Institutionen för systemteknik

Department of Electrical Engineering

Master Thesis

Tree-Structured Linear-Phase Nyquist FIR Filter Interpolators and Decimators

Master thesis performed in Electronics Systems
at Tekniska högskolan vid Linköpings universitet
by

Jimmie Lahti

LiTH-ISY-EX--12/4590--SE

Linköping 2012

Linköpings universitet
TEKNISKA HÖGSKOLAN

Tree-Structured Linear-Phase Nyquist FIR Filter Interpolators and Decimators

Master thesis performed in Electronics Systems
at Tekniska högskolan vid Linköpings universitet
by

Jimmie Lahti

LiTH-ISY-EX--12/4590--SE

Supervisor: **Ph.D. Amir Eghbali**
isy, Linköpings universitet

Examinator: **Prof. Håkan Johansson**
isy, Linköpings universitet

Linköping, 14 maj 2012

Avdelning, Institution Division, Department Department of Electronics Eystems Department of Electrical Engineering SE-581 83 Linköping	Datum Date 2012-05-14
---	---	--

Språk Language ☐ Svenska/Swedish ☒ Engelska/English ☐ _____	Rapporttyp Report category ☐ Licentiatavhandling ☒ Examensarbete ☐ C-uppsats ☐ D-uppsats ☐ Övrig rapport ☐ _____	ISBN _____ ISRN LiTH-ISY-EX--12/4590--SE Serietitel och serienummer ISSN Title of series, numbering _____
--	---	---

URL för elektronisk version http://urn.kb.se/resolve?urn=urn:nbn:se:liu:diva-XXXXX	
---	--

Titel Title	Tree-Structured Linear-Phase Nyquist FIR Filter Interpolators and Decimators
Författare Author	Jimmie Lahti

Sammanfattning Abstract	<p>The master thesis is based upon a new type of linear-phase Nyquist finite impulse response interpolator and decimator implemented using a tree-structure. The tree-structure decreases the complexity, considerably, compared to the ordinary single-stage interpolator structure. The computational complexity is comparable to a multi-stage Nyquist interpolator structure, but the proposed tree-structure has slightly higher delay. The tree-structure should still be considered since it can interpolate with an arbitrary number and all subfilters operate at the base rate which is not the case for multi-stage Nyquist interpolators.</p>
-----------------------------------	---

Nyckelord Keywords	filter, interpolation, decimation, tree-structure, Nyquist, linear-phase
------------------------------	--

Abstract

The master thesis is based upon a new type of linear-phase Nyquist finite impulse response interpolator and decimator implemented using a tree-structure. The tree-structure decreases the complexity, considerably, compared to the ordinary single-stage interpolator structure. The computational complexity is comparable to a multi-stage Nyquist interpolator structure, but the proposed tree-structure has slightly higher delay. The tree-structure should still be considered since it can interpolate with an arbitrary number and all subfilters operate at the base rate which is not the case for multi-stage Nyquist interpolators.

Acknowledgments

I would like to thank my supervisor Ph.D. Amir Eghbali for inspiring me to do a master thesis in the digital filters area with his interesting lessons on the subject. His good explanations and motivation, to always make sure the students understand, has helped me during the master thesis. I would also like to thank Prof. Håkan Johansson for his contributions with his expertise and time. I will always be impressed by his great knowledge and ability to pass it on to me.

Linköping, May 2012
Jimmie Lahti

Contents

Notation	ix
1 Introduction	1
1.1 Motivation and Problem Formulation	1
1.2 Thesis Outline	2
2 Background	3
2.1 Transform Basics	3
2.2 Linear Time Invariant Systems	4
2.3 Filter Basics	5
2.4 FIR Filters	6
2.5 Linear-Phase Filters	7
2.6 Fractional Delay Filters	7
2.7 Interpolation	8
2.8 Multistage Interpolation	10
2.9 Decimation	10
2.10 Polyphase Decomposition	11
2.11 Nyquist (M th band) Filters	12
2.A Polyphase Decomposition	13
3 Computational Complexity and Delay	15
3.1 FIR Filters	15
3.2 Linear-Phase Filters	16
3.3 Fractional Delay Filters	16
3.4 Nyquist (M th Band) Filters	17
3.5 Combination of Linear-Phase and Nyquist (M th Band) Filters . . .	17
4 Filter Design	19
4.1 Optimization Using Fminimax	19
4.2 Optimization of a Non-Convex Problems Using Fminimax	20
5 Theory of Tree-Structured Nyquist Linear-Phase Interpolator/Decimator	23

5.1	Introduction	23
5.2	Compensation with Equalizer Filter	25
5.3	Tree-Structure	25
5.4	Tree-Structure Computational Complexity and Delay	26
6	Order Estimation	29
6.1	Fractional Delay Filters	29
6.2	Equalizer Filters	29
7	Designing Tree-Structured Nyquist Linear-Phase Interpolator/Decimator	41
7.1	Problem Formulation	41
7.2	Finding Initial Solution	42
7.3	Optimizing Filter on Global Level	45
7.4	Optimizing Considering Several Potential Filters	46
7.5	Analyzing Optimization Result	46
8	Results	47
8.1	Prime Numbers	47
8.2	Non-Prime Numbers	49
9	Conclusions And Future Work	53
	Bibliography	55

Notation

ACRONYMS

Notation	Explanation
LTI	Linear Time Invariant (system)
FIR	Finite Impulse Response (filter)
FD	Fractional Delay (filter)

DENOTATIONS

Notation	Explanation
ωT	frequency
$\omega_c T$	bandwidth frequency
$\Omega_c T$	region of frequency
ρ	excess bandwidth
$H(z)$	transfer function
$H(e^{j\omega T})$	transfer function
$h[n]$	discrete-time impulse response
$x[n]$	discrete-time input signal
$y[n]$	discrete-time output signal
$x_a(t)$	analog input signal

1

Introduction

1.1 Motivation and Problem Formulation

Today's communication systems are using interpolation and decimation, as conversion between different sampling rates, to make other signal processing algorithms work at their most beneficial sampling rate [Vaidyanathan, 1993]. The interpolation and decimation operations can be used to simplify analog-to-digital/digital-to-analog converters and to improve sound quality. During the past years, different interpolators and decimators have been introduced aiming to reduce the computational complexity. However, many of them address the problem with reconfigurable interpolators [Mahesh and Vinod, 2008] [Johansson, 2011]. This thesis introduces some improvements to the fixed types which demonstrates that there are still improvements to be made in this area of research.

Consider performing a L th polyphase decomposition on a simple interpolation by L . The result will be L branches of different fractional delay (FD) filters. This thesis introduces a new type of structure that implements the FD filters in an effective manner, the structure is called tree-structure. The tree-structured Nyquist interpolator/decimator, in this thesis, uses as the name suggests a tree-structure to produce the FD filters. The root nodes produce two FD filters which in steps are combined with each other to create the desired FD filters, obtained in the leaf nodes. The principle is an add and equalize principle where the FD filters are added together and the error introduced, by the addition, is equalized using filters, here after called equalizer filter.

The scope of the thesis is to investigate the pros and cons of the tree-structured Nyquist interpolator/decimator and compare it to the single-stage and multi-

stage alternatives. Another part of the thesis is to obtain a good order estimation of the equalizer filters which will be helpful when designing the tree-structured Nyquist interpolator/decimator. Important parameters when designing a filter, that both are included in the thesis, is the cost of implementation and the delay that affects the signal.

1.2 Thesis Outline

The thesis will in the first chapter go through the basics in filters and thereafter introduce some new concepts used in the tree-structured Nyquist interpolator. The decimation part is not included as it is simply the transposed operation of interpolation and hence the implementation is similar. Then, some basic tools will be introduced like the order estimation of equalizer filters. Thereafter, a more thorough chapter about how to design the tree-structured Nyquist interpolator will be presented. After that, the result and conclusion chapters will conclude the result in this thesis. The experienced reader in the filter field is recommended to skip the background chapter. However, fractional delay filters is not a conventional filter and, therefore might, require even the experienced reader to read.

2

Background

This chapter will give the basics of the filters discussed in this master thesis. The assumption is that the reader has a master of science degree or similar and is familiar with transforms, especially the z transform and discrete Fourier transform.

2.1 Transform Basics

For repetition purpose, the transforms with their respective inverses are defined below.

o Discrete Fourier transform:

$$X(e^{j\omega T}) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega T n},$$

where $x[n]$ is the time domain signal. The Fourier transform is used to transform a signal from the time domain to the frequency domain.

o Inverse discrete Fourier transform:

$$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega T}) e^{j\omega T n} d\omega,$$

where $X(e^{j\omega T})$ is the frequency domain signal. The inverse Fourier transform is used to transform a signal from the frequency domain to the time domain.

◦ Z transform:

$$X(z) = \sum_{n=-\infty}^{\infty} x[n]z^{-n},$$

where $x[n]$ is the discrete-time signal. The z transform is used to transform a signal from the discrete-time domain to the z domain.

◦ Inverse z transform:

$$x[n] = \frac{1}{j2\pi} \oint_C X(z)z^{n-1} dz,$$

where $X(z)$ is the z domain signal and C is a closed contour around origo and inside the region of convergence. The inverse z transform is used to transform a signal from z domain to discrete-time domain.

2.2 Linear Time Invariant Systems

An Linear Time Invariant (LTI) system is a mathematically described system but has many applications, in this case signal theory. The theory describes how a linear and time invariant system behaves. The linearity property is, if the input signal is multiplied with k , the output will be multiplied with k as well. Another linearity property is if several signals are added together the output signal will be the same as if the input signals were applied on the system separately and the output signals added together. The time invariant property is the fact that the output signal is not affected by the time the input signal is applied, the output signal is simply just delayed in the same manner as the input signal. The most appealing property of an LTI system is that the system can be characterized fully by obtaining the impulse response which is the answer of a dirac impulse. An LTI system can be described in the time domain using the impulse response, $h[n]$, or in the frequency domain using frequency response, $H(e^{j\omega T})$. In Fig. 2.1, an LTI system in discrete-time domain can be seen.

Figure 2.1: LTI system in time domain.

In the Fig. 2.1, $x[n]$ is the input, $h[n]$ is the impulse response and $y[n]$ is the output. The output signal is produced by convolution between the impulse response and the input signal.

Figure 2.2: LTI system in the frequency domain.

In Fig. 2.2, $X(e^{j\omega T})$ is the input, $H(e^{j\omega T})$ is the frequency response and $Y(e^{j\omega T})$ is the output. The output signal is obtained by multiplication between the frequency response and the signal in the frequency domain.

An LTI system is basically an addition of multiple delayed versions of the signal that is multiplied with a constant dependent on the delay. Due to this, it is possible to describe an LTI system using a difference equation $y[n] = \sum_{k=1}^N a_k x[n-k]$. The difference equation can be described by a convolution $y[n] = \sum_{k=1}^N h[k]x[n-k]$ which when transformed to the frequency domain, using the Fourier transform, can be seen as $Y(e^{j\omega T}) = H(e^{j\omega T})X(e^{j\omega T})$. This means that the spectrum out from the filter is the spectrum of the input signal multiplied with the filter spectrum. The spectrum function $H(e^{j\omega T})$ of a filter is called frequency response and describes what will be transferred to the output.

A property that LTI systems possess is time invariance which expresses how the system reacts in time. The time invariant property only means that the system will behave the same when ever the input signal is received. However the system can theoretically react before the input signal is received. This is called an anti-causal system. If the system reacts after the input signal has been received, the LTI system is a causal system. The third option of causality is non-causality which is when the system is a combination of a anti-causal system and causal system. If a system is not expressed in a causal manner, it can be described like one by inserting enough delays.

To describe the shape of the frequency response some terms are used. The amplitude is defined as $|H(e^{j\omega T})|$ and is how much the frequencies are scaled with. The phase is defined as $\arg(H(e^{j\omega T}))$ and is how much the frequencies are delayed when passing through the system. Both of these properties are important when designing filters.

There are mainly two types of digital filters Finite Impulse Response (FIR) and Infinite Impulse Response (IIR). The difference is how they produce the output signal. Where a FIR filter only produces the output signal using the input signal a IIR filter produces the output signal using the output signal as well. An IIR filter can be described using a difference equation $y[n] = \sum_{k=1}^{N_1} a_k x[n-k] + \sum_{k=1}^{N_2} b_k y[n-k]$, as viewable in the equation y is produced using the input signal and output signal. The FIR counterpart equation is $y[n] = \sum_{k=1}^N a_k x[n-k]$ and if compared to the IIR difference equation, it lacks the part where y is used.

2.3 Filter Basics

When designing a filter, there are some parameters to take into account. In this section the bandwidth, amplitude, attenuation, phase and the ripple will be explained. If we assume the specification for the filter is the same as the dashed line in Fig. 2.3, we want the amplitude to be 1 from $\omega T = 0$ to $\pi/2$ and after $\pi/2$, the amplitude should be 0. And ideally the phase, ϕ , should be 0 to not delay the signal. If it is not possible to make the phase 0, it should be linear so

the signal will keep the shape and hence only be delayed. The dilemma is that to make an ideal filter, we would need to make a non-causal filter which implies that the filter has to be able to acquire the signal from the future samples which is impossible. Therefore, it is good enough with a filter that has linear-phase for as high frequencies as it is needed, for example the required bandwidth, $\omega_c T$.

Figure 2.3: Example of result when designing a filter.

In Fig. 2.3, there is a solid line which indicates how the hypothetical designed filter became. As visualized with exaggeration in Fig. 2.3, it was not possible to obtain an ideal filter. There is, first of all, variations in amplitude called ripple. The vertical line at $\pi/2$ consists of a sloped curve. The attenuation at $\pi/2$ to π is not 0. The phase is not 0. It is however possible to make the variations from the ideal filter smaller by using a higher order filter, i.e., more calculations and higher cost. As shown above, it is not possible to create perfect filters, but it is possible to create a filter that is good enough.

By easing the requirement from the ideal case, some parameters are introduced. Starting from the left side, the first parameter that occur are δ_c that is the deviation from the desired amplitude also called ripple. The next parameter is $\omega_c T$ which is the passband edge that decides how wide the passband is. The grey marked area is the transition band and is how sharp the filter should be. The subsequent parameter is $\omega_s T$ and is the stopband edge which decides where the stopband starts. The last parameter is δ_s which is the stopband ripple which is related to the minimum attenuation as $A_{min} = -20 \log_{10}(\frac{1+\delta_c}{\delta_s})$.

2.4 FIR Filters

FIR filters are a class of filters that only use the input signal to produce the output signal, i.e., the impulse response will be of finite length. The transfer function $H(z)$ of a causal FIR filter of order N can be described by

$$H(z) = \sum_{n=0}^N h[n]z^{-n}. \quad (2.1)$$

Hence the output signal, if we assume that the input signal is $X(z)$, will be

$$Y(z) = H(z)X(z). \quad (2.2)$$

which can be transformed to the time domain using the inverse z transform

$$y[n] = \sum_{k=0}^N h[k]x[n-k]. \quad (2.3)$$

2.5 Linear-Phase Filters

Linear-phase filters are a special class of FIR filters which have, like the name suggests, linear-phase. There are 4 types of linear-phase filters but in this thesis only type I will be used so that is the focus in this section. Type I linear-phase filters have an even order, N , and are symmetric in the sense that the impulse response satisfies

$$h[n] = h[N - n]. \quad (2.4)$$

This implies, due to less freedom, that the order of the filter will be higher for the same amplitude requirements. At a closer look, it also implies that $N/2$ coefficients, i.e., multiplications will be the same. This will actually make it possible to save multiplications by implementing the filter carefully. However, the increased order will increase the delay which is also of importance.

Figure 2.4: Symmetric impulse response for a filter of order 4.

2.6 Fractional Delay Filters

FD filters are in this thesis a common FIR filter with the following specification. The amplitude should be constant 1. The phase should approximate $-j\omega T(d + N/2)$ where N is the filter order. This is not possible so to ease the requirement, it is only required for the filter to follow the specification when the frequency is below $\omega_c T$. A FD filter is a class of filters which is intended to delay the signal a fraction of a sampling interval [Välimäki and Laakso, 2000]. This is obtained by designing an FIR filter using the specification above.

Figure 2.5: FD filter in frequency domain (upper) and discrete-time domain (lower) with order N .

Figure 2.5 is an example of how the FD filter will affect the signal in the time domain and the frequency domain. Notice how the signal is delayed by $d + N/2$, however this is only true for frequencies below $\omega_c T$ but if $\omega_c T$ is selected to an appropriate value, the delay can be assumed to affect the whole signal.

Figure 2.6: FD filter in discrete-time domain approximating a delay of -0.25 .

To visualize the delay, Fig. 2.6 is provided where a hypothetical FD filter produce a delay of -0.25 samples. The sampled samples are the circles whereas the delayed samples are the squares. As viewable below, the samples are delayed by -0.25 samples resulting in a shift to the left. The negative delay seems anti-causal but when the delay $N/2$, made by the order N , is considered there is no doubt that this is a causal filter.

Figure 2.7 shows the frequency response of an FD filter approximating the delay 0.4 up to the bandwidth $\omega_c T = 0.6\pi$.

As viewable in Fig. 2.7, the delay approximates $-j\omega T(0.4 + N/2)$ until the bandwidth 0.6π . The amplitude approximates 1 until 0.6π . Notice how the phase ends with an integer of π at the maximum frequency π and deviates from the 0.4 delay dashed line at 0.6π .

2.7 Interpolation

Interpolation is the operation of producing new samples between existing samples on an underlying continuous signal marked as a solid line in Fig. 2.8. This is of interest in signal processing and communication. In Fig. 2.8, there is an

Figure 2.7: FD filter with 0.4 delay, 0.6π bandwidth and order 6.

original analog signal and the original sampling points marked as dots. These sampling points are the ones that were actually sampled by the ADC, analog-to-digital converter, and therefore their origin is from the actual signal. The purpose of interpolation is to produce the sample points marked as squares.

Figure 2.8: Time domain interpolation.

As almost always in signal theory, it is easier to look at the signal in the frequency domain. In Fig. 2.9, a block diagram is presented to explain the basics of interpolation. The input signal $x[n]$ is first upsampled by insertion of $L - 1$ zeros in between every original sample. Thereafter, the sample rate is L times faster than before. The insertion of zeros introduces some images in the spectrum that needs to be filtered out using the filter $H(z)$. After the filtering, the signal is an interpolated version of the original signal [Wanhammar and Johansson, 2011].

In Fig. 2.10 (a), a hypothetical signal is shown in the frequency domain and the signal after $L - 1$ zeros have been inserted in Fig. 2.10 (b). After inserting $L - 1$

Figure 2.9: Block diagram of interpolation.

zeros between every sample, there is as viewable in Fig. 2.10 (b), $L-1$ copies of the signal at higher frequencies. Those copies have to be filtered out and after that the interpolation is finished. The dashed line in the Fig. 2.10 (b) is a suggestion for how to filter out the extra spectrum copies.

Figure 2.10: Interpolation example in the frequency domain.

2.8 Multistage Interpolation

To be able to do a multistage interpolation, the interpolation factor needs to be divided into components. For example 15, can be divide into 3 and 5. Then, to achieve a interpolation by 15, the signal is first interpolated by 5 and then by 3 or vice versa. See Fig. 2.11 for example where $L = 5$ and $M = 3$. The property that the interpolation factor needs to be dividable can be rewritten as that the interpolation factor should not be a prime number.

Figure 2.11: Example of a 2-stage interpolator.

In Fig. 2.11, the signal is first upsampled by L and then the $L-1$ copies are filtered out. After that, the signal is upsampled by M and the $M-1$ copies are filtered. After these steps, the signal has been interpolated by $L \cdot M$ times using 2 steps.

2.9 Decimation

Decimation is the transposed operation of interpolation, meaning that samples will be removed. Due to the relation to interpolation, decimation can be done by

transposing the interpolation structure. The decimation operation will result in that the frequencies will be stretched by the decimation factor, resulting in that if the signal has frequencies higher than π/L , where L is the decimation factor, information will be lost.

In Fig. 2.12, a block diagram is presented to explain the basics of decimation. The input signal $x[n]$ is first filtered to avoid aliasing. The filtering does not change the sample rate. After the anti-aliasing filter, the signal is downsampled by removal of $L - 1$ samples. This operation will stretch the spectrum and the decimation is completed.

Figure 2.12: Block diagram of decimation.

In Fig. 2.13 (a), a hypothetical signal is shown in frequency domain and the signal after $L - 1$ samples have been removed in Fig. 2.13 (b). Before decimation can be done, the frequencies above π/L have to be filtered out to avoid aliasing. This is done by a filter $H(z)$ marked as a dashed line in the figure. Thereafter, the decimation can be done by removal of $L - 1$ samples. This operation will stretch the frequencies by L and thereafter the signal will become as in Fig. 2.13 (b).

Figure 2.13: Decimation example in the frequency domain.

2.10 Polyphase Decomposition

To do an L -polyphase decomposition from (2.1), the rewriting to a polyphase sum is

$$H(z) = \sum_{i=0}^{L-1} z^{-i} H_i(z^L). \quad (2.5)$$

See proof 2.A on page 13. Polyphase decomposition can be used to split up a filter, Fig. 2.14, into several parts see, Fig. 2.15. The parts will be allowed to operate at lowest possible sample rate which reduces the implementation cost [Eghbali, 2010].

Figure 2.14: Before polyphase decomposition.

The decomposition is made by separating the transfer function $H(z)$ into several transfer functions $H_1(z), H_2(z), \dots, H_L(z)$ and then adding the phases together using a commutator interleaver 2.15 (b) or by the method in 2.15 (a). A fact to notice is that all three figures are mathematical equivalents. The commutator is used in this thesis as in most practical implementations.

Figure 2.15: Example of polyphase decomposition.

When calculating the delay of polyphase decompositions, it is important to remember to multiply the branch delay by the decomposition factor N . However, this is just approximately since the branches can have different delays.

2.11 Nyquist (M th band) Filters

Nyquist filters are a class of filters for which the impulse response $h[n]$ at every M th coefficient is 0 and the center coefficient is $1/M$. The coefficients which are 0 will not need a multiplication and hence hardware will be saved.

By applying polyphase decomposition on the impulse response $h[n]$, the resulting filter, $H_0(z)$, for the 0th polyphase branch will become a constant. That is because the only coefficient for that polyphase branch will be $1/M$ and hence the output to that polyphase branch will be the signal multiplied by $1/M$.

Another property for Nyquist filters is that the passband ripple is dependent on the stopband ripple as $\delta_s \leq (M - 1)\delta_c$. This property is used in Chapter 7 to optimize the tree-structured Nyquist interpolator.

Appendix

2.A Polyphase Decomposition

This proof will show how to obtain (2.5) by performing polyphase decomposition on (2.1) on page 6.

Proof:

$$\begin{aligned} H(z) &= \sum_{n=0}^N h[n]z^{-n} \\ &= \sum_{n=-\infty}^{\infty} h[nN]z^{-nN} \\ &\quad + z^{-1} \sum_{n=-\infty}^{\infty} h[nN+1]z^{-nN} \\ &\quad \dots \\ &\quad + z^{-(N-1)} \sum_{n=-\infty}^{\infty} h[nN+N-1]z^{-nN}. \end{aligned}$$

This can be expressed in a more compact manner if $H_i(z)$ is defined as

$$H_i(z^N) = \sum_{n=-\infty}^{\infty} h[nN+i]z^{-nN}. \quad (2.6)$$

The sub transfer function $H_i(z)$ is composed of every N th coefficient in $h[n]$ with starting index i . Using the definition $H_i(z)$ the transfer function $H(z)$ can be rewritten as [Wanhammar and Johansson, 2011]

$$H(z) = \sum_{i=0}^{N-1} z^{-i} H_i(z^N). \quad (2.7)$$

The result above is equal to (2.5) which should be proved. $\square$

3

Computational Complexity and Delay

In this chapter, the computational complexity and delay of each filter type is discussed. The computational complexity of the filters is, for simplicity reasons, only counting the number of multiplications since that is the most expensive part. There are other components like additions and delays but they are almost always proportional to the number of multiplications.

3.1 FIR Filters

An N th-order FIR filter has $N + 1$ coefficients and therefore $N + 1$ multiplications, see Fig. 3.1. The number of additions is N and is equal to the number of delays as viewable in Fig.3.1.

Figure 3.1: Filter implementation with order N .

The z^{-1} objects are delays elements and the triangles are the multipliers. The circles with a “+” in are the adders. Notice how the adders and delays are proportional to the number of multiplications.

3.2 Linear-Phase Filters

Linear-phase filters of type I with order N have due to symmetric coefficients, $N/2 + 1$ multiplications. The additions are not affected by the symmetry, so the filter requires N adders. The linear-phase filter has a delay of $N/2$. The symmetry can be visualized in Fig. 3.2 by thinking that the coefficients a_0 and a_N are equal and a_1 and a_{N-1} are equal, etc. compared to Fig. 3.2

Figure 3.2: Linear-phase filter implementation with order N .

By moving half of the adders in front of the multipliers, addition of the inputs that use the same multiplication coefficient can be combined. This rearranging will make the number of multiplications divided by two. The other half of the adders are used after the multipliers and hence the number of adders does not change whereas the number of multipliers is divided by two.

3.3 Fractional Delay Filters

Fractional delay filters, designed using FIR filters, are not any different in cost than a regular FIR filter. Fortunately in the tree-structured interpolator, the structure uses two FD filters with the same FD but opposite sign and due to that they can be combined, see proof below, and therefore the two FD filters can be realized by 1 extra adder and half the multipliers compared to two FD filters. This is due to the fact that the two delays $e^{j\omega Td}$ and $e^{-j\omega Td}$ are complex conjugates. Since the FD filters, in this thesis, are designed by simple FIR filters, the delay for the FD filter is $N/2$ plus or minus the FD.

The following is a proof of how two FD filters with opposite FD can be combined to only require 1 extra adder and half of the multiplications compared to realizing both of the FD filters separately.

Proof: Let $F_1(z) = e^{-j\omega Td}$ and $F_2(z) = e^{j\omega Td}$. Since they are complex conjugate of each other, they can be written as

$$F_1(z) = F_2(z^{-1}) \leftrightarrow f_1(n) = f_2(-n).$$

This symmetry will make it possible to realize the two filters as a sum of two

linear-phase filters $C_1(z)$ and $C_2(z)$. Where $C_1(z)$ is a symmetric linear-phase filter and $C_2(z)$ is an anti-symmetric linear-phase filter. The filters $C_1(z)$ and $C_2(z)$ are given by

$$C_1(z) = \frac{F_1(z) + F_2(z)}{2}, \quad (3.1)$$

$$C_2(z) = \frac{F_1(z) - F_2(z)}{2}, \quad (3.2)$$

$$(3.3)$$

and by reversing it with some algebra, $F_1(z)$ and $F_2(z)$ can be realized by

$$F_1(z) = C_1(z) + C_2(z), \quad (3.4)$$

$$F_2(z) = C_1(z) - C_2(z). \quad (3.5)$$

$$(3.6)$$

Hence, the FD filter with opposite delay can be realized by doubling the amount of adders and one extra adder [Johansson et al., 2011].

□

3.4 Nyquist (M th Band) Filters

As explained in section 2.11 in the background chapter, Nyquist filters have every M th coefficients set to 0 except the center coefficient. This will make the cost calculation more complex than for a common FIR filter. The cost formula is $N + 1 - 2 \cdot \lfloor \frac{N}{2M} \rfloor$ which is less than for a common FIR filter. This is expected since the filter have every M th coefficient set to 0. The delay for a linear-phase Nyquist filter is $N/2$ which is the same as for a common linear-phase FIR filter.

3.5 Combination of Linear-Phase and Nyquist (M th Band) Filters

To obtain a formula for this combination of filter properties, the Nyquist filter formula will be analyzed. The linear-phase property will make every coefficient, except the center tap, to have a symmetric counterpart. This will basically reduce the coefficients in half. Therefore, to obtain a formula that combines the linear-phase and Nyquist filter properties, the terms where N is included have to be divided by two. Hence, the result is $N/2 + 1 - \lfloor \frac{N}{2M} \rfloor$.

In order to test the formula, two examples are provided below. For example if the filter is a half (2th) band LP filter and the filter order is 30, the half band makes every other coefficient except the center coefficient 0. And the symmetry due to linear-phase makes every other except the center coefficient equal. This reduces the multiplications to 9. By using the formula we get $30/2 + 1 - \lfloor \frac{30}{2 \cdot 2} \rfloor = 9$ which confirms the formula above.

And for example if the filter is a 6th band LP filter and the filter order is 20, the 6th band makes every 6th coefficient except the center coefficient 0. And the symmetry due to LP makes every other except the center coefficient equal. This reduces the multiplications to 10. By using the formula we get $20/2 + 1 - \lfloor \frac{20}{2 \cdot 6} \rfloor = 10$ which confirms the formula above.

4

Filter Design

In this thesis, filter design is done by minimax optimization. Other optimization methods are assumed to work as well, for example least-square method. Minimax criterion is that the maximum error from the desired frequency response is minimized.

4.1 Optimization Using Fminimax

When using fminimax some parameters are needed. There are mainly two parameters fminimax requires to perform the optimization, the error vector and the filter parameter vector, i.e., filter coefficient vector. To be able to explain it further some denotations are introduced.

Notation	Explanation
c	filter coefficient vector
ωT	frequency
Ω	set of frequencies to optimize
$D(e^{j\omega T})$	desired frequency response
$H(e^{j\omega T}, c)$	frequency response of c
$E(j\omega T)$	Error function

Table 4.1: Denotations

The frequency response of $H(e^{j\omega T}, c)$ will change when the parameters in c are altered. However the order of each filter is predeclared before starting the optimization, therefore it is desired to try different combinations of orders. By iter-

ating over the orders the static order problem is circumvented and the solution room is tested for solutions with different orders as well.

Using the denotations above, it is possible to define the error vector generating function as

$$E(j\omega T) = H(e^{j\omega T}, c) - D(e^{j\omega T}), \omega T \in \Omega, \quad (4.1)$$

where Ω is the optimization region, i.e., the region of interest in the frequency domain.

The task of fminimax is then to minimize the maximum of $|E(j\omega T)|$ when $\omega T \in \Omega$ by modifying the parameters in c . This is for a common FIR filter a simple task for fminimax since the problem is convex and therefore the optimal solution is found.

To obtain the filters in the background on page 3 the requirement on the desired function, i.e., amplitude and phase specifications should be met. Also the requirement on the impulse response for example symmetry of coefficients should be met for linear-phase filters.

4.2 Optimization of a Non-Convex Problems Using Fminimax

If the problem is non-convex, there is no guarantee that the global optimum will be found. Therefore, it is of great importance that a good initial solution is provided to fminimax. As will be understood in subsequent chapters, the tree-structured Nyquist interpolator has cascaded filters and is therefore a non-convex problem. However the tree-structured Nyquist interpolator has a mathematical solution that is used as initial state and therefore a optimum close to the global is assumed to be obtained.

Figure 4.1: Example of global and local optima.

If a poor initial state is given to fminimax, there is a risk that it will find a poor local optimum. For example, in the Fig. 4.1, if the initial state were to the left of

A , fminimax would find the local optimum. Instead if the initial state where to the right of A the global optimum will be found.

As will be understood in subsequent chapters, the tree-structured Nyquist interpolator consist of several steps of subfilters, where each step has its own specifications for the filters it contains. The parameter that can be varied the most is the order of the filters. The steps in the tree-structured Nyquist interpolator are cascaded and therefore the orders are as well. This will result in a very large solution room.

5

Theory of Tree-Structured Nyquist Linear-Phase Interpolator/Decimator

In this chapter, the theory behind the tree-structured Nyquist interpolator will be explained. It is good to have the Fourier- and z -transform recently repeated, see Section 2.1 on page 3 for repetition. The reason for not mentioning decimators is that it is basically the transposed operation of interpolation. See section 2.7 about interpolators and Section 2.9 about decimators for a proof.

5.1 Introduction

The tree-structured interpolator expressed using polyphase decomposition in a non-causal manner is the starting point in this explanation. However, a causal expression like in the background chapter is easily obtained by inserting delays. If M is the interpolation for an odd interpolator the transfer function can be expressed in polyphase form as

$$H(z) = \sum_{m=-(M-1)/2}^{(M-1)/2} z^{-m} H_m(z^M). \quad (5.1)$$

By looking at each polyphase branch and knowing that for a non-causal M th band filter, with transfer function as above, the frequency response of $z^{-m} H_m(z^M)$ must approximately satisfy, where the substitution $z = e^{j\omega T}$ is used,

$$e^{-j\omega T m} H_m(e^{j\omega T M}) = 1, \omega T \in [0, (1 - \rho)\pi/M], 0 < \rho < 1,$$

where ρ is the excess bandwidth [Saramäki, 1993]. The assumption that the pass-band energy is preserved by multiplying by M when interpolating is used. From

the equation above, it follows that $H_m(e^{j\omega T})$ must approximately satisfy

$$H_m(e^{j\omega T}) = e^{j\omega T m/M}, \omega T \in [0, (1 - \rho)\pi].$$

This means that the polyphase branches should approximate FD filters with delays of

$$m/M, m = -(M - 1)/2, -(M - 1)/2 - 1, \dots, (M - 1)/2. \quad (5.2)$$

Also note that for $m = 0$, there is the trivial case of $H_0(e^{j\omega T}) = 1$. To implement the interpolator, all the FD filters should be combined using (5.1).

For interpolation by an even M the same tree-structure can be used by using the polyphase decomposition

$$H(z) = \sum_{m=-M/2}^{M/2} w \cdot z^{-m} H_m(z^M), \quad (5.3)$$

where

$$w = \begin{cases} 1/2 & \text{when } |m| = M/2 \\ 1 & \text{otherwise} \end{cases}$$

With the polyphase decomposition, stated above, the polyphase branches can be derived for even order of M with similar method as for odd order of M .

The second fact behind the tree-structured Nyquist interpolator is that the average of two samples is a cheap estimation of the value in between them. This method however introduces some errors. For example, in the Fourier domain, the current sample is denoted $e^{-j\omega T 0}$ and the previous sample is denoted as $e^{-j\omega T 1}$. Adding together the samples gives

$$e^{-j\omega T 0} + e^{-j\omega T 1} = e^{-j\omega T (0+1)/2} \times 2 \cos\left(\frac{(0+1)\omega T}{2}\right), \quad (5.4)$$

where $2 \cos\left(\frac{(0+1)\omega T}{2}\right)$ is the introduced error. In the more general case when we have samples of delay d_1 and delay d_2 which in the Fourier domain is denoted $e^{-j\omega T d_1}$, respectively, $e^{-j\omega T d_2}$ the result of adding them together becomes

$$e^{-j\omega T d_1} + e^{-j\omega T d_2} = e^{-j\omega T (d_1+d_2)/2} \times 2 \cos\left(\frac{(d_1+d_2)\omega T}{2}\right), \quad (5.5)$$

where $2 \cos\left(\frac{(d_1+d_2)\omega T}{2}\right)$ is the introduced error. This suggest that we can obtain a delay of 1/4 and 3/4 using the result in (5.4) after compensation, i.e., phase 1/2 and the original delays of 0 and 1 that were used to produce the 1/2 delay. This can be done by using the delay 0 and 1/2 to produce the 1/4 delay and delay 1 and 1/2 to produce the 3/4 delay. By producing an arbitrary initial delay, with the help of a FD filter, the delays created can be arbitrary.

5.2 Compensation with Equalizer Filter

As stated in Section 5.1, adding together two FD filters give an error of $2 \cos\left(\frac{(d_1+d_2)\omega T}{2}\right)$ compared to the ideal case. This must be compensated after each addition. The solution is to use a filter that equalizes the error, or as called in this thesis, equalizer filter. The equalizer filter is designed by inverting the error hence the filter approximates

$$\frac{1}{2 \cos\left(\frac{(d_1+d_2)\omega T}{2}\right)}. \quad (5.6)$$

If we assume that we compensate the result in (5.5) with the filter above, the result will become

$$e^{-j\omega T(d_1+d_2)/2} \times 2 \cos\left(\frac{(d_1+d_2)\omega T}{2}\right) \times \frac{1}{2 \cos\left(\frac{(d_1+d_2)\omega T}{2}\right)} = e^{-j\omega T(d_1+d_2)/2}. \quad (5.7)$$

Hence the result is the expected FD filter with a delay of $(d_1 + d_2)/2$. A fact to remember though is that the filter approximates the inverse of the error so the compensation will not be perfect. And for the most extreme case, i.e., the result in (5.4) it is not even possible for the whole spectrum since at π it has to compensate an amplitude of 0, when the delays are related as $d_1 + d_2 = 0$.

5.3 Tree-Structure

To use the results in Sections 5.1 and 5.2 in a systematic manner, a tree-structure is introduced. The tree-structure is almost like a binary tree, see Fig. 5.1. As viewable in Fig. 5.1, the next FD filters are generated by using the FD filters in the previous step and thereafter compensated using the filters $G_i(z)$. To know how many steps, N , to use a simple formula is provided below. When the interpolation M is odd, the number of stages will be

$$N = \lceil \log_2(M - 1) \rceil - 1. \quad (5.8)$$

For the case when M is even, the following modification is made

$$N = \lceil \log_2(M) \rceil - 1. \quad (5.9)$$

The last part of designing the tree-structured interpolator is to select which delay the FD filters should have. The FD filters delay are selected as

$$d_{F_1} = 2^N/M, \quad (5.10)$$

$$d_{F_2} = -2^N/M, \quad (5.11)$$

for the upper respective lower FD filter.

In this part, the trivial delay 0 should not be forgotten. If for simplicity only the upper half of the tree-structure is considered, the result is as follows. The delay 0 and $2^N/M$ can be used to produce the delay $2^N/2M$. Delay $2^N/2M$ and 0

Figure 5.1: Tree-structured Nyquist interpolator for $M \in [6, 7, 8, 9]$.

can be used to produce the delay $2^N/4M$. Delay $2^N/2M$ and $2^N/M$ can be used to produce the delay $2^N 3/4M$. Now, if $M = 7$ then $N = 2$ as in Fig. 5.1 then the produced delays would have been 0 , $4/4M$, $4/2M$ and $4/M$. And with origin from delay 0 and $-2^N/M$, the corresponding negative delays can be produced in the same manner. The process of producing the new delays using the add and equalize principle, explained in previous sections.

In the figure above, the delays on the branches parallel to the filters are not shown. These delays are implicitly assumed to be there to compensate for the delay of $N/2$ the filters introduces. This introduces a constraint which is that the order, N , should be an even number for all sub filters. This is to make sure the delays parallel to the filters can be an integer since a non integer delay is out of question.

5.4 Tree-Structure Computational Complexity and Delay

In order to obtain the computational complexity and delay of the tree-structure a more comprehensive analysis has to be done than in the case with simple filters as in Chapter 3. Since the tree-structured Nyquist interpolator for the example where $M = 11$ only uses 11 of 17 branches in the tree-structure in Fig. 7.3 on page 44 some of the G_i filters are not needed. The filters not needed are the

most upper and lower G_3 filters. This is due to if $M = 11$ in (5.1) there will only be 11 branches. This must be taken into consideration when calculating the computational complexity. The multiplications required are the sum of the following: the order of the FD filters + 1, $2 \cdot (\text{the order of } G_1)/2 + 1$, $4 \cdot (\text{the order of } G_2)/2 + 1$ and $6 \cdot (\text{the order of } G_3)/2 + 1$.

Another important part is the delay. The delay is not as hard as computational complexity to calculate but the use of polyphase decomposition has to be considered. To obtain the delay of the tree-structured interpolator, add the delay of all steps together and multiply by the interpolation factor M : $M \cdot (\text{the order of the FD filters})/2 + (\text{the order of } G_1)/2 + (\text{the order of } G_2)/2 + (\text{the order of } G_3)/2$.

6

Order Estimation

The order estimations are used to select which orders to test for in each step in the tree-structure. A better order estimation will decrease the design time, which is preferable.

6.1 Fractional Delay Filters

The order estimation for the FD filters are solved with no elegance, just pure brute force. That is because in an upcoming thesis at Linköpings university, an order estimation formula will be presented. Therefore, no effort was put into making any good solution to this problem. However, to get a better starting order than 0, there is a method to estimate the order. The method is as follows, create a single-stage interpolator with the same specifications as for the tree-structured Nyquist interpolator that should be designed. The previous step gives an order, N_s , for the single-stage interpolator. Take the order, N_s , and divide it by the interpolation factor M . Hence the order of the FD filter, N_{FD} , can be estimated as $N_{FD} = N_s/M$. However, this is a simple estimation, so precautions in the order should be taken when using it. For example the orders tested could be 20% in deviation, from the obtained order.

6.2 Equalizer Filters

This section will describe how the formula to estimate the order of the equalizer filters were obtained and suggestions of how to use it. As stated in the previous

chapter the equalizer filters approximate

$$\frac{1}{2 \cos\left(\frac{(d_1+d_2)\omega T}{2}\right)}. \quad (6.1)$$

Hence after simplifications the following is obtained

$$\frac{1}{\cos\left(\frac{\omega T}{P}\right)}, \quad (6.2)$$

where the substitution $P = \frac{2}{(d_1+d_2)}$ and removal of scaling by 2 has been made to simplify. Since scaling of 2 is just a arithmetic right shift it can be neglected without consequence. The approximation above is hereafter used in this chapter when the equalizer filters are mentioned. Due to the nature of the tree-structured Nyquist linear-phase interpolator, P can only obtain values of 2 or larger. This is because the largest difference in delay between samples is 1 and for that extreme point $P = \frac{2}{(1+0)} = 2$ will be obtained. P has no upper limit since the delay between different samples can have arbitrary small difference depending on the interpolation factor. However to limit the problem the upper limit of 20 is introduced for P . The limit of 20 is selected due to the fact that for P larger than 20, the order will be close to 0 and therefore very easy to find without a formula. The bandwidth's lower limit where selected to 0.1π respective upper limit to 0.98π . The reason for not selecting the upper limit to π is that the order for the most extreme case of the equalizer filter become very large due to that it has to compensate for 0. The approximation error δ were selected to be 10^{-2} , 10^{-3} and 10^{-4} .

The order estimation for the equalizer filters are more of an analytical manner compared to doing a simple polynomial that is used in some papers. The analytical nature comes from that the dependency between the coefficients and P were found by guessing elemental functions to match the curves. This was done by an iterative method of guessing and trying to fit each coefficient on it's own and thereafter combining them to produce the formula below.

The result obtained by following the steps described below is as follows

$$N(P) = \frac{-2\pi A(P) \cdot \log_{10}(\delta)^{B(P)}}{(\pi - \omega_c T)^{\frac{1+C(P)}{\log_{10}(\delta)}}}, \quad (6.3)$$

$$A(P) = 2.2587 \cdot 10^3 \cdot e^{-5.4965 \cdot P} + \frac{1.2709}{P^{1.3455}}, \quad (6.4)$$

$$B(P) = 1.7131 \log_{10}(41.7886 \cdot P + 2.2540 \cdot 10^2) - 8.6818, \quad (6.5)$$

$$C(P) = 0.2162 \log_{10}(27.3875 \cdot P - 57.1356) - 0.3455. \quad (6.6)$$

The maximum error by using this formula is 1.2 for $2.1 \leq P \leq 20$ and $\omega_c T \leq 0.8\pi$. Due to the small error the margin for the order estimation can be as low as simply rounding the result up to the closest even number and adding 2 to the order obtained. As stated above the lower limit for P were 2 but it is not in the region in the final result. The reason for not including 2 in the region for P is that it, as

viewable in the figures of the constants below, has a big deviation from the other points and therefore will produce a large error compared to the maximum error 1.2 as is the case for $2.1 \leq P$. However, the formula above can be used for 2 as well, but then it is recommended to have a precaution margin of 10 for the order. The bandwidth region were originally 0.98π but when fitting the curve, fminimax produced an numerical error which was not possible to resolve. However this is not considered a big loss due to insights during the fitting process, that the error for the Type I order estimation formula for each P for frequencies around 0.9π is very large.

The first step were to produce all the data required to know what the order are, for each combination of the parameters. The parameters in (6.2) to sweep are P and the bandwidth $\omega_c T$. For this purpose a script was designed to sweep all parameters and store them. The sweeping was performed in the following region $2 \leq P \leq 20$ and $0.1\pi \leq \omega_c T \leq 0.98\pi$. The density used when sweeping were 0.1 for P and 0.01π for $\omega_c T$. After this step a list of the order N versus each combination of P and $\omega_c T$ were obtained. This resulted in approximately 16k grid points. To improve the speed of this multiple threads were used.

The second step were to use the common order estimation formula for Type I filters since the filters used in this thesis are Type I. The order estimation formula is showed below.

$$N = \frac{-2\pi A \cdot \log_{10}(\delta)^B}{(\pi - \omega_c T)^{\frac{1+C}{\log_{10}(\delta)}}}, \quad (6.7)$$

where A , B and C are unknown constants that will be found when solving the curve fitting with fminimax. As understood when inspecting the order estimation formula above, the deviation, δ , will affect the order in a $\log(x)$ manner and the bandwidth, ω_c , in a $1/x$ manner. This implies that δ will not be as important as ω_c . However this is not correct since δ is chosen to be varied in a 10^x way due to its relation to the attenuation and therefore could be considered to affect the order in a linear way.

As notable in the order estimation formula above, δ and ω_c are included in the formula. Compare these parameters to the equalizer filter specifications that includes δ , ω_c and P . The assumption is that δ and ω_c will be included by the order estimation formula and therefore the constants A , B and C will solely depend on P . The grid constructed by δ , ω_c and P where used to produce for each P a set of constants A_P , B_P and C_P by performing curve fitting using the formula above and fminimax. After iteration over all possible P were done, each P had three constants A_P , B_P and C_P to be used in further analysis in the next step.

The third step was to find out the relation between P and the set of constants A_P , B_P and C_P depending on P solely. This was performed by plotting each constant versus P . The difficult part was to estimate the analytical expression describing the relation between P and the constants. After a guess on an analytical function was done it was fitted to the plot using fminimax. This part in the process is an iterative step and several guesses were tested until the final result was obtained.

Below are the 3 constants and P with the final guessed function plotted as well.

Figure 6.1: The constants A .

In Fig. 6.1 the red/solid line are the A_p coefficients and the blue/dashed line is the approximation optimized for $2.1 \leq P \leq 10$. The final approximation is eq. 6.4 above.

Figure 6.2: The constants B .

In Fig. 6.2 the red/solid line are the B_p coefficients and the blue/dashed line is the approximation optimized for $2.1 \leq P \leq 10$. The final approximation is eq. 6.5 above.

In Fig. 6.3 the red/solid line are the C_p coefficients and the blue/dashed line is the approximation optimized for $2.1 \leq P \leq 10$. The final approximation is eq. 6.6 above.

Figure 6.3: The constants C .

The fourth step were to put it all together by inserting the expressions $A(P)$, $B(P)$ and $C(P)$ obtained in step three in the common Type 1 order formula as written in (6.7). After the insertion an overall optimization using fminimax were performed and the end result was found as can be seen in (6.3). The optimization had to be done using a smaller sub problem to ease the optimization for fminimax. The final result contains limits in the variables compared to the initial limits but still a satisfying result were found.

Appendix

The code below was used to produce the results above. The reason for including it is that it is a non-standard way of solving the curve fitting issue. The normal procedure is to solve it using a polynomial when the method used in this thesis are a more analytical method by fitting the curve to combinations of elemental functions.

Some of the code is commented but were used in certain steps in the curve fitting process. For example, the grid density was changed to ease the optimization. In matlab, blocks of code are separated by a %% whereas the single % is used for comments.

The first block loads the orders and sets the region of optimization. The second block until the fourth block is used to find an appropriate expression for the coefficients A, B and C versus P . The fifth block combines the 3 guessed expressions for A, B and C to create the universal expression as it is called in the code. The second block to the fifth block contains optimization with `fminimax` to modify the parameters to fit the expression to the curves. The sixth block of code will print some statistics including max error and average error.

```
1 %This file is used to get an idea of what function that could describe the
2 %coeff curves
3
4 global wcT_notpi_list;
5 global delta_c_des_list;
6 global P_list;
7
8 wcT_notpi_list = 0.8:-0.1:0.2; %OBS under 0.8 so svd must not contain nan or inf
9 delta_c_des_list = power(10,[-2 -3 -4]); %obs
10 P_list = [2.1:0.1:20]; %OBS skipping P=2
11
12
13 load('Est_data_all_dec2.mat','Est_order_all_dec','Est_h_all_dec' ...
14      , 'Est_Flg_all_dec', 'Est_plot_all_dec');
15
16
17
18
19 %%    coeff 1
```



```

20
21 %guess 1/(Ax) + B
22 [P, y] = plot_plot_coeff(P_list, Est_plot_all_dec, 1, false);
23
24
25 % %testing
26 % y_values = [];
27 % for i=1:length(P)
28 %     y_values = [y_values (1.2/P(i).^3 + 7)];
29 % end
30 % y=y_values;
31
32 % P = P(2:80);
33 % y = y(2:80);
34
35 %guessed_func1 = @(x, P)x(2)/P^x(3) + x(6); %almost good
36 guessed_func1 = @(x, P)x(2)*exp(-x(3)*P) + x(4)/P^x(6) + x(5); %good enough
37 %guessed_func1 = @(x, P)x(2)/P + x(6) + -1*log(P + x(5));
38 %guessed_func1 = @(x, P)x(2)/P^2 + x(3)/P + x(7);
39
40 %
41 %             TolCon = 1e-10 ; Iters = 1e5 ; Evals = 1e6 ; x1 = ones(1,8);
42 %             %x = x1;
43 % Options = optimset('Display','notify','TolX',TolCon,'TolCon',TolCon ...
44 %             'TolFun',TolCon,'TolConSQP',TolCon,'MaxIter',Iters,'MaxFunEvals',Evals,...
45 %             'TolConSQP',TolCon,'Diagnostics','off','FinDiffType','central' ...
46 %             'UseParallel','always','TypicalX',[]);
47 %
48 % x1 = fminimax(@(x)x(1),x1,[],[],[],[],[],[],@(x)coeff_total_error(x, P ...
49 %             , y, guessed_func1),Options) ; Params = [] ; Params = x(2:end);
50 %
51
52 figure(1);
53 plot(P,y, 'r');
54 hold on;
55 y_values = [];
56 for i=1:length(P)
57     y_values = [y_values guessed_func1(x1,P(i))];
58 end
59
60 plot(P, y_values, 'b');
61 hold off;
62
63 title('coeff_1');
64 max_error = max(abs(coeff_total_error(x1, P, y, guessed_func1)))
65
66
67 %% coeff 2
68
69 %guess 1/(Ax) + B
70 [P, y] = plot_plot_coeff(P_list, Est_plot_all_dec, 2, false);
71
72
73 % %testing
74 % y_values = [];
75 % for i=1:length(P)
76 %     y_values = [y_values (1.2/P(i).^3 + 7)];
77 % end

```

```

78 % y=y_values;
79 %
80 % P = P(2:80);
81 % y = y(2:80);
82
83 guessed_func2 = @(x, P)x(2)*log(x(3)*P + x(5)) + x(4); % good enough
84
85 % TolCon = 1e-10 ; Iters = 1e5 ; Evals = 1e6 ; x2 = rand(1,8);
86 % Options = optimset('Display','notify','TolX',TolCon,'TolCon',TolCon ...
87 % 'TolFun',TolCon,'TolConSQP',TolCon,'MaxIter',Iters,'MaxFunEvals',Evals,...
88 % 'TolConSQP',TolCon,'Diagnostics','off' ...
89 % 'FinDiffType','central','UseParallel','always','TypicalX',[]);
90 %
91 % x2 = fminimax(@(x)x(1),x2,[],[],[],[],[],[],[],@(x)coeff_total_error(x, P ...
92 % , y, guessed_func2),Options); Params = []; Params = x(2:end);
93
94 figure(2);
95 plot(P,y, 'r');
96 hold on;
97 y_values = [];
98 for i=1:length(P)
99     y_values = [y_values guessed_func2(x2,P(i))];
100 end
101
102 plot(P, y_values, 'b');
103 hold off;
104
105 title('coeff_2');
106 max_error = max(abs(coeff_total_error(x2, P, y, guessed_func2)))
107
108 %% coeff 3
109
110 %guess 1/(Ax) + B
111 [P, y] = plot_plot_coef(P_list, Est_plot_all_dec, 3, false);
112
113
114 % %testing
115 % y_values = [];
116 % for i=1:length(P)
117 %     y_values = [y_values (1.2/P(i).^3 + 7)];
118 % endx
119 % y=y_values;
120 %
121 % P = P(2:80);
122 % y = y(2:80);
123
124
125 % P = P(2:end);
126 % y = y(2:end);
127
128
129 %guessed_func3 = @(x, P)x(2)*(log(x(3)*P +x(5))) + x(8);
130 %guessed_func3 = @(x, P)x(2)*exp(x(3)*P +x(5)) + x(6);
131 %guessed_func3 = @(x, P)-x(2)/(x(3) + log(x(6)*P + x(4))) + x(5);
132 guessed_func3 = @(x, P)x(2)*log(x(3)*P + x(5)) + x(4); %good enough
133
134 %
135 % TolCon = 1e-10 ; Iters = 1e5 ; Evals = 1e6 ; x3 = rand(1,8);

```

```

136 % Options = optimset('Display','notify','TolX',TolCon,'TolCon',TolCon ...
137     , 'TolFun',TolCon,'TolConSQP',TolCon,'MaxIter',Iters,'MaxFunEvals',Evals,...
138 %     'TolConSQP',TolCon,'Diagnostics','off','FinDiffType','central' ...
139     , 'UseParallel','always','TypicalX',[]) ;
140 %
141 %
142 % x3 = fminimax(@(x)x(1),x3,[],[],[],[],[],[],@(x)coeff_total_error(x, P ...
143     , y, guessed_func3),Options) ; Params = [] ; Params = x(2:end);
144
145 figure(3);
146 plot(P,y, 'r');
147 hold on;
148 y_values = [];
149 for i=1:length(P)
150     y_values = [y_values guessed_func3(x3,P(i))];
151 end
152
153 plot(P, y_values, 'b');
154 hold off;
155
156 title('coeff_3');
157 max_error = max(abs(coeff_total_error(x3, P, y, guessed_func3)))
158
159
160
161
162
163 %% the universal formulas initial version
164
165 % make function that takes x's & P_list & Est_order_dec &
166 % guessed_funcs and outputs the difference
167 % name it universal_coeff_total_error
168 % then input it to fminimax and calc how good it is.
169 % make use of x1 x2 x3 for initial soloution!
170 length_x1 = length(x1);
171 length_x2 = length(x2);
172 length_x3 = length(x3);
173
174 x_univ = [x1 x2 x3];
175
176 guessed_func_univ = @(x, P, wcT, delta)2 * pi * guessed_func1(x(1:length_x1)...
177     , P) * ( (-log10(delta))^(guessed_func2(x( (length_x1+1):(length_x1+length_x2) )...
178     , P)) ) / ( (pi-wcT)^(1+guessed_func3( x( (length_x1+1+length_x2):...
179     (length_x1+length_x2+length_x3) ), P)/log10(delta));
180
181 [c, ceq] = universal_coeff_total_error(x_univ, P_list, wcT_notpi_list ...
182     , delta_c_des_list, Est_order_all_dec, guessed_func_univ, true);
183
184 x_univ_opt = x_univ;
185 %% (re)optimize
186 x = x_univ_opt;
187 %x = rand(1,24) + 0.1;
188
189 TolCon = 1e-5 ; Iters = 20 ; Evals = 1e6 ;
190 Options = optimset('Display','iter-detailed','TolX',TolCon,'TolCon',TolCon...
191     , 'TolFun',TolCon,'TolConSQP',TolCon,'MaxIter',Iters,'MaxFunEvals',Evals,...
192     'TolConSQP',TolCon,'Diagnostics','off','FinDiffType','central','UseParallel'...
193     , 'always','TypicalX',[]) ;

```

```

194
195 x_univ_opt = fminimax(@(x)x(1),x,[],[],[],[],[],[] ...
196 ,@(x)universal_coeff_total_error(x, P_list, wcT_notpi_list, delta_c_des_list...
197 , Est_order_all_dec, guessed_func_univ, false),Options) ;
198 Params = [] ; Params = x(2:end);
199
200 %% get result
201 % the true flag outputs error statistics (average, max ...)
202 [c, ceq] = universal_coeff_total_error(x_univ_opt, P_list, wcT_notpi_list,...
203     delta_c_des_list, Est_order_all_dec, guessed_func_univ, true);

```

The following code is the error calculation function used for optimization in `fminimax`. By setting a flag to true the function will print statistics about the error.

```

1  function [c, ceq] = universal_coeff_total_error(x, P_list, wcT_notpi_list, delta_c_des_
2      %function to guess parameters on
3      wcT_min = 0.6;
4      max_error = 0;
5      max_point = {};
6
7      avg_error = 0;
8      avg_count = 0;
9
10     avg_error_percent = 0;
11
12     max_N = 0;
13     max_value = 0;
14
15     f=x(1); c = []; ceq = [];
16     for delta_c = delta_c_des_list
17         for wcT_notpi = wcT_notpi_list(wcT_notpi_list > wcT_min)
18             P = P_list
19             %index
20             N = cell2mat( Est_order_all_dec( int64(wcT_notpi*100), int64(P*10), int6
21             value = funktion(x, P, wcT_notpi*pi, delta_c);
22             Error = value - N;
23             c = [c Error-f -Error-f];
24             %c = [c abs(Error-f)];
25
26             avg_error = avg_error + abs(Error);
27             if(not( N == 0))
28                 avg_error_percent = avg_error_percent + abs(Error)/N;
29             end
30             avg_count = avg_count + 1;
31
32             if(abs(Error) > max_error)
33                 max_error = abs(Error);
34                 max_point = {P, wcT_notpi, delta_c};
35                 max_N = abs(N);
36                 max_value = abs(value);
37             end
38             if(abs(value) > max_value)
39
40
41             end
42         end
43     end
44 end

```

```
45     if (output)
46         max_error
47         max_point
48         max_N
49         max_value
50
51         avg_value = avg_error/avg_count
52         avg_percent = avg_error_percent/avg_count
53     %         avg_error
54     %         avg_count
55     %         avg_error_percent
56     end
57 end
```


7

Designing Tree-Structured Nyquist Linear-Phase Interpolator/Decimator

In this chapter the details in designing the tree-structured Nyquist Linear-Phase interpolator will be explained. To clarify the details an example filter will be designed in the end of each section following the steps in the section.

7.1 Problem Formulation

Assuming that the interpolation factor, M , and the attenuation, A and the bandwidth, $\omega_c T$ are given. The tree-structure comes with a price, namely a special amplitude function that have bumps in the amplitude on π/M , $3\pi/M$, $5\pi/M$ and so on. The consequence of this is that it will be hard to make the stopband attenuated at the bumps or even impossible. If it is impossible to attenuate the bumps is not known since fminimax failed to optimize with such high orders that was tested. This gives two choices either increase the order to force the bumps down or introduce don't-care bands. Don't-care bands can be acceptable if the upsampled signals' frequencies does not overlay with the frequencies where the bumps are. Typically it is too expensive to just increase the order and therefore other filters are more suited if don't-care bands are not applicable. However if the attenuation is less than 40dB the bumps are not a big issue.

The two alternative stopbands are described in a mathematical manner as follows where Ω_1 is without don't care bands and Ω_2 is with don't-care bands.

$$\Omega_1 = [(1 + \rho)\pi/M, \pi], \quad (7.1)$$

$$\Omega_2 = \bigcup_{p=1}^{\lfloor M/2 \rfloor} \left[\frac{(2p-1+\rho)\pi}{M}, \min\left(\frac{(2p-1-\rho)\pi}{M}, \pi\right) \right], \quad (7.2)$$

where ρ is the excess bandwidth [Saramäki, 1993].

Since Nyquist filters have a property that the passband ripple is dependent on the stopband ripple there is no reason to optimize the passband region and therefore Ω_1 and Ω_2 are only including the stopband.

In Fig. 7.1 below the desired filter is plotted with a dashed line and Ω_2 is plotted as a solid line. The interpolation factor M , is assumed to be 11.

Figure 7.1: Specification for interpolation by 11 with don't-care bands.

As explained in the background chapter on page 8 in the section about interpolation. Interpolation by 11 will add 10 extra copies of the signal. This implies that the interpolation filter have to filter out all the frequencies above $\omega T > \pi/11$. As seen in Fig. 7.1 above the stopband starts at $\pi/11$ and goes all the way to π except for some don't-care bands. In the examples in the end of the following sections the attenuation is assumed to be 60dB and $\rho = 0.125$.

7.2 Finding Initial Solution

To find the initial solution, the result in Chapter 5 on page 23 is used. First the number of steps required are determined using (5.8) or (5.9) depending on whether M is odd or even. Subsequently the FD filters are designed with a delay using (5.10) and (5.11). The next step is to produce the equalizer filters as explained in the theory chapter in Section 5.2. The ideal frequency response can be produced using the formula

$$G_n(\omega T) = \frac{1}{2 \cos\left(\frac{(2^{(N-n)}/M)\omega T}{2}\right)}, \quad (7.3)$$

where n is the stage, M is the interpolation factor and N the number of steps as in Chapter 5.

When the filter specifications are done the next step is to find out which order to use. To obtain the orders, use the method in Chapter 6. However δ_c is a requirement that is yet to be decided. Therefore a table that was obtained designing filters for $M \in [5, 11, 12, 24]$ and $\delta_c \in [10^{-2}, 10^{-3}, 10^{-4}]$ is provided in Table 7.1.

initial δ_c	Attenuation after optimization
10^{-4}	90dB
10^{-3}	70dB
10^{-2}	50dB

Table 7.1: Table with practical rules to select δ_c for initial solution when $\rho = 0.125$

The practical rule is as expected since the attenuation is defined as $A = -20 \log_{10}(\delta_c)$ for a common filter. From the formula for attenuation it says that for every power of 10 the attenuation is decreased by 20dB. But this is before optimization and without cascaded filters as opposite to the tree-structure, so it has to be adjusted with a constant of 10 hence $A_p = -20 \log_{10}(\delta_c) + 10$. A_p gains 10dB compared to A because the optimization that is performed after the initial orders are selected and are purely for selecting δ_c . To ease the selection A_p is by algebra rewritten as, initial $\delta_c = 10^{-\frac{A_p - 10}{20}}$ which can be used to pick the initial δ_c .

Figure 7.2: Initial solution tree-structured Nyquist interpolator for $M = 11$, $\rho = 0.125$ and $\delta_c = 10^{-2.5} = 0.00316$.

Since ρ is related to the bandwidth it is assumed that it will be considered in the order formula indirectly via the bandwidth $\omega_c T$ and therefore only a table for $\rho = 0.125$ were produced.

Figure 7.3: Tree-structured Nyquist interpolator for $M \in [10, 11, 12, 13, 14, 15, 16, 17]$.

To conclude the information above the example from the section above is continued with the steps provided in this section. The number of steps N is calculated as $N = \lceil \log_2(11 - 1) \rceil - 1 = \lceil 3.3 \rceil - 1 = 3$. Hence the structure can be seen in Fig. 7.3.

In order to find δ_c the table 7.1 above is used and interpolated since the attenuation, 60dB, in this example is not in the table. By solving $60 = -20 \log_{10}(\delta_c) + 10$, δ_c is found to be $10^{-2.5}$. The subsequent step is to do as in Chapter 6 on page 30 to obtain the orders. The orders found are if the order for G_i is denoted N_i as following $N_1 = 14$, $N_2 = 8$, $N_3 = 4$. The order of the FD filter is $N_{FD} = 24$.

By creating all the filters separately and combining according to the tree-structure above and thereafter using polyphase composition to add all the different branches together the result in the Fig. 7.2 below were obtained.

As viewable in Fig. 7.2 the filter does not meet the specification. However this is not the optimal configuration as it is only the initial solution.

7.3 Optimizing Filter on Global Level

To optimize the tree-structured Nyquist interpolator a good initial solution is needed since it is a non-convex problem due to the cascaded filters. The initial solution were obtained following the steps in the previous section. The initial solution is a good solution so it is assumed to be close to the optimal. To globally optimize the tree-structured Nyquist interpolator it has to be parameterized. To parameterize there is a need to take into account that the G_i filters are symmetrical due to that they are linear-phase filters and that the FD filters are using the same coefficients. Taking this into consideration the number of parameters are the order + 1 for FD filters, $N_i/2 + 1$ for each step for the equalizer filters. Hence using the orders from the example in Section 7.2 above the total number of parameters for the example is $24 + 1 + 14/2 + 1 + 8/2 + 14/2 + 1 = 41$ these are inserted into the fminimax optimization as the variable c . See Chapter 4 for reference. The use of symmetry eases the optimization for fminimax to find the optimum by minimizing the parameters. The symmetry is the reason for the division by 2.

After global optimization, the interpolator will become as viewable in Fig. 7.4. Notice how the attenuation increased. There is no difference in order here compared to Fig. 7.2 above so global optimization is worth doing due to the large gain in attenuation.

Figure 7.4: Globally optimized tree-structured Nyquist interpolator for $M = 11$, $\rho = 0.125$ and $\delta_c = 10^{-2.5} = 0.00316$.

If global optimization did not meet the requirements, the suggested method is to increase the order manually or by inserting a lower δ_c into the order estimation step. Then repeat previous steps from Section 7.2 to this step.

7.4 Optimizing Considering Several Potential Filters

If the global optimization of the tree-structured Nyquist interpolator achieved greater attenuation than required there is a good reason to minimize the order, since minimizing the order is equal to saving cost. Since the initial solution is good enough the assumption that all the filters only need an order equal or less than the current order is used. To obtain a lower order a local search technique is used. For each order combination a initial solution is acquired as in the previous section. The brute force tests all possible combinations that have a lower order for each filter. All the filter combinations are saved in a list for later analysis.

In the example used above the orders were 24, 14, 8, 4. Hence there are $24/2 \cdot 14/2 \cdot 8/2 \cdot 4/2 = 672$ combinations since only even orders are considered.

7.5 Analyzing Optimization Result

The list of different filters produced in previous section will be analyzed in this step. The filters that have an attenuation less than the required are removed. At this point all filters in the list should meet the requirements. Thereafter sort the list by complexity in an ascending order. Pick the filter that has the lowest complexity. If several filters have the same complexity pick the one with lowest delay.

The filter with lowest complexity that meets the requirements are $N_1 = 10$, $N_2 = 6$, $N_3 = 2$. The order of the FD filter is 24. There are no other filters with the same complexity hence the filter is the best. The attenuation is 60.22dB and it uses 65 multiplications and have a delay of 231.

Figure 7.5: Final result of optimizing computational complexity, $M = 11$ and an attenuation of 60dB.

8

Results

This chapter contains design examples to prove the design effectiveness of the tree-structured Nyquist interpolator and also compares the result to those of the conventional interpolators [Saramäki, 1987]. Cases with both prime and non-prime numbers will be considered to show that the proposed structure can be used for either. Designing interpolators with the method in Chapter 7 on page 41 will be performed in this chapter.

8.1 Prime Numbers

In this section the two prime numbers 5 and 11 will be considered and the region Ω_2 with don't-care bands defined as (7.2).

If $M = 5$ and $\rho = 0.5$ and the attenuation requirement is 40dB the result is as in Fig. 8.1 below. The final result uses the orders 6 and 2, which will end up in 11 multiplications and a delay of 20. The regular one-stage interpolator uses 12 multiplications and has a delay of 14.

If $M = 11$ and $\rho = 0.125$ and the attenuation requirement is 40dB the result is as in Fig. 8.2 below. The final result uses the orders 14, 2, 2 and 0, which will end up in 33 multiplications and a delay of 99. The regular one-stage interpolator uses 50 multiplications and has a delay of 63.

Figure 8.1: Tree-structured Nyquist interpolator where $M = 5$ and $\rho = 0.5$ and attenuation is 60dB.

Figure 8.2: Tree-structured Nyquist interpolator where $M = 11$ and $\rho = 0.125$ and attenuation is 40dB.

8.2 Non-Prime Numbers

In this section the two numbers 12 and 24 will be considered and the region Ω_1 without don't-care bands defined as (7.1).

Figure 8.3: Tree-structured Nyquist interpolator where $M = 12$ and $\rho = 0.125$ and attenuation is 40dB.

If $M = 12$ and $\rho = 0.125$ and the attenuation requirement is 40dB the result is as in Fig. 8.3 below. The final result uses the orders 14, 4, 2 and 0, which will end up in 35 multiplications and a delay of 120. The regular one-stage interpolator uses 79 multiplications and has a delay of 85. The two-stage interpolator uses 28 multiplications and has a delay of 100. With a three-stage design the complexity can be reduced slightly.

Figure 8.4: Tree-structured Nyquist interpolator where $M = 24$ and $\rho = 0.125$ and attenuation is 40dB.

If $M = 24$ and $\rho = 0.125$ and the attenuation requirement is 40dB the result is as

in Fig. 8.4 below. The final result uses the orders 14, 4, 2, 0 and 0, which will end up in 47 multiplications and a delay of 240. The regular one-stage interpolator uses 164 multiplications and has a delay of 179. The two-stage interpolator use 47 multiplications and has a delay of 195. With a three-stage design the complexity can be reduced slightly.

As viewable in Table 8.1 the tree-structured Nyquist interpolator decreases the computational complexity slightly compared to the single-stage design. However this is expected since there is not much to gain since the orders are small. The interpolation factor, M , is a prime number and hence a multi-stage implementation is not applicable.

Type	Multiplicators	Delay
Single-stage	12	20
Tree-structured	11	20

Table 8.1: Table to summarize the result for $M = 5$, 60dB attenuation and $\rho = 0.5$ with don't-care bands.

As viewable in Table 8.2 the tree-structured Nyquist interpolator decreases the computational complexity compared to the single-stage design, however the delay increases. The interpolation factor, M , is a prime number and hence a multi-stage implementation is not applicable.

Type	Multiplicators	Delay
Single-stage	50	63
Tree-structured	33	99

Table 8.2: Table to summarize the result for $M = 11$, 40dB attenuation and $\rho = 0.125$ with don't-care bands.

As viewable in Table 8.3 the tree-structured Nyquist interpolator decreases the implementation cost compared to the single-stage design, however the delay increases. Since the interpolation factor is a non-prime number the two-stage implementation is applicable. The two-stage implementation has less computational complexity and less delay than the tree-structured Nyquist interpolator.

Type	Multiplicators	Delay
Single-stage	79	85
Two-stage	28	100
Tree-structured	35	120

Table 8.3: Table to summarize the result for $M = 12$, 40dB attenuation and $\rho = 0.125$ without don't-care bands.

As viewable in Table 8.4 the tree-structured Nyquist interpolator decreases the

implementation cost compared to the single-stage design, however the delay increases. Since the interpolation factor is a non-prime number the two-stage implementation is applicable. The two-stage implementation has the same computational complexity and less delay than the tree-structured Nyquist interpolator.

Type	Multiplicators	Delay
Single-stage	164	179
Two-stage	47	195
Tree-structured	47	240

Table 8.4: Table to summarize the result for $M = 24$, 40dB attenuation and $\rho = 0.125$ without don't-care bands.

In Tables 8.1 to 8.4, the results are summarized to ease the comparison between the different implementations. As seen the tree-structured Nyquist interpolator is comparable to the two-stage interpolator. The main advantage is that the tree-structured Nyquist interpolator can interpolate by a prime number as easily as a non-prime number.

9

Conclusions And Future Work

The tree-structured Nyquist interpolator is comparable to a multistage Nyquist interpolator but have slightly higher complexity and hence an alternative to consider. The introduced structure's main advantage is that it is equally effective to implement any interpolation factor, even prime numbers. Implementing prime numbers with a multistage interpolator is not possible since the interpolation factor should be possible to split up in steps. Another advantage is that all the filters run at the input sample rate. A multistage interpolator can however do likewise but then require polyphase decomposition, which for many steps can be troublesome to implement. As usual when decreasing computational complexity the delay increases which is also the case for the tree-structured interpolator, which can be seen in Chapter 8. The tree-structured Nyquist interpolator shows that it is possible to improve the fixed interpolators, however the fixed interpolators can be used as a component in the reconfigurable interpolators and therefore the tree-structured Nyquist interpolator will indirectly improve the reconfigurable interpolators as well.

Future work should include an analysis of the relation between the subfilters orders of each step and the final interpolation filter attenuation. In contradiction to the solution used now, this new insight could make it easier to pick the orders of the subfilters better and therefore save design time. Another optimization is that the current implementation goes through a lot of unnecessary combinations since it does not look at what the other combinations gave as solution. There is not a certain rule that might be used to filter out unnecessary combinations by looking at other combinations, but surely with some safety margin it could be done.

Possible optimization can be done by designing the equalizer filters using a band-

width for the phase as well. However doing this would probably require more freedom in parameters and hence the symmetry property of linear-phase filters might not be possible to use. Consider the equalizer filters, G_i , as designed today they have a bandwidth, $\omega_c T$, which is used to limit the optimization region for the amplitude. However the $\omega_c T$ could be used for the phase as well. By allowing more freedom when designing the equalizer filters and not forcing the coefficients to be symmetrical a lower order is expected to be obtained. However the symmetry would disappear and more multiplications are needed but the delay would decrease with the order. If it were possible to still use symmetry and do some tricks with the coefficients, like in a Nyquist filter, to only make the phase linear until $\omega_c T$, the savings could be large.

To gain more insight in how good the tree-structured Nyquist interpolator is, more comparisons versus the multi-stage interpolator should be done. By designing more multi-stage Nyquist interpolators and tree-structured Nyquist interpolators with the same specifications, a better comparison could be achieved as opposed to comparing the four examples provided in this thesis. To design multi-stage Nyquist interpolators using the fminimax method, more work than was within this master thesis scope would be needed. However the comparison could compare more filters with different specifications not already found in papers.

A solution to the don't-care band problem could be achieved by placing a filter in the 0-phase branch and hence making the tree-structure a non-Nyquist filter. By introducing this filter the cost would increase. An analytical analysis should be performed to understand in which step the filter should be inserted. But an initial guess is in step 0 since it then will affect all the other branches and the introduced filter might decrease the orders of the subsequent filters.

Bibliography

- A. Eghbali. *Contributions to Reconfigurable Filter Banks and Transmultiplexers*. Linköping University, 2010.
- H. Johansson. Farrow-structure-based reconfigurable bandpass linear-phase fir filters for integer sampling rate conversion. *IEEE Trans. Circuits Syst. II*, 58(1), 2011.
- H. Johansson, A. Eghbali, and J. Lahti. Tree-structured linear-phase nyquist fir filter interpolators and decimators. *IEEE Int. Symp. Circuits Syst.*, 2011.
- R. Mahesh and A. P. Vinod. Reconfigurable frequency response masking filters for software radio channelization. *IEEE Trans. Circuits Syst. II*, 55(3):274–278, 2008.
- T. Saramäki. A class of fir nyquist (nth band) filters with zero intersymbol interference. *IEEE Trans. Circuits Syst.*, CAS-34(10), 1987.
- T. Saramäki. *Handbook for Digital Signal Processing*. Wiley, 1993.
- P. P. Vaidyanathan. *Multirate systems and filter banks*. Prentice Hall, Inc., first edition, 1993.
- V. Välimäki and T. I. Laakso. Principles of fractional delay filters. *IEEE*, 2000.
- L. Wanhammar and H. Johansson. *Digital filters using matlab*. Linköping University, 2011.

Upphovsrätt

Detta dokument hålls tillgängligt på Internet — eller dess framtida ersättare — under 25 år från publiceringsdatum under förutsättning att inga extraordinära omständigheter uppstår.

Tillgång till dokumentet innebär tillstånd för var och en att läsa, ladda ner, skriva ut enstaka kopior för enskilt bruk och att använda det oförändrat för icke-kommersiell forskning och för undervisning. Överföring av upphovsrätten vid en senare tidpunkt kan inte upphäva detta tillstånd. All annan användning av dokumentet kräver upphovsmannens medgivande. För att garantera äktheten, säkerheten och tillgängligheten finns det lösningar av teknisk och administrativ art.

Upphovsmannens ideella rätt innefattar rätt att bli nämnd som upphovsman i den omfattning som god sed kräver vid användning av dokumentet på ovan beskrivna sätt samt skydd mot att dokumentet ändras eller presenteras i sådan form eller i sådant sammanhang som är kränkande för upphovsmannens litterära eller konstnärliga anseende eller egenart.

För ytterligare information om Linköping University Electronic Press se förlagets hemsida <http://www.ep.liu.se/>

Copyright

The publishers will keep this document online on the Internet — or its possible replacement — for a period of 25 years from the date of publication barring exceptional circumstances.

The online availability of the document implies a permanent permission for anyone to read, to download, to print out single copies for his/her own use and to use it unchanged for any non-commercial research and educational purpose. Subsequent transfers of copyright cannot revoke this permission. All other uses of the document are conditional on the consent of the copyright owner. The publisher has taken technical and administrative measures to assure authenticity, security and accessibility.

According to intellectual property law the author has the right to be mentioned when his/her work is accessed as described above and to be protected against infringement.

For additional information about the Linköping University Electronic Press and its procedures for publication and for assurance of document integrity, please refer to its www home page: <http://www.ep.liu.se/>