DSP Exam

Thomas Debelle

KU LEUVEN

July 30, 2025

Contents

1 January 11 2024

1.1 Question 1

1. Chapter-2 p36 (“Example: HP anti-aliasing…”): Explain the relevance of the presented operations in the filter bank context (=Chapters 12-13-14).

> This demonstrates how that we can, after downsampling and upconverting, obtain the same signal back at the condition that we apply a HP Anti-Aliasing filter first.
>
> This HP AA is important as demonstrated in the formula page 33 where we can have aliasing after downconverting leading to *aliased* signal that cannot be reconstructed back. Right now, this result can be seen as stupid or useless, but the fact we can downsample and upsample and get the same signal is a sign of promising efficiency. Indeed, after downsampling, we can run a filter at a much lower frequency (the goal of downsampling) which is much more efficient (lower clock !). In other terms, we can perform an operation at lower speed which could be equivalent to a more power-hungry high speed operation.
>
> This equivalence is key for filter banks and is the motivation of their study.

2. Chapter-3 p.29 (“At the Rx, throw away L samples corresponding…”). Rewrite the formula for the case where the order of the FIR channel is larger than the cyclic prefix length.

> L should be adequately chosen to represent the FIR of the channel. Ideally L should be equal than the actual filter order and then the rest of the OFDM theory can be done.
>
> With L the cyclic prefix length and $\lambda =$ actual order $- L$ we can find the following:

$$\begin{bmatrix} y_k \\ y_{k+1} \\ y_{k+2} \\ y_{k+3} \\ y_{k+4} \\ \vdots \\ y_{k+N-1} \end{bmatrix} = \begin{bmatrix} h_L & h_{L-1} & \cdots & h_0 & 0 & \cdots & 0 \\ h_{L+1} & h_L & \cdots & h_1 & h_0 & \cdots & 0 \\ h_{L+2} & h_{L+1} & \cdots & h_2 & h_1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots & \vdots & \ddots & \vdots \\ 0 & h_{L+\lambda} & \cdots & h_{L+1} & h_L & \cdots & h_0 \end{bmatrix} \begin{bmatrix} x_{k+N-L} \\ x_{k+N-L+1} \\ \vdots \\ x_{k-1} \\ x_k \\ x_{k+1} \\ \vdots \\ x_{k+N-1} \end{bmatrix}$$

> In short, the matrix is **no longer** be made circulant which will void the OFDM principle. In practice, higher order h coefficient are near zero and thus they can be seen as 0 compared to h_0 and other low order coefficient. So the OFDM technique doesn’t just implode if you mispredict, just the math doesn’t really hold up and you may find some

> surprising excessive noise in your filter coefficient estimation.

3. Chapter-4 (“Filter Design”): Explain how the filter phase response is controlled in IIR filter design, and compare this to linear-phase FIR filter design.

> The phase response of a filter $H(z)$ is simply $\angle H(e^{j\omega})$. Moreover, we now that a z-transform is nothing else than $H(z) = \sum_{k=-\infty}^{\infty} h_k z^{-k} \implies H(e^{j\omega}) = \sum_{k=-\infty}^{\infty} h_k e^{-j\omega k}$. Then it is just a matter to cancel out the complex part which can only be done in **non-causal** filter. The only causal filter will be **linear phase** one with an added $e^{-j\omega L_0}$ where $L_0 = L/2$ or $L_0 = L - 1/2$ depending on a symmetric or anti-symmetric filter. So the phase response is controlled through 1. Real value symmetric anti-symmetric filters 2. Length of the filter. 1. ensures linear phase response and 2. the *steepness* of the linear phase function.
>
> The phase response of IIR filters is tougher to control. Having a linear phase is impossible in IIR or we will lead into stability issues. To control it we could think of tweaking the optimization problem and make larger weight on the frequency we want to avoid ripple, but this is really not straightforward and is not part of standard quadratic problems.
>
> A simpler way is to simply use filters designed for such response such as Bessel filter.
>
> In short, for FIR, by design we can choose a specific family through its coefficient (reducing flexibility by a factor 2) and then we use a common quadratic optimization problem that is easily solvable. For IIR, we can’t have linear phase by design, controlling the phase is hard and we will use some higher order optimization problem, solver do exists but will never be perfect, rely on well studied filters such as Bessel to simplify IIR filter deisgn with well controlled phase reponse.

4. Chapter-5 p.30 (“Derivation similar to p.22…”): Explain why the highlighted element in the first formula has to be a zero (unlike in p.21 and p.22).

> In the first formula on p.22, we are interested in block processing and to ultimately use cheap algorithm such as the DFT. The basic principle is to compute K samples at once and thus we will have a $Y = HU$ of dimension $L : 1 = (L : 2 * L)(2 * L : 1)$ where the H matrix is filled with all the filter coefficient in a Toeplitz fashion, then we can make it circulant which allow us to introduce DFT, … The size of the matrixes are directly linked with the length of the filter order as we need to create a square circulant matrix to use the theorem otherwise we will have a possible rank deficient matrix.
>
> The first algorithm can have extensive delay for high order filter, another approach is to use *poly-phase decomposition*. Important to remember that when we talk about a filter order L there is actually L+1 components in it. So this is why our polyphase decomposition D will have terms up to $L_D = (L + 1)/D$ order for each polyphase component. So this makes sense that our matrix won’t be “*filled anymore up until the bottom*”. So if we took back our first example with order L=4, we will have $L_D = 1$ with a 5-fold decomposition. And we have to remember that to transform our matrix into a square one with our trick, we need $2D$ columns in this case and so we **must virtually add** this void column at the end. This can be seen as weird but makes sense from a mathematical point of view.

5. Chapter-6: Draw a “parallel realization” of $H(z) = (1 + az^{-1})^{-1} + (1 + bz^{-1})^{-1}$. Insert all relevant quantization noise sources and compute the corresponding noise transfer functions. Can some of these noise sources be lumped into an equivalent noise source? Why (not)?

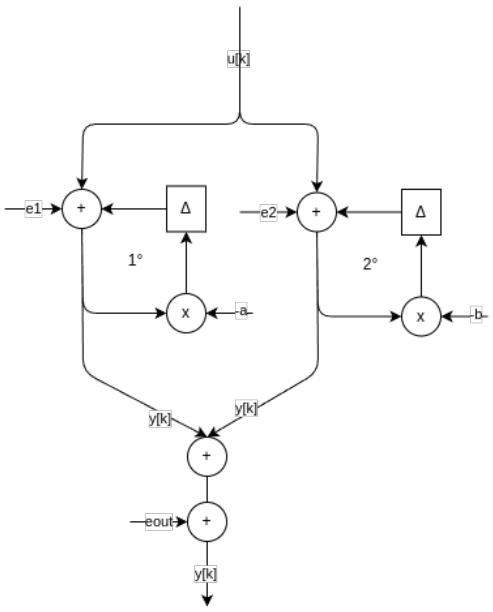

Figure 1: Realization using Transposed direct form

> For this, I chose a Transposed direct form and already lumped the components that could be lumped according to p.34. But if we take a closer look, the magnitude TF of e1 is simply 1 and same for e2, so we could also theoretically lump e1 and e2 to eout as they all share the same transfer function.
>
> Of course this is assuming that quantization noise can be a stochastic process.

1.2 Question 2

1. Chapter-7 p.24 ("MMSE cost function can be expanded as…"): How does the Wiener filter formula ($w_{WF} = (X_{uu})^{-1}X_{du}$) and/or its components (Xuu and Xdu) change in the case of a multi-channel FIR problem (as on p.19)?

> In the case of the multi-channel FIR filter, each one takes a different but the we compare it against the same desired signal as indicated on p.19. So we can write a new notation like $y_{k|i} = u_{k|i}^T w_i$ where the i index indicates the channel specific result. The error is then expressed as $e_k = d_k - \sum_i y_{k|i}$. To simplify notation let's denote $\sum_i y_{k|i} = y_k$, $\sum_i u_{k|i} = u_k$ and $\sum_i w_i = w$ By re-using the same MMSE criterion we can find that:

$$J_{mse}(w) = \mathbb{E}\{e_k^2\} = \varepsilon\{d_k^2\} + w^T\varepsilon\{u_k u_k^T\}w - 2w^T\varepsilon\{u_k d_k\}$$

> So, the optimal w_{wf} stays the same but what's behind $\mathbb{X}_{uu}$ and $\mathbb{X}_{du}$ changes. This is now the cross-correlation between each inputs, it now sums up everything together and the input signal is a sort of big soup of the various inputs. This makes sense as the desired output is compared with the results of all the filters.

2. Chapter-8: Explain how the LMS algorithm can be viewed as an RLS algorithm with a specific substitution for the input signal correlation matrix. Based on this link with RLS, provide an intuitive explanation for the statement that the convergence of the LMS depends on the correlation matrix eigenvalue spread.

The idea of the LMS algorithm is based on Wiener filter theory, but made practical. Typically, we don't have the statical informations about the signals so we average them out over time and this kinda behave as the expectancy $\mathbb{E}$. This gives $w_k = w_{k-1} + \mu u_k(d_k - u_k^T w_{k-1})$.

The LS is also in the same vein but we have a more direct memory idea, we average out the past k samples which should represents the observed correlation and cross-correlation matrices. This gives, after a few algebric tricks, the following results $w_k = w_{k-1} + \mathbb{X}_{uu}^{-1} u_k(d_k - u_k^T w_{k-1})$

Those two formulas are the same besides for the μ and $\mathbb{X}_{uu}^{-1} = (U_k^T U_k)^{-1}$, input signal correlation matrix. So if we swap them, we can obtain the other algorithm. So if the correlation matrix is the identity it's the same. Meaning that for a noiseless input signal we have actually the same results !

Those are some recursive algorithm so we must ensure stability by making sure the eigenvalues are not superior to 1 or we may run into some troubles. Ideally, as explained, we should have an unit matrix and thus no spread in the eigenvalues and the idea stable eigenvalues of 1. A bad spread of eingenvalues will be problematic for inverting the matrix. So good spread = good power correlation which is expected and is physical. Same idea can be derived for the stepsize selection in LMS algorithm. A bad spread is problematic and is symptomatic of a noisy, fast-changing environment that will lead to poor convergence as the algorithm can't lock into the actual input signal.

3. Chapter-9 p.38 ("Residual extraction…"): Consider the case where uk is an all-zero vector and dk is non-zero (and R[k] is full-rank). What would be the corresponding rotation angles and epsilon, and hence the a posteriori and a priori residual.

We know from slide p.33, that $R[k-1]$ is a upper triangular matrix as expected. To update our matrix, we append horizontally the new input and desired output. If the inputs are zeros, this means our matrix is already upper triangular and thus $\tilde{x_i} = x_i$ which means that $\theta = 0°$ based on notation from p.32

Taking back the notation from slide p.38, knowing that u_k is a zero-filled vector and that all $cos(\theta_i) = 1$, we can simplify this equation with $d_k = \epsilon$. The a posteriori residual is the desired output itself. This represents the error.

4. Chapter-10 p.20 ("The main trick…"): Redraw the signal flow graph when the "main trick" is used to remove the column with R15, R25, … Define the relevant epsilon-signals in the signal flow graph (with subscripts & superscripts).

Same as the figure presented in p.20, but we need to add one more cell below the column of $u(k+1)$ and the move our 3 cells connections one row down and deleting all the cells above the last celle from the row $u(k-4)$. Add back all the cells in the column $u(k-3)$ and connect its θ to the output.

5. In Chapter-11 p.25 ("Recursive Square-Root…"): Could residual extraction (cfr. Chapter 9) be added to this algorithm (would it also require only the "lower-right/lower part" as stated on p.23)? What exactly would be the meaning of the extracted residuals?

The residual signal should take the following form when using Givens transform to make a matrix triangular superior $\varepsilon = \frac{d_k - u_k^T w_{k,LS}}{\Pi_{i=1}^{L+1} cos(\theta_i)}$. This should be possible as RLS is simply a specific case of Kalman filtering as explained on slide p.17. So we could rewrite the **measurement noise** linked with Kalman as the actual residual extraction with $\varepsilon = \frac{d_k - C_k x_k - D_k u_k}{\Pi_{i=1}^{L+1} cos(\theta_i)}$ to generalize this concept.

Here we are getting the measurement noise that is also measuring the statespace model error with the residual noise from the inputs. But, this will not just be one element of the matrix as we are doing more than a rank-1 update in

> comparison with RLS. This would thus correspond to the vector … at the bottom right which won't be a simple value anymore.

1.3 Question 3

1. Chapter-12 p.12 ("Noise reduction…"): If D=4 (instead of D=3) and if the Gi's are not all equal to 1, what would be a condition for alias-free operation (a general formula is sufficient here) and what would be the resulting (linear) "distortion function"?

> If D=4, we will be having a *maximally decimated* FB which is more prone to possible alias. If we are looking at slide p.37 that is based on chapter 2 p.33, we can see we could have some potential aliasing but if we can guarantee that the $A_n(z)$ (alias function) is 0 then we can have an alias-free results. Adapting from those results give the following formula

$$Y(z) = \underbrace{1/N \sum_{n=0}^{N-1} G_n H_n(z) F(z)}_{T(z)} \cdot U(z) + \underbrace{1/N \sum_{n=0}^{N-1} G_n \sum_{\bar{n}=0}^{N-1} H_{\bar{n}}(zW^n) F_{\bar{n}}(z)}_{A_n(z)=0} \cdot U(zW^n)$$

> Where $T(z)$ is the distortion function.

2. Chapter-13 p.11 ("This can be verified…"). Provide an equivalent verification where in the first step R(z)E(z) is swapped with the upsampling (instead of the downsampling).

> So if we move the $R(z)E(z)$ to the right instead of the left we will find thanks to the noble identities that we again get $R(z^4)E(z^4)$. But this makes our life a tad harder as down converting can cause alias, the new formula is thus:

$$\mathbf{R}(z^4) \cdot \mathbf{E}(z^4) \cdot 1/4 \sum_{d=0}^{D-1} \cdot \begin{bmatrix} 1 \\ z^{-1}e^{2\pi jd} \\ z^{-2}e^{4\pi jd} \\ z^{-3}e^{6\pi jd} \end{bmatrix} U(ze^{-2\pi jd}) = \mathbf{R}(z^4) \cdot \mathbf{E}(z^4) \cdot 1/4 \begin{bmatrix} 4 \\ z^{-1}4 \\ z^{-2}4 \\ z^{-3}4 \end{bmatrix} \cdot U(z)$$

$$= \ldots = \begin{bmatrix} 1 \\ z^{-1} \\ z^{-2} \\ z^{-3} \end{bmatrix} \underbrace{\left(p_0(z^{-1}) + z^{-1}p_1(z^{-1}) + z^{-2}p_2(z^{-1}) + z^{-3}p_3(z^{-1})\right)}_{T(z)} U(z)$$

3. Chapter-13 p.31 ("Given E(z)…"). Consider the case with N=4, D=1 and LE=3. Construct a (simple) example with transfer functions Hi(z) and Fi(z) that provide perfect reconstruction.

> To make sure this problem is solvable, we must ensure we are not in a contrived case. L_R is not given but can be selected such that we are in an open case $L_R \geqslant \frac{1}{4-1}3 - 1$ so $L_R \geqslant 0$. To respect this property and to make our life easier we can select $E(z) = 1/4z^{-4}$ which is indeed a 4th order filter and then choose a zero order H filter:

$$\begin{bmatrix} 1/4z^{-4} & 1/4z^{-4} & 1/4z^{-4} & 1/4z^{-4} \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 1 \\ 1 \\ 1 \end{bmatrix} = z^{-4}$$

4. Chapter-14 p.27 ("Example-1: Define B(z4)…"):

a) Specify B(z) for the case where N=7 and D=4.

> Knowing that $N' = ND/gcd(N,D) = 4 * 7/1 = 28$ this means that we will fill up our matrix

$$B(z^4) = \begin{pmatrix} E_0(z^{28}) & z^{-20}E_{21}(z^{28}) & z^{-12}E_{14}(z^{28}) & z^{-4}E_7(z^{28}) \\ z^{-8}E_8(z^{28}) & E_1(z^{28}) & z^{-20}E_{22}(z^{28}) & z^{-12}E_{15}(z^{28}) \\ z^{-16}E_{16}(z^{28}) & z^{-8}E_9(z^{28}) & E_2(z^{28}) & z^{-20}E_{23}(z^{28}) \\ z^{-24}E_{24}(z^{28}) & z^{-16}E_{17}(z^{28}) & z^{-8}E_{10}(z^{28}) & E_3(z^{28}) \\ z^{-4}E_4(z^{28}) & z^{-24}E_{25}(z^{28}) & z^{-16}E_{18}(z^{28}) & z^{-8}E_{11}(z^{28}) \\ z^{-12}E_{12}(z^{28}) & z^{-4}E_5(z^{28}) & z^{-24}E_{26}(z^{28}) & z^{-16}E_{19}(z^{28}) \\ z^{-20}E_{20}(z^{28}) & z^{-12}E_{13}(z^{28}) & z^{-4}E_6(z^{28}) & z^{-24}E_{27}(z^{28}) \end{pmatrix}$$

b) Provide the corresponding proof (similar to the proof on p.27) (with explanation in words for non-trivial steps) that a 7-channel DFT-modulated filter bank is indeed obtained with this B(z).

> Here, we just started from a 28 poly-phase decomposition and referring from slide p.27 we have $F^{-1}B(z^4) \begin{bmatrix} 1 \\ z^{-1} \\ z^{-2} \\ z^{-3} \end{bmatrix} U(z)$
>
> The third vector represents the delay elements present in the signal flow graph. We will focus on the part in the middle first which yield the following result:

$$B(z^4) \begin{bmatrix} 1 \\ z^{-1} \\ z^{-2} \\ z^{-3} \end{bmatrix} = \begin{pmatrix} E_0(z^{28}) + z^{-21}E_{21}(z^{28}) + z^{-14}E_{14}(z^{28}) + z^{-7}E_7(z^{28}) \\ z^{-8}E_8(z^{28}) + z^{-1}E_1(z^{28}) + z^{-22}E_{22}(z^{28}) + z^{-15}E_{15}(z^{28}) \\ z^{-16}E_{16}(z^{28}) + z^{-9}E_9(z^{28}) + z^{-2}E_2(z^{28}) + z^{-23}E_{23}(z^{28}) \\ z^{-24}E_{24}(z^{28}) + z^{-17}E_{17}(z^{28}) + z^{-10}E_{10}(z^{28}) + z^{-3}E_3(z^{28}) \\ z^{-4}E_4(z^{28}) + z^{-25}E_{25}(z^{28}) + z^{-18}E_{18}(z^{28}) + z^{-11}E_{11}(z^{28}) \\ z^{-12}E_{12}(z^{28}) + z^{-5}E_5(z^{28}) + z^{-26}E_{26}(z^{28}) + z^{-19}E_{19}(z^{28}) \\ z^{-20}E_{20}(z^{28}) + z^{-13}E_{13}(z^{28}) + z^{-6}E_6(z^{28}) + z^{-27}E_{27}(z^{28}) \end{pmatrix}$$

> Already, we can feel something special when looking at the matrix, the $z^{-\delta}$ is "aligned" with its polyphase decomposition ! Something even more spectacular, is that they are evenly spaced by 7 the channel decomposition value ! So we can see that actually, each rows is a lower rate polyphase decomposition namely:

$$E_0(z^{28}) + z^{-21}E_{21}(z^{28}) + z^{-14}E_{14}(z^{28}) + z^{-7}E_7(z^{28}) = E_0(z^7)$$

> If we remember, a polyphase decomposition is selecting the elements spaced by D, If we divide our polyphase in 28 pieces and we select the zeors, the seventh, … this becomes the same as a lower rate decomposition. This idea can be repeated for the second line, but we must remove the z^{-1} factor as it is relative to the lower order factor.

$$B(z^4)\begin{bmatrix}1\\z^{-1}\\z^{-2}\\z^{-3}\end{bmatrix} = \begin{pmatrix}E_0(z^7)\\z^{-1}E_1(z^{28})\\z^{-2}E_2(z^{28})\\z^{-3}E_3(z^{28})\\z^{-4}E_4(z^{28})\\z^{-5}E_5(z^{28})\\z^{-6}E_6(z^{28})\end{pmatrix}$$