
DSP & Digital Filters

Mike Brookes

▷ **1: Introduction**

Organization

Signals

Processing

Syllabus

Sequences

Time Scaling

z-Transform

**Region of
Convergence**

z-Transform examples

Rational z-Transforms

Rational example

Inverse z-Transform

MATLAB routines

Summary

1: Introduction

Organization

1: Introduction

▷ Organization

Signals

Processing

Syllabus

Sequences

Time Scaling

z-Transform

Region of

Convergence

z-Transform examples

Rational z-Transforms

Rational example

Inverse z-Transform

MATLAB routines

Summary

- ☐ 18 lectures: feel free to ask questions
- ☐ Textbook:
 - (a) Mitra “Digital Signal Processing” ISBN:0071289461 £35 covers most of the course except for some of the multirate stuff
 - (b) Harris “Multirate Signal Processing” ISBN:0137009054 £58 covers multirate material in more detail but less rigour than Mitra
- ☐ Lecture slides available via Blackboard or on my website:
<http://www.ee.ic.ac.uk/hp/staff/dmb/courses/dspdf/dspdf.htm>
- ☐ Prerequisites: 3rd year DSP - attend lectures if dubious
- ☐ Exam + Formula Sheet

Signals

1: Introduction

Organization

▷ Signals

Processing

Syllabus

Sequences

Time Scaling

z-Transform

Region of Convergence

z-Transform examples

Rational z-Transforms

Rational example

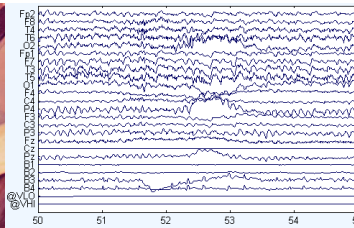
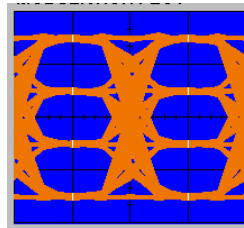
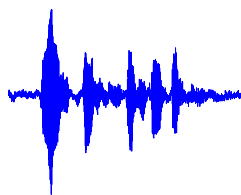
Inverse z-Transform

MATLAB routines

Summary

- A signal is a numerical quantity that is a function of one or more independent variables such as time or position.
- Real-world signals are analog and vary continuously and take continuous values.
- Digital signals are sampled at discrete times and are quantized to a finite number of discrete values
- We will mostly consider one-dimensionalal real-valued signals with regular sample instants; except in a few places, we will ignore the quantization.
 - Extension to multiple dimensions and complex-valued signals is mostly straightforward.

Examples:



Processing

1: Introduction

Organization

Signals

▷ Processing

Syllabus

Sequences

Time Scaling

z-Transform

Region of
Convergence

z-Transform examples

Rational z-Transforms

Rational example

Inverse z-Transform

MATLAB routines

Summary

- ☐ Aims to “improve” a signal in some way or extract some information from it
- ☐ Examples:
 - Modulation/demodulation
 - Coding and decoding
 - Interference rejection and noise suppression
 - Signal detection, feature extraction
- ☐ We are concerned with linear, time-invariant processing

Syllabus

1: Introduction

Organization

Signals

Processing

▷ Syllabus

Sequences

Time Scaling

z-Transform

Region of
Convergence

z-Transform examples

Rational z-Transforms

Rational example

Inverse z-Transform

MATLAB routines

Summary

Main topics:

- ☐ Introduction/Revision (2 lectures)
- ☐ Discrete Time Systems (2 lectures)
- ☐ Filter Design (6 lectures)
- ☐ Multirate Fundamentals (3 lectures)
- ☐ Multirate Filters (2 lectures)
- ☐ Subband processing (3 lectures)

Sequences

1: Introduction

Organization

Signals

Processing

Syllabus

▷ Sequences

Time Scaling

z-Transform

Region of

Convergence

z-Transform examples

Rational z-Transforms

Rational example

Inverse z-Transform

MATLAB routines

Summary

We denote the n^{th} sample of a signal as $x[n]$ where $-\infty < n < +\infty$ and the entire sequence as $\{x[n]\}$ although we will often omit the braces.

Special sequences:

- **Unit step:** $u[n] = \begin{cases} 1 & n \geq 0 \\ 0 & \text{otherwise} \end{cases}$
- **Unit impulse:** $\delta[n] = \begin{cases} 1 & n = 0 \\ 0 & \text{otherwise} \end{cases}$
- **Condition:** $\delta_{\text{condition}}[n] = \begin{cases} 1 & \text{condition is true} \\ 0 & \text{otherwise} \end{cases}$ (e.g. $u[n] = \delta_{n \geq 0}$)
- **Right-sided:** $x[n] = 0$ for $n < N_{min}$
- **Left-sided:** $x[n] = 0$ for $n > N_{max}$
- **Finite length:** $x[n] = 0$ for $n \notin [N_{min}, N_{max}]$
- **Causal:** $x[n] = 0$ for $n < 0$, **Anticausal:** $x[n] = 0$ for $n > 0$
- **Finite Energy:** $\sum_{n=-\infty}^{\infty} |x[n]|^2 < \infty$ (e.g. $x[n] = n^{-1}u[n-1]$)
- **Absolutely Summable:** $\sum_{n=-\infty}^{\infty} |x[n]| < \infty \Rightarrow$ **Finite energy**

Time Scaling

1: Introduction

Organization

Signals

Processing

Syllabus

Sequences

▷ Time Scaling

z-Transform

Region of

Convergence

z-Transform examples

Rational z-Transforms

Rational example

Inverse z-Transform

MATLAB routines

Summary

For sampled signals, the n^{th} sample is at time $t = nT = \frac{n}{f_s}$ where $f_s = \frac{1}{T}$ is the sample frequency.

Often easiest to scale time so that $f_s = 1$ Hz. E.g. to design a 1 kHz low-pass filter for $f_s = 44.1$ kHz we can design a 0.0227 Hz filter for $f_s = 1$ Hz.

To scale back to real-world values: *Every quantity* of dimension $(\text{Time})^n$ is multiplied by T^n (or equivalently by f_s^{-n}). Thus all *times* are multiplied by T and all *frequencies* and *angular frequencies* by T^{-1} .

We use Ω for “real” angular frequencies and ω for normalized angular frequency. The units of ω are “*radians per sample*”.

Warning: Several MATLAB routines scale time so that $f_s = 2$ Hz. Weird, non-standard and irritating.

z-Transform

1: Introduction

Organization

Signals

Processing

Syllabus

Sequences

Time Scaling

▷ z-Transform

Region of Convergence

z-Transform examples

Rational z-Transforms

Rational example

Inverse z-Transform

MATLAB routines

Summary

The z-Transform converts a sequence, $\{x[n]\}$, into a function of an arbitrary complex-valued variable z , $X(z)$.

Why do it?

- Complex functions are easier to manipulate than sequences
- Useful operations on sequences correspond to simple operations on the z-transform:
 - addition, multiplication, scalar multiplication, time-shift, convolution
- Definition: $X(z) = \sum_{n=-\infty}^{+\infty} x[n]z^{-n}$

Region of Convergence

1: Introduction

Organization

Signals

Processing

Syllabus

Sequences

Time Scaling

z-Transform

Region of Convergence

z-Transform examples

Rational z-Transforms

Rational example

Inverse z-Transform

MATLAB routines

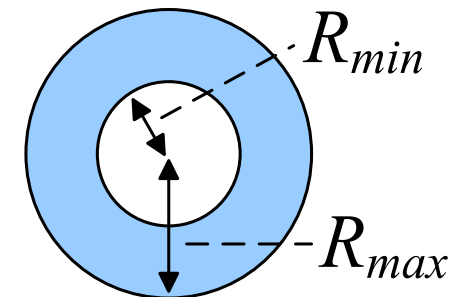
Summary

The set of z for which $X(z)$ converges is its *Region of Convergence* (ROC).

Complex analysis \Rightarrow : the ROC of a power series (if it exists at all) is always an annular region of the form $0 \leq R_{min} < |z| < R_{max} \leq \infty$.

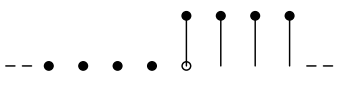
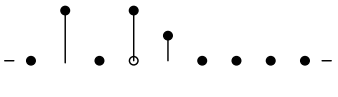
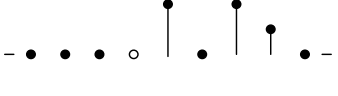
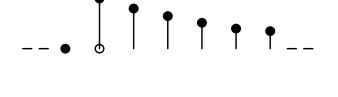

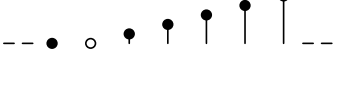
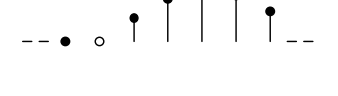

$X(z)$ will always **converge absolutely** inside the ROC and may converge on some, all, or none of the boundary.

- “**converge absolutely**” $\Leftrightarrow \sum_{n=-\infty}^{+\infty} |x[n]z^{-n}| < \infty$
- **finite** $\Leftrightarrow R_{min} = 0, R_{max} = \infty$
 - ROC may include either, both or none of 0 and ∞
- **absolutely summable** $\Leftrightarrow X(z)$ converges for $|z| = 1$.
- **right-sided** $\Leftrightarrow R_{max} = \infty$
 - **causal** $\Leftrightarrow X(\infty)$ converges
- **left-sided** $\Leftrightarrow R_{min} = 0$
 - **anticausal** $\Leftrightarrow X(0)$ converges



z-Transform examples

The sample at $n = 0$ is indicated by an open circle.

$u[n]$		$\frac{1}{1-z^{-1}}$	$1 < z \leq \infty$
$x[n]$		$2z^2 + 2 + z^{-1}$	$0 < z < \infty$
$x[n-3]$		$z^{-3} (2z^2 + 2 + z^{-1})$	$0 < z \leq \infty$
$\alpha^n u[n]_{\alpha=0.8}$		$\frac{1}{1-\alpha z^{-1}}$	$\alpha < z \leq \infty$
$-\alpha^n u[-n-1]$		$\frac{1}{1-\alpha z^{-1}}$	$0 \leq z < \alpha$
$nu[n]$		$\frac{z^{-1}}{1-2z^{-1}+z^{-2}}$	$1 < z \leq \infty$
$\sin(\omega n)u[n]_{\omega=0.5}$		$\frac{z^{-1} \sin(\omega)}{1-2z^{-1} \cos(\omega)+z^{-2}}$	$1 < z \leq \infty$
$\cos(\omega n)u[n]_{\omega=0.5}$		$\frac{1-z^{-1} \cos(\omega)}{1-2z^{-1} \cos(\omega)+z^{-2}}$	$1 < z \leq \infty$

Note: Examples 4 and 5 have the same z-transform but different ROCs.

Geometric Progression: $\sum_{n=q}^r \alpha^n z^{-n} = \frac{\alpha^q z^{-q} - \alpha^{r+1} z^{-r-1}}{1 - \alpha z^{-1}}$

Rational z-Transforms

1: Introduction

Organization

Signals

Processing

Syllabus

Sequences

Time Scaling

z-Transform

Region of Convergence

z-Transform examples

Rational

▷ z-Transforms

Rational example

Inverse z-Transform

MATLAB routines

Summary

Most z-transforms that we will meet are **rational polynomials** with real coefficients, usually one polynomial in z^{-1} divided by another.

$$G(z) = g \frac{\prod_{m=1}^M (1 - z_m z^{-1})}{\prod_{k=1}^K (1 - p_k z^{-1})} = g z^{K-M} \frac{\prod_{m=1}^M (z - z_m)}{\prod_{k=1}^K (z - p_k)}$$

Completely defined by the **poles**, **zeros** and **gain**.

The **absolute values** of the poles define the ROCs:

$\exists R + 1$ different ROCs

where R is the number of distinct pole magnitudes.

Note: There are $K - M$ zeros or $M - K$ poles at $z = 0$ (easy to overlook)

Rational example

1: Introduction

Organization

Signals

Processing

Syllabus

Sequences

Time Scaling

z-Transform

Region of Convergence

z-Transform examples

Rational z-Transforms

▷ Rational example

Inverse z-Transform

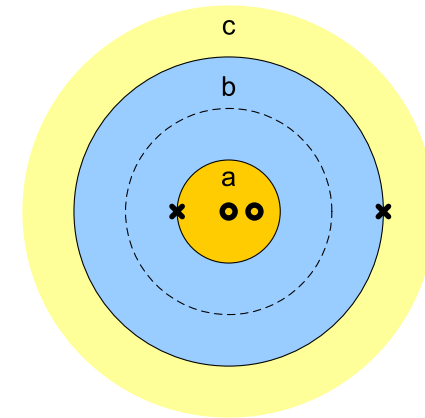
MATLAB routines

Summary

$$G(z) = \frac{8-2z^{-1}}{4-4z^{-1}-3z^{-2}}$$

Poles/Zeros: $G(z) = \frac{2z(z-0.25)}{(z+0.5)(z-1.5)}$

\Rightarrow **Poles** at $z = \{-0.5, +1.5\}$,
Zeros at $z = \{0, +0.25\}$



Partial Fractions: $G(z) = \frac{0.75}{1+0.5z^{-1}} + \frac{1.25}{1-1.5z^{-1}}$

ROC	ROC	$\frac{0.75}{1+0.5z^{-1}}$	$\frac{1.25}{1-1.5z^{-1}}$	$G(z)$
a	$0 \leq z < 0.5$			
b	$0.5 < z < 1.5$			
c	$1.5 < z \leq \infty$			

Inverse z-Transform

1: Introduction

Organization

Signals

Processing

Syllabus

Sequences

Time Scaling

z-Transform

Region of Convergence

z-Transform examples

Rational z-Transforms

Rational example

Inverse ▷ z-Transform

MATLAB routines

Summary

$g[n] = \frac{1}{2\pi j} \oint G(z) z^{n-1} dz$ where the integral is anti-clockwise around a circle within the ROC, $z = Re^{j\theta}$.

Proof:

$$\begin{aligned} \frac{1}{2\pi j} \oint G(z) z^{n-1} dz &= \frac{1}{2\pi j} \oint \left(\sum_{m=-\infty}^{\infty} g[m] z^{-m} \right) z^{n-1} dz \\ &\stackrel{(i)}{=} \sum_{m=-\infty}^{\infty} g[m] \frac{1}{2\pi j} \oint z^{n-m-1} dz \\ &\stackrel{(ii)}{=} \sum_{m=-\infty}^{\infty} g[m] \delta[n-m] = g[n] \end{aligned}$$

(i) depends on the circle with radius R lying within the ROC

(ii) Cauchy's theorem: $\frac{1}{2\pi j} \oint z^{k-1} dz = \delta[k]$ for $z = Re^{j\theta}$ anti-clockwise.

$$\begin{aligned} \frac{dz}{d\theta} = jRe^{j\theta} &\Rightarrow \frac{1}{2\pi j} \oint z^{k-1} dz = \frac{1}{2\pi j} \int_{\theta=0}^{2\pi} R^{k-1} e^{j(k-1)\theta} \times jRe^{j\theta} d\theta \\ &= \frac{R^k}{2\pi} \int_{\theta=0}^{2\pi} e^{jk\theta} d\theta \\ &= R^k \delta(k) = \delta(k) \end{aligned}$$

In practice use a combination of partial fractions and table of z-transforms.

MATLAB routines

1: Introduction

Organization

Signals

Processing

Syllabus

Sequences

Time Scaling

z-Transform

Region of

Convergence

z-Transform examples

Rational z-Transforms

Rational example

Inverse z-Transform

▷ MATLAB routines

Summary

tf2zp,zp2tf	$\frac{b(z^{-1})}{a(z^{-1})} \leftrightarrow \{z_m, p_k, g\}$
residuez	$\frac{b(z^{-1})}{a(z^{-1})} \rightarrow \sum_k \frac{r_k}{1-p_k z^{-1}}$
tf2sos,sos2tf	$\frac{b(z^{-1})}{a(z^{-1})} \leftrightarrow \prod_l \frac{b_{0,l}+b_{1,l}z^{-1}+b_{2,l}z^{-2}}{1+a_{1,l}z^{-1}+a_{2,l}z^{-2}}$
zp2sos,sos2zp	$\{z_m, p_k, g\} \leftrightarrow \prod_l \frac{b_{0,l}+b_{1,l}z^{-1}+b_{2,l}z^{-2}}{1+a_{1,l}z^{-1}+a_{2,l}z^{-2}}$
zp2ss,ss2zp	$\{z_m, p_k, g\} \leftrightarrow \begin{cases} x' = Ax + Bu \\ y = Cx + Du \end{cases}$
tf2ss,ss2tf	$\frac{b(z^{-1})}{a(z^{-1})} \leftrightarrow \begin{cases} x' = Ax + Bu \\ y = Cx + Du \end{cases}$

Summary

1: Introduction

Organization

Signals

Processing

Syllabus

Sequences

Time Scaling

z-Transform

Region of Convergence

z-Transform examples

Rational z-Transforms

Rational example

Inverse z-Transform

MATLAB routines

▷ Summary

- **Time scaling:** assume $f_s = 1$ Hz so $-\pi < \omega \leq \pi$
- **z-transform:** $X(z) = \sum_{n=-\infty}^{+\infty} x[n]z^{-n}$
- **ROC:** $0 \leq R_{min} < |z| < R_{max} \leq \infty$
 - **Causal:** $\infty \in \text{ROC}$
 - **Absolutely summable:** $|z| = 1 \in \text{ROC}$
- **Inverse z-transform:** $g[n] = \frac{1}{2\pi j} \oint G(z)z^{n-1}dz$
 - **Not unique** unless ROC is specified
 - Use **partial fractions** and/or a **table**

For further details see Mitra:1 & 6.

▷ **2: Three Different
Fourier Transforms**

Fourier Transforms

**Convergence of
DTFT**

DTFT Properties

DFT Properties

Symmetries

Parseval's Theorem

Convolution

Sampling Process

Zero-Padding

Phase Unwrapping

Summary

MATLAB routines

2: Three Different Fourier Transforms

Fourier Transforms

- 2: Three Different Fourier Transforms
 - ▷ Fourier Transforms
 - Convergence of DTFT
 - DTFT Properties
 - DFT Properties
 - Symmetries
 - Parseval's Theorem
 - Convolution
 - Sampling Process
 - Zero-Padding
 - Phase Unwrapping
 - Summary
 - MATLAB routines

Three different Fourier Transforms:

- Continuous-Time Fourier Transform (CTFT): $x(t) \rightarrow X(j\Omega)$
- Discrete-Time Fourier Transform (DTFT): $x[n] \rightarrow X(e^{j\omega})$
- Discrete Fourier Transform (DFT): $x[n] \rightarrow X[k]$

Forward Transform

Inverse Transform

CTFT	$X(j\Omega) = \int_{-\infty}^{\infty} x(t)e^{-j\Omega t} dt$	$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(j\Omega)e^{j\Omega t} d\Omega$
DTFT	$X(e^{j\omega}) = \sum_{-\infty}^{\infty} x[n]e^{-j\omega n}$	$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega})e^{j\omega n} d\omega$
DFT	$X[k] = \sum_0^{N-1} x[n]e^{-j2\pi \frac{kn}{N}}$	$x[n] = \frac{1}{N} \sum_0^{N-1} X[k]e^{j2\pi \frac{kn}{N}}$

We use Ω for “real” and $\omega = \Omega T$ for “normalized” angular frequency.
Nyquist frequency is $\Omega = 2\pi \frac{f_s}{2} = \frac{\pi}{T}$ and $\omega = \pi$.

For “power signals” (energy \propto time interval), CTFT & DTFT are unbounded.

Fix this by normalizing:

$$X(j\Omega) = \lim_{A \rightarrow \infty} \frac{1}{2A} \int_{-A}^A x(t)e^{-j\Omega t} dt$$
$$X(e^{j\omega}) = \lim_{A \rightarrow \infty} \frac{1}{2A} \sum_{-A}^A x[n]e^{-j\omega n}$$

Convergence of DTFT

2: Three Different Fourier Transforms

Fourier Transforms Convergence of ▷ DTFT

DTFT Properties

DFT Properties

Symmetries

Parseval's Theorem

Convolution

Sampling Process

Zero-Padding

Phase Unwrapping

Summary

MATLAB routines

DTFT: $X(e^{j\omega}) = \sum_{-\infty}^{\infty} x[n]e^{-j\omega n}$ does not converge for all $x[n]$.

Consider the finite sum: $X_K(e^{j\omega}) = \sum_{-K}^K x[n]e^{-j\omega n}$

Strong Convergence:

$x[n]$ absolutely summable $\Rightarrow X(e^{j\omega})$ converges **uniformly**

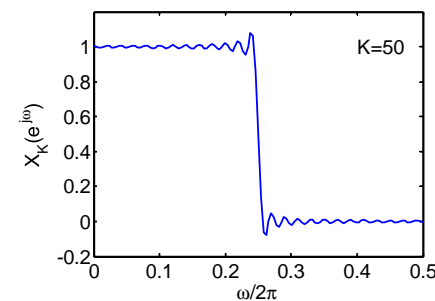
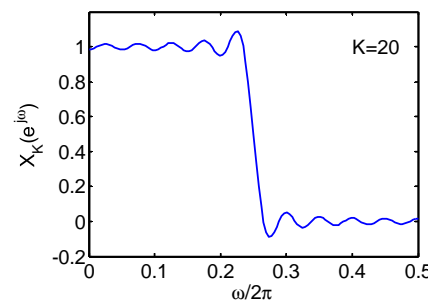
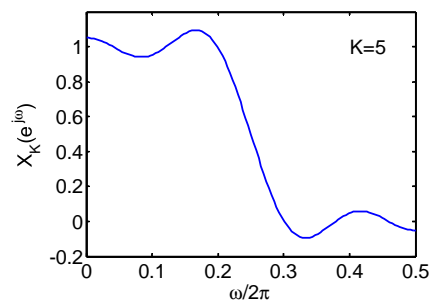
$$\sum_{-\infty}^{\infty} |x[n]| < \infty \Rightarrow \sup_{\omega} |X(e^{j\omega}) - X_K(e^{j\omega})| \xrightarrow{K \rightarrow \infty} 0$$

Weaker convergence:

$x[n]$ finite energy $\Rightarrow X(e^{j\omega})$ converges in **mean square**

$$\sum_{-\infty}^{\infty} |x[n]|^2 < \infty \Rightarrow \frac{1}{2\pi} \int_{-\pi}^{\pi} |X(e^{j\omega}) - X_K(e^{j\omega})|^2 d\omega \xrightarrow{K \rightarrow \infty} 0$$

Example: $x[n] = \frac{\sin 0.5\pi n}{\pi n}$



Gibbs phenomenon:

Converges at each ω as $K \rightarrow \infty$ but peak error does not get smaller.

DTFT Properties

2: Three Different Fourier Transforms

Fourier Transforms Convergence of DTFT

▷ DTFT Properties

DFT Properties

Symmetries

Parseval's Theorem

Convolution

Sampling Process

Zero-Padding

Phase Unwrapping

Summary

MATLAB routines

DTFT: $X(e^{j\omega}) = \sum_{-\infty}^{\infty} x[n]e^{-j\omega n}$

- **DTFT** is **periodic** in ω : $X(e^{j(\omega+2m\pi)}) = X(e^{j\omega})$ for integer m .

- **DTFT** is the **z-Transform** evaluated at the point $e^{j\omega}$:

$$X(z) = \sum_{-\infty}^{\infty} x[n]z^{-n}$$

DTFT converges iff the ROC includes $|z| = 1$.

- **DTFT** is the same as the CTFT of a signal comprising **impulses at the sample times** (dirac delta functions) :

$$x_{\delta}(t) = \sum x[n]\delta(t - nT) = x(t) \times \sum_{-\infty}^{\infty} \delta(t - nT)$$

Equivalent to multiplying a continuous $x(t)$ by an impulse train.

Proof: $X(e^{j\omega}) = \sum_{-\infty}^{\infty} x[n]e^{-j\omega n}$

$$\sum_{n=-\infty}^{\infty} x[n] \int_{-\infty}^{\infty} \delta(t - nT) e^{-j\omega \frac{t}{T}} dt$$

$$\stackrel{(i)}{=} \int_{-\infty}^{\infty} \sum_{n=-\infty}^{\infty} x[n]\delta(t - nT) e^{-j\omega \frac{t}{T}} dt$$

$$\stackrel{(ii)}{=} \int_{-\infty}^{\infty} x_{\delta}(t) e^{-j\Omega t} dt$$

(i) OK if $\sum_{-\infty}^{\infty} |x[n]| < \infty$. (ii) use $\omega = \Omega T$.

DFT Properties

2: Three Different Fourier Transforms

Fourier Transforms Convergence of DTFT

DTFT Properties

▷ DFT Properties

Symmetries

Parseval's Theorem

Convolution

Sampling Process

Zero-Padding

Phase Unwrapping

Summary

MATLAB routines

$$\text{DFT: } X[k] = \sum_0^{N-1} x[n] e^{-j2\pi \frac{kn}{N}}$$

$$\text{DTFT: } X(e^{j\omega}) = \sum_{-\infty}^{\infty} x[n] e^{-j\omega n}$$

Case 1: $x[n] = 0$ for $n \notin [0, N-1]$

DFT is the same as DTFT at $\omega_k = \frac{2\pi}{N}k$.

The $\{\omega_k\}$ are uniformly spaced from $\omega = 0$ to $\omega = 2\pi \frac{N-1}{N}$.

DFT is the z-Transform evaluated at N equally spaced points around the unit circle beginning at $z = 1$.

Case 2: $x[n]$ is periodic with period N

DFT equals the normalized DTFT

$$X[k] = \lim_{K \rightarrow \infty} \frac{N}{2K+1} \times X_K(e^{j\omega_k})$$

$$\text{where } X_K(e^{j\omega}) = \sum_{-K}^K x[n] e^{-j\omega n}$$

Symmetries

2: Three Different
Fourier Transforms

Fourier Transforms
Convergence of
DTFT

DTFT Properties

DFT Properties

▷ Symmetries

Parseval's Theorem

Convolution

Sampling Process

Zero-Padding

Phase Unwrapping

Summary

MATLAB routines

If $x[n]$ has a special property then $X(e^{j\omega})$ and $X[k]$ will have corresponding properties as shown in the table (and vice versa):

One domain	Other domain
Discrete	Periodic
Symmetric	Symmetric
Antisymmetric	Antisymmetric
Real	Conjugate Symmetric
Imaginary	Conjugate Antisymmetric
Real + Symmetric	Real + Symmetric
Real + Antisymmetric	Imaginary + Antisymmetric

Symmetric: $x[n] = x[-n]$
 $X(e^{j\omega}) = X(e^{-j\omega})$
 $X[k] = X[(-k)_{\text{mod } N}] = X[N - k]$ for $k > 0$

Conjugate Symmetric: $x[n] = x^*[-n]$

Conjugate Antisymmetric: $x[n] = -x^*[-n]$

Parseval's Theorem

2: Three Different Fourier Transforms

Fourier Transforms Convergence of DTFT

DTFT Properties

DFT Properties

Symmetries

Parseval's ▷ Theorem

Convolution

Sampling Process

Zero-Padding

Phase Unwrapping

Summary

MATLAB routines

Fourier transforms preserve “energy”

$$\text{CTFT} \quad \int |x(t)|^2 dt = \frac{1}{2\pi} \int |X(j\Omega)|^2 d\Omega$$

$$\text{DTFT} \quad \sum_{-\infty}^{\infty} |x[n]|^2 = \frac{1}{2\pi} \int_{-\pi}^{\pi} |X(e^{j\omega})|^2 d\omega$$

$$\text{DFT} \quad \sum_0^{N-1} |x[n]|^2 = \frac{1}{N} \sum_0^{N-1} |X[k]|^2$$

More generally, they actually preserve complex inner products:

$$\sum_0^{N-1} x[n]y^*[n] = \frac{1}{N} \sum_0^{N-1} X[k]Y^*[k]$$

Unitary matrix viewpoint for DFT:

If we regard \mathbf{x} and \mathbf{X} as vectors, then $\mathbf{X} = \mathbf{F}\mathbf{x}$ where \mathbf{F} is a symmetric matrix defined by $f_{k+1,n+1} = e^{-j2\pi \frac{kn}{N}}$.

The inverse DFT matrix is $\mathbf{F}^{-1} = \frac{1}{N}\mathbf{F}^H$
equivalently, $\mathbf{G} = \frac{1}{\sqrt{N}}\mathbf{F}$ is a unitary matrix with $\mathbf{G}^H\mathbf{G} = \mathbf{I}$.

Convolution

2: Three Different Fourier Transforms

Fourier Transforms Convergence of DTFT

DTFT Properties

DFT Properties

Symmetries

Parseval's Theorem

▷ Convolution

Sampling Process

Zero-Padding

Phase Unwrapping

Summary

MATLAB routines

DTFT: Convolution \rightarrow Product

$$x[n] = g[n] * h[n] = \sum_{k=-\infty}^{\infty} g[k]h[n-k]$$

$$\Rightarrow X(e^{j\omega}) = G(e^{j\omega})H(e^{j\omega})$$

DFT: Circular convolution \rightarrow Product

$$x[n] = g[n] \circledast_N h[n] = \sum_{k=0}^{N-1} g[k]h[(n-k)_{\text{mod } N}]$$

$$\Rightarrow X[k] = G[k]H[k]$$

DTFT: Product \rightarrow Circular Convolution $\div 2\pi$

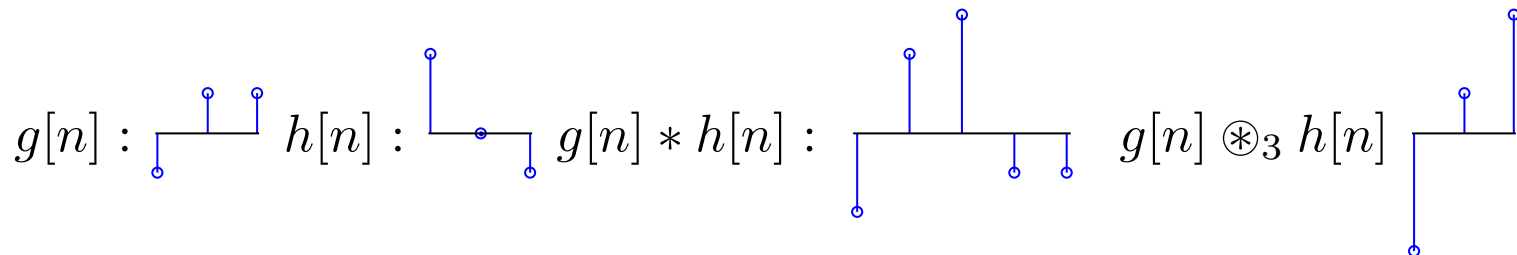
$$y[n] = g[n]h[n]$$

$$\Rightarrow Y(e^{j\omega}) = \frac{1}{2\pi} G(e^{j\omega}) \circledast H(e^{j\omega}) = \frac{1}{2\pi} \int_{-\pi}^{\pi} G(e^{j\theta})H(e^{j(\omega-\theta)})d\theta$$

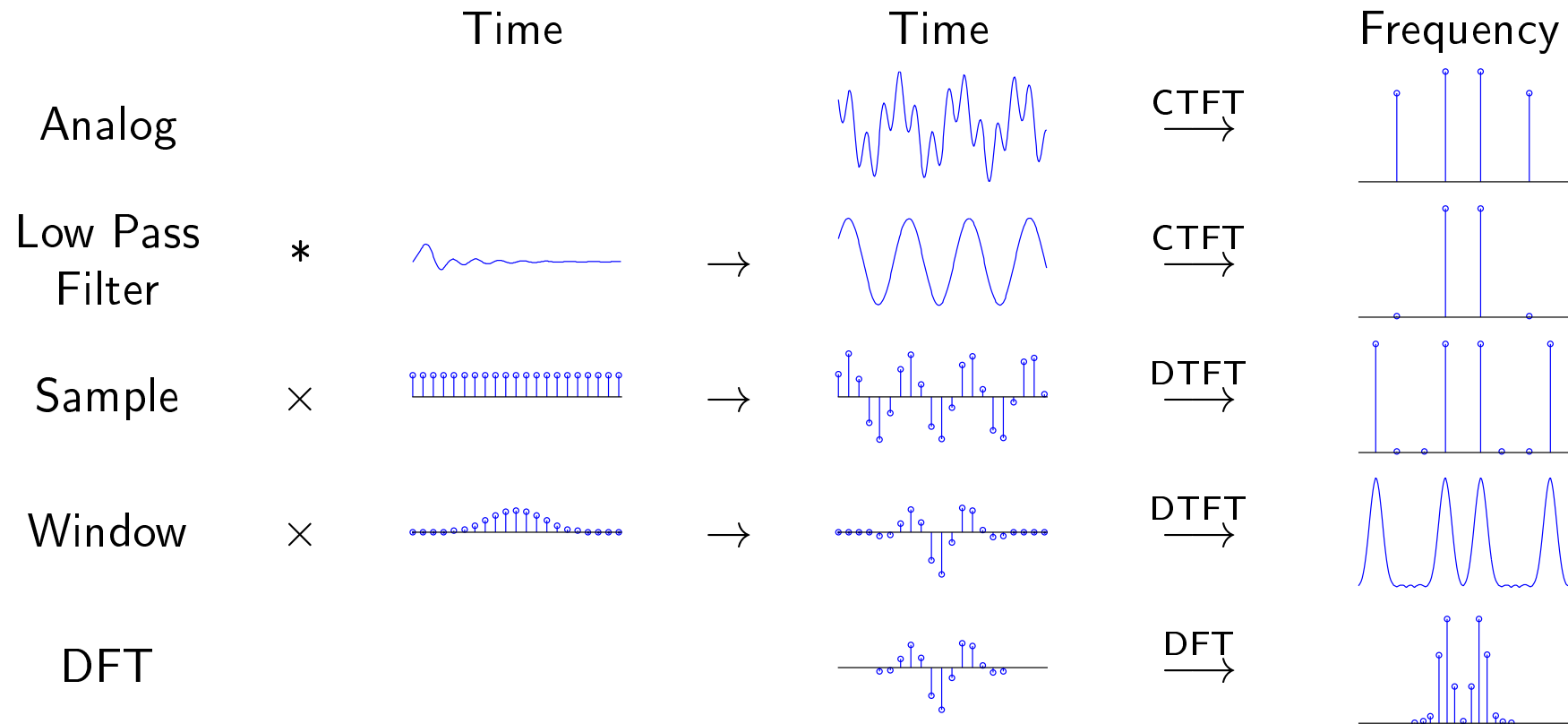
DFT: Product \rightarrow Circular Convolution $\div N$

$$y[n] = g[n]h[n]$$

$$\Rightarrow X[k] = \frac{1}{N} G[k] \circledast_N H[k]$$



Sampling Process



Zero-Padding

2: Three Different Fourier Transforms

Fourier Transforms Convergence of DTFT

DTFT Properties

DFT Properties

Symmetries

Parseval's Theorem

Convolution

Sampling Process

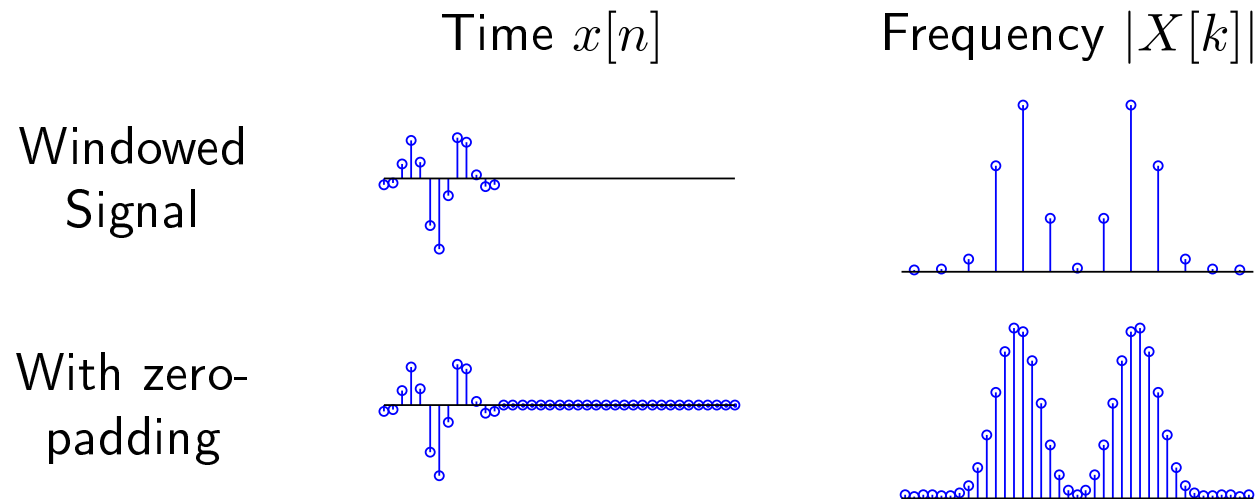
▷ Zero-Padding

Phase Unwrapping

Summary

MATLAB routines

Zero padding means added extra zeros onto the end of $x[n]$ before performing the DFT.



- Zero-padding causes the DFT to evaluate the DTFT at more values of ω_k . Denser frequency samples.
- Width of the peaks remains constant: determined by the length and shape of the window.
- Smoother graph but increased frequency resolution is an illusion.

Phase Unwrapping

2: Three Different Fourier Transforms

Fourier Transforms Convergence of DTFT

DTFT Properties

DFT Properties

Symmetries

Parseval's Theorem

Convolution

Sampling Process

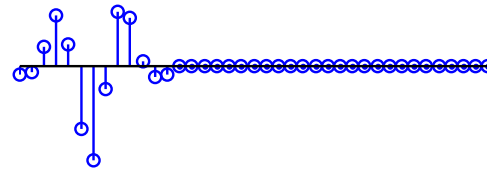
Zero-Padding

▷ Phase Unwrapping

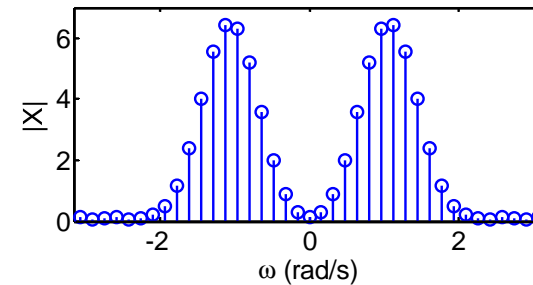
Summary

MATLAB routines

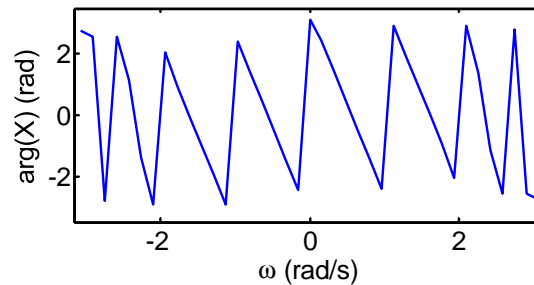
Phase of a DTFT is only defined to within an integer multiple of 2π .



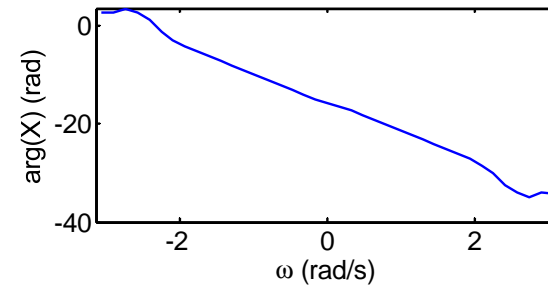
$x[n]$



$|X[k]|$



$\angle X[k]$



$\angle X[k]$ unwrapped

Phase unwrapping adds multiples of 2π onto each $\angle X[k]$ to make the phase as continuous as possible.

Summary

2: Three Different Fourier Transforms

Fourier Transforms

Convergence of DTFT

DTFT Properties

DFT Properties

Symmetries

Parseval's Theorem

Convolution

Sampling Process

Zero-Padding

Phase Unwrapping

▷ Summary

MATLAB routines

- Three types: CTFT, DTFT, DFT
 - DTFT = CTFT of continuous signal \times impulse train
 - DFT = DTFT of periodic or finite support signal
 - ▷ DFT is a scaled unitary transform
- DTFT: Convolution \rightarrow Product; Product \rightarrow Circular Convolution
- DFT: Product \leftrightarrow Circular Convolution
- DFT: Zero Padding \rightarrow Denser freq sampling but same resolution
- Phase is only defined to within a multiple of 2π .
- Whenever you integrate frequency components you get a [scale factor](#)
 - $\frac{1}{2\pi}$ for CTFT and DTFT or $\frac{1}{N}$ for DFT
 - Inverse transform, Parseval, frequency domain convolution

For further details see Mitra: 3 & 5.

MATLAB routines

2: Three Different Fourier Transforms
Fourier Transforms
Convergence of DTFT
DTFT Properties
DFT Properties
Symmetries
Parseval's Theorem
Convolution
Sampling Process
Zero-Padding
Phase Unwrapping
Summary
▷ MATLAB routines

fft, ifft	DFT with optional zero-padding
fftshift	swap the two halves of a vector
conv	convolution or polynomial multiplication (not circular)
$x[n] \circledast y[n]$	$\text{real}(\text{ifft}(\text{fft}(x) \cdot \text{fft}(y)))$
unwrap	remove 2π jumps from phase spectrum

**3: Discrete Cosine
▷ Transform**

DFT Problems

DCT

DCT of sine wave

DCT/DFT

Equivalence

DCT Properties

IDCT

Energy Conservation

Energy Compaction

Frame-based coding

Lapped Transform

MDCT

Summary

MATLAB routines

3: Discrete Cosine Transform

DFT Problems

3: Discrete Cosine Transform

▷ DFT Problems

DCT

DCT of sine wave

DCT/DFT

Equivalence

DCT Properties

IDCT

Energy Conservation

Energy Compaction

Frame-based coding

Lapped Transform

MDCT

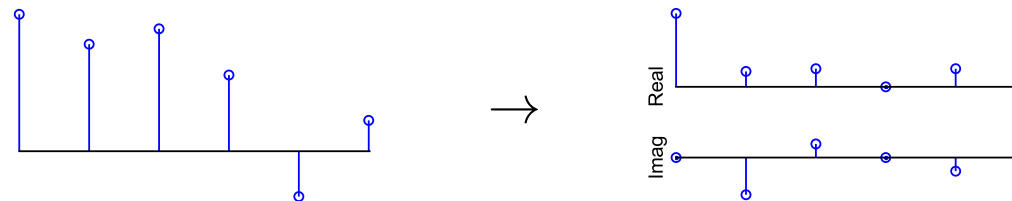
Summary

MATLAB routines

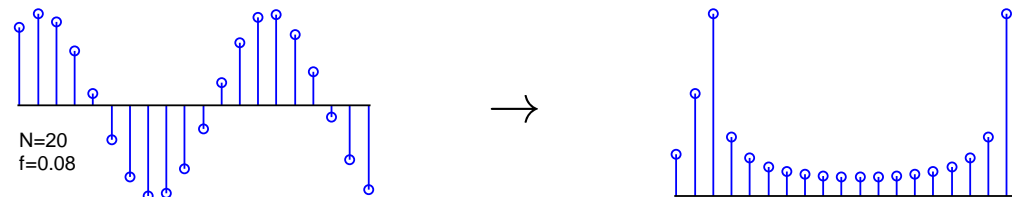
For processing 1-D or 2-D signals (especially coding), a common method is to divide the signal into “frames” and then apply an invertible transform to each frame that compresses the information into few coefficients.

The DFT has some problems when used for this purpose:

- N real $x[n] \leftrightarrow N$ complex $X[k] : 2 \text{ real}, \frac{N}{2} - 1 \text{ conj pairs}$



- $\text{DFT} \propto \text{the DTFT of a periodic signal formed by replicating } x[n]$.
 \Rightarrow Spurious frequency components from boundary discontinuity.



The Discrete Cosine Transform (DCT) overcomes these problems.

DCT

3: Discrete Cosine Transform

DFT Problems

▷ DCT

DCT of sine wave

DCT/DFT

Equivalence

DCT Properties

IDCT

Energy Conservation

Energy Compaction

Frame-based coding

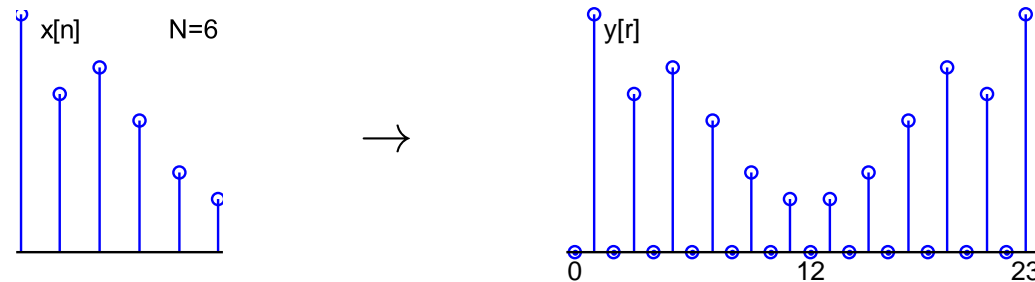
Lapped Transform

MDCT

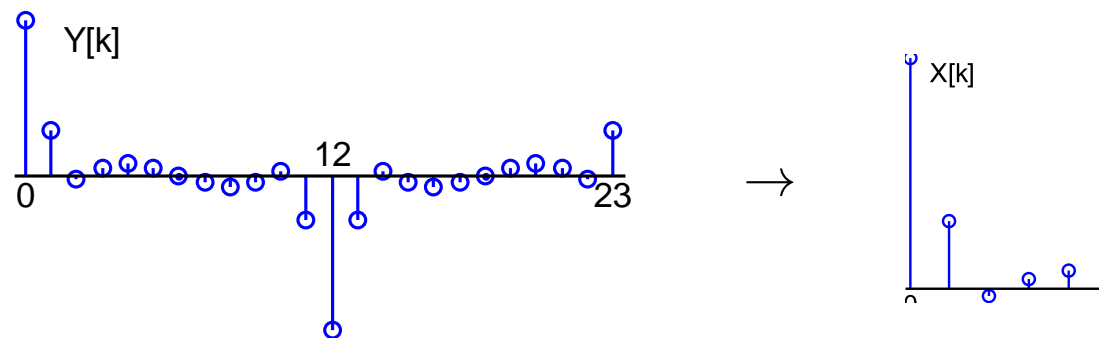
Summary

MATLAB routines

To form the Discrete Cosine Transform (DCT), replicate $x[0 : N - 1]$ but in reverse order and insert a zero between each pair of samples:



Take the DFT of length $4N$ real symmetric sequence
Result is real, symmetric and anti-periodic: only need first N values



$$\text{Forward DCT: } X_C[k] = \sum_{n=0}^{N-1} x[n] \cos \frac{2\pi(2n+1)k}{4N} \text{ for } k = 0 : N - 1$$

$$\text{Compare DFT: } X_F[k] = \sum_{n=0}^{N-1} x[n] \exp \frac{-j2\pi(4n+0)k}{4N}$$

DCT of sine wave

3: Discrete Cosine Transform

DFT Problems

DCT

▷ DCT of sine wave

DCT/DFT

Equivalence

DCT Properties

IDCT

Energy Conservation

Energy Compaction

Frame-based coding

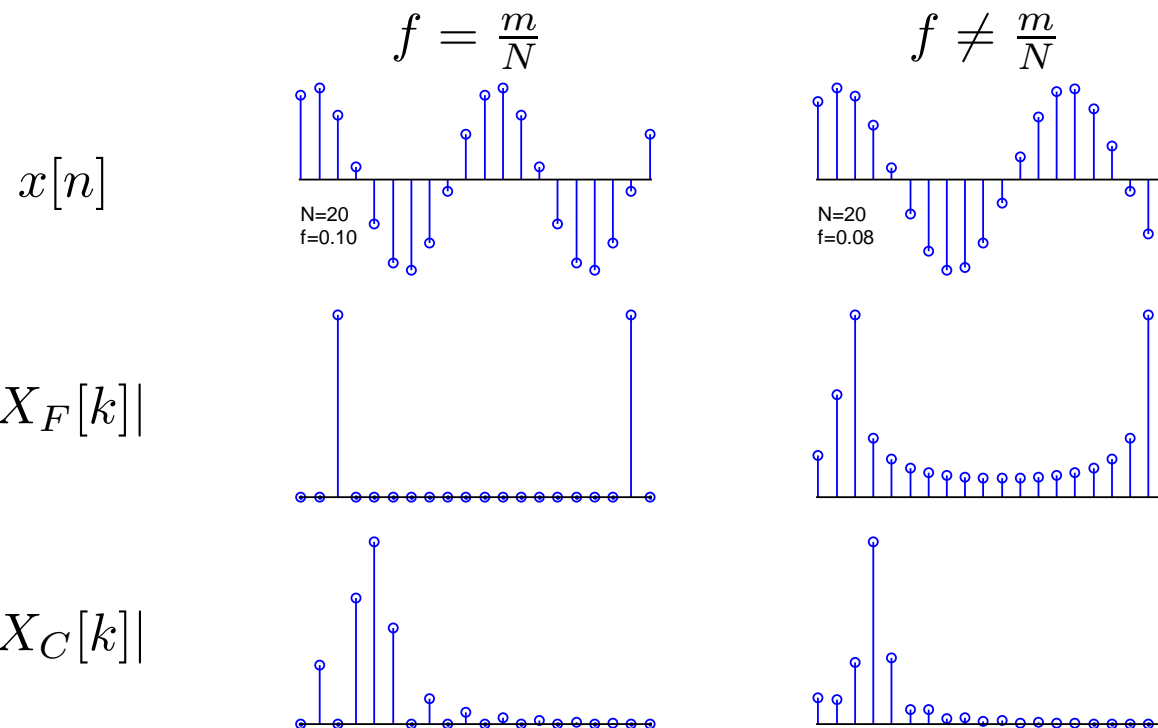
Lapped Transform

MDCT

Summary

MATLAB routines

$$\text{DCT: } X_C[k] = \sum_{n=0}^{N-1} x[n] \cos \frac{2\pi(2n+1)k}{4N}$$



DFT Real→Complex, Freq range $[0, 1]$, Poorly localized unless $f \approx \frac{m}{N}$: $\propto k^{-1}$ for $m < k \ll \frac{N}{2}$

DCT Real→Real, Freq range $[0, 0.5]$, well localized, $\propto k^{-2}$ for $m < k < N$

DCT/DFT Equivalence

3: Discrete Cosine Transform

DFT Problems

DCT

DCT of sine wave

DCT/DFT

▷ Equivalence

DCT Properties

IDCT

Energy Conservation

Energy Compaction

Frame-based coding

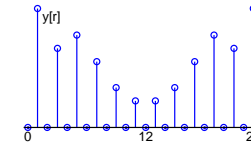
Lapped Transform

MDCT

Summary

MATLAB routines

$$X_C[k] = \sum_{n=0}^{N-1} x[n] \cos \frac{2\pi(2n+1)k}{4N}$$



$$\text{Define } y[r] = \begin{cases} 0 & r \text{ even} \\ x\left[\frac{r-1}{2}\right] & r = 1 : 2 : 2N - 1 \\ x\left[\frac{4N-1-r}{2}\right] & r = 2N + 1 : 2 : 4N - 1 \end{cases}$$

$$\begin{aligned} Y_F[k] &= \sum_{r=0}^{4N-1} y[r] W_{4N}^{kr} \quad \text{where } W_M = e^{-\frac{j2\pi}{M}} \\ &\stackrel{(i)}{=} \sum_{n=0}^{2N-1} y[2n+1] W_{4N}^{(2n+1)k} \\ &\stackrel{(ii)}{=} \sum_{n=0}^{N-1} y[2n+1] W_{4N}^{(2n+1)k} \\ &\quad + \sum_{m=0}^{N-1} y[4N-2m-1] W_{4N}^{(4N-2m-1)k} \\ &\stackrel{(iii)}{=} \sum_{n=0}^{N-1} x[n] W_{4N}^{(2n+1)k} + \sum_{m=0}^{N-1} x[m] W_{4N}^{-(2m+1)k} \\ &= 2 \sum_{n=0}^{N-1} x[n] \cos \frac{2\pi(2n+1)k}{4N} = 2X_C[k] \end{aligned}$$

(i) odd r only: $r = 2n + 1$

(ii) reverse order for $n \geq N$: $m = 2N - 1 - n$

(iii) substitute y definition & $W_{4N}^{4Nk} = e^{-j2\pi \frac{4Nk}{4N}} \equiv 1$

DCT Properties

3: Discrete Cosine Transform

DFT Problems

DCT

DCT of sine wave

DCT/DFT

Equivalence

▷ DCT Properties

IDCT

Energy Conservation

Energy Compaction

Frame-based coding

Lapped Transform

MDCT

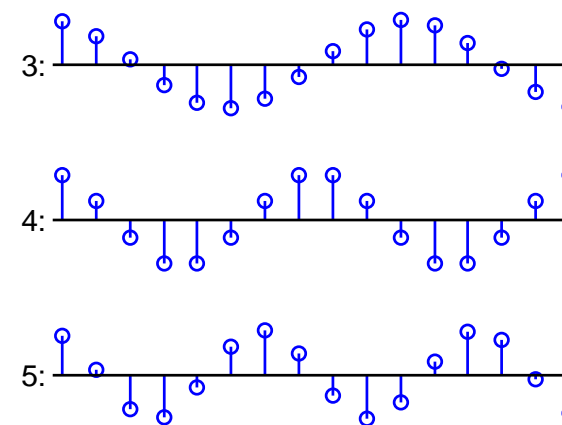
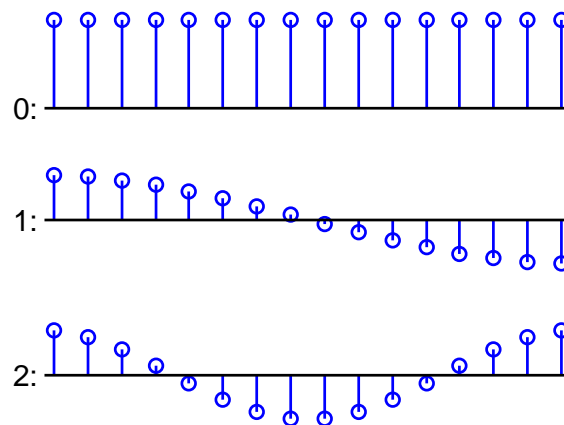
Summary

MATLAB routines

Definition: $X[k] = \sum_{n=0}^{N-1} x[n] \cos \frac{2\pi(2n+1)k}{4N}$

- **Linear:** $\alpha x[n] + \beta y[n] \rightarrow \alpha X[k] + \beta Y[k]$
- **DFT Convolution property does not hold** ☹
- **Symmetric:** $X[-k] = X[k]$ since $\cos -\alpha k = \cos +\alpha k$
- **Anti-periodic:** $X[k + 2N] = -X[k]$ because:
 - $\cos(\theta + \pi) = -\cos \theta$
 - $2\pi(2n+1)(k+2N) = 2\pi(2n+1)k + 8\pi Nn + 4\pi$
 $\Rightarrow X[N] = 0$ since $X[N] = -X[-N] = -X[-N+2N]$
- **Periodic:** $X[k + 4N] = X[k]$ since $\cos(\theta + 2\pi) = \cos \theta$

DCT basis functions:



3: Discrete Cosine Transform

DFT Problems

DCT

DCT of sine wave

DCT/DFT

Equivalence

DCT Properties

▷ IDCT

Energy Conservation

Energy Compaction

Frame-based coding

Lapped Transform

MDCT

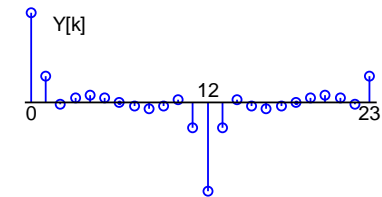
Summary

MATLAB routines

$$y[r] = \frac{1}{4N} \sum_{k=0}^{4N-1} Y[k] W_{4N}^{-rk} = \frac{1}{2N} \sum_{k=0}^{4N-1} X[k] W_{4N}^{-rk}$$

$$x[n] = y[2n+1] = \frac{1}{2N} \sum_{k=0}^{4N-1} X[k] W_{4N}^{-(2n+1)k}$$

$$[Y[k] = 2X[k]]$$



$$\begin{aligned} & \stackrel{(i)}{=} \frac{1}{2N} \sum_{k=0}^{2N-1} X[k] W_{4N}^{-(2n+1)k} \\ & \quad - \frac{1}{2N} \sum_{l=0}^{2N-1} X[l] W_{4N}^{-(2n+1)(l+2N)} \end{aligned}$$

$$\stackrel{(ii)}{=} \frac{1}{N} \sum_{k=0}^{2N-1} X[k] W_{4N}^{-(2n+1)k}$$

$$W_a^b = e^{-j \frac{2\pi b}{a}}$$

$$\begin{aligned} & \stackrel{(iii)}{=} \frac{1}{N} X[0] + \frac{1}{N} \sum_{k=1}^{N-1} X[k] W_{4N}^{-(2n+1)k} + \frac{1}{N} X[N] \\ & \quad + \frac{1}{N} \sum_{r=1}^{N-1} X[2N-r] W_{4N}^{-(2n+1)(2N-r)} \end{aligned}$$

$$\begin{aligned} & \stackrel{(iv)}{=} \frac{1}{N} X[0] + \frac{1}{N} \sum_{k=1}^{N-1} X[k] W_{4N}^{-(2n+1)k} \\ & \quad + \frac{1}{N} \sum_{r=1}^{N-1} -X[r] W_{4N}^{(2n+1)r+2N} \end{aligned}$$

$$x[n] = \frac{1}{N} X[0] + \frac{2}{N} \sum_{k=1}^{N-1} X[k] \cos \frac{2\pi(2n+1)k}{4N} = \text{Inverse DCT}$$

$$(i) \ k = l + 2N \text{ for } k \geq 2N \text{ and } X[k + 2N] = -X[k]$$

$$(ii) \ \frac{(2n+1)(l+2N)}{4N} = \frac{(2n+1)l}{4N} + n + \frac{1}{2}$$

$$(iii) \ k = 2N - r \text{ for } k > N \quad (iv) \ X[N] = 0 \text{ and } X[2N - r] = -X[r]$$

Energy Conservation

3: Discrete Cosine Transform

DFT Problems

DCT

DCT of sine wave

DCT/DFT

Equivalence

DCT Properties

IDCT

Energy

▷ Conservation

Energy Compaction

Frame-based coding

Lapped Transform

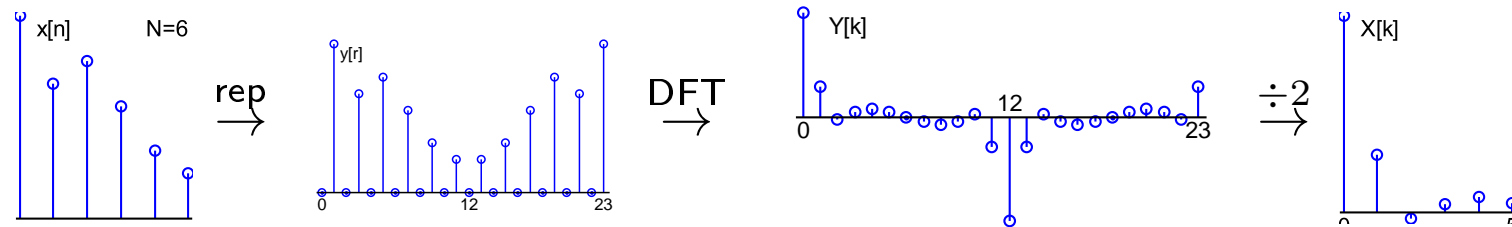
MDCT

Summary

MATLAB routines

$$\text{DCT: } X[k] = \sum_{n=0}^{N-1} x[n] \cos \frac{2\pi(2n+1)k}{4N}$$

$$\text{IDCT: } x[n] = \frac{1}{N} X[0] + \frac{2}{N} \sum_{k=1}^{N-1} X[k] \cos \frac{2\pi(2n+1)k}{4N}$$



$$\text{Energy: } E = \sum_{n=0}^{N-1} |x[n]|^2: E \rightarrow 2E \rightarrow 8NE \rightarrow \approx 0.5NE$$

$$E = \frac{1}{N} |X|^2[0] + \frac{2}{N} \sum_{n=1}^{N-1} |X|^2[n]$$

Orthogonal DCT (preserves energy)

$$\text{Define: } c[k] = \sqrt{\frac{2-\delta_k}{N}} \Rightarrow c[0] = \sqrt{\frac{1}{N}}, c[k \neq 0] = \sqrt{\frac{2}{N}}$$

$$\text{ODCT: } X[k] = c[k] \sum_{n=0}^{N-1} x[n] \cos \frac{2\pi(2n+1)k}{4N}$$

$$\text{IODCT: } x[n] = \sum_{k=0}^{N-1} c[k] X[k] \cos \frac{2\pi(2n+1)k}{4N}$$

Note: MATLAB `dct()` calculates the ODCT

Energy Compaction

3: Discrete Cosine Transform

DFT Problems

DCT

DCT of sine wave

DCT/DFT

Equivalence

DCT Properties

IDCT

Energy Conservation

Energy

▷ Compaction

Frame-based coding

Lapped Transform

MDCT

Summary

MATLAB routines

If consecutive $x[n]$ are positively correlated, DCT concentrates energy in a few $X[k]$ and decorrelates them.

Example: Markov Process: $x[n] = \rho x[n-1] + \sqrt{1-\rho^2}u[n]$

where $u[n]$ is i.i.d. unit Gaussian.

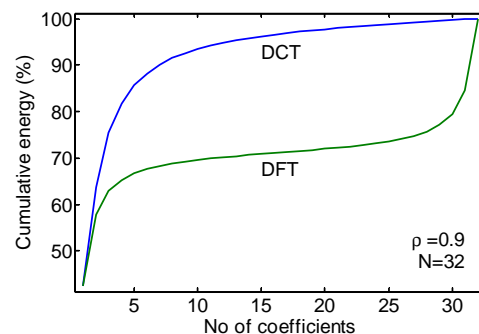
Then $\langle x^2[n] \rangle = 1$ and $\langle x[n]x[n-1] \rangle = \rho$.

Covariance of vector \mathbf{x} is $\mathbf{S}_{i,j} = \langle \mathbf{x}\mathbf{x}^H \rangle_{i,j} = \rho^{|i-j|}$.

Suppose ODCT of \mathbf{x} is $\mathbf{C}\mathbf{x}$ and DFT is $\mathbf{F}\mathbf{x}$.

Covariance of $\mathbf{C}\mathbf{x}$ is $\langle \mathbf{C}\mathbf{x}\mathbf{x}^H\mathbf{C}^H \rangle = \mathbf{C}\mathbf{S}\mathbf{C}^H$ (similarly $\mathbf{F}\mathbf{S}\mathbf{F}^H$)

Diagonal elements give mean coefficient energy.



- Used in MPEG and JPEG (superseded by JPEG2000 using wavelets)
- Used in speech recognition to decorrelate: DCT of log spectrum

Energy compaction good for coding (low-valued coefficients can be set to 0)

Decorrelation good for coding and for probability modelling

Frame-based coding

3: Discrete Cosine Transform

DFT Problems

DCT

DCT of sine wave

DCT/DFT

Equivalence

DCT Properties

IDCT

Energy Conservation

Energy Compaction

Frame-based

▷ coding

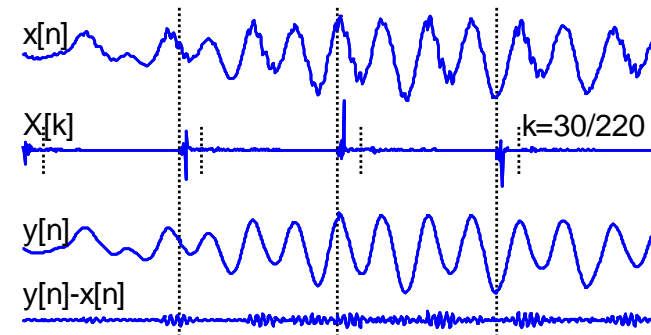
Lapped Transform

MDCT

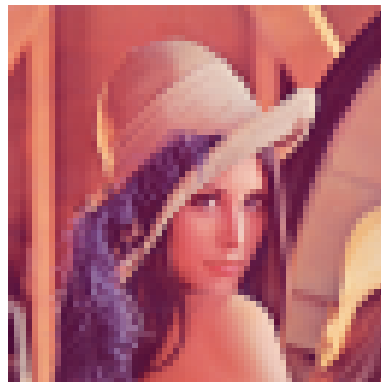
Summary

MATLAB routines

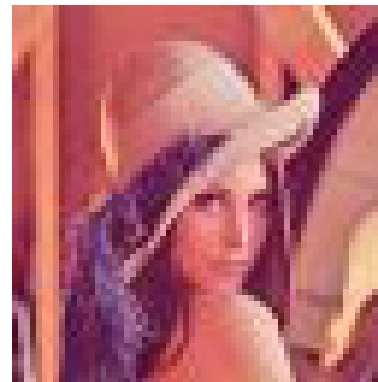
- Divide continuous signal into frames
- Apply DCT to each frame
- Encode DCT
 - e.g. keep only 30 $X[k]$
- Apply IDCT $\rightarrow y[n]$



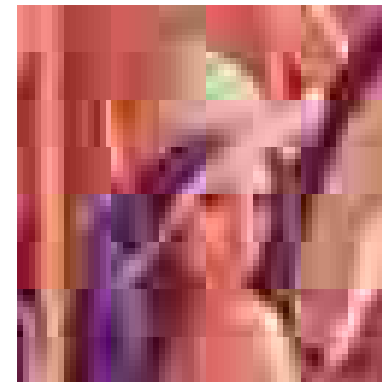
Problem: Coding may create discontinuities at frame boundaries
e.g. JPEG, MPEG use 8×8 pixel blocks



8.3 kB (PNG)



1.6 kB (JPEG)



0.5 kB (JPEG)

Lapped Transform

3: Discrete Cosine Transform

DFT Problems

DCT

DCT of sine wave

DCT/DFT

Equivalence

DCT Properties

IDCT

Energy Conservation

Energy Compaction

Frame-based coding

▷ Lapped Transform

MDCT

Summary

MATLAB routines

Modified Discrete Cosine Transform (MDCT): overlapping frames $2N$ long

$$\begin{aligned} x[0 : 2N - 1] \\ \xrightarrow{\text{MDCT}} X[0 : N - 1] \\ \xrightarrow{\text{IMDCT}} y_A[0 : 2N - 1] \end{aligned}$$

$$\begin{aligned} x[N : 3N - 1] \\ \xrightarrow{\text{MDCT}} X[N : 2N - 1] \\ \xrightarrow{\text{IMDCT}} y_B[N : 3N - 1] \end{aligned}$$

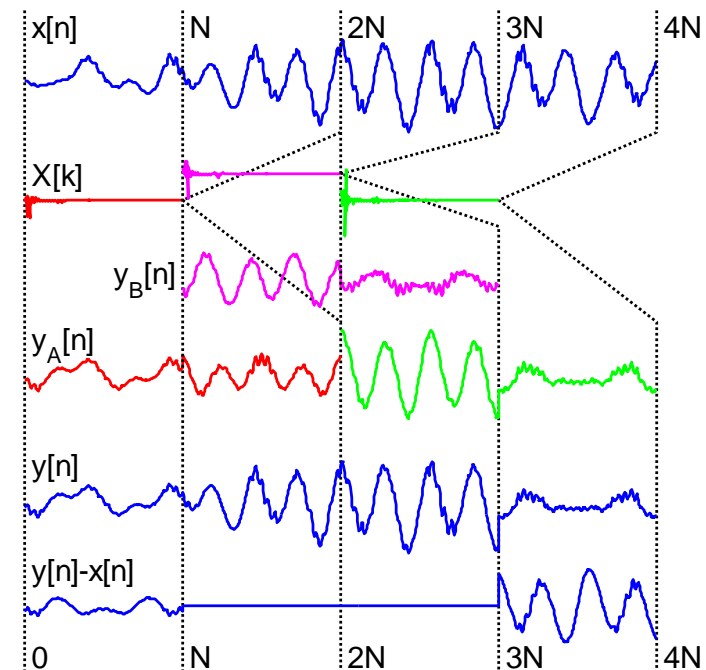
$$\begin{aligned} x[2N : 4N - 1] \\ \xrightarrow{\text{MDCT}} X[2N : 3N - 1] \\ \xrightarrow{\text{IMDCT}} y_A[2N : 3N - 1] \end{aligned}$$

$$y[*] = y_A[*] + y_B[*]$$

MDCT: $2N \rightarrow N$ coefficients, **IMDCT:** $N \rightarrow 2N$ samples

Even frames $\rightarrow y_A$, Odd frames $\rightarrow y_B$ then $y = y_A + y_B$

Errors cancel exactly: **Time-domain alias cancellation (TDAC)**



$$\text{MDCT: } X_M[k] = \sum_{n=0}^{2N-1} x[n] \cos \frac{2\pi(2n+1+N)(2k+1)}{8N}$$

$$\text{IMDCT: } y_{A,B}[n] = \frac{1}{N} \sum_{k=0}^{N-1} X_M[k] \cos \frac{2\pi(2n+1+N)(2k+1)}{8N}$$

MDCT can be made from a DCT of length $2N$

Split $x[n]$ into four $\frac{N}{2}$ -vectors: $\mathbf{x} = \begin{bmatrix} \mathbf{a} & \mathbf{b} & \mathbf{c} & \mathbf{d} \end{bmatrix}$

Now form the N -vector $\mathbf{u} = \begin{bmatrix} -\mathbf{d} - \overleftarrow{\mathbf{c}} & \mathbf{a} - \overleftarrow{\mathbf{b}} \end{bmatrix}$

where $\overleftarrow{\mathbf{c}}$ is the vector \mathbf{c} but in reverse order

Now form $\mathbf{v} = \begin{bmatrix} \mathbf{u} & -\overleftarrow{\mathbf{u}} \end{bmatrix}$ and take $2N$ DCT

$V_C[2k] = 0$ because of symmetry, so set $X_M[k] = V_C[2k+1]$

IMDCT

Undo above to get $X_M[k] \rightarrow \mathbf{u} = \begin{bmatrix} -\mathbf{d} - \overleftarrow{\mathbf{c}} & \mathbf{a} - \overleftarrow{\mathbf{b}} \end{bmatrix}$

Create $\mathbf{y}_A[n] = \begin{bmatrix} \mathbf{a} - \overleftarrow{\mathbf{b}} & \mathbf{b} - \overleftarrow{\mathbf{a}} & \mathbf{c} + \overleftarrow{\mathbf{d}} & \mathbf{d} + \overleftarrow{\mathbf{c}} \end{bmatrix}$

Next frame: $\mathbf{y}_B[n] = \begin{bmatrix} \mathbf{c} - \overleftarrow{\mathbf{d}} & \mathbf{d} - \overleftarrow{\mathbf{c}} & \mathbf{e} + \overleftarrow{\mathbf{f}} & \mathbf{f} + \overleftarrow{\mathbf{e}} \end{bmatrix}$

Adding together, the $\overleftarrow{\mathbf{d}}$ and $\overleftarrow{\mathbf{c}}$ terms cancel but \mathbf{c} and \mathbf{d} add.

Used in audio coding: MP3, WMA, AC-3, AAC, Vorbis, ATRAC

Summary

3: Discrete Cosine Transform

DFT Problems

DCT

DCT of sine wave

DCT/DFT

Equivalence

DCT Properties

IDCT

Energy Conservation

Energy Compaction

Frame-based coding

Lapped Transform

MDCT

▷ Summary

MATLAB routines

DCT: Discrete Cosine Transform

- ☐ Equivalent to a DFT of time-shifted $\left[\mathbf{x} \quad \overleftarrow{\mathbf{x}} \right]$
- ☐ Often scaled to make an orthogonal transform
- ☐ Better than DFT for energy compaction and decorrelation 😊
- ☐ Nice convolution property of DFT is lost ☹️

MDCT: Modified Discrete Cosine Transform

- ☐ Lapped transform: $2N \rightarrow N \rightarrow 2N$
- ☐ Aliasing errors cancel out when overlapping output frames are added
- ☐ Like DCT for energy compaction and decorrelation 😊
- ☐ Overlapping frames can avoid edge effects 😊

For further details see Mitra: 5.

MATLAB routines

3: Discrete Cosine Transform

DFT Problems

DCT

DCT of sine wave

DCT/DFT

Equivalence

DCT Properties

IDCT

Energy Conservation

Energy Compaction

Frame-based coding

Lapped Transform

MDCT

Summary

▷ MATLAB routines

dct, idct

DFT with optional zero-padding

4: Linear Time
▷ Invariant Systems

LTI Systems

Convolution

Properties

BIBO Stability

Frequency Response

Causality

Passive and Lossless

Parallel and Series

Convolution

Complexity

Circular Convolution

Frequency-domain
convolution

Overlap Add

Overlap Save

Summary

MATLAB routines

4: Linear Time Invariant Systems



Linear Time-invariant (LTI) systems have two properties:

Linear: $\mathcal{H}(\alpha u[n] + \beta v[n]) = \alpha \mathcal{H}(u[n]) + \beta \mathcal{H}(v[n])$

Time Invariant: $y[n] = \mathcal{H}(x[n]) \Rightarrow y[n-r] = \mathcal{H}(x[n-r]) \forall r$

The behaviour of an LTI system is **completely defined by its impulse response**: $h[n] = \mathcal{H}(\delta[n])$

Proof:

We can always write $x[n] = \sum_{r=-\infty}^{\infty} x[r] \delta[n-r]$

$$\begin{aligned} \text{Hence } \mathcal{H}(x[n]) &= \mathcal{H}\left(\sum_{r=-\infty}^{\infty} x[r] \delta[n-r]\right) \\ &= \sum_{r=-\infty}^{\infty} x[r] \mathcal{H}(\delta[n-r]) \\ &= \sum_{r=-\infty}^{\infty} x[r] h[n-r] \\ &= x[n] * h[n] \end{aligned}$$

Convolution Properties

4: Linear Time Invariant Systems

LTI Systems

Convolution ▷ Properties

BIBO Stability

Frequency Response

Causality

Passive and Lossless

Parallel and Series

Convolution Complexity

Circular Convolution

Frequency-domain convolution

Overlap Add

Overlap Save

Summary

MATLAB routines

Convolution: $x[n] * v[n] = \sum_{r=-\infty}^{\infty} x[r]v[n-r]$

Convolution obeys **normal arithmetic rules for multiplication:**

Commutative: $x[n] * v[n] = v[n] * x[n]$

Proof: $\sum_r x[r]v[n-r] \stackrel{(i)}{=} \sum_p x[n-p]v[p]$
(i) substitute $p = n - r$

Associative: $x[n] * (v[n] * w[n]) = (x[n] * v[n]) * w[n]$
 $\Rightarrow x[n] * v[n] * w[n]$ is **unambiguous**

Proof: $\sum_{r,s} x[n-r]v[r-s]w[s] \stackrel{(i)}{=} \sum_{p,q} x[p]v[q-p]w[n-q]$
(i) substitute $p = n - r, q = n - s$

Distributive over +:

$x[n] * (\alpha v[n] + \beta w[n]) = (x[n] * \alpha v[n]) + (x[n] * \beta w[n])$

Proof: $\sum_r x[n-r] (\alpha v[r] + \beta w[r]) =$
 $\alpha \sum_r x[n-r]v[r] + \beta \sum_r x[n-r]w[r]$

Identity: $x[n] * \delta[n] = x[n]$

Proof: $\sum_r \delta[r]x[n-r] \stackrel{(i)}{=} x[n]$ (i) all terms zero except $r = 0$.

BIBO Stability

4: Linear Time Invariant Systems

LTI Systems

Convolution Properties

▷ BIBO Stability

Frequency Response

Causality

Passive and Lossless

Parallel and Series

Convolution

Complexity

Circular Convolution

Frequency-domain convolution

Overlap Add

Overlap Save

Summary

MATLAB routines

BIBO Stability: Bounded Input, $x[n] \Rightarrow$ Bounded Output, $y[n]$

The following are equivalent:

- (1) An LTI system is **BIBO stable**
- (2) $h[n]$ is **absolutely summable**, i.e. $\sum_{n=-\infty}^{\infty} |h[n]| < \infty$
- (3) $H(z)$ **region of convergence includes $|z| = 1$.**

Proof (1) \Rightarrow (2):

$$\text{Define } x[n] = \begin{cases} 1 & h[-n] \geq 0 \\ -1 & h[-n] < 0 \end{cases}$$

$$\text{then } y[0] = \sum x[0-n]h[n] = \sum |h[n]|.$$

$$\text{But } |x[n]| \leq 1 \forall n \text{ so BIBO } \Rightarrow y[0] = \sum |h[n]| < \infty.$$

Proof (2) \Rightarrow (1):

Suppose $\sum |h[n]| = S < \infty$ and $|x[n]| \leq B$ is bounded.

$$\begin{aligned} \text{Then } |y[n]| &= \left| \sum_{r=-\infty}^{\infty} x[n-r]h[r] \right| \\ &\leq \sum_{r=-\infty}^{\infty} |x[n-r]| |h[r]| \\ &\leq B \sum_{r=-\infty}^{\infty} |h[r]| \leq BS < \infty \end{aligned}$$

Frequency Response

4: Linear Time Invariant Systems

LTI Systems

Convolution

Properties

BIBO Stability

Frequency

▷ Response

Causality

Passive and Lossless

Parallel and Series

Convolution

Complexity

Circular Convolution

Frequency-domain convolution

Overlap Add

Overlap Save

Summary

MATLAB routines

For a BIBO stable system $Y(e^{j\omega}) = X(e^{j\omega})H(e^{j\omega})$
where $H(e^{j\omega})$ is the DTFT of $h[n]$ i.e. $H(z)$ evaluated at $z = e^{j\omega}$.

Example: $h[n] = [1 \ 1 \ 1]$

$$\begin{aligned} H(e^{j\omega}) &= 1 + e^{-j\omega} + e^{-j2\omega} \\ &= e^{-j\omega} (1 + 2\cos\omega) \end{aligned}$$

$$|H(e^{j\omega})| = |1 + 2\cos\omega|$$

$$\angle H(e^{j\omega}) = -\omega + \pi \frac{1 - \text{sgn}(1 + 2\cos\omega)}{2}$$

Sign change in $(1 + 2\cos\omega)$ at $\omega = 2.1$ gives

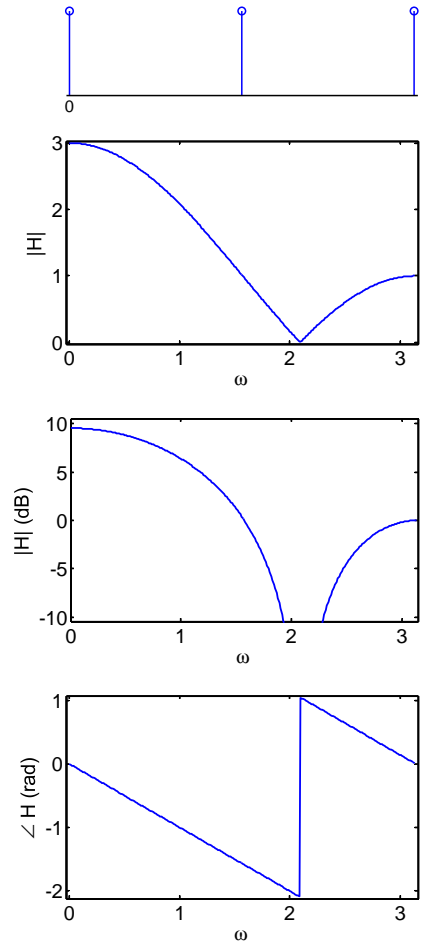
(a) **gradient discontinuity** in $|H(e^{j\omega})|$

(b) an **abrupt phase change** of $\pm\pi$.

Group delay is $-\frac{d}{d\omega}\angle H(e^{j\omega})$: gives delay of the modulation envelope at each ω .

Normally varies with ω but for a symmetric filter it is constant: in this case +1 samples.

Discontinuities of $\pm k\pi$ do not affect group delay.



Causal System: cannot see into the future

i.e. output at time n depends only on inputs up to time n .

Formal definition:

If $v[n] = x[n]$ for $n \leq n_0$ then $\mathcal{H}(v[n]) = \mathcal{H}(x[n])$ for $n \leq n_0$.

The following are equivalent:

- (1) An LTI system is causal
- (2) $h[n]$ is causal $\Leftrightarrow h[n] = 0$ for $n < 0$
- (3) $H(z)$ converges for $z = \infty$

Any right-sided sequence can be made causal by adding a delay.

All the systems we will deal with are causal.

Passive and Lossless

- 4: Linear Time Invariant Systems
- LTI Systems
- Convolution Properties
- BIBO Stability
- Frequency Response
- Causality
 - Passive and Lossless
- Parallel and Series
- Convolution Complexity
- Circular Convolution
- Frequency-domain convolution
- Overlap Add
- Overlap Save
- Summary
- MATLAB routines



A **passive system** can never gain energy:

$$\sum_{n=-\infty}^{\infty} |y[n]|^2 \leq \sum_{n=-\infty}^{\infty} |x[n]|^2 \text{ for any finite energy input } x[n].$$

A **passive LTI** system must have $|H(e^{j\omega})| \leq 1 \forall \omega$

Somewhat analogous to a circuit consisting of R, L and C only.

A **lossless system** always preserves energy exactly:

$$\sum_{n=-\infty}^{\infty} |y[n]|^2 = \sum_{n=-\infty}^{\infty} |x[n]|^2 \text{ for any finite energy input } x[n].$$

A **lossless LTI** system must have $|H(e^{j\omega})| = 1 \forall \omega$
called an **allpass** system.

Somewhat analogous to a circuit consisting of L and C only.

Passive and lossless building blocks can be used to design systems which are insensitive to coefficient changes

Parallel and Series

4: Linear Time Invariant Systems

LTI Systems

Convolution

Properties

BIBO Stability

Frequency Response

Causality

Passive and Lossless

▷ Parallel and Series

Convolution

Complexity

Circular Convolution

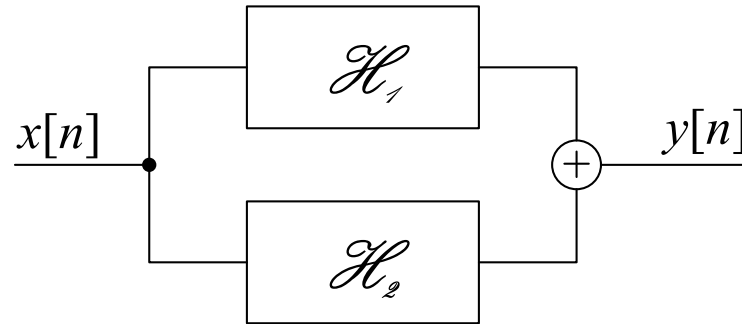
Frequency-domain convolution

Overlap Add

Overlap Save

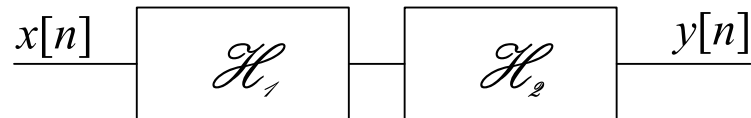
Summary

MATLAB routines



Two LTI systems in **parallel**:

$$h[n] = h_1[n] + h_2[n] \quad H(z) = H_1(z) + H_2(z)$$



Two LTI systems in **series** (a.k.a. **cascade**):

$$h[n] = h_1[n] * h_2[n] \quad H(z) = H_1(z)H_2(z)$$

Proof: $y[n] = h_2[n] * (h_1[n] * x[n]) = (h_2[n] * h_1[n]) * x[n]$
associativity of convolution

Convolution Complexity

4: Linear Time Invariant Systems

LTI Systems

Convolution Properties

BIBO Stability

Frequency Response

Causality

Passive and Lossless

Parallel and Series

Convolution

▷ Complexity

Circular Convolution

Frequency-domain convolution

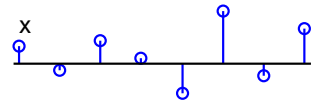
Overlap Add

Overlap Save

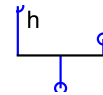
Summary

MATLAB routines

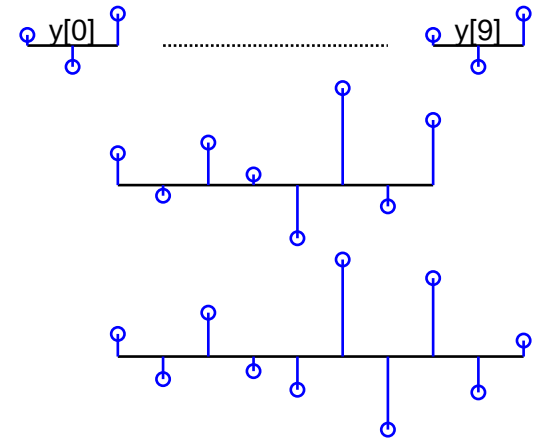
$y[n] = x[n] * h[n]$: convolve $x[0 : N - 1]$ with $h[0 : M - 1]$



*



→



Convolution sum:

$$y[n] = \sum_{r=0}^{M-1} h[r]x[n-r]$$

$y[n]$ is only non-zero in the range

$$0 \leq n \leq M + N - 2$$

Thus $y[n]$ has only

$M + N - 1$ non-zero values

Algebraically:

$$\begin{aligned} x[n-r] \neq 0 &\Rightarrow 0 \leq n-r \leq N-1 \\ &\Rightarrow n+1-N \leq r \leq n \end{aligned}$$

$$\text{Hence: } y[n] = \sum_{r=\max(0, n+1-N)}^{\min(M-1, n)} h[r]x[n-r]$$

We must multiply each $h[n]$ by each $x[n]$ and add them to a total

\Rightarrow total arithmetic complexity (\times or $+$ operations) $\approx 2MN$

$$\begin{aligned} N &= 8, M = 3 \\ M + N - 1 &= 10 \end{aligned}$$

Circular Convolution

4: Linear Time Invariant Systems

LTI Systems

Convolution Properties

BIBO Stability

Frequency Response

Causality

Passive and Lossless

Parallel and Series

Convolution Complexity

Circular Convolution

Frequency-domain convolution

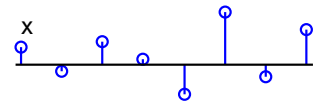
Overlap Add

Overlap Save

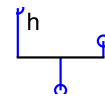
Summary

MATLAB routines

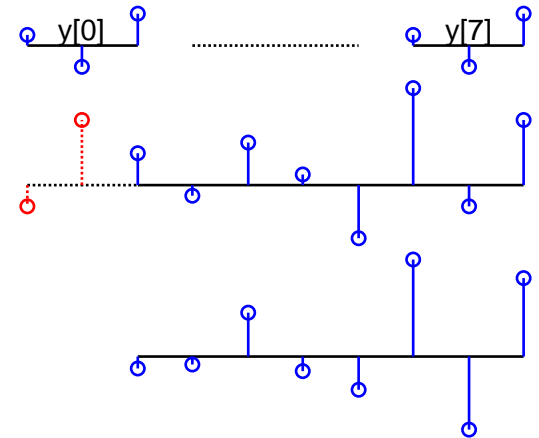
$y_{\circledast}[n] = x[n] \circledast_N h[n]$: circ convolve $x[0 : N - 1]$ with $h[0 : M - 1]$



\circledast_N



\rightarrow



Convolution sum:

$$y_{\circledast_N}[n] = \sum_{r=0}^{M-1} h[r]x[(n-r)_{\text{mod } N}]$$

$y_{\circledast_N}[n]$ has period N

$\Rightarrow y_{\circledast_N}[n]$ has N distinct values

$$N = 8, M = 3$$

Only the first $M - 1$ values are affected by the circular repetition:

$$y_{\circledast_N}[n] = y[n] \text{ for } M - 1 \leq n \leq N - 1$$

If we append $M - 1$ zeros (or more) onto $x[n]$, then the circular repetition has no effect at all and:

$$y_{\circledast_{N+M-1}}[n] = y[n] \text{ for } 0 \leq n \leq N + M - 2$$

Frequency-domain convolution

Idea: Use DFT to perform circular convolution - less computation

- (1) Choose $L \geq M + N - 1$ (normally round up to a power of 2)
- (2) Zero pad $x[n]$ and $h[n]$ to give sequences of length L : $\tilde{x}[n]$ and $\tilde{h}[n]$
- (3) Use DFT: $\tilde{y}[n] = \mathcal{F}^{-1}(\tilde{X}[k]\tilde{H}[k]) = \tilde{x}[n] \circledast_L \tilde{h}[n]$
- (4) $y[n] = \tilde{y}[n]$ for $0 \leq n \leq M + N - 2$.

Arithmetic Complexity:

DFT or IDFT take $4L \log_2 L$ operations if L is a power of 2
(or $16L \log_2 L$ if not).

Total operations: $\approx 12L \log_2 L \approx 12(M + N) \log_2 (M + N)$

Beneficial if both M and N are $> \sim 100$.

Example: $M = 10^3$, $N = 10^4$:

Direct: $2MN = 2 \times 10^7$

with DFT: $12(M + N) \log_2 (M + N) = 1.8 \times 10^6 \odot$

But: (a) DFT may be very long if N is large

(b) No output samples until all $x[n]$ has been input.

Overlap Add

4: Linear Time Invariant Systems

LTI Systems

Convolution

Properties

BIBO Stability

Frequency Response

Causality

Passive and Lossless

Parallel and Series

Convolution

Complexity

Circular Convolution

Frequency-domain convolution

▷ Overlap Add

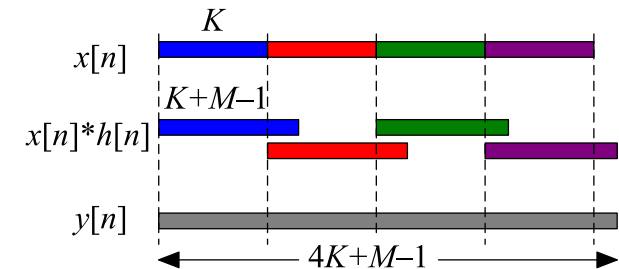
Overlap Save

Summary

MATLAB routines

If N is very large:

- (1) chop $x[n]$ into $\frac{N}{K}$ chunks of length K
- (2) convolve each chunk with $h[n]$
- (3) add up the results



Each output chunk is of length $K + M - 1$ and overlaps the next chunk

Operations: $\approx \frac{N}{K} \times 8(M + K) \log_2(M + K)$

Computational saving if $\approx 100 < M \ll K \ll N$

Example: $M = 500$, $K = 10^4$, $N = 10^7$

Direct: $2MN = 10^{10}$

single DFT: $12(M + N) \log_2(M + N) = 2.8 \times 10^9$

overlap-add: $\frac{N}{K} \times 8(M + K) \log_2(M + K) = 1.1 \times 10^9 \odot$

Other advantages:

- (a) Shorter DFT
- (b) Can cope with $N = \infty$
- (c) Can calculate $y[0]$ as soon as $x[K - 1]$ has been read:
algorithmic delay = $K - 1$ samples

Overlap Save

4: Linear Time Invariant Systems

LTI Systems

Convolution

Properties

BIBO Stability

Frequency Response

Causality

Passive and Lossless

Parallel and Series

Convolution

Complexity

Circular Convolution

Frequency-domain convolution

Overlap Add

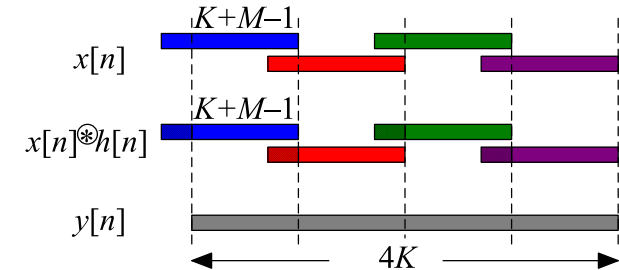
▷ Overlap Save

Summary

MATLAB routines

Alternative method:

- (1) chop $x[n]$ into $\frac{N}{K}$ overlapping chunks of length $K + M - 1$
- (2) \otimes_{K+M-1} each chunk with $h[n]$
- (3) discard first $M - 1$ from each chunk
- (4) concatenate to make $y[n]$



The first $M - 1$ points of each output chunk are invalid

Operations: slightly less than overlap-add because no addition needed to create $y[n]$

Advantages: same as overlap add

Strangely, rather less popular than overlap-add

Summary

4: Linear Time Invariant Systems

LTI Systems

Convolution

Properties

BIBO Stability

Frequency Response

Causality

Passive and Lossless

Parallel and Series

Convolution

Complexity

Circular Convolution

Frequency-domain convolution

Overlap Add

Overlap Save

▷ Summary

MATLAB routines

- ☐ LTI systems: impulse response, frequency response, group delay
- ☐ BIBO stable, Causal, Passive, Lossless systems
- ☐ Convolution and circular convolution properties
- ☐ Efficient methods for convolution
 - single DFT
 - overlap-add and overlap-save

For further details see Mitra: 4 & 5.

MATLAB routines

4: Linear Time Invariant Systems

LTI Systems

Convolution

Properties

BIBO Stability

Frequency Response

Causality

Passive and Lossless

Parallel and Series

Convolution

Complexity

Circular Convolution

Frequency-domain convolution

Overlap Add

Overlap Save

Summary

▷ MATLAB routines

fftfilt	Convolution using overlap add
$x[n] \circledast y[n]$	$\text{real}(\text{ifft}(\text{fft}(x) \cdot \text{fft}(y)))$

▷ 5: Filters

Finite Dimensional

LTI Systems

FIR Filters

IIR Frequency

Response

Negating z

Cubing z

Scaling z

IIR Impulse response

examples

IIR Impulse response

Impulse response

proof

Other BIBO

responses

Stability Triangle

Low-pass filter

Allpass filters

Group Delay

Minimum Phase

Summary

MATLAB routines

5: Filters

Finite Dimensional LTI Systems

- 5: Filters
 - Finite Dimensional
 - ▷ LTI Systems
 - FIR Filters
 - IIR Frequency Response
 - Negating z
 - Cubing z
 - Scaling z
 - IIR Impulse response examples
 - IIR Impulse response proof
 - Other BIBO responses
 - Stability Triangle
 - Low-pass filter
 - Allpass filters
 - Group Delay
 - Minimum Phase
 - Summary
 - MATLAB routines

Most useful LTI systems can be described by a difference equation:

$$y[n] = \sum_{r=0}^M b[r]x[n-r] - \sum_{r=1}^N a[r]y[n-r]$$

$$\Leftrightarrow \sum_{r=0}^N a[r]y[n-r] = \sum_{r=0}^M b[r]x[n-r]$$

$$\Leftrightarrow a[n] * y[n] = b[n] * x[n]$$

$$\Leftrightarrow Y(z) = \frac{B(z)}{A(z)}X(z)$$

$$\Leftrightarrow Y(e^{j\omega}) = \frac{B(e^{j\omega})}{A(e^{j\omega})}X(e^{j\omega})$$

- (1) Always **causal**.
- (2) **Order** of system is $\max(M, N)$, the highest r with $a[r] \neq 0$ or $b[r] \neq 0$.
- (3) We assume that $a[0] = 1$; if not, divide $A(z)$ and $B(z)$ by $a[0]$.
- (4) Filter is BIBO stable iff roots of $A(z)$ all lie within the unit circle.

Note **negative sign** in first equation.

Authors in some SP fields reverse the sign of the $a[n]$: BAD IDEA.

FIR Filters

- 5: Filters
- Finite Dimensional LTI Systems
 - ▷ FIR Filters
 - IIR Frequency Response
 - Negating z
 - Cubing z
 - Scaling z
 - IIR Impulse response examples
 - IIR Impulse response proof
 - Other BIBO responses
 - Stability Triangle
 - Low-pass filter
 - Allpass filters
 - Group Delay
 - Minimum Phase
 - Summary
 - MATLAB routines

$A(z) = 1$: **Finite Impulse Response** (FIR) filter: $Y(z) = B(z)X(z)$.
Impulse response is $b[n]$ and is of length $M + 1$.

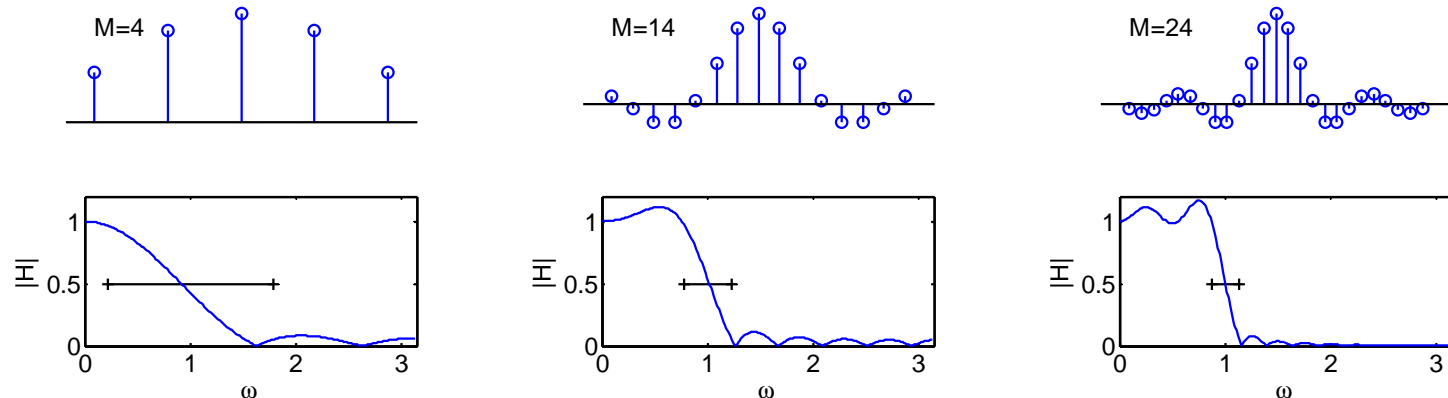
Frequency response is $B(e^{j\omega})$ and is the DTFT of $b[n]$.

Comprises M complex sinusoids:

$$b[0] + b[1]e^{-j\omega} + \dots + b[M-1]e^{-j(M-1)\omega}$$

Small $M \Rightarrow$ response contains only low “**quefrecencies**”

Symmetrical $b[n] \Rightarrow |H(e^{j\omega})|$ consists of $\frac{M}{2}$ cosine waves + const



Rule of thumb: Fastest possible transition $\Delta\omega \geq \frac{2\pi}{M}$ (marked line)

IIR Frequency Response

- 5: Filters
- Finite Dimensional LTI Systems
- FIR Filters
- IIR Frequency Response
- Negating z
- Cubing z
- Scaling z
- IIR Impulse response examples
- IIR Impulse response proof
- Other BIBO responses
- Stability Triangle
- Low-pass filter
- Allpass filters
- Group Delay
- Minimum Phase
- Summary
- MATLAB routines

$$\text{Factorize } H(z) = \frac{B(z)}{A(z)} = \frac{b[0] \prod_{i=1}^M (1 - q_i z^{-1})}{\prod_{i=1}^N (1 - p_i z^{-1})}$$

Roots of $A(z)$ and $B(z)$ are the “poles” $\{p_i\}$ and “zeros” $\{q_i\}$ of $H(z)$
 Also an additional $N - M$ zeros at the origin (affect phase only)

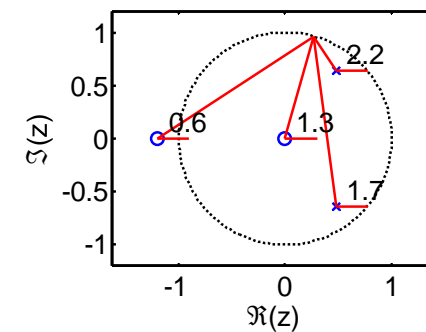
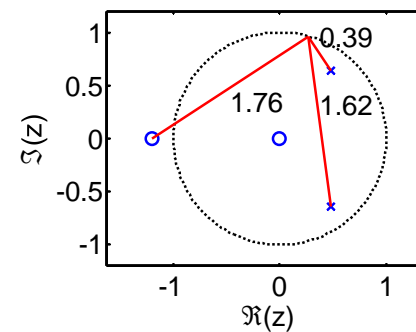
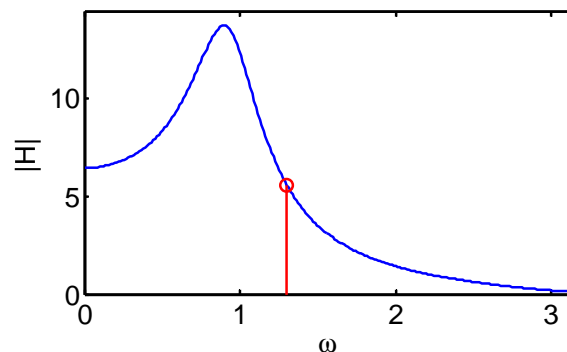
$$|H(e^{j\omega})| = \frac{|b[0]| |z^{-M}| \prod_{i=1}^M |z - q_i|}{|z^{-N}| \prod_{i=1}^N |z - p_i|} \text{ for } z = e^{j\omega}$$

Example:

$$H(z) = \frac{2 + 2.4z^{-1}}{1 - 0.96z^{-1} + 0.64z^{-2}} = \frac{2(1 + 1.2z^{-1})}{(1 - (0.48 - 0.64j)z^{-1})(1 - (0.48 + 0.64j)z^{-1})}$$

At $\omega = 1.3$: $|H(e^{j\omega})| = \frac{2 \times 1.76}{1.62 \times 0.39} = 5.6$

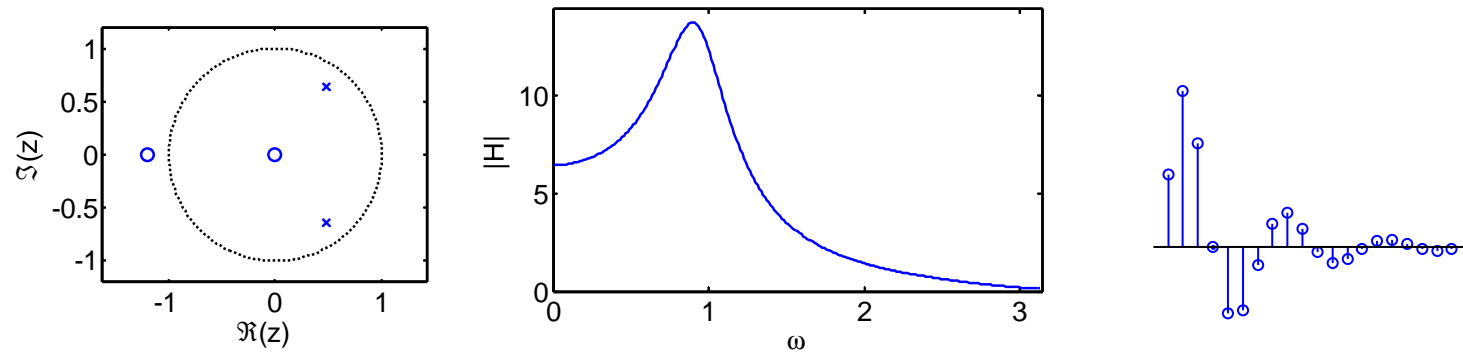
$$\angle H(e^{j\omega}) = (0.6 + 1.3) - (1.7 + 2.2) = -1.97$$



Negating z

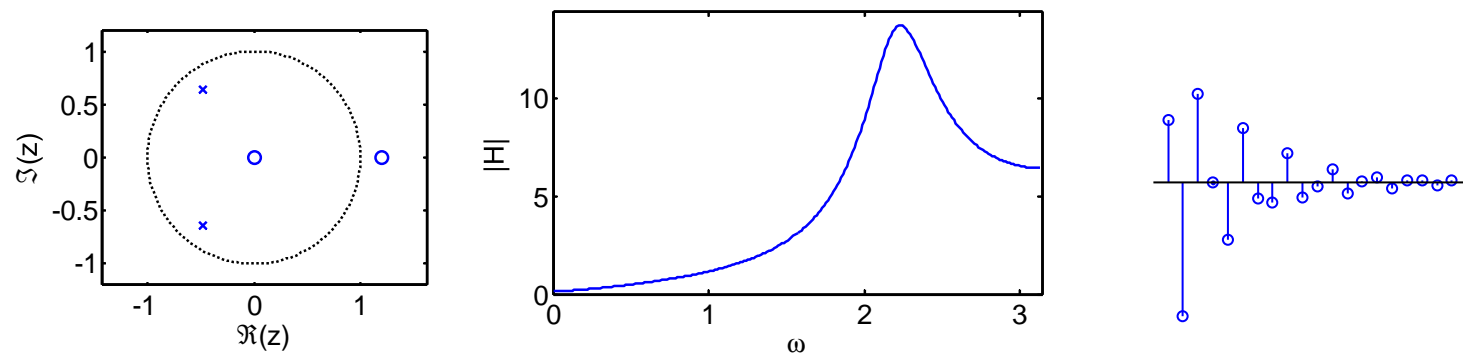
Given a filter $H(z)$ we can form a new one $H_R(z) = H(-z)$
 Negate all odd powers of z , i.e. negate alternate $a[n]$ and $b[n]$

Example: $H(z) = \frac{2+2.4z^{-1}}{1-0.96z^{-1}+0.64z^{-2}}$



Negate z: $H_R(z) = \frac{2-2.4z^{-1}}{1+0.96z^{-1}+0.64z^{-2}}$

Negate odd coefficients



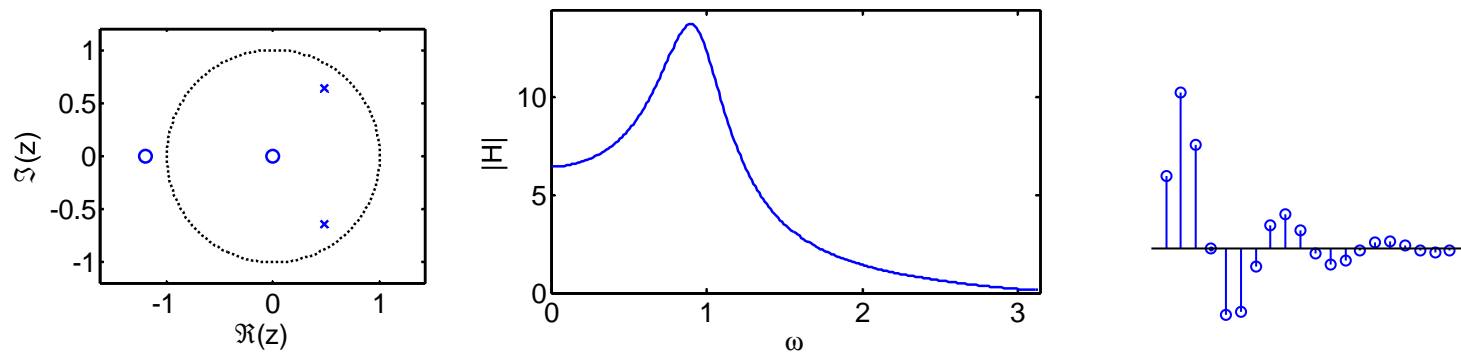
Pole and zero positions are **negated**, frequency response **flipped**.

Cubing z

- 5: Filters
- Finite Dimensional LTI Systems
- FIR Filters
- IIR Frequency Response
- Negating z
- ▷ Cubing z
- Scaling z
- IIR Impulse response examples
- IIR Impulse response proof
- Other BIBO responses
- Stability Triangle
- Low-pass filter
- Allpass filters
- Group Delay
- Minimum Phase
- Summary
- MATLAB routines

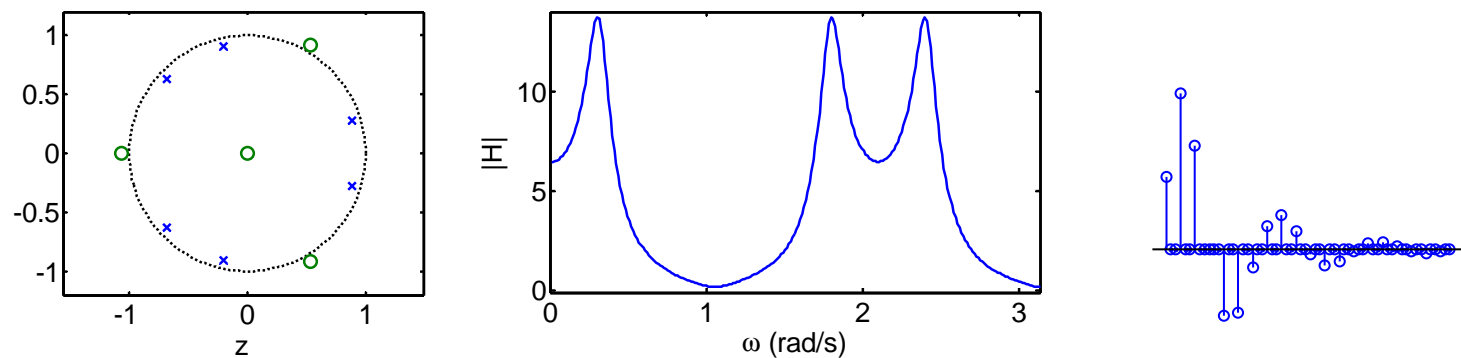
Given a filter $H(z)$ we can form a new one $H_C(z) = H(z^3)$
 Insert two zeros between each $a[n]$ and $b[n]$ term

Example: $H(z) = \frac{2+2.4z^{-1}}{1-0.96z^{-1}+0.64z^{-2}}$



Cube z: $H_C(z) = \frac{2+2.4z^{-3}}{1-0.96z^{-3}+0.64z^{-6}}$

Insert 2 zeros between coefs



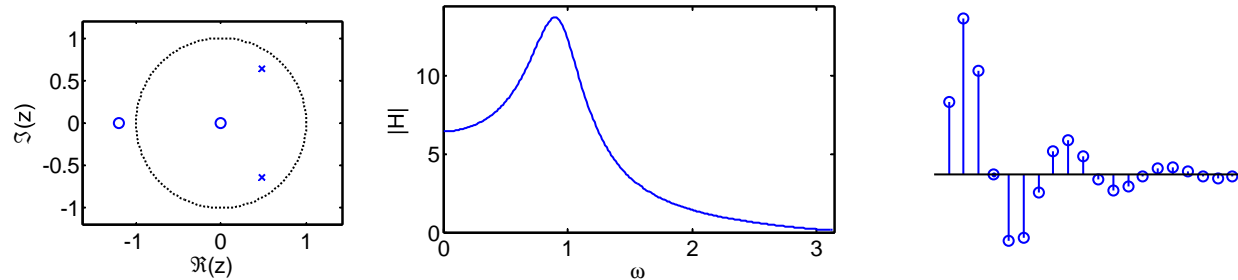
Pole and zero positions are **replicated**, magnitude response **replicated**.

Scaling z

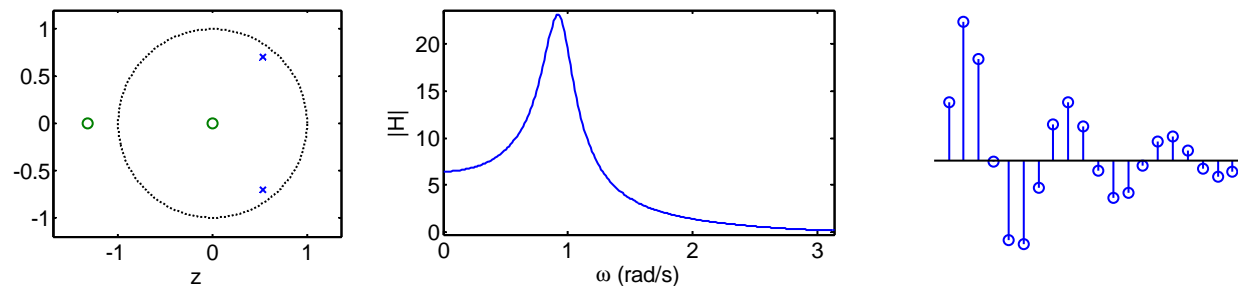
- 5: Filters
- Finite Dimensional LTI Systems
- FIR Filters
- IIR Frequency Response
- Negating z
- Cubing z
- ▷ Scaling z
- IIR Impulse response examples
- IIR Impulse response proof
- Other BIBO responses
- Stability Triangle
- Low-pass filter
- Allpass filters
- Group Delay
- Minimum Phase
- Summary
- MATLAB routines

Given a filter $H(z)$ we can form a new one $H_S(z) = H(\frac{z}{\alpha})$
 Multiply $a[n]$ and $b[n]$ by α^n

Example: $H(z) = \frac{2+2.4z^{-1}}{1-0.96z^{-1}+0.64z^{-2}}$



Scale z: $H_S(z) = H(\frac{z}{1.1}) = \frac{2+2.64z^{-1}}{1-1.056z^{-1}+0.7744z^{-2}}$



Pole and zero positions are **multiplied by α** , $\alpha > 1 \Rightarrow$ peaks **sharpened**.

Pole at $z = p$ gives peak bandwidth $\approx 2 |\log |p|| \approx 2 (1 - |p|)$

For pole near unit circle, **decrease bandwidth** by $\approx 2 \log \alpha$

IIR Impulse response examples

- 5: Filters
- Finite Dimensional LTI Systems
- FIR Filters
- IIR Frequency Response
- Negating z
- Cubing z
- Scaling z
- IIR Impulse response examples
- IIR Impulse response proof
- Other BIBO responses
- Stability Triangle
- Low-pass filter
- Allpass filters
- Group Delay
- Minimum Phase
- Summary
- MATLAB routines

To find the impulse response of $\frac{B(z)}{A(z)}$, use partial fractions:

Example 1: $M < N$

$$\frac{B(z)}{A(z)} = \frac{1-3.4z^{-1}}{1-0.3z^{-1}-0.4z^{-2}} = \frac{3}{1+0.5z^{-1}} + \frac{-2}{1-0.8z^{-1}}$$
$$h[n] = (3(-0.5)^n - 2(0.8)^n) u[n]$$

Example 2: $M \geq N$

$$\frac{B(z)}{A(z)} = \frac{2-5.7z^{-1}+0.2z^{-2}+0.8z^{-3}}{1-0.3z^{-1}-0.4z^{-2}}$$
$$= 1 - 2z^{-1} + \frac{1-3.4z^{-1}}{1-0.3z^{-1}-0.4z^{-2}}$$
$$= 1 - 2z^{-1} + \frac{3}{1+0.5z^{-1}} + \frac{-2}{1-0.8z^{-1}}$$
$$h[n] = \delta[n] - 2\delta[n-1] + (3(-0.5)^n - 2(0.8)^n) u[n]$$

Example 3: repeated pole at $z = -0.5$

$$\frac{B(z)}{A(z)} = \frac{3-5z^{-1}-1.3z^{-2}}{1+0.3z^{-1}-0.55z^{-2}-0.2z^{-3}}$$
$$= \frac{2}{1+0.5z^{-1}} + \frac{3}{(1+0.5z^{-1})^2} + \frac{-2}{1-0.8z^{-1}}$$
$$h[n] = (2(-0.5)^n + 3(n+1)(-0.5)^n - 2(0.8)^n) u[n]$$

IIR Impulse response

- 5: Filters
- Finite Dimensional LTI Systems
- FIR Filters
- IIR Frequency Response
- Negating z
- Cubing z
- Scaling z
- IIR Impulse response examples
 - IIR Impulse response proof
- Other BIBO responses
- Stability Triangle
- Low-pass filter
- Allpass filters
- Group Delay
- Minimum Phase
- Summary
- MATLAB routines

$H(z) = \frac{B(z)}{A(z)}$ we want $h[n]$, the inverse z-Transform of $H(z)$.

(1) Assume $M < N$

[Proofs on next slide]

If not, do a long division so that $H(z) = D(z) + \frac{\tilde{B}(z)}{A(z)}$

(2) Factorize $A(z) = \prod_{i=1}^{N_p} (1 - p_i z^{-1})^{K_i}$ ($N_p = \#$ of distinct poles)

(3) Then $\frac{\tilde{B}(z)}{A(z)} = \sum_{i=1}^{N_p} \sum_{k=1}^{K_i} \frac{r_{i,k}}{(1 - p_i z^{-1})^k}$ ($K_i = \text{multiplicity of } p_i$)

$$\text{where } r_{i,k} = \frac{(-p_i)^{k-K_i}}{(K_i-k)!} \frac{d^{K_i-k}}{d(z^{-1})^{K_i-k}} \left(\frac{\tilde{B}(z)}{A(z)} (1 - p_i z^{-1})^{K_i} \right) \Big|_{z=p_i}$$

For the common case $k = K_i = 1$ (e.g. all poles distinct)

$$\text{this simplifies to } r_{i,1} = \frac{\tilde{B}(z)}{A(z)} (1 - p_i z^{-1}) \Big|_{z=p_i}$$

(4) The impulse response is given by

$$h[n] = \sum_{i=1}^{N_p} \sum_{k=1}^{K_i} C_{k-1}^{n+k-1} r_{i,k} p_i^n \text{ for } n \geq 0$$

$$\text{where } C_{\beta}^{\alpha} \text{ is a binomial coefficient } C_{k-1}^{n+k-1} \sim n^{k-1}$$

Impulse response proof

- 5: Filters
- Finite Dimensional LTI Systems
- FIR Filters
- IIR Frequency Response
- Negating z
- Cubing z
- Scaling z
- IIR Impulse response examples
- IIR Impulse response
 - Impulse response
 - ▷ proof
- Other BIBO responses
- Stability Triangle
- Low-pass filter
- Allpass filters
- Group Delay
- Minimum Phase
- Summary
- MATLAB routines

Proof of (3):

(1) Multiply $\frac{\tilde{B}(z)}{A(z)} = \sum_{j=1}^{N_p} \sum_{l=1}^{K_j} \frac{r_{j,l}}{(1-p_j z^{-1})^l}$ by $(1-p_i z^{-1})^{K_i}$

summands for $j = i$ will have $(1-p_i z^{-1})^{K_i-l}$ in numerator

summands for $j \neq i$ will have $(1-p_i z^{-1})^{K_i}$ in numerator

(2) Differentiate $K_i - k$ times: $\frac{d^{K_i-k}}{d(z^{-1})^{K_i-k}}$ and evaluate at $z = p_i$

All terms that still include $(1-p_i z^{-1})$ in numerator will vanish:
i.e. all terms with $j \neq i$ and with $j = i, l < k$

for $j = i, l = k$ term is now a constant: $(-p)^{K_i-k} (K_i - k)! r_{i,k}$

for $j = i, l > k$, the terms differentiate to zero

Proof of (4):

Induction on k by differentiating and then setting $n = n' + 1$

$$\begin{aligned} \frac{d}{d(z^{-1})} \left\{ (1-pz^{-1})^{-k} = \sum_{n=0}^{\infty} C_{k-1}^{n+k-1} p^n z^{-n} \right\} \\ \rightarrow pk (1-pz^{-1})^{-(k+1)} = \sum_{n=0}^{\infty} n C_{k-1}^{n+k-1} p^n z^{-(n-1)} \end{aligned}$$

Other BIBO responses

- 5: Filters
- Finite Dimensional LTI Systems
- FIR Filters
- IIR Frequency Response
- Negating z
- Cubing z
- Scaling z
- IIR Impulse response examples
- IIR Impulse response proof
- Other BIBO responses
- Stability Triangle
- Low-pass filter
- Allpass filters
- Group Delay
- Minimum Phase
- Summary
- MATLAB routines

All LTI systems described by difference equations have $h[n]$ decaying exponentially or exponentially times a power of n .

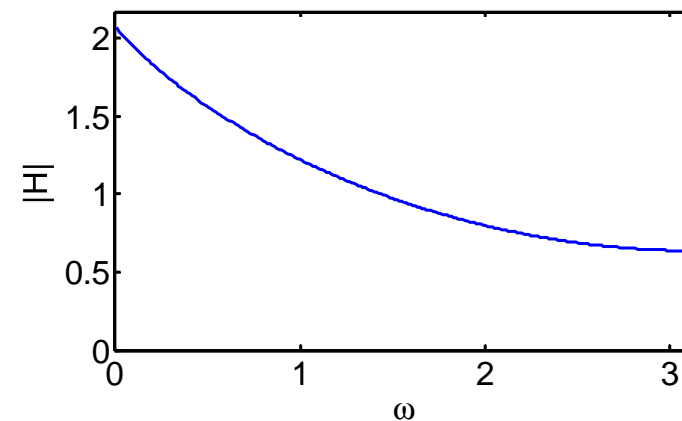
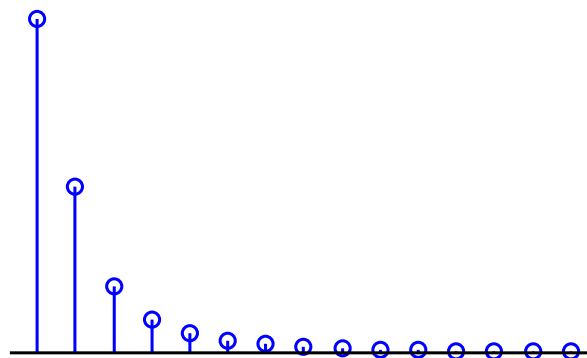
It is perfectly possible to have BIBO stable LTI systems whose impulse responses decay more slowly than this.

Example: $h[n] = \frac{1}{1+n^2}$ for $n \geq 0$

BIBO stable since $\sum |h[n]| = 2.077... < \infty$

Hence $H(e^{j\omega})$ exists.

However, no finite difference equation can have this impulse response.



Stability Triangle

- 5: Filters
- Finite Dimensional LTI Systems
- FIR Filters
- IIR Frequency Response
- Negating z
- Cubing z
- Scaling z
- IIR Impulse response examples
- IIR Impulse response proof
- Other BIBO responses
- ▷ Stability Triangle
- Low-pass filter
- Allpass filters
- Group Delay
- Minimum Phase
- Summary
- MATLAB routines

Not normally easy to tell if poles lie within $|z| = 1$, but \exists easy test for a 2nd order filter:

Suppose $A(z) = 1 + a[1]z^{-1} + a[2]z^{-2}$

The roots are $p_{1,2} = \frac{-a[1] \pm \sqrt{a[1]^2 - 4a[2]}}{2}$

Want the conditions that $|p_{1,2}| < K$

Case 1: Real roots:

$$(P) \quad a[1]^2 \geq 4a[2]$$

$$(Q) \quad -2K < -a[1] \pm \sqrt{a[1]^2 - 4a[2]} < 2K$$

$$\Rightarrow \sqrt{a[1]^2 - 4a[2]} < 2K \pm a[1]$$

$$\Rightarrow a[1]^2 - 4a[2] < 4K^2 \pm 4a[1]K + a[1]^2$$

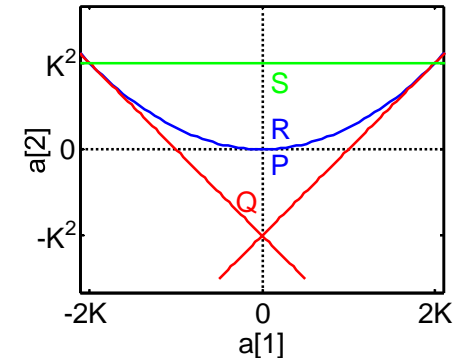
$$\Rightarrow a[2] > -K^2 \pm a[1]K$$

Case 2: Complex roots:

$$(R) \quad a[1]^2 < 4a[2]$$

$$(S) \quad |p_{1,2}|^2 = a[2] < K^2$$

Hence coefficients must lie within a **stability triangle**.



Low-pass filter

5: Filters

Finite Dimensional

LTI Systems

FIR Filters

IIR Frequency

Response

Negating z

Cubing z

Scaling z

IIR Impulse response
examples

IIR Impulse response
proof

Other BIBO
responses

Stability Triangle

▷ Low-pass filter

Allpass filters

Group Delay

Minimum Phase

Summary

MATLAB routines

1st order low pass filter: extremely common

$$y[n] = (1 - p)x[n] + py[n - 1] \Rightarrow H(z) = \frac{1-p}{1-pz^{-1}}$$

Impulse response:

$$h[n] = (1 - p)p^n = (1 - p)e^{-\frac{n}{\tau}}$$

where $\tau = \frac{-1}{\ln p}$ is the time constant in samples.

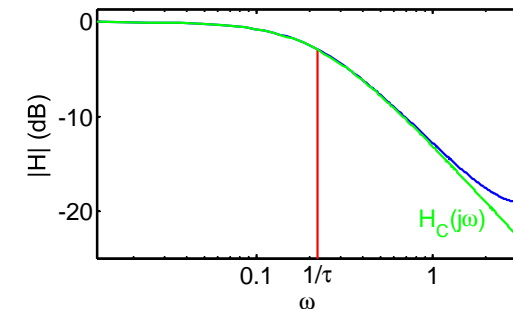
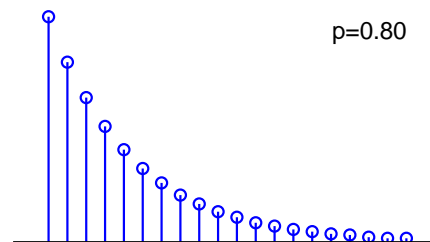
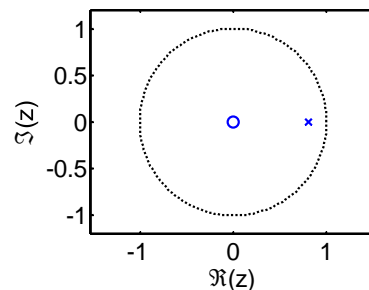
Magnitude response: $|H(e^{j\omega})| = \frac{1-p}{\sqrt{1-2p \cos \omega + p^2}}$

Low-pass filter with DC gain of unity.

3 dB frequency is $\omega_{3dB} = \cos^{-1} \left(1 - \frac{(1-p)^2}{2p} \right) \approx 2 \frac{1-p}{1+p} \approx \frac{1}{\tau}$

Compare continuous time: $H_C(j\omega) = \frac{1}{1+j\omega\tau}$

Indistinguishable for low ω but $H(e^{j\omega})$ is periodic, $H_C(j\omega)$ is not



Allpass filters

- 5: Filters
- Finite Dimensional LTI Systems
- FIR Filters
- IIR Frequency Response
- Negating z
- Cubing z
- Scaling z
- IIR Impulse response examples
- IIR Impulse response proof
- Other BIBO responses
- Stability Triangle
- Low-pass filter
- ▷ Allpass filters
- Group Delay
- Minimum Phase
- Summary
- MATLAB routines

If $a[n] = b[M - n]$ we have an **allpass filter**

$$H(e^{j\omega}) = \frac{\sum_{r=0}^M b[r]e^{-j\omega r}}{\sum_{r=0}^M b[r]e^{-j\omega(M-r)}} = e^{j\omega M} \frac{\sum_{r=0}^M b[r]e^{-j\omega r}}{\sum_{r=0}^M b[r]e^{j\omega r}}$$

The two sums are complex conjugates \Rightarrow they have the same magnitude

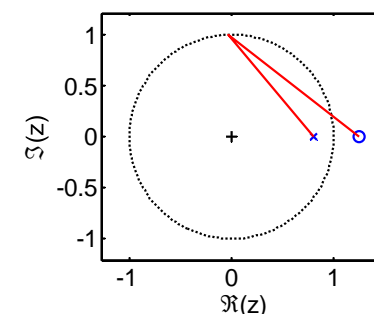
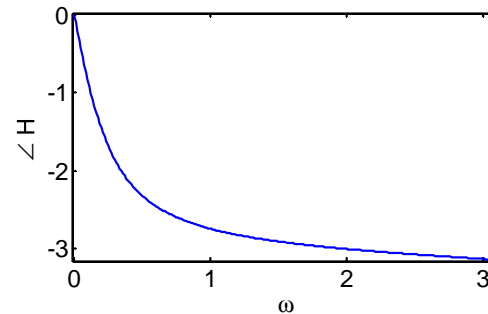
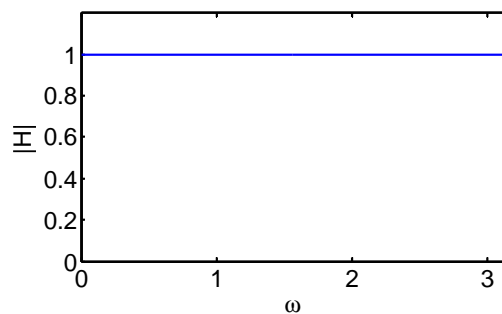
Hence $|H(e^{j\omega})| = 1 \forall \omega \Leftrightarrow$ “**allpass**”

However phase is **not** constant: $\angle H(e^{j\omega}) = \omega M + 2\angle B(e^{j\omega})$

1st order allpass: $H(z) = \frac{-p+z^{-1}}{1-pz^{-1}} = -p \frac{1-p^{-1}z^{-1}}{1-pz^{-1}}$

Pole at p and zero at p^{-1} : “**reflected in unit circle**”

Constant distance ratio: $|e^{j\omega} - p| = |p| \left| e^{j\omega} - \frac{1}{p} \right| \forall \omega$



In an allpass filter, the **zeros are the poles reflected in the unit circle.**

Group Delay

5: Filters

Finite Dimensional
LTI Systems

FIR Filters
IIR Frequency
Response

Negating z

Cubing z

Scaling z

IIR Impulse response
examples

IIR Impulse response
proof

Other BIBO
responses

Stability Triangle

Low-pass filter

Allpass filters

▷ Group Delay

Minimum Phase

Summary

MATLAB routines

Group delay: $\tau_H(e^{j\omega}) = -\frac{d\angle H(e^{j\omega})}{d\omega} = \text{delay of the modulation envelope.}$

Trick to get at phase: $\ln H(e^{j\omega}) = \ln |H(e^{j\omega})| + j\angle H(e^{j\omega})$

$$\tau_H = \frac{-d(\Im(\ln H(e^{j\omega})))}{d\omega} = \Im \left(\frac{-1}{H(e^{j\omega})} \frac{dH(e^{j\omega})}{d\omega} \right) = \Re \left(\frac{-z}{H(z)} \frac{dH}{dz} \right) \Big|_{z=e^{j\omega}}$$

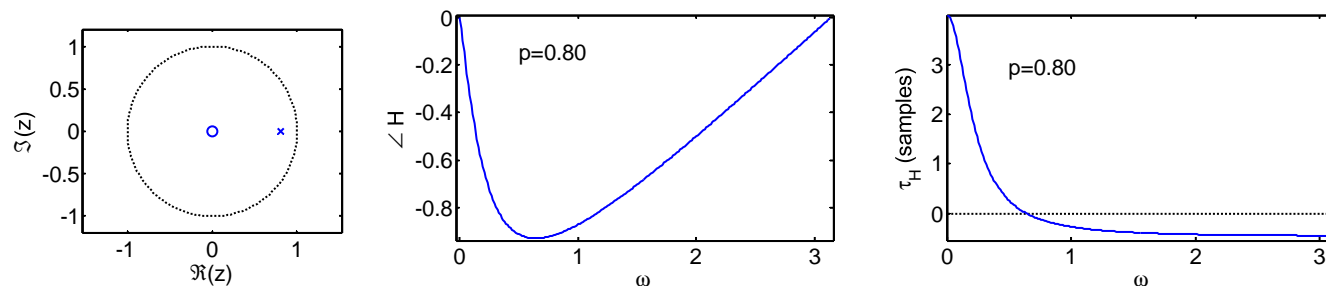
$$H(e^{j\omega}) = \sum_{n=0}^{\infty} h[n]e^{-jn\omega} = \mathcal{F}(h[n])$$

[\mathcal{F} = DTFT]

$$\frac{dH(e^{j\omega})}{d\omega} = \sum_{n=0}^{\infty} -jnh[n]e^{-jn\omega} = -j\mathcal{F}(nh[n])$$

$$\tau_H = \Im \left(\frac{-1}{H(e^{j\omega})} \frac{dH(e^{j\omega})}{d\omega} \right) = \Im \left(\frac{j\mathcal{F}(nh[n])}{\mathcal{F}(h[n])} \right) = \Re \left(\frac{\mathcal{F}(nh[n])}{\mathcal{F}(h[n])} \right)$$

Example: $H(z) = \frac{1}{1-pz^{-1}} \Rightarrow \tau_H = -\tau_{[1-p]} = -\Re \left(\frac{-pe^{-j\omega}}{1-pe^{-j\omega}} \right)$



Average group delay (over ω) = (# poles - # zeros) within the unit circle
Zeros on the unit circle count $-\frac{1}{2}$

++ Group Delay

The group delay of a filter $H(z)$ at a frequency ω , measured in seconds or samples, gives the time delay of the envelope of a modulated sine wave at a frequency ω . It is defined as $\tau_H(e^{j\omega}) = -\frac{d\angle H(e^{j\omega})}{d\omega}$. For example, $H(z) = z^{-k}$ defines a filter that delays its input by k samples and we can calculate the group delay by evaluating

$$\tau_H(e^{j\omega}) = -\frac{d\angle H(e^{j\omega})}{d\omega} = -\frac{d}{d\omega} \left(\angle e^{-jk\omega} \right) = -\frac{d}{d\omega} (-k\omega) = k$$

which tells us that this filter has a constant group delay of k samples that is independent of ω .

The average value of τ_H equals the total change in $-\angle H(e^{j\omega})$ as ω goes from $-\pi$ to $+\pi$ divided by 2π . If you imagine an elastic string connecting a pole or zero to the point $z = e^{j\omega}$, you can see that as ω goes from $-\pi$ to $+\pi$ the string will wind once around the pole or zero if it is inside the unit circle but not if it is outside. Thus, the total change in $\angle H(e^{j\omega})$ is equal to 2π times the number of poles inside the unit circle minus the number of zeros inside the unit circle. A zero that is exactly on the unit circle counts $\frac{1}{2}$ since there is a sudden discontinuity of π in $\angle H(e^{j\omega})$ as ω passes through the zero position.

When you multiply or divide complex numbers, their phases add or subtract, so it follows that when you multiply or divide transfer functions their group delays will add or subtract. Thus, for example, the group delay of an IIR filter, $H(z) = \frac{B(z)}{A(z)}$, is given by $\tau_H = \tau_B - \tau_A$. This means too that we can determine the group delay of a factorized transfer function by summing the group delays of the individual factors.

++ Group Delay

The slide shows how to determine the group delay, τ_H , from either the impulse response, $h[n]$, or the transfer function, $H(z)$. We start by using a trick that is very common: if you want to get at the magnitude and phase of a complex number separately, you can do so by taking its natural log: $\ln(re^{j\theta}) = \ln|r| + j\theta$ or, in general, $\ln H = \ln|H| + j\angle H$. By rearranging this equation, we get $\angle H = \Im(\ln H)$ where $\Im(\cdot)$ denotes taking the imaginary part of a complex number. Using this, we can write

$$\tau_H = \frac{-d(\Im(\ln H(e^{j\omega})))}{d\omega} = \Im\left(\frac{-d(\ln H(e^{j\omega}))}{d\omega}\right) = \Im\left(\frac{-1}{H(e^{j\omega})} \frac{dH(e^{j\omega})}{d\omega}\right). \quad (1)$$

By going back to the definition of the DTFT, we find that $H(e^{j\omega}) = \mathcal{F}(h[n])$ and $\frac{dH(e^{j\omega})}{d\omega} = -j\mathcal{F}(nh[n])$ where $\mathcal{F}(\cdot)$ denotes the DTFT. Substituting these expressions into the above equation gives us a formula for τ_H in terms of the impulse response $h[n]$.

$$\tau_H = \Re\left(\frac{\mathcal{F}(nh[n])}{\mathcal{F}(h[n])}\right) \quad (2)$$

In order to express τ_H in terms of z , we first note that if $z = e^{j\omega}$ then $\frac{dz}{d\omega} = jz$. By substituting $z = e^{j\omega}$ into equation (1), we get

$$\tau_H = \Im\left(\frac{-1}{H(z)} \frac{dH(z)}{d\omega}\right) = \Im\left(\frac{-1}{H(z)} \frac{dH(z)}{dz} \frac{dz}{d\omega}\right) = \Im\left(\frac{-jz}{H(z)} \frac{dH(z)}{dz}\right) = \Re\left(\frac{-z}{H(z)} \frac{dH(z)}{dz}\right)\bigg|_{z=e^{j\omega}}.$$

++ Group Delay

As an example, suppose we want to determine the group delay of : $H(z) = \frac{1}{1-pz^{-1}}$. As noted above, if $H(z) = \frac{B(z)}{A(z)}$, then $\tau_H = \tau_B - \tau_A$. In this case $\tau_B = 0$ so $\tau_H = -\tau_{[1-p]}$.

Using equation (2) gives $\tau_H = -\Re \left(\frac{\mathcal{F}([0 \ -p])}{\mathcal{F}([1 \ -p])} \right)$ since $nh[n] = [0 \ 1] \times [1 \ -p]$.

Applying the definition of the DTFT, we get

$$\tau_H = -\Re \left(\frac{-pe^{-j\omega}}{1 - pe^{-j\omega}} \right) = \Re \left(\frac{p}{e^{j\omega} - p} \right) = \frac{\Re(p(e^{-j\omega} - p))}{(e^{j\omega} - p)(e^{-j\omega} - p)} = \frac{p \cos \omega - p^2}{1 - 2p \cos \omega + p^2}$$

As demonstrated above, the average value of τ_H is zero for this filter because there is one pole and one zero inside the unit circle.

Minimum Phase

5: Filters

Finite Dimensional

LTI Systems

FIR Filters

IIR Frequency

Response

Negating z

Cubing z

Scaling z

IIR Impulse response
examples

IIR Impulse response
proof

Other BIBO
responses

Stability Triangle

Low-pass filter

Allpass filters

Group Delay

▷ Minimum Phase

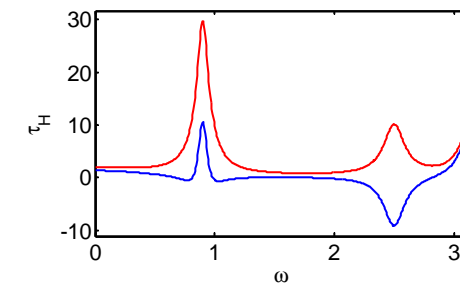
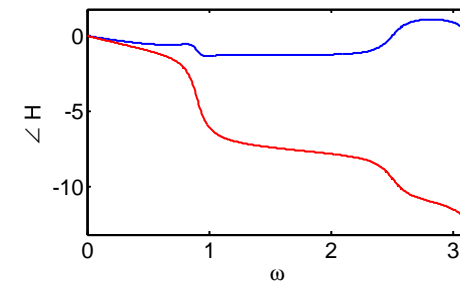
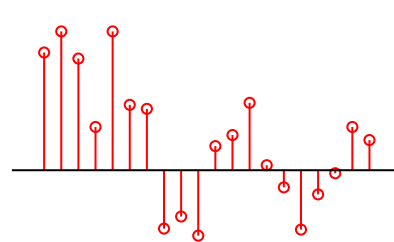
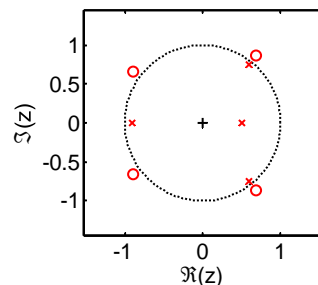
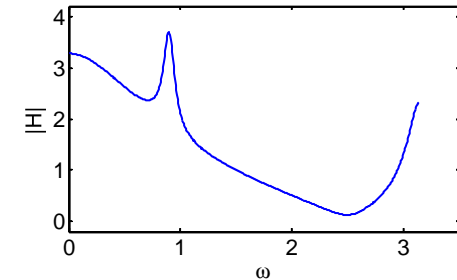
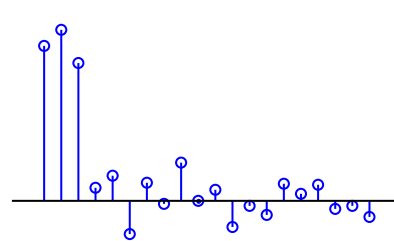
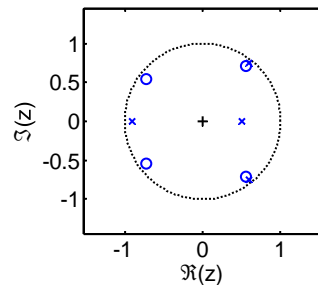
Summary

MATLAB routines

Average group delay (over ω) = (# poles – # zeros) within the unit circle

- zeros on the unit circle count $-\frac{1}{2}$

Reflecting an interior zero to the exterior multiplies $|H(e^{j\omega})|$ by a constant but **increases average group delay** by 1 sample.



A filter with all zeros inside the unit circle is a **minimum phase** filter:

- **Lowest possible group delay** for a given magnitude response
- Energy in $h[n]$ is **concentrated towards $n = 0$**

Summary

- 5: Filters
- Finite Dimensional LTI Systems
- FIR Filters
- IIR Frequency Response
- Negating z
- Cubing z
- Scaling z
- IIR Impulse response examples
- IIR Impulse response proof
- Other BIBO responses
- Stability Triangle
- Low-pass filter
- Allpass filters
- Group Delay
- Minimum Phase
- ▷ Summary
- MATLAB routines

Useful filters have **difference equations**:

Freq response determined by pole/zero positions

$N - M$ zeros at origin (or $M - N$ poles)

Geometric construction of $|H(e^{j\omega})|$

Stable if poles have $|p| < 1$

Allpass filter: $a[n] = b[M - n]$

Reflecting a zero in unit circle leaves $|H(e^{j\omega})|$ unchanged

Group delay: $\tau_H(e^{j\omega}) = -\frac{d\angle H(e^{j\omega})}{d\omega}$ samples

Symmetrical $h[n] \Leftrightarrow \tau_H(e^{j\omega}) = \frac{M}{2} \forall \omega$

Average τ_H over $\omega = (\# \text{ poles} - \# \text{ zeros})$ within the unit circle

Minimum phase if zeros have $|q| \leq 1$

Lowest possible group delay for given $|H(e^{j\omega})|$

For further details see Mitra: 6, 7.

MATLAB routines

5: Filters

Finite Dimensional

LTI Systems

FIR Filters

IIR Frequency
Response

Negating z

Cubing z

Scaling z

IIR Impulse response
examples

IIR Impulse response
Impulse response
proof

Other BIBO
responses

Stability Triangle

Low-pass filter

Allpass filters

Group Delay

Minimum Phase

Summary

▷ MATLAB routines

filter	filter a signal
impz	Impulse response
residuez	partial fraction expansion
grpdelay	Group Delay
freqz	Calculate filter frequency response

▷ **6: Window Filter
Design**

Inverse DTFT

Rectangular window

Dirichlet Kernel

Window relationships

Hanning Window

Common Windows

Uncertainty principle

Order Estimation

Example Design

Frequency sampling

Linear Phase Filters

Summary

MATLAB routines

6: Window Filter Design

Inverse DTFT

6: Window Filter Design

▷ Inverse DTFT

Rectangular window

Dirichlet Kernel

Window relationships

Hanning Window

Common Windows

Uncertainty principle

Order Estimation

Example Design

Frequency sampling

Linear Phase Filters

Summary

MATLAB routines

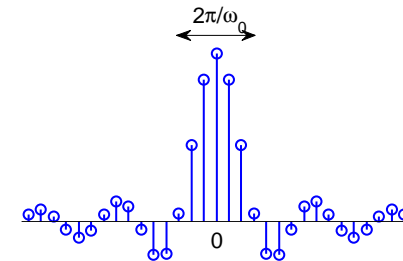
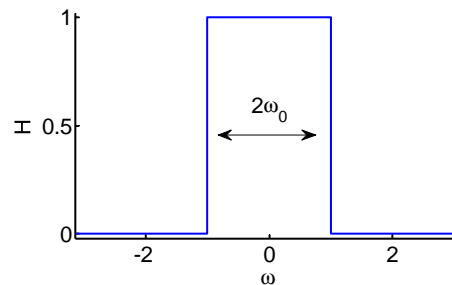
For any BIBO stable filter, $H(e^{j\omega})$ is the DTFT of $h[n]$

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

If we know $H(e^{j\omega})$ exactly, the IDTFT gives the ideal $h[n]$

Example: Ideal Lowpass filter

$$H(e^{j\omega}) = \begin{cases} 1 & |\omega| \leq \omega_0 \\ 0 & |\omega| > \omega_0 \end{cases} \Leftrightarrow h[n] = \frac{\sin \omega_0 n}{\pi n}$$



Note: Width in ω is $2\omega_0$, width in n is $\frac{2\pi}{\omega_0}$: **product is 4π always**
Sadly $h[n]$ is **infinite** and **non-causal**. **Solution:** multiply $h[n]$ by a window

Rectangular window

6: Window Filter Design

Inverse DTFT

▷ Rectangular window

Dirichlet Kernel

Window relationships

Hanning Window

Common Windows

Uncertainty principle

Order Estimation

Example Design

Frequency sampling

Linear Phase Filters

Summary

MATLAB routines

Truncate to $\pm \frac{M}{2}$ to make finite; $h_1[n]$ is now of length $M + 1$

MSE Optimality:

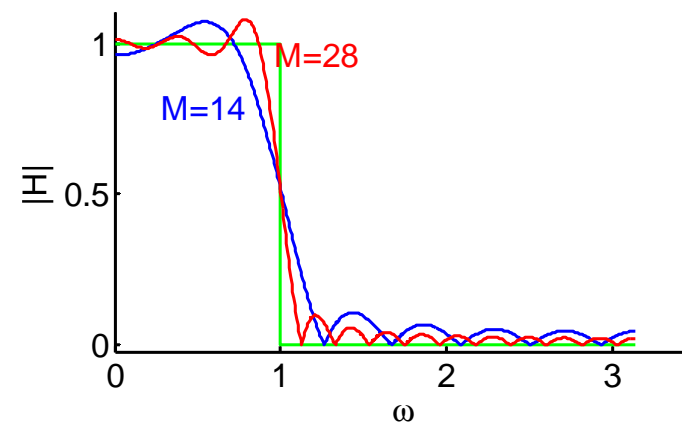
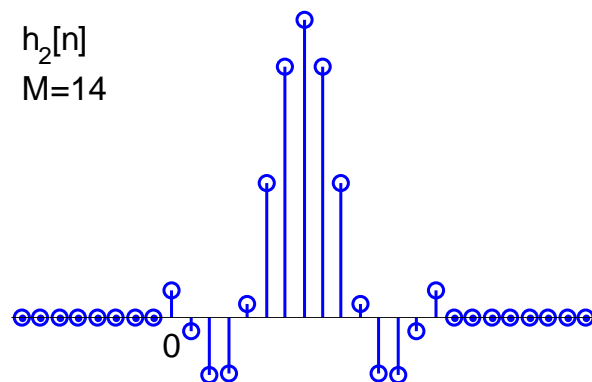
Define mean square error (MSE) in frequency domain

$$\begin{aligned} E &= \frac{1}{2\pi} \int_{-\pi}^{\pi} |H(e^{j\omega}) - H_1(e^{j\omega})|^2 d\omega \\ &= \frac{1}{2\pi} \int_{-\pi}^{\pi} \left| H(e^{j\omega}) - \sum_{-\frac{M}{2}}^{\frac{M}{2}} h_1[n] e^{-j\omega n} \right|^2 d\omega \end{aligned}$$

Minimum E is when $h_1[n] = \Re(h[n])$

Proof: Differentiate w.r.t. $h[r]$ and set to zero

However: 9% overshoot at a discontinuity even for large n .



Normal to delay by $\frac{M}{2}$ to make causal. Multiplies $H(e^{j\omega})$ by $e^{-j\frac{M}{2}\omega}$.

Dirichlet Kernel

6: Window Filter Design

Inverse DTFT

Rectangular window

▷ Dirichlet Kernel

Window relationships

Hanning Window

Common Windows

Uncertainty principle

Order Estimation

Example Design

Frequency sampling

Linear Phase Filters

Summary

MATLAB routines

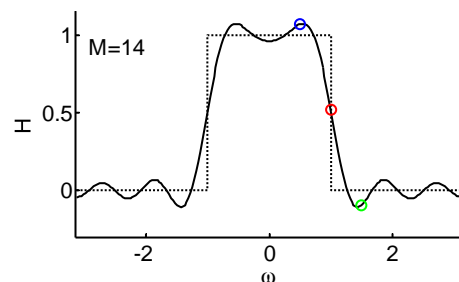
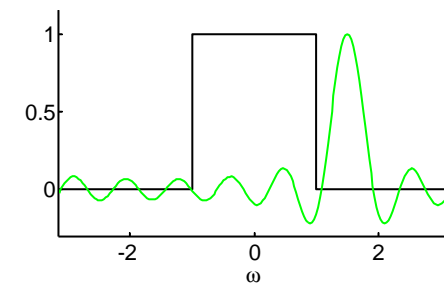
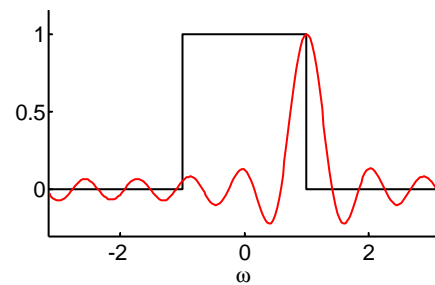
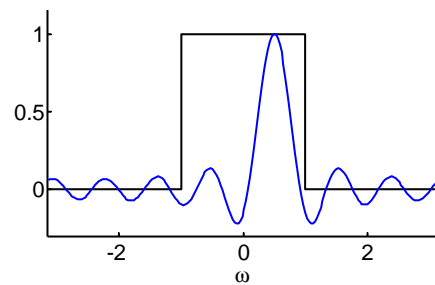
Truncation \Leftrightarrow Multiply $h[n]$ by a rectangular window, $w[n] = \delta_{-\frac{M}{2} \leq n \leq \frac{M}{2}}$

$$\Leftrightarrow \text{Circular Convolution } H_1(e^{j\omega}) = \frac{1}{2\pi} H(e^{j\omega}) \circledast W(e^{j\omega})$$

$$W(e^{j\omega}) = \sum_{-\frac{M}{2}}^{\frac{M}{2}} e^{-j\omega n} \stackrel{(i)}{=} 1 + 2 \sum_1^{0.5M} \cos(n\omega) \stackrel{(ii)}{=} \frac{\sin 0.5(M+1)\omega}{\sin 0.5\omega}$$

Proof: (i) $e^{-j\omega(-n)} + e^{-j\omega(+n)} = 2 \cos(n\omega)$ (ii) Sum geom. progression

Effect: convolve ideal freq response with **Dirichlet kernel** (aliased sinc)



Provided that $\frac{4\pi}{M+1} < 2\omega_0 \Leftrightarrow M+1 > \frac{2\pi}{\omega_0}$:

Passband ripple: $\Delta\omega \approx \frac{4\pi}{M+1}$, stopband $\frac{2\pi}{M+1}$

Transition pk-to-pk: $\Delta\omega \approx \frac{4\pi}{M+1}$

Transition Gradient: $\left. \frac{d|H|}{d\omega} \right|_{\omega=\omega_0} \approx \frac{M+1}{2\pi}$

Window relationships

6: Window Filter Design

Inverse DTFT

Rectangular window

Dirichlet Kernel

Window relationships

Hanning Window

Common Windows

Uncertainty principle

Order Estimation

Example Design

Frequency sampling

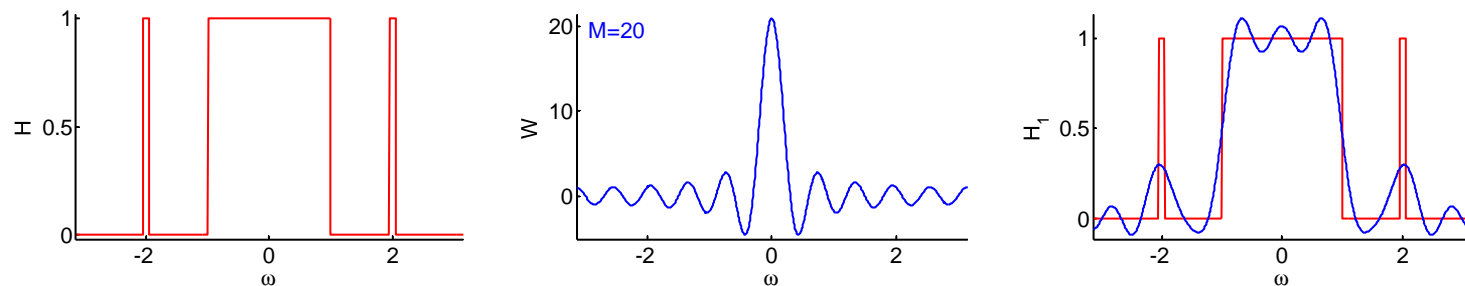
Linear Phase Filters

Summary

MATLAB routines

When you multiply an impulse response by a window $M + 1$ long

$$H_1(e^{j\omega}) = \frac{1}{2\pi} H(e^{j\omega}) \otimes W(e^{j\omega})$$



(a) passband gain $\approx w[0]$; peak $\approx \frac{w[0]}{2} + \frac{1}{4\pi} \times \int_{\text{mainlobe}} W(e^{j\omega}) d\omega$
 rectangular window: passband gain = 1; peak gain = 1.09

(b) stopband gain is an integral over oscillating sidelobes
 hence lower ripple than the sidelobes due to cancellation

(c) transition **bandwidth**, $\Delta\omega$ = width of the main lobe
 transition **amplitude**, ΔH = integral of main lobe $\div 2\pi$
 rectangular window: $\Delta\omega = \frac{4\pi}{M+1}$, $\Delta H \approx 1.18$

(d) features narrower than the main lobe will be broadened and attenuated

Hanning Window

6: Window Filter Design

Inverse DTFT

Rectangular window

Dirichlet Kernel

Window relationships

▷ Hanning Window

Common Windows

Uncertainty principle

Order Estimation

Example Design

Frequency sampling

Linear Phase Filters

Summary

MATLAB routines

Hanning (von Hann) window:

$$w[n] = 0.5 + 0.5 \cos \frac{2\pi n}{M+1}$$

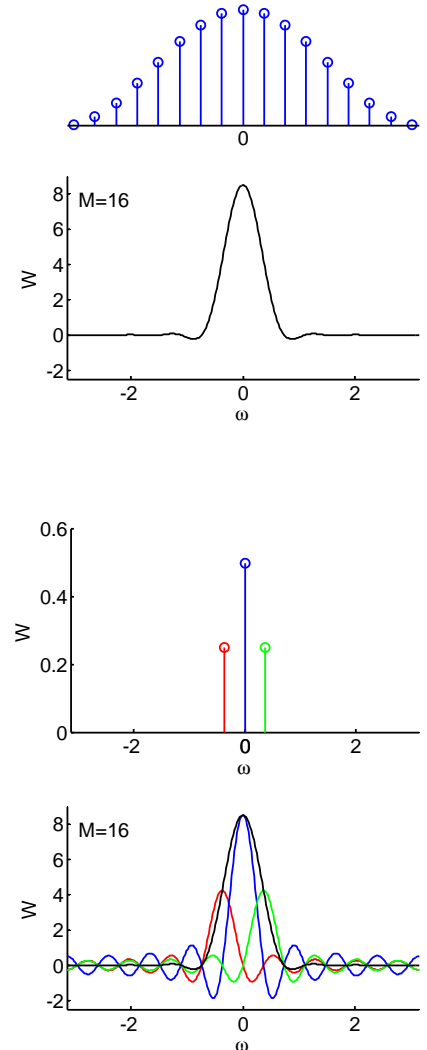
Some use M or $M + 2$ in denominator [bad]

- Smoothly tapers to zero at ends
- DTFT has much **reduced ripples** ☺
- Central lobe **width doubled**: $\Delta\omega = \frac{8\pi}{M+1}$ ☹

Regard $w[n]$ as infinite signal, $0.5 + 0.5 \cos \frac{2\pi n}{M+1}$, multiplied by a rectangular window

$W(e^{j\omega})$ is spectrum of signal \circledast Dirichlet kernel

- Add components to get $W(e^{j\omega})$
- Anti-phase sidelobes cancel
- Central lobe width doubled



Common Windows

6: Window Filter Design

Inverse DTFT

Rectangular window

Dirichlet Kernel

Window relationships

Hanning Window

▷ Common Windows

Uncertainty principle

Order Estimation

Example Design

Frequency sampling

Linear Phase Filters

Summary

MATLAB routines

Rectangular: $w[n] \equiv 1$

don't use

Hanning: $0.5 + 0.5c_1$

$$c_k = \cos \frac{2\pi kn}{M+1}$$

rapid sidelobe decay

Hamming: $0.54 + 0.46c_1$

best peak sidelobe

Blackman-Harris 3-term:

$$0.42 + 0.5c_1 + 0.08c_2$$

best peak sidelobe

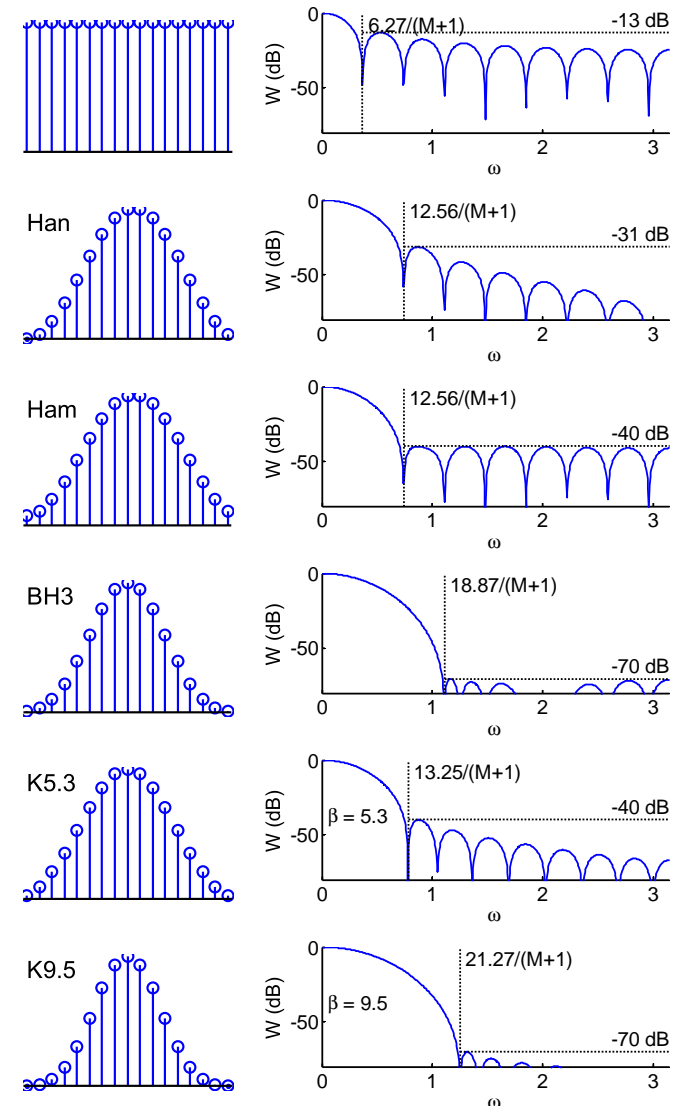
Kaiser:

$$\frac{I_0\left(\beta\sqrt{1-\left(\frac{2n}{M}\right)^2}\right)}{I_0(\beta)}$$

β controls width v sidelobes

Good compromise:

Width v sidelobe v decay



Uncertainty principle

6: Window Filter Design

Inverse DTFT

Rectangular window

Dirichlet Kernel

Window relationships

Hanning Window

Common Windows

Uncertainty

▷ principle

Order Estimation

Example Design

Frequency sampling

Linear Phase Filters

Summary

MATLAB routines

$$\text{CTFT: } \left(\frac{\int t^2 |x(t)|^2 dt}{\int |x(t)|^2 dt} \right)^{\frac{1}{2}} \left(\frac{\int \omega^2 |X(\omega)|^2 d\omega}{\int |X(\omega)|^2 d\omega} \right)^{\frac{1}{2}} \geq \frac{1}{2}$$

First term measures the “width” of $x(t)$ around $t = 0$.

like σ if $|x(t)|^2$ was a zero-mean probability distribution

Second term is similarly the “width” of $X(\omega)$.

A signal **cannot be concentrated in both time and frequency**.

So a short window cannot have a narrow main lobe.

Proof:

Assume $\int |x(t)|^2 dt = 1 \Rightarrow \int |X(\omega)|^2 d\omega = 2\pi$ [Parseval]

Set $y(t) = \frac{dx}{dt} \Rightarrow Y(\omega) = j\omega X(\omega)$ [by parts]

Now $\int tx \frac{dx}{dt} dt = \frac{1}{2} tx^2 \Big|_{t=-\infty}^{\infty} - \int \frac{1}{2} x^2 dt = -\frac{1}{2}$ [by parts]

So $\frac{1}{4} = \left| \int tx \frac{dx}{dt} dt \right|^2 \leq \left(\int t^2 x^2 dt \right) \left(\int \left| \frac{dx}{dt} \right|^2 dt \right)$ [Schwartz]

$$= \left(\int t^2 x^2 dt \right) \left(\int |y|^2 dt \right) = \left(\int t^2 x^2 dt \right) \left(\frac{1}{2\pi} \int |Y|^2 d\omega \right)$$

$$= \left(\int t^2 x^2 dt \right) \left(\frac{1}{2\pi} \int \omega^2 |X|^2 d\omega \right)$$

No exact equivalent for DTFT/DFT but a similar effect is true

Order Estimation

6: Window Filter Design

Inverse DTFT

Rectangular window

Dirichlet Kernel

Window relationships

Hanning Window

Common Windows

Uncertainty principle

▷ Order Estimation

Example Design

Frequency sampling

Linear Phase Filters

Summary

MATLAB routines

Several formulae estimate the required order of a filter, M .

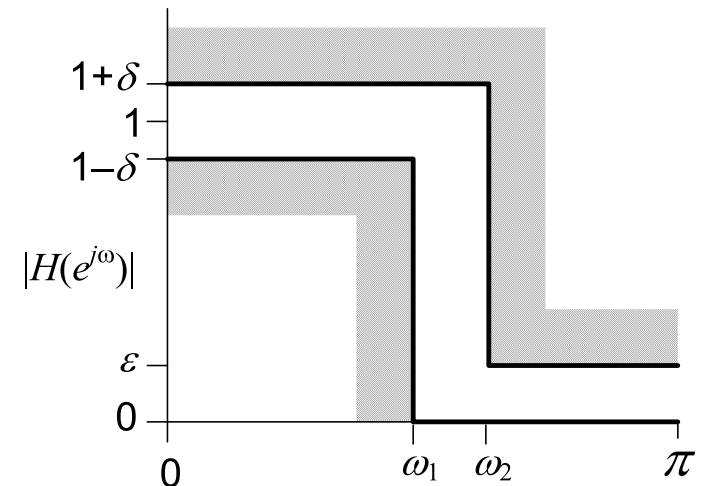
E.g. for lowpass filter

Estimated order is

$$M \approx \frac{-5.6 - 4.3 \log_{10}(\delta\epsilon)}{\omega_2 - \omega_1} \approx \frac{1/\epsilon(\text{in dB}) - 8}{2.2\Delta\omega}$$

Required M increases as either the transition width, $\omega_2 - \omega_1$, or the gain tolerances δ and ϵ get smaller.

Only approximate.



Example:

Transition band: $f_1 = 1.8$ kHz, $f_2 = 2.0$ kHz, $f_s = 12$ kHz,

$$\omega_1 = \frac{2\pi f_1}{f_s} = 0.943, \quad \omega_2 = \frac{2\pi f_2}{f_s} = 1.047$$

Ripple: $\delta = 0.1$ dB, $\epsilon = -35$ dB

$$\delta = 10^{\frac{0.1}{20}} - 1 = 0.0116, \quad \epsilon = 10^{\frac{-35}{20}} = 0.0178$$

$$M \approx \frac{-5.6 - 4.3 \log_{10}(2 \times 10^{-4})}{1.047 - 0.943} = \frac{10.25}{0.105} = 98 \quad \text{or} \quad \frac{35 - 8}{2.2\Delta\omega} = 117$$

Example Design

6: Window Filter Design

Inverse DTFT

Rectangular window

Dirichlet Kernel

Window relationships

Hanning Window

Common Windows

Uncertainty principle

Order Estimation

▷ Example Design

Frequency sampling

Linear Phase Filters

Summary

MATLAB routines

Specifications:

Bandpass: $\omega_1 = 0.5$, $\omega_2 = 1$

Transition bandwidth: $\Delta\omega = 0.1$

Ripple: $\delta = \epsilon = 0.02$

$$20 \log_{10} \epsilon = -34 \text{ dB}$$

$$20 \log_{10} (1 + \delta) = 0.17 \text{ dB}$$

Order:

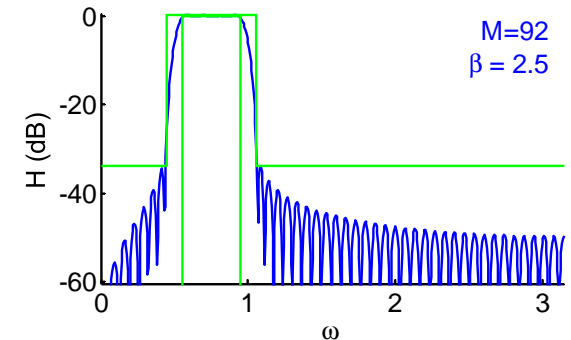
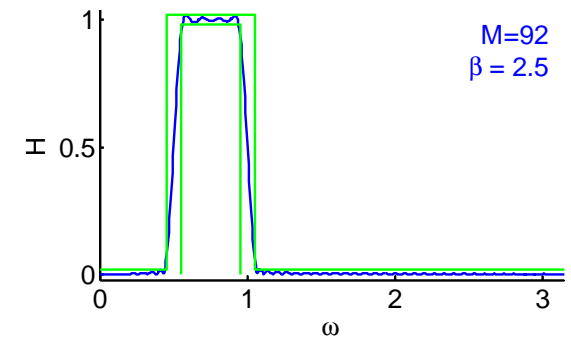
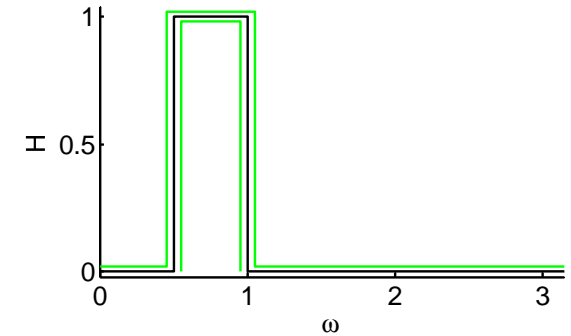
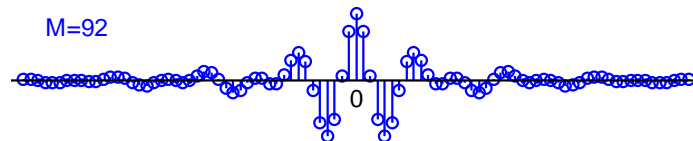
$$M \approx \frac{-5.6 - 4.3 \log_{10}(\delta\epsilon)}{\omega_2 - \omega_1} = 92$$

Ideal Impulse Response:

Difference of two lowpass filters

$$h[n] = \frac{\sin \omega_2 n}{\pi n} - \frac{\sin \omega_1 n}{\pi n}$$

Kaiser Window: $\beta = 2.5$



Frequency sampling

6: Window Filter Design

Inverse DTFT

Rectangular window

Dirichlet Kernel

Window relationships

Hanning Window

Common Windows

Uncertainty principle

Order Estimation

Example Design

Frequency

▷ sampling

Linear Phase Filters

Summary

MATLAB routines

Take $M + 1$ uniform samples of $H(e^{j\omega})$; take IDFT to obtain $h[n]$

Advantage:

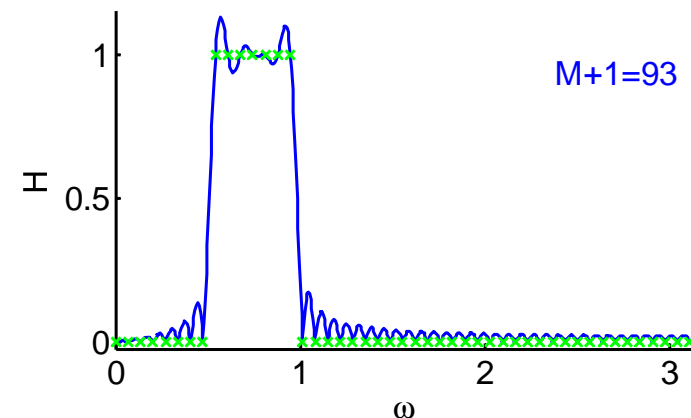
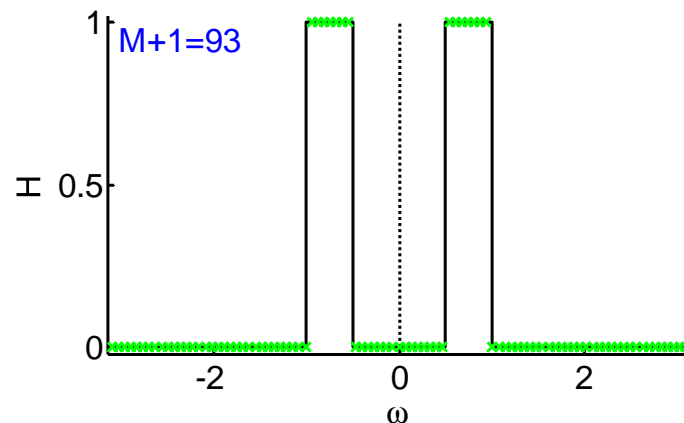
exact match at sample points

Disadvantage:

poor intermediate approximation if spectrum is varying rapidly

Solutions:

- (1) make the filter transitions smooth over $\Delta\omega$ width
- (2) oversample and do least squares fit (can't use IDFT)
- (3) use non-uniform points with more near transition (can't use IDFT)



Linear Phase Filters

6: Window Filter Design

Inverse DTFT

Rectangular window

Dirichlet Kernel

Window relationships

Hanning Window

Common Windows

Uncertainty principle

Order Estimation

Example Design

Frequency sampling

Linear Phase ▷ Filters

Summary

MATLAB routines

All FIR filters discussed have had **linear phase**: $\angle H(e^{j\omega}) = \theta_0 - \frac{M}{2}\omega$

Equivalently **constant group delay**: $\tau_H = -\frac{d\angle H(e^{j\omega})}{d\omega} = \frac{M}{2}$

A filter has linear phase iff $h[n]$ is **symmetric** or **antisymmetric**:

$$h[n] = h[M - n] \forall n \text{ or else } h[n] = -h[M - n] \forall n$$

M can be even ($\Rightarrow \exists$ mid point) or odd ($\Rightarrow \nexists$ mid point)

Proof \Leftarrow :

$$\begin{aligned} 2H(e^{j\omega}) &= \sum_0^M h[n]e^{-j\omega n} + \sum_0^M h[M - n]e^{-j\omega(M-n)} \\ &= e^{-j\omega \frac{M}{2}} \sum_0^M h[n]e^{-j\omega(n - \frac{M}{2})} + h[M - n]e^{j\omega(n - \frac{M}{2})} \end{aligned}$$

$h[n]$ symmetric:

$$2H(e^{j\omega}) = 2e^{-j\omega \frac{M}{2}} \sum_0^M h[n] \cos\left(n - \frac{M}{2}\right) \omega$$

$h[n]$ anti-symmetric:

$$\begin{aligned} 2H(e^{j\omega}) &= -2je^{-j\omega \frac{M}{2}} \sum_0^M h[n] \sin\left(n - \frac{M}{2}\right) \omega \\ &= 2e^{-j\left(\frac{\pi}{2} + \omega \frac{M}{2}\right)} \sum_0^M h[n] \sin\left(n - \frac{M}{2}\right) \omega \end{aligned}$$

Summary

6: Window Filter Design

Inverse DTFT

Rectangular window

Dirichlet Kernel

Window relationships

Hanning Window

Common Windows

Uncertainty principle

Order Estimation

Example Design

Frequency sampling

Linear Phase Filters

▷ Summary

MATLAB routines

- Make an FIR filter by windowing the IDTFT of ideal response
Ideal lowpass has $h[n] = \frac{\sin \omega_0 n}{\pi n}$
Add/subtract lowpass filters to make any piecewise constant response
- Ideal response is \otimes with window DTFT
Rectangular window (Dirichlet kernel) has -13 dB sidelobes and is always a bad idea
Hamming, Blackman-Harris are good; Kaiser good with β
- Uncertainty principle: cannot be concentrated in both time and frequency
- Frequency sampling: IDFT of uniform frequency samples: not so great
- Linear phase = Constant group Delay = symmetric/antisymmetric $h[n]$
- For further details see Mitra: 7, 10.

MATLAB routines

6: Window Filter Design

Inverse DTFT
Rectangular window
Dirichlet Kernel
Window relationships
Hanning Window
Common Windows
Uncertainty principle
Order Estimation
Example Design
Frequency sampling
Linear Phase Filters
Summary
▷ MATLAB routines

diric(x,n)	Dirichlet kernel: $\frac{\sin 0.5nx}{\sin 0.5x}$
hanning hamming kaiser	Window functions (Note 'periodic' option)
kaiserord	Estimate required filter order and β

▷ **7: Optimal FIR
filters**

Optimal Filters

Alternation Theorem

Chebyshev

Polynomials

Maximal Error

Locations

Remez Exchange

Algorithm

Determine Polynomial

Example Design

FIR Pros and Cons

Summary

MATLAB routines

7: Optimal FIR filters

Optimal Filters

7: Optimal FIR filters

▷ Optimal Filters

Alternation Theorem

Chebyshev

Polynomials

Maximal Error

Locations

Remez Exchange

Algorithm

Determine Polynomial

Example Design

FIR Pros and Cons

Summary

MATLAB routines

We restrict ourselves to zero-phase filters of odd length $M + 1$, symmetric around $h[0]$, i.e. $h[-n] = h[n]$.

$$\overline{H}(\omega) = H(e^{j\omega}) = \sum_{-\frac{M}{2}}^{\frac{M}{2}} h[n] e^{-jn\omega} = h[0] + 2 \sum_1^{\frac{M}{2}} h[n] \cos n\omega$$

$\overline{H}(\omega)$ is real but not necessarily positive (unlike $|H(e^{j\omega})|$).

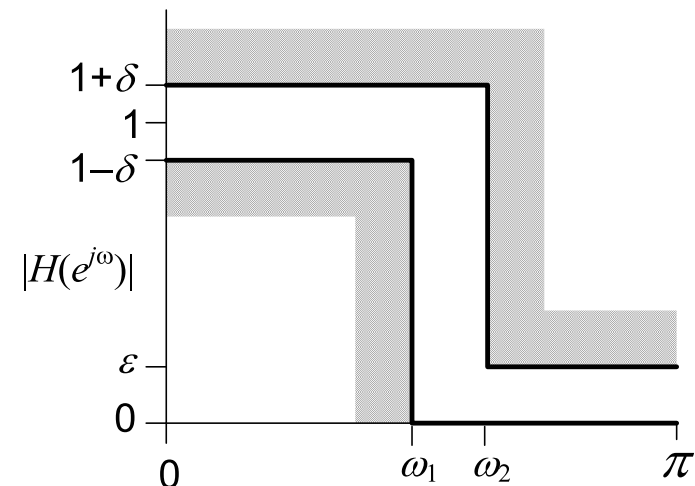
Weighted error: $e(\omega) = w(\omega) (\overline{H}(\omega) - d(\omega))$ where $d(\omega)$ is the target.

Choose $w(\omega)$ to control the error variation with ω .

Example: lowpass filter

$$d(\omega) = \begin{cases} 1 & 0 \leq \omega \leq \omega_1 \\ 0 & \omega_2 \leq \omega \leq \pi \end{cases}$$

$$w(\omega) = \begin{cases} \delta^{-1} & 0 \leq \omega \leq \omega_1 \\ \epsilon^{-1} & \omega_2 \leq \omega \leq \pi \end{cases}$$



$e(\omega) = \pm 1$ when $\overline{H}(\omega)$ lies at the edge of the specification.

Minimax criterion: $h[n] = \arg \min_{h[n]} \max_{\omega} |e(\omega)|$: minimize max error

Alternation Theorem

7: Optimal FIR filters

Optimal Filters

▷ Alternation Theorem

Chebyshev Polynomials

Maximal Error

Locations

Remez Exchange Algorithm

Determine Polynomial

Example Design

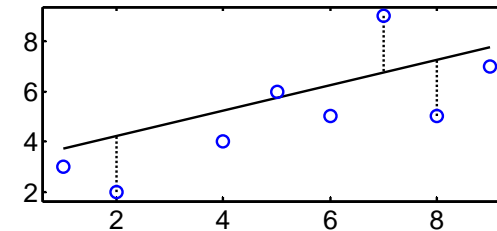
FIR Pros and Cons

Summary

MATLAB routines

Want to find the best fit line: with the smallest maximal error.

Best fit line always attains the maximal error three times with alternate signs



Proof:

Assume the first maximal deviation from the line is negative as shown.

There must be an equally large positive deviation; or else just move the line downwards to reduce the maximal deviation.

This must be followed by another maximal negative deviation; or else you can rotate the line and reduce the deviations.

Alternation Theorem:

A polynomial fit of degree n to a bounded set of points is minimax if and only if it attains its maximal error at $n + 2$ points with alternating signs.

There may be additional maximal error points.

Fitting to a continuous function is the same as to an infinite number of points.

Chebyshev Polynomials

7: Optimal FIR filters

Optimal Filters

Alternation Theorem

► Chebyshev Polynomials

Maximal Error

Locations

Remez Exchange

Algorithm

Determine Polynomial

Example Design

FIR Pros and Cons

Summary

MATLAB routines

$$\overline{H}(\omega) = H(e^{j\omega}) = h[0] + 2 \sum_{n=1}^{\frac{M}{2}} h[n] \cos n\omega$$

But $\cos n\omega = T_n(\cos \omega)$: **Chebyshev polynomial** of 1st kind

$$\cos 2\omega = 2 \cos^2 \omega - 1 = T_2(\cos \omega)$$

$$\cos 3\omega = 4 \cos^3 \omega - 3 \cos \omega = T_3(\cos \omega)$$

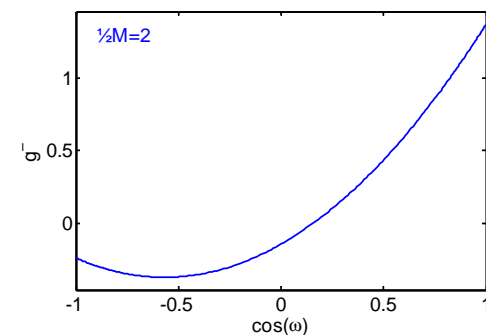
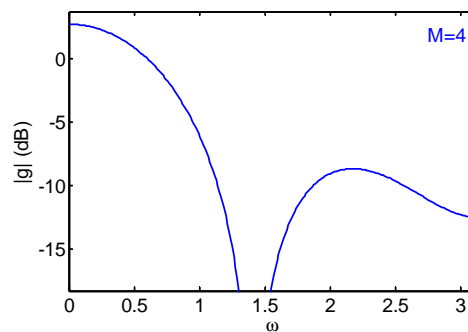
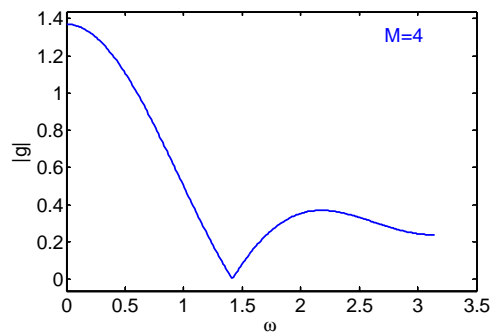
Recurrence Relation:

$$T_{n+1}(x) = 2xT_n(x) - T_{n-1}(x) \text{ with } T_0(x) = 1, T_1(x) = x$$

Proof: $\cos(n\omega + \omega) + \cos(n\omega - \omega) = 2 \cos \omega \cos n\omega$

So $\overline{H}(\omega)$ is an $\frac{M}{2}$ order polynomial in $\cos \omega$: **alternation theorem** applies.

Example: Low-pass filter of length 5 ($M = 4$)



Maximal Error Locations

7: Optimal FIR filters

Optimal Filters

Alternation Theorem

Chebyshev

Polynomials

Maximal Error

Locations

Remez Exchange

Algorithm

Determine Polynomial

Example Design

FIR Pros and Cons

Summary

MATLAB routines

Maximal error locations occur either at band edges or when $\frac{d\bar{H}}{d\omega} = 0$

$$\begin{aligned}\bar{H}(\omega) &= h[0] + 2 \sum_{n=1}^{\frac{M}{2}} h[n] \cos n\omega \\ &= P(\cos \omega)\end{aligned}$$

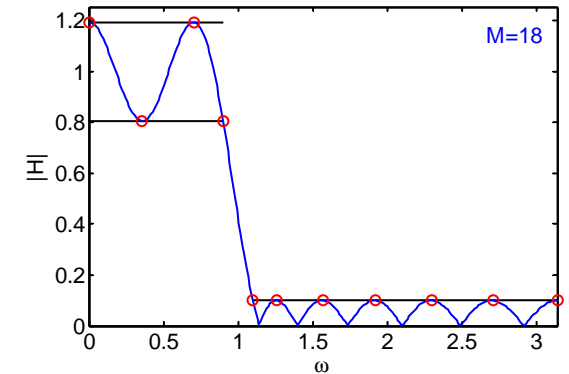
where $P(x)$ is a polynomial of order $\frac{M}{2}$.

$$\begin{aligned}\frac{d\bar{H}}{d\omega} &= -P'(\cos \omega) \sin \omega \\ &= 0 \text{ at } \omega = 0, \pi \text{ and at most } \frac{M}{2} - 1 \text{ zeros of polynomial } P'(x).\end{aligned}$$

\therefore With two bands, we have at most $\frac{M}{2} + 3$ maximal error frequencies. We require $\frac{M}{2} + 2$ of alternating signs for the optimal fit.

Only three possibilities exist (try them all):

- (a) $\omega = 0 + \text{two band edges} + \frac{M}{2} - 1$ zeros of $P'(x)$.
- (b) $\omega = \pi + \text{two band edges} + \frac{M}{2} - 1$ zeros of $P'(x)$.
- (c) $\omega = \{0 \text{ and } \pi\} + \text{two band edges} + \frac{M}{2} - 2$ zeros of $P'(x)$.



Remez Exchange Algorithm

7: Optimal FIR filters

Optimal Filters

Alternation Theorem

Chebyshev

Polynomials

Maximal Error

Locations

Remez Exchange

▷ Algorithm

Determine Polynomial

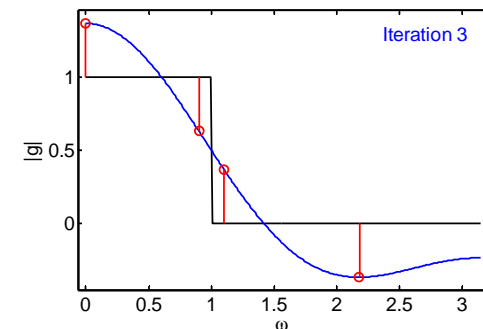
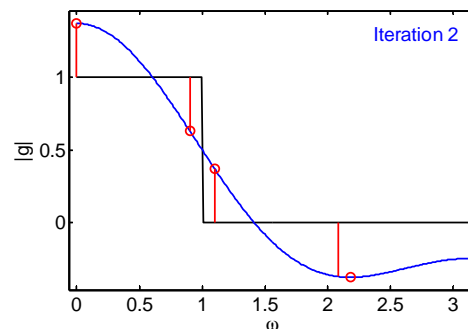
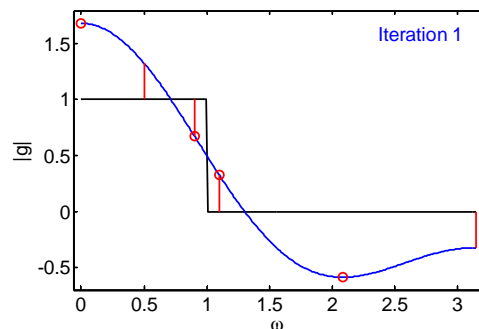
Example Design

FIR Pros and Cons

Summary

MATLAB routines

1. **Guess** the positions of the $\frac{M}{2} + 2$ maximal error frequencies and give alternating signs to the errors (e.g. choose evenly spaced ω).
2. **Determine** the error magnitude, ϵ , and the $\frac{M}{2} + 1$ coefficients of the polynomial that passes through the maximal error locations.
3. **Find the local maxima** of the error function by evaluating $e(\omega) = w(\omega) (\overline{H}(\omega) - d(\omega))$ on a dense set of ω .
4. **Update the maximal error frequencies** to be an alternating subset of the local maxima + band edges + $\{0 \text{ and/or } \pi\}$.
If maximum error is $> \epsilon$, go back to step 2. (typically 15 iterations)
5. **Evaluate** $\overline{H}(\omega)$ on $M + 1$ evenly spaced ω and do an **IDFT** to get $h[n]$.



Determine Polynomial

7: Optimal FIR filters

Optimal Filters

Alternation Theorem

Chebyshev Polynomials

Maximal Error

Locations

Remez Exchange Algorithm

Determine Polynomial

Example Design

FIR Pros and Cons

Summary

MATLAB routines

For each extremal frequency, ω_i for $1 \leq i \leq \frac{M}{2} + 2$

$$d(\omega_i) = \overline{H}(\omega_i) + \frac{(-1)^i \epsilon}{w(\omega_i)} = h[0] + 2 \sum_{n=1}^{\frac{M}{2}} h[n] \cos n\omega_i + \frac{(-1)^i \epsilon}{w(\omega_i)}$$

Method 1: (Computation time $\propto M^3$)

Solve $\frac{M}{2} + 2$ equations in $\frac{M}{2} + 2$ unknowns for $h[n] + \epsilon$.

In step 3, evaluate $\overline{H}(\omega) = h[0] + 2 \sum_{n=1}^{\frac{M}{2}} h[n] \cos n\omega_i$

Method 2: Don't calculate $h[n]$ explicitly (Computation time $\propto M^2$)

Multiply equations by $c_i = \prod_{j \neq i} \frac{1}{\cos \omega_i - \cos \omega_j}$ and add:

$$\sum_{i=1}^{\frac{M}{2}+2} c_i \left(h[0] + 2 \sum_{n=1}^{\frac{M}{2}} h[n] \cos n\omega + \frac{(-1)^i \epsilon}{w(\omega_i)} \right) = \sum_{i=1}^{\frac{M}{2}+2} c_i d(\omega_i)$$

All terms involving $h[n]$ sum to zero leaving

$$\sum_{i=1}^{\frac{M}{2}+2} \frac{(-1)^i c_i}{w(\omega_i)} \epsilon = \sum_{i=1}^{\frac{M}{2}+2} c_i d(\omega_i)$$

Solve for ϵ then calculate the $\overline{H}(\omega_i)$ then use Lagrange interpolation:

$$\overline{H}(\omega) = P(\cos \omega) = \sum_{i=1}^{\frac{M}{2}+2} \overline{H}(\omega_i) \prod_{j \neq i} \frac{\cos \omega - \cos \omega_j}{\cos \omega_i - \cos \omega_j}$$

$(\frac{M}{2} + 1)$ -polynomial going through all the $\overline{H}(\omega_i)$ [actually order $\frac{M}{2}$]

Example Design

7: Optimal FIR filters

Optimal Filters

Alternation Theorem

Chebyshev

Polynomials

Maximal Error

Locations

Remez Exchange

Algorithm

Determine Polynomial

▷ Example Design

FIR Pros and Cons

Summary

MATLAB routines

Filter Specifications:

Bandpass $\omega = [0.5, 1]$, Transition widths: $\Delta\omega = 0.2$

Stopband Attenuation: -25 dB and -15 dB

Passband Ripple: ± 0.3 dB

Determine **gain tolerances** for each band:

$$-25 \text{ dB} = 0.056, -0.3 \text{ dB} = 1 - 0.034, -15 \text{ dB} = 0.178$$

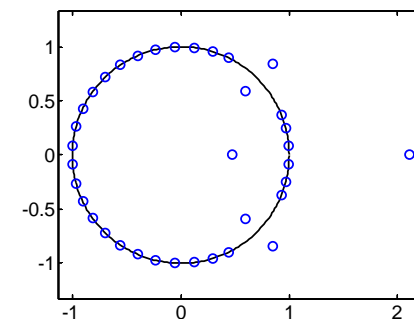
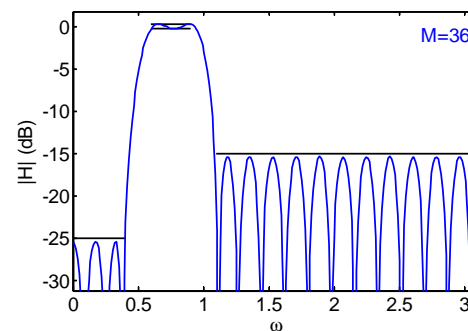
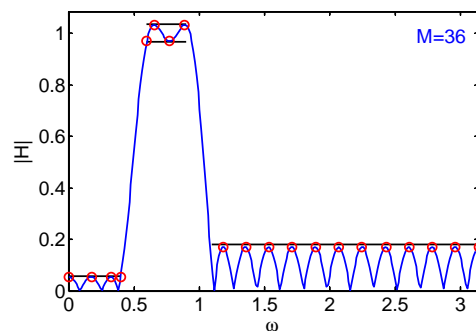
Predicted order: $M = 36$

$\frac{M}{2} + 2$ extremal frequencies are distributed between the bands

Filter meets specs ☺; clearer on a decibel scale

Most zeros are on the unit circle + three **reciprocal pairs**

Reciprocal pairs give a linear phase shift



FIR Pros and Cons

7: Optimal FIR filters

Optimal Filters

Alternation Theorem

Chebyshev

Polynomials

Maximal Error

Locations

Remez Exchange

Algorithm

Determine Polynomial

Example Design

▷ FIR Pros and Cons

Summary

MATLAB routines

- Can have **linear phase**
 - no envelope distortion, all frequencies have the same delay ☺
 - symmetric or antisymmetric: $h[n] = h[-n]\forall n$ or $-h[-n]\forall n$
 - antisymmetric filters have $H(e^{j0}) = H(e^{j\pi}) = 0$
 - symmetry means you only need $\frac{M}{2} + 1$ multiplications to implement the filter.
- Always **stable** ☺
- **Low coefficient sensitivity** ☺
- **Optimal design method** fast and robust ☺
- Normally needs **higher order** than an IIR filter ☹
 - Filter order $M \approx \frac{\text{dB}_{\text{atten}}}{3.5\Delta\omega}$ where $\Delta\omega$ is the most rapid transition
 - Filtering complexity $\propto M \times f_s \approx \frac{\text{dB}_{\text{atten}}}{3.5\Delta\omega} f_s = \frac{\text{dB}_{\text{atten}}}{3.5\Delta\Omega} f_s^2 \propto f_s^2$ for a given specification in unscaled Ω units.

Summary

7: Optimal FIR filters

Optimal Filters

Alternation Theorem

Chebyshev

Polynomials

Maximal Error

Locations

Remez Exchange

Algorithm

Determine Polynomial

Example Design

FIR Pros and Cons

▷ Summary

MATLAB routines

Optimal Filters: minimax error criterion

- use weight function, $w(\omega)$, to allow different errors in different frequency bands
- symmetric filter has zeros on unit circle or in reciprocal pairs
- Response of symmetric filter is a polynomial in $\cos \omega$
- Alternation Theorem: $\frac{M}{2} + 2$ maximal errors with alternating signs

Remez Exchange Algorithm (also known as Parks-McLellan Algorithm)

- multiple constant-gain bands separated by transition regions
- very robust, works for filters with $M > 1000$
- Efficient: computation $\propto M^2$
- can go mad in the transition regions

Modified version works on arbitrary gain function

- Does not always converge

For further details see Mitra: 10.

MATLAB routines

7: Optimal FIR filters

Optimal Filters

Alternation Theorem

Chebyshev

Polynomials

Maximal Error

Locations

Remez Exchange

Algorithm

Determine Polynomial

Example Design

FIR Pros and Cons

Summary

▷ MATLAB routines

firpm	optimal FIR filter design
firpmord	estimate require order for firpm
cfirpm	arbitrary-response filter design
remez	[obsolete] optimal FIR filter design

8: IIR Filter

▷ Transformations

**Continuous Time
Filters**

Bilinear Mapping

**Continuous Time
Filters**

**Mapping Poles and
Zeros**

**Spectral
Transformations**

Constantinides

Transformations

Impulse Invariance

Summary

MATLAB routines

8: IIR Filter Transformations

Continuous Time Filters

Classical continuous-time filters: optimize passband ripple v stopband ripple v transition width

There are explicit formulae for pole/zero positions.

Butterworth: $\tilde{G}^2(\omega) = \left| \tilde{H}(j\omega) \right|^2 = \frac{1}{1+\omega^{2N}}$

- Monotonic $\forall \omega$
- $\tilde{G}(\omega) = 1 - \frac{1}{2}\omega^{2N} + \frac{3}{8}\omega^{4N} + \dots$
“Maximally flat”: $2N - 1$ derivatives are zero

Chebyshev: $\tilde{G}^2(\omega) = \frac{1}{1+\epsilon^2 T_N^2(\omega)}$

where polynomial $T_N(\cos x) = \cos Nx$

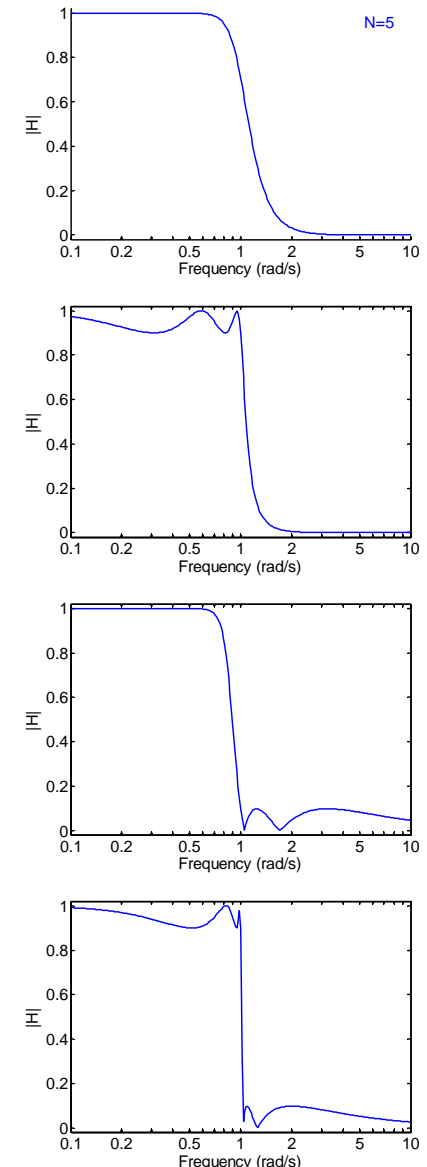
- passband equiripple + very flat at ∞

Inverse Chebyshev: $\tilde{G}^2(\omega) = \frac{1}{1+(\epsilon^2 T_N^2(\omega^{-1}))^{-1}}$

- stopband equiripple + very flat at 0

Elliptic: [no nice formula]

- Very steep + equiripple in pass and stop bands



Bilinear Mapping

Change variable: $z = \frac{1+s}{1-s} \Leftrightarrow s = \frac{z-1}{z+1}$: a one-to-one invertible mapping

- \Re axis (s) \leftrightarrow \Re axis (z)

- \Im axis (s) \leftrightarrow Unit circle (z)

Proof: $z = e^{j\omega} \Leftrightarrow s = \frac{e^{j\omega} - 1}{e^{j\omega} + 1} = \frac{e^{j\frac{\omega}{2}} - e^{-j\frac{\omega}{2}}}{e^{j\frac{\omega}{2}} + e^{-j\frac{\omega}{2}}} = j \tan \frac{\omega}{2} = j\Omega$

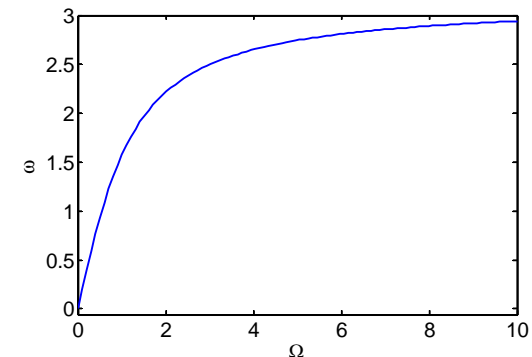
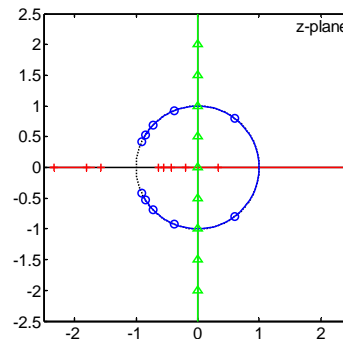
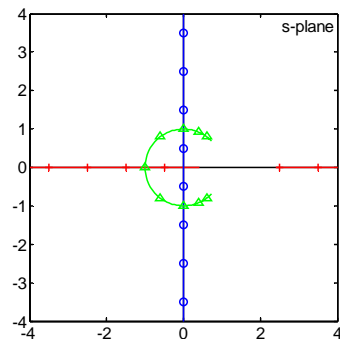
- Left half plane(s) \leftrightarrow inside of unit circle (z)

Proof: $s = x + jy \Leftrightarrow |z|^2 = \frac{|(1+x)+jy|^2}{|(1-x)+jy|^2}$

$$= \frac{1+2x+x^2+y^2}{1-2x+x^2+y^2} = 1 + \frac{4x}{|(1-x)+jy|^2}$$

$$x < 0 \Leftrightarrow |z| < 1$$

- Unit circle (s) \leftrightarrow \Im axis (z)



Continuous Time Filters

- 8: IIR Filter Transformations
- Continuous Time Filters
- Bilinear Mapping
 - Continuous Time Filters
- Mapping Poles and Zeros
- Spectral Transformations
- Constantinides Transformations
- Impulse Invariance
- Summary
- MATLAB routines

Take $\tilde{H}(s) = \frac{1}{s^2 + 0.2s + 4}$

Substitute: $s = \frac{z-1}{z+1}$

[extra zeros at $z = -1$]

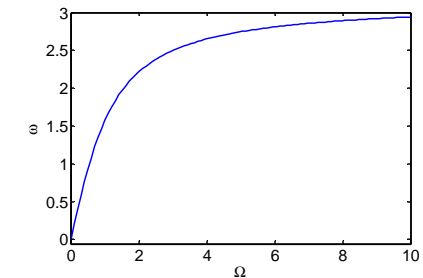
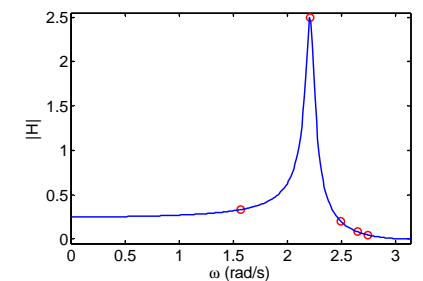
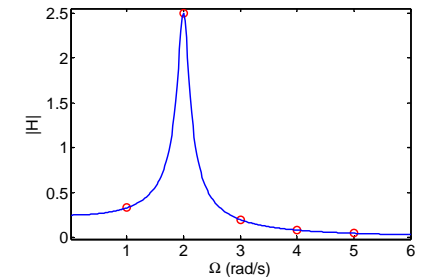
$$\begin{aligned}
 H(z) &= \frac{1}{\left(\frac{z-1}{z+1}\right)^2 + 0.2 \frac{z-1}{z+1} + 4} \\
 &= \frac{(z+1)^2}{(z-1)^2 + 0.2(z-1)(z+1) + 4(z+1)^2} \\
 &= \frac{z^2 + 2z + 1}{5.2z^2 + 6z + 4.8} = 0.19 \frac{1 + 2z^{-1} + z^{-2}}{1 + 1.15z^{-1} + 0.92z^{-2}}
 \end{aligned}$$

Frequency response is identical (both magnitude and phase) but with a distorted frequency axis:

Frequency mapping: $\omega = 2 \tan^{-1} \Omega$

$$\begin{aligned}
 \Omega &= [1 \quad 2 \quad 3 \quad 4 \quad 5] \\
 \rightarrow \omega &= [1.6 \quad 2.2 \quad 2.5 \quad 2.65 \quad 2.75]
 \end{aligned}$$

Scaling s : Substitute $s = \left(\frac{\Omega_0}{\tan \frac{1}{2}\omega_0}\right) \frac{z-1}{z+1}$ to map $\Omega_0 \rightarrow \omega_0$



Mapping Poles and Zeros

Alternative method: $\tilde{H}(s) = \frac{1}{s^2 + 0.2s + 4}$

Find the poles and zeros: $p_s = -0.1 \pm 2j$

Map using $z = \frac{1+s}{1-s} \Rightarrow p_z = -0.58 \pm 0.77j$

After the transformation we will always end up with the same number of poles and zeros:

Add extra poles or zeros at $z = -1$

$$H(z) = g \times \frac{(1+z^{-1})^2}{(1+(0.58-0.77j)z^{-1})(1+(0.58+0.77j)z^{-1})}$$

$$= g \times \frac{1+2z^{-1}+z^{-2}}{1+1.15z^{-1}+0.92z^{-2}}$$

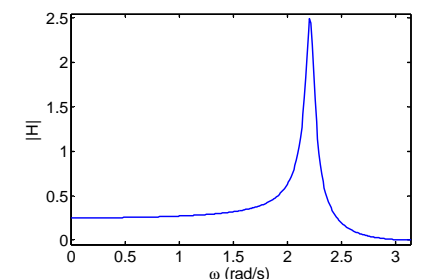
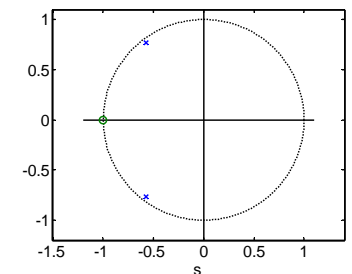
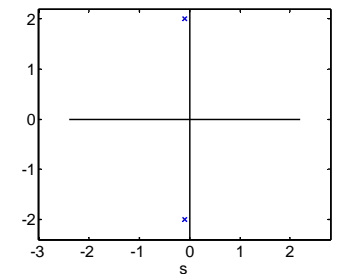
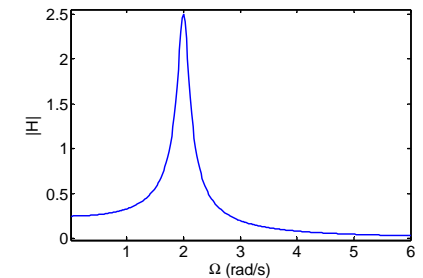
Choose overall scale factor, g , to give the same gain at any convenient pair of mapped frequencies:

$$\text{At } \Omega = 0 \Rightarrow s = 0 \Rightarrow |\tilde{H}(s)| = 0.25$$

$$\text{At } \omega = 2 \tan^{-1} \Omega = 0 \Rightarrow z = e^{j\omega} = 1$$

$$\Rightarrow |H(z)| = g \times \frac{4}{3.08} = 0.25 \Rightarrow g = 0.19$$

$$H(z) = 0.19 \frac{1+2z^{-1}+z^{-2}}{1+1.15z^{-1}+0.92z^{-2}}$$



Spectral Transformations

- 8: IIR Filter Transformations
- Continuous Time Filters
- Bilinear Mapping
- Continuous Time Filters
- Mapping Poles and Zeros
- Spectral
- ▷ Transformations
- Constantinides Transformations
- Impulse Invariance
- Summary
- MATLAB routines

We can transform the z -plane to change the cutoff frequency by substituting

$$z = \frac{\hat{z} - \lambda}{1 - \lambda \hat{z}} \Leftrightarrow \hat{z} = \frac{z + \lambda}{1 + \lambda z}$$

Frequency Mapping:

If $z = e^{j\omega}$, then $\hat{z} = z \frac{1 + \lambda z^{-1}}{1 + \lambda z}$ has modulus 1 since the numerator and denominator are complex conjugates.

Hence the **unit circle is preserved**.

$$\Rightarrow e^{j\hat{\omega}} = \frac{e^{j\omega} + \lambda}{1 + \lambda e^{j\omega}}$$

Some algebra gives: $\tan \frac{\omega}{2} = \left(\frac{1 + \lambda}{1 - \lambda} \right) \tan \frac{\hat{\omega}}{2}$

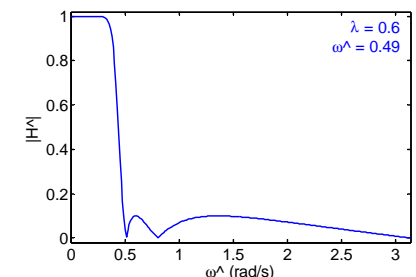
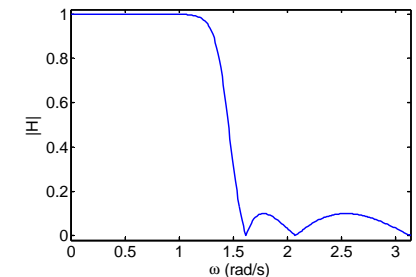
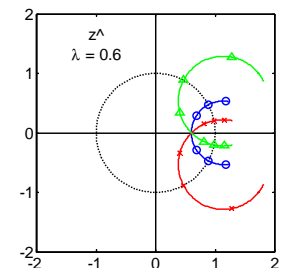
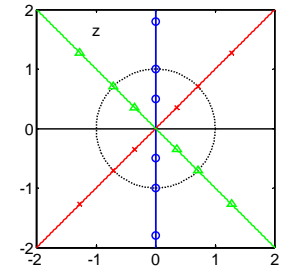
Equivalent to:

$$z \longrightarrow s = \frac{z-1}{z+1} \longrightarrow \hat{s} = \frac{1-\lambda}{1+\lambda} s \longrightarrow \hat{z} = \frac{1+\hat{s}}{1-\hat{s}}$$

Lowpass Filter example:

Inverse Chebyshev

$$\omega_0 = \frac{\pi}{2} = 1.57 \xrightarrow{\lambda=0.6} \hat{\omega}_0 = 0.49$$

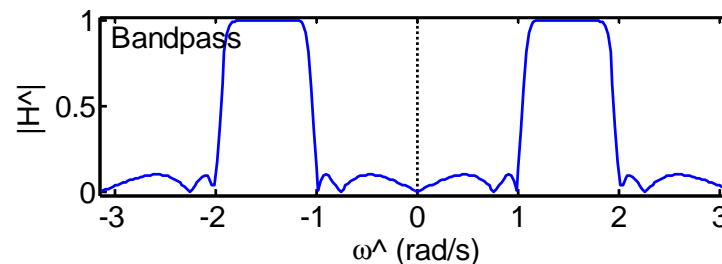
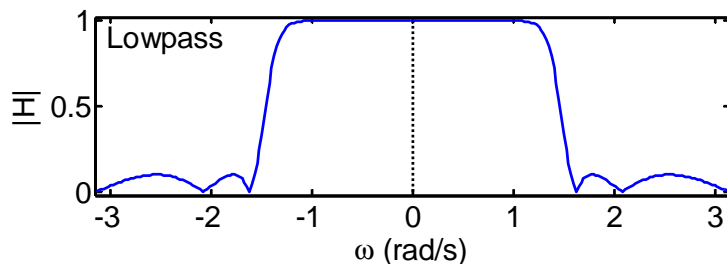


Constantinides Transformations

Transform a lowpass filter with cutoff frequency ω_0 to:

Target	Substitute	Parameters
Lowpass $\hat{\omega} < \hat{\omega}_1$	$z^{-1} = \frac{\hat{z}^{-1} - \lambda}{1 - \lambda \hat{z}^{-1}}$	$\lambda = \frac{\sin\left(\frac{\omega_0 - \hat{\omega}_1}{2}\right)}{\sin\left(\frac{\omega_0 + \hat{\omega}_1}{2}\right)}$
Highpass $\hat{\omega} > \hat{\omega}_1$	$z^{-1} = -\frac{\hat{z}^{-1} + \lambda}{1 + \lambda \hat{z}^{-1}}$	$\lambda = \frac{\cos\left(\frac{\omega_0 + \hat{\omega}_1}{2}\right)}{\cos\left(\frac{\omega_0 - \hat{\omega}_1}{2}\right)}$
Bandpass $\hat{\omega}_1 < \hat{\omega} < \hat{\omega}_2$	$z^{-1} = -\frac{(\rho-1) - 2\lambda\rho\hat{z}^{-1} + (\rho+1)\hat{z}^{-2}}{(\rho+1) - 2\lambda\rho\hat{z}^{-1} + (\rho-1)\hat{z}^{-2}}$	$\lambda = \frac{\cos\left(\frac{\hat{\omega}_2 + \hat{\omega}_1}{2}\right)}{\cos\left(\frac{\hat{\omega}_2 - \hat{\omega}_1}{2}\right)}$ $\rho = \cot\left(\frac{\hat{\omega}_2 - \hat{\omega}_1}{2}\right) \tan\left(\frac{\omega_0}{2}\right)$
Bandstop $\hat{\omega}_1 \nless \hat{\omega} \nless \hat{\omega}_2$	$z^{-1} = \frac{(1-\rho) - 2\lambda\hat{z}^{-1} + (\rho+1)\hat{z}^{-2}}{(\rho+1) - 2\lambda\hat{z}^{-1} + (1-\rho)\hat{z}^{-2}}$	$\lambda = \frac{\cos\left(\frac{\hat{\omega}_2 + \hat{\omega}_1}{2}\right)}{\cos\left(\frac{\hat{\omega}_2 - \hat{\omega}_1}{2}\right)}$ $\rho = \tan\left(\frac{\hat{\omega}_2 - \hat{\omega}_1}{2}\right) \tan\left(\frac{\omega_0}{2}\right)$

Bandpass and bandstop transformations are quadratic and so will double the order:



Impulse Invariance

Bilinear transform works well for a lowpass filter but the non-linear compression of the frequency distorts any other response.

Alternative method:

Express $\tilde{H}(s)$ as a sum of partial fractions $H(s) = \sum_{i=1}^N \frac{g_i}{s - \tilde{p}_i}$

Impulse response is $\tilde{h}(t) = u(t) \times \sum_{i=1}^N g_i e^{\tilde{p}_i t}$

Digital filter $H(z) = \sum_{i=1}^N \frac{g_i}{1 - e^{\tilde{p}_i T} z^{-1}}$ has identical impulse response

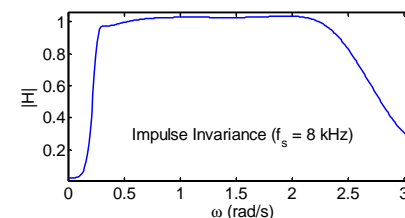
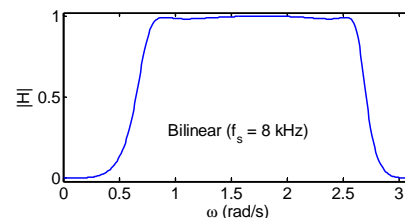
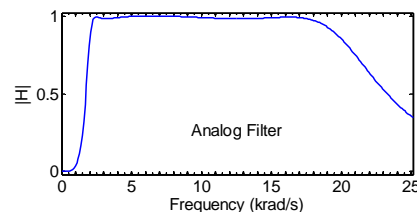
Poles of $H(z)$ are $p_i = e^{\tilde{p}_i T}$ (where $T = \frac{1}{f_s}$ is sampling period)

Zeros do not map in a simple way

Properties:

- ☺ Impulse response correct
- ☺ No distortion of frequency axis
- ☹ Frequency response is aliased

Example: Standard telephone filter - 300 to 3400 Hz bandpass



Summary

8: IIR Filter Transformations
Continuous Time Filters
Bilinear Mapping
Continuous Time Filters
Mapping Poles and Zeros
Spectral Transformations
Constantinides Transformations
Impulse Invariance
▷ Summary
MATLAB routines

- **Classical filters** have optimal tradeoffs in continuous time domain
 - Order \leftrightarrow transition width \leftrightarrow pass ripple \leftrightarrow stop ripple
 - Monotonic passband and/or stopband
- **Bilinear mapping**
 - Exact preservation of frequency response
 - non-linear frequency axis distortion
 - can scale s to map $\Omega_0 \rightarrow \omega_0$ for one specific frequency
- **Spectral transformations**
 - lowpass \rightarrow lowpass, highpass, bandpass or bandstop
 - bandpass and bandstop double the filter order
- **Impulse Invariance**
 - Aliasing distortion of frequency response
 - preserves frequency axis and impulse response

For further details see Mitra: 9.

MATLAB routines

8: IIR Filter
Transformations
Continuous Time
Filters
Bilinear Mapping
Continuous Time
Filters
Mapping Poles and
Zeros
Spectral
Transformations
Constantinides
Transformations
Impulse Invariance
Summary
▷ MATLAB routines

bilinear	Bilinear mapping
impinvar	Impulse invariance
butter butterord	Analog or digital Butterworth filter
cheby1 cheby1ord	Analog or digital Chebyshev filter
cheby2 cheby2ord	Analog or digital Inverse Chebyshev filter
ellip ellipord	Analog or digital Elliptic filter

▷ **9: Optimal IIR
Design**

Error choices

Linear Least Squares

Frequency Sampling

Iterative Solution

Newton-Raphson

Magnitude-only

Specification

Hilbert Relations

Phase Relation

Summary

MATLAB routines

9: Optimal IIR Design

Error choices

9: Optimal IIR Design

▷ Error choices

Linear Least Squares

Frequency Sampling

Iterative Solution

Newton-Raphson

Magnitude-only

Specification

Hilbert Relations

Phase Relation

Summary

MATLAB routines

We want to find a filter $H(e^{j\omega}) = \frac{B(e^{j\omega})}{A(e^{j\omega})}$ that approximates a target response $D(\omega)$. Assume A is order N and B is order M .

Two possible error measures:

Solution Error: $E_S(\omega) = W_S(\omega) \left(\frac{B(e^{j\omega})}{A(e^{j\omega})} - D(\omega) \right)$

Equation Error: $E_E(\omega) = W_E(\omega) (B(e^{j\omega}) - D(\omega)A(e^{j\omega}))$

We may know $D(\omega)$ completely or else only $|D(\omega)|$

We minimize $\int_{-\pi}^{\pi} |E_*(\omega)|^p d\omega$
where $p = 2$ (least squares) or ∞ (minimax).

Weight functions $W_*(\omega)$ are chosen to control relative errors at different frequencies. $W_S(\omega) = |D(\omega)|^{-1}$ gives constant dB error.

We actually want to minimize E_S but E_E is easier because it gives rise to linear equations.

However if $W_E(\omega) = \frac{W_S(\omega)}{|A(e^{j\omega})|}$, then $|E_E(\omega)| = |E_S(\omega)|$

Linear Least Squares

9: Optimal IIR Design

Error choices

Linear Least Squares

Frequency Sampling

Iterative Solution

Newton-Raphson

Magnitude-only

Specification

Hilbert Relations

Phase Relation

Summary

MATLAB routines

Overdetermined set of equations $\mathbf{Ax} = \mathbf{b}$ (#equations > #unknowns)

We want to minimize $\|\mathbf{e}\|^2$ where $\mathbf{e} = \mathbf{Ax} - \mathbf{b}$

$$\|\mathbf{e}\|^2 = \mathbf{e}^T \mathbf{e} = (\mathbf{x}^T \mathbf{A}^T - \mathbf{b}^T) (\mathbf{Ax} - \mathbf{b})$$

Differentiate:

$$d(\mathbf{e}^T \mathbf{e}) = d\mathbf{x}^T \mathbf{A}^T (\mathbf{Ax} - \mathbf{b}) + (\mathbf{x}^T \mathbf{A}^T - \mathbf{b}^T) \mathbf{A} d\mathbf{x}$$

[since $d(\mathbf{uv}) = d\mathbf{u} \mathbf{v} + \mathbf{u} d\mathbf{v}$]

[since $\mathbf{u}^T \mathbf{v} = \mathbf{v}^T \mathbf{u}$]

$$= 2d\mathbf{x}^T \mathbf{A}^T (\mathbf{Ax} - \mathbf{b})$$

$$= 2d\mathbf{x}^T (\mathbf{A}^T \mathbf{Ax} - \mathbf{A}^T \mathbf{b})$$

This is zero for any $d\mathbf{x}$ iff $\mathbf{A}^T \mathbf{Ax} = \mathbf{A}^T \mathbf{b}$

Thus $\|\mathbf{e}\|^2$ is minimized if $\mathbf{x} = (\mathbf{A}^T \mathbf{A})^{-1} \mathbf{A}^T \mathbf{b}$

These are the **Normal Equations** ("Normal" because $\mathbf{A}^T \mathbf{e} = 0$)

The **pseudoinverse** $\mathbf{x} = \mathbf{A}^+ \mathbf{b}$ works even if $\mathbf{A}^T \mathbf{A}$ is singular and finds the \mathbf{x} with minimum $\|\mathbf{x}\|^2$ that minimizes $\|\mathbf{e}\|^2$.

This is a very widely used technique.

Frequency Sampling

9: Optimal IIR Design

Error choices
Linear Least Squares

Frequency
▷ Sampling

Iterative Solution

Newton-Raphson

Magnitude-only
Specification

Hilbert Relations

Phase Relation

Summary

MATLAB routines

Want to approximate: $W(\omega) (B(e^{j\omega}) - D(\omega)A(e^{j\omega})) = 0$

$$W(\omega) \left(\sum_{m=0}^M b[m]e^{-jm\omega} - D(\omega) \left(1 + \sum_{n=1}^N a[n]e^{-jn\omega} \right) \right)$$

As a matrix equation:

$$\begin{pmatrix} \mathbf{u}(\omega)^T & \mathbf{v}(\omega)^T \end{pmatrix} \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \end{pmatrix} = W(\omega)D(\omega)$$

$$\text{where } \mathbf{u}(\omega)^T = -W(\omega)D(\omega) \begin{bmatrix} e^{-j\omega} & e^{-j2\omega} & \dots & e^{-jN\omega} \end{bmatrix}$$

$$\mathbf{v}(\omega)^T = W(\omega) \begin{bmatrix} 1 & e^{-j\omega} & e^{-j2\omega} & \dots & e^{-jM\omega} \end{bmatrix}$$

Choose K values of ω , $\{ \omega_1 \dots \omega_K \}$ where $K \geq \frac{M+N+1}{2}$

$$\begin{pmatrix} \mathbf{U}^T & \mathbf{V}^T \end{pmatrix} \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \end{pmatrix} = \mathbf{d}^T$$

$$\text{where } \mathbf{U} = \begin{bmatrix} \mathbf{u}(\omega_1) & \dots & \mathbf{u}(\omega_K) \end{bmatrix},$$

$$\mathbf{V} = \begin{bmatrix} \mathbf{v}(\omega_1) & \dots & \mathbf{v}(\omega_K) \end{bmatrix},$$

$$\mathbf{d} = \begin{bmatrix} W(\omega_1)D(\omega_1) & \dots & W(\omega_K)D(\omega_K) \end{bmatrix}$$

We want to force \mathbf{a} and \mathbf{b} to be real; find least squares solution to

$$\begin{pmatrix} \Re(\mathbf{U}^T) & \Re(\mathbf{V}^T) \\ \Im(\mathbf{U}^T) & \Im(\mathbf{V}^T) \end{pmatrix} \begin{pmatrix} \mathbf{a} \\ \mathbf{b} \end{pmatrix} = \begin{pmatrix} \Re(\mathbf{d}^T) \\ \Im(\mathbf{d}^T) \end{pmatrix}$$

Iterative Solution

9: Optimal IIR Design

Error choices

Linear Least Squares

Frequency Sampling

▷ Iterative Solution

Newton-Raphson

Magnitude-only

Specification

Hilbert Relations

Phase Relation

Summary

MATLAB routines

Least squares solution minimizes the E_E rather than E_S .

However $E_E = E_S$ if $W_E(\omega) = \frac{W_S(\omega)}{A(e^{j\omega})}$.

We can use an iterative solution technique:

- 1 Select K frequencies $\{\omega_k\}$ (e.g. uniformly spaced)
- 2 Initialize $W_E(\omega_k) = W_S(\omega_k)$
- 3 Find least squares solution to
$$W_E(\omega_k) (B(e^{j\omega_k}) - D(\omega_k)A(e^{j\omega_k})) = 0 \forall k$$
- 4 Force $A(z)$ to be stable
Replace pole p_i by $(p_i^*)^{-1}$ whenever $|p_i| \geq 1$
- 5 Update weights: $W_E(\omega_k) = \frac{W_S(\omega_k)}{A(e^{j\omega_k})}$
- 6 Return to step 3 until convergence

Newton-Raphson

9: Optimal IIR Design

Error choices
Linear Least Squares
Frequency Sampling
Iterative Solution
▷ Newton-Raphson
Magnitude-only
Specification
Hilbert Relations
Phase Relation
Summary
MATLAB routines

Newton: To solve $f(x) = 0$ given an initial guess x_0 , we write

$$f(x) \approx f(x_0) + (x - x_0)f'(x_0) \Rightarrow x = x_0 - \frac{f(x_0)}{f'(x_0)}$$

Converges very rapidly once x_0 is close to the solution

So for each ω_k , we can write (omitting the ω and $e^{j\omega}$ arguments)

$$\begin{aligned} E_S &\approx W_S \left(\frac{B_0}{A_0} - D \right) + \frac{W_S}{A_0} (B - B_0) - \frac{W_S B_0}{A_0^2} (A - A_0) \\ &= \frac{W_S}{A_0} \left(-\frac{B_0}{A_0} (A - 1) + B - A_0 D - \frac{B_0}{A_0} + B_0 \right) \end{aligned}$$

From which we get a linear equation for each ω_k :

$$\left(\begin{array}{cc} \frac{B_0}{DA_0} \mathbf{u}^T & \mathbf{v}^T \end{array} \right) \left(\begin{array}{c} \mathbf{a} \\ \mathbf{b} \end{array} \right) = W \left(A_0 D + \frac{B_0}{A_0} - B_0 \right)$$

where $W = \frac{W_S}{A_0}$ and, as before, $u_n(\omega) = -W(\omega)D(\omega)e^{-jn\omega}$

for $n \in 1 : N$ and $v_m(\omega) = W(\omega)e^{-jm\omega}$ for $m \in 0 : M$.

At each iteration, calculate $A_0(e^{j\omega_k})$ and $B_0(e^{j\omega_k})$ based on \mathbf{a} and \mathbf{b} from the previous iteration.

Then use linear least squares to minimize the linearized E_S using the above equation replicated for each of the ω_k .

Magnitude-only Specification

9: Optimal IIR Design

Error choices

Linear Least Squares

Frequency Sampling

Iterative Solution

Newton-Raphson

▷ Magnitude-only

Specification

Hilbert Relations

Phase Relation

Summary

MATLAB routines

If the filter specification only dictates the target magnitude: $|D(\omega)|$, we need to select the target phase.

Solution:

Make an initial guess of the phase and then at each iteration update $\angle D(\omega) = \angle \frac{B(e^{j\omega})}{A(e^{j\omega})}$.

Initial Guess:

If $H(e^{j\omega})$ is causal and minimum phase then the magnitude and phase are not independent:

$$\begin{aligned}\angle H(e^{j\omega}) &= -\ln |H(e^{j\omega})| \circledast \cot \frac{\omega}{2} \\ \ln |H(e^{j\omega})| &= \ln |H(\infty)| + \angle H(e^{j\omega}) \circledast \cot \frac{\omega}{2}\end{aligned}$$

where $\cot x$ is taken to be zero for $-\epsilon < x < \epsilon$ for some small value of ϵ and we take the limit as $\epsilon \rightarrow 0$.

Hilbert Relations

9: Optimal IIR Design

Error choices
Linear Least Squares
Frequency Sampling
Iterative Solution
Newton-Raphson
Magnitude-only
Specification
▷ Hilbert Relations
Phase Relation
Summary
MATLAB routines

We define $t[n] = u[n-1] - u[1-n]$

$$T(z) = \frac{z^{-1}}{1-z^{-1}} - \frac{z}{1-z} = \frac{1+z^{-1}}{1-z^{-1}}$$

$$\begin{aligned} T(e^{j\omega}) &= \frac{1+e^{-j\omega}}{1-e^{-j\omega}} = \frac{e^{j\frac{\omega}{2}} + e^{-j\frac{\omega}{2}}}{e^{j\frac{\omega}{2}} - e^{-j\frac{\omega}{2}}} \\ &= \frac{2 \cos \frac{\omega}{2}}{2j \sin \frac{\omega}{2}} = -j \cot \frac{\omega}{2} \end{aligned}$$

Even/odd parts: $h_e[n] = \frac{1}{2} (h[n] + h[-n])$

$$h_o[n] = \frac{1}{2} (h[n] - h[-n])$$

$$\text{so } \Re(H(e^{j\omega})) = H_e(e^{j\omega})$$

$$\Im(H(e^{j\omega})) = -jH_o(e^{j\omega})$$

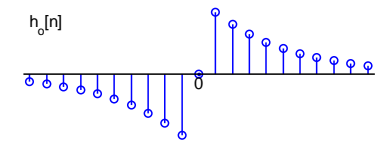
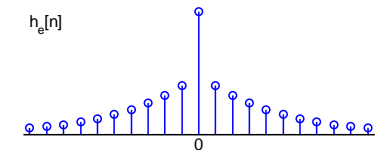
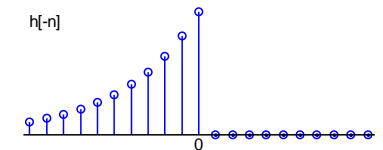
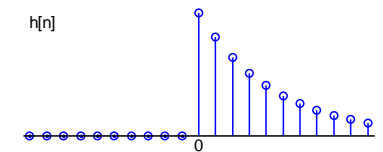
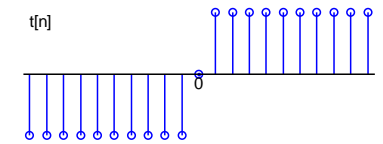
If $h[n]$ is causal: $h_o[n] = h_e[n]t[n]$

$$h_e[n] = h[0]\delta[n] + h_o[n]t[n]$$

Hence, for causal $h[n]$:

$$\begin{aligned} \Im(H(e^{j\omega})) &= -j(\Re(H(e^{j\omega})) \circledast -j \cot \frac{\omega}{2}) \\ &= -\Re(H(e^{j\omega})) \circledast \cot \frac{\omega}{2} \end{aligned}$$

$$\begin{aligned} \Re(H(e^{j\omega})) &= H(\infty) + j\Im(H(e^{j\omega})) \circledast -j \cot \frac{\omega}{2} \\ &= H(\infty) + \Im(H(e^{j\omega})) \circledast \cot \frac{\omega}{2} \end{aligned}$$



Phase Relation

9: Optimal IIR Design

Error choices

Linear Least Squares

Frequency Sampling

Iterative Solution

Newton-Raphson

Magnitude-only Specification

Hilbert Relations

▷ Phase Relation

Summary

MATLAB routines

$$\text{Given } H(z) = g \frac{\prod (1 - q_m z^{-1})}{\prod (1 - p_n z^{-1})}$$

$$\begin{aligned} \ln H(z) &= \ln(g) + \sum \ln(1 - q_m z^{-1}) \\ &\quad - \sum \ln(1 - p_n z^{-1}) \\ &= \ln |H(z)| + j \angle H(z) \end{aligned}$$

Taylor Series:

$$\ln(1 - az^{-1}) = -az^{-1} - \frac{a^2}{2}z^{-2} - \frac{a^3}{3}z^{-3} - \dots$$

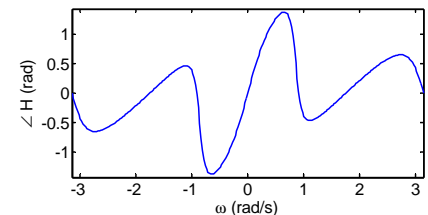
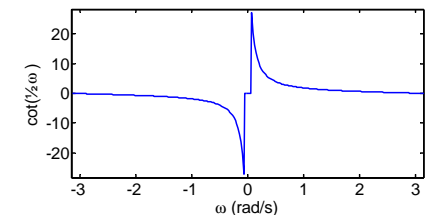
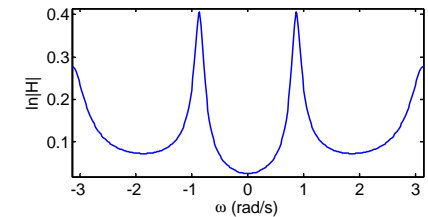
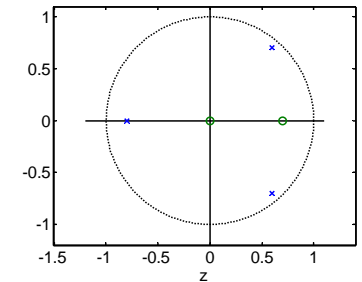
causal and stable provided $|a| < 1$

So, if $H(z)$ is **minimum phase** (all p_n and q_m inside unit circle) then $\ln H(z)$ is the z-transform of a stable causal sequence and:

$$\begin{aligned} \angle H(e^{j\omega}) &= -\ln |H(e^{j\omega})| \circledast \cot \frac{\omega}{2} \\ \ln |H(e^{j\omega})| &= \ln |g| + \angle H(e^{j\omega}) \circledast \cot \frac{\omega}{2} \end{aligned}$$

Example: $H(z) = \frac{10 - 7z^{-1}}{100 - 40z^{-1} - 11z^{-2} + 68z^{-3}}$

Note **symmetric dead band** in $\cot \frac{\omega}{2}$ for $|\omega| < \epsilon$



Summary

9: Optimal IIR Design

Error choices

Linear Least Squares

Frequency Sampling

Iterative Solution

Newton-Raphson

Magnitude-only Specification

Hilbert Relations

Phase Relation

▷ Summary

MATLAB routines

- Want solution error E_S but E_E gives linear equations
 - use $W(\omega)$ to weight errors
- Linear least squares: solution to overdetermined $\mathbf{Ax} = \mathbf{b}$
- Closed form solution: least squares E_E at $\{\omega_k\}$
 - use $W(\omega) = \frac{W_S(\omega)}{A(e^{j\omega})}$ to approximate E_S
 - use Taylor series to approximate E_S better
- Hilbert relations
 - relate $\Re(H(e^{j\omega}))$ and $\Im(H(e^{j\omega}))$ for causal stable sequences
 - relate $\ln|H(e^{j\omega})|$ and $\angle H(e^{j\omega})$ for causal stable minimum phase sequences

For further details see Mitra: 9.

MATLAB routines

9: Optimal IIR Design

Error choices

Linear Least Squares

Frequency Sampling

Iterative Solution

Newton-Raphson

Magnitude-only

Specification

Hilbert Relations

Phase Relation

Summary

▷ MATLAB routines

invfreqz

IIR design for complex response

▷ **10: Digital Filter Structures**

Direct Forms
Transposition
State Space
State Space Transfer Function
Precision Issues
Coefficient Sensitivity
Cascaded Biquads
Pole-zero Pairing/Ordering
Linear Phase
Hardware Implementation
Allpass Filters
Lattice Stage
Allpass Lattice
Lattice Filter
Lattice Example
Summary
MATLAB routines

10: Digital Filter Structures

Direct Forms

10: Digital Filter Structures

▷ Direct Forms

Transposition

State Space

State Space Transfer Function

Precision Issues

Coefficient Sensitivity

Cascaded Biquads

Pole-zero

Pairing/Ordering

Linear Phase

Hardware

Implementation

Allpass Filters

Lattice Stage

Allpass Lattice

Lattice Filter

Lattice Example

Summary

MATLAB routines

Filter: $H(z) = \frac{B(z)}{A(z)}$ with input $x[n]$ and output $y[n]$

$$y[n] = \sum_{k=0}^M b[k]x[n-k] - \sum_{k=1}^N a[k]y[n-k]$$

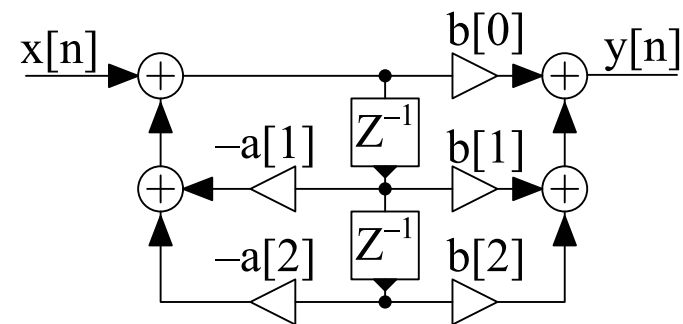
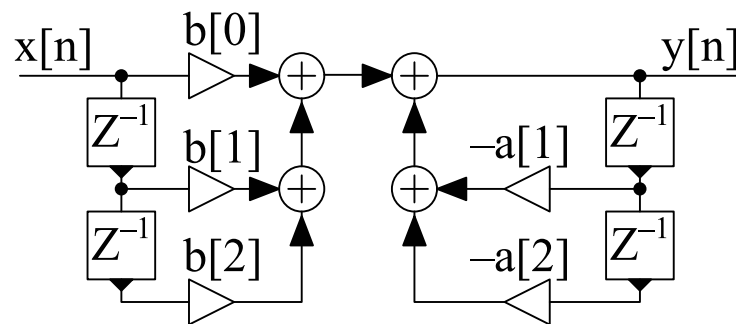
Direct forms use coefficients $a[k]$ and $b[k]$ directly

Direct Form 1:

- Direct implementation of difference equation
- Can view as $B(z)$ followed by $\frac{1}{A(z)}$

Direct Form II:

- Implements $\frac{1}{A(z)}$ followed by $B(z)$
- Saves on delays (=storage)



Transposition

10: Digital Filter Structures

Direct Forms

▷ Transposition

State Space

State Space Transfer Function

Precision Issues

Coefficient Sensitivity

Cascaded Biquads

Pole-zero

Pairing/Ordering

Linear Phase

Hardware

Implementation

Allpass Filters

Lattice Stage

Allpass Lattice

Lattice Filter

Lattice Example

Summary

MATLAB routines

Can convert any block diagram into an equivalent **transposed form**:

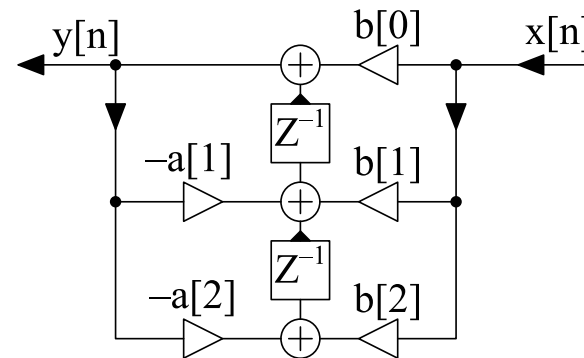
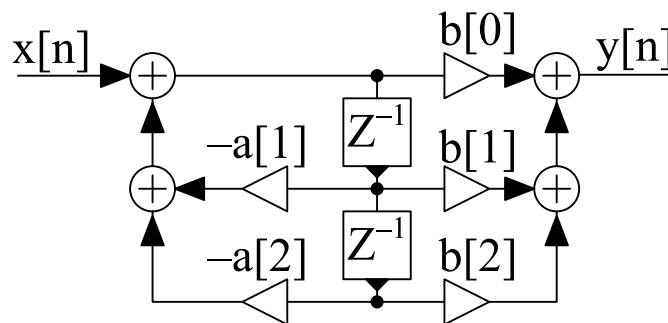
- Reverse direction of each interconnection
- Reverse direction of each multiplier
- Change junctions to adders and vice-versa
- Interchange the input and output signals

Example:

Direct form II \rightarrow Direct Form II_t

Would normally be drawn with input on the left

Note: A valid block diagram must never have any feedback loops that don't go through a delay (z^{-1} block).



State Space

10: Digital Filter Structures

Direct Forms

Transposition

▷ State Space

State Space Transfer Function

Precision Issues

Coefficient Sensitivity

Cascaded Biquads

Pole-zero

Pairing/Ordering

Linear Phase

Hardware

Implementation

Allpass Filters

Lattice Stage

Allpass Lattice

Lattice Filter

Lattice Example

Summary

MATLAB routines

$\mathbf{v}[n]$ is a vector of **delay element outputs**

Can write: $\mathbf{v}[n+1] = \mathbf{P}\mathbf{v}[n] + \mathbf{q}x[n]$

$$y[n] = \mathbf{r}^T \mathbf{v}[n] + sx[n]$$

$\{\mathbf{P}, \mathbf{q}, \mathbf{r}^T, s\}$ is the **state-space representation** of the filter structure.

The transfer function is given by:

$$H(z) = \frac{B(z)}{A(z)} = \frac{\det(z\mathbf{I} - \mathbf{P} + \mathbf{q}\mathbf{r}^T)}{\det(z\mathbf{I} - \mathbf{P})} + s - 1$$

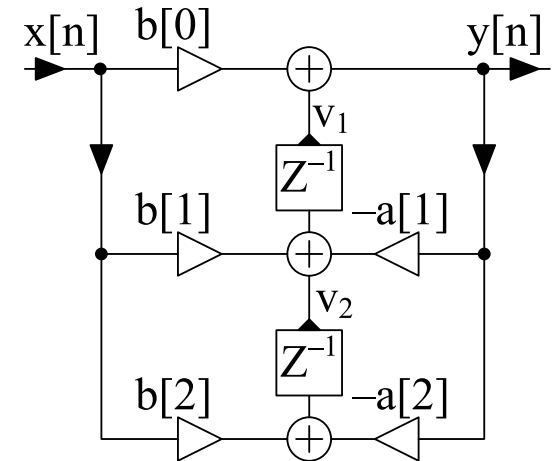
The transposed form has $\mathbf{P} \rightarrow \mathbf{P}^T$ and $\mathbf{q} \leftrightarrow \mathbf{r} \Rightarrow$ same $H(z)$

Example: Direct Form II_t

$$\mathbf{P} = \begin{pmatrix} -a[1] & 1 \\ -a[2] & 0 \end{pmatrix} \quad \mathbf{q} = \begin{pmatrix} b[1] - b[0]a[1] \\ b[2] - b[0]a[2] \end{pmatrix}$$

$$\mathbf{r}^T = \begin{pmatrix} 1 & 0 \end{pmatrix} \quad s = b[0]$$

$$\text{From which } H(z) = \frac{b[0]z^2 + b[1]z + b[2]}{z^2 + a[1]z + a[2]}$$



State Space Transfer Function

Matrix determinant lemma:

$$\begin{pmatrix} 1 & \mathbf{r}^T \\ \mathbf{0} & \mathbf{A} \end{pmatrix} \begin{pmatrix} 1 + \mathbf{r}^T \mathbf{A}^{-1} \mathbf{q} & \mathbf{0}^T \\ -\mathbf{A}^{-1} \mathbf{q} & \mathbf{I} \end{pmatrix} = \begin{pmatrix} 1 & \mathbf{0}^T \\ -\mathbf{q} & \mathbf{I} \end{pmatrix} \begin{pmatrix} 1 & \mathbf{r}^T \\ \mathbf{0} & \mathbf{A} + \mathbf{q} \mathbf{r}^T \end{pmatrix}$$
$$\Rightarrow \det(\mathbf{A}) \times (1 + \mathbf{r}^T \mathbf{A}^{-1} \mathbf{q}) = 1 \times \det(\mathbf{A} + \mathbf{q} \mathbf{r}^T)$$

[Expand $\det()$ by first row or column]

State space:

$$\begin{aligned} \mathbf{v}[n+1] &= \mathbf{P} \mathbf{v}[n] + \mathbf{q} x[n] \\ y[n] &= \mathbf{r}^T \mathbf{v}[n] + s x[n] \end{aligned}$$

Take z-transform:

$$\begin{aligned} z\mathbf{V} &= \mathbf{P}\mathbf{V} + \mathbf{q}X \\ Y &= \mathbf{r}^T \mathbf{V} + sX \end{aligned}$$

Eliminate \mathbf{V} between the equations to give the transfer function:

$$\frac{Y}{X} = \mathbf{r}^T (z\mathbf{I} - \mathbf{P})^{-1} \mathbf{q} + s = \frac{\det(z\mathbf{I} - \mathbf{P} + \mathbf{q} \mathbf{r}^T)}{\det(z\mathbf{I} - \mathbf{P})} + s - 1$$

[Use MDL with $\mathbf{A} = z\mathbf{I} - \mathbf{P}$; $\det(z\mathbf{I} - \mathbf{P})$ is the characteristic polynomial of \mathbf{P}]

If all computations were exact, it would not make any difference which of the equivalent structures was used. However ...

- Coefficient precision

Coefficients are stored to finite precision and so are not exact. The filter actually implemented is therefore incorrect.

- Arithmetic precision

Arithmetic calculations are not exact.

- Worst case for arithmetic errors is when calculating the difference between two similar values:

$$1.23456789 - 1.23455678 = 0.00001111: 9 \text{ s.f.} \rightarrow 4 \text{ s.f.}$$

Arithmetic errors introduce noise that is then filtered by the transfer function between the point of noise creation and the output.

Coefficient Sensitivity

10: Digital Filter Structures

Direct Forms

Transposition

State Space

State Space Transfer Function

Precision Issues

▷ Coefficient

▷ Sensitivity

Cascaded Biquads

Pole-zero

Pairing/Ordering

Linear Phase

Hardware

Implementation

Allpass Filters

Lattice Stage

Allpass Lattice

Lattice Filter

Lattice Example

Summary

MATLAB routines

The roots of high order polynomials can be very sensitive to small changes in coefficient values.

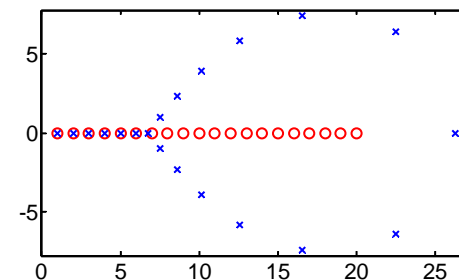
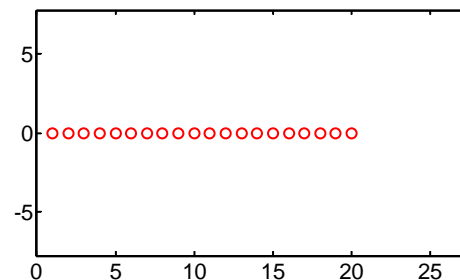
Wilkinson's polynomial: (famous example)

$$f(x) = \prod_{n=1}^{20} (x - n) = x^{20} - 210x^{19} + 20615x^{18} - \dots$$

has roots well separated on the real axis.

Multiplying the coefficient of x^{19} by 1.000001 moves the roots a lot.

“Speaking for myself I regard it as the most traumatic experience in my career as a numerical analyst”, James Wilkinson 1984



Moral: Avoid using direct form for filters orders over about 10.

Cascaded Biquads

10: Digital Filter Structures

Direct Forms

Transposition

State Space

State Space Transfer Function

Precision Issues

Coefficient Sensitivity

▷ Cascaded Biquads

Pole-zero

Pairing/Ordering

Linear Phase

Hardware

Implementation

Allpass Filters

Lattice Stage

Allpass Lattice

Lattice Filter

Lattice Example

Summary

MATLAB routines

Avoid high order polynomials by **factorizing into quadratic terms**:

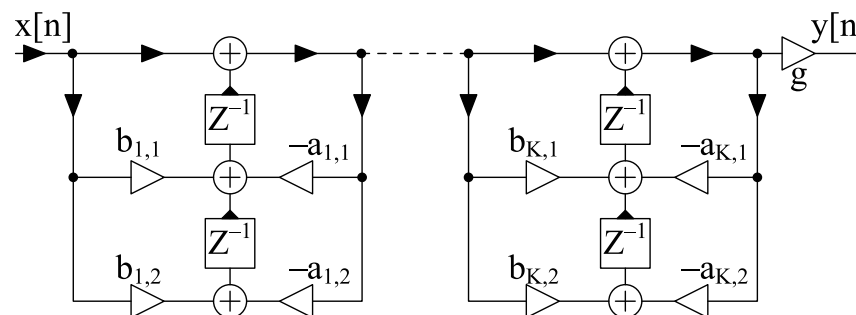
$$\frac{B(z)}{A(z)} = g \frac{\prod (1 + b_{k,1} z^{-1} + b_{k,2} z^{-2})}{\prod (1 + a_{k,1} z^{-1} + a_{k,2} z^{-2})} = g \prod_{k=1}^K \frac{1 + b_{k,1} z^{-1} + b_{k,2} z^{-2}}{1 + a_{k,1} z^{-1} + a_{k,2} z^{-2}}$$

where $K = \max \left(\left\lceil \frac{M}{2} \right\rceil, \left\lceil \frac{N}{2} \right\rceil \right)$.

The term $\frac{1 + b_{k,1} z^{-1} + b_{k,2} z^{-2}}{1 + a_{k,1} z^{-1} + a_{k,2} z^{-2}}$ is a **biquad** (bi-quadratic section).

We need to choose:

- which poles to **pair** with which zeros in each biquad
- how to **order** the biquads



Pole-zero Pairing/Ordering

10: Digital Filter Structures

Direct Forms

Transposition

State Space

State Space Transfer Function

Precision Issues

Coefficient Sensitivity

Cascaded Biquads

Pole-zero

▷ Pairing/Ordering

Linear Phase

Hardware

Implementation

Allpass Filters

Lattice Stage

Allpass Lattice

Lattice Filter

Lattice Example

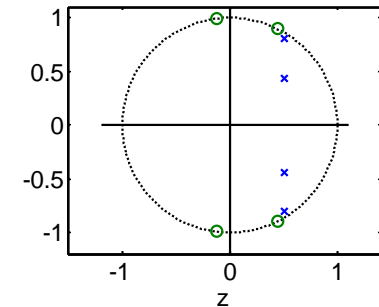
Summary

MATLAB routines

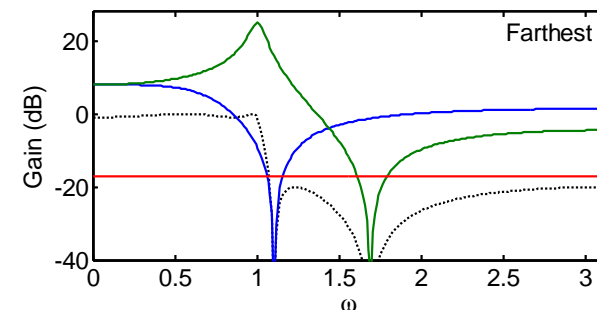
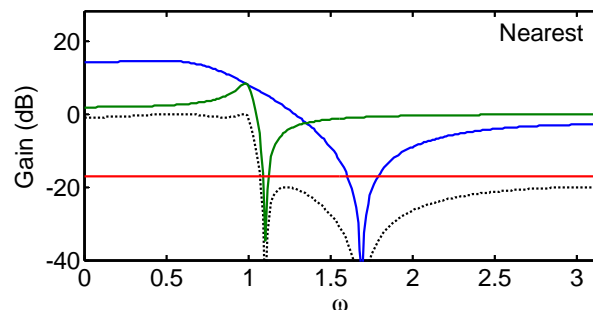
Example: Elliptic lowpass filter

2 pole pairs and 2 zero pairs
need 2 biquads

Noise introduced in one biquad is amplified
by all the subsequent ones:



- Make the peak gain of each biquad as small as possible
 - Pair poles with nearest zeros to get lowest peak gain
begin with the pole nearest the unit circle
 - Pairing with farthest zeros gives higher peak biquad gain
- Poles near the unit circle have the highest peaks and introduce most noise so place them last in the chain



Linear Phase

10: Digital Filter Structures

Direct Forms

Transposition

State Space

State Space Transfer Function

Precision Issues

Coefficient Sensitivity

Cascaded Biquads

Pole-zero

Pairing/Ordering

▷ Linear Phase

Hardware

Implementation

Allpass Filters

Lattice Stage

Allpass Lattice

Lattice Filter

Lattice Example

Summary

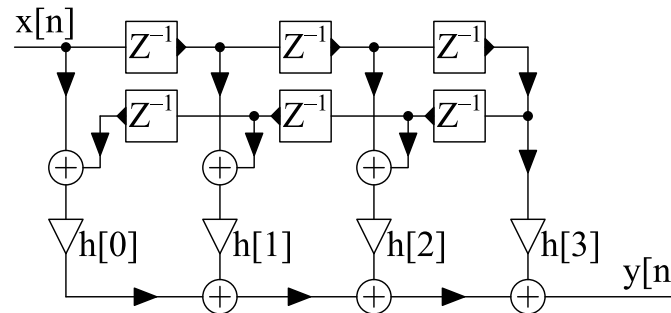
MATLAB routines

Implementation can take advantage of any symmetry in the coefficients.

Linear phase filters are always FIR and have **symmetric** (or, more rarely, **antisymmetric**) coefficients.

$$H(z) = \sum_{m=0}^M h[m] z^{-m} \quad h[M - m] = h[m]$$
$$= h\left[\frac{M}{2}\right] z^{-\frac{M}{2}} + \sum_{m=0}^{\frac{M}{2}-1} h[m] (z^{-m} + z^{m-M}) \quad [m \text{ even}]$$

We can implement this with $\frac{M}{2} + 1$ multiplies instead of $M + 1$



Hardware Implementation

10: Digital Filter Structures

Direct Forms

Transposition

State Space

State Space Transfer Function

Precision Issues

Coefficient Sensitivity

Cascaded Biquads

Pole-zero

Pairing/Ordering

Linear Phase

Hardware

▷ Implementation

Allpass Filters

Lattice Stage

Allpass Lattice

Lattice Filter

Lattice Example

Summary

MATLAB routines

Software Implementation:

All that matters is the total number of multiplies and adds

Hardware Implementation:

Delay elements (z^{-1}) represent storage registers

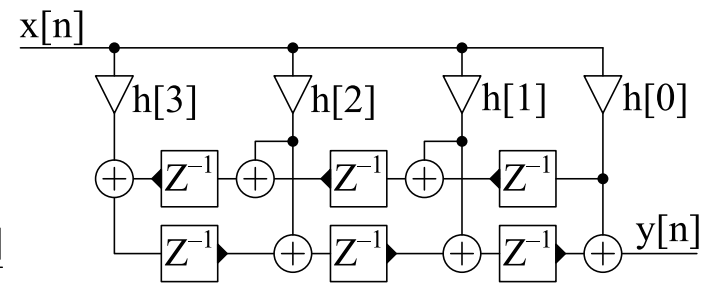
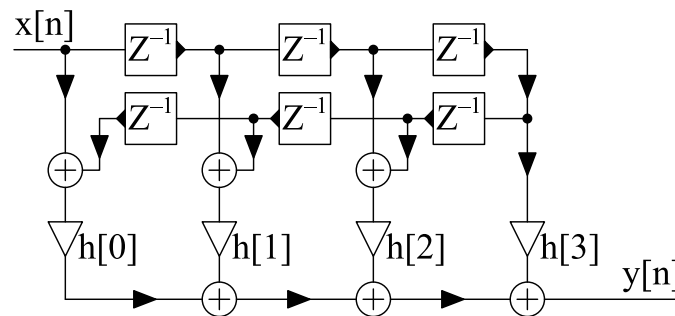
The maximum clock speed is limited by the number of sequential operations between registers

Example: Symmetric Linear Phase Filter

Direct form: Maximum sequential delay = $4a + m$

Transpose form: Maximum sequential delay = $a + m$ ☺

a and m are the delays of adder and multiplier respectively



Allpass Filters

10: Digital Filter Structures

Direct Forms

Transposition

State Space

State Space Transfer Function

Precision Issues

Coefficient Sensitivity

Cascaded Biquads

Pole-zero

Pairing/Ordering

Linear Phase

Hardware

Implementation

▷ Allpass Filters

Lattice Stage

Allpass Lattice

Lattice Filter

Lattice Example

Summary

MATLAB routines

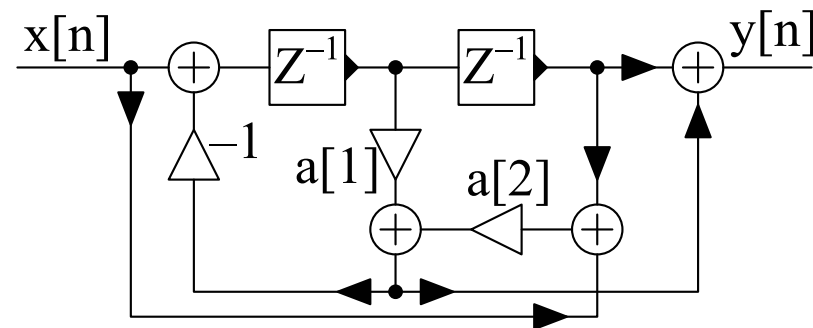
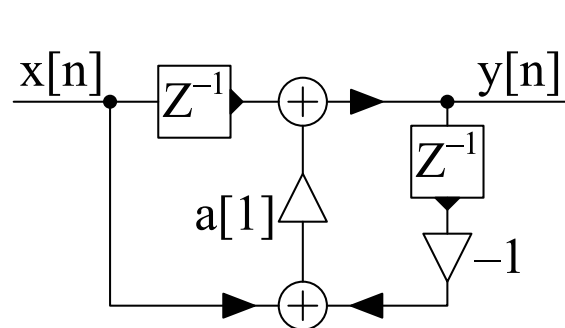
Allpass filters have **mirror image** numerator and denominator coefficients:

$$b[n] = a[N - n] \quad \Leftrightarrow \quad B(z) = z^{-N} A(z^{-1})$$

$$\Rightarrow |H(e^{j\omega})| \equiv 1 \forall \omega$$

There are several efficient structures, e.g.

- **First Order:** $H(z) = \frac{a[1] + z^{-1}}{1 + a[1]z^{-1}}$
- **Second Order:** $H(z) = \frac{a[2] + a[1]z^{-1} + z^{-2}}{1 + a[1]z^{-1} + a[2]z^{-2}}$



Allpass filters have a gain magnitude of 1 even with coefficient errors.

Lattice Stage

10: Digital Filter Structures

Direct Forms

Transposition

State Space

State Space Transfer Function

Precision Issues

Coefficient Sensitivity

Cascaded Biquads

Pole-zero

Pairing/Ordering

Linear Phase

Hardware

Implementation

Allpass Filters

▷ Lattice Stage

Allpass Lattice

Lattice Filter

Lattice Example

Summary

MATLAB routines

Suppose G is allpass: $G(z) = \frac{z^{-N} A(z^{-1})}{A(z)}$

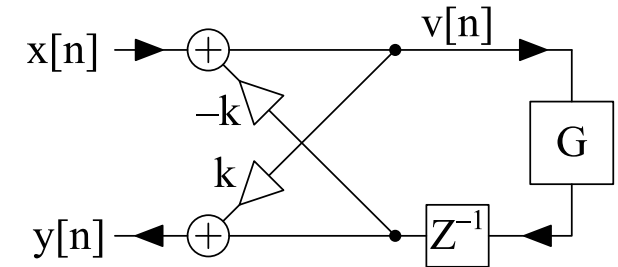
$$V(z) = X(z) - kGz^{-1}V(z)$$

$$\Rightarrow V(z) = \frac{1}{1+kGz^{-1}} X(z)$$

$$Y(z) = kV(z) + Gz^{-1}V(z) = \frac{k+z^{-1}G}{1+kGz^{-1}} X(z)$$

If $|k| < 1$ then $\frac{Y(z)}{X(z)}$ is stable since $|kG(e^{j\omega})e^{-j\omega}| < 1$

$$\frac{Y(z)}{X(z)} = \frac{kA(z) + z^{-N-1}A(z^{-1})}{A(z) + kz^{-N-1}A(z^{-1})} \triangleq \frac{z^{-(N+1)}B(z^{-1})}{B(z)}$$



Obtaining $\{b[n]\}$ from $\{a[n]\}$:

$$b[n] = \begin{cases} 1 & n = 0 \\ a[n] + ka[N+1-n] & 1 \leq n \leq N \\ k & n = N+1 \end{cases}$$

Obtaining $\{a[n]\}$ from $\{b[n]\}$:

$$k = b[N+1] \quad a[n] = \frac{b[n] - kb[N+1-n]}{1-k^2}$$

Allpass Lattice

10: Digital Filter Structures

Direct Forms

Transposition

State Space

State Space Transfer Function

Precision Issues

Coefficient Sensitivity

Cascaded Biquads

Pole-zero

Pairing/Ordering

Linear Phase

Hardware

Implementation

Allpass Filters

Lattice Stage

▷ Allpass Lattice

Lattice Filter

Lattice Example

Summary

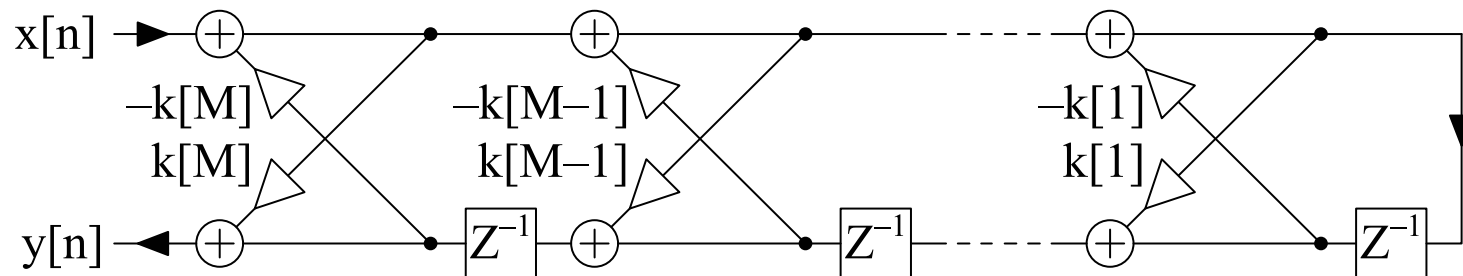
MATLAB routines

We can implement **any allpass filter** $H(z) = \frac{z^{-M} A(z^{-1})}{A(z)}$ as a lattice filter with M stages:

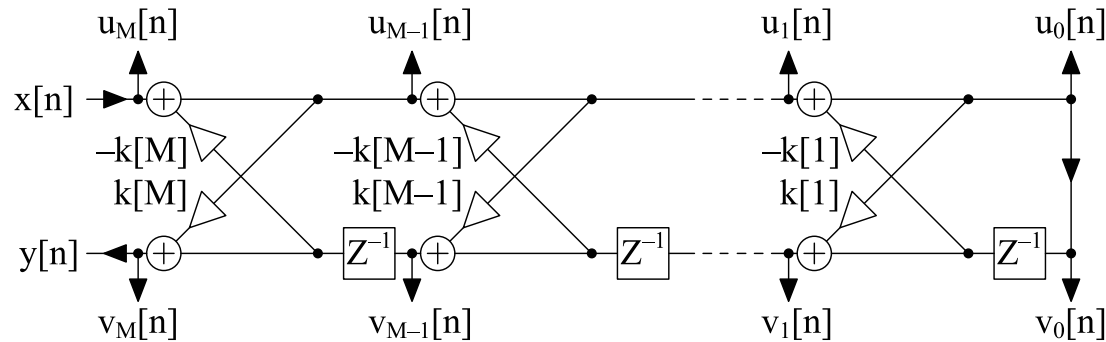
- **Initialize** $A_M(z) = A(z)$
- **Repeat** for $m = M : -1 : 1$
 - $k[m] = a_m[m]$
 - $a_{m-1}[n] = \frac{a_m[n] - k[m]a_m[m-n]}{1 - k^2[m]}$ for $0 \leq n \leq m-1$

equivalently $A_{m-1}(z) = \frac{A_m(z) - k[m]z^{-m}A_m(z^{-1})}{1 - k^2[m]}$

$A(z)$ is stable iff $|k[m]| < 1$ for all m (good stability test)



Lattice Filter



Label outputs $u_m[n]$ and $v_m[n]$ and define $H_m(z) = \frac{V_m(z)}{U_m(z)} = \frac{z^{-m} A_m(z^{-1})}{A_m(z)}$

From earlier slide:

$$\frac{U_{m-1}(z)}{U_m(z)} = \frac{1}{1 + k[m]z^{-1}H_{m-1}(z)} = \frac{A_{m-1}(z)}{A_{m-1}(z) + k[m]z^{-m}A_{m-1}(z^{-1})} = \frac{A_{m-1}(z)}{A_m(z)}$$

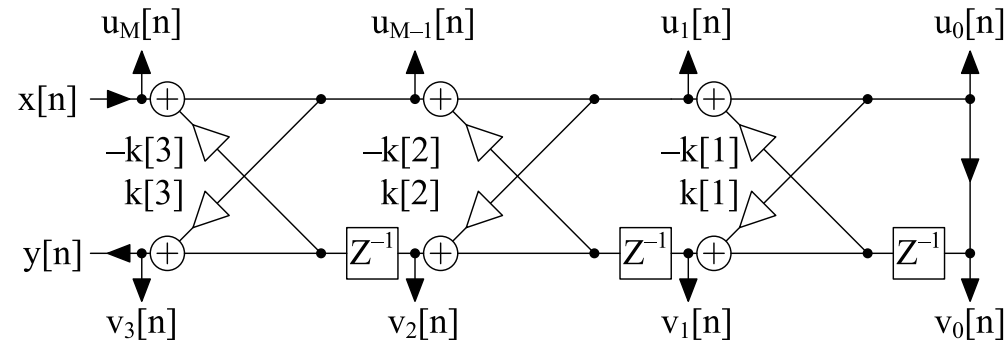
Hence:

$$\frac{U_m(z)}{X(z)} = \frac{A_m(z)}{A(z)} \quad \text{and} \quad \frac{V_m(z)}{X(z)} = \frac{U_m(z)}{X(z)} \times \frac{V_m(z)}{U_m(z)} = \frac{z^{-m} A_m(z^{-1})}{A(z)}$$

The numerator of $\frac{V_m(z)}{X(z)}$ is of order m so you can create **any numerator of order M** by summing appropriate multiples of $V_m(z)$:

$$g[n] = \sum_{m=0}^M c[m]v_m[n] \Rightarrow G(z) = \frac{\sum_{m=0}^M c[m]z^{-m} A_m(z^{-1})}{A(z)}$$

Lattice Example



$$A(z) = A_3(z) = 1 + 0.2z^{-1} - 0.23z^{-2} + 0.2z^{-3}$$

- $k[3] = 0.2 \Rightarrow a_2[\] = \frac{[1, 0.2, -0.23] - 0.2[0.2, -0.23, 0.2]}{1 - 0.2^2} = [1, 0.256, -0.281]$
- $k[2] = -0.281 \Rightarrow a_1[\] = \frac{[1, 0.256] + 0.281[-0.281, 0.256]}{1 - 0.281^2} = [1, 0.357]$
- $k[1] = 0.357 \Rightarrow a_0[\] = 1$

$$\frac{V_0(z)}{X(z)} = \frac{1}{1 + 0.2z^{-1} - 0.23z^{-2} + 0.2z^{-3}}$$

$$\frac{V_1(z)}{X(z)} = \frac{0.357 + z^{-1}}{1 + 0.2z^{-1} - 0.23z^{-2} + 0.2z^{-3}}$$

$$\frac{V_2(z)}{X(z)} = \frac{-0.281 + 0.256z^{-1} + z^{-2}}{1 + 0.2z^{-1} - 0.23z^{-2} + 0.2z^{-3}}$$

$$\frac{V_3(z)}{X(z)} = \frac{0.2 - 0.23z^{-1} + 0.2z^{-2} + z^{-3}}{1 + 0.2z^{-1} - 0.23z^{-2} + 0.2z^{-3}}$$

Add together multiples of $V_i(z)$ to create an arbitrary $\frac{B(z)}{1 + 0.2z^{-1} - 0.23z^{-2} + 0.2z^{-3}}$

Summary

10: Digital Filter Structures

Direct Forms

Transposition

State Space

State Space Transfer Function

Precision Issues

Coefficient Sensitivity

Cascaded Biquads

Pole-zero

Pairing/Ordering

Linear Phase

Hardware

Implementation

Allpass Filters

Lattice Stage

Allpass Lattice

Lattice Filter

Lattice Example

▷ Summary

MATLAB routines

- Filter block diagrams
 - Direct forms
 - Transposition
 - State space representation
- Precision issues: coefficient error, arithmetic error
 - cascaded biquads
- Allpass filters
 - first and second order sections
- Lattice filters
 - Arbitrary allpass response
 - Arbitrary IIR response by summing intermediate outputs

For further details see Mitra: 8.

MATLAB routines

10: Digital Filter Structures

Direct Forms

Transposition

State Space

State Space Transfer Function

Precision Issues

Coefficient Sensitivity

Cascaded Biquads

Pole-zero

Pairing/Ordering

Linear Phase

Hardware

Implementation

Allpass Filters

Lattice Stage

Allpass Lattice

Lattice Filter

Lattice Example

Summary

▷ MATLAB routines

residuez	$\frac{b(z^{-1})}{a(z^{-1})} \rightarrow \sum_k \frac{r_k}{1-p_k z^{-1}}$
tf2sos,sos2tf	$\frac{b(z^{-1})}{a(z^{-1})} \leftrightarrow \prod_l \frac{b_{0,l}+b_{1,l}z^{-1}+b_{2,l}z^{-2}}{1+a_{1,l}z^{-1}+a_{2,l}z^{-2}}$
zp2sos,sos2zp	$\{z_m, p_k, g\} \leftrightarrow \prod_l \frac{b_{0,l}+b_{1,l}z^{-1}+b_{2,l}z^{-2}}{1+a_{1,l}z^{-1}+a_{2,l}z^{-2}}$
zp2ss,ss2zp	$\{z_m, p_k, g\} \leftrightarrow \begin{cases} x' = Ax + Bu \\ y = Cx + Du \end{cases}$
tf2ss,ss2tf	$\frac{b(z^{-1})}{a(z^{-1})} \leftrightarrow \begin{cases} x' = Ax + Bu \\ y = Cx + Du \end{cases}$
poly	$\text{poly}(\mathbf{A}) = \det(z\mathbf{I} - \mathbf{A})$

▷ **11: Multirate Systems**

Multirate Systems

Building blocks

Resampling Cascades

Noble Identities

Noble Identities Proof

Upsampled

z-transform

Downsampled

z-transform

Downsampled

Spectrum

Perfect

Reconstruction

Commutators

Summary

MATLAB routines

11: Multirate Systems

Multirate Systems

11: Multirate Systems

▷ Multirate Systems

Building blocks

Resampling Cascades

Noble Identities

Noble Identities Proof

Upsampled

z-transform

Downsampled

z-transform

Downsampled

Spectrum

Perfect

Reconstruction

Commutators

Summary

MATLAB routines

Multirate systems include more than one sample rate

Why bother?:

- May need to **change the sample rate**
e.g. Audio sample rates include 32, 44.1, 48, 96 kHz
- Can **relax** analog or digital **filter requirements**
e.g. Audio DAC increases sample rate so that the reconstruction filter can have a more gradual cutoff
- **Reduce computational complexity**
FIR filter length $\propto \frac{f_s}{\Delta f}$ where Δf is width of transition band
Lower $f_s \Rightarrow$ shorter filter + fewer samples \Rightarrow computation $\propto \frac{1}{f_s^2}$

Building blocks

11: Multirate Systems

Multirate Systems

▷ Building blocks

Resampling Cascades

Noble Identities

Noble Identities Proof

Upsampled

z-transform

Downsampled

z-transform

Downsampled

Spectrum

Perfect

Reconstruction

Commutators

Summary

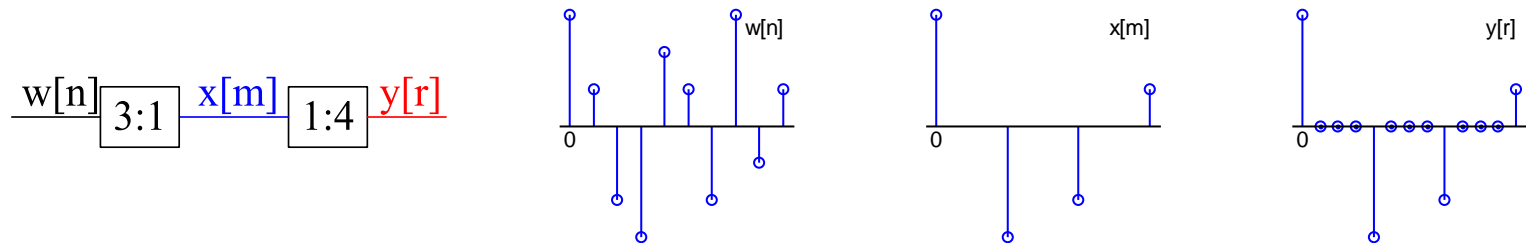
MATLAB routines

Downsample $x[n] \xrightarrow{K:1} y[m] \quad y[m] = x[Km]$

Upsample $u[m] \xrightarrow{1:K} v[n] \quad v[n] = \begin{cases} u\left[\frac{n}{K}\right] & K \mid n \\ 0 & \text{else} \end{cases}$

Example:

Downsample by 3 then upsample by 4



- We use different index variables (n, m, r) for different sample rates
- Use different colours for signals at different rates (sometimes)
- **Synchronization:** all signals have a sample at $n = 0$.

Resampling Cascades

11: Multirate Systems

Multirate Systems

Building blocks

▷ Resampling

Cascades

Noble Identities

Noble Identities Proof

Upsampled

z-transform

Downsampled

z-transform

Downsampled

Spectrum

Perfect

Reconstruction

Commutators

Summary

MATLAB routines

Successive downsamplers or upsamplers can be combined

$$\boxed{P:1} \text{---} \boxed{Q:1} \text{---} = \text{---} \boxed{PQ:1} \text{---}$$

$$\boxed{1:P} \text{---} \boxed{1:Q} \text{---} = \text{---} \boxed{1:PQ} \text{---}$$

Upsampling can be exactly inverted

$$\boxed{1:P} \text{---} \boxed{P:1} \text{---} = \text{---}$$

Downsampling **destroys information permanently** \Rightarrow uninvertible

$$\boxed{P:1} \text{---} \boxed{1:P} \text{---} \neq \text{---}$$

Resampling can be interchanged
iff **P and Q are coprime** (surprising!)

$$\boxed{P:1} \text{---} \boxed{1:Q} \text{---} = \text{---} \boxed{1:Q} \text{---} \boxed{P:1} \text{---}$$

Proof: Left side: $y[n] = x\left[\frac{P}{Q}n\right]$ if $Q \mid n$ else $y[n] = 0$.

Right side: $y[n] = x\left[\frac{P}{Q}n\right]$ if $Q \mid Pn$.

But $\{Q \mid Pn \Rightarrow Q \mid n\}$ iff P and Q are coprime.

Noble Identities

11: Multirate Systems

Multirate Systems

Building blocks

Resampling Cascades

▷ Noble Identities

Noble Identities Proof

Upsampled

z-transform

Downsampled

z-transform

Downsampled

Spectrum

Perfect

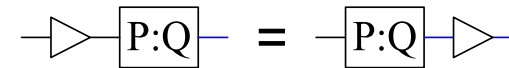
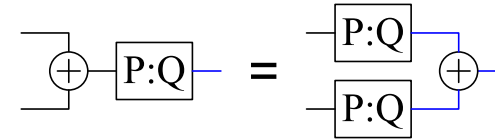
Reconstruction

Commutators

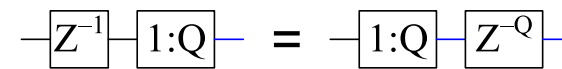
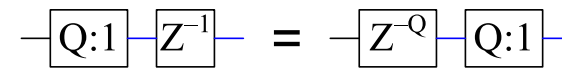
Summary

MATLAB routines

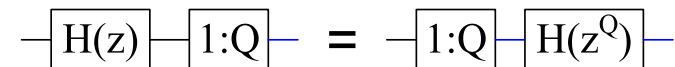
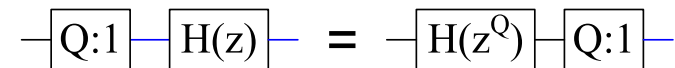
Resamplers commute with addition and multiplication



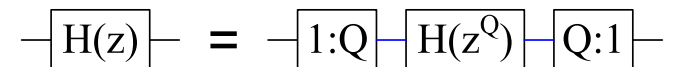
Delays must be multiplied by the resampling ratio



Noble identities:
Exchange resamplers and filters



Corrollary



Example: $H(z) = h[0] + h[1]z^{-1} + h[2]z^{-2} + \dots$
 $H(z^3) = h[0] + h[1]z^{-3} + h[2]z^{-6} + \dots$

Noble Identities Proof

11: Multirate Systems

Multirate Systems

Building blocks

Resampling Cascades

Noble Identities

▷ Noble Identities

▷ Proof

Upsampled

z-transform

Downsampled

z-transform

Downsampled

Spectrum

Perfect

Reconstruction

Commutators

Summary

MATLAB routines

Define $h_Q[n]$ to be the impulse response of $H(z^Q)$.

$$\underline{x[n]} \boxed{Q:1} \underline{u[r]} \boxed{H(z)} \underline{y[r]} = \underline{x[n]} \boxed{H(z^Q)} \underline{v[n]} \boxed{Q:1} \underline{w[r]}$$

Assume that $h[r]$ is of length $M + 1$ so that $h_Q[n]$ is of length $QM + 1$. We know that $h_Q[n] = 0$ except when $Q \mid n$ and that $h[r] = h_Q[Qr]$.

$$\begin{aligned} w[r] &= v[Qr] = \sum_{s=0}^{QM} h_Q[s] x[Qr - s] \\ &= \sum_{m=0}^M h_Q[Qm] x[Qr - Qm] = \sum_{m=0}^M h[m] x[Q(r - m)] \\ &= \sum_{m=0}^M h[m] u[r - m] = y[r] \quad \text{☺} \end{aligned}$$

Upsampled Noble Identity:

$$\underline{x[r]} \boxed{H(z)} \underline{u[r]} \boxed{1:Q} \underline{y[n]} = \underline{x[r]} \boxed{1:Q} \underline{v[n]} \boxed{H(z^Q)} \underline{w[n]}$$

We know that $v[n] = 0$ except when $Q \mid n$ and that $v[Qr] = x[r]$.

$$\begin{aligned} w[n] &= \sum_{s=0}^{QM} h_Q[s] v[n - s] = \sum_{m=0}^M h_Q[Qm] v[n - Qm] \\ &= \sum_{m=0}^M h[m] v[n - Qm] \end{aligned}$$

If $Q \nmid n$, then $v[n - Qm] = 0 \forall m$ so $w[n] = 0 = y[n]$

$$\begin{aligned} \text{If } Q \mid n = Qr, \text{ then } w[Qr] &= \sum_{m=0}^M h[m] v[Qr - Qm] \\ &= \sum_{m=0}^M h[m] x[r - m] = u[r] = y[Qr] \quad \text{☺} \end{aligned}$$

Upsampled z-transform

11: Multirate Systems

Multirate Systems

Building blocks

Resampling Cascades

Noble Identities

Noble Identities Proof

Upsampled

▷ z-transform

Downsampled

z-transform

Downsampled

Spectrum

Perfect

Reconstruction

Commutators

Summary

MATLAB routines

$$\begin{aligned} V(z) &= \sum_n v[n] z^{-n} = \sum_{n:K|n} u\left[\frac{n}{K}\right] z^{-n} \\ &= \sum_m u[m] z^{-Km} = U(z^K) \end{aligned}$$

$$\underline{u[m]} \boxed{1:K} \underline{v[n]}$$

$$\underline{U(z)} \boxed{1:K} \underline{U(z^K)}$$

Frequency Spectrum:

$$V(e^{j\omega}) = U(e^{jK\omega})$$

Frequency spectrum is horizontally shrunk and replicated K times.

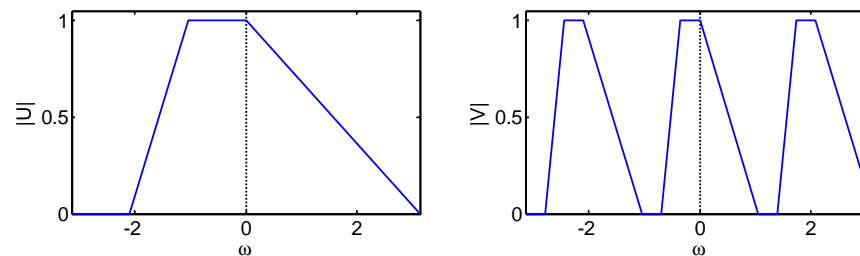
Upsampling normally **followed** by a LP filter to remove images.

Example:

Asymmetric real spectrum (\Rightarrow complex signal)

$K = 3$: three images of the original spectrum in all.

$$\text{Energy unchanged: } \frac{1}{2\pi} \int |U(e^{j\omega})|^2 d\omega = \frac{1}{2\pi} \int |V(e^{j\omega})|^2 d\omega$$



Downsampled z-transform

11: Multirate Systems

Multirate Systems

Building blocks

Resampling Cascades

Noble Identities

Noble Identities Proof

Upsampled

z-transform

Downsampled

z-transform

Downsampled

Spectrum

Perfect

Reconstruction

Commutators

Summary

MATLAB routines

Define $c_K[n] = \delta_{K|n}[n] = \frac{1}{K} \sum_{k=0}^{K-1} e^{\frac{j2\pi kn}{K}}$

$$\underline{x[n]}_{K:1} \underline{y[m]}_{1:K} \underline{x_K[n]}$$

Now define $x_K[n] = \begin{cases} x[n] & K | n \\ 0 & K \nmid n \end{cases} = c_K[n]x[n]$

$$\begin{aligned} X_K(z) &= \sum_n x_K[n] z^{-n} = \frac{1}{K} \sum_n \sum_{k=0}^{K-1} e^{\frac{j2\pi kn}{K}} x[n] z^{-n} \\ &= \frac{1}{K} \sum_{k=0}^{K-1} \sum_n x[n] \left(e^{\frac{-j2\pi k}{K}} z \right)^{-n} = \frac{1}{K} \sum_{k=0}^{K-1} X(e^{\frac{-j2\pi k}{K}} z) \end{aligned}$$

From previous slide:

$$\underline{X(z)}_{K:1} \underline{\frac{1}{K} \sum_{k=0}^{K-1} X(e^{\frac{-j2\pi k}{K}} z^{\frac{1}{K}})}$$

$$\begin{aligned} X_K(z) &= Y(z^K) \\ \Rightarrow Y(z) &= X_K(z^{\frac{1}{K}}) = \frac{1}{K} \sum_{k=0}^{K-1} X(e^{\frac{-j2\pi k}{K}} z^{\frac{1}{K}}) \end{aligned}$$

Frequency Spectrum:

$$\begin{aligned} Y(e^{j\omega}) &= \frac{1}{K} \sum_{k=0}^{K-1} X(e^{\frac{j(\omega-2\pi k)}{K}}) \\ &= \frac{1}{K} \left(X(e^{\frac{j\omega}{K}}) + X(e^{\frac{j\omega}{K} - \frac{2\pi}{K}}) + X(e^{\frac{j\omega}{K} - \frac{4\pi}{K}}) + \dots \right) \end{aligned}$$

Horizontally expanded by a factor of K and aliased K times.

Downsampling is normally **preceded** by a LP filter to prevent aliasing.

Downsampled Spectrum

$$Y(e^{j\omega}) = \frac{1}{K} \sum_{k=0}^{K-1} X(e^{j(\omega - 2\pi k)/K})$$

$$\underline{x[n]} \boxed{K:1} \underline{y[m]}$$

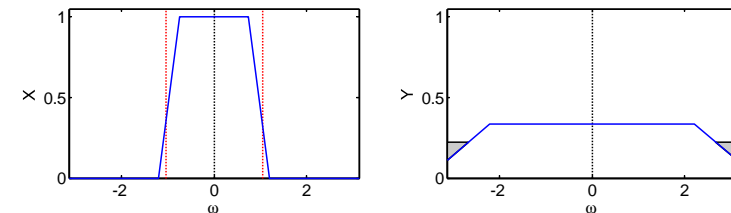
Example 1:

$$K = 3$$

Not quite limited to $\pm \frac{\pi}{K}$

Shaded region shows aliasing

$$\text{Energy decreases: } \frac{1}{2\pi} \int |Y(e^{j\omega})|^2 d\omega \approx \frac{1}{2\pi K} \int |X(e^{j\omega})|^2 d\omega$$

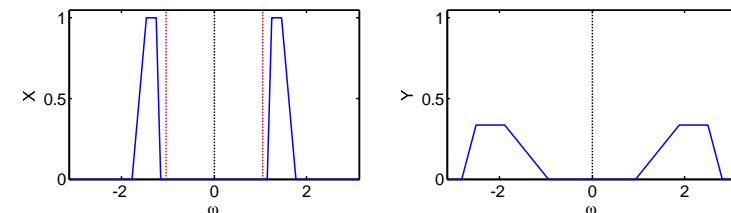


Example 2:

$$K = 3$$

Limited to $\frac{\pi}{K} \leq |\omega| \leq 2\frac{\pi}{K}$

No aliasing: ☺



Criterion for no aliasing:

Normally quoted Nyquist criterion is $|\omega| \leq \frac{\pi}{K}$

Actually OK if spectral energy is restricted to $r\frac{\pi}{K} \leq |\omega| \leq (r+1)\frac{\pi}{K}$

Perfect Reconstruction

11: Multirate Systems

Multirate Systems

Building blocks

Resampling Cascades

Noble Identities

Noble Identities Proof

Upsampled

z-transform

Downsampled

z-transform

Downsampled

Spectrum

Perfect

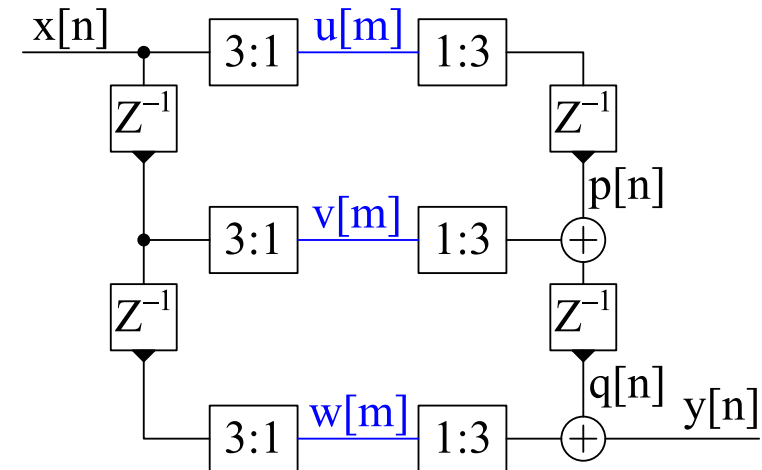
▷ Reconstruction

Commutators

Summary

MATLAB routines

$x[n]$	c d e f g h i j k l m n
$u[m]$	c f i l
$p[n]$	- c - - f - - i - - l
$v[m]$	b e h k
$q[n]$	- b c - e f - h i - k l
$w[m]$	a d g j
$y[n]$	a b c d e f g h i j k l



Input sequence $x[n]$ is split into three streams:

$$u[m] = x[3m], \quad v[m] = x[3m - 1], \quad w[m] = x[3m - 2]$$

Following upsampling, the streams are aligned by the delays and then added to give:

$$y[n] = x[n - 2]$$

Perfect Reconstruction: output is a delayed scaled replica of the input

Commutators

11: Multirate Systems

Multirate Systems

Building blocks

Resampling Cascades

Noble Identities

Noble Identities Proof

Upsampled

z-transform

Downsampled

z-transform

Downsampled

Spectrum

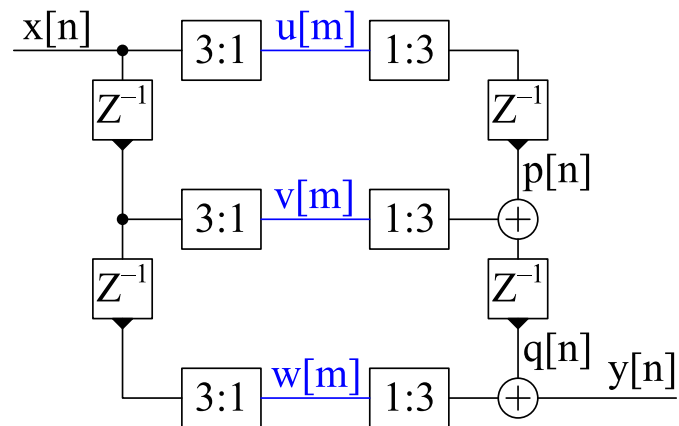
Perfect

Reconstruction

► Commutators

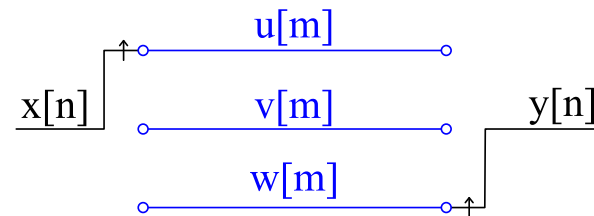
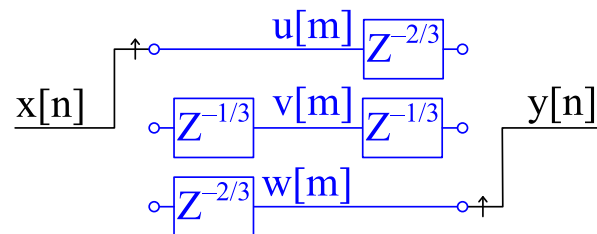
Summary

MATLAB routines



$x[n]$	c d e f g h i j k l m n
$u[m]$	c f i l
$v[m]$	b e h k
$w[m]$	a d g j
$v[m + \frac{1}{3}]$	e h k l
$w[m + \frac{2}{3}]$	d g j m
$y[n]$	a b c d e f g h i j k l

The combination of delays and downsamplers can be regarded as a **commutator** that **distributes values in sequence** to u , w and v . Fractional delays, $z^{-\frac{1}{3}}$ and $z^{-\frac{2}{3}}$ are needed to synchronize the streams. The **output commutator** takes values from the streams in sequence. For clarity, we omit the fractional delays and regard each terminal, \circ , as holding its value until needed. **Initial commutator position has zero delay.**



The commutator direction is **against the direction** of the z^{-1} delays.

Summary

11: Multirate Systems

Multirate Systems

Building blocks

Resampling Cascades

Noble Identities

Noble Identities Proof

Upsampled

z-transform

Downsampled

z-transform

Downsampled

Spectrum

Perfect

Reconstruction

Commutators

▷ Summary

MATLAB routines

- Multirate Building Blocks

- **Upsample:** $X(z) \xrightarrow{1:K} X(z^K)$
Invertible, Inserts $K - 1$ zeros between samples
Shrinks and replicates spectrum
Follow by LP filter to remove images
- **Downsample:** $X(z) \xrightarrow{K:1} \frac{1}{K} \sum_{k=0}^{K-1} X(e^{-j\frac{2\pi k}{K}} z^{\frac{1}{K}})$
Destroys information, keeps every K^{th} sample
Expands and aliases spectrum
Precede by LP filter to prevent aliases

- Equivalences

- Noble Identities: $H(z) \longleftrightarrow H(z^K)$
- Interchange $P : 1$ and $1 : Q$ iff P and Q coprime

- Commutators

- Combine delays and down/up sampling

For further details see Mitra: 13.

MATLAB routines

11: Multirate Systems

Multirate Systems

Building blocks

Resampling Cascades

Noble Identities

Noble Identities Proof

Upsampled

z-transform

Downsampled

z-transform

Downsampled

Spectrum

Perfect

Reconstruction

Commutators

Summary

▷ MATLAB routines

resample

change sampling rate

▷ 12: Polyphase

Filters

Heavy Lowpass
filtering

Maximum Decimation
Frequency

Polyphase
decomposition

Downsampled
Polyphase Filter

Polyphase Upsampler

Complete Filter

Upsampler
Implementation

Downsampler
Implementation

Summary

12: Polyphase Filters

Heavy Lowpass filtering

12: Polyphase Filters

Heavy Lowpass

filtering

Maximum Decimation Frequency

Polyphase decomposition

Downsampled Polyphase Filter

Polyphase Upsampler

Complete Filter

Upsampler

Implementation

Downsampler Implementation

Summary

Filter Specification:

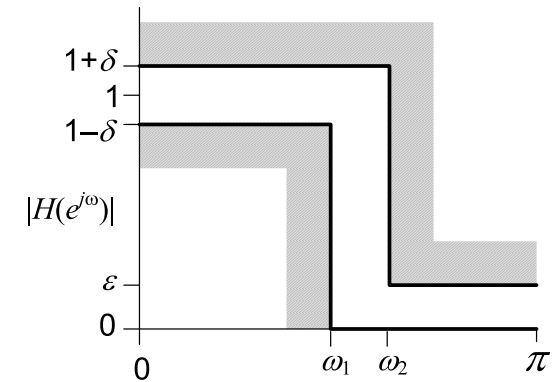
Sample Rate: 20 kHz

Passband edge: 100 Hz ($\omega_1 = 0.03$)

Stopband edge: 300 Hz ($\omega_2 = 0.09$)

Passband ripple: ± 0.05 dB ($\delta = 0.006$)

Stopband Gain: -80 dB ($\epsilon = 0.0001$)

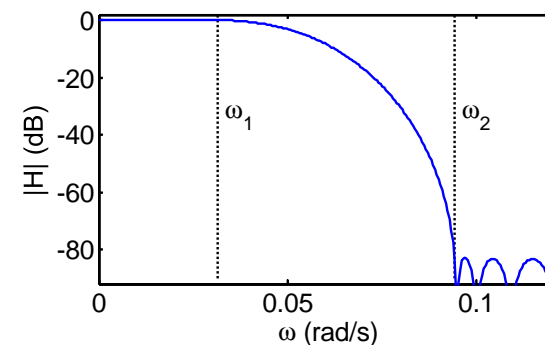
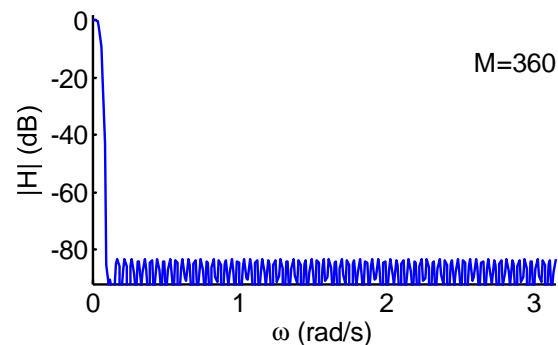


This is an extreme filter because the cutoff frequency is only 1% of the Nyquist frequency.

Symmetric FIR Filter:

Design with Remez-exchange algorithm

Order = 360



Maximum Decimation Frequency

12: Polyphase Filters

Heavy Lowpass filtering

Maximum Decimation Frequency

Polyphase decomposition

Downsampled Polyphase Filter

Polyphase Upsampler

Complete Filter

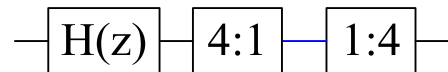
Upsampler

Implementation

Downsampler Implementation

Summary

If a filter passband occupies only a small fraction of $[0, \pi]$, we can downsample then upsample without losing information.



Downsample: aliased components at offsets of $\frac{2\pi}{K}$ are all zero because of $H(z)$

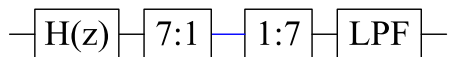
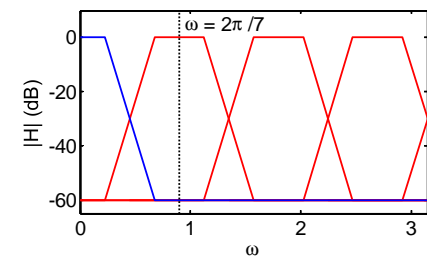
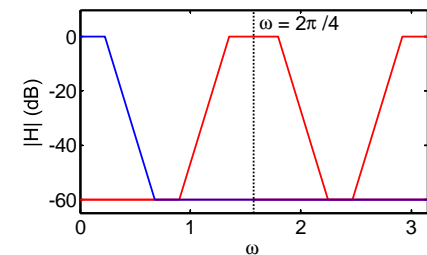
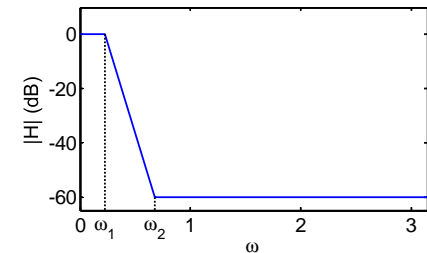
Upsample: Images spaced at $\frac{2\pi}{K}$ can be removed using another low pass filter

To avoid aliasing in the passband, we need

$$\frac{2\pi}{K} - \omega_2 \geq \omega_1 \Rightarrow K \leq \frac{2\pi}{\omega_1 + \omega_2}$$

Normally place the centre of the transition band at the intermediate Nyquist frequency.

We must add a lowpass filter to remove the images:



Polyphase decomposition

12: Polyphase Filters
Heavy Lowpass
filtering
Maximum Decimation
Frequency
Polyphase
decomposition
Downsampled
Polyphase Filter
Polyphase Upsampler
Complete Filter
Upsampler
Implementation
Downsampler
Implementation
Summary

For our filter: original Nyquist frequency = 10 kHz and transition band centre is at 200 Hz so we can use $K = 50$.

For convenience, zero-pad $h[n]$ to order $M' = RK - 1$.

$$M = 360 \Rightarrow R = 8 \Rightarrow M' = 399$$

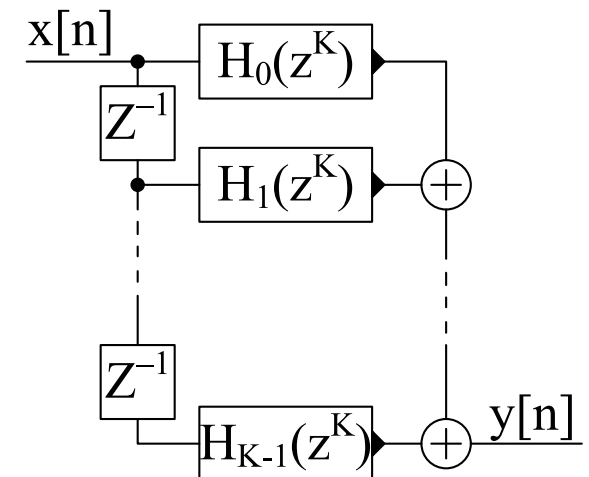
$$\begin{aligned} H(z) &= \sum_{m=0}^{M'} h[m] z^{-m} \\ &= \sum_{m=0}^{K-1} h[m] z^{-m} + \sum_{m=K}^{2K-1} h[m] z^{-m} + \dots \\ &= \sum_{m=0}^{K-1} \sum_{r=0}^{R-1} h[m + Kr] z^{-m-Kr} \\ &= \sum_{m=0}^{K-1} z^{-m} \sum_{r=0}^{R-1} h_m[r] z^{-Kr} \end{aligned}$$

$$\text{where } h_m[r] = h[m + Kr]$$

Example:

$$\begin{aligned} h_0[r] &= [h[0] \quad h[50] \quad \dots \quad h[350]] \\ h_1[r] &= [h[1] \quad h[51] \quad \dots \quad h[351]] \end{aligned}$$

This is a **polyphase** implementation of the filter $H(z)$
Split $H(z)$ into K filters each of order $R - 1$



Downsampled Polyphase Filter

12: Polyphase Filters

Heavy Lowpass
filtering

Maximum Decimation
Frequency

Polyphase
decomposition

Downsampled
Polyphase Filter

Polyphase Upsampler

Complete Filter

Upsampler

Implementation

Downsampler

Implementation

Summary

$H(z)$ is low pass so we downsample its output by K without aliasing.

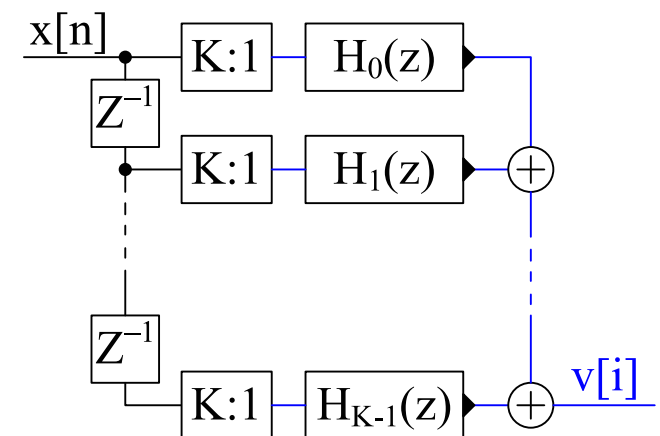
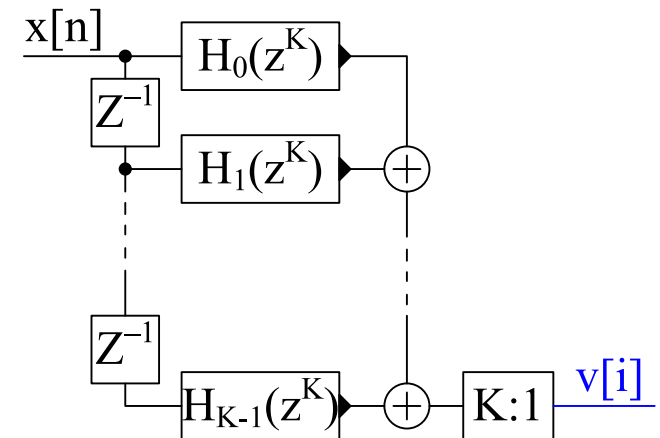
The number of multiplications per input sample is $M + 1 = 361$.

Note that $H_0(z)$ to $H_{10}(z)$ are of order 7 while the rest are of order 6.

Using the Noble identities, we can move the resampling back through the adders and filters. $H_m(z^K)$ turns into $H_m(z)$ at a lower sample rate.

We still perform 361 multiplications but now only once for every K input samples.

Multiplications per input sample = 7.2 (down by a factor of 50 😊) but $v[n]$ has the wrong sample rate (😞).



Polyphase Upsampler

12: Polyphase Filters

Heavy Lowpass filtering

Maximum Decimation Frequency

Polyphase decomposition

Downsampled Polyphase Filter

▷ Polyphase Upsampler

Complete Filter

Upsampler Implementation

Downsampler Implementation

Summary

To restore sample rate: upsample and then lowpass filter to remove images

We can use the same lowpass filter, $H(z)$, in polyphase form:

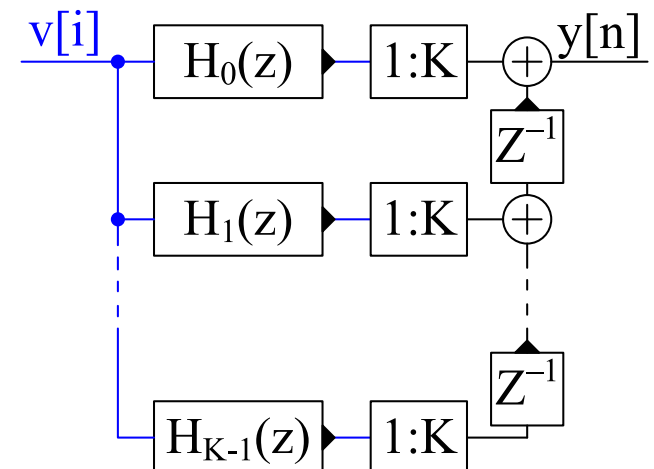
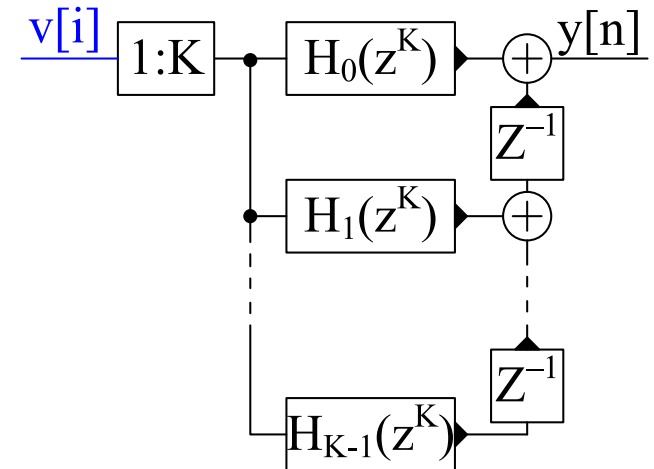
$$\sum_{m=0}^{K-1} z^{-m} \sum_{r=0}^{R-1} h_m[r] z^{-Kr}$$

This time we put the delay z^{-m} after the filters.

Multiplications per output sample = 361

Using the Noble identities, we can move the resampling forwards through the filters. $H_m(z^K)$ turns into $H_m(z)$ at a lower sample rate.

Multiplications per output sample = 7.2 (down by a factor of 50 😊).



Complete Filter

12: Polyphase Filters

Heavy Lowpass
filtering

Maximum Decimation
Frequency

Polyphase
decomposition

Downsampled
Polyphase Filter

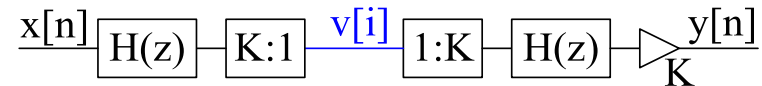
Polyphase Upsampler

▷ Complete Filter

Upsampler
Implementation

Downsampler
Implementation

Summary



The overall system implements:

Need an extra gain of K to compensate for the downsampling energy loss.

Filtering at downsampled rate requires 14.4 multiplications per input sample (7.2 for each filter). Reduced by $\frac{K}{2}$ from the original 361.

$H(e^{j\omega})$ reaches -10 dB at the downsampler

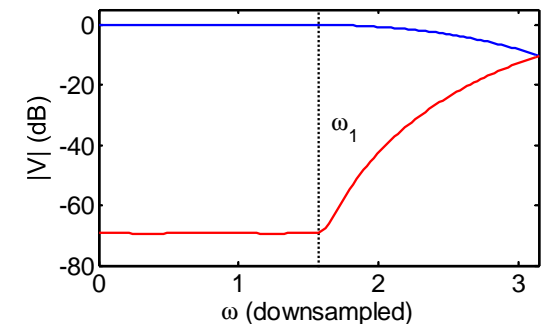
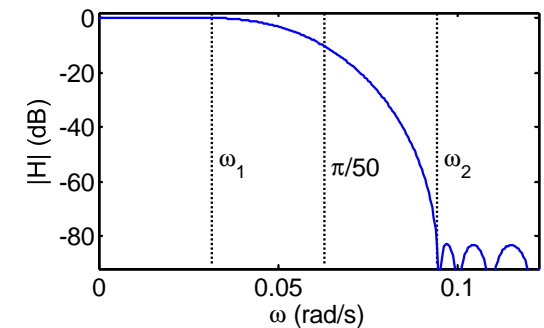
Nyquist frequency of $\frac{\pi}{K}$.

Spectral components $> \frac{\pi}{K}$ will be aliased down in frequency in $V(e^{j\omega})$.

For $V(e^{j\omega})$, passband gain (blue curve) follows the same curve as $X(e^{j\omega})$.

Noise arises from K aliased spectral intervals.

Unit white noise in $X(e^{j\omega})$ gives passband noise floor at -69 dB (red curve) even though stop band ripple is below -83 dB (due to $K - 1$ aliased stopband copies).



Upsampler Implementation

- 12: Polyphase Filters
- Heavy Lowpass filtering
- Maximum Decimation Frequency
- Polyphase decomposition
- Downsampled Polyphase Filter
- Polyphase Upsampler
- Complete Filter
 - Upsampler
 - Implementation
 - Downsampler
 - Implementation
 - Summary

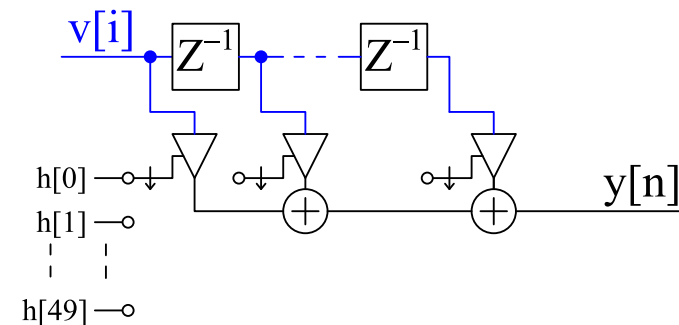
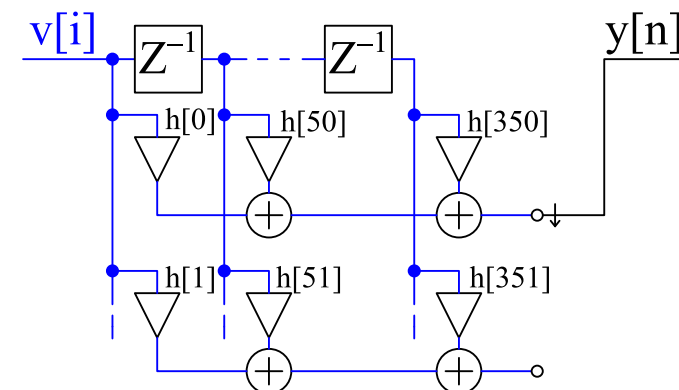
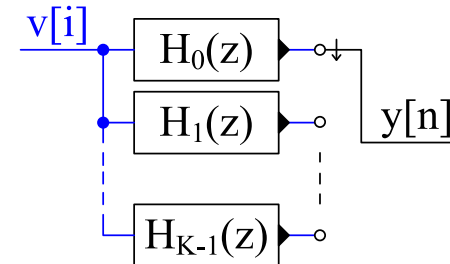
We can represent the upsampler compactly using a commutator.

$H_0(z)$ comprises a sequence of 7 delays, 7 adders and 8 gains.

We can share the delays between all 50 filters.

We can also share the gains and adders between all 50 filters and use commutators to switch the coefficients.

We now need 7 delays, 7 adders and 8 gains for the entire filter.



Downsampler Implementation

- 12: Polyphase Filters
- Heavy Lowpass filtering
- Maximum Decimation Frequency
- Polyphase decomposition
- Downsampled Polyphase Filter
- Polyphase Upsampler
- Complete Filter
- Upsampler Implementation
- Downsampler Implementation
- Summary

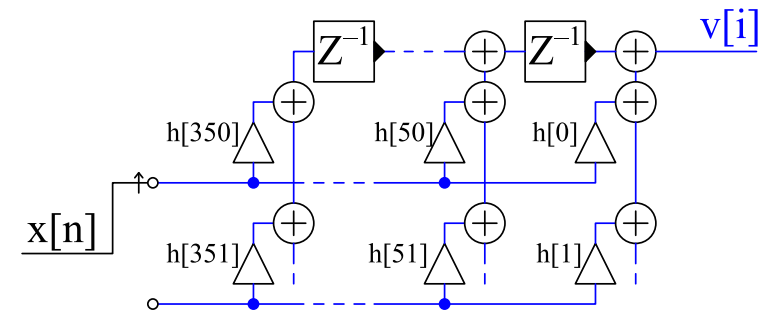
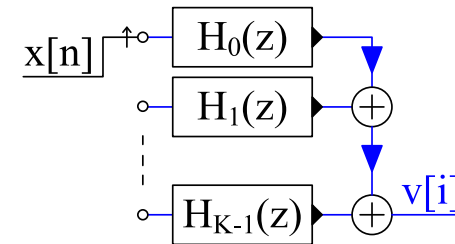
We can again use a commutator. The outputs from all 50 filters are added together to form $v[i]$.

We use the transposed form of $H_m(z)$ because this will allow us to share components.

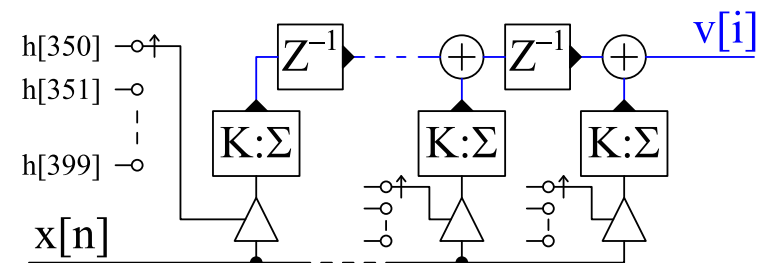
We can sum the outputs of the gain elements using an **accumulator** which sums the previous K samples.

Now we can share all the components and use commutators to switch the gain coefficients.

We need 7 delays, 7 adders, 8 gains and 8 accumulators in total.



$$u[n] \xrightarrow{K:\Sigma} w[i] \quad w[i] = \sum_{r=0}^{K-1} u[Ki - r]$$



Summary

12: Polyphase Filters

Heavy Lowpass
filtering

Maximum Decimation
Frequency

Polyphase
decomposition

Downsampled
Polyphase Filter

Polyphase Upsampler

Complete Filter

Upsampler

Implementation

Downsampler

Implementation

▷ Summary

- Filtering should be performed at the lowest possible sample rate
 - reduce filter computation by K
 - actual saving is only $\frac{K}{2}$ because you need a second filter
 - downsampled Nyquist frequency $\geq \max \omega_{\text{passband}} + \frac{\Delta\omega}{2}$
- Polyphase decomposition: split $H(z)$ as $\sum_{m=0}^{K-1} z^{-m} H_m(z^K)$
 - each $H_m(z^K)$ can operate on subsampled data
 - combine the filtering and down/up sampling
- Noise floor is higher because it arises from K spectral intervals that are aliased together
- Share components between the K filters
 - multiplier gain coefficients switch at the original sampling rate
 - need a new component: accumulator/downsampler ($K : \Sigma$)

For further details see Harris 5.

▷ **13: Resampling
Filters**

Resampling
Halfband Filters
Dyadic 1:8 Upsampler
Rational
Downsampling
Arbitrary Resampling
Polynomial
Approximation
Farrow Filter
Summary
MATLAB routines

13: Resampling Filters

Resampling

13: Resampling Filters

▷ Resampling
Halfband Filters
Dyadic 1:8 Upsampler
Rational
Downsampling
Arbitrary Resampling
Polynomial
Approximation
Farrow Filter
Summary
MATLAB routines

Suppose we want to change the sample rate while preserving information:
e.g. Audio 44.1 kHz ↔ 48 kHz ↔ 96 kHz

Downsample:

LPF to **new** Nyquist bandwidth: $\omega_0 = \frac{\pi}{K}$



Upsample:

LPF to **old** Nyquist bandwidth: $\omega_0 = \frac{\pi}{K}$



Rational ratio: $f_s \times \frac{P}{Q}$

LPF to **lower of old and new** Nyquist
bandwidths: $\omega_0 = \frac{\pi}{\max(P, Q)}$



- The transition band centre should be at the Nyquist frequency, ω_0 .
- Filter order $M \approx \frac{a}{3.5\Delta\omega}$ where a is stopband attenuation in dB and $\Delta\omega$ is the transition bandwidth.
- Transition bandwidth, $\Delta\omega$, is typically a fixed fraction of ω_0 , e.g. $0.1\omega_0$. This gives $M \approx \frac{aK}{0.35\pi} = 0.9aK$ (using Remez-exchange estimate)
- Polyphase decomposition reduces computation by K or $\max(P, Q)$.

Halfband Filters

13: Resampling Filters

Resampling

▷ Halfband Filters

Dyadic 1:8 Upsampler

Rational

Downsampling

Arbitrary Resampling

Polynomial

Approximation

Farrow Filter

Summary

MATLAB routines

If $K = 2$ then the new Nyquist frequency is $\omega_0 = \frac{\pi}{2}$.

We multiply ideal response $\frac{\sin \omega_0 n}{\pi n}$ by a Kaiser window. All even numbered points are zero except $h[0] = 0.5$.

If $4 \mid M$ and we make the filter causal ($\times z^{-\frac{M}{2}}$),

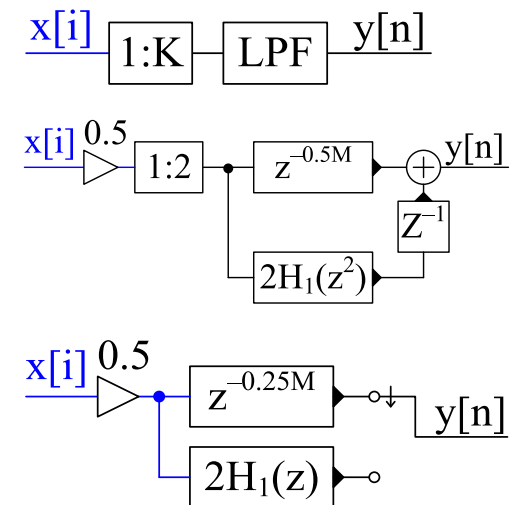
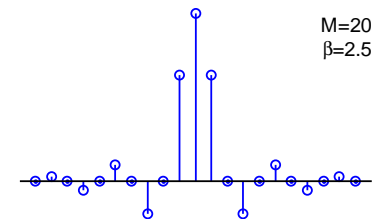
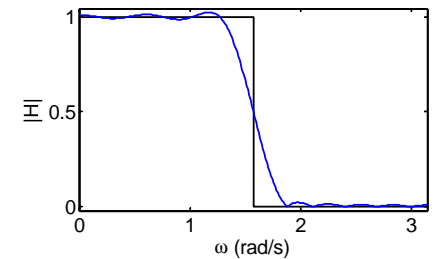
$$H(z) = 0.5z^{-\frac{M}{2}} + z^{-1} \sum_{r=0}^{\frac{M}{2}-1} h_1[r]z^{-2r}$$

Half-band upampler:

We interchange the filters with the 1:2 block and use the commutator notation.

$H_1(z)$ is symmetrical with $\frac{M}{2}$ coefficients so we need $\frac{M}{4}$ multipliers in total (input gain of 0.5 can usually be absorbed elsewhere).

Computation: $\frac{M}{4}$ multiplies per input sample.



Dyadic 1:8 Upsampler

13: Resampling Filters

Resampling Halfband Filters

Dyadic 1:8 ▷ Upsampler

Rational Downsampling

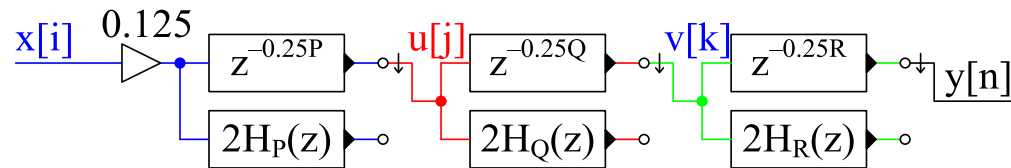
Arbitrary Resampling

Polynomial Approximation

Farrow Filter

Summary

MATLAB routines



Suppose $X(z)$: BW = 0.8π @ sample freq f_s

Upsample 1:2 $\rightarrow U(z)$:

Filter $H_P(z)$ must remove image: $\Delta\omega = 0.2\pi$

For attenuation = 60 dB, $P \approx \frac{60}{3.5\Delta\omega} = 27.3$

Round up to a multiple of 4: $P = 28$

Upsample 1:2 $\rightarrow V(z)$: $\Delta\omega = 0.6\pi \Rightarrow Q = 12$

Upsample 1:2 $\rightarrow Y(z)$: $\Delta\omega = 0.8\pi \Rightarrow R = 8$

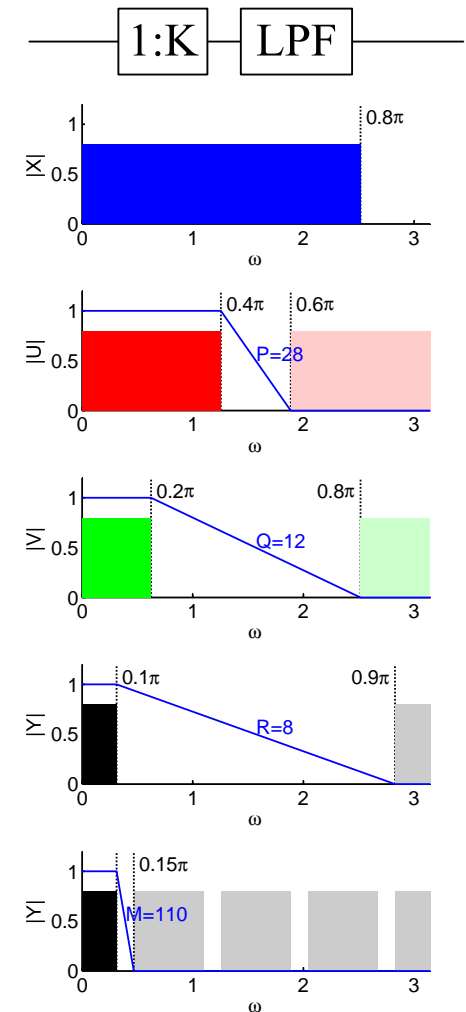
[diminishing returns + higher sample rate]

Multiplication Count:

$$\left(1 + \frac{P}{4}\right) \times f_s + \frac{Q}{4} \times 2f_s + \frac{R}{4} \times 4f_s = 22f_s$$

Alternative approach using direct 1:8 upsampling:

$\Delta\omega = 0.05\pi \Rightarrow M = 110 \Rightarrow 111f_s$ multiplications (polyphase)



Rational Downsampling

13: Resampling Filters

Resampling

Halfband Filters

Dyadic 1:8 Upsampler

Rational

Downsampling

Arbitrary Resampling

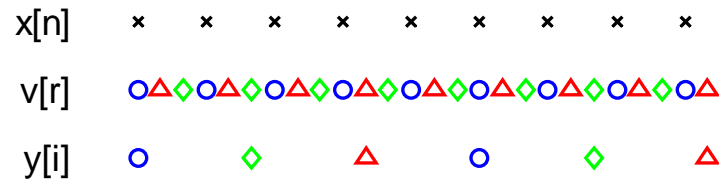
Polynomial

Approximation

Farrow Filter

Summary

MATLAB routines



Resample by $\frac{P}{Q} = \frac{3}{5}$:

$$H(z): \omega_0 = (1 - \alpha) \frac{\pi}{Q}, \Delta\omega = \frac{2\alpha\pi}{Q}$$

Polyphase: $H(z) = \sum_{p=0}^{P-1} z^{-p} H_p(z^P)$

Commutate coefficients:

calculate three $v[r]$ for each $x[n]$

Keep only every Q^{th} output:

$y[i]$ uses $H_p(z)$ with $p = Qi \bmod P$

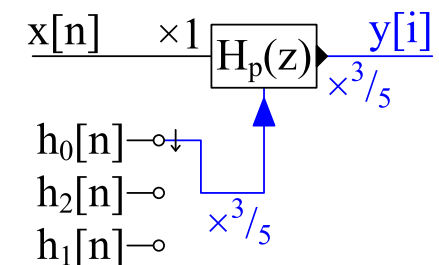
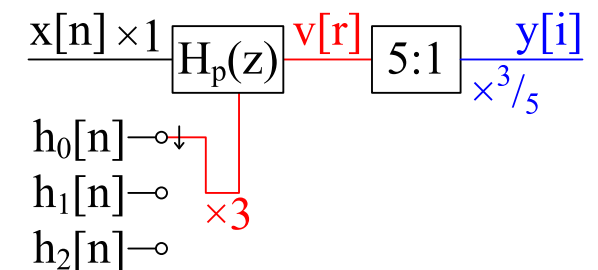
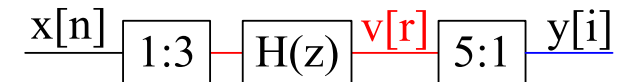
Multiplication Count:

$$H(z) \text{ order: } \frac{60 \text{ [dB]}}{3.5\Delta\omega} = \frac{2.7Q}{\alpha}$$

$$H_p(z) \text{ order: } \frac{2.7Q}{\alpha P}$$

$$\text{Multiplication rate: } \frac{2.7Q}{\alpha P} \times \frac{P}{Q} f_x = \frac{2.7}{\alpha} f_x \text{ (or } \frac{2.7}{\alpha} f_y \text{ for upsampling)}$$

To resample by $\frac{P}{Q}$ do 1:P then LPF, then Q:1.



$$\frac{2.7Q}{\alpha} \text{ coefficients in total}$$

Arbitrary Resampling

13: Resampling Filters

Resampling
Halfband Filters
Dyadic 1:8 Upsampler
Rational
Downsampling

Arbitrary
▷ Resampling
Polynomial
Approximation
Farrow Filter
Summary
MATLAB routines

Sometimes need very large P and Q :

$$\text{e.g. } \frac{44.1 \text{ kHz}}{48 \text{ kHz}} = \frac{147}{160}$$

Multiplication rate $\propto \frac{f_x}{\alpha}$ indep of P , Q .

However need P sets of coefficients.

Alternatively, use a conveniently large P and round down to the nearest sample:

for $y[i]$ at time $i\tau$ use

$$p = (\text{floor}(i\tau P))_{\text{mod } P}$$

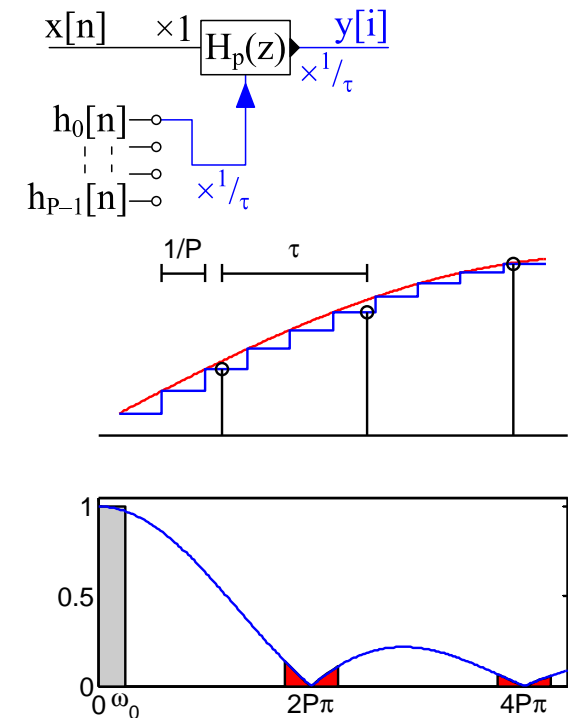
Equivalent to converting to analog with zero-order hold and resampling at $f_y = \frac{1}{\tau}$.

Zero-order hold convolves with rectangular window \Rightarrow multiplies periodic spectrum by $\frac{\sin \frac{\omega}{2P}}{\frac{\omega}{2P}}$. Resampling aliases Ω to $\Omega_{\text{mod } \frac{2\pi}{\tau}}$.

Unit power component at ω gives alias components with total power:

$$\sin^2 \frac{\omega}{2P} \sum_{n=1}^{\infty} \left(\frac{1}{\frac{2nP\pi}{2P} + \frac{\omega}{2P}} \right)^2 + \left(\frac{1}{\frac{2nP\pi}{2P} - \frac{\omega}{2P}} \right)^2 \approx \frac{\omega^2}{4P^2} \frac{2\pi^2}{6\pi^2} = \frac{\omega^2}{12P^2}$$

For -60 dB at $\omega = \pi$, need $P = 906$ ☹



Polynomial Approximation

13: Resampling Filters

Resampling
Halfband Filters
Dyadic 1:8 Upsampler
Rational
Downsampling
Arbitrary Resampling
 Polynomial
 ▷ Approximation
Farrow Filter
Summary
MATLAB routines

Suppose $P = 50$ and $H(z)$ has order $M = 249$

$H(z)$ is lowpass filter with $\omega_0 = \frac{\pi}{50}$

Split into 50 filters of length 5: $h_p[n] = h[p + nP]$

$h_p[0]$ is the first P samples of $h[n]$

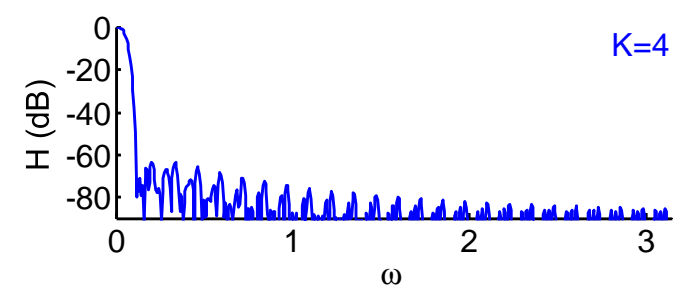
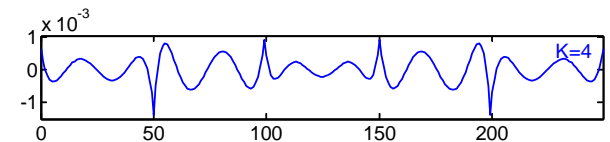
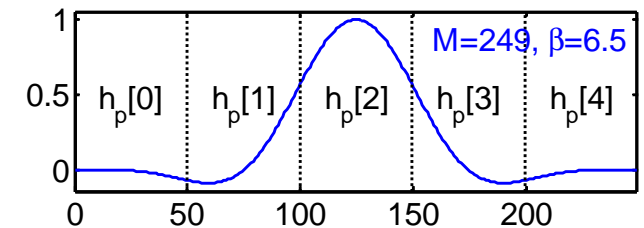
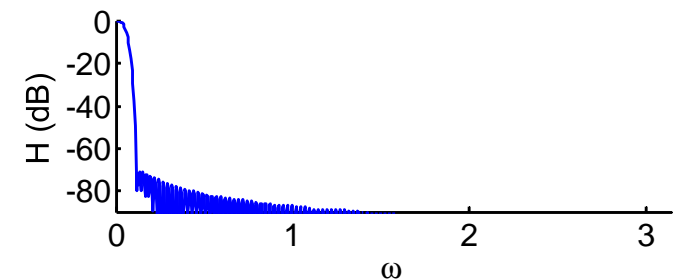
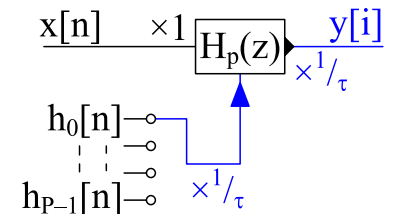
$h_p[1]$ is the next P samples, etc.

Use a polynomial of order K to approximate each segment:

$$h_p[n] \approx f_n\left(\frac{p}{P}\right)$$

$h[n]$ is smooth so errors are low even for $K = 4$ ($< 10^{-3}$)

- Resultant filter almost as good
- Only store 5 ($K + 1$) coefficients instead of $5P$
- Total coefficients: $\frac{(K+1)(M+1)}{P}$
- Can calculate $f_n(\frac{p}{P})$ for **any** p
Use $f_n(i\tau - \lfloor i\tau \rfloor)$ for $y[i]$



Farrow Filter

13: Resampling Filters

Resampling
Halfband Filters
Dyadic 1:8 Upsampler
Rational
Downsampling
Arbitrary Resampling
Polynomial
Approximation
▷ Farrow Filter
Summary
MATLAB routines

Filter coefficients depend on **fractional part** of $i\tau$:

$$\Delta[i] = i\tau - n \text{ where } n = \lfloor i\tau \rfloor$$

$$y[i] = \sum_{r=0}^R f_r(\Delta[i]) x[n-r]$$

$$\text{where } f_r(\Delta) = \sum_{k=0}^K b_k[r] \Delta^k$$

$$y[i] = \sum_{r=0}^R \sum_{k=0}^K b_k[r] \Delta[i]^k x[n-r]$$

$$= \sum_{k=0}^K \Delta[i]^k \sum_{r=0}^R b_k[r] x[n-r]$$

$$= \sum_{k=0}^K \Delta[i]^k v_k[n]$$

$$\text{where } v_k[n] = b_k[n] * x[n]$$

[like a Taylor series expansion]

Horner's Rule:

$$y[i] = v_0[n] + \Delta (v_1[n] + \Delta (v_2[n] + \Delta (\dots)))$$

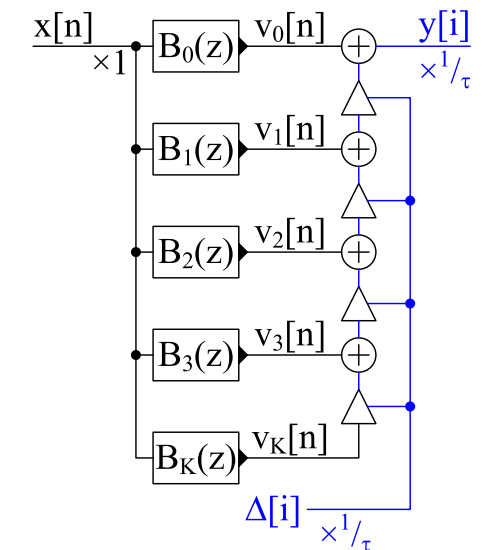
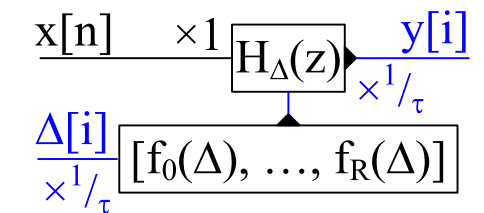
Multiplication Load:

Each $B_k(z)$ needs $R+1$ per input sample

Horner needs K per output sample

Total: $(K+1)(R+1) + \frac{K}{\tau}$

$$R+1 = \frac{M+1}{P} = 5$$



$$R+1 \approx \frac{70 \text{ [dB]}}{3.5 \Delta \omega_x}$$

$$\Delta \omega = \frac{2\alpha\pi}{\max(\tau, 1)}$$

Summary

13: Resampling Filters

Resampling
Halfband Filters
Dyadic 1:8 Upsampler
Rational
Downsampling
Arbitrary Resampling
Polynomial
Approximation
Farrow Filter
▷ Summary
MATLAB routines

- Transition band centre at ω_0 , the lower of the old and new Nyquist frequencies
 - width = $\Delta\omega = \alpha\omega_0$, typically $\alpha \approx 0.1$
- Factorizing resampling ratio can reduce computation
 - halfband filters very efficient (half the coefficients are zero)
- Rational resampling $\times \frac{P}{Q}$
 - computation $\propto \frac{\max(1, \frac{P}{Q})}{\alpha}$
 - coefficients $\propto P$
- Farrow Filter
 - filter impulse response \approx polynomial segments
 - arbitrary resampling ratios
 - computation $\times (K + 1) \approx 5$
 - coefficients independent of resolution

For further details see Mitra: 13 and Harris: 7, 8.

MATLAB routines

13: Resampling Filters

Resampling
Halfband Filters
Dyadic 1:8 Upsampler
Rational
Downsampling
Arbitrary Resampling
Polynomial
Approximation
Farrow Filter
Summary
▷ MATLAB routines

<code>gcd(p,q)</code>	Find $\alpha p + \beta q = 1$ for coprime p, q
<code>polyfit</code>	Fit a polynomial to data
<code>polyval</code>	Evaluate a polynomial

14: FM Radio
▷ Receiver

FM Radio Block
Diagram

Aliased ADC

Channel Selection

Channel Selection (1)

Channel Selection (2)

Channel Selection (3)

FM Demodulator

Differentiation Filter

Pilot tone extraction

Polyphase Pilot tone

Summary

14: FM Radio Receiver

FM Radio Block Diagram

14: FM Radio Receiver

FM Radio Block Diagram

Aliased ADC

Channel Selection

Channel Selection (1)

Channel Selection (2)

Channel Selection (3)

FM Demodulator

Differentiation Filter

Pilot tone extraction

Polyphase Pilot tone

Summary

FM spectrum: 87.5 to 108 MHz

Each channel: ± 100 kHz

Baseband signal:

Mono (L + R): ± 15 kHz

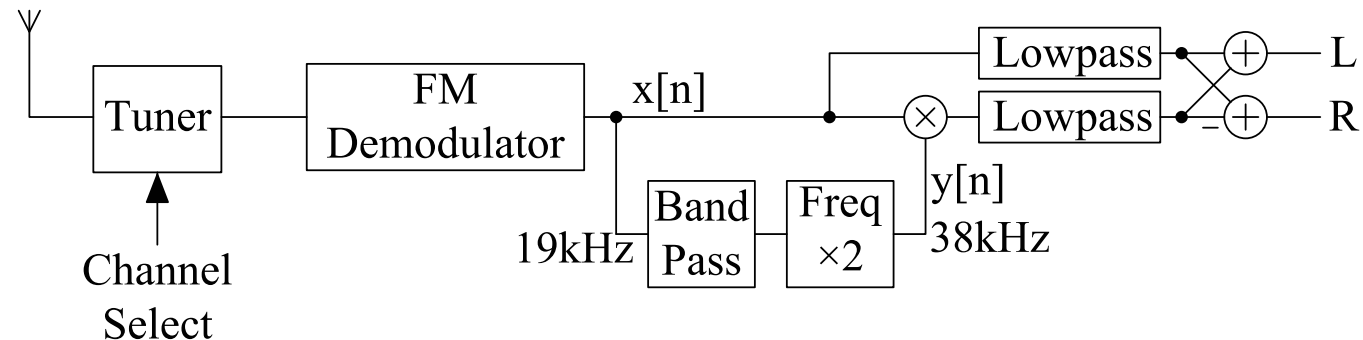
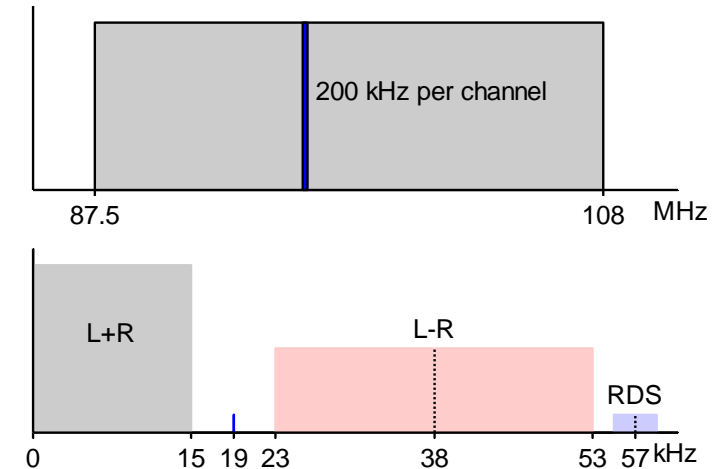
Pilot tone: 19 kHz

Stereo (L - R): 38 ± 15 kHz

RDS: 57 ± 2 kHz

FM Modulation:

Freq deviation: ± 75 kHz



L-R signal is multiplied by 38 kHz to shift it to baseband

Aliased ADC

14: FM Radio Receiver
FM Radio Block Diagram
▷ Aliased ADC
Channel Selection
Channel Selection (1)
Channel Selection (2)
Channel Selection (3)
FM Demodulator
Differentiation Filter
Pilot tone extraction
Polyphase Pilot tone
Summary

FM band: 87.5 to 108 MHz
Normally sample at $f_s > 2f$

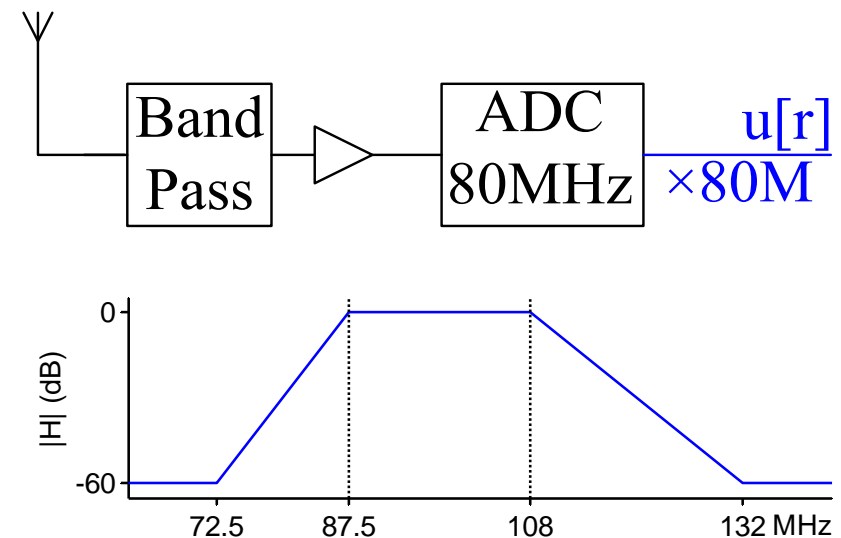
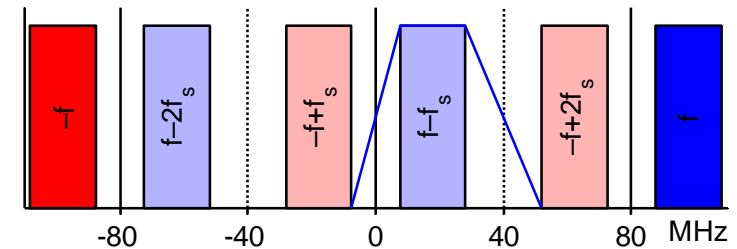
However:

$f_s = 80$ creates aliased images at intervals of f_s

–ve frequencies also generate images

We want the image between 7.5 and 28 MHz

Need an analogue bandpass filter to extract the FM band



Transition band mid-points are at f_s and $\frac{3}{2}f_s$

You can use an aliased analog-digital converter (ADC) provided that the target band fits entirely between two consecutive multiples of $\frac{1}{2}f_s$.

Lower ADC sample rate ☺. Image = undistorted frequency-shifted copy.

Channel Selection

14: FM Radio Receiver

FM Radio Block Diagram

Aliased ADC

▷ Channel Selection

Channel Selection (1)

Channel Selection (2)

Channel Selection (3)

FM Demodulator

Differentiation Filter

Pilot tone extraction

Polyphase Pilot tone

Summary

FM band shifted to 7.5 to 28 MHz (from 87.5 to 108 MHz)

We need to select a single channel 200 kHz wide

We downsample to $f_s = 400$ kHz and shift selected channel to DC.

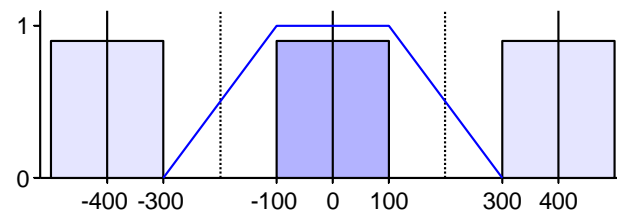
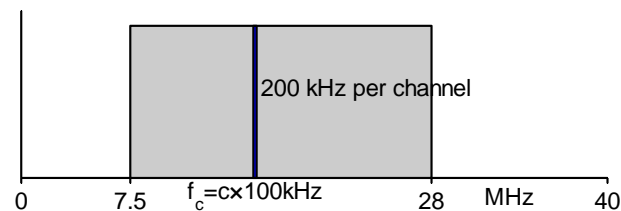
Assume channel centre frequency is $f_c = c \times 100$ kHz

We must apply a filter before downsampling to remove unwanted images

The downsampled signal is **complex** since positive and negative frequencies contain different information.

Three methods:

- 1 Freq shift, then polyphase lowpass filter
- 2 Polyphase bandpass complex filter
- 3 Polyphase bandpass real filter



Channel Selection (1)

Multiply by $e^{-j2\pi r \frac{f_c}{80M}}$ to shift channel at f_c to DC.

$$f_c = c \times 100 \text{ k} \Rightarrow \frac{f_c}{80M} = \frac{c}{800}$$

Result of multiplication is complex (thick lines on diagram)

Next, lowpass filter to $\pm 100 \text{ kHz}$

$$\Delta\omega = 2\pi \frac{200 \text{ k}}{80 \text{ M}} = 0.157$$

$$\Rightarrow M = \frac{60 \text{ dB}}{3.5\Delta\omega} = 1091$$

Finally, downsample 200:1

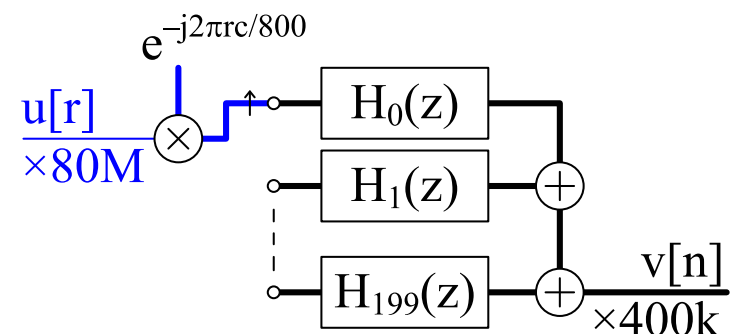
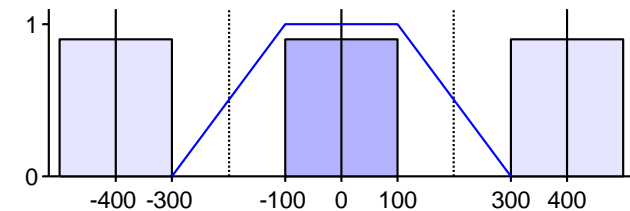
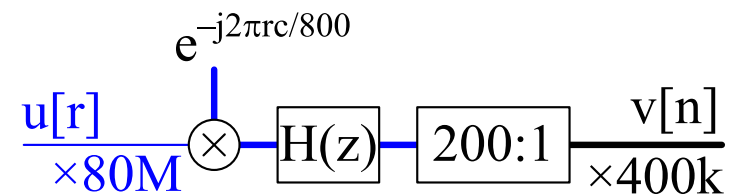
Polyphase:

$$H_p(z) \text{ has } \left\lceil \frac{1092}{200} \right\rceil = 6 \text{ taps}$$

Complex data \times Real Coefficients (needs 2 multiplies per tap)

Multiplication Load:

$$2 \times 80 \text{ MHz (freq shift)} + 12 \times 80 \text{ MHz } (H_p(z)) = 14 \times 80 \text{ MHz}$$



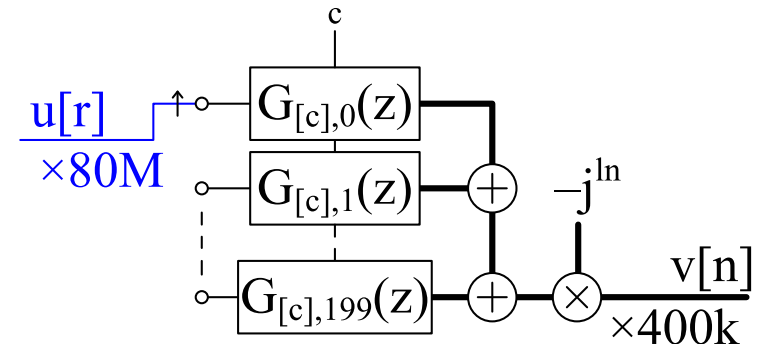
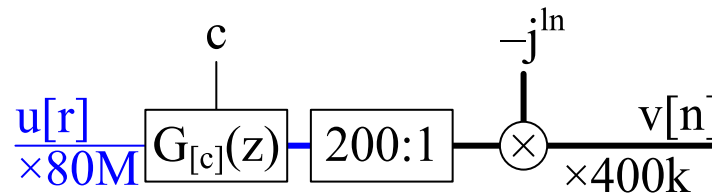
Channel Selection (2)

14: FM Radio Receiver
 FM Radio Block Diagram
 Aliased ADC
 Channel Selection
 Channel Selection (1)
 Channel Selection (2)
 Channel Selection (3)
 FM Demodulator
 Differentiation Filter
 Pilot tone extraction
 Polyphase Pilot tone
 Summary

Channel centre frequency $f_c = c \times 100$ kHz where c is an integer.

Write $c = 4k + l$

where $k = \lfloor \frac{c}{4} \rfloor$ and $l = c_{\text{mod } 4}$



We multiply $u[r]$ by $e^{-j2\pi r \frac{c}{800}}$ then convolve with $h[m]$:

$$\begin{aligned}
 v[n] &= \sum_{m=0}^M h[m] u[200n - m] e^{-j2\pi(200n-m) \frac{c}{800}} & [r = 200n] \\
 &= \sum_{m=0}^M h[m] e^{j2\pi \frac{mc}{800}} u[200n - m] e^{-j2\pi 200n \frac{4k+l}{800}} & [c = 4k + l] \\
 &= \sum_{m=0}^M g_{[c]}[m] u[200n - m] e^{-j2\pi \frac{ln}{4}} & [g_{[c]}[m] \triangleq h[m] e^{j2\pi \frac{mc}{800}}] \\
 &= (-j)^{ln} \sum_{m=0}^M g_{[c]}[m] u[200n - m] & [e^{-j2\pi \frac{ln}{4}} \text{ indep of } m]
 \end{aligned}$$

Multiplication Load for polyphase implementation:

$G_{[c],p}(z)$ has complex coefficients \times real input \Rightarrow 2 mults per tap

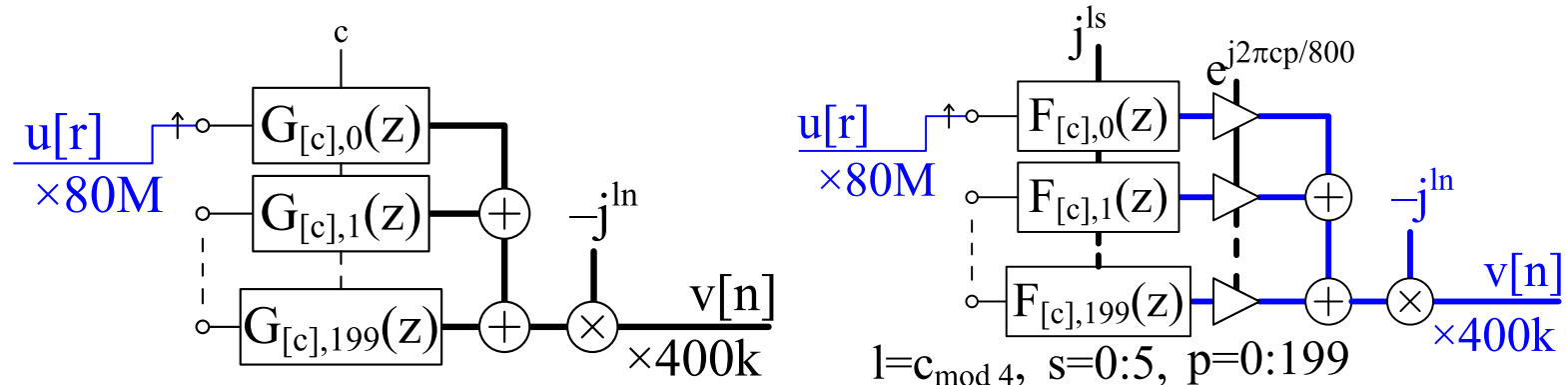
$(-j)^{ln} \in \{+1, -j, -1, +j\}$ so no actual multiplies needed

Total: 12×80 MHz ($G_{[c],p}(z)$) + 0 ($(-j)^{ln}$) = 12×80 MHz

Channel Selection (3)

- 14: FM Radio Receiver
- FM Radio Block Diagram
- Aliased ADC
- Channel Selection
- Channel Selection (1)
- Channel Selection (2)
- Channel Selection (3)
- FM Demodulator
- Differentiation Filter
- Pilot tone extraction
- Polyphase Pilot tone
- Summary

Channel frequency $f_c = c \times 100 \text{ kHz}$ where $c = 4k + l$ is an integer



$$g_{[c]}[m] = h[r]e^{j2\pi \frac{cm}{800}}$$

$$g_{[c],p}[s] = g_c[200s + p] = h[200s + p]e^{j2\pi \frac{c(200s+p)}{800}}$$

$$= h[200s + p]e^{j2\pi \frac{ls}{4}} e^{j2\pi \frac{cp}{800}} \quad [l = c_{\text{mod } 4}]$$

Define $f_{[c],p}[s] = h[200s + p]e^{j2\pi \frac{ls}{4}} = j^{ls} h[200s + p]$

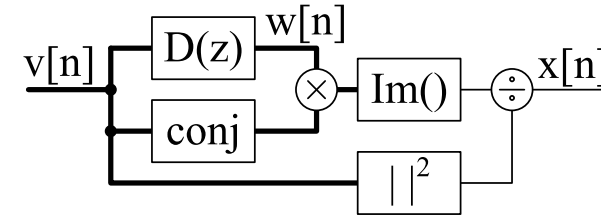
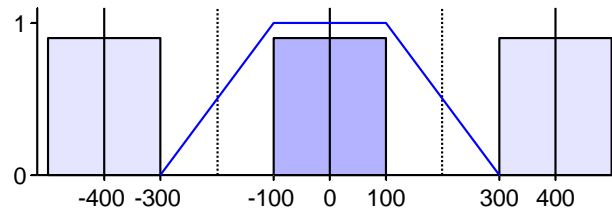
Although $f_{[c],p}[s]$ is complex it requires only one multiplication per tap because each tap is either purely real or purely imaginary.

Multiplication Load:

$$6 \times 80 \text{ MHz } (F_p(z)) + 4 \times 80 \text{ MHz } (\times e^{j2\pi \frac{cp}{800}}) = 10 \times 80 \text{ MHz}$$

FM Demodulator

- 14: FM Radio Receiver
- FM Radio Block Diagram
- Aliased ADC
- Channel Selection
- Channel Selection (1)
- Channel Selection (2)
- Channel Selection (3)
- FM Demodulator
- Differentiation Filter
- Pilot tone extraction
- Polyphase Pilot tone
- Summary



Complex FM signal centred at DC: $v(t) = |v(t)|e^{j\phi(t)}$

We know that $\log v = \log |v| + j\phi$

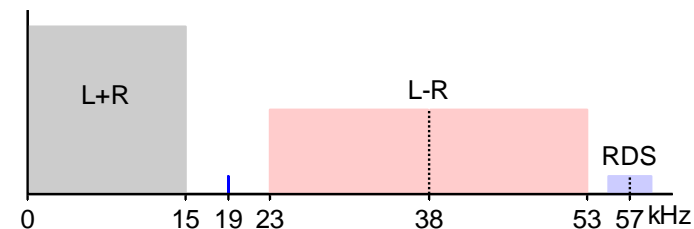
The instantaneous frequency of $v(t)$ is $\frac{d\phi}{dt}$.

We need to calculate $\frac{d\phi}{dt} = \frac{d\Im(\log v)}{dt} = \Im\left(\frac{1}{v} \frac{dv}{dt}\right) = \frac{1}{|v|^2} \Im\left(v^* \frac{dv}{dt}\right)$

We need:

- (1) Differentiation filter, $D(z)$
- (2) Complex multiply, $w[n] \times v^*[n]$ (only need \Im part)
- (3) Real Divide by $|v|^2$

$x[n]$ is baseband signal (real):



Differentiation Filter

14: FM Radio Receiver

FM Radio Block Diagram

Aliased ADC

Channel Selection

Channel Selection (1)

Channel Selection (2)

Channel Selection (3)

FM Demodulator

Differentiation

▷ Filter

Pilot tone extraction

Polyphase Pilot tone

Summary

Window design method:

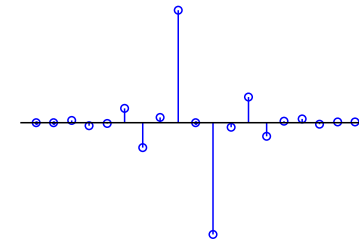
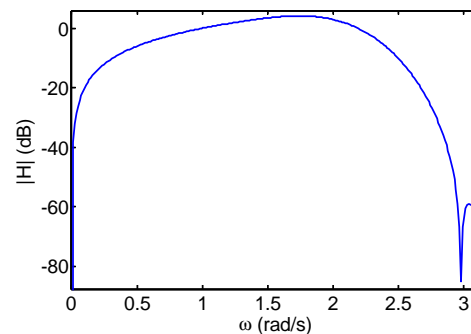
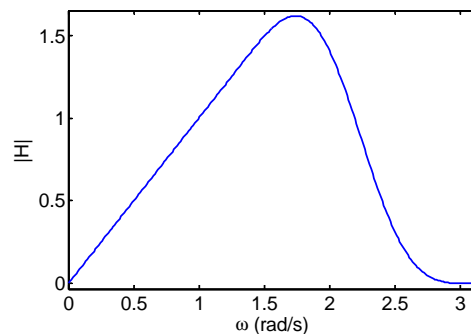
- (1) calculate $d[n]$ for the ideal filter
- (2) multiply by a window to give finite support

$$\underline{x[n]} \boxed{D(z)} \underline{w[n]}$$

Differentiation of a complex tone: $\frac{d}{dt}e^{j\omega t} = j\omega e^{j\omega t}$

We want a frequency response: $D(e^{j\omega}) = \begin{cases} j\omega & |\omega| \leq \omega_0 \\ 0 & |\omega| > \omega_0 \end{cases}$

$$\text{Hence } d[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} j\omega e^{j\omega n} d\omega = \frac{n\omega_0 \cos n\omega_0 - \sin n\omega_0}{\pi n^2}$$



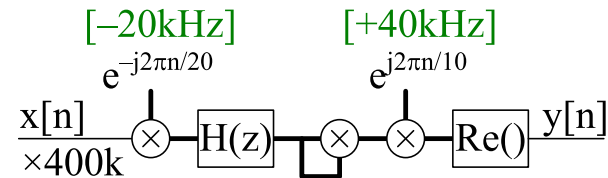
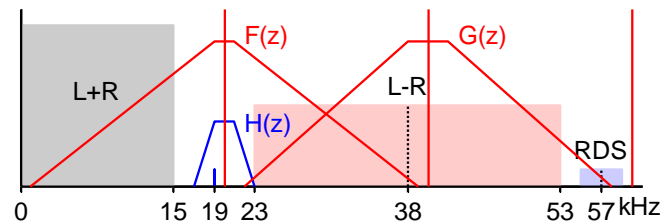
Using $\omega_0 = 2.2$, $M = 18$, Kaiser window, $\beta = 7$:

Near perfect differentiation for $\omega \leq 1.6$

Broad transition region allows shorter filter

Pilot tone extraction

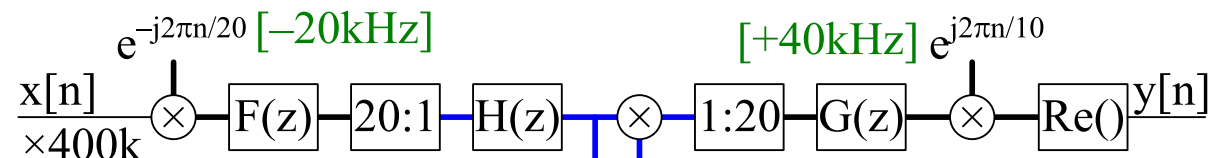
- 14: FM Radio Receiver
- FM Radio Block Diagram
- Aliased ADC
- Channel Selection
- Channel Selection (1)
- Channel Selection (2)
- Channel Selection (3)
- FM Demodulator
- Differentiation Filter
- Pilot tone
- ▷ extraction
- Polyphase Pilot tone
- Summary



Aim: extract 19 kHz pilot tone, double freq \rightarrow real 38 kHz tone.

- (1) shift spectrum down by 20 kHz: multiply by $e^{-j2\pi n \frac{20k}{400k}}$
- (2) low pass filter to ± 1 kHz to extract complex pilot at -1 kHz: $H(z)$
- (3) square to double frequency to -2 kHz
- (4) shift spectrum up by 40 kHz: multiply by $e^{+j2\pi n \frac{40k}{400k}}$
- (5) Take real part

More efficient to do low pass filtering at a low sample rate:

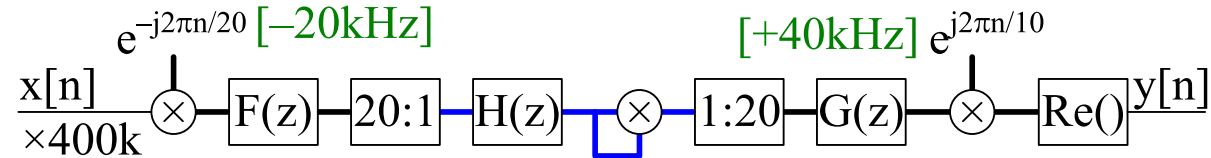


Transition bands:

$F(z)$: $1 \rightarrow 19$ kHz, $H(z)$: $1 \rightarrow 3$ kHz, $G(z)$: $2 \rightarrow 18$ kHz

Polyphase Pilot tone

- 14: FM Radio Receiver
- FM Radio Block Diagram
- Aliased ADC
- Channel Selection
- Channel Selection (1)
- Channel Selection (2)
- Channel Selection (3)
- FM Demodulator
- Differentiation Filter
- Pilot tone extraction
- Polyphase Pilot tone
- Summary

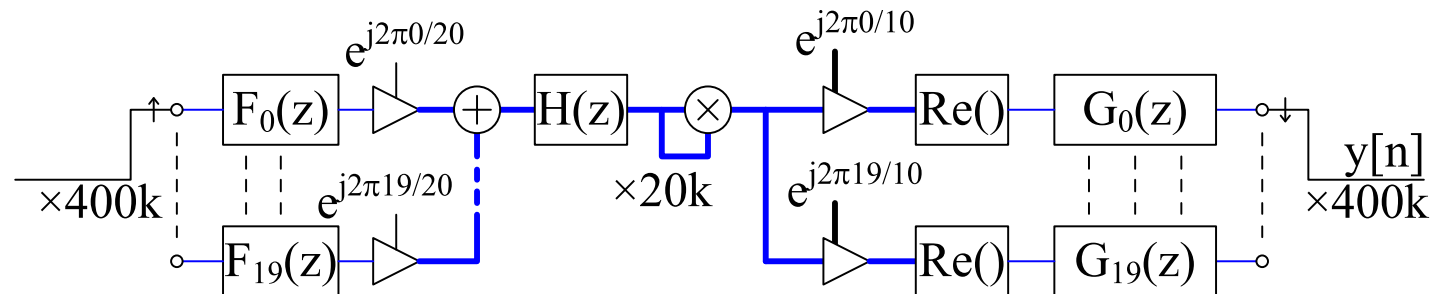


Anti-alias filter: $F(z)$

Each branch, $F_p(z)$, gets every 20^{th} sample and an identical $e^{j2\pi \frac{n}{20}}$
 So $F_p(z)$ can filter a real signal and then multiply by fixed $e^{j2\pi \frac{p}{20}}$

Anti-image filter: $G(z)$

Each branch, $G_p(z)$, multiplied by identical $e^{j2\pi \frac{n}{10}}$
 So $G_p(z)$ can filter a real signal



Multiples:

F and G each: $(3 + 2) \times 400 \text{ kHz}$, $H + x^2$: $(2 \times 30 + 4) \times 20 \text{ kHz}$

Total: $13.2 \times 400 \text{ kHz}$ [Full-rate bandpass $H(z)$ needs $273 \times 400 \text{ kHz}$]

Summary

14: FM Radio Receiver

FM Radio Block Diagram

Aliased ADC

Channel Selection

Channel Selection (1)

Channel Selection (2)

Channel Selection (3)

FM Demodulator

Differentiation Filter

Pilot tone extraction

Polyphase Pilot tone

▷ Summary

- Aliased ADC allows sampling below the Nyquist frequency
 - Only works because the wanted signal fits entirely within a Nyquist band image
- Polyphase filter can be combined with complex multiplications to select the desired image
 - subsequent multiplication by $-j^{ln}$ shifts by the desired multiple of $\frac{1}{4}$ sample rate
No actual multiplications required
- FM demodulation uses a differentiation filter to calculate $\frac{d\phi}{dt}$
- Pilot tone bandpass filter has narrow bandwidth so better done at a low sample rate
 - double the frequency of a complex tone by squaring it
 - Select the the first image of 20 kHz when downsampling and the second image when upsampling

This example is taken from Harris: 13.

▷ **15: Subband
Processing**

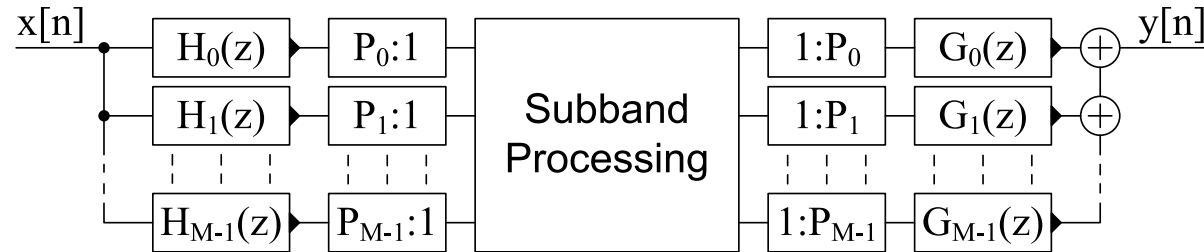
Subband processing
2-band Filterbank
Perfect
Reconstruction
Quadrature Mirror
Filterbank (QMF)
Polyphase QMF
QMF Options
Linear Phase QMF
IIR Allpass QMF
Tree-structured
filterbanks
Summary
Merry Xmas

15: Subband Processing

Subband processing

15: Subband Processing

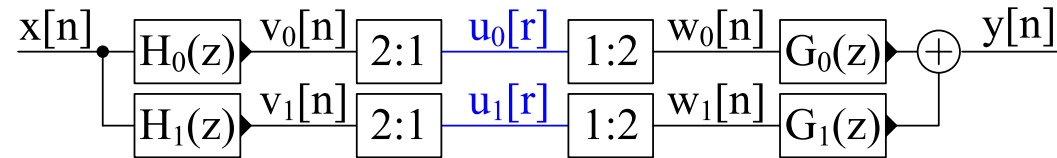
Subband
processing
2-band Filterbank
Perfect
Reconstruction
Quadrature Mirror
Filterbank (QMF)
Polyphase QMF
QMF Options
Linear Phase QMF
IIR Allpass QMF
Tree-structured
filterbanks
Summary
Merry Xmas



- The $H_m(z)$ are bandpass *analysis filters* and divide $x[n]$ into frequency bands
- Subband processing often processes frequency bands independently
- The $G_m(z)$ are *synthesis filters* and together reconstruct the output
- The $H_m(z)$ outputs are bandlimited and so can be subsampled without loss of information
 - Sample rate multiplied overall by $\sum \frac{1}{P_i}$
 - $\sum \frac{1}{P_i} = 1 \Rightarrow$ *critically sampled*: good for coding
 - $\sum \frac{1}{P_i} > 1 \Rightarrow$ *oversampled*: more flexible
- **Goals:**
 - (a) good frequency selectivity in $H_m(z)$
 - (b) *perfect reconstruction*: $y[n] = x[n - d]$ if no processing
- **Benefits:** Lower computation, faster convergence if adaptive

2-band Filterbank

15: Subband Processing
 Subband processing
 ▷ 2-band Filterbank
 Perfect Reconstruction
 Quadrature Mirror Filterbank (QMF)
 Polyphase QMF
 QMF Options
 Linear Phase QMF
 IIR Allpass QMF
 Tree-structured filterbanks
 Summary
 Merry Xmas



$$V_m(z) = H_m(z)X(z)$$

$$U_m(z) = \frac{1}{K} \sum_{k=0}^{K-1} V_m(e^{-j\frac{2\pi k}{K}} z^{\frac{1}{K}}) = \frac{1}{2} \left\{ V_m\left(z^{\frac{1}{2}}\right) + V_m\left(-z^{\frac{1}{2}}\right) \right\}$$

$$W_m(z) = U_m(z^2) = \frac{1}{2} \{ V_m(z) + V_m(-z) \} \quad [K = 2]$$

$$= \frac{1}{2} \{ H_m(z)X(z) + H_m(-z)X(-z) \}$$

$$Y(z) = \begin{bmatrix} G_0(z) & G_1(z) \end{bmatrix} \begin{bmatrix} W_0(z) \\ W_1(z) \end{bmatrix}$$

$$= \frac{1}{2} \begin{bmatrix} G_0(z) & G_1(z) \end{bmatrix} \begin{bmatrix} H_0(z) & H_0(-z) \\ H_1(z) & H_1(-z) \end{bmatrix} \begin{bmatrix} X(z) \\ X(-z) \end{bmatrix}$$

$$= \begin{bmatrix} T(z) & A(z) \end{bmatrix} \begin{bmatrix} X(z) \\ X(-z) \end{bmatrix} \quad [A(z)X(-z) \text{ is aliased term}]$$

We want (a) $T(z) = \frac{1}{2} \{ H_0(z)G_0(z) + H_1(z)G_1(z) \} = z^{-d}$
 and (b) $A(z) = \frac{1}{2} \{ H_0(-z)G_0(z) + H_1(-z)G_1(z) \} = 0$

Perfect Reconstruction

For perfect reconstruction without aliasing, we require

$$\frac{1}{2} \begin{bmatrix} G_0(z) & G_1(z) \end{bmatrix} \begin{bmatrix} H_0(z) & H_0(-z) \\ H_1(z) & H_1(-z) \end{bmatrix} = \begin{bmatrix} z^{-d} & 0 \end{bmatrix}$$

Hence:

$$\begin{aligned} \begin{bmatrix} G_0(z) \\ G_1(z) \end{bmatrix} &= \begin{bmatrix} H_0(z) & H_1(z) \\ H_0(-z) & H_1(-z) \end{bmatrix}^{-1} \begin{bmatrix} 2z^{-d} \\ 0 \end{bmatrix} \\ &= \frac{2z^{-d}}{H_0(z)H_1(-z) - H_0(-z)H_1(z)} \begin{bmatrix} H_1(-z) & -H_1(z) \\ -H_0(-z) & H_0(z) \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} \\ &= \frac{2z^{-d}}{H_0(z)H_1(-z) - H_0(-z)H_1(z)} \begin{bmatrix} H_1(-z) \\ -H_0(-z) \end{bmatrix} \end{aligned}$$

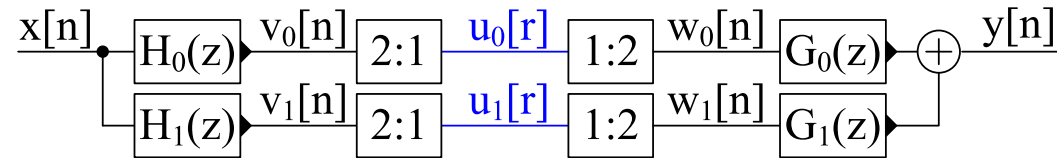
For all filters to be FIR, we need the denominator to be:

$$H_0(z)H_1(-z) - H_0(-z)H_1(z) = cz^{-k} \text{ from which:}$$

$$\begin{bmatrix} G_0(z) \\ G_1(z) \end{bmatrix} = \frac{2}{c} z^{k-d} \begin{bmatrix} H_1(-z) \\ -H_0(-z) \end{bmatrix} \stackrel{d=k}{=} \frac{2}{c} \begin{bmatrix} H_1(-z) \\ -H_0(-z) \end{bmatrix}$$

Quadrature Mirror Filterbank (QMF)

15: Subband Processing
 Subband processing
 2-band Filterbank
 Perfect Reconstruction
 Quadrature Mirror
 ▷ Filterbank (QMF)
 Polyphase QMF
 QMF Options
 Linear Phase QMF
 IIR Allpass QMF
 Tree-structured filterbanks
 Summary
 Merry Xmas



QMF satisfies:

- (a) $H_0(z)$ is causal and real
- (b) $H_1(z) = H_0(-z)$: i.e. $H_0(e^{j\omega})$ is reflected around $\omega = \frac{\pi}{2}$
- (c) $G_0(z) = 2H_1(-z) = 2H_0(z)$
- (d) $G_1(z) = -2H_0(-z) = -2H_1(z)$

QMF is alias-free:

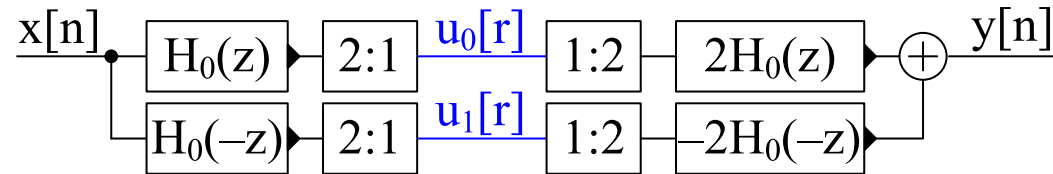
$$\begin{aligned} A(z) &= \frac{1}{2} \{H_0(-z)G_0(z) + H_1(-z)G_1(z)\} \\ &= \frac{1}{2} \{2H_1(z)H_0(z) - 2H_0(z)H_1(z)\} = 0 \end{aligned}$$

QMF Transfer Function:

$$\begin{aligned} T(z) &= \frac{1}{2} \{H_0(z)G_0(z) + H_1(z)G_1(z)\} \\ &= H_0^2(z) - H_1^2(z) = H_0^2(z) - H_0^2(-z) \end{aligned}$$

Polyphase QMF

15: Subband Processing
 Subband processing
 2-band Filterbank
 Perfect Reconstruction
 Quadrature Mirror Filterbank (QMF)
 ▷ Polyphase QMF
 QMF Options
 Linear Phase QMF
 IIR Allpass QMF
 Tree-structured filterbanks
 Summary
 Merry Xmas



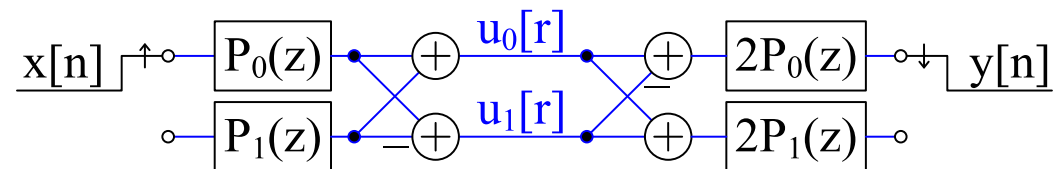
Polyphase decomposition:

$$H_0(z) = P_0(z^2) + z^{-1}P_1(z^2)$$

$$H_1(z) = H_0(-z) = P_0(z^2) - z^{-1}P_1(z^2)$$

$$G_0(z) = 2H_0(z) = 2P_0(z^2) + 2z^{-1}P_1(z^2)$$

$$G_1(z) = -2H_0(-z) = -2P_0(z^2) + 2z^{-1}P_1(z^2)$$



Transfer Function:

$$T(z) = H_0^2(z) - H_1^2(z) = 4z^{-1}P_0(z^2)P_1(z^2)$$

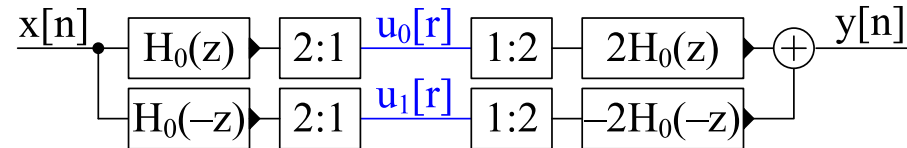
$$T(z) = z^{-d} \Rightarrow P_0(z) = a_0 z^{-k}, P_1(z) = a_1 z^{k+1-d}$$

$\Rightarrow H_0(z)$ has only two non-zero taps \Rightarrow poor freq selectivity

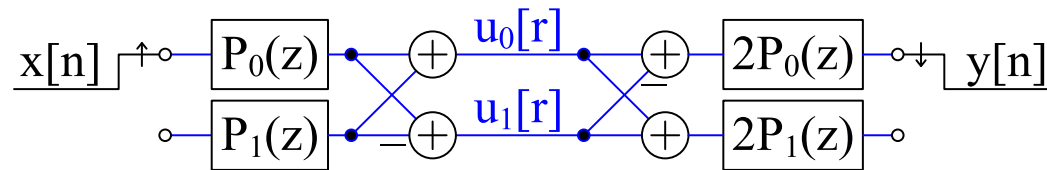
\therefore Perfect reconstruction QMF filterbanks cannot have good freq selectivity

QMF Options

- 15: Subband Processing
- Subband processing
- 2-band Filterbank
- Perfect Reconstruction
- Quadrature Mirror Filterbank (QMF)
- Polyphase QMF
- ▷ QMF Options
- Linear Phase QMF
- IIR Allpass QMF
- Tree-structured filterbanks
- Summary
- Merry Xmas



Polyphase decomposition:



$A(z) = 0 \Rightarrow$ no alias term

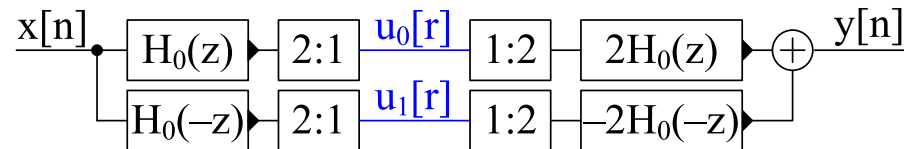
$$T(z) = H_0^2(z) - H_1^2(z) = H_0^2(z) - H_0^2(-z) = 4z^{-1}P_0(z^2)P_1(z^2)$$

Options:

- Perfect Reconstruction $T(z) = z^{-d} \Rightarrow H_0(z)$ is a bad filter.
- $T(z)$ is Linear Phase FIR
 \Rightarrow Tradeoff: $|T(e^{j\omega})| \approx 1$ versus $H_0(z)$ stopband attenuation
- $T(z)$ is Allpass IIR : $H_0(z)$ can be Butterworth or Elliptic filter
 \Rightarrow Tradeoff: $H_0(z)$ stopband attenuation versus $\angle T(e^{j\omega}) \propto \omega$ approx

Linear Phase QMF

15: Subband Processing
 Subband processing
 2-band Filterbank
 Perfect Reconstruction
 Quadrature Mirror Filterbank (QMF)
 Polyphase QMF
 QMF Options
 ▷ Linear Phase QMF
 IIR Allpass QMF
 Tree-structured filterbanks
 Summary
 Merry Xmas



$$T(z) \approx 1$$

$$H_0(z) \text{ order } M, \text{ linear phase} \Rightarrow H_0(e^{j\omega}) = \pm e^{-j\omega \frac{M}{2}} |H_0(e^{j\omega})|$$

$$\begin{aligned} T(e^{j\omega}) &= H_0^2(e^{j\omega}) - H_1^2(e^{j\omega}) = H_0^2(e^{j\omega}) - H_0^2(-e^{j\omega}) \\ &= e^{-j\omega M} |H_0(e^{j\omega})|^2 - e^{-j(\omega-\pi)M} |H_0(e^{j(\omega-\pi)})|^2 \\ &= e^{-j\omega M} \left(|H_0(e^{j\omega})|^2 - (-1)^M |H_0(e^{j(\pi-\omega)})|^2 \right) \end{aligned}$$

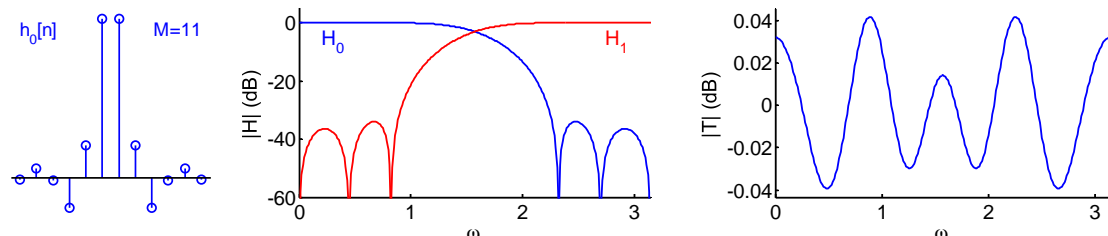
$$M \text{ even} \Rightarrow T(e^{j\frac{\pi}{2}}) = 0 \odot \text{ so choose } M \text{ odd} \Rightarrow -(-1)^M = +1$$

Select $h_0[n]$ by numerical iteration to minimize

$$\alpha \int_{\frac{\pi}{2}+\Delta}^{\pi} |H_0(e^{j\omega})|^2 d\omega + (1-\alpha) \int_0^{\pi} (|T(e^{j\omega})| - 1)^2 d\omega$$

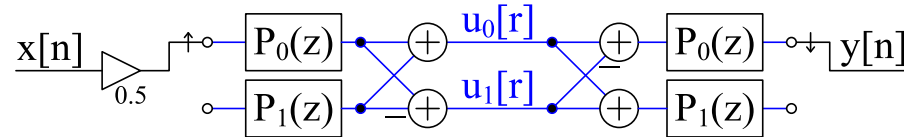
$\alpha \rightarrow$ balance between $H_0(z)$ being lowpass and $T(e^{j\omega}) \approx 1$

Johnston filter
 ($M = 11$):



IIR Allpass QMF

15: Subband Processing
 Subband processing
 2-band Filterbank
 Perfect Reconstruction
 Quadrature Mirror Filterbank (QMF)
 Polyphase QMF
 QMF Options
 Linear Phase QMF
 ▷ IIR Allpass QMF
 Tree-structured filterbanks
 Summary
 Merry Xmas



$$|T(z)| = 1$$

Choose $P_0(z)$ and $P_1(z)$ to be allpass IIR filters:

$$H_{0,1}(z) = \frac{1}{2} (P_0(z^2) \pm z^{-1}P_1(z^2)), \quad G_{0,1}(z) = \pm 2H_{0,1}(z)$$

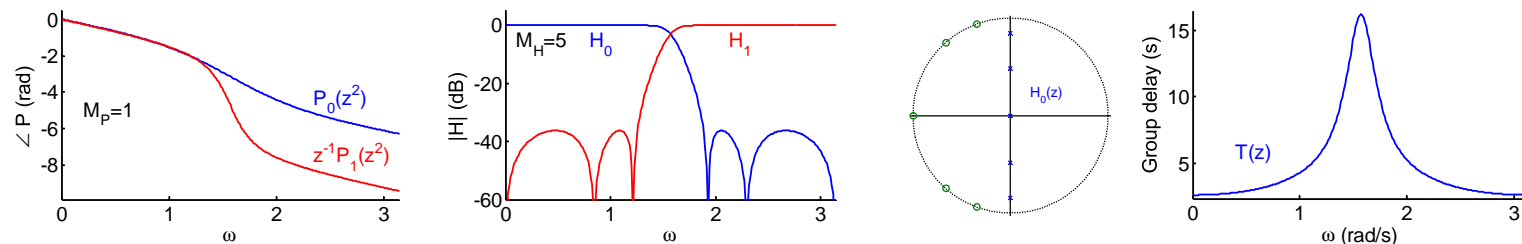
$A(z) = 0 \Rightarrow$ **No aliasing**

$T(z) = H_0^2 - H_1^2 = \dots = z^{-1}P_0(z^2)P_1(z^2)$ is an **allpass filter**.

$H_0(z)$ and $H_1(z)$ are **power complementary**:

$$\begin{aligned} |H_0(e^{j\omega})|^2 + |H_1(e^{j\omega})|^2 &= H_0(e^{j\omega})H_0(e^{-j\omega}) + H_1(e^{j\omega})H_1(e^{-j\omega}) \\ &= \dots = \frac{1}{2} |P_0(e^{j\omega})|^2 + \frac{1}{2} |P_1(e^{j\omega})|^2 = 1 \end{aligned}$$

$H_0(z)$ can be chosen to be an odd order **Butterworth** or **Elliptic** filter:



Phase cancellation: $\angle z^{-1}P_1 = \angle P_0 + \pi$; Ripples in H_0 and H_1 cancel.

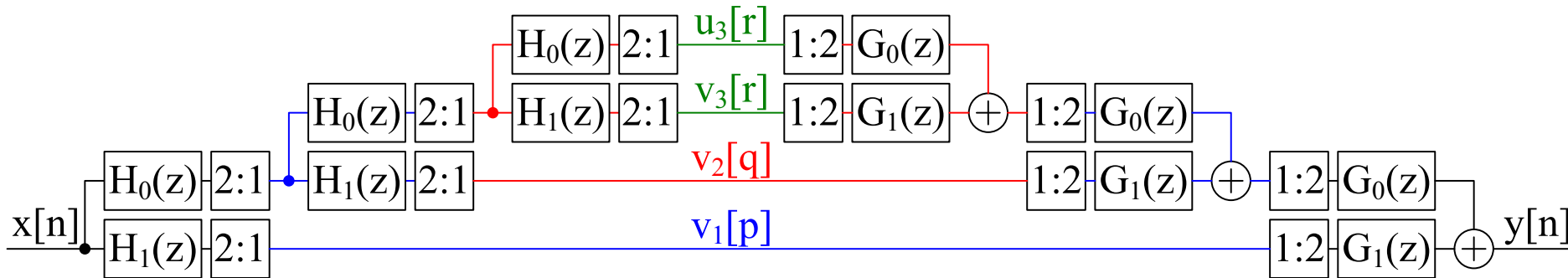
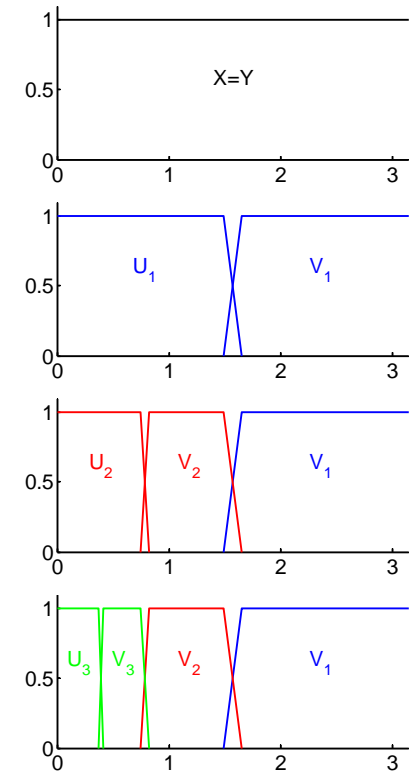
Tree-structured filterbanks

A *half-band filterbank* divides the full band into two equal halves.

You can repeat the process on either or both of the signals $u_1[p]$ and $v_1[p]$.

Dividing the lower band in half repeatedly results in an *octave band filterbank*. Each subband occupies one octave (= a factor of 2 in frequency).

The properties “*perfect reconstruction*” and “*allpass*” are preserved by the iteration.



Summary

- 15: Subband Processing
- Subband processing
- 2-band Filterbank
- Perfect Reconstruction
- Quadrature Mirror Filterbank (QMF)
- Polyphase QMF
- QMF Options
- Linear Phase QMF
- IIR Allpass QMF
- Tree-structured filterbanks
- ▷ Summary
- Merry Xmas

- **Half-band filterbank:**
 - Reconstructed output is $T(z)X(z) + A(z)X(-z)$
 - Unwanted alias term is $A(z)X(-z)$
- **Perfect reconstruction:** imposes strong constraints on analysis filters $H_i(z)$ and synthesis filters $G_i(z)$.
- **Quadrature Mirror Filterbank (QMF)** adds an additional symmetry constraint $H_1(z) = H_0(-z)$.
 - Perfect reconstruction now impossible except for trivial case.
 - Neat polyphase implementation with $A(z) = 0$
 - Johnston filters: Linear phase with $T(z) \approx 1$
 - Allpass filters: Elliptic or Butterworth with $|T(z)| = 1$
- Can iterate to form a tree structure with equal or unequal bandwidths.

See Mitra chapter 14 (which also includes some perfect reconstruction designs).

Merry Xmas



FORMULA SHEET AVAILABLE IN EXAM

The following formulae will be available in the exam:

Notation

- All signals and filter coefficients are real-valued unless explicitly noted otherwise.
- Unless otherwise specified, upper and lower case letters are used for sequences and their z -transforms. The signal at a block diagram node V is $v[n]$ and its z -transform is $V(z)$.
- $x[n] = [a, b, c, d, e, f]$ means that $x[0] = a, \dots, x[5] = f$ and that $x[n] = 0$ outside this range.
- $\Re(z)$, $\Im(z)$, z^* , $|z|$ and $\angle z$ denote respectively the real part, imaginary part, complex conjugate, magnitude and argument of a complex number z .

Abbreviations

BIBO	Bounded Input, Bounded Output
CTFT	Continuous-Time Fourier Transform
DCT	Discrete Cosine Transform
DFT	Discrete Fourier Transform
DTFT	Discrete-Time Fourier Transform
LTI	Linear Time-Invariant
MDCT	Modified Discrete Cosine Transform
SNR	Signal-to-Noise Ratio

Standard Sequences

- $\delta[n] = 1$ for $n = 0$ and 0 otherwise.
- $\delta_{\text{condition}}[n] = 1$ whenever "condition" is true and 0 otherwise.
- $u[n] = 1$ for $n \geq 0$ and 0 otherwise.

Geometric Progression

- $\sum_{n=0}^r \alpha^n z^{-n} = \frac{1 - \alpha^{r+1} z^{-r-1}}{1 - \alpha z^{-1}}$ or, more generally, $\sum_{n=q}^r \alpha^n z^{-n} = \frac{\alpha^q z^{-q} - \alpha^{r+1} z^{-r-1}}{1 - \alpha z^{-1}}$

Forward and Inverse Transforms

z:	$X(z) = \sum_{-\infty}^{\infty} x[n]z^{-n}$	$x[n] = \frac{1}{2\pi j} \oint X(z)z^{n-1}dz$
CTFT:	$X(j\Omega) = \int_{-\infty}^{\infty} x(t)e^{-j\Omega t}dt$	$x(t) = \frac{1}{2\pi} \int_{-\infty}^{\infty} X(j\Omega)e^{j\Omega t}d\Omega$
DTFT:	$X(e^{j\omega}) = \sum_{-\infty}^{\infty} x[n]e^{-j\omega n}$	$x[n] = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\omega})e^{j\omega n}d\omega$
DFT:	$X[k] = \sum_{n=0}^{N-1} x[n]e^{-j2\pi \frac{kn}{N}}$	$x[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k]e^{j2\pi \frac{kn}{N}}$
DCT:	$X[k] = \sum_{n=0}^{N-1} x[n] \cos \frac{2\pi(2n+1)k}{4N}$	$x[n] = \frac{X[0]}{N} + \frac{2}{N} \sum_{k=1}^{N-1} X[k] \cos \frac{2\pi(2n+1)k}{4N}$
MDCT:	$X[k] = \sum_{n=0}^{2N-1} x[n] \cos \frac{2\pi(2n+1+N)(2k+1)}{8N}$	$y[n] = \frac{1}{N} \sum_{k=0}^{N-1} X[k] \cos \frac{2\pi(2n+1+N)(2k+1)}{8N}$

Convolution

DTFT:	$v[n] = x[n] * y[n] = \sum_{r=-\infty}^{\infty} x[r]y[n-r]$	\Leftrightarrow	$V(e^{j\omega}) = X(e^{j\omega})Y(e^{j\omega})$
	$v[n] = x[n]y[n]$	\Leftrightarrow	$V(e^{j\omega}) = \frac{1}{2\pi} X(e^{j\omega}) \otimes Y(e^{j\omega}) = \frac{1}{2\pi} \int_{-\pi}^{\pi} X(e^{j\theta})Y(e^{j(\omega-\theta)})d\theta$
DFT:	$v[n] = x[n] \otimes_N y[n] = \sum_{r=0}^{N-1} x[r]y[(n-r) \bmod N]$	\Leftrightarrow	$V[k] = X[k]Y[k]$
	$v[n] = x[n]y[n]$	\Leftrightarrow	$V[k] = \frac{1}{N} X[k] \otimes_N Y[k] = \frac{1}{N} \sum_{r=0}^{N-1} X[r]Y[(k-r) \bmod N]$

Group Delay

The group delay of a filter, $H(z)$, is $\tau_H(e^{j\omega}) = -\frac{d\angle H(e^{j\omega})}{d\omega} = \Re \left(\frac{-z}{H(z)} \frac{dH(z)}{dz} \right) \Big|_{z=e^{j\omega}} = \Re \left(\frac{\mathcal{F}(nh[n])}{\mathcal{F}(h[n])} \right)$ where $\mathcal{F}(\cdot)$ denotes the DTFT.

Order Estimation for FIR Filters

Three increasingly sophisticated formulae for estimating the minimum order of an FIR filter with unity gain passbands:

1. $M \approx \frac{a}{3.5\Delta\omega}$
2. $M \approx \frac{a-8}{2.2\Delta\omega}$
3. $M \approx \frac{a-1.2-20\log_{10} b}{4.6\Delta\omega}$

where a = stop band attenuation in dB, b = peak-to-peak passband ripple in dB and $\Delta\omega$ = width of smallest transition band in normalized rad/s.

z-plane Transformations

A lowpass filter, $H(z)$, with cutoff frequency ω_0 may be transformed into the filter $H(\hat{z})$ as follows:

Target $H(\hat{z})$	Substitute	Parameters
Lowpass $\hat{\omega} < \hat{\omega}_1$	$z^{-1} = \frac{\hat{z}^{-1} - \lambda}{1 - \lambda \hat{z}^{-1}}$	$\lambda = \frac{\sin\left(\frac{\omega_0 - \hat{\omega}_1}{2}\right)}{\sin\left(\frac{\omega_0 + \hat{\omega}_1}{2}\right)}$
Highpass $\hat{\omega} > \hat{\omega}_1$	$z^{-1} = -\frac{\hat{z}^{-1} + \lambda}{1 + \lambda \hat{z}^{-1}}$	$\lambda = \frac{\cos\left(\frac{\omega_0 + \hat{\omega}_1}{2}\right)}{\cos\left(\frac{\omega_0 - \hat{\omega}_1}{2}\right)}$
Bandpass $\hat{\omega}_1 < \hat{\omega} < \hat{\omega}_2$	$z^{-1} = -\frac{(\rho-1) - 2\lambda\rho\hat{z}^{-1} + (\rho+1)\hat{z}^{-2}}{(\rho+1) - 2\lambda\rho\hat{z}^{-1} + (\rho-1)\hat{z}^{-2}}$	$\lambda = \frac{\cos\left(\frac{\hat{\omega}_2 + \hat{\omega}_1}{2}\right)}{\cos\left(\frac{\hat{\omega}_2 - \hat{\omega}_1}{2}\right)}, \rho = \cot\left(\frac{\hat{\omega}_2 - \hat{\omega}_1}{2}\right) \tan\left(\frac{\omega_0}{2}\right)$
Bandstop $\hat{\omega}_1 \not< \hat{\omega} \not< \hat{\omega}_2$	$z^{-1} = \frac{(1-\rho) - 2\lambda\hat{z}^{-1} + (\rho+1)\hat{z}^{-2}}{(\rho+1) - 2\lambda\hat{z}^{-1} + (1-\rho)\hat{z}^{-2}}$	$\lambda = \frac{\cos\left(\frac{\hat{\omega}_2 + \hat{\omega}_1}{2}\right)}{\cos\left(\frac{\hat{\omega}_2 - \hat{\omega}_1}{2}\right)}, \rho = \tan\left(\frac{\hat{\omega}_2 - \hat{\omega}_1}{2}\right) \tan\left(\frac{\omega_0}{2}\right)$