Finite-Precision Filter Implementation

This lecture covers some of the details of implementing FIR and IIR filters using finite-precision hardware. After this lecture you should be able to: (1) quantize filter coefficients for use in integer hardware, (2) check the effect of coefficient quantization on the filter response, and (3) convert an IIR transfer function to cascade and parallel forms. This material is covered is sections 6.9.1, 7.5 and parts of 7.6 of the text.

Fixed-Point Implementations

Integer arithmetic hardware is often used in real-time DSP systems because it's less expensive than floating point hardware. We have already considered the effect of finite word length on A/D converter quantization noise. Here we consider the effect of finite-precision (integer) arithmetic on the response of a filter.

There are two main effects. The first is that quantization of the filter coefficients changes the response of the filter. This may prevent the filter from meeting the original specifications or, in the case of IIR filters, it may make the filter unstable by moving poles outside the unit circle. The second effect is that the results of some operations (addition or multiplication) may give results that overflow the number of bits available.

There is a lot of material on these topics in the text. Due to limited time we will only cover the basic principles.

Coefficient Quantization

For a fixed-point implementation the filter coefficients must be quantized to B bits. A typical format is to consider that the integer represent a fractional number whose absolute value is less than 1. Quantization can be done by multiplying the coefficient values by 2^{B-1}, truncating to an integer and dividing by 2^{B-1} (the extra bit is required for the sign). Any required scaling of the coefficients should be done first (see below).

Quantization changes the values of the coefficients. To make sure the filter specifications are still met the filter response must be re-computed, for example, using `freqz()`.

Exercise: Quantize the coefficients 0.1 and 0.2 to 4 bits.

Alternative IIR Filter Structures

For FIR filters a straightforward implementation of the direct form flow graph results in a transversal filter structure. This structure is not very sensitive to the effects of quantization and overflow is relatively easy to control. DSP microprocessors have special hardware for efficient implementation of the transversal filter structure. For these reasons this is the preferred structure for FIR filters.

On the other hand, IIR filters are seldom implemented in direct form because small errors caused by quantization of the coefficients can lead to large changes in the filter response, especially for poles that are close to the unit circle. This sensitivity to coefficient errors increases with filter order.

Instead, IIR filters are typically converted to cascade (serial) or parallel combinations of first or second order filters. The overall response is the same as that of the original filter but it is less sensitive to coefficient quantization errors.

The new filter structure will also have completely different filter coefficients. To determine the effect of quantizing the coefficients *of a cascade or parallel realization*, the new coefficients are quantized and then the filter is converted back to direct form. The frequency response and pole positions of this direct form filter will be those of the filter with the cascade/parallel structure and can be found as before.

Cascade Structure

To implement an IIR filter as a cascade of lower-order sections the numerator and denominator of the transfer function are factored. The numerator and denominator can then be expressed as products of first and second order polynomials (if the coefficients are to be real). The numerator and denominator polynomials can be paired in different combinations and arranged in different orders. This pairing and ordering should be chosen to minimize the gain of each stage and to reduce the magnitude of the intermediate values. For example, poles are paired with their closest zeros and the poles closest to the unit circle are placed later in the cascade.

Parallel Structure

For a parallel realization the transfer function is expanded into a sum of terms using partial fraction expansion. Each term will be a ratio of a first and a second order section or a zero and first order section. In this case ordering is not important and scaling is not as important because the values from each of the sections are only combined once.

Overflow and Scaling

When using integer arithmetic there is the possibility that products or sums will overflow. This can be controlled by scaling the filter coefficients and/or the input. However, if the coefficients are made too small the effective number of bits is reduced. This increases the coefficient quantization error and reduces the signal to noise ratio. Scaling thus involves a compromise between overflow and signal to noise ratio. There are a number of ways of estimating an appropriate scaling factor so as to compromise between the likelihood of overflow and signal to noise ratio. We will probably not have time to study these.

In normal 2's-complement arithmetic an overflow causes the result to "wrap around" and the sign of the signal to change. As long as the final result is representable with the number of bits in the adder this is not a problem. However if the final result is too large the result will be a very large error. In some cases this can lead to instabilities and "large scale limit cycles." Some DSP adders can be set up to use so-called "saturation arithmetic" where overflows cause the result to be set to the maximum value rather than to wrap around. This may be desirable in some (but not all) cases.

There are a number of other relatively minor effects due to finite precision arithmetic that we will not cover.