APPLIED SIGNAL PROCESSING
DIGITAL FILTERS

Digital filters are discrete-time linear systems

\[ \{ x[n] \} \rightarrow G \rightarrow \{ y[n] \} \]

Impulse response:

\[ y[n] = h[0]x[n] + h[1]x[n - 1] + \cdots \]
DIGITAL FILTER TYPES

• FIR (Finite Impulse Response) filters
  - have finite memory; output depends only on a finite number of inputs
  - modeled by (weighted) moving average models

\[ y[n] = h[0]x[n] + h[1]x[n-1] + \cdots + h[N-1]x[n+1-N] \]

• IIR (Infinite Impulse Response) filters
  - have infinite memory; output depends on an infinite number of (past) inputs
  - modeled by difference equations

\[ y[n] + a_1 y[n-1] + \cdots + a_N y[n-N] = b_0 x[n] + b_1 x[n-1] + \cdots + b_M x[n-M] \]
SIMPLE EXAMPLE OF IIR FILTER

First-order filter \( y[n] - ay[n - 1] = bx[n] \)
\[ \Rightarrow \]
\[ y[n] = ay[n - 1] + bx[n] \]
\[ = a^2y[n - 2] + abx[n - 1] + bx[n] \]
\[ = a^3y[n - 3] + a^2bx[n - 2] + abx[n - 1] + bx[n] \]
\[ \vdots \]
\[ = bx[n] + abx[n - 1] + a^2bx[n - 2] + \cdots + a^kbx[n - k] + \cdots \]
PROS AND CONS

FIR filters:

+ In general less sensitive to round-off errors, always stable

+ Straightforward to achieve linear phase property

- High filter order (many taps) may be required to achieve filter specs

- FIR filters exist only in the digital world: only IIR analogue filters can be constructed
IIR filters:

+ More flexible than FIR: given filter specs can be achieved with a simpler IIR filter, with fewer parameters than corresponding FIR filter

- Feedback from past filter outputs may give sensitivity to round-off errors, or even stability problems

- Linear phase properties can be achieved only approximately
FREQUENCY RESPONSE OF FILTERS

\[ \{ x[n] \} \rightarrow G \rightarrow \{ y[n] \} \]

For input

\[ x[n] = \cos(\omega n) \]

the output is

\[ y[n] = A(\omega) \cos(\omega n + \varphi(\omega)) \]

\( A(\omega) \): filter gain or magnitude

\( \varphi(\omega) \): filter phase
Reason: the output can always be written as

\[ y[n] = h[0] \cos(\omega n) + h[1] \cos(\omega n - \omega) + \cdots + h[k] \cos(\omega n - \omega k) + \cdots \]

\[ = A(\omega) \cos(\omega n + \varphi(\omega)) \]

The functions \( A(\omega), \varphi(\omega) \) form the frequency response of a filter

Matlab routine:

\texttt{freqz(B,A,W)}

where \( B = [b_0 a_2 \ldots b_M] \), \( A = [1 a_1 a_2 \ldots a_N] \),
\( W \): vector with frequencies (if specified)
Low-pass filter specs:

- maximum deviation in passband, \( 1 - \delta_p \leq A(f) \leq 1 + \delta_p \)
- damping in stopband, \( A(f) \leq \delta_s \)
- width of transition band (normalized)
Filter specs often defined in dB units:

- maximum deviation in passband: $A_p = 20 \log(1 + \delta_p)$ (dB)
  Note: $A_p \approx 8.7\delta_p$ for small $\delta_p$

- damping in stopband: $A_s = -20 \log \delta_s$ (dB)
LINEAR PHASE FILTERS

Observation: Phase shift in passband may cause signal distortion.

**Linear phase property:** All frequencies in passband delayed by the same time

\[ \{x[n]\} \xrightarrow{G} \{y[n]\} \]

For input \( x[n] = \cos(\omega n) \) we have the output

\[ y[n] = A(\omega) \cos(\omega n + \varphi(\omega)) \]

To avoid phase distortion we require that

\[ y[n] = A(\omega) \cos(\omega(n - \tau)) \]
in passband.

This is achieved if the filter gives phase shift

$$\varphi(\omega) = -\omega \cdot \tau$$

PHASE SHOULD BE A LINEAR FUNCTION OF FREQUENCY IN PASSBAND!

Linear phase property can be achieved with symmetric FIR filters

Linear property can be achieved only approximately with IIR filters
LINEAR PHASE FIR FILTERS

Linear phase property is achieved by symmetric FIR filters:

\[ y[n] = h[0]x[n] + h[1]x[n - 1] + \cdots \]
\[ + h[1]x[n + 2 - N] + h[0]x[n + 1 - N], \quad n = 0, 1, 2, \ldots, M \]

where \( h[k] = h[N - 1 - k], \quad k = 0, 1, \ldots, N - 1 \)

For input \( x[n] = \cos(\omega n) \) the output is

\[ y[n] = A(\omega) \cos(\omega n - \omega(N - 1)/2) \]
\[ = A(\omega) \cos(\omega(n - (N - 1)/2)) \]

⇒ Each frequency component delayed by \((N - 1)/2\) time steps.
REASON:
Pairwise combination of terms $k$ and $N - 1 - k$ gives

$$h[k](x[n-k] + x[n + 1 - N + k]) =$$

$$= h[k]\left(\cos(\omega(n-k)) + \cos(\omega(n + 1 - N + k))\right)$$

$$= h[k]\left(\cos(\omega(n - \frac{N-1}{2}) + \omega(\frac{N-1}{2} - k))
+ \cos(\omega(n - \frac{N-1}{2}) - \omega(\frac{N-1}{2} - k))\right)$$

$$= h[k]2\cos(\omega(\frac{N-1}{2} - k)) \cdot \cos(\omega(n - \frac{N-1}{2}))$$

where we have used

$$\cos(\alpha \pm \beta) = \cos \alpha \cdot \cos \beta \mp \sin \alpha \cdot \sin \beta$$
IIR FILTER PROTOTYPES

Good filter performance requires a sufficiently long memory in order to allow distinct response to different frequency components. In FIR filters this implies a large number of taps.

IIR filters have infinite memory due to recursiveness ⇒ simpler filter for given magnitude performance specifications

Standard IIR filters:

- Butterworth
- Chebyshev
- Elliptic
Butterworth filters

Digital low-pass Butterworth filter with cut-off frequency $\omega_c = 2\pi f_c$ has frequency response magnitude

$$|B_N(e^{j\omega})| = \left[ \frac{1}{1 + \left[\frac{\tan(\omega/2)}{\tan(\omega_c/2)}\right]^{2N}} \right]^{1/2}$$

- monotonically varying gain in both passband and stopband

Matlab routines:

- `[B,A]=butter(N,Wn,'ftype')` designs filter
- `[N,Wn]=buttord(Wp,Ws,Rp,Rs)` computes required minimum order
Chebyshev filters

- Chebyshev filter type I has equiripple gain in passband and monotonically varying gain in stopband.

- Chebyshev filter type II has monotonically varying gain in passband and equiripple gain in stopband.
Elliptic filters

- Equiripple frequency response in both passband and stopband
- Achieve a given set of magnitude specifications with the lowest frequency order
- However, elliptic filters have less linear phase response than Butterworth or Chebyshev filters
COMPUTATION OF FILTER OUTPUT

Difference equation

\[ y[n] + a_1 y[n - 1] + \cdots + a_N y[n - N] = b_0 x[n] + b_1 x[n - 1] + \cdots + b_M x[n - M] \]

can be computed using Matlab routine

\[ y = \text{filter}(B, A, x) \]

where

\( x \): input sequence,

\( B = [b_0 \ b_1 \ \cdots \ b_M] \) and

\( A = [1 \ a_1 \ a_2 \ \cdots \ a_N] \)
**FILTER IMPLEMENTATION**

**IMPLEMENTATION OF FIR FILTERS**

The operation

\[ y[n] = h[0]x[n] + h[1]x[n-1] + \cdots + h[N-1]x[n+1-N], \quad n = 0, 1, 2, \ldots, M \]

is a *convolution* between the impulse response sequence \( \{h[k]\} = \{h[0], h[1], \ldots, h[N-1]\} \) and the input sequence \( \{x[n]\} = \{x[0], x[1], \ldots, x[M-1]\} \)

Often denoted as

\[ \{y[n]\} = \{h[k]\} \ast \{x[n]\} \]
Computational burden: calculation of output sequence requires \( NM \) flops

Example:

Filter length \( N = 1000 \), sequence length \( M = 10000 \) requires 10 million flops

Efficient software for DSPs exist

In Matlab: \( y = \text{conv}(h,x) \)

or

\( y = \text{filter}(h,1,x) \) (cf. above)
FFT CONVOLUTION (’high-speed convolution’)

IMPORTANT PROPERTY:

Convolution corresponds to polynomial multiplication:

If we define the polynomials

\[ Y(z) = y[0] + y[1]z^{-1} + \cdots + y[M-1]z^{-M+1} \]
\[ X(z) = x[0] + x[1]z^{-1} + \cdots + x[M-1]z^{-M+1} \]
\[ H(z) = h[0] + h[1]z^{-1} + \cdots + h[N-1]z^{-N+1} \]
then we see that

\[ H(z)X(z) = h[0]x[0] + \left( h[0]x[1] + h[1]x[0] \right)z^{-1} + \cdots \]

\[ + \left( h[0]x[n] + h[1]x[n-1] + \cdots + h[N-1]x(n+1-N) \right)z^{-n} + \cdots \]

\[ = y[0] + y[1]z^{-1} + \cdots + y[M-1]z^{-M+1} \]

\[ = Y(z) \]

⇒ the convolution \( \{y[n]\} = \{h[k]\} \ast \{x[n]\} \) can be represented as

\[ Y(z) = H(z)X(z) \]

\( X(z) \): \textit{z-transform} of sequence \( \{x[n]\} \)

\( Y(z) \): \textit{z-transform} of sequence \( \{y[n]\} \)

\( H(z) \): \textit{transfer function} of the system having impulse response \( h[0], h[1], \ldots \)
Relation between convolution and polynomial multiplication leads to an efficient implementation of convolution

**FFT CONVOLUTION ('high-speed convolution')**

The Fourier transform of \( \{x[n]\} \) is precisely a polynomial in \( z = e^{-j2\pi/N} \).

Recall the Fourier transform of \( \{x[n]\} \):

\[
X[k] = x[0] + x[1]e^{-j2\pi k/M} + \cdots + x[M - 1]e^{-j2\pi(M-1)k/M} \\
= x[0] + x[1]e^{-j2\pi k/M} + \cdots + x[M - 1] \left(e^{-j2\pi k/M}\right)^{M-1}
\]

Comparison with \( X(z) \) shows that:

\[
X[k] = X(z_k), \quad z_k = e^{-j2\pi k/M}
\]

Hence \( X[k] \) evaluates \( X(z) \) at \( M \) points \( z_k, k = 0, 1, \ldots, M-1 \).
Hence we can compute the convolution as follows:

1. Compute FFT $X[k]$ of sequence $\{x[n]\}$. Then $X[k] = X(z)$, $z = e^{-j2\pi k/M}$.

2. Compute FFT $H[k]$ of sequence $\{h[n]\}$. Use zero padding to obtain sequences of the same length. Then $H[k] = H(z)$, $z = e^{-j2\pi k/N}$.

3. Compute $Y[k]$ as the element-wise product $Y[k] = H[k]X[k]$. Then $Y[k] = Y(z)$, $z = e^{-j2\pi k/M}$.

4. Compute inverse Fourier transform $\{y[n]\}$ of $\{Y[k]\}$. Then $\{y[n]\}$ is the convolution $\{y[n]\} = \{h[k]\} \ast \{x[n]\}$.
IMPLEMENTATION OF IIR FILTERS

Recursive structure:

\[ y[n] = a_1 y[n - 1] + \cdots + a_N y[n - N] = \]
\[ = b_0 x[n] + b_1 x[n - 1] + \cdots + b_M x[n - M] \]

Makes IIR filters sensitive to round-off errors if implemented naively!

IIR FILTERS SHOULD ALWAYS BE IMPLEMENTED AS A CASCADE STRUCTURE CONSISTING OF A SERIES COUPLING OF SECOND ORDER SYSTEMS (SOS):

\[ Y(z) = H(z)X(z) = H_L(z) \cdot H_{L-1}(z) \cdots H_1(z)X(z) \]
or

\[ Y_1(z) = H_1(z)X(z) \]
\[ Y_2(z) = H_2(z)Y_1(z) \]
\[ \vdots \]
\[ Y(z) = H_L(z)Y_{L-1}(z) \]

where \( H_k(z) \) are second-order systems:

\[ H_1(z) : \quad y_1[n] + a_{11}y_1[n-1] + a_{12}y_1[n-2] = b_{10}x[n] + b_{11}x[n-1] \]
\[ H_2(z) : \quad y_2[n] + a_{21}y_2[n-1] + a_{22}y_2[n-2] = b_{20}y_1[n] + b_{21}y_1[n-1] \]
\[ \vdots \]
\[ H_k(z) : \quad y_k[n] + a_{k1}y_k[n-1] + a_{k2}y_k[n-2] = b_{k0}y_{k-1}[n] + b_{k1}y_{k-1}[n-1] \]
\[ \vdots \]
\[ H_L(z) : \quad y_L[n] + a_{L1}y_L[n-1] + a_{L2}y_L[n-2] = b_{L0}y_{L-1}[n] + b_{L1}y_{L-1}[n-1] \]