DSP Exam Questions

An Open-Source Summary

Thomas Debelle & Anonymous student

KU LEUVEN

September 26, 2025

Contents

Prompt:

```
Answer all of those equations, answer following the same structure, use the provided chapter and use inline Latex in yours answers:
```

1 10 August 2020

1.1 Question 1

1. **Explain how WLS for FIR filter design reduces the degrees of freedom when imposing a linear phase requirement.**

   For a Finite Impulse Response (FIR) filter, imposing a linear phase requirement significantly reduces the number of independent coefficients, thereby lowering the degrees of freedom in the design process. A linear phase FIR filter is typically achieved by designing the filter with a **symmetric or anti-symmetric impulse response**.

   For example, for a causal linear-phase FIR filter of length $L+1$ (where L is the filter order) with a symmetric impulse response and L being even, the impulse response coefficients $h[k]$ satisfy $h[k] = h[L-k]$ for $k = 0, \ldots, L$. This means that if you determine $h, h, \ldots, h[L/2]$, the remaining coefficients $h[L/2+1], \ldots, h[L]$ are automatically determined by the symmetry property.

   The frequency response of such a filter can be written as a real-valued function $G(\omega)$ multiplied by a linear phase term $e^{-j\omega L/2}$:

$$H(e^{j\omega}) = e^{-j\omega L/2} \cdot \sum_{k=0}^{L/2} d_k \cos(\omega k)$$

Here, d_k are combinations of the $h[k]$ coefficients, where $d_0 = h[L/2]$ and $d_k = 2h[L/2 - k]$ for $k > 0$. Instead of optimizing $L + 1$ independent coefficients $h[k]$, the Weighted Least Squares (WLS) design, for example using Type-1 linear-phase filters, only needs to optimize $L/2 + 1$ coefficients d_k. This is a **reduction in degrees of freedom** from $L + 1$ to $L/2 + 1$.

2. **Can QRD be used in WLS FIR filter design? How? Chapter 9, QRD for LS estimation?**

   Yes, the **QR Decomposition (QRD) can be used in Weighted Least Squares (WLS) FIR filter design**. The WLS design problem for FIR filters can be formulated as a quadratic optimization problem. If a discrete set of sample frequencies is used for the optimization, this problem can be transformed into a **linear least squares (LS) problem for an overdetermined set of equations**.

   Chapter 9 explains that the LS problem, typically written as $\min_w ||d - Uw||_2^2$, can be solved using QRD. The QRD of the matrix U (the input matrix in the LS problem) is $U = QR$, where Q is an orthogonal matrix ($Q^TQ = I$) and R is an upper triangular matrix.

   By substituting $U = QR$ into the LS problem and multiplying by Q^T, the minimization problem becomes:

   $$\min_w ||d - Uw||_2^2 = \min_w ||Q^T(d - Uw)||_2^2 = \min_w ||Q^Td - Q^TUw||_2^2$$

   Let $Q^Td = \begin{bmatrix} z \\ \star \end{bmatrix}$ and $Q^TU = \begin{bmatrix} R \\ 0 \end{bmatrix}$. The problem simplifies to:

   $$\min_w \left\| \begin{bmatrix} z \\ \star \end{bmatrix} - \begin{bmatrix} R \\ 0 \end{bmatrix} w \right\|_2^2 = \min_w ||z - Rw||_2^2 + || \star ||_2^2$$

   The minimization is achieved by setting $z - Rw = 0$, which yields the system of equations $Rw = z$. Since R is an upper triangular matrix, this system can be efficiently solved for w (the filter coefficients) using **back substitution**.

   Therefore, the discretized WLS FIR filter design problem can be cast as an LS problem, and then QRD, followed by back substitution, can be applied to find the optimal filter coefficients.

3. **When WLS FIR filter design is used for filters with an arbitrary required phase response, how does the design procedure change? Give relevant formulas.**

   When WLS FIR filter design is used for filters with an **arbitrary required phase response**, the fundamental assumption of a linear phase, which leads to a symmetric or anti-symmetric impulse response, **cannot be made**. This changes the design procedure significantly:

   - **Impulse Response:** The impulse response coefficients $h[k]$ (or b_k in the source) would be general, potentially complex-valued, and would not be constrained by symmetry. All $L + 1$ coefficients $h, \ldots, h[L]$ become independent design variables, increasing the degrees of freedom.
   - **Frequency Response:** The filter's frequency response $H(e^{j\omega})$ would be a general complex-valued function, as it would not necessarily factor into a real-valued amplitude response and a linear phase term. The transfer function for an FIR filter is $H(z) = \sum_{k=0}^{L} b_k z^{-k}$. Its frequency response is:

     $$H(e^{j\omega}) = \sum_{k=0}^{L} b_k e^{-j\omega k}$$

     This expression directly involves the complex exponentials and the coefficients b_k.

- **Desired Response:** The desired frequency response $H_d(\omega)$ would also be an arbitrary complex-valued function, specifying both desired magnitude and phase at each frequency.
- **Optimization Criterion:** The WLS optimization criterion would directly minimize the weighted squared difference between the complex-valued filter's frequency response and the desired complex frequency response. The formulation would be:

$$\min_{b_0,\ldots,b_L} \int_{-\pi}^{+\pi} W(\omega)|H(e^{j\omega}) - H_d(\omega)|^2 d\omega$$

where $W(\omega) \geq 0$ is the weighting function. This typically leads to a more complex optimization problem compared to the linear phase case, as the variables are generally complex, and the objective function involves the squared magnitude of a complex difference.

1.2 Question 2

1. **For overdetermined Ax=b, LS is x_LS=(A$^{T*A)}$(-1) Aˆ T*b, manipulate this to justify back substitution in QRD RLS.**

   Given an overdetermined system of linear equations $Ax = b$, the Least Squares (LS) solution is $x_{LS} = (A^T A)^{-1} A^T b$. To justify back substitution in the context of QRD Recursive Least Squares (RLS), we introduce the QR Decomposition of matrix A. The QR decomposition of A is given by $A = QR$, where Q is an orthogonal matrix ($Q^T Q = I$) and R is an upper triangular matrix.

   Substitute $A = QR$ into the LS solution:

$$x_{LS} = ((QR)^T QR)^{-1} (QR)^T b$$

$$x_{LS} = (R^T Q^T QR)^{-1} R^T Q^T b$$

   Since $Q^T Q = I$ (due to Q being orthogonal), this simplifies to:

$$x_{LS} = (R^T R)^{-1} R^T Q^T b$$

   This expression gives the LS solution. To justify back substitution, we consider the original LS minimization problem:

$$\min_x ||Ax - b||_2^2$$

   We can multiply the argument of the norm by the orthogonal matrix Q^T without changing the norm (since orthogonal transformations preserve Euclidean norm):

$$\min_x ||Q^T(Ax - b)||_2^2 = \min_x ||Q^T Ax - Q^T b||_2^2$$

   Now, substitute $A = QR$:

$$\min_x ||Q^T(QR)x - Q^T b||_2^2 = \min_x ||(Q^T Q)Rx - Q^T b||_2^2 = \min_x ||Rx - Q^T b||_2^2$$

   Let $Q^T b = \begin{bmatrix} z \\ \star \end{bmatrix}$ where z is the portion corresponding to the rows of R, and $\star$ is the remaining part (typically zero for

the case A is $k \times (L+1)$ and R is $(L+1) \times (L+1)$ where $k > L+1$). The transformed problem becomes:

$$\min_x \left\| Rx - \begin{bmatrix} z \\ \star \end{bmatrix} \right\|_2^2$$

This can be rewritten as:

$$\min_x \left(||Rx - z||_2^2 + || \star ||_2^2 \right)$$

To minimize this expression, we need to make $||Rx - z||_2^2$ as small as possible. This is achieved by setting $Rx - z = 0$, or $Rx = z$. Since R is an **upper triangular matrix**, the system $Rx = z$ is trivial to solve using **back substitution** (starting from the last equation and working backwards). This process is central to QRD RLS algorithms for computing the filter coefficients $w_{LS}[k] = R^{-1}[k]z[k]$.

2. **How is Chapter-9, slide 32 used in derivation of QRD LSL? See january 24 2020: Q2.1**

   Chapter-9, slide 32, introduces the **Givens rotation** as a basic tool for QR decomposition and updating. A Givens rotation is an orthogonal transformation that can selectively zero out an element in a vector. Specifically, for a vector $\mathbf{x}$, applying a Givens rotation $G_{i,j,\theta}$ modifies elements x_i and x_j to $x_i' = \cos\theta \cdot x_i + \sin\theta \cdot x_j$ and $x_j' = -\sin\theta \cdot x_i + \cos\theta \cdot x_j$, while leaving other elements unchanged. The rotation angle θ can be chosen such that x_j' becomes zero.

   In the context of QRD Recursive Least Squares (RLS) algorithms, particularly the **QRD Least Squares Lattice (QRD-LSL) algorithm** (Chapter 10, p.7), these Givens rotations are the fundamental building blocks. The QRD-LSL algorithm is a "fast" RLS algorithm that achieves $O(L)$ computational complexity per update, unlike the standard $O(L^2)$ RLS.

   The derivation of QRD-LSL relies on recursively applying sequences of Givens rotations in a structured way to propagate the triangular factor R and the vector z (from $Q^T d$) in time. Each "layer" in the lattice structure of QRD-LSL corresponds to a stage in the recursive updating, typically involving multiple Givens rotations. Chapter 10, slide 21, confirms this by stating that the QRD-LSL algorithm, which results from such derivations, involves "Six rotations per layer." These rotations are precisely the Givens rotations described on Chapter 9, slide 32. Therefore, Chapter 9, slide 32 provides the **mathematical operation** (Givens rotation) that is graphically represented as a "rotation cell" in the signal flow graph for QRD updating (Chapter 9, slide 35) and forms the core computational element within each layer of the QRD-LSL structure.

3. **Chapter-9, slide 33, the set of input signals [u[k] u[k-1] u[k-2] u[k-3] u[k-4]] is extended to [u[k] u[k-1] u[k-2] u[k-3] u[k-4] v[k]] with v[k] independent of u[k], is QRD LSL still possible? Sketch a block scheme, omit details.**

   Yes, if the input signals are extended to $\begin{bmatrix} u[k] & u[k-1] & u[k-2] & u[k-3] & u[k-4] & v[k] \end{bmatrix}$, with $v[k]$ independent of $u[k]$, the **QRD-LSL algorithm is still possible**. This scenario describes a **multi-channel adaptive filter**, where the input vector includes samples from more than one distinct source.

   Chapter 10, slides 26 to 29, specifically discusses the "Multi-channel QRD-LSL Algorithm." It states that the derivation of the single-channel QRD-LSL can be "straightforwardly generalized" to obtain the multi-channel version. This means that the core principles and recursive structure, based on Givens rotations, still apply, but the elements being processed would become vectors or matrices, and the internal operations within each lattice layer would become more complex, operating on multiple input streams simultaneously.

   Sketch of a conceptual block scheme (omitting detailed internal structure):

```
Input Channel 1 (e.g., u[k]) --------|
                                     |
Input Channel 2 (e.g., v[k]) --------|
                                       |
                                       |
                                  [Multi-channel]
                                  Lattice Stage 1
                                       |
                                       |
                                       |
                                  [Multi-channel]
                                  Lattice Stage 2
                                       |
                                       |
                                       |
                                      ...
                                       |
                                       |
                                       |
                                 [Multi-channel]
                                 Lattice Stage L
                                       |
                                       |
                                       |
                                   [Output]
```

In this conceptual scheme, each "Multi-channel Lattice Stage" would take multiple input signals (e.g., $u[k]$ and $v[k]$ and their delayed versions, as well as intermediate "forward" and "backward" prediction errors) and produce corresponding output signals, much like the 2-channel example shown in Chapter 10, slide 29. The lattice structure maintains its cascaded, modular nature, but the individual processing blocks (e.g., rotation cells) are adapted for vector/matrix operations.

1.3 Question 3

1. **For maximally decimated filter banks, provide an intuitive explanation for perfect reconstruction despite aliasing in each channel.**

   In a **maximally decimated filter bank**, the decimation factor D is equal to the number of channels N ($D = N$). A critical consequence of decimation is that it **introduces aliasing** in the individual subband signals. This means that the frequency content of the original signal gets folded back onto itself within each subband after downsampling.

   However, **perfect reconstruction (PR)** can still be achieved. This seemingly counter-intuitive result is possible because the **synthesis filter bank is specifically designed to cancel out these aliasing effects**. The distortion introduced by aliasing in the analysis stage is precisely compensated by the synthesis stage.

   Intuitively, imagine that the information lost or distorted by aliasing in one channel (subband) is present in another channel in a complementary way. The synthesis filters act as a "decoder" that intelligently combines these aliased subband signals. While each individual subband signal might be heavily aliased, the collective set of subband signals contains all the necessary information, and the synthesis filters know exactly how to linearly combine these aliased components such that the aliasing terms cancel out, leaving only the original signal (or a delayed version of it). The `DFT/IDFT filter bank` example is a simple case where this is shown to work even with non-ideal filters that produce significant aliasing. The overall system's transfer function becomes a pure delay, $Y(z) = z^{-\delta}U(z)$, despite the presence of aliasing terms in intermediate stages.

2. **Compare maximally decimated filter banks with oversampled filter banks. What are the advantages/disadvantages of oversampling?**

   Here's a comparison between maximally decimated filter banks (MDFB) and oversampled filter banks (OSFB):

   Maximally Decimated Filter Banks (MDFB):

   - **Definition:** The decimation factor D is equal to the number of channels N ($D = N$).
   - **Sampling Rate:** The total number of samples across all subbands is equal to the number of samples in the original fullband signal. This implies **maximal efficiency** in terms of data rate.
   - **Aliasing:** Aliasing is inherently introduced in the subband signals due to critical downsampling. While perfect reconstruction is possible, it relies heavily on the synthesis bank precisely canceling this aliasing.
   - **Design Complexity for PR:** Achieving perfect reconstruction (PR) can be challenging. The PR condition for $D = N$ involves the product of square polyphase matrices $R(z)E(z) = z^{-\delta}I_D$. If analysis filters $E(z)$ are FIR, synthesis filters $R(z)$ often turn out to be IIR (infinite impulse response), which can lead to stability concerns. Designing FIR PR-FBs in this case requires very specific "unimodular" or "paraunitary" matrix transfer functions for $E(z)$, which significantly **limits design flexibility**. For DFT-modulated FBs, this leaves "not much design freedom".

   Oversampled Filter Banks (OSFB):

   - **Definition:** The decimation factor D is less than the number of channels N ($D < N$).
   - **Sampling Rate:** The total number of samples across all subbands is greater than the number of samples in the original fullband signal. This results in **redundancy**.
   - **Aliasing:** Oversampling can allow for reduced aliasing in subbands, although aliasing might still be present depending on the design.
   - **Design Complexity for PR:** The PR condition $R(z)E(z) = z^{-\delta}I_D$ involves a product of a "short-fat" matrix $R(z)$ (D-by-N) and a "tall-thin" matrix $E(z)$ (N-by-D). This structural difference provides **additional design flexibility**. It is generally easier to find FIR synthesis filters $R(z)$ for FIR analysis filters $E(z)$ compared to the maximally decimated case.

   Advantages of Oversampling:

   - **Increased Design Flexibility:** This is the primary advantage, making it easier to meet both filter specifications (e.g., stopband attenuation, passband ripple) and the perfect reconstruction property.
   - **Robustness:** While not extensively detailed in the sources provided, the redundancy often makes oversampled systems more robust to quantization noise and errors introduced by subband processing (e.g., compression, noise reduction).
   - **Improved Performance in Applications:** For example, in hearing aids, oversampling with multiple microphones allows for advanced noise reduction techniques like multi-microphone beamforming.

   Disadvantages of Oversampling:

   - **Higher Computational Cost:** Due to the redundancy (processing more samples in total across subbands), OSFBs generally incur higher computational complexity compared to MDFBs.
   - **Increased Latency:** Processing multiple samples can lead to increased processing delay compared to time-domain sample-by-sample processing.

3. **Design a DFT modulated filter bank with 4 channels and 3 fold decimation. Give relevant formulas.**

   To design a DFT-modulated filter bank with 4 channels and 3-fold decimation, we have the following parameters:

- Number of channels: $N = 4$
- Decimation factor: $D = 3$

Since $D < N$ (3 < 4), this is an **oversampled DFT-modulated filter bank**.

The design procedure for DFT-modulated filter banks revolves around designing a single "prototype" analysis filter, $H_0(z)$, from which all other analysis filters $H_n(z)$ are derived by frequency shifting:

- **Analysis Filters:**
$$H_n(z) = H_0(z \cdot e^{-j2\pi n/N}), \quad \text{for } n = 0, \ldots, N-1$$
For $N = 4$, this means: $H_0(z) = H_0(z)$ $H_1(z) = H_0(z \cdot e^{-j2\pi/4})$ $H_2(z) = H_0(z \cdot e^{-j4\pi/4})$ $H_3(z) = H_0(z \cdot e^{-j6\pi/4})$
- **Polyphase Decomposition:** For efficient realization, the prototype filter $H_0(z)$ is decomposed into polyphase components. The appropriate number of polyphase components N' is given by $N' = \frac{N \cdot D}{\text{gcd}(N,D)}$. In this case, $N = 4$ and $D = 3$, so $\text{gcd}(4, 3) = 1$.
$$N' = \frac{4 \cdot 3}{1} = 12$$
So, $H_0(z)$ is expressed using its 12-fold polyphase components $E'_{n'}(z^{12})$:
$$H_0(z) = \sum_{n'=0}^{N'-1} z^{-n'} E'_{n'}(z^{N'}) = \sum_{n'=0}^{11} z^{-n'} E'_{n'}(z^{12})$$
- **Analysis and Synthesis Polyphase Matrices:** The analysis filter bank can be represented by a tall-thin matrix $E(z)$ (N-by-D) and the synthesis filter bank by a short-fat matrix $R(z)$ (D-by-N). These matrices are structured using the polyphase components of $H_0(z)$ and $F_0(z)$ respectively, in conjunction with DFT matrices. In general, for oversampled DFT-modulated FBs:
$$E(z) = F^{-1} \cdot B(z^{D'})$$
$$R(z) = C(z^{D'}) \cdot F$$
Where F is the $N \times N$ DFT matrix, and $B(z)$ (N-by-D) and $C(z)$ (D-by-N) are sparse matrices containing the polyphase components. For this specific case ($N = 4, D = 3$), $B(z)$ would be 4×3, and $C(z)$ would be 3×4. The construction of $B(z)$ and $C(z)$ from the $E'_{n'}(z^{12})$ components involves mapping based on the decimation and channel numbers, similar to Example 2 in Chapter 14.
- **Perfect Reconstruction (PR) Condition:** The PR condition for filter banks (assuming subband processing has 'output=input') is given by the product of the synthesis and analysis polyphase matrices:
$$R(z)E(z) = z^{-\delta} I_D$$
where δ is an integer delay and I_D is the $D \times D$ identity matrix. For DFT-modulated filter banks, this condition simplifies to:
$$C(z^{D'}) \cdot B(z^{D'}) = z^{-\delta} I_D$$
For oversampled filter banks ($D < N$), this condition is generally easier to satisfy while keeping $E(z)$ and $R(z)$ as FIR filters, providing more design freedom. The challenge is to design the prototype $H_0(z)$ (and thus its polyphase components $E'_{n'}(z^{N'})$) such that it meets the filter specifications (e.g., passband ripple, stopband attenuation), and then find corresponding synthesis filter components $R'_{n'}(z^{N'})$ that satisfy the PR condition.

1.4 Question 4 (about acoustic modem)

1. **What is the relevance of channel equalisation in the acoustic modem?**

   The acoustic modem transmits digital signals over an acoustic channel, which includes a loudspeaker, the physical room, and a microphone. This channel is modelled by a **linear discrete-time transfer function** $H(z)$, representing the acoustic impulse response.

   The relevance of channel equalisation in this context is paramount because real-world acoustic channels introduce significant distortions to the transmitted signal. These distortions include:

   - **Dead time, early reflections, and reverberation:** The acoustic impulse response contains multiple echoes and decays over time, causing transmitted symbols to "smear" into subsequent symbols.
   - **Inter-Symbol Interference (ISI):** This smearing effect, known as ISI, makes it difficult for the receiver to distinguish between consecutive transmitted data symbols, leading to errors in decoding the digital information.

   Channel equalisation is crucial to undo these distorting effects. In the Orthogonal Frequency Division Multiplexing (OFDM) scheme used by the acoustic modem, equalisation is ingeniously simplified:

   - The channel's linear convolution in the time domain is transformed into **scalar multiplication in the frequency domain for each subcarrier**. This is achieved by adding a "cyclic prefix" to each transmitted OFDM symbol.
   - This transformation allows for a very efficient equalisation technique called **"1-tap FEQ" (Frequency-domain Equalisation)**. After performing a Fast Fourier Transform (FFT) at the receiver, the received signal on each subcarrier $Y_{\tilde{k}}(n)$ is simply the product of the transmitted symbol $X_{\tilde{k}}(n)$ and the channel's frequency response H_n at that specific carrier:
     $$Y_{\tilde{k}}(n) = H_n \cdot X_{\tilde{k}}(n)$$
   - Equalisation then involves a simple **component-wise division** by H_n for each carrier in the frequency domain to estimate the original transmitted symbol:
     $$\text{estimate}\{X_{\tilde{k}}(n)\} = (H_n)^{-1} \cdot Y_{\tilde{k}}(n)$$

     Without effective channel equalisation, the severe distortions introduced by the acoustic environment would make reliable digital communication impossible.

2. **What is the relevance of the prefix in OFDM? What are the requirements for this prefix?**

   The "cyclic prefix" (CP) is a **fundamental component of OFDM** that addresses the crucial issue of channel distortion.

   - **Relevance:** The primary relevance of the CP is to **transform linear convolution (caused by the channel) into circular convolution**. This is essential for the OFDM principle to work, as it allows the channel's effect to be represented as simple scalar multiplications in the frequency domain for each subcarrier. Without the CP, inter-symbol interference (ISI) from previous symbols and inter-carrier interference (ICI) between subcarriers would corrupt the received signal, complicating or preventing frequency-domain equalisation.
   - **Mechanism:** At the transmitter, the CP is formed by copying the last L samples of an OFDM symbol (of length N) and appending them to the *beginning* of that symbol, resulting in an $(N + L)$-sample segment for transmission. At the receiver, these L samples corresponding to the CP are then discarded before the FFT. This ensures that when the channel performs a linear convolution on the extended symbol, the central N samples of the received signal correspond to a circular convolution of the original N-sample symbol with the channel's impulse response.

Requirements for this prefix:

- The length of the cyclic prefix, L, must be **at least as long as the order (or effective length) of the FIR channel model**. If the channel impulse response extends beyond the CP, residual ISI will occur.
- It ensures that the matrix representing the channel's effect on the received segment (after CP removal) is a **circulant matrix**. This is a critical property because circulant matrices are diagonalized by the Discrete Fourier Transform (DFT) matrix, which is precisely why OFDM can use FFT/IFFT for modulation and demodulation and achieve simple frequency-domain equalisation.

3. **How does one derive a channel model for the acoustic modem? What if one wants to equalise without modelling the channel?**

Deriving a Channel Model for the Acoustic Modem: The acoustic modem project involves characterising the composite channel, from the digital transmit signal (Tx) through the loudspeaker, acoustic environment, microphone, and back to the digital receive signal (Rx). This entire transmission channel is modelled as a **discrete-time FIR (Finite Impulse Response) transfer function**, $H(z) = h_0 + h_1 z^{-1} + \cdots + h_L z^{-L}$.

The process to derive this channel model is a **parameter estimation experiment**:

1. **Transmit a Known Signal (x_k):** A pre-defined "favourite signal" x_k is sent from the transmitter through the acoustic channel.
2. **Record the Corresponding Output Signal (y_k):** The signal received at the microphone and converted back to digital form is recorded as y_k.
3. **Least Squares (LS) Estimation:** The relationship between the transmitted signal x_k and the received signal y_k can be expressed as a convolution:

$$\begin{bmatrix} y_k \\ y_{k+1} \\ \vdots \\ y_{k+K} \end{bmatrix} = \begin{bmatrix} x_k & x_{k-1} & \cdots & x_{k-L} \\ x_{k+1} & x_k & \cdots & x_{k+1-L} \\ \vdots & \vdots & \ddots & \vdots \\ x_{k+K} & x_{k+K-1} & \cdots & x_{k+K-L} \end{bmatrix} \cdot \begin{bmatrix} h_0 \\ h_1 \\ \vdots \\ h_L \end{bmatrix}$$

This equation is an overdetermined system $Y = X \cdot H_{coeffs}$. The channel coefficients $h_0, \ldots, h_L$ are then estimated by minimizing the squared error between the observed y_k and the predicted output. This is a linear least squares problem solved by finding $\min_{h_0,\ldots,h_L} ||\mathbf{Y} - \mathbf{X} \cdot \mathbf{H}_{coeffs}||_2^2$. The solution can be obtained using standard numerical methods (e.g., in Matlab, this is a single line of code).

What if one wants to equalise without modelling the channel? The provided sources primarily describe **channel modelling as a prerequisite for equalisation** in the OFDM acoustic modem. The 1-tap FEQ approach explicitly relies on having the channel's frequency response H_n for each subcarrier, which is derived from the estimated channel model $H(z)$.

However, the concept of **adaptive equalisation** is introduced for later weeks of the acoustic modem project (Week 7-8). This suggests a scenario where an explicit, fixed "model" of the channel may not be needed upfront, but rather the equaliser continuously *learns* and *adapts* its parameters to the channel's characteristics. The source notes that this involves "decision-directed adaptive equalization" (see Part III of the course). While the source does not detail how this is done "without modelling the channel", it refers to "channel estimation and/or equalizer (FEQ) initialization/updating based on transmitted training symbols" for Weeks 5-6.

This implies that the system does not need a pre-computed model. Instead, it uses:

- **Training Sequences:** Known data sequences are transmitted, allowing the receiver to estimate the channel's properties (or the required equalization parameters directly).
- **Adaptive Algorithms (e.g., LMS, RLS):** These algorithms (discussed in Chapters 8 and 9) continuously adjust the equaliser's coefficients based on the error between the received signal and the desired (or reconstructed) signal, effectively adapting to the channel without requiring a precise, static model. The goal is to track time-varying channels. While these algorithms inherently estimate channel characteristics (or their inverse), they do not necessarily require a full, explicit model $H(z)$ to be computed and maintained, but rather adapt the equalization parameters directly.

2 13 augustus 2018

2.1 Question 1

1. **What are noise transfer functions? Explain how such noise transfer functions are used in the analysis of quantization effects in filter implementation.** In the context of analysing quantization effects in filter implementation, a **noise transfer function** refers to the transfer function from a **noise source**, representing the quantization error, to the **filter output** .

   These noise transfer functions are primarily used in the **statistical analysis** of quantization effects. This analysis assumes that the filter itself is a linear system . To model quantization, a **noise source is conceptually inserted after each multiplier or adder** within the filter's block scheme or flow graph . Each of these noise sources represents the quantization error introduced at that specific arithmetic operation .

   Since the filter is treated as linear, the output noise generated by each individual noise source is assumed to simply **add to the desired output signal** . The effect of each noise source on the overall filter output is then computed by determining its respective noise transfer function `G(z)` . This `G(z)` allows for the calculation of the **mean and variance** of the noise at the filter's output, based on the statistical properties (mean and variance) of the individual quantization error `e[k]` at its source . This procedure is repeated for all noise sources, and their effects are combined to understand the overall quantization noise at the filter's output .

   For certain filter realizations, this analysis can be simplified:

   - In a **transposed direct form** realization, all noise transfer functions (up to a delay) are equal, meaning all individual noise sources can be **lumped into one equivalent noise source** . This significantly simplifies the analysis .
   - Similarly, in a **direct form** realization, all noise sources can be **lumped into two equivalent noise sources** .

2. **Do the properties of the quantization mechanism play a role in such analysis? Yes, the properties of the quantization mechanism play a crucial role** in such statistical analysis . Different quantization mechanisms exhibit distinct statistical characteristics that directly influence the assumed properties of the noise sources in the model .

   The main quantization mechanisms discussed are:

   - **Rounding**: The quantization error has a **mean of 0** and a variance of $\frac{1}{12}\text{LSB}^2$.
   - **Truncation**: The quantization error is **biased**, with a mean of $(-0.5)\text{LSB}$ and a variance of $\frac{1}{12}\text{LSB}^2$.
   - **Magnitude Truncation**: The quantization error has a **mean of 0** and a variance of $\frac{1}{6}\text{LSB}^2$.

These different mean and variance values are fundamental inputs to the statistical analysis, determining the expected level and characteristics of the noise propagated through the filter system . For instance, a biased quantization mechanism (like truncation) would introduce a DC offset in the output noise, which needs to be accounted for in the analysis ($\mu_e \cdot G(z)|_{z=1}$ for the output mean) .

3. **The analysis relies on linearity (explain) whereas quantization is a non-linear operation. Could this be a contradiction?** The statistical analysis of quantization effects simplifies the problem by assuming the underlying filter is a **linear system** , allowing for the superposition of noise effects. However, **quantization itself is inherently a non-linear and deterministic operation** .

   This difference indeed presents a **potential contradiction** . While the statistical model (which assumes the quantization error is random, uncorrelated, and uniformly distributed) is "simple/convenient," the sources explicitly acknowledge that "quantization is truly a non-linear effect, and should be analyzed as a deterministic process" .

   The limitations of the linear statistical analysis become apparent when considering **non-linear phenomena** like **limit cycle oscillations** . These are persistent oscillations at the filter's output even in the absence of any input signal (i.e., `u[k]=0`) . Such oscillations are a direct result of the non-linear nature of quantization . Examples demonstrate that the type of quantization (rounding vs. truncation) and even the initial state can lead to different or absent limit cycle behaviors . These unwanted oscillations can be particularly problematic in applications like speech or audio, where they might be audible .

   It's important to note that limit cycle oscillations **can only occur in filters with feedback** (i.e., **IIR filters**), because FIR filters do not have feedback and therefore cannot sustain such oscillations .

   Despite the general difficulty of deterministic analysis, good news exists: for **lossless lattice or lattice-ladder realizations** of IIR filters, if **magnitude truncation** is used, the implementation is **guaranteed to be free of limit cycles** . This is intuitively explained by the fact that magnitude truncation consumes energy, while the orthogonal operations in these lattice structures do not generate power to sustain limit cycles .

2.2 Question 2

1. **In a maximally decimated (MD) filter bank aliasing usually occurs because of the down-sampling operation. Provide an intuitive explanation why perfect reconstruction (PR) is still possible, in spite of the downsampling and aliasing.** In a maximally decimated (MD) filter bank, the input signal is first split into several frequency channels by an analysis filter bank, and then each subband signal is **downsampled by a factor `D=N`** (where N is the number of channels), which typically equals the decimation factor . This downsampling operation inevitably introduces **aliasing** in the individual subband signals, as spectral replicas overlap due to the reduced sampling rate .

   Despite this aliasing, **perfect reconstruction (PR)** is still possible. The **intuitive explanation** lies in the **design of the synthesis filter bank** . While aliasing is introduced in the analysis (downsampling) stage, the synthesis filters are **specifically designed to cancel out these aliasing effects** during the reconstruction phase . This means that the aliased components, although present in the individual subbands, cancel each other out precisely when they are combined by the synthesis filters and upsampling operations .

   Even when using "non-ideal filters" (filters that are not perfect brick-wall filters and thus have overlapping frequency responses) that introduce "significant aliasing," PR can be achieved . The key is that the **synthesis filters are matched to the analysis filters** in such a way that the distortion and aliasing terms in the overall input-output

relationship cancel out, leaving only a pure delay . For example, in a simple DFT/IDFT filter bank, the inverse DFT (IDFT) matrix is used to perfectly reconstruct the signal, despite the aliasing present in the intermediate subbands .

2. **What are DFT modulated filter banks? What are the advantages of such FB's? DFT-modulated filter banks** are a type of **uniform filter bank** . In a uniform filter bank, all the analysis filters $H_n(z)$ are **frequency-shifted versions of a single prototype filter $H_0(z)$** . That is, $H_n(z) = H_0(z \cdot e^{-j2\pi n/N})$. This means that only the prototype filter $H_0(z)$ needs to be designed, and the other filters are automatically defined .

   These filter banks are called "DFT-modulated" because they can be **efficiently realized using polyphase decompositions and the Discrete Fourier Transform (DFT) or Inverse DFT (IDFT) algorithms** (often implemented via Fast Fourier Transform (FFT) for speed) . The analysis bank, for instance, can be implemented by filtering with the polyphase components of $H_0(z)$ followed by an inverse DFT .

   The main **advantages** of DFT-modulated filter banks are their **economy in design and implementation complexity** :

   - **Reduced Implementation Complexity**: Instead of implementing N separate filters, the system only requires the implementation of the polyphase components of the single prototype filter, followed by an efficient IFFT operation. This means N filters can be obtained "for the price of 1, plus inverse DFT (=FFT)!" .
   - **Reduced Design Complexity**: The engineer only needs to design the single prototype filter $H_0(z)$ to meet the desired specifications (e.g., passband ripple, stopband suppression) . The characteristics of the other $H_n(z)$ filters (e.g., their ripple, transition bands) are automatically "co-designed" due to their uniform relationship to the prototype .

3. **How can DFT-modulated PR filter banks be designed?** The design of DFT-modulated Perfect Reconstruction (PR) filter banks depends significantly on whether they are maximally decimated or oversampled:

   - **Maximally Decimated (D=N) DFT-Modulated PR-FBs**: The general PR condition for maximally decimated filter banks is given by the product of the polyphase matrices of the synthesis and analysis banks, $R(z)E(z) = z^{-\delta}I$. For DFT-modulated filter banks, the analysis polyphase matrix $E(z)$ has a specific structure involving the DFT matrix and the polyphase components of the prototype filter . However, designing such systems to be both FIR and achieve PR imposes **significant limitations, resulting in very little design freedom** . If the polyphase components $E_n(z)$ are constant or simple all-pass functions (e.g., $E_n(z) = w_n$), this leads to "windowed" IDFT/DFT filter banks, also known as the "short-time Fourier transform" . If $E(z)$ is forced to be paraunitary (all-pass and FIR), it results in only trivial modifications of the basic IDFT/DFT filter bank . This means that while maximally decimated DFT-modulated filter banks offer implementation and design economy, they offer **limited flexibility for designing filters with specific frequency response characteristics while maintaining PR** .
   - **Oversampled (D<N) DFT-Modulated PR-FBs**: **Oversampling provides additional design flexibility** for DFT-modulated PR filter banks . The PR condition remains the same: $R(z)E(z) = z^{-\delta}I_D$ (where I_D is the identity matrix of size D), but now $E(z)$ is a tall-thin $N \times D$ matrix, and $R(z)$ is a short-fat $D \times N$ matrix . The **design procedure** generally involves:

     1. **Designing the prototype analysis filter $H_0(z)$** (e.g., as an FIR filter) . This automatically determines the polyphase components $E_n(z)$ (which form the matrix $B(z)$ within the polyphase representation of $E(z)$) .
     2. **Computing the synthesis filter bank $R(z)$**: Given $E(z)$, the corresponding $R(z)$ can often be found by solving the PR condition $R(z)E(z) = z^{-\delta}I_D$. If $D < N$, this typically leads to an **underdetermined set**

of linear equations for the coefficients of R(z) . This implies that there are generally **infinitely many solutions** for R(z) .

3. **Selecting a specific solution**: An "easy design procedure" involves choosing the **minimum-norm solution** for R(z) . This minimum-norm solution is derived from a least-squares problem and provides desirable properties for the synthesis filter bank .

In summary, oversampling introduces sufficient degrees of freedom to enable the design of FIR DFT-modulated PR filter banks with good filter characteristics, providing a flexible and relatively easy design procedure .

2.3 Question 3

1. **Explain the main trick that is used in the derivation of the QRD-LSL algorithm.** The QRD-LSL (QR Decomposition Least Squares Lattice) algorithm is a "fast" RLS (Recursive Least Squares) algorithm, noteworthy because it achieves a **computational complexity of O(L) per update**, in contrast to the O(L^2) complexity of standard RLS algorithms .

   The main trick in its derivation lies in the **recursive application of QR decomposition (QRD) using Givens rotations** . The core idea is to express the least squares (LS) problem $\min_w \|d - Uw\|_2^2$ in a numerically superior way by performing a QR decomposition of the data matrix U as $U = Q \cdot R$. The LS solution can then be found by solving the triangular system $R \cdot w_{LS} = z$, where z is derived from $Q^T \cdot d$. This method is numerically more stable than directly inverting $U^T U$.

   The "trick" for the *recursive* derivation of QRD-LSL then proceeds as follows:

   - It operates on the **"information matrix"** (or square-root of it) and the corresponding right-hand side vector .
   - Instead of directly updating $U^T U$ (which is a full matrix, leading to O(L^2) operations), the QRD-LSL algorithm recursively updates the **upper triangular matrix** $R(k)$ **and vector** $z(k)$ at each time step k .
   - This updating is achieved by applying a sequence of **Givens rotations** . A Givens rotation is an orthogonal transformation that can selectively zero-out elements in a vector .
   - By strategically applying these rotations, the algorithm effectively "triangularizes" the incoming new data sample (u_k^T, d_k) with the existing $R(k-1)$ and $z(k-1)$ values, producing the updated $R(k)$ and $z(k)$ and leaving a single residual value (ϵ) . This process, graphically represented as layers where the "prediction order is increased when going from one layer to the next lower layer" , ensures that the computational complexity scales linearly with the filter length L . The \epsilon value (the last element after the Givens rotations) is directly related to the a posteriori and a priori residuals .

2. **Compare the complexity of the QRD-LSL algorithm with the complexity of LMS.** Both the QRD-LSL algorithm and the LMS (Least Mean Squares) algorithm offer a **computational complexity of O(L)** per time update, where L is the filter length or order . However, there are significant differences in their constant factors and performance:

   - **LMS (Least Mean Squares) Algorithm**:
     - **Complexity**: Approximately **2L multiplications** per time update . This makes it a very **computationally cheap** algorithm .
     - **Convergence**: LMS generally exhibits **slow convergence** . Its convergence speed is affected by the eigenvalue spread of the input signal's autocorrelation matrix ($\lambda_{max}/\lambda_{min}$) . For "colored" input signals (where $\lambda_{min} << \lambda_{max}$), convergence can be particularly slow .

- **QRD-LSL (QR Decomposition Least Squares Lattice) Algorithm**:
  - **Complexity**: Approximately **24(L+1) multiplications** per time update . This means it is **significantly more computationally intensive** than LMS, despite both being O(L) .
  - **Convergence**: QRD-LSL, being a type of Recursive Least Squares (RLS) algorithm, does **not compromise convergence properties** . RLS algorithms generally offer **much faster convergence** and are less sensitive to the coloring of the input signal compared to LMS, making them suitable for time-varying environments .
  - **Numerical Properties**: QRD-based algorithms like QRD-LSL have **better numerical properties** and are more robust in finite-precision implementations compared to standard RLS algorithms .

In summary, while both are linear in complexity, **LMS is much cheaper per update but slower to converge**, whereas **QRD-LSL is more expensive per update but offers much faster and more robust convergence**.

3. **In an acoustic echo cancellation scenario with one loudspeaker and two microphones (ie. where an echo contribution is to be cancelled in each of the two microphone signals), is it possible to apply a QRD-LSL-algorithm? Sketch an efficient realisation (clearly defining input and output signals). Yes, it is possible to apply a QRD-LSL algorithm** in an acoustic echo cancellation scenario with one loudspeaker and two microphones. The sources explicitly state that the **Multi-channel QRD-LSL** algorithm is a straightforward generalization of the QRD-LSL algorithm, providing an example for a 2-channel system with equal FIR filter orders .

   Efficient Realization Sketch (Conceptual Description): For this scenario, we have:

   - **Input signal `u[k]`**: This is the "far-end signal" originating from the loudspeaker, which is the common input to the echo paths for both microphones (e.g., in `DSP-CIS Chapter-7: Optimal Filters - Wiener Filters` for acoustic echo cancellation and multi-channel FIR filters).
   - **Desired signals `d_1[k]` and `d_2[k]`**: These are the signals recorded by the two microphones. Each `d_i[k]` contains the near-end speech signal plus the echo from the loudspeaker (`u[k]`) .
   - **Output signals `e_1[k]` and `e_2[k]`**: These are the residual echo signals after the adaptive filter has estimated and cancelled the echo in each microphone channel . These error signals are crucial for steering the adaptation .

   The **Multi-channel QRD-LSL algorithm** (`DSP-CIS Chapter-10: Fast RLS Algorithms`, p.29) would adapt two sets of filter coefficients, say $w_1[k]$ and $w_2[k]$, to model the two acoustic echo paths from the loudspeaker to each microphone.

   A conceptual sketch for this efficient realization would involve a block diagram similar to that shown in (which points to `p.28` and `p.29` of `chapter10-Fast RLS Algorithms.pdf` for multi-channel QRD-LSL):

   1. **Input Side**: The far-end signal `u[k]` (from the loudspeaker) serves as the primary input to the multi-channel adaptive filter.
   2. **Adaptive Filter Core (Multi-channel QRD-LSL)**: This core processes the input `u[k]` to generate two estimated echo signals, $y_1[k]$ and $y_2[k]$, corresponding to the echo path for each microphone. The multi-channel QRD-LSL structure, as generalized from the single-channel lattice structure , would contain parallel processing paths or a combined matrix operation that allows simultaneous adaptation for both channels. This involves layers of orthogonal transformations (Givens rotations) that update reflection coefficients (or similar internal parameters) for each channel .
   3. **Error Calculation**: The estimated echo signals $y_1[k]$ and $y_2[k]$ are subtracted from their respective microphone signals $d_1[k]$ and $d_2[k]$ to produce the error signals:

- $e_1[k] = d_1[k] - y_1[k]$
- $e_2[k] = d_2[k] - y_2[k]$ These errors are then fed back to the adaptation process within the QRD-LSL algorithm to continuously update the filter coefficients and minimize the residual echo .

The efficient realization leverages the **recursive and numerically stable properties** of QRD-LSL, making it suitable for real-time acoustic echo cancellation even with multiple channels . While the source mentions specific pages (p.28, p.29) showing the setup and the detailed graphical realization, conceptually, it extends the single-channel lattice structure to handle parallel outputs.

2.4 Question 4: Acoustic modem project

1. **Explain how the OFDM modulation format (including a cyclic prefix) provides an easy equalization of the transmission channel. What are conditions that the transmission channel has to satisfy for this 'cyclic prefix trick' to work properly?** The OFDM (Orthogonal Frequency Division Multiplexing) modulation format, particularly when combined with a cyclic prefix, dramatically simplifies channel equalization.

   How OFDM with Cyclic Prefix Provides Easy Equalization:

   1. **OFDM Basics**: In OFDM, a high-rate bit stream is divided into multiple low-rate bit streams, each modulating a different orthogonal carrier frequency . At the transmitter, these modulated carriers are combined, typically through an **Inverse Fast Fourier Transform (IFFT)**, to create time-domain signal segments .
   2. **Cyclic Prefix (CP) Insertion**: Before transmission, a **cyclic prefix (CP)** is appended to each OFDM symbol (time-domain segment) . This prefix consists of a copy of the **last L samples** of the symbol, prepended to the beginning of the symbol, where L is typically the estimated order of the channel's impulse response . Thus, instead of transmitting N samples, N+L samples are sent .
   3. **Circular Convolution**: At the receiver, the **CP is removed** . Crucially, if the channel's impulse response length does not exceed the CP length, the linear convolution of the transmitted signal with the channel's impulse response is effectively converted into a **circular convolution** .
   4. **Simplified Frequency-Domain Operation**: A fundamental theorem states that **circular convolution in the time domain is equivalent to point-wise multiplication in the frequency domain** . Therefore, after removing the CP and performing an **FFT** at the receiver, the received frequency-domain symbols $\hat{Y}_k(n)$ (for the n-th carrier of the k-th symbol) are simply the transmitted frequency-domain symbols $\hat{X}_k(n)$ multiplied by a **scalar complex gain Hn** : $\hat{Y}_k(n) = H_n \cdot \hat{X}_k(n)$. Here, Hn is the channel's frequency response evaluated at the n-th carrier frequency .
   5. **1-Tap Frequency Domain Equalization (FEQ)**: This simple relationship allows for straightforward channel equalization in the frequency domain. The receiver can recover an estimate of the transmitted symbol $\hat{X}_k(n)$ by performing a simple **point-wise division**: estimate$\{\hat{X}_k(n)\} = (H_n)^{-1} \cdot \hat{Y}_k(n)$. This is referred to as "1-tap FEQ" because each frequency bin (carrier) requires only a single complex multiplication (by the inverse of Hn), significantly simplifying the equalization process compared to complex multi-tap time-domain equalizers .

   Conditions for the 'Cyclic Prefix Trick' to Work Properly: For the cyclic prefix trick to work effectively, the transmission channel must satisfy specific conditions:

   - The channel's impulse response must be **adequately modelled as an FIR filter** .

- The **length of the cyclic prefix (L)** must be **greater than or equal to the length (order) of the channel's impulse response** . If the channel's impulse response extends beyond the length of the cyclic prefix, it will lead to **inter-symbol interference (ISI)** (overlap between consecutive OFDM symbols) and **inter-carrier interference (ICI)** (loss of orthogonality between carriers), degrading system performance.

2. **If a channel model is required for the equalization, explain how it can be derived. Is it possible to run the equalization without first deriving a channel model?**

   How to Derive a Channel Model (if required for equalization): Yes, a channel model is typically required for equalization . For the acoustic channel, it can be modeled as a **discrete-time FIR transfer function `H(z)`** . The derivation of this model is usually achieved through a **least squares (LS) parameter estimation problem** . The process involves:

   1. **Transmission of a Known Training Sequence**: A pre-defined signal, often a "training sequence" (`xk`), is transmitted through the acoustic channel .
   2. **Recording of the Corresponding Received Signal**: The output signal (`yk`) from the microphone, corresponding to the transmitted training sequence, is recorded .
   3. **Formulating a Least Squares Problem**: Given that the received signal `yk` is a convolution of the transmitted signal `xk` with the unknown channel impulse response `h` (coefficients `h0, h1, ..., hL`), a system of linear equations can be formed . The goal is to find the channel coefficients `h` that minimize the squared difference between the observed `yk` and the `yk` predicted by the model . This minimization is expressed as:
      $$\min_{h_0,\dots,h_L} \left\| \begin{pmatrix} y_k \\ y_{k+1} \\ \vdots \\ y_{k+K} \end{pmatrix} - \begin{pmatrix} x_k & x_{k-1} & \dots & x_{k-L} \\ x_{k+1} & x_k & \dots & x_{k+1-L} \\ \vdots & \vdots & \ddots & \vdots \\ x_{k+K} & x_{k+K-1} & \dots & x_{k+K-L} \end{pmatrix} \cdot \begin{pmatrix} h_0 \\ h_1 \\ \vdots \\ h_L \end{pmatrix} \right\|_2^2$$
   4. **Solving the LS Problem**: This is a standard linear least squares problem, which can be solved directly (e.g., in Matlab with a single line of code, like `h = U \ y`) . The solution provides the estimated FIR channel coefficients `h0, ..., hL`, effectively yielding the channel model `H(z)` . Once `h[k]` is known, the frequency response `Hn` for each carrier can be computed by taking its DFT .

   Is it possible to run the equalization without first deriving a channel model?

AI bullshit, it is possible through adaptive filtering chap.7 page 14

~~**No, it is not possible to run the frequency-domain equalization (1-tap FEQ) *without* knowledge of the channel's frequency response `Hn` for each carrier** . The equalization step fundamentally requires dividing by `Hn` to undo the channel's attenuation and phase shift ..~~

~~.While a full, explicit "channel model" (i.e., the impulse response `h[k]`) might not always be derived and stored as a separate step for every single sample, the **`Hn` values (the channel's frequency response at each carrier)** must be **continuously estimated or tracked**. In practical OFDM systems, especially those operating in time-varying channels (like mobile communications where channel can change 100s of times per second), this is often achieved through:. .* **Periodic Transmission of Training Symbols**: Known training symbols are embedded within the data stream. The receiver uses these known symbols to estimate `Hn` for each carrier by comparing the received training symbol to the known transmitted one .. .* **Adaptive Equalization / Decision-Directed Mode**: Once an initial estimate of the channel is obtained (e.g., in a "training mode"), the equalizer can switch to a "decision-directed" mode . In this mode, the receiver's decoded symbols (which are typically very reliable) are treated as if they were known training symbols, and used to continuously refine and update the channel estimates (`Hn` values) . This implies that the system is constantly "deriving" or refining its knowledge~~

~~of the channel, even without a distinct, explicit "channel modeling" phase for every data block. Therefore, continuous channel knowledge (in the form of Hn values) is essential for effective equalization.~~

3 16 augustus 2017

3.1 Question 1

1. **What are noise transfer functions? Explain how such noise transfer functions are used in the analysis of quantization effects in filter implementation.**

   In the context of filter implementation and quantization effects, a **noise transfer function** $G(z)$ is defined as the transfer function from a noise source (representing quantization error) to the filter's output.

   The analysis of quantization effects typically assumes that quantization introduces random errors. These errors are modelled as **noise sources** that are inserted after each arithmetic operation (multiplier or adder) within the filter's realization. The core idea is that, because the filter is a linear system, the output noise generated by each of these individual noise sources simply adds to the main output signal.

   To determine the impact of these noise sources on the filter's output, the following steps are generally taken:

   - Each quantization error $e[k]$ is assumed to be a **white noise** source with a specific mean (μ_e) and variance (σ_e^2).
   - For each noise source, the **noise transfer function** $G(z)$ (or its impulse response $g[k]$) from the point of noise injection to the filter output is determined.
   - The **mean of the output noise** is calculated as $\mu_e \cdot G(z)|_{z=1}$ (i.e., the noise mean multiplied by the DC-gain of the noise transfer function).
   - The **variance of the output noise** is computed as $\sigma_e^2 \cdot \sum_{k=0}^{\infty} g^2[k] = \sigma_e^2 \cdot \frac{1}{2\pi} \int_{-\pi}^{+\pi} |G(e^{j\omega})|^2 d\omega$. This procedure is then repeated for every noise source in the filter, and their effects are combined to give the total output noise. For instance, in a transposed direct form realization, all noise transfer functions are effectively the same (up to a delay), simplifying the analysis by lumping all noise sources into one equivalent source.

2. **Do the properties of the quantization mechanism play a role in such analysis?**

   Yes, the properties of the quantization mechanism play a crucial role in this analysis. Different quantization mechanisms, such as **rounding**, **truncation**, and **magnitude truncation**, produce quantization errors with distinct statistical properties.

   Specifically, these mechanisms result in different values for:

   - **Mean of the quantization error**: Rounding typically leads to a zero-mean error, whereas truncation can result in a biased error (e.g., mean of $(-0.5)LSB$).
   - **Variance of the quantization error**: While rounding and standard truncation may have the same variance ($\frac{1}{12}LSB^2$), magnitude truncation has a different variance ($\frac{1}{6}LSB^2$).

   These specific mean and variance values for each quantization mechanism directly feed into the formulas for calculating the mean and variance of the output noise. Therefore, the choice of quantization mechanism directly impacts the predicted noise performance of the filter implementation. Furthermore, the type of quantization (e.g., magnitude truncation) can also affect the **deterministic non-linear behavior**, such as the guaranteed absence of limit cycle oscillations in certain filter realizations like lossless lattice or lattice-ladder forms.

3. **The analysis relies on linearity (explain) whereas quantization is a non-linear operation. Could this be a contradiction?**

   Yes, this can be considered a contradiction, or more accurately, a **simplification with limitations**.

   The statistical analysis of quantization effects relies on the assumption that the quantization error can be modelled as a **random, uncorrelated noise source**. This allows for the linear superposition of the effects of multiple noise sources on the filter's output, essentially treating the filter as a **linear time-invariant (LTI) system** through which these noise sources propagate. This linearity assumption simplifies the mathematical analysis considerably.

   However, **quantization itself is inherently a deterministic non-linear operation**. This means that the statistical model, while convenient and often useful for predicting average noise performance, **does not fully capture all the behaviors** that can arise from quantization. For instance, the non-linear nature of quantization can lead to phenomena such as **limit cycle oscillations** in the absence of input signals (zero-input limit cycles), which are not predicted by the linear statistical model. These oscillations are a direct consequence of the non-linear feedback within the quantized system.

   Therefore, while the linearity assumption allows for a tractable analysis of noise propagation, it is important to acknowledge that it is an approximation. It is generally effective for understanding the average-case performance but can miss critical non-linear behaviors that require a more complex, deterministic analysis.

3.2 Question 2

1. **What are DFT modulated filter banks? What are the advantages of such FB's?**

   DFT-modulated filter banks (FBs) are a type of **uniform filter bank** where all analysis filters ($H_n(z)$) are generated by uniformly shifting the frequency response of a single "prototype" filter ($H_0(z)$) over the unit circle. This means $H_n(z) = H_0(z \cdot e^{-j2\pi n/N})$ for $n = 0, \ldots, N-1$, where N is the number of channels. The time-domain equivalent is $h_n[k] = h_0[k] \cdot e^{j2\pi k \cdot n/N}$.

   The **advantages** of DFT-modulated filter banks are primarily related to their **design and implementation efficiency**:

   - **Reduced Design Complexity**: Instead of designing each of the N analysis filters independently, only the single prototype filter $H_0(z)$ needs to be designed. The other $N-1$ filters are then automatically "co-designed," inheriting the same filter specifications (e.g., passband ripple, stopband suppression) from the prototype. This significantly simplifies the design process compared to designing each filter individually.
   - **Reduced Implementation Complexity**: Uniform filter banks can be realized very efficiently using polyphase decompositions combined with Discrete Fourier Transform (DFT) or Fast Fourier Transform (FFT) operations. This means that, for FIR filters, N filters can be implemented "for the price of 1," plus an Inverse DFT (IFFT) operation. This leads to substantial computational savings, especially for large N.

2. **How can maximally decimated DFT filter banks be designed?**

   Maximally decimated DFT filter banks are those where the decimation factor D is equal to the number of channels N ($D = N$). For such filter banks to achieve Perfect Reconstruction (PR), meaning the output signal is equal to the input signal (possibly with a delay) when subband processing is identity, a specific condition involving their polyphase matrices must be met.

   The PR condition for maximally decimated filter banks (based on their polyphase representation) is $R(z) \cdot E(z) = z^{-\delta} I_N$, where $E(z)$ is the $N \times N$ analysis polyphase matrix and $R(z)$ is the $N \times N$ synthesis polyphase matrix, and I_N is the identity matrix.

The design procedure for maximally decimated DFT-modulated FBs is as follows:

1. **Design the Prototype Analysis Filter** $H_0(z)$: This filter should ideally meet the desired frequency domain specifications (e.g., passband ripple, stopband attenuation).
2. **Determine Polyphase Components** $E_n(z)$: The prototype filter $H_0(z)$ has N-fold polyphase components, which define the analysis polyphase matrix $E(z)$.
3. **Choose Synthesis Filters**: To satisfy the PR condition, the polyphase components of the synthesis bank prototype filter, $R_n(z)$, are obtained by inverting the polyphase components of the analysis bank prototype filter. Specifically, $R_n(z) = z^{-\delta} \cdot E_{N-1-n}^{-1}(z)$, where δ is an integer delay to ensure causality.

However, a significant challenge arises here: If the prototype analysis filter $H_0(z)$ is a Finite Impulse Response (FIR) filter (which is generally desired), its polyphase components $E_n(z)$ will also be FIR. However, inverting these FIR polyphase components $E_n(z)$ to get $R_n(z)$ will generally result in **Infinite Impulse Response (IIR) synthesis filters**, which can pose stability concerns. Perfect Reconstruction with FIR analysis and synthesis filters in maximally decimated DFT-modulated filter banks is only achieved in very restrictive cases, specifically when each $E_n(z)$ is "unimodular" (i.e., $E_n(z) = \text{constant} \cdot z^{-d}$) or "paraunitary" (i.e., all-pass filters where $|E_n(e^{j\omega})| = 1$). This inherent restriction means there is **not much design freedom** for maximally decimated DFT-modulated FBs to achieve PR with desired filter characteristics.

3. **How can oversampled DFT filter banks be designed?**

Oversampled DFT filter banks are characterized by a decimation factor D that is less than the number of channels N ($D < N$). This "oversampling" provides **additional design flexibility** compared to maximally decimated filter banks.

The Perfect Reconstruction (PR) condition for oversampled filter banks is $R(z) \cdot E(z) = z^{-\delta} I_D$, where $E(z)$ is an $N \times D$ "tall-thin" analysis polyphase matrix, and $R(z)$ is a $D \times N$ "short-fat" synthesis polyphase matrix.

The design procedure for oversampled DFT-modulated FBs, especially for FIR filters, is significantly more flexible:

1. **Design Analysis Filters**: Design the prototype analysis filter $H_0(z)$ (and implicitly all $H_n(z)$ via modulation) to meet desired frequency specifications.
2. **Determine Analysis Polyphase Matrix** $E(z)$: From the analysis filters, the $N \times D$ polyphase matrix $E(z)$ is determined.
3. **Compute Synthesis Polyphase Matrix** $R(z)$: The key is to find an $R(z)$ that satisfies the PR condition $R(z) \cdot E(z) = z^{-\delta} I_D$.
    - Because $D < N$, the system of equations involved in finding $R(z)$ (given $E(z)$) is generally **underdetermined**. This means there are typically **infinitely many solutions** for $R(z)$.
    - This flexibility implies that, unlike the maximally decimated case, an **FIR** $R(z)$ **can almost always be found** (except in contrived cases) if $E(z)$ is FIR.
    - A commonly chosen solution from the infinite possibilities is the **"minimum-norm solution"**. This solution is obtained using the pseudo-inverse of $E(z)$ and corresponds to a specific least squares estimator (from Chapter 11). This choice is desirable as it provides a synthesis filter bank with good properties.
        - If there is **no subband processing** (i.e., output signals equal input signals within subbands), the minimum-norm synthesis filter bank guarantees **Perfect Reconstruction** (the output signal equals the input signal up to a delay).
        - If there **is subband processing**, the minimum-norm synthesis filter bank provides an output signal that corresponds to the input signal for which the analysis filter bank yields subband signals optimally

close (in a least squares sense) to the processed subband signals. This is achieved by an orthogonal projection.

In conclusion, oversampling significantly enhances the design flexibility for DFT-modulated filter banks, allowing for straightforward design procedures that guarantee Perfect Reconstruction with FIR filters.

3.3 Question 3

1. **What is a rank-1 update? How does it define the complexity?**

   A **rank-1 update** refers to the modification of a matrix by adding or subtracting an outer product of two vectors, which results in a matrix of rank 1. In the context of Recursive Least Squares (RLS) algorithms, this typically involves updating the inverse of the input correlation matrix (Ψ_{uu}). For instance, when a new data sample u_k arrives, the correlation matrix $\Psi_{uu}[k]$ can be expressed as an update of the previous one, $\Psi_{uu}[k-1]$, plus a rank-1 term $u_k u_k^T$.

   The **matrix inversion lemma** is a crucial tool that exploits this rank-1 update structure. It provides a direct way to compute the inverse of the updated matrix from the inverse of the previous matrix, without having to perform a full matrix inversion from scratch at each time step.

   This mechanism **defines the complexity** of the standard RLS algorithm. By using the matrix inversion lemma to recursively update the inverse of the input correlation matrix, the computational complexity per time update is reduced to $O(L^2)$ **arithmetic operations**, where L is the filter order (or number of taps). This is significantly more computationally intensive than LMS algorithms, which have an $O(L)$ complexity, but it offers advantages in terms of convergence speed and tracking ability. Faster RLS algorithms aim to reduce this complexity to $O(L)$.

2. **Residuals: explain a priori and a posteriori residuals of SFG.**

   In the context of Recursive Least Squares (RLS) algorithms, particularly those based on QR Decomposition (QRD-RLS), the "star" ($\star$) or epsilon (ϵ) term generated at the bottom of the **signal flow graph (SFG)** of the QRD updating process is significant because it can be used to compute the least squares residuals **without explicitly computing the least squares filter vector**.

   There are two types of residuals that can be extracted:

   - **A priori residual**: This is the error calculated **before** the filter coefficients are updated using the current data. It is defined as $d_k - u_k^T w_{LS}[k-1]$. This represents the error given the current input u_k and the filter coefficients $w_{LS}[k-1]$ from the *previous* time step.
   - **A posteriori residual**: This is the error calculated **after** the filter coefficients have been updated using the current data. It is defined as $d_k - u_k^T w_{LS}[k]$. This represents the error given the current input u_k and the *newly updated* filter coefficients $w_{LS}[k]$.

   In QRD-RLS, both the a priori and a posteriori residuals can be directly obtained from the ϵ term of the QRD update, scaled by a product of cosines. This ability to extract residuals directly is a valuable feature for various applications.

3. **Is it useful to use two exponential weighting factors if the output and input are very different: static input and non static output (for example)? (something like this)**

   The provided sources discuss **exponentially weighted RLS** using a single weighting factor, λ, also known as a "forget factor". This factor is applied to past error terms in the cost function, giving a smaller weight to older data. The purpose of this exponential weighting is to allow the algorithm to adapt to **time-varying environments** and track changes in the underlying system or statistics.

The sources **do not explicitly discuss the use of two separate exponential weighting factors** for input and output, or for different aspects of the data (e.g., one for the input vector and another for the desired output signal). If the relationship between a *static input* and a *non-static output* is due to a time-varying "plant" or channel (as implied in system identification or channel equalization scenarios), the single exponential weighting factor (λ) in RLS is designed to address this. It causes the algorithm to emphasize more recent data, allowing the filter coefficients to track the evolving relationship between input and output.

While using multiple, distinct weighting factors might be explored in more advanced adaptive filtering literature for specific scenarios (e.g., if input statistics and noise statistics evolve at different rates), the provided sources do not suggest or justify such an approach. Based solely on the given material, a single, uniformly applied exponential weighting factor is the standard method for enabling RLS to track time-varying systems.

3.4 Question 4 (Acoustic modem project)

1. **Explain how the OFDM modulation format (include cyclic prefix) provides easy equalization of the transmission channel. What conditions are to be met for the cyclic prefix?**

   The **OFDM (Orthogonal Frequency Division Multiplexing)** modulation format facilitates easy equalization of a transmission channel by transforming the complex time-domain convolution into simpler frequency-domain scalar multiplications.

   Here's how it works, including the role of the **cyclic prefix (CP)**:

   - **OFDM Principle**: OFDM works by modulating many different carriers with low-rate bit streams. These modulated carriers are then summed and transmitted as a high-rate bit stream, which is divided across hundreds of low-rate sub-streams. The modulation at the transmitter (Tx) is typically performed using an Inverse Fast Fourier Transform (IFFT), and demodulation at the receiver (Rx) uses a Fast Fourier Transform (FFT).
   - **Channel Effect**: A typical transmission channel, such as the acoustic channel in the project, can be modelled as a linear FIR filter $H(z)$. Without any special precautions, transmitting data over such a channel would result in **linear convolution** in the time domain. This causes **inter-symbol interference (ISI)** (where past symbols interfere with the current symbol) and **inter-carrier interference (ICI)** (where different carriers interfere with each other), making equalization complex.
   - **Role of Cyclic Prefix**: To combat ISI and ICI, a **cyclic prefix (CP)** is inserted at the Tx. For each block of N transmitted samples (an OFDM symbol), the last L samples of the block are copied and prepended to the beginning of the block, forming a new block of $N + L$ samples. Here, L represents the order of the FIR channel.
   - **Easy Equalization Mechanism**:
     - At the Rx, the L samples of the cyclic prefix are simply **discarded**.
     - This action effectively converts the **linear convolution** operation of the channel into a **circular convolution**.
     - A fundamental property of circulant matrices (which represent circular convolution) is that they can be **diagonalized by the DFT matrix**. This means that the effect of the channel in the frequency domain becomes a simple scalar multiplication for each carrier.
     - Specifically, if $X_{\tilde{k}}(n)$ is the transmitted symbol for the n-th carrier in the $\tilde{k}$-th OFDM block, and $Y_{\tilde{k}}(n)$ is the corresponding received symbol after demodulation (FFT at the Rx), their relationship becomes $Y_{\tilde{k}}(n) = H_n \cdot X_{\tilde{k}}(n)$. Here, H_n is the **channel frequency response evaluated at the n-th carrier**.
     - This simplifies equalization immensely: instead of a complex multi-tap equalizer for the entire signal, a **1-tap Frequency-Domain Equalizer (FEQ)** can be applied independently to each carrier by simply dividing

the received symbol by the estimated channel gain for that carrier: estimate$\{X_{\tilde{k}}(n)\} = (H_n)^{-1} \cdot Y_{\tilde{k}}(n)$.

Conditions for the Cyclic Prefix:

- The length of the cyclic prefix (L) must be at least **equal to or greater than the order (length minus one) of the channel's impulse response**. This ensures that the channel's memory is fully covered within the CP, thus preventing ISI between OFDM symbols and enabling circular convolution.
- The length L should ideally be much smaller than the OFDM symbol length N (i.e., $L \ll N$) to minimize overhead and maintain efficient data transmission.

2. **Explain how RLS is used for channel equalization in the acoustic modem project. What input for the adaptive filter? What is the desired output signal? What is the order of the filter?**

   In the acoustic modem project, **Recursive Least Squares (RLS)** algorithms are employed for **adaptive equalization** of the transmission channel, particularly in Weeks 7-8 for decision-directed adaptive equalization. RLS is an adaptive filtering algorithm that recursively computes the least squares solution to an estimation problem.

   Here's how RLS is used in this context:

   - **Context of Equalization**: In OFDM, as explained, equalization is performed in the frequency domain, with each carrier essentially experiencing a simple scalar attenuation H_n. Therefore, the equalization task reduces to estimating these complex scalar values for each carrier and then performing a simple division.
   - **Input for the Adaptive Filter**: For each individual carrier (or "subband"), the **input for the adaptive filter** (which is estimating H_n or its inverse) is the **received frequency-domain symbol** $Y_{\tilde{k}}(n)$ for that specific carrier. In the overall system, this means there is an adaptive filter for each of the N carriers/subbands.
   - **Desired Output Signal**: The **desired output signal** for the adaptive filter for carrier n is the **original transmitted frequency-domain symbol** $X_{\tilde{k}}(n)$ for that carrier.
     - In a **training phase**, known, pre-defined training sequences are transmitted. The receiver knows these original symbols, so $X_{\tilde{k}}(n)$ is directly available as the desired signal.
     - In a **decision-directed mode** (as mentioned for Weeks 7-8), after an initial training phase, the receiver's best estimate (decision) of the transmitted symbol is used as the desired signal for further adaptation.
   - **Order of the Filter**: Because the cyclic prefix transforms the channel's frequency-domain effect into a simple scalar multiplication per carrier, the equalizer required for each carrier (or "subband") is a **1-tap Frequency-Domain Equalizer (FEQ)**. This means the **order of the adaptive filter for each carrier/subband is 1**. The RLS algorithm for each carrier estimates this single complex coefficient, effectively determining the inverse of the channel gain H_n for that specific frequency.

4 License

4.1 Modification

To contribute to this work or any other one from this project please find more information at the Github repository.
