ECE-501 **Introduction to Analog and Digital Communications** **Winter 2008**

Homework #2 Jan. 11, 2008

HOMEWORK ASSIGNMENT #2

Due Fri. Jan. 18, 2008 (in class)

Reading:

1. 2.6-2.7, though the sections on Paley-Wiener Criterion, Pulse Response of Ideal LPFs, and Approximation of Ideal LPFs are optional.
2. Ch. 2.10-2.12, though the sections on Fast Fourier Transform Algorithms and Computation of the IDFT are optional.

Problems:

1. (a) Prove that the ideal (zero-phase) LPF has a sinc impulse response:

$$H(f) \;=\; \begin{cases} 1 & |f| \le B \\ 0 & |f| > B \end{cases} \;\;\overset{\mathcal{F}}{\longleftrightarrow}\;\; h(t) \;=\; 2B\,\mathrm{sinc}(2Bt)$$

   (b) Prove that the ideal linear-phase LPF has a delayed sinc impulse response:

$$G(f) \;=\; \begin{cases} e^{-j2\pi f t_o} & |f| \le B \\ 0 & |f| > B \end{cases} \;\;\overset{\mathcal{F}}{\longleftrightarrow}\;\; g(t) \;=\; 2B\,\mathrm{sinc}(2B(t-t_o))$$

   (*Hint:* Use the result of part (a) with FT property 5 from Haykin.)

2. In this problem you will use MATLAB to study causal linear-phase LPFs.

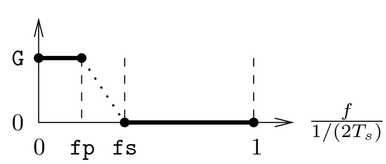
   (a) One way to design a causal linear-phase LPF is to truncate the t_o-shifted sinc impulse response

$$h(t) \;=\; 2B\,\mathrm{sinc}(2B(t-t_o))$$

   so that $h(t) = 0$ for $t < 0$ and $t > 2t_o$. For a single-sided bandwidth of $B = 20$ Hz and an impulse response length of $2t_o = 0.5$ seconds, generate a $T_s = 0.001$-sampled version of this impulse response in MATLAB and use `plottf.m` to plot the impulse and frequency magnitude responses. Comment on the non-ideality of the magnitude response of this filter.

   (b) Another way to generate a causal linear-phase LPF is to use MATLAB's built-in filter design routines. Here you will use `firls.m` to repeat the filter design task in part (a). As described in the lecture, `firls` is used as follows:

```
h = firls(Lf, [0,fp,fs,1], [G,G,0,0])/Ts;
```



   `Lf`+1 = impulse response length

   {0, `fp`}, {`fs`, 1} = *normalized* freq pairs

   {`G`, `G`}, {0, 0} = corresp. magnitude pairs

For a fair comparison with the truncated-sinc design of part (a), use the same values of t_o, T_s, and passband gain, and set the passband and stopband cutoffs to be $0.9B$ and $1.1B$ Hz, respectively, for the same B. (*Hint:* This implies `Lf` $= 2t_o/T_s$ and `G` $= 1$. Also, don't forget to normalize the cutoff frequencies by $\frac{1}{2T_s}$ when setting `fp` and `fs`!)

(c) Repeat (b) using `firpm` in place of `firls`.

(d) Repeat (b) using `fir2` in place of `firls`.

(e) Comment on the qualitative differences between the filters designed in parts (a)-(d).

3. In this problem you will experiment with the effects of filtering in MATLAB. Use a sampling rate of $\frac{1}{T_s} = 1000$ Hz throughout.

   (a) Generate a random noise waveform of duration $t_{\max} = 1$ sec in MATLAB using `randn`. Plot the magnitude of the signal's Fourier transform using `plottf` (with the `'f'` option).

   (b) Design a causal linear-phase LPF with unit passband gain, single-sided bandwidth $B = 100$ Hz, and group delay $t_o = 0.25$ sec, as described in problem 2(b). Lowpass filter the noise waveform using `conv`, and plot the magnitude of the output's Fourier transform using `plottf` (with the `'f'` option). Remember to multiply the output of `conv` by T_s.

   (c) Design a causal linear-phase HPF with unit passband gain, cutoff $B = 100$ Hz, and group delay $t_o = 0.25$ sec, similar to problem 3(a). Highpass filter the noise waveform using `conv`, and plot the magnitude of the output's Fourier transform using `plottf` (with the `'f'` option).

   (d) Do the results of (a)-(c) look as expected? To compare them, it might be nice to plot the three magnitude responses on a single plot via the `subplot` command. (*Hint:* `help subplot`.)

4. In this problem we will learn about the inner workings of `plottf`, in particular how `plottf` approximates the Fourier transform (FT).

   Suppose that we are interested in computing the FT of an $x(t)$ which is non-zero on the interval $t \in [0, t_{\max}]$. For this we know $X(f) = \int_{-\infty}^{\infty} x(t)e^{-j2\pi ft}dt = \int_0^{t_{\max}} x(t)e^{-j2\pi ft}dt$. Though `plottf` is given access only to $\frac{1}{T_s}$-rate samples of $x(t)$, a Riemann-sum[1] approximation of the integral says

$$X(f) \approx T_s \sum_{n=0}^{N-1} x(nT_s)e^{-j2\pi f nT_s} \quad \text{for} \quad N = t_{\max}/T_s. \tag{1}$$

   Now, since we are going to plot pixels on a screen, we don't need to compute $X(f)$ at *all* values of f. Say instead that we only care to sample $X(f)$ at $f = \frac{k}{NT_s}$ for the N integers $k \in \{-\frac{N}{2}, \dots, \frac{N}{2} - 1\}$. (Here we have assumed that N is even; the odd case is treated similarly.) Thus, from (1),

$$T_s \sum_{n=0}^{N-1} x(nT_s)e^{-j\frac{2\pi}{N}kn} \approx X(\tfrac{k}{NT_s}) \quad \text{for} \quad k \in \{-\tfrac{N}{2}, \dots, \tfrac{N}{2} - 1\}. \tag{2}$$

   The routine `plottf` approximates the FT according to (2). If you find that `plottf` returns an answer which does not make immediate sense, the explanation can probably be found in (2).

   In the following MATLAB experiments, use sampling rate $\frac{1}{T_s} = 1000$ Hz and $t_{\max} = 3$ sec unless told otherwise.

[1] You learned about the Riemann sum in your first calculus class. Recall that the approximation in (1) becomes exact as $T_s \to 0$.

(a) In MATLAB, approximate the Dirac delta using the sampled waveform[2]

$$x(nT_s) = \begin{cases} \frac{1}{T_s} & n = 0 \\ 0 & n \neq 0 \end{cases}.$$

How does the FT returned by `plottf` compare to $\mathcal{F}\{\delta(t)\}$?

(b) In MATLAB, generate the sampled version of $\exp(j2\pi f_o t)$ on $t \in [0, t_{\max}]$ for $f_o = 9$ Hz. How does the FT returned by `plottf` compare to $\mathcal{F}\{\exp(j2\pi f_o t)\}$?

(c) Repeat (b) with $t_{\max} = 5$ and comment.

[2] We use $\frac{1}{T_s}$ here so that the Riemann-sum approximation gives $\int x(t)dt \approx T_s \sum_n x(nT_s) = 1$, since we know that $\int \delta(t)dt = 1$ for Dirac delta $\delta(t)$.