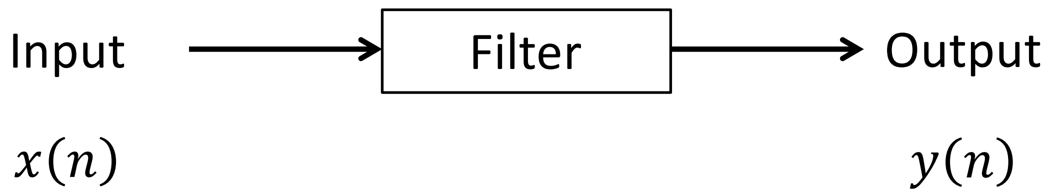
CTP431- Music and Audio Computing
Audio Signal Processing (Part #1)

Graduate School of Culture Technology
KAIST
Juhan Nam

Types of Audio Signal Processing

- Amplitude
  - Gain, fade in/out, automation curve, compressor
- Timbre
  - EQ, distortion, modulation, chorus, flanger
- Spatial effect
  - Delay, reverberation
- Pitch
  - Pitch shifting (e.g. auto-tune)
- Duration
  - Time stretching
- Playback Rate Conversion (resampling)
  - pitch-shifting /time-stretching / timbre change

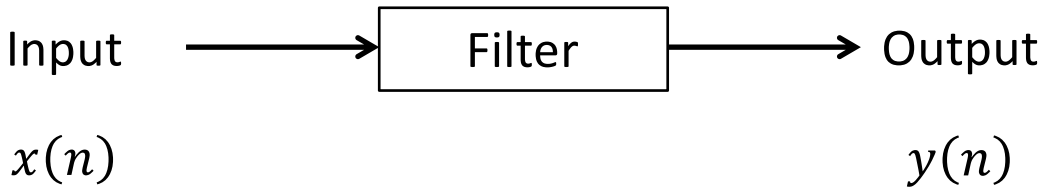
Filter



- A system that takes signals and modifies them in some way

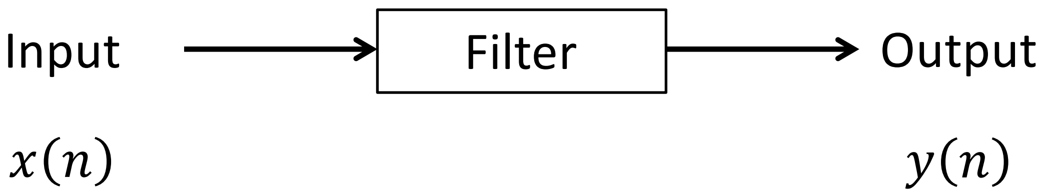
- In particular, we are interested in **digital filters** that takes discrete number sequences as input and output

Basic Operations in Digital Filters



- Multiplication: $y(n) = b_0 \cdot x(n)$
- Delaying: $y(n) = x(n-1)$
- Summation: $y(n) = x(n) + x(n-1)$

Linear Time-Invariant (LTI) Digital Filters



- Linearity
  - Homogeneity: if $x(n) \rightarrow y(n)$, then $\mathrm{a} \cdot x(n) \rightarrow \mathrm{a} \cdot y(n)$
  - Superposition: if $x_1(n) \rightarrow y_1(n)$ and $x_2(n) \rightarrow y_2(n)$, then $x_1(n) + x_2(n) \rightarrow y_1(n) + y_2(n)$

- Time-Invariance
  - If $x(n) \rightarrow y(n)$, then $x(n-N) \rightarrow y(n-N)$ for any N
  - This means that the system does not change its behavior over time

LTI Digital Filters

- A LTI digital filters performs a combination of the three operations
  - $y(n) = b_0 \cdot x(n) + b_1 \cdot x(n-1) + b_2 \cdot x(n-2) + \cdots + b_M \cdot x(n-M)$

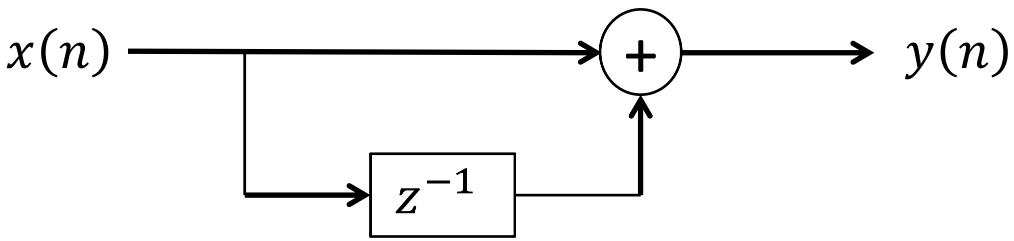
- This is a general form of **Finite Impulse Response (FIR) filter**

The Simplest Lowpass Filter

- Difference equation

$$y(n) = x(n) + x(n-1)$$

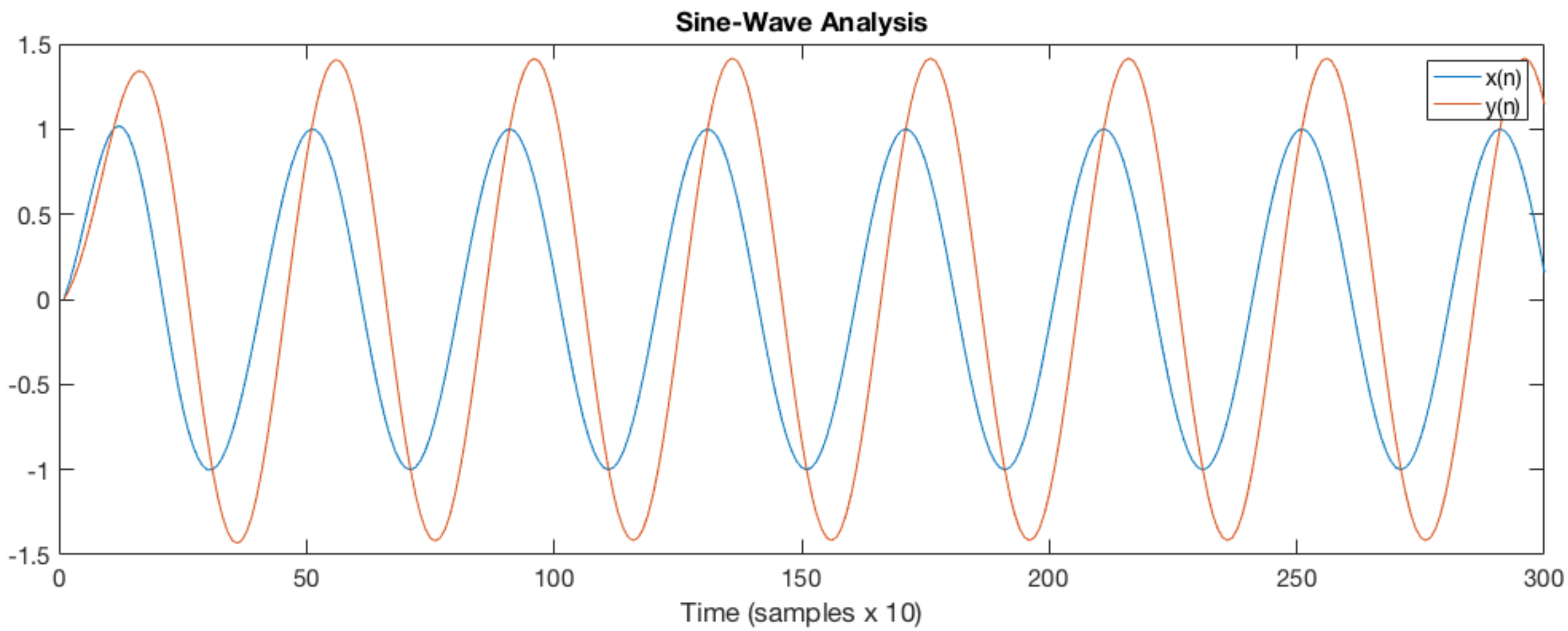
- Signal flow graph



“Delay Operator”

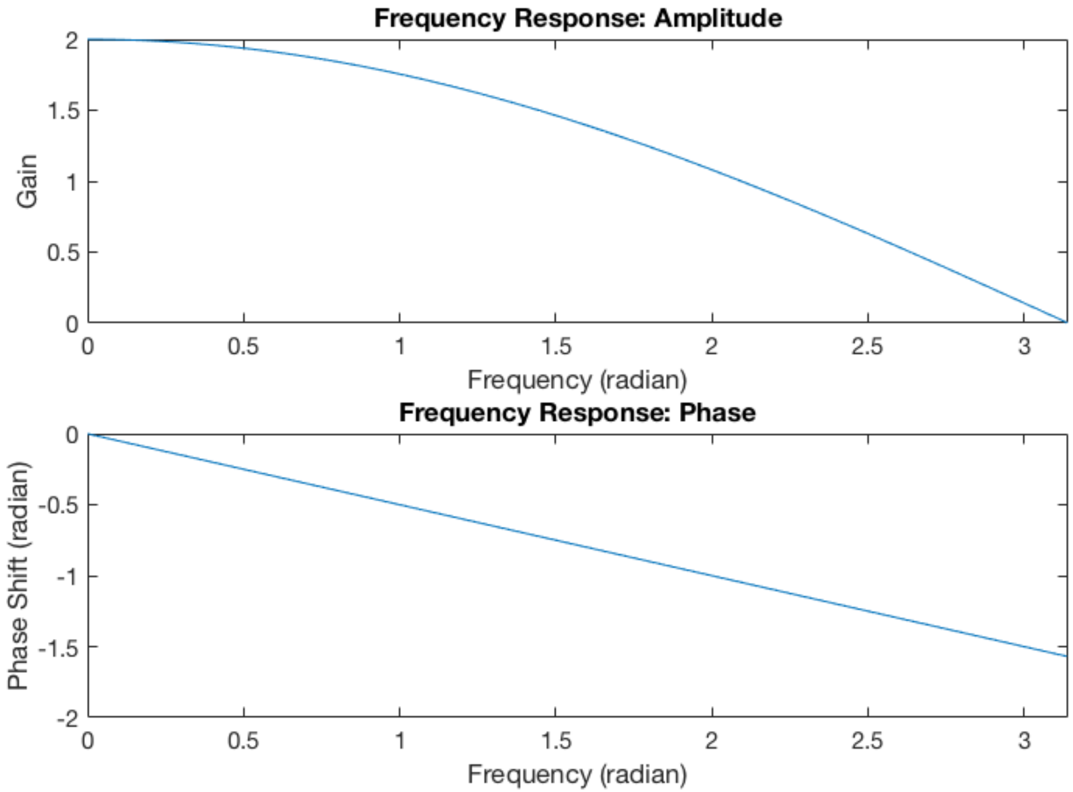
The Simplest Lowpass Filter: Sine-Wave Analysis

- Measure the amplitude and phase changes given a sinusoidal signal input



The Simplest Lowpass Filter: Frequency Response

- Plot the amplitude and phase change over different frequency
  – The frequency sweeps from 0 to the Nyquist rate



The Simplest Lowpass Filter: Frequency Response

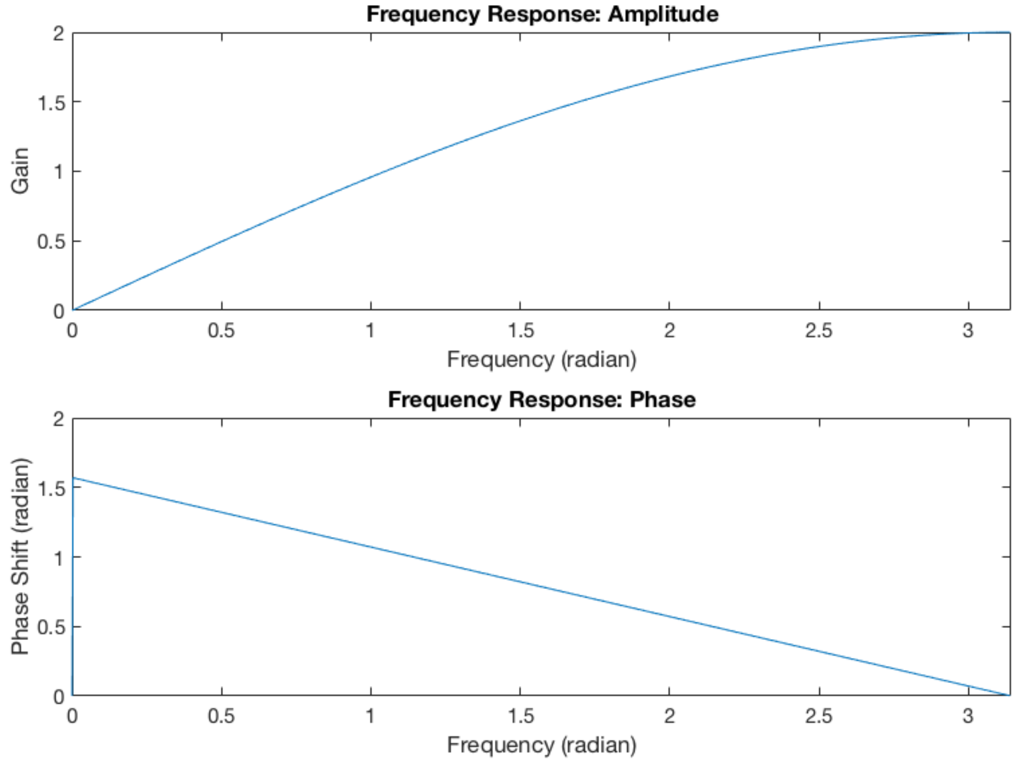
- Mathematical approach
  - Use complex sinusoid as input: $x(n) = e^{j\omega n}$
  - Then, the output is:

    $$y(n) = x(n) + x(n-1) = e^{j\omega n} + e^{j\omega(n-1)} = \left(1 + e^{-j\omega}\right) \cdot e^{j\omega n} = \left(1 + e^{-j\omega}\right) \cdot x(n)$$

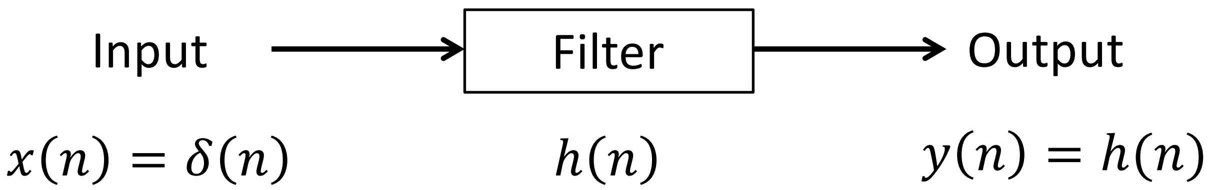
  - Frequency response: $H(\omega) = \left(1 + e^{-j\omega}\right) = \left(e^{j\frac{\omega}{2}} + e^{-j\frac{\omega}{2}}\right) e^{-j\frac{\omega}{2}} = 2\cos(\frac{\omega}{2})e^{-j\frac{\omega}{2}}$
  - Amplitude response: $|H(\omega)| = 2\cos\left(\frac{\omega}{2}\right)$
  - Phase response: $\angle H(\omega) = -\frac{\omega}{2}$

The Simplest Highpass Filter

- Difference equation: $y(n) = x(n) - x(n-1)$
- Frequency response



Impulse Response



- The filter output when the input is a unit impulse
  - $x(n) = \delta(n) = [1, 0, 0, 0, \dots] \rightarrow y(n) = h(n)$
- Characterizes **the digital system as a sequence of numbers**
  - A system is represented just like audio samples!

Examples: Impulse Response

- The simplest lowpass filter
  - $\mathrm{h}(n) = [1, 1]$

- The simplest highpass filter
  - $\mathrm{h}(n) = [1, -1]$

- Moving-average filter (order=5)
  - $\mathrm{h}(n) = \left[\frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}, \frac{1}{5}\right]$

- General FIR Filter
  - $\mathrm{h}(n) = [b_0, b_1, b_2, \ldots, b_M]$ → A finite length of impulse response

Convolution

- The output of LTI digital filters is represented by **convolution operation** between $x(n)$ and $h(n)$

$$y(n) = x(n) * h(n) = \sum_{i=0}^{M} x(i) \cdot h(n-i)$$

- Deriving convolution
  - The input can be represented as a time-ordered set of weighted impulses
    - $x(n) = [x_0, x_1, x_2, \dots, x_M] = x_0 \cdot \delta(n) + x_1 \cdot \delta(n-1) + x_2 \cdot \delta(n-2) + \cdots + x_M \cdot \delta(n-M)$
  - By the linearity and time-invariance
    - $y(n) = x_0 \cdot h(n) + x_1 \cdot h(n-1) + x_2 \cdot h(n-2) + \cdots + x_M \cdot h(n-M) = \sum_{i=0}^{M} x(i) \cdot h(n-i)$

Convolution In Practice

- The practical expression of convolution

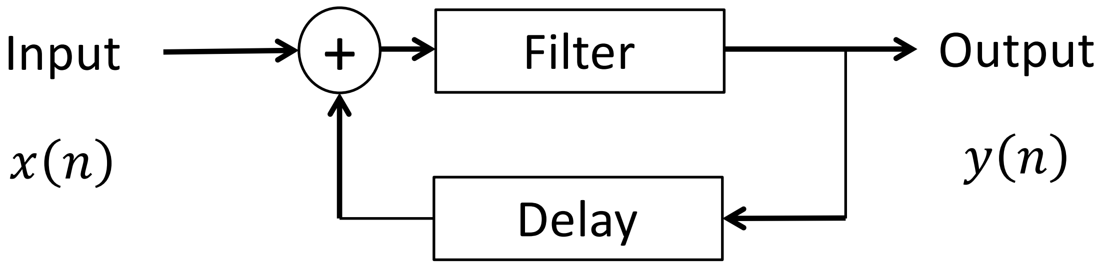
$$y(n) = x(n) * h(n) = \sum_{i=0}^{M} x(i) \cdot h(n-i) = \boxed{\sum_{i=0}^{M} h(i) \cdot x(n-i)}$$

  - This represents input $x(n)$ as a streaming data to the filter $h(n)$

- Matlab Animation Demo
  - http://mac.kaist.ac.kr/~juhan/ctp431/convolution_demo.html

- The length of convolution output
  - If the length of $x(n)$ is M and the length of $h(n)$ is N, the length of $y(n)$is M+N-1

Feedback Filter



- LTI digital filters allow to use the past outputs as input
  - Past outputs: $y(n-1), y(n-2), \dots, y(n-N)$

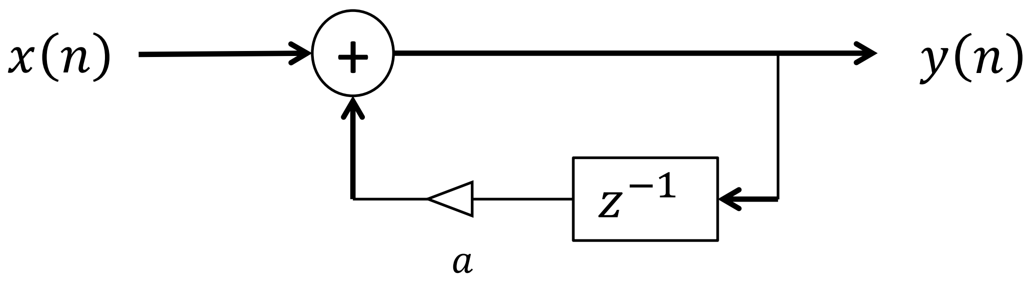
- The whole system can be represented as
  - $y(n) = b_0 \cdot x(n) + a_1 \cdot y(n-1) + a_2 \cdot y(n-2) + \cdots + a_N \cdot y(n-N)$
  - This is a general form of **Infinite Impulse Response (IIR) filter**

A Simple Feedback Lowpass Filter

- Difference equation

$$y(n) = x(n) + a \cdot y(n-1)$$

- Signal flow graph



  – When a is slightly less than 1, it is called "Leaky Integrator"

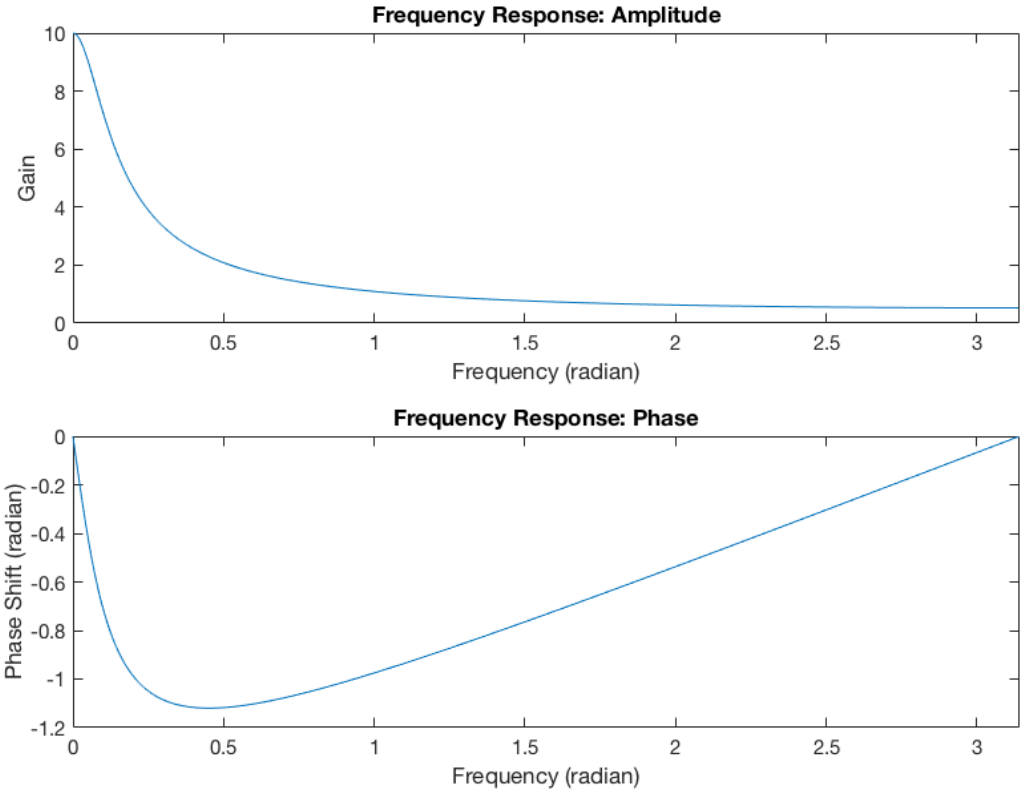
A Simple Feedback Lowpass Filter: Impulse Response

- Impulse response
  - $y(0) = x(0) = 1$
  - $y(1) = x(1) + a \cdot y(0) = a$
  - $y(2) = x(2) + a \cdot y(1) = a^2$
  - …
  - $y(n) = x(n) + a \cdot y(n-1) = a^n$

- **Stability!**
  - If $a < 1$, the filter output converges (stable)
  - If $a = 1$, the filter output oscillates (critical)
  - If $a > 1$, the filter output diverges (unstable)

A Simple Feedback Lowpass Filter: Frequency Response

- More dramatic change than the simplest lowpass filter (FIR)
  - Phase response is not linear



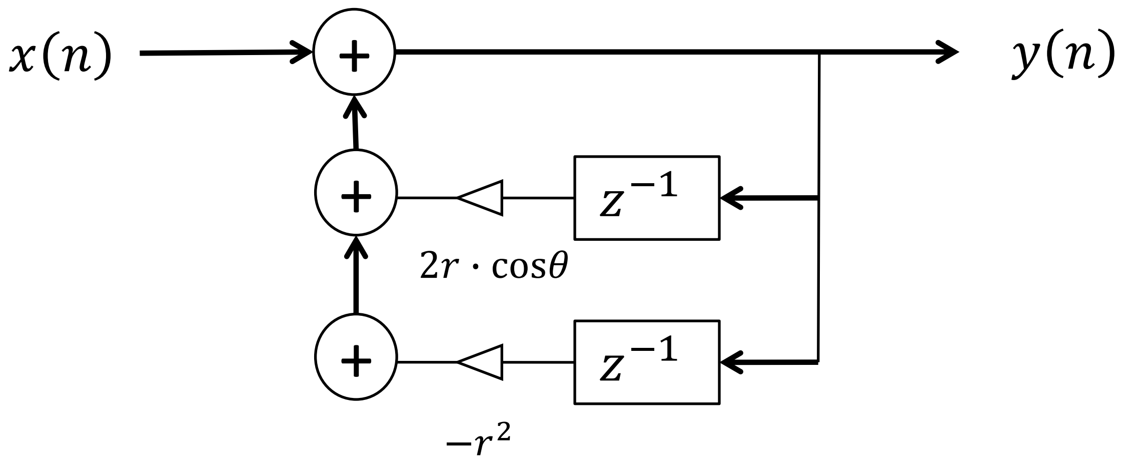
$$y(n) = x(n) + 0.9 \cdot y(n-1)$$

Reson Filter

- Difference equation

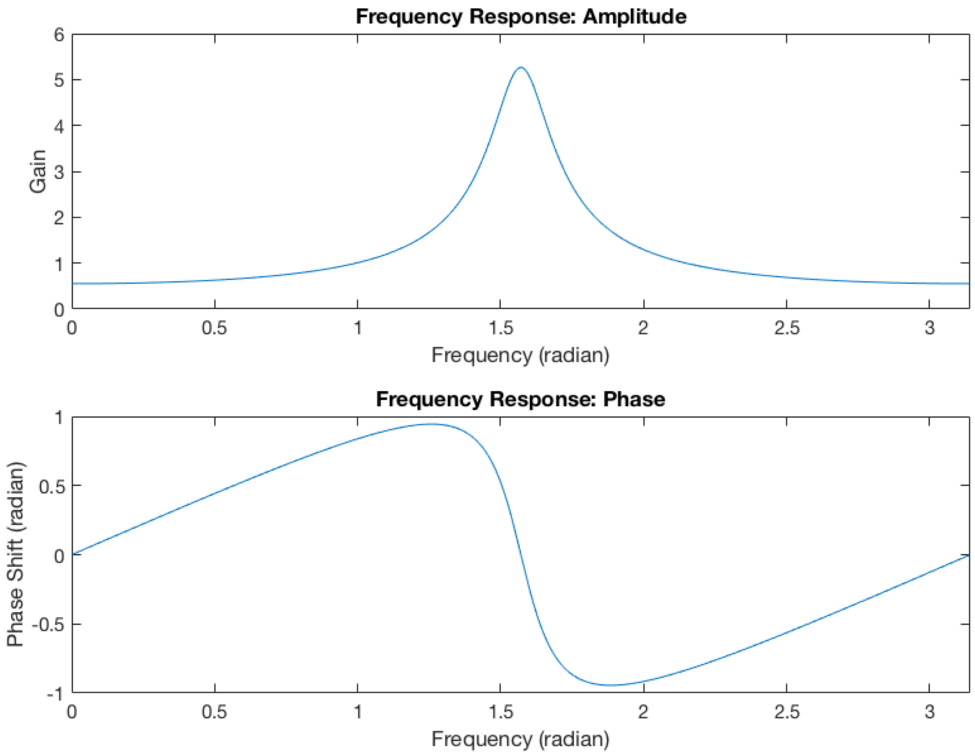
$$y(n) = x(n) + 2r \cdot \cos\theta \cdot y(n-1) - r^2 \cdot y(n-2)$$

- Signal flow graph



Reson Filter: Frequency Response

- Generate resonance at a particular frequency
  - Control the peak height by r and the peak frequency by θ



For stability: $r < 1$

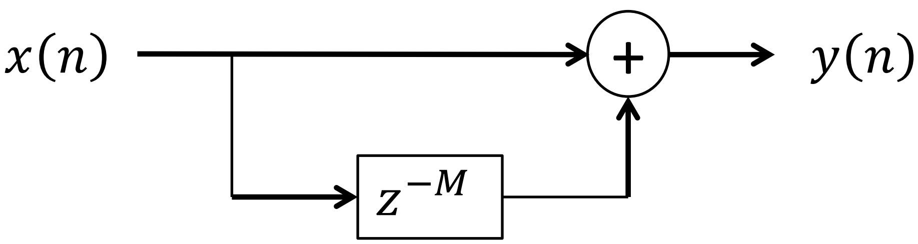
Comb Filter

- Difference equation

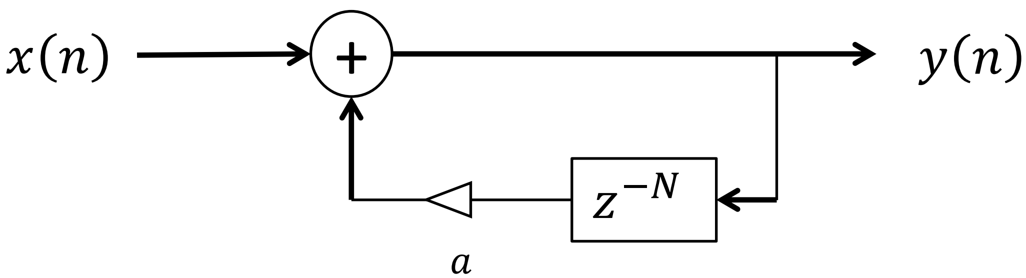
$$y(n) = x(n) + x(n - M) \qquad \text{(Feedforward)}$$

$$y(n) = x(n) + a \cdot y(n - N) \qquad \text{(Feedback)}$$

- Signal flow graph



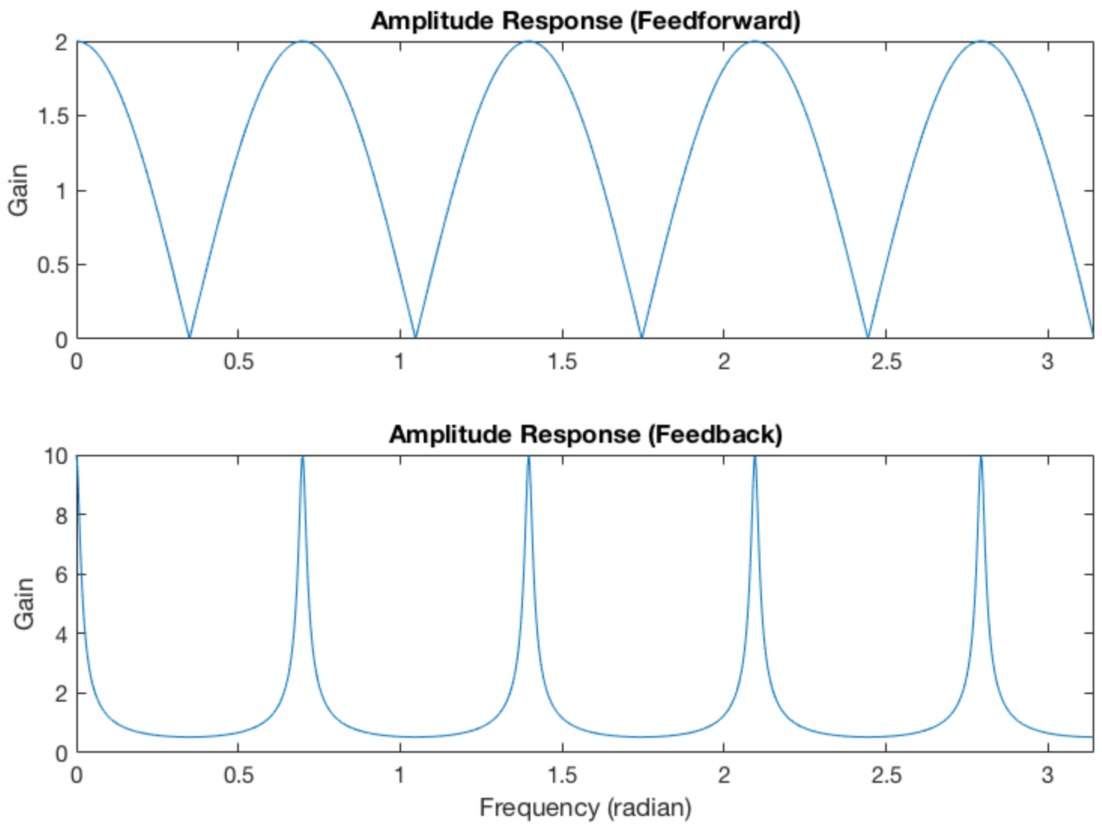
Feedforward



Feedback

Comb Filter: Frequency Response

- ”Combs” become shaper in the feedback type



$$y(n) = x(n) + x(n-8)$$

$$y(n) = x(n) + 0.9 \cdot y(n-8)$$

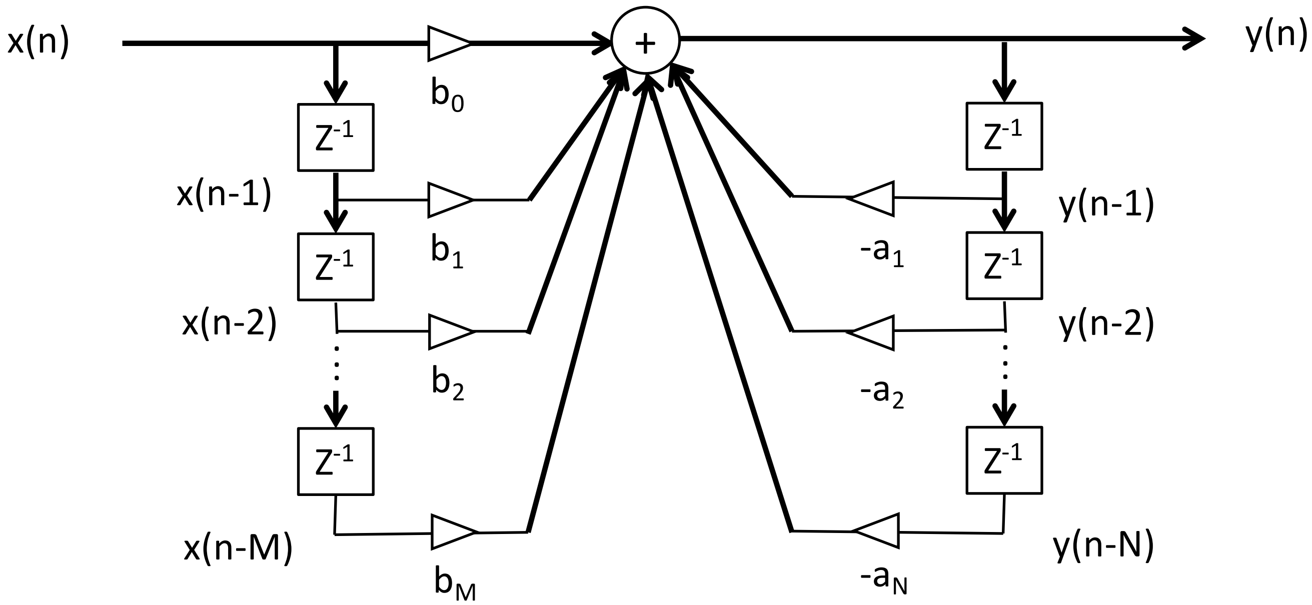
Perception of Time Delay

- The 30 Hz transition
  - Given a repeated click sound (e.g. impulse train):
    - If the rate is less than 30Hz, they are perceived as discrete events.
    - As the rate is above 30 Hz, they are perceive as a tone
  - Demo: http://auditoryneuroscience.com/?q=pitch/click_train

- Feedback comb filter: $y(n) = x(n) + a \cdot y(n - N)$
  - If $\mathrm{N} < \frac{F_s}{30}$ (F_s: sampling rate): models sound propagation and reflection with energy loss on a string (**Karplus-strong model**)
  - If $\mathrm{N} < \frac{F_s}{30}$ (F_s: sampling rate): generate a looped delay

General Filter Form

- The general form of digital Filters
  - $y(n) = b_0 \cdot x(n) + b_1 \cdot x(n-1) + b_2 \cdot x(n-2) + \ldots + \ b_N \cdot x(n-M)$
    $+a_1 \cdot y(n-1) + a_2 \cdot y(n-2) + \cdots + a_N \cdot y(n-N)$



Frequency Response

- Sine-wave Analysis
  - $x(n) = e^{j\omega n} \rightarrow x(n-m) = e^{j\omega(n-m)} = e^{-j\omega m}x(n)$ for any m
  - Let's assume that $y(n) = G(\omega)e^{j(\omega n+\theta(\omega))} \rightarrow y(n-m) = e^{-j\omega m}y(n)$ for any m

- Putting this into the different equation

$$y(n) = b_0 \cdot x(n) + b_1 \cdot e^{-j\omega} \cdot x(n) + b_2 \cdot e^{-j2\omega} \cdot x(n) + \ldots + b_M \cdot e^{-j\omega M} \cdot x(n)$$
$$+a_1 \cdot e^{-j\omega} \cdot y(n) + a_2 \cdot e^{-j2\omega} \cdot y(n) + \cdots + a_N \cdot e^{-j\omega N} \cdot y(n)$$

$$y(n) = \frac{b_0 + b_1 \cdot e^{-j\omega} + b_2 \cdot e^{-j2\omega} + \ldots + b_M \cdot e^{-j\omega M}}{1 + a_1 \cdot e^{-j\omega} + a_2 \cdot e^{-j2\omega} + \ldots + a_N \cdot e^{-j\omega N}} x(n)$$

$$\boxed{H(\omega) = \frac{b_0 + b_1 \cdot e^{-j\omega} + b_2 \cdot e^{-j2\omega} + \ldots + b_M \cdot e^{-j\omega M}}{1 + a_1 \cdot e^{-j\omega} + a_2 \cdot e^{-j2\omega} + \ldots + a_N \cdot e^{-j\omega N}}}$$

$H(\omega)$: frequency response
$G(\omega) = |H(\omega)|$: amplitude response
$\theta(\omega) = \angle H(\omega)$: phase response

Z-Transform

- Z-transform
  - Define z to be a variable in complex plane: we call it z-plane
  - When $z = e^{j\omega}$ (on unit circle), the frequency response is a particular case of the following form

$$H(z) = \frac{B(z)}{A(z)} = \frac{b_0 + b_1 \cdot z^{-1} + b_2 \cdot z^{-2} + \ldots + b_M \cdot z^{-M}}{1 + a_1 \cdot z^{-1} + a_2 \cdot z^{-2} + \ldots + a_N \cdot z^{-N}}$$

  - We call this z-transform or the transfer function of the filter
  - z^{-1} corresponds to one sample delay: delay operator or delay element
  - Filters are often expressed as z-transform: polynomial of z^{-1}

Z-Transform

- Poles and Zeros
  - $H(z)$ can be factorized and we can find roots for each of polynomials
  - Zeros: the numerator roots
  - Poles: the denominator roots

$$H(z) = \frac{B(z)}{A(z)} = \frac{(1 - q_1 z^{-1})(1 - q_2 z^{-1})(1 - q_3 z^{-1})\ldots(1 - q_M z^{-1})}{(1 - p_1 z^{-1})(1 - p_2 z^{-1})(1 - p_3 z^{-1})\ldots(1 - p_N z^{-1})}$$

- We can analyze frequency response of filters more easily with poles and zeros than numerator or denominator coefficient

Practical Filters

- One-pole one-zero filters
  - Leaky integrator
  - Moving average
  - DC-removal filters
  - Bass / treble shelving filter

$$H(z) = \frac{b_0 + b_1 z^{-1}}{a_0 + a_1 z^{-1}}$$

- Biquad filters
  - Reson filter
  - Band-pass / notch filters
  - Equalizer: a set of biquad filters

$$H(z) = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{a_0 + a_1 z^{-1} + a_2 z^{-2}}$$

- Any high-order filter can be factored into a combination of one-pole one-zero filters or bi-quad filters!