Decimation and Interpolation

This lecture describes how to change the sampling rate of sampled signals.
After this lecture you should be able to: (1) design and implement single-stage interpolators and decimators, (2) combine interpolators and decimators to achieve sampling rate conversion by a rational number, (3) specify the filter required in each stage of a multi-stage decimator or interpolator, and (4) find the optimum number of stages and rates for a particular interpolation/decimation ratio.
This material is covered is Chapter 8 of the text.

Introduction

It's often desirable to change the sampling rate of a sampled signal. Decreasing the sampling rate is called *decimation* and increasing the sampling rate is called *interpolation*.

One application of sampling rate conversion is to feed data sampled at one rate into a system designed for a different rate. For example, CDs use a sampling rate of 44.1 kHz while DAT recorders use a sampling rate of 48 kHz.

Another common application is to sample an analog signal at a rate higher than twice the signal bandwidth and then reduce the sampling rate using a decimator. This relaxes the specifications required for the analog anti-aliasing filter.

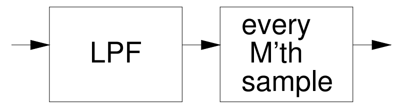
Decimation and interpolation are techniques from a field called *Multirate DSP*. These techniques are important because they often make it possible to process signals sampled at rates that would otherwise be impractical.

Decimation

Consider a signal sampled at a rate F_s but with a bandwith of $F_s/2M$. This signal has been sampled at a rate M times higher than necessary. We can reduce the sampling rate by a factor of M by extracting only every M'th sample.

However, if the bandwidth of the original signal was not limited to $F_s/2M$ then taking every M'th sample (which is equivalent to sampling) will cause aliasing. In this case we can low-pass filter the digital signal to reduce the bandwidth to $F_s/2M$ before taking every M'th sample.

A decimator thus consists of a digital low-pass filter followed by a device that takes every M'th sample:

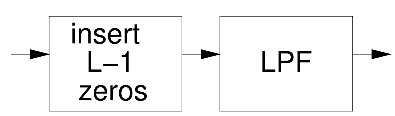


Exercise: Assume the decimator input is a signal whose spectrum was a triangle centered at DC, extending to ± 2 kHz that was sampled at 4 kHz. Sketch the input spectrum, the spectrum after ideal low-pass filtering to 1 kHz and after taking every 2nd sample. Include frequencies over the range ± 4 kHz.

Interpolation

An interpolator inserts $L-1$ zero samples in between each input sample to raise the sampling rate by a factor L. If F_s is the sampling rate of the input, then the output signal is at a sampling rate LF_s. This increases the sampling rate but also produces a "sampled" signal. The "sampling" causes the spectrum of the original signal to be replicated about multiples of the sampling rate, F_s.

To remove these "images" the signal must be low-pass filtered to remove components above $F_s/2$. The output must also be scaled by L to obtain the same average output level as the input.



Exercise: Assume the decimator input is the same as in the previous exercise. Sketch the input spectrum, the spectrum after inserting one zero sample between each input sample, and after ideal low-pass filtering to 2 kHz. Include frequencies over ± 8 kHz.

Multi-Stage Conversion

If the required sampling rate conversion is not an integer, then a rate-L interpolator can be followed by a rate-M decimator to obtain a sampling rate change of L/M.

Another reason for using multiple interpolation or decimation steps is to reduce the computational difficulty of each step. For example, if decimation by a factor of 12 is required, it is more efficient to decimate by $M = 4$ and then to decimate by $M = 3$. The first decimation step reduces the sampling rate and thus reduces the amount of processing that needs to be done by the second decimator.

Design

The design of a single-stage decimator or interpolator simply requires choosing M or L and designing the low-pass filter. The filter is usually an FIR filter designed using the optimum equiripple method. As usual, the filter specifications will be determined by the application.

When multiple stages are cascaded the (worst-case) passband ripple will be sum of the passband ripple for each stage and the (worst-case) minimum stopband attenuation will be the minimum stopband attenuation of the stage with the "worst" attenuation.

Exercise: A two-stage decimator requires a maximum passband ripple of ± 0.01 and minimum stopband attenuation of $.01$ (linear units). If each stage will be designed using the same specifications, what should the ripple specifications be for each stage?

The stopband and passband edge frequencies will depend on the input signal and interpolation rate for that stage. Typically the passband will include the original signal frequencies (which remain fixed throughout) and the stopband will cover the alias or image frequencies (which might change from stage to stage).

Selecting the number of stages and the interpolation/decimation rate for each stage can be done by evaluating each possible combination. For each possible set of factors of the desired overall decimation/interpolation ratio the required filter order for each stage is estimated (e.g. using the `remezord()` function). The set of factors that results in the smallest overall number of processing steps is then chosen as the best solution. Note that the number of operations for each stage will depend not only on the number of filter coefficients for that stage but also on the sampling rate at that stage.

Exercise: What are the possible choices (number of stages and rates) for decimating by an overall rate of 12?

Note that the logic for real-time rate conversion can become rather complicated because the different filter stages need to execute at different rates.

Implementation

One of the reasons FIR filters are preferred for interpolation and decimation filters is that the transversal filter structure allows significant reductions in the number of operations required. For the interpolator filter only one output sample has to be computed for every M input samples. In the case of the decimator filter only one of each L input samples will be non-zero. These facts can be exploited to reduce the number of computations required to compute the filter output.