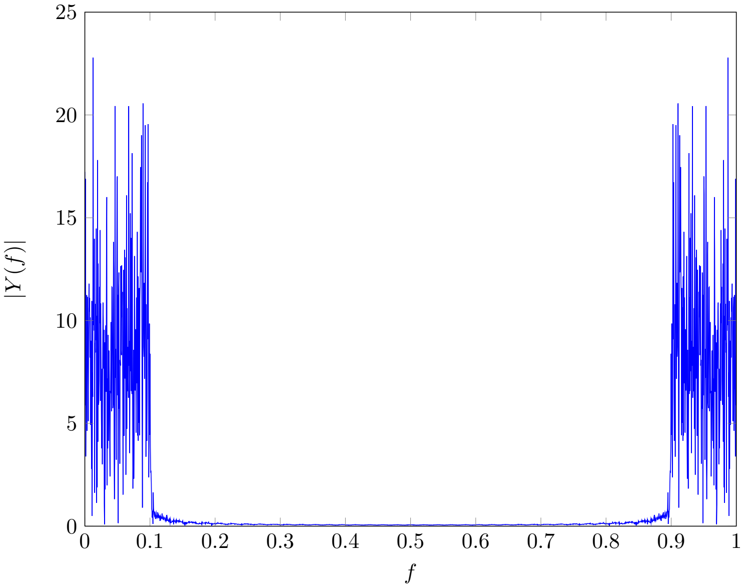

Figure 3.14: Spectrum for the sampled signal $y[n]$ generated as as filtered white Gaussian noise

As we can see from figure 3.14, the spectrum for the signal generated as filtered white Gaussian noise has some components above the normalized frequency $f = 0.1$. This type of input signal will be used to evaluate performance of the reconstruction filter-bank in Chapter 4, for time-skew estimation in Chapters 5 and 6, and of the whole system in Chapter 7.

4 Reconstruction of band-limited signals from recurrent non-uniform samples using continuous-time filter-bank

In this chapter we present an efficient method for reconstructing a signal from recurrent non-uniform samples. First of all, the development of continuous-time filter-bank is presented. As examples, filter-banks for two systems (with two and three branches) are presented here. Then we evaluate performance for a reconstruction filter-bank using a sinc-function as input signal. Also the influence of window functions on filter-bank design is discussed in this chapter.

Recurrent non-uniform sampling is defined as a combination of M sequences of uniform samples taken at $1/M$ of the Nyquist rate [9]. We model the output of the ith constituent A/D converter (see figure 2.1) as:

$$y_i[n] = x(nMT_s + iT_s + \tau_i) + \omega_i[n] \tag{4.1}$$

where τ_i model the corresponding to the ith branch time-skew and $\omega_i[n]$ represents the aggregate noise. As was mentioned in section 2.1, we assume the quantizer to be uniform, and signal values distributed uniformly over the given range. We know from [6] that the sum of many independent random variables are Gaussian distributed, therefore we model $\omega_i[n]$ as white Gaussian noise whose variance depends on the number of bits to which the input signal is quantized. Without loss of generality, for this chapter we assume high resolution for A/D converters (or $\omega_i[n] \approx 0$) for ease of analysis. The effects of quantisation noise are considered in Chapter 7. The sampling distribution for the case with two branches is illustrated by figure 4.1:

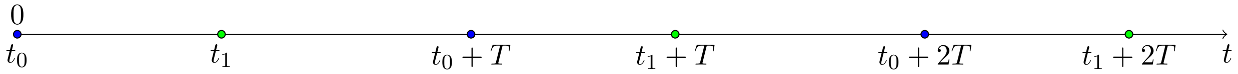

Figure 4.1: Sampling distribution for $M = 2$

Each period consists of M non-uniformly distributed sampling points. We denote the points of one period by t_i, $i = 0, 1, \dots M-1$ (or using notation for $y_i[n]$, we can express $i = n(modM)$), then the complete set of sampling points is:

$$t_i + nT, \quad i = 0,\ 1,\ \ldots\ M-1; \quad n \in (-\infty, \infty) \tag{4.2}$$

It is well established that a continuous-time signal $x(t)$ can be reconstructed from its samples at a set of samplings times t_n if the average sampling period is smaller than the Nyquist period. The average sampling period is defined as $\lim_{x\to\infty} \frac{t_n}{n}$. For recurrent non-uniform distribution the following theorem is valid [1]

Theorem 1 (Yao and Thomas Theorem)**.** *Let $x(t)$ be a finite energy band-limited signal such that $X(\omega) = 0$ for $|\omega| > W - \epsilon$ for some $0 < \epsilon < W$. Then $x(t)$ is uniquely determined by its samples $x(t_n)$ if*

$$\left| t_n - n\frac{\pi}{W} \right| < L < \infty$$
$$|t_n - t_k| > \delta > 0, \qquad n \neq k$$

The reconstruction is given by

$$x(t) = \sum_{n=-\infty}^{\infty} x_n \frac{G(t)}{G'_n(t - t_n)} \tag{4.3}$$

where

$$G(t) = (t - t_0) \prod_{\substack{n=-\infty \\ n\neq 0}}^{\infty} (1 - \frac{t}{t_n}) \tag{4.4}$$

$G'(t_n)$ is the derivative of $G(t)$ evaluated at $t = t_n$, and if $t_n = 0$ for some n, then $t_0 = 0$.

Based on the Theorem 4, we can therefore reconstruct a continuous-time signal $x(t)$ where the sampling times are given by 4.2. In particular, substituting 4.2 in 4.3 and 4.4, we obtain the following reconstruction formula for the system with M A/D converters [1]:

$$x(t) = \gamma(t) \sum_{n=-\infty}^{\infty} \sum_{i=0}^{M-1} y[nM + i] \frac{a_i(-1)^{nM}}{\pi(t - nT - \tau'_i)/T} \tag{4.5}$$

where

$$a_i = \frac{1}{\prod\limits_{\substack{k=0 \\ k\neq i}}^{M-1} \sin(\pi(\tau_i' - \tau_k')/T)} \tag{4.6}$$

$$\gamma(t) = \prod_{k=0}^{M-1} \sin(\pi(t - \tau_k')/T) \tag{4.7}$$

$$T = MT_s \tag{4.8}$$

$$\tau_k' = kT_s + \tau[k] \tag{4.9}$$

Because of direct implementation of the reconstruction given by 4.5 is computationally difficult, a continuous-time filter-bank implementation of 4.5 can be used. To do this we interchange the order of summation in 4.5 and denote the inner sum by $f_i(t)$ [1], i.e. ,

$$\begin{aligned} f_i(t) &= \sum_{n=-\infty}^{\infty} x(nT + t_i) \frac{a_i(-1)^{nM} \prod\limits_{k=0}^{M-1} \sin(\pi(t - t_k)/T)}{\pi(T - nT - t_i)/T} &(4.10) \\ &= s_i(t) * h_i(t)^1 &(4.11) \end{aligned}$$

The s_i here represent sub-sequences corresponding to the $i-$branch (see equation 4.12), i.e. shifted by τ_i impulse train of samples. In the last equation we have used the relation $\sin(t - n\pi) = (-1)^n \sin(t)$.

$$s_i(t) = \sum_{n=-\infty}^{\infty} x(nT + \tau_i')\delta(t - nT - \tau_i'). \tag{4.12}$$

Using the definition of f_i, we can rewrite $x(t)$ as a sum of M convolutions:

$$x(t) = \sum_{i=0}^{M-1} s_i(t) * h_i(t) \tag{4.13}$$

It means that each of the sub-sequences corresponds to samples at $\frac{1}{M}$ of the Nyquist rate of a time-shifted version of $x(t)$. Therefore, the output of the filter-bank is an aliased and filtered version of $x(t)$. Filter impulse response h_i for every branch can be expressed as:

$$h_i(t) = a_i \frac{\prod\limits_{k=0}^{M-1} \sin(\pi(t + \tau_i' - \tau_k')/T)}{\pi t/T} \tag{4.14}$$

where

$$a_i = \frac{1}{\prod\limits_{\substack{k=0\\k\neq i}}^{M-1} \sin(\pi(\tau'_i - \tau'_k)/T)} \tag{4.15}$$

As we can see from 4.14, the expression for filter impulse response of each filter of the filter-bank includes the same term: $\frac{\sin(\pi t/T)}{\pi t/T}$ at the point where $\tau'_i = \tau'_k$, which corresponds to an ideal low-pass filter with cut-off frequency W/M. All the other combinations of $\tau'_i - \tau'_k$ will compensate aliasing between sequences at the output of the filter-bank. In the next section we will show by examples how the filter-banks for system with two and three branches can be designed.

4.1 Implementation of the continuous-time filter-bank for the signal reconstructing

In this section we demonstrate by examples how the reconstruction filter-bank can be designed for systems with two and three brunches.

4.1.1 Filter-bank for systems with two branches

For the case with two branches (odd end even) we have:

$$h_{odd} = a_{odd} \frac{\sin(\pi t/T)}{\pi t/T} \sin(\pi(t - (t_{odd} - t_{even}))/T) \tag{4.16}$$

$$h_{even} = a_{even} \frac{\sin(\pi t/T)}{\pi t/T} \sin(\pi(t + (t_{odd} - t_{even}))/T) \tag{4.17}$$

where

$$a_{odd} = \frac{1}{\sin(\pi(t_{odd} - t_{even})/T)} \tag{4.18}$$

$$a_{even} = \frac{1}{\sin(\pi(t_{even} - t_{odd})/T)} \tag{4.19}$$

Without loss of generality we assume $T = 1$. If we deal with uniform samples (or equivalently the same time offset between all the samples) the reconstruction filter-bank has the same filter in each branch (figure 4.2). But in the case of non-uniform samples, the filter-bank is constructed with respect to a delay introduced by the even branch as illustrated by figure 4.3

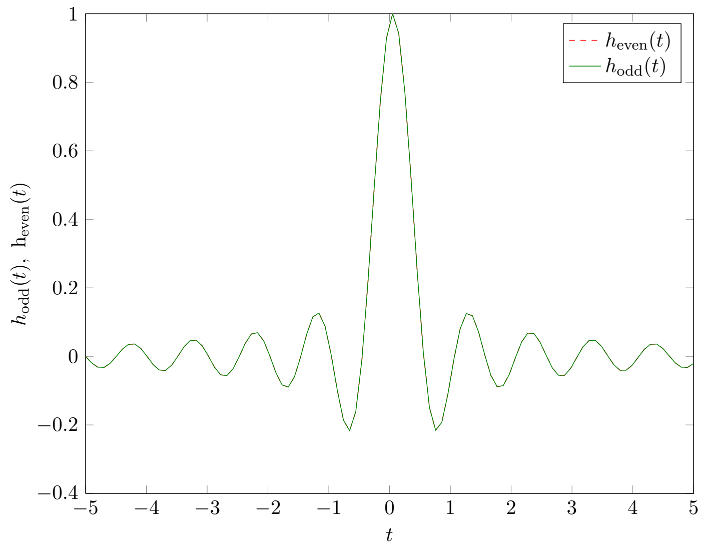

Figure 4.2: Representation of the filter impulse response for the system with 2 branches. ($\tau = [0\,0]T_s$)

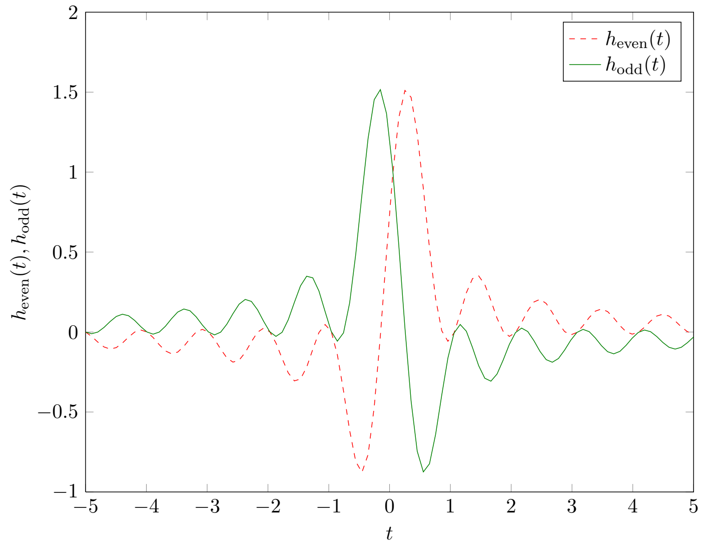

Figure 4.3: Representation of the filter impulse response for the system with 2 branches. ($\tau = [0\,0.3]T_s$)

From figure 4.3 we see that filter impulse responses for both filters equal to 1 at the point where $t = 0$, but at the points $t = \pm 1,\ 2, \ldots\ K$ (as was mentioned above, $T = 1$) the filter impulse responses are not equal to 0 as is for the case with uniform sampling (see figure 4.2). In section 4.2 we will illustrate how the filter-bank compensates time-skews by examples.

4.1.2 Filter-bank for systems with three branches

Now we have $T = 3T_s$ which means we have three samples at $\frac{1}{3}th$ of the Nyquist rate in one period. This is represented by figure 4.4:

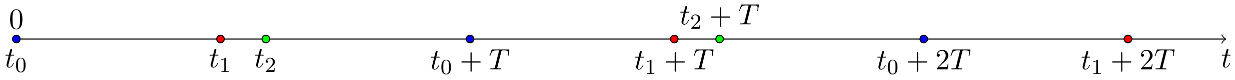

Figure 4.4: Sampling distribution for $M = 3$

For three-path implementation we can rewrite the equations in 4.14 as:

$$h_0 = a_0 h_{com} \sin(\pi(t + t_0 - t_1)/T) \sin(\pi(t + t_0 - t_2)/T); \quad (4.20)$$

$$h_1 = a_1 h_{com} \sin(\pi(t + t_1 - t_0)/T) \sin(\pi(t + t_1 - t_2)/T); \quad (4.21)$$

$$h_2 = a_2 h_{com} \sin(\pi(t + t_2 - t_0)/T) \sin(\pi(t + t_2 - t_1)/T) \quad (4.22)$$

$$a_0 = \frac{1}{\sin(\pi(t_0 - t_1)/T) \sin(\pi(t_0 - t_2)/T)}; \quad (4.23)$$

$$a_1 = \frac{1}{\sin(\pi(t_1 - t_0)/T) \sin(\pi(t_1 - t_2)/T)}; \quad (4.24)$$

$$a_2 = \frac{1}{\sin(\pi(t_2 - t_0)/T) \sin(\pi(t_2 - t_1)/T)} \quad (4.25)$$

The examples of the filter-banks for systems with three branches for uniform and non-uniform sampling are represented below:

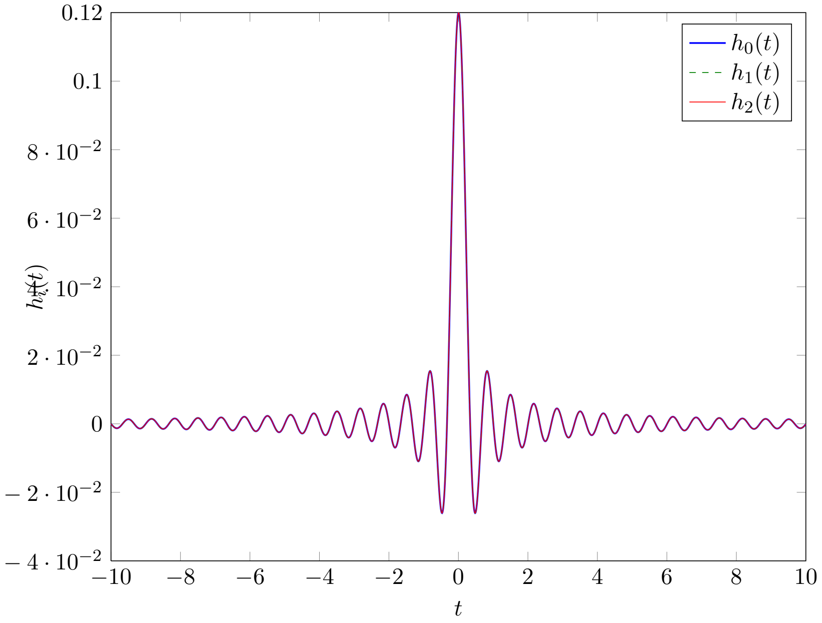

Figure 4.5: Representation of the filter impulse response for the system with 3 branches. ($\tau = [0\,0\,0]T_s$)

If we deal with uniform sampling ($\tau = 0$), the filter bank have the same filters for each branch as shown by figure 4.5.

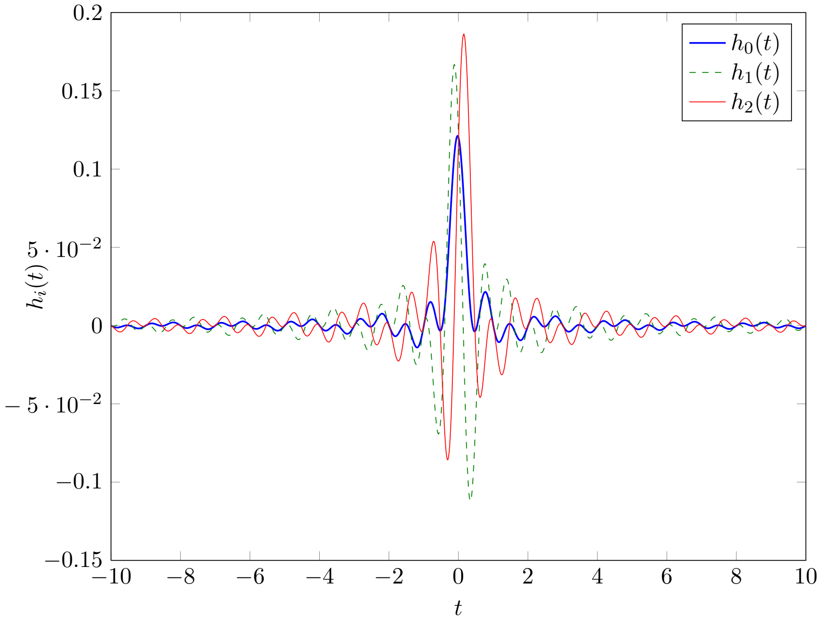

Figure 4.6: Representation of the filter impulse response for the system with 3 branches. ($\tau = [0\,0.2\ -0.4]T_s$)

As we can see from figure 4.6, if we deal with non-uniform sampling, filters are not the same for each branch because time-skews have to be compensated. It is not obvious from figure 4.6 how the filter-bank compensates time-skews, but we will demonstrate it with simulation results in the next section.

4.2 Evaluating performance of reconstruction using continuous-time filter-bank

To measure distortions introduced by reconstruction the filter-bank, we assume that time-offsets are estimated accurately (i.e. for our simulations we simply take the same value of τ for filter-bank design as for signal design). Results will be presented for systems with two and three branches.

Distortions D are defined as the difference between the output samples and the uniformly sampled signal, normalized by signal effect, i.e.:

$$D = \frac{\sum\limits_{n} |\hat{x}[n] - x[n]|^2}{\sum\limits_{n} |x[n]|^2} \tag{4.26}$$

Before beginning the simulations, we have to define the length of the filters in the reconstruction filter-bank K_{rec}. We define the filter length as $(2K_{rec}+1)T$ for $K_{rec} = 1,\ 2,\ 3, \ldots$ measured in seconds. There is always a trade-off between the delay introduced by the filter and distortions: the smaller the length of the filter, the smaller the delay, but distortions grow if we reduce length of the filter. In order to show how filter length influence reconstruction signals, we simulate signal reconstruction with values for the K_{rec} in the range $K_{rec} \in (3, 20)T$ while keeping all the other parameters for the system the same, i.e. $l = 200$, $time = 200$, $w_c = 0.2\pi$ and $\tau = [0\,0.2]T_s$. Figure 4.7 shows how distortions depend on K_{rec}. The K_{rec} first samples and K_{rec} last samples of the reconstructed signal are distorted because of the filtering operation. As we know from figures 4.2 and 4.3, the magnitude of h_i grows with magnitude of τ. It means that distortions introduced by the reconstruction filter grow with magnitude of τ. Since for now we are not interesting in distortions introduced by filtering operations in general, we reduce the length of the $x[n]$ and $\hat{x}[n]$ by K_{rec} samples for both sides in order to measure distortions.

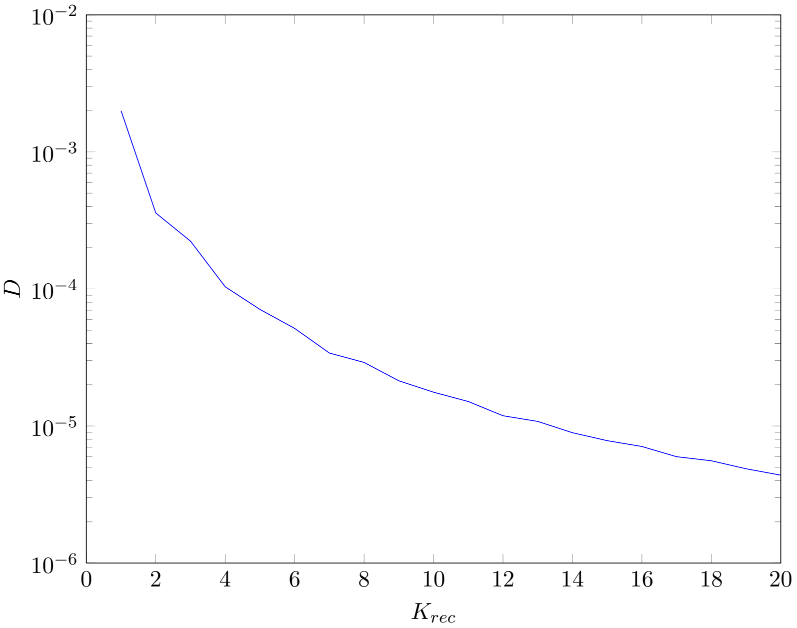

Figure 4.7: Distortions introduced by the reconstruction filter-bank for an input signal generated as the sinc-function depending of the filter-length K_{rec}

As we can see from figure 4.7, the distortions reduce with increasing K_{rec} Since we have no information about the value of allowable distortions for the system, the $K_{rec} = 10$ may be a reasonable trade-off between the distortions and delay, introduced by the reconstruction filter, for our simulations.

4.2.1 System with two branches

For the case with two branches we denoted the second term (the first one is always zero) of τ as the offset between uniform samples relative to the first branch and real (i.e. including time-offset introducing by system) ones. Samplings period in each branch T is two times bigger than overall samplings interval T_s ($T = 2T_s$). That means that time-offset can not be greater than sampling interval or $\tau_1 < |T_s|$. For our simulations we change the second value of the time-skew vector in the range $\tau_1 \in (-0.98T_s, 0.98T_s)$ with step size $\tau_s = 0.02T_s$. Values for other parameters for the input signal are the same as in section 5.3, i.e signal length $l = 200$, $time = 200$, $w_c = 0.2\pi$. The figure below represents how distortions, introduced by reconstruction filter-bank, depend on time-skews.

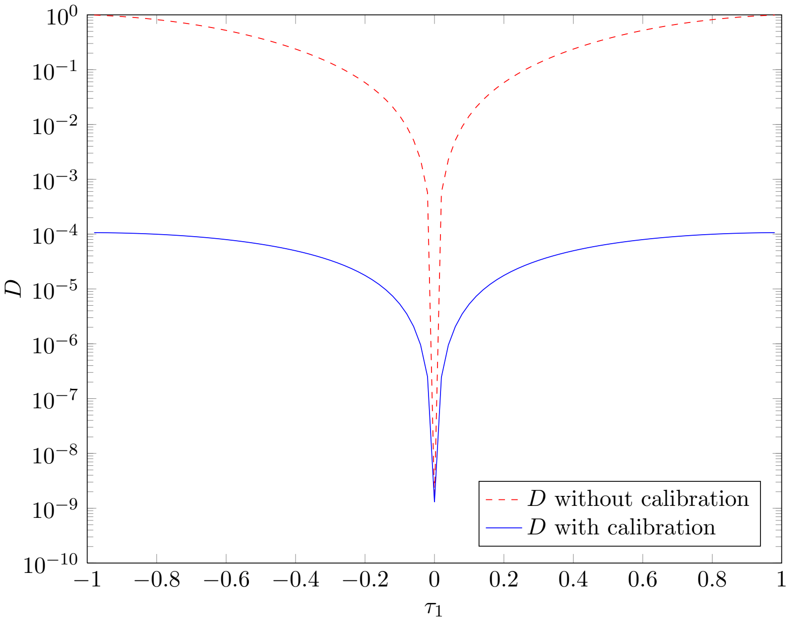

Figure 4.8: Distortions introduced by reconstruction filter-bank for input signal generated as sinc-function.

From figure 4.8 we see that reconstruction is more accurate if the information about time-skews is used by the filter-bank. We see also that distortions grows with magnitude of time-skew, but after the $|\tau_1| = 0.4T_s$, distortions remain almost the same. This may be due to the inaccuracies in filter-bank implementation and can be improved by using windowing functions as discussed in section 4.2.1.

Designing the reconstruction filter-bank using windows

In general, the unit sample response for each filter of the reconstruction filter-bank obtained from equation 4.14 is infinite in duration. In the last section we simply truncated the input sample response h_i to the length $(2K_{rec} + 1)T$ measured in seconds (in our examples K_{rec} was chosen to be equal to 10). The edge effects were neglected. In this section we consider the effect of using window functions for the filter design. We evaluate distortions introduced by the reconstruction filter-bank for the case then window functions are used for it's design.

The truncation of the h_i in length is equivalent to multiplying h_i by a "rectangular window", defined as

$$w_{rect}[n] = \begin{cases} 1, & n = 0,\ 1,\ 2,\ \dots,\ 2K \\ 0, & \text{otherwise.} \end{cases} \tag{4.27}$$

By using a window function we can design a filter with finite unit sample response (FIR-filter) which can be realized in practise.

There are a lot of window functions which can be used for FIR filter design. From reference [3] (Chapter 10.2.2) we know that using the ”rectangular window” results in a filter with relative small main lobes and the lowest stop-band attenuation in the frequency domain. In this report we do not consider the frequency domain, but we know that the window functions that do not contain abrupt discontinuities in their time-domain characteristics gives better results. The choice of window function for reconstruction filter-bank depends on system specification. In this report we illustrate the effect of windowing by using the Blackman window function as example which is defined as (references [3], Chapter 10):

$$w_{Blackman}[n] = \begin{cases} 0.54 - 0.46\cos\frac{2\pi n}{2K}, & n = 0,\ 1,\ \ldots,\ 2K \\ 0, & \text{otherwise.} \end{cases} \tag{4.28}$$

The shape of Blackman window function for $K = 10$ is shown in figure 4.9.

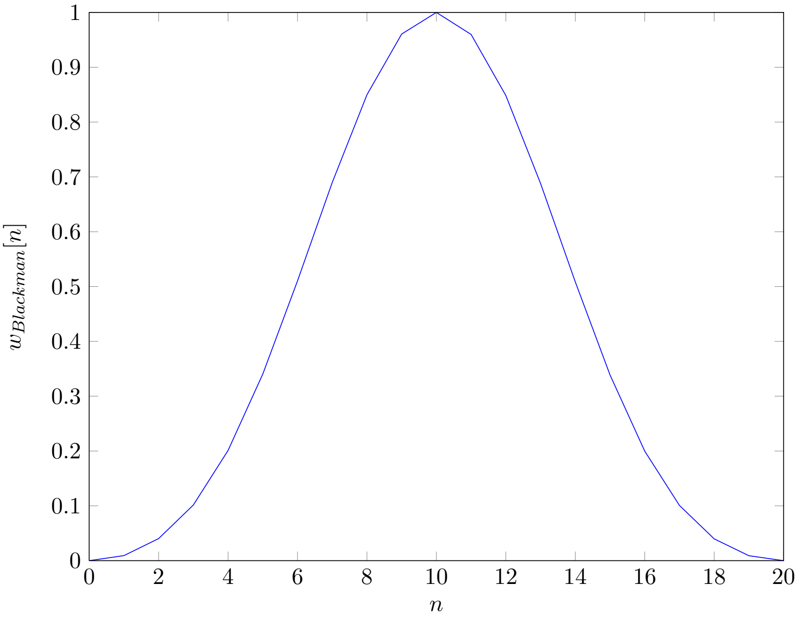

Figure 4.9: Shape of the Blackman window function.

Multiplying h_i by $w_{Blackman}$ result in a new unit sample response for reconstruction filter-bank. The unit sample responses for the reconstruction filter-bank for the case with two branches are illustrated in figures 4.10.

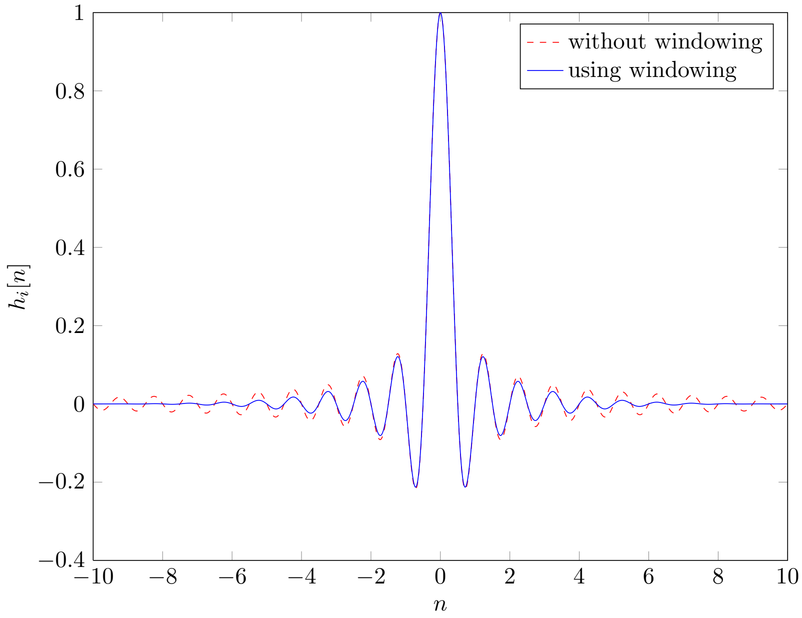

Figure 4.10: Representation of the filter impulse response for the filter-bank. ($\tau = [0\ 0]T_s$, $K_{rec} = 10$)

From figure 4.10 we see that for the case with Blackman window used for filter design, the filter impulse response converges to zero more rapidly with time than for the case with "rectangular window". It means that we can either use the same filter length K_{rec} and reduce the distortions or we can reduce the length of the filter. This trade-off is shown in figure 4.11.

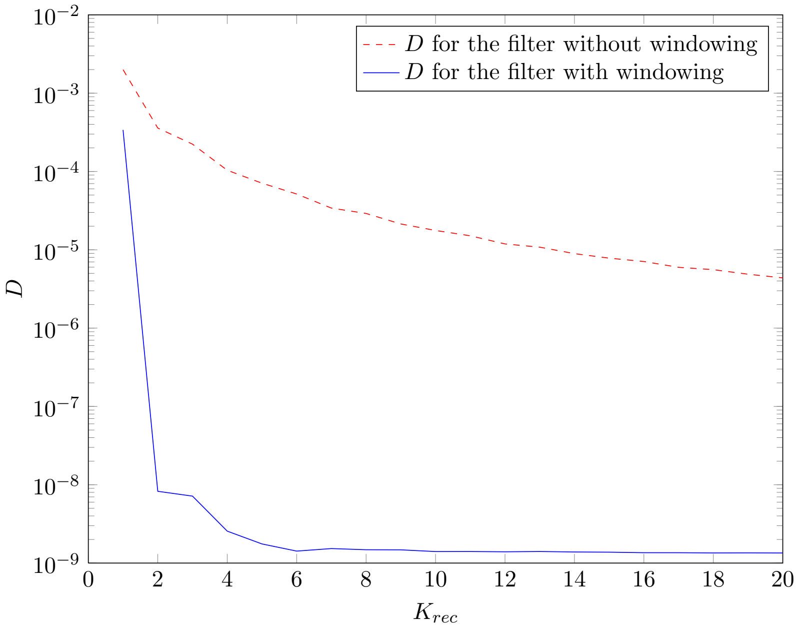

Figure 4.11: Distortions introduced by the reconstruction filter-bank depending of the filter-length K_{rec}

From figure 4.11 we see that using window functions for the filter design gives us better results. It means we can expect that distortions introduced by this filter (the one which is designed using Blackman window) are reduced. To demonstrate the advantage of using Blackman window function (and window function in general) for filter-bank design, we will see how distortions depend on the value of τ for two filter-banks as was done in section 4.2.1. As input signal the sinc-function with the same parameters as in section 4.2.1 is used. The difference between distortions introduced by systems with a filter-bank designed with and without using Blackman window is shown in figure below.

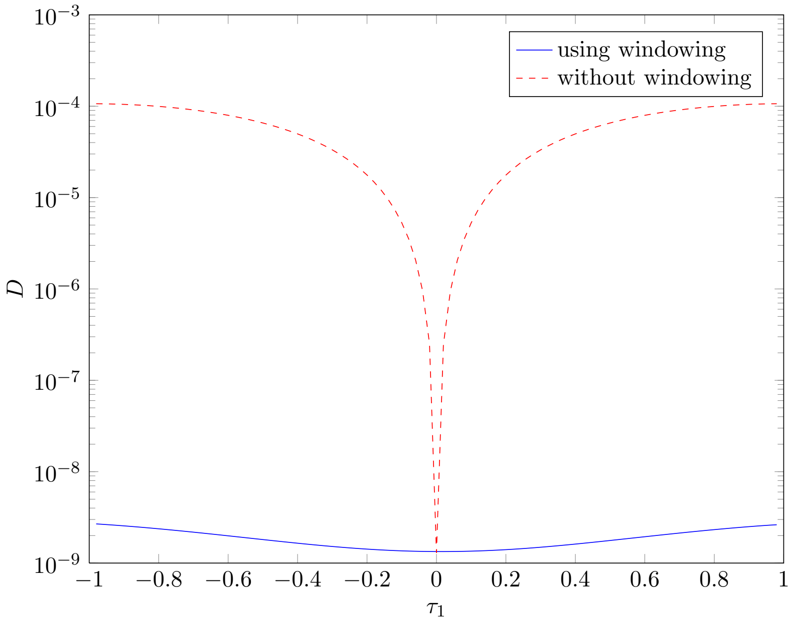

Figure 4.12: Distortions introduced by reconstruction filter-banks designed with and without using Blackman window function for the system with $M = 2$.

From figure 4.12 we see that by using Blackman window for filter-bank design we have reduced distortions almost to the same value as for the case with uniform sampling. This is due to the fact that windowing reduces the length of non-zero values of the filter impulse response. We conclude that using window functions improve the system performance, and we will use the Blackman window function in our simulations.

System with three branches

For the case with three branches we denoted the last two terms (the first one is always zero) of τ as the offsets between uniform samples relative to the first branch and real (i.e. including time-offset introducing by system) ones. Sampling period for each branch T is three times bigger than overall saplings interval T_s ($T = 3T_s$). That means that time-offset can not be greater than the sampling interval or $\tau_i < T_s$ for $i = 1,\ 2$. We have to take into account that $\tau_1 - \tau_2 < T_s$. If we do not make this restrictions, then samples at times $t_1 + T$ and $t_2 + T$ may be swapped. It violates the system restrictions because for such a case the samples can not be defined as recurrent non-uniform samples. So, for our simulations we change last two values of time-skew vector: $\tau_1 \in (-0.9T_s, 0.45T_s)$, $\tau_2 \in (-0.45T_s, 0.9T_s)$ with step size $\tau_s = 0.15T_s$. The other parameters are the same as for the case with two branches. Distortions, introduced by the reconstruction filter-bank depending on time-skews are represented in figure 4.13.

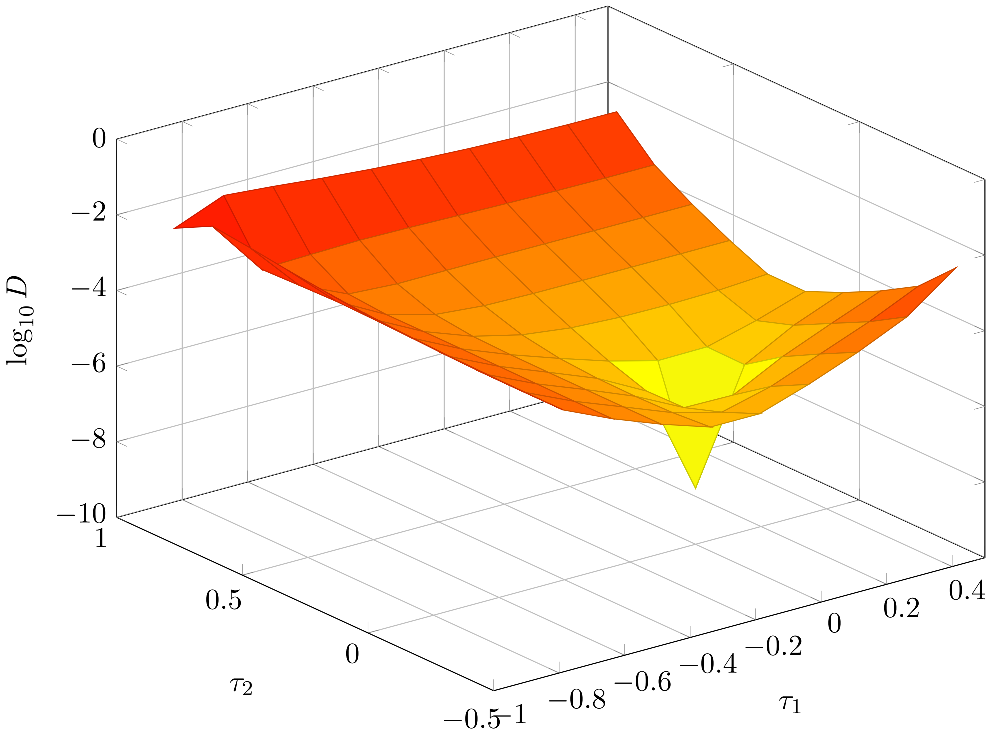

Figure 4.13: Distortions introduced by reconstruction filter-bank for the system with three branches.

As we can see from figure 4.13, the distortions grow with magnitude of τ. The worst cases are the ones where $|\tau_1 - \tau_2|$ is highest or where we have τ large in magnitude.

As we can see, the filter-bank for system with $M = 3$ A/D converters introduces more distortions than the system with $M = 2$ for the case with non-uniform sampling. It is not obvious how distortions depends on M. Results for simulations for the systems with $M = 2$, 4 are represented in Chapter 7.

In this section we evaluated performance for the reconstruction filter-bank assuming ideal quantization of the system, or equivalently quantization noise $w_i \approx 0$. In Chapter 7 we will see how reconstruction of the non-uniformly sampled signal depends on the number of bits using for quantization.

5 Estimation of time-skew and gain using Least Square method

In this section we represent a least squares (LS) method for estimating at first the unknown time-skews and then gains for each branch of the system.

The input signal $x(t)$ is modelled as a band-limited signal with cut-off frequency Ω_c, i.e., the continuous time Fourier transform $X(j\Omega) = 0$ for $\Omega_c <\mid \Omega \mid$. The overall sampling period T_s of the system is chosen to ensure that the sampling rate strictly exceeds the Nyquist frequency, i.e., $T_s < \dfrac{1}{2\Omega_c}$. Each converter operates with sampling period $T = MT_s$. Thus, we can rewrite the continuous signal $x(t)$ in term of its samplings as:

$$x(t) = \sum_m x[m] sinc(t - mT_s) \tag{5.1}$$

where the *sinc* terms are defined with period T_s.

The received signal $y[n]$ obtained by multiplexing the A/D converters outputs $y_i[n]$ (see 4.1):

$$\begin{aligned} y[n] &= y_i \left[\frac{n-i}{M} \right]; \qquad \text{where } n(mod\, M) = i. && (5.2) \\ &= x(nT_s + \tau_i) + \omega[n] && (5.3) \\ &= \sum_m x[m] sinc((n-m)T_s + \tau_i) + \omega[n] && (5.4) \end{aligned}$$

The goal of the signal recovery problem is to estimate $x[n] = x(nT_s)$ as accurately as possible. To do it, we develop an algorithm for estimating the unknown parameter τ.

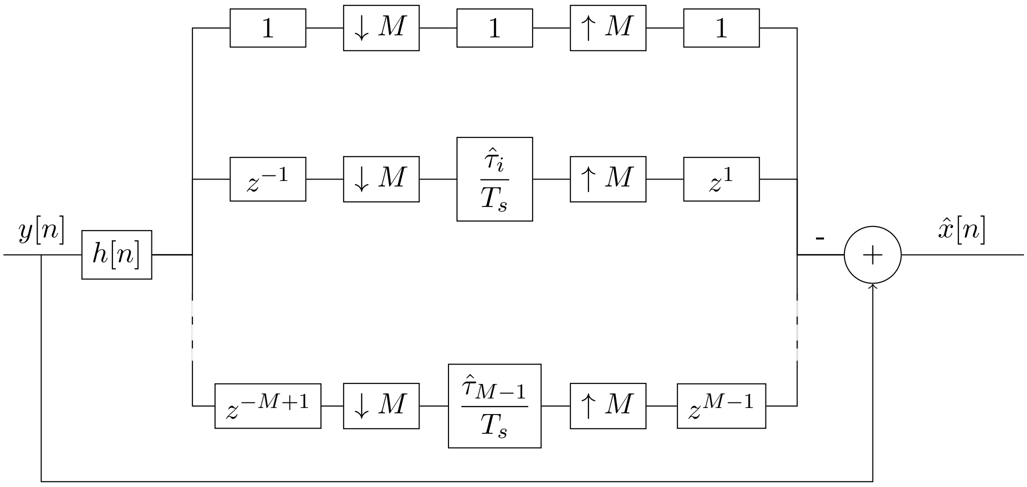

Figure 5.1: Filter-bank for reconstructing a signal using time-skew estimates $\hat{\tau}$ for a small mismatch regime.

If we approximate the reconstruction expression given by 4.5 (assuming a small mismatch regime) by using the first order Taylor series of the function around the point $\tau = 0$ [1] it can be reduced at times $t = nT$ to:

$$\hat{x}[n] \approx y[n] - \frac{\tau_i}{T_s}(b * y)[n],\ n(modM) = i^{\,2} \tag{5.5}$$

and

$$b[n] = \begin{cases} 0, & n = 0 \\ \dfrac{(-1)^n}{n}, & \text{otherwise.} \end{cases} \tag{5.6}$$

is a discrete-time filter implementing band-limited differentiation. A derivation of 5.5 is given by reference [2] (Appendix I). Using 5.5 we can also derive the approximation equation for the received signal:

$$y[n] \approx x[n] + \frac{\tau_i}{T_s}(b * y)[n],\ n(modM) = i \tag{5.7}$$

Figure 5.1 represents the filter-bank implementation of the reconstruction approximation given by 5.5. The method for signal reconstruction from recurrent non-uniform samples was described in more detail in Chapter 4.

For convenience we rewrite our reconstruction equation in matrix representation.

[1] the τ here is a vector containing unknown time-skews

[2] $*$ represents the convolution operation here

$$
\begin{aligned}
y &= \mathbf{F}(\tau)x && (5.8)\\
&= x + \sum_{i=1}^{M-1} \frac{\tau_i}{T_s}\mathbf{D}_i\mathbf{B}x && (5.9)\\
&= (\mathbf{I} + \mathbf{TB})x && (5.10)
\end{aligned}
$$

Toeplitz matrix $\mathbf{B}$ represents a discrete time filter implementing bandlimited differentiation ($\mathbf{B}_{k,l} = b[k-l]$).

Matrix $\mathbf{D}_i$ of size $N \times N$ selects the entries from the ith branch.

$$
[\mathbf{D}_i]_{k,l} = \begin{cases} 1, & k = l,\ i = k(modM) \\ 0, & \text{otherwise.} \end{cases} \qquad (5.11)
$$

Matrix $\mathbf{T}$ is a diagonal matrix containing the unknown time-skews

$$
\mathbf{T}_{k,l} = \begin{cases} \frac{\tau_i}{T_s}, & k = l,\ i = k(modM) \\ 0, & \text{otherwise.} \end{cases} \qquad (5.12)
$$

5.1 Estimation of the time-skew parameter

Time-skew τ in our system is an unknown but constant parameter equal to time mismatch between the first and the ith converters ($i = 0, 1, \dots M-1$), i.e. $\tau_i = t - iTs$. Because there is no time difference for the first converter, the first element in the vector τ is always equal to zero, so we can simply discard the first element and represent τ as a vector of $M-1$ elements.

We estimate the timing skews and input signal using the method of maximum likelihood, modelling the time-skews as non random and unknown parameters. We assume that the noise introduced by the system is white and Gaussian, so the maximum likelihood estimate of the non-random unknown parameter τ reduces to the least-square problem [5]. With other words, the optimization is formulated as a least-square problem that computes the timing skews $\hat{\tau}$ minimizing out-of-band energy in $\hat{x}$

$$
\begin{aligned}
\hat{\tau} &= \arg\min_{\tau} \parallel (\mathbf{L} - \mathbf{I})(\mathbf{I} - \mathbf{TH})y \parallel^2 && (5.13)\\
&= \arg\min_{\tau} \parallel \gamma - \mathbf{R}\tau \parallel^2 && (5.14)
\end{aligned}
$$

where

$$\mathbf{R} = [\mathbf{r}_1\, \mathbf{r}_2 \ldots \mathbf{r}_{M-1}],\ \mathbf{r}_i = (\mathbf{L} - \mathbf{I})\mathbf{D}_i\mathbf{H}y \tag{5.15}$$

$$\gamma = (\mathbf{L} - \mathbf{I})y \tag{5.16}$$

Matrix $\mathbf{L}$ of size $N \times N$ represents low-pass filter with cut-off frequency ω_c such that $\hat{x} = \mathbf{L}\hat{x}$. (Edge effects of filtering operation are neglected).

The solution to the over-constrained least-squares estimation problem is given by

$$\hat{\tau} = (\mathbf{R}^T\mathbf{R})^{-1}\mathbf{R}^T\gamma \tag{5.17}$$

5.2 Estimation of gain parameter

An approximation equation for the output of the ith A/D converter is modelled as

$$y_i[n] = g_i x(nMT_s + iT_s + \tau_i) \tag{5.18}$$

where the g are unknown gains. It means that non-uniform gain introduced by the system can be compensated by multiplying output from each branch $y_i[n]$ by $\frac{1}{g_i}$. Without loss of generality, we assume $g_0 = 1$, and unknown vector g consists of $M - 1$ elements

$$g = [g_1\ g_2\ \ldots\ g_{M-1}] \tag{5.19}$$

Now we can compute the first order Taylor series approximation around the point $\tau = 0$ and $g = 1$. Vector $\hat{\theta}$ includes $2(M - 1)$ unknown parameters:

$$\hat{\theta} = \arg\min_{\theta} \parallel \gamma - \mathbf{R}\theta \parallel^2 \tag{5.20}$$

where

$$\theta = [\tau\ g']^T \tag{5.21}$$

$$R = [r_1\ r_2\ \ldots\ r_{M-1}\ s_1\ s_2\ \ldots\ s_{M-1}] \tag{5.22}$$

$$r_i = (\mathbf{L} - \mathbf{I})\mathbf{D}_i\mathbf{H}y \tag{5.23}$$

$$s_i = -(\mathbf{L} - \mathbf{I})\mathbf{D}_i y \tag{5.24}$$

$$g' = \left[\frac{1}{g_1}\ \frac{1}{g_2}\ \ldots\ \frac{1}{g_{M-1}}\right] \tag{5.25}$$

5.3 Evaluation of the performance for time-skew estimation

In this section we evaluate the performance of time-skew estimation based on the algorithm which was described in section 5.1. We test the system by using both deterministic signal (sinc-function) and random signal (band-limited white Gaussian noise). Inaccuracies are measured as

$$\xi = \frac{|\tau - \hat{\tau}|^2}{M - 1} \tag{5.26}$$

First we have to choose the length of the low-pass filter L (section 5.1) and the length of the input signal, then we simulate estimation of time-skews for system with two branches using both the sinc-function and the band-limited white Gaussian noise as input signals, and at the end we generalise our results for systems with three branches using the sinc-function as input signal.

5.3.1 Choosing parameters for simulation

In this section we argue for such parameters as length of the signal and length of the low-pass filter L.

Low-pass filter L with cut-off frequency w_c for our system ensures that the signal is band-limited and that aliasing in the spectrum is minimized. The input signal is modelled as a band-limited signal as was described in Chapter 3. It means the system dependence on the the filter length (or on the sharp of the spectrum of the low-pass filter) shouldn't be too sensitive. As example we evaluate the dependence of the time-skew estimation on the length of the filter L. In this case we choose the τ to be $[0\,0.5]T_s$ (for other values of τ we have the same tendency), $l = 200$, $time = 200$ and $w_c = 0.2\pi$. Simulation results are illustrated in figure 5.2:

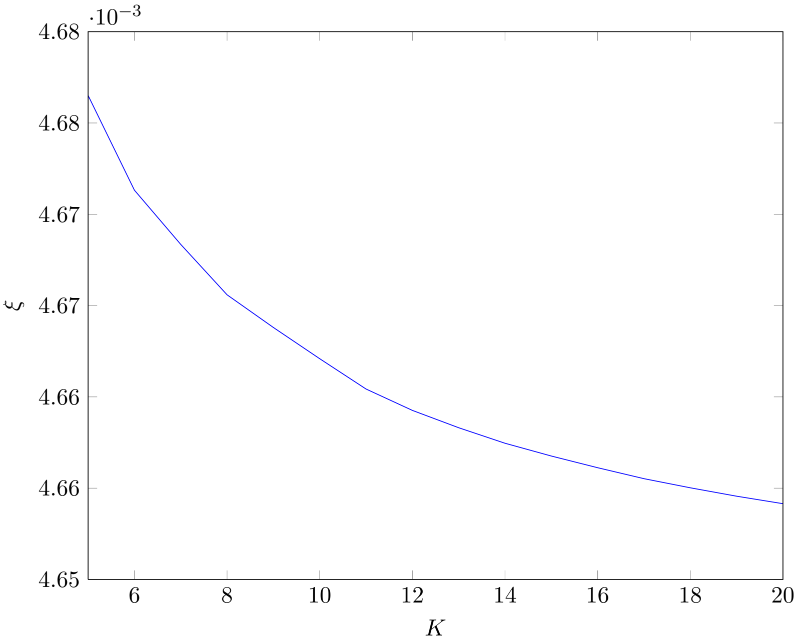

Figure 5.2: Inaccuracies in estimator of τ for different values of K.

From results representing in figure 5.2, we may conclude that filter length more than approximately $K = 10$ does not give significant improvement for estimation. Based on this, we chose $K = 10$ for further measurements.

In order to find an optimal value for l for our simulations we evaluate how accuracies of the estimator depend on the signal length l. To do it, we will change values of l in a range from $(10, 800)$ samples while keeping bandwidth w_c equal to 0.2π radians. We choose the value of $\tau = [0\ 0.1]$, because estimation method described in this chapter performs better for values of time-skew small in magnitude. Results are presented in figure 5.3:

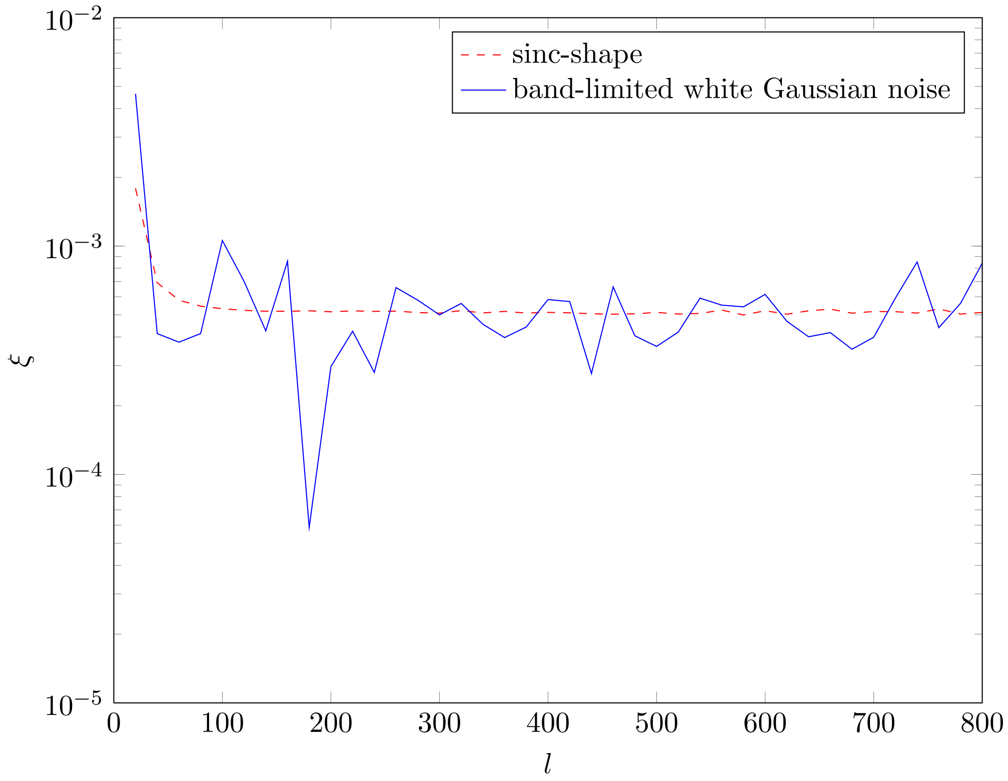

Figure 5.3: Inaccuracies in estimator of τ for different values of l.

As we can see from figure 5.3, estimates for the case then the input signal is modelled as sinc-shape give best results for values of $l > 100$. For the case then the input signal is modelled as band-limited white Gaussian noise, estimates become more stable for $l > 300$ We use $l = 400$, $time = 100$, $w_c = 0.8\pi$ for further measurements.

5.3.2 Inaccuracies in estimation of time-skew

The LS algorithm performs accurate estimation in systems where skews and gains are small in magnitude, because the reconstruction formula given by equation 5.5 is based on Taylor series expansion around the point $\tau = 0$. In this section we will evaluate dependence of estimator on the time-skew values.

The input signals are modelled as either sinc-functions or band-limited white Gaussian noise with parameters described in section 5.3.1. In this section we will show how inaccuracies in the estimator increase with magnitudes of τ. First of all we evaluate performance of time-skew estimation for the system with two branches. To do it, we change the second value of time-skew vector τ_1 in the range $\tau_1 \in (-0.9T_s, 0.9T_s)$ with step-size $\tau_s = 0.01T_s$ while keeping the value of the gain vector the same ($g = [1\ 1]$). Results are presented in the figure below.

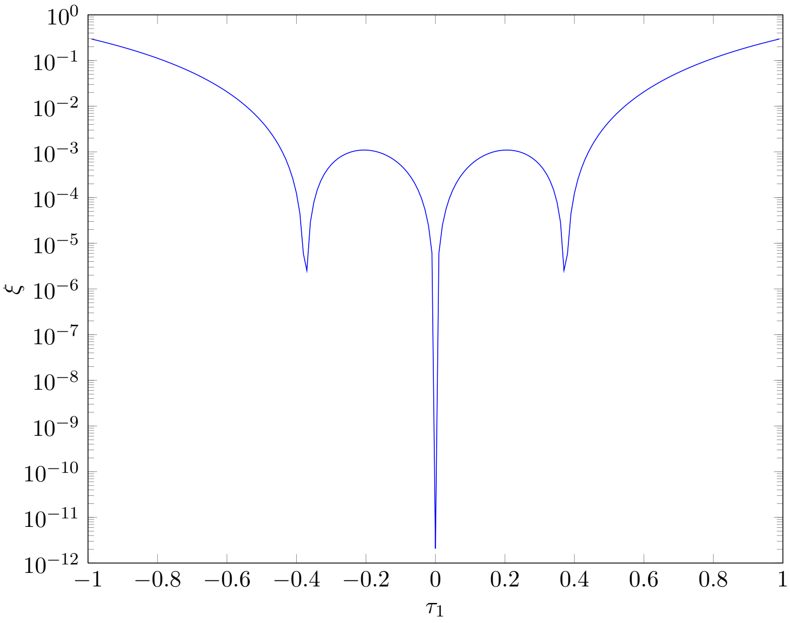

Figure 5.4: Inaccuracies in estimation of time-skew for the signal $y[n]$ modelled as a sinc-function

From figure 5.4 we can see that for the same values of τ estimates are more accurate than for others. The two minima in $\tau \approx \pm 0.37$ are due to the cut-off frequency (i.e. if we choose cut-off frequency to be larger than in this example, these two minima move farther from the point $\tau = 0$).

The next figure shows results for time-skew estimation for the input signal is modelled as band-limited white Gaussian noise. The parameters for signal modelling are the same, but now we take the step-size $\tau_s = 0.1T_s$ and produce $S = 10$ simulations for each value of τ. The result value for inaccuracies computed as average value for S simulations.

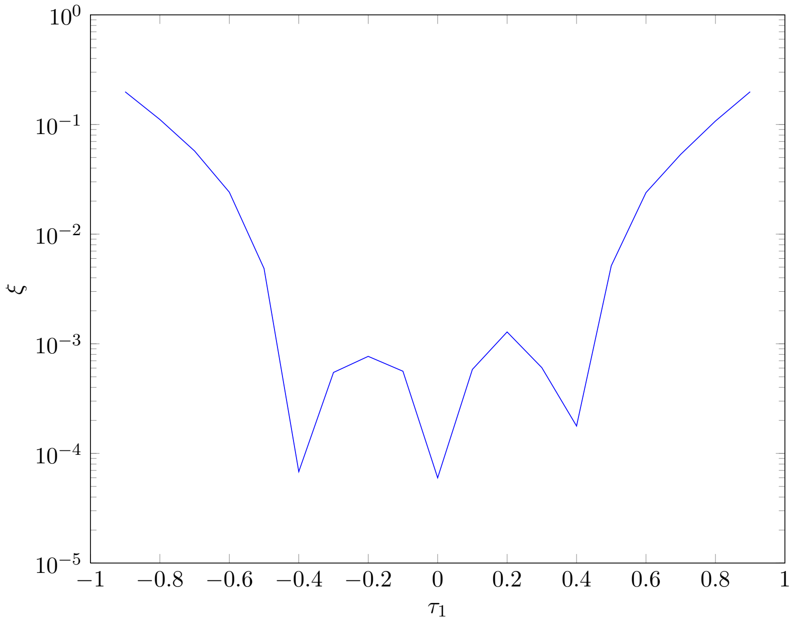

Figure 5.5: Inaccuracies in estimation of time-skew for the signal $y[n]$ modelled as bandlimited white Gaussian noise.

As we can see from figures 5.4 and 5.5, inaccuracies between the estimation of τ grows exactly with a magnitude of τ. As was mentioned earlier, this fact is due to the Taylor series expansion around $\tau = 0$ which is used for finding an approximation for the estimated signals reconstruction value. For values of τ_i which are relatively large in magnitude (for example, $\tau_1 > 0.2$), an iterative method can be used in order to improve estimation accuracy as described in the next section.

5.3.3 Results for the iterative method

The iterative method presented here, for improving the accuracy of the initial LS method, is derived based on Newton's method. Specifically, after we have found $\hat{\tau}$ in the first iteration, we compute the first order Taylor series approximation of $x[n]$ around the new point of τ which is the result of the first iteration. As a result we find another value of $\hat{\tau}$ which is closer to the desired value. In this section we present results for two and three iterations.

In the Matlab implementation for the system we simply update τ in each iteration ($\tau = \tau - \hat{\tau}$, where $\hat{\tau}$ is the least-squares estimate).

Estimate of τ based on two iterations

Parameters for modelling are the same as in previous section, but now we take two iterations for each value of τ. Results for the systems with two branches using a sinc-function as the input signals are presented in the figure below.

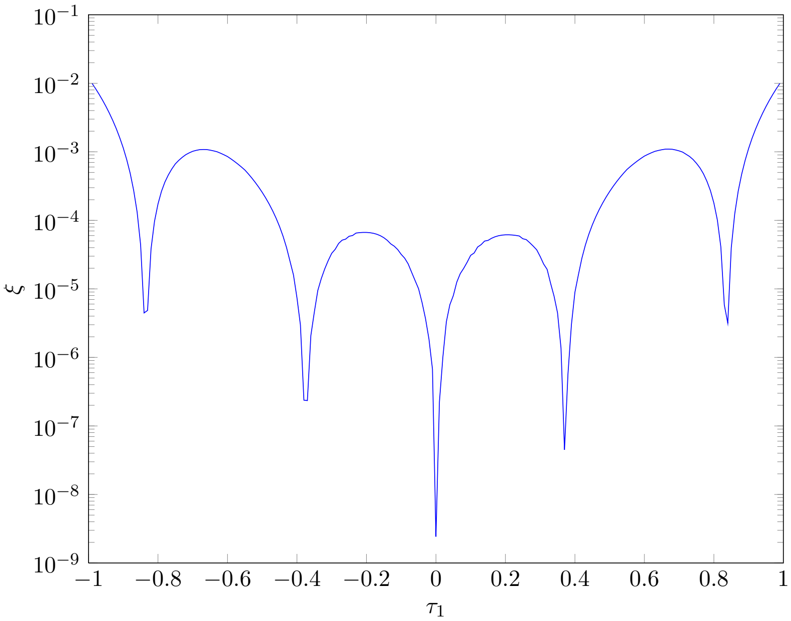

Figure 5.6: Inaccuracies in estimation of time-skew for the iterative method with two iterations.

For the case with two iterations, again some values of τ_1 give better estimates (the second iteration gives two extra minima in figure 5.6 compared to figure 5.4. Now all the inaccuracies values ξ are below 10^{-2}. Compared to the case with one iteration, estimate dependence on value of τ is slightly weakened as expected.

The next figure show the results for the systems using band-limited white Gaussian noise as the input signal.

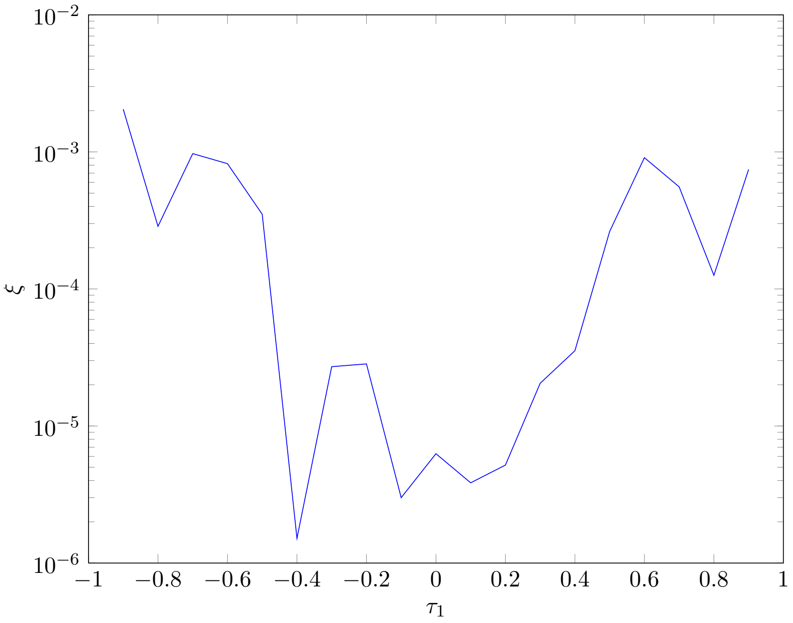

Figure 5.7: Inaccuracies in estimation of time-skew for the iterative method with two iterations using the signal modelled as non-uniformly sampled band-limited white Gaussian noise.

For signals modelled as band-limited white Gaussian noise, performance for estimation of τ grows as well. From figures 5.5 and 5.7 we see that dependence of $\hat{\tau}$ on the magnitude of τ is slightly reduced. We can see that all the values of τ are estimated with inaccuracy $\xi \leq 10^{-2}$.

Estimate of τ based on three iterations

The parameters for modelling the input signals are the same as in previous sections, but we now take three iterations for computing each estimate value of τ. The next two figures show the simulation results.

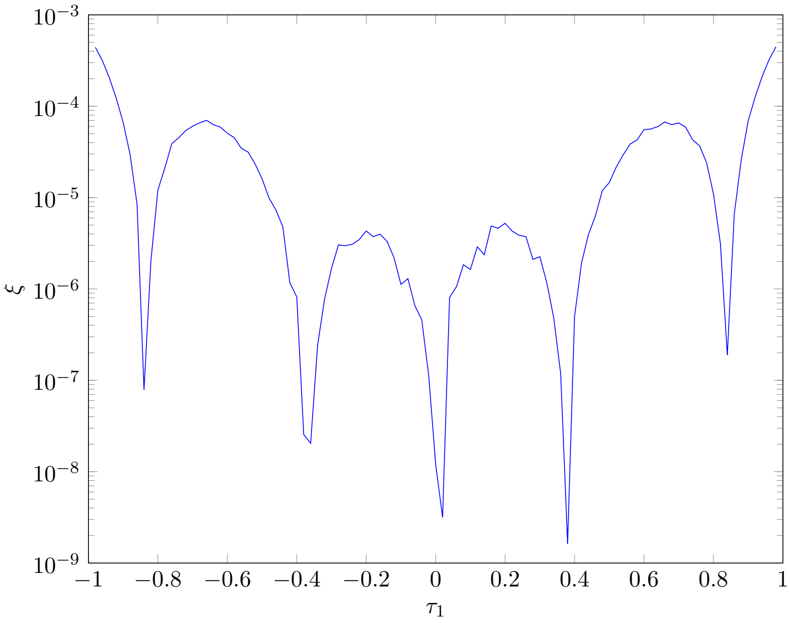

Figure 5.8: Inaccuracies in estimation of time-skew for iterative method with three iterations.

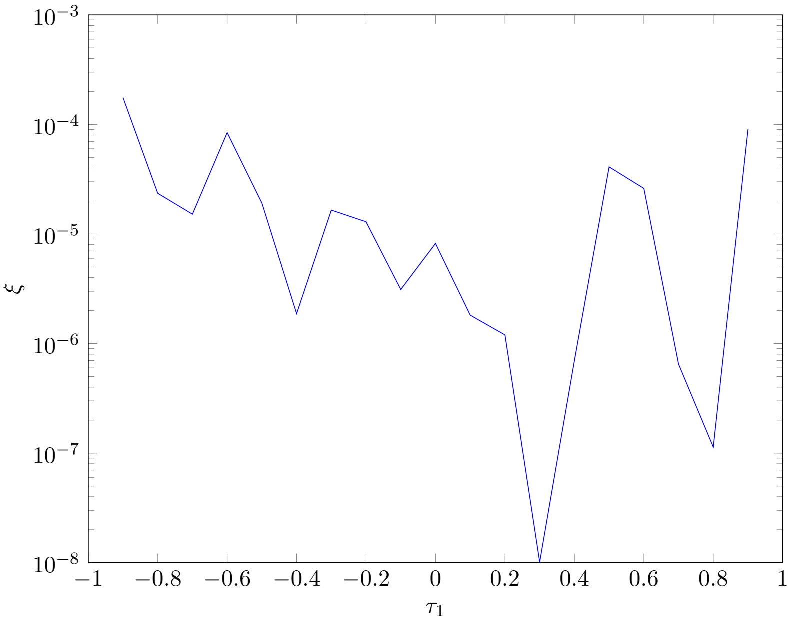

Figure 5.9: Inaccuracies in estimation of time-skew for iterative method with three iterations.

If we compare figures 5.8 and 5.9 with figures 5.4 to 5.5 respectively, we can see that performance for the estimator is improved further more than for case with two iterations: all the values of ξ are below 10^{-3}. We also see that dependence $\hat{\tau}$ on τ is reduced even more than for the case with two iterations as was expected.

Results given above confirm our theoretical expectation about ability to improve estimate performance. This improvement is due to the increasing system complexity. Instead of $O(M^2N)$ complexity (for system with only one iteration), we have $I * O(M^2N)$ where I is number of iterations.

5.3.4 System with three branches

In this section we evaluate the performance of time-skew estimation for systems with three branches (τ is a vector of length equal to 3 with the first element τ_0 equal to 0). In our simulations we change the values of τ_1 and τ_2 in a range from -0.45 to 0.45 with step-size equal to 0.05. Other parameters are the same as in previous sections. Results are presented in figure 5.10.

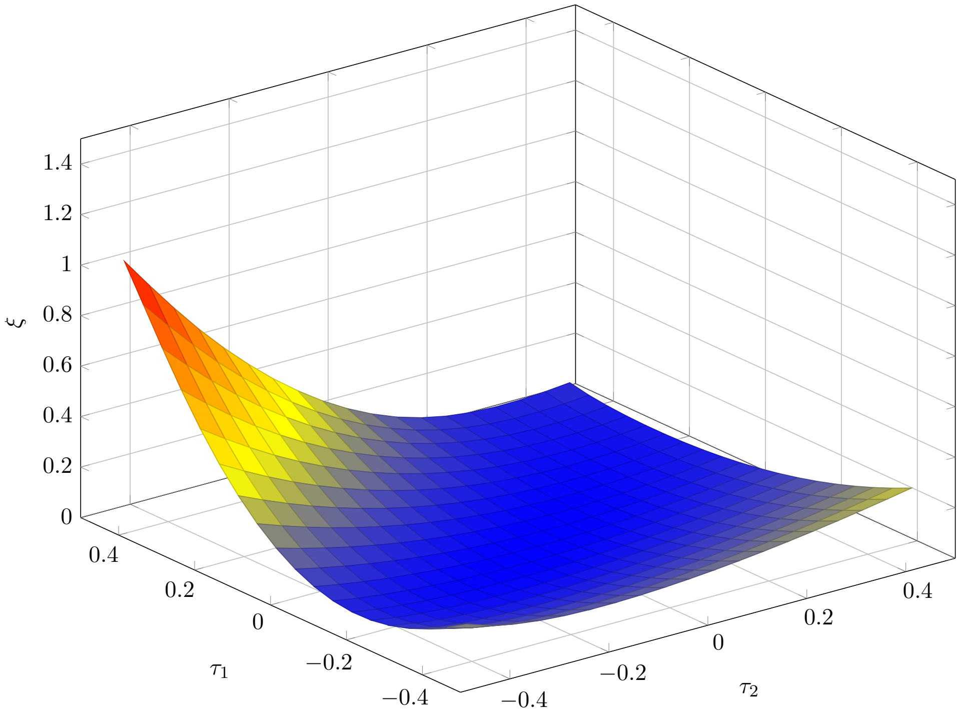

Figure 5.10: Measured inaccuracies for the estimator of τ. Parameters: Filter length $K = 10$, signal length $l = 402$, $time = 100$ and bandwidth $w_c = 0.8\pi$.

As we can see from the figure 5.10, inaccuracies grow with magnitude of τ. We also see that differences between values of each of the elements of τ make sense: inaccuracies for the estimator grow with $|\tau_1 - \tau_2|$. Estimation of time-skew parameter for the case with $M = 3$ may be improved by using more than one iteration.

5.4 Simulations where the input signal bandwidth does not fulfil the input signal restrictions

To fulfil input class restrictions the input signal $x(t)$ has to be band-limited such that sampling of $x(t)$ with F_s fulfil the Nyquist criteria and with non-zero time-skews will produce the signal $y[n]$ having aliasing in spectrum in the frequency band $w_c < w < \pi$ (see Chapter 3). It means that non-uniform periodic sampling of $x(t)$ will produce $y[n]$ which is unique.

The parameters for modelling are the same as in sections 3.1.1 but now we take $w_c = 20\pi$. That means we have $F_c = \dfrac{w_c}{2\pi} = \dfrac{20\pi}{2\pi} = 10$, with sampling rate $F_s = \dfrac{l}{time} = \dfrac{2000}{100} = 20$. We then have $F_s = 2F_c$. This sampling rate fulfil Nyquist criteria, but does not fulfil the requirements for our system. The spectrum for this signal (assuming no time-skew) is represented by figure 5.11.

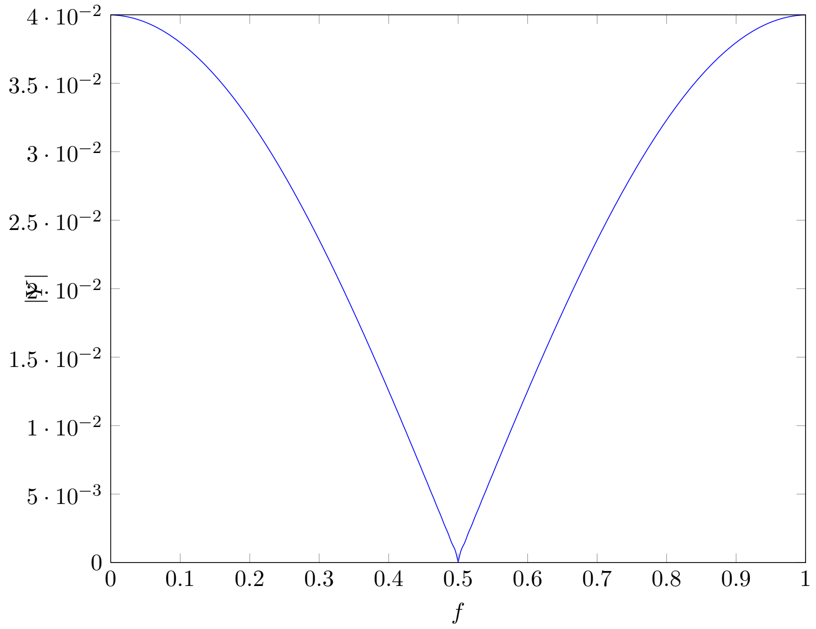

Figure 5.11: Spectrum for a signal sampled with sampling rate $F_s = 2F_c$.

If we introduce any time-skew for the signal represented by figure 5.11, we have aliasing in spectrum as described in chapter 3. The spectrum for a signal modelled with time-skew vector equal $\tau = [0\,0.3]T_s$ is represented by figure 5.12:

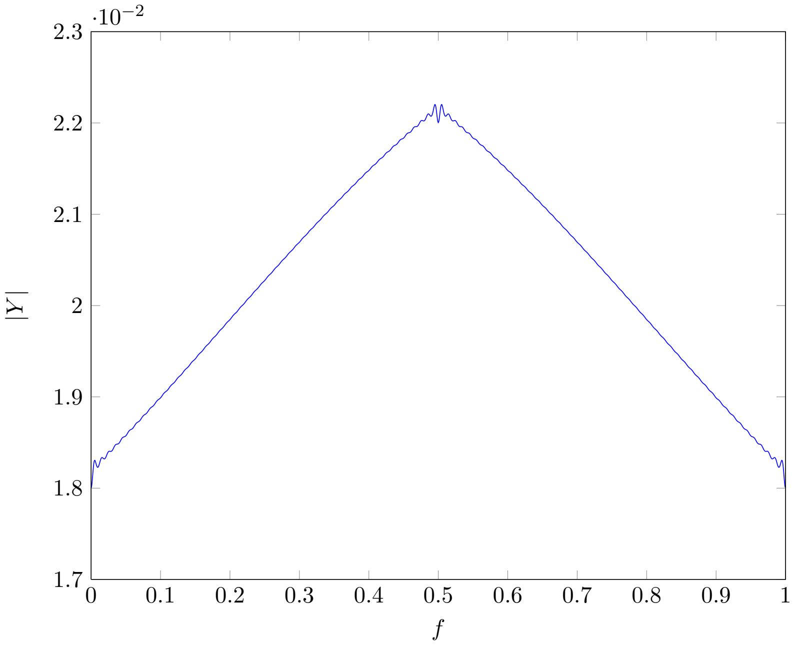

Figure 5.12: The spectrum for a signal sampled with sampling rate $F_s = 2F_c$ and time-skew $\tau = [0\,0.3]$.

If we compare figures 5.11 and 5.12, we can conclude that it is not possible to estimate τ by using the technique described here. This fact is illustrated by figure 5.13.

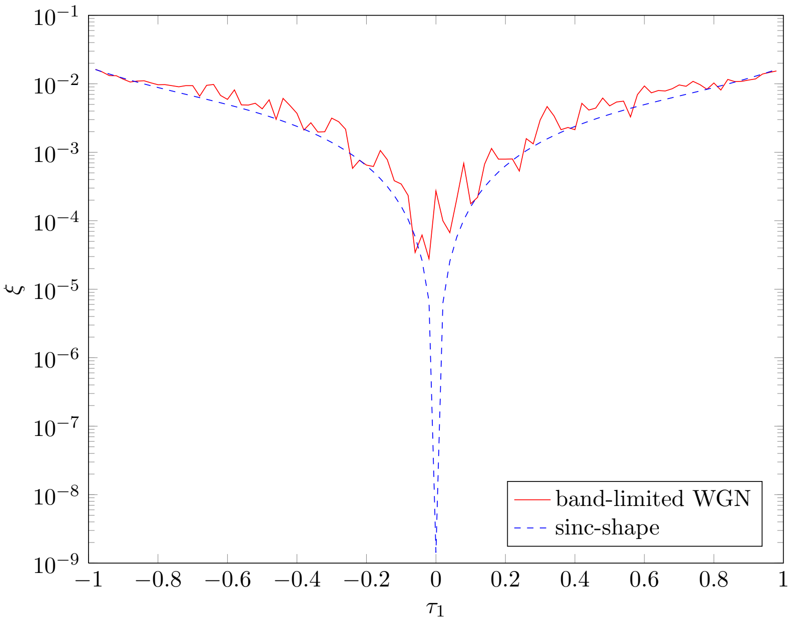

Figure 5.13: Inaccuracies in estimation of time-skew without fulfilling input restriction for $x(t)$.

As was expected, for input signals which does not fulfil the input signal restrictions, the system doesn't work: even for values of τ small in magnitude, the difference between $\hat{\tau}$ and τ is very big.

5.5 Evaluating the performance for gain estimation

To evaluate performance of gain estimation we use the same parameters for the simulation as in previous section (with $M = 2$), but now we change the value of the second value in the gain vector g_1 to the range $g_1 \in (0.1, 2)$ using step size $g_s = 0.1$ while keeping the value of the time-skew vector the same ($\tau = [0\,0]$). The estimator inaccuracies ξ_g are measured as as

$$\xi_g = \frac{|g - \hat{g}|^2}{M - 1} \tag{5.27}$$

The results are presented in figure 5.14.

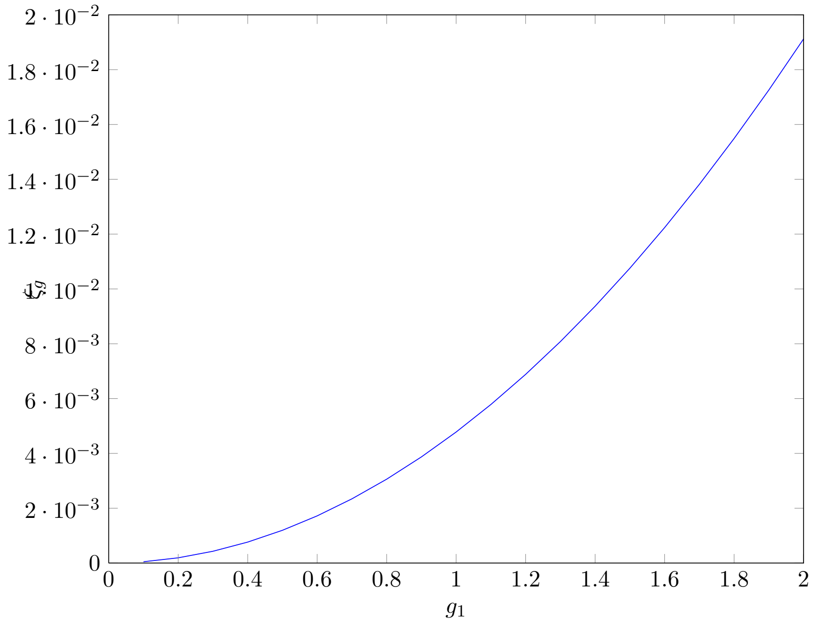

Figure 5.14: Inaccuracies in estimation of gain for signal $y[n]$ modelled as sinc-function

As we can see from figure 5.14, inaccuracies for the estimator grow with magnitude of g and this dependence is linear. If we normalize the inaccuracies by the g^2 (or measure inaccuracies as $\xi_g = \dfrac{(|g - \hat{g}|^2}{(M-1)g^2}$), we see that the normalized inaccuracies are the same for the whole range of g and equal to 0.0047 or 0.47% of the gain. As we know from section 5.3, the dependency for time-skew estimator on the time-skew is not linear. For the rest of the report we concentrate our attention on the estimation of the time-skews and assume that the system has the gain g always equal to 1 [3].

In this chapter we described the LS-method for estimation of unknown parameters (time-skew and gain) for a non-uniformly sampled signal. The performance of the system using LS-method will be discussed in Chapter 7. In the next chapter we will evaluate the adaptive method for estimation of the time-skew.

[3] the 1 here represents the vector of length M with each element equal to 1

6 Adaptive filter realization

In an adaptive filter realization we can either use the desired response (pilot signal) or use only the ADC inputs. Because the disadvantages of the first method (extra hardware, decrease of sampling resolution and delays), we restrict our attention to the second method.

The block implementation of the time-skew estimate described in Chapter 5 has $O(M^2N)$ complexity and takes a large amount of memory. We develop a recursive least-squares (RLS) implementation that distributes the computation over time and adapts to shifts in the parameters. Instead of $O(M^2N)$ complexity every N samples RLS takes $O(M^2)$ complexity every sample. To simplify further, the RLS algorithm can be replaced by the least mean-squares (LMS) algorithm which takes lower complexity due to the fact that no estimation of the covariance matrix P is made and only the gradient is used.

6.1 Recursive Least Squares (RLS) method

In recursive implementations of the method of least squares (RLS), we start the computation with known initial conditions and use information contained in new data samples to update the old estimates. Therefore the length of the observable data is variable.

Compared to block implementation of the timing skew estimate (5.1), we simply take one row $\mathbf{u}[n]$ of the matrix R (5.15) for each iteration.

The initialization for the RLS algorithm includes two equations:

$$\hat{\tau}[0] = \mathbf{0} \tag{6.1}$$

$$\mathbf{P}[0] = \delta \mathbf{I} \tag{6.2}$$

The recommended choice for the regularization constant δ is that it should be small compared to $0.01\sigma_u^2$, where σ_u^2 is the variance of a data sample $u(n)$. The value of σ has an influence on convergence of the estimator to the true value. For large data lengths, the exact value of the regularization constant δ has an insignificant effect.

With RLS algorithm the equation for updating the time-skew value for each iteration can be expressed as:

$$\pi[n] = \mathbf{P}[n-1]\mathbf{u}[n] \tag{6.3}$$

$$\mathbf{k}[n] = \frac{\pi[n]}{\lambda + \mathbf{u}^T[n]\pi[n]} \tag{6.4}$$

$$\zeta[n] = \gamma[n] - \hat{\tau}^T[n-1]\mathbf{u}[n] \tag{6.5}$$

$$\hat{\tau}^T[n] = \hat{\tau}^T[n-1] + \mathbf{k}[n]\zeta[n] \tag{6.6}$$

$$\mathbf{P}[n] = \lambda^{-1}\mathbf{P}[n-1] - \lambda^{-1}\mathbf{k}[n]\mathbf{u}^T[n]\mathbf{P}[n-1] \tag{6.7}$$

where λ is a positive constant close to, but less than, 1. (If we have $\lambda = 1$, we have the ordinary method of least squares). The use of the λ, in general, is intended to ensure that data in the distant past are "forgotten" in order to afford the possibility of following the statistical variations of the observable data when we deal with non-stationary environments. We define a forgetting factor λ such that the weight of the $(n-i)$th sample is λ^{-i}. The case with $\lambda = 1$ corresponds to infinite memory.

6.1.1 Results for RLS-method

For evaluating the performance for adaptive method of time-skew estimation, we measure mismatches between the true value of time skew and the estimated ones. As was described in Chapter 2, the magnitude of time-skews can take values between $-T_s < \tau < T_s$. For our simulations, we use $M = 2$ (the number of A/D converters for the system) or $M - 1 = 1$ time-skews to be estimated. We change the τ_1 in the range $(-0.9T_s, 0.9T_s)$ with step size $\tau_s = 0.1$ and produce ten simulations for each value of time-skew. The result value is computed as the average value for ten iterations. The tests are performed using band-limited Gaussian noise as input. The δ is chosen to be equal to 10^{-2}.

First, we decide how long the signal has to be or how many iterations is needed for convergence, and then evaluate the performance for the method. Figure 6.1 illustrates how estimate of time-skew depends on the number of iterations.

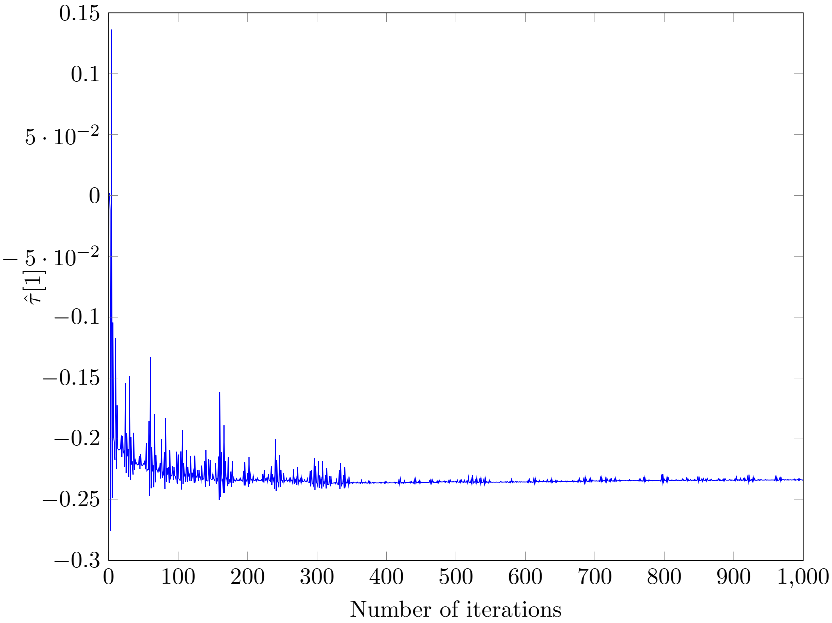

Figure 6.1: Dependency of time-skew estimation value on the number of iterations for the RLS method. ($\tau_1 = -0.2$)

As we can see from figure 6.1, for the parameters described above, the time-skew estimator converges to the true value (with a bias) after approximately 350 iterations (the example illustrates estimation of τ for $\tau_1 = -0.2$, but the same tendency holds for all values of τ). We set the number of iterations equal to 400 (the same length for the input signal as was used for evaluation performance for LS-algorithm in Chapter 5) to evaluate performance of the method. Figure 6.2 represents the results of our simulations.

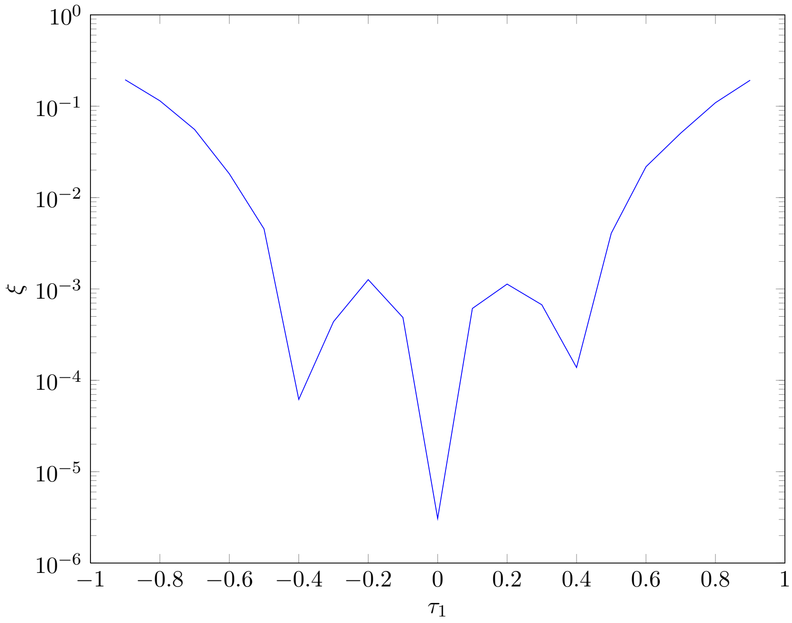

Figure 6.2: Inaccuracies in estimation of time-skew for RLS method.

As we can see, the RLS method performs better for values of τ_1 close to zero. If we compare figure 6.2 and figure 5.5, we can see that the RLS method (adaptive) performs almost the same as the LS-estimation (which is based on blocks of data).

6.2 The method of Least Mean Squares (LMS)

The main advantage of the LMS method is its lower complexity. The difference between the RLS algorithm and LMS is that no estimation of the covariance matrix $\mathbf{P}$ is made. Instead estimates $\hat{\tau}$ are updated by the gradient direction that minimizes the error $\xi[n]$.

The equations for updating estimators using LMS algorithm becomes

$$\zeta[n] = \gamma[n] - \hat{\tau}^T[n-1]\mathbf{u}[n] \tag{6.8}$$

$$\hat{\tau}^T[n] = \hat{\tau}^T[n-1] + \mu\mathbf{u}[n]\zeta[n] \tag{6.9}$$

where μ denotes the step-size parameter of the update which has to satisfy the condition: $0 < \mu < \dfrac{\lambda_{max}}{2}$, where λ_{max} is the largest eigenvalue of the correlation matrix $\mathbf{R}$ ($\mathbf{R}$ is the M by M correlation matrix of the tap inputs $u[n]$, $u[n-1]$ ... $u[n-M+1]$) [7].

6.2.1 Results for LMS-method

Figure 6.3 shows the dependency of the time-skew estimation value on the number of iterations for two values of μ.

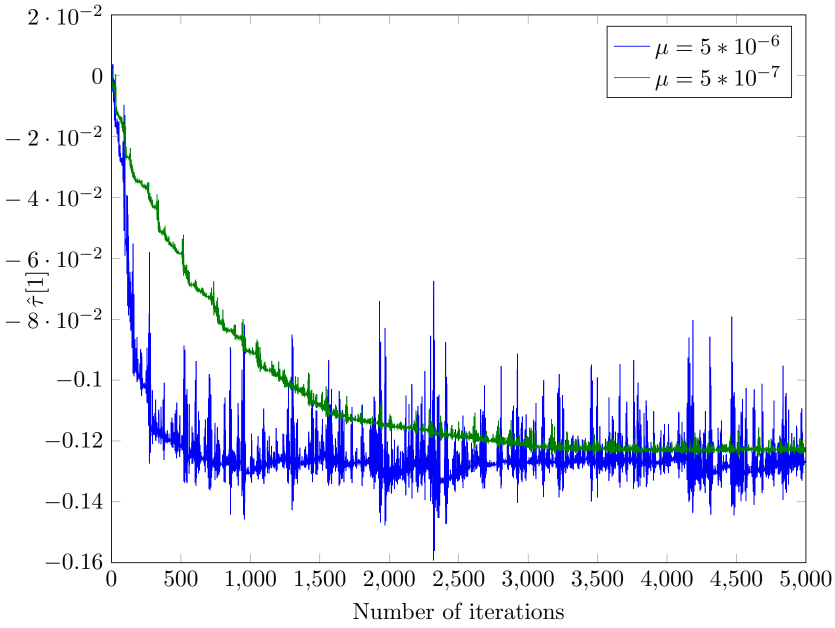

Figure 6.3: The Dependency of the time-skew estimation value on the number of iterations for LMS method. ($\tau_1 = -0.1$)

As we can see from figure 6.3, the estimate produced using the value of $\mu = 5 * 10^{-7}$ has the lower variance, but it converges only after approximately $4 * 10^3$ iterations. For $\mu = 5 * 10^{-6}$, estimate converges to the same value after approximately 10^3 iterations, but its variance is larger.

Although the LMS algorithm is slower than RLS in convergence, it requires only $O(M)$ computation per sample. This fact makes the LMS algorithm very attractive for real system implementations. In simulations performed with sufficiently small step-size, the LMS algorithm converges to the same skew estimates as LS and RLS methods [7].

7 Results

In this section, we numerically evaluate the performance characteristics of the blind calibration method for system with 2 branches. We can compare either the signal-to-quantization-noise (power) ration (SQNR) or number of effective bits for calibrated and uncalibrated signals. If the sampling rate F_s fulfil the Nyquist criteria, quantization is the only error in the A/D conversion process. Thus, we can evaluate the quantization error by quantizing analog signal $x(t)$ instead of the discrete-time signal $x[n] = x_a(nT_s)$. The SQNR for the analog signal (assuming uniform quantization)can be expressed on a logarithmic scale as ([4] Chapter 1.4.4.):

$$SQNR = 10 \log_{10} \frac{P_x}{P_n} = 10 \log_{10} \frac{3}{2} 2^L = 1.76 + 6.02b \; dB \tag{7.1}$$

where $P_x = \sigma_x^2 = E[x^2(n)]$ is the signal power, $P_n = \sigma_e^2 = E[e_q^2(n)]$ is the power of quantization noise and b is the number of bits using for quantization ($b = log_2 L$).

Using 7.1, we can express the effective number of bits (that may be somewhat less than the number of bits in the A/D converter) as

$$b = \frac{SQNR - 1.76}{6.02} \tag{7.2}$$

In our test we use 12-bits A/D converters which generate the quantization noise $w_i[n]$. The tests are performed using the system with two branches and band-limited Gaussian noise as input signal. Only the LS-algorithm (which is based on blocks of data) is used in this chapter for time-skew estimation. We know from Chapter 6 that the RLS and LMS (adaptive algorithms) converge to the same value as LS-algorithm.

7.1 Evaluation of the system performance depending on excess bandwidth

The excess bandwidth is a result of the rate using for sampling of the analog band-limited signal. According to Nyquist criteria, $F_s = 2F_c$ results in no exceed bandwidth. For simulation in the previous section sampling rate was chosen to be five times bigger than the Nyquist rate ($F_s = 5 * 2F_c$). It means that information about the input signal takes only 20% of the spectrum in the range $(0, \pi)$ and gives the factor of 80% oversampling.

From Chapter 3 we know that time-skews between the branches result in overlapping in the spectrum. To avoid distortions of the part of the spectrum $Y(w)$ which inhold information about the input signal and estimate the time-skew correctly, we need the exceed bandwidth to be equal to at least 1/3 or approximately 33% (It means that only 67% of the spectrum contains information about $x(t)$). In this case overlapping for the spectrum $Y(w)$ occurs in the band $\frac{2\pi}{3} \leq |f| \leq \pi$ and does not affect the information about the signal which is in the band $\leq |f| \leq \frac{2\pi}{3}$.

To evaluate how the system performance depends on exceed bandwidth, we change value of w_c in the range $(0.1\pi, \pi)$ with step size $w_{cs} = 0.025\pi$ The results for the values of $\tau[1] = 0.4$ are shown in figure 7.1.

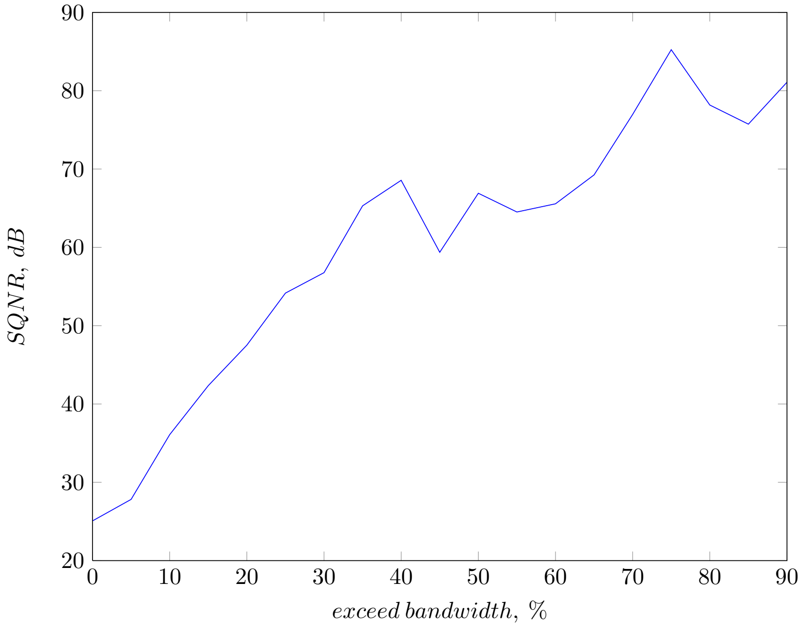

Figure 7.1: Dependency of the SQNR on the excess bandwidth.

As we can see from the figure above, SQNR increases with increasing the excess bandwidth (for other values for the $\tau[1]$ results are similar). As was expected, the excess bandwidth smaller than 30%, reduces the value for SQNR significantly.

7.2 Evaluation of the system performance depending on time-skew

In this section we show how the SQNR depends on time-skew. Since from previous chapters we know that dependence reconstructed signal on τ is symmetric around $\tau = 0$, we take only positive values of τ_1 in the range $(0, T_s))$ for our tests. To do it for the system with $M = 2$, we change value of τ_1 in the range $(0, T_s)$ with step size $\tau_s = 0.1T_s$. The block size of 200 samples and the factor of 80% oversampling are used for modelling of the input signal.

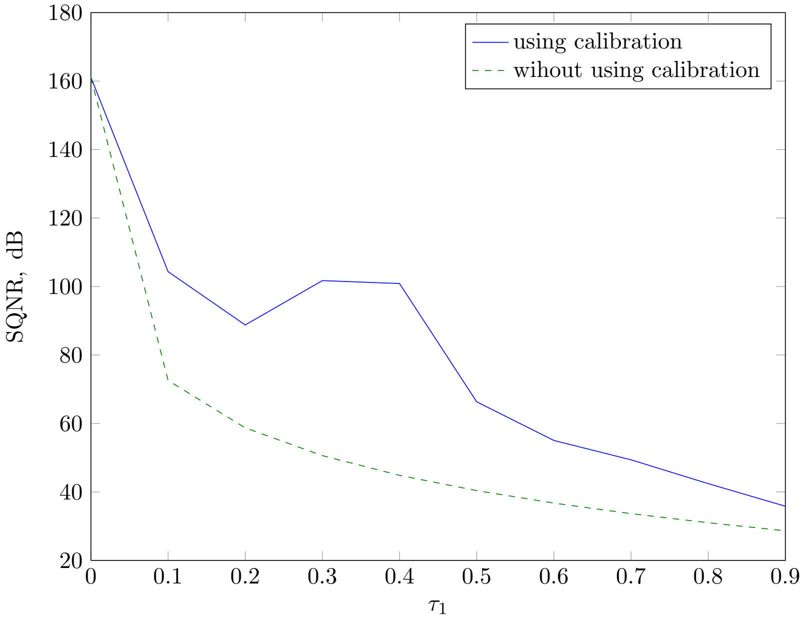

Figure 7.2: Dependency of the SQNR on the time-skew value.

For the uniform sampling case, we have the same SQNR for both calibrated and uncalibrated signals. With time-skew small in magnitude ($|\tau_i| \leq 0.4$), the system with calibration performs better, but if the magnitude of τ becomes larger, the SQNR goes down for both systems. From Chapter 5 we know that our system can estimate some values of τ better than the others. Because of this we have minima for SQNR at the point where $\tau_1 = 0.1$ and maxima at the interval $\tau_1 = (0.3, 0.4)$.

From Chapter 5 we also know that the performance for system using calibration can be improved using relinearization for time-skew estimation. The results for the systems using two and three iterations for the time-skew estimation are shown in figure 7.3.

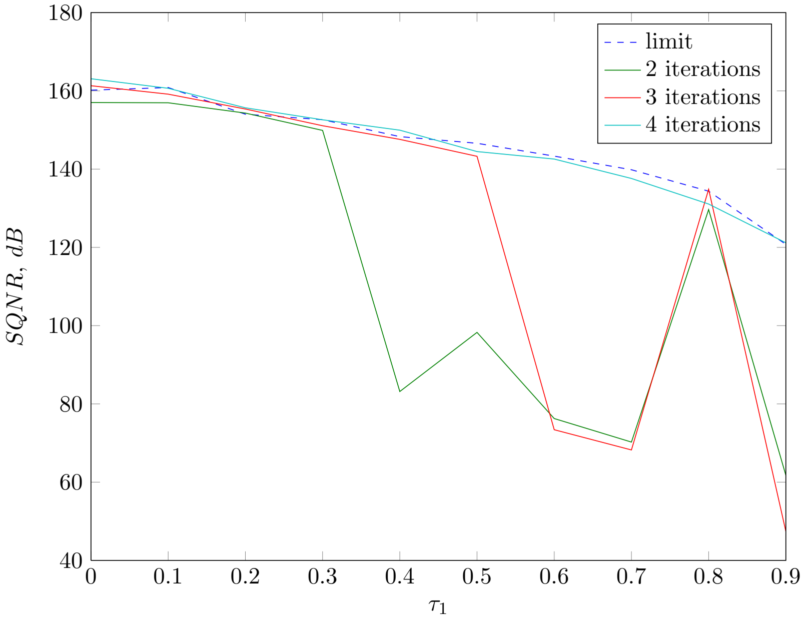

Figure 7.3: Dependency of the SQNR on the time-skew value for the systems which use iterative method for time-skew estimation.

If we use iterative method for the time-skew estimation, we extend the range of values for τ for which estimator is more accurate (section 5.3.3). As we can see from figure 7.3, SQNR for the system using 4 iterations approximately reaches the limit[1]. For the system using 2 iterations, we have the SQNR values in the range $\tau_1 \in (0.4, 0.7)$ smaller than for the other values of τ_1. This is due to the fact that some values of τ_1 are estimated more accurate than the others (see figure **??**). For the system using three iterations we also have minima in the range $\tau_1 \in (0.6, 0.7)$ and maxima at $\tau_1 = 0.9$. This is also due to the fact that accuracies for time-skew estimator vary with values of τ_1 (see figure **??**), and we see that the range for values of τ_1, which are estimated more accurate, is extended from $\tau[1] \in (0, 0.3)$ to $\tau_1 \in (0, 0.5)$. Because of the input signal is modelled as filtered white Gaussian noise (as a random process) and its spectrum does not equal exactly to zero for $|w| \geq w_c$, it cause to that the SQNR values may vary slightly. This explains the fact that same SQNR values for systems using 3 and 4 iterations exceed the limit value.

7.3 Evaluation of the system performance depending on the number of bits used for quantization

In order to evaluate how the system performance depends on the number of bits using for quantization ($Nbit$), we test the system for values of $\tau_1 = 0,\ 0.1,\ 0.2,\ 0.3$ (results for other

[1] we define the limit as SQNR for a system which uses time-skew estimation without error

values are analogous). For our simulations we change value of $Nbit$ in the range $(5, 30)$. The parameters for the input signal are the same as in the previous sections.

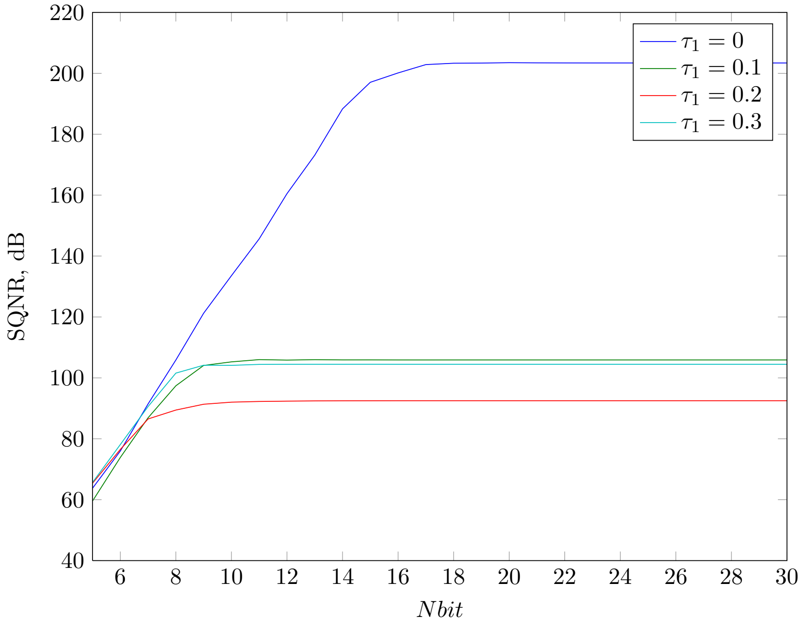

Figure 7.4: Dependency of the SQNR on the number of bits using for quantization.

As we can see from figure 7.4, for the case with non-uniform sampling, the SQNR grows with $Nbit$ linearly until the number of bits reaches the value $Nbit = 10$ for our system. For the case with uniform sampling this limit is much higher. If we increase the range of the input signal (and the range of the quantizers), then the increasing of $Nbit$ may result in higher SQNR, but with a new limit for $Nbits$ after which where is no significant improvement.

7.4 Evaluation of the system performance depending on the number of A/D converters

For our simulations we chose $Nbit = 10$, and we compare SQNR for two systems with $M = 2$ and $M = 4$ A/D converters. From Chapters 5 we know that estimation of the time-skew parameter performs better for the case with $M = 2$ than for the case with $M = 3$. The reconstruction filter-bank introduces minimal distortions for the system with $M = 2$ A/D converters for the whole range of τ_1 (see Chapter 4). Thus, we can expect what the system with $M = 2$ converters will have the best results.

Because of the length of τ is different for these two systems, we represent the system timing error as the sum of magnitudes of the time-skews $\sum_{i=1}^{M} |\tau_i|/(M-1)$ for our simulations. We use only one iteration for time-skew estimation, therefore we assuming small mismatch regime and change the values of τ_i for in the range $\tau_i \in (-0.2T_s,\ 0.2T_s)$ with step size $\tau_s = 0.1T_s$. Results are shown in figure 7.5.

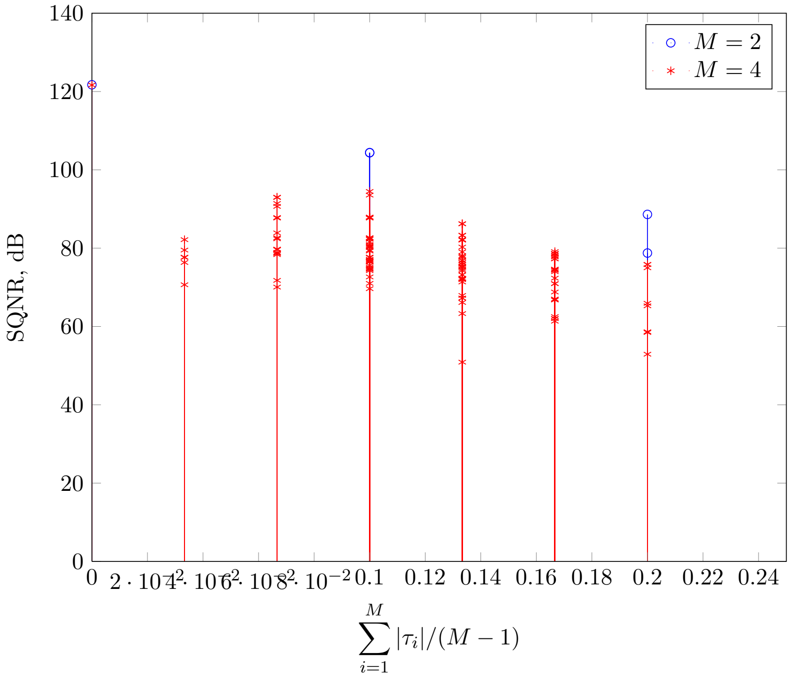

Figure 7.5: Dependency of the SQNR on the number of A/D converters.

From figure 7.5 we see that the SQNR for the case with uniform sampling is the same. If we have some time-skews, the SQNR for the system with two A/D converters is higher for all values of $\sum_{i=1}^{M} |\tau_i|/(M-1)$. If we apply the linear fitting (from Matlab graphic toolbox) to the data sets, we see that the SQNR value in the middle of the range (at the point where $\sum_{i=1}^{M} |\tau_i|/(M-1) = 0.1$) for the system with $M = 2$ equals approximately 100 dB while for the system with $M = 4$ it equals 75 dB (see figure 7.6).

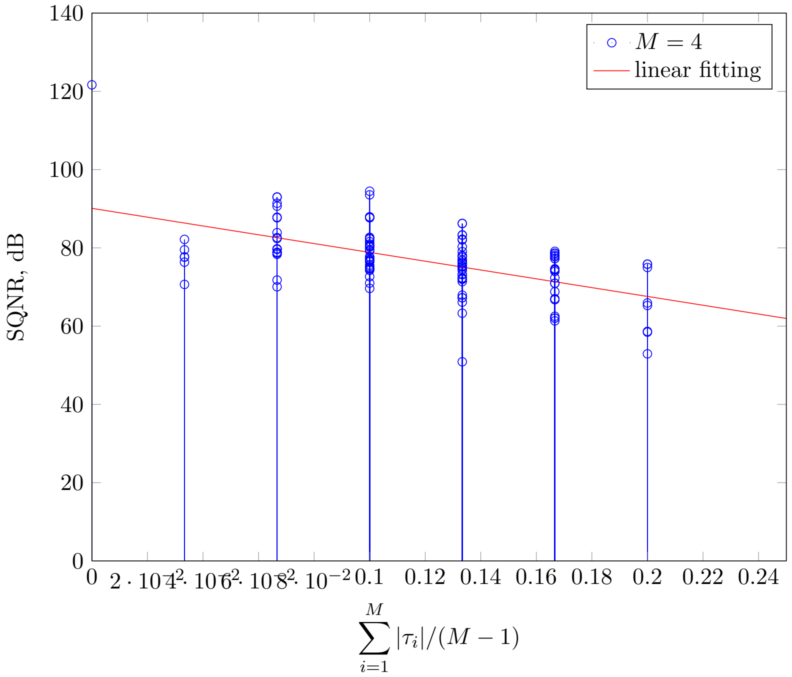

Figure 7.6: Dependency of the SQNR on the magnitude of time-skew for the system with $M = 4$ A/D converters.

For the system with $M = 5$ the SQNR value in the point $\sum_{i=1}^{M} |\tau_i|/(M-1) = 0.1$ equals to approximately 75 dB. Based on this results we conclude that with increasing of the number of the A/D converters, the performance of the system decreases, as was expected.

8 Conclusion

In this report we have described the method for the calibration of the time-interleaved analog-to-digital converters. The algorithms which was represented here, produces accurate reconstruction based on the estimation of the unknown system parameters.

At first, we discussed the reconstruction of the signal represented by the non-uniform samples. We have seen that the non-uniform samples can be represented as sum of M sequences of uniform samples and that the efficient reconstruction may by performed by the filter-bank which uses information about time-skews between the samples. The restrictions for the input signal are taken into account.

Then we represented some methods for estimation of the unknown parameters, such as the time-skew and the gain, for the non-uniform sampled signal. This parameters can be estimated using methods based on both the blocks of data (LS) and sequence of samples (adaptive methods such as RLS and LMS). These methods perform well assuming the time-skews are small relative the the sampling interval, but for time-skews large in magnitude the method of realization, based on Newton's method, may be used. We focused out attention to the LS-estimation, but we know that all of this methods converges to the same value.

We have evaluate performance for the system using band-limited white Gaussian noise as the input signal. The results have shown that this kind of system may be a good decision for the cases then the load is distributed along many A/D converters.

Bibliography

[1] Yonina C. Eldar, Alan V. Oppenheim *Reconstraction of Bandlimited Signals from Nonuniform and Generalized Samples* (IEEE Transaction on signal processing; Volume 48, No 10, October 2000).

[2] Vijay Divi, Gregory W. Wornell *Blind Calibration of Timing Skew in Time-Interleaved Analog-ti Digital Converters* (IEEE Journal of selected topics in signal processing; Volume 3, No 3, June 2009).

[3] John R. Barry, Edward A. Lee, David G. Messerschmitt *Digital Communication* (Springer, (third edition) 2004).

[4] John G. Proakis, Dimitris G. Manolakis *Digital Signal Processing* (Pearson, (fourth edition) 2007).

[5] Steven M. Kay *Fundamentals of Statistical Signal Processing: Estimation Theory* (Prentice Hall PTR, 1993).

[6] Simon Haykin *Communication Systems, 4th edition* (J. Wiley 2001).

[7] Simon Haykin *Adaptive Filter Theory, third edition* (Prentice Hall Information end System Sciences Series...).

[8] Torbjorn Svendsen *Course literature for TTT4225: Applied signal processing - Speech Communication System; pp 10-11* (January, 2012).

[9] J. L. Yen *On nonuniform sampling of band-limited signals* (IEEE Trans. Circuit Theory; vol. CT-3, pp 251-257, Dec. 1956).

A Appendices

A.1 The function for the generation of the input signal

```
function x = gensignal_as_noise(tau, l, time, w_c)
%produces vector of l elements representing signal which is generated as white Gaussian no

% tau is a vector, represents time-skews for each branch;
% l is length of the signal in samples;
% time is length of the signal in sec;
% w_c is bandwidth of the signal;

K = 10; %the length of the low-pass filter which is used for filtering of the input signal

M = length(tau);
l_path = l/M;   %length of the signal in each path
X = zeros(M, l_path);

Fs_a = 1000;  %sampling rate for the analog signal
Ts_a = 1/Fs_a;
t_ref = linspace(0,time,time*Fs_a);
n_ref = 1:length(t_ref);
F_c = w_c/(2*pi); T_c = 1/F_c;  %cut-off frequency
T_s = time/l; %F_s = 1/T_s; %T = T_s*M;

% White Gaussian noise generation
%rng(s);
x_t_noise = randn(1,length(t_ref));

% Filter with digital cut-off requency w_c
k = -K*T_c:Ts_a:K*T_c;
F = zeros(1,length(k));
for i = 1:length(k)
    if k(i) == 0
         F(i) = w_c/pi;
      else
         F(i) = sin(w_c*k(i))/(pi*k(i));
    end
end

%Signal filtering
x_f = conv(x_t_noise, F);
x = x_f(1, (length(F)-1)/2:end-(length(F)-1)/2-1);

%Sampling points choosing
n = zeros(M, l_path);
for a = 1:M
    i = round(tau(a)*T_s*Fs_a);    %tau_i in samples
    n(a,1)= n_ref(round((a-1)*T_s*Fs_a+i+1));
     for b = 2:l_path
       j = round(((b-1)*M+(a-1))*T_s*Fs_a+i+1);
       n(a,b) = n_ref(j);
    end
end

%Sampling
 for r = 1:M
     for c = 1:l_path
         X(r,c) = x(n(r,c));
     end
 end

 x = matrix_stream(X)'; %representation of the signal as a vector
end
```

A.2 Skript for estimation of the time-skew and gain parameters

```
clear all
%input parameters
tau = [0]';     % time-skews for the elements 1, 2, ... for the time-skew vector
gain = [1]';    %gains for the elements 1, 2, ... for the gain vector
tau = [0; tau];  %the first element in time-skew vector is always 0;
gain = [1; gain];  %the first element in gainv vector is always 1;

l = 400;          % signal length (for each path)
K = 10;            % length of low-pass filter
wc = 0.4*2*pi;          %f_c = w_c/(2*pi) cutoff frequency
```

```
time = 100;

% creating some matricies for estimation
I = ident(l); % creationg identity matrix
T = timeskewsmatrix(tau, l);   %matrix containing information about time-skews
H = diffiltermatrix(l); %creating the Toeplits matrix H representing the dif.filter
L = f_rc_coeff(K, l); %low-pass filter design

tau_hatt = zeros(size(tau));
res_tau = 0; res_gain = 0;
res_quad_tau = 0; res_quad_gain = 0;

for tau_2 = -0.9:0.1:0.9
    tau(2) = tau_2;
    for gain_2 = 0.1:0.1:2;
        gain(2) = gain_2;

y = gensignal_gain(tau, gain, l, time, wc);

%creating matrix R
tau_hatt = tau_hatt(2:end);
[R, gamma] = Rmatrix_tau_gain(tau_hatt, l, L, H, y);

%finding tau_est
teta = (R'*R)^(-1)*R'*gamma;
tau_hatt = [0; teta(1:M-1)];
gain_hatt_1 = [1; teta(M:end)]; gain_hatt = 1./gain_hatt_1;


feil_quad_tau = 0; feil_quad_gain = 0;
feil_tau = 0; feil_gain = 0;
for i = 2:length(tau)
    feil_quad_tau = feil_quad_tau + (tau(i)-tau_hatt(i))^2;
    feil_quad_gain = feil_quad_gain + (gain(i)-gain_hatt(i))^2;
    end
res_quad_tau = [res_quad_tau; feil_quad_tau];
res_quad_gain = [res_quad_gain; feil_quad_gain];
    end
```

```
end

% Plotting Results
t =  -0.9:0.1:0.9;
semilogy(t,res_quad_tau(2:end));
figure
plot(t,res_quad_gain(2:end));
```

A.3 Skript for signal reconstruction with using information about time-skews

```
tau = [0]';    % time-skew vector
tau = [0; tau];
M = length(tau);  %number og path
l = 200;          %signal length (for each path)
time = 100;
K_rec = 10;        %length of reconstruction filter
w_c = 0.2*2*pi;    %cutoff frequency for low-pass filter

I = 100;   %Interpolation factor

% signal designing in matrix (each row represents path og the system)
s = rng; %to generate the same random sequence next time
rng(s);
x = gensignal([0 0], l, time, w_c);
rng(s);  %to use the same random seguence to create y[n] as was used to create x[n]

tau_hatt = zeros(size(tau));
res_quad = 0;

for tau_new = -0.98:0.02:0.98;
    tau_hatt(2) = tau_new;
rng(s)
y = gensignal(tau_hatt, l, time, w_c);
Y = stream_matrix(y',M);
Y_i = interp(Y, I);      %interpolation with factor > M
%   time-skew correcting
y_interp_2 = timeskew_zeros(Y_i(2,:), 0.5+tau_new/2, I);
y_interp_1 = zeros(1, length(y_interp_2));
for n = 1:length(Y_i(1,:))
```

```
    y_interp_1(n) = Y_i(1,n);
end

h = h_rec_wind(K_rec, I, tau_hatt);   %reconstruction filter

x_hatt_1 = conv(y_interp_1, h(1,:));
x_hatt_2 = conv(y_interp_2, h(2,:));

x_hatt = x_hatt_1 + x_hatt_2;
x_hatt_cut = x_hatt(K_rec*I+1:end-K_rec*I+1); %!!!!!

x_hatt_sampl = sampling(x_hatt, I/2, 0);
x_hatt_res = x_hatt_sampl(2*K_rec+1:end-2*K_rec-2);

feil_quad = 0;
feil = 0;
for i = 1:length(tau)
    feil_quad = sum((x(K_rec:end-K_rec) - x_hatt_res(K_rec:end-K_rec)' ).^2);
end
res_quad = [res_quad; feil_quad];
end
res_quad = res_quad(2:end);

%plotting result
D_norm = res_quad./sum(x(K_rec:end-K_rec).^2);
t = -0.98:0.02:0.98;
semilogy(t, D_norm);
```

A.4 Function for creating filter-bank for the case with $M = 2$

```
function h = h_rec_wind(K, I, tau)
%The function computes impulse filter response of the reconstruction filterbank consisting
%(2*K+1) - length of the filter;
%tau is vector representing time-skews between branches (the first element is always zero)
% I - imterpolation factor.

T = 1; %- period of saplings T = 2*T_s
L = I*100;
t = linspace(-K, K, L*K*2);
```

```
tau_r = 0.5 + tau(2)*0.5;

%a - coeff vector
a = zeros(1,2);
a(1) = 1/(sin(pi*(-tau_r)/T));   %a_0
a(2) = 1/(sin(pi*( tau_r)/T));   %a_1

%filters
h_1 = a(1).* T./(pi*t).*sin(pi.*t./T).* sin(pi.*(t-tau_r)./T);
h_2 = a(2).* T./(pi*t).*sin(pi.*t./T).* sin(pi.*(t+tau_r)./T);

h_sampl_1= sampling(h_1,100,0);
h_sampl_2= sampling(h_2,100,0);

%windowing
M = L*K*2/100;
n = linspace(0,M-1, M);
h_ham = 0.42 - 0.5.*cos((2*pi*n)./(M-1)) + 0.08.*cos((4*pi*n)./(M-1)); %Blackman window

h_1_ham = h_sampl_1.*h_ham;
h_2_ham = h_sampl_2.*h_ham;

h = [h_1_ham; h_2_ham];

end
```

A.5 Skript for time-skew estimation using RLS algorithm

```
tau = [-0.1]';    % time-skew vector
tau = [0; tau];
M = length(tau);  %number of path
l = 1000;          % signal length (for each path)
K = 10;           % length of low-pass filter
wc = 1*2*pi;        % cut-off frequency for signal forming
time = 100;
%RLS initialization
delta = 0.01;
lambda = 1;

%creationg identity matrix
```

```
I = ident(l);

%creating the Toeplits matrix H representing the dif.filter
H = diffiltermatrix(l);

%Low-pass filter
L = f_rc_coeff(K, l);

J = 10;    % Number og iterations for each value of tau[1]

tau_est_end = 0;
t = -0.9:0.1:0.9;
res = zeros(1,l);

for tau_2 = -0.2%:0.1:0.9;
        tau(2) = tau_2;
        tau_res = zeros(1,l);
        for j = 1:J
y = gensignal_as_noise(tau, l, time, wc);

%creating matrix R and gamma
tau_hatt = zeros(M-1,1);
R = Rmatrix(tau_hatt, l, L, H, y);
gamma = (L-I)*y;

%RLS initialization
P = delta.*ident(M-1);

%RLS update
est_tau = zeros(M-1, 1);
for n = 1:length(y)
pi = P*R(n,:)';
k = pi./(lambda + R(n,:)*pi);
ksi = gamma(n) - tau_hatt'*R(n,:)';
tau_hatt = tau_hatt+ k*ksi;
P = lambda^(-1)*P-lambda^(-1)*k*R(n,:)*P;
est_tau = [est_tau; tau_hatt];
end
```

```
tau_res = tau_res + est_tau(2:end)';
        end
    tau_est = sum(tau_res(1, end-100:end))/100/J;
    tau_est_end = [tau_est_end; tau_est];
end

tau_est_end = (tau_est_end(2:end) - t').^2;

%plotting results
semilogy(t, tau_est_end);
```

A.6 Skript for time-skew estimation using RLS algorithm

```
tau = [-0.1]';    % time-skew vector
tau = [0; tau];
M = length(tau);  %number og path
l = 5000;          % signal length (for each path)
K = 10;           % length of low-pass filter
wc = 5*2*pi;        %cutoff frequency for signal forming
time = 100;
%LMS initialization
mu = 0.5*10^(-5);

%creationg identity matrix
I = ident(l);

%creating the Toeplits matrix H representing the dif.filter
H = diffiltermatrix(l);

%Low-pass filter
L = f_rc_coeff(K, l);

%update
y = gensignal_as_noise(tau, l, time, wc);

%creating matrix R and gamma
tau_hatt = zeros(M-1,1);
R = Rmatrix(tau_hatt, l, L, H, y);
gamma = (L-I)*y;
```

```
%LMS update
est_tau = zeros(M-1, 1);
for n = 1:length(y)
ksi = gamma(n) - tau_hatt*R(n,:);
tau_hatt = tau_hatt + mu*R(n,:)*ksi;
est_tau = [est_tau tau_hatt];
end

est_tau = est_tau(:, 2:end);
```

A.7 Skript for system testing using the iterative time-skew estimation for $M = 2$

```
tau = [0]';      % time-skew vector
tau = [0; tau];
M = length(tau);  %number of path
l = 200;         % signal length (for each path)
K = 10;           % length of low-pass filter
K_rec = 10;       % length of the reconstruction filter
wc = 0.2*2*pi;        %f_c = w_c/(2*pi) cutoff frequency for signal forming
time = 100;
I = 100;    %interpolation factor
Nbit = 12;
iteration = 3;    %number of iteration using for time-skew estimation

s = rng;
rng(s);
x = gensignal(tau, l, time, wc);
x_q = quantize_w_range(x, Nbit); %quantazing x(t)

% creationg identity matrix
Ident = ident(l);

T = timeskewsmatrix(tau, l);

%creating the Toeplits matrix H representing the dif.filter
H = diffiltermatrix(l);

%low-pass filter design
L = f_rc_coeff(K, l);
```

```
tau_hatt = zeros(size(tau));
sqnr = 0; sqnr_un = 0;
h_un = h_rec_wind(K_rec, I, zeros(size(tau)));   % uncolibrated reconstruction filter

for tau_2 = 0:0.1:0.9
tau(2) = tau_2;

rng(s);
y = gensignal(tau, 1, time, wc);
y_q = quantize_w_range(y, Nbit);
Y = stream_matrix(y_q',M);
Y_i = interp(Y, I);      %interpolation with factor I

%%%%%%%%% reconstruction without using tau %%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%   time-skew correcting
y_interp_2_un = timeskew_zeros(Y_i(2,:), 0.5, I);
y_interp_1_un = zeros(1, length(y_interp_2_un));
 for n = 1:length(Y_i(1,:))
    y_interp_1_un(n) = Y_i(1,n);
 end

x_hatt_1_un = conv(y_interp_1_un, h_un(1,:)); %???????????????????????
x_hatt_2_un = conv(y_interp_2_un, h_un(2,:));

x_hatt_un = x_hatt_1_un + x_hatt_2_un;
x_hatt_cut_un = x_hatt_un(K_rec*I+1:end-K_rec*I+1); %!!!!!

x_hatt_sampl_un = sampling(x_hatt_un, I/2, 0);
x_hatt_res_un = x_hatt_sampl_un(2*K_rec+1:end-2*K_rec-2);

 for i = 1:length(tau)
    sqnr_un_res = 10*log( sum(x.^2)/ sum((x' - x_hatt_res_un).^2));
 end
sqnr_un = [sqnr_un; sqnr_un_res];


%%%%%%%%% estimation tau using iter iterations %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
tau_hatt = zeros(size(tau));
tau_new = tau;
tau_end = 0;

  for iter = 1:iteration
      rng(s);
  y = gensignal(tau_new, l, time, wc);

   tau_hatt = zeros(size(tau));

   %creating matrix R
   tau_hatt = tau_hatt(2,:);
   R = Rmatrix(tau_hatt, l, L, H, y);

   %finding tau_est
   tau_hatt = (R'*R)^(-1)*R'*(L-I)*y;
   tau_hatt = [0; tau_hatt];

   tau_new = tau_new - tau_hatt;
   tau_end = tau_end + tau_hatt;
   end
  tau_hatt = tau_end;

%%%%%%%%% reconstruction with using tau %%%%%%%%%%%%%%%%%%%%%%%%%%%%%

h = h_rec_wind(K_rec, I, tau_hatt);   %reconstruction filter

%   time-skew correcting
y_interp_2 = timeskew_zeros(Y_i(2,:), 0.5+tau_hatt(2)/2, I);
y_interp_1 = zeros(1, length(y_interp_2));
 for n = 1:length(Y_i(1,:))
    y_interp_1(n) = Y_i(1,n);
 end

x_hatt_1 = conv(y_interp_1, h(1,:));
x_hatt_2 = conv(y_interp_2, h(2,:));

x_hatt = x_hatt_1 + x_hatt_2;
```

```
x_hatt_cut = x_hatt(K_rec*I+1:end-K_rec*I+1); %!!!!!

x_hatt_sampl = sampling(x_hatt, I/2, 0);
x_hatt_res = x_hatt_sampl(2*K_rec+1:end-2*K_rec-2);

 for i = 1:length(tau)
    sqnr_res = 10*log(sum(x(K+1:end - K).^2)/sum((x(K+1:end - K)' - x_hatt_res(K+1:end - K
 end
sqnr = [sqnr; sqnr_res];

end

SQNR = sqnr(2:end); b = (SQNR-1.76)./6.02;
SQNR_un = sqnr_un(2:end); b_un = (SQNR_un-1.76)./6.02;
```

List of Figures