Overview of Filter Design

The second part of the course deals with filter design. This lecture is an overview of the steps involved in the design of a digital filter.
After this lecture you should be able to: (1) apply the basic terminology used to specify filters in the frequency domain (passband, stopband, ripple, etc), (2) decide whether an FIR or IIR design is the most appropriate to meet particular design requirements, and (3) list the possible sources of degradation due to a finite word length implementations.
This material is covered in Chapter 5 of the textbook.

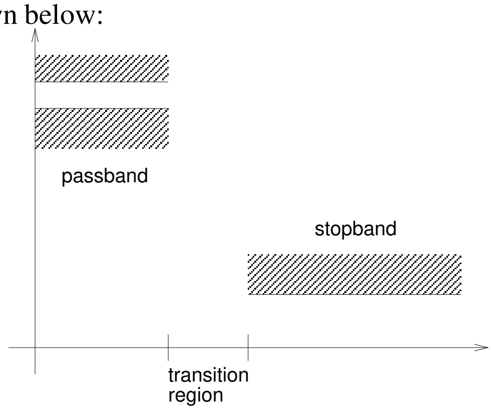
The Design Process

The first two steps essential to the design of a digital filter are: (1) the filter specification, and (2) the design of the pole/zero (or coefficient) values. When the filter is to be implemented in hardware such as a fixed-point DSP microprocessor additional design steps will include: (3) selection of a realization structure, (4) finite word length analysis and (5) the implementation of the filter in hardware.

Frequency-Domain Filter Specifications

The first step in the design to obtain a complete specification. Typically this will include the filter's amplitude response in the frequency domain. It might also include requirements on the phase response and implementation constraints such as the hardware to be used, the number of bits in the digitized signal, the sampling rate, the minimum SNR, etc.

The filter's frequency response is usually specified by the passband and stopband frequency ranges, the minimum stopband attenuation and the maximum allowed attenuation in the passband (the "ripple")as shown below:



The same specifications are used for the different types of filters: low-pass, bandpass, high-pass, all-pass, and band-reject or notch.

FIR versus IIR Designs

The first major design decision is whether to use an FIR or IIR filter. This decision will affect the characteristics of the filter and determine the design methods that can be used. Some of the questions to consider when selecting an IIR or FIR design are:

Linear Phase A linear phase filter exhibits the same delay at all frequencies. This ensures minimum distortion of time-domain characteristics. For example, non-linear phase could have a serious effect on the signaling pulses used in data communications systems but would have little effect on the intelligibility of speech signals.

FIR filters can be easily designed to have linear phase. If the filter needs to have linear phase then an FIR filter is usually the most appropriate.

Efficiency A filter implemented using poles (i.e. an IIR filter) can have a sharper transition region than a filter using only the same number of zeros (an FIR filter).

Therefore an IIR filter can usually be designed to meet the same amplitude response specifications using fewer delay elements and multipliers.

An IIR filter is appropriate if the implementation must use the minimum of memory and multipliers.

On the other hand, FIR filters can be implemented by fast convolution using FFTs. These are usually only appropriate for non-real-time applications because of the long latency.

Stability Since an FIR filter has no feedback elements and no poles it is always stable. An IIR filter can become unstable under certain circumstances (e.g. if the input causes arithmetic overflows).

A FIR filter is appropriate if some input conditions might cause the filter to become unstable.

Conversion of Analog Filters Conventional analog filters are described in the s-domain as rational functions and include poles. When these designs are converted to equivalent digital filters the results are IIR filters.

Therefore IIR filters are used when converting conventional analog filter responses (e.g. Butterworth or Chebyshev) to digital form.

Pole/Zero or Coefficient Estimation

The most critical part of the design process – and the one we will concentrate on – is determining the locations of the poles and/or zeros of the filter's transfer function.

Many design methods are available and the choice depends on the specifications to be met and whether the filter is FIR or IIR. In this course we will only study the most popular design method for each type of filter. These two techniques are sufficient for the large majority of filter designs.

We will use software that implements the so-called "Parks-McClellan" algorithm to design FIR filters with an "equiripple" amplitude response. This design method is optimal in the sense that the filters will meet the design requirements using the minimum number of zeroes.

For IIR design we will use the "Bilinear z-transform" to convert well-known s-domain transfer functions, (Butterworth, Chebyshev, Bessel, elliptic) into z-domain transfer functions with equivalent amplitude responses.

Realization Structures

If the filter will be implemented on a general-purpose computer with floating point hardware then the selection of the poles/zeros (or filter coefficients) normally completes the design.

However, digital filters are often implemented with DSP microprocessors or special-purpose hardware using integer arithmetic with relatively few (e.g. 16) bits of precision. Because of coefficient quantization and roundoff errors the structure of the filter (i.e. the sequence of steps used to compute the result) can have a significant effect on the performance of the filter.

We have seen that a given transfer function can be expressed in several equivalent ways (a polynomial in z^{-n}, a product of factors, or a sum of partial fractions). Each of these will be seen to corresponds to a particular computational structure. Although equivalent in the absence of roundoff and quantization errors these different structures will perform differently when implemented with finite-precision hardware.

We will study ways to convert our standard filter structure (direct I) to alternative structures (direct II, cascade and parallel).

Finite Word Length Effects

A digital filter using integer hardware must quantize the filter coefficients to a fixed number of bits (e.g. 16). These coefficients must also be scaled to avoid overflows.

Quantization changes the effective values of the coefficients and thus changes the response of the filter. The filter response must be re-evaluated with the quantized and scaled coefficients to make sure the requirements are still met.

Implementation

The details of the implementation will depend on the sampling rate, complexity of the filter and whether real-time operation is required.

For low sampling rates or off-line analysis conventional computers can be used. For audio frequencies special purpose "DSP" microprocessors are used. These have special features such as single-cycle multiply/accumulate instructions and dual memory buses which speed up common DSP algorithms such as FIR filtering and the FFT. For high-speed applications (e.g. video) hardware must be custom-designed.