EE379A EXTRA CREDIT PROJECT (WINTER 2005)

Hoi Wong

(wonghoi@ieee.org)

ABSTRACT

Problem 3.34 in the course reader (a final exam problem in year 2000) was quite a challenge to quite a few of us. Scaling, oversampling, anti-aliasing, etc, are often confusing. It was quite difficult for me to get what part a) is actually asking for due to the way it was worded. The phrase 'decide a set of parameters' in part b) and c) were not that clear to me in terms of what parameters we were asked for.

The anti-alias filter specified in Fig. 3.65 has $l = 2$ plugged into $\frac{l}{2T}$ already. That made the anti-aliasing filter quite obscure to me when I first looked at it.

Therefore, this article tries to clarify the problem statement, discuss implementation issues, and provide some implementation tools to aid solving 'real-life' problem like this one.

Problem 3.34 *Finite-Length Equalizer Design*

A stream of message x_k is to be sent through a channel[1] with impulse response

$$h(t) = \frac{1}{T}\frac{1}{1+(\frac{10^7}{3}t)^2}$$

and with 1 mW transmission power, i.e.

$$P_{\mathbf{x}} = \frac{\varepsilon_{\mathbf{x}}}{T} = 1 \text{ mW}$$

The channel is AWGN with noise power spectral density

$$S_n(f) = \frac{N_0}{2} = \text{-86.5 dBm/Hz}$$

QAM transmission system is used with the following parameters:

$$\frac{1}{T} = 1 \text{ MHz} \quad (1)$$

$$f_c = 600 \text{ kHz} \quad (2)$$

[1] This is not a baseband channel

For the QAM transmission, the baseband pulse shape $\varphi(t)$ is given to be a SRRC pulse with 10% excess bandwidth. This is also commonly called the 'roll-off factor', α. For our case, $\alpha = 0.1$.

The goal of the problem is to design a *digital* FIR equalizer. However, we are only asked for design parameters (such as ν, the number of significant taps) and performance parameters such as SNR. There's no need to find the actual filter taps, though `dfecolor.m` does it for you.

Since we are going to design a *digital* equalizer (that includes *digital* matched filter), we need to pass the channel output into an anti-aliasing filter and then oversample. We are given oversampling factor

$$l = 2$$

and the following anti-aliasing filter:

$$F(f) = \begin{cases} \sqrt{T} & |f| \le \frac{l}{2T} \\ 0 & else \end{cases}$$

Figure 3.65 illustrates the transmission system. Please note the following typos:

i) -85 dBm/Hz should be changed to -86.5 dBm/Hz
ii) $p(t)$ does NOT include the anti-aliasing filter

Using our regular notations, the pulse response is

$$p(t) = \varphi(t) * h(t)$$

The overall pulse response that incorporates the anti-aliasing filter $f(t)$ is defined to be

$$\tilde{p}(t) = \varphi(t) * h(t) * f(t)$$

Tips

i) Since we are trying to simulate continuous-time signals/channels on a computer, we should sample the continous time signals (with given function) fast enough to avoid aliasing. $H(f)$ is a symmetrical exponential in frequency

domain, so sampling at T/100 would be fast enough. (Thanks to Edward for pointing that out in problem session.)

ii) When we 'fake' continuous time signals operations on in the computer by operating with digital samples (obtained from sampling above Nyquist frequency, of course), we need to correct them with scaling factors according to Table 3.1 in the course reader.

There are people who notationally distinguish vectors from scalars and there are people who don't. Having taken EE263, I started to expect the possibility that vectors may be not be boldfaced, underlined or denoted with an arrow. However, in EE379A, Prof. Cioffi DOES boldface/underline vectors. So, $||p||^2$ in the table actually meant $||p(t)||^2$, not $||\mathbf{p}||^2$ or $\sum_k |p_k|^2$.

Note that we also 'faked' continous time noise $n(t)$ as digital samples in the computer, so the noise variance should be scaled by its sampling period. Make sure you consider this to SNR_{MFB} consistently.

iii) Implementing SRRC manually might be a painful task. MATLAB Communications Toolbox offer 'Raised Cosine Filter Design' and SRRC pulse can be obtained through the function `rcosine` with appropriate parameters. Even easier, you can directly filter a signal with SRRC filter using `rcosflt`.

Here are some hints to use `rcosine` routine for this problem:

`Fd`	(input) symbol frequency, i.e. $\frac{1}{T}$
`Fs`	filter sampling frequency, `Fs`=$l\cdot$`Fd`
`'sqrt/fir'`	get FIR coefficients of SRRC
`R`	Roll-off factor α
`DELAY`	For FIR, it's half the length of taps in the pulse in terms of T (i.e. `1/Fd`. Make it long enough to 'fake' an IIR for part a).

`rcosflt` essentially the same style as `rcosine`. It basically filter your signal with the pulse response generated with `rcosine`. The flag `'Fs'` can be used if your input signal is already oversampled. Otherwise, `rcosflt` will do the oversampling for you according to l=`Fd/Fs` entered.

Here are the questions:

a) Since we are working with FIR, we need to truncate our IIR pulse response. Find ν, the number of significant taps (the taps we keep during truncation), that gives less than 5% truncation[2] error in $||p(t)||^2$.

NOTE[3]: $p(t)$ in the norm above really meant the continuous time version, while we are actually putting in ν discrete time taps for `dfecolor.m`. To compute the L_2-norm properly for finding the ν that satisfy the criterion, we need to obtain the $\hat{p}(t)$ by truncating $p(t)$ that we 'faked' by sampling at $T/100$. This can be easily done by corresponding $\nu = 1$ to 100 samples in the 'faked' continuous-time $p(t)$.

b) i) Calculate SNR_{MFB}

b) ii) Determine a reasonable set of parameters (basically all the inputs to `dfecolor.m`) for designing a FIR MMSE-LE equalizer.

b) iii) Under the condition that $P_e = 10^{-7}$, calculate the data rate R of the above system with equalizer in ii) using both integer and possibly non-integer number of bits/symbol b.

c) i) Repeat part (ii) and (iii) of part b using FIR MMSE-DFE equalizer design.

c) ii) Draw that FIR MMSE-DFE receiver.

d) Can we improve data rate R on this channel by changing symbol rate $\frac{1}{T}$ or carrier frequency f_c? If so, suggest a way to do it.

Final comments

The notational covention in the course reader has $\sqrt{T}$ absorbed[4] in the anti-aliasing filter to make the mathematics easier. Therefore, you can think of it as the true anti-aliasing filter has unit gain and we multiply by $\sqrt{T}$ before sampling.

Therefore, if the input signal doesn't need anti-aliasing, you still need to multiply it by $\sqrt{T}$ before sampling!

[2]For people who have lighter background in signal processing, that means $||p(t) - \hat{p}(t)||^2 < 0.05||p(t)||^2$, where $\hat{p}(t)$ is the truncated pulse response.

[3]It's natural to mistaken the question by computing the criterion with p_k obtained by sampling at T/l. That doesn't work!

[4]Yes. The $\sqrt{T}$ makes it LOOK like a scaling factor to make the power spectrum unit area. That's especially misleading when the simplified anti-alias filter icon in the figure attains non-zero value from 0 to $1/T$. Note that the $1/T$ in the anti-alasing filter actually comes from plugging $l = 2$ into $0.5T$. The $\sqrt{T}$ factor won't change for any l you use, so the $\sqrt{T}$ is actually an external scaling factor that has nothing to do with the anti-alasing operation