MASSACHUSETTS INSTITUTE OF TECHNOLOGY

DEPARTMENT OF MECHANICAL ENGINEERING

2.161 Signal Processing - Continuous and Discrete
Fall Term, 2008

Quiz #2 (Take Home)
Due in class (1pm) on December 9, 2008

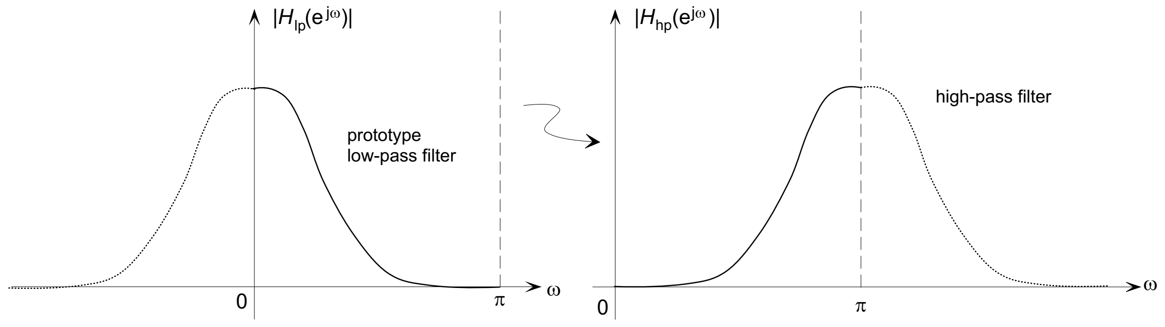
Notes:

- This is a take-home quiz.
- Collaboration (with anybody) is expressly forbidden.
- **Do not** spend excessive time on this quiz!
- There are 4 problems, answer them all.
- Partial credit will be given.
- You may ask the teaching staff for help - but that help will be generally limited to ensuring that you have access to the required data files.

Problem 1: (20 points)

Here is a simple way to design a digital high-pass filter from a low-pass design (this is *not* a method we talked about in class). Consider a prototype causal digital low-pass filter $\bar{H}_{lp}(e^{j\omega})$. The figure below shows how a digital low-pass filter may be transformed to a high-pass filter by simply translating (shifting) its frequency response $H(e^{j\omega})$ by π, that is we create the high-pass filter from the prototype:

$$H_{hp}(e^{j\omega}) = H_{lp}(e^{j(\omega-\pi)}).$$



(a) Assume that the original low-pass filter has a recursive difference equation

$$y(n) = -\sum_{k=1}^{N} a_k y(n-k) + \sum_{k=0}^{M} b_k x(n-k)$$

corresponding to the discrete-time transfer function

$$H_{lp}(z) = \frac{\sum_{k=0}^{M} b_k z^{-k}}{1 + \sum_{k=1}^{N} a_k z^{-k}}$$

Show that the difference equation of the new high-pass filter is

$$y(n) = -\sum_{k=1}^{N} (-1)^k a_k y(n-k) + \sum_{k=0}^{M} (-1)^k b_k x(n-k)$$

In other words, a high-pass filter may be constructed by simply alternating the signs of the coefficients in the low-pass difference equation!

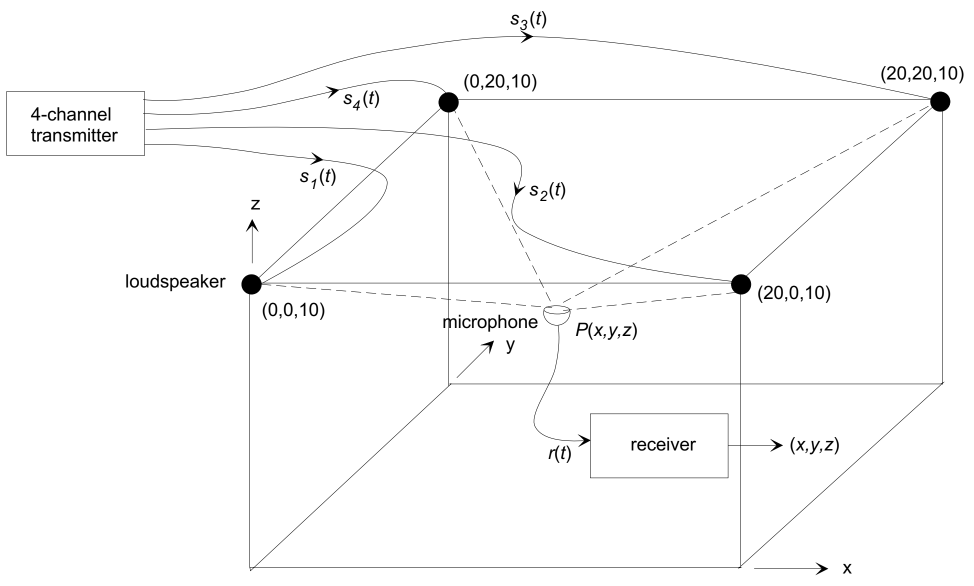
(b) Show that the two discrete-time impulse responses are related by

$$h_{hp}(n) = (-1)^n h_{lp}(n)$$

(c) How is the phase response $\angle H_{hp}(e^{j\omega})$ of the high-pass filter related to the phase response $\angle H_{lp}(e^{j\omega})$ of the prototype low-pass filter?

Problem 2: (25 points) An acoustic micro-GPS system.

An experiment on human movement requires that we monitor the position of a person's head during the experiment. The experiment is to be conducted in a room 20×20 ft. with a 10 ft. ceiling. Four loudspeakers have been installed in the top four corners of the room as shown below:



The human subject wears a microphone mounted on a helmet.

- At regular intervals the transmitter broadcasts four uncorrelated wide-band acoustic signals $s_i(t)$, $i = 1 \ldots 4$, simultaneously - each to one of the loudspeakers as shown above.
- The sampling rate used in both the transmitted and received signals is 100,000 samples/sec.
- The microphone signal $r(t)$ is fed to the receiver, which computes and stores the cartesian coordinates x, y, z of the subject's head.
- The transmitter and receiver have precision clocks that are synchronized, in other words the receiver knows the time origin of each transmitted signal.
- The speed of sound is 1125 ft/s.
- There is "significant" noise in the room.

The Problem: You are given a file `Q2Prob2.mat` containing two MATLAB data arrays: (1) A two-dimensional array `s(4,500)` containing the four transmitted waveforms s, and (2) a one-dimensional array `r` containing a single data record from the microphone. Your task is to determine the cartesian coordinates (x, y, z) of the location of the microphone at this time.

Notes:

- The system is (deliberately) over-determined. In fact you only need three speakers to solve this problem. Just as in the GPS system the use of redundant sources allows for situations when one source might be occluded from the microphone. Your solution should use all four speakers. Use a least-squares approach to estimate the coordinates x, y, z. Suggestion: use MATLAB's `lsqnonlin()`.

- Since this is *not* a course in numerical optimization, most of the grade will be based on setting up the data to pass to the least-squares solver.

- MATLAB's `lsqnonlin()` requires you to write a function that returns an error vector (not the objective function). For example `x = lsqnonlin(@myfun, x0)` uses the function `myfun()`. See the help.

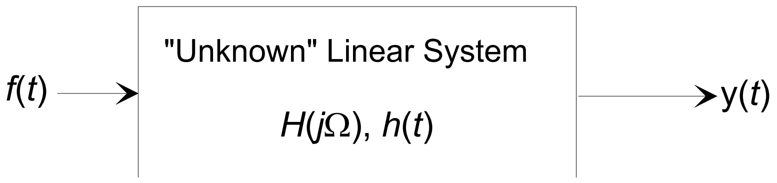
Problem 3: (25 points) System Identification

We talked about the correlation method of system identification briefly in class, that is

$$\hat{H}(\mathrm{j}\,\Omega) = \frac{R_{fy}(\mathrm{j}\,\Omega)}{R_{ff}(\mathrm{j}\,\Omega)}$$

where $\hat{H}(\mathrm{j}\,\Omega)$ is the estimated frequency response, $R_{ff}(\mathrm{j}\,\Omega) = \mathcal{F}\left\{\rho_{ff}(\tau)\right\}$ is the auto-power spectrum at the input, and $R_{fy}(\mathrm{j}\,\Omega) = \mathcal{F}\left\{\rho_{fy}(\tau)\right\}$ is the cross-power spectrum between input and output. This is all very nice in theory but, as we will see in this problem, things don't always work out quite so well in practice.

We have an unknown plant with frequency response $H(\mathrm{j}\,\Omega)$ and impulse response $h(t)$. Our goal is to find estimates $\hat{H}(\mathrm{j}\,\Omega)$ and $\hat{h}(t)$ of these quantities from input-output measurements. The system was driven with a wide-band input $f(t)$, and the input and output $y(t)$ were sampled.



The MATLAB file `Q2Prob3.mat` contains two data records (each of length 4096), recorded at 10,000 samples/sec. The vector `f` contains the input, and `y` contains the output.

(a) Use the two data vectors to estimate $H(\mathrm{j}\,\Omega)$ using the correlation method. Make plots of the magnitude and phase responses. (They are a bit messy!)

(b) From your $\hat{H}(\mathrm{j}\,\Omega)$ find an estimate of the impulse response $\hat{h}(n)$. Make a plot of the first 40 samples, using the `stem()` function.

(c) Now repeat (a) and (b), but this time truncate the correlation functions so as to save only the first 100 lags (a total of 201 samples). Apply a Hamming window of length 201 to the new correlation sequences $\rho_{ff}(n)$, and cross-correlation $\rho_{fy}(n)$, $-100 \leq n \leq 100$, before computing the power spectra.

(d) Discuss any improvements you see in the estimates of the frequency response and/or impulse response resulting from the windowing.

Notes:

- In `xcorr(x,y)` MATLAB defines the cross-correlation function of two real sequences as

$$R_{xy}(m) = \mathcal{E}\left\{x(n+m)y(n)\right\}.$$

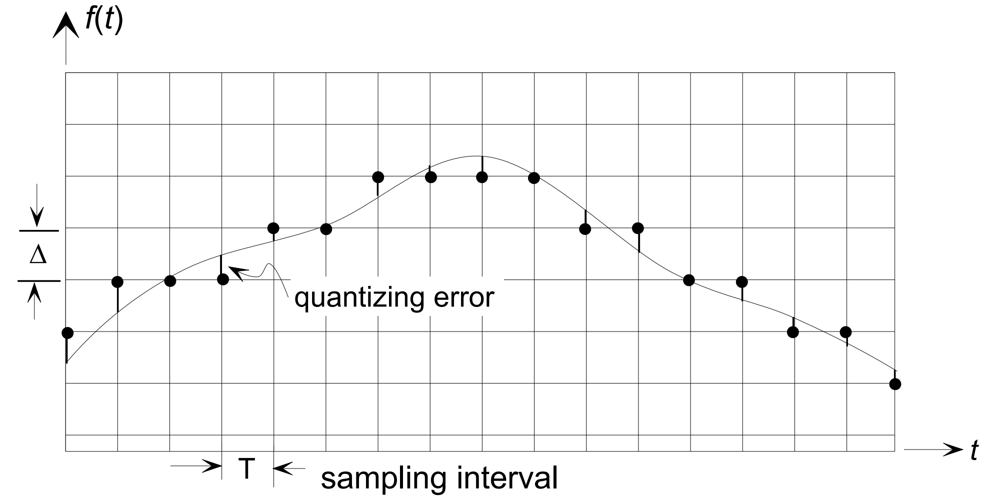
- Keep the following a secret: to help you assess your results, the "unknown" plant was created using

```
[b, a] = butter(3,[0.15 0.5])
```

- Hand in your MATLAB script and the plots you made.

Problem 4: (30 points) Image Deblurring Revisited

In the image deblurring problem of Prob. Set 9 we were very careful to give you high precision values (MATLAB's double precision) for the RGB components of each pixel. Practical images are not stored with such accuracy; the intensity values are *quantized* to a fixed precision, commonly 8 bits or a total of $2^8 = 256$ possible levels for each of the RGB values. This introduces an error into each of the pixels in the image. The following indicates a 1-D time-based quantized signal with quantization interval Δ:



The upshot is that quantization introduces random errors (or additive noise) into a sampled data set, and common (dubious) assumptions about the noise are that

- The noise is uniformly distributed across the quantization interval Δ, and
- the noise is white.

Whether or not the noise is truly "white" can be debated, but the significant thing for us is that it has significant high frequency content. The simple inverse filter you constructed for Prob. Set 9 had *extremely* large high-frequency gain ($\approx 10^{13}$), with the result that it could not tolerate any high frequency content in the blurred image without numerical failure.

In this problem we want you to modify your solution to the deblurring problem so that it will handle an image that has only eight bits of precision in the RGB values (with the resulting noise). Here are two simple empirical methods.

Method 1: A common way of dealing with noise amplification in inverse filters is to set an empirically chosen hard limit on the maximum gain in the inverse filter. In one-dimensional filtering, if the distorting filter is $H(\mathrm{j}\,\Omega)$ and the inverse filter is

$$H_{\mathrm{inv}}(\mathrm{j}\,\Omega) = \frac{1}{H(\mathrm{j}\,\Omega)}$$

we can choose a maximum value γ for $|H_{\text{inv}}(\mathrm{j}\,\Omega)|$ in a modified inverse filter

$$H'_{\text{inv}}(\mathrm{j}\,\Omega) = \begin{cases} \dfrac{1}{H(\mathrm{j}\,\Omega)} & \text{if } |H(\mathrm{j}\,\Omega)| > \dfrac{1}{\gamma} \\ \dfrac{\gamma\,|H(\mathrm{j}\,\Omega)|}{H(\mathrm{j}\,\Omega)} & \text{if } |H(\mathrm{j}\,\Omega)| \le \dfrac{1}{\gamma}. \end{cases}$$

Notice that this retains the phase response, but limits the magnitude to a maximum of γ.

Method 2: Reza made the following suggestion. He reasoned that although the deblurred image has higher frequency spectral content, is still essentially band-limited and we do not need to retain all of the high frequency content. Therefore, eliminate high frequencies completely if the gain exceed the criterion γ using the modified inverse filter

$$H'_{\text{inv}}(\mathrm{j}\,\Omega) = \begin{cases} \dfrac{1}{H(\mathrm{j}\,\Omega)} & \text{if } |H(\mathrm{j}\,\Omega)| > \dfrac{1}{\gamma} \\ 0 & \text{if } |H(\mathrm{j}\,\Omega)| \le \dfrac{1}{\gamma}. \end{cases}$$

To save you time, we have provided a basic deblurring file `Q2Prob4.m` that is based on our solution to Prob. Set 9. Modify and use this file to investigate this problem. We also have provided one of the blurred images used in Prob. Set 9, renamed to `Q2Prob4.mat`.

- `Q2Prob4.m` starts by quantizing the image to 8 bits and converting it back to double precision (but quantized) image with the line

```
        blurred8 = double(uint8(255*blurred))/255;
```

(a) Modify `Q2Prob4.m` to implement Method 1. Investigate the effect of γ on the resulting image. Start with $\gamma = 1000$, and find a value that you think is a good compromise between image sharpness and noise. (There is no "correct" answer)

(b) Use your value of γ chosen in (a) and implement Method 2.

(c) Compare the images from (a) and (b) (with the same value of γ), explain any differences you see. In particular, look at the edges of the images.

(d) Define the "energy" in a RGB image as the sum of the squared values of all pixels across the whole image plane and all the color planes. Also define the *signal to noise ratio* (SNR) as

$$\text{SNR} = \frac{\text{energy in the inverse filtered image}}{\text{energy in the inverse filtered noise}}.$$

Compute the SNR for Method 1 and Method 2 for values of $\gamma = 1000, 100, 30, 3$ before any intensity scaling. Does a high SNR generate a "better" output? Briefly (within a few lines) discuss any compromises you see with regard to SNR and deblurring.