Universitetet
i Stavanger

FACULTY OF SCIENCE AND TECHNOLOGY

MASTER'S THESIS

Study program/specialization: Computer Engineering - Computing Technology	Spring semester, 2016 Open / ~~Confidential~~
Author: Kjell Le	………………………………………… (signature author)
Supervisor: Professor John Håkon Husøy	
Title of Master's Thesis: Investigation of Delayless Subband Adaptive Filters within the Unified Framework	
ECTS: 30	
Subject headings: Adaptive filter, Delayless subband adaptive filter, Signal processing.	Pages: 66 Stavanger, 15th June 2016

Investigation of Delayless Subband Adaptive Filters within the Unified Framework

Kjell Le
LKjell@gmail.com

June 2016

Abstract

Conventional subband adaptive filter (SAF) solved a complexity and convergence rate problem from long adaptive filter by doing adaptive filtering in subbands. The complexity is reduced because the subband adaptive filters have a much lower length and run parallel in a decimated rate. Faster convergence is achieved because each subband spectrum is stretch out making it more flatter, such that it resembles a white signal. Essentially we decorrelate the signal.

However, conventional SAF is plague with delay in the signal path. The delay is introduced mainly from the convolution of analysis and synthesis filter bank.

With the delayless subband adaptive filter (DSAF) introduced in 1995 by Morgan and Thi the delay is eliminated by doing a fullband adaptive filtering, but the (fullband) weight-update is done by the subband adaptive filters' weights through a weight transformation. We keep the benefit of convergence speed, but have increased the computational complexity.

DSAF comes in two varieties. An open loop version which resemble the initial problem with Wiener filter based adaptive filter algorithm such as the (N)LMS algorithm. This version does not converge to the true Wiener solution. The other variety is the closed loop version. This version converges to the true Wiener solution. However, the open loop is redundant because the closed loop requires less computation.

In this thesis we have derived an equation set which describe the closed loop version of DSAF. From a unified framework we can derive different adaptive filter algorithms within 3 steps. Therefore a proposition is to reverse these 3 steps to find the underlying equation set for the adaptive filter. Where we can investigate convergence properties from tools available from linear algebra.

A central part of DSAF is the weight transform which can be viewed as a reconstruction problem with a synthesis filter bank. We have therefore optimized the synthesis and analysis filters, with the help of the derived equation set, to gain better convergence speed.

Acknowledgement

I would like to thank my supervisor Jonh Håkon Husøy for a his great help and advice during these years at University of Stavanger. He has been an inspiration and helped me cope with the daunting discipline of signal processing. Without him I would have never been interested in signal processing. Nevertheless it has been an enjoying and challenging moment writing this thesis.

Kjell Le
Stavanger, June 2016

Contents

List of Abbreviations

APA	Affine projection algorithm
DC	Direct current
DFT	Discrete Fourier transform
DSAF	Delayless subband adaptive filter
FFT	Fast Fourier transform
FIR	Finite impulse response
gcd	Greatest common divisor
IDFT	Inverse DFT
LMS	Least mean squares
MSE	Mean squared error
NLMS	Normalized least mean squares
SAF	Subband adaptive filter
WSS	Wide-sense stationary

Chapter 1

Introduction and Background

The delayless subband adaptive filter (DSAF) is a filter bank architecture which was introduced in 1995 by Morgan and Thi [16] to solve a problem with delays in conventional subband adaptive filter (SAF) [6, 14]. The delay throughout the multirate system is introduced mainly from the convolution of analysis and synthesis filters. This delay is unwanted in some delay critical application such as echo cancellation and active noise control [14].

In this thesis we wish to describe the delayless subband adaptive filter within the unified framework as proposed in [10]. This framework simplify (or make it more complex) the train of thought to derive an adaptive filter scheme, which contains 3 steps.

1. An equation set.
2. A preconditioning strategy.
3. An estimation strategy.

These 3 steps will be clarified later.

Within the unified framework various adaptive algorithms can be derived. From simple algorithms such as LMS and NLMS to more sophisticated ones such as Pradhan Reddy subband adaptive filters (PRSAF) and transform domain adaptive filters (TDAF) [10].

The main idea is to reverse these 3 steps of the unified framework to establish an equation set for DSAF. With this equation set we will be able to describe the convergence in the mean for DSAF. Therefore we will be able to optimize parameters such that we gain faster convergence.

The Adaptive Filter

The bread and butter of an adaptive filter is the Wiener filter [8]. The Wiener filter is an optimum linear filter that minimize the mean squared error (MSE) between a desired signal $d(n)$ with the output of the filter, $\hat{d}(n)$. The input of the filter is an observable signal $x(n)$. The problem is illustrated in Figure 1.1.

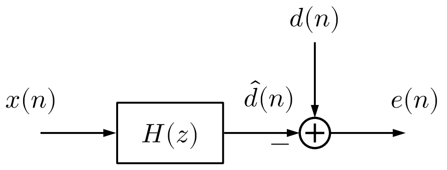


Figure 1.1: Wiener filter problem.

Given the input signal vector $\underline{x}(n) = [x(n), x(n-1), \dots, x(n-M+1)]^T$ and the FIR Wiener filter $\underline{h} = [h_0, h_1, \dots, h_{M-1}]^T$, where T denotes the transpose. Then from the orthogonal principle [8] it follows that

$$\mathrm{E}\left\{\underline{x}^*(n)e(n)\right\} = 0 \tag{1.1}$$

$$\mathrm{E}\left\{\underline{x}^*(n)[d(n) - \underline{x}^T(n)\underline{h}]\right\} = 0 \tag{1.2}$$

$$\underline{r}_{dx} - \mathbf{R}_x\underline{h} = 0 \tag{1.3}$$

where $*$ denotes the complex conjugate, $\underline{r}_{dx} = \mathrm{E}\left\{\underline{x}(n)d(n)\right\}$ is the crosscorrelation vector between $d(n)$ and $\underline{x}(n)$, and $\mathbf{R}_x = \mathrm{E}\left\{\underline{x}^*(n)\underline{x}^T(n)\right\}$ is the autocorrelation matrix of the input signal $x(n)$. $\underline{x}^T(n)\underline{h}$ is just another way to write the convolution since $\underline{h}$ is a FIR filter. The solution of the Wiener-Hopf equation (1.3) will be denoted as $\underline{h}_t$, the true Wiener solution

$$\underline{h}_t = \mathbf{R}_x^{-1}\underline{r}_{dx}. \tag{1.4}$$

From the Wiener-Hopf equation (1.3) we can create an adaptive filter scheme (e.g. LMS) by creating a Richardson iteration [17]

$$h(k+1) = \underline{h}(k) + \mu[\underline{r}_{dx} - \mathbf{R}_x\underline{h}(k)] \tag{1.5}$$

$$= [\mathbf{I} - \mu\mathbf{R}_x]\underline{h}(k) + \mu\underline{r}_{dx} \tag{1.6}$$

follow with point estimates of $\underline{r}_{dx}$ and $\mathbf{R}_x$

$$h(k+1) = \underline{h}(k) + \mu[d(k)\underline{x}^*(k) - \underline{x}^*(k)\underline{x}^T(k)\underline{h}(k)] \tag{1.7}$$

$$= \underline{h}(k) + \mu\underline{x}^*(k)e(k). \tag{1.8}$$

$[\mathbf{I} - \mu\mathbf{R}_x]$ is called the iteration matrix where convergence is solely dependent on. The iteration (1.8) is the LMS algorithm. μ is the step size which controls how fast the algorithm converges. LMS converges in the mean, with the independent assumption [8], when $0 < \mu < 2/\lambda_{max}$. λ_{max} is the largest eigenvalue of $\mathbf{R}_x$.

A high step size increases, to a certain point, the convergence rate. Though often with the trade off that the excessive MSE increases. The excessive MSE is the difference between the observable asymptotic error and minimum MSE, $e_{min}(k) = d(k) - \underline{h}_t^T\underline{x}(k)$. However, from literature [8, 20] we also know that high eigenvaluespread increases the convergence rate while low eigenvaluespread decreases. Fastest convergence rate corresponds to a white input signal.

The Unified Framework

In the previous section we have used the unified framework, as described in [10], to derive the LMS algorithm. There are 3 main steps in the unified framework [11]:

1. An equation set with its simplest iteration (Richardson iteration)

$$\mathbf{C}\mathbf{R}_x \underline{h}_t = \mathbf{C}\underline{r}_{dx} \tag{1.9}$$

$$\underline{h}(k+1) = \underline{h}(k) + \mu\mathbf{C}[\underline{r}_{dx} - \mathbf{R}_x\underline{h}(k)]. \tag{1.10}$$

2. A preconditioning strategy, i.e. the selection of $\mathbf{C}$.

3. An estimation strategy, i.e. estimation of $\mathbf{R}_x$ and $\mathbf{r}_{dx}$.

The LMS algorithm was derived with the selection $\mathbf{C} = \mathbf{I}$, the identity matrix, and point estimates of $\mathbf{R}_x$ and $\underline{r}_{dx}$.

NLMS can be derived as follows. We let $\mathbf{C} = [\epsilon\mathbf{I} + \mathbf{R}_x]^{-1}$ to be our preconditioning strategy. ϵ is a (small) regularization factor to ensure that $\mathbf{C}$ is invertible. The estimation strategy is the same as in LMS, i.e. point estimation. Let $\hat{\mathbf{C}} = [\epsilon\mathbf{I} + \underline{x}^*(k)\underline{x}^T(k)]^{-1}$ then our iteration can be written as

$$\begin{aligned}\underline{h}(k+1) &= \underline{h}(k) + \mu\hat{\mathbf{C}}[d(k)\underline{x}^*(k) - \underline{x}^*(k)\underline{x}^T(k)\underline{h}(k)] && (1.11)\\ &= \underline{h}(k) + \mu\hat{\mathbf{C}}x^*(k)e(k). && (1.12)\end{aligned}$$

To simplify further we use the Sherman-Morrison formula, a special case of Woodbury formula (matrix inversion identity [8, 14]),

$$[\mathbf{A} + \underline{u}\underline{v}^T]^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\underline{u}\underline{v}^T\mathbf{A}^{-1}}{1 + \underline{v}^T\mathbf{A}^{-1}\underline{u}}. \tag{1.13}$$

Applied on $\hat{\mathbf{C}}\underline{x}^*(k)$ yields

$$\hat{\mathbf{C}}\underline{x}^*(k) = \frac{1}{\epsilon + \|\underline{x}(k)\|_2^2}\underline{x}^*(k) \tag{1.14}$$

where $\|\cdot\|_2$ is the Euclidean norm. Hench the NLMS algorithm is derived,

$$\underline{h}(k+1) = \underline{h}(k) + \frac{\mu}{\epsilon + \|\underline{x}(k)\|_2^2}\underline{x}^*(k)e(k). \tag{1.15}$$

We ultimately wish to reverse these 3 steps to find an equation set which describe the convergence properties of DSAF in the mean.

Delayless Subband Adaptive Filter

There are 2 motivations for doing (conventional) subband adaptive filtering [6, 14]:

- Reduce computational complexity.
- Increase convergence rate.

The reduction of the computational complexity is because we can divide the fullband adaptive filter into smaller subband adaptive filters which run in parallel at a lower rate. Though there are some overhead in analysis and synthesis filter. The Increased convergence speed is due to frequency stretching when the signal is decimated in each subband. This leads to a flatter spectrum, i.e. decorrelate the subband signal.

Though there are problems such as aliasing and band-edge effect [14] which degrade the performance. Aliasing can be reduced with oversampled filter bank. This gives us a lower fullband minimum mean squared error. But oversampling creates band-edge effect in each subband, such that we get slower convergence.

With the introduction to DSAF by Morgan and Thi in 1995 [16] we have sacrificed a bit of the reduction of computational complexity of SAF to gain a delayless (subband) adaptive filter.

Figure 1.2 and 1.3 show the two configurations of DSAF. We perform fullband convolution similar to (fullband) LMS algorithm. The (fullband) weight-update is different. To do the weight-update we divide our signals into subbands. Do subband weight-updates with an adaptive filter algorithm. Perform a weight transform (frequency stacking + IDFT) from subband adaptive filter weights to fullband adaptive filter weights every M/J input samples. M is the fullband adaptive filter length, and $J = 1, 2, \ldots, 8$ depends on application [16]. For closed loop we need a high update rate because the delay of the error feedback will affect convergence. However, from simulations it appear that an update rate of every M input samples is suitable for stationary signal.

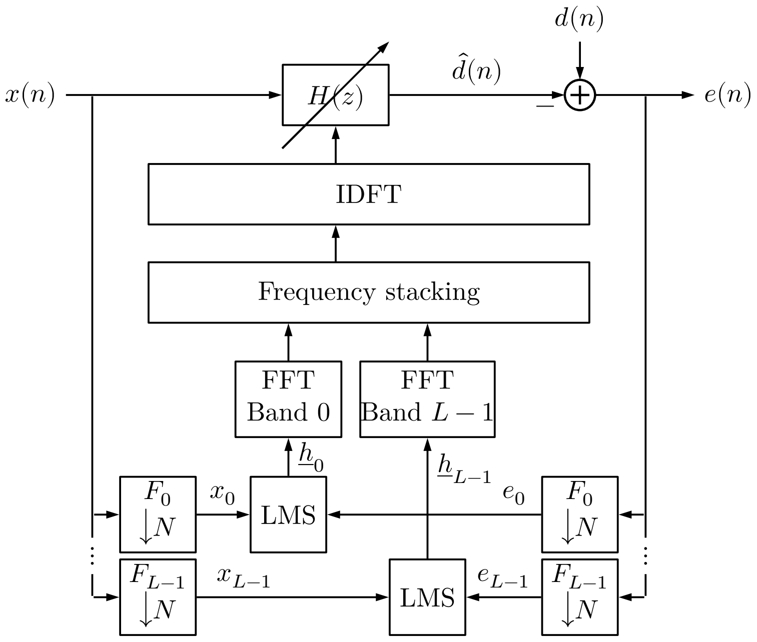


Figure 1.2: Delayless SAF with LMS, closed loop.

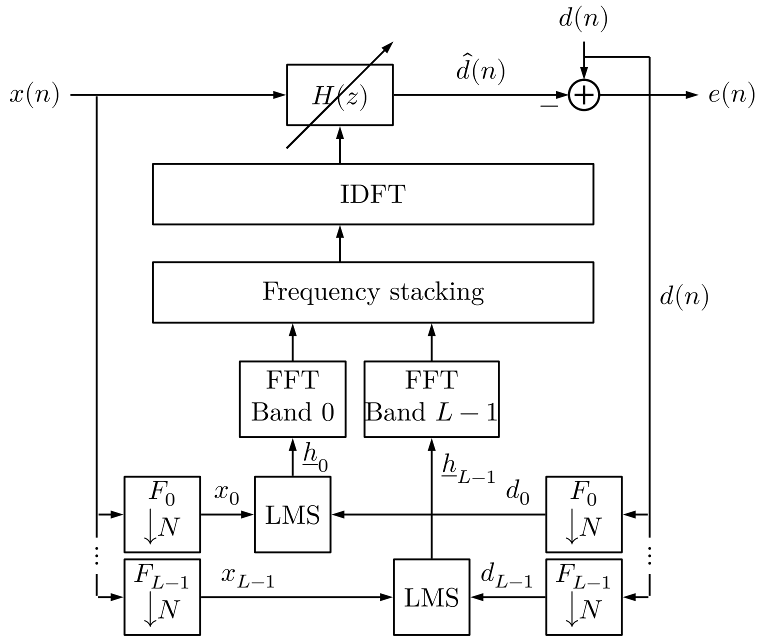


Figure 1.3: Delayless SAF with LMS, open loop.

Each LMS block is different on the two configurations. Basically we can exchange the LMS block with another adaptive filter algorithm such as NLMS. The differences in the LMS block is how the local subband error signal $e_i(k)$ is derived. k is time index after decimation. For closed loop we use the decimated fullband error signal directly as the local subband error. In open loop we need to calculate the local subband error signal with the fullband desired signal as

$$\hat{d}_i(k) = \underline{h}_i^T(k)\underline{x}_i(k) \tag{1.16}$$

$$e_i(k) = d_i(k) - \hat{d}_i(k). \tag{1.17}$$

Because of this extra computation, the open loop requires more computation than its counterpart. And since the open loop does not converge to the true Wiener solution we will primarily investigate the closed loop version of DSAF.

After obtaining the error signal we can do a subband weight-update with LMS algorithm, i.e.

$$\underline{h}_i(k+1) = \underline{h}_i(k) + e_i(k)\underline{x}_i^*(k) \tag{1.18}$$

for both configurations.

To explain the weight transform is a bit tricky. It can be viewed as a reconstruction problem with a synthesis filter bank [9]. The main idea in the original article [16] is to align the passband of each subband such that they form the fullband spectrum. Their weight transform, i.e. frequency stacking + IDFT, depends on the oversampling factor and filter modulation.

Throughout this thesis we shall use DFT modulated filter banks with an oversampling factor 2, i.e. $L = 2N$. L is the number of subbands and N is the decimation factor [1, 2].

With an ideal prototype filter $F_0(\omega_x)$ with cutoff frequency π/L the fullband spectrum can be divided as shown in Figure 1.4. ω_x is the frequency before decimation. The other bandpass filter $F_i(\omega_x)$ is a shifted version of the prototype filter.

$$F_i(\omega_x) = F_0\left(\omega_x - \frac{2\pi i}{L}\right), \quad i = 0, 1, 2, \ldots, L-1 \tag{1.19}$$

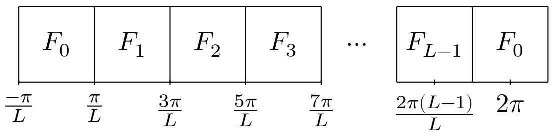


Figure 1.4: Fullband spectrum, ω_x.

Let us now look at a single subband branch as shown in Figure 1.5.

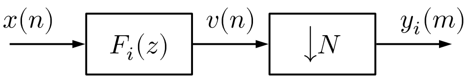


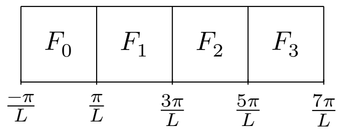
Figure 1.5: Bandpass signal decimation

We can assume that $x(n)$ is white noise, with unit variance, signal. The Fourier transform of $v(n)$ is then $V(\omega_x) = F_i(\omega_x)$. And the Fourier transform of $y_i(m)$ is

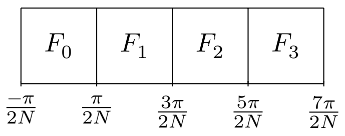
$$Y_i(\omega_y) = \frac{1}{N}\sum_{k=0}^{N-1} F_i\left(\frac{\omega_y - 2\pi k}{N}\right) \tag{1.20}$$

where ω_y is the frequency after decimation. The relationship between the frequencies is $\omega_y = N\omega_x$. Which means a frequency range from 0 to 2π in ω_y corresponding to a frequency range from 0 to $2\pi/N$ in ω_x. When $N \leq L$ then $Y_i(\omega_y)$ contains no aliasing and is periodic in 2π [20]. This means we can draw the spectrum as shown in Figure 1.4 with frequencies having the common denominator N. When we omit the common denominator N we get frequencies in ω_y. Since there are no aliasing when $N \leq L$ the resulting frequencies are the passband frequencies we wish to align.

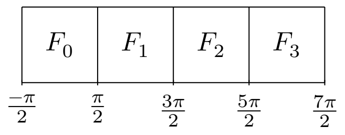
Let us look at an example for $L = 4$ subbands with a downsampling factor $N = 2$. Further we let the fullband adaptive tap be $M = 8$ such that each subband adaptive tap is $P_H = M/N = 4$. The adaptive filter length can be altered, but we will keep this ratio throughout the thesis. Drawing now as in Figure 1.4.



These are the fullband frequencies. We then substitute L with $2N$.



Omitting now N to get the passband frequencies in the subbands.



Thus the passband frequencies for each subband is as follows

$$F_0 : \left[\frac{-\pi}{2}, \frac{\pi}{2}\right] \tag{1.21}$$

$$F_1 : \left[\frac{\pi}{2}, \frac{3\pi}{2}\right] \tag{1.22}$$

$$F_2 : \left[\frac{3\pi}{2}, \frac{5\pi}{2}\right] \overset{periodic}{=} \left[\frac{-\pi}{2}, \frac{\pi}{2}\right] \tag{1.23}$$

$$F_3 : \left[\frac{5\pi}{2}, \frac{7\pi}{2}\right] \overset{periodic}{=} \left[\frac{\pi}{2}, \frac{3\pi}{2}\right] \tag{1.24}$$

Pausing for a moment, we can easily see and generalize that for even subbands the passband frequency range is between $-\pi/2$ and $\pi/2$, and for odd subbands the frequency range is between $\pi/2$ and $3\pi/2$. Therefore we need to pick out these frequency ranges for the respective subbands.

Since the subband adaptive tap is $P_H = 4$ we perform a 4 point DFT (DFT-4), and divide into 4 segments. The first segment corresponds to frequency range 0 to $\pi/2$. The second segment corresponds to frequency range $\pi/2$ to π etc.

Let now D_{ij} denote DFT segment $j = 1, 2, 3, 4$ of subband number i. Then we stack as follows

$$D_{01}, D_{12}, D_{13}, D_{24}, D_{21}, D_{32}, D_{33}, D_{04} \tag{1.25}$$

i.e. align passband sequentially such that they align as in fullband from frequency 0 to 2π. Finally we do M point DFT (DFT-8) on the sequence (1.25) to get the fullband adaptive filter weights. That's the gist of it.

Since subband 1 and 3 are conjugate of each other and their Fourier transform exhibit conjugate symmetry, we therefore only need to process $L/2 + 1$ subbands.

We can now carry out a fullband weight-update complexity analysis for closed loop. There are mainly 3 parts R_1 for analysis filter bank, R_2 for subband adaptive filtering and R_3 for weight transform.

Let now L_f be the analysis filter length. For an effective analysis polyphase implementation [3, 5] we need $L\log_2(L)$ real multiplication for (I)DFT, and L_f real multiplication for the convolution with the prototype filter. This is carried out in the decimated rate, thus for both signal $x(n)$ and $e(n)$ we have

$$R_1 = 2\frac{L\log_2(L)}{N} + 2\frac{L_f}{N} \tag{1.26}$$

real multiplication per input samples for the analysis part.

Using LMS in each subband yields P_H complex multiplication between $\underline{x}_i^*(k)$ and $e_i(k)$. We need to process $L/2+1$ subbands, but 2 of them are real subbands. Therefore the subband weight-updates require

$$R_2 = \frac{4P_H(L/2-1)+2P_H}{N} \tag{1.27}$$

real multiplication per input samples.

The weight transform need $(P_H/2)\log_2(P_H)$ complex multiplication for DFT in $L/2-1$ subbands, and $P_H \log_2(P_H)$ real multiplication for the other (real) subbands. The IDFT takes $M\log_2(M)$ real multiplication. But we only apply the weight transform every M/J input samples. Therefore we have

$$R_3 = \left[\frac{2P_H \log_2(P_H)(L/2+1)}{M} + \log_2(M)\right] J \tag{1.28}$$

real multiplication per input samples for the fullband weight transform.

If we would have used NLMS in each subbands the complexity does not increase much.

Chapter 2

Weight Transform Approaches

In this chapter we will look at different weight transform $\mathcal{G}$ which can be looked as a reconstruction problem with a synthesis filter bank [9]. The frequency stacking + IDFT in [16] for DSAF can be described as a transformation

$$\underline{h} = \mathcal{G}\underline{h}_s \tag{2.1}$$

where $\underline{h}_s$ is a vector composed of subband adaptive filters, i.e.

$$\underline{h}_s = [\underline{h}_0^T, \underline{h}_1^T, \dots, \underline{h}_{L-1}^T]^T. \tag{2.2}$$

A weight transform $\mathcal{G}$ can be decomposed as

$$\begin{aligned} \mathcal{G} &= \mathcal{G}\mathbf{I} && (2.3)\\ &= [\mathcal{G}\underline{e}_0, \mathcal{G}\underline{e}_1, \dots, \mathcal{G}\underline{e}_{P_H L-1}] && (2.4) \end{aligned}$$

where $\underline{e}_j$ is a standard basis vector column in the identity matrix of length $P_H L$. Thus feeding a weight transform algorithm with $\underline{e}_j$ yields us column j of $\mathcal{G}$.

An important notice is we need to use all subbands L and omit taking the real part or force it to be real at the end of an algorithm. i.e.

$$\underline{h} = \mathrm{Re}\,(\mathcal{G}\underline{h}_s) \tag{2.5}$$

If we take the real part it effectively means we are creating $\mathrm{Re}\,(\mathcal{G})$, which is something we do not wish to do. Though we do want to take the real part at some point since we need to ensure that $\underline{h}$ is real. In the following we will show the structure of $\mathcal{G}$ without going through the decomposition method.

FFT-1 Scheme

The FFT-1 is the original stacking scheme proposed by Morgan and Thi [16]. This stacking scheme was also explained in section 1.3 on page 7.

It is possible to view the stacking scheme as a weight transformation matrix [7]

$$\mathcal{G} = \mathbf{W}_M^{-1}\mathbf{S}_2\,\mathrm{diag}(\mathbf{W}_{P_H}, \dots, \mathbf{W}_{P_H}) \tag{2.6}$$

where $\mathbf{W}_n$ is a DFT matrix of dimension $n \times n$. M is the fullband adaptive filter length, and P_H is subband adaptive filter length. $\mathbf{S}_2$ is a 2 times oversampled stacking matrix for DFT filter banks,

$$\mathbf{S}_2 = \begin{bmatrix} \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & & & & & & & & & & & & & \\ & & & & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} & & & & & & & & & \\ & & & & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} & & & & & & & & & \\ & & & & & & & & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & & & & & \\ & & & & & & & & \mathbf{I} & \mathbf{0} & \mathbf{0} & \mathbf{0} & & & & & \\ & & & & & & & & & & & & \ddots & & & & \\ & & & & & & & & & & & & & \mathbf{0} & \mathbf{I} & \mathbf{0} & \mathbf{0} \\ & & & & & & & & & & & & & \mathbf{0} & \mathbf{0} & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{I} & & & & & & & & & & & & & \end{bmatrix} \tag{2.7}$$

where each block is of dimension $P_H/4 \times P_H/4$. For comparison, a critically sampled filter bank the stacking matrix would have been given by

$$\mathbf{S}_1 = \begin{bmatrix} \mathbf{I} & \mathbf{0} & & & & & \\ & & \mathbf{0} & \mathbf{I} & & & \\ & & \mathbf{I} & \mathbf{0} & & & \\ & & & & \ddots & & \\ & & & & & \mathbf{0} & \mathbf{I} \\ & & & & & \mathbf{I} & \mathbf{0} \\ \mathbf{0} & \mathbf{I} & & & & & \end{bmatrix} \tag{2.8}$$

where the dimension of each block is now $P_H/2 \times P_H/2$.

However, Morgan and Thi [16], and Huo et al. [9] suppress the coefficient corresponding to frequency π and force the spectrum to be conjugated symmetric, i.e. force $\underline{h}$ to be real.

> ... to form points 0-255 of a 512-point array. The array is then completed by setting point 256 equal to zero and using the complex conjugate of points 1-255 in reverse order.[16]

If we utilize all subbands and align the passbands sequentially as shown in the stacking matrix we do not get a conjugated symmetric spectrum, e.g. points 257-511 are not conjugate symmetric to points 1-255 in the above quote. Hence our $\mathcal{G}$ matrix does not corresponds exactly to their method. However, the differences in simulation are meager for lowpass signal. For highpass signal it is a different story since they suppress the high frequency.

FFT-2 Scheme

According to [9] the FFT-1 introduces zeroes in the spectrum of the fullband adaptive filter at frequencies $\frac{(2k+1)\pi}{M}, k \in \mathbb{N}$. The FFT-2 scheme does not have this deficiency, but its implementation is very similar to FFT-1. The weight transform can now be written as [7]

$$\mathcal{G} = [\mathbf{I}_M, \mathbf{0}_M]\mathbf{W}_{2M}^{-1}\mathbf{S}_2 \operatorname{diag}(\mathbf{W}_{2P_H}\mathbf{P}, \dots, \mathbf{W}_{2P_H}\mathbf{P}) \tag{2.9}$$

$$\mathbf{P} = \begin{bmatrix} \mathbf{I}_{P_H} \\ \mathbf{0}_{P_H} \end{bmatrix} \tag{2.10}$$

each block in $\mathbf{S}_2$ has dimension $P_H/2 \times P_H/2$, doubled of what we had in FFT-1 scheme. What we basically do is perform DFT-$2P_H$ on each subband. Stack the frequency as in FFT-1 scheme. Perform IDFT-$2M$ and discard the last M samples. Note that the lowest P_H we can have is 2 for 2 times oversampled filter bank.

The computational cost evaluation is $P_H \log_2(2P_H)$ complex multiplication for DFT in $L/2-1$ subbands and $2P_H \log_2(2P_H)$ real multiplication for the 2 real subbands, and another $2M\log_2(2M)$ real multiplication for IDFT. The total cost is then

$$R_3 = \left[\frac{4P_H \log_2(2P_H)(L/2+1)}{M} + 2\log_2(2M)\right] J \tag{2.11}$$

real multiplication per input samples. Recall we only need to update the weights every M/J samples.

DFT-FIR Scheme

Since both FFT-1 and FFT-2 scheme can be viewed as a reconstruction problem with a synthesis filter bank. We can therefore generalize and construct a weight transform from a synthesis filter bank with arbitrary synthesis filters. However, in [9] there is a section they describe which is difficult to completely and precisely decipher:

> A Q tap linear phase FIR filter can be used to approximate the ideal lowpass prototype filter $F_0(\omega)$. By doing so, an initial extra delay of $(Q-1)/2$ samples is added to the fullband filter impulse response. This extra delay can be removed by starting the convolution at sample $(Q-1)/2$. [9]

With the quote above it is reasonable to believe the equivalent to remove the delay is to omit the first $(Q-1)/2$ samples from the output. Though, this does not work. Empirically, convergence was observed for DFT filter banks when we omit $Q/2$ samples for even $Q = 2KL$, $K = 1, 2, 3, \ldots$. Generally for $Q = 2KL + n$ we had to remove $(Q-n)/2$ i.e. remove KL samples to observe convergence. $|n|$ is a small number. A greater $|n|$ may converge at the expense of convergence speed and excessive mean squared error. We will refer this delay removal as L_g offset. Where L_g is the length of the synthesis filters.

Implementation of DFT-FIR

An effective way to implement the synthesis filter bank is through polyphase decomposition. Polyphase decomposition makes it easier to decouple the analysis and synthesis filter bank from each other. Especially synthesis part is much easier to implement because we do all the filtering at lower rate.

For oversampled filter bank the type II polyphase matrix $\mathbf{R}(z)$ is of dimension $N \times L$ [1]. L is number of subbands and N is the downsampling factor. With the polyphase decomposition of synthesis filters $G_k(z)$

$$G_k(z) = \sum_{i=0}^{N-1} z^{-(N-1-i)} R_{i,k}(z^N) \tag{2.12}$$

where

$$R_{i,k}(z) = \sum_{n=-\infty}^{\infty} g_k(nN + N - 1 - i)z^{-n} \tag{2.13}$$

the type II polyphase matrix is given by

$$\mathbf{R}(z^N) = \begin{bmatrix} R_{0,0}(z^N) & R_{0,1}(z^N) & ... & R_{0,L-1}(z^N) \\ R_{1,0}(z^N) & R_{1,1}(z^N) & ... & R_{1,L-1}(z^N) \\ \vdots & \vdots & & \vdots \\ R_{N-1,0}(z^N) & R_{N-1,1}(z^N) & ... & R_{N-1,L-1}(z^N) \end{bmatrix} \tag{2.14}$$

i.e. the polyphase components of filter $G_k(z)$ is in column k. Another way to construct the type II polyphase matrix is to first construct a type I polyphase matrix $\mathbf{E}(z)$ then use the identity [5]

$$\mathbf{R}(z) = \mathbf{J}\mathbf{E}^T(z) \tag{2.15}$$

where $\mathbf{J}$ is an anti-diagonal matrix with ones, i.e. we flip the rows of $\mathbf{E}^T(z)$ up down. For polyphase decomposition type I

$$G_k(z) = \sum_{j=0}^{N-1} z^{-j} E_{k,j}(z^N) \tag{2.16}$$

the polyphase matrix is given by

$$\mathbf{E}(z^N) = \begin{bmatrix} E_{0,0}(z^N) & E_{0,1}(z^N) & ... & E_{0,N-1}(z^N) \\ E_{1,0}(z^N) & E_{1,1}(z^N) & ... & E_{1,N-1}(z^N) \\ \vdots & \vdots & & \vdots \\ E_{L-1,0}(z^N) & E_{L-1,1}(z^N) & ... & E_{L-1,N-1}(z^N) \end{bmatrix} \tag{2.17}$$

i.e. the polyphase components of filter $G_k(z)$ is in row k. And the polyphase component is given by

$$E_{k,j} = \sum_{n=-\infty}^{\infty} g_k(nN + j)z^{-n} \tag{2.18}$$

The polyphase components between type I and II are simple a permutation of each other, i.e.

$$R_{l,k}(z) = E_{k,N-1-l}(z). \tag{2.19}$$

This polyphase decomposition will work fine when we already have modulated filters. However, for DFT filter banks we can construct even more effective with polyphase design in [3, 5]. It follows we can decomposed the polyphase matrix $\mathbf{E}(z)$ as

$$\mathbf{E}(z) = \mathbf{W}^*\mathbf{B}(z) \tag{2.20}$$

where $\mathbf{W}$ is the DFT matrix. Given

$$n = \gcd(L, N) \tag{2.21}$$

$$K = LN/n \tag{2.22}$$

$$j + qN \equiv i \pmod{L} \tag{2.23}$$

$$q \leq K/N - 1 \tag{2.24}$$

$$J = L/n \tag{2.25}$$

elements of $\mathbf{B}(z)$ are generated by

$$[\mathbf{B}(z)]_{i,j} = \begin{cases} z^{-q}E_{j+qN}(z^J) & \text{if } (i-j) \equiv 0 \pmod{n} \\ 0 & \text{else} \end{cases} \tag{2.26}$$

where E_k is a polyphase component of K polyphase decomposition of the prototype filter, i.e.

$$G_0(z) = \sum_{k=0}^{K-1} z^{-k}E_k(z^K). \tag{2.27}$$

A way to interpret elements of $\mathbf{B}(z)$ is that z^{-q} specifies how many q zeroes to add before the polyphase component, and z^J specifies how many $J-1$ zeroes to add between the samples. After constructing $\mathbf{B}(z)$ we use equation (2.15) to get the polyphase matrix type II

$$\mathbf{R}(z) = \mathbf{J}\mathbf{E}^T(z) \tag{2.28}$$
$$= \mathbf{J}[\mathbf{W}^*\mathbf{B}(z)]^T \tag{2.29}$$
$$= \mathbf{J}\mathbf{B}^T(z)\mathbf{W}^H \tag{2.30}$$
$$= \mathbf{J}\mathbf{B}^T(z)\mathbf{W}^*. \tag{2.31}$$

The synthesis filter bank can then be constructed as shown in Figure 2.1.

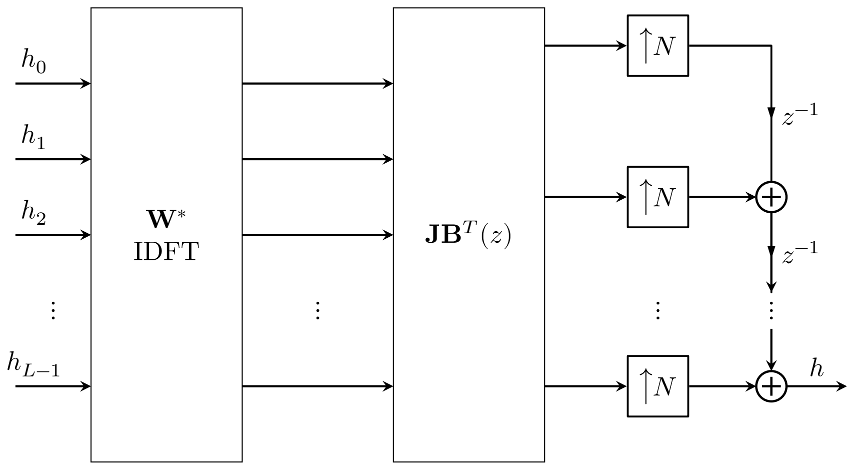


Figure 2.1: Synthesis filter bank

The complexity cost is $L\log_2(L)$ real multiplication for the real IDFT plus L_g real multiplication for convolution with the prototype filter g with length L_g. Since we need to run through P_H times and update every M/J samples ($J = 1, \dots, 8$), the total cost is then

$$R_3 = J\frac{P_H(L\log_2(L) + L_g)}{M} \tag{2.32}$$

real multiplication per input samples.

Construction of Weight Transform

Since the weight transform can be viewed as a reconstruction problem with a synthesis filter bank, we will now try to construct $\mathcal{G}$ directly from the synthesis filters. We will first try to find a corresponding analysis matrix then use (2.15) to convert to a synthesis matrix.

Let us define $\underline{x}_i(k) \triangleq \mathcal{F}_i^T \underline{x}(kN) = [x_i(k), x_i(k-1), \dots, x_i(k-P_H+1)]^T$, the decimated input signal of subband i. The fullband signal $\underline{x}(kN)$ is given by

$$\begin{bmatrix} x(kN) \\ x(kN-1) \\ x(kN-2) \\ \vdots \\ x(kN-(L_x-1)) \end{bmatrix} \tag{2.33}$$

where $L_x = L_f + (P_H - 1)N$. And $\mathcal{F}_i^T$ is given by

$$\mathcal{F}_i^T = \begin{bmatrix} \underline{f}_i^T & \underline{0}^T & & \\ \underline{0}_N^T & \underline{f}_i^T & \underline{0}^T & \\ \underline{0}_N^T & \underline{0}_N^T & \underline{f}_i^T & \underline{0}^T \\ \vdots & \vdots & \ddots & \\ \underline{0}_N^T & \underline{0}_N^T & \cdots & \underline{f}_i^T \end{bmatrix} \tag{2.34}$$

where $\underline{0}_N^T$ is a zero row vector of N elements. The dimension of $\mathcal{F}_i^T$ is $P_H \times L_x$. We can verify this by direct computation.

Stacking now all subband signals such that we have

$$\begin{bmatrix} \underline{x}_0(k) \\ \underline{x}_1(k) \\ \vdots \\ \underline{x}_{L-1}(k) \end{bmatrix} = \begin{bmatrix} \mathcal{F}_0^T \\ \mathcal{F}_1^T \\ \vdots \\ \mathcal{F}_{L-1}^T \end{bmatrix} \underline{x}(kN). \tag{2.35}$$

If we look at polyphase domain then

$$\begin{bmatrix} \mathcal{F}_0^T \\ \mathcal{F}_1^T \\ \vdots \\ \mathcal{F}_{L-1}^T \end{bmatrix} \tag{2.36}$$

corresponds to the polyphase matrix type I. With (2.15) we can turn this into a corresponding synthesis matrix, i.e. transpose and row reverse. But then the output signal would be row reversed therefore we omit the row reverse.

Our $\mathcal{G}$ matrix is therefore given by

$$\mathcal{G} = [\mathbf{0}_{L_g offset}, \mathbf{I}_M, \mathbf{0}][\mathcal{G}_0, \mathcal{G}_1, \dots, \mathcal{G}_{L-1}] \tag{2.37}$$

where $\mathcal{G}_i$ has the same structure as $\mathcal{F}_i$ with synthesis filter instead of analysis filter. What we have done is to omit L_g offset (delay) rows and removed the bottom rows such that we get overall M rows.

However, since we only need to process $L/2+1$ subbands we can exploit the conjugate symmetry of the subbands to create a half $\mathcal{G}$. Let's redefine

$\mathcal{G} = [\mathcal{G}_0, \mathcal{G}_1, \dots, \mathcal{G}_{L-1}]$, where each $\mathcal{G}_i$ has dimension $M \times P_H$, i.e. we have trunked the original $\mathcal{G}_i$. With

$$\underline{h}_{s,half} = [\underline{h}_0^T, \underline{h}_1^T, \dots, \underline{h}_{L/2}^T]^T \tag{2.38}$$

and define the new half weight transform as

$$\mathcal{G}_{half} = [\mathbf{G}_0, \mathbf{G}_1, \dots, \mathbf{G}_{L/2}] \tag{2.39}$$

where

$$\mathbf{G}_i = \begin{cases} \mathcal{G}_i + \mathcal{G}_{L-i}^* & i = 1, 2, \dots, L/2 - 1 \\ \mathcal{G}_i & i = 0, L/2 \end{cases} \tag{2.40}$$

we can write the new subband to fullband transformation as

$$\underline{h} = \mathrm{Re}(\mathcal{G}_{half} \underline{h}_{s,half}). \tag{2.41}$$

Note that this works because we take real part of fullband adaptive filter to ensure it is real.

An example for $L = 4$. The subband adaptive filter $\underline{h}_1$ and $\underline{h}_3$ are conjugate to each other since $\mathcal{G}_1$ and $\mathcal{G}_3$ are conjugate of each other. We can now write

$$\begin{aligned} \underline{h} &= [\mathcal{G}_0, \mathcal{G}_1, \mathcal{G}_2, \mathcal{G}_3] \begin{bmatrix} \underline{h}_0 \\ \underline{h}_1 \\ \underline{h}_2 \\ \underline{h}_3 \end{bmatrix} && (2.42) \\ &= \mathcal{G}_0 \underline{h}_0 + \mathcal{G}_1 \underline{h}_1 + \mathcal{G}_2 \underline{h}_2 + \mathcal{G}_3 \underline{h}_3 && (2.43) \\ &= \mathcal{G}_0 \underline{h}_0 + \mathcal{G}_1 \underline{h}_1 + \mathcal{G}_2 \underline{h}_2 + \mathcal{G}_1^* \underline{h}_1^* && (2.44) \\ &= \mathcal{G}_0 \underline{h}_0 + 2\,\mathrm{Re}\,(\mathcal{G}_1 \underline{h}_1) + \mathcal{G}_2 \underline{h}_2 && (2.45) \\ &\overset{\mathrm{Re}}{=} \mathrm{Re}(\mathcal{G}_0 \underline{h}_0) + 2\,\mathrm{Re}\,(\mathcal{G}_1 \underline{h}_1) + \mathrm{Re}(\mathcal{G}_2 \underline{h}_2) && (2.46) \\ &= \mathrm{Re}(\mathcal{G}_{half} \underline{h}_{s,half}). && (2.47) \end{aligned}$$

Chapter 3

Delayless Subband Adaptive Filter Closed Loop

In this chapter we will derive the equation set for the DSAF closed loop of the form:

$$\mathbf{Q}\mathbf{R}_x = \mathbf{Q}\underline{r}_{dx}. \tag{3.1}$$

The main difference between the Wiener-Hopf equation (1.3) with a preconditioner $\mathbf{C}$ is that the matrices in (3.1) are rectangular. $\mathbf{Q}$ can be interpret as a rectangular preconditioner for the fullband rectangular autocorrelation matrix $\mathbf{R}_x$.

In the following we will reverse the 3 steps of the unified framework as described in section 1.2 on page 6 in an attempt to find the equation set. Previously attempts to establish the equation set has been done in [7, 11].

However, a central problem in deriving the equation set will be to rearrange certain matrix-vector products, which we will examine first.

Let $\mathbf{A}$ be Toeplitz and $\underline{f}$ a column vector. Then we wish to reorder the product $\mathbf{A}\underline{f}$ to $\tilde{\mathcal{F}}^T\underline{a}$, where $\tilde{\mathcal{F}}^T$ can be associated with $\underline{f}$ and $\mathbf{A}$ with $\underline{a}$. Let us look at an example.

$$\mathbf{A} = \begin{bmatrix} a_1 & a_2 & a_3 \\ a_4 & a_1 & a_2 \\ a_5 & a_4 & a_1 \\ a_6 & a_5 & a_4 \end{bmatrix} \tag{3.2}$$

The Toeplitz matrix can be identified by the last (or first) column and last (or first) row. Where we trace from the top of the last column, down to the last row, then to the beginning of the last row.

$$\underline{a} = \begin{bmatrix} a_3 & a_2 & a_1 & a_4 & a_5 & a_6 \end{bmatrix}^T \tag{3.3}$$

Constructing now a vector $\tilde{\underline{f}}$ which is row reversed of $\underline{f}$. With careful inspection and direct computation $\tilde{\mathcal{F}}^T$ is given by

$$\begin{bmatrix} f_3 & f_2 & f_1 & 0 & 0 & 0 \\ 0 & f_3 & f_2 & f_1 & 0 & 0 \\ 0 & 0 & f_3 & f_2 & f_1 & 0 \\ 0 & 0 & 0 & f_3 & f_2 & f_1 \end{bmatrix} = \begin{bmatrix} \tilde{\underline{f}} & \underline{0}_3^T & & \\ 0 & \tilde{\underline{f}} & \underline{0}_2^T & \\ 0 & 0 & \tilde{\underline{f}} & 0 \\ 0 & 0 & 0 & \tilde{\underline{f}} \end{bmatrix} \tag{3.4}$$

We can therefore reorder any product of Toeplitz matrix with a column vector, $\mathbf{A}\underline{f} = \tilde{\mathcal{F}}^T\underline{a}$, by this method.

The Equation Set

With the LMS algorithm in each subband yields us the weight update iteration

$$\underline{h}_i(k+1) = \underline{h}_i(k) + \mu e_i(k)\underline{x}_i^*(k), \qquad i = 0, 1, \dots, L-1 \tag{3.5}$$

where $\underline{x}_i^*(k)$ is the conjugated subband input signal of length $P_H = M/N$.

In order to uncover the equation set will view this as the simplest estimation of

$$\underline{h}_i(k+1) = \underline{h}_i(k) + \mu\,\mathrm{E}\left\{e_i(k)\underline{x}_i^*(k)\right\}. \tag{3.6}$$

Note that this actually corresponds to convergence in the mean, i.e. the iteration matrix of a Richardson iteration with $\underline{h}_i(k)$ is the same as the iteration matrix when we investigate convergence in the mean $\mathrm{E}\left\{\underline{h}_i(k)\right\}$ with the independent assumption [8]. Something we will elaborate further in chapter 4.

As we have seen previously the subband signal $\underline{x}_i^*(k)$ can be written as

$$\underline{x}_i^*(k) = \begin{bmatrix} \underline{f}_i^H & \underline{0}^T & & \\ \underline{0}_N^T & \underline{f}_i^H & \underline{0}^T & \\ \underline{0}_N^T & \underline{0}_N^T & \underline{f}_i^H & \underline{0}^T \\ \vdots & \vdots & \ddots & \\ \underline{0}_N^T & \underline{0}_N^T & \dots & \underline{f}_i^H \end{bmatrix} \begin{bmatrix} x^*(kN) \\ x^*(kN-1) \\ x^*(kN-2) \\ \vdots \\ x^*(kN-(L_x-1)) \end{bmatrix} \tag{3.7}$$

$$= \mathcal{F}_i^H \underline{x}^*(kN) \tag{3.8}$$

where the fullband signal $\underline{x}(kN)$ is of length $L_x = (P_H - 1)N + L_f$, and $\mathcal{F}_i^H$ has dimension $P_H \times L_x$ (empty entries is filled with zeroes). To simplify the math, and with the presumption that update rates between every M to 1 input samples do not influence much the convergence performance, we let the update rate to be every N input samples. This let us write the subband error signal $e_i(k)$ as

$$e_i(k) = \underline{f}_i^T \underline{e}(kN) \tag{3.9}$$

$$= \underline{f}_i^T [\underline{d}(kN) - \mathbf{X}^T(kN)\underline{h}(k)] \tag{3.10}$$

where $\underline{h}(k)$ is the fullband adaptive filter of length M. $\underline{d}(kN)$ is the fullband desired signal with length equal to the analysis filter $\underline{f}_i$ of length L_f, i.e.

$$\underline{d}(kN) = \begin{bmatrix} d(kN) \\ d(kN-1) \\ d(kN-2) \\ \vdots \\ d(kN-L_f+1) \end{bmatrix} \tag{3.11}$$

and $\mathbf{X}^T(kN)$ is a signal matrix,

$$\mathbf{X}^T(kN) = \begin{bmatrix} x(kN) & x(kN-1) & \dots & x(kN-M+1) \\ x(kN-1) & x(kN-2) & \dots & x(kN-M) \\ x(kN-2) & x(kN-3) & \dots & x(kN-M-1) \\ \vdots & \vdots & & \vdots \\ x(kN-L_f+1) & x(kN-L_f) & \dots & x(kN-L_f-M+2) \end{bmatrix} \tag{3.12}$$

with dimension $L_f \times M$.

Next is to simplify the subband iteration with the matrix-vector product reordering that was mention earlier. We first expand $e_i(k)\underline{x}_i^*(k)$,

$$\begin{aligned} e_i(k)\underline{x}_i^*(k) &= \underline{x}_i^*(k)e_i(k) && (3.13)\\ &= \mathcal{F}_i^H \underline{x}^*(kN)\underline{f}_i^T[\underline{d}(kN) - \mathbf{X}^T(kN)\underline{h}(k)] && (3.14)\\ &= \mathcal{F}_i^H \underline{x}^*(kN)\underline{d}^T(kN)\underline{f}_i - \mathcal{F}_i^H \underline{x}^*(kN)\underline{f}_i^T \mathbf{X}^T(kN)\underline{h}(k). && (3.15)\end{aligned}$$

The first term with the expectation operator is

$$\mathrm{E}\left\{\mathcal{F}_i^H \underline{x}^*(kN)\underline{d}^T(kN)\underline{f}_i\right\} = \mathcal{F}_i^H \mathbf{R}_{dx}\underline{f}_i. \tag{3.16}$$

$\mathbf{R}_{dx}$ is a Toeplitz matrix, which is multiplied with $\underline{f}_i$:

$$\begin{bmatrix} r_{dx}(0) & r_{dx}(-1) & ... & r_{dx}(-L_f+1)\\ r_{dx}(1) & r_{dx}(0) & ... & r_{dx}(-L_f+2)\\ r_{dx}(2) & r_{dx}(1) & ... & r_{dx}(-L_f+3)\\ \vdots & \vdots & & \vdots\\ r_{dx}(L_x-1) & r_{dx}(L_x-2) & ... & r_{dx}(-L_f+L_x)\end{bmatrix}\begin{bmatrix} f_i(0)\\ f_i(1)\\ f_i(2)\\ \vdots\\ f_i(L_f-1)\end{bmatrix}. \tag{3.17}$$

With the reordering method mention previously we can construct a crosscorrelation vector

$$\underline{r}_{dx} \triangleq \begin{bmatrix} r_{dx}(-L_f+1)\\ r_{dx}(-L_f+2)\\ \vdots\\ r_{dx}(-1)\\ r_{dx}(0)\\ r_{dx}(1)\\ \vdots\\ r_{dx}(L_x-1)\end{bmatrix} \tag{3.18}$$

and let $\tilde{\underline{f}}_i = [f_i(L_f-1), f_i(L_f-2), ... f_i(0)]^T$. Then $\mathbf{R}_{dx}\underline{f}_i$ can be written as

$$\begin{bmatrix} \tilde{\underline{f}}_i^T & \underline{0}^T & & & \\ 0 & \tilde{\underline{f}}_i^T & \underline{0}^T & & \\ 0 & 0 & \tilde{\underline{f}}_i^T & \underline{0}^T & \\ \vdots & & & \ddots & \\ \underline{0}^T & & & & \tilde{\underline{f}}_i^T\end{bmatrix}\underline{r}_{dx} = \tilde{\mathcal{F}}_i^T \underline{r}_{dx} \tag{3.19}$$

where $\tilde{\mathcal{F}}_i^T$ has dimension $L_x \times (L_x + L_f - 1)$, and empty entries are filled with zeroes. After reordering, the first term is given by

$$\mathrm{E}\left\{\mathcal{F}_i^H \underline{x}^*(kN)\underline{d}^T(kN)\underline{f}_i\right\} = \mathcal{F}_i{}^H \tilde{\mathcal{F}}_i^T \underline{r}_{dx}. \tag{3.20}$$

We will now look at the second term, in particular $\underline{x}^*(kN)\underline{f}_i^T\mathbf{X}^T(kN)$. Expanding and rewrite

$$\begin{aligned}
&\underline{x}^*(kN)\underline{f}_i^T\mathbf{X}^T(kN) && (3.21)\\
&= \underline{x}^*(kN)\left[\underline{f}_i^T\underline{x}_0, \underline{f}_i^T\underline{x}_1, \dots, \underline{f}_i^T\underline{x}_{M-1}\right] && (3.22)\\
&= \left[\underline{x}^*(kN)\underline{f}_i^T\underline{x}_0, \underline{x}^*(kN)\underline{f}_i^T\underline{x}_1, \dots, \underline{x}^*(kN)\underline{f}_i^T\underline{x}_{M-1}\right] && (3.23)\\
&= \left[\underline{x}^*(kN)\underline{x}_0^T\underline{f}_i, \underline{x}^*(kN)\underline{x}_1^T\underline{f}_i, \dots, \underline{x}^*(kN)\underline{x}_{M-1}^T\underline{f}_i\right] && (3.24)
\end{aligned}$$

where $\underline{x}_j$ is j column in $\mathbf{X}^T(kN)$:

$$\underline{x}_j^T = [x(kN-j), x(kN-1-j), \dots, x(kN-L_f+1-j)]. \tag{3.25}$$

Applying the expectation operator yields us

$$\begin{aligned}
\left[\mathbf{R}_{x,0}\underline{f}_i, \mathbf{R}_{x,1}\underline{f}_i, \dots, \mathbf{R}_{x,M-1}\underline{f}_i\right] &= \left[\tilde{\mathcal{F}}_i^T\underline{r}_{x,0}, \tilde{\mathcal{F}}_i^T\underline{r}_{x,1}, \dots, \tilde{\mathcal{F}}_i^T\underline{r}_{x,M-1}\right] && (3.26)\\
&= \tilde{\mathcal{F}}_i^T\left[\underline{r}_{x,0}, \underline{r}_{x,1}, \dots, \underline{r}_{x,M-1}\right] && (3.27)
\end{aligned}$$

where each $\mathbf{R}_{x,j}$ is a rectangular autocorrelation matrix,

$$\mathbf{R}_{x,j} = \begin{bmatrix} r_x(-j) & r_x(-j-1) & \dots & r_x(-j-L_f+1) \\ r_x(-j+1) & r_x(-j) & \dots & r_x(-j-L_f+2) \\ r_x(-j+2) & r_x(-j+1) & \dots & r_x(-j-L_f+3) \\ \vdots & \vdots & & \vdots \\ r_x(-j+L_x-1) & r_x(-j+L_x-2) & \dots & r_x(-j+L_x-L_f) \end{bmatrix} \tag{3.28}$$

and $\tilde{\mathcal{F}}_i^T$ has the same structure as in equation (3.19) with dimension $L_x \times (L_x + L_f - 1)$. From $\mathbf{R}_{x,j}$ we can define $\underline{r}_{x,j}$ to be

$$\underline{r}_{x,j} \triangleq [r_x(-j-L_f+1), r_x(-j-L_f+2), \dots, r_x(-j+L_x-1)]^T \tag{3.29}$$

with length $L_x + L_f - 1$. Thus $\left[\underline{r}_{x,0}, \underline{r}_{x,1}, \dots, \underline{r}_{x,M-1}\right]$ is given by

$$\mathbf{R}_x \triangleq \begin{bmatrix} r_x(-L_f+1) & r_x(-L_f) & \dots & r_x(-L_f+1-M+1) \\ r_x(-L_f+2) & r_x(-L_f+1) & \dots & r_x(-L_f+2-M+1) \\ r_x(-L_f+3) & r_x(-L_f+2) & \dots & r_x(-L_f+3-M+1) \\ \vdots & \vdots & & \vdots \\ r_x(0) & r_x(-1) & \dots & r_x(-M+1) \\ r_x(1) & r_x(0) & \dots & r_x(-M+2) \\ \vdots & \vdots & & \vdots \\ r_x(M-1) & r_x(M-2) & \dots & r_x(0) \\ \vdots & \vdots & & \vdots \\ r_x(L_x-1) & r_x(L_x-2) & \dots & r_x(L_x-M) \end{bmatrix} \tag{3.30}$$

with dimension $(L_x + L_f - 1) \times M$. $\mathbf{R}_x$ can be written differently by exploiting the conjugate symmetry of the autocorrelation, $r_x(k) = r_x^*(-k)$. The second term with the independent assumption can therefore be written as

$$\mathrm{E}\left\{\mathcal{F}_i^H\underline{x}^*(kN)\underline{f}_i^T\mathbf{X}^T(kN)\underline{h}(k)\right\} = \mathcal{F}_i^H\tilde{\mathcal{F}}_i^T\mathbf{R}_x\underline{h}(k). \tag{3.31}$$

After gathering the terms the subband iteration is given by

$$\underline{h}_i(k+1) = \underline{h}_i(k) + \mu \,\mathrm{E}\left\{e(k)\underline{x}_i^*(k)\right\} \tag{3.32}$$

$$= \underline{h}_i(k) + \mu \left[\mathcal{F}_i^H \tilde{\mathcal{F}}_i^T \underline{r}_{dx} - \mathcal{F}_i^H \tilde{\mathcal{F}}_i^T \mathbf{R}_x \underline{h}(k)\right]. \tag{3.33}$$

Until now we have used LMS in each of the subbands. However, we could have used NLMS. With NLMS, the subband preconditioner is

$$\mathbf{C}_i = \frac{1}{\underline{x}_i^H(k)\underline{x}_i(k)} \mathbf{I}_{P_H} \tag{3.34}$$

where $\underline{x}_i(k) = [x_i(k), x_i(k-1), \ldots, x_i(k-P_H+1)]^T$ and $\mathbf{I}_{P_H}$ is the identity matrix of size P_H. $\underline{x}_i^H(k)\underline{x}_i(k)$ can be viewed as an estimation of

$$P_H \,\mathrm{E}\left\{x_i^*(k)x_i(k)\right\} = P_H \underline{f}_i^H \,\mathrm{E}\left\{\underline{x}_i^*(kN)\underline{x}^T(kN)\right\} \underline{f}_i \tag{3.35}$$

$$= P_H \|\underline{f}_i\|_{\mathbf{R}}^2 \tag{3.36}$$

where $\|\cdot\|_{\mathbf{R}}$ is a weighted $\mathbf{R}$ norm. $\mathbf{R}$ is the normal fullband autocorrelation matrix of size $L_f \times L_f$. Absorbing P_H into μ, and name this the NLMS subband preconditioner strategy gives us

$$\mathbf{C}_{i,NLMS} \triangleq \frac{1}{\|\underline{f}_i\|_{\mathbf{R}}^2} \mathbf{I}_{P_H}. \tag{3.37}$$

This power normalization subband preconditioner strategy is used previously by [7, 11].

However, a more correct subband preconditioner strategy is perhaps using ideas from [10] and realize that $\mathbf{C}_i$ is an inverse of a subband autocorrelation, i.e.

$$\mathbf{C}_i = \mathrm{E}\left\{\underline{x}_i^*(k)\underline{x}_i^T(k)\right\}^{-1} = [\mathcal{F}_i^H \underline{x}^*(kN)\underline{x}^T(kN)\mathcal{F}_i]^{-1} \tag{3.38}$$

This comes from the fact we can write

$$[\epsilon \mathbf{I} + \underline{x}_i^*(k)\underline{x}_i^T(k)]^{-1}\underline{x}_i^*(k) = \frac{1}{\epsilon + \|\underline{x}_i(k)\|_2^2}\underline{x}_i^*(k) \tag{3.39}$$

where the right hand side (neglect $\underline{x}_i^*(k)$) is a power normalization with a regularization factor. This is the same method we used to derived NLMS from the unified framework, in section 1.2. We will name this the APA subband preconditioner strategy

$$\mathbf{C}_{i,APA} \triangleq \mathrm{E}\left\{\underline{x}_i^*(k)\underline{x}_i^T(k)\right\}^{-1} = [\mathcal{F}_i^H \underline{x}^*(kN)\underline{x}^T(kN)\mathcal{F}_i]^{-1} \tag{3.40}$$

since it is also the same subband preconditioner strategy if we had used APA in each subband. Note that for white input signal and $P_H = 1$ then both subband preconditioner strategies are the same.

Including a subband preconditioner (strategy) the subband iteration is given by

$$\underline{h}_i(k+1) = \underline{h}_i(k) + \mu \mathbf{C}_i \mathcal{F}_i^H \tilde{\mathcal{F}}_i^T [\underline{r}_{dx} - \mathbf{R}_x \underline{h}(k)] \tag{3.41}$$

where $\mathbf{C}_i$ can either be the NLMS or APA subband preconditioner strategy. We can now stack all the subband iterations together such that we have

$$\begin{bmatrix} \underline{h}_0(k+1) \\ \underline{h}_1(k+1) \\ \vdots \\ \underline{h}_{L-1}(k+1) \end{bmatrix} = \begin{bmatrix} \underline{h}_0(k) \\ \underline{h}_1(k) \\ \vdots \\ \underline{h}_{L-1}(k) \end{bmatrix} + \mu \begin{bmatrix} \mathbf{C}_0 \mathcal{F}_0^H \tilde{\mathcal{F}}_0^T [\underline{r}_{dx} - \mathbf{R}_x \underline{h}(k)] \\ \mathbf{C}_1 \mathcal{F}_1^H \tilde{\mathcal{F}}_1^T [\underline{r}_{dx} - \mathbf{R}_x \underline{h}(k)] \\ \vdots \\ \mathbf{C}_{L-1} \mathcal{F}_{L-1}^H \tilde{\mathcal{F}}_{L-1}^T [\underline{r}_{dx} - \mathbf{R}_x \underline{h}(k)] \end{bmatrix}. \quad (3.42)$$

Defining now the stacked subband adaptive filter weights as

$$\underline{h}_s(k) = \begin{bmatrix} \underline{h}_0(k) \\ \underline{h}_1(k) \\ \vdots \\ \underline{h}_{L-1}(k) \end{bmatrix} \quad (3.43)$$

and use the fact that

$$\underline{h}(k) = \mathcal{G} \underline{h}_s(k) \quad (3.44)$$

to get the fullband iteration:

$$\underline{h}(k+1) = \underline{h}(k) + \mu \mathcal{G} \begin{bmatrix} \mathbf{C}_0 \mathcal{F}_0^H \tilde{\mathcal{F}}_0^T [\underline{r}_{dx} - \mathbf{R}_x \underline{h}(k)] \\ \mathbf{C}_1 \mathcal{F}_1^H \tilde{\mathcal{F}}_1^T [\underline{r}_{dx} - \mathbf{R}_x \underline{h}(k)] \\ \vdots \\ \mathbf{C}_{L-1} \mathcal{F}_{L-1}^H \tilde{\mathcal{F}}_{L-1}^T [\underline{r}_{dx} - \mathbf{R}_x \underline{h}(k)] \end{bmatrix}. \quad (3.45)$$

Letting now

$$\mathbf{Q} \triangleq \mathcal{G} \begin{bmatrix} \mathbf{C}_0 \mathcal{F}_0^H \tilde{\mathcal{F}}_0^T \\ \mathbf{C}_1 \mathcal{F}_1^H \tilde{\mathcal{F}}_1^T \\ \vdots \\ \mathbf{C}_{L-1} \mathcal{F}_{L-1}^H \tilde{\mathcal{F}}_{L-1}^T \end{bmatrix} \quad (3.46)$$

and comparing the Richardson iteration with

$$\underline{h}(k+1) = \underline{h}(k) + \mu \mathbf{Q} [\underline{r}_{dx} - \mathbf{R}_x \underline{h}(k)] \quad (3.47)$$

we can see that $\mathbf{Q}$ is a (rectangular) preconditioner. Thus we have establish the preconditioner. Finally reversing the Richardson iteration yields us the equation set for the DSAF closed loop:

$$\mathbf{Q}\mathbf{R}_x \underline{h} = \mathbf{Q} \underline{r}_{dx}. \quad (3.48)$$

Notice that both $\mathbf{R}_x$ and $\underline{r}_{dx}$ have $L_x + L_f - 1$ rows. By using the Wiener-Hopf equation

$$\sum_{l=0}^{M-1} h_l r_x(k-l) = r_{dx}(k) \quad (3.49)$$

it is easily to see that

$$\mathbf{R}_x \underline{h} = \underline{r}_{dx} \quad (3.50)$$

which is over determined and consistent equation set, M unknowns but we have $L_x + L_f - 1$ equations.

From the equation set we can see that the scaling of analysis filter bank and weight transform is insignificant because it gets cancelled.

Chapter 4

Convergence and Objective Function

One of the essential connection between the equation set and convergence in the mean will be elaborated in this chapter. Previously it was mention both iterations have the same iteration matrix.

Indeed when we look at the (normal) Wiener-Hopf equation

$$\mathbf{R}_x \underline{h} = \underline{r}_{dx} \tag{4.1}$$

the Richardson iteration with an iteration matrix formulation is given by

$$\underline{h}(k+1) = (\mathbf{I} - \mu \mathbf{R}_x)\underline{h}(k) + \mu \underline{r}_{dx} \tag{4.2}$$

where we can define $\mathbf{T}_1 \triangleq \mathbf{I} - \mu \mathbf{R}_x$ as the iteration matrix [17].

Now if we work backwards from the (fullband) LMS algorithm and look at convergence in the mean, with the independent assumption, gives us the same iteration, but in terms of $\mathrm{E}\left\{\underline{h}(k)\right\}$ [8]

$$\mathrm{E}\left\{\underline{h}(k+1)\right\} = (\mathbf{I} - \mu \mathbf{R}_x)\,\mathrm{E}\left\{\underline{h}(k)\right\} + \mu \underline{r}_{dx}. \tag{4.3}$$

From these two similar iterations, the convergence properties are solely determined by the iteration matrix $\mathbf{T}_1$.

With the Richardson iteration of DSAF closed loop equation set, we do also assume it corresponds to a similar iteration equation with convergence in the mean. However, in addition to the independent assumption between $\underline{h}(k)$ and $\underline{x}_i(k)$, we also assume that the subband preconditioner is independent of the rest.

After this clarification we are now ready to look at the convergence properties of the Richardson iteration of DSAF closed loop equation set

$$\underline{h}(k+1) = (\mathbf{I} - \mu \mathbf{Q}\mathbf{R}_x)\underline{h}(k) + \mu \mathbf{Q}\underline{r}_{dx}. \tag{4.4}$$

where we define $\mathbf{T}_2 \triangleq \mathbf{I} - \mu \mathbf{Q}\mathbf{R}_x$ as the iteration matrix. The iteration converges when the spectral radius $\rho(\mathbf{T}_2)$ is less than one [4], i.e. the eigenvalues of $\mathbf{T}_2$ are inside the unit circle. A crucial difference between $\mathbf{T}_1$ and $\mathbf{T}_2$ is that $\mathbf{R}_x$ from $\mathbf{T}_1$ is Hermitian, and therefore is a normal matrix. Because of this the eigenvectors, $\{\underline{v}_1, \underline{v}_2, \dots, \underline{v}_M\}$ of $\mathbf{R}_x$ form an orthonormal basis. Let it be an ordered basis such that the corresponding eigenvalues are ordered from highest to lowest. $\lambda_1 \geq \lambda_2 \geq \dots \geq \lambda_M$. The error sequence (coefficient deviation)

$\underline{\epsilon}(k) = \underline{h}_t - \underline{h}(k)$ given that $\underline{h}(0) = \underline{0}$ can be written as [13]

$$\begin{aligned}
\underline{\epsilon}(k) &= \mathbf{T}_1^k e(0) && (4.5)\\
&= [\mathbf{I} - \mu \mathbf{R}_x]^k e(0) && (4.6)\\
&= [\mathbf{I} - \mu \mathbf{R}_x]^k \underline{h}_t && (4.7)\\
&= \sum_{i=1}^{M} [\mathbf{I} - \mu \mathbf{R}_x]^k \operatorname{proj}_{\underline{v}_i} \underline{h}_t && (4.8)\\
&= \sum_{i=1}^{M} [\mathbf{I} - \mu \mathbf{R}_x]^k (\underline{h}_t^H \underline{v}_i) \underline{v}_i && (4.9)\\
&= \sum_{i=1}^{M} [1 - \mu \lambda_i]^k (\underline{h}_t^H \underline{v}_i) \underline{v}_i && (4.10)
\end{aligned}$$

where proj is the projection operator. To get to the last line we observe that

$$\begin{aligned}
(\mathbf{I} - \mu \mathbf{R}_x)\underline{v}_i &= \underline{v}_i - \mu \mathbf{R}_x \underline{v}_i && (4.11)\\
&= \underline{v}_i - \mu \lambda_i \underline{v}_i && (4.12)\\
&= (1 - \mu \lambda_i)\underline{v}_i && (4.13)
\end{aligned}$$

hence do this k times gives the result. What we can see is that we need $0 < \mu < 2/\lambda_{max}$ to ensure convergence. However, the alignment of $\underline{h}_t$ has a significant impact on convergence rate. If $\underline{h}_t$ aligns well with the eigenvector that corresponds to the greatest eigenvalue i.e. $\underline{h}_t^H \underline{v}_i = 0 \ \forall i \neq 1$ we see that we have the fastest convergence rate. Opposite if $\underline{h}_t$ aligns well with the eigenvector that corresponds to the lowest eigenvalue, the convergence rate is the slowest. Hence the eigenvaluespread determines the overall convergence.

If we carry similar calculation with $\mathbf{T}_2$ instead we encounter a problem. $\mathbf{QR}_x$ (in $\mathbf{T}_2$) does not necessarily give us an unitary eigenvector matrix ($\mathbf{V}^{-1} = \mathbf{V}^H$). Assume now that $\underline{h}_t$ aligns well with $\underline{v}_1$ but there exist an eigenvector $\underline{v}_n$ such that $\underline{h}_t^H \underline{v}_i = 0 \ \forall i \neq 1, n$. Then the (smaller) eigenvalue $|\lambda_n|$ will be the determining factor for the convergence rate. We use the absolute value because the eigenvalues are not necessary real. Nevertheless we could just improve the eigenvaluespread since in the real world we do not know the alignment of $\underline{h}_t$. But by enforcing the eigenvectors to be nearly perpendicular to each other we are limiting the impact of poorly aligned $\underline{h}_t$. Therefore with these considerations we wish to create an objective function which contains these two objectives, eigenvaluespread and orthogonal eigenvectors.

However, checking for whether the eigenvector matrix is unitary is not feasible, i.e. check that

$$\mathbf{A} = \mathbf{V}\mathbf{\Lambda}\mathbf{V}^{-1} \tag{4.14}$$

such that $\mathbf{V}^{-1} = \mathbf{V}^H$. An alternative is to check that $\mathbf{QR}_x$ is a normal matrix. The definition for a normal matrix $\mathbf{A}$ is

$$\mathbf{A}^H\mathbf{A} = \mathbf{A}\mathbf{A}^H \tag{4.15}$$

which is equivalent to the above, since a matrix $\mathbf{A}$ is normal if and only if it is unitary diagonalizable.

With a relative weight $0 \leq \alpha \leq 1$ factor between eigenvaluespread (of $\mathbf{QR}_x$) and normality our objective function is given by

$$\alpha \left(1 - \frac{|\lambda_{min}|}{|\lambda_{max}|}\right) + (1-\alpha) \left\| [\mathbf{QR}_x]^H \mathbf{QR}_x - \mathbf{QR}_x [\mathbf{QR}_x]^H \right\|_2 \tag{4.16}$$

where $\| \cdot \|_2$ is the 2-norm, i.e. maximum singular value for matrices and Euclidean norm for vectors.

This objective function is based solely on a linear algebra perspective. Though we also know from a signal processing perspective that high stop band attenuation and small transition band gives us a good filter response with less aliasing. A proposed objective function is to mix from the two disciplines. When $\|\mathbf{T}_2\|_2 < 1$ the error sequence is bounded by [4]

$$\|\underline{\epsilon}(k)\|_2 \leq \|\mathbf{T}_2^k\|_2 \|\underline{\epsilon}(0)\|_2. \tag{4.17}$$

Which is something we want because we want $\mathbf{Q}$ to be as close as a pseudoinverse to the rectangular $\mathbf{R}_x$ matrix. Therefore we wish to minimize $\|\mathbf{T}_2\|_2$, while also minimize the stop band energy, E_s, to make sure the filter response is consistent. The stop band energy can be defined as [21, 18]

$$E_s = \frac{1}{\pi} \int_{\omega_s}^{\pi} |H(e^{j\omega})|^2 d\omega \tag{4.18}$$

$$= \underline{h}^T \mathbf{P} \underline{h} \tag{4.19}$$

where ω_s is the cutoff frequency and $\underline{h}^T = [h_0, h_1, \dots, h_{L_f - 1}]$ are the filter coefficients for the prototype filter. Without going into details and omit any scaling $\mathbf{P}$ is given by

$$[\mathbf{P}]_{m,n} = \int_{\omega_s}^{\pi} \cos[(m-n)\omega] d\omega \tag{4.20}$$

$$= \frac{1}{m-n} \left\{ \sin[(m-n)\pi] - \sin[(m-n)\omega_s] \right\} \tag{4.21}$$

for $m = 0, 1, \dots, L_{L_f - 1}$ and $n = 0, 1, \dots, L_{L_f - 1}$. We can now define our alternative objective function as

$$\alpha \|\mathbf{I} - \mathbf{QR}_x\|_2 + (1-\alpha) E_s. \tag{4.22}$$

With these two objective functions there are 3 cases of interest to investigate:

1. Optimize analysis filter (FFT-2 scheme).
2. Optimize synthesis filter (DFT-FIR scheme).
3. Optimize analysis and synthesis filter (DFT-FIR scheme).

A reasonable assumption is that the input signal statistics is independent of our optimized filters. Therefore we will optimize with respect to a white input signal. And to keep it simple we also let $\mu = 1$ in the iteration matrix.

Chapter 5

Simulation and Result

With the two objective functions mention in the previous chapter we are now ready to test them with our DSAF equation set closed loop. The underlying algorithm we will use in each subband is NLMS. However, we will optimize analysis and synthesis prototype filter based on two different subband preconditioner strategies:

$$\mathbf{C}_{i,NLMS} \triangleq \frac{1}{\|\underline{f}_i\|^2_{\mathbf{R}}} \mathbf{I}_{P_H} \tag{5.1}$$

$$\mathbf{C}_{i,APA} \triangleq [\mathcal{F}_i^H \mathbf{R} \mathcal{F}_i]^{-1} \tag{5.2}$$

i.e. we view the power normalization factor $1/\|\underline{x}_i\|_2^2$ as an estimation of $\mathbf{C}_{i,NLMS}$ or $\mathbf{C}_{i,APA}$. Optimizations are done with fmincon command from Matlab. After optimization the DC gain of the prototype filter is normalized to 1. The reference filter is generated by the fir1$(L_f - 1, 1/L)$ Matlab command. (L_f is referred to the filter length and can be either analysis and synthesis.) Reference filter is the initial point for fmincon and unoptimized prototype filter when we compare.

If nothing else is mentioned we will use the following settings

$$N = 16 \tag{5.3}$$

$$L = 2N \tag{5.4}$$

$$M = 32, 64, 128 \tag{5.5}$$

$$P_H = M/N \tag{5.6}$$

$$L_f = 2L \tag{5.7}$$

$$L_g = 2L \tag{5.8}$$

$$\mu = 0.1 \frac{2}{|\lambda_{max}|} \tag{5.9}$$

where $|\lambda_{max}|$ is the largest (absolute) eigenvalue of $\mathbf{QR}_x$. $\mathbf{R}_x$ is based on the (simulated) input signal statistics. But the optimized filters are optimized with respect to white input signal. $\mathbf{Q}$'s composition is different, based on the subband preconditioner strategy and M that is used. In addition we will use an innovation representation for the WSS input signal $x(n)$. To keep it simple we will use an autoregressive model of order 1, i.e $x(n) = \rho x(n-1) + w(n)$. $w(n)$ is white noise with unit variance, and $|\rho| < 1$. When $\rho = 0$ the input signal is white noise. The more we increase ρ the more colored the signal will be. For our desired signal we will use a system identification framework and let $d(n) = \underline{h}_t^T \underline{x}(n) + v(n)$. $v(n)$ is also white noise.

The following signal generation method will be used.

1. Create $x(n) = \rho x(n-1) + w(n)$ with $\rho = \pm 0.95$.
2. Create $d(n) = \underline{h}_t^T \underline{x}(n)$.
3. Remove M first samples of $x(n)$ and $d(n)$.
4. Normalize $x(n)$ and $d(n)$ with $d(n)$'s power.
5. Add Gaussian white noise ($v(n)$) of -40dB to $d(n)$.

Throughout the simulation we will also use the half weight transform matrix, equation (2.39) on page 19, to update the fullband tap-weights. The half weight transform matrix is created from the (full) weight transform matrix, which in turn is generated by the decomposition method described in the beginning of chapter 2. This is to keep roundoff error equal. The update rate will be every M input samples, after all taps are filled up.

Optimization of Analysis Filters

In optimization of analysis filters we use the FFT-2 scheme. Because of the difficulty of this optimization only $M = 32$ with APA subband preconditioner strategy with the first objective function, eigenvaluespread and normality, is shown. Figure 5.1 and 5.2 show learning curves of 1000 runs. Figure 5.3 shows the filter response compared to the reference filter of length $2L$.

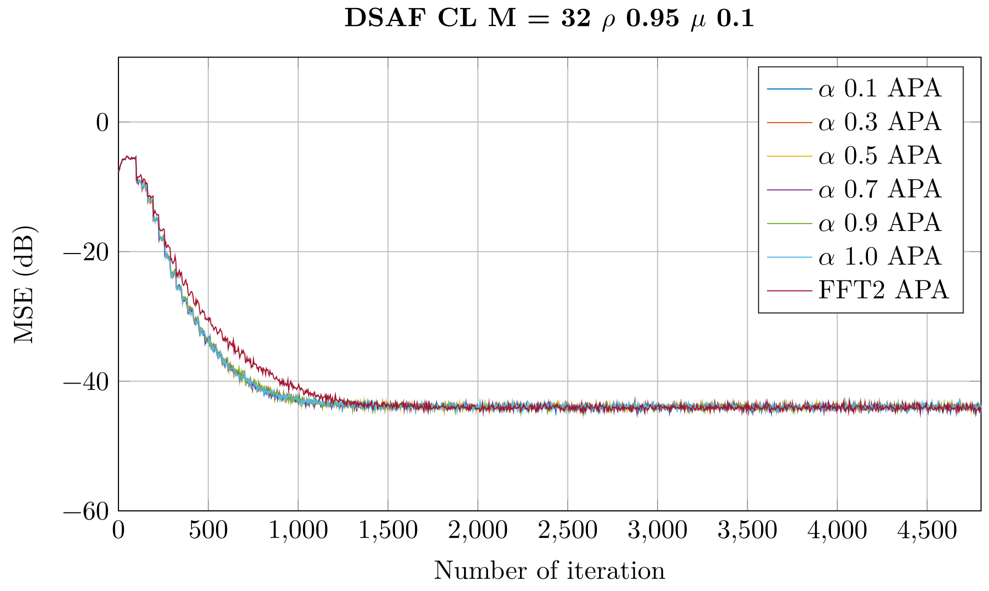


Figure 5.1: Comparison of analysis filter optimization $M = 32$, APA.

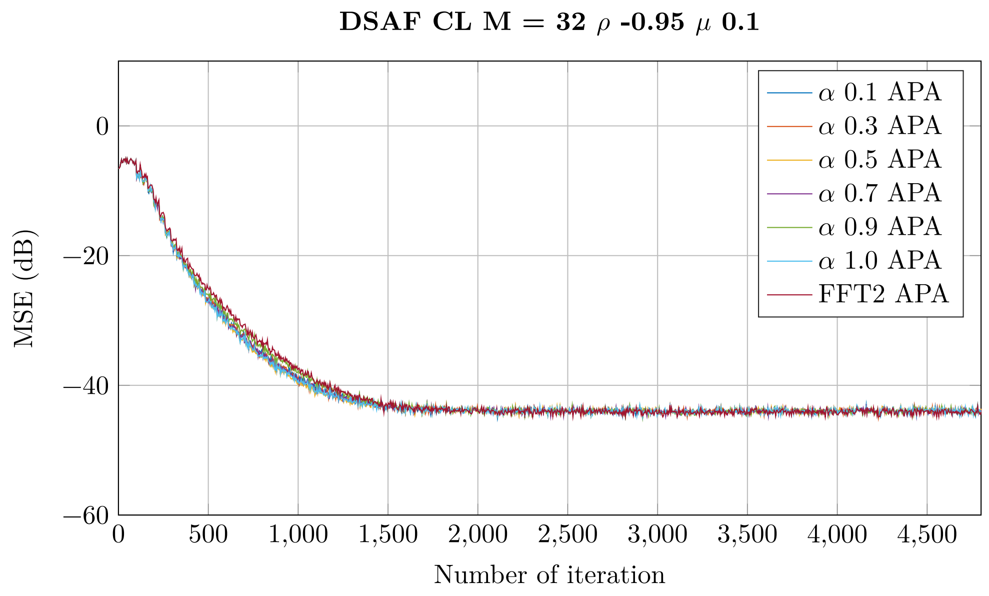


Figure 5.2: Comparison of analysis filter optimization $M = 32$, APA.

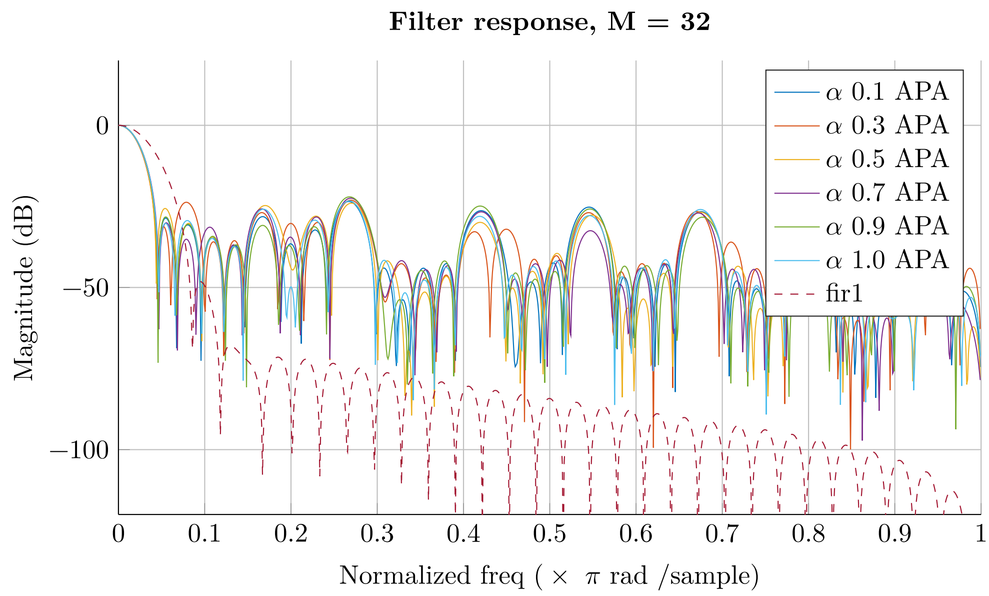


Figure 5.3: Filter response for analysis prototype filter $M = 32$, APA.

Optimization of Synthesis Filters

For optimization of synthesis filters we will use the DFT-FIR scheme. Analysis prototype filter is the same unoptimized synthesis prototype filter.

Further on the second objective function is used, maximum singular value and stop band energy. Figure 5.4 and 5.5 show learning curves for $M = 128$ with the APA subband preconditioner strategy. And Figure 5.6 and 5.7 show learning curves for the NLMS subband preconditioner strategy. The learning curves are generated from 1000 runs. Filter responses are found in Figure 5.8 and 5.9.

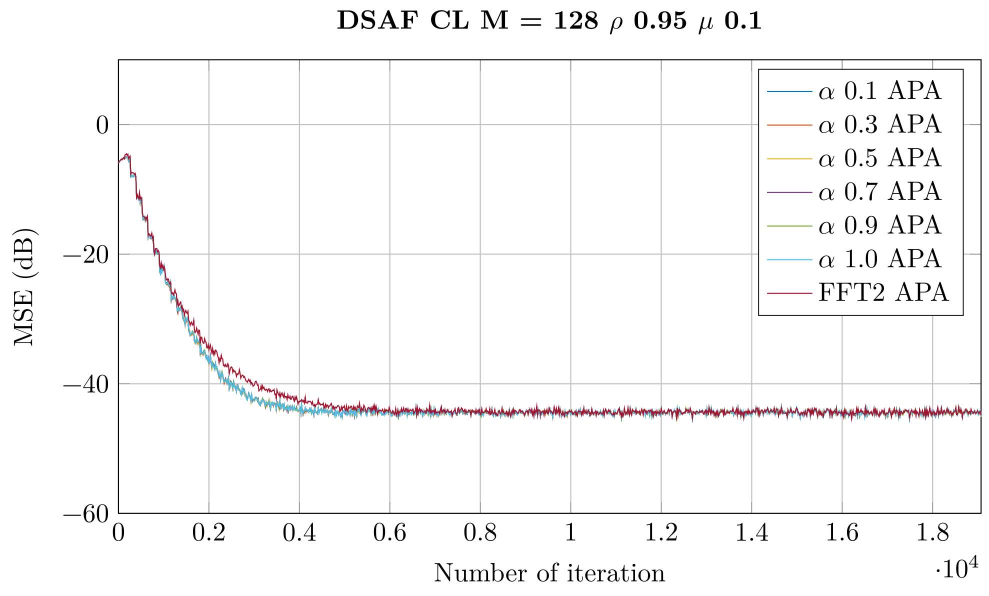


Figure 5.4: Comparison of synthesis filter optimization $M = 128$, APA.

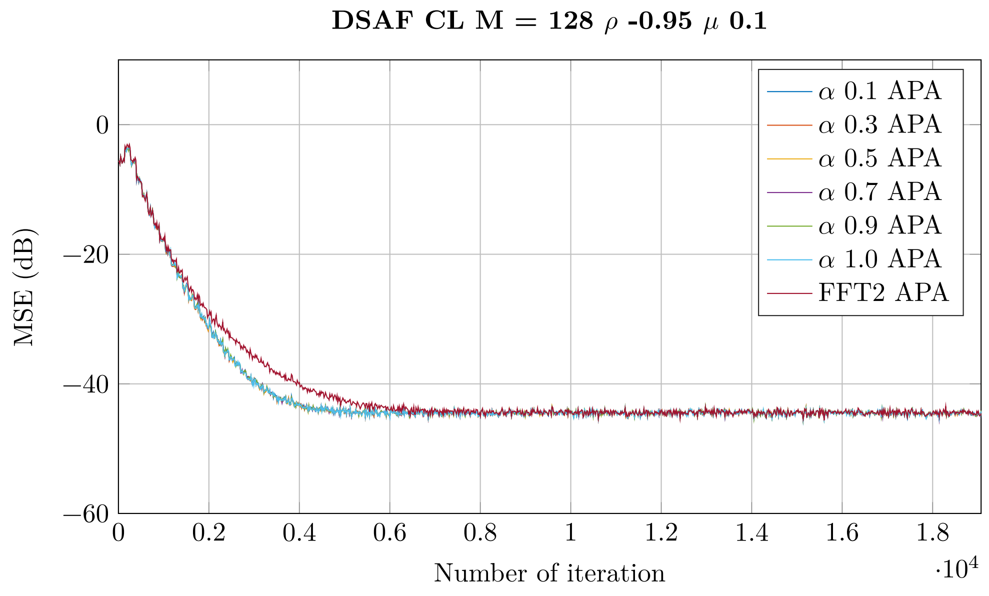


Figure 5.5: Comparison of synthesis filter optimization $M = 128$, APA.

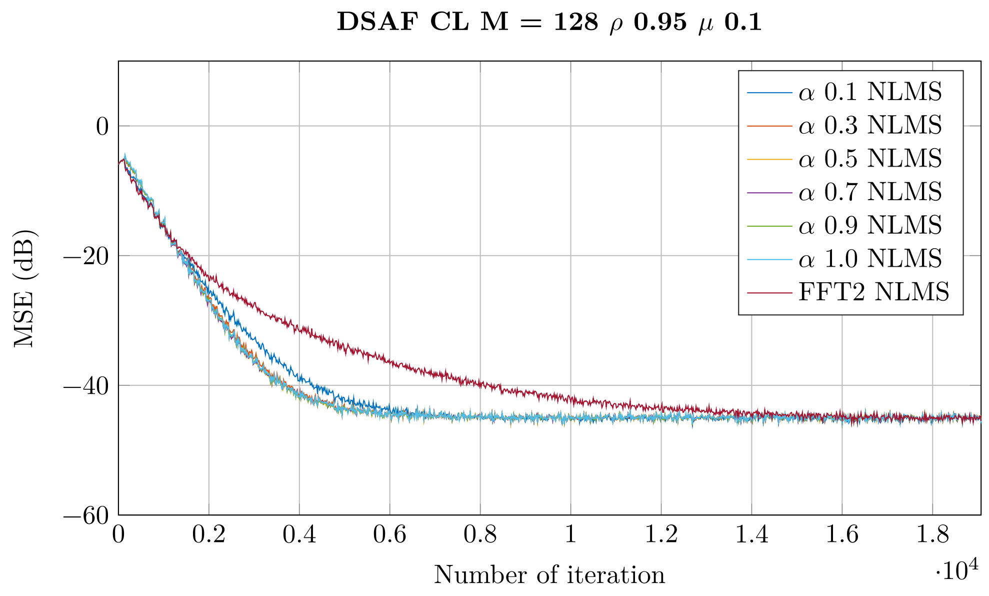


Figure 5.6: Comparison of synthesis filter optimization $M = 128$, NLMS.

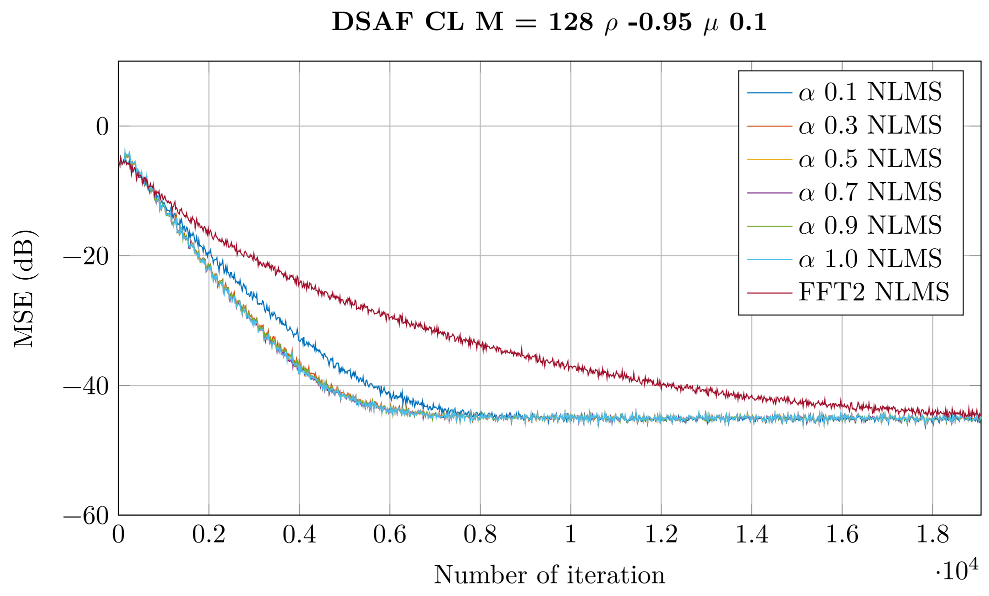


Figure 5.7: Comparison of synthesis filter optimization $M = 128$, NLMS.

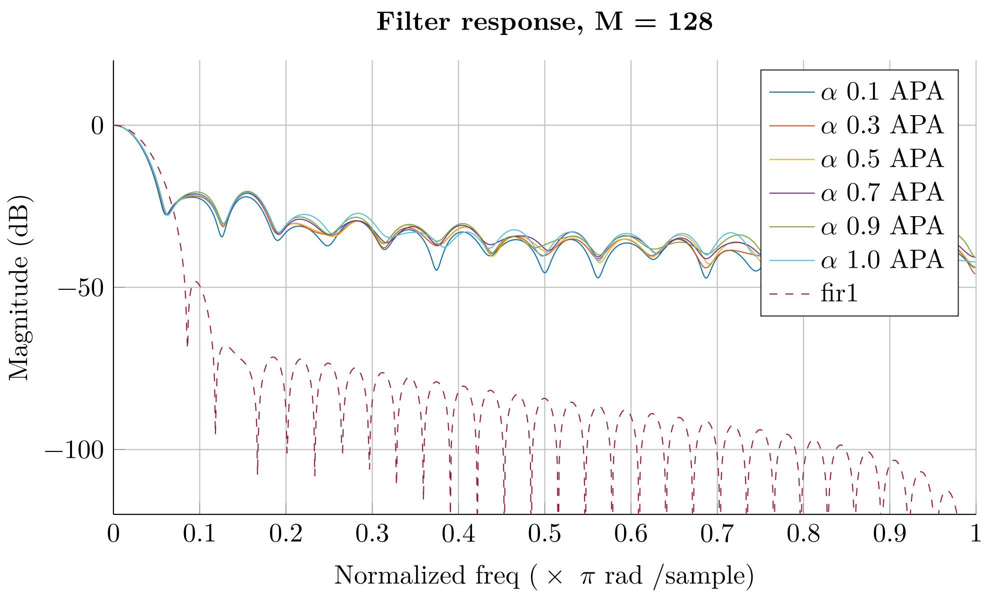


Figure 5.8: Filter response for synthesis prototype filter $M = 128$, APA.

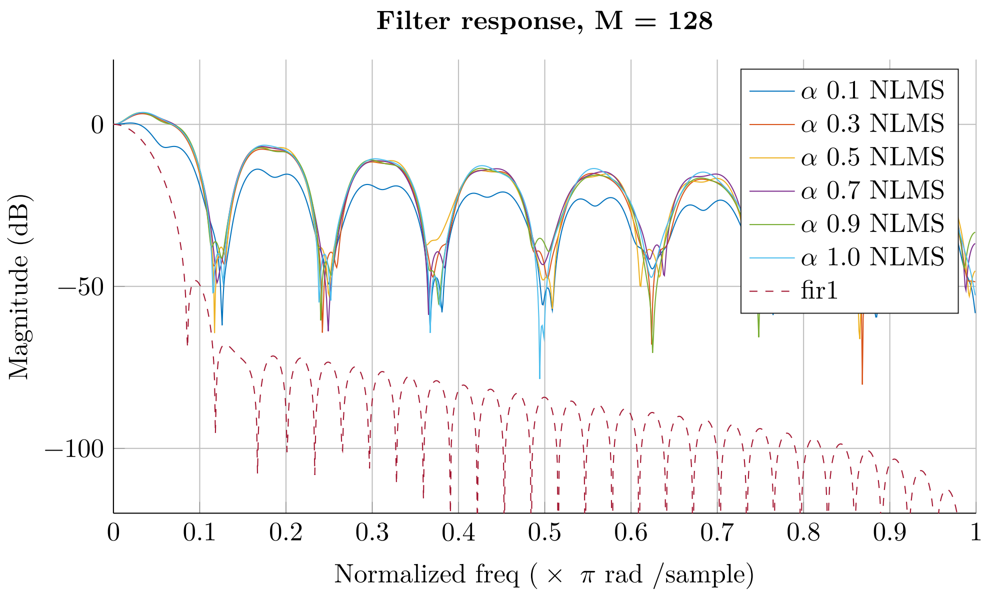


Figure 5.9: Filter response for synthesis prototype filter $M = 128$, NLMS.

Optimization of Analysis and Synthesis Filters

In this optimization, the analysis and synthesis prototype filter are the same. We have not time-reversed the synthesis filters. The second objective function which minimize maximum singular value and stop band energy is used.

Figure 5.10 and 5.11 show learning curves for 1000 runs with the APA subband preconditioner strategy and $M = 64$. Similar Figure 5.12 and 5.13 show learning curves for the NLMS subband preconditioner strategy. And Figure 5.14 and 5.15 show their respective filter response.

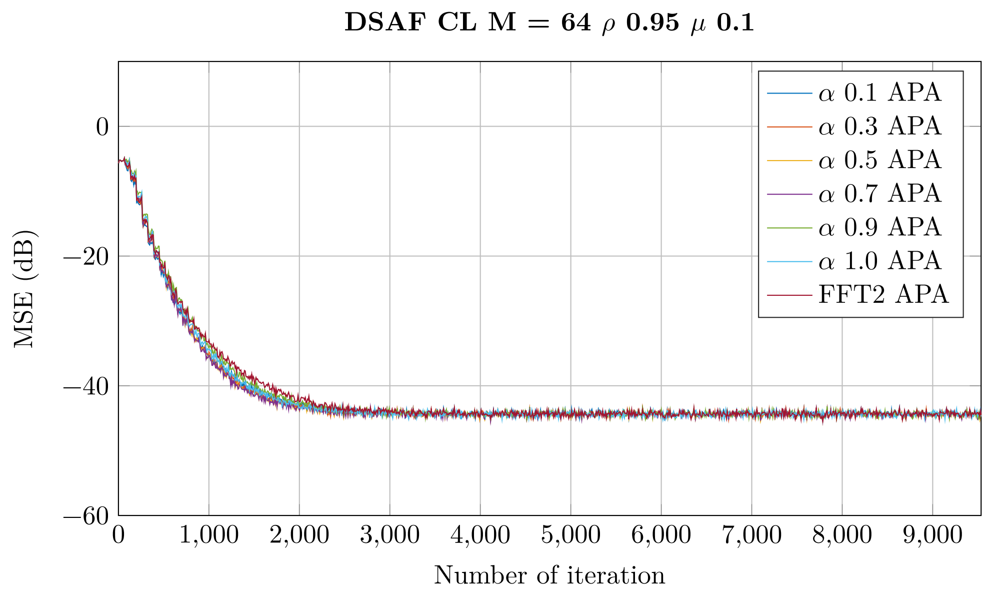


Figure 5.10: Comparison of analysis and synthesis filter optimization $M = 64$, APA. Max singular value E_s.

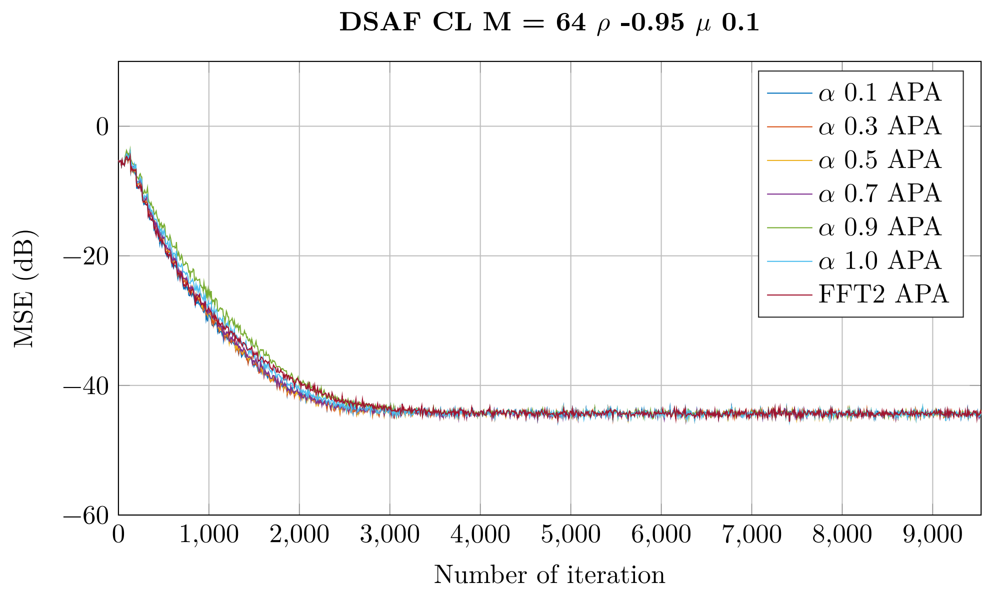


Figure 5.11: Comparison of analysis and synthesis filter optimization $M = 64$, APA. Max singular value E_s.

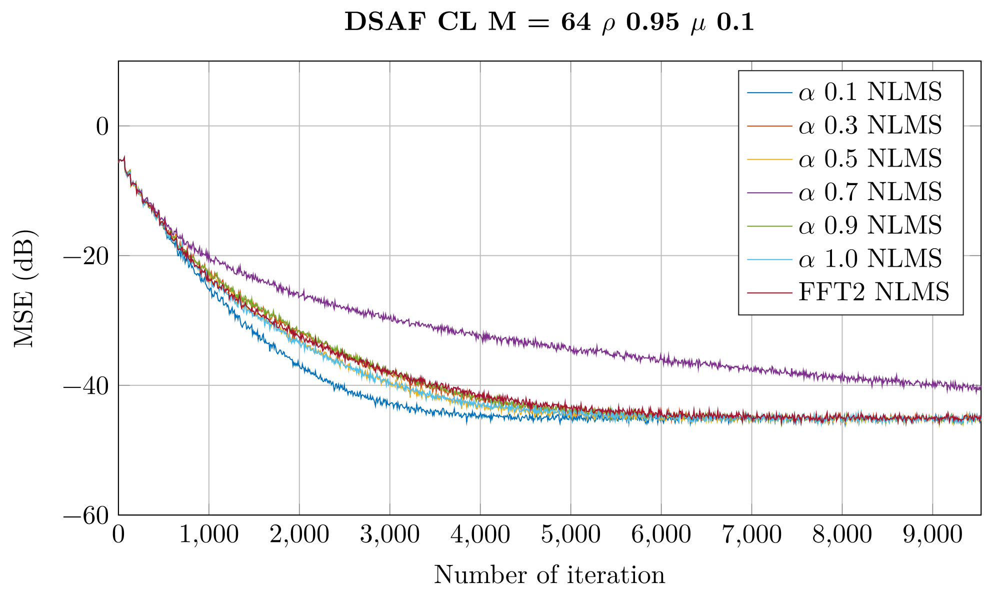


Figure 5.12: Comparison of analysis and synthesis filter optimization $M = 64$, NLMS. Max singular value E_s.

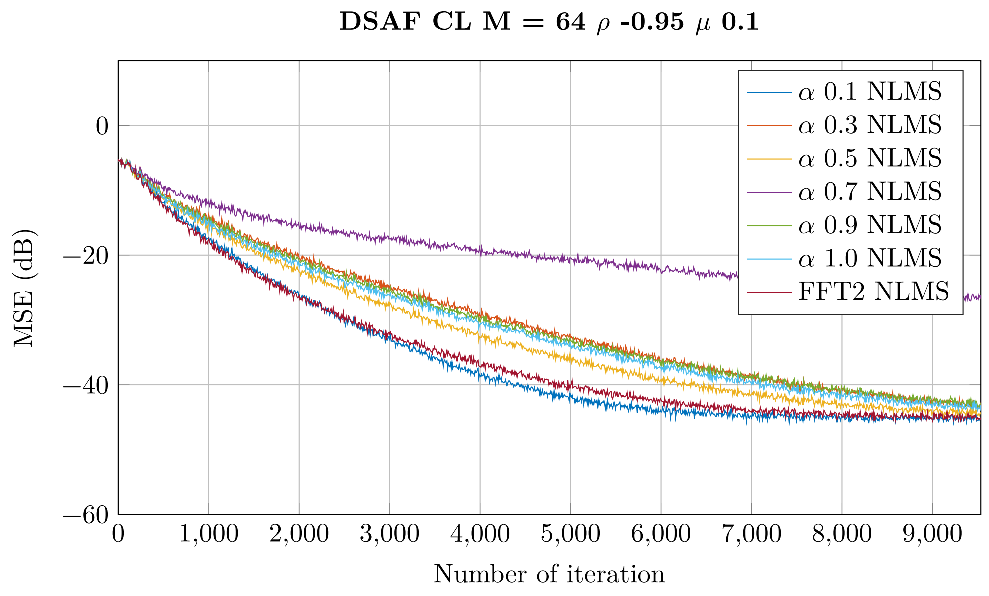


Figure 5.13: Comparison of analysis and synthesis filter optimization $M = 64$, NLMS. Max singular value E_s.

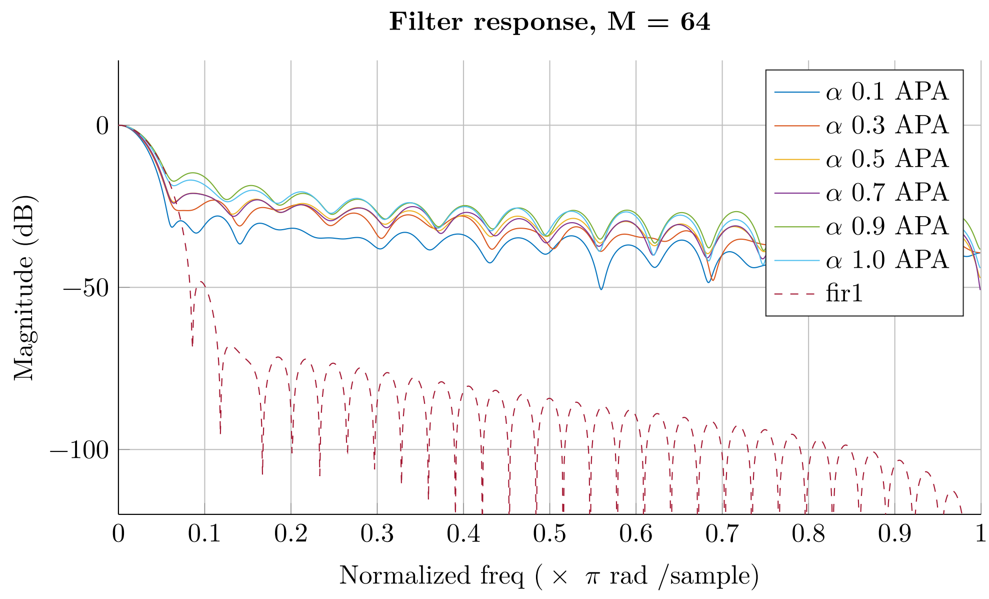


Figure 5.14: Filter response for analysis and synthesis prototype filter $M = 64$, APA. Max singular value E_s.

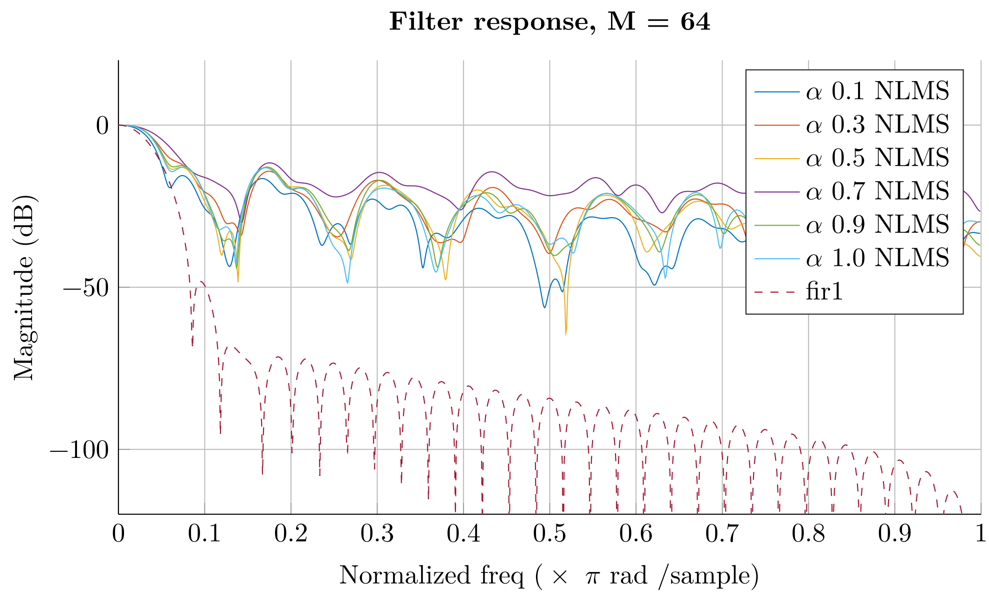


Figure 5.15: Filter response for analysis and synthesis prototype filter $M = 64$, NLMS. Max singular value E_s.

Chapter 6

Discussion, Conclusion and Further Works

Discussion

A selected simulations were picked out to show the overall findings. A quick glance on the plots shows some satisfactory results with better convergence. However, optimizing the analysis prototype filter is very hard. Only the first objective function and APA subband preconditioner strategy with $M = 32$ gave any improvements. In fact further investigation showed that when $P_H = K$ where K is the oversampling factor, the $\mathbf{QR}_x$ matrix is normal. Or very close to normal. ($\mathbf{R}_x$ is the rectangular autocorrelation matrix that corresponds to white noise.) Therefore we were able to use the eigenvaluespread as a criterion. When P_H was increased, e.g. $P_H = 4, 8$ for $M = 64, 128$ respectively, the convergence rate between optimized and unoptimized were fairly the same for APA subband preconditioner strategy. However, testing with different filter responses (e.g. least square method with different cutoff frequency) showed that very high stop band attenuation is more stable for increased step size μ. But low transition band, as in Figure 5.3, improves convergence speed when we have low step size μ. Therefore if we can live with higher excessive mean squared error then just generate the prototype filter with the window method. The alternative to increase filter length L_f to get a better filter response is not a viable option since we increase the computational cost and delay from the filter convolution. And a lower filter length ($L_f = L$) does not give better performance whether it is optimized or not. That is why $L_f = 2L$ was chosen. Though when we look at normalized misalignment plots in appendix, on page 44, we can see that the optimized filters gave poorer alignments. If the result is the same for highly correlated and non-stationary signal, e.g. speech signal, then the optimized analysis prototype filter will give a degraded performance than the reference filter [14]. Another important test that was conducted showed that we cannot use filters that are optimized for $M = 32$ for higher M e.g. $M = 512$ and expect better convergence. To wind up the discussion for optimization of analysis filter, notice the peculiar that there are no big differences between different α for lowpass signal ($\rho = 0.95$). But for highpass signal ($\rho = -0.95$) there is a slightly disparity with $\alpha = 0.9$, even though the filter response plot look almost the same. This just reinforce the difficulty of optimizing analysis filter.

However, most of the potential lies in optimizing the synthesis prototype filter. Both of our subband preconditioner strategies gave better convergence rate than without optimization. Since we omit $1/P_H$ factor in $\mathbf{C}_{i,NLMS}$ the overall step size is less. Hence it appears that the optimal filter try to compensate this. This is visible in Figure 5.9 as the overall filter response gain is a bit higher than its counterpart in Figure 5.8.

The question whether it is fair to absorb $1/P_H$ factor into μ might be asked. Well since the equation set can be written as

$$\mathbf{QR}_x = \sum_{i=0}^{L-1} \mathcal{G}_i \mathbf{C}_i \mathcal{F}_i^H \tilde{\mathcal{F}}_i^T \mathbf{R}_x = \mathbf{Q}\underline{r}_{dx} \tag{6.1}$$

we see that this factor can be cancelled. And at the time we optimize the filters we let μ to be 1. Of course we could have picked a μ that cancel the factor. Therefore as long we compare with the same subband preconditioner strategy, i.e. omit this factor throughout, this should not be a problem. Though, the excessive MSE for $M = 32$ with NLMS subband preconditioner strategy is a bit higher than the reference FFT-2 scheme. This problem does not occur with the APA subband preconditioner strategy.

Further on when we look at the learning curves for APA subband preconditioner strategy, Figure 5.4 and 5.5, we see that the α factor does not differentiate much. Similar when we look at the filter response plot Figure 5.8, each prototype filter's passband and transition band look the same. But for the NLMS subband preconditioner strategy we have an outlier with $\alpha = 0.1$. And when we look at the filter response in Figure 5.8 the passband does not align well with the others, too. Which suggest that NLMS subband preconditioner strategy is unstable when it comes to optimization of synthesis prototype filter. Nevertheless, it does imply that we can just use the maximum singular value as a criterion and omit stopband energy.

These plots, Figure 5.4-5.7, are plotted with the FFT-2 scheme as the reference. To get an indication on how unoptimized synthesis filters are compared to optimized synthesis filters with the DFT-FIR scheme, some plots are included in appendix from page 43. Clearly unoptimized synthesis filters of length $L_g = 2L$ give us poorer convergence performance compared to the FFT-2 scheme. However, it is also interesting to see that we can increase the filter length and get a convergence performance that coincide with the FFT-2 scheme. Actually, tests with optimized filter with tap length of $L_g = 2L$ or $L_g = 2P_H L$, the lower length gave better performance. Which can be seen in section A.5 on page 49. We can also see that the first objective function ($\alpha = 0.5$) outperforms the second objective function, with slightly better alignment.

Perhaps the most significant result is that we can use a synthesis prototype filter that is optimized for $M = 32$ ($P_H = 2$) and still get comparably the same convergence performance for a configuration with higher M. Optimizing for a lower M is faster than a bigger M. Included plots are shown in section A.6 on page 52. Here we have optimized for $M = 32$ in a configuration that runs with $M = 512$. We could have optimized for $M = 16$ ($P_H = 1$), but this is predicted to be a bad choice since we get the same filter response for both subband preconditioner strategies, i.e. $\mathbf{C}_{i,NLMS} = \mathbf{C}_{i,APA}$, just a filter scaling.

With the optimizing of analysis and synthesis filter we can also see that the APA preconditioner strategy is better than the NLMS preconditioner strategy. We also see the significant of the transition band. In the filter response plot for APA subband preconditioner strategy Figure 5.14 we can see that the green and teal curve, $\alpha = 0.9$ and $\alpha = 1.0$, are slightly off. Similar with the NLMS preconditioner strategy, the filter response plot Figure 5.15 shows the sharp transition band of the blue curve, $\alpha = 0.1$ which is also the only one that gave better convergence. What we can say is that we need to emphasize the stopband energy such that we get sharp transition band. Though, not always sharp transition band is good, e.g. section A.5 we see sharp transition band on the synthesis filter, but does not give better convergence. But this might be because the sharp transition band also introduces higher cutoff frequency.

Conclusion

Optimizing synthesis prototype filter gives greatest potential. Both objective functions are able to give viable filters, with $L_f = L_g = 2L$ and an L_g offset of L. But the first objective function is better. When $P_H = 2$ we get a normal $\mathbf{QR}_x$ matrix, resulting in a very easy optimization with the eigenvaluespread criterion. Lastly the APA subband preconditioner strategy describe the DSAF closed loop best when NLMS algorithm is used in each subband.

Further Works

Throughout this thesis we have been using the ratio $P_H = M/N$ for subband adaptive filter length. Question that arises is can we fix this ratio such that we have $P_H = 2$, or for any arbitrary length less than M/N without compromise convergence performance. This will reduce the complexity cost for the subband adaptive filtering.

Since one of the strength of DFT-FIR scheme is that we can have arbitrary M. Of course if we set $P_H = M/N$ then M must be a multiple of N. Though, there is a less restriction on choice on M. Whereas in the FFT-2 scheme we need P_H to be divisible by 2, i.e. $M = 2KN$, $K = 1, 2, 3, \dots$. Because of this there are some configurations we cannot compare DFT-FIR scheme with FFT-2. Thus making it hard to say whether DFT-FIR scheme would be a better choice. However, it would still be interesting to see whether optimized filter is better than unoptimized in configurations where we cannot use FFT-2 scheme.

Further on our optimized filters are not strictly linear phase, i.e. does not exhibit coefficient symmetry. It would be interesting to see if linear phase affects the performance. Or in the case where we optimize analysis and synthesis filter together, whether paraunitary filters, i.e. time reverse the synthesis filters, will affect the performance.

In addition, we have not tested with non-stationary signals. It would be interesting to see whether our optimized synthesis prototype filter works well for non-stationary signals, too. Perhaps we would have to increase the update rate a bit.

Another curiosity is to figure out how DFT-FIR scheme actually works. The removal of $L_g/2$ when L_g is even, is because this is the delay that is introduced when we run a (finite) signal through a multirate system. However, there is an edge effect at the end of the convolution. This edge effect of course has an intimate connection with the length of P_H. Further there is a need to know why $L_g = 2KL$, $K = 1, 2, 3, ...$ converges yet e.g. $L_g = 3L$ does not converge when we omit the first $L_g/2$ samples. This of course means we cannot use a much lower L_g than $2L$ with L_g offset of $L_g/2$. However, convergence does appear when we use L_g offset of L and lower L_g a bit below $L_g = 2L$. Whether this lowering of complexity affects performance is yet to see.

Challenging is to try to modify the DFT-FIR scheme or find a synthesis (and analysis) prototype filter such that it gives a satisfactory result for other type of filter banks, e.g. critically sampled cosine modulated and DFT filter banks. For critically sampled DFT filter banks we have the weight transform proposed by Merched et al. [15]. Thus we need to find a suitable synthesis prototype filter that outperforms the weight transform proposed by Merched et al. This might require new objective functions and constrains since these type of filter banks are plague with aliasing.

Finally an idea which is purely of theoretical and can't seem to have any practical value is to try to create a weight transform based on the decimated input signal, i.e. an adaptive weight transform. From the equation set we have $\mathbf{QR}_x$ therefore create a $\mathcal{G}$ such that $\mathbf{QR}_x$ is closed to the identity matrix. We need to estimate $\mathbf{R}_x$ based on the input signal and regularize it such that it has full rank.

Appendices

Appendix A

Further Simulation Results

Simulation settings are the same as described before.

DFT-FIR vs FFT-2 Scheme Unoptimized Filters

Here we see for $L_g = 2L$ we get poorer convergence when we do not optimized the filters. As we increase L_g, e.g. $L_g = 2P_H L$, we get similar result as with FFT-2 scheme. Unoptimized filters are created with fir1$(L_g - 1, 1/L)$ Matlab command.

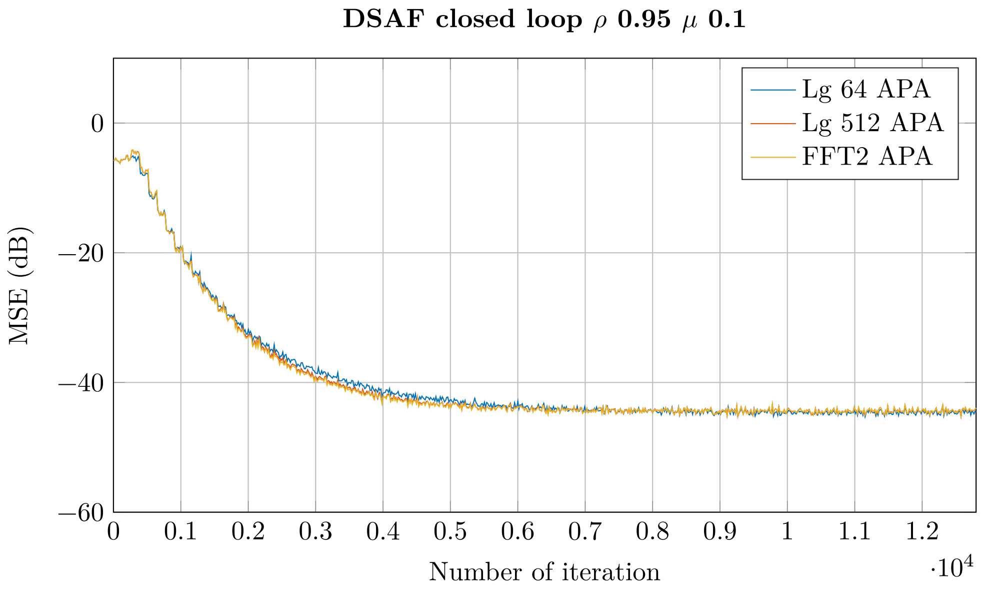


Figure A.1: Comparison between DFT-FIR vs FFT-2 Scheme. $M = 128$.

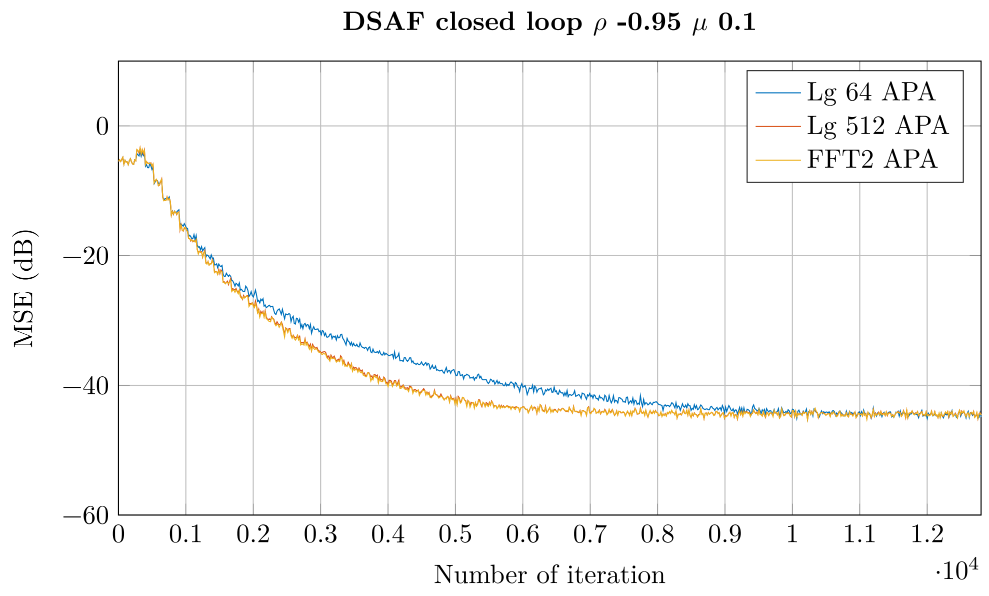


Figure A.2: Comparison between DFT-FIR vs FFT-2 Scheme. $M = 128$.

Optimization of Analysis Filters - Misalignment

These are the normalized misalignment, $\|\underline{h}_t - \underline{h}(k)\|_2 / \|\underline{h}_t\|_2$, plot for optimization of analysis filter simulation. The zigzag pattern in the plot is due to weight-update every M samples. Note that even if FFT-2 has better alignment, its convergence speed is not better. The objective function is eigenvaluespread and normality.

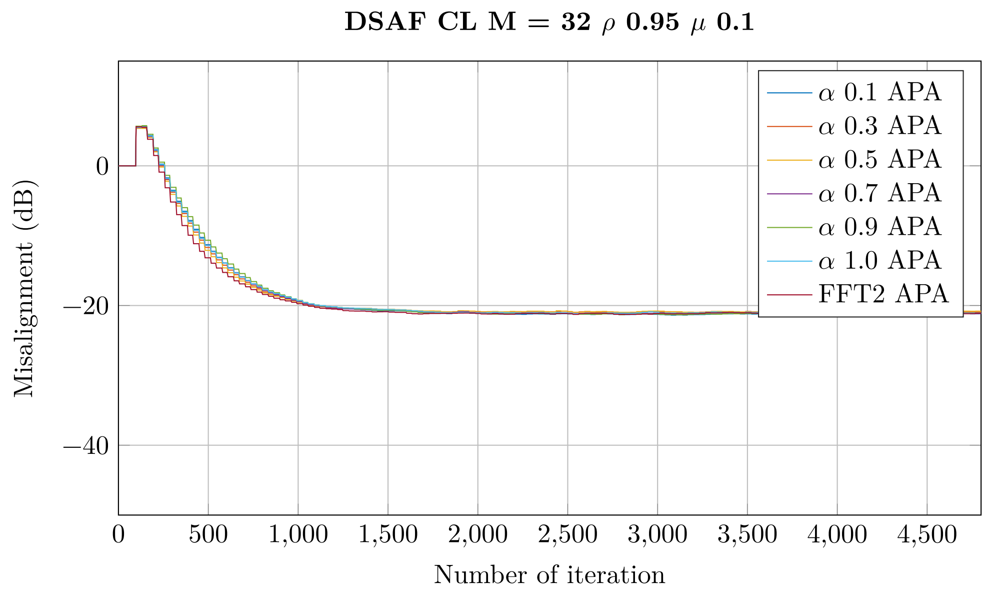


Figure A.3: Misalignment for optimized analysis filter. $M = 32$.

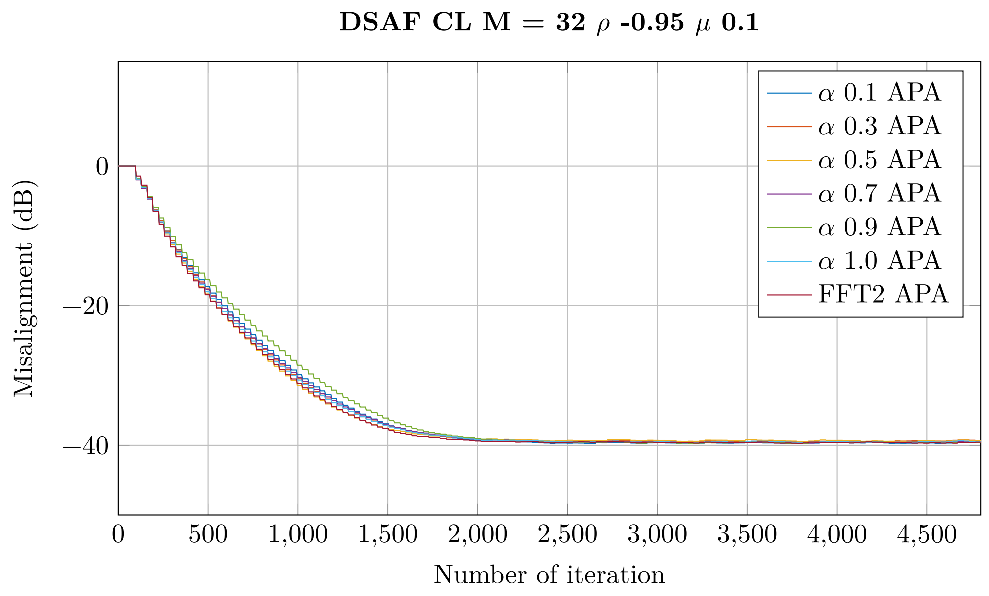


Figure A.4: Misalignment for optimized analysis filter. $M = 32$.

Optimization of Synthesis Filters - Misalignment

These are the normalized misalignment, $\|\underline{h}_t - \underline{h}(k)\|_2 / \|\underline{h}_t\|_2$, plot for optimization of synthesis filter simulation. The zigzag pattern in the plot is due to weight-update every M samples. The objective function is the largest singular value and stopband energy.

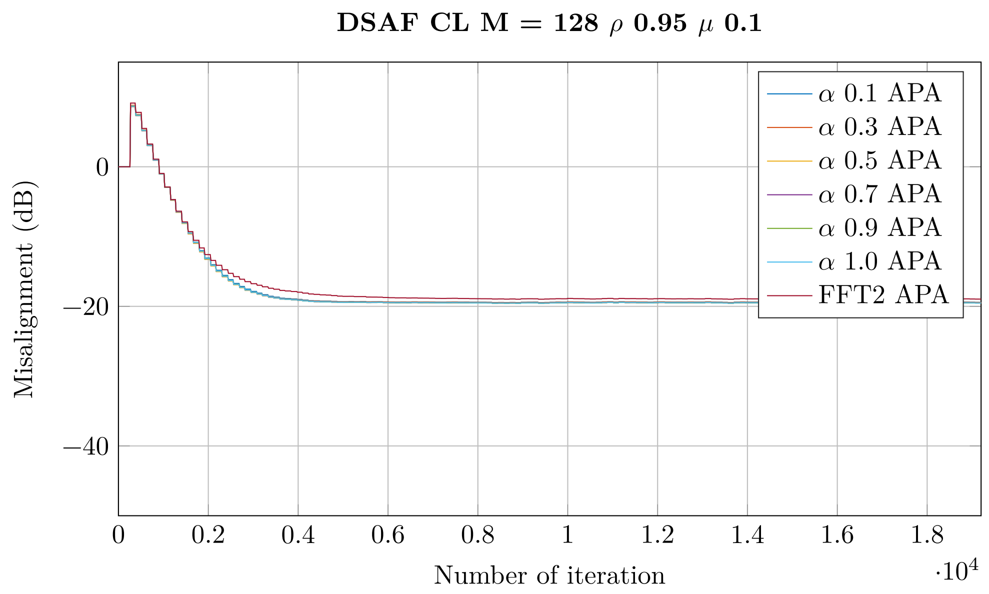


Figure A.5: Misalignment for optimized synthesis filter. $M = 128$.

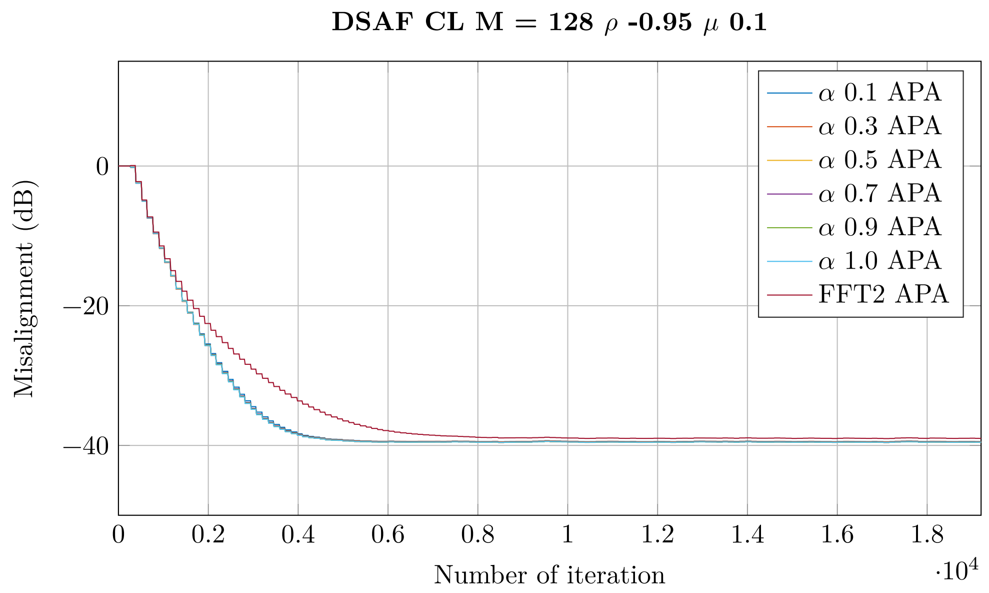


Figure A.6: Misalignment for optimized synthesis filter. $M = 128$.

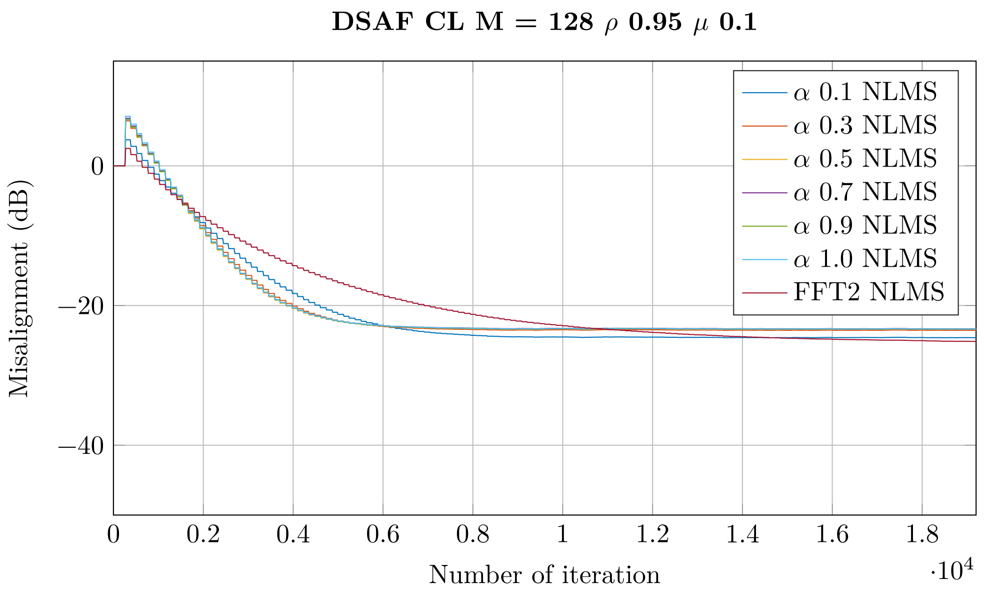


Figure A.7: Misalignment for optimized synthesis filter. $M = 128$.

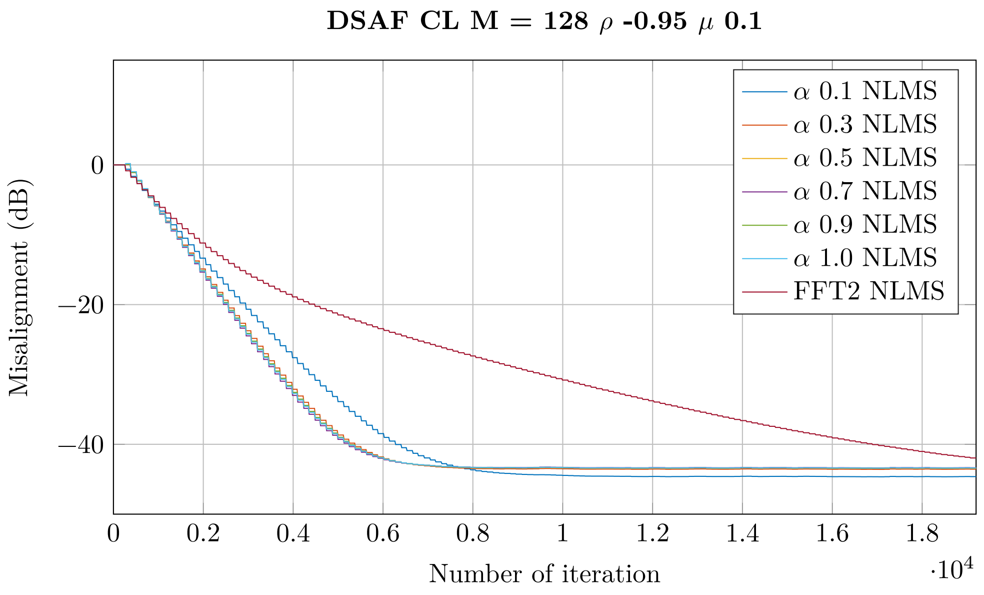


Figure A.8: Misalignment for optimized synthesis filter. $M = 128$.

Optimization of Analysis and Synthesis Filters - Misalignment

These are the normalized misalignment, $\|\underline{h}_t - \underline{h}(k)\|_2 / \|\underline{h}_t\|_2$, plot for optimization of analysis and synthesis filter simulation. The zigzag pattern in the plot is due to weight-update every M samples. The objective function is the largest singular value and stopband energy.

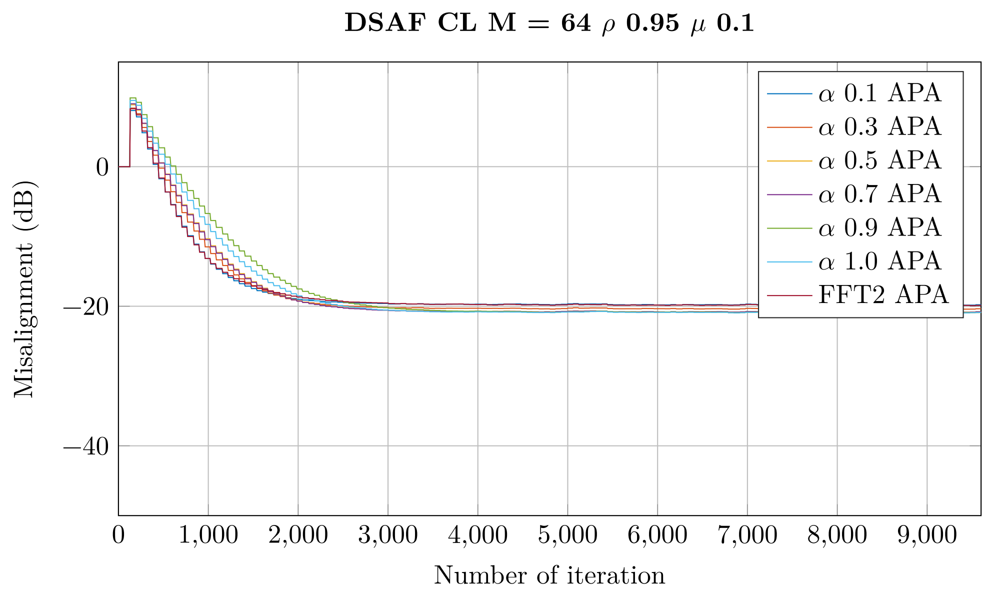


Figure A.9: Misalignment for optimized analysis and synthesis filter. $M = 64$.

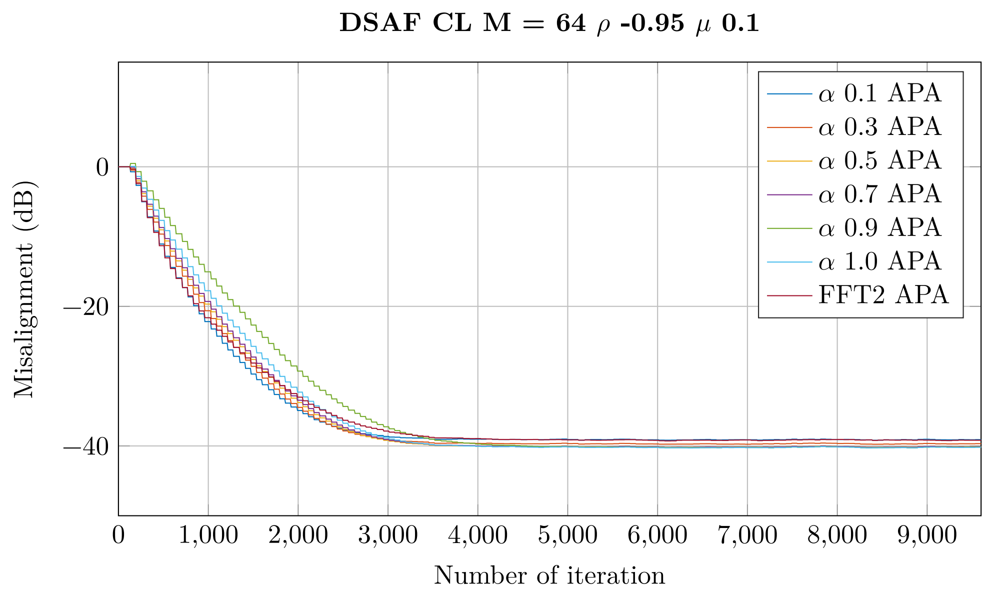


Figure A.10: Misalignment for optimized analysis and synthesis filter. $M = 64$.

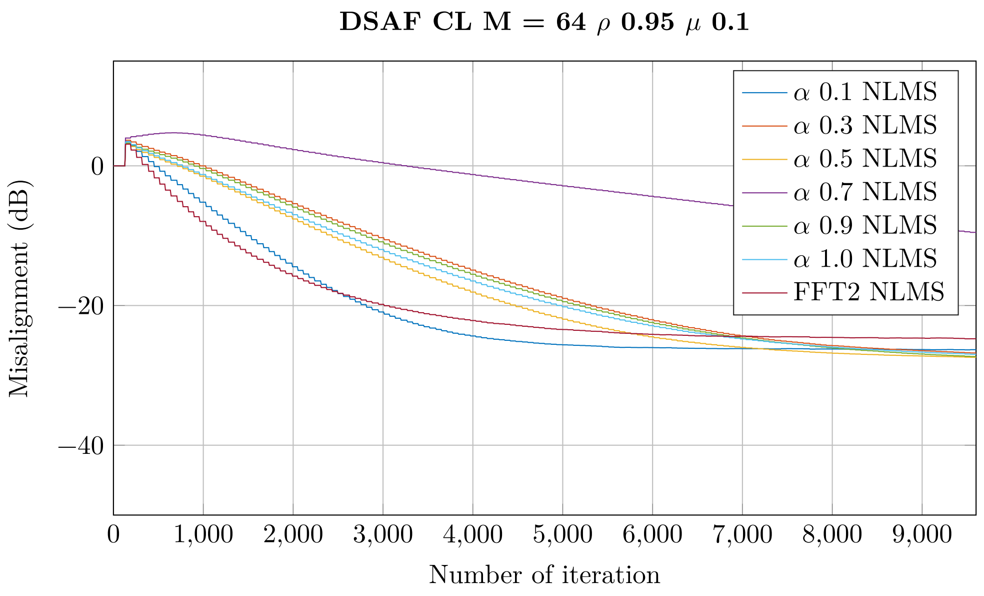


Figure A.11: Misalignment for optimized analysis and synthesis filter. $M = 64$.

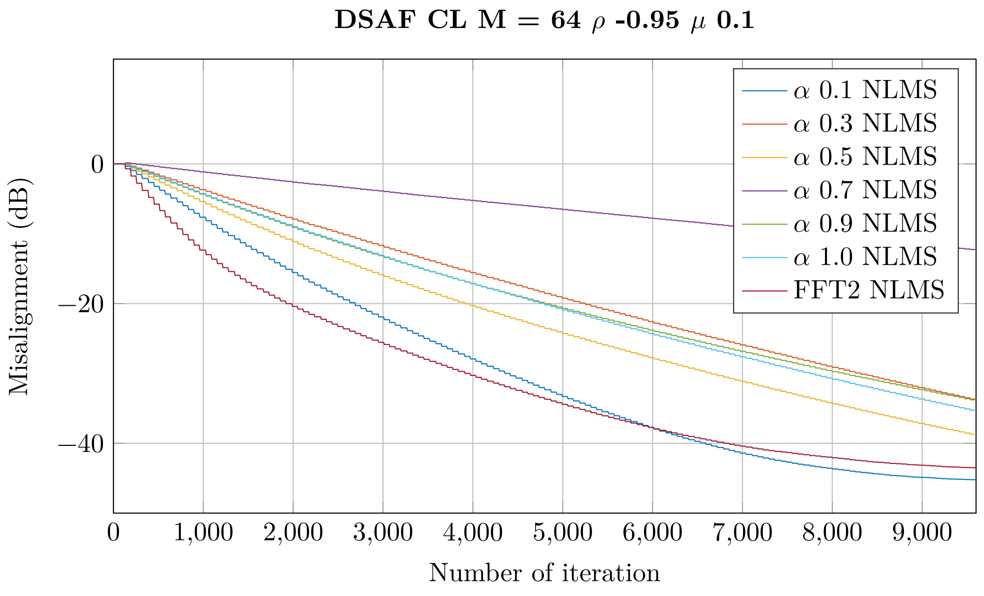


Figure A.12: Misalignment for optimized analysis and synthesis filter. $M = 64$.

Optimization of Synthesis Filters - High Fullband Tap Weights

Optimization with both objective function where $\alpha = 0.5$. The keywords dspread and normal refer to the first objective function, eigenvaluespread and normality. And smax and es is for the second objective function, maximum singular value and stopband energy. Analysis prototype filter is the same as before with $L_f = 2L$.

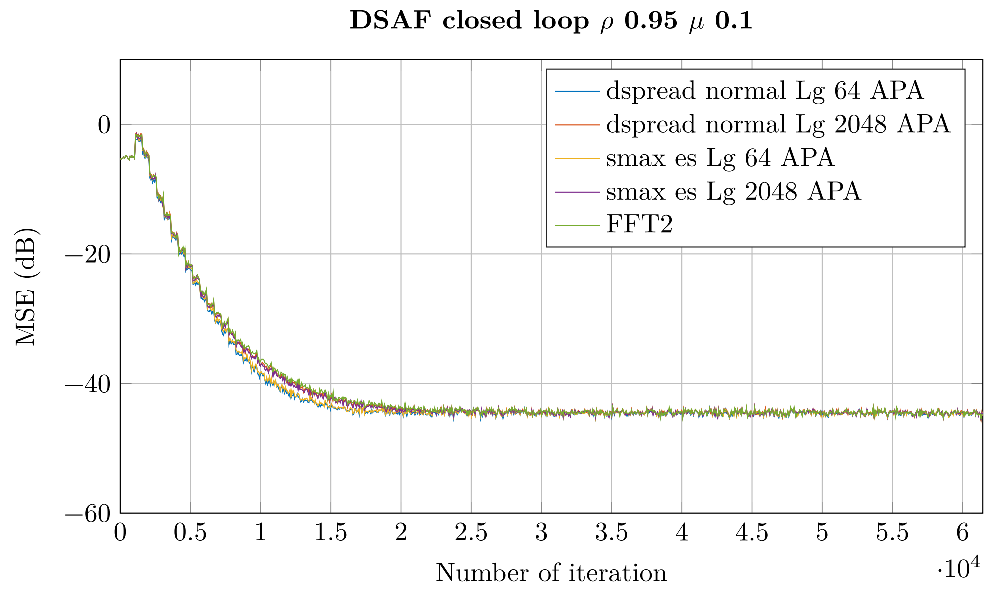


Figure A.13: Optimize synthesis filter for high adaptive taps. $M = 512$.

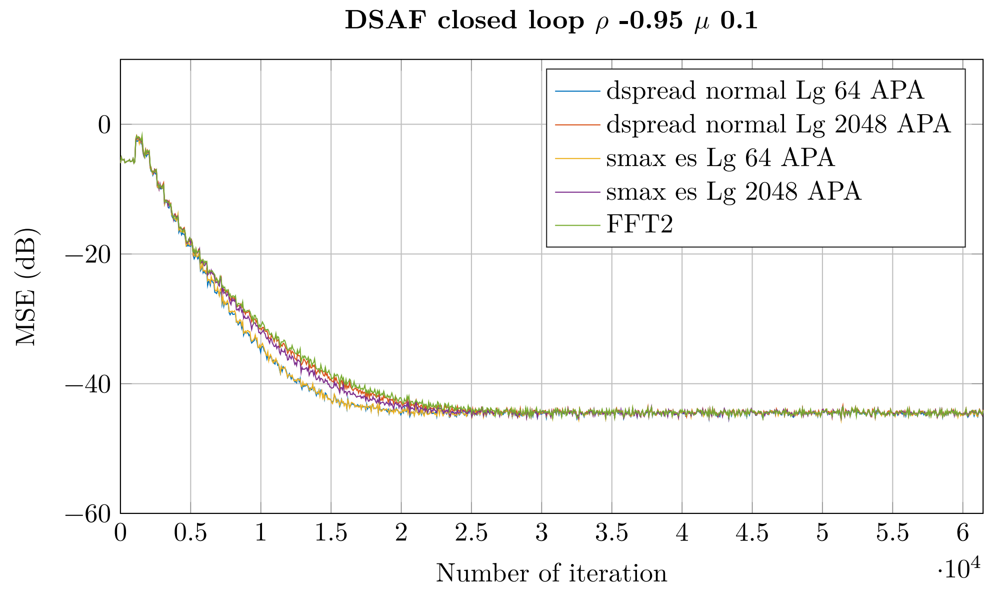


Figure A.14: Optimize synthesis filter for high adaptive taps. $M = 512$.

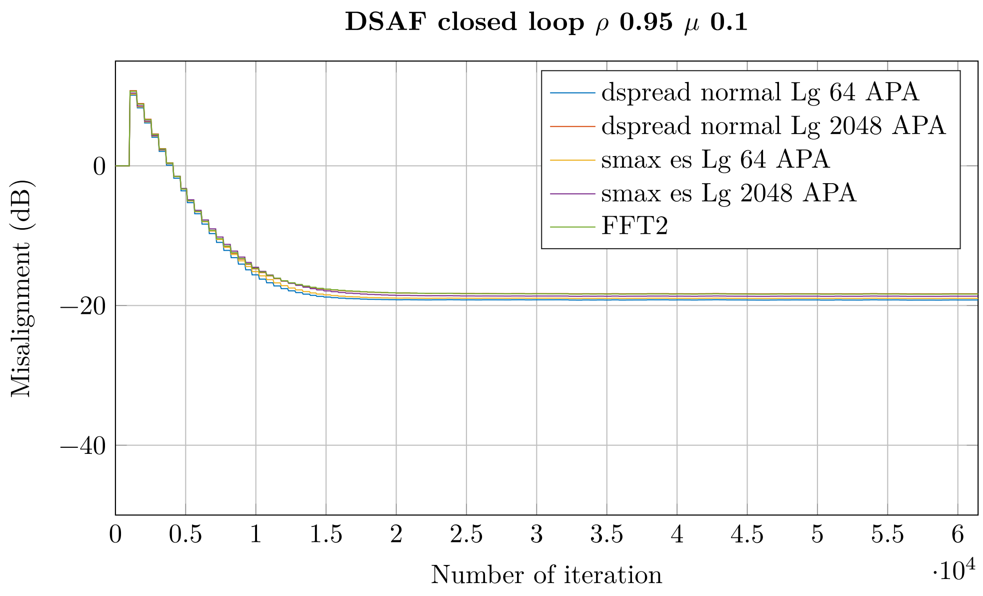


Figure A.15: Optimize synthesis filter for high adaptive taps, misalignment. $M = 512$.

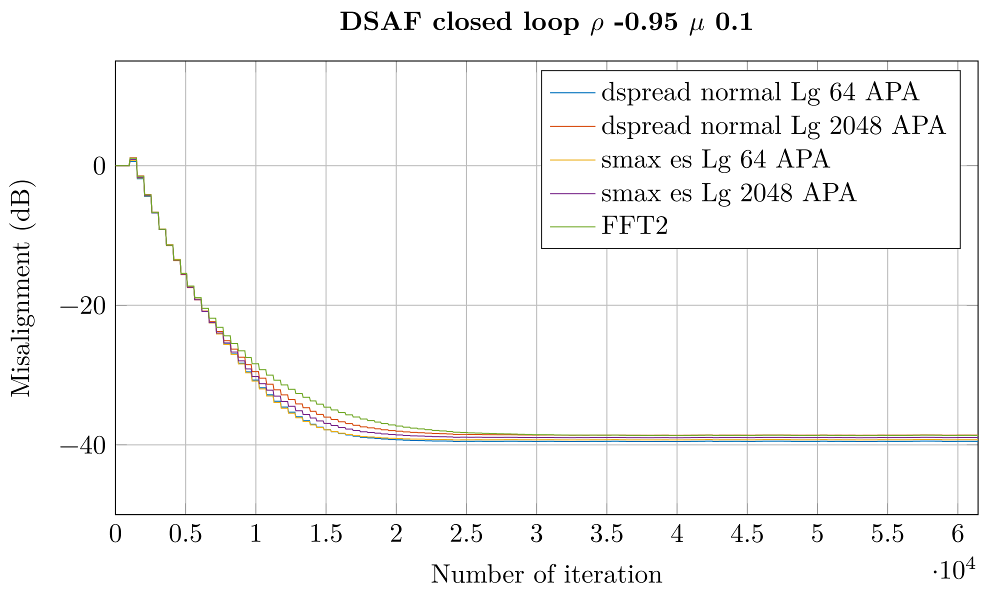


Figure A.16: Optimize synthesis filter for high adaptive taps, misalignment. $M = 512$.

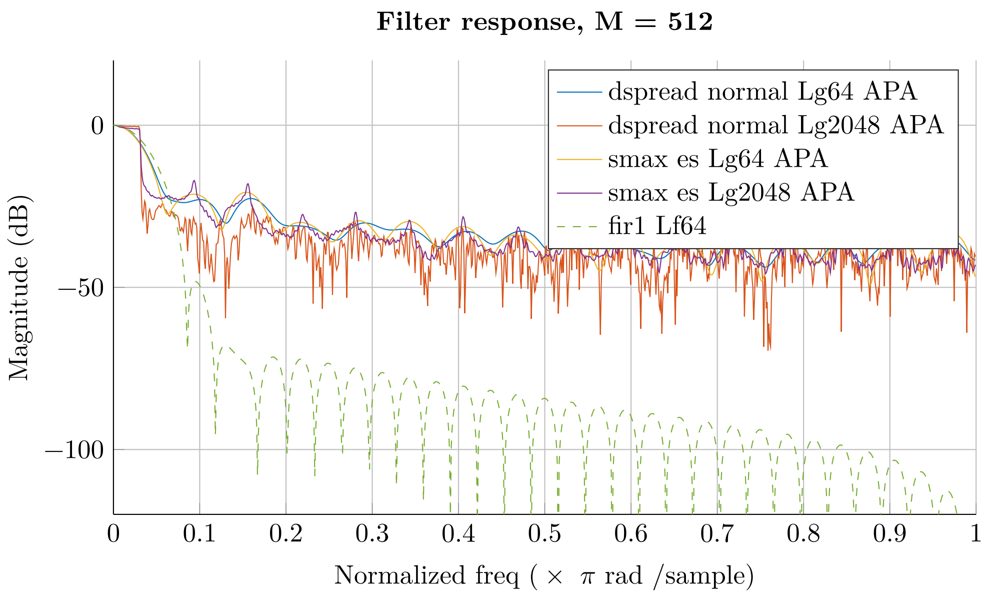


Figure A.17: Filter response for synthesis prototype filter high $M = 512$.

Mix Synthesis Filter Optimization

We are running a 512 tap fullband adaptive filter with synthesis filters that are optimized for $M = 32$. $\alpha = 0.5$. Note that we can optimized for $M = 16$ which corresponds to $P_H = 1$. But that will make $\mathbf{C}_{i,APA} = \mathbf{C}_{i,NLMS}$ for white signal. Hence optimized filters are the same for both subband preconditioner strategy.

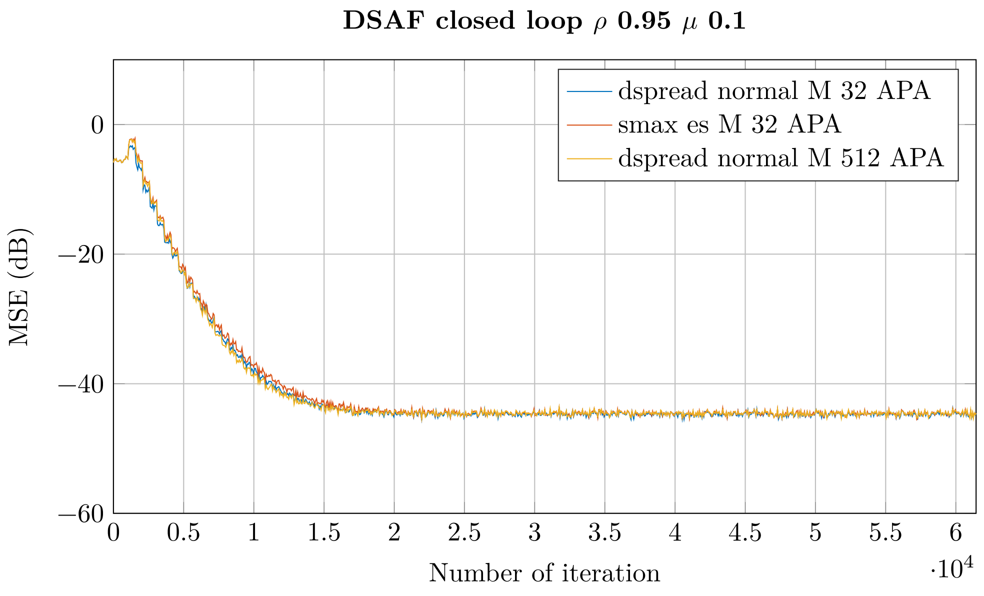


Figure A.18: Optimize synthesis filter for high adaptive taps. $M = 512$.

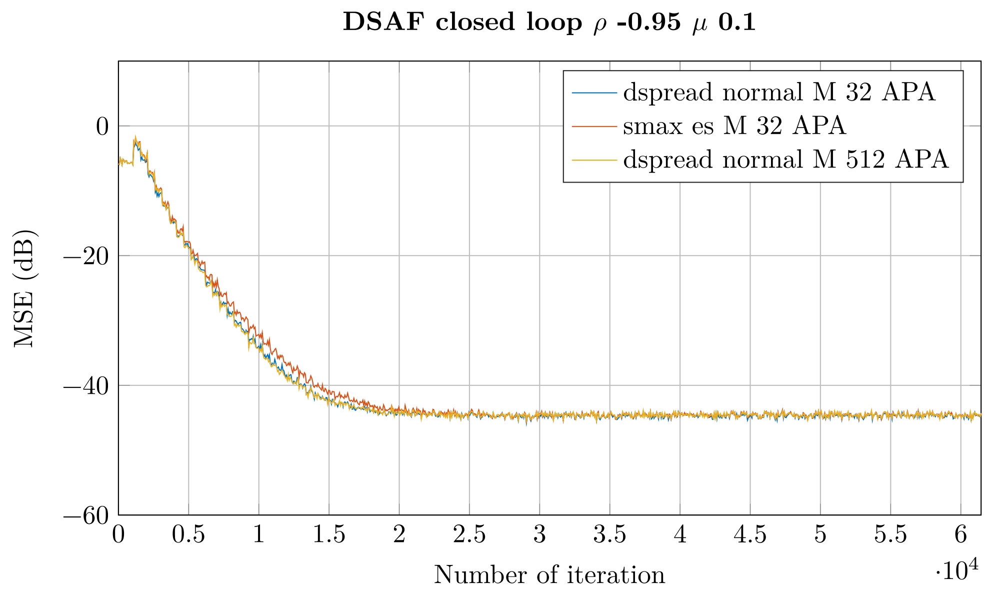


Figure A.19: Optimize synthesis filter for high adaptive taps. $M = 512$.

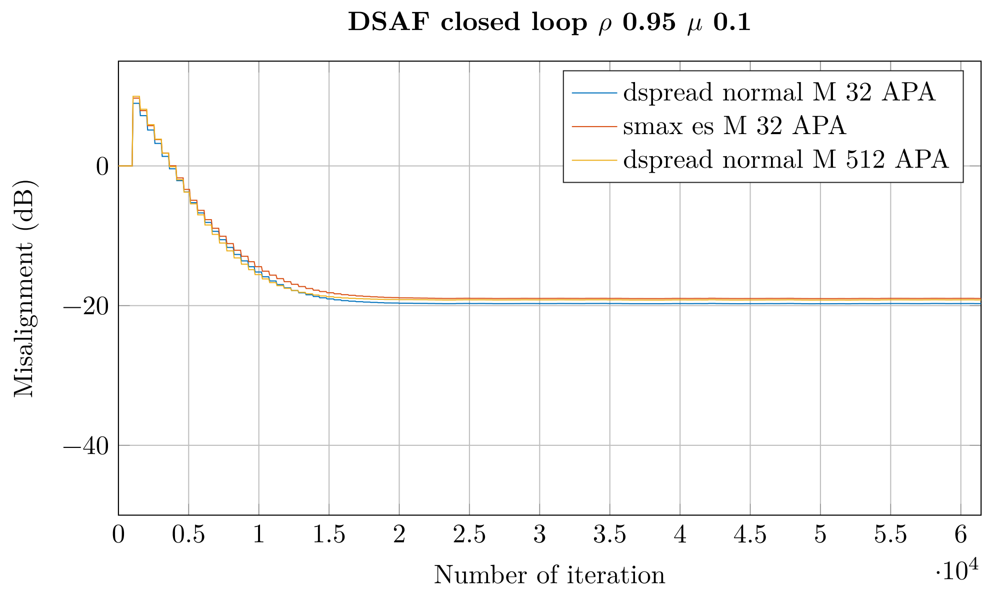


Figure A.20: Optimize synthesis filter for high adaptive taps, misalignment. $M = 512$.

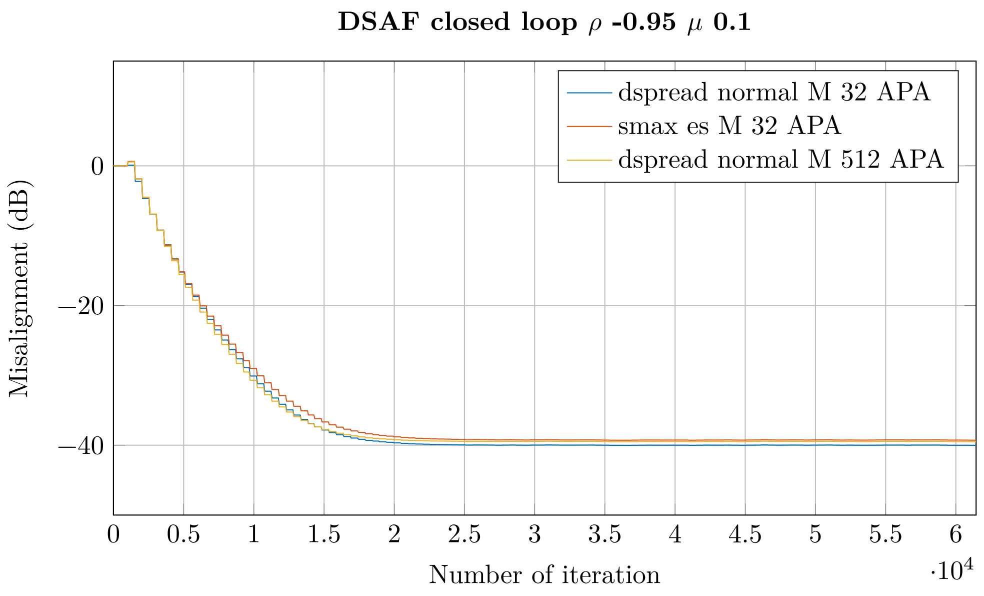


Figure A.21: Optimize synthesis filter for high adaptive taps, misalignment. $M = 512$.

Appendix B

Implication of Overdetermined Wiener-Hopf

From the Wiener-Hopf equation we know that

$$\mathbf{R}_x \underline{h} = \underline{r}_{dx} \tag{B.1}$$

is consistent for the overdetermined equation set, i.e. $\underline{r}_{dx}$ is in column space of $\mathbf{R}_x$. However, assume that $\underline{r}_{dx}$ is not in the column space of $\mathbf{R}_x$ and we wish to find an approximation $\underline{\hat{h}}$ such that $\|\underline{r}_{dx} - \mathbf{R}_x \underline{\hat{h}}\|_2^2$ is minimized. This is a least-squares problem from linear algebra. The minimum is satisfied when $\mathbf{R}_x \underline{\hat{h}} = \underline{\hat{r}}_{dx} = \operatorname{proj}_W \underline{r}_{dx}$, where W is the column space of $\mathbf{R}_x$. Any least-squares solution must satisfy

$$\mathbf{R}_x^H \mathbf{R}_x \underline{h} = \mathbf{R}_x^H \underline{r}_{dx} \tag{B.2}$$

from the fact that $\underline{r}_{dx} - \underline{\hat{r}}_{dx}$ is orthogonal to columns of $\mathbf{R}_x$.

We create now a Richardson iteration with a preconditioner $\mathbf{C}$

$$\underline{h}(k+1) = \underline{h}(k) + \mu \mathbf{C} \mathbf{R}_x^H [\underline{r}_{dx} - \mathbf{R}_x \underline{h}(k)] \tag{B.3}$$

and substitute with the simplest estimation of $\mathbf{R}_x$ and $\underline{r}_{dx}$,

$$\mathbf{R}_x = \underline{x}_e^*(k) \underline{x}^T(k) \tag{B.4}$$

$$\underline{r}_{dx} = d(k) \underline{x}_e^*(k) \tag{B.5}$$

where $\underline{x}_e(k)$ is an extended vector with greater length than $\underline{x}(k)$. Substitutes into the Richardson iteration gives us

$$\begin{aligned} \underline{h}(k+1) &= \underline{h}(k) + \mu \mathbf{C} [\underline{x}_e^*(k) \underline{x}^T(k)]^H [d(k) \underline{x}_e^*(k) - \underline{x}_e^*(k) \underline{x}^T(k) \underline{h}(k)] && \text{(B.6)} \\ &= \underline{h}(k) + \mu \mathbf{C} \underline{x}^*(k) \underline{x}_e^T(k) \underline{x}_e^*(k) e(k) && \text{(B.7)} \\ &= \underline{h}(k) + \mu \mathbf{C} \underline{x}^*(k) \|\underline{x}_e(k)\|_2^2 e(k) && \text{(B.8)} \end{aligned}$$

We get a normalized iteration with approaches in [10] by choosing

$$\mathbf{C} = [\epsilon \mathbf{I} + \mathbf{R}_x^H \mathbf{R}_x]^{-1} \tag{B.9}$$

follow with estimation of the autocorrelation matrix

$$\begin{aligned} \hat{\mathbf{C}} &= [\epsilon \mathbf{I} + [\underline{x}_e^*(k) \underline{x}^T(k)]^H \underline{x}_e^*(k) \underline{x}^T(k)]^{-1} && \text{(B.10)} \\ &= [\epsilon \mathbf{I} + \|\underline{x}_e(k)\|_2^2 \underline{x}^*(k) \underline{x}^T(k)]^{-1} && \text{(B.11)} \end{aligned}$$

where ϵ is a small regularization factor to make sure the inversion of the matrix possible. With the Sherman-Morrison formula, a special case of Woodbury formula (matrix inversion identity [8, 14]),

$$[\mathbf{A} + \underline{u}\underline{v}^T]^{-1} = \mathbf{A}^{-1} - \frac{\mathbf{A}^{-1}\underline{u}\underline{v}^T\mathbf{A}^{-1}}{1 + \underline{v}^T\mathbf{A}^{-1}\underline{u}} \tag{B.12}$$

applied on $\hat{\mathbf{C}}\underline{x}^*(k)$ we get

$$\hat{\mathbf{C}}\underline{x}^*(k) = \frac{1}{\epsilon + \|\underline{x}_e(k)\|_2^2\|\underline{x}(k)\|_2^2}\underline{x}^*(k) \tag{B.13}$$

Substitutes this estimation into the Richardson iteration yields

$$\underline{h}(k+1) = \underline{h}(k) + \frac{\mu\|\underline{x}_e(k)\|_2^2}{\epsilon + \|\underline{x}_e(k)\|_2^2\|\underline{x}(k)\|_2^2}\underline{x}^*(k)e(k) \tag{B.14}$$

Because ϵ is a very small number we can cancel $\|\underline{x}_e(k)\|_2^2$ and get the NLMS algorithm

$$\underline{h}(k+1) = \underline{h}(k) + \frac{\mu}{\epsilon + \|\underline{x}(k)\|_2^2}\underline{x}^*(k)e(k) \tag{B.15}$$

What we have shown is that whether we have overdetermined equation set or not with appropriate estimation and preconditioner strategy we can derive the NLMS algorithm.

But since NLMS is a special case of Affine Projection Algorithm (APA) in time domain [14], we could derive APA by using the estimation strategy,

$$\mathbf{R}_x = \frac{1}{L}\mathbf{X}_e^*(k)\mathbf{X}^T(k) \tag{B.16}$$

$$\underline{r}_{dx} = \frac{1}{L}\mathbf{X}_e^*(k)\underline{d}(k) \tag{B.17}$$

where

$$\mathbf{X}_e^*(k) = [\underline{x}_e^*(k), \underline{x}_e^*(k-1), \dots, \underline{x}_e^*(k-L+1)] \tag{B.18}$$

$$\mathbf{X}^T(k) = \begin{bmatrix} \underline{x}^T(k) \\ \underline{x}^T(k-1) \\ \vdots \\ \underline{x}^T(k-L+1) \end{bmatrix} \tag{B.19}$$

$$\underline{d}(k) = [d(k), d(k-1), \dots, d(k-L+1)]^T \tag{B.20}$$

The Richardson iteration with a preconditioner becomes

$$\begin{aligned} \underline{h}(k+1) &= \underline{h}(k) + \mu\mathbf{C}\mathbf{R}^H[\underline{r}_{dx} - \mathbf{R}_x\underline{h}(k)] && \text{(B.21)} \\ &= \underline{h}(k) + \mu\mathbf{C}[\mathbf{X}_e^*(k)\mathbf{X}^T(k)]^H[\mathbf{X}_e^*(k)\underline{d}(k) - \mathbf{X}_e^*(k)\mathbf{X}^T\underline{h}(k)] && \text{(B.22)} \\ &= \underline{h}(k) + \mu\mathbf{C}\mathbf{X}^*(k)\mathbf{X}_e^T(k)\mathbf{X}_e^*(k)\underline{e}(k) && \text{(B.23)} \end{aligned}$$

The scaling factor $1/L$ is omitted since we can absorb it into μ or be cancelled with the scaling factor in the preconditioner. Verily we write the preconditioner as

$$\mathbf{C} = [\epsilon\mathbf{I} + \mathbf{R}^H\mathbf{R}]^{-1} \tag{B.24}$$

follow with estimate

$$\hat{\mathbf{C}} = [\epsilon\mathbf{I} + \mathbf{X}^*(k)\mathbf{X}_e^T(k)\mathbf{X}_e^*(k)\mathbf{X}^T(k)]^{-1} \tag{B.25}$$

For brevity we shall omit time index k and apply the Woodbury formula, which is

$$[\mathbf{A} + \mathbf{UBV}]^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{U}[\mathbf{B}^{-1} + \mathbf{VA}^{-1}\mathbf{U}]^{-1}\mathbf{VA}^{-1} \tag{B.26}$$

With assumption $\mathbf{B} = \mathbf{X}_e^T\mathbf{X}_e^*$ of rank L, nonsingular i.e. columns of $\mathbf{X}_e^*$ are linear independent [19], we have then

$$[\epsilon\mathbf{I} + \mathbf{X}^*\mathbf{X}_e^T\mathbf{X}_e^*\mathbf{X}^T]^{-1} = \frac{1}{\epsilon}\mathbf{I} - \frac{1}{\epsilon}\mathbf{X}^*\left[(\mathbf{X}_e^T\mathbf{X}_e^*)^{-1} + \frac{1}{\epsilon}\mathbf{X}^T\mathbf{X}^*\right]^{-1}\frac{1}{\epsilon}\mathbf{X}^T \tag{B.27}$$

We then post multiply with $\mathbf{X}^*$,

$$\begin{aligned}
\hat{\mathbf{C}}\mathbf{X}^* &= \frac{1}{\epsilon}\mathbf{X}^* - \frac{1}{\epsilon}\mathbf{X}^*\left[(\mathbf{X}_e^T\mathbf{X}_e^*)^{-1} + \frac{1}{\epsilon}\mathbf{X}^T\mathbf{X}^*\right]^{-1}\frac{1}{\epsilon}\mathbf{X}^T\mathbf{X}^* && \text{(B.28)}\\
&= \frac{1}{\epsilon}\mathbf{X}^*\left\{\mathbf{I} - \left[(\mathbf{X}_e^T\mathbf{X}_e^*)^{-1} + \frac{1}{\epsilon}\mathbf{X}^T\mathbf{X}^*\right]^{-1}\frac{1}{\epsilon}\mathbf{X}^T\mathbf{X}^*\right\} && \text{(B.29)}\\
&= \frac{1}{\epsilon}\mathbf{X}^*\left[(\mathbf{X}_e^T\mathbf{X}_e^*)^{-1} + \frac{1}{\epsilon}\mathbf{X}^T\mathbf{X}^*\right]^{-1}(\mathbf{X}_e^T\mathbf{X}_e^*)^{-1} && \text{(B.30)}\\
&= \mathbf{X}^*\left[\epsilon(\mathbf{X}_e^T\mathbf{X}_e^*)^{-1} + \mathbf{X}^T\mathbf{X}^*\right]^{-1}(\mathbf{X}_e^T\mathbf{X}_e^*)^{-1} && \text{(B.31)}
\end{aligned}$$

where we in the intermediate step have used

$$\mathbf{I} = \left[(\mathbf{X}_e^T\mathbf{X}_e^*)^{-1} + \frac{1}{\epsilon}\mathbf{X}^T\mathbf{X}^*\right]^{-1}\left[(\mathbf{X}_e^T\mathbf{X}_e^*)^{-1} + \frac{1}{\epsilon}\mathbf{X}^T\mathbf{X}^*\right] \tag{B.32}$$

to to simplify further. With the simplification the Richardson iteration becomes

$$\begin{aligned}
\underline{h}(k+1) &= \underline{h}(k) + \mu\hat{\mathbf{C}}\mathbf{X}^*(k)\mathbf{X}_e^T(k)\mathbf{X}_e^*(k)\underline{e}(k) && \text{(B.33)}\\
&= \underline{h}(k) + \mu\mathbf{X}^*(k)[\epsilon(\mathbf{X}_e^T(k)\mathbf{X}_e^*(k))^{-1} + \mathbf{X}^T(k)\mathbf{X}^*(k)]^{-1}\underline{e}(k) && \text{(B.34)}
\end{aligned}$$

Essentially ϵ is a very small number which drift $\epsilon(\mathbf{X}_e^T\mathbf{X}_e^*)^{-1}$ to $\mathbf{0}$. Thus we can ommit this term. However, $\mathbf{X}^T\mathbf{X}^*$ (and $\mathbf{X}_e^T\mathbf{X}_e^*$) might be ill-conditioned therefore we add an extra term $\delta\mathbf{I}$ to regularize it. We therefore have

$$\underline{h}(k+1) = \underline{h}(k) + \mu\mathbf{X}^*(k)[\delta\mathbf{I} + \mathbf{X}^T(k)\mathbf{X}^*(k)]^{-1}\underline{e}(k) \tag{B.35}$$

Which is APA.

Further on, the question that has to be asked is whether it would be more appropriate to develop preconditioner strategy from

$$\mathbf{CR}_x^H\mathbf{R}_x\underline{h} = \mathbf{CR}_x^H\underline{r}_{dx} \tag{B.36}$$

or from an overdetermined equation set

$$\mathbf{QR}_x\underline{h} = \mathbf{Q}\underline{r}_{dx} \tag{B.37}$$

Appendix C

Delayless Subband Adaptive Filter Open Loop

The open loop share similar structure with a conventional SAF [14], where we do adaptive filtering in subband. Differences is we use a weight transform in delayless SAF and do a fullband filtering. With a conventional SAF we would have reconstructed the subband signals through a synthesis filter bank. The open loop cannot be set in the unified framework with a fullband autocorrelation matrix. However, with the Toeplitz matrix reordering mention earlier in DSAF closed loop we can still get some interesting result.

The Equation Set

We will start with defining a subband Wiener-Hopf equation

$$\mathbf{R}_{x,i}\underline{h}_i = \underline{r}_{dx,i} \tag{C.1}$$

For brevity we shall omit the subscript x and dx. We have then

$$\begin{align} \mathbf{R}_i &= \mathrm{E}\left\{\underline{x}_i^*(k)\underline{x}_i^T(k)\right\} \tag{C.2}\\ \underline{x}_i(k) &= [x_i(k), x_i(k-1), \ldots, x_i(k-(P_H-1))]^T \tag{C.3}\\ r_i(k) &= \mathrm{E}\left\{d_i(k+k)x^*(k)\right\} \tag{C.4}\\ \underline{r}_i &= [r_i(0), r_i(1), \ldots, r_i(P_H-1)]^T \tag{C.5}\\ \underline{h}_i &= [h_{i,0}, h_{i,1}, \ldots, h_{i,P_H-1}]^T \tag{C.6} \end{align}$$

Next is to try to express $\mathbf{R}_i$ and $\underline{r}_i$ in terms of their fullband equivalent. We will look at $\mathbf{R}_i$ first. $\underline{r}_i$ depends on whether it is closed or open loop.

The relationship between fullband $x(n)$ and subband $x_i(k)$ is

$$x_i(k) = \underline{f}_i^T\underline{x}(kN) \tag{C.7}$$

$\underline{f}_i$ is the analysis filter with length L_f. An element in (C.3) has the form

$$\begin{align} x_i(k-l) &= \underline{f}_i^T\underline{x}((k-l)N) \tag{C.8}\\ &= \underline{f}_i^T[x(kN-lN), \ldots, x(kN-lN-(L_f-1))]^T \tag{C.9}\\ &= \underline{f}_i^T[x(kN-lN), \ldots, x(kN-(lN+L_f-1)]^T \tag{C.10} \end{align}$$

By setting $l = P_H - 1$, the last element of x is $x(kN - ((P_H - 1)N + L_f - 1))$. The first element of x is $x(kN)$, with $l = 0$. We now construct a new vector $\underline{x}(kN)$ with length $L_x = (P_H - 1)N + L_f$.

$$\underline{x}(kN) = [x(kN), x(kN-1), \ldots, x(kN - (L_x - 1))]^T \tag{C.11}$$

With this new vector $\underline{x}(kN)$ we can write $\underline{x}_i(k)$ as

$$\underline{x}_i(k) = \begin{bmatrix} \underline{f}_i^T & \underline{0}^T & & \\ \underline{0}_N^T & \underline{f}_i^T & \underline{0}^T & \\ \underline{0}_N^T & \underline{0}_N^T & \underline{f}_i^T & \underline{0}^T \\ \vdots & \vdots & \ddots & \\ \underline{0}_N^T & \underline{0}_N^T & \ldots & \underline{f}_i^T \end{bmatrix} \begin{bmatrix} x(kN) \\ x(kN-1) \\ x(kN-2) \\ \vdots \\ x(kN-(L_x-1)) \end{bmatrix} \tag{C.12}$$

$$= \mathcal{F}_i^T \underline{x}(kN) \tag{C.13}$$

where $\underline{0}_N^T$ is a row vector with N zeroes, and $\underline{0}^T$ indicates we add zeroes to the matrix row such that the length of the row is equal L_x. Thus the analysis filter matrix $\mathcal{F}_i^T$ has dimension $P_H \times L_x$. The subband autocorrelation matrix can now be written in terms of fullband autocorrelation matrix.

$$\mathbf{R}_i = \mathrm{E}\left\{\underline{x}_i^*(k)\underline{x}_i^T(k)\right\} \tag{C.14}$$

$$= \mathrm{E}\left\{\mathcal{F}_i^H \underline{x}^*(kN) \mathcal{F}_i^T \underline{x}(kN)\right\} \tag{C.15}$$

$$= \mathcal{F}_i^H \, \mathrm{E}\left\{\underline{x}^*(kN)\underline{x}^T(kN)\right\} \mathcal{F}_i \tag{C.16}$$

$$= \mathcal{F}_i^H \mathbf{R}_x \mathcal{F}_i \tag{C.17}$$

Next we need to find the crosscorrelation vector, which can be written as

$$\underline{r}_i = \mathrm{E}\left\{d_i(k)\underline{x}_i^*(k)\right\} \tag{C.18}$$

$$= \mathrm{E}\left\{\underline{x}_i^*(k) d_i(k)\right\} \tag{C.19}$$

$$= \mathrm{E}\left\{\mathcal{F}_i^H \underline{x}^*(kN) \underline{d}^T(kN) \underline{f}_i\right\} \tag{C.20}$$

$$= \mathcal{F}_i^H \mathbf{R}_{dx} \underline{f}_i \tag{C.21}$$

where, similar to the autocorrelation matrix, $\mathbf{R}_{dx} = \mathrm{E}\left\{\underline{x}^*(kN)\underline{d}^T(kN)\right\}$ is the crosscorrelation matrix with dimension $L_x \times L_f$.

The subband Wiener-Hopf equation can now be written as

$$\mathcal{F}_i^H \mathbf{R}_x \mathcal{F}_i \underline{h}_i = \mathcal{F}_i^H \mathbf{R}_{dx} \underline{f}_i \tag{C.22}$$

When gathering everything into an expression we get

$$\begin{bmatrix} \mathcal{F}_0^H \mathbf{R}_x \mathcal{F}_0 & \mathbf{0} & \ldots & \mathbf{0} \\ \mathbf{0} & \mathcal{F}_1^H \mathbf{R}_x \mathcal{F}_1 & \ddots & \vdots \\ \vdots & \ddots & \ddots & \mathbf{0} \\ \mathbf{0} & \ldots & \mathbf{0} & \mathcal{F}_{L-1}^H \mathbf{R}_x \mathcal{F}_{L-1} \end{bmatrix} \begin{bmatrix} \underline{h}_0 \\ \underline{h}_1 \\ \vdots \\ \underline{h}_{L-1} \end{bmatrix} = \begin{bmatrix} \mathcal{F}_0^H \mathbf{R}_{dx} \underline{f}_0 \\ \mathcal{F}_1^H \mathbf{R}_{dx} \underline{f}_1 \\ \vdots \\ \mathcal{F}_{L-1}^H \mathbf{R}_{dx} \underline{f}_{L-1} \end{bmatrix} \tag{C.23}$$

0 is zero block matrix with dimension $P_H \times P_H$. The right hand side is the same as in the closed loop version. However, the left hand side is a bit tricky. For real input signal the fullband autocorrelation matrix is given by

$$\mathbf{R}_x = \begin{bmatrix} r_x(0) & r_x(1) & r_x(2) & \dots & r_x(L_x-1) \\ r_x(1) & r_x(0) & r_x(1) & \dots & r_x(L_x-2) \\ r_x(2) & r_x(1) & r_x(0) & \dots & r_x(L_x-3) \\ \vdots & \vdots & \vdots & & \vdots \\ r_x(L_x-1) & r_x(L_x-2) & r_x(L_x-3) & \dots & r_x(0) \end{bmatrix} \tag{C.24}$$

and

$$\mathscr{F}_i = \begin{bmatrix} \underline{f}_i & \underline{0}_N & \underline{0}_N & & \\ & \underline{f}_i & \underline{0}_N & \underline{0}_N & \\ & & \underline{f}_i & \underline{0}_N & \underline{0}_N \\ & & & \ddots & \vdots \\ & & & & \underline{f}_i \end{bmatrix} \tag{C.25}$$

where column j is a shift of Nj position downwards. Here is the interesting part. We wish to move Nj position upwards in each columns of $\mathscr{F}_i$ such that they are the same as the first column. This also means we can delete the first Nj columns of $\mathbf{R}_x$ that is associated with the column j of $\mathscr{F}_i$ and append arbitrary Nj columns to the end of the matrix. Let us look at an example with $N = 2$. The product between the first column in $\mathscr{F}_i$ and $\mathbf{R}_x$ gives us $\tilde{\mathscr{F}}_i^T \underline{r}_0$. Where the vector associate with $\mathbf{R}_x$ is

$$\underline{r}_0 = [r_x(L_x-1), r_x(L_x-2), \dots, r_x(0), \dots, r_x(L_x-1)]^T \tag{C.26}$$

The product between the second column of $\mathscr{F}_i$ and $\mathbf{R}_x$ gives us $\tilde{\mathscr{F}}_i^T \underline{r}_1$. To find the associated $\underline{r}_1$ we delete the first N columns of $\mathbf{R}_x$ and append suitable columns at the end such that we get a Toeplitz matrix, e.g.

$$\begin{bmatrix} r_x(2) & \dots & r_x(L_x-1) & r_x(L_x) & r_x(L_x+1) \\ r_x(1) & \dots & r_x(L_x-2) & r_x(L_x-1) & r_x(L_x) \\ r_x(0) & \dots & r_x(L_x-3) & r_x(L_x-2) & r_x(L_x-1) \\ \vdots & & \vdots & \vdots & \vdots \\ r_x(L_x-3) & \dots & r_x(0) & r_x(1) & r_x(2) \end{bmatrix} \tag{C.27}$$

thus $\underline{r}_1$ is given by

$$\underline{r}_1 = [r_x(L_x+1), r_x(L_x), \dots, r_x(0), \dots, r_x(L_x-3)]^T. \tag{C.28}$$

If we do this for the other columns of $\mathscr{F}_i$, then we have

$$\mathbf{R}_x \mathscr{F} = \tilde{\mathscr{F}}_i^T [\underline{r}_0, \underline{r}_1, \dots \underline{r}_{P_H-1}]. \tag{C.29}$$

$[\underline{r}_0, \underline{r}_1, \dots \underline{r}_{P_H-1}]$ can be viewed as an autocorrelation matrix with missing columns. The next part is to find an inverse weight transform such we can reintroduce the missing columns and make the whole equation resemble the Wiener-Hopf equation.

Appendix D

Matlab Code

Some of the Matlab codes are included. They contain some crufts and may not have appropriate comments. Only relevant files are listed here with a brief description. Though there are some helper functions included. Some files are taken from [14]. License from these files are in the Readme.html file. License for parfor_progress.m is in its own zip file.

CFmtx.m	Create $\mathbf{C}\mathcal{F}$ matrices of $\mathbf{Q}$.
dsaf_cl_dft_fir_high_M.m	Learning curves for $M = 512$.
dsaf_cl_dft_fir_vs_fft2.m	Compare unoptimized filter, DFT-FIR vs. FFT-2
dsaf_cl_dft_mix_filter.m	Optimized $M = 32$ in $M = 512$ setting.
dsaf_cl_dft_opt_analysis_filters.m	Create learning curves, analysis.
dsaf_cl_dft_opt_analysis_synthesis_filters.m	Create learning curves, synthesis and both.
dsaf_closed_loop_alg.m	DSAF algorithm.
opt_func_analysis_synthesis_fmincon.m	Optimization objective function.
opt_solver_analysis_synthesis_fmincon.m	Optimization solver.
Rmtx.m	Create the autocorrelation matrix.
WeightTransform_DFT_FIR.m	Weight transform DFT-FIR scheme.
WeightTransform_FFT2_Full.m	Weight transform FFT2 scheme using all subbands.
WeightTransform_G_DFT_FIR.m	Obtain $\mathcal{G}$ from DFT-FIR scheme.
WeightTransform_G_DFT_FIR2.m	Create $\mathcal{G}$ directly from synthesis filters.
WeightTransform_G_FFT2_Full.m	Obtain $\mathcal{G}$ from FFT-2 algorithm.
WeightTransform_G_Full_2_Half.m	Convert a full $\mathcal{G}$ to a half $\mathcal{G}$.

Chapter 7

Bibliography

[1] Helmut Bölcskei. “Oversampled Filter Banks and Predictive Subband Coders”. PhD thesis. Universität Wien, 1997.

[2] Helmut Bölcskei and Franz Hlawatsch. “Oversampled Cosine Modulated Filter Banks with Perfect Reconstruction”. In: *IEEE Trans. On Circuits and Systems II: Analog and Digital Signal Processing* Vol. 45.No. 8 (Aug. 1998), pp. 1057–1071.

[3] Zoran Cvetković. “Oversampled Modulated Filter Banks and Tight Gabor Frames in $l^2(\mathbb{Z})$”. In: *Proceedings of the IEEE International Conference on Acoust., Speech and Signal Processing (ICASSP95).* 1995, pp. 1456–1459.

[4] Michael Eiermann. “Fields of Values and Iterative Methods”. In: *Elsevier Science* (1993).

[5] Koen Eneman and Marc Moonen. “DFT modulated filter bank design for oversampled subband systems”. In: *Elsevier Signal Processing* Vol. 81 (2001), pp. 1947–1973.

[6] André Gilloire and Martin Vetterli. “Adaptive Filtering in Sub-Bands”. In: *Proceedings of the IEEE International Conference on Acoust., Speech and Signal Processing (ICASSP88).* Vol. 3. 1988, pp. 1572–1575.

[7] J.M. de Haan. *Convergence and Complexity Analysis of Delayless Subband Adaptive Filters.* Research report. Blekinge Institute of Technology, 2004.

[8] Monson H. Hayes. *Statistical Digital Signal Processing and Modeling.* Wiley, 1996.

[9] Jiaquan Huo, Sven Nordholm, and Zhuquan Zang. “New Weight Transform Schemes for Delayless Subband Adaptive Filtering”. In: *IEEE Global Telecommunications Conference, 2001.* Vol. 1. 2001, pp. 197–201.

[10] J.H. Husøy and M.S.E. Abadi. “Unified approach to adaptive filters and their performance”. In: *IET Signal Processing* Vol. 2.No. 2 (2008), pp. 97–109.

[11] John Håkon Husøy. “New insights into delayless subband adaptive filters”. In: *Proceedings of the APSIPA Annual Summit and Conference 2014.* 2014.

[12] Jonh Håkon Husøy. “A Circulantly Preconditioned NLMS-type Adaptive Filter”. In: *17th Intl. Conf. Radioelektronika.* Brno, Czech Republic, 2007, pp. 141–145.

[13] Jonh Håkon Husøy. “An Adaptive Filter Based on Multigrid Linear Equation Solvers”. In: *18th Intl. Conf. Radioelektronika.* Prague, 2008.

[14] Kong-Aik Lee, Woon-Seng Gan, and Sen M. Kuo. *Subband Adaptive Filtering: Theory and Implementation.* Wiley, 2009.

[15] Ricardo Merched, Paulo S. R. Diniz, and Mariane R. Petraglia. “A New Delayless Subband Adaptive Filter Structure”. In: *IEEE Trans. On Signal Processing* Vol. 47.No. 6 (June 1999), pp. 1580–1591.

[16] Dennis R. Morgan and James C. Thi. “A Delayless Subband Adaptive Filter Architecture”. In: *IEEE Trans. Signal Processing* Vol. 43.No. 8 (Aug. 1995), pp. 1819–1830.

[17] Michael K. Ng. *Iterative Methods for Toeplitz Systems.* Oxford, 2004.

[18] Truong Q. Nguyen. “The Design of Arbitrary FIR Digital Filters Using the Eigenfilter Method”. In: *IEEE Trans. On Signal Processing* Vol. 41.No. 3 (Mar. 1993), pp. 1128–1139.

[19] Kazuhiko Ozeki. *Theory of Affine Projection Algorithms for Adaptive Filtering.* Springer, 2015.

[20] John G. Proakis and Dimitris G. Manolakis. *Digital Signal Processing - Principles, Algorithms, and Applications 4th edition.* Pearson Education, 2007.

[21] P.P Vaidyanathan and Truong Q. Nguyen. “Eigenfilters: A New Approach to Least-Squares FIR Filter Design and Applications Including Nyquist Filters”. In: *IEEE Trans. On Circuits and Systems* Vol. cas-34.No. 1 (Jan. 1987), pp. 11–23.