DIGITAL SIGNAL PROCESSING: COSC390

Jordan Hanson

January 22, 2019

Whittier College Department of Physics and Astronomy

1. Types of filters (reading: ch. 3, ch. 5)
    - Butterworth
    - Bessel
    - Chebyshev
2. LTI systems and their properties (reading: ch. 5)
3. Convolution (reading: ch. 7)
    - Implementation with FFT
    - Impulse and step response

These lectures will cover: (Reading: chapter 19)

1. **Common filter kernels: Moving Average**
2. **General Recursive Filters: HP, LP, and Notch Examples**
3. **FIR and IIR definitions**

COMMON FILTER KERNELS: MOVING AVERAGE

The out of an LTI system such as a filter man be expressed as a *convolution* of the transfer function coefficients a_k and the data x. If the output is y, we may write

$$y[n] = \sum_{k=0}^{N-1} a_k x[n-k] \tag{1}$$

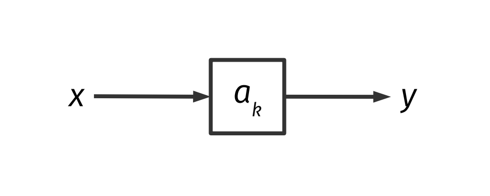

Figure 1: A simple model for convolution as an LTI response.

The out of an LTI system such as a filter man be expressed as a *convolution* of the transfer function coefficients a_k and the data x. If the output is y, we may write

$$y[n] = \sum_{k=0}^{N-1} a_k x[n-k] + \sum_{k=0}^{N-1} b_k y[n-k] \tag{2}$$

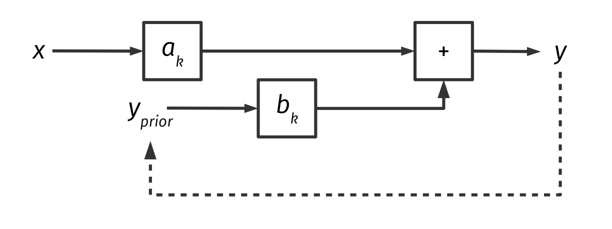

Figure 2: A more general model for convolution as an LTI response. These are the a and b coefficients of the octave *filter* function.

Moving average filter with window-length n:

1. a_k are all equal to $1/n$
2. b_k are zero

Example:

- window-3 moving average: $a_0 = \frac{1}{3}, a_1 = \frac{1}{3}, a_2 = \frac{1}{3}$

```
y = filter(ones(window,1)/window,1,y);
```

The filter function is expecting the a_k and b_k coefficients[1].

[1] For maximum confusion: the textbook and Octave use opposite conventions for a_k and b_k.

OCTAVE PROGRAMMING EXERCISE: MOVING AVERAGE

Download the code **movingAverage.m** from Moodle, and run it at your desk.

1. What is happening to the spectrum?
2. Change the window parameter to various values. What further changes do you observe in the noise and the spectrum?
3. What function shape do you observe in the spectrum with sufficient window size?

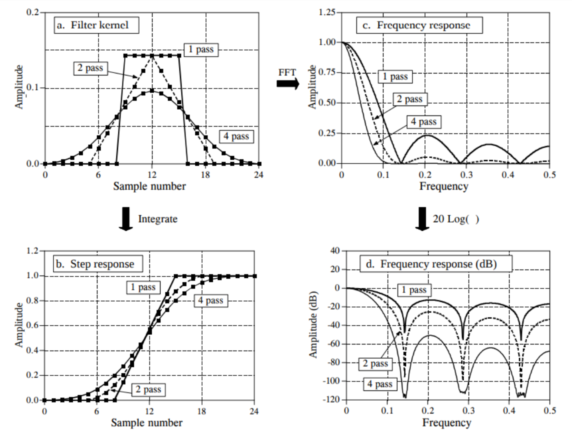

Figure 3: Various effects of the moving average filter kernel. The moving average efficiently filters high-frequency noise in the time-domain.

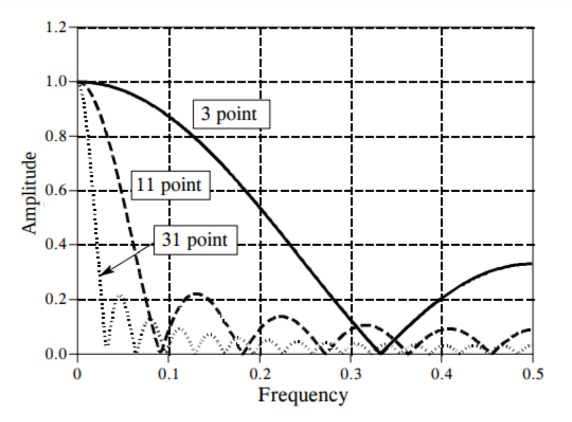

Figure 4: The frequency response of a moving average filter is a *sync*. Why? (Calculations on board of filter kernel).

COMMON FILTER KERNELS: MOVING AVERAGE

Frequency response of moving average filter, n-window:

$$h(f) = \frac{\sin(\pi f n)}{n \sin(\pi f)} \tag{3}$$

- Basically, a sync
- Modified by n-window instead of period

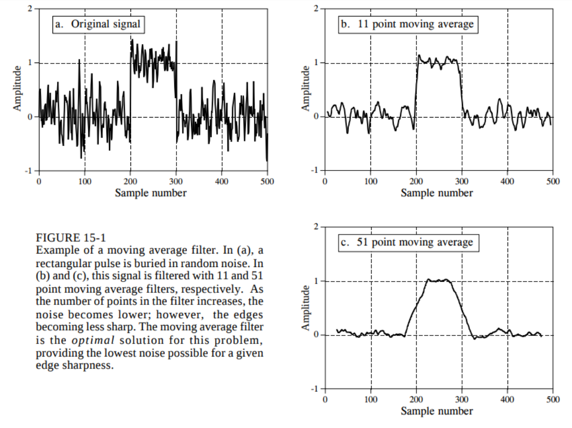

FIGURE 15-1
Example of a moving average filter. In (a), a rectangular pulse is buried in random noise. In (b) and (c), this signal is filtered with 11 and 51 point moving average filters, respectively. As the number of points in the filter increases, the noise becomes lower; however, the edges becoming less sharp. The moving average filter is the *optimal* solution for this problem, providing the lowest noise possible for a given edge sharpness.

Figure 5: The moving averege filter removes noise optimally while preserving step response.

OCTAVE PROGRAMMING EXERCISE: FINDING THE SIGNAL

Download the code **movingAverage2.m** from Moodle, and run it at your desk.

1. By tuning the moving average filter, can you reveal the signal?
2. Bonus: using older code, can you *play this signal as audio* for varying window sizes?

OCTAVE PROGRAMMING EXERCISE: SINGLE-POLE RECURSION FORMULAS

Single-pole LP filter recursion:

1. $a_0 = 1 - x$
2. $b_1 = x$

The variable x varies from $[0, 1]$. It is the amount of decay between samples:

$$x = \exp(-1/d) \tag{4}$$

The x parameter is related to the cutoff-frequency:

$$x = \exp(-2\pi f_c) \tag{5}$$

Exercise: implement this in movingAverage2.m and recover the signal with the single-pole LP filter. Note: we are not using the **butter** function...How would you achieve the effect of multiple poles?

Single-pole HP filter recursion:

1. $a_0 = (1 + x)/2$
2. $a_1 = -(1 + x)/2)$
3. $b_1 = x$

The variable x varies from $[0, 1]$. It is the amount of decay between samples:

$$x = \exp(-1/d) \tag{6}$$

The x parameter is related to the cutoff-frequency:

$$x = \exp(-2\pi f_c) \tag{7}$$

Exercise: implement this in movingAverage2.m and recover the signal with the single-pole HP+LP filter.

OCTAVE PROGRAMMING EXERCISE: NOTCH AND NARROW BAND-PASS

Implement the following recursive formula, and see what it does to noise[2]:

1. $a_0 = 1 - K$
2. $a_1 = 2(K - R)\cos(2\pi f)$
3. $a_2 = R^2 - K$
4. $b_1 = 2R\cos(2\pi f)$
5. $b_2 = -R^2$

($R = 1 - 3BW$, where BW is the bandwidth, centered on the frequency f). For K, we have

$$K = \frac{1 - 2R\cos(2\pi f) + R^2}{2 - 2\cos(2\pi f)} \tag{8}$$

[2] It's probably best to recycle code from movingAverage.m to a new file.

Implement the following recursive formula, and see what it does to noise[3]:

1. $a_0 = K$
2. $a_1 = -2K\cos(2\pi f)$
3. $a_2 = K$
4. $b_1 = 2R\cos(2\pi f)$
5. $b_2 = -R^2$

($R = 1 - 3BW$, where BW is the bandwidth, centered on the frequency f).

[3] It's probably best to recycle code from movingAverage.m to a new file.

1. Add a sine-tone to your noise sample, and then isolate it with the band-pass filter.
2. Now make the noise small, but add several other *unwanted sine-tones* to the data, and filter them out with the band-reject.

So now we start to see how simple it is to clean the data of unwanted noise and signals, using only recursive relationships between input and output data! However, we must eventually make a distinction between **FIR and IIR.**

FIR - Finite Impulse Response:

$$y[n] = \sum_{k=0}^{N-1} a_k x[n-k] \tag{9}$$

IIR - Infinite Impulse Response:

$$y[n] = \sum_{k=0}^{N-1} a_k x[n-k] + \sum_{k=0}^{N-1} b_k y[n-k] \tag{10}$$

THEORY AND EXAMPLES: PHASE RESPONSE

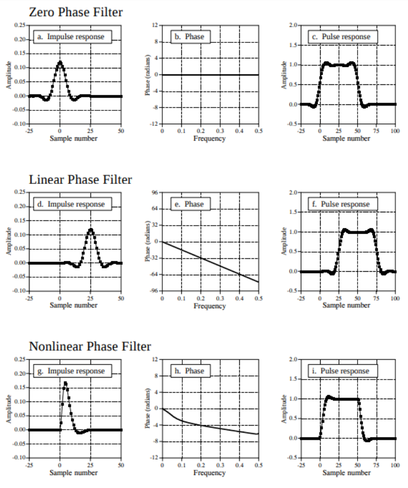

Figure 6: The problem with many filters is that they introduce non-zero or non-linear phase response. How do we eliminate this?

1. Review **group delay:** $\tau_g = -\frac{d\phi}{d\omega}$.
2. Discuss *unwrapping* if necessary.
3. Changing zero group delay to linear (time-shift, causality).
4. Taking $t \to -t$? What phase shift does this cause? What is the total group delay?