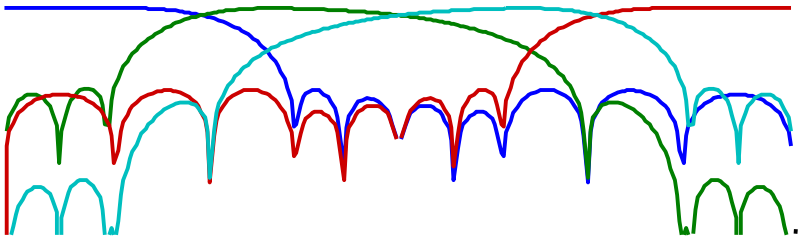


Digital Signal Processing II

Prof. Dr.-Ing. Stefan Wolter

stefan.wolter@hs-bremen.de

Department of Electrical Engineering and Computer Science
University of Applied Sciences Bremen

CONTENTS

MATERIALS

https://wolter.hs-bremen.de/skripte/DSP_II/

- Lectures Slides
- Lab Exercises and Solutions
- Further Materials

REFERENCES

Applied Digital Signal Processing
D. G. Manolakis, V. K. Ingle, Cambridge University Press

Digital Signal Processing
S. K. Mitra, McGraw-Hill

Digital Signal Processing
J. G. Proakis, D. G. Manolakis, Pearson

Digitale Signalverarbeitung
K.-D. Kammeyer, K. Kroschel, Vieweg + Teubner Studium

Multiraten-Signalverarbeitung
N. Fliege, B. G. Teubner Verlag

Signaltheorie
A. Mertens, Vieweg + Teubner Studium

1. Efficient Computation of the Discrete Fourier Transform

Prelimimary Note

- Recall the N-point DFT $X[k], 0 \le k \le N-1$, of a length-N sequence $x[n], 0 \le n \le N-1$, defined by

$$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}, \text{ where } W_N = e^{-j2\pi/N}$$

- The direct computation of all N DFT samples requires approximately N^2 complex multiplications and $N(N-1)$ complex additions
- However, there is much that we can do to reduce the computational effort of the DFT

1.1 Efficient DFT Computation of Real Sequences

- In most practical applications, sequences of interest are real
- In such cases, the symmetry properties of the DFT can be exploited to make the DFT computations more efficient
- Next, we exploit the symmetry properties in two different cases

N-Point DFT of Two Length-N Real Sequences

- Let $g[n]$ and $h[n]$ be two length-N real sequences with $G[k]$ and $H[k]$ denoting their respective N-point DFTs
- These two N-point DFTs can be computed efficiently using a single N-point DFT
- Define a complex length-N sequence

$$x[n] = g[n] + j\,h[n]$$

where $g[n] = \mathcal{R}e\{x[n]\}$ and $h[n] = \mathcal{I}m\{x[n]\}$

N-Point DFT of Two Length-*N* Real Sequences

- Let $X[k]$ denote the N-point DFT of $x[n]$
- Then, from the symmetry properties of the DFT of a complex sequence we arrive at

$$G[k]=\frac{1}{2}\{X[k]+X^*[\langle -k\rangle_N]\}$$

$$H[k]=\frac{1}{2j}\{X[k]-X^*[\langle -k\rangle_N]\}$$

- Recall the circular folding

$$X^*[\langle -k\rangle_N]=X^*[\langle N-k\rangle_N]$$

N-Point DFT of Two Length-*N* Real Sequences

- Example: Compute the 4-point DFTs of the two real sequences $g[n]$ and $h[n]$ given below

$$g[n]=\{\underset{\uparrow}{1}\quad 2\quad 0\quad 1\},\quad h[n]=\{\underset{\uparrow}{2}\quad 2\quad 1\quad 1\}$$

- Then $x[n]=g[n]+j\cdot h[n]$ is given by

$$x[n]=\{\underset{\uparrow}{1+j2}\quad 2+j2\quad j\quad 1+j\}$$

N-Point DFT of Two Length-*N* Real Sequences

- Its DFT $X[k]$ is then

$$\begin{bmatrix}X[0]\\X[1]\\X[2]\\X[3]\end{bmatrix}=\begin{bmatrix}1&1&1&1\\1&-j&-1&j\\1&-1&1&-1\\1&j&-1&-j\end{bmatrix}\begin{bmatrix}1+j2\\2+j2\\j\\1+j\end{bmatrix}=\begin{bmatrix}4+j6\\2\\-2\\j2\end{bmatrix}$$

- From the above

$$X^*[k]=[4-j6\quad 2\quad -2\quad -j2]$$

N-Point DFT of Two Length-*N* Real Sequences

- Hence

$$X^*[\langle 4-k\rangle_4]=[4-j6\quad -j2\quad -2\quad 2]$$

- Therefore

$$G[k]=\{4\quad 1-j\quad -2\quad 1+j\}$$

$$H[k]=\{6\quad 1-j\quad 0\quad 1+j\}$$

N-point DFT of a 2*N*-Point Real Sequence

- Let $v[n]$ be a length-$2N$ real sequence with an $2N$-point DFT $V[k]$
- Define two length-N real sequences $g[n]$ and $h[n]$ as follows

  $g[n]=v[2n],\ \ h[n]=v[2n+1],\ \ 0\le n\le N-1$
- Let $G[k]$ and $H[k]$ denote their respective N-point DFTs

N-point DFT of a 2*N*-Point Real Sequence

- Now, we define a length-N complex sequence

  $$x[n]=g[n]+j\cdot h[n]$$

  with an N-point DFT $X[k]$
- Then, we determine $G[k]$ and $H[k]$ recalling

  $$G[k]=\frac{1}{2}\{X[k]+X^*[\langle -k\rangle_N]\}$$

  $$H[k]=\frac{1}{2j}\{X[k]-X^*[\langle -k\rangle_N]\}$$

N-point DFT of a 2*N*-Point Real Sequence

- Calculate the DFT $V[k]=\sum_{n=0}^{2N-1} v[n]W_{2N}^{nk}$ as

$$=\sum_{n=0}^{N-1} v[2n]W_{2N}^{2nk}+\sum_{n=0}^{N-1} v[2n+1]W_{2N}^{(2n+1)k}$$

$$=\sum_{n=0}^{N-1} g[n]W_N^{nk}+\sum_{n=0}^{N-1} h[n]W_N^{nk}W_{2N}^{k}$$

$$=\sum_{n=0}^{N-1} g[n]W_N^{nk}+W_{2N}^{k}\sum_{n=0}^{N-1} h[n]W_N^{nk},\ 0\le k\le 2N-1$$

N-point DFT of a 2*N*-Point Real Sequence

- The first sum is the N-point DFT $G[k]$ of $g[n]$, whereas the second sum is the N-point DFT $H[k]$ of $h[n]$ and hence we can express $V[k]$ as

  $$V[k]=G[\langle k\rangle_N]+W_{2N}^{k}H[\langle k\rangle_N],\ \ 0\le k\le 2N-1$$

- Example: 8-point DFT of the real sequence

  $$v[n]=\{1\ \ 2\ \ 2\ \ 2\ \ 0\ \ 1\ \ 1\ \ 1\}$$
  $$\uparrow$$

N-point DFT of a 2*N*-Point Real Sequence

- We form two length-4 real sequences as follows

$$g[n] = v[2n] = \{\underset{\uparrow}{1} \quad 2 \quad 0 \quad 1\}$$

$$h[n] = v[2n+1] = \{\underset{\uparrow}{2} \quad 2 \quad 1 \quad 1\}$$

- Now

$$V[k] = G[\langle k \rangle_4] + W_8^k H[\langle k \rangle_4], \ 0 \le k \le 7$$

- Substituting the values of the 4-point DFTs $G[k]$ and $H[k]$ computed earlier we get $V[k]$

N-point DFT of a 2*N*-Point Real Sequence

$$V[0] = G[0] + H[0] = 4 + 6 = 10$$
$$V[1] = G[1] + W_8^1 H[1] = 1 - j2.4142$$
$$V[2] = G[2] + W_8^2 H[2] = -2$$
$$V[3] = G[3] + W_8^3 H[3] = 1 - j0.4142$$
$$V[4] = G[0] + W_8^4 H[0] = -2$$
$$V[5] = G[1] + W_8^5 H[1] = 1 + j0.4142$$
$$V[6] = G[2] + W_8^6 H[2] = -2$$
$$V[7] = G[3] + W_8^7 H[3] = 1 + j2.4142$$

1.2 Goertzel Algorithm

- Using the identity $W_N^{-kN} = 1$ we can write the DFT of a length-N sequence as

$$X[k] = W_N^{-kN} \sum_{\ell=0}^{N-1} x[\ell] W_N^{k\ell} = \sum_{\ell=0}^{N-1} x[\ell] W_N^{-k(N-\ell)}$$

- This appears to be a linear convolution providing a filtering perspective to the DFT
- Next, we derive the **Goertzel algorithm** using a recursive computation applying an IIR digital filter

Goertzel Algorithm

- For $n \ge 0$, we define $y_k[n] = \sum_{\ell=0}^{n} x_e[\ell] h_k[n-\ell]$
- $y_k[n]$ is the linear convolution of the causal input

$$x_e[n] = \begin{cases} x[n], & 0 \le n \le N-1 \\ 0, & otherwise \end{cases}$$

with the complex-valued causal impulse response

$$h_k[n] = W_N^{-kn} \mu[n] = \begin{cases} W_N^{-kn}, & n \ge 0 \\ 0, & n < 0 \end{cases}$$

- Observe that $X[k] = y_k[n]\big|_{n=N}$

Goertzel Algorithm

- z-transform of $y_k[n]$ yields

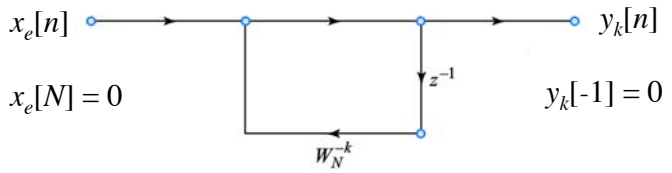
$$Y_k(z) = Z\{y_k[n]\} = \frac{X_e(z)}{1 - W_N^{-k} z^{-1}} = H_k(z) X_e(z)$$

where $H_k(z) = Z\{h_k[n]\} = 1/(1 - W_N^{-k} z^{-1})$

- Thus, $y_k[n]$ is the output of an initially relaxed LTI digital filter $H_k(z)$ of first-order with an input $x_e[n]$
- When $n = N$, the output sequence of this filter is one DFT sample X [k]

Goertzel Algorithm

- Flow graph of the first-order recursive filter



- The recursive DFT computation is given by

$$y_k[n] = x_e[n] + W_N^{-k} y_k[n-1], \quad 0 \le n \le N$$

with $y_k[-1] = 0$ and $x_e[N] = 0$

Goertzel Algorithm

- The computation of each new value of $y_k[n]$ requires 4 real multiplications and 4 real additions
- Thus, a single DFT sample $X[k]$ requires $4N$ real multiplications and $4N$ real additions
- The computation of all N DFT samples requires $4N^2$ real multiplications and $4N^2$ real additions

Goertzel Algorithm

- The direct computation of all N DFT samples requires N^2 complex multiplications and $N(N-1)$ complex additions
- Equivalently, direct computation of all N DFT samples requires $4N^2$ real multiplications and $N(4N-2)$ real additions
- Thus, the recursive computation with the first-order IIR filter requires $2N$ more real additions than the direct DFT computation

Goertzel Algorithm

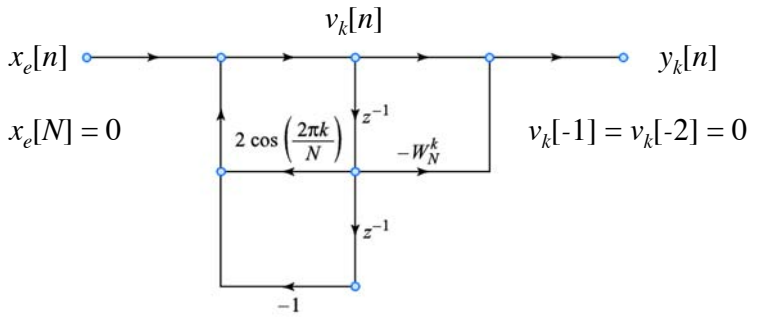
- The algorithm can be made computationally more efficient using a little mathematical trick

$$H_k(z) = \frac{1}{1 - W_N^{-k} z^{-1}} = \frac{1 - W_N^k z^{-1}}{(1 - W_N^{-k} z^{-1})(1 - W_N^k z^{-1})}$$

$$= \frac{1 - W_N^k z^{-1}}{1 - 2\cos(2\pi k / N) z^{-1} + z^{-2}}$$

- The flow graph to compute the DFT with the second-order Goertzel filter is given next

Goertzel Algorithm



$$v_k[n] = x_e[n] + 2\cos(2\pi k / N) v_k[n-1]$$
$$- v_k[n-2],\ 0 \le n \le N$$
$$X[k] = y_k[N] = v_k[N] - W_N^k v_k[N-1]$$

Goertzel Algorithm

- Each sample of the intermediate signal $v_k[n]$ involves 2 real multiplications and 4 real add.
- Complex multiplication by W_N^k needs to be performed only once at $n = N$
- Therefore, the computation of one sample of $X[k]$ requires $(2N + 4)$ real multiplications and $(4N + 4)$ real additions
- All N DFT samples require $2N(N + 2)$ real multiplications and $4N(N + 1)$ real additions

Goertzel Algorithm

- In the realization of $H_{N-k}(z)$, the feedback multiplier is the same as that in $H_k(z)$, i.e.

$$2\cos(2\pi(N - k)/N) = 2\cos(2\pi k / N)$$

- Thus, $v_{N-k}[n] = v_k[n]$, i.e., the intermediate variables computed to determine $X[k]$ can be used again to determine $X[N - k]$
- Only difference between the two structures is the feed-forward multiplier which is now W_N^{-k}, that is the complex conjugate of W_N^k

Goertzel Algorithm

- Thus, the computation of $X[k]$ and $X[N - k]$ require $2(N+4)$ real multiplications and $4(N+2)$ real additions
- Computation of all N DFT samples require approximately N^2 real multiplications and approximately $2N^2$ real additions
- The number of real multiplications is about one-fourth and the number of real additions is about one-half of those needed in the direct DFT computation

Goertzel Algorithm

- The MATLAB function **`goertzel(x,k)`** computes the ***k***-th DFT-sample of the input **`x`** using the second-order Goertzel algorithm
- Note that the algorithm is very attractive in applications requiring the computation of a few samples of the DFT
- One example is the dual-tone multifrequency (DTMF) signal detection in the TOUCH-TONE telephone dialing system

1.3 Chirp Transform Algorithm

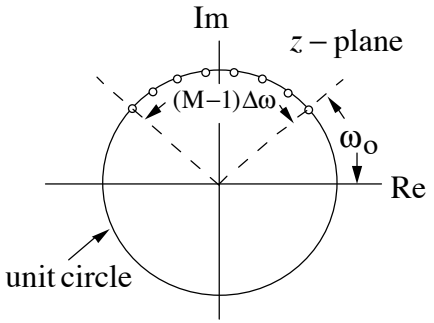
- Let $x[n]$ be a length-N sequence with a Fourier transform $X(e^{j\omega})$
- We consider the evaluation of M samples of $X(e^{j\omega})$ that are equally spaced in angle on the unit circle at frequencies

$$\omega_k = \omega_o + k\Delta\omega, \quad 0 \le k \le M-1$$

where the starting frequency ω_o and the frequency increment $\Delta\omega$ can be chosen arbitrarily

Chirp Transform Algorithm

- Figure below illustrates the problem



$$\omega_k = \omega_o + k\Delta\omega, \quad 0 \le k \le M-1$$

Chirp Transform Algorithm

- Thus, the problem is to evaluate

$$X(e^{j\omega_k}) = \sum_{n=0}^{N-1} x[n]e^{-j\omega_k n}, \quad 0 \le k \le M-1$$

or, with W defined as

$$W = e^{-j\Delta\omega}$$

to obtain

$$X(e^{j\omega_k}) = \sum_{n=0}^{N-1} x[n]e^{-j\omega_o n} W^{nk}$$

Chirp Transform Algorithm

- Using the identity $nk = \frac{1}{2}[n^2 + k^2 - (k-n)^2]$ we can write

$$X(e^{j\omega_k}) = \sum_{n=0}^{N-1} x[n]e^{-j\omega_o n} W^{\frac{n^2}{2}} W^{\frac{k^2}{2}} W^{\frac{-(k-n)^2}{2}}$$

- Let $g[n] = x[n]e^{-j\omega_o n} W^{\frac{n^2}{2}}$ we arrive at

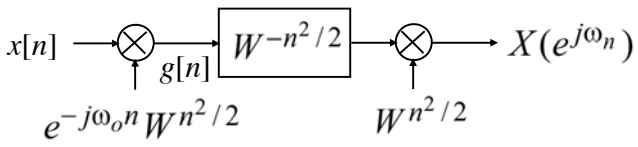
$$X(e^{j\omega_k}) = W^{\frac{k^2}{2}} \left(\sum_{n=0}^{N-1} g[n] W^{\frac{-(k-n)^2}{2}} \right), 0 \le k \le M-1$$

Chirp Transform Algorithm

- Interchanging k and n we get

$$X(e^{j\omega_n}) = W^{n^2/2} \left(\sum_{k=0}^{N-1} g[k] \cdot W^{-(n-k)^2/2} \right)$$

- Thus, $X(e^{j\omega_n})$ corresponds to the convolution of $g[n]$ with $W^{-n^2/2}$ followed by a sequence multiplication with $W^{n^2/2}$ as indicated below



Chirp Transform Algorithm

- The sequence $W^{-n^2/2}$ can be thought as a complex exponential sequence with linearly increasing frequency $(\Delta\omega)n$
- Signals are called chirp signals, hence the name **Chirp Transform Algorithm** (CTA)
- Now we evaluate

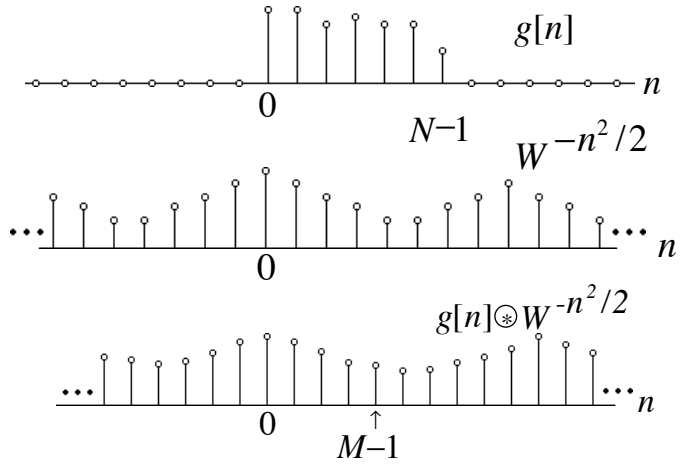
$$X(e^{j\omega_n}) = W^{\frac{n^2}{2}} \left(\sum_{k=0}^{N-1} g[k] W^{\frac{-(n-k)^2}{2}} \right)$$

Chirp Transform Algorithm

- The algorithm requires sequences of various lengths and hence care must be taken in the evaluation of the convolution
- Since $g[n]$ is a length-N sequence, only a finite segment of the infinite length sequence $W^{-n^2/2}$ is used in obtaining the convolution sum over the interval $0 \le n \le M-1$
- To obtain the finite segment, we consider the involved sequences

Chirp Transform Algorithm

- Typical sequences are shown below



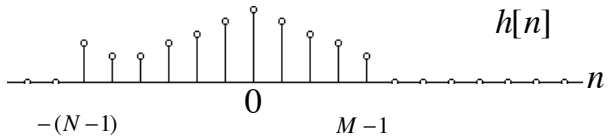
Chirp Transform Algorithm

- The segment of the sequence $W^{-n^2/2}$ used in the convolution sum is from the interval

$$-(N-1) \le n \le M-1$$

- Now, we define the impulse response

$$h[n] = \begin{cases} W^{-n^2/2}, & -(N-1) \le n \le (M-1) \\ 0, & \text{otherwise} \end{cases}$$

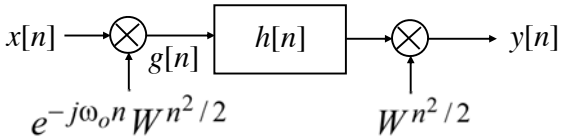


Chirp Transform Algorithm

- Therefore

$$g[n] \circledast W^{-n^2/2} = g[n] \circledast h[n], 0 \le n \le M-1$$

- Hence, the computation of the frequency samples $X(e^{j\omega_n})$ can be performed using a three-step procedure as indicated below



where $y[n] = X(e^{j\omega_n}), 0 \le n \le M-1$

Chirp Transform Algorithm

- Advantages of the CTA:
  1) $M = N$ is not required as in FFT algorithms
  2) Neither N nor M do not have to be composite numbers (can be prime)
  3) Parameters ω_o and $\Delta\omega$ are arbitrary
- The Chirp z-transform Algorithm (CZT) generalizes the CTA to compute a spiral contour in the z-plane
- MATLAB provides `czt` in its SP Toolbox

1.4 Decimation-in-Time FFT Algorithm

- The basic idea behind all FFT algorithms is a divide-and-combine approach to decompose a DFT of length N into smaller length DFTs that are merged to form the DFT of length N
- To this end, the **Decimation-In-Time** (DIT) **FFT algorithm** decomposes or decimates the input $x[n]$ into smaller sequences that are used to form smaller length DFTs

Decimation-in-Time FFT Algorithm

- The algorithm exploits the periodicity and symmetry properties of the twiddle factors
- These properties are the **periodicity in *n* and *k***

$$W_N^{nk} = W_N^{n(k+N)} = W_N^{k(n+N)}$$

and the **complex conjugate symmetry**

$$W_N^{k(N-n)} = W_N^{-nk} = (W_N^{nk})^*$$

Decimation-in-Time FFT Algorithm

- Consider a sequence $x[n]$ of length $N = 2^\mu$
- We can express its z-transform using a so called 2-band polyphase decomposition as

$$X(z) = X_0(z^2) + z^{-1}X_1(z^2)$$

where

$$X_0(z) = \sum_{n=0}^{(N/2)-1} x_0[n]z^{-n} = \sum_{n=0}^{(N/2)-1} x[2n]z^{-n}$$

$$X_1(z) = \sum_{n=0}^{(N/2)-1} x_1[n]z^{-n} = \sum_{n=0}^{(N/2)-1} x[2n+1]z^{-n}$$

Decimation-in-Time FFT Algorithm

- Evaluating $X(z)$ on the unit circle at N equally spaced points, $z = W_N^{-k}$, $0 \le k \le N-1$, we arrive at the N–point DFT of $x[n]$

$$X[k] = X_0[\langle k \rangle_{N/2}] + W_N^k X_1[\langle k \rangle_{N/2}], 0 \le k \le N-1$$

where $X_0[k]$ and $X_1[k]$ are the $(N/2)$–point DFTs of the $(N/2)$–point sequences $x_0[n]$ and $x_1[n]$

Decimation-in-Time FFT Algorithm

$$\text{i.e., } X_0[k] = \sum_{r=0}^{(N/2)-1} x_0[r] W_{N/2}^{rk}$$

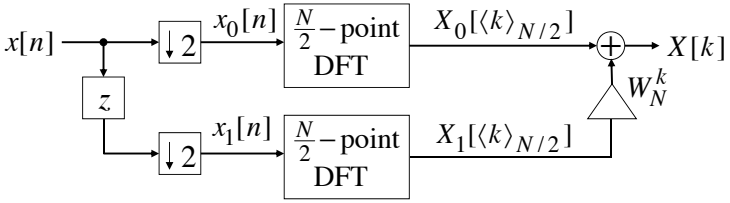
$$= \sum_{r=0}^{(N/2)-1} x[2r] W_{N/2}^{rk}, \; 0 \le k \le \frac{N}{2}-1$$

$$X_1[k] = \sum_{r=0}^{(N/2)-1} x_1[r] W_{N/2}^{rk}$$

$$= \sum_{r=0}^{(N/2)-1} x[2r+1] W_{N/2}^{rk}, \; 0 \le k \le \frac{N}{2}-1$$

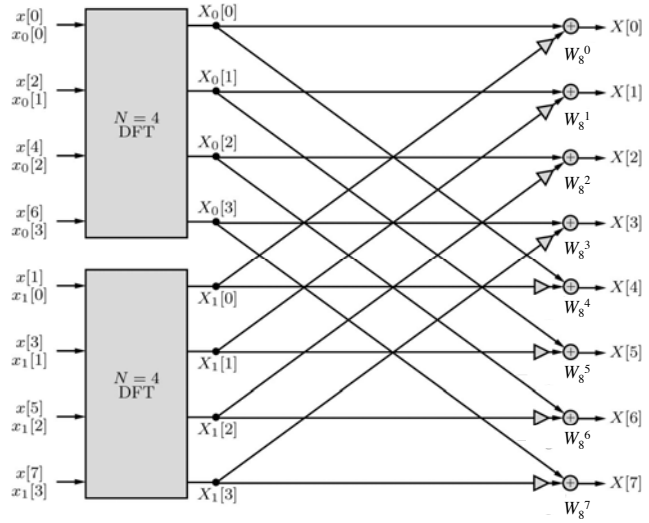
Decimation-in-Time FFT Algorithm

- A DIT decomposition dividing an N–point DFT into two $(N/2)$–point DFTs is shown below



- The block diagram using two 4–point DFTs for $N = 8$ is given next

Decimation-in-Time FFT Algorithm



Decimation-in-Time FFT Algorithm

- Continuing the process we can express $X_0[k]$ and $X_1[k]$ as a weighted combination of two $(N/4)$–point DFTs
- For $X_0[k]$ we can write

$$X_0[k] = X_{00}[\langle k \rangle_{N/4}] + W_{N/2}^k X_{01}[\langle k \rangle_{N/4}], \quad 0 \le k \le (N/2)-1$$

where $X_{00}[k]$ and $X_{01}[k]$ are the $(N/4)$–point DFTs of the $(N/4)$-point sequences $x_{00}[n] = x_0[2n]$ and $x_{01}[n] = x_0[2n+1]$

Decimation-in-Time FFT Algorithm

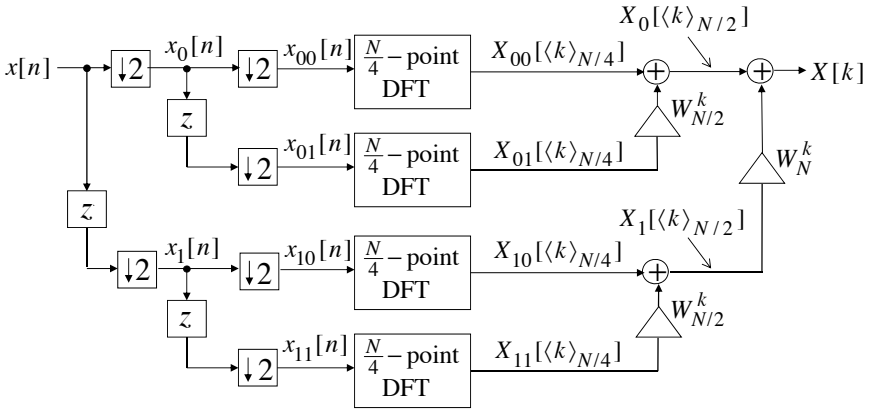
- Likewise, we can express

$$X_1[k] = X_{10}[\langle k \rangle_{N/4}] + W_{N/2}^k X_{11}[\langle k \rangle_{N/4}], \quad 0 \le k \le (N/2)-1$$

where $X_{10}[k]$ and $X_{11}[k]$ are the $(N/4)$–point DFTs of the $(N/4)$–point sequences $x_{10}[n] = x_1[2n]$ and $x_{11}[n] = x_1[2n+1]$

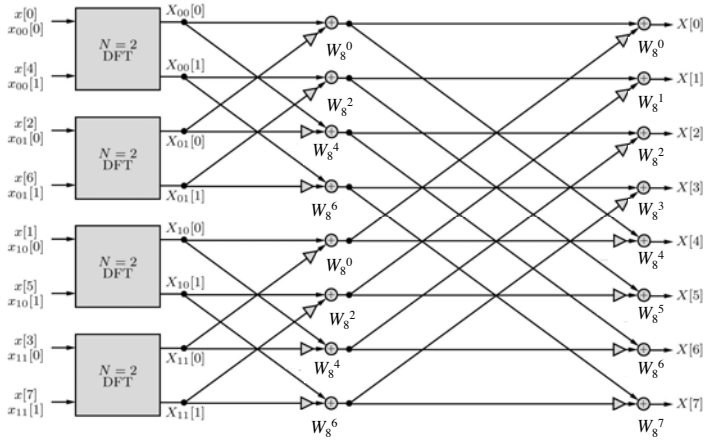
Decimation-in-Time FFT Algorithm

- DIT decomposition dividing an N–point DFT into four $(N/4)$–point DFTs



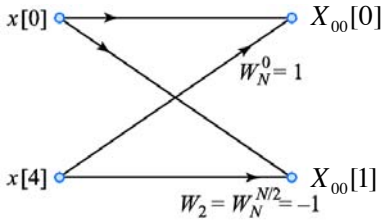
Decimation-in-Time FFT Algorithm

- Block diagram for $N = 8$ using 2–point DFTs



Decimation-in-Time FFT Algorithm

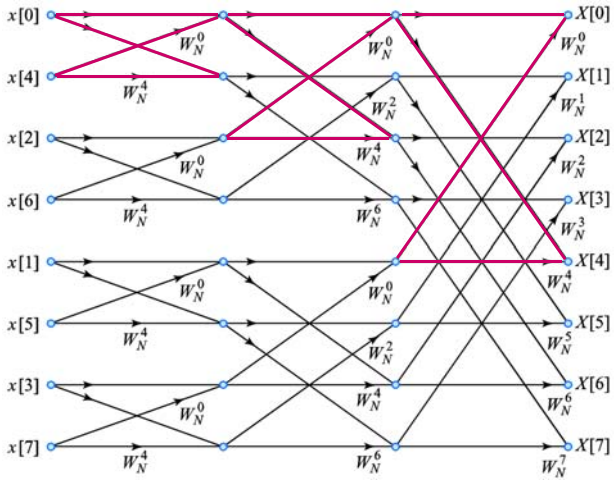
- For $N = 8$, the (N/4)–point DFT is a 2–point DFT and no further decomposition is possible
- The four 2–point DFTs $X_{ij}[k],\ \ i, j = 0,1$ can be computed easily, e.g. for $i = 0$ and $j = 0$ we have $X_{00}[k] = x[0] + W_2^k x[4],\ \ k = 0,1$



Decimation-in-Time FFT Algorithm

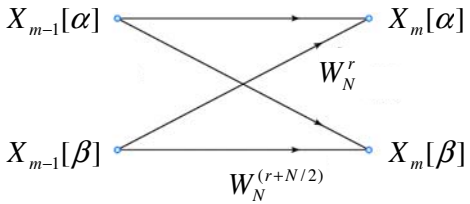
- When $N = 2^\mu$, the number of stages for the decimation-in-time FFT algorithm is given by $\mu = \log_2 N$
- Thus, the computation of the N–point DFT with $\log_2 N$ stages results in a fast algorithm with computational complexity proportional to $N \log_2 N$, i.e. it has the order $O\{N \log_2 N\}$
- The flow-graph developed so far shows that each stage employs the same basic module

Decimation-in-Time FFT Algorithm



Decimation-in-Time FFT Algorithm

- In the basic module two output variables are generated by a weighted combination of two input variables as shown below



- This is the so-called butterfly module

Decimation-in-Time FFT Algorithm

- Input-output relations of a butterfly module

$$X_m[\alpha] = X_{m-1}[\alpha] + W_N^r X_{m-1}[\beta]$$

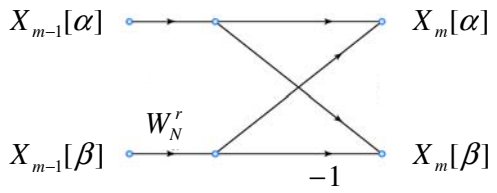
$$X_m[\beta] = X_{m-1}[\alpha] + W_N^{(r+N/2)} X_{m-1}[\beta]$$

- Substituting $W_N^{(r+N/2)} = -W_N^r$ in the second equation given above we get

$$X_m[\beta] = X_{m-1}[\alpha] - W_N^r X_{m-1}[\beta]$$

Decimation-in-Time FFT Algorithm

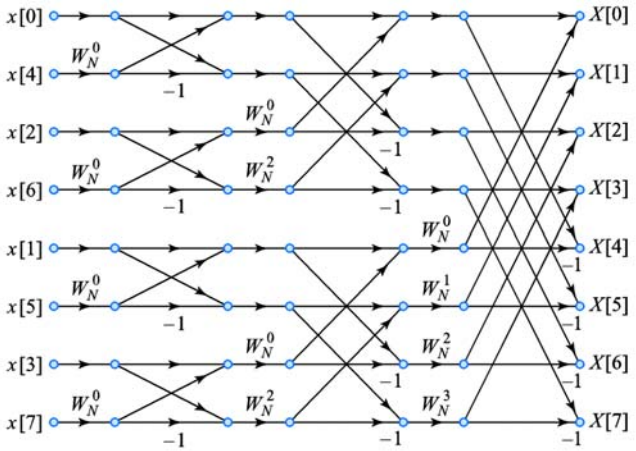
- The modified butterfly module requires only one complex multiplication as shown below



- Using the modified module reduces the total number of complex multiplications by 50%

Decimation-in-Time FFT Algorithm

- Flow-graph for $N = 8$



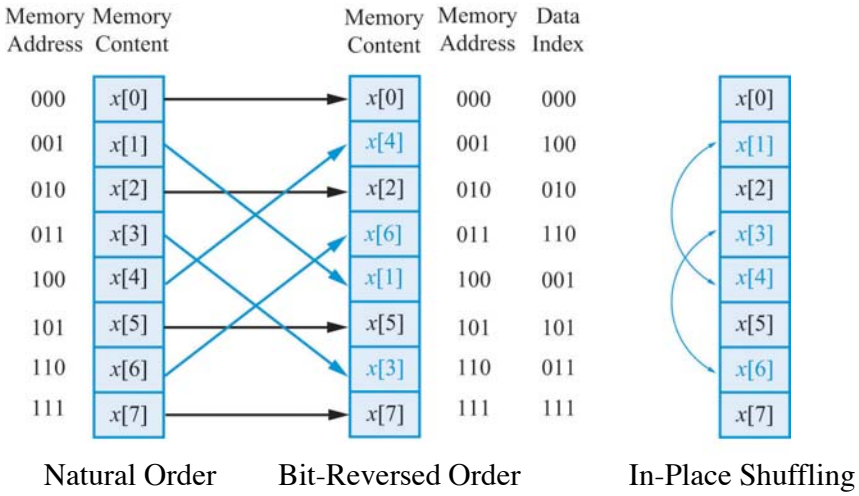
Decimation-in-Time FFT Algorithm

- Computational complexity can be reduced further by avoiding multiplications by $W_N^0 = 1$, $W_N^{N/2} = -1$, $W_N^{N/4} = j$, and $W_N^{3N/4} = -j$
- The algorithm is efficient with regard to memory requirements
- At the end of the computation at any stage, the outputs can be stored in the same memory locations previously occupied by the input variables (in-place computation)

Decimation-in-Time FFT Algorithm

- Note that the sequentially ordered input $x[n]$ must be reordered in bit-reversed order
- A sample with the binary index $b_2b_1b_0$ is moved to another location with index $b_0b_1b_2$
- The next figure illustrates the bit-reversed shuffling process of a sequence for $N = 8$
- We do not need a second memory, because bit-reversing only involves swapping pairs of elements (in-place shuffling)

Decimation-in-Time FFT Algorithm



Natural Order Bit-Reversed Order In-Place Shuffling

Radix-R Decimation-in-Time FFT Algorithm

- In our discussion so far, we have used the decimation of the input sequence by 2, known as radix–2 FFT
- There are algorithms that use successive decimations of the input sequence, e.g. by $R = 4, 8$, and so on
- These radix–R FFTs do save number of computations, but the savings are not huge, and the structure complexity increases

Radix-4 Decimation-in-Time FFT Algorithm

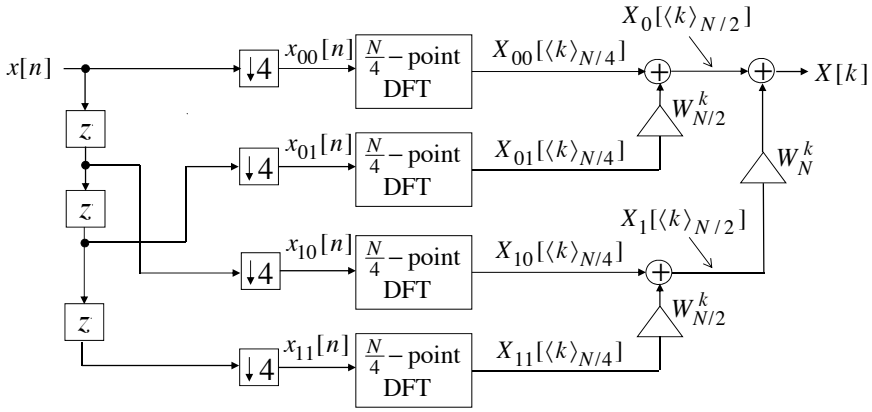
- Example: The four subsequences $x_{00}[n]$, $x_{01}[n]$, $x_{10}[n]$, and $x_{11}[n]$, in the radix–2 decomposition for $N = 8$ are given by

$x[n]$	$x[0]$	$x[1]$	$x[2]$	$x[3]$	$x[4]$	$x[5]$	$x[6]$	$x[7]$
$x_{00}[n]$	$x[0]$				$x[4]$			
$x_{01}[n]$			$x[2]$				$x[6]$	
$x_{10}[n]$		$x[1]$				$x[5]$		
$x_{11}[n]$				$x[3]$				$x[7]$

Radix-4 Decimation-in-Time FFT Algorithm

- The subsequences $x_{00}[n]$, $x_{01}[n]$, $x_{10}[n]$, and $x_{11}[n]$ are generated by a radix–2 decimation, leading to a three-stage radix–2 FFT
- However, the subsequences can be generated directly by a radix–4 decimation, leading to a two-stage radix–4 DIT FFT
- The next figure illustrates the structure of a radix–4 DIT FFT

Radix-4 Decimation-in-Time FFT Algorithm



Mixed-Radix FFT Algorithms

- Depending on the value of N, various combinations of decompositions can be used to develop different types of FFT algorithms
- If a mixture of decimations by different factors is used, the FFT algorithm is known as a **mixed-radix** or **split-radix FFT algorithm**

1.5 Decimation-in-Frequency FFT Algorithm

- The same idea behind the DIT FFT algorithm can be applied to the N–point DFT sequence $X[k]$ to decompose it sequentially into sets of smaller and smaller subsequences
- This approach leads to another class of DFT algorithms collectively known as the **Decimation-In-Frequency** (DIF) **FFT algorithms**

Decimation-in-Frequency FFT Algorithm

- Consider a sequence $x[n]$ of length $N = 2^{\mu}$
- Its z-transform can be expressed as

$$X(z) = X_a(z) + z^{-N/2} X_b(z)$$

where

$$X_a(z) = \sum_{n=0}^{(N/2)-1} x[n] z^{-n}$$

$$X_b(z) = \sum_{n=0}^{(N/2)-1} x[\frac{N}{2} + n] z^{-n}$$

Decimation-in-Frequency FFT Algorithm

- Evaluating $X(z)$ on the unit circle at W_N^{-k}

$$X[k] = \sum_{n=0}^{(N/2)-1} x[n] W_N^{nk} + W_N^{(N/2)k} \sum_{n=0}^{(N/2)-1} x[\frac{N}{2} + n] W_N^{nk}$$

and using $W_N^{(N/2)k} = (-1)^k$ we arrive at

$$X[k] = \sum_{n=0}^{(N/2)-1} (x[n] + (-1)^k x[\frac{N}{2} + n]) W_N^{nk}$$

Decimation-in-Frequency FFT Algorithm

- For k even

$$X[2\ell] = \sum_{n=0}^{(N/2)-1} (x[n] + x[\frac{N}{2} + n]) W_N^{2n\ell}$$

$$= \sum_{n=0}^{(N/2)-1} (x[n] + x[\frac{N}{2} + n]) W_{N/2}^{n\ell}, \quad 0 \le \ell \le \frac{N}{2} - 1$$

- For k odd

$$X[2\ell + 1] = \sum_{n=0}^{(N/2)-1} (x[n] - x[\frac{N}{2} + n]) W_N^{n(2\ell+1)}$$

$$= \sum_{n=0}^{(N/2)-1} (x[n] - x[\frac{N}{2} + n]) W_N^n W_{N/2}^{n\ell}, \quad 0 \le \ell \le \frac{N}{2} - 1$$

Decimation-in-Frequency FFT Algorithm

- Thus $X[2\ell]$ and $X[2\ell + 1]$ given below

$$X[2\ell] = \sum_{n=0}^{(N/2)-1} x_0[n] W_N^{n(2\ell)}$$

$$X[2\ell + 1] = \sum_{n=0}^{(N/2)-1} x_1[n] W_N^{n(2\ell)}, \quad 0 \le \ell \le \frac{N}{2} - 1$$

are the $(N/2)$–point DFTs of the subsequences

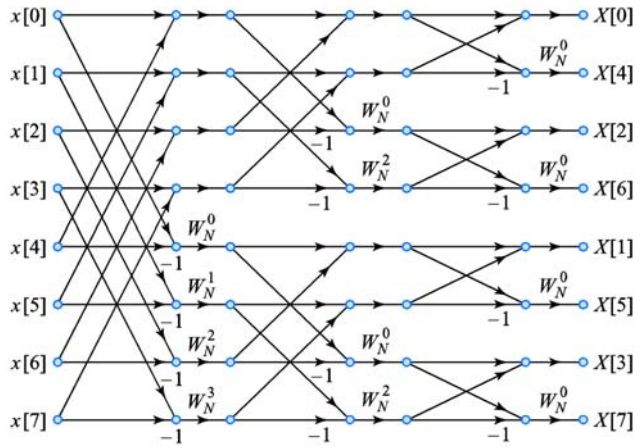
$$x_0[n] = (x[n] + x[\frac{N}{2} + n]),$$

$$x_1[n] = (x[n] - x[\frac{N}{2} + n]) W_N^n, \quad 0 \le n \le \frac{N}{2} - 1$$

Decimation-in-Frequency FFT Algorithm

- We next express the even-indexed and odd-indexed samples of each one of the two ($N/2$)–point DFTs as a sum of two ($N/4$)–point DFTs
- The process can continue until the smallest DFTs are 2–point DFTs
- The complete flow graph of the DIF FFT algorithm for $N = 8$ is shown next

Decimation-in-Frequency FFT Algorithm



Decimation-in-Frequency FFT Algorithm

- Input samples are in sequential order and output samples appear in bit-reversed order
- Computational complexity of the radix–2 DIF FFT is same as that of the radix–2 DIT FFT
- Radix–R or split radix DIF FFT algorithms can be developed similarly
- DIT and DIF FFT algorithms are collectively known as the **Cooley-Tukey FFT algorithms**

Computational Complexity of Cooley-Tukey FFT algorithms

- The computational complexity is a measure for the cost of FFT algorithms in terms of additions and multiplications, see next table
- Note that the computational complexity is just one measure to answer the question Which is the best FFT algorithm?
- Other factors to determine the performance of FFT algorithms are e.g. the structure complexity or memory requirements

Computational Complexity of Cooley-Tukey FFT algorithms

	Real Multiplications				Real Additions			
N	Radix 2	Radix 4	Radix 8	Split Radix	Radix 2	Radix 4	Radix 8	Split Radix
16	24	20		20	152	148		148
32	88			68	408			388
64	264	208	204	196	1,032	976	972	964
128	712			516	2,504			2,308
256	1,800	1,392		1,284	5,896	5,488		5,380
512	4,360		3,204	3,076	13,566		12,420	12,292
1,024	10,248	7,856		7,172	30,728	28,336		27,652

1.6 Cooley-Tukey Mappings for FFTs with Factorizable Lengths

- As we have seen one key for the previous FFTs is to define DFTs of reduced lengths
- We now introduce a general approach that can be used to derive DIT FFTs and DIF FFTs as special cases
- To this end, let us assume that the length N is a composite number, i.e. $N = N_1 N_2$
- First, we represent the indices n and k in 2–D arrays defined by two index mappings

Cooley-Tukey Mappings for FFTs with Factorizable Lengths

- $x[n]$ is entered into an 2–D array according to the row-wise mapping

$$n = N_2 n_1 + n_2, \quad \begin{cases} 0 \le n_1 \le N_1 - 1 \\ 0 \le n_2 \le N_2 - 1 \end{cases}$$

- $X[k]$ need to be extracted from an 2–D array according to the column-wise mapping

$$k = k_1 + N_1 k_2, \quad \begin{cases} 0 \le k_1 \le N_1 - 1 \\ 0 \le k_2 \le N_2 - 1 \end{cases}$$

Cooley-Tukey Mappings for FFTs with Factorizable Lengths

- Using these index mappings we can write the N–point DFT

as $$X[k] = \sum_{n=0}^{N-1} x[n] W_N^{nk}$$

$$X[k_1 + N_1 k_2] = \sum_{n_2=0}^{N_2-1} \sum_{n_1=0}^{N_1-1} x[N_2 n_1 + n_2] W_N^{(N_2 n_1 + n_2)(k_1 + N_1 k_2)}$$

$$= \sum_{n_2=0}^{N_2-1} \sum_{n_1=0}^{N_1-1} x[N_2 n_1 + n_2] W_N^{N_2 n_1 k_1} W_N^{n_2 k_1} W_N^{N_1 n_2 k_2} W_N^{N_1 N_2 n_1 k_2}$$

Cooley-Tukey Mappings for FFTs with Factorizable Lengths

- Substituting $W_N^{N_2 n_1 k_1} = W_{N_1}^{n_1 k_1}$, $W_N^{N_1 n_2 k_2} = W_{N_2}^{n_2 k_2}$, and $W_N^{N_1 N_2 n_1 k_2} = 1$, we obtain

$$X[k_1 + N_1 k_2]$$

$$= \sum_{n_2=0}^{N_2-1} \left[\left(\sum_{n_1=0}^{N_1-1} x[N_2 n_1 + n_2] W_{N_1}^{n_1 k_1} \right) W_N^{n_2 k_1} \right] W_{N_2}^{n_2 k_2}$$

where $0 \le k_1 \le N_1 - 1$ and $0 \le k_2 \le N_2 - 1$

Cooley-Tukey Mappings for FFTs with Factorizable Lengths

- The effect of the index mapping is that $x[n]$ is represented as a 2–D array with n_1 specifying the rows and n_2 specifying the columns
- Inner parentheses of the last equation is the set of the N_1–point DFTs for the N_2 columns

$$G[k_1, n_2] = \sum_{n_1=0}^{N_1-1} x[N_2 n_1 + n_2] W_{N_1}^{n_1 k_1}, \quad \begin{cases} 0 \le k_1 \le N_1 - 1 \\ 0 \le n_2 \le N_2 - 1 \end{cases}$$

Cooley-Tukey Mappings for FFTs with Factorizable Lengths

- Next, these column DFTs are multiplied by the twiddle factors $W_N^{n_2 k_1}$ yielding

$$\tilde{G}[k_1, n_2] = W_N^{n_2 k_1} G[k_1, n_2], \quad \begin{cases} 0 \le k_1 \le N_1 - 1 \\ 0 \le n_2 \le N_2 - 1 \end{cases}$$

- Finally, the outer sum are the N_2–point DFTs of the N_1 rows of the array $\tilde{G}[k_1, n_2]$

$$X[k_1 + N_1 k_2] = \sum_{n_2=0}^{N_2-1} \tilde{G}[k_1, n_2] W_{N_2}^{n_2 k_2}, \quad \begin{cases} 0 \le k_1 \le N_1 - 1 \\ 0 \le k_2 \le N_2 - 1 \end{cases}$$

Cooley-Tukey Mappings for FFTs with Factorizable Lengths

- Example: $N = 8$ with $N_1 = 2$ and $N_2 = 4$

It follows for the DFT

$$X[k_1 + 2k_2] = \sum_{n_2=0}^{N_2-1} \left[\left(\sum_{n_1=0}^{N_1-1} x[N_2 n_1 + n_2] W_{N_1}^{k_1 n_1} \right) W_N^{k_1 n_2} \right] W_{N_2}^{k_2 n_2}$$

$$= \sum_{n_2=0}^{3} \left[\left(\sum_{n_1=0}^{1} x[4n_1 + n_2] W_2^{k_1 n_1} \right) W_8^{k_1 n_2} \right] W_4^{k_2 n_2}$$

where $0 \le k_1 \le 1$ and $0 \le k_2 \le 3$

Cooley-Tukey Mappings for FFTs with Factorizable Lengths

- The 2–D array representation of $x[n]$ is

n_1 \ n_2	0	1	2	3
0	x[0]	x[1]	x[2]	x[3]
1	x[4]	x[5]	x[6]	x[7]

- The column DFTs are 2–point DFTs given by

$$G[k_1,n_2]=x[n_2]+(-1)^{k_1}x[4+n_2],\quad \begin{cases}0\le k_1\le 1\\ 0\le n_2\le 3\end{cases}$$

- Thus, these DFTs require no multiplications

Cooley-Tukey Mappings for FFTs with Factorizable Lengths

- The 2–D array of the 4 column DFTs is

k_1 \ n_2	0	1	2	3
0	$G[0,0]$	$G[0,1]$	$G[0,2]$	$G[0,3]$
1	$G[1,0]$	$G[1,1]$	$G[1,2]$	$G[1,3]$

- 2–D array becomes $\tilde{G}[k_1,n_2]=W_8^{n_2k_1}G[k_1,n_2]$

k_1 \ n_2	0	1	2	3
0	$\tilde{G}[0,0]$	$\tilde{G}[0,1]$	$\tilde{G}[0,2]$	$\tilde{G}[0,3]$
1	$\tilde{G}[1,0]$	$\tilde{G}[1,1]$	$\tilde{G}[1,2]$	$\tilde{G}[1,3]$

Cooley-Tukey Mappings for FFTs with Factorizable Lengths

- Finally, the 4–point DFTs of the 2 rows are computed by

$$X[k_1+2k_2]=\sum_{n_2=0}^{3}\tilde{G}[k_1,n_2]W_4^{n_2k_2},\quad \begin{cases}0\le k_1\le 1\\ 0\le k_2\le 3\end{cases}$$

- The 2–D output array of the DFT is given by

k_1 \ k_2	0	1	2	3
0	X[0]	X[2]	X[4]	X[6]
1	X[1]	X[3]	X[5]	X[7]

Cooley-Tukey Mappings for FFTs with Factorizable Lengths

- The above mappings lead for $N_1=2$ and $N_2=4$ to the radix-2 DIF FFT, and in the case $N_1=4$ and $N_2=2$ to the radix-2 DIT FFT
- DIT-FFTs and DIF-FFTs can be obtained from the other by forming the transpose operation of the flow graphs (the matrix decompositions)
- We also can use a column-wise mapping for $x[n]$ and a row-wise mapping for $X[k]$ to derive FFTs

1.7 Prime Factor FFT Algorithms

- The twiddle factors in the FFTs with Cooley-Tukey mapping between the column- and row-DFTs can be eliminated if the factors in $N = N_1 N_2$ are relatively prime (coprime)
- To this end, we define the index mappings as

$$n = <An_1 + Bn_2>_N \ , \begin{cases} 0 \le n_1 \le N_1 - 1 \\ 0 \le n_2 \le N_2 - 1 \end{cases}$$

$$k = <Ck_1 + Dk_2>_N \ , \begin{cases} 0 \le k_1 \le N_1 - 1 \\ 0 \le k_2 \le N_2 - 1 \end{cases}$$

Prime Factor FFT Algorithms

- If $A = N_2, B = C = 1$ and $D = N_1$, the index mapping leads to the DIT FFT algorithm
- If $A = D = 1, B = N_1$ and $C = N_2$, the index mapping leads to the DIF FFT algorithm
- Note that in the cases when the factors in $N = N_1 N_2$ are not coprime, the stated twiddle factors cannot be avoided at all
- Mappings that eliminate the twiddle factors can be found using number theoretic concepts

Prime Factor FFT Algorithms

- If N_1 and N_2 are coprime, i.e. $\gcd(N_1, N_2) = 1$, it is possible to find index mappings to avoid the complex multiplications between the column- and row-DFTs
- Appropriate constants A, B, C and D for the two index mappings must satisfy the relation

$$W_N^{(An_1+Bn_2)(Ck_1+Dk_2)} = W_{N_1}^{k_1 n_1} W_{N_2}^{k_2 n_2}$$

- From the above relation and from

$$W_N^{nk} = W_N^{ACn_1k_1} W_N^{ADn_1k_2} W_N^{BCn_2k_1} W_N^{BDn_2k_2}$$

Prime Factor FFT Algorithms

follow the four congruences

$$\langle AC \rangle_N = N_2 \ , \langle BD \rangle_N = N_1$$

$$\langle AD \rangle_N = 0 \ , \langle BC \rangle_N = 0$$

- We choose $A = N_2$ and $B = N_1$ to satisfy the congruences $\langle AC \rangle_N = N_2$ and $\langle BD \rangle_N = N_1$
- The resulting index mapping for n (Good's mapping) is then given by

$$n = < N_2 n_1 + N_1 n_2 >_N \ , \begin{cases} 0 \le n_1 \le N_1 - 1 \\ 0 \le n_2 \le N_2 - 1 \end{cases}$$

Prime Factor FFT Algorithms

- C and D used in the index mapping of $X[k]$ can be found using the Chinese Remainder Theorem as follows

$$C = N_2\langle N_2^{-1}\rangle_{N_1}, D = N_1\langle N_1^{-1}\rangle_{N_2}$$

where $\langle N_1^{-1}\rangle_{N_2}$ denotes the multiplicative inverse of N_1 modulo N_2, and $\langle N_2^{-1}\rangle_{N_1}$ denotes the multiplicative inverse of N_2 modulo N_1

- If $\langle N_1^{-1}\rangle_{N_2} = \alpha$, it follows $\langle N_1\alpha\rangle_{N_2} = 1$, or $N_1\alpha = N_2\beta + 1$, where β is any integer

Prime Factor FFT Algorithms

- Likewise, if $\langle N_2^{-1}\rangle_{N_1} = \gamma$, then $N_2\gamma = N_1\delta + 1$, where δ is any integer
- Using the above, for $\langle AC\rangle_N$ we arrive at

$$\langle AC\rangle_N = \langle N_2 \cdot (N_2\langle N_2^{-1}\rangle_{N_1})\rangle_N$$

$$= \langle N_2(N_1\delta + 1)\rangle_N = \langle N_2N_1\delta + N_2\rangle_N = N_2$$

- Similarly, we have

$$\langle BD\rangle_N = \langle N_1 \cdot (N_1\langle N_1^{-1}\rangle_{N_2})\rangle_N$$

$$= \langle N_1(N_2\beta + 1)\rangle_N = \langle N_1N_2\beta + N_1\rangle_N = N_1$$

Prime Factor FFT Algorithms

- We observe that

$$\langle AD\rangle_N = \langle N_2 \cdot (N_1\langle N_1^{-1}\rangle_{N_2})\rangle_N = \langle N\alpha\rangle_N = 0$$

$$\langle BC\rangle_N = \langle N_1 \cdot (N_2\langle N_2^{-1}\rangle_{N_1})\rangle_N = \langle N\gamma\rangle_N = 0$$

- Thus, the DFT can be expressed as

$$X[k] = X[\langle Ck_1 + Dk_2\rangle_N]$$

$$= \sum_{n_2=0}^{N_2-1}\sum_{n_1=0}^{N_1-1} x[\langle An_1 + Bn_2\rangle_N]W_N^{N_2n_1k_1}W_N^{N_1n_2k_2}$$

Prime Factor FFT Algorithms

- Finally, we express $X[\langle Ck_1 + Dk_2\rangle_N]$ in the form

$$X[\langle Ck_1 + Dk_2\rangle_N] = \sum_{n_2=0}^{N_2-1} G[k_1, n_2]\, W_{N_2}^{n_2k_2}$$

where

$$G[k_1, n_2] = \sum_{n_1=0}^{N_1-1} x[\langle An_1 + Bn_2\rangle_N]W_{N_1}^{n_1k_1}$$

$$0 \le k_1 \le N_1 - 1, 0 \le k_2 \le N_2 - 1$$

are the N_1–point DFTs for the N_2 columns

Prime Factor FFT Algorithms

- Example: Let $N = 12, N_1 = 4$ and $N_2 = 3$ Then, $A = 3, B = 4, C = 3\langle 3^{-1}\rangle_4 = 3\times 3 = 9$ and $D = 4\langle 4^{-1}\rangle_3 = 4\times 1 = 4$ result in the index mappings

$$n = \langle 3n_1 + 4n_2\rangle_{12}, \begin{cases} 0 \le n_1 \le 3 \\ 0 \le n_2 \le 2 \end{cases}$$

$$k = \langle 9k_1 + 4k_2\rangle_{12}, \begin{cases} 0 \le k_1 \le 3 \\ 0 \le k_2 \le 2 \end{cases}$$

Prime Factor FFT Algorithms

- The 2–D representation of $x[n]$ is given by

n_1 \ n_2	0	1	2
0	$x[0]$	$x[4]$	$x[8]$
1	$x[3]$	$x[7]$	$x[11]$
2	$x[6]$	$x[10]$	$x[2]$
3	$x[9]$	$x[1]$	$x[5]$

- The 4–point DFTs of the columns are

k_1 \ n_2	0	1	2
0	$G[0,0]$	$G[0,1]$	$G[0,2]$
1	$G[1,0]$	$G[1,1]$	$G[1,2]$
2	$G[2,0]$	$G[2,1]$	$G[2,2]$
3	$G[3,0]$	$G[3,1]$	$G[3,2]$

Prime Factor FFT Algorithms

- The final 2–D DFT array for $X[k]$ is given by

k_1 \ k_2	0	1	2
0	$X[0]$	$X[4]$	$X[8]$
1	$X[9]$	$X[1]$	$X[5]$
2	$X[6]$	$X[10]$	$X[2]$
3	$X[3]$	$X[7]$	$X[11]$

- The 4–point DFTs require no multiplications, whereas all 3–point DFTs require 4 complex multiplications
- Totally, 16 real multiplications are required

1.8 IDFT and MATLAB Computation

- FFT algorithms can also be applied to calculate the **Inverse DFT** (IDFT)
- Consider a length-N sequence $x[n]$ with an N–point DFT $X[k]$
- Recall the IDFT computation

$$x[n] = \frac{1}{N}\sum_{k=0}^{N-1} X[k]W_N^{-nk}$$

IDFT Computation

- Multiplying both sides by N and taking the complex conjugate we arrive at

$$Nx^*[n] = \sum_{k=0}^{N-1} X^*[k]\, W_N^{nk}$$

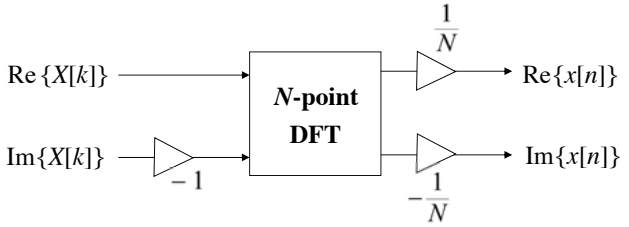
- The right-hand side of the above expression can be recognized as the N–point DFT of a sequence $X^*[k]$

IDFT Computation

- Then $x[n]$ is obtained as

$$x[n] = \frac{1}{N}\left\{\sum_{k=0}^{N-1} X^*[k] W_N^{nk}\right\}^*$$

- The IDFT computation is shown below



DFT and IDFT Computation in MATLAB

- The MATLAB functions **`fft`**, **`fft2`**, **`fftn`**, **`ifft`**, **`ifft2`**, and **`ifftn`** are based on a free library called FFTW (Fastest Fourier Transform in the West)
- To compute a DFT of length $N = N_1 N_2$, the FFTW library decomposes the problem using the Cooley-Tukey mapping, which first computes N_1 DFTs of size N_2, and then it computes N_2 DFTs of size N_1

DFT and IDFT Computation in MATLAB

- The decomposition is applied recursively to both the N_1–point and N_2–point DFTs until the problem can be solved using one of several fixed-size algorithms (codelets)
- These codelets in turn use several algorithms in combination, including a variation of Cooley-Tukey, prime factor, and split-radix
- The particular factorization of N is chosen heuristically

2. Spectral Analysis

2.1 Introduction

- Spectral analysis considers the problem of determining the spectral content of a signal
- Spectral analysis finds applications in many fields, e.g. in speech analysis, in radar and sonar systems, in medicine (ECG, EEG), or in seismology
- We begin our study of frequency analysis with the DFT analysis of sinusoidal signals

Introduction

- Next, we will discuss the Short-time Fourier transform to analyze signals with time-varying frequencies
- Because many real-world signals can be characterized as being random, we will then be concerned with estimating the spectral characteristics of random signals
- For practical reasons, we can only analyze finite-length signals

2.2 DFT Analysis of Sinusoidal Signals

- In this section, we focus on the DFT analysis of sinusoidal signals
- We assume that the parameters such as amplitude, frequency, and phase, do not change with time (stationary signals)
- For such a signal $g[n]$, the spectral analysis can be carried out by computing the DTFT

$$G(e^{j\omega}) = \sum_{n=-\infty}^{\infty} g[n]e^{-j\omega n}$$

DFT Analysis of Sinusoidal Signals

- The infinite-length sequence $g[n]$ is first windowed with a length-R window $w[n]$ to convert it into a length-R sequence $\gamma[n]$
- DTFT $\Gamma(e^{j\omega})$ of $\gamma[n]$ is assumed to provide a reasonable estimate of $G(e^{j\omega})$
- $\Gamma(e^{j\omega})$ is evaluated by a N-point DFT of $\gamma[n]$, where the condition $N \geq R$ must be met to avoid time-domain aliasing

DFT Analysis of Sinusoidal Signals

- Next, we analyze the evaluation of the DTFT via the DFT given by

$$\Gamma[k] = \Gamma(e^{j\omega})\Big|_{\omega=2\pi k/N}, \quad 0 \leq k \leq N-1$$

- The normalized digital angular frequency ω_k corresponding to the DFT bin number k (DFT frequency sample) is given by

$$\omega_k = \frac{2\pi k}{N}$$

DFT Analysis of Sinusoidal Signals

- Thus, the continuous-time angular frequency Ω_k corresponding to the DFT bin number k is

$$\Omega_k = 2\pi F_k = \frac{\omega_k}{T} = \frac{2\pi k}{NT} = 2\pi k \frac{F_s}{N}$$

- Note that $\Delta F = F_s/N$ represents the value of the frequency resolution, i.e. the smallest difference in frequencies that can be detected

DFT Analysis of Sinusoidal Signals

- Consider $g[n] = \cos(\omega_o n + \phi), \quad -\infty < n < \infty$
- It can be expressed as

$$g[n] = \frac{1}{2}\left(e^{j(\omega_o n+\phi)} + e^{-j(\omega_o n+\phi)}\right)$$

- Its DTFT is given by

$$G(e^{j\omega}) = \pi \sum_{\ell=-\infty}^{\infty} e^{j\phi}\delta(\omega - \omega_o + 2\pi\ell) + \pi \sum_{\ell=-\infty}^{\infty} e^{-j\phi}\delta(\omega + \omega_o + 2\pi\ell)$$

DFT Analysis of Sinusoidal Signals

- $G(e^{j\omega})$ is periodic in ω with a period 2π containing two impulses in each period
- In the range $-\pi \le \omega \le \pi$, there is an impulse at $\omega = \omega_o$ of amplitude $\pi e^{j\phi}$ and an impulse at $\omega = -\omega_o$ of amplitude $\pi e^{-j\phi}$
- To analyze $g[n]$ using DFT, we employ a finite-length version of the sequence given by $\gamma[n] = \cos(\omega_o n + \phi),\ 0 \le n \le R-1$

DFT Analysis of Sinusoidal Signals

- Example: Determine the 32-point DFT of a length-32 sequence $\gamma[n]$ obtained by a sinusoidal signal of frequency $F = 10$ Hz sampled at a sampling rate of $F_s = 64$ Hz
- Since F_s is much larger than the Nyquist frequency, there is no aliasing due to sampling and its DTFT contains only two impulses at $F = +10$ Hz and $F = -10$ Hz

DFT Analysis of Sinusoidal Signals

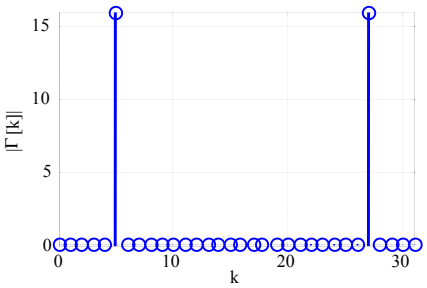
- Its 32-point DFT is obtained by sampling $\Gamma(e^{j\omega})$ at $\omega = \omega_k = 2\pi k/32,\ 0 \le k \le 31$
- The impulse at $F = +10$ Hz appears as $\Gamma[5]$ at the DFT frequency bin location

$$k = \frac{F_k \times N}{F_S} = \frac{10 \times 32}{64} = 5$$

and the impulse at $F = -10$ Hz appears as $\Gamma[27]$ at bin location $k = 32 - 5 = 27$

DFT Analysis of Sinusoidal Signals

- The magnitude plot of the 32-point DFT $\Gamma[k]$ is shown below



DFT Analysis of Sinusoidal Signals

- For an N-point DFT, the first half of the DFT samples for $k = 0$ to $k = (N/2) - 1$ corresponds to positive frequencies from $F = 0$ to $F = F_s/2$
- Furthermore, the second half of the DFT samples for $k = N/2$ to $k = N - 1$ corresponds to negative frequencies from $F = -F_s/2$ to $F = 0$
- Next, we consider a sinusoid not coinciding with the DFT bins

DFT Analysis of Sinusoidal Signals

- Example: Determine the 32-point DFT of a length-32 sequence $\gamma[n]$ obtained by a sinusoidal signal of frequency $F = 11$ Hz sampled at a sampling rate of $F_s = 64$ Hz
- Since $\gamma[n]$ is a pure sinusoid, its DTFT contains two impulses at $F = +11$ Hz and $F = -11$ Hz and is zero elsewhere

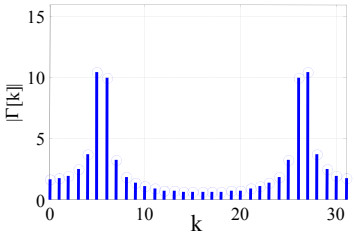
DFT Analysis of Sinusoidal Signals

- Since $\dfrac{F \times N}{F_s} = \dfrac{11\ Hz \times 32}{64\ Hz} = 5.5$

  the impulse at $F = 11$ Hz of the DTFT is between the DFT bin locations $k = 5$ and $k = 6$
- Likewise, the impulse at $F = -11$ Hz of the DTFT is between the DFT bin locations $k = 26$ and $k = 27$

DFT Analysis of Sinusoidal Signals

- Magnitude plot of the 32-point DFT $\Gamma[k]$



- DFT contains nonzero bins all over, strong bins at $k = 5$ and $k = 6$, and at $k = 26$ and $k = 27$

DFT Analysis of Sinusoidal Signals

- The wrong frequency components at all bins are due to **spectral spreading** and **leakage** caused by the main lobe and the sidelobes of the window spectrum
- To understand this effects, recall that the N-point DFT $\Gamma[k]$ of a length-R sequence $\gamma[n]$ is given by the samples of its DTFT $\Gamma(e^{j\omega})$

$$\Gamma[k] = \Gamma(e^{j\omega_k})\Big|_{\omega_k = 2\pi k/N},\ 0 \le k \le N-1$$

DFT Analysis of Sinusoidal Signals

- Recall that the finite-length sequence

$$\gamma[n] = \cos(\omega_o n + \phi),\ 0 \le n \le R-1$$

has been obtained by windowing the original infinite-length sequence $g[n]$ with a length-R rectangular window $w[n]$ given by

$$w[n] = \begin{cases} 1, & 0 \le n \le R-1 \\ 0, & \text{otherwise} \end{cases}$$

DFT Analysis of Sinusoidal Signals

- The DTFT $\Gamma(e^{j\omega})$ of $\gamma[n]$ is given by

$$\Gamma(e^{j\omega}) = \frac{1}{2\pi}\int_{-\pi}^{\pi} G(e^{j\varphi})\Psi(e^{j(\omega-\varphi)})\,d\varphi$$

where $G(e^{j\omega})$ is the DTFT of $g[n]$

$$G(e^{j\omega}) = \pi\sum_{\ell=-\infty}^{\infty} e^{j\phi}\delta(\omega-\omega_o+2\pi\ell) + \pi\sum_{\ell=-\infty}^{\infty} e^{-j\phi}\delta(\omega+\omega_o+2\pi\ell)$$

and $\Psi(e^{j\omega})$ is the DTFT of $w[n]$

$$\Psi(e^{j\omega}) = e^{-j\omega(N-1)/2}\cdot\frac{\sin(\omega N/2)}{\sin(\omega/2)}$$

DFT Analysis of Sinusoidal Signals

- Hence $\Gamma(e^{j\omega})$ given by

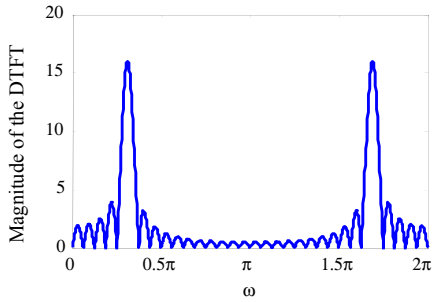
$$\Gamma(e^{j\omega}) = \frac{1}{2}e^{j\phi}\Psi(e^{j(\omega-\omega_o)}) + \frac{1}{2}e^{-j\phi}\Psi(e^{j(\omega+\omega_o)})$$

is a sum of the frequency shifted and amplitude scaled DTFT $\Psi(e^{j\omega})$ with the amount of shifts given by $\pm\omega_o$

- Example: The normalized angular frequency of the length-32 sinusoid of frequency 11 Hz sampled at 64 Hz is $\omega_0 = 11/64\times 2\pi = 0.344\pi$

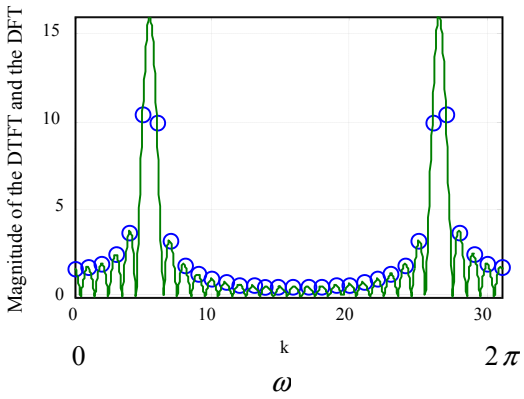
DFT Analysis of Sinusoidal Signals

- Magnitude plot of the DTFT $\Gamma(e^{j\omega})$ with two peaks at 0.344π and $2\pi - 0.344\pi = 1.656\pi$



DFT Analysis of Sinusoidal Signals

- Plot of $|\Gamma(e^{j\omega})|$ and the 32-point DFT $|\Gamma[k]|$



There two peaks at bin locations $k = 5$ and $k = 6$, and two peaks at bin locations $k = 26$ and $k = 27$

DFT Analysis of Sinusoidal Signals

- Windowing causes spectral distortions due to the main lobe and the sidelobes of the window
- A major effect of the main lobe is to spread the spectrum resulting in the loss of resolution
- A major effect of the sidelobes is to transfer signal power from frequency bands with large amounts of power into bands with less power creating additional non-zero bins

DFT Analysis of Sinusoidal Signals

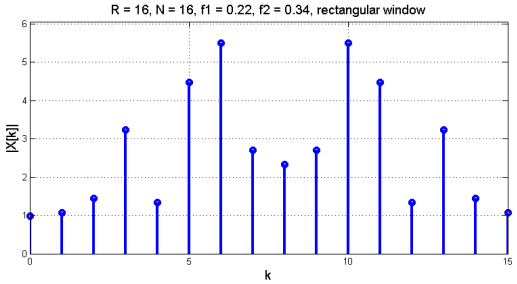
- Problem gets more complicated if the signal being analyzed has more than one sinusoid
- Example: Consider the sum of two sinusoids

$$x[n] = \frac{1}{2}\sin(2\pi f_1 n) + \sin(2\pi f_2 n),\ 0 \le n \le R-1$$

i.e. the given two sinusoids are windowed by a rectangular window of length R

DFT Analysis of Sinusoidal Signals

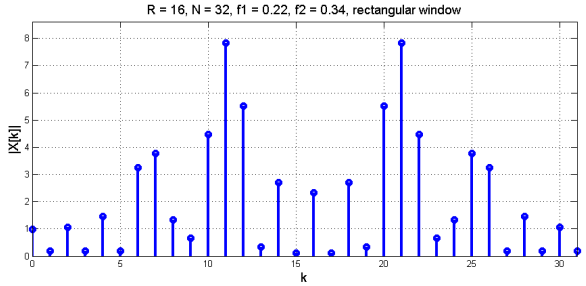
- The plot of $|X[k]|$ for $f_1 = 0.22, f_2 = 0.34$ and for $R = N = 16$ is shown below



- The bins do not represent the exact frequencies

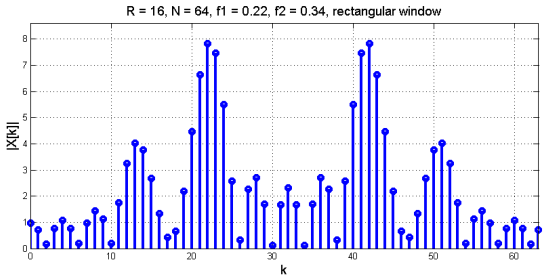
DFT Analysis of Sinusoidal Signals

- An increase in the DFT length to $N = 32$ leads to some nonzero bins around $k = 7$ and $k = 11$



DFT Analysis of Sinusoidal Signals

- Increasing the DFT length to $N = 64$ leads to more nonzero bins around $k = 13$ and $k = 22$



DFT Analysis of Sinusoidal Signals

- As can be seen, an increase in the DFT length N provides more peaks around the bins near to the frequencies of the given two sinusoids
- However, none of the bins represent a right frequency value because of the low frequency resolution due to $\Delta F = F_s / N$
- In cases with many peaks, it is often not clear whether additional sinusoids are present or not

DFT Analysis of Sinusoidal Signals

- Example: $x[n] = \frac{1}{2}\sin(2\pi f_1 n) + \sin(2\pi f_2 n)$

windowed by a length-R rectangular window

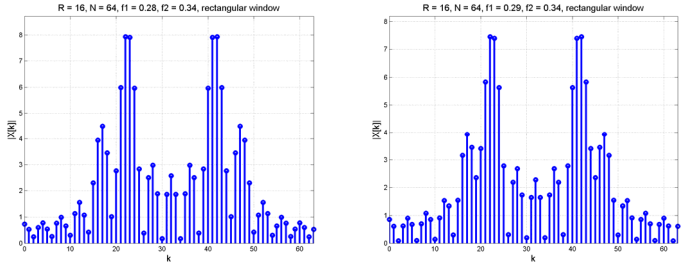
We choose $R = 16$, and a DFT of length $N = 64$

$f_2 = 0.34$ is fixed

$f_1 = 0.28$, 0.29, 0.30 or 0.31

DFT Analysis of Sinusoidal Signals

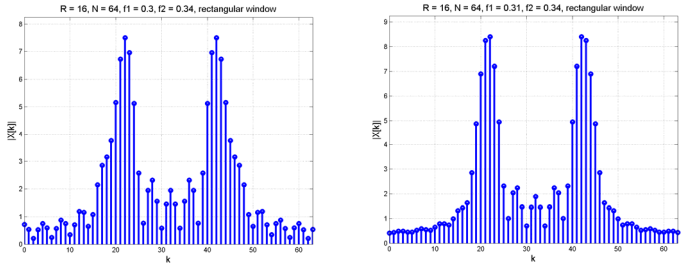
$f_1 = 0.28$ $f_1 = 0.29$



- Here, two distinct groups of peaks are visible

DFT Analysis of Sinusoidal Signals

$f_1 = 0.30$ $f_1 = 0.31$

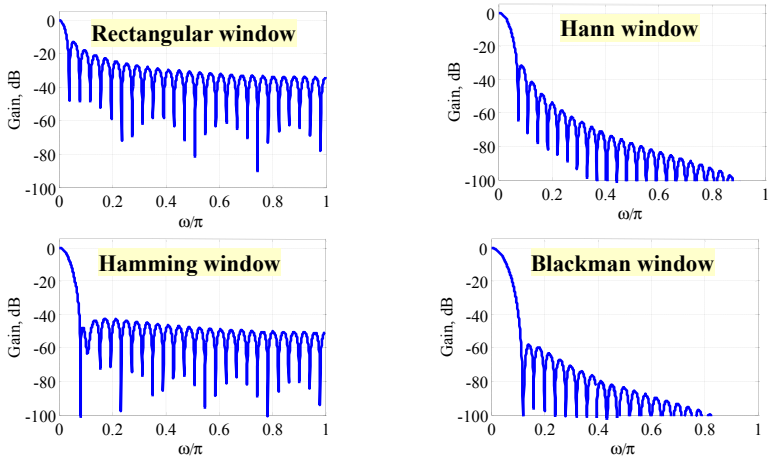


- Here, the distinct peaks of the two sinusoids are not clearly visible

DFT Analysis of Sinusoidal Signals

- As the frequency difference between the two sinusoids becomes smaller, the two main lobes get closer and can eventually overlap
- If there is a significant overlap, it will be difficult or impossible to resolve the two peaks
- To improve the frequency analysis in such cases it is necessary to consider the windows
- Next figure shows four typical fixed windows

DFT Analysis of Sinusoidal Signals



DFT Analysis of Sinusoidal Signals

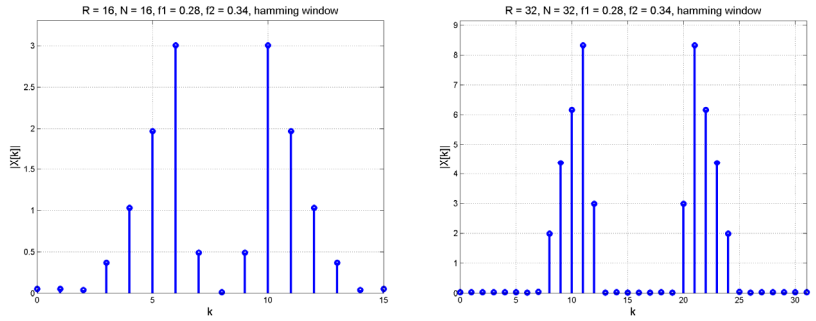
- The main lobe width Δ_{ML} and the relative sidelobe level $A_{s\ell}$ of the windows are given by
  - Rectangular window: $\Delta_{ML} = 4\pi/R$, $A_{s\ell} = 13$ dB
  - Hann window: $\Delta_{ML} = 8\pi/R$, $A_{s\ell} = 31$ dB
  - Hamming window: $\Delta_{ML} = 8\pi/R$, $A_{s\ell} = 54$ dB
  - Blackman window: $\Delta_{ML} = 12\pi/R$, $A_{s\ell} = 58$ dB
- A rectangular window has the smallest main lobe width, but the largest sidelobe level

DFT Analysis of Sinusoidal Signals

- Example: $x[n] = \frac{1}{2}\sin(2\pi f_1 n) + \sin(2\pi f_2 n)$ with $f_1 = 0.28$ and $f_2 = 0.34$ windowed by a Hamming window
- The resulting plots for $R = N = 16$ and for $R = N = 32$ are shown next
- As can been seen, the two sinusoids cannot be resolved in both cases

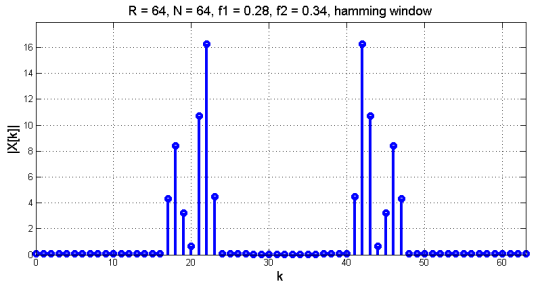
DFT Analysis of Sinusoidal Signals

$R = N = 16$ $R = N = 32$



DFT Analysis of Sinusoidal Signals

- For $R = N = 64$, two distinct peaks are clearly visible



DFT Analysis of Sinusoidal Signals

- In addition to the windows with fixed relative sidelobe level there are windows with adjustable relative sidelobe level, e.g. the Chebyshev window or the Kaiser window
- As in the case of the fixed windows, the main lobe width of the Chebyshev or the Kaiser window can be reduced by increasing the window length

DFT Analysis of Sinusoidal Signals

Summary

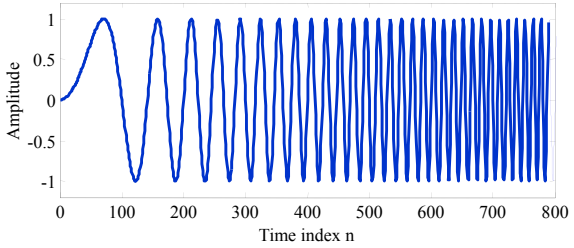
- The performance of the DFT-based spectral analysis of sinusoidal signals depends on the three factors:
  1. Window length R
  2. DFT length N (Note: $\Delta F = F_s / N$)
  3. Type of window

2.3 Short-Time Fourier Transform

- DFT spectrum analysis can be employed if the frequencies, amplitudes and phases of each sinusoidal component of the analyzed signal do not change with time within the analysis window
- The spectral characteristics of many signals of practical interest are instead time-varying
- Some examples of such signals are speech, radar and sonar signals

Short-Time Fourier Transform

- Example: Chirp signal $x[n] = A\cos(\omega_o n^2)$ as a time-varying signal for $\omega_o = 10\pi \times 10^{-5}$



- The instantaneous frequency of $x[n]$ is $2\omega_o n$

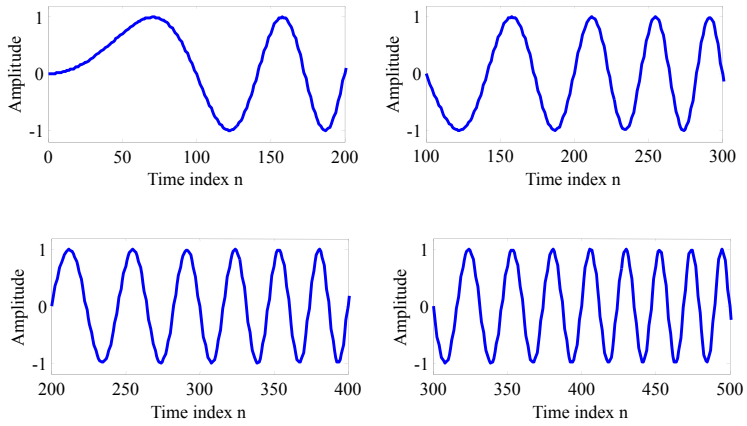
Short-Time Fourier Transform

- If the spectral content changes significantly inside the long analysis window, a DFT analysis will provide erroneous results
- A practical solution to this problem is to break the signal into a set of subsequences
- The length of each subsequence should be sufficiently short to assure that the spectral characteristics do not vary

Short-Time Fourier Transform

- To represent a signal $x[n]$ in terms of a set of short-length subsequences, $x[n]$ can be moved through a window
- The frequency-domain description of the long sequence is then given by a set of short-length DFTs (time-dependent DFTs)
- Next figure shows four segments of the chirp signal as seen through a length-200 rectangular window

Short-Time Fourier Transform



Short-Time Fourier Transform

- The **Short-time Fourier transform** (STFT) of a signal $x[n]$ is defined by

$$X_{STFT}[n, e^{j\omega}) = \sum_{m=-\infty}^{\infty} x[n+m]\,w[m]\,e^{-j\omega m}$$

where $w[m]$ is a suitably chosen window that is stationary with respect to time and $x[n]$ is moved through the window $w[m]$

- Note that if $w[m] = 1$ for all values of m, the STFT reduces to the DTFT of $x[n]$

Short-Time Fourier Transform

- $X_{STFT}[n, e^{j\omega})$ is a function of two variables:
  1. integer variable time index n
  2. continuous frequency variable ω

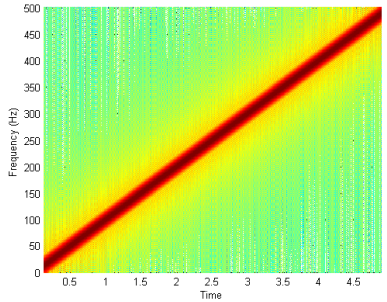
  Note: **[** is used for n and **)** is used for ω
- STFT is a periodic function of ω with a period 2π
- Magnitude of STFT is usually referred to as the **spectrogram**

Short-Time Fourier Transform

- Display of spectrogram requires normally three dimensions
- Often, STFT magnitude is plotted in two dimensions with magnitudes represented by the grey level or in color
- Plot of STFT magnitude of the chirp signal $x[n] = A\cos(\omega_o n^2)$ with $\omega_o = 10\pi \times 10^{-5}$ for a length of 20,000 samples and a Hamming window of length 200 is shown next

Short-Time Fourier Transform

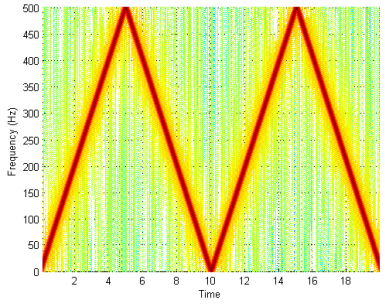
- Spectrogram of the chirp signal with duration τ = 5 s and sampling frequency F_s = 1,000 Hz



- Instantaneous frequency grows linearly

Short-Time Fourier Transform

- Next, we increase the duration until $\tau = 20$ s



- Since the highest frequency is $F_s/2$, it appears a triangular shape due to aliasing

Sampling in the Time and Frequency Dimensions

- In practice, the STFT is computed at a finite set of N values and windows of length R with nonzero samples in the range $0 \le m \le R-1$
- The STFT is accurately represented by its frequency samples as long as the number N is not smaller than the window length R
- Then the subsequence of $x[n]$ inside the window can be fully recovered from the frequency samples of the STFT

Sampling in the Time and Frequency Dimensions

- Sampling $X_{STFT}[n, e^{j\omega})$ at N equally spaced frequencies $\omega_k = 2\pi k / N$ is given by

$$X[n,k] = X_{\text{STFT}}[n, e^{j\omega})\Big|_{\omega=2\pi k/N}$$

$$= \sum_{m=0}^{R-1} x[n+m]w[m]e^{-j2\pi km/N}, 0 \le k \le N-1$$

- $X[n, k]$ is simply the N-point DFT of the windowed subsequence given by $x[n+m]w[m]$

Sampling in the Time and Frequency Dimensions

- $X[n, k]$ is a two-dimensional sequence that is periodic in k with a period N
- Applying the IDFT of $X[n,k]$ we obtain the subsequence $x[n+m]$ by

$$x[n+m] = \frac{1}{Nw[m]} \sum_{k=0}^{N-1} X[n,k]e^{j2\pi km/N}, 0 \le m \le R-1$$

- Thus, $x[n]$ can be fully recovered from $X[n, k]$ in the range from n to $(n + R - 1)$

Sampling in the Time and Frequency Dimensions

- Without sampling $x[n]$, the subsequences are shifted along the window one sample at a time
- Now, we define the STFT with a sampling period L in time as

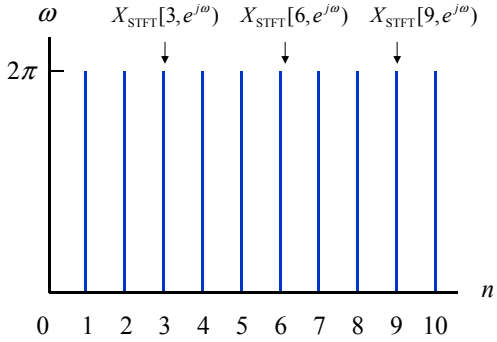
$$X_\ell[k] = X[\ell L, k] = \sum_{m=0}^{R-1} x[\ell L + m] w[m] e^{-j2\pi km/N}$$

where $-\infty < \ell < \infty$ and $0 \le k \le N-1$

- Thus, the location of the windowed blocks change in steps of L points in time

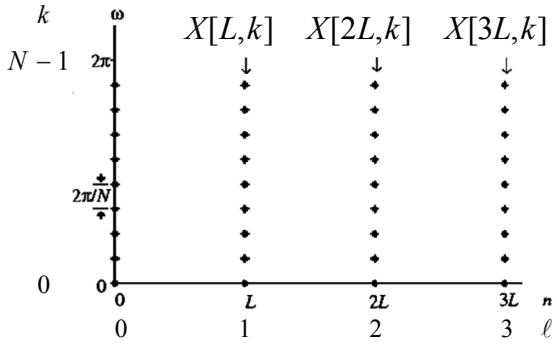
Sampling in the Time and Frequency Dimensions

- [n,ω)-plane corresponding to $X_{STFT}[n, e^{j\omega})$ without sampling



Sampling in the Time and Frequency Dimensions

- Grid of sampling points in the [n, ω)-plane for $N = 10$ and $L = 3$



Sampling in the Time and Frequency Dimensions

- Subsequences of $x[nL+m]$ are shifted with L samples at a time and windowed by $w[m]$
- The overlap between window blocks is $R - L$
- It is possible to uniquely reconstruct the original signal $x[n]$ from such a 2-D discrete representation provided $N \ge R \ge L$, where N is the DFT length, R is the window length and L is the sampling period in time

Frequency and Time Localization Trade-off

- For a reasonably good analysis of a time-varying signal, the window should have a very small relative sidelobe level
- The window length depends on the required frequency and time localization
- The uncertainty principle determines the trade-off between fine localization in the time and in the frequency domain

Frequency and Time Localization Trade-off

- Signals that are finely localized in time cannot finely localized in frequency
- Conversely, signals that are finely localized in frequency cannot be finely localized in time
- Conceptually, this trade-off can be illustrated using the following two transform pairs with Dirac functions

$$\delta(t) \leftrightarrow 1$$

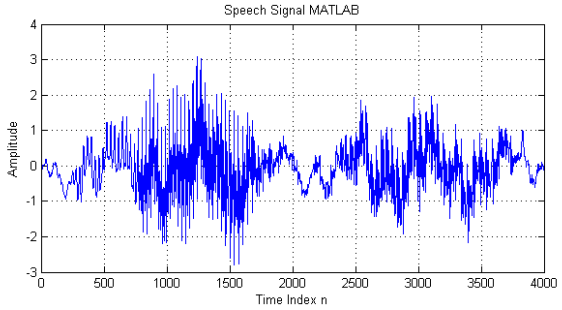
$$1 \leftrightarrow 2\pi\delta(\omega)$$

Frequency and Time Localization Trade-off

- Therefore, a decrease of the window length increases the time resolution
- An increase of the window length increases the frequency resolution
- Thus, a shorter window provides a wideband spectrogram, whereas a longer window results in a narrowband spectrogram

Frequency and Time Localization Trade-off

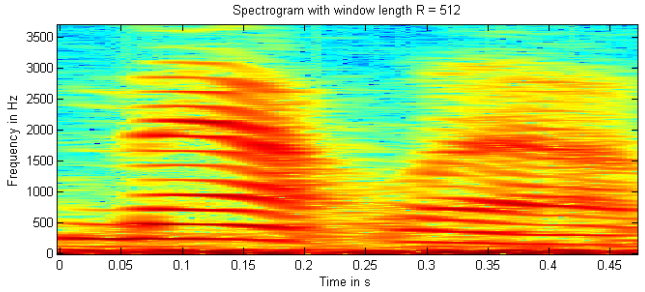
- Example: Speech signal MATLAB (mtlb.mat)



- The MATLAB function **spectrogram** can be used to compute and plot the STFT

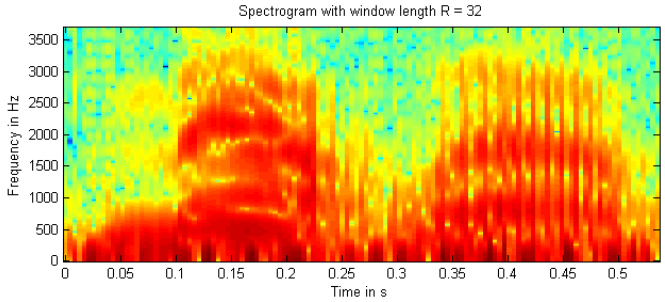
Frequency and Time Localization Trade-off

- The **narrowband spectrogram** with window length $R = 512$ is shown below



Frequency and Time Localization Trade-off

- The **wideband spectrogram** with window length $R = 32$ is shown below



2.4 Overview of Power Spectral Density Estimation

- Many signals are either partly or wholly random
- We assume a wide-sense stationary process, i.e. that the mean and the variance are constant and that the autocovariance depends only on the lag
- Random signals do not have finite energy, and therefore they do not possess DTFTs
- However, they usually have finite average power, and can thus be characterized by an average power spectral density (PSD)

Overview of PSD Estimation

- In practice, we evaluate a sample signal $x[n]$ in an interval $0 \le n \le L-1$, and we assume that the sample mean is zero
- Otherwise, we estimate the sample mean by

$$\hat{m}_x = \frac{1}{L}\sum_{n=0}^{L-1} x[n]$$

  and subtract $\hat{m}_x$ from $x[n]$
- Thus, the random signal can be described by its autocorrelation sequence (ACS)

Overview of PSD Estimation

- The PSD is given by the **Wiener-Khintchine Theorem**

$$\Phi_{xx}(e^{j\omega}) = \sum_{m=-\infty}^{+\infty} r_{xx}[m]\, e^{-jm\omega}$$

  i.e. it is the DTFT of the (true) ACS $r_{xx}[m]$
- Since the ACS measures the average power for a given lag, the PSD represents the distribution of the average power over the frequency axis

Overview of PSD Estimation

- Since the true ACS is not known a priori, we can only estimate the PSD
- Hereafter, the hat (^) over $\hat{\Phi}_{xx}(e^{j\omega})$ denotes a PSD estimate coming from a PSD estimator
- Three metrics are used to characterize the quality of a PSD estimator: **bias**, **variance**, and **mean squared error**
- Another criterion for a PSD estimator is its **frequency resolution**

Overview of PSD Estimation

- The **bias of the PSD estimate** is defined by

$$Bias\{\hat{\Phi}_{xx}(e^{j\omega})\} = E\left\{\hat{\Phi}_{xx}(e^{j\omega})\right\} - \Phi_{xx}(e^{j\omega})$$

  $E\{\cdot\}$ denotes the expectation operator
- If the bias is zero, the PSD estimate is said to be **unbiased**, otherwise **biased**
- An asymptotically unbiased estimator has zero bias for $L \to \infty$

Overview of PSD Estimation

- The **variance of the PSD estimate** is defined by

$$Var\left\{\hat{\Phi}_{xx}(e^{j\omega})\right\} = E\left\{\left[\hat{\Phi}_{xx}(e^{j\omega}) - E\{\hat{\Phi}_{xx}(e^{j\omega})\}\right]^2\right\}$$

- The variance of the PSD estimate is the spread of its values about its expected values
- A good PSD estimate should have the smallest possible variance

Overview of PSD Estimation

- The **Mean Squared Error** (MSE) **of the PSD estimate** is defined by

$$MSE\left\{\hat{\Phi}_{xx}(e^{j\omega})\right\} = E\left\{\left[\hat{\Phi}_{xx}(e^{j\omega}) - \Phi_{xx}(e^{j\omega})\right]^2\right\}$$

- By separately considering the bias and the variance of the MSE, we obtain

$$MSE\left\{\hat{\Phi}_{xx}(e^{j\omega})\right\} = Var\{\hat{\Phi}_{xx}(e^{j\omega})\} + Bias^2\{\hat{\Phi}_{xx}(e^{j\omega})\}$$

Overview of PSD Estimation

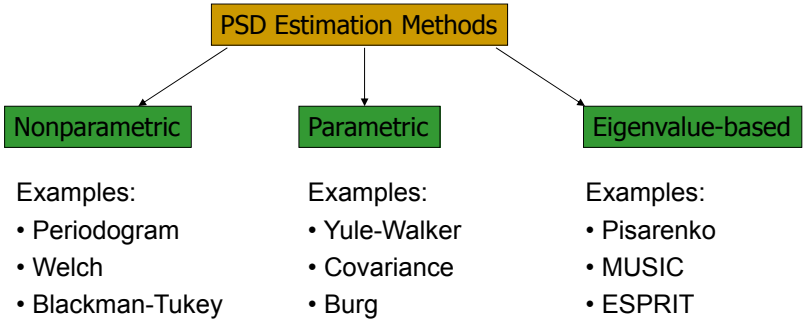
- Although, a random sequence exists for all time, often a finite portion of the random sequence with length L is available
- As L increases, an estimator should give a better estimate, and in the limiting case, the estimator should give the true value
- An estimator is said to be **consistent** if both its bias and its variance converge to zero as L tends to infinity

Overview of PSD Estimation

- A **sufficient condition for an estimator to be consistent** is that the MSE of the PSD estimate converges to zero as L tends to infinity
- However, there are trade-offs between the bias and the variance of an estimator
- In many cases, an estimator with low variance and some bias is more desirable than an unbiased estimator with higher variance

Overview of PSD Estimation

- Classification of PSD estimation methods



Overview of PSD Estimation

- Methods that rely on the direct use of the given random signal to estimate the PSD are called nonparametric methods
- Parametric methods rely on a model for the replication of the random signal to estimate the PSD from the model (e.g. AR) parameters
- Methods that rely on the eigenvalues of the underlying signal model are referred to as eigenvalue-based or subspace methods

2.5 Periodogram Analysis

- Assume a random signal $y[n]$ windowed by a length-L window $w[n], 0 \le n \le L-1$, resulting in the sequence $x[n] = w[n] \cdot y[n]$
- The DTFT of $x[n]$ is given by

$$X(e^{j\omega}) = \sum_{n=0}^{L-1} w[n] \cdot y[n]\, e^{-j\omega n}$$

- A PSD estimate is then obtained by

$$\hat{\Phi}_{xx}^{Per}(e^{j\omega}) = \frac{1}{L \cdot U} \left| X(e^{j\omega}) \right|^2$$

Periodogram Analysis

- $\hat{\Phi}_{xx}^{Per}(e^{j\omega})$ is called the periodogram if $w[n]$ is the rectangular window, otherwise it is called the modified periodogram
- Note that for each value of ω, the amplitude of the periodogram is a random variable
- U is a normalizing factor to compensate the power of $w[n]$ defined by

$$U = \frac{1}{L} \sum_{n=0}^{L-1} \left| w[n] \right|^2$$

Note: $U = 1$ for a rectangular window

Periodogram Analysis

- In practice, the periodogram is evaluated at a discrete set of equally spaced R frequencies, $\omega_k = 2\pi k / R,\ 0 \le k \le R-1$, by replacing the DTFT $X(e^{j\omega})$ with an R-point DFT $X[k]$ of the length-L sequence $x[n]$

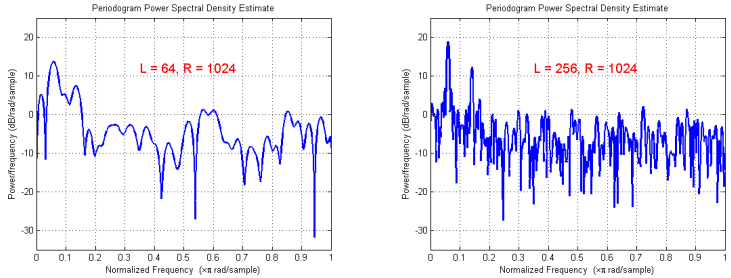
$$\hat{\Phi}_{xx}^{Per}[k] = \frac{1}{L \cdot U} \left| X[k] \right|^2$$

- Usually R is chosen to be greater than L to get a finer grid of samples of the periodogram

Periodogram Analysis

- A periodogram can be computed using the MATLAB function **`periodogram`**
- Example: Signal composed of two sinusoidal components of angular frequencies 0.06 π and 0.14 π, corrupted by a normally distributed random signal of zero mean and unity variance
- Periodograms are generated for $L = 64$ and $L = 256$ using a DFT of length $R = 1024$

Periodogram Analysis



- With increasing L the peaks at 0.06π and 0.14π get sharper

Periodogram Analysis

- The statistical properties of $\hat{\Phi}_{xx}^{Per}(e^{j\omega})$ can be explained by considering its bias and variance
- Taking the expectation of $\hat{\Phi}_{xx}^{Per}(e^{j\omega})$ using the ACS estimate gives the relation

$$E\{\hat{\Phi}_{xx}^{Per}(e^{j\omega})\} = \frac{1}{LU}\sum_{k=-(L-1)}^{L-1} E\{\hat{r}_{xx}[k]\}\, e^{-j\omega k}$$

$$= \frac{1}{LU}\sum_{k=-(L-1)}^{L-1} r_{ww}[k]\cdot r_{xx}[k]\cdot e^{-j\omega k}$$

where $E\{\hat{r}_{xx}[k]\} = r_{ww}[k]\cdot r_{xx}[k]$

Periodogram Analysis

- The ACS $r_{ww}[k]$ of the window $w[n]$ of length L is given by

$$r_{ww}[k] = \sum_{n=0}^{L-1} w[n]w[n+k]$$

- Because the ACS $r_{ww}[k]$ can be expressed as the convolution sum of $w[n]$ and $w[-n]$, the DTFT of $r_{ww}[k]$ is given by

$$\left|W(e^{j(\omega)})\right|^2$$

Periodogram Analysis

- $E\{\hat{\Phi}_{xx}^{Per}(e^{j\omega})\}$ can be expressed using the DTFTs of the true ACS $r_{xx}[k]$ and the ACS of $w[n]$

$$E\{\hat{\Phi}_{xx}^{Per}(e^{j\omega})\} = \frac{1}{2\pi LU}\int_{-\pi}^{\pi}\Phi_{xx}(e^{ju})\cdot\left|W(e^{j(\omega-u)})\right|^2\,du$$

- Since $E\{\hat{\Phi}_{xx}^{Per}(e^{j\omega})\} \neq \Phi_{xx}(e^{j\omega})$, the periodogram is a biased PSD estimator
- For $L\to\infty$, $W(e^{j\omega})\to\delta(\omega)$, i.e. the average periodogram is asymptotically unbiased

Periodogram Analysis

- For a rectangular window, the variance of the PSD estimate is (after long algebra) given by

$$\lim_{L\to\infty} Var\{\hat{\Phi}_{xx}^{Per}(e^{j\omega})\} = \begin{cases}\Phi_{xx}^2(e^{j\omega}), & 0<\omega<\pi\\ 2\cdot\Phi_{xx}^2(e^{j\omega}), & \omega=0,\pm\pi\end{cases}$$

- Independently of L, the standard deviation of the PSD estimate is as large as the true PSD
- Thus, the periodogram is not a consistent PSD estimator

Periodogram Analysis

- If we rewrite the periodogram ($U = 1$) as

$$\hat{\Phi}_{xx}^{Per}(e^{j\omega}) = \frac{1}{L}X(e^{j\omega})\cdot X^*(e^{j\omega}) =$$
$$\frac{1}{L}\sum_{m=0}^{L-1}\sum_{n=0}^{L-1}x^*[m]x[n]\,e^{-j\omega(m-k)} = \frac{1}{L}\sum_{\kappa=-(L-1)}^{L-1}\hat{r}_{xx}[\kappa]\,e^{-j\omega\kappa}$$

  we can see that $\hat{\Phi}_{xx}^{Per}(e^{j\omega})$ indeed relates to the ACS estimate $\hat{r}_{xx}[\kappa]$
- $\hat{r}_{xx}[\kappa]$ is the so-called biased ACS estimate, obtained from the length-L random signal

Periodogram Analysis

- The biased ACS estimate is given by

$$\hat{r}_{xx}[\kappa] = \begin{cases}\frac{1}{L}\sum_{k=0}^{L-1-\kappa}x^*[k]\,x[k+\kappa], & 0\le\kappa\le L-1\\ \frac{1}{L}\sum_{k=-\kappa}^{L-1}x^*[k]\,x[k+\kappa], & -(L-1)\le\kappa\le 0\end{cases}$$

- The sum of the periodogram including the biased ACS estimate and the double sum of the periodogram involving $X(e^{j\omega})\,X^*(e^{j\omega})$ add up the same elements in different order

Periodogram Analysis

- Note that when κ is close to L, only a few samples are involved in the computation of $\hat{r}_{xx}[\kappa]$ resulting in a poor ACS estimate
- Recall that the variance of $\hat{\Phi}_{xx}^{Per}(e^{j\omega})$ is of order $\Phi_{xx}^{2}(e^{j\omega})$ and does not decrease as L increases
- Therefore, periodograms have erratic and wild fluctuating forms
- Smoothing of periodograms can be achieved by averaging periodograms of segments of $x[n]$

Welch Method

- The **Welch method** subdivides the input into multiple segments, computes and averages the periodograms of these segments
- Assume an input of length N, P segments of L samples each, and an offset S between two successive segments
- The windowed k-th segment is given by

$$x^{(k)}[n] = x[n+kS]\cdot w[n]$$
$$0 \le n \le L-1,\ S \le L,\ 0 \le k \le P-1$$

Welch Method

- The periodogram of the k-th segment is given by

$$\hat{\Phi}_{xx}^{Per(k)}(e^{j\omega}) = \frac{1}{LU}\left|X^{(k)}(e^{j\omega})\right|^2$$
$$= \frac{1}{LU}\left|\sum_{n=0}^{L-1} x[n+kS]\cdot w[n]\cdot e^{-j\omega n}\right|^2$$

- The PSD Welch estimate is the average of all P periodograms, i.e.

$$\hat{\Phi}_{xx}^{Welch}(e^{j\omega}) = \frac{1}{P}\sum_{k=0}^{P-1}\hat{\Phi}_{xx}^{Per(k)}(e^{j\omega})$$

Welch Method

- We assume large values of L, so that the P periodograms are approximately independent and uniformly distributed random variables
- Then, the expected value of $\hat{\Phi}_{xx}^{Welch}(e^{j\omega})$ yields

$$E\{\hat{\Phi}_{xx}^{Welch}(e^{j\omega})\} \approx \frac{1}{P}\sum_{k=0}^{P-1} E\{\hat{\Phi}_{xx}^{Per(k)}(e^{j\omega})\}$$
$$\approx E\{\hat{\Phi}_{xx}^{Per(k)}(e^{j\omega})\}$$
$$\approx \frac{1}{2\pi LU}\int_{-\pi}^{\pi}\Phi_{xx}(e^{ju})\cdot\left|W(e^{j(\omega-u)})\right|^2 du$$

Welch Method

- For large values of L follows

$$E\{\hat{\Phi}_{xx}^{Welch}(e^{j\omega})\} \approx \frac{1}{2\pi LU}\Phi_{xx}(e^{j\omega}) \int_{-\pi}^{\pi}\left|W(e^{j\omega})\right|^2 d\omega$$

- Using the Parseval's relation

$$\frac{1}{2\pi}\int_{-\pi}^{\pi}\left|W(e^{j\omega})\right|^2 d\omega = \sum_{n=0}^{L-1}\left|w[n]\right|^2$$

we obtain

$$E\{\hat{\Phi}_{xx}^{Welch}(e^{j\omega})\} \approx \frac{1}{LU}\sum_{n=0}^{L-1}\left|w[n]\right|^2 \cdot \Phi_{xx}(e^{j\omega})$$

Welch Method

- Choosing $U = \frac{1}{L}\sum_{n=0}^{L-1}\left|w[n]\right|^2$ we arrive at

$$E\{\hat{\Phi}_{xx}^{Welch}(e^{j\omega})\} \approx \Phi_{xx}(e^{j\omega})$$

- In the limiting case, the PSD Welch estimate is asymptotically unbiased, i.e.

$$\lim_{L\to\infty} E\{\hat{\Phi}_{xx}^{Welch}(e^{j\omega})\} = \Phi_{xx}(e^{j\omega})$$

- The variance of the PSD Welch estimate is more difficult to analyze

Welch Method

- Assuming P independent periodograms, it can be shown that the variance of the PSD Welch estimate reduces to

$$Var\{\hat{\Phi}_{xx}^{Welch}(e^{j\omega})\} \approx \frac{1}{P}\Phi_{xx}^2(e^{j\omega})$$

- Hence, the PSD Welch estimate is consistent
- Note that for increasing L, P becomes smaller
- For a fixed length N of the input signal, we have a trade-off between bias and variance

Welch Method

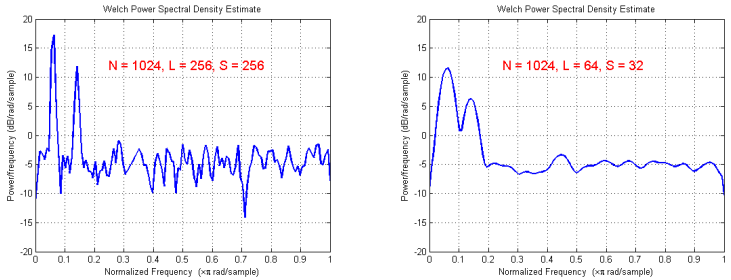
- For a fixed input length N, the corresponding values are generally related by $(P-1)S + L \le N$
- For variance reduction, choose L as small as possible, so that P will be large
- To obtain maximum frequency resolution, choose L as large as possible
- Thus, the decrease in variance is achieved at the cost of poorer frequency resolution, also referred to as the resolution-variance trade-off

Welch Method

- Since these are conflicting requirements, a suitable number of periodograms is required to get a good compromise between frequency resolution and variance
- In practice, 8 segments with 50% overlap and Hamming window are a good choice
- Note: The Bartlett method uses a rectangular window and no overlap and can be considered as a special case of the Welch method

Welch Method

- Example: Consider the sequence of the previous example



- The Welch PSD estimate is much smoother than a single periodogram

2.6 Blackman-Tukey Method

- We consider a sequence $x[k]$ of length N given by $x[k], 0 \le k \le N-1$ and its biased ACS estimate (see chapter 2.5) given by

$$\hat{r}_{xx}[|\kappa|] = \frac{1}{N} \sum_{k=0}^{N-1-|\kappa|} x^*[k]\, x[k+|\kappa|],\ 0 \le |\kappa| \le N-1$$

where $\hat{r}_{xx}[|\kappa|] = 0$ for $|\kappa| \ge N$

- $\hat{r}_{xx}[|\kappa|]$ is a conjugate symmetric sequence (as is the true ACS), i.e. $\hat{r}_{xx}[-|\kappa|] = \hat{r}^*_{xx}[|\kappa|]$

Blackman-Tukey Method

- The expected value of $\hat{r}_{xx}[|\kappa|]$ is given by

$$E\{\hat{r}_{xx}[|\kappa|]\} = \frac{N-|\kappa|}{N} \cdot r_{xx}[|\kappa|],\ |\kappa| \le N-1$$

i.e., $r_{xx}[|\kappa|]$ is weighted by a triangular (Bartlett) window

$$w_B[\kappa] = \begin{cases} \dfrac{N-|\kappa|}{N}, & |\kappa| \le N-1 \\ 0, & \textit{otherwise} \end{cases}$$

- Therefore, $\hat{r}_{xx}[|\kappa|]$ is a biased ACS estimate

Blackman-Tukey Method

- The bias is caused by the constant factor $1/N$, i.e. it is scaled independent from the effective quantity of $N-|\kappa|$ summands
- An unbiased ACS estimate is obtained by simply changing the scaling factor from $1/N$ to $1/(N-|\kappa|)$, resulting in

$$\hat{r}_{xx}^{ub}[|\kappa|] = \frac{1}{N-|\kappa|} \sum_{k=0}^{N-1-|\kappa|} x^*[k]\, x[k+|\kappa|], \quad |\kappa| \le N-1$$

where $\hat{r}_{xx}^{ub}[|\kappa|]$ is also conjugate symmetric

Blackman-Tukey Method

- It can be shown, that both types of ACS estimates are consistent
- However, if κ approaches N, the variance of $\hat{r}_{xx}^{ub}[|\kappa|]$ increases significantly
- For large lags κ, only a few samples of $x[k]$ enter into the computation of $\hat{r}_{xx}^{ub}[|\kappa|]$ which results in a very unreliable estimate
- Therefore, in practice the biased ACS estimate is most commonly used

Blackman-Tukey Method

- The **Blackman-Tukey method**, also referred to as **Correlogram method**, estimates the PSD as

$$\hat{\Phi}_{xx}^{BT}(e^{j\omega}) = \sum_{k=-(L-1)}^{L-1} \hat{r}_{xx}[k] \cdot w_C[k] \cdot e^{-j\omega k}$$

- It is the DTFT of the product of the biased ACS estimate and a correlation window $w_C[k]$
- The so-obtained PSD estimate differs from the periodogram by the additional correlation window

Blackman-Tukey Method

- $w_C[k]$ must have an even symmetry to produce a PSD estimate that is an even function of ω, i.e.

$$w_C[k] = \begin{cases} w_C[-k], & 0 \le k \le L-1, \; L \ll N \\ 0, & otherwise \end{cases}$$

- In the frequency domain, $\hat{\Phi}_{xx}^{BT}(e^{j\omega})$ is given by the convolution of the DTFT of the correlation window and the periodogram, i.e.

$$\hat{\Phi}_{xx}^{BT}(e^{j\omega}) = \frac{1}{2\pi} \int_{-\pi}^{\pi} \hat{\Phi}_{xx}^{Per}(e^{ju})\, W_C(e^{j(\omega-u)})\, du$$

Blackman-Tukey Method

- The correlation window has the effect of smoothing the fluctuating periodogram
- Note that we have to ensure that $\hat{\Phi}_{xx}^{BT}(e^{j\omega})$ is nonnegative, i.e. $W_C(e^{j\omega})$ has to be nonnegative
- The Bartlett and Parzen window satisfy this property (e.g. the Hamming window does not)
- The expected value of $\hat{\Phi}_{xx}^{BT}(e^{j\omega})$ is given by

$$E\{\hat{\Phi}_{xx}^{BT}(e^{j\omega})\} = \sum_{k=-(L-1)}^{L-1} E\{\hat{r}_{xx}[k]\} \cdot w_C[k] \cdot e^{-j\omega k}$$

Blackman-Tukey Method

- Using $E\{\hat{r}_{xx}[k]\} = w_B[k] \cdot r_{xx}[k]$, we obtain

$$E\{\hat{\Phi}_{xx}^{BT}(e^{j\omega})\} = \sum_{k=-(L-1)}^{L-1} r_{xx}[k] \cdot w_B[k] \cdot w_C[k] \cdot e^{-j\omega k}$$

  which is the DTFT of the true ACS weighted by the Bartlett window and the correlation window
- In the frequency domain, we have

$$E\{\hat{\Phi}_{xx}^{BT}(e^{j\omega})\} = \Phi_{xx}(e^{j\omega}) \circledast W_B(e^{j\omega}) \circledast W_C(e^{j\omega})$$

Blackman-Tukey Method

- For $N >> L$, the expected value of the PSD estimate $\hat{\Phi}_{xx}^{BT}(e^{j\omega})$ can be approximated by

$$E\{\hat{\Phi}_{xx}^{BT}(e^{j\omega})\} \approx \frac{1}{2\pi} \int_{-\pi}^{\pi} \Phi_{xx}(e^{ju}) W_C(e^{j(\omega-u)}) du$$

- Hence, $\hat{\Phi}_{xx}^{BT}(e^{j\omega})$ is biased due the frequency response of the correlation window
- However, as $\lim_{L\to\infty} E\{\hat{\Phi}_{xx}^{BT}(e^{j\omega})\} = w_C[0] \cdot \Phi_{xx}(e^{j\omega})$ the correlogram is asymptotically unbiased under the condition $w_C[0] = 1$

Blackman-Tukey Method

- For $N >> L$, the variance can be approximated as

$$Var\{\hat{\Phi}_{xx}^{BT}(e^{j\omega})\} \approx \left[\frac{1}{N} \sum_{k=-(L-1)}^{L-1} w_C^2[k]\right] \cdot \Phi_{xx}^2(e^{j\omega})$$

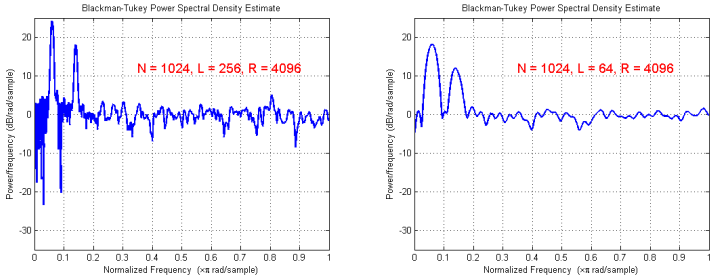
- Compared to the variance of the periodogram, the variance is weighted by a factor which depends on the type and the length L of the correlation window and the input length N
- For a rectangular or a Bartlett windows this factor is $2L/N$ and $2L/3N$, respectively

Blackman-Tukey Method

- Assuming $N >> L$, the factor is less than unity and hence the variance is reduced compared to the variance of the periodogram
- As in the Welch method, there is a frequency resolution and variance trade-off
- For higher resolution, L must be large so that $W_C(e^{j\omega})$ is closer to a dirac impulse $\delta(\omega)$
- For a small variance, the fraction L/N should be as small as possible

Blackman-Tukey Method

- Example: Consider the sequence of the previous example



- A decreased value of L/N reduces the variance and reduces the frequency resolution

2.7 AR-Modeling and Yule-Walker Equations

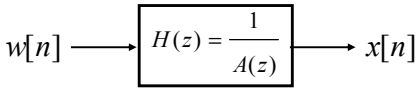
- The most commonly used kind of model in parametric PSD estimation methods is an autoregressive (AR) model, i.e. an all-pole filter
- The output of such a model for white noise input is called an AR process
- For this reason, AR-model based methods are also referred to as AR methods of spectral estimation

AR-Modeling

- Assume a causal AR-model of order M with the transfer function

$$H(z) = \frac{1}{A(z)} = \frac{1}{1 + \sum_{k=1}^{M} a_M[k] \cdot z^{-k}}$$

- The input $w[n]$ of $H(z)$ is a white noise signal with zero mean and variance σ_w^2, the output $x[n]$ models the AR process



PSD of AR-Model Output

- The PSD of the AR-model output is due to the **Wiener-Lee relation** given by

$$\Phi_{xx}^{AR}(e^{j\omega}) = \sigma_w^2 \cdot \left|H(e^{j\omega})\right|^2 = \frac{\sigma_w^2}{\left|A(e^{j\omega})\right|^2}$$

- The difference equation for the AR-model is the relationship between the white-noise input $w[n]$ and the output $x[n]$ to be analyzed

$$x[n] + \sum_{k=1}^{M} a_M[k] \cdot x[n-k] = w[n], -\infty < n < \infty$$

AR-Modeling and Yule-Walker Equations

- The Yule-Walker equations represent a basic relationship between the ACS (estimate) and the model coefficients
- The equations can be derived by multiplying the difference equation by $x^*[n-m]$ and then taking the expectation, i.e.

$$E\{x[n] \cdot x^*[n-m]\} = E\{w[n] \cdot x^*[n-m]\} - \sum_{k=1}^{M} a_M[k] \cdot E\{x[n-k] \cdot x^*[n-m]\}$$

AR-Modeling and Yule-Walker Equations

- The ACS given by $r_{xx}[m] = E\{x^*[n] \cdot x[n+m]\}$ is conjugate-symmetric, i.e.

$$r_{xx}[m] = r_{xx}^*[-m] = E\{x^*[n] \cdot x[n-m]\}^* = E\{x[n] \cdot x^*[n-m]\}$$

- The cross-correlation between $w[n]$ and $x[n]$ is given by $r_{wx}[m] = E\{w^*[n] \cdot x[n+m]\}$ and we deploy

$$E\{w[n] \cdot x^*[n-m]\} = r_{wx}^*[-m]$$

AR-Modeling and Yule-Walker Equations

- Since $x[n]$ is the convolution of $w[n]$ and $h[n]$, we can rewrite $r_{wx}[m]$ as

$$r_{wx}[m] = E\{w^*[m] \cdot \sum_{i=0}^{\infty} h[i] \cdot w[n-i+m]\} = \sum_{i=0}^{\infty} h[i] \cdot E\{w^*[m] \cdot w[n-i+m]\} = h[m] \circledast r_{ww}[m]$$

- Recall $w[n]$ is a white noise with zero mean and variance σ_w^2, therefore, we finally obtain

$$r_{wx}[m] = h[m] \circledast \sigma_w^2 \cdot \delta[m] = \sigma_w^2 \cdot h[m]$$

AR-Modeling and Yule-Walker Equations

- $h[n]$ is a causal sequence with $h[0] = 1$
- Using the relation $r_{wx}[m] = \sigma_w^2 \cdot h[m]$, it follows

$$r_{wx}^*[-m] = \begin{cases} 0, & m > 0 \\ \sigma_w^2, & m = 0 \end{cases}$$

- Thus, we obtain the relationship in the form

$$r_{xx}[m] = \begin{cases} -\sum_{k=1}^{M} a_M[k] \cdot r_{xx}[m-k] + \sigma_w^2, & m = 0 \\ -\sum_{k=1}^{M} a_M[k] \cdot r_{xx}[m-k], & m > 0 \end{cases}$$

AR-Modeling and Yule-Walker Equations

- In the case of an AR-model the relationship between the ACS estimate and the coefficients of the AR-model becomes linear
- For this reason, AR-models are widely used in practical spectral estimation methods
- The linear system is of Toeplitz structure and it can be solved recursively using the Levinson-Durbin algorithm providing the corresponding lattice filter structures

AR-Modeling and Yule-Walker Equations

- For $m = 1, 2, \ldots, M$ (i.e. $m > 0$), the $M + 1$ ACS values relate to the M model coefficients given by the **Yule-Walker equations**

$$\underbrace{\begin{bmatrix} r_{xx}[0] & r_{xx}[-1] & \cdots & r_{xx}[-(M-1)] \\ r_{xx}[1] & r_{xx}[0] & \cdots & r_{xx}[-(M-2)] \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}[M-1] & r_{xx}[M-2] & \cdots & r_{xx}[0] \end{bmatrix}}_{\boldsymbol{R}_{xx}} \cdot \underbrace{\begin{bmatrix} a_M[1] \\ a_M[2] \\ \vdots \\ a_M[M] \end{bmatrix}}_{\boldsymbol{a}} = - \underbrace{\begin{bmatrix} r_{xx}[1] \\ r_{xx}[2] \\ \vdots \\ r_{xx}[M] \end{bmatrix}}_{\boldsymbol{r}_{xx}}$$

Note: $\boldsymbol{R}_{xx}$ is a Toeplitz matrix, i.e. $\boldsymbol{R}_{xx}(i, j) = \boldsymbol{R}_{xx}(i+1, j+1)$, and $\boldsymbol{R}_{xx}$ is a Hermitian matrix, i.e. $\boldsymbol{R}_{xx}^*(i, j) = \boldsymbol{R}_{xx}(j, i)$

AR-Modeling and Yule-Walker Equations

- Using the autocorrelation vector $\boldsymbol{r}_{xx}$, the coefficient vector $\boldsymbol{a}$ and the autocorrelation matrix $\boldsymbol{R}_{xx}$, we can express the Yule-Walker equations in matrix form as follows

$$\boldsymbol{R}_{xx} \cdot \boldsymbol{a} = -\boldsymbol{r}_{xx}$$

- Thus, knowing the $M+1$ values $r_{xx}[0] \ldots r_{xx}[M]$ of the ACS, the M coefficients of the AR-model are given by $\boldsymbol{a} = -\boldsymbol{R}_{xx}^{-1} \cdot \boldsymbol{r}_{xx}$

AR-Modeling and Yule-Walker Equations

- Using $\boldsymbol{R}_{xx}^{*}(\mathrm{i},\mathrm{j}) = \boldsymbol{R}_{xx}(\mathrm{j},\mathrm{i})$ or $r_{xx}[-k] = r_{xx}^{*}[k]$, the **Yule-Walker equations** are also given by

$$\begin{bmatrix} r_{xx}[0] & r_{xx}^{*}[1] & \cdots & r_{xx}^{*}[M-1] \\ r_{xx}[1] & r_{xx}[0] & \cdots & r_{xx}^{*}[M-2] \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}[M-1] & r_{xx}[M-2] & \cdots & r_{xx}[0] \end{bmatrix} \cdot \begin{bmatrix} a_M[1] \\ a_M[2] \\ \vdots \\ a_M[M] \end{bmatrix} = - \begin{bmatrix} r_{xx}[1] \\ r_{xx}[2] \\ \vdots \\ r_{xx}[M] \end{bmatrix}$$

- Note that for $m = 0$ and knowing the model coefficients, the noise variance σ_w^2 is given by

$$\sigma_w^2 = r_{xx}[0] + \sum_{k=1}^{M} a_M[k] \cdot r_{xx}^{*}[k]$$

AR-Model Order Selection

- For AR-models, a model order must be chosen
- A low model order produces a smooth spectral estimate, and a too high model order tends to introduce spurious details in the PSD
- Several criteria are available for order selection, but none of them appear to be well suited for determining the model order in all cases
- For a data length N, model orders between $N/3$ and $N/2$ appear to produce satisfactory results

2.8 Yule-Walker Method and Covariance Method

- The Yule-Walker method and the covariance method determine the coefficients of an AR-model by solving the Yule-Walker equations
- Since the ACS is not known, an ACS estimate must be used in this two methods
- There is a one-to-one correspondence between the AR-model parameters and the coefficients of a linear forward predictor (prediction filter)

Yule-Walker Method

- Initially, we consider the one-step forward linear predictor to predict $\hat{x}[n]$ by a weighted linear combination of the P past samples

$$x[n-1], x[n-2], \cdots, x[n-P]$$

given by $\hat{x}[n] = \sum_{k=1}^{P} p_{P,f}[k] \cdot x[n-k]$

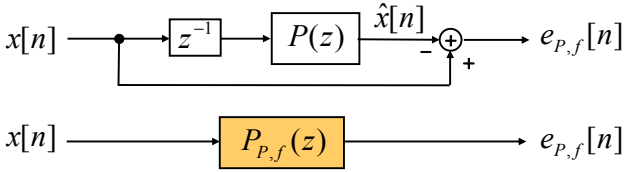
$p_{P,f}[k]$ are the forward prediction coefficients

- The forward prediction error of order P is given by

$$e_{P,f}[n] = x[n] - \hat{x}[n] = x[n] - \sum_{k=1}^{P} p_{P,f}[k] \cdot x[n-k]$$

Yule-Walker Method

- Linear prediction is equivalent to FIR filtering



- The transfer function of the forward prediction filter is given by

$$P_{P,f}(z) = 1 - z^{-1} \cdot P(z) = 1 - \sum_{k=1}^{P} p_{P,f}[k] \cdot z^{-k} = \sum_{k=0}^{P} p_{P,f}[k] \cdot z^{-k}$$

where $P(z) = \sum_{k=1}^{P} p_{P,f}[k] \cdot z^{-k+1}$ and $p_{P,f}[0] = 1$

Yule-Walker Method

- By minimizing the mean squared error of the forward prediction error, i.e.

$$E\left\{ \left| e_{P,f}[n] \right|^2 \right\} \stackrel{!}{=} Min = \varepsilon_{P,f}^{YW}$$

we obtain the optimal coefficients as follows

$$\boldsymbol{p}_{P,f} = \boldsymbol{R}_{xx}^{-1} \cdot \boldsymbol{r}_{xx}$$

where $\boldsymbol{p}_{P,f} = [p_{P,f}[1] \;\; p_{P,f}[2] \;\cdots\; p_{P,f}[P]]^T$ is the forward predictor coefficient vector of order P, $\boldsymbol{R}_{xx}$ is the autocorrelation matrix, and $\boldsymbol{r}_{xx}$ is the autocorrelation vector, respectively

Yule-Walker Method

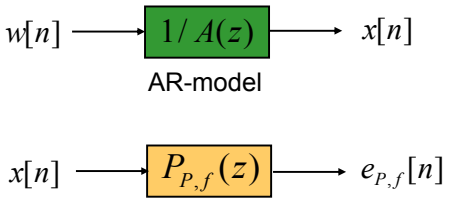
- The solution is also referred to as the normal equations of linear prediction (Wiener-Hopf)
- Note that the Yule-Walker equations for the coefficient vector $\boldsymbol{a}_P$ of order P are given by

$$\boldsymbol{a}_P = -\boldsymbol{R}_{xx}^{-1} \cdot \boldsymbol{r}_{xx}$$

- Hence, the coefficients $p_{Pf}[k]$ of the forward prediction filter are obtained by the negative model coefficients of the AR-model, i.e.

$$\boldsymbol{p}_{P,f} = -\boldsymbol{a}_P$$

Yule-Walker Method



$$\frac{1}{A(z)} = \frac{1}{1 + \sum_{k=1}^{P} a_P[k] \cdot z^{-k}}$$

$$P_{P,f}(z) = 1 - \sum_{k=1}^{P} p_{P,f}[k] \cdot z^{-k}$$

$$\boldsymbol{a}_P = -\boldsymbol{p}_{P,f}, \; A(z) = P_{P,f}(z)$$

- The random signal (process) to be analyzed is the response of the AR-model to a white noise
- Conversely, the random signal is transformed into a white noise by the prediction filter

Yule-Walker Method

- The MMSE for a prediction filter of order P determined for a signal of length N is given by

$$\hat{\mathcal{E}}_{P,f}^{YW} = \frac{1}{N+P}\sum_{n=0}^{N+P-1}\left|e_{P,f}[n]\right|^2 = \hat{r}_{xx}[0] - \sum_{k=1}^{P} p_{P,f}[k]\cdot\hat{r}_{xx}^{*}[k]$$

- This quantity is an estimate for the variance of the white noise σ_w^2 at the AR-model input
- Therefore, the prediction filter is also referred to as a whitening filter

Yule-Walker Method

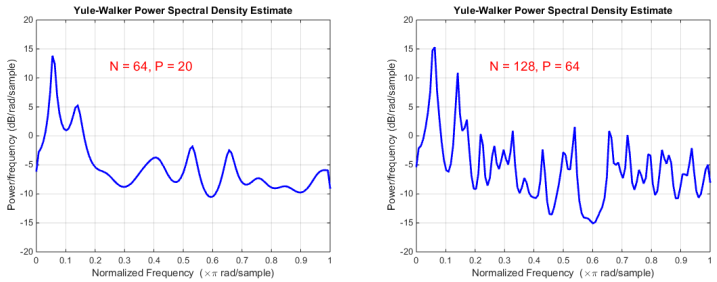
- The Yule-Walker PSD estimate is given by

$$\hat{\Phi}_{xx}^{YW}(e^{j\omega}) = \frac{\hat{\mathcal{E}}_{P,f}^{YW}}{\left|A(e^{j\omega})\right|^2} = \frac{\hat{\mathcal{E}}_{P,f}^{YW}}{\left|1+\sum_{k=1}^{P} a_P[k]\cdot e^{-j\omega k}\right|^2}$$

- It makes use of the biased ACS estimate, thus the Yule-Walker PSD estimate is biased
- Because the FIR forward prediction filter is minimum-phase, the AR-model is always stable

Yule-Walker Method

- Example: Noise-corrupted sinusoidal sequence of the previous example for $N = 64$ or $N = 128$



- Note that the Yule-Walker PSD estimate is relatively poor for short data records

Covariance Method

- The covariance method only involves the steady-state response of the prediction filter
- Thus, the forward prediction error is only minimized over the sample range from $n = P$ to $n = N - 1$, i.e.

$$\hat{\mathcal{E}}_{P,f}^{Cov} = \frac{1}{N-P}\sum_{n=P}^{N-1}\left|e_{P,f}[n]\right|^2 \overset{!}{=} Min$$

- It can be shown that the covariance method deploys the unbiased ACS estimate, therefore the covariance PSD estimate is unbiased

Covariance Method

- The covariance method differs from the Yule-Walker method in the underlying matrix to determine the coefficients of the AR-model
- Unfortunately, the matrix is not a Toeplitz matrix and thus the Levinson-Durbin recursion (see next chapter) cannot be applied
- Furthermore, the FIR prediction filter is not necessarily minimum-phase, and thus the AR-model is not necessarily stable

2.9 Levinson-Durbin Recursion and Burg Method

- The Burg method estimates the coefficients of an AR-model based on the minimization of forward and backward prediction errors
- The Levinson-Durbin recursion (LDR) is used to develop an order-recursive relation for these errors along with a lattice filter structure
- The method produces a stable model, and it provides high resolution for short data records

Levinson-Durbin Recursion

- Consider the Yule-Walker equations $\boldsymbol{R}_{xx}\,\boldsymbol{a} = -\boldsymbol{r}_{xx}$ for the AR-model of order P given by

$$\begin{bmatrix} r_{xx}[0] & r_{xx}^{*}[1] & \cdots & r_{xx}^{*}[P-1] \\ r_{xx}[1] & r_{xx}[0] & \cdots & r_{xx}[P-2] \\ \vdots & \vdots & \ddots & \vdots \\ r_{xx}[P-1] & r_{xx}[P-2] & \cdots & r_{xx}[0] \end{bmatrix} \cdot \begin{bmatrix} a_p[1] \\ a_p[2] \\ \vdots \\ a_p[P] \end{bmatrix} = - \begin{bmatrix} r_{xx}[1] \\ r_{xx}[2] \\ \vdots \\ r_{xx}[P] \end{bmatrix}$$

$\boldsymbol{R}_{xx}$ is a Toeplitz matrix, i.e. $\boldsymbol{R}_{xx}(i, j) = \boldsymbol{R}_{xx}(i+1, j+1)$ and it is a Hermitian matrix, i.e. $\boldsymbol{R}_{xx}^{*}(i, j) = \boldsymbol{R}_{xx}(j, i)$; $i, j = 1,2, \ldots, P\text{-}1$

Levinson-Durbin Recursion

- The key to the LDR algorithm is a recursion, beginning with a model of order $P = 1$, i.e.

$$r_{xx}[1] = -a_1[1] \cdot r_{xx}[0]$$

- Thus the 1st-order model coefficient is

$$a_1[1] = -\frac{r_{xx}[1]}{r_{xx}[0]}$$

and the MMSE is given by

$$\mathcal{E}_1 = r_{xx}[0] + a_1[1] \cdot r_{xx}[-1] = r_{xx}[0] + a_1[1] \cdot r_{xx}^{*}[1]$$

$$= r_{xx}[0] \cdot (1 - |a_1[1]|^2)$$

Levinson-Durbin Recursion

- We denote the jth coefficient of the jth-order model by the reflection coefficient K_j, i.e.

$$a_j[j] = K_j$$

- K_j is shown to be the parameter of the jth-stage of the lattice structure
- For $P = 1$, $a_1[1]$ is the only coefficient given by

$$K_1 = a_1[1] = -\frac{r_{xx}[1]}{r_{xx}[0]}$$

Levinson-Durbin Recursion

- The next step is to solve for the coefficients $a_2[k], k = 1, 2$ of the model of 2nd-order
- The corresponding two equations are

$$a_2[1] \cdot r_{xx}[0] + a_2[2] \cdot r_{xx}^*[1] = -r_{xx}[1]$$

$$a_2[1] \cdot r_{xx}[1] + a_2[2] \cdot r_{xx}[0] = -r_{xx}[2]$$

- Using the solution for $a_1[1]$, we obtain

$$a_2[2] = -\frac{r_{xx}[2] + a_1[1] \cdot r_{xx}[1]}{r_{xx}[0](1 - |a_1[1]|^2)} = -\frac{r_{xx}[2] + a_1[1] \cdot r_{xx}[1]}{\mathcal{E}_1}$$

$$a_2[1] = a_1[1] + a_2[2] \cdot a_1^*[1]$$

Levinson-Durbin Recursion

- Thus, the second reflection coefficient is given by $K_2 = a_2[2]$
- Next, we express the coefficients of the 2nd-order model using the 1st-order model and the reflection coefficient K_2 as follows

$$\boldsymbol{a}_2 = \begin{bmatrix} a_2[1] \\ a_2[2] \end{bmatrix} = \begin{bmatrix} a_1[1] \\ 0 \end{bmatrix} + \begin{bmatrix} K_2 \cdot a_1^*[1] \\ K_2 \end{bmatrix}$$

- In general, the coefficient vector $\boldsymbol{a}_j$ may be expressed in terms of the vector $\boldsymbol{a}_{j-1}$ and the reflection coefficient K_j

Levinson-Durbin Recursion

- Thus, we obtain for the coefficient vector $\boldsymbol{a}_j$

$$\mathbf{a}_j = \begin{bmatrix} a_j[1] \\ a_j[2] \\ \vdots \\ a_j[j] \end{bmatrix} = \begin{bmatrix} \mathbf{a}_{j-1} \\ 0 \end{bmatrix} + \begin{bmatrix} K_j \cdot \mathbf{c}_{j-1} \\ K_j \end{bmatrix}$$

where $\boldsymbol{a}_{j-1}$ is the coefficient vector for the AR-model of order $j-1$

- Assume $\boldsymbol{a}_{j-1}$ is known, then the vector $\boldsymbol{c}_{j-1}$ and the reflection coefficient K_j of order j must be determined

Levinson-Durbin Recursion

- Let the $j \times j$ autocorrelation matrix $\boldsymbol{R}_{xx}^{j\times j}$ be partitioned as

$$\boldsymbol{R}_{xx}^{j\times j} = \boldsymbol{R}_j = \begin{bmatrix} \boldsymbol{R}_{j-1} & \boldsymbol{E}_{j-1}\cdot\boldsymbol{r}_{j-1}^{*} \\ \boldsymbol{r}_{j-1}^{T}\cdot\boldsymbol{E}_{j-1} & r_{xx}[0] \end{bmatrix}$$

  where $\boldsymbol{r}_{j-1}$ denotes the autocorrelation vector $\boldsymbol{r}_{j-1} = [r_{xx}^{j-1}[1]\, r_{xx}^{j-1}[2] \cdots r_{xx}^{j-1}[j-1]]^T$ and $\boldsymbol{E}_{j-1}$ is the exchange matrix of order $(j-1)\times(j-1)$
- The exchange matrix $\boldsymbol{E}_{j-1}$ flips the vector, i.e. the vector elements are in reversed order

$$\boldsymbol{E}_{j-1}\cdot\boldsymbol{r}_{j-1}^{T} = [r_{xx}^{j-1}[j-1]\, r_{xx}^{j-1}[j-2] \cdots r_{xx}^{j-1}[1]]^T$$

Levinson-Durbin Recursion

- $\boldsymbol{E}_{j-1}$ is given by

$$\boldsymbol{E}_{j-1} = \begin{bmatrix} 0 & 0 & \cdots & 1 \\ 0 & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ 0 & 1 & \cdots & \vdots \\ 1 & 0 & \cdots & 0 \end{bmatrix}$$

- Then, the set of equations $\boldsymbol{R}_j \cdot \boldsymbol{a}_j = -\boldsymbol{r}_j$ of order j can be expressed as follows

$$\begin{bmatrix} \boldsymbol{R}_{j-1} & \boldsymbol{E}_{j-1}\cdot\boldsymbol{r}_{j-1}^{*} \\ \boldsymbol{r}_{j-1}^{T}\cdot \mathrm{E}_{j-1} & r_{xx}[0] \end{bmatrix} \cdot \left\{ \begin{bmatrix} \boldsymbol{a}_{j-1} \\ 0 \end{bmatrix} + \begin{bmatrix} K_j \cdot \boldsymbol{c}_{j-1} \\ K_j \end{bmatrix} \right\} = -\begin{bmatrix} \boldsymbol{r}_{j-1} \\ r_{xx}[j] \end{bmatrix} = -\boldsymbol{r}_j$$

Levinson-Durbin Recursion

- Carrying out the multiplications leads to the following two equations

$$\boldsymbol{R}_{j-1}\cdot\boldsymbol{a}_{j-1} + K_j \cdot \boldsymbol{R}_{j-1}\cdot\boldsymbol{c}_{j-1} + K_j \cdot \boldsymbol{E}_{j-1}\cdot\boldsymbol{r}_{j-1}^{*} = -\boldsymbol{r}_{j-1}$$

$$\boldsymbol{r}_{j-1}^{T}\cdot\boldsymbol{E}_{j-1}\cdot\boldsymbol{a}_{j-1} + K_j \cdot \boldsymbol{r}_{j-1}^{T}\cdot\boldsymbol{E}_{j-1}\cdot\boldsymbol{c}_{j-1} + K_j \cdot r_{xx}[0] = -r_{xx}[j]$$

- Since $\boldsymbol{R}_{j-1}\cdot\boldsymbol{a}_{j-1} = -\boldsymbol{r}_{j-1}$ we get from the first equation

$$\boldsymbol{R}_{j-1}\cdot\boldsymbol{c}_{j-1} = -\boldsymbol{E}_{j-1}\cdot\boldsymbol{r}_{j-1}^{*}$$

- Let $\boldsymbol{I}_{j-1}$ the $(j-1)\times(j-1)$ identity matrix

Levinson-Durbin Recursion

- Then, exploiting the two relations

$$\boldsymbol{R}_{j-1} = \boldsymbol{E}_{j-1}\cdot\boldsymbol{R}_{j-1}^{*}\cdot\boldsymbol{E}_{j-1}$$

$$\boldsymbol{E}_{j-1}\cdot\boldsymbol{E}_{j-1} = \boldsymbol{I}_{j-1}$$

  in $\boldsymbol{R}_{j-1}\cdot\boldsymbol{c}_{j-1} = -\boldsymbol{E}_{j-1}\cdot\boldsymbol{r}_{j-1}^{*}$ results in

$$\boldsymbol{c}_{j-1} = \boldsymbol{E}_{j-1}\cdot\boldsymbol{a}_{j-1}^{*}$$

- That is, the vector $\boldsymbol{c}_{j-1}$ can be obtained from the model vector $\boldsymbol{a}_{j-1}$ with conjugated elements in reversed order

Levinson-Durbin Recursion

- Substituting $\boldsymbol{c}_{j-1} = \boldsymbol{E}_{j-1} \cdot \boldsymbol{a}_{j-1}^{*}$ in the equation for $r_{xx}[j]$ and using $\boldsymbol{\varepsilon}_{j-1}$ for the model of order $j-1$

$$\boldsymbol{\varepsilon}_{j-1} = r_{xx}[0] + \boldsymbol{r}_{j-1}^{T} \cdot \boldsymbol{a}_{j-1}^{*}$$

we obtain for the reflection coefficient K_j

$$K_j = a_j[j] = -\frac{r_{xx}[j] + \boldsymbol{r}_{j-1}^{T} \cdot \boldsymbol{E}_{j-1} \cdot \boldsymbol{a}_{j-1}}{\boldsymbol{\varepsilon}_{j-1}}$$

- Note: K_j depends only on the known quantities, namely $\boldsymbol{a}_{j-1}$ and $\boldsymbol{\varepsilon}_{j-1}$, each of order $j-1$

Levinson-Durbin Recursion

- Now, substituting $\boldsymbol{c}_{j-1} = \boldsymbol{E}_{j-1} \cdot \boldsymbol{a}_{j-1}^{*}$ in the equation

$$\boldsymbol{a}_j = \begin{bmatrix} \boldsymbol{a}_{j-1} \\ 0 \end{bmatrix} + \begin{bmatrix} K_j \cdot \boldsymbol{c}_{j-1} \\ K_j \end{bmatrix}$$

we can write the coefficient vector $\boldsymbol{a}_j$ in terms of the known quantities as follows

$$\boldsymbol{a}_j = \begin{bmatrix} \boldsymbol{a}_{j-1} \\ 0 \end{bmatrix} + \begin{bmatrix} K_j \cdot \boldsymbol{E}_{j-1} \cdot \boldsymbol{a}_{j-1}^{*} \\ K_j \end{bmatrix}$$

- Then, the only quantity left is the error $\boldsymbol{\varepsilon}_j$, which can be expressed in terms of $\boldsymbol{\varepsilon}_{j-1}$ and K_j

Levinson-Durbin Recursion

- To this end, we use $\boldsymbol{\varepsilon}_j = r_{xx}[0] + \boldsymbol{r}_j^{T} \cdot \boldsymbol{a}_j^{*}$ and the equations

$$K_j = -\frac{r_{xx}[j] + \boldsymbol{r}_{j-1}^{T} \cdot \boldsymbol{E}_{j-1} \cdot \boldsymbol{a}_{j-1}}{\boldsymbol{\varepsilon}_{j-1}}$$

$$\boldsymbol{a}_j = \begin{bmatrix} \boldsymbol{a}_{j-1} \\ 0 \end{bmatrix} + \begin{bmatrix} K_j \cdot \boldsymbol{E}_{j-1} \cdot \boldsymbol{a}_{j-1}^{*} \\ K_j \end{bmatrix}$$

to obtain the error $\boldsymbol{\varepsilon}_j$ for a model of order j as

$$\boldsymbol{\varepsilon}_j = \boldsymbol{\varepsilon}_{j-1} \cdot (1 - |K_j|^2)$$

Levinson-Durbin Recursion

Summary

Step 1 Initialize the recursion for $\boldsymbol{j = 1}$ with

$$K_1 = a_1[1] = -\frac{r_{xx}[1]}{r_{xx}[0]}$$

$$\boldsymbol{\varepsilon}_1 = r_{xx}[0] \cdot (1 - |K_1|^2)$$

Step 2 For $\boldsymbol{j = 2, 3, \ldots, P}$ compute recursively

$$K_j = -\frac{r_{xx}[j] + \boldsymbol{r}_{j-1}^{T} \cdot \boldsymbol{E}_{j-1} \cdot \boldsymbol{a}_{j-1}}{\boldsymbol{\varepsilon}_{j-1}}$$

$\boldsymbol{E}_{j-1}$ denotes the $(j-1) \times (j-1)$ exchange matrix

Levinson-Durbin Recursion

$$\boldsymbol{a}_j = \begin{bmatrix} \boldsymbol{a}_{j-1} \\ 0 \end{bmatrix} + K_j \cdot \begin{bmatrix} \boldsymbol{E}_{j-1} \cdot \boldsymbol{a}^*_{j-1} \\ 1 \end{bmatrix}$$

and $\mathcal{E}_j = \mathcal{E}_{j-1} \cdot (1 - |K_j|^2)$

- For an AR process with ACS `r`, the statement `[a,e,k] = levinson(r,P)` finds the AR model vector `a`, the prediction error `e`, and the reflection coefficient vector `k` of order `P`

Backward Linear Prediction

- In addition to the forward prediction, we also predict the past using a backward predictor
- Assume $x[n]$, $x[n-1]$, …, $x[n-P+1]$ to predict the past sample $x[n-P]$ using a P-step backward linear predictor as follows

$$\hat{x}[n-P] = \sum_{k=0}^{P-1} p_{P,b}[k] \cdot x[n-k]$$

where $p_{P,b}[k]$ are the backward prediction coefficients of order P

Backward Linear Prediction

- The backward prediction error is given by

$$e_{P,b}[n] = x[n-P] - \hat{x}[n-P]$$
$$= x[n-P] - \sum_{k=0}^{P-1} p_{P,b}[k] \cdot x[n-k]$$

- The coefficients $p_{P,b}[k]$ of the P-step backward linear predictor can be determined in the same way as the coefficients $p_{Pf}[k]$ of the one-step forward linear predictor of order P by minimization the MSE of the prediction error

Backward Linear Prediction

- As result, the backward predictor coefficients are the complex conjugates of the forward prediction coefficients in reversed order, i.e.

$$p_{P,b}[k] = p^*_{P,f}[P-k],\ k = 0, 1, \cdots, P$$

- Using the exchange matrix, there are given by

$$\boldsymbol{p}_{P,b} = \boldsymbol{E}_P \cdot \boldsymbol{p}^*_{P,f}$$

- The MMSE of the P-step backward predictor is equal to the MMSE of the one-step forward predictor of order P, i.e. $\mathcal{E}_{P,b} = \mathcal{E}_{P,f}$

Lattice Structure for the Prediction Filter

- The LDR suggests an order-recursive lattice structure to obtain the order-recursive forward and backward prediction error sequences
- We use the forward and backward prediction error sequences of order P given by

$$e_{P,f}[n] = x[n] - \sum_{k=1}^{P} p_{P,f}[k] \cdot x[n-k]$$

$$e_{P,b}[n] = x[n-P] - \sum_{k=0}^{P-1} p_{P,b}[k] \cdot x[n-k]$$

Lattice Structure for the Prediction Filter

- Since the backward prediction coefficients are related by $\boldsymbol{p}_{P,b} = \boldsymbol{E}_P \cdot \boldsymbol{p}_{P,f}^*$

  the backward prediction error sequence can be expressed in terms of the forward predictor coefficients as follows

$$e_{P,b}[n] = x[n-P] - \sum_{k=0}^{P-1} p_{P,f}^*[P-k] \cdot x[n-k]$$

Lattice Structure for the Prediction Filter

- The forward predictor coefficients are due to the LDR given by

$$\mathbf{p}_{j,f} = \begin{bmatrix} \mathbf{p}_{j-1,f} \\ 0 \end{bmatrix} - K_j \cdot \begin{bmatrix} \mathbf{E}_{j-1} \cdot \mathbf{p}_{j-1,f}^* \\ -1 \end{bmatrix}$$

- After some algebraic manipulations, the order-recursive forward prediction error sequence $e_{j,f}[n]$ can be expressed as follows

$$e_{j,f}[n] = e_{j-1,f}[n] - K_j \cdot e_{j-1,b}[n-1]$$

Lattice Structure for the Prediction Filter

- The order-recursive backward prediction error sequence $e_{j,b}[n]$ can be derived with the help of the backward predictor coefficients given by

$$\mathbf{p}_{j,b} = \mathbf{E}_j \cdot \mathbf{p}_{j,f}^* = \begin{bmatrix} 0 \\ \mathbf{E}_{j-1} \cdot \mathbf{p}_{j-1,f}^* \end{bmatrix} - K_j^* \cdot \begin{bmatrix} -1 \\ \mathbf{p}_{j-1,f} \end{bmatrix}$$

- In this case, we obtain

$$e_{j,b}[n] = e_{j-1,b}[n-1] - K_j^* \cdot e_{j-1,f}[n]$$

Lattice Structure for the Prediction Filter

- The order-recursive equations for the two sequences for $1 \le j \le P$ are given by

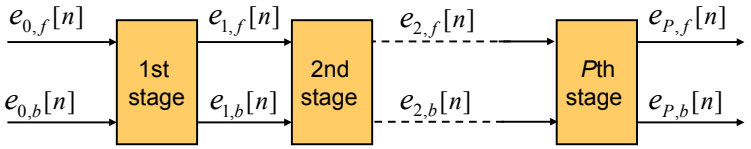
$$e_{j,f}[n] = e_{j-1,f}[n] - K_j \cdot e_{j-1,b}[n-1]$$

$$e_{j,b}[n] = e_{j-1,b}[n-1] - K_j^* \cdot e_{j-1,f}[n]$$

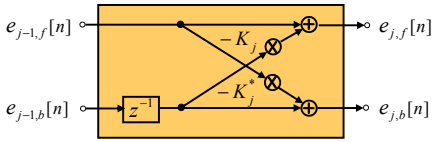
- This set of equations describes a lattice filter structure for the FIR prediction filter
- The MATLAB function **`arburg`** can be used to estimate the reflection coefficients K_j

Lattice Structure for the Prediction Filter

- ***P*-stage lattice structure** with inputs $e_{0,f}[n]$ and $e_{0,b}[n]$ and outputs are $e_{P,f}[n]$ and $e_{P,b}[n]$



j-th stage of the lattice structure



Burg Method

- **Goal:** Minimize the arithmetic mean of the forward and the backward prediction error variances over the steady-state response of the prediction filter of order j, i.e.

$$\frac{1}{2}\sum_{n=j}^{N-1}\{|e_{j,f}[n]|^2 + |e_{j,b}[n]|^2\} \stackrel{!}{=} Min$$

- In contrast to the previous methods, the method initially estimates the reflection coefficients
- Then, the LDR is used to obtain the AR-model parameters from the reflection coefficients

Burg Method

- The reflection coefficients are attained with the minimization approach

$$\frac{\partial}{\partial K_j}\left\{\sum_{n=j}^{N-1}|e_{j,f}[n]|^2 + |e_{j,b}[n]|^2\right\} = 0;\ j = 1, 2, \cdots, P$$

as follows

$$K_j = \frac{2 \cdot \sum_{n=j}^{N-1} e_{j-1,f}[n] \cdot e_{j-1,b}^*[n-1]}{\sum_{n=j}^{N-1}|e_{j-1,f}[n]|^2 + |e_{j-1,b}[n-1]|^2}$$

Burg Algorithm

Step 1 Initialize the algorithm for $j = 0$ with

$$a_0 = 1,\ e_{0,f}[n] = e_{0,b}[n] = x[n],\ \varepsilon_0 = \frac{1}{N}\sum_{n=0}^{N-1} |x[n]|^2$$

Step 2 Compute recursively for $j = 1, 2, \ldots, P$

- the reflection coefficient K_j
- the error sequences $e_{j,f}[n]$ and $e_{j,b}[n]$
- the error variance $\varepsilon_j = \varepsilon_{j-1}(1 - |K_j|^2)$
- the AR coefficients $a_j[k]$ using the LDR

Finally, the AR coefficient vector $\mathbf{a}_P$ is available

Burg PSD Estimate

- The **Burg PSD estimate** with an AR-model of order P is given by

$$\hat{\Phi}_{xx}^{Burg,P}(e^{j\omega}) = \frac{\hat{\varepsilon}_P^{Burg}}{\left|1 + \sum_{k=1}^{P} a_P[k] \cdot e^{-j\omega k}\right|^2}$$

where the error variance estimate yields

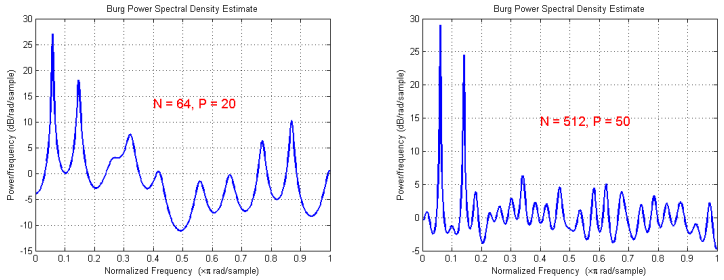
$$\hat{\varepsilon}_P^{Burg} = \varepsilon_0 \cdot \prod_{j=1}^{P} (1 - |K_j|^2)$$

Burg Method

- Since all reflection coefficients are less than unity, the Burg algorithm ensures a minimum-phase prediction filter, and therefore it always provides a stable all-pole model (AR-model)
- Burg PSD estimates are very close to the true values in the case of closely spaced sinusoids in signals with low noise levels, and provides high resolution for short data records
- The Burg PSD estimate is unbiased

Burg Method

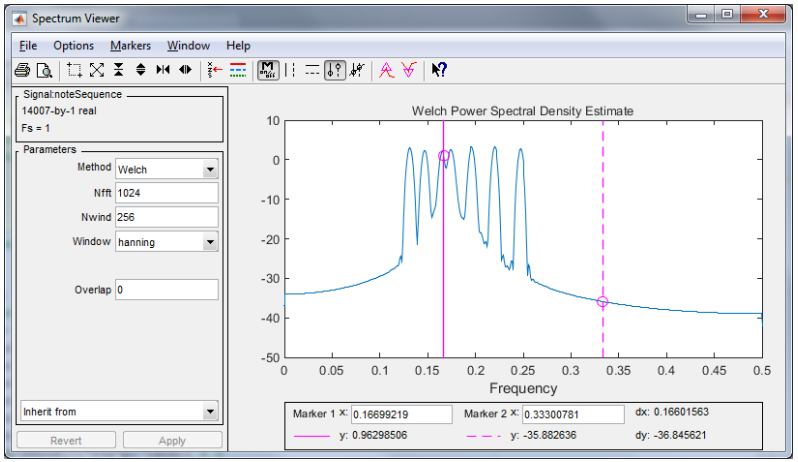
- Example: Noise-corrupted sinusoidal sequence of the previous example



- Good PSD estimate for short data length N
- Improved PSD estimate by increasing N and P

2.10 Spectral Analysis using MATLAB

- Spectrum Viewer (**sptool**)



Spectral Analysis using MATLAB

- Spectral Estimation Functions (SP Toolbox)

Method	Description	Function
Periodogram	poor estimates with low resolution	`periodogram`
Welch	averaged periodograms with better resolution than a single peridogram	`pwelch`
Yule-Walker AR	relatively poor estimates for short data records	`pyulear`
Covariance	better resolution than Yule-Walker estimate for short data records, unstable AR models possible	`pcov`
Burg	high resolution for short data records	`pburg`

Spectral Analysis using MATLAB

- System Objects (DSP System Toolbox)

`dsp.SpectrumAnalyzer`	Display frequency spectrum of time-domain signals
`dsp.SpectrumEstimator`	Estimate power spectrum or power density spectrum
`dsp.CrossSpectrumEstimator`	Estimate cross-spectral density
`dsp.TransferFunctionEstimator`	Estimate transfer function
`dsp.BurgAREstimator`	Estimate of autoregressive (AR) model parameters using Burg method
`dsp.BurgSpectrumEstimator`	Parametric spectral estimate using Burg method

Spectral Analysis using MATLAB

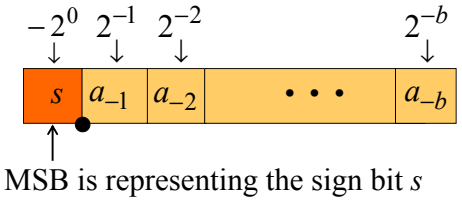
- Simulink Blocks (DSP System Toolbox)

Burg AR Estimator	Compute estimate of autoregressive (AR) model parameters using Burg method
Burg Method	Power spectral density estimate using Burg method
Covariance AR Estimator	Compute estimate of autoregressive (AR) model parameters using covariance method
Covariance Method	Power spectral density estimate using covariance method
Discrete Transfer Function Estimator	Compute estimate of frequency-domain transfer function of system
Magnitude FFT	Compute nonparametric estimate of spectrum using periodogram method
Modified Covariance AR Estimator	Compute estimate of autoregressive (AR) model parameters using modified covariance method
Modified Covariance Method	Power spectral density estimate using modified covariance method
Periodogram	Power spectral density or mean-square spectrum estimate using periodogram method
Short-Time FFT	Nonparametric estimate of spectrum using short-time, fast Fourier transform (FFT) method
Spectrum Analyzer	Display frequency spectrum of time-domain signals
Yule-Walker AR Estimator	Compute estimate of autoregressive (AR) model parameters using Yule-Walker method
Yule-Walker Method	Power spectral density estimate using Yule-Walker method

3. Finite Wordlength Effects

3.1 Quantization Characteristics

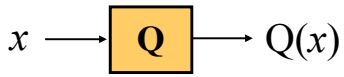
- Assume the two's complement representation of a number x with b+1 bit in the 1.f-format



- The number range or dynamic range of x is $-1 \le x < 1$, denoted as [-1,1)

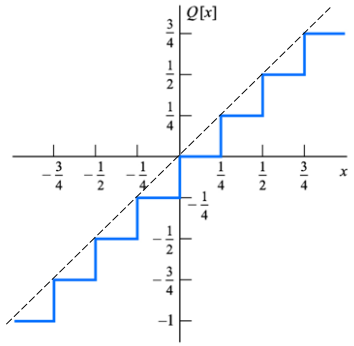
Quantization Characteristics

- Numbers are quantized in steps of $\delta = 2^{-b}$
- An original data x represented as a (β+1)-bit fraction (assuming β >> b) is converted into a (b+1)-bit fraction Q(x) either by truncation or rounding
- Block diagram representation for a quantizer with input x and output Q(x)

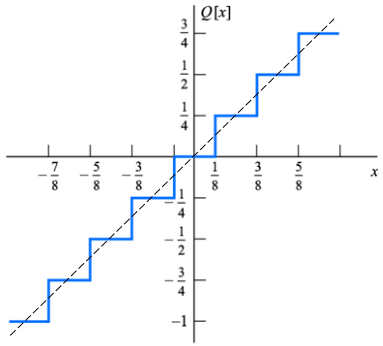


Quantization Characteristics

- Quantization characteristics for 3-bit (b = 2)



Truncation



Rounding

Quantization Characteristics

- The error range for quantized numbers depend on the quantization characteristic
- Range of truncation error $\varepsilon_t = \mathrm{Q}(x) - x$ is given by

$$-\delta < \varepsilon_t \le 0$$

- Range of rounding error $\varepsilon_r = \mathrm{Q}(x) - x$ is given by

$$-\frac{\delta}{2} < \varepsilon_r \le \frac{\delta}{2}$$

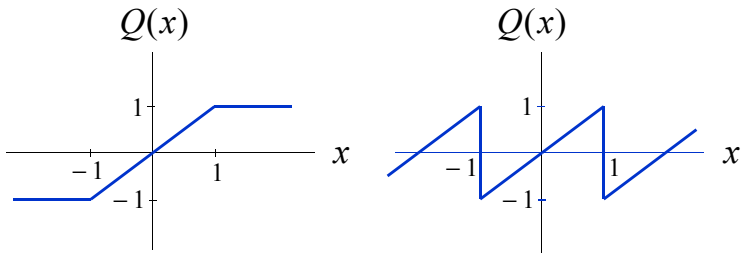
Note: $\delta = 2^{-b}$

Quantization Characteristics

- If a number x exceeds the permitted dynamic range, an overflow has occurred and the number x should be replaced with a number that belongs to the permitted dynamic range
- In practice we wish to completely avoid overflow or at least reducing the probability of overflow
- Next, we consider two common schemes to handle an overflow

Quantization Characteristics

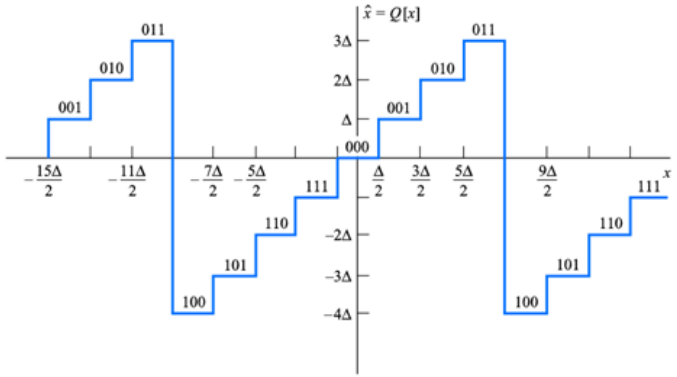
- If x exceeds [-1,1), the quantizer substitutes x with a number Q(x) within the permitted range using saturation or wrap overflow



Saturation overflow Two's-complement (wrap) overflow

Quantization Characteristics

- Example: 3-bit two's complement number x with rounding and wrap overflow characteristic



Quantization Characteristics

- Denote the differences between the quantized values $Q(x[n]) = \hat{x}[n]$ and the input sequence $x[n]$ as the quantization error sequence $e[n]$

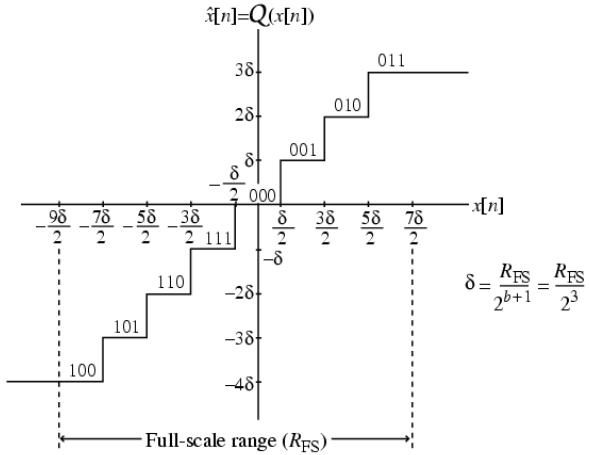
$$e[n] = Q(x[n]) - x[n] = \hat{x}[n] - x[n]$$

- The full-scale range R_{FS} for a $(b+1)$-bit quantizer employing rounding is given by

$$-(2^{b+1}+1)\frac{\delta}{2} < x[n] \le (2^{b+1}-1)\frac{\delta}{2}$$

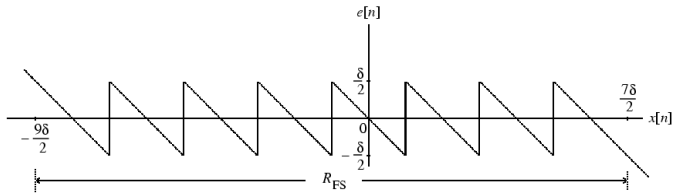
Quantization Characteristics

- Example: R_{FS} for a 3-bit (b = 2) quantizer



Quantization Characteristics

- Assume a sample exactly halfway between two levels is rounded up to the nearest higher level
- Then the quantization error $e[n]$, also called the granular noise, is bounded due to $-\frac{\delta}{2} < e[n] \le \frac{\delta}{2}$
- Example: Granular noise $e[n]$ for (b+1)=3

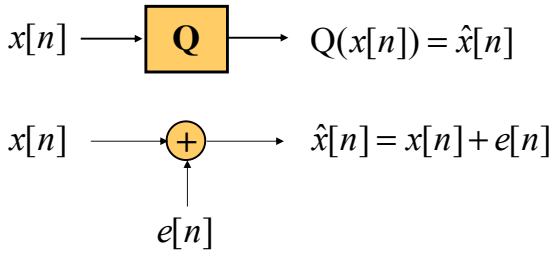


Quantization Characteristics

- When the input analog sample is outside of the full-scale range of the quantizer, the magnitude of the error $e[n]$ increases linearly with an increase in the magnitude of the input
- In such a situation, the quantization error is called the saturation or overload noise as the quantizer output is clipped to the maximum value $1-2^{-b}$ (if the input is positive) or the minimum value -1 (if the input is negative)

Quantization Noise Model

- We assume that the quantization error $e[n]$ is a random variable additive to the error-free input $x[n]$, i.e. the output of the quantizer is as follows

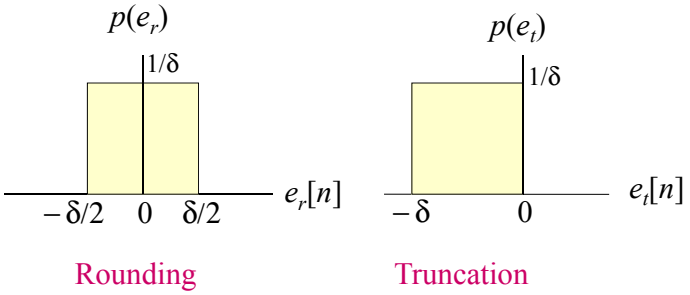


Quantization Noise Model

- **Assumptions:**
  1. We consider only granular noise assuming the quantization step size δ is very small
  2. The input $x[n]$ and the quantizer error $e[n]$ are uncorrelated
  3. The quantization error $e[n]$ is a wide-sense stationary process with uniform distribution for the amplitude

Quantization Noise Model

- The probability density functions for the rounding and truncation quantization errors $e_r[n]$ and $e_t[n]$ are given below



Quantization Noise Model

- Mean and variance of the quantization error in the case of rounding are given by

$$\mu_{er} = 0 \qquad \sigma_{er}^2 = \frac{\delta^2}{12}$$

- Mean and variance of the quantization error in the case of truncation are given by

$$\mu_{et} = -\frac{\delta}{2} \qquad \sigma_{et}^2 = \frac{\delta^2}{12}$$

Signal-to-Quantization Noise Ratio

- The signal-to-quantization noise ratio (*SQNR*) in dB is given by

$$SQNR = 10\log_{10}\left(\frac{\sigma_x^2}{\sigma_e^2}\right)$$

where σ_x^2 represents the input signal variance or signal power, and σ_e^2 the noise variance or quantization noise power

Signal-to-Quantization Noise Ratio

- For a (b+1)-bit quantizer the full-scale range is given by $R_{FS} = 2^{(b+1)}\delta$
- Hence, the quantization noise power results in

$$\sigma_e^2 = \frac{2^{-2b}(R_{FS})^2}{48}$$

arriving at

$$SQNR = 10\log_{10}\left(\frac{48\,\sigma_x^2}{2^{-2b}(R_{FS})^2}\right)$$

Signal-to-Quantization Noise Ratio

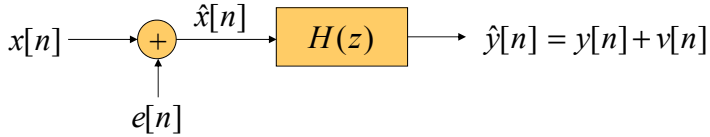
- Finally, we obtain the *SQNR* given by

$$SQNR = 6.02b + 16.81 - 20\log\left(\frac{R_{FS}}{\sigma_x}\right)$$

- This expression can be used to determine the minimum word length (the number of bits) to meet a specified *SQNR*
- Note that the *SQNR* increases by 6 dB for each bit added to the word length

3.2 Propagation of Input Quantization Noise to Digital Filter Output

- To determine the quantization noise at the filter output, we assume that the filter is implemented using infinite precision without internal noise sources
- The input quantization noise is propagated as follows



Propagation of Input Quantization Noise to Digital Filter Output

- The output noise power spectrum is due to the Wiener-Lee relation given by

$$\Phi_{vv}(e^{j\omega}) = \sigma_e^2 \left| H(e^{j\omega}) \right|^2$$

- The normalized output noise variance can be obtained by

$$\sigma_{v,n}^2 = \frac{\sigma_v^2}{\sigma_e^2} = \frac{1}{2\pi} \int_{-\pi}^{\pi} \left| H(e^{j\omega}) \right|^2 d\omega$$

Propagation of Input Quantization Noise to Digital Filter Output

- Parseval theorem states that

$$\frac{1}{2\pi} \int_{-\pi}^{\pi} \left| H(e^{j\omega}) \right|^2 d\omega = \sum_{n=-\infty}^{\infty} \left| h[n] \right|^2$$

- Thus, the normalized output noise variance is given by the squared L_2-norm of the impulse response sequence of the digital filter

$$\sigma_{v,n}^2 = \sum_{n=-\infty}^{\infty} \left| h[n] \right|^2 = \left\| h \right\|_2^2$$

Propagation of Input Quantization Noise to Digital Filter Output

- The function **`filternorm`** can be used to compute the normalized output noise variance
- Example: Normalized output noise variance at the output of a digital Butterworth filter

```
[b, a] = butter(5, 0.5);
L2 = filternorm(b, a);  % L2-norm
```

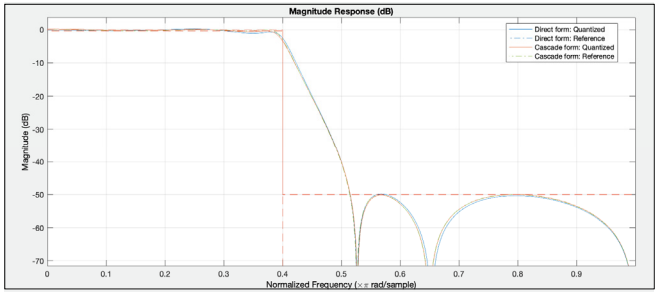
Result is $\sigma_{v,n}^2 = \left\| h \right\|_2^2 = 0.5$

3.3 Coefficient Quantization

- The transfer function $\hat{H}(z)$ of a digital filter with quantized coefficients is different from the desired transfer function $H(z)$
- Main effect of coefficient quantization is to move the poles and zeros to different locations from the original desired locations
- The actual frequency response is thus different from the desired frequency response

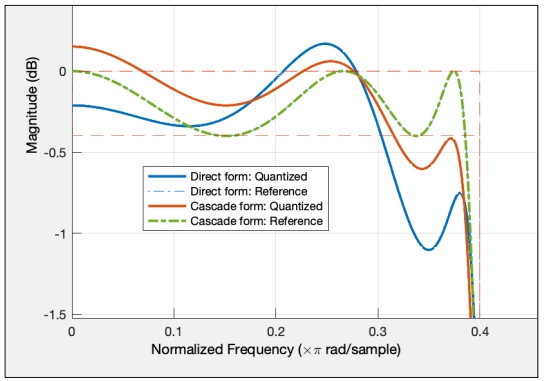
Coefficient Quantization Effects On a IIR Digital Filter

- Example: Gain of a 5-th order elliptic LP filter with coefficients truncated to 8-bit in direct form and in cascade form structure



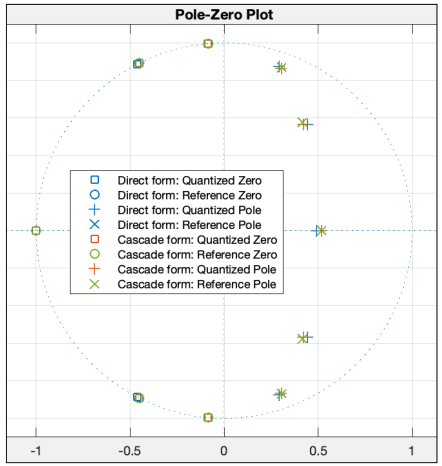
Coefficient Quantization Effects On a IIR Digital Filter

- Passband Details



Coefficient Quantization Effects On a IIR Digital Filter

- Pole-Zero Plot



Coefficient Quantization Effects On a IIR Digital Filter

- Effects in the passbands are higher passband ripples and smaller transition bands
- Effects in the stopbands are minimal
- Effects in the cascade form structure are not as severe as in the direct form structures
- Thus, higher-order IIR digital filters should be realized as cascade form structures

Coefficient Sensitivity Analysis

- Next, we analyze the sensitivity for the root displacements due to coefficient quantization
- Consider an N-th degree polynomial $A(z)$ with simple roots given by

$$A(z) = \sum_{i=0}^{N} a_i z^i = \prod_{k=1}^{N} (z - z_k)$$

with $a_N = 1$

- Roots z_k of $A(z)$ are given by $z_k = r_k e^{j\theta_k}$

Coefficient Sensitivity Analysis

- Effect of coefficient quantization is to change the polynomial coefficient from a_i to $a_i + \Delta a_i$
- Thus, the polynomial $A(z)$ after coefficient quantization becomes

$$\hat{A}(z) = \sum_{i=0}^{N} (a_i + \Delta a_i) z^i$$

$$= A(z) + \sum_{i=0}^{N} (\Delta a_i) z^i = \prod_{k=1}^{N} (z - \hat{z}_k)$$

Coefficient Sensitivity Analysis

- $\hat{z}_k$ denotes the roots of $\hat{A}(z)$, i.e. the new locations of the roots of $A(z)$ given by

$$\hat{z}_k = z_k + \Delta z_k = (r_k + \Delta r_k) e^{j(\theta_k + \Delta\theta_k)}$$

- If Δa_i is assumed to be very small, $e^{j\Delta\theta_k}$ can be expressed as a Taylor series

$$e^{j\Delta\theta_k} \cong 1 + j\Delta\theta_k$$

neglecting higher order terms

Coefficient Sensitivity Analysis

- Therefore, for very small changes Δa_i the new pole locations are given by

$$\hat{z}_k \cong (r_k + \Delta r_k)(1 + j\Delta\theta_k) e^{j\theta_k}$$

$$\cong r_k e^{j\theta_k} + (\Delta r_k + j r_k \Delta\theta_k) e^{j\theta_k}$$

- Then, we can write for the root displacement

$$\Delta z_k = \hat{z}_k - z_k \cong (\Delta r_k + j r_k \Delta\theta_k) e^{j\theta_k}$$

Coefficient Sensitivity Analysis

- Now we express $1/A(z)$ by its partial-fraction expansion as

$$\frac{1}{A(z)} = \sum_{k=1}^{N} \frac{\rho_k}{z - z_k}$$

where ρ_k is the residue of $1/A(z)$ at the pole $z = z_k$, i.e.

$$\rho_k = \left. \frac{(z - z_k)}{A(z)} \right|_{z=z_k} = R_k + jX_k$$

Coefficient Sensitivity Analysis

- If we assume that $\hat{z}_k$ is very close to z_k, then we can write

$$\frac{1}{A(\hat{z}_k)} \cong \frac{\rho_k}{\hat{z}_k - z_k}$$

or equivalently

$$\Delta z_k \cong \rho_k \cdot A(\hat{z}_k)$$

- But

$$\hat{A}(\hat{z}_k) = 0 = A(\hat{z}_k) + \sum_{i=0}^{N-1} (\Delta a_i)(\hat{z}_k)^i$$

Coefficient Sensitivity Analysis

- Therefore

$$\Delta z_k = -\rho_k \left\{ \sum_{i=0}^{N-1} (\Delta a_i)(\hat{z}_k)^i \right\} \cong -\rho_k \left\{ \sum_{i=0}^{N-1} (\Delta a_i)(z_k)^i \right\}$$

assuming that $\hat{z}_k$ is very close to z_k

- Rewriting the above equation we get

$$(\Delta r_k + j r_k \Delta\theta_k) e^{j\theta_k} = -(R_k + jX_k) \left\{ \sum_{i=0}^{N-1} (\Delta a_i)(z_k)^i \right\}$$

Coefficient Sensitivity Analysis

- Equating real and imaginary parts of the above we finally (after some algebra) arrive at

$$\Delta r_k = (-R_k \mathbf{P}_k + X_k \mathbf{Q}_k) \cdot \Delta\mathbf{A} = \mathbf{S}_a^{r_k} \cdot \Delta\mathbf{A}$$

$$\Delta\theta_k = -\frac{1}{r_k} (X_k \mathbf{P}_k + R_k \mathbf{Q}_k) \cdot \Delta\mathbf{A} = \mathbf{S}_a^{\theta_k} \cdot \Delta\mathbf{A}$$

where

$$\mathbf{P}_k = [\cos\theta_k \quad r_k \quad r_k^2 \cos\theta_k \quad \cdots \quad r_k^{N-1} \cos(N-2)\theta_k]$$

$$\mathbf{Q}_k = [-\sin\theta_k \quad 0 \quad r_k^2 \sin\theta_k \quad \cdots \quad r_k^{N-1} \sin(N-2)\theta_k]$$

$$\Delta\mathbf{A} = [\Delta a_0 \quad \Delta a_1 \quad \Delta a_2 \quad \cdots \quad \Delta a_{N-1}]^T$$

Coefficient Sensitivity Analysis

- $\mathbf{S}_a^{r_k}$ and $\mathbf{S}_a^{\theta_k}$ are the sensitivity vectors for the pole displacements given by

$$\mathbf{S}_a^{r_k} = -R_k\mathbf{P}_k + X_k\mathbf{Q}_k$$

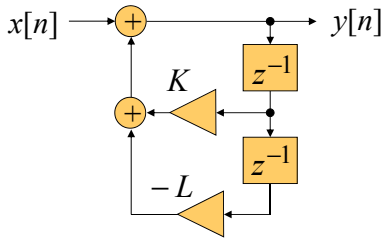
$$\mathbf{S}_a^{\theta_k} = -(X_k\mathbf{P}_k + R_k\mathbf{Q}_k)$$

- These vectors can be used to calculate the pole displacements due to coefficient changes $\Delta\mathbf{A}$
- Note that the sensitivity vectors are for pole displacements of direct form structures

Coefficient Sensitivity Analysis

- Example: Consider the direct form structure of the 2nd-order transfer function given by

$$H(z) = \frac{z^2}{z^2 - Kz + L} = \frac{z^2}{A(z)}$$



Coefficient Sensitivity Analysis

- Assume $A(z) = (z - z_1)(z - z_2)$, where the two poles are given by $z_1 = re^{j\theta}$ and $z_2 = re^{-j\theta}$
- The vector $\Delta\mathbf{A}$ is given by

$$\Delta\mathbf{A} = [\Delta L \quad -\Delta K]^T$$

- The residues are $\rho_1 = \left.\frac{z - z_1}{A(z)}\right|_{z=z_1} = -\frac{j}{2r\sin\theta} = -\rho_2$

  and thus $R_1 = R_2 = 0$, $X_1 = -\frac{1}{2r\sin\theta} = -X_2$
- Therefore

$$\mathbf{P}_1 = [\cos\theta \quad r] \qquad \mathbf{Q}_1 = [-\sin\theta \quad 0]$$

Coefficient Sensitivity Analysis

- Substituting these values we finally obtain

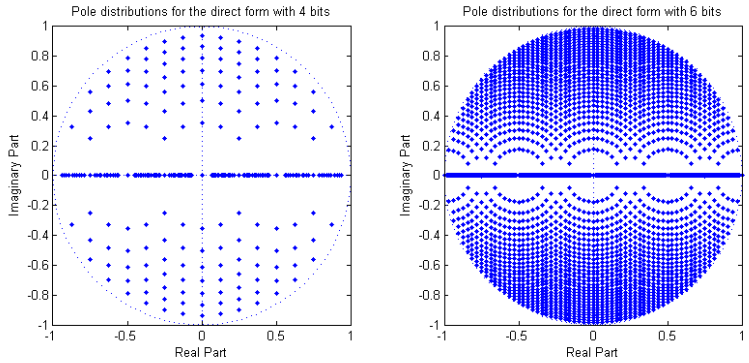
$$\Delta r = (X_1\mathbf{Q}_1)\cdot\Delta\mathbf{A} = \frac{\Delta L}{2r}$$

$$\Delta\theta = -\frac{1}{r}(X_1\mathbf{P}_1)\cdot\Delta\mathbf{A} = \frac{\Delta L}{2r^2\tan\theta} - \frac{\Delta K}{2r\sin\theta}$$

- Thus, the direct form structure of the 2nd-order transfer function is highly sensitive to coefficient quantizations for poles close to $\theta = 0$ or for poles close to $\theta = \pi$

Coefficient Sensitivity Analysis

- Pole distributions for a 2nd-order direct form



Coefficient Sensitivity Analysis

- Consider an arbitrary digital filter structure with R multipliers given by c_k, $k = 1, 2, \ldots, R$
- The multiplier coefficients c_k are functions of the coefficients a_i of the polynomial $A(z)$
- Thus, when c_k changes into $c_k + \Delta c_k$ due to coefficient quantization, the change Δa_i can be expressed as

$$\Delta a_i = \sum_{k=1}^{R} \frac{\partial a_i}{\partial c_k} \Delta c_k, \qquad i = 1, 2, \ldots, N-1$$

Coefficient Sensitivity Analysis

- In matrix form we have $\Delta \mathbf{A} = \mathbf{C} \cdot \Delta \mathbf{c}$, where

$$\mathbf{C} = \begin{bmatrix} \frac{\partial a_0}{\partial c_1} & \frac{\partial a_0}{\partial c_2} & \cdots & \frac{\partial a_0}{\partial c_R} \\ \frac{\partial a_1}{\partial c_1} & \frac{\partial a_1}{\partial c_2} & \cdots & \frac{\partial a_1}{\partial c_R} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial a_{N-1}}{\partial c_1} & \frac{\partial a_{N-1}}{\partial c_2} & \cdots & \frac{\partial a_{N-1}}{\partial c_R} \end{bmatrix}$$

$$\Delta \mathbf{c} = [\Delta c_1 \quad \Delta c_2 \quad \Delta c_3 \quad \cdots \quad \Delta c_R]^T$$

Coefficient Sensitivity Analysis

- Here, the root displacements are given by

$$\Delta \mathbf{r}_k = \mathbf{S}_a^{r_k} \cdot \mathbf{C} \cdot \Delta \mathbf{c}$$

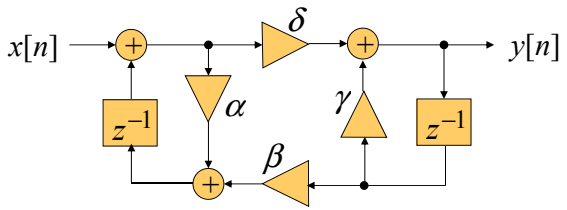
$$\Delta \boldsymbol{\theta}_k = \mathbf{S}_a^{\theta_k} \cdot \mathbf{C} \cdot \Delta \mathbf{c}$$

  where $\mathbf{S}_a^{r_k}$ and $\mathbf{S}_a^{\theta_k}$ are the sensitivity vectors of the direct form structure
- The matrix $\mathbf{C}$ depends on the filter structure and needs to be computed only once

Coefficient Sensitivity Analysis

- Example: Consider the 2nd-order coupled form structure with a transfer function given by

$$H(z)=\frac{\gamma z^2}{z^2-(\alpha+\delta)z+(\alpha\delta-\beta\gamma)}$$



Coefficient Sensitivity Analysis

- If $\alpha=\delta=r\cos\theta$ and $\beta=-\gamma=r\sin\theta$, then the transfer function becomes

$$H(z)=\frac{\gamma z^2}{z^2-2r\cos\theta z+r^2}$$

- Comparing the denominator with that of the transfer function of the direct form structure we obtain

$$K=\alpha+\delta=2\alpha$$

$$L=\alpha\delta-\beta\gamma=\alpha^2+\beta^2=r^2$$

Coefficient Sensitivity Analysis

- Taking the partial derivatives of the last two equations we get

$$\begin{bmatrix}\Delta L\\ \Delta K\end{bmatrix}=\begin{bmatrix}2r\cos\theta & 2r\sin\theta\\ 2 & 0\end{bmatrix}\begin{bmatrix}\Delta\alpha\\ \Delta\beta\end{bmatrix}$$

- Substituting the results of the direct form structure we arrive at

$$\begin{bmatrix}\Delta r\\ \Delta\theta\end{bmatrix}=\begin{bmatrix}\frac{1}{2r} & 0\\ \frac{1}{2r^2\tan\theta} & -\frac{1}{2r\sin\theta}\end{bmatrix}\begin{bmatrix}2r\cos\theta & 2r\sin\theta\\ 2 & 0\end{bmatrix}\begin{bmatrix}\Delta\alpha\\ \Delta\beta\end{bmatrix}$$

Coefficient Sensitivity Analysis

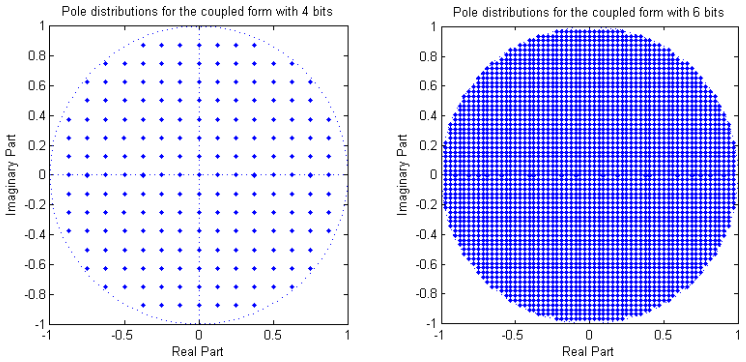
- Finally, the root displacements of the coupled form structure due to coefficient quantization are given by

$$\begin{bmatrix}\Delta r\\ \Delta\theta\end{bmatrix}=\begin{bmatrix}\cos\theta & \sin\theta\\ \frac{-\sin\theta}{r} & \frac{\cos\theta}{r}\end{bmatrix}\cdot\begin{bmatrix}\Delta\alpha\\ \Delta\beta\end{bmatrix}$$

- The coupled form structure is less sensitive to multiplier coefficient quantization than the direct form structure

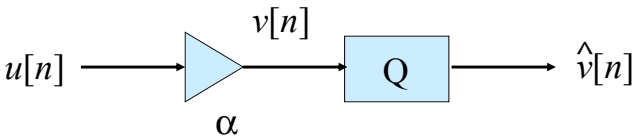
Coefficient Sensitivity Analysis

- Pole distributions for a 2nd-order coupled form



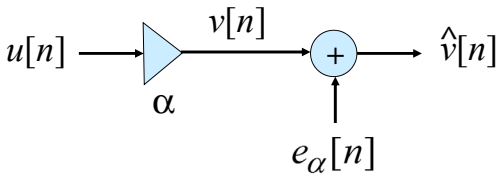
3.4 Analysis of Product Round-Off Errors

- In a fixed-point implementation of a digital filter structure we quantize only products or a sum of products
- A multiplier with a quantizer at its output is represented as shown below



Analysis of Product Round-Off Errors

- The product round-off error model is shown below



- The output $v[n]$ of the ideal multiplier is quantized to a value $\hat{v}[n]$, where

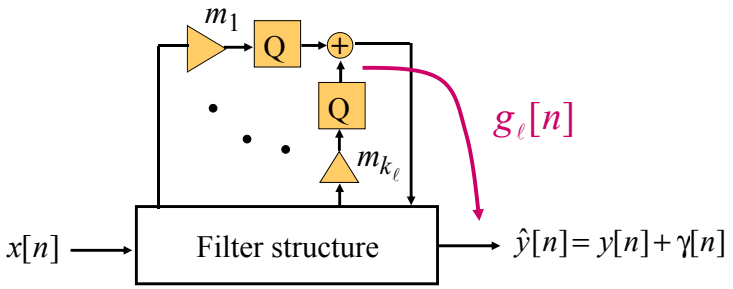
$$\hat{v}[n] = v[n] + e_\alpha[n]$$

Analysis of Product Round-Off Errors

- **Assumptions:**
  1. The product round-off noise error $e_\alpha[n]$ is a wide-sense stationary white noise process
  2. $e_\alpha[n]$ has a uniform probability density function in the quantization interval
  3. $e_\alpha[n]$ is uncorrelated with the input $v[n]$ to the quantizer, all other round-off noise sources, and the input signal $u[n]$

Analysis of Product Round-Off Errors

- Model for a filter with a quantizer after each multiplier (quantization before accumulation)



Analysis of Product Round-Off Errors

- The model illustrates a single internal part of a filter structure picked the $\ell - th$ adder to add up k_ℓ quantized products
- $g_\ell[n]$ represents the impulse response from the input of the $\ell - th$ adder to the digital filter output
- $G_\ell(z) = Z\{g_\ell[n]\}$ is referred to as the noise transfer function for the quantization noise

Analysis of Product Round-Off Errors

- If σ_o^2 denotes the product round-off variance at the output of a single multiplier, the total product round-off variance at the output of the $\ell - th$ adder is simply $k_\ell \sigma_o^2$
- The product round-off variance at the digital filter output caused by $k_\ell \sigma_o^2$ is given by

$$\sigma_{\gamma,\ell}^2 = k_\ell \sigma_o^2 \frac{1}{2\pi} \int_{-\pi}^{\pi} \left| G_\ell(e^{j\omega}) \right|^2 d\omega = k_\ell \sigma_o^2 \left\| G_\ell(e^{j\omega}) \right\|_2^2$$

Analysis of Product Round-Off Errors

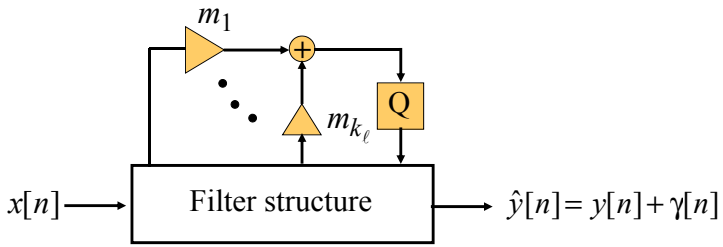
- For N adders, the total output noise power due to all product round-offs is given by

$$\sigma_\gamma^2 = \sigma_o^2 \sum_{\ell=1}^{N} k_\ell \frac{1}{2\pi} \int_{-\pi}^{\pi} \left| G_\ell(e^{j\omega}) \right|^2 d\omega = \sigma_o^2 \sum_{\ell=1}^{N} k_\ell \left\| G_\ell(e^{j\omega}) \right\|_2^2$$

- However, most DSP chips have accumulators with 40 or more bits
- Then, quantization is performed after the accumulation of all products

Analysis of Product Round-Off Errors

- Model for a filter with a single quantizer after the accumulation of all products



Analysis of Product Round-Off Errors

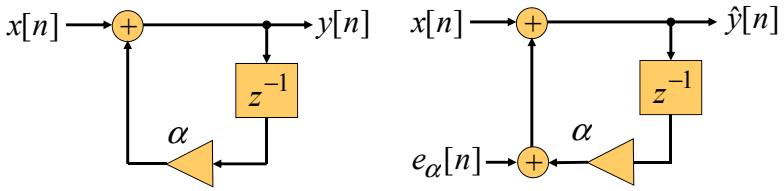
- If the sum of all products is quantized, the resulting total output noise power is given by

$$\sigma_\gamma^2 = \sigma_o^2 \sum_{\ell=1}^{N} \frac{1}{2\pi} \int_{-\pi}^{\pi} \left| G_\ell(e^{j\omega}) \right|^2 d\omega = \sigma_o^2 \sum_{\ell=1}^{N} \left\| G_\ell(e^{j\omega}) \right\|_2^2$$

- Hence, quantization after accumulation results in a considerably lower product round-off noise at the digital filter output

Analysis of Product Round-Off Errors

- Example: Product round-off noise power analysis of a first-order IIR transfer function



$$H(z) = \frac{z}{z-\alpha}$$

Analysis of Product Round-Off Errors

- From the noise analysis model it can be seen that the noise transfer function $G_\alpha(z)$ is the same as the filter transfer function $H(z)$, i.e.,

$$G_\alpha(z) = H(z) = \frac{z}{z-\alpha}$$

- Thus, the output noise variance due to the product round-off is same as that due to input quantization

$$\sigma_\gamma^2 = \sigma_o^2 \sum_{n=0}^{\infty} \left| h[n] \right|^2 = \sigma_o^2 \sum_{n=0}^{\infty} \left| \alpha^n \right|^2 = \frac{\sigma_o^2}{1-\alpha^2}$$

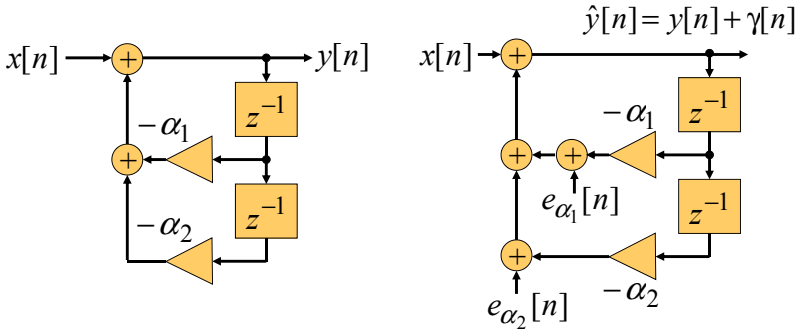
Analysis of Product Round-Off Errors

- The quantity $\sigma^2_{\gamma,n} = \sigma^2_\gamma / \sigma^2_o$ is the noise gain or the normalized round-off noise power
- Example: Product round-off noise power analysis of a causal 2nd-order IIR transfer function in direct form II structure

$$H(z) = \frac{Y(z)}{X(z)} = \frac{z^2}{z^2 + \alpha_1 z + \alpha_2}$$

Analysis of Product Round-Off Errors

- The filter structure and the model for product round-off noise power analysis is shown below



Analysis of Product Round-Off Errors

- $H(z)$ and the noise transfer functions $G_{\alpha_1}(z)$ and $G_{\alpha_2}(z)$ are equal
- The normalized round-off noise variance is given by

$$\sigma^2_{\gamma,n} = 2 \cdot \left(\frac{1}{2\pi} \int_{-\pi}^{\pi} \left| H(e^{j\omega}) \right|^2 d\omega \right)$$

$$= 2 \cdot \left(\frac{1+\alpha_2}{1-\alpha_2} \right) \left(\frac{1}{1+\alpha_2^2+2\alpha_2-\alpha_1^2} \right)$$

Analysis of Product Round-Off Errors

- In terms of the pole locations $re^{\pm j\theta}$, we have $\alpha_1 = -2r\cos\theta$ and $\alpha_2 = r^2$
- Substituting these values in the expression for $\sigma^2_{\gamma,n}$ we obtain

$$\sigma^2_{\gamma,n} = 2 \cdot \left(\frac{1+r^2}{1-r^2} \right) \cdot \left(\frac{1}{1+r^4-2r^2\cos 2\theta} \right)$$

Analysis of Product Round-Off Errors

- If the poles are close to the unit circle, i.e., $r = 1 - \varepsilon$, where ε is a very small positive number, we can express $\sigma^2_{\gamma,n}$ as

$$\sigma^2_{\gamma,n} \cong \frac{1}{2\sin^2\theta}\left(\frac{1-\varepsilon}{\varepsilon}\right)\left(\frac{1}{1-2\varepsilon}\right)$$

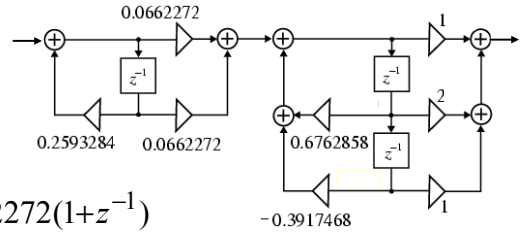
- Thus, as the poles get closer to the unit circle, the total output noise power increases rapidly

Computation of Product Round-Off Noise Power using MATLAB

- The method **`noisepsd`** of **dfilt** objects computes the power spectral density (PSD) of filter outputs due to product round-off noise
- Use **dfilt.<strucname>** to select a structure
- Set the fixed-point properties in the object
- The average power of the output noise (the integral of the PSD) can be computed using **`avgpower`**, a method of **dspdata** objects

Computation of Product Round-Off Noise Power using MATLAB

- Example: 3rd-order IIR filter in cascade form



$$H_1(z) = \frac{0.0662272(1+z^{-1})}{1-0.2593284z^{-1}}$$

$$H_2(z) = \frac{1+2z^{-1}+z^{-2}}{1-0.6762858z^{-1}+0.3917468z^{-2}}$$

Computation of Product Round-Off Noise Power using MATLAB

- MATLAB code snippets:

```
Hd = dfilt.df2sos(sos, g);
set(Hd, 'Arithmetic', 'fixed' , …
    'CoeffWordLength', 13, …
    'AccumWordLength', 28, …);
npsd = noisepsd(Hd);
avgnpow = avgpower(npsd);
avgnpowdb = pow2db(avgnpow); % in dB
```

Computation of Product Round-Off Noise Power using MATLAB

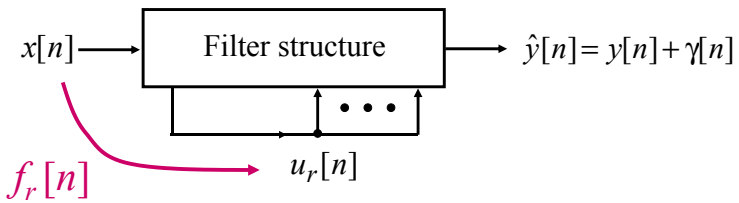
- Note the properties in the dfilt object to set the word lengths used for the input, the states, the coefficients, and the accumulator
- In the example, the average product round-off noise power using quantization before and after accumulation is -40.7 dB and -52.8 dB, respectively

3.5 Scaling to Prevent Overflows

- Assume that all fixed-point numbers are represented in two's complement 1.*f* format, i.e. within the number range given by $[-1, +1)$
- **Objective of scaling:**
  Ensure that each internal node variable $u_r[n]$ is bounded by unity, i.e. $|u_r[n]| < 1$
- Different scaling rules are applied to avoid or to reduce overflows

Scaling to Prevent Overflows

- Filter structure with the picked internal r-th node to be scaled and the impulse response $f_r[n]$ of the scaling transfer function from the filter input to the r-th node



Scaling to Prevent Overflows

- First, we consider a scaling rule to guarantee no overflows
- From the linear convolution

$$u_r[n] = \sum_{r=-\infty}^{\infty} f_r[k]\,x[n-k]$$

using the scaling impulse response $f_r[k]$ and assuming $|x[n]| \le 1$, we obtain

$$|u_r[n]| = \left| \sum_{r=-\infty}^{\infty} f_r[k]\,x[n-k] \right| \le \sum_{k=-\infty}^{\infty} |f_r[k]|$$

Scaling to Prevent Overflows

- Thus the condition $|u_r[n]| < 1$ is satisfied if
$$\sum_{k=-\infty}^{\infty} |f_r[k]| = \|f_r\|_1 < 1$$
- The condition is both necessary and sufficient to guarantee no overflow
- If the condition is not satisfied for all nodes, the input can be scaled with K as follows
$$K \le \frac{1}{\max_r \sum_{k=-\infty}^{\infty} |f_r[k]|} = \frac{1}{\max_r \|f_r\|_1}$$

Scaling to Prevent Overflows

- The above scaling method is for the worst-case scenario, and the scaling factor is most over top
- The scaling does not fully utilize the dynamic range resulting in a significant reduction in the signal-to-noise ratio at the filter output
- In addition, it is difficult to compute the value of the scaling constant K analytically

Scaling to Prevent Overflows

- More practical easy to use scaling rules can be derived using **L_p-norms** for the scaling transfer function $F(e^{j\omega})$
- The **L_p-norm** $\|F\|_p$ $(p \ge 1)$ of $F(e^{j\omega})$ is defined as
$$\|F\|_p \triangleq \left(\frac{1}{2\pi} \int_{-\pi}^{\pi} |F(e^{j\omega})|^p \, d\omega \right)^{1/p}$$
- In practice, three norms are particularly suited

Scaling to Prevent Overflows

- $\|F\|_1$ is the **L_1-norm**, the mean absolute value of $F(e^{j\omega})$ over $[-\pi, \pi]$
- $\|F\|_2$ is the **L_2-norm**, the root mean square (RMS) value of $F(e^{j\omega})$ over $[-\pi, \pi]$
- $\lim_{p\to\infty} \|F\|_p$ is the **L_∞-norm**, the peak value of $F(e^{j\omega})$ over $[-\pi, \pi]$, i.e.
$$\|F\|_\infty = \max_{-\pi \le \omega \le \pi} |F(e^{j\omega})|$$

Scaling to Prevent Overflows

- A **general scaling rule** is obtained using Hölder's inequality given by

$$|u_r[n]| \le \|F_r\|_p \cdot \|X\|_q$$

for all $p, q \ge 1$ satisfying $\frac{1}{p} + \frac{1}{q} = 1$

- **L_∞-scaling** is obtained for $p = \infty$ and $q = 1$
- **L_2-scaling** is obtained for $p = 2$ and $q = 2$
- **L_1-scaling** is obtained for $p = 1$ and $q = \infty$

Scaling to Prevent Overflows

- First, we consider the scaling rule using the **L_2-norm**
- Applying the Hölder's inequality to

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

we get the scaling rule

$$|u_r[n]| \le \left\|F_r(e^{j\omega})\right\|_2 \cdot \left\|X(e^{j\omega})\right\|_2 = \|F_r\|_2 \cdot \|X\|_2$$

Scaling to Prevent Overflows

- Thus, if the filter input has finite energy bounded by unity, i.e., $\|X\|_2 \le 1$, the filter can be scaled such that the **L_2-norm** of all scaling transfer functions is bounded by unity, i.e.,

$$\|F_r\|_2 \le 1, \quad r = 1, 2, \ldots, R$$

- This norm is easy to compute and very often used for finite energy inputs, however it is not applicable for power signals

Scaling to Prevent Overflows

- The scaling rule with the **L_∞-norm is** given by

$$|u_r[n]| \le \left\|F_r(e^{j\omega})\right\|_\infty \cdot \left\|X(e^{j\omega})\right\|_1 = \|F_r\|_\infty \cdot \|X\|_1$$

- This scaling rule is rarely used since $\|X\|_1 \le 1$ is required, i.e. the mean value of $X(e^{j\omega})$ must be bounded by unity
- It can be used for sinusoidal and narrowband signals (also known as harmonic scaling)

Scaling to Prevent Overflows

- The norms satisfy $\|F_r\|_2 \le \|F_r\|_\infty \le \|f_r\|_1$
- Thus, the scaling factors K are related as given below

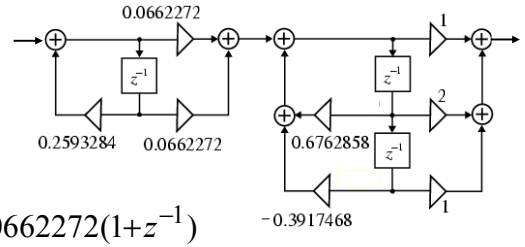
$$K \le K_1 = \frac{1}{\max_r \|f_r\|_1}$$

$$K_1 \le K_\infty = \frac{1}{\max_r \|F_r\|_\infty}$$

$$K_\infty \le K_2 = \frac{1}{\max_r \|F_r\|_2}$$

Scaling to Prevent Overflows

- Example: Norms for an unscaled 3rd-order transfer function in cascade structure



$$H_1(z) = \frac{0.0662272(1+z^{-1})}{1-0.2593284z^{-1}}$$

$$H_2(z) = \frac{1+2z^{-1}+z^{-2}}{1-0.6762858z^{-1}+0.3917468z^{-2}}$$

Scaling to Prevent Overflows

- MATLAB Code snippets:

```
Hd = dfilt.df2sos(sos, g);
set(Hd, …
    'Arithmetic', 'fixed' , …);
l1 = norm(Hd, 'l1');
L2 = norm(Hd, 'L2');
Linf = norm(Hd, 'Linf');
```

- Results:

$\|h\|_1 = 1.3094$

$\|H\|_\infty = 1.0026$

$\|h\|_2 = 0.5880$

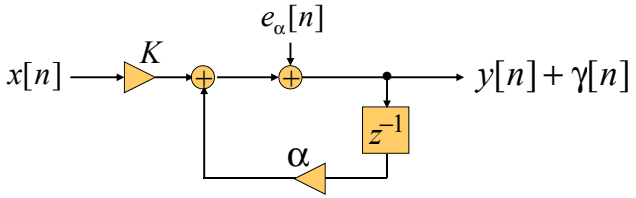
$\|H\|_2 = 0.5880$

$\|H\|_1 = 0.4115$

$\|h\|_\infty = 0.4061$

3.6 Tradeoff between Scaling and Product Round-off Noise

- Consider the first-order filter structure with scaled input and round-off noise model



$$F(z) = H(z) = \frac{1}{1-\alpha z^{-1}},\ f[n] = h[n] = \boldsymbol{Z}^{-1}\{H(z)\} = \alpha^n \mu[n]$$

Tradeoff between Scaling and Product Round-off Noise

- The product round-off noise variance at the filter output assuming a (b+1)-bit quantizer is given by

$$\sigma_\gamma^2 = \sigma_o^2 \sum_{n=0}^{\infty} |h[n]|^2 = \frac{2^{-2b}}{12} \sum_{n=0}^{\infty} |\alpha^n|^2 = \frac{2^{-2b}}{12(1-\alpha^2)}$$

- If the pole α is close to the unit circle, then the high gain of the filter structure means that we have to prevent overflows

Tradeoff between Scaling and Product Round-off Noise

- To guarantee no overflow, we choose K as

$$K = \frac{1}{\sum_{n=0}^{\infty} |f[n]|} = 1-\alpha$$

- Hence, the scaled input $x[n]$ satisfies the relation $-(1-\alpha) \le x[n] < (1-\alpha)$
- Now that the filter structure is properly scaled, we determine the signal to noise ratio

Tradeoff between Scaling and Product Round-off Noise

- We assume $x[n]$ to be uniformly distributed between -1 and $+1$ with the scaled signal average power given by

$$\sigma_x^2 = K^2 \frac{\delta^2}{12} = (1-\alpha)^2 \frac{2^2}{12} = \frac{(1-\alpha)^2}{3}$$

- Then, the signal power σ_y^2 at the filter output is given by

$$\sigma_y^2 = \sigma_x^2 \left(\sum_{n=0}^{\infty} |h[n]|^2 \right) = \frac{(1-\alpha)^2}{3(1-\alpha^2)}$$

Tradeoff between Scaling and Product Round-off Noise

- Hence the *SNR* at the filter output is equal to

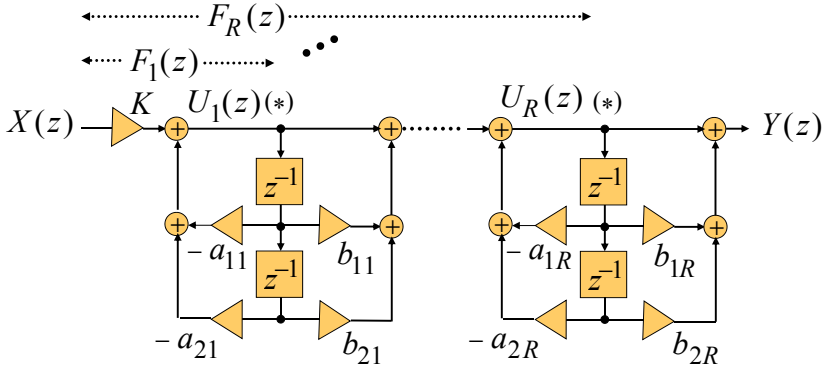
or in dB $$SNR = \frac{\sigma_\gamma^2}{\sigma_y^2} = 2^{2(b+1)}(1-\alpha)^2$$

$$SNR = 6.02(b+1) + 20\log_{10}(1-\alpha)\ dB$$

- Since the pole α is inside the unit circle, the second term is negative
- The scaling has reduced the dynamic range of the input thereby lowering the *SNR*

3.7 Scaling of IIR Digital Filters in Cascade Form

- Consider the unscaled cascade structure with R 2nd-order sections realized in direct form II



Scaling of IIR Digital Filters in Cascade Form

- The transfer function is given by

$$H(z) = K\prod_{\ell=1}^{R} H_\ell(z)$$

where

$$H_\ell(z) = \frac{B_\ell(z)}{A_\ell(z)} = \frac{1+b_{1\ell}z^{-1}+b_{2\ell}z^{-2}}{1+a_{1\ell}z^{-1}+a_{2\ell}z^{-2}}$$

- The scaling transfer functions are given by

$$F_1(z) = \frac{K}{A_1(z)};\; F_r(z) = \frac{K}{A_r(z)}\prod_{\ell=1}^{r-1} H_\ell(z);\;\; r = 2, 3, \ldots, R$$

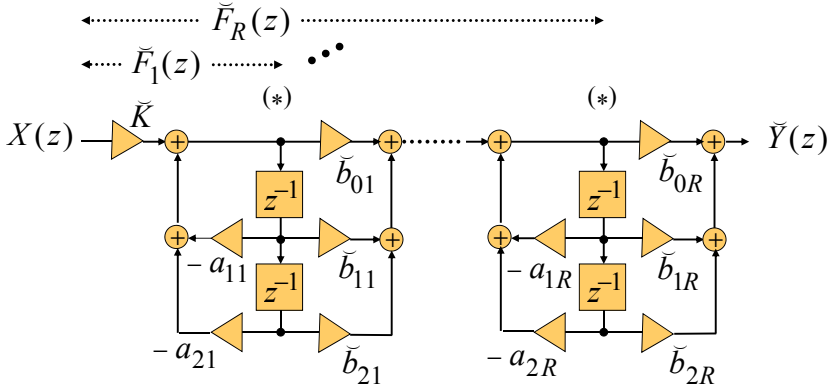
Scaling of IIR Digital Filters in Cascade Form

- The nodes to be scaled are marked by (*)
- All scaling factors can be absorbed by the multiplier K for input scaling and the feed-forward multipliers $b_{\ell r}$ in each 2nd-order section choosing

$$\breve{K} = \beta_0 K;\quad \breve{b}_{\ell r} = \beta_r b_{\ell r}$$
$$\ell = 0,1,2;\; r = 1,2,\ldots,R$$

Scaling of IIR Digital Filters in Cascade Form

- The scaled filter structure is shown below



Scaling of IIR Digital Filters in Cascade Form

- From the scaled structure it can be seen that

$$\breve{F}_1(z)=\frac{\breve{K}}{A_1(z)};\ \breve{F}_r(z)=\frac{\breve{K}}{A_r(z)}\prod_{\ell=1}^{r-1}\breve{H}_\ell(z);\ r=2,3,\ldots,R$$

where $\breve{H}_\ell(z)=\dfrac{\breve{b}_{0\ell}+\breve{b}_{1\ell}z^{-1}+\breve{b}_{2\ell}z^{-2}}{1+a_{1\ell}z^{-1}+a_{2\ell}z^{-2}}$

- Note that the scaling has introduced a new multiplier $\breve{b}_{0\ell}$ in each 2nd-order section

Scaling of IIR Digital Filters in Cascade Form

- For $r = 2, 3, \ldots, R$ the scaled scaling transfer functions can be expressed as

$$\breve{F}_r(z)=\frac{\beta_0 K}{A_r(z)}\prod_{\ell=1}^{r-1}\beta_\ell H_\ell(z)=\left(\prod_{\ell=0}^{r-1}\beta_\ell\right)F_r(z)$$

- The scaled filter transfer function is given by the product of all scaled sections, i.e.

$$\breve{H}(z)=\beta_0 K\prod_{\ell=1}^{R}\beta_\ell H_\ell(z)=\left(\prod_{\ell=0}^{R}\beta_\ell\right)H(z)$$

Scaling of IIR Digital Filters in Cascade Form

- Denote the L_p-norms of the unscaled transfer functions as

$$\|F_r\|_p=k_r;\ r=1,2,\ldots,R$$

$$\|H\|_p=k_{R+1}$$

- Using the scaling constants β_r, we require for the L_p-norms of the scaled transfer functions

$$\|\breve{F}_r\|_p=\left(\prod_{\ell=0}^{r-1}\beta_\ell\right)\cdot\|F_r\|_p=\left(\prod_{\ell=0}^{r-1}\beta_\ell\right)\cdot k_r=1,\ r=1,2,\ldots,R$$

Scaling of IIR Digital Filters in Cascade Form

and

$$\|\breve{H}\|_p=\left(\prod_{\ell=0}^{R}\beta_\ell\right)\cdot\|H\|_p=\left(\prod_{\ell=0}^{R}\beta_\ell\right)\cdot k_{R+1}=1$$

- Solving the above we obtain for the required scaling constants

$$\beta_0=\frac{1}{k_1},\ \beta_r=\frac{k_r}{k_{r+1}},\ r=1,2,\ldots,R$$

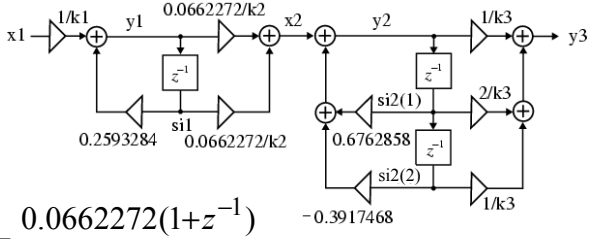
Scaling of IIR Digital Filters in Cascade Form

Scaling Procedure

- First, compute the L_p-norm $\|F_1\|_p$ of the first section and divide the input by $k_1 = \|F_1\|_p$
- Next, compute the L_p-norm $\|\breve{F}_2\|_p$ of the second section and divide the feedforward multipliers of the first section by $k_2 = \|\breve{F}_2\|_p$
- Continue this procedure for all sections until the last section has been scaled by $k_R = \|\breve{F}_R\|_p$

Scaling by Structure Simulation

- Example: L2-Scaling of a 3rd-order cascade form using structure simulation



$$H_1(z) = \frac{0.0662272(1+z^{-1})}{1-0.2593284z^{-1}}$$

$$H_2(z) = \frac{1+2z^{-1}+z^{-2}}{1-0.6762858z^{-1}+0.3917468z^{-2}}$$

Scaling by Structure Simulation

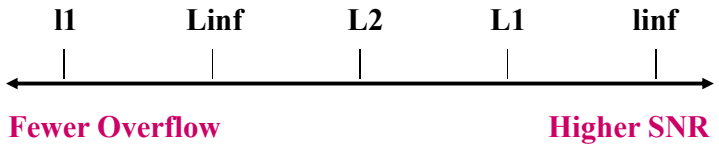
```
% L2-Scaling of a 3rd-order Cascade Form
k1 = 1; k2 = 1; k3 = 1;
x1 = 1/k1; si1 = 0; si2 = [0 0];
varnew = 0; delta = 1;
while delta > 0.000001
  y1 = 0.2593284*si1 + x1;
  x2 = (0.0662272/k2)*(y1+si1);
  si1 = y1;
  y2 = 0.6762858*si2(1) - 0.3917468*si2(2) + x2;
  y3 = (y2 + 2*si2(1) + si2(2))/k3;
  si2(2) = si2(1); si2(1) = y2;
  varold = varnew;
  varnew = varnew + y1^2;          % first run
  % varnew = varnew + y2^2;        % second run
  % varnew = varnew + y3^2;        % third run
  delta = varnew - varold; x1 = 0;
end
disp('L2-norm squared = '); disp(varnew);
```


Scaling by Structure Simulation

- The first run is with initialization constants $k1 = k2 = k3 = 1$ and choosing the output $y1$, resulting in the square of the L_2-norm at the output $y1$
- For the second run, we set $k1$ to the L_2-norm at $y1$, where $k2 = k3 = 1$ and $y2$ is choosen, resulting in the square of the L_2-norm at the output $y2$ and therefore the scaling factor $k2$
- In the third run we obtain scaling factor $k3$

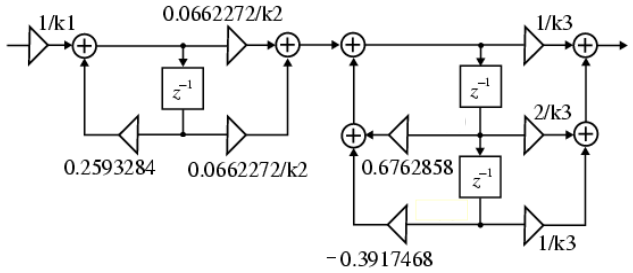
Scaling by the Scale Method

- The method **`scale`** of **dfilt** objects can be used to scale the cascade filter form structures
- It allows to select either the norms 'l1', 'l2', and 'linf', or the norms 'L1', 'L2', and 'Linf'
- Note that a more stringent norm reduces the number of overflows, but has a worse SNR



Scaling by the Scale Method

- Example: 3rd-order IIR filter in cascade form



- The three scaling factors k1, k2, and k3 are determined using different scaling rules

Scaling by the Scale Method

- Assume the filter input $x[n]$ to be uniformly distributed between -1 and $+1$
- Simulation of the scaled filter structures for 10,000 input samples results in

  1133 overflows using L_2-scaling

  3 overflows using L_∞-scaling

  0 overflows using l_1-scaling

  compared to 641 overflows without scaling

Scaling by the Scale Method

- The average product round-off noise power with quantization before accumulation is given by

  -56.3 dB using L_2-scaling

  -53.8 dB using L_∞-scaling

  -47.8 dB using l_1-scaling

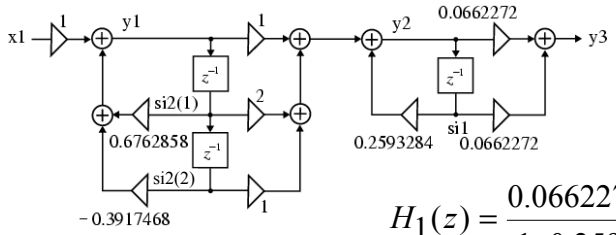
  compared to -40.7 dB without scaling

3.8 Optimum Pole-Zero Pairing and Section Order

- There are many possible cascade filter structures of a higher order IIR transfer function obtained by different section order and different pole-zero pairings
- Each of these structures usually have different output noise power due to product round-offs
- It is of interest to determine the cascade structure with the lowest output noise power

Optimum Pole-Zero Pairing and Section Order

- Example: Unscaled 3rd-order cascade structure with sections in reversed order



$$H_1(z) = \frac{0.0662272(1+z^{-1})}{1-0.2593284z^{-1}}$$

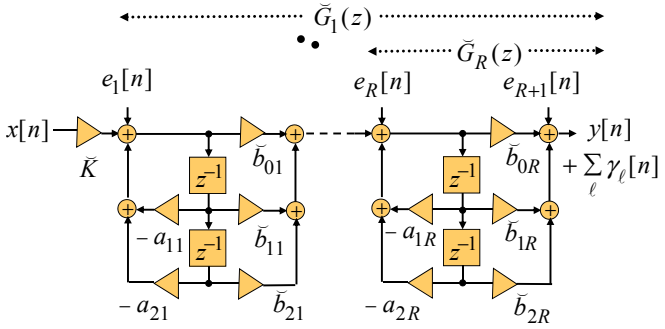
$$H_2(z) = \frac{1+2z^{-1}+z^{-2}}{1-0.6762858z^{-1}+0.3917468z^{-2}}$$

Optimum Pole-Zero Pairing and Section Order

- Continue fixed-point arithmetic settings in the previous example with quantization before accumulation
- The average product round-off noise power of the re-ordered cascade structure is -31.5 dB compared to -40.7 dB of the original order
- Apparently, the re-ordered cascade structure yields a higher product round-off noise power

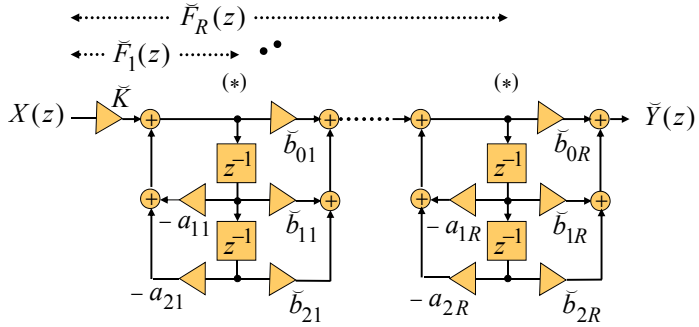
Optimum Pole-Zero Pairing and Section Order

- Consider the scaled noise transfer functions in the scaled cascade form



Optimum Pole-Zero Pairing and Section Order

- Consider the scaled scaling transfer functions in the scaled cascade form



Optimum Pole-Zero Pairing and Section Order

- The product round-off noise power at the filter output using quantization after accumulation is then given by

$$\sigma_\gamma^2 = \sigma_o^2 \left[\sum_{\ell=1}^{R+1} \left\| \breve{G}_\ell \right\|_2^2 \right]$$

- Recall the L_p-norms of the scaled transfer functions

$$\left\| \breve{F}_r \right\|_p = \frac{1}{\left(\prod_{i=0}^{r-1} \beta_i \right)}, \left\| \breve{H} \right\|_p = \frac{1}{\left(\prod_{i=0}^{R} \beta_i \right)}$$

Optimum Pole-Zero Pairing and Section Order

- Using the relation

$$\frac{\left\| \breve{F}_r \right\|_p}{\left\| \breve{H} \right\|_p} = \frac{\prod_{i=0}^{R} \beta_i}{\prod_{i=0}^{r-1} \beta_i} = \prod_{i=r}^{R} \beta_i$$

the output noise variance can be expressed as

$$\sigma_\gamma^2 = \frac{\sigma_o^2}{\left\| \breve{H} \right\|_p^2} \left[\left\| \breve{H} \right\|_p^2 + \sum_{\ell=1}^{R} \left\| \breve{F}_\ell \right\|_p^2 \left\| \breve{G}_\ell \right\|_2^2 \right]$$

Optimum Pole-Zero Pairing and Section Order

- A scaling transfer function $\breve{F}_\ell(z)$ contains the product of the section transfer functions $\breve{H}_i(z), i = 1,2,\ldots,\ell-1$
- A noise transfer function $\breve{G}_\ell(z)$ contains the product of the section transfer functions $\breve{H}_i(z), i = \ell, \ell+1,\ldots,R$
- This implies the rules for pole-zero pairing and ordering of paired sections

Optimum Pole-Zero Pairing and Section Order

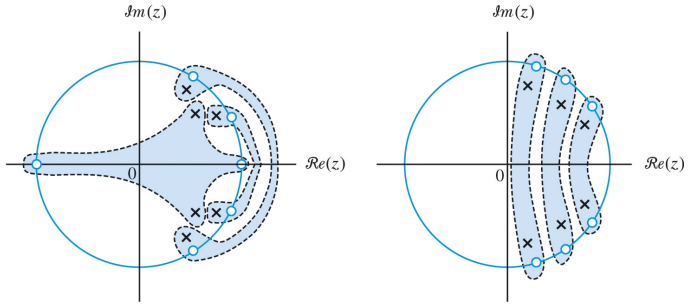
Rule 1:

Pole-Zero Pairing

- Pair the pole (or the pole pair) which is closest to the unit circle with the zero (or the zero pair) which is the closest to it
- Repeat this process until all poles (pole pairs) and all zeros (zero pairs) have been paired

Optimum Pole-Zero Pairing and Section Order

- Illustration of the pole-zero pairing rule



Sixth-order bandpass Sixth-order bandstop

Optimum Pole-Zero Pairing and Section Order

Rule 2:

Order of Paired Sections

- Order the second-order sections obtained by the pole-zero pairing rule according to the closeness of their poles to the unit circle
- Use either decreasing or increasing closeness for ordering

Optimum Pole-Zero Pairing and Section Order in MATLAB

- The function `reorder` can be used to find the optimum pole-zero pairing and ordering
- If `reorder(hd,'up')` is used, the first section will contain the poles closest to the origin, and the last section will contain the poles closest to the unit circle
- If `reorder(hd,'down')` is used, then all sections are ordered in the opposite direction

Optimum Pole-Zero Pairing and Section Order

- If e.g. the L_2-norm is used, then the section order does not affect too much the output product round-off noise power
- If e.g. the L_∞-norm is being employed, the sections with the poles closest to the unit circle exhibit a peaking magnitude response and should therefore be placed closer to the filter output

3.9 Limit Cycles in IIR Filters

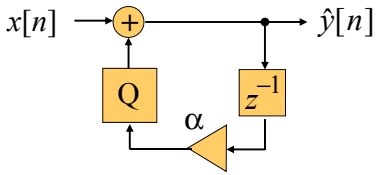
- When a filter input is zero or constant, the filter output should approach zero or a steady-state constant value
- However, finite wordlength effects can create oscillations at the filter output, the so-called limit-cycles
- Due to the presence of feedback, such limit cycles can only occur in IIR filter structures

Limit Cycles in IIR Filters

- Types of limit cycles:
  1. **Quantization or granular limit cycles** due to quantization of products
  2. **Overflow limit cycles** due to addition overflows
- Quantization limit cycles have low amplitudes, whereas overflow limit cycles can cover the whole dynamic range, i.e. they are much more serious in nature

Quantization Limit Cycles

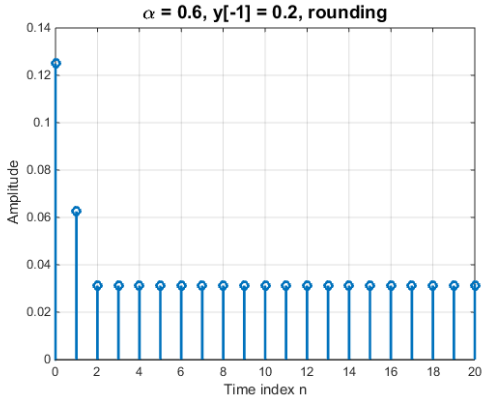
- Example: Consider the first-order IIR filter



- The nonlinear difference equation for the filter is given by $\hat{y}[n] = Q(\alpha \cdot \hat{y}[n-1]) + x[n]$
- We assume zero-input, a non-zero initial condition y[-1] = 0.2, and 6 bit fraction

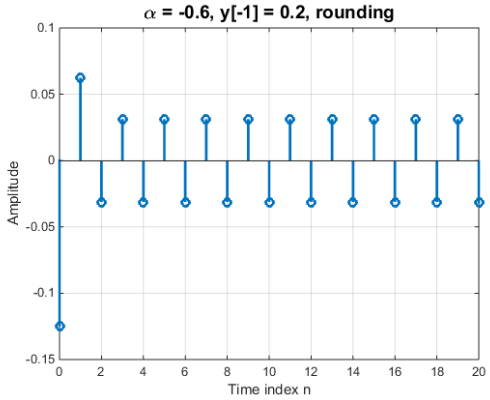
Quantization Limit Cycles

- Case 1: $\alpha = 0.6$ and rounding



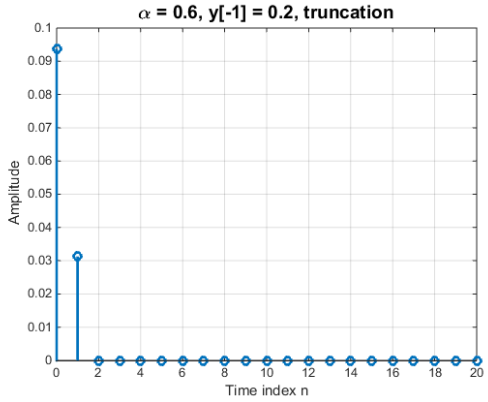
Quantization Limit Cycles

- Case 2: $\alpha = -0.6$ and rounding



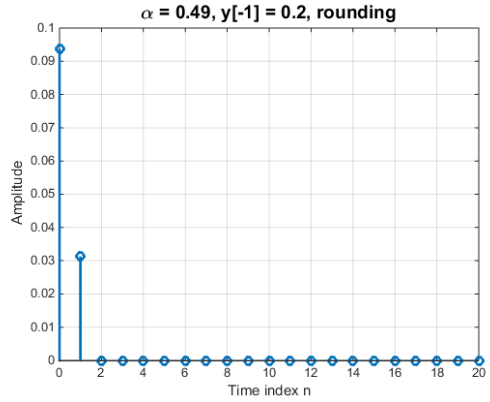
Quantization Limit Cycles

- Case 3: $\alpha = 0.6$ and truncation



Quantization Limit Cycles

- Case 4: $\alpha = 0.49$ and rounding



Quantization Limit Cycles

- Using magnitude truncation, quantization limit cycles are absent in first-order IIR filters
- In 2nd-order IIR filters with magnitude truncation, the probability of quantization limit cycles is very small
- Hence we can greatly reduce the occurrence of quantization limit cycles by using cascade structures and magnitude truncation
- Quantization limit cycles can be minimized having more bits in the quantizer

Quantization Limit Cycles

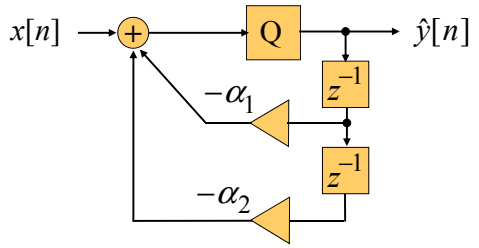
- For first-order IIR filters using rounding, it has been shown that quantization limit cycles are absent if the coefficient is bounded by $|\alpha| < 0.5$
- For 2nd-order IIR filters using rounding, quantization limit cycles are absent if the coefficients are into one of the two regions

$$|\alpha_1| \le 0.25 \; ; -0.5 \le \alpha_2 \le -0.25$$

$$|\alpha_1| \le 0.5 \; ; -0.25 \le \alpha_2 \le 0.5$$

Overflow Limit Cycles

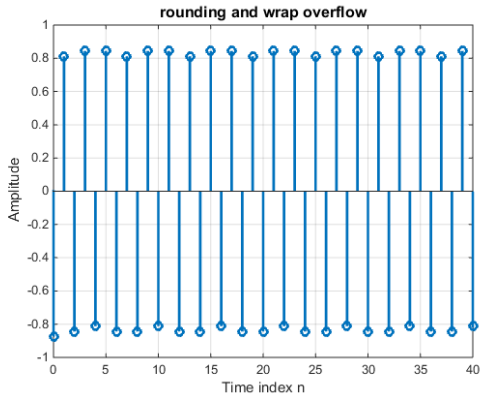
- Example: Consider the 2nd-order IIR filter



- Assume zero-input, 6 bit fraction, y[-1] = 0.8, y[-2] = - 0.8, α_1 = - 0.9, and α_2 = 0.5

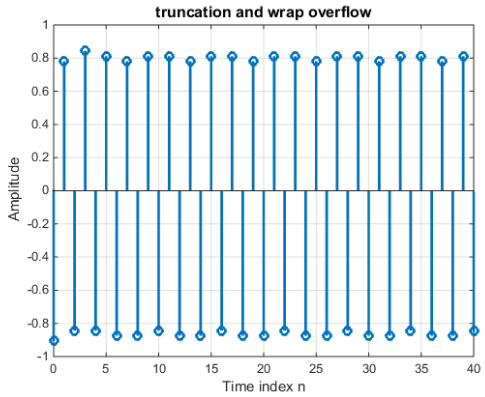
Overflow Limit Cycles

- Case 1: Rounding and wrap overflow



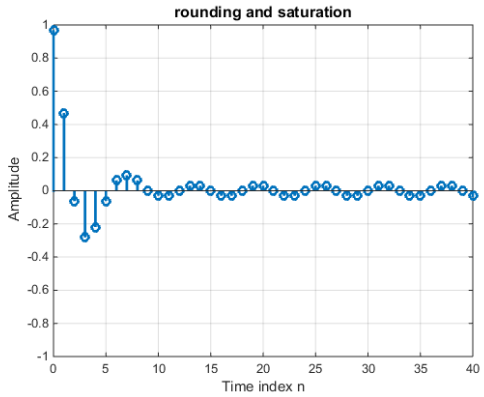
Overflow Limit Cycles

- Case 2: Truncation and wrap overflow



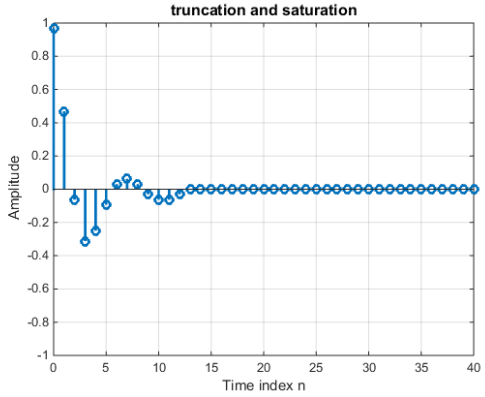
Overflow Limit Cycles

- Case 3: Rounding and saturation



Overflow Limit Cycles

- Case 4: Truncation and saturation

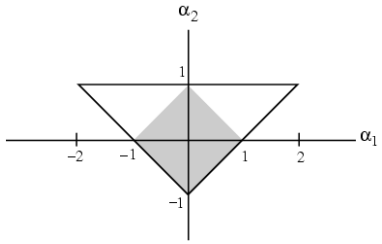


Overflow Limit Cycles

- As can been seen in case 3 and 4, overflow limit cycles are avoided using saturation
- Note that a single overflow occurs in case 3 and 4, and note the continually granular limit cycles in case 3
- Thus, accumulators with large wordlengths (e.g. 40 bits) and saturation are used in fixed-point DSPs

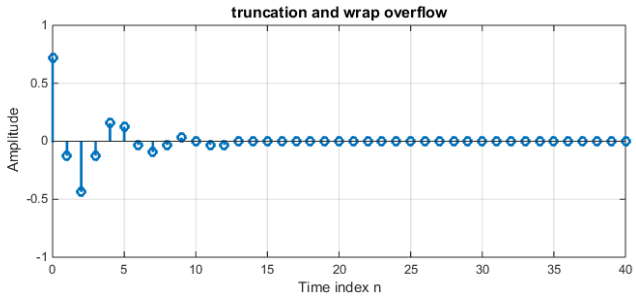
Overflow Limit Cycles

- A 2nd-order IIR filter is stable if $|\alpha_2| < 1$ and $|\alpha_1| < 1 + \alpha_2$ (stability triangle)
- Overflow limit cycles cannot occur under zero-input if the coefficients are bounded by $|\alpha_1| + |\alpha_2| < 1$, indicated by the shaded region



Overflow Limit Cycles

- Example: Consider the 2nd-order IIR filter
- Assume zero-input, 6 bit fraction, $y[-1] = 0.8$, $y[-2] = -0.8$, $\alpha_1 = -0.4$, and $\alpha_2 = 0.5$



3.10 Limit Cycle Free Structures

- Conditions for filter structures that are free of zero-input limit cycles have been derived for state-space models
- For example, a 2nd-order state-space model is given by

$$\begin{bmatrix} s_1[n+1] \\ s_2[n+1] \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} \begin{bmatrix} s_1[n] \\ s_2[n] \end{bmatrix} + \begin{bmatrix} b_1 \\ b_2 \end{bmatrix} x[n]$$

$$y[n] = \begin{bmatrix} c_1 & c_2 \end{bmatrix} \begin{bmatrix} s_1[n] \\ s_2[n] \end{bmatrix} + d\, x[n]$$

Limit Cycle Free Structures

- Let $\mathbf{s}[n] = \begin{bmatrix} s_1[n] & s_2[n] \end{bmatrix}^T$ and

$$\mathbf{A} = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}, \mathbf{B} = \begin{bmatrix} b_1 \\ b_2 \end{bmatrix}, \mathbf{C} = \begin{bmatrix} c_1 & c_2 \end{bmatrix}$$

- Now, the state-space equations are given by

$$\mathbf{s}[n+1] = \mathbf{A}\,\mathbf{s}[n] + \mathbf{B}\,x[n]$$
$$y[n] = \mathbf{C}\,\mathbf{s}[n] + d\,x[n]$$

- $\mathbf{A}$ is called the state-transition matrix or the system matrix, and $\mathbf{s}[n]$ is the state-vector

Limit Cycle Free Structures

- The errors caused by the quantization of the state vector

$$\mathbf{s}[n+1] = \mathbf{A}\,\mathbf{s}[n] + \mathbf{B}\,x[n]$$

  go through the feedback loop of the state space structure and are responsible for the generation of limit cycles
- Assume zero-input, thus $\mathbf{s}[n+1]$ is quantized
- The delayed version of $\mathbf{s}[n+1]$ is $\mathbf{s}[n]$

Limit Cycle Free Structures

- Overflow oscillations occur because the length of the state vector **s** (defined by its vector norm) increases
- Since overflow functions always decrease or do not change the length of a state vector **s**, repeated oscillations cannot occur unless the multiplication of **s** by the system matrix **A**
- Therefore, a necessary condition for overflow oscillations is that the length of **As** is greater than the length of **s**

Limit Cycle Free Structures

- It has been shown that so called minimum norm filters are free of zero-input limit cycles
- The used norm of the matrix **A** is given by

$$\|\mathbf{A}\| = \max_{\mathbf{s}\neq 0} \frac{\|\mathbf{As}\|}{\|\mathbf{s}\|} = \max_{\mathbf{s}\neq 0} \left(\frac{(\mathbf{As})^T\,\mathbf{As}}{\mathbf{s}^T\mathbf{s}} \right)^{1/2}$$

- ||**A**|| is the "gain" of **A** or the maximum increase in the length of the state vector **s**
- If **A** satisfies ||**A**|| < 1, then **s** decreases in length and no zero-input overflow limit cycles occur

Limit Cycle Free Structures

- The design of limit cycle free 2nd-order filter structures reduces to a very simple condition involving the 4 elements of the 2×2 matrix **A**
- For complex eigenvalues $\lambda_{1,2} = \alpha \pm j\beta$ follows for **A**

$$\mathbf{A} = \begin{bmatrix} \alpha & -\beta \\ \beta & \alpha \end{bmatrix}$$

and therefore

$$\mathbf{A}^T\mathbf{A} = \begin{bmatrix} \alpha & \beta \\ -\beta & \alpha \end{bmatrix} \begin{bmatrix} \alpha & -\beta \\ \beta & \alpha \end{bmatrix} = \left(\alpha^2 + \beta^2\right)\mathbf{I}$$

Limit Cycle Free Structures

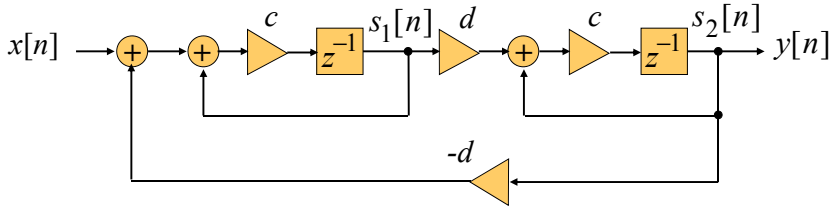
- Finally, we obtain for the matrix norm

$$\|\mathbf{A}\| = \max_{\mathbf{s}\neq 0}\left((\alpha^2+\beta^2)\,\frac{\mathbf{s}^T\mathbf{I}\,\mathbf{s}}{\mathbf{s}^T\mathbf{s}}\right)^{1/2} = (\alpha^2+\beta^2)^{1/2}$$

- For a stable filter, the eigenvalues (poles) must lie in the unit circle, i.e. $r = (\alpha^2+\beta^2)^{1/2} < 1$
- This proves that for a stable 2nd-order filter the matrix norm is less than unity, i.e. $\|\mathbf{A}\| < 1$
- Thus no zero-input limit cycles can occur

Limit Cycle Free Structures

- Example: Consider the 2nd-order filter structure shown below



Analysis yields

$$s_1[n+1] = c\,s_1[n] - cd\,s_2[n] + cx[n]$$

$$s_2[n+1] = cd\,s_1[n] + c\,s_2[n]$$

Limit Cycle Free Structures

- The system matrix is given by

$$\mathbf{A} = \begin{bmatrix} c & -cd \\ cd & c \end{bmatrix}$$

- For the transfer function $H(z)$ we obtain

$$H(z) = \frac{c^2 d z^{-2}}{1 - 2cz^{-1} + c^2(1+d^2)z^{-2}}$$

- For poles at $z = re^{\pm j\theta}$ ($r < 1$ for stability) the coefficients are $c = r\cos\theta$ and $d = \tan\theta$

Limit Cycle Free Structures

- From the above, it follows

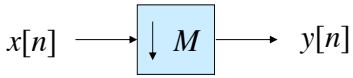
$$\mathbf{A} = \begin{bmatrix} r\cos\theta & -r\sin\theta \\ r\sin\theta & r\cos\theta \end{bmatrix}$$

- The matrix $\mathbf{A}$ satisfies the matrix equation $\mathbf{A}^T\mathbf{A} = \mathbf{A}\mathbf{A}^T = r^2\mathbf{I}$ and the minimum norm condition, i.e. $\|\mathbf{A}\| = r < 1$
- Since the system matrix $\mathbf{A}$ is a normal matrix, the filter structure is a normal form structure

4. Sampling Rate Conversion

4.1 Down-Sampler

- A down-sampler with a down-sampling factor M, where M is a positive integer, develops an output sequence $y[n]$ with a sampling rate that is $(1/M)$-th of that of the input sequence $x[n]$
- Down-sampling element is represented as



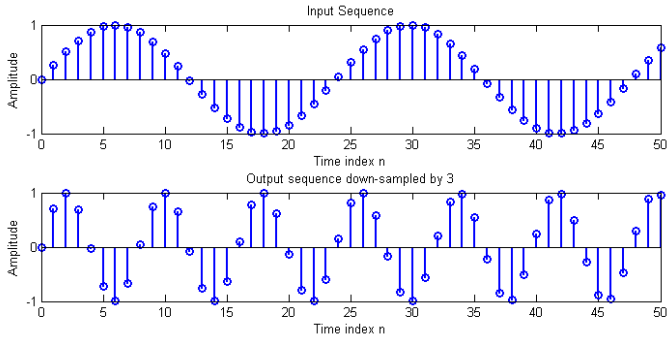
where the input-output relation is $y[n] = x[nM]$

Down-Sampler

- Down-sampling operation is implemented by keeping every M-th sample of $x[n]$ and by removing M-1 in-between samples to generate the output $y[n]$
- Note that a down-sampler is a linear but a time-varying discrete-time system
- The MATLAB function **`downsample`** can be employed to decrease the sampling rate by an integer factor

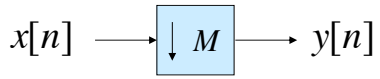
Down-Sampler

- Example: Down-sampling of a sinusoidal sequence by a factor of 3



Down-Sampler

- A down-sampler is a sampling-rate compressor



- The input sampling rate F_x is

$$F_x = \frac{1}{T}$$

and the output sampling rate F_M is given by

$$F_M = \frac{1}{T_M} = \frac{1}{MT} = \frac{F_x}{M}$$

Down-Sampler

- We now derive the frequency-domain relation of a down-sampler
- Applying the z-transform to the input-output relation of a factor-of-M down-sampler

$$y[n] = x[Mn]$$

we get

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

- The expression on the right-hand side can not be directly expressed in terms of $X(z)$

Down-Sampler

- To get around this problem, we define a new intermediate high-rate sequence $x_{\text{int}}[n]$

$$x_{\text{int}}[n] = \begin{cases} x[n], & n = 0, \pm M, \pm 2M, \ldots \\ 0, & \text{otherwise} \end{cases}$$

- Then

$$Y(z) = \sum_{n=-\infty}^{\infty} x[Mn] z^{-n} = \sum_{n=-\infty}^{\infty} x_{\text{int}}[Mn] z^{-n}$$

$$= \sum_{k=-\infty}^{\infty} x_{\text{int}}[k] z^{-k/M} = X_{\text{int}}(z^{1/M})$$

Down-Sampler

- Now, $x_{\text{int}}[n]$ can be formally related to $x[n]$ through

$$x_{\text{int}}[n] = p[n] \cdot x[n]$$

where

$$p[n] = \begin{cases} 1, & n = 0, \pm M, \pm 2M, \ldots \\ 0, & \text{otherwise} \end{cases}$$

is an periodic impulse train which can be represented as a Fourier series given by

$$p[n] = \frac{1}{M} \sum_{k=0}^{M-1} W_M^{-kn}$$

Down-Sampler

- Taking the z-transform of $x_{\text{int}}[n] = p[n] \cdot x[n]$ and making use of the above representation of $p[n]$ we arrive at

$$X_{\text{int}}(z) = \sum_{n=-\infty}^{\infty} p[n]x[n]z^{-n}$$

$$= \frac{1}{M}\sum_{n=-\infty}^{\infty}\left(\sum_{k=0}^{M-1} W_M^{-kn}\right)x[n]z^{-n}$$

$$= \frac{1}{M}\sum_{k=0}^{M-1}\left(\sum_{n=-\infty}^{\infty} x[n]W_M^{-kn}z^{-n}\right) = \frac{1}{M}\sum_{k=0}^{M-1} X\left(zW_M^k\right)$$

Down-Sampler

- The input-output relation in the z-transform domain is then obtained by substituting the above result for $X_{\text{int}}(z)$ in $Y(z) = X_{\text{int}}(z^{1/M})$ resulting in

$$Y(z) = \frac{1}{M}\sum_{k=0}^{M-1} X(z^{1/M}W_M^k)$$

- This relation implies that the output spectrum is a sum of M stretched and shifted replicas of the input spectrum, scaled by a factor of $1/M$

Down-Sampler

- The output spectrum of a factor-of-M down-sampler is given by

$$Y(e^{j\omega}) = \frac{1}{M}\sum_{k=0}^{M-1} X(e^{j(\omega-2\pi k)/M})$$

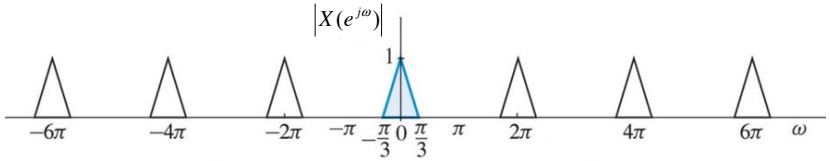
- Aliasing in $Y(e^{j\omega})$ is absent if and only if

$$X(e^{j\omega}) = 0 \quad for \quad \pi/M \le |\omega| \le \pi$$

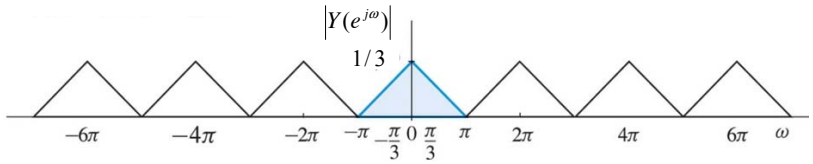
i.e. $x[n]$ must be band-limited to $\pm\pi/M$

Down-Sampler

- Example: Consider a factor-of-3 down-sampler with an input spectrum as shown below



- The output spectrum is then given by



Down-Sampler

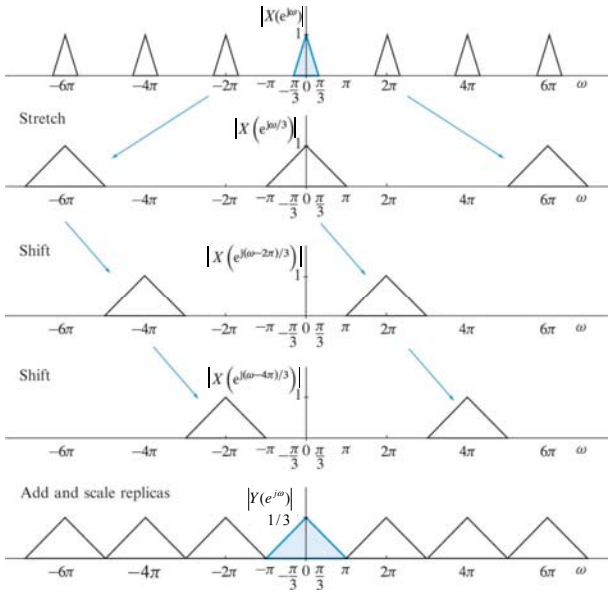
- To understand the meaning of the relation between the input and output spectrum of the factor-of-3 down-sampler given by

$$Y(e^{j\omega}) = \frac{1}{3}\{X(e^{j\omega/3}) + X(e^{j(\omega-2\pi)/3}) + X(e^{j(\omega-4\pi)/3})\}$$

  we now consider the graphical construction of $Y(e^{j\omega})$ directly from $X(e^{j\omega})$
- The construction process involves three steps

Down-Sampler

- First, stretch $X(e^{j\omega})$ by a factor of 3 to obtain $X(e^{j\omega/3})$, i.e. the input spectrum is widened
- Second, create and put 2 copies of $X(e^{j\omega/3})$ at the frequencies $\omega = 2\pi k$ for $k = 1$ and 2
- Third, add the stretched and shifted replicas and scale the sum by 1/3 to obtain $Y(e^{j\omega})$
- Due to sampling rate compression, the input and output frequency variables of a factor-of-M down-sampler are related by $\omega_y = M \cdot \omega_x$



4.2 Up-Sampler

- An up-sampler with an up-sampling factor L, where L is a positive integer, has an output sequence $x_u[n]$ with a sampling rate that is L times larger than that of the input $x[n]$
- Up-sampler is represented as

$$x[n] \longrightarrow \boxed{\uparrow L} \longrightarrow x_u[n]$$

  where the input-output relation is given by

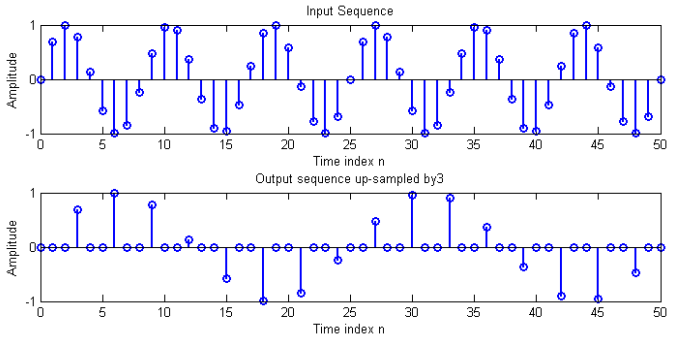
$$x_u[n] = \begin{cases} x[n/L], & n = 0, \pm L, \pm 2L, \cdots \\ 0, & \text{otherwise} \end{cases}$$

Up-Sampler

- An up-sampler inserts $L-1$ zero-valued samples between two consecutive samples of the input sequence $x[n]$
- In practice, the inserted zero-valued samples are replaced with interpolated values to obtain an extended sequence (see interpolators)
- The MATLAB function **`upsample`** can be employed to increase the sampling rate by an integer factor

Up-Sampler

- Example: Up-sampling of a sinusoidal input sequence by a factor of 3



Up-Sampler

- An up-sampler is a sampling rate expander

$$x[n] \longrightarrow \boxed{\uparrow L} \longrightarrow x_u[n]$$

- The input sampling rate F_x is

$$F_x = \frac{1}{T}$$

and the output sampling rate F_L is given by

$$F_L = \frac{1}{T_L} = \frac{L}{T} = LF_x$$

Up-Sampler

- Consider a factor-of-L up-sampler with the input-output relation given by

$$x_u[n] = \begin{cases} x[n/L], & n = 0, \pm L, \pm 2L, \ldots \\ 0, & \text{otherwise} \end{cases}$$

- In the z-transform domain we obtain

$$X_u(z) = \sum_{n=-\infty}^{\infty} x_u[n] z^{-n} = \sum_{n=-\infty}^{\infty} \left\{ x[n/L] \cdot \sum_{k=-\infty}^{\infty} \delta[n - kL] \right\} \cdot z^{-n}$$

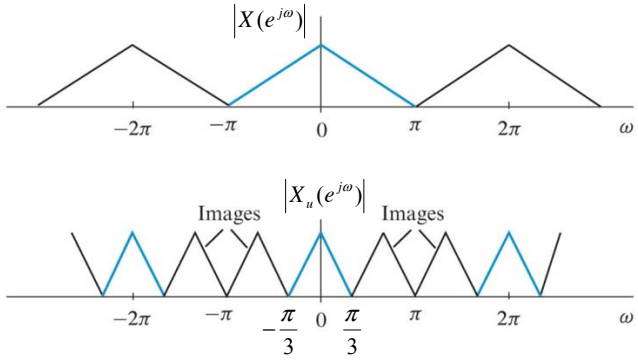
$$= \sum_{m=-\infty}^{\infty} x[m] z^{-mL} = \sum_{m=-\infty}^{\infty} x[m] (z^L)^{-m} = X(z^L)$$

Up-Sampler

- Hence, the spectrum of the up-sampled signal is given by $X_u(e^{j\omega}) = X(e^{j\omega L})$
- Due to the sampling rate expansion, the input and output frequency variables of a factor-of-L up-sampler are related by $\omega_{x_u} = \omega_x / L$
- Thus, L periods of $X(e^{j\omega})$ are compressed to form one period of $X_u(e^{j\omega})$
- The extra $(L-1)$ copies of the compressed spectrum are called images

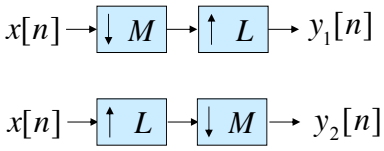
Up-Sampler

- Example: Consider the compressed output spectrum for an up-sampler with factor $L = 3$



4.3 Simple Cascade Structures

- In many applications cascades of up-sampler and down-sampler devices are required, e.g. to change the sampling rate by a rational factor
- Consider the two cascade connections of an up-sampler and a down-sampler shown below



Simple Cascade Structures

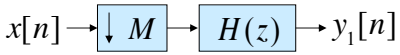
- A cascade of a factor-of-M down-sampler and a factor-of-L up-sampler is interchangeable with no change in the input-output relation

$$y_1[n] = y_2[n]$$

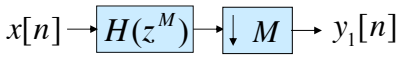
  if and only if M and L are relatively prime, i.e., M and L do not have any common factor that is an integer $k > 1$
- Two further cascade structures, often denoted as multirate identities, are given next

Simple Cascade Structures

- **Multirate identity #1**



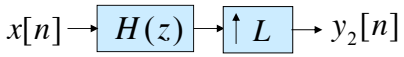
$$\equiv \qquad Y_1(z) = H(z)\frac{1}{M}\sum_{k=0}^{M-1} X(z^{1/M}W_M^k)$$



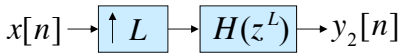
- We can interchange the order of down-sampling and filtering if we up-sample the impulse response of the filter

Simple Cascade Structures

- **Multirate identity #2**



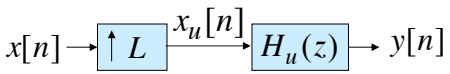
$$\equiv \qquad Y_2(z) = H(z^L)X(z^L)$$



- We can interchange the order of filtering and up-sampling if we up-sample the impulse response of the filter

4.4 Fractional Sampling Rate Converter

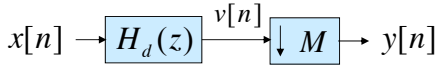
- Since up-sampling causes periodic repetition of the basic spectrum, the unwanted images in the spectrum of the up-sampled signal $x_u[n]$ must be removed by using an additional LP filter $H_u(z)$, called the **interpolation filter**



- The overall system is called an **interpolator**

Fractional Sampling Rate Converter

- Prior to down-sampling, a signal $v[n]$ must be band-limited to $|\omega| < \pi/M$ to avoid aliasing using an additional LP filter $H_d(z)$, called the **decimation filter**



- The overall system is called a **decimator**

Interpolation Filter

- Assume $x[n]$ has been obtained by sampling a continuous-time signal $x_a(t)$ at the Nyquist rate
- If $X_a(j\Omega)$ and $X(e^{j\omega})$ denote the Fourier transforms of $x_a(t)$ and $x[n]$, respectively, then we have

$$X(e^{j\omega}) = \frac{1}{T_0}\sum_{k=-\infty}^{\infty} X_a\left(\frac{j\omega - j2\pi k}{T_0}\right)$$

where T_0 is the sampling period

Interpolation Filter

- Since the sampling is being performed at the Nyquist rate, there is no overlap between the shifted spectra of $X_a(j\omega/T_0)$
- If we instead sample $x_a(t)$ at a much higher rate $1/T = L/T_o$ yielding $y[n]$, its Fourier transform $Y(e^{j\omega})$ is related to $X_a(j\Omega)$ through

$$Y(e^{j\omega}) = \frac{L}{T_o}\sum_{k=-\infty}^{\infty} X_a\left(\frac{j\omega - j2\pi k}{T_o/L}\right)$$

Interpolation Filter

- On the other hand, if we pass $x[n]$ through a factor-of-L up-sampler generating $x_u[n]$, the relation between the Fourier transforms of $x[n]$ and $x_u[n]$ are given by

$$X_u(e^{j\omega}) = X(e^{j\omega L})$$

- It follows that if $x_u[n]$ is passed through an ideal LP filter $H(z)$ with a cutoff frequency π/L and a gain L, the filter output will be precisely $y[n]$

Interpolation Filter

- If ω_c is the highest frequency that needs to be preserved in $x[n]$, then the passband edge of the LP filter should be at

$$\omega_p = \omega_c / L$$

- Thus, the specification of the interpolation LP filter is given by

$$\left|H_u(e^{j\omega})\right| = \begin{cases} L, & |\omega| \le \omega_c / L \\ 0, & otherwise \end{cases}$$

Decimation Filter

- In a similar manner, the specification of the decimation LP filter is given by

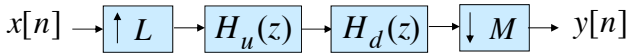
$$\left|H_d(e^{j\omega})\right| = \begin{cases} 1, & |\omega| \le \omega_c / M \\ 0, & otherwise \end{cases}$$

where ω_c denotes the highest frequency that needs to be preserved in the decimated signal

- The design of the interpolation or decimation filter is a standard LP filter design problem

Fractional Sampling Rate Converter

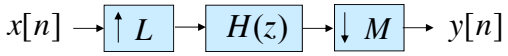
- A fractional sampling rate converter can be achieved by cascading a factor-of-L interpolator with a factor-of-M decimator



- As both the interpolation and the decimation filter operate at the same sampling rate, they can be replaced by a single filter

Fractional Sampling Rate Converter

- Thus, a fractional sampling rate converter can be realized as shown below



- The specification of $H(z)$ is given by

$$\left|H(e^{j\omega})\right| = \begin{cases} L, & |\omega| \le \min\left(\frac{\omega_c}{L}, \frac{\omega_c}{M}\right) \\ 0, & otherwise \end{cases}$$

where ω_c denotes the highest frequency in $x[n]$

Fractional Sampling Rate Converter

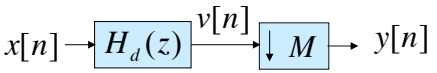
- Example: Consider the design of a sampling rate converter that will transform a signal sampled at 48 kHz (DAT) to a signal with a sampling frequency of 44.1 kHz (CD)
- The required frequency conversion ratio is 44.1/48 = 441/480 = 147/160
- Consequently, this application requires a fractional sampling rate converter with factors $L = 147$ and $M = 160$

Computational Requirements

- The decimation or interpolation LP filter can be implemented either as a FIR or a IIR filter
- In the case of single rate DSP systems, IIR digital filters are, in general, computationally more efficient than equivalent FIR digital filter
- This issue is not quite the same in the case of multirate DSP systems

Computational Requirements

- Consider the factor-of-M decimator shown below



- If the decimation filter $H_d(z)$ is an FIR filter of length N and implemented in direct form, then

$$v[n] = \sum_{m=0}^{N-1} h[m]\,x[n-m]$$

Computational Requirements

- The down-sampler keeps only every M-th sample of $v[n]$ at its output
- Hence, it is sufficient to compute $v[n]$ only for values of n that are multiples of M and skip the computations of in-between samples
- This leads to a factor of M savings in the computational complexity

Computational Requirements

- Now assume $H_d(z)$ to be an IIR filter

$$\frac{V(z)}{X(z)} = H_d(z) = \frac{\sum_{n=0}^{K} p_n z^{-n}}{1 + \sum_{n=1}^{K} d_n z^{-n}}$$

- The direct form II filter structure involves the following computations

$$w[n] = -d_1 w[n-1] - d_2 w[n-2] - \cdots - d_K w[n-K] + x[n]$$

$$v[n] = p_0 w[n] + p_1 w[n-1] + \cdots + p_K w[n-K]$$

Computational Requirements

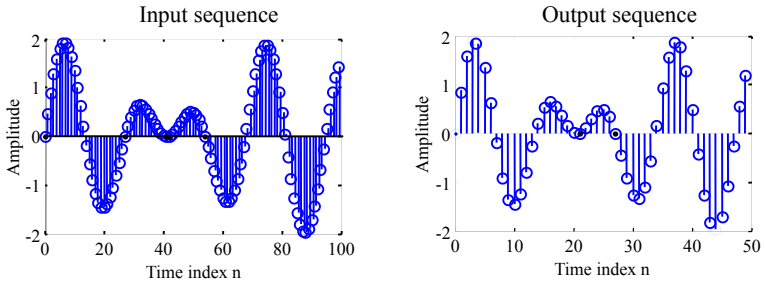
- Since $v[n]$ is being down-sampled, it is sufficient to compute $v[n]$ only for values of n that are integer multiples of M
- However, the intermediate signal $w[n]$ must be computed for all values of n
- As a result, the savings in the computation in this case is going to be less than a factor of M

Sampling Rate Conversion Using MATLAB

- The function **`decimate`** can be employed to filter an input signal with a LP filter and then reduce the sampling rate by an integer factor M
- The function **`interp`** can be employed to increase the sampling rate of an input signal by an integer factor L and then applying a LP filter

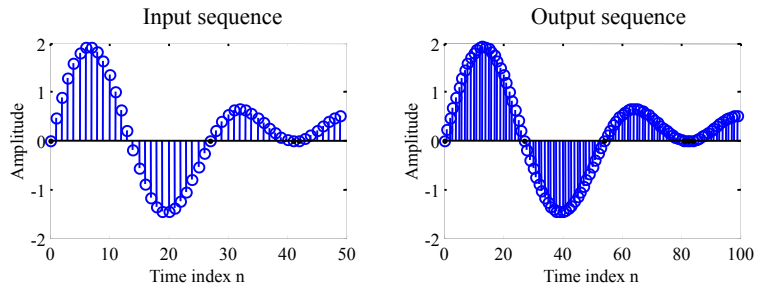
Sampling Rate Conversion Using MATLAB

- Example: Input and output sequence of a factor-of-2 decimator using **`decimate`**



Sampling Rate Conversion Using MATLAB

- Example: Input and output sequence of a factor-of-2 interpolator using **`interp`**

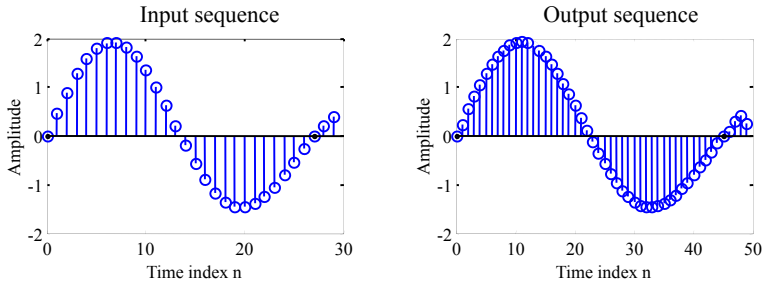


Sampling Rate Conversion Using MATLAB

- The function **`resample(x,L,M)`** returns a signal with **`L/M`** times the sampling rate of **`x`**
- It performs an FIR filter design, followed by a rate change using the function **`upfirdn`**
- Function **`upfirdn`** performs 3 operations:
  1. Up-sampling of the input signal
  2. FIR filtering of the up-sampled signal
  3. Down-sampling of the filtered signal

Sampling Rate Conversion Using MATLAB

- Example: Changing the sampling rate by a rational factor 5/3 using **`resample`**



4.5 Interpolated FIR Filter Design

- An interpolated FIR (IFIR) filter is based on the cascade of two filters with the overall transfer function

$$H_{IFIR}(z) = F(z^L) \cdot G(z)$$

- The periodic filter $F(z^L)$ with interpolation factor L has the factor-L up-sampled impulse response of $H(z)$
- $G(z)$ interpolates the zero valued samples and suppresses the unwanted images of $F(z^L)$

Interpolated FIR Filter Design

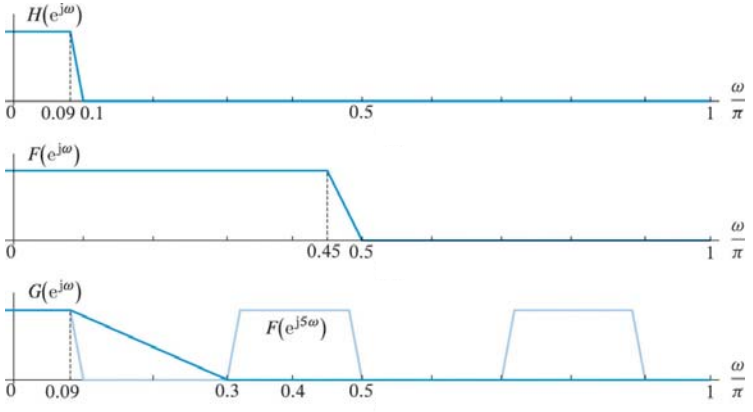
- The cascade of these two filters forms an efficient implementation to meet the desired filter specification
- The advantage of the filter cascade is that the desired frequency response is shared between two low-order filters
- For example, the required computational costs for a narrowband lowpass filter can be reduced drastically

Interpolated FIR Filter Design

- Example: Consider the IFIR design of a narrowband LP filter $H(z)$ with $\omega_p = 0.09\pi$ and $\omega_s = 0.1\pi$
- We choose $F(z^L)$ with factor $L = 5$, i.e. the prototype $F(z)$ with $\omega_p = 0.09\pi \times 5 = 0.45\pi$ and $\omega_s = 0.1\pi \times 5 = 0.5\pi$
- $G(z)$ is the image-suppressor filter of the unwanted images of $F(z^5)$, however with a much wider transition band than $H(z)$

Interpolated FIR Filter Design

- Gain responses of $H(z)$, $F(z)$, $F(z^5)$ and $G(z)$



Interpolated FIR Filter Design

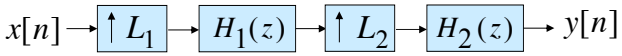
- The periodic filter $F(z^5)$ is "responsible" for achieving the wider desired transition band
- $F(z^5)$ has the first stopband from $\omega_s = 0.1\pi$ to $\omega_s = 2\pi/L - \omega_s = 0.3\pi$ due its multiple images
- We can choose 0.3π as the stopband edge for $G(z)$ instead of 0.1π to reduce the filter order
- IFIR filters are often used in the multistage design of interpolators and decimators

Interpolated FIR Filter Design using MATLAB

- **`Hd = design(Hf,'ifir')`** designs an FIR filter using the IFIR method and returns the object Hd to meet the specifications in Hf
- **`[h,g] = ifir(L,type,f,dev)`** returns a periodic filter **`h`** with interpolation factor **`L`** and an image-suppressor filter **`g`**, to meet the response type (**`type`**), band edge frequencies (**`f`**) and maximum deviations (**`dev`**)

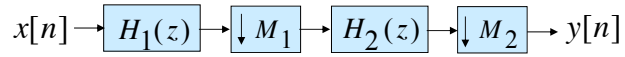
4.6 Multistage Design of Interpolators and Decimators

- Interpolators and decimators can efficiently be implemented in multistage form
- For example, if the interpolation factor L can be expressed as a product of two integers, L_1 and L_2, then the factor-of-L interpolator can be realized in two stages as shown below



Multistage Design of Interpolators and Decimators

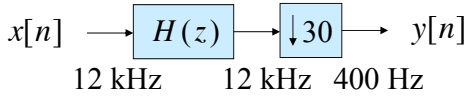
- Likewise if the decimator factor M can be expressed as a product of two integers, M_1 and M_2, then the factor-of-M interpolator can be realized in two stages as shown below



- The designs can also involve more than two stages, depending on the number of factors used to express L and M, respectively

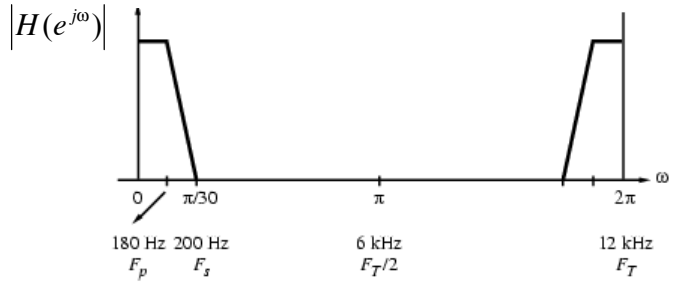
Multistage Design of Interpolators and Decimators

- Example: Consider the design of a decimator for reducing the sampling rate of a signal from 12 kHz to 400 Hz
- The desired decimator with a down-sampling factor $M = 30$ is shown below



Multistage Design of Interpolators and Decimators

- The specifications of $H(z)$ are $F_p = 180$ Hz, $F_s = 200$ Hz, $\delta_p = 0.002$, and $\delta_s = 0.001$



Multistage Design of Interpolators and Decimators

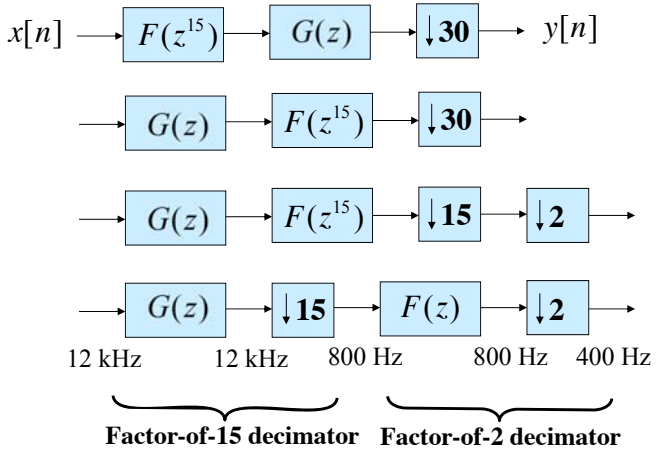
- Assume $H(z)$ to be an equiripple FIR filter designed with the MATLAB function **`firpm`**
- The filter order of $H(z)$ estimated by the function **`firpmord`** is $N_H = 1827$
- Hence, the required number of multiplications per second for the decimation filter is given by

$$R_{M,H} = 1828 \times 400 = 731{,}200 \; mult/\sec$$

Multistage Design of Interpolators and Decimators

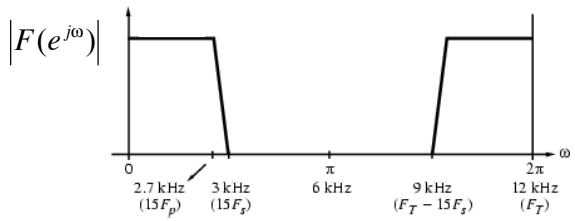
- Now, to reduce the computational costs, we implement the decimation LP filter as an interpolated FIR (IFIR) filter
- The down-sampling factor M is factorized as $30 = 15 \cdot 2$, such that the interpolation factor of the periodic filter can be choosen as $L = 15$
- The steps to design the two-stage decimator are illustrated next

Multistage Design of Interpolators and Decimators



Multistage Design of Interpolators and Decimators

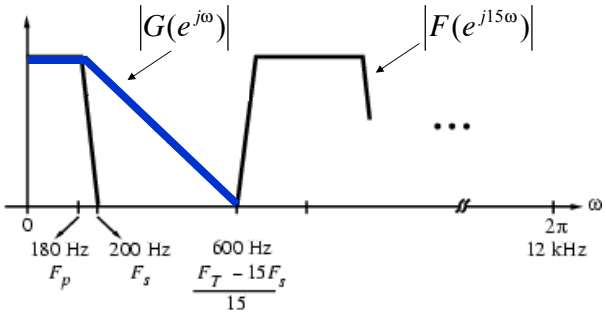
- Magnitude response of the filter $F(z)$



- This corresponds to specifications of $H(z)$ stretched by a factor of $L = 15$

Multistage Design of Interpolators and Decimators

- Magnitude response of the image-supressor filter $G(z)$ and of the periodic filter $F(z^{15})$



Multistage Design of Interpolators and Decimators

- The overall ripple of the cascade is given by the sum of the two passband ripples
- This can be compensated by designing $G(z)$ and $F(z)$ each having a passband ripple of $\delta_p = 0.001$
- The cascade of $G(z)$ and $F(z)$ has a stopband at least as good as $F(z)$ or $G(z)$, individually
- So we can choose $\delta_s = 0.001$ for both filters

Multistage Design of Interpolators and Decimators

- The band edge frequencies of the two filters are summarized below

$$F(z): \quad F_p = 2.7\ kHz,\ F_s = 3\ kHz$$

$$G(z): \quad F_p = 180\ Hz,\ F_s = 600\ Hz$$

- The filter order of $F(z)$ and $G(z)$ estimated by the `firpmord` function are $N_F = 130$ and $N_G = 93$, respectively

Multistage Design of Interpolators and Decimators

- The implementation of $F(z)$ followed by a factor-of-2 down-sampler requires

$$R_{M,F} = 131 \times 400 = 52{,}400\ \ mult/\sec$$

- The implementation of $G(z)$ followed by a factor-of-15 down-sampler requires

$$R_{M,G} = 94 \times 800 = 75{,}200\ \ mult/\sec$$

- Therefore, the savings compared to the single stage design is approximately 83 %

4.7 Polyphase FIR Filter Structures

- Consider an arbitrary sequence $x[n]$ with a z-transform $X(z)$ given by

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

- We can rewrite $X(z)$ as

$$X(z)=\sum_{k=0}^{M-1}z^{-k}X_k(z^M)$$

where

$$X_k(z)=\sum_{n=-\infty}^{\infty}x_k[n]z^{-n}=\sum_{n=-\infty}^{\infty}x[Mn+k]z^{-n}$$

$$0\le k\le M-1$$

Polyphase Decomposition

- The relation between the M polyphase components $x_k[n]$ and $x[n]$ is given by

$$x_k[n]=x[Mn+k],\ 0\le k\le M-1$$

- $X(z)$ can be expressed in matrix form as follows

$$X(z)=\begin{bmatrix}1 & z^{-1} & \cdots & z^{-(M-1)}\end{bmatrix}\begin{bmatrix}X_0(z^M)\\X_1(z^M)\\\vdots\\X_{M-1}(z^M)\end{bmatrix}$$

Polyphase FIR Filter Structures

- The most important application of polyphase decompositions is in polyphase structures for FIR filter
- The M-branch polyphase decomposition breaks the impulse response of length $N=ML$ into M subsequences of length L
- When N is a composite number there are multiple polyphase decompositions
- If N is not a multiple of M, we append zeros

Polyphase FIR Filter Structures

- Example: Consider a length-9 FIR transfer function $H(z)=\sum_{n=0}^{8}h[n]z^{-n}$

We define the following 4 subfilters given by

$$E_0(z)=h[0]+h[4]z^{-1}+h[8]z^{-2}$$

$$E_1(z)=h[1]+h[5]z^{-1}$$

$$E_2(z)=h[2]+h[6]z^{-1}$$

$$E_3(z)=h[3]+h[7]z^{-1}$$

Polyphase FIR Filter Structures

- Then, the transfer function can be expressed in 4-branch polyphase decomposition as

$$H(z) = E_0(z^4) + z^{-1}E_1(z^4) + z^{-2}E_2(z^4) + z^{-3}E_3(z^4)$$

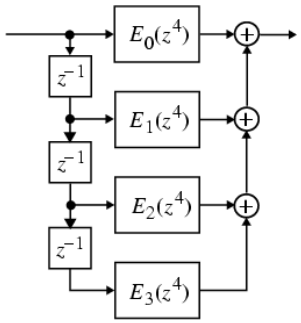
where

$$E_0(z^4) = h[0] + h[4]z^{-4} + h[8]z^{-8}$$

$$E_1(z^4) = h[1] + h[5]z^{-4}$$

$$E_2(z^4) = h[2] + h[6]z^{-4}$$

$$E_3(z^4) = h[3] + h[7]z^{-4}$$



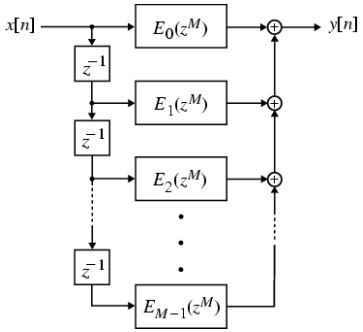
Type I and Type II Polyphase FIR Filter Structures

- Polyphase FIR filter structures can be arranged in different forms
- Consider the M-branch Type I polyphase structure of $H(z)$ given by

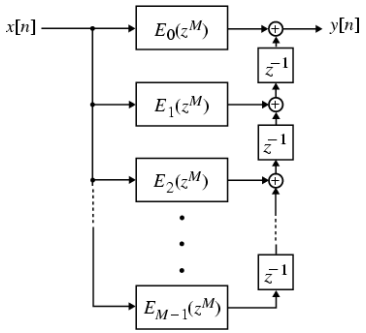
$$H(z) = \sum_{k=0}^{M-1} z^{-k} E_k(z^M)$$

- The Type I polyphase FIR filter structure and its transposed form are shown next

Type I and Type II Polyphase FIR Filter Structures



M-branch Type I polyphase FIR filter structure



M-branch Transposed Type I polyphase FIR filter structure

Type I and Type II Polyphase FIR Filter Structures

- Substituting

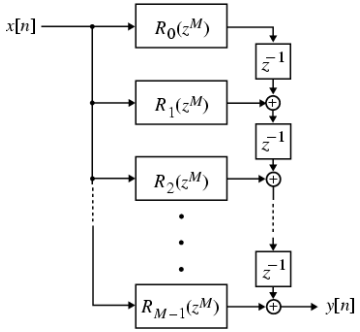
$$R_\ell(z^M) = E_{M-1-\ell}(z^M),\ 0 \le \ell \le M-1$$

in the transposed Type I polyphase FIR filter structure we arrive at the Type II polyphase FIR filter structure given by

$$H(z) = \sum_{\ell=0}^{M-1} z^{-(M-1-\ell)} R_\ell(z^M)$$

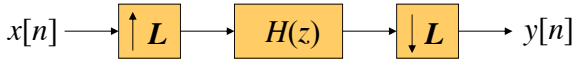
Type I and Type II Polyphase FIR Filter Structures

- The Type II polyphase FIR filter structure is shown below

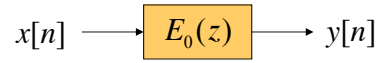


Polyphase Identity

- The multirate structure shown below appears in a number of applications

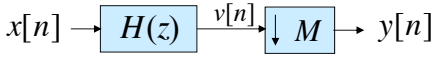


- An equivalent implementation using $H(z)$ in its L-term Type I polyphase form $\sum_{k=0}^{L-1} z^{-k} E_k(z^L)$ is shown below



Polyphase Structures for Decimators and Interpolators

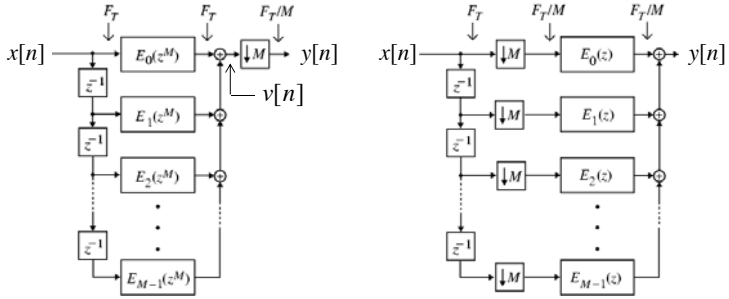
- Computationally efficient decimator and interpolator structures can be derived by using polyphase FIR filter structures
- Consider the factor-of-M decimator



- We realize the LP filter $H(z)$ using the Type I polyphase FIR filter structure as shown next

Polyphase Decimator Structure

- Using the multirate identity #1 we arrive at the decimator structure shown below

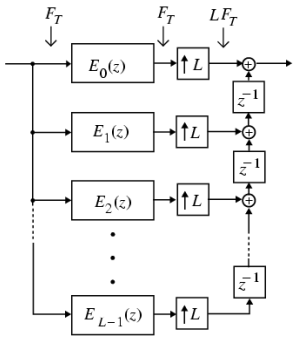


Decimator using Type I polyphase FIR filter structure

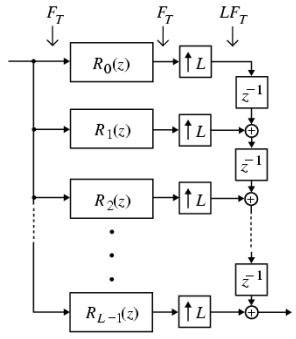
Polyphase Decimator Structure

- The decimator output $y[n]$ in the original structure is obtained by down-sampling the filter output $v[n]$ by a factor of M
- Now, in the modified decimator structure, the arithmetic units are operative with F_T/M, i.e. M times smaller than in the original structure
- Computationally efficient interpolator structures using transposed Type I and Type II polyphase FIR filter structures are shown next

Polyphase Interpolator Structure



Interpolator using the Transposed Type I polyphase FIR filter structure



Interpolator using the Type II polyphase FIR filter structure

Polyphase Structures for Decimators and Interpolators

- More efficient interpolator and decimator structures can be realized by exploiting the symmetry of the impulse response sequence in the case of linear-phase FIR filters
- Example: Consider a factor-of-3 ($M = 3$) decimator using a length-12 Type 1 linear-phase FIR LP filter

Polyphase Structures for Decimators and Interpolators

- The corresponding FIR transfer function is

$$H(z) = h[0] + h[1]z^{-1} + h[2]z^{-2} + h[3]z^{-3} + h[4]z^{-4} + h[5]z^{-5} + h[5]z^{-6} + h[4]z^{-7} + h[3]z^{-8} + h[2]z^{-9} + h[1]z^{-10} + h[0]z^{-11}$$

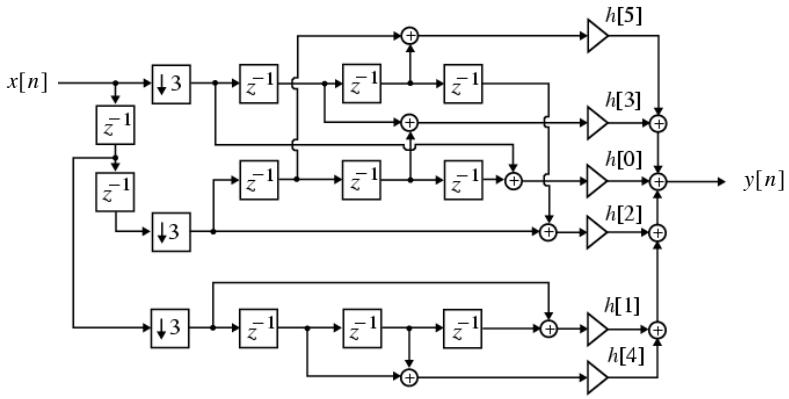
- A conventional 3-branch Type I polyphase filter structure yields the following 3 subfilters

$$E_0(z) = h[0] + h[3]z^{-1} + h[5]z^{-2} + h[2]z^{-3}$$
$$E_1(z) = h[1] + h[4]z^{-1} + h[4]z^{-2} + h[1]z^{-3}$$
$$E_2(z) = h[2] + h[5]z^{-1} + h[3]z^{-2} + h[0]z^{-3}$$

Polyphase Structures for Decimators and Interpolators

- Note that $E_1(z)$ still has a symmetric impulse response, whereas $E_0(z)$ is the mirror image of $E_2(z)$
- These relations can be used in developing a computationally efficient realization using only 6 multipliers and 11 two-input adders
- The factor-of-3 decimator structure with a linear-phase filter is shown next

Polyphase Structures for Decimators and Interpolators



4.8 Nyquist Filter

- In this section we introduce a LP FIR filter with an impulse response that, by design, has certain zero-valued coefficients
- These filters, called Lth-band or Nyquist filters, are therefore computationally more efficient than other LP filters of same order
- Nyquist filters are very useful in multirate systems, e.g. in decimators or interpolators, and in many other applications

Nyquist Filter

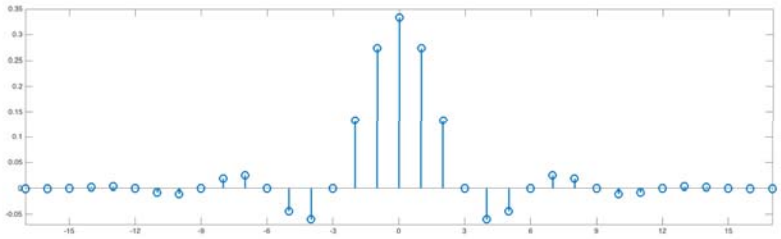
- The Lth-band or Nyquist filter is defined as a zero-phase FIR filter whose impulse response satisfies the time-domain condition

$$h[n] = \frac{1}{L}\frac{\sin\frac{\pi}{L}(n-k)}{\frac{\pi}{L}(n-k)} = \begin{cases} 1/L, & n = k \\ 0, & n-k = \pm L, \pm 2L, \ldots \end{cases}$$

with cutoff frequency $\omega_c = \dfrac{\pi}{L}$

Nyquist Filter

- Example: Consider the impulse response of a 3*th*-band Nyquist filter, where $k = 0$



- Nyquist filters are Type I FIR filters, i.e. they have odd length $N = M + 1$

Nyquist Filter

- Consider the *L*-band Type I polyphase form of a LP filter $H(z)$ given by

$$H(z) = \sum_{k=0}^{L-1} z^{-k} E_k(z^L)$$

- Assume that the k-th polyphase component of $H(z)$ is a constant, i.e. $E_k(z) = \alpha$
- Then, the filter $H(z) = \alpha z^{-k} + \sum_{\substack{\ell=0 \\ \ell \neq k}}^{L-1} z^{-\ell} E_\ell(z^L)$ is an *L*th-band filter

Nyquist Filter

- The output of a factor-of-*L* interpolator using the *L*th-band filter $H(z)$ is given by

$$Y(z) = H(z)X(z^L) = \alpha z^{-k} X(z^L) + \sum_{\substack{\ell=0 \\ \ell \neq k}}^{L-1} z^{-\ell} E_\ell(z^L) X(z^L)$$

- As a result, for a given k, the output of the interpolator yields

$$y[Ln + k] = \alpha x[n]$$

- Thus, the input samples appear at the output of the interpolator at time instants $Ln + k$

Nyquist Filter

- If the zero-th polyphase component $E_0(z)$ of $H(z)$ is given by $E_0(z) = 1/L$, we obtain

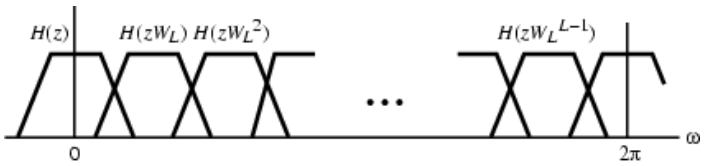
$$\sum_{k=0}^{L-1} H(zW_L^k) = 1$$

- Evaluating this relation on the unit circle yields the frequency-domain condition for an *L*th-band filter given by

$$\sum_{k=0}^{L-1} H(e^{j(\omega - 2\pi k/L)}) = 1$$

Nyquist Filter

- Thus, the sum of L copies of the frequency response of a Nyquist filter, shifted successively by $2\pi/L$, is equal to unity
- The figure below illustrates the frequency-domain condition for an Lth-band filter



Half-Band Filter

- Decimation and interpolation by a factor of 2 require LP filters with a cutoff frequency $\omega_c = \pi/2$, also referred to as half-band filter
- The polyphase structure of a half-band filter ($L = 2$) is given by

$$H(z) = 0.5 + z^{-1}E_1(z^2)$$

with its impulse response satisfying

$$h[0] = 0.5$$

$$h[2n] = 0,\ n = \pm 1, \pm 2, \ldots$$

Half-Band Filter

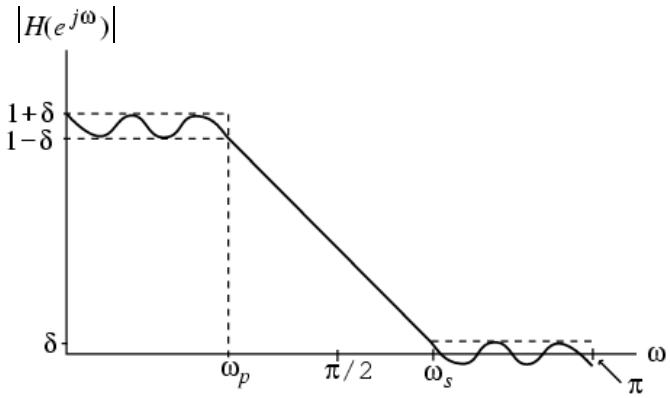
- For $L = 2$ and $\alpha = 0.5$ the sum $\sum_{k=0}^{L-1} H(zW_L^k)$ reduces to $H(z) + H(-z) = 1$
- Evaluating the above relationship on the unit circle yields $H(e^{j\omega}) + H(e^{j(\omega-\pi)}) = 1$
- If we change the variable ω to $\pi/2 + \theta$ we obtain the following relation

$$H(e^{j(\pi/2+\theta)}) - \frac{1}{2} = -(H(e^{j(\pi/2-\theta)} - \frac{1}{2})$$

- Thus, $H(e^{j\omega})$ has odd symmetry about $\omega = \pi/2$

Half-Band Filter

- Figure below illustrates the symmetry of the magnitude response of the half-band filter



Half-Band Filter

- **Implications of the symmetry:**
  1. The passband ripple value δ_p and the stopband ripple value δ_s are equal
  2. The passband edge frequency ω_p and the stopband edge frequency ω_s are symmetric with respect to $\pi/2$, i.e. $\omega_p + \omega_s = \pi$
- An attractive property is that about 50% of the filter coefficients are zero, reducing the amount of computations by almost one-half

Design of a *L*th-Band Filter

- A linear-phase *L*th-band LP FIR filter can be readily designed using the window technique
- The impulse response coefficients of the LP filter are chosen as $h[n] = h_{LP}[n] \cdot w[n]$
- $h_{LP}[n]$ is the impulse response of an ideal LP filter with a cutoff frequency $\omega_c = \pi/L$
- $w[n]$ is a suitable window, e.g. a Hamming window

Design of a *L*th-Band Filter

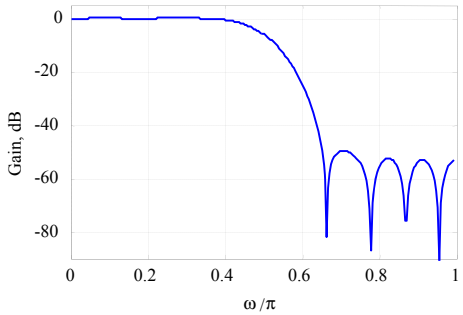
- The impulse response of an ideal *L*th-band filter obtained for $k = 0$ is given by

$$h_{LP}[n] = \frac{1}{L}\frac{\sin(\pi n / L)}{\pi n / L}, \quad -\infty < n < \infty$$

- The design of a practical *L*th-band filter is straightforward, as long as the window has even symmetry, odd length and the center coefficient is equal to 1

Design of a *L*th-Band Filter

- Example: Gain response of a half-band filter of length-23 using a Hamming window



Design of a *L*th-Band Filter

- The filter coefficients are given by

$h[-11]=h[11]=-0.002315;\ \ h[-10]=h[10]=0;$

$h[-9]=h[9]=0.005412;\ \ h[-8]=h[8]=0;$

$h[-7]=h[7]=-0.001586;\ \ h[-6]=h[6]=0;$

$h[-5]=h[5]=0.003584;\ \ h[-4]=h[4]=0;$

$h[-3]=h[3]=-0.089258;\ \ h[-2]=h[2]=0;$

$h[-1]=h[1]=0.3122379;\ \ h[0]=0.5;$

- As expected, $h[n] = 0$ for $n = \pm 2, \pm 4, \pm 6, \pm 8, \pm 10$

Nyquist Filter Design Using MATLAB

- Design functions for Nyquist filters:
  - **`firnyquist`**
  - **`firhalfband`**
- Design objects for Nyquist filters:
  - **`fdesign.nyquist`**
  - **`fdesign.halfband`**
  - Available design methods are returned using **`designmethods`**

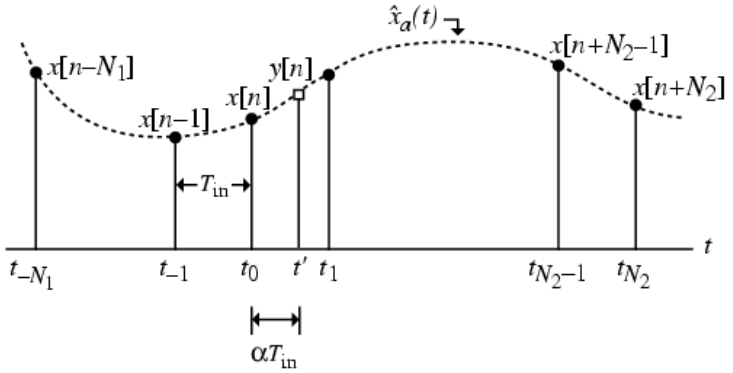
4.9 Polynomial Sampling Rate Converter

- Interpolation can be used to find new sample values at arbitrary points between the existing samples, i.e. to design a fractional rate converter
- Linear interpolation is the simplest interpolation technique, in which the interpolated values between two consecutive samples lie on the straight line connecting these two samples
- Improved performance can be obtained by using higher-order Lagrange or Spline interpolation

Interpolation Problem

- Given $N_2 + N_1 + 1$ input signal samples, $x[k]$, $k = -N_1, \ldots, N_2$, obtained e.g. by sampling an analog signal $x_a(t)$ at $t = t_k = t_0 + k\,T_{in}$
- Problem statement: Determine the sample value $x_a(t_0 + \alpha T_{in}) = y[\alpha]$ at an arbitrary time $t' = t_0 + \alpha T_{in}$ where $-N_1 \le \alpha \le N_2$
- The next figure illustrates the interpolation process by an arbitrary factor α

Interpolation Problem



Lagrange Interpolation

- Here, a polynomial approximation $\hat{x}_a(t)$ to $x_a(t)$ is defined as

$$\hat{x}_a(t) = \sum_{k=-N_1}^{N_2} P_k(t)\, x[n+k]$$

where $P_k(t)$ are the Lagrange polynomials given by

$$P_k(t) = \prod_{\substack{\ell=-N_1 \\ \ell \neq k}}^{N_2} \left(\frac{t - t_\ell}{t_k - t_\ell} \right), \quad -N_1 \le k \le N_2$$

Lagrange Interpolation

- Since

$$P_k(t_r) = \begin{cases} 1, & k = r \\ 0, & k \neq r \end{cases}, \quad -N_1 \le r \le N_2$$

it follows

$$\hat{x}_a(t_k) = x_a(t_k), \quad -N_1 \le k \le N_2$$

- The value of $\hat{x}_a(t')$ at an arbitrary time value $t' = t_0 + \alpha T_{in}$ is given by

$$\hat{x}_a(t_0 + \alpha T_{in}) = y[n] = \sum_{k=-N_1}^{N_2} P_k(\alpha)\, x[n+k]$$

Lagrange Interpolation

where the Lagrange polynomials are

$$P_k(\alpha) = P_k(t_0 + \alpha T_{in}) = \prod_{\substack{\ell=-N_1 \\ \ell \neq k}}^{N_2} \left(\frac{\alpha - \ell}{k - \ell} \right), \quad -N_1 \le k \le N_2$$

- Now, we apply Lagrange polynomials to design a sampling rate converter with arbitrary conversion factor

Polynomial Sampling Rate Converter

- Example: Consider a factor-of-3/2 sampling rate converter using a 3rd-order Lagrange interpolation with $N_1 = 2$ and $N_2 = 1$
- The output $y[n]$ of the converter, using the 4 input samples $x[n\text{-}2]$ through $x[n\text{+}1]$, is then given by

$$y[n] = P_{-2}(\alpha)x[n-2] + P_{-1}(\alpha)x[n-1] + P_0(\alpha)x[n] + P_1(\alpha)x[n+1]$$

Polynomial Sampling Rate Converter

- Here, the Lagrange polynomials are given by

$$P_{-2}(\alpha) = \frac{(\alpha+1)\alpha(\alpha-1)}{-6} = \frac{1}{6}(-\alpha^3 + \alpha)$$

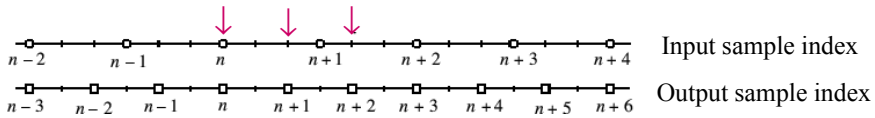
$$P_{-1}(\alpha) = \frac{(\alpha+2)\alpha(\alpha-1)}{2} = \frac{1}{2}(\alpha^3 + \alpha^2 - 2\alpha)$$

$$P_0(\alpha) = \frac{(\alpha+2)(\alpha+1)(\alpha-1)}{-2} = -\frac{1}{2}(\alpha^3 + 2\alpha^2 - \alpha - 2)$$

$$P_1(\alpha) = \frac{(\alpha+2)(\alpha+1)\alpha}{6} = \frac{1}{6}(\alpha^3 + 3\alpha^2 + 2\alpha)$$

Polynomial Sampling Rate Converter

- Next, we determine $y[n]$, $y[n\text{+}1]$, and $y[n\text{+}2]$
- Thus, we have to determine the corresponding three values for the factor α
- Locations of the output samples y[n], $y[n\text{+}1]$, and $y[n\text{+}2]$ in the input sample domain are marked with an arrow



Polynomial Sampling Rate Converter

- For $y[n]$ the value α is denoted as α_0 and the equation is of the form

$$y[n] = P_{-2}(\alpha_0)x[n-2] + P_{-1}(\alpha_0)x[n-1] + P_0(\alpha_0)x[n] + P_1(\alpha_0)x[n+1]$$

- From the previous figure we observe $\alpha_0 = 0$
- Substituting $\alpha_0 = 0$ in the Lagrange polynomials we get $P_{-2}(\alpha_0) = 0$ $P_{-1}(\alpha_0) = 0$ $P_0(\alpha_0) = 1$ $P_1(\alpha_0) = 0$

Polynomial Sampling Rate Converter

- The value of α in the computation of $y[n+1]$ is $\alpha_1 = 2/3$
- Substituting this value in $P_k(\alpha_1)$ we obtain

$$P_{-2}(\alpha_1) = 0.0617 \quad P_{-1}(\alpha_1) = -0.2963$$
$$P_0(\alpha_1) = 0.7407 \quad P_1(\alpha_1) = 0.4938$$

- The value of α in the computation of $y[n+2]$ is $\alpha_2 = 4/3$, and for $P_k(\alpha_2)$ we get

$$P_{-2}(\alpha_2) = -0.1728 \quad P_{-1}(\alpha_2) = 0.7407$$
$$P_0(\alpha_2) = -1.2963 \quad P_1(\alpha_2) = 1.7284$$

Polynomial Sampling Rate Converter

- Combining the three equations we arrive at the input-output relation

$$\begin{bmatrix} y[n] \\ y[n+1] \\ y[n+2] \end{bmatrix} = \begin{bmatrix} P_{-2}(\alpha_0) & P_{-1}(\alpha_0) & P_0(\alpha_0) & P_1(\alpha_0) \\ P_{-2}(\alpha_1) & P_{-1}(\alpha_1) & P_0(\alpha_1) & P_1(\alpha_1) \\ P_{-2}(\alpha_2) & P_{-1}(\alpha_2) & P_0(\alpha_2) & P_1(\alpha_2) \end{bmatrix} \begin{bmatrix} x[n-2] \\ x[n-1] \\ x[n] \\ x[n+1] \end{bmatrix}$$

- This matrix form can be compactly written as

$$\begin{bmatrix} y[n] \\ y[n+1] \\ y[n+2] \end{bmatrix} = \mathbf{H} \begin{bmatrix} x[n-2] \\ x[n-1] \\ x[n] \\ x[n+1] \end{bmatrix}$$

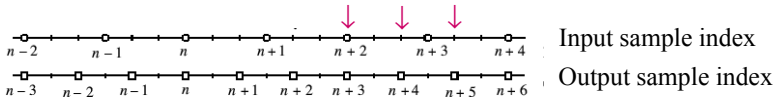
where $\mathbf{H}$ is the block coefficient matrix

Polynomial Sampling Rate Converter

- For the factor-of-3/2 converter, we obtain

$$\mathbf{H} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0.0617 & -0.2963 & 0.7407 & 0.4938 \\ -0.1728 & 0.7407 & -1.2963 & 1.7284 \end{bmatrix}$$

- From the input and output sample locations

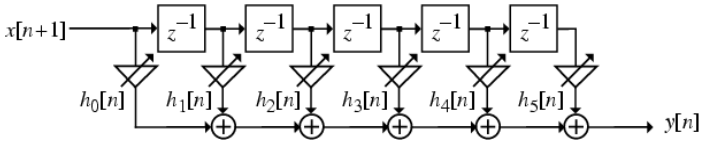


we observe that the coefficients to compute $y[n+3]$, $y[n+4]$, and $y[n+5]$ are given by the same block matrix $\mathbf{H}$

Polynomial Sampling Rate Converter

- Hence, the value of the factor α changes periodically with a period of 3 samples
- This leads to a realization of the factor-of-3/2 polynomial sampling rate converter with a time-varying FIR filter
- The structure of the 5-th order time-varying FIR filter is shown next
- The filter coefficients are changed with a period of 3 samples as assigned in the table below

Polynomial Sampling Rate Converter



Time	$h_0[n]$	$h_1[n]$	$h_2[n]$	$h_3[n]$	$h_4[n]$	$h_5[n]$
3ℓ	$P_1(\alpha_0)$	$P_0(\alpha_0)$	$P_{-1}(\alpha_0)$	$P_{-2}(\alpha_0)$	0	0
$3\ell+1$	0	$P_1(\alpha_1)$	$P_0(\alpha_1)$	$P_{-1}(\alpha_1)$	$P_{-2}(\alpha_1)$	0
$3\ell+2$	0	0	$P_1(\alpha_2)$	$P_0(\alpha_2)$	$P_{-1}(\alpha_2)$	$P_{-2}(\alpha_2)$

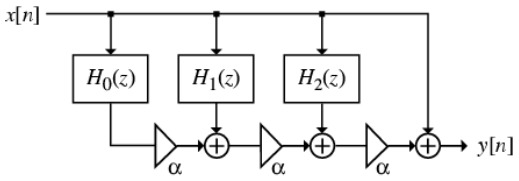
Farrow Structure

- Substituting the Lagrange polynomials in $y[n]$ we arrive at a polynomial in the variable α

$$y[n] = \alpha^3(-\tfrac{1}{6}x[n-2]+\tfrac{1}{2}x[n-1]-\tfrac{1}{2}x[n]+\tfrac{1}{6}x[n+1]) + \alpha^2(\tfrac{1}{2}x[n-1]-x[n]+\tfrac{1}{2}x[n+1]) + \alpha(\tfrac{1}{6}x[n-2]-x[n-1]+\tfrac{1}{2}x[n]+\tfrac{1}{3}x[n+1]) + x[n]$$

- This leads to the Farrow structure shown next

Farrow Structure



Note:

α changes periodically with a period of 3 samples

$$H_0(z) = -\frac{1}{6}z^{-2} + \frac{1}{2}z^{-1} - \frac{1}{2} + \frac{1}{6}z$$

$$H_1(z) = \frac{1}{2}z^{-1} - 1 + \frac{1}{2}z$$

$$H_2(z) = \frac{1}{6}z^{-2} - z^{-1} + \frac{1}{2} + \frac{1}{3}z$$

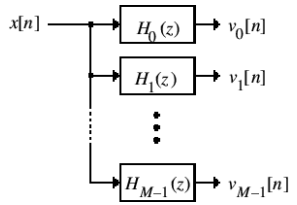
Design of a Fractional Sampling Rate Converter Using MATLAB

- Polynomial SRC using a Farrow structure:
  - `fdesign.polysrc`
  - `dsp.FarrowRateConverter`
- Direct-form FIR polyphase fractional SRC:
  - `mfilt.firsrc`
  - `dsp.FIRRateConverter`
  - `dsp.SampleRateConverter`

5. Filter Banks

5.1 Introduction

- A digital filter bank is a set of filters with either a common input or a summed output
- An *M*-band analysis filter bank with a common input is shown below

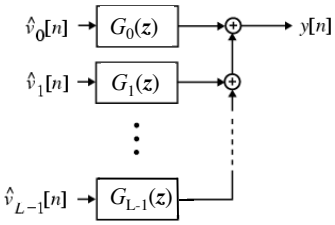


Introduction

- The M subfilters $H_k(z)$ in the analysis filter bank are known as analysis filters
- The analysis filter bank is used to decompose the input signal $x[n]$ into a set of subband signals $v_k[n]$
- Each subband signal occupies a portion of the original frequency band
- If all subbands have the same width, the filter bank is called an uniform filter bank

Introduction

- An *L*-band synthesis filter bank with a summed output is shown below



- The L subfilters $G_k(z)$ in the synthesis filter bank are known as synthesis filters

Introduction

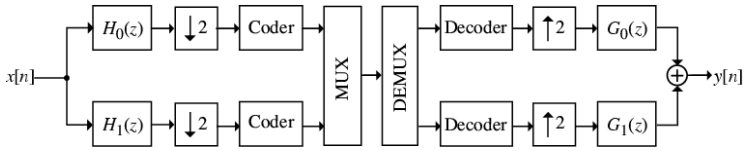
- The outputs of the synthesis filters are added to obtain the reconstructed signal $y[n]$
- Filter banks play an important role in many modern signal processing applications such as audio and image coding
- The reason for their popularity is the fact that they easily allow the extraction of spectral components of a signal while providing very efficient multirate implementations

5.2 Analysis of Two-Channel Filter Banks

- In many applications, an input signal $x[n]$ is divided into two subband signals $v_k[n]$ using a two-channel analysis filter bank
- Before processing the subband signals are decimated by a factor of 2
- After processing, the subband signals are then interpolated by a factor of 2 before being combined by the two-channel synthesis filter bank into the output signal $y[n]$

Two-Channel Filter Bank-based Subband Codec

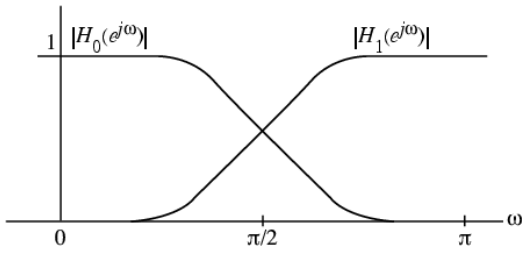
- Figure below shows the basic two-channel filter bank-based subband codec (SBC)



- Apart from the filter banks, the SBC includes coders, the multiplexer and demultiplexer for transmission, and decoders

Two-Channel Filter Bank-based Subband Codec

- The analysis filters $H_0(z)$ and $H_1(z)$ have typically a lowpass and highpass frequency responses, respectively, with a cutoff at $\pi/2$



Two-Channel Filter Bank-based Subband Codec

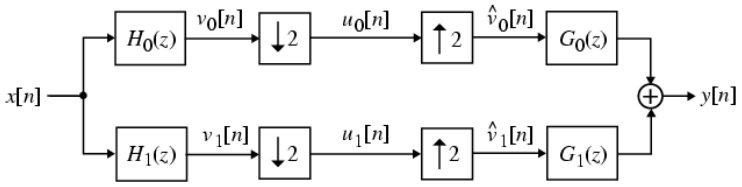
- Clearly, each down-sampled subband signal is encoded by exploiting the special spectral properties of the signal
- The reconstructed output $y[n]$ should be a reasonably close replica of the input $x[n]$
- It follows from the figure that the sampling rates of the output $y[n]$ and the input $x[n]$ are the same

Two-Channel Filter Bank-based Subband Codec

- In practice, various errors are generated in this two-channel filter bank-based subband codec
- In addition to the coding errors and the errors caused by transmission of the coded signals through the channel, the filter bank itself introduces several errors due to the sampling rate alterations and imperfect filters
- Next, we ignore the coding and channel errors

Analysis of Two-Channel Filter Banks

- We only consider the errors caused by the sampling rate alterations and their effects
- The general two-channel filter bank structure is shown below



Analysis of Two-Channel Filter Banks

- Making use of the input-output relations of the down-sampler and the up-sampler in the z-domain we arrive at

$$V_k(z) = H_k(z)X(z),\ k = 0,1$$

$$U_k(z) = \frac{1}{2}\{V_k(z^{1/2}) + V_k(-z^{1/2})\},\ k = 0,1$$

$$\hat{V}_k(z) = U_k(z^2),\ k = 0,1$$

Analysis of Two-Channel Filter Banks

- From the first and the last equations we obtain after some algebra

$$\hat{V}_k(z) = \frac{1}{2}\{V_k(z) + V_k(-z)\}$$
$$= \frac{1}{2}\{H_k(z)X(z) + H_k(-z)X(-z)\}$$

- The reconstructed output is given by

$$Y(z) = G_0(z)\hat{V}_0(z) + G_1(z)\hat{V}_1(z)$$

Analysis of Two-Channel Filter Banks

- For the output we arrive at

$$Y(z) = \frac{1}{2}\{H_0(z)G_0(z) + H_1(z)G_1(z)\}X(z)$$
$$+ \frac{1}{2}\{H_0(-z)G_0(z) + H_1(-z)G_1(z)\}X(-z)$$

- The first term includes the system function between $X(z)$ and $Y(z)$ and may cause magnitude and phase distortion
- The second term between $X(-z)$ and $Y(z)$ is the aliasing term due to sampling rate alteration

Analysis of Two-Channel Filter Banks

- The above equation can be compactly written as

$$Y(z) = T(z)X(z) + A(z)X(-z)$$

- $T(z)$ is the distortion transfer function given by

$$T(z) = \frac{1}{2}\{H_0(z)G_0(z) + H_1(z)G_1(z)\}$$

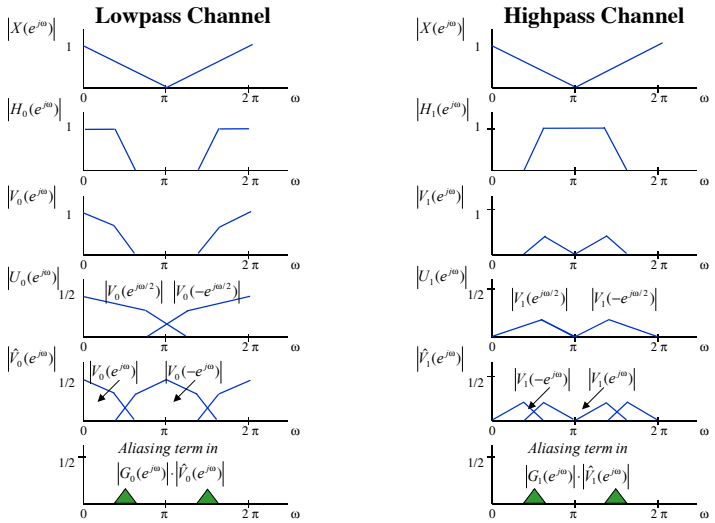
- $A(z)$ is the aliasing transfer function given by

$$A(z) = \frac{1}{2}\{H_0(-z)G_0(z) + H_1(-z)G_1(z)\}$$

Analysis of Aliasing

- To understand the aliasing term we look at the internal spectra using an appropriate input
- In the LP channel, the aliasing component $\left|V_0(-e^{j\omega/2})\right|$ in the output of the down-sampler overlaps with the desired component $\left|V_0(e^{j\omega/2})\right|$
- The contribution of $\left|X(-e^{j\omega})\right|$ to the output $\left|G_0(e^{j\omega})\right| \cdot \left|\hat{V}_0(e^{j\omega})\right|$ of the LP channel (shaded area) is the undesired alias component
- The HP channel generates a similar aliasing

Analysis of Aliasing



Analysis of Aliasing

- Note that the aliasing term is the result of the aliasing and imaging components generated by the down-sampler and the up-sampler
- It is clear that the input can only be recovered if no aliasing occurs, but the aliased case is the normal operation in such a filter bank
- However, the synthesis filters $G_0(z)$ and $G_1(z)$ can be designed to eliminate the aliasing term

Analysis of Two-Channel Filter Banks

- Note that the up-sampler and down-sampler are linear but time-varying components
- In the aliased case the two-channel filter bank is a time-varying system
- However, it is possible to design the analysis and synthesis filters so that the filter bank is alias-free, i.e. $A(z) = 0$
- Then, the alias-free filter bank becomes a linear time-invariant system

Reconstruction Conditions

- To cancel aliasing distortion we need to ensure that $A(z) = 0$, i.e.

$$H_0(-z)G_0(z) + H_1(-z)G_1(z) = 0$$

- Then, the output of the alias-free two-channel filter bank is given by $Y(z) = T(z)\,X(z)$
- On the unit circle, we have

$$\begin{aligned} Y(e^{j\omega}) &= T(e^{j\omega})\,X(e^{j\omega}) \\ &= \left|T(e^{j\omega})\right| e^{j\phi(\omega)}\,X(e^{j\omega}) \end{aligned}$$

Reconstruction Conditions

- If we restrict $T(z)$ to be an allpass function, i.e. $|T(e^{j\omega})| = d$ with $d \neq 0$ then

$$|Y(e^{j\omega})| = d\,|X(e^{j\omega})|$$

- Thus, the output has the same magnitude response as that of the input (scaled by d) but exhibits phase distortion
- In this case, the filter bank is said to be magnitude preserving

Reconstruction Conditions

- If $T(z)$ has a linear phase, i.e.

$$\arg\{T(e^{j\omega})\} = \phi(\omega) = \alpha\omega + \beta$$

then

$$\arg\{Y(e^{j\omega})\} = \arg\{X(e^{j\omega})\} + \alpha\omega + \beta$$

- The filter bank is called phase-preserving, but it exhibits magnitude distortion
- If an alias-free filter bank has no magnitude distortion and no phase distortion, it is called a perfect reconstruction (PR) filter bank

Reconstruction Conditions

- In the case of a PR filter bank, the distortion transfer function is $T(z) = d\,z^{-\ell}$ resulting in

$$Y(z) = d\,z^{-\ell}X(z)$$

- In the time-domain, the input-output relation for all possible inputs is given by

$$y[n] = d\,x[n-\ell]$$

- Therefore, for a PR filter bank, the output is a scaled and delayed replica of the input

5.3 Two-Channel QMF Banks

- A two-channel Quadrature Mirror Filter (QMF) bank is obtained if we choose

$$H_0(z) = H(z), G_0(z) = H(z)$$

$$H_1(z) = H(-z), G_1(z) = -H(-z)$$

- Hence, all filters are essentially determined from one prototype lowpass filter $H(z)$
- The QMF bank provides a complete aliasing cancellation but satisfies PR only approximately

Two-Channel QMF Banks

- The relation $\left|H_1(e^{j\omega})\right| = \left|H_0(e^{j(\omega-\pi)})\right|$ implies that if $H_0(z)$ is a LP then $H_1(z)$ will be a HP filter
- The magnitude responses of the two filters are symmetric to the "mirror" frequency $\omega = \pi/2$
- Moreover, a perfect reconstruction filter bank requires $H_0(z)$ and $H_1(z)$ to be power-complementary, i.e. $|H_0(e^{j\omega})|^2 + |H_1(e^{j\omega})|^2 = 1$
- The name QMF is due to these two important properties

Two-Channel QMF Banks

- The QMF bank is aliase-free per construction
- The distortion function $T(z)$ reduces to

$$T(z) = \frac{1}{2}\{H_0^2(z) - H_1^2(z)\}$$

$$= \frac{1}{2}\{H_0^2(z) - H_0^2(-z)\}$$

- A computationally efficient realization of a two-channel FIR QMF bank is obtained by realizing the FIR filters in polyphase form

Two-Channel QMF Banks

- Now, let the 2-band Type 1 polyphase form of $H_0(z)$ be given by

$$H_0(z) = E_0(z^2) + z^{-1}E_1(z^2)$$

- From the relation $H_1(z) = H_0(-z)$ it follows

$$H_1(z) = E_0(z^2) - z^{-1}E_1(z^2)$$

- The distortion transfer function $T(z)$ is then given by

$$T(z) = 2z^{-1}E_0(z^2)E_1(z^2)$$

Two-Channel QMF Banks

- From the above equations, the analysis filters can be expressed in matrix form as

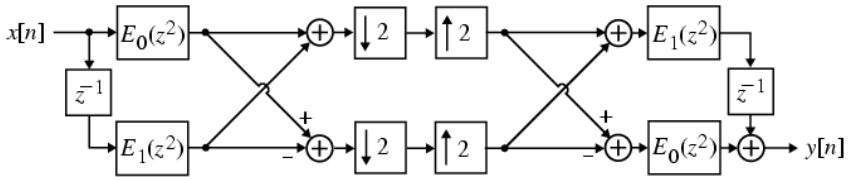
$$\begin{bmatrix} H_0(z) \\ H_1(z) \end{bmatrix} = \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix} \begin{bmatrix} E_0(z^2) \\ z^{-1}E_1(z^2) \end{bmatrix}$$

- Likewise, the synthesis filters are given in matrix form as

$$\begin{bmatrix} G_0(z) & G_1(z) \end{bmatrix} = \begin{bmatrix} z^{-1}E_1(z^2) & E_0(z^2) \end{bmatrix} \begin{bmatrix} 1 & 1 \\ 1 & -1 \end{bmatrix}$$

Two-Channel QMF Banks

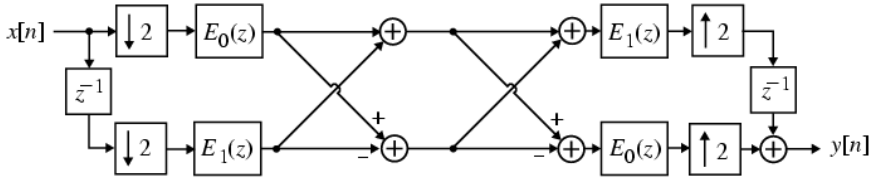
- Using the last two equations we can redraw the QMF bank as shown below



Analysis filter bank **Synthesis filter bank**

Two-Channel QMF Banks

- Utilizing multirate identities, the QMF bank structure can be simplified as shown below



Analysis filter bank **Synthesis filter bank**

Two-Channel QMF Banks

- If $H_0(z)$ is a linear-phase FIR filter, then its polyphase components $E_0(z)$ and $E_1(z)$ are also linear-phase FIR transfer functions
- In this case, the distortion transfer function

$$T(z) = 2z^{-1}E_0(z^2)E_1(z^2)$$

exhibits a linear-phase characteristic

- As a result, the corresponding FIR QMF bank has no phase distortion

Two-Channel QMF Banks

- Example: Assume $H_0(z) = 1 + z^{-1}$

Its polyphase components are simply given by

$$E_0(z^2) = 1, \quad E_1(z^2) = 1$$

and hence

$$H_1(z) = H_0(-z) = E_0(z^2) - z^{-1}E_1(z^2) = 1 - z^{-1}$$

$$G_0(z) = z^{-1}E_1(z^2) + E_0(z^2) = 1 + z^{-1}$$

$$G_1(z) = z^{-1}E_1(z^2) - E_0(z^2) = -1 + z^{-1}$$

Two-Channel QMF Banks

- The distortion transfer function $T(z)$ reduces to

$$T(z) = 2z^{-1}E_0(z^2)E_1(z^2) = 2z^{-1}$$

- Thus, the resulting FIR QMF bank using these trivial filters is of perfect reconstruction type
- However, the only FIR filters satisfying the PR condition are the trivial filters given above
- As a result, it is not possible to design practical FIR QMF banks with perfect reconstruction

QMF Banks with Johnston Filters

- Therefore, the best we can do is to design the prototype LP filter $H_0(z)$ in such a way that the PR condition is satisfied as closely as possible
- Next, we outline an approach to minimize the residual amplitude distortion
- First, let $H_0(z)$ be a linear-phase FIR transfer function of order N, i.e.

$$H_0(z) = \sum_{n=0}^{N} h_0[n]\,z^{-n}$$

QMF Banks with Johnston Filters

- The impuse response of $H_0(z)$ must satisfy the symmetry condition

$$h_0[n] = h_0[N-n]$$

- For the frequency response we can write

$$H_0(e^{j\omega}) = e^{-j\omega N/2}\left|H_0(e^{j\omega})\right|$$

where the magnitude function $\left|H_0(e^{j\omega})\right|$ is an even function of ω

QMF Banks with Johnston Filters

- The frequency response of $T(z)$ can now be written as

$$T(e^{j\omega}) = \frac{e^{-jN\omega}}{2}\{|H_0(e^{j\omega})|^2 - (-1)^N |H_0(e^{j(\pi-\omega)})|^2\}$$

- From the above, it can be seen that if N is even, $T(e^{j\omega}) = 0$ at $\omega = \pi/2$, implying severe amplitude distortion at the output
- Therefore, the filter order N must be odd

QMF Banks with Johnston Filters

- For N odd, we obtain

$$T(e^{j\omega}) = \frac{e^{-jN\omega}}{2}\{|H_0(e^{j\omega})|^2 + |H_0(e^{j(\pi-\omega)})|^2\}$$

$$= \frac{e^{-jN\omega}}{2}\{|H_0(e^{j\omega})|^2 + |H_1(e^{j\omega})|^2\}$$

- Then, the FIR QMF bank will be of perfect reconstruction type if

$$|H_0(e^{j\omega})|^2 + |H_1(e^{j\omega})|^2 = 1$$

QMF Banks with Johnston Filters

- In general, the FIR QMF bank will always exhibit amplitude distortion unless $|T(e^{j\omega})|$ is a constant for all ω
- The degree of distortion is determined by the amount of overlap between their magnitude responses and their stopband attenuations
- Maximum amplitude distortion occurs in the transition band of these two filters

QMF Banks with Johnston Filters

- The distortion can be minimized by controlling the overlap, which in turn can be controlled by appropriately choosing the passband edge of the filter $H_0(z)$
- One way to minimize the amplitude distortion is to iteratively adjust the filter coefficients $h_0[n]$ such that

$$|H_0(e^{j\omega})|^2 + |H_1(e^{j\omega})|^2 \cong 1$$

for all frequencies

QMF Banks with Johnston Filters

- One approach has been proposed by Johnston to design optimal FIR filters for QMF banks
- To this end, an appropriate objective function to be minimized has been chosen
- The objective function involves the
  1. flatness (or ripple energy) of the distortion transfer function $T(z)$, and
  2. stopband energy of $H_0(z)$

QMF Banks with Johnston Filters

- The objective function ϕ is given by

$$\phi = \alpha\phi_1 + (1-\alpha)\phi_2$$

where

$$\phi_1 = \int_{\omega_s}^{\pi} \left|H_0(e^{j\omega})\right|^2 d\omega$$

and

$$\phi_2 = \int_0^{\pi} \left(1 - \left(\left|H_0(e^{j\omega})\right|^2 + \left|H_1(e^{j\omega})\right|^2\right)\right)^2 d\omega$$

where $0 < \alpha < 1$, $\omega_s = \frac{\pi}{2} + \varepsilon$ for small $\varepsilon > 0$

QMF Banks with Johnston Filters

- The technique tries to approximate the power complementary condition while minimizing the stopband energy
- After ϕ has been made very small by the optimization procedure, both ϕ_1 and ϕ_2 will also be very small
- Johnston has designed a family of linear-phase FIR filters for QMF banks and has tabulated their impulse response coefficients

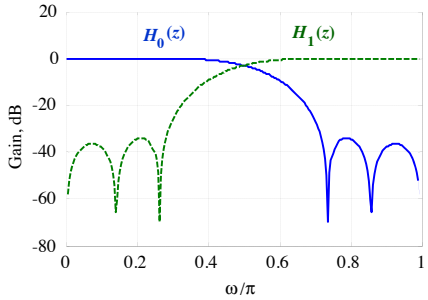
QMF Banks with Johnston Filters

- Example: Length-12 linear-phase LP filter $H_0(z)$ 12B of Johnston with the coefficients

$$h_0[0] = -0.006443977 = h_0[11]$$
$$h_0[1] = 0.02745539 = h_0[10]$$
$$h_0[2] = -0.00758164 = h_0[9]$$
$$h_0[3] = -0.0913825 = h_0[8]$$
$$h_0[4] = 0.09808522 = h_0[7]$$
$$h_0[5] = 0.4807962 = h_0[6]$$

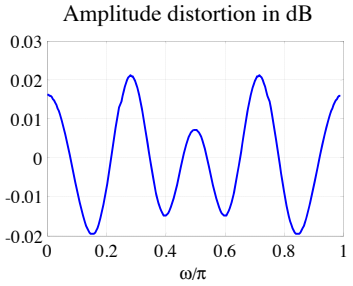
QMF Banks with Johnston Filters

- Gain responses of the LP filter $H_0(z)$ 12B of Johnston and its power-complementary HP filter $H_1(z)$ are shown below



QMF Banks with Johnston Filters

- The amplitude distortion function in dB is shown below



QMF Banks with Johnston Filters

- The stopband edge ω_s of the LP filter is about 0.71π, which corresponds to a transition bandwidth of $(\omega_s - 0.5\pi)/2 = 0.105\pi$
- The minimum stopband attenuation of the two filters is approximately 34 dB
- The amplitude distortion function is very close to 0 dB in both the passbands and stopbands of the two filters, with a peak value of ± 0.02 dB

5.4 Perfect Reconstruction Two-Channel Filter Banks

- A method for the construction of two-channel filter banks to eliminate all three distortions, i.e. aliasing distortion, magnitude distortion, and phase distortion, is derived next
- Since matrix representations are a convenient and a compact way of describing and characterizing filter banks, we first introduce the modulation matrices to derive the perfect reconstruction condition

Modulation Matrices

- To develop the pertinent design equations we observe that the input-output relation of the two-channel filter bank

$$Y(z) = \frac{1}{2}\{H_0(z)G_0(z) + H_1(z)G_1(z)\}X(z) + \frac{1}{2}\{H_0(-z)G_0(z) + H_1(-z)G_1(z)\}X(-z)$$

can be expressed in matrix form as

$$Y(z) = \frac{1}{2}\begin{bmatrix} G_0(z) & G_1(z) \end{bmatrix}\begin{bmatrix} H_0(z) & H_0(-z) \\ H_1(z) & H_1(-z) \end{bmatrix}\begin{bmatrix} X(z) \\ X(-z) \end{bmatrix}$$

Modulation Matrices

- From the previous equation we obtain

$$Y(-z)=\frac{1}{2}[G_0(-z)\;\; G_1(-z)]\begin{bmatrix}H_0(z) & H_0(-z)\\ H_1(z) & H_1(-z)\end{bmatrix}\begin{bmatrix}X(z)\\ X(-z)\end{bmatrix}$$

- Combining the two matrix equations we get

$$\begin{bmatrix}Y(z)\\ Y(-z)\end{bmatrix}=\frac{1}{2}\begin{bmatrix}G_0(z) & G_1(z)\\ G_0(-z) & G_1(-z)\end{bmatrix}\begin{bmatrix}H_0(z) & H_0(-z)\\ H_1(z) & H_1(-z)\end{bmatrix}\begin{bmatrix}X(z)\\ X(-z)\end{bmatrix}$$

$$=\frac{1}{2}\mathbf{G}^{(m)}(z)[\mathbf{H}^{(m)}(z)]^T\begin{bmatrix}X(z)\\ X(-z)\end{bmatrix}$$

Modulation Matrices

- $\mathbf{H}^{(m)}(z)$ is called the analysis modulation matrix and $\mathbf{G}^{(m)}(z)$ is called the synthesis modulation matrix
- The modulation matrices are given by

$$\mathbf{G}^{(m)}(z)=\begin{bmatrix}G_0(z) & G_1(z)\\ G_0(-z) & G_1(-z)\end{bmatrix}$$

$$\mathbf{H}^{(m)}(z)=\begin{bmatrix}H_0(z) & H_1(z)\\ H_0(-z) & H_1(-z)\end{bmatrix}$$

Conditions for Perfect Reconstruction

- Perfect reconstruction requires $Y(z)=z^{-\ell}X(z)$ and correspondingly $Y(-z)=(-z)^{-\ell}X(-z)$
- Substituting these relations in the equation

$$\begin{bmatrix}Y(z)\\ Y(-z)\end{bmatrix}=\frac{1}{2}\mathbf{G}^{(m)}(z)[\mathbf{H}^{(m)}(z)]^T\begin{bmatrix}X(z)\\ X(-z)\end{bmatrix}$$

we conclude that the PR condition is satisfied if

$$\frac{1}{2}\mathbf{G}^{(m)}(z)[\mathbf{H}^{(m)}(z)]^T=\begin{bmatrix}z^{-\ell} & 0\\ 0 & (-z)^{-\ell}\end{bmatrix}$$

Conditions for Perfect Reconstruction

- Thus, knowing the analysis filters $H_0(z)$ and $H_1(z)$, the synthesis filters $G_0(z)$ and $G_1(z)$ can be determined from

$$\mathbf{G}^{(m)}(z)=2\cdot\begin{bmatrix}z^{-\ell} & 0\\ 0 & (-z)^{-\ell}\end{bmatrix}\cdot\left([\mathbf{H}^{(m)}(z)]^T\right)^{-1}$$

- Choosing the synthesis filters according this linear system of equations ensures aliasing cancellation for any choice of analysis filters, i.e. it is alias-free by construction

Conditions for Perfect Reconstruction

- The synthesis filters are given by

$$G_0(z) = \frac{2z^{-\ell}}{\det[\mathbf{H}^{(m)}(z)]} \cdot H_1(-z)$$

$$G_1(z) = -\frac{2z^{-\ell}}{\det[\mathbf{H}^{(m)}(z)]} \cdot H_0(-z)$$

provided $\det[\mathbf{H}^{(m)}(z)] \neq 0$, where $\det[\mathbf{H}^{(m)}(z)]$ is the determinant of the matrix obtained by $\det[\mathbf{H}^{(m)}(z)] = H_0(z)H_1(-z) - H_0(-z)H_1(z)$

Conditions for Perfect Reconstruction

- For FIR analysis filters $H_0(z)$ and $H_1(z)$, the synthesis filters $G_0(z)$ and $G_1(z)$ will also be FIR filters if and only if $\det[\mathbf{H}^{(m)}(z)] = cz^{-k}$ where c is a real number, k is a positive integer
- Then, the solution for the two FIR synthesis filters is given by

$$G_0(z) = \frac{2}{c} z^{-(\ell-k)} H_1(-z)$$

$$G_1(z) = -\frac{2}{c} z^{-(\ell-k)} H_0(-z)$$

Conditions for Perfect Reconstruction

- Let $H_0(z)$ be a FIR filter of odd order N and choose $H_1(z)$ so that $H_1(z) = z^{-N} H_0(-z^{-1})$
- A filter $H_1(z)$ satisfying the above property is called a Conjugate Quadrature Filter (CQF)
- The determinant of the analysis modulation matrix reduces to $\det[\mathbf{H}^{(m)}(z)] = -z^{-N}$ and the PR condition becomes

$$H_0(z)H_0(z^{-1}) + H_0(-z)H_0(-z^{-1}) = 1$$

aka the power-symmetric condition for $H_0(z)$

Conditions for Perfect Reconstruction

- Comparing the above result with the condition $\det[\mathbf{H}^{(m)}(z)] = cz^{-k}$ to obtain FIR filters, we observe that $c = -1$ and $k = N$
- Using $c = -1$, $k = N$ and $H_1(z) = z^{-N} H_0(-z^{-1})$ in

$$G_0(z) = \frac{2}{c} z^{-(\ell-k)} H_1(-z)$$

$$G_1(z) = -\frac{2}{c} z^{-(\ell-k)} H_0(-z)$$

and choosing $\ell = k = N$ we finally obtain

$$G_0(z) = 2z^{-N} H_0(z^{-1}), G_1(z) = 2H_0(-z)$$

Conditions for Perfect Reconstruction

- Thus, if the two-channel filter bank defined by the FIR filters $H_0(z), H_1(z) = z^{-N} H_0(-z^{-1})$, $G_0(z) = 2z^{-N} H_0(z^{-1}), G_1(z) = 2H_0(-z)$ satisfies the power-symmetric condition

$$H_0(z)H_0(z^{-1}) + H_0(-z)H_0(-z^{-1}) = 1$$

  then, it is a perfect reconstruction filter bank
- Note that the factor 2 in $|G_i(e^{j\omega})| = 2 \cdot |H_i(e^{j\omega})|$ preserves the amplitude of the input signal

Orthogonal Filter Banks

- Now, we need to know under what conditions a FIR filter $H_0(z)$ of odd order N exists so that the power-symmetric condition is satisfied
- To check this, we define a product filter $F(z)$

$$F(z) = H_0(z)H_0(z^{-1})$$

- Then, the PR condition becomes

$$F(z) + F(-z) = 1$$

- The spectral factorization of $F(z)$ determines the properties and the type of the filter bank

Orthogonal Filter Banks

- Next, we consider the case if the product filter $F(z) = H_0(z)\, H_0(z^{-1})$ is the z-transform of an autocorrelation sequence $f[n] = h_0[n] * h_0[-n]$
- This requires the design of a valid half-band filter leading to an orthogonal (para-unitary) filter bank
- If the product filter $F(z) = H_0(z)\, G_0(z)$ is the z-transform of a cross-correlation sequence, the resulting filter bank is called bi-orthogonal

Orthogonal Filter Banks

- If $h_0[n]$ has real coefficients, $F(z)$ must be a zero-phase half-band LP filter with a non-negative frequency response, i.e.

$$H_0(e^{j\omega})H_0(e^{-j\omega}) = |H_0(e^{j\omega})|^2 = F(e^{j\omega}) \geq 0$$

- Such a valid half-band filter can be obtained by adding a constant δ to the frequency response of a zero-phase half-band filter $Q(z)$ such that $F(e^{j\omega}) = Q(e^{j\omega}) + \delta \geq 0$ for all ω

Orthogonal Filter Banks

- The length of a valid half-band filter is always of the form $4i-1$, where i is a positive integer, since the order of the spectral factor $H_0(z)$ must be odd
- Zeros of $F(z) = H_0(z)\,H_0(z^{-1})$ appear with mirror-image symmetry in the z-plane with the zeros on the unit circle being of multiplicity 2
- Any appropriate half of these zeros can be grouped to form the spectral factor $H_0(z)$

Orthogonal Filter Banks

- For example, a minimum-phase $H_0(z)$ can be formed by grouping all the zeros inside the unit circle along with half of the zeros on the unit circle
- However, it is not possible to form a spectral factor with linear phase
- The stopband edge frequency is the same for $F(z)$ and $H_0(z)$

Design of Valid Half-Band Filters

- **Steps to design a valid half-band LP filter:**

1. Design a zero-phase half-band LP filter $Q(z)$ of order $2N$, where N is an odd integer, i.e.

$$Q(z) = \sum_{n=-N}^{N} q[n]z^{-n}$$

2. Define the product filter $F(z) = Q(z) + \delta$ to guarantee $F(e^{j\omega}) \geq 0$, where δ is the peak stopband ripple of $Q(e^{j\omega})$

Design of Valid Half-Band Filters

If $q[n]$ denotes the impulse response of $Q(z)$, then the impulse response of $F(z)$ is given by

$$f[n] = \begin{cases} q[n]+\delta, & \text{for } n=0 \\ q[n], & \text{for } n\neq 0 \end{cases}$$

3. Determine the minimum-phase filter $H_0(z)$ of $F(z)$ by solving the spectral factorization task

Note: If the minimum stopband attenuation of $H_0(z)$ is α_s, then the minimum stopband attenuation of $F(z)$ is roughly $2\alpha_s + 6$ dB

Design of Valid Half-Band Filters

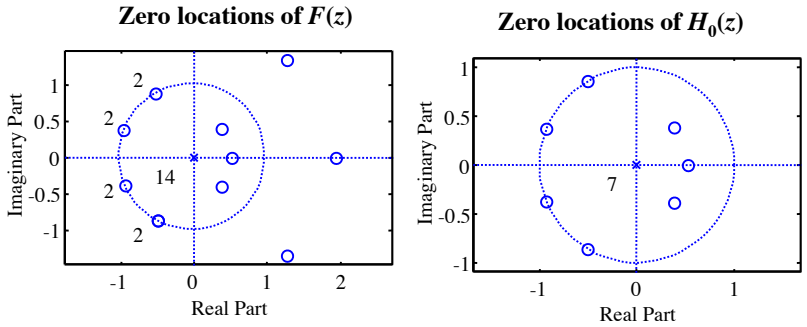
- Example: Design the filter $H_0(z)$ with the specifications $\omega_s = 0.63\pi$ and $\alpha_s = 17\ dB$
- Thus, $F(z)$ has approximately $\alpha_s = 40\ dB$
- The desired stopband ripple is $\delta_s = 0.01$, the passband edge is $\omega_p = \pi - 0.63\pi = 0.37\pi$
- Assume the filter order of $F(z)$ is 14
- This implies that the filter order of $H_0(z)$ is 7, i.e. it is odd as required

Design of Valid Half-Band Filters

- To determine the coefficients of $F(z)$ we add the maximum stopband ripple δ_s to the central coefficient $q[7]$
- Next, we determine the roots of $F(z)$ which should theoretically exhibit mirror-image symmetry with double roots on the unit circle
- Choosing the roots inside the unit circle along with one set of unit circle roots we get the minimum-phase spectral factor, i.e. $H_0(z)$

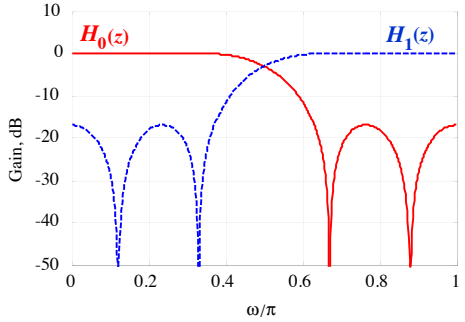
Design of Valid Half-Band Filters

- The zero locations of $F(z)$ and $H_0(z)$ are shown below



Design of Valid Half-Band Filters

- The figure below shows the gain responses of the analysis filters $H_0(z)$ and $H_1(z)$

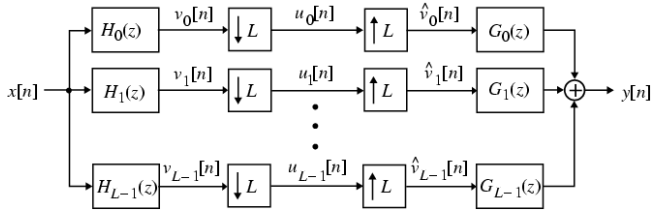


Design of a Two-Channel PR FIR Filter Bank Using MATLAB

- **`firpr2chfb`** can be used to design a perfect reconstruction orthogonal FIR filter bank
- **`[h0,h1,g0,g1]=firpr2chfb(N,fp)`** designs the analysis filters **`h0`** and **`h1`** and the synthesis filters **`g0`** and **`g1`**, whereas the filter order **`N`** of all four filters must be odd, **`fp`** is the passband-edge for the two LP filters h0 and g0

5.5 Analysis of Uniform Multi-Channel Filter Banks

- We generalize the analysis of an uniform two-channel filter bank with more than two channels
- The general structure of an uniform L-channel filter bank is shown below



Input-Output Relations

- The intermediate signals are given by

$$V_k(z) = H_k(z)X(z)$$

$$U_k(z) = \frac{1}{L}\sum_{\ell=0}^{L-1} H_k(z^{1/L}W_L^{\ell})X(z^{1/L}W_L^{\ell})$$

$$\hat{V}_k(z) = U_k(z^L), 0 \le k \le L-1$$

- Vector of down-sampled subband signals $\mathbf{u}(z)$ is

$$\mathbf{u}(z) = \begin{bmatrix} U_0(z) & U_1(z) & \cdots & U_{L-1}(z) \end{bmatrix}^T$$

Input-Output Relations

- Define the modulation vector of the input signal $\mathbf{x}^{(m)}(z)$ as

$$\mathbf{x}^{(m)}(z) = \begin{bmatrix} X(z) & X(zW_L) & \cdots & X(zW_L^{L-1}) \end{bmatrix}^T$$

and the analysis modulation matrix $\mathbf{H}^{(m)}(z)$ as

$$\mathbf{H}^{(m)}(z) = \begin{bmatrix} H_0(z) & H_1(z) & \cdots & H_{L-1}(z) \\ H_0(zW_L) & H_1(zW_L) & \cdots & H_{L-1}(zW_L) \\ \vdots & \vdots & \ddots & \vdots \\ H_0(zW_L^{L-1}) & H_1(zW_L^{L-1}) & \cdots & H_{L-1}(zW_L^{L-1}) \end{bmatrix}$$

Input-Output Relations

- Define the modulation vector of the output signal $\mathbf{y}^{(m)}(z)$ as

$$\mathbf{y}^{(m)}(z) = [Y(z) \quad Y(zW_L) \quad \cdots \quad Y(zW_L^{L-1})]^T$$

and the synthesis modulation matrix $\mathbf{G}^{(m)}(z)$ as

$$\mathbf{G}^{(m)}(z) = \begin{bmatrix} G_0(z) & G_1(z) & \cdots & G_{L-1}(z) \\ G_0(zW_L) & G_1(zW_L) & \cdots & G_{L-1}(zW_L) \\ \vdots & \vdots & \ddots & \vdots \\ G_0(zW_L^{L-1}) & G_1(zW_L^{L-1}) & \cdots & G_{L-1}(zW_L^{L-1}) \end{bmatrix}$$

Input-Output Relations

- Then we can write for the vector of the down-sampled subband signals $\mathbf{u}(z)$

$$\mathbf{u}(z) = \frac{1}{L}[\mathbf{H}^{(m)}(z^{1/L})]^T \mathbf{x}^{(m)}(z^{1/L})$$

- The output of the filter bank is given by

$$Y(z) = \sum_{k=0}^{L-1} G_k(z)\, \hat{V}_k(z)$$

$$= \sum_{k=0}^{L-1} G_k(z)\, U_k(z^L)$$

Input-Output Relations

- In matrix form we can write for the output

$$Y(z) = \mathbf{g}^T(z)\, \mathbf{u}(z^L)$$

where

$$\mathbf{g}(z) = [G_0(z) \quad G_1(z) \quad \cdots \quad G_{L-1}(z)]^T$$

- From this equation, the modulated versions of the output signal are given by

$$Y(zW_L^k) = \mathbf{g}^T(zW_L^k)\mathbf{u}(z^L W_L^{kL})$$

$$= \mathbf{g}^T(zW_L^k)\mathbf{u}(z^L), 0 \le k \le L-1$$

Input-Output Relations

- The modulation vector of the output signal $\mathbf{y}^{(m)}(z)$ can be expressed as

$$\mathbf{y}^{(m)}(z) = \mathbf{G}^{(m)}(z)\mathbf{u}(z^L)$$

- Combining the above and

$$\mathbf{u}(z) = \frac{1}{L}[\mathbf{H}^{(m)}(z^{1/L})]^T \mathbf{x}^{(m)}(z^{1/L})$$

we arrive at

$$\mathbf{y}^{(m)}(z) = \frac{1}{L}\mathbf{G}^{(m)}(z)[\mathbf{H}^{(m)}(z)]^T \mathbf{x}^{(m)}(z)$$

Input-Output Relations

- Using the transfer matrix $\mathbf{T}(z)$ given by

$$\mathbf{T}(z) = \frac{1}{L}\mathbf{G}^{(m)}(z)[\mathbf{H}^{(m)}(z)]^T$$

we can write for the modulation vector of the output signal

$$\mathbf{y}^{(m)}(z) = \mathbf{T}(z)\,\mathbf{x}^{(m)}(z)$$

- $\mathbf{T}(z)$ relates the input $X(z)$ and its modulated versions $X(zW_L^k)$ with the output signal $Y(z)$ and its modulated versions $Y(zW_L^k)$

Alias-Free Condition

- The filter bank is alias-free if and only if the transfer matrix $\mathbf{T}(z)$ is a diagonal matrix of the form

$$\mathbf{T}(z) = \text{diag}[T(z) \quad T(zW_L) \quad \cdots \quad T(zW_L^{L-1})]$$

- The first element $T(z)$ of the above diagonal matrix is the distortion transfer function of the L-channel filter bank, i.e.

$$Y(z) = T(z)X(z)$$

Alias-Free Condition

- From the first row of $\mathbf{y}^{(m)}(z) = \mathbf{T}(z)\,\mathbf{x}^{(m)}(z)$ we arrive at the output $Y(z)$

$$Y(z) = \sum_{\ell=0}^{L-1} A_\ell(z)X(zW_L^\ell)$$

where

$$A_\ell(z) = \frac{1}{L}\sum_{k=0}^{L-1} H_k(zW_L^\ell)G_k(z), \quad 0 \le \ell \le L-1$$

are the aliasing transfer functions

- Thus, the output $Y(z)$ depends on the original input $X(z)$ and its alias components $X(zW_L^\ell)$

Alias-Free Condition

- Clearly, the first term in the relation

$$Y(z) = A_0(z)X(z) + \sum_{\ell=1}^{L-1} A_\ell(z)X(zW_L^\ell)$$

is the alias-free output of the filter bank and the second term is the aliasing component

- Note that the output spectrum is a weighted sum of the input spectrum and its uniformly shifted versions $X(e^{j(\omega-2\pi\ell/L)}), \ell = 1,2,\ldots,L-1$ caused by the sampling rate alterations

Alias-Free Condition

- Aliasing is completely eliminated by forcing the condition

$$A_\ell(z) = 0, \ell = 1,2,\ldots,L-1$$

- With the elimination of the alias terms, the L-channel filter bank satisfies the input-output relation

$$Y(z) = A_0(z)\, X(z)$$

- In this case, the L-channel uniform filter bank becomes a linear and time-invariant system

Alias-Free Condition

- The distortion transfer function $T(z)$ is given by $T(z) = A_0(z) = \frac{1}{L}\sum_{k=0}^{L-1} H_k(z)G_k(z)$
- If $T(z)$ has a constant magnitude, then the filter bank is magnitude-preserving
- If $T(z)$ has a linear phase, then the filter bank is phase-preserving
- If $T(z)$ is a pure delay, the filter bank is a perfect reconstruction filter bank

Alias-Free Condition

- Define $\mathbf{A}(z) = [A_0(z) \quad A_1(z) \quad \cdots \quad A_{L-1}(z)]$, then it follows

$$L \cdot \mathbf{A}(z) = \mathbf{H}^{(m)}(z)\mathbf{g}(z)$$

- The aliasing cancellation condition can now be rewritten as

$$\mathbf{H}^{(m)}(z)\mathbf{g}(z) = \mathbf{t}(z)$$

where

$$\mathbf{t}(z) = [L\ A_0(z) \quad 0 \quad \cdots \quad 0]^T$$

Alias-Free Condition

- Knowing the set of analysis filters $H_k(z)$, we can determine the set of synthesis filters $G_k(z)$ as

$$\mathbf{g}(z) = [\mathbf{H}^{(m)}(z)]^{-1}\mathbf{t}(z)$$

provided $\det \mathbf{H}^{(m)}(z) \neq 0$

- Moreover, a perfect reconstruction uniform L-channel bank results for $T(z) = A_0(z) = z^{-k}$, where k is a positive constant

Polyphase Component Matrices

- Now, consider the *L*-band Type I polyphase representation of the analysis filters

$$H_k(z)=\sum_{\ell=0}^{L-1}z^{-\ell}E_{k\ell}(z^L),\quad 0\le k\le L-1$$

- The matrix equation for the vector of analysis filters is given by

where $$\mathbf{h}(z)=\mathbf{E}^{(\mathrm{p})}(z^L)\mathbf{e}(z)$$

$$\mathbf{h}(z)=[H_0(z)\quad H_1(z)\quad\cdots\quad H_{L-1}(z)]^T$$

$$\mathbf{e}(z)=[1\quad z^{-1}\quad\cdots\quad z^{-(L-1)}]^T$$

Polyphase Component Matrices

- The matrix $\mathbf{E}^{(p)}(z)$ is the Type I polyphase component matrix given by

$$\mathbf{E}^{(p)}(z)=\begin{bmatrix}E_{00}(z) & E_{01}(z) & \cdots & E_{0,L-1}(z)\\ E_{10}(z) & E_{11}(z) & \cdots & E_{1,L-1}(z)\\ \vdots & \vdots & \vdots & \vdots\\ E_{L-1,0}(z) & E_{L-1,1}(z) & \cdots & E_{L-1,L-1}(z)\end{bmatrix}$$

- $E_{k\ell}(z)$ are the L type I polyphase components of the k-th analysis filter $H_k(z)$

Polyphase Component Matrices

- Likewise, we can represent the L synthesis filters in a L-band Type II polyphase form

$$G_k(z)=\sum_{\ell=0}^{L-1}z^{-(L-1-\ell)}R_{\ell k}(z^L),\quad 0\le k\le L-1$$

- The matrix equation for the vector of synthesis filters is given by

where $$\mathbf{g}^T(z)=z^{-(L-1)}\tilde{\mathbf{e}}(z)\mathbf{R}^{(\mathrm{p})}(z^L)$$

$$\mathbf{g}(z)=[G_0(z)\quad G_1(z)\quad\cdots\quad G_{L-1}(z)]^T$$

$$\tilde{\mathbf{e}}(z)=[1\quad z\quad\cdots\quad z^{L-1}]^T=\mathbf{e}^T(z^{-1})$$

Polyphase Component Matrices

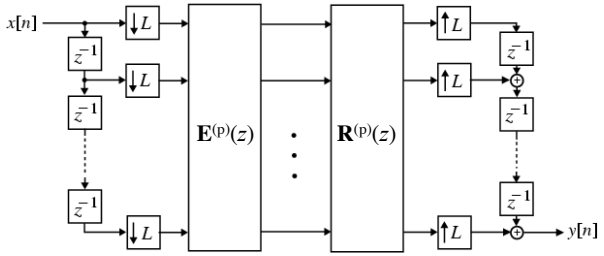
- The matrix $\mathbf{R}^{(p)}(z)$ is the Type II polyphase component matrix given by

$$\mathbf{R}^{(p)}(z)=\begin{bmatrix}R_{00}(z) & R_{01}(z) & \cdots & R_{0,L-1}(z)\\ R_{10}(z) & R_{11}(z) & \cdots & R_{1,L-1}(z)\\ \vdots & \vdots & \vdots & \vdots\\ R_{L-1,0}(z) & R_{L-1,1}(z) & \cdots & R_{L-1,L-1}(z)\end{bmatrix}$$

- $R_{\ell k}(z)$ are the L type II polyphase components of the k-th synthesis filter $G_k(z)$

Polyphase Component Matrices

- After substituting the polyphase component matrices and utilizing the multirate identities we arrive at



- Note the reduced computational requirements obtained by the sampling rate reduction

Perfect Reconstruction Condition

- Using the two polyphase component matrices $\mathbf{E}^{(p)}(z)$ and $\mathbf{R}^{(p)}(z)$, the PR condition reduces to

$$\mathbf{E}^{(p)}(z)\,\mathbf{R}^{(p)}(z) = c\,z^{-k}\,\mathbf{I}$$

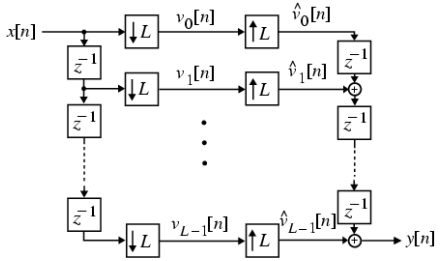
where $\mathbf{I}$ is an $L\times L$ identity matrix, c and k are constants

- The multi-channel filter bank reduces to a delay chain PR system
- Example: Consider the special case

$$H_k(z) = z^{-k},\quad G_k(z) = z^{-(L-1-k)},\quad 0\le k\le L-1$$

Perfect Reconstruction Condition

- The resulting filter bank is shown below



- Even though the filters have trivial frequency responses, this example proves the existence of a multi-channel PR filter bank

Perfect Reconstruction Condition

- Proof: Substituting

$$H_k(z) = z^{-k},\quad G_k(z) = z^{-(L-1-k)},\quad 0\le k\le L-1$$

in

$$A_\ell(z) = \frac{1}{L}\sum_{k=0}^{L-1} H_k(zW_L^{\ell})G_k(z),\quad 0\le \ell\le L-1$$

we obtain

$$A_\ell(z) = z^{-(L-1)}\,\frac{1}{L}\sum_{\ell=0}^{L-1} W_L^{-\ell k},\quad 0\le \ell\le L-1$$

Perfect Reconstruction Condition

- Now, $\frac{1}{L}\sum_{\ell=0}^{L-1} W_L^{-\ell k} = \begin{cases} 1, & \ell = 0 \\ 0, & 1 \le \ell \le L-1 \end{cases}$
- Hence, it follows

$$A_0(z) = 1, \quad A_\ell(z) = 0 \quad \text{for } \ell \neq 0$$

or, equivalently $T(z) = z^{-(L-1)}$

i.e. the filter bank satisfies the PR condition

Perfect Reconstruction Condition

- For a given polyphase matrix of the analysis filter bank, a multi-channel PR filter bank can be designed by constructing a synthesis filter bank with a polyphase matrix given by

$$\mathbf{R}^{(p)}(z) = c\, z^{-k} \left[\mathbf{E}^{(p)}(z)\right]^{-1}$$

i.e., we have to compute the inverse of $\mathbf{E}^{(p)}(z)$

- Thus, FIR solutions for both the analysis and synthesis filters require that the determinants of the polyphase matrices are just delays

Near Perfect Reconstruction

- A lot of works have been done on the design of uniform multi-channel PR filter banks
- However, PR is not absolute necessary in any case, i.e. small distortions can be tolerated
- Thus, a practical approach is to keep all the distortions as small as possible resulting in multi-channel filter banks with near perfect reconstruction (NPR)

Pseudo-QMF Banks

- A widely used approach for the design of NPR filter banks is known as Pseudo-QMF
- The Pseudo-QMF approach includes 3 steps:
  1. Design a LP prototype and derive all other filters by shifts of the frequency response of this prototype
  2. Ensure that the aliasing components of adjacent channels compensate exactly (by power complementary pairs of filters)

Pseudo-QMF Banks

3. Suppress all other (not adjacent) alias spectra by designing the LP prototype with sufficiently high stopband attenuation

- DFT filter banks and cosine-modulated filter banks belongs to the class of Pseudo-QMF filter banks, where all filters are derived from a prototype via modulation

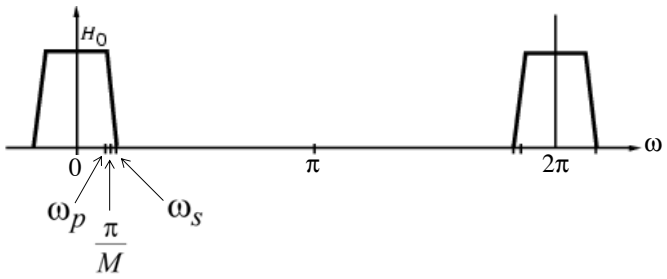
5.6 DFT Filter Banks

- Now, a simple modulation technique to design uniform multi-channel filter banks is outlined
- Let $H_0(z)$ represent a causal lowpass filter with an impulse response $h_0[n]$
- The filter $H_0(z)$ is assumed to be an FIR filter without any loss of generality

$$H_0(z) = \sum_{n=-\infty}^{\infty} h_0[n] z^{-n}$$

DFT Filter Banks

- Assume that $H_0(z)$ has its passband edge ω_p and stopband edge ω_s around π/M, where M is some arbitrary integer, as indicated below



DFT Filter Banks

- Now, consider the transfer function $H_k(z)$ whose impulse response $h_k[n]$ is given by

$$h_k[n] = h_0[n] e^{j2\pi kn/M} = h_0[n] W_M^{-kn}, \quad 0 \le k \le M-1$$

- The transfer function is then given by

$$H_k(z) = \sum_{n=-\infty}^{\infty} h_k[n] z^{-n} = \sum_{n=-\infty}^{\infty} h_0[n] \left(z W_M^k\right)^{-n}$$

or simply

$$H_k(z) = H_0(z W_M^k), \quad 0 \le k \le M-1$$

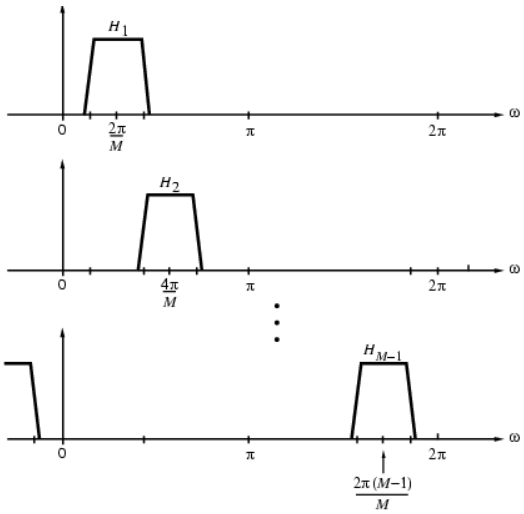
DFT Filter Banks

- Thus, the corresponding frequency response is given by

$$H_k(e^{j\omega}) = H_0(e^{j(\omega - 2\pi k/M)}),\ 0 \le k \le M-1$$

- The frequency response of $H_k(z)$ is obtained by shifting the response of $H_0(z)$ to the right by an amount $2\pi k/M$
- The next figure represents the frequency responses of $H_k(z), 1 \le k \le M-1$

DFT Filter Banks



DFT Filter Banks

- The M filters defined by

$$H_k(z) = H_0(zW_M^k),\ \ 0 \le k \le M-1$$

could be used as the analysis filters in the analysis filter bank or as the synthesis filters in the synthesis filter bank

- Thus, the magnitude responses of all M filters are uniformly shifted versions of that of the prototype LP filter

DFT Filter Banks

- Let the prototype LP transfer function be represented in its M-band polyphase form

$$H_0(z) = \sum_{\ell=0}^{M-1} z^{-\ell} E_\ell(z^M)$$

where $E_\ell(z)$ is the ℓ-th polyphase component of $H_0(z)$, that is

$$E_\ell(z) = \sum_{n=0}^{\infty} e_\ell[n] z^{-n} = \sum_{n=0}^{\infty} h_0[\ell + nM] z^{-n},$$

$$0 \le \ell \le M-1$$

DFT Filter Banks

- Substituting z with zW_M^k in the expression for $H_0(z)$ we arrive at the M-band polyphase form of $H_k(z)$

$$H_k(z) = \sum_{\ell=0}^{M-1} z^{-\ell} W_M^{-k\ell} E_\ell(z^M W_M^{kM})$$

$$= \sum_{\ell=0}^{M-1} z^{-\ell} W_M^{-k\ell} E_\ell(z^M), \quad 0 \le k \le M-1$$

where we have used the identity $W_M^{kM} = 1$

DFT Filter Banks

- The above equation can be written in matrix form as given by

$$H_k(z) = \begin{bmatrix} 1 & W_M^{-k} & W_M^{-2k} & \cdots & W_M^{-(M-1)k} \end{bmatrix} \begin{bmatrix} E_0(z^M) \\ z^{-1}E_1(z^M) \\ z^{-2}E_2(z^M) \\ \vdots \\ z^{-(M-1)}E_{M-1}(z^M) \end{bmatrix}$$

$$0 \le k \le M-1$$

DFT Filter Banks

- All transfer functions can be combined into a single matrix equation as

$$\begin{bmatrix} H_0(z) \\ H_1(z) \\ H_2(z) \\ \vdots \\ H_{M-1}(z) \end{bmatrix} = \underbrace{\begin{bmatrix} 1 & 1 & 1 & \cdots & 1 \\ 1 & W_M^{-1} & W_M^{-2} & \cdots & W_M^{-(M-1)} \\ 1 & W_M^{-2} & W_M^{-4} & \cdots & W_M^{-2(M-1)} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & W_M^{-(M-1)} & W_M^{-2(M-1)} & \cdots & W_M^{-(M-1)^2} \end{bmatrix}}_{M\,\mathbf{D}^{-1}} \begin{bmatrix} E_0(z^M) \\ z^{-1}E_1(z^M) \\ z^{-2}E_2(z^M) \\ \vdots \\ z^{-(M-1)}E_{M-1}(z^M) \end{bmatrix}$$

- $\mathbf{D}$ is the $M \times M$ DFT matrix

DFT Filter Banks

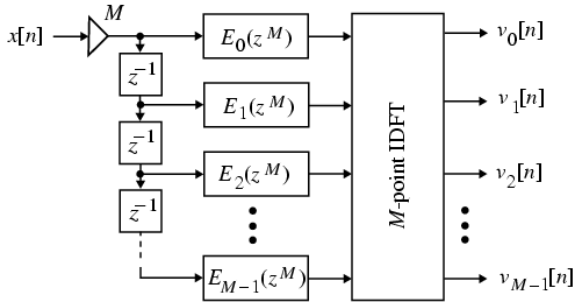
- $E_i(z^M)$ can be expressed in terms of $H_0(z)$ and its modulated versions $H_k(z)$

$$\begin{bmatrix} E_0(z^M) \\ z^{-1}E_1(z^M) \\ z^{-2}E_2(z^M) \\ \vdots \\ z^{-(M-1)}E_{M-1}(z^M) \end{bmatrix} = \frac{1}{M}\mathbf{D} \begin{bmatrix} H_0(z) \\ H_1(z) \\ H_2(z) \\ \vdots \\ H_{M-1}(z) \end{bmatrix} = \frac{1}{M}\mathbf{D} \begin{bmatrix} H_0(z) \\ H_0(z\,W_M) \\ H_0(z\,W_M^2) \\ \vdots \\ H_0(z\,W_M^{M-1}) \end{bmatrix}$$

- The above can be used to derive the polyphase components of the DFT analysis filter bank

DFT Filter Banks

- The first step towards a polyphase-based uniform M-channel DFT analysis filter bank is shown below

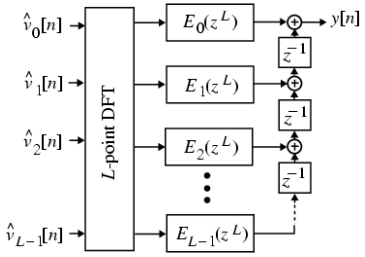


DFT Filter Banks

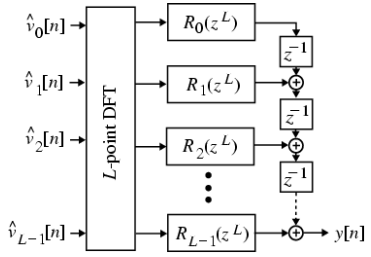
- Note that we can move the down-samplers at the output of the IDFT before the IDFT and using the multirate identity for down-sampling
- The computational complexity of a polyphase-based M-band uniform DFT filter bank is much smaller than that of a direct implementation
- In a similar way we can obtain a polyphase-based L-band uniform DFT synthesis filter bank as shown next

DFT Filter Banks

- First steps towards a polyphase-based uniform L-channel DFT synthesis filter bank



Uniform DFT synthesis filter bank with Type I transposed polyphase form



Uniform DFT synthesis filter bank with Type II polyphase form

5.7 Cosine-Modulated Filter Banks

- In cosine-modulated filter banks, the complex modulation of a DFT filter bank is replaced by cosine modulation
- Cosine-modulated filter banks are very popular due to their real-valued nature, e.g. in the MP3 audio codec
- Near perfect reconstruction is easily achieved by designing an appropriate prototype

Cosine-Modulated Filter Banks

- Now, let $P_0(z) = \sum_{n=0}^{N} p_0[n] z^{-n}$

  denote the prototype LP filter with real coefficients and a cutoff at $\pi/2L$
- First, we generate a set of filters $Q_k(z)$ from $P_0(z)$ by complex modulation at frequencies $(2k+1)\pi/2L = (k+0.5)\pi/L$ such that

$$Q_k(z) = P_0(zW_{2L}^{k+0.5}),\ 0 \le k \le 2L-1$$

Cosine-Modulated Filter Banks

- Note: $W_{2L} = e^{-j\pi/L}$
- According to the complex modulation, the set of filters $Q_k(z), 0 \le k \le 2L-1$ have complex-valued impulse responses
- The response of $Q_0(z)$ is a right-shifted version of the response of $P_0(z)$ shifted by an amount of $\pi/2L$

Cosine-Modulated Filter Banks

- As a result of the shift, we obtain

$$|Q_k(e^{j\omega})| = |Q_{2L-1-k}(e^{j\omega})|$$

- The impulse response of $Q_{2L\text{-}1\text{-}k}(z)$ is the complex conjugated version of the impulse response of $Q_k(z)$
- Thus, the filter pair $Q_k(z)$ and $Q_{2L\text{-}1\text{-}k}(z)$ can be combined to create a filter with real-valued impulse response coefficients

Cosine-Modulated Filter Banks

- Define the intermediate transfer functions

$$U_k(z) = c_k Q_k(z), \quad V_k(z) = c_k^* Q_{2L-1-k}(z)$$

- Then, the analysis filters are formed according to

$$H_k(z) = \sum_{n=0}^{N} h_k[n] z^{-n}$$

$$= a_k U_k(z) + a_k^* V_k(z), \quad 0 \le k \le L-1$$

Cosine-Modulated Filter Banks

- Likewise, the L synthesis filters are formed according to

$$G_k(z) = \sum_{n=0}^{N} g_k[n] z^{-n}$$
$$= b_k U_k(z) + b_k^* V_k(z), \quad 0 \le k \le L-1$$

- The constants a_k, b_k, and c_k are chosen to provide alias cancellation between adjacent channels

Cosine-Modulated Filter Banks

- For example, a_k, b_k, and c_k are chosen as

$$a_k = e^{j\frac{\pi}{4}(-1)^k}, b_k = a_k^*$$

$$c_k = e^{-j\frac{\pi}{2L}(k+0.5)N}, 0 \le k \le L-1$$

  where N is the order of the prototype LP filter
- The chosen constants lead to closed-form expressions for the coefficients of the analysis and synthesis filters

Cosine-Modulated Filter Banks

- The coefficients for the analysis and synthesis filters are related to the prototype filter $P_0(z)$ by cosine modulation

$$h_k[n] = 2p_0[n]\cos\left[(k+\tfrac{1}{2})(n-\tfrac{N}{2})\tfrac{\pi}{L} + (-1)^k\tfrac{\pi}{4}\right]$$

$$g_k[n] = 2p_0[n]\cos\left[(k+\tfrac{1}{2})(n-\tfrac{N}{2})\tfrac{\pi}{L} - (-1)^k\tfrac{\pi}{4}\right]$$

- If $P_0(z)$ has linear phase, then the distortion transfer function $T(z)$ has also linear phase

Cosine-Modulated Filter Banks

- The aliasing between adjacent channels is cancelled structurally
- Thus, the design of the cosine-modulated filter bank reduces to the design of the L-th band prototype LP filter $P_0(z)$ so that all distortions are eliminated
- This can be accomplished by reflecting the requirements in the design process of $P_0(z)$

Cosine-Modulated Filter Banks

- The first constraint in the design of $P_0(z)$ is

$$\phi_1 = \int_0^{\pi/L} \left(\left|P_0(e^{j\omega})\right|^2 + \left|P_0(e^{j(\omega-\pi/L)})\right|^2 - 1\right) d\omega \stackrel{!}{=} \min$$

- Recall that amplitude distortion arises if the magnitude response of the distortion transfer function $\left|T(e^{j\omega})\right|$ is not acceptable flat
- The integral reflects the passband ripple in the above limits because $\left|T(e^{j\omega})\right|$ has period π/L

Cosine-Modulated Filter Banks

- The second constraint in the design of $P_0(z)$ is

$$\phi_2 = \int_{\frac{\pi}{2L}+\varepsilon}^{\pi} \left(\left|P_0(e^{j\omega})\right|^2 d\omega \stackrel{!}{=} \min$$

- The integral is an appropriate measure of the stopband energy of $P_0(z)$
- The choice of $\varepsilon < 0$ depends on the acceptable transition bandwidth

Cosine-Modulated Filter Banks

- We can optimize the coefficients of $P_0(z)$ by minimizing the composite objective function

$$\phi = \alpha\,\phi_1 + (1-\alpha)\,\phi_2 \stackrel{!}{=} \min$$

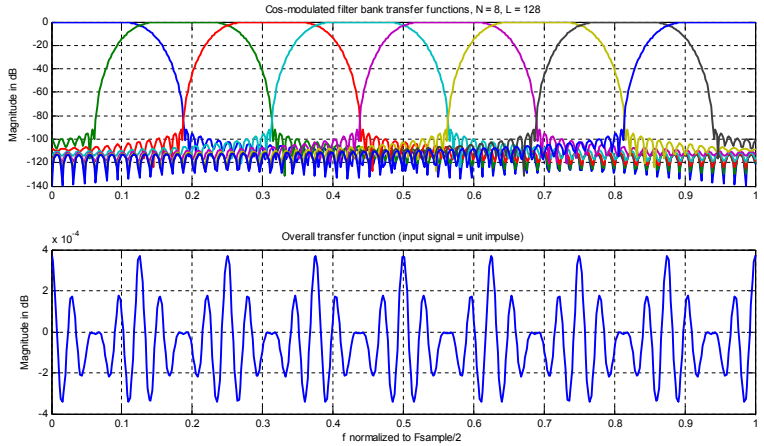
where α is a tradeoff parameter with $0 < \alpha < 1$

- Standard nonlinear optimization methods can be used to search optimal coefficients
- However, further methods are known to avoid the problematic nonlinear optimization

Cosine-Modulated Filter Bank Design Example

- Example: Design of an uniform 8th-band cosine-modulated filter bank using a prototype lowpass FIR filter of length 128
- The M-file **`unicmfb`** (provided by Doblinger) is used to design all the transfer functions of the filter bank
- The design results are illustrated next

Cosine-Modulated Filter Bank Design Example

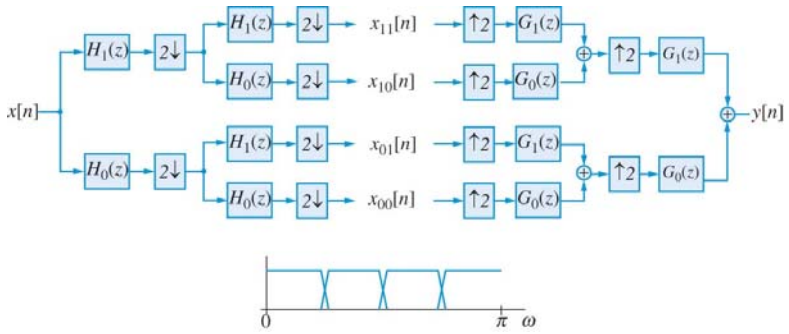


5.8 Tree-Structured Filter Banks

- In most applications one needs a signal decomposition into more than two frequency bands
- A simple way of designing the required filters is to build cascades of two-channel filter banks
- The most widely used structures are the uniform-band tree structure, and the octave-band tree structure

Tree-Structured Filter Banks

- A two-level uniform-band tree-structured filter bank is shown below

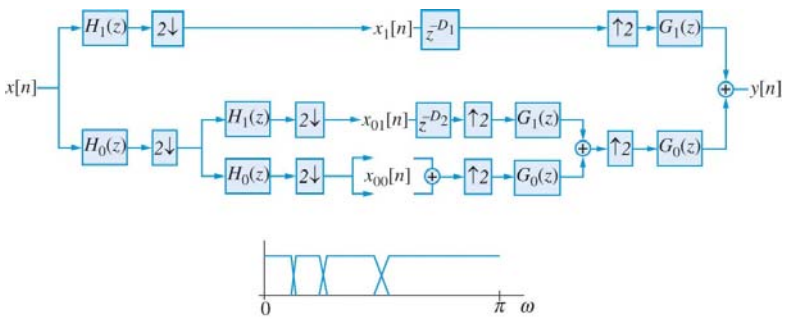


Tree-Structured Filter Banks

- The input signal is split into two equal frequency bands by a LP half-band filter $H_0(z)$ and a HP half-band filter $H_1(z)$
- After down-sampling by a factor of 2, each sub-band is again divided into two sub-bands using the same two-channel filter banks
- The magnitude responses of the four analysis filters for the two-level tree are also shown

Tree-Structured Filter Banks

- A two-level octave-band tree-structured filter bank is shown below



Tree-Structured Filter Banks

- At first, the input signal is split into two equal frequency bands
- The further non-uniform decomposition is chosen such that it is best matched to the given problem
- Such multi-resolution decompositions are closely related to the Discrete Wavelet Transform (DWT) and widely used e.g. in image compression

Tree-Structured Filter Banks

- Tree-structured filter banks are a supplement to uniform multi-channel filter banks
- In all cases, PR is easily obtained if the two-channel filter bank used as a basic building block provides PR
- Note that the number of channels is limited due to the tree structure
- Additionally, the overall delay of the system is rather long

Tree-Structured Filter Bank Design in MATLAB

- DSP System Toolbox provides system objects for dyadic analysis and synthesis filter banks
  - `dsp.DyadicAnalysisFilterBank`
  - `dsp.DyadicSynthesisFilterBank`
- The dyadic filter banks can be constructed either with a symmetric (uniform-band) or an asymmetric (octave-band) tree structure