A C++ class to implement low-pass, high-pass, and band-pass filters

November 13th, 2013 by Mike Perkins

We recently needed a simple C++ class for linear phase FIR filtering, and I figured it might be useful to others, as well. You can download it here.

Here's how to use this class:

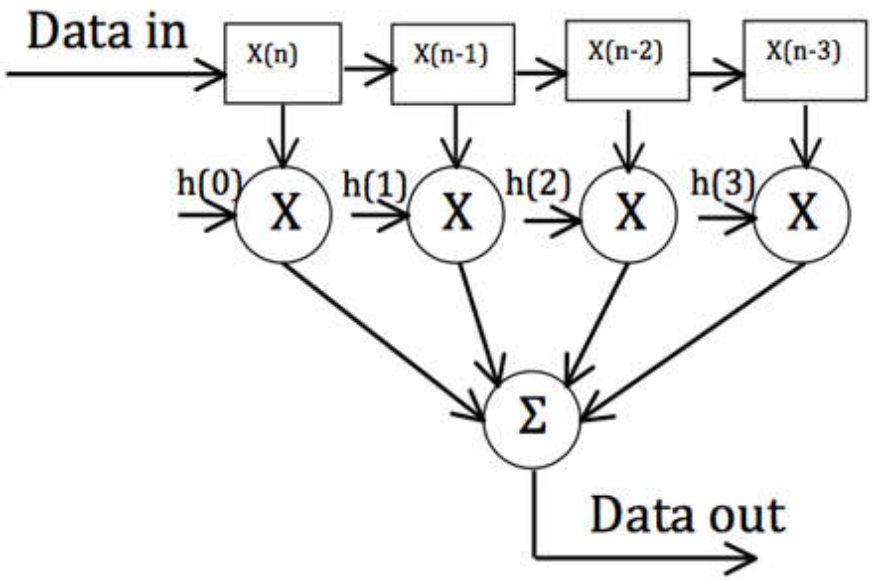
1. Specify the desired filter type (low-pass, high-pass, or band-pass) in the constructor, along with the other needed parameters: the number of taps, the transition frequencies, and the sampling frequency of the data you'll be filtering.
2. Make one call to the class's filter function, `do_sample()`, for every value in the data stream you're filtering. `do_sample()` returns a filtered value each time it is called.
3. Use optional class helper functions as desired, for example, to load an array with the filter taps, write the taps to a file, or write the frequency response to a file.

Details on using the class are contained in the file `filt.h`, and the implementation is in `filt.cpp`. A sample program demonstrating its use on an audio file is in `devel.cpp`.

If all you need is code that works, you can stop here. For those who want to know how the taps are computed, keep reading. I assume that you have some exposure to digital filtering and just need a brief refresher.

Reminder one:

An FIR filter can be implemented using the following structure:



The data to be filtered is shifted into the array on the left (each box holds one data sample). Each time a new value is shifted in, the value currently stored in each position is multiplied by its corresponding tap (the "h(k)" value), and the outputs of the multipliers are summed together to yield a single data output value. The computation of each output value requires 4 multiplies and 3 adds in this four tap filter example. (The class function `do_sample()` performs this computation.)

Reminder two:

The frequency response of an *N*-tap FIR filter is given by the Discrete Time Fourier Transform (DTFT) of its taps. The DTFT is periodic on the interval where corresponds to (half the sampling frequency) and is given by:

$$F(\omega) = \sum_{n=0}^{N-1} h(n) e^{-i\omega n}$$

Reminder three:

The expression above for the DTFT is a weighted sum of complex sinusoids; it looks exactly like a Fourier Series.

Therefore:

One way to design the taps of a digital filter is the following:

a. Choose a desired frequency response on the interval $[-\pi,\pi]$ where π corresponds to F_s / 2. Note that in the class `filt.cpp`, the desired frequency response is chosen to be that of an ideal filter of the specified type.
b. Choose a number of taps.

c. Compute the Fourier Series coefficients of the desired frequency response.
d. Plot the frequency response to see how well it meets your needs (the class function `write_freqres_to_file()` will output the frequency response to a file in a form suitable for plotting, for example, by using Excel).

If you are a purist, you may have noticed that the DTFT equation I gave looks like a Fourier Series but with one exception: The sign of the exponent in the complex sinusoids is negative instead of positive. But not to worry—when computing the Fourier Series coefficients, just flip the sign of the exponent in that formula too. In other words, if is your desired frequency response, compute the taps as follows:

$$h(n) = \frac{1}{2\pi}\int_{-\pi}^{\pi} H(\omega)e^{i\omega n}\,d\omega$$

For the ideal frequency responses of the filters used in `filt.cpp`, these computations are easy. You can see the resulting formulas for the coefficients by looking at the code.

Categories: Perk, Theory
Tags: mathematics, signal processing

AUDIO & VIDEO

- Alexa Voice Service
- Mobile Apps
- STB Apps
- Cloud Server Infrastructure
- Bluetooth & BLE
- DRM
- Codecs & Protocols
- Linux Kernel
- Signal Processing
- FPGA Design
- Hardware Design

INTERNET OF THINGS

- Mobile & Web Apps
- User Experience
- IoT Platforms
- Cloud Infrastructure
- Networking Protocols
- Embedded Operating Systems
- Signal Processing
- Hardware Design

ABOUT US

- Why Outsourced Engineering?
- Our Process
- Management Team
- Clients
- Industry Partnerships
- News
- Careers

NEW PROJECT ON THE HORIZON?

- (303) 665-3962
- info@cardinalpeak.com
- **BLOG**
- **CASECRACKER**

LinkedInTwitterFacebook
