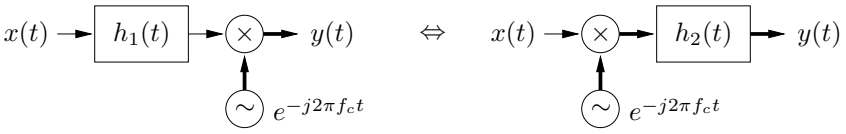
HOMEWORK ASSIGNMENT #6

Due Fri. Feb. 22, 2008 (in class)

Problems:

1. Write an expression for $h_2(t)$ in terms of $h_1(t)$.



   (*Hint*: We did something similar when deriving the complex-baseband channel representation.)

2. Show that
$$\sum_{n=-\infty}^{\infty} \delta(t-nT) \xrightarrow{\mathcal{F}} \frac{1}{T} \sum_{k=-\infty}^{\infty} \delta(f-\frac{k}{T}).$$
   (*Hint*: First show that $\sum_{n=-\infty}^{\infty} \delta(t-nT) = \frac{1}{T}\sum_{k=-\infty}^{\infty} e^{j2\pi kt/T}$ using Fourier series.)

3. Finally, you get a chance to play with analog FM communication. Use a sampling rate of 10 kHz.

   (a) Generate a message signal of single-sided bandwidth 50 Hz and length 0.5 sec, and normalize the message so that it's maximum amplitude equals 1. Plot the message in time and frequency domains using `plottf` to verify that it looks as expected.

   (b) FM modulate the message using carrier frequency 500 Hz and modulation index $D = 5$. (*Hint:* Use `Ts*cumsum(x)` to implement $\int_0^t x(t)dt$.) Plot the transmitted signal in time and frequency domains using `plottf` to verify that it looks as expected. Does Carson's rule hold?

   (c) FM demodulate the transmitted signal using the "discriminator" approach. Recall that this requires differentiating, rectifying, LPFing, subracting the DC offset term $2\pi f_c$, and scaling by $(2\pi k_{\text{f}})^{-1}$. (*Hint*: Use `diff(x)/Ts` to implement $\frac{d}{dt}x(t)$.) For your discriminator LPF, you are allowed a group delay of *at most* 5 *msec*.

   - Plot the rectified differentiated signal in time and frequency domains. A zoomed plot should help you choose your LPF stopband edge.
   - Plot the discriminator LPF in the time and frequency domains. Does it look as you expect? (Don't forget to try `firls`, `firpm`, and `fir2` before settling on one approach.)
   - Plot the recovered message in the time domain and superimpose the original message using a dashed red line for comparison. Are they close? (They should be!)

   I suggest including `randn('seed',0)` before generating the random message. This forces MATLAB to generate the same message every time, making it is easier to notice the effect of changes in your code.