Filtering Tricks

DSP Trick: Simple Filter Coefficient Interpolation

```
From: ericj@primenet.com.nospam (Eric Jacobsen)
Subject: DSP Trick - Simple Filter Coefficient Interpolation
Date: 23 Oct 1999 00:00:00 GMT
Newsgroups: comp.dsp
THIS WORK IS PLACED IN THE PUBLIC DOMAIN
```

Often it is necessary to design a FIR filter coefficient set that is longer than can be practically computed using common automated routines like Parks-McLellan or its Remez subroutine. In most cases this problem can be alleviated by using P-M or Remez to design a short coefficient set and then interpolating it up to the desired length. A simple interpolation method that avoids some of the distortions that may be associated with linear, spline, or polynomial interpolation utilizes the interpolating capabilities of the DFT. The procedure may be achieved as follows:

1. Design a shortened FIR filter that can be computed using an appropriate design technique.
2. Take a forward DFT of this shortened FIR coefficient set.
3. Zero pad the result up to the desired length of the final filter.
4. Take the inverse-DFT of the zero-padded vector.
5. The real part of the inverse-DFT is the interpolated coefficient set of the desired length.

It is always prudent to verify that the interpolated filter has the desired response.

Eric Jacobsen Minister of Algorithms, EF Data Corp.

Editor's Note: This trick has some similarities to another dspGuru article, "How to Interpolate in the Time-Domain by Zero-Padding in the Frequency Domain".

DSP Trick: FIR Filtering in C

```
Subject: DSP Trick: Filtering
From: rainer.storn@infineon.com
Date: 1999/04/21
Newsgroups: comp.dsp
THIS WORK IS PLACED IN THE PUBLIC DOMAIN
```

Name: Filtering in C

Category: Programming Trick

Application: FIR-Filtering

Advantages: Simple and fast

Introduction:

This is more like a trick in C but might also be applicable in DSP environments. It allows to compute FIR-filtering in a fast manner when the filter length contains a factor of, let's say 4 (other factors are also possible).

The Trick:

Let's suppose you have an array for the FIR filter coefficients w[LEN] and an array for the delay line x[LEN]. A straightforward approach to do the filtering operation would be:

```
   y=0; //will contain your result
   for (i=0; i<LEN; i++)
   {
      y = y + w[i]*x[i];
   }
```

However, many processors don't like the index calculations, so it is sometimes advantageous to do less index calculations. Here's how it goes (suppose LEN is divisible by four. As stated above, the trick basically also works for other factors):

```
//-----Initialization--------------------------------------------
   w_end_ptr  = &w[LEN]; // load sentinel
   w_ptr      = w;
   x_ptr      = x;
   y0         = 0.; // we assume floating point here, so scaling is
   y1         = 0.; // not an issue
   y2         = 0.;
   y3         = 0.;
//----Do the filtering-------------------------------------------
   while(w_ptr != w_end_ptr)
   {
      y0 = y0 + w_ptr[0]*x_ptr[0];
      y1 = y1 + w_ptr[1]*x_ptr[1];
      y2 = y2 + w_ptr[2]*x_ptr[2];
      y3 = y3 + w_ptr[3]*x_ptr[3];
      w_ptr = w_ptr + 4;
      x_ptr = x_ptr + 4;
   }
   y = y0+y1+y2+y3;         // y will contain the filtering result
```

I had pretty good result with this on a SPARC.

Rainer Storn

DSP Trick: Filtering in QAM transmitters and receivers

```
Subject: Re: DSP Tricks
From: Allan Herriman
Date: 1999/04/22
```

```
Newsgroups: comp.dsp
THIS WORK IS PLACED IN THE PUBLIC DOMAIN
```

Name: Filtering in QAM transmitters and receivers. When *NOT* to do what the textbooks tell you to do.

Category: Hardware architecture, or implementation

Application: QAM receivers (with hardware emphasis)

Advantages: The textbook descriptions of QAM receivers sometimes miss practical details. "Optimal" solutions may not be the best ones... These tricks are simply a list of possible reasons for deviating from "normal" QAM filter design.

Introduction: (what the textbooks tell you to do)

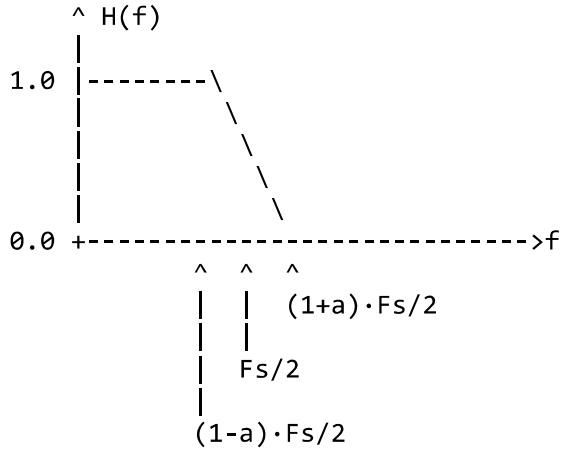
Any modem which modulates a linear channel (AM, ASK, FM, FSK, PM, PSK, BPSK, QAM, QPSK, or even baseband signalling) subject to noise will use filtering to improve the error rate in the receiver.

In general, there will be filters on the output signals (tx) and the input signals (rx), and also the bit in the middle (the channel).

There are two parts to this:

1. Eliminate ISI (Inter Symbol Interference) due to limitations in the frequency or phase response of the channel.
2. Use matched filtering to produce a maximum likelihood receiver.

1. ISI can be eliminated if the channel (including tx and rx filters) frequency response is (1) linear phase, and (2) has symmetry about a point at half the symbol rate (Fs/2).



'a' here is actually 'alpha' - the rolloff factor. (Although I have seen (1+alpha) used instead of alpha.) Alpha can be between 0 and 1, but commonly this will be 0.3 to 0.5 (or 1.3 to 1.5 using the other definition.) for 30% to 50% excess bandwidth

For some reason (because the maths isn't too hard?), most implemenations use a "raised cosine" response. The rolloff section between (1-a)·Fs/2 and (1+a)·Fs/2 is actually a half cycle of a cosine wave.

Note: it is also possible to use an adaptive equaliser in the receiver (either before or after the symbol decisions are made) which can reduce ISI. But this is "simply" a filter which adapts its response so the above requirement is met. Adaptive equalisers are used when the channel response is unknown or changing. They may either "adapt" to a training sequence and then remain fixed (like a fax machine), or they continuously adapt.

2. A matched filter will produce the lowest errors in the receiver output for a channel which adds white gaussian noise (an AWGN channel) if the rx filter impulse response is the time inverse of the tx pulse shape. (The tx pulse shape is determined by the tx filter.) In the frequency domain, this means that the magnitude responses of the rx and tx filters are the same, but the phase responses are opposite (and the combination has zero phase (linear phase in practice)). The matched filter output is only valid at the symbol sampling instant.

(This was inherent in the maths. If you want to know more, look at a textbook.) For example, if we transmit square pulses, then the rx filter should have a square impulse response. This would be an integrate-and-dump filter.

3. Combining 1 and 2 results in the following:

An optimal modem will use root-raised cosine filtering in the tx and rx filters. (A root-raised cosine filter puts "half" the response in the tx and "half" in the rx filter, so that the product in the frequency domain is a raised cosine.) The total channel reponse will have zero ISI, and the tx and rx filters are the same, so we have minimised the probability of errors.

The Tricks

The above description can be found in any communications textbook. Now for what the textbooks leave out: some examples of when *not* to use "optimal" filters.

Trick #1:

Must meet transmit spectral mask because:

1. Certain regulatory bodies place restrictions on the tx spectrum from a modem. For RF modems, the out-of-band emissions sometimes have to be < -80dBc.
2. Sometimes, the tx signal will interfere with the rx signal at the same end of the link in nearby channels. This is known as NEXT (Near End Cross Talk). In the case of an RF modem, the tx signal can be more than 100dB stronger than the rx signal, so NEXT can be a big problem.

Both of these place limits on the tx filter. This will entail:

1. Using a small alpha.
2. Truncating the tails of the tx filter frequency response.
    - This will result in degraded performance.
    - Truncate the rx filter reponse as well.
3. Using a non-root-raised cosine tx filter. Pick one that allows a sharper rolloff.
4. Allocating more of the raised cosine filter to the tx, and less to the rx

Trick #2:

Interfering signal has non-white spectrum. (AWGN assumption was made in the matched filter derivation.) Known narrowband interferers can be handled by putting a notch in the rx filter. If the notch is very narrow, the tx filter needn't be changed. Adjacent channel interference can be handled by making the rx filter slightly narrower. (See Trick #1 above)

Trick #3:

Symbol timing recovery problems. A matched filter produces a maximum likelihood estimate of the input symbol at a particular instant only. This assumes that this instant is known. Some simpler symbol-timing recovery schemes may require sub-optimal filtering to work. For example, wideband rx and tx filters allow signal transition detection to be used for symbol timing recovery. (This is how a UART works.) Symbol timing recovery is usually easier with larger alpha. (Books could be written about symbol-timing recovery. Any takers?)

Trick #4:

When one of the filters cannot be controlled. Perhaps the receiver uses analog filtering only, possibly in a SAW filter in the IF (passband) or RLC filter at baseband (BTW, 2nd and 3rd order butterworth have been used here). This filter will only be rough approximation for a root-raised-cosine, and will not have a linear phase response. This can be compensated for in the (FIR) tx filter.

Trick #5:

When there are significant non-linearities (in the tx output amplifier). Usually, the requirement will be to have the smallest amount of AM in the tx, which allows the average output power to be higher for a given amount of spectral spreading (due to the non-linearity). This may require wider tx filters and narrower rx filters. Useful where power efficiency is important (satellite links, handheld equipment, etc). There is also a case for using a larger alpha here. In extreme cases, it is possible to pick a modulation scheme that has a constant-amplitude constellation. (OQPSK, GMSK, etc.)

Trick #6:

When the rx filter is inside a feedback loop controlling carrier phase or frequency tracking. The group delay of the rx filter limits the tracking bandwidth of these loops (due to stability considerations). If a wider loop bandwidth is required (perhaps because of capture range or perhaps poor phase noise performance in the up- and downconverters), then the rx filter may need to be changed if it is not possible to move it outside the loop. In this case, allocate more of the raised cosine filter to the tx, and less to the rx (or try harder to move it outside the loop).

DSP Trick: Using Parks-McClellan to Design a Non-Linear Phase FIR Filter

```
From: ericj@primenet.com.nospam (Eric Jacobsen)
Subject: DSP Trick - Using P-M to design a non-linear phase FIR filter.
Date: 23 Oct 1999 00:00:00 GMT
Newsgroups: comp.dsp
```

It is possible to use the Parks-McClellan algorithm to design FIR filters with non-linear phase response. For example, a FIR filter with the equivalent response of a Butterworth filter can be designed using the P-M routine.

First, take a common Butterworth description like that in Parks and Burrus where H(x) is a complex function. Create two PM input grids using the real and imaginary components of the Butterworth response. Use PM (or "remez" or whatever it's called on your favorite system) to design two FIR filters using the respective input grids, but turn the 'Hilbert' switch on for the one derived from the imaginary component. Sum the results, i.e., add together the nth coefficients of each filter to create a single N-tap filter.

The resulting FIR filter (assuming you've done your job to make sure everything converges) will have the desired response from the original expression used to generate the PM input grids.

Eric Jacobsen
Minister of Algorithms