Digital Filters

What is a filter?

- Take input and create output
- Programs are filters
- People are filters
- Physics is filters

- That's too general.

2

Filters

- Filters *shape* the frequency spectrum of a sound signal.
  - Filters generally do not add frequency components to a signal that are not there to begin with.
  - Boost or attenuate selected frequency regions

3

Which of the Following is not a Filter?

- Graphic Equalizer on a stereo system
- Tone control on a stereo system
- Microphone
- Mixing Board
- Loudspeaker
- Vocal Tract
- Ear

4

Definition

- A filter is any operation on a signal
  (From Rabiner et al, "Terminology in Digital Signal Processing.")

- Commonly, we limit the term filter to devices (hardware or software) that were designed specifically to boost or attenuate regions of a signal spectrum.

5

Linear Time-Invariant Filters

- 95% of what we need will be:
- Linear
  - **Homogeneity:** Input and output scales at the same rate.
    - If filter(x) = y then filter(a*x)=a*y (a is a real number)
  - **Superposition:** The sum of two inputs is the sum of two outputs.
    - If filter(x1)=y1 and filter(x2)=y2 then filter(x1+x2)=y1+y2
- Time Invariant
  - Time delayed inputs yield time delayed output without change of the filter
    - If filter(x(now))=y(now)then filter(x(now+time))=y(now+time)

6

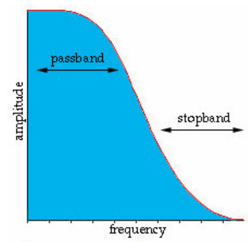
Filter types: frequency response

- Functions which take one sequence of numbers (the input signal) and produces a new sequence of numbers (the filtered output signal)
- The Four Basic Types
  1. Lowpass
  2. Highpass
  3. Bandpass
  4. Band reject

cutoff frequency

7

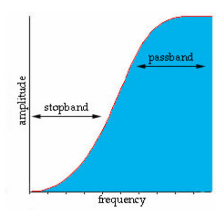
Lowpass



- lets through low frequencies
- Stops high frequencies

8

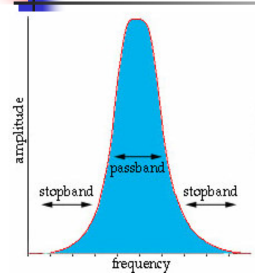
Highpass



- lets through high frequencies
- Stops low frequencies

9

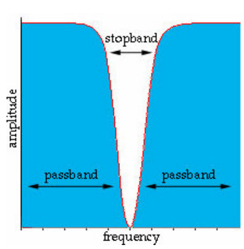
Bandpass



- combinations of lowpass and highpass filters
- lets through only frequencies above a certain point and below another, so there is a band of frequencies that get through

10

Band Reject



- combinations of lowpass and highpass filters
- stops a band of frequencies
- sometimes called *notch* filters, because they can notch out a particular part of a sound

11

Defining Filter Properties

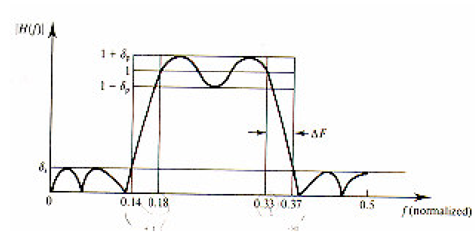
- Cutoff Frequency
  - Defined at the half-power point (filtered output amplitude at 0.707 of original signal)
- Bandwidth (in Bandpass filters) is distance (in Hz) between half-power points
- Center Frequency is point of maximum amplitude.
- Slope of filter (or rolloff) is usually described in dB/octave.

12

Passband & Stopband

- Passband
  - The area where frequencies are passed
- Stopband
  - The area where frequencies are stopped
- The center frequency
  - the frequency in the middle of the band
- The filter's bandwidth
  - The width of the band is called

Slope, passband, stopband



Bandpass Filter

Cutoff Frequency

- Real filters can't just stop all frequencies at a certain point
- Instead the ways that frequencies die out according to a sort of curve around the corner of their cutoff frequency
- the pictures in the figures above (the four different types of filter) don't have right-angles at the cutoff frequencies

Transition Band

- The area between where a filter "turns the corner" and where it "hits the bottom“
- The steepness of the slope in the transition band
  - important in defining the sound of a particular filter
  - If the slope is very steep, the filter is said to be "sharp”
  - If the slope is more gradual, the filter is "soft" or "gentle"

More on Slope

- The number of delayed samples a filter “looks back” to mix with new input gives the filter’s *order*.
  - 1 sample = first order; 2 samples = second order, etc.
  - Each order represents an additional 6 dB/octave increase in slope.
- The greater the slope (more dB/octave change), the greater the phase distortion of the filter.

Filter Q (Bandpass and Notch)

- *Q* represents the “quality” of the filter, represented by the equation:

$$Q = \frac{f_{center}}{f_{highcutoff} - f_{lowcutoff}}$$

The Impulse

- An impulse is a very short pulse—a waveform that has significant amplitude only for a very short time. (usually unipolar)
- For filters, we use a one-sample pulse, or unit impulse.
- The response of the filter to the unit impulse is the filter's Impulse Response (IR).

19

Impulse Response

- Impulse response describes filter behavior completely.
- Why? At all time delays response will be the same and input can be chopped into infinite stream of impulses. Sum of impulse inputs -> Total response

20

Impulses: FIR and IIR

- Delaying the input signal creates a *Finite Impulse Response* filter (FIR).
- Delaying the output signal creates an *Infinite Impulse Response* filter (IIR).
  - IIR filter is essentially a feedback loop.

21

Impulses: FIR and IIR

- Filters work by using one or both of the following methods:
  - Delay a copy of the *input signal* (by *x* number of samples), and combine the delayed input signal with the new input signal.
    - (**Finite Impulse Response, FIR, or feedforward filter**)
  - Delay a copy of the *output signal* (by *x* number of samples), and combine it with the new input signal.
    - (**Infinite Impulse Response, IIR, feedback filter**)

22

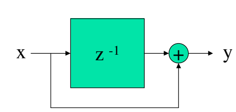
The Math

- First, how we label things:
  - *x* is the input signal
  - *y* is the output signal
  - *n* is the sample index (all samples are numbered, or *indexed*)
- $x[0]$ is the first sample of input; $y[0]$ is the first sample of output. $x[n]$ is the current sample; $x[n-1]$ is the previous sample.

23

FIR: What does this do?

- Add time-delayed signal to itself



- Can we find out properties?
- This is a feed-forward filter. Why?

24

Finite Impulse Response (FIR)

- Delaying a signal and averaging (in a wide variety of ways) the delayed signal and the non-delayed one
- Delays mean that the sound that comes out at a given time uses some of the previous samples
- The sound has been delayed before it gets used
- FIR comes out uses a finite number of samples, and a sample only has a finite effect

25

FIR Math [0]

- Simple Lowpass Filter (averaging):
  - output = half_of_current_input + half_of_previous_input
  - $y[n] = (0.5 \times x[n]) + (0.5 \times x[n-1])$
- Simple Highpass Filter (difference):
  - output = half_of_current input - half_of_previous_input
  - $y[n] = (0.5 \times x[n]) - (0.5 \times x[n-1])$

26

FIR Math[1]

- Filters generally use more than one sample delay, with independently determined coefficients.

$$y[n] = (a_0 \times x[n]) \pm (a_1 \times x[n-1]) \pm \ldots (a_i \times n[n-i])$$

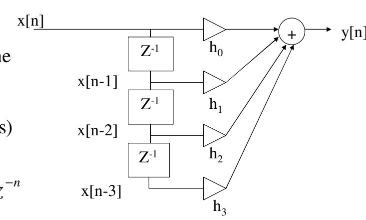
  - a_i is the last coefficient in the series, and $x[n - i]$ is the last delayed sample
  - The Order of the filter is equal to the number of samples you look back. Generally, the higher the order—the more samples you look back to take an average or difference—the more attenuation of frequencies.

27

Basic Structure for FIR

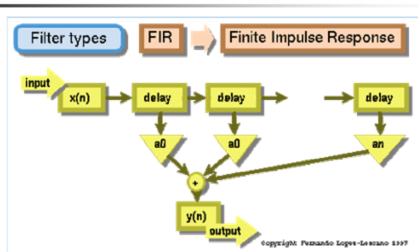
- Tapped-delay line
(N-1) delays
- N multipliers
- 1 adder (N inputs)

$$H(z) = \sum_{n=0}^{3} h_n z^{-n}$$



28

FIR



29

FIR Code for 56300 DSP

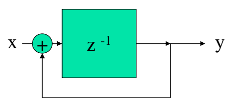
- Filter order 'n'
- Input and Output in accumulator 'a'
- r0: samples, r4: coefs, m0 & m4: n-1

```
move   a,x(r0)
clr    a          x:(r0)+,x0        y:(r4)+,y0
rep    #n-1
mac    x0,y0,a    x:(r0)+,x0        y:(r4)+,y0
macr   x0,y0,a    (r0)-
```

30

IIR: How about this one?

- Remember &
- “accumulate” output



- Any exciting properties?
- A feedback filter?
- Real-world examples?

IIR Math[0]

- The feedback loop introduced creates the possibility of an infinite impulse (delayed sample).
- The simple averaging filter becomes an *Exponential Time Averaging* Filter (ETA Filter), equivalent to an infinitely long FIR filter.

$$y[n] = (0.5 \times x[n]) + (0.5 \times y[n-1])$$

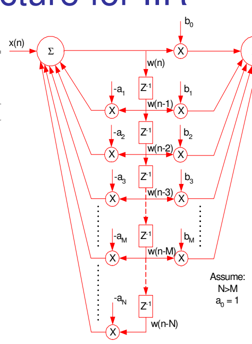
$$y[n] = (1/2 \times x[n]) + (1/4 \times x[n-1]) + (1/8 + x[n-2])\ldots$$

Basic Structure for IIR

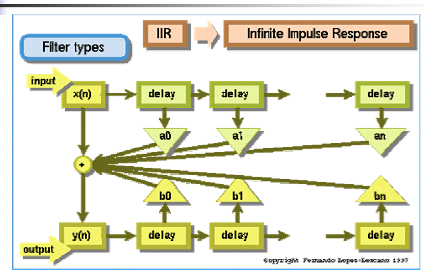
•Tapped-delay line (N or M delays

•N + M + 1 multipliers

•2 adders (N and M+1 inputs)



IIR



IIR Filters

- IIR (infinite impulse response) filters allow zeros *and* poles; FIR allow zeros only. IIR can be more *selective* for a given filter order
- IIR also called *recursive* filters: output depends on past inputs *and* past outputs
- IIR designs are not guaranteed to be stable
- IIR filters can be particularly sensitive to coefficient quantization

IIR Issues: Stability and Sensitivity

- Finite precision of coefficients can lead to several issues:
  - In order to be unconditionally stable and causal, all system poles must be inside the unit circle (|z|<1). Coefficient roundoff may inadvertently move a pole outside unit circle
  - Finite coefficient precision “quantizes” pole locations: may change frequency response from ideal case even if still stable

Overflow Issues

- Gain from input to storage nodes in the filter may exceed unity. This can cause filter state to be saturated (clipped), resulting in distortion
- Typically must scale down (attenuate) the input signal, then scale up (amplify) by an equal amount on the output

Q and Gain

- High-Q filters can self-oscillate when fed frequencies near their center frequency.
- Gain is the amount of boost or attenuation of a frequency band.
  - Care must be taken with high *Q* filters so that the gain at the center frequency does not distort.

IIR Filters and Resonance

- Because of the feedback loop, IIR filters can provide an amplitude increase around the cutoff or center frequency.
- This amplitude increase is usually referred to as *resonance*.
- IIR filters also provide steeper slopes with less computation.

Second-Order Sections

- High-order filter polynomials involve terms that are products and sums involving many poles and zeros. Small roundoff errors when implementing filter can lead to large response errors
- As with analog filters, it is typical to reduce sensitivity by using *second-order sections*

Implementing 2nd Order Sections

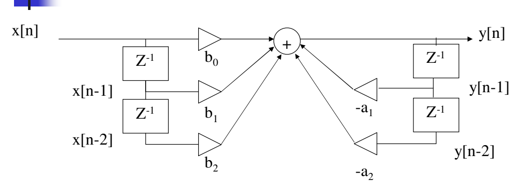
- 2nd Order (bi-quad) expression

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

$$= b_0 \frac{1 + k_1 z^{-1} + k_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

- Numerator implements 2 zeros, denominator implements 2 poles

Direct Form Bi-Quad



IIR Code for 56300

- Direct Form II, with equations:
  $w(n)=x(n)-a_{i1}w(n-1)-a_{i2}w(n-2)$
  $y(n)=w(n)+b_{i1}w(n-1)+b_{i2}w(n-2)$
- Since a_{i1} and b_{i1} may be > 1, need to divide all coefs by 2, then use special *scaling mode* for ×2 on read from accumulator:

```
ori     #$08,MR
```

  → sets "scale up": 1-bit left shift on acc read

43

IIR for 56300 (cont.)

- *N* = number of second-order sections
- Filter state (w) in X memory: r0
- Filter coefs (a,b) in Y memory: r4
  - Coefs stored in order:
    - $a_{12}/2$, $a_{11}/2$, $b_{12}/2$, $b_{11}/2$, $a_{22}/2$, … $b_{N2}/2$
  - State (data) stored in order:
    - $w_1(n-2)$, $w_1(n-1)$, $w_2(n-2)$, $w_2(n-1)$, … $w_N(n-1)$
- m0 = 2*N-1, m4 = 4*N-1
- Initial gain in y1, input in y0, output in 'a'

44

IIR for 56300 (cont.)

```
 mpy y0,y1,a   x:(r0)+,x0     y:(r4)+,y0
 do  #N,end_cell
 mac -x0,y0,a  x:(r0)-,x1     y:(r4)+,y0
 macr -x1,y0,a x1,x:(r0)+     y:(r4)+,y0
 mac x0,y0,a   a,x:(r0)+           y:(r4)+,y0
 mac x1,y0,a   x:(r0)+,x0     y:(r4)+,y0
end_cell
 rnd a
```

45

Other Filter Structures

- Direct Form I and Direct Form II
- Cascade and Parallel Realizations
- Transpose Forms
- Lattice Forms

46

Care and Handling

- Care has to be taken with any feedback system.
  - Feedback coefficients have to remain below 1.0, or the filter becomes unstable.
- IIR filters are computationally less expensive than FIR filters for greater shaping potential.
  - drawbacks: phase distortion and ringing.
- FIR filters are always stable, and prevent phase distortion.
  - drawbacks: more computation than an IIR with similar effect.

47

Complicated Filters

- More complicated filters can generally be built by using combinations of second-order IIR filters, or combinations of FIR and IIR filters.
  - feedforward paths (FIR) usually contribute notches to the frequency spectrum.
  - feedback paths (IIR) usually contribute peaks.

48