2.161 Signal Processing: Continuous and Discrete

Fall 2008

MASSACHUSETTS INSTITUTE OF TECHNOLOGY
DEPARTMENT OF MECHANICAL ENGINEERING

2.161 Signal Processing - Continuous and Discrete
Fall Term 2008

Problem Set 8: FIR Linear Filters

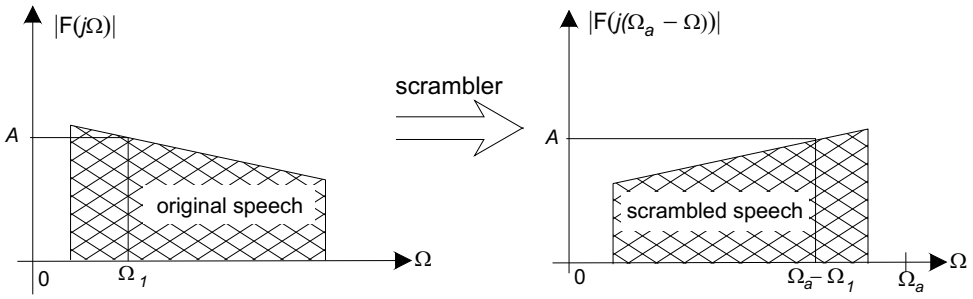
Assigned: November 6, 2008 **Due:** November 18, 2008

Problem 1: A Mini-project: An Audio Scrambler
Hint: We looked at this problem in Quiz 1. Look at the published solution.

Text-messaging is out-of-date! You and I have been communicating by sending voice-messages to each other as ".wav" files. It's not that I am paranoid, but everybody is out to get me! In particular, "they" have been intercepting all of our voice messages. But we are going to outwit "them" by using MATLAB to *scramble* our voice messages before sending them, making them unintelligible using a secret encoding scheme, and then we will *descrambling* the messages when we get them.

your task is to design and implement the speech encoder and decoder. the requirement is that the scrambled message should occupy the same audio bandwidth (approximately 300 – 3000 Hz) as the original message, and the descrambled message should have good fidelity.

We will use a frequency domain "mirroring" scheme, where the frequency components are flipped, so that high frequencies are translated to low frequencies and vice-versa. A spectral component with frequency Ω_1 will be translated to a new frequency $\Omega_a - \Omega_1$, where Ω_a is known to the sender and recipient. The resulting waveform is unintelligible until the frequency relationships have been restored. The scheme is shown below:

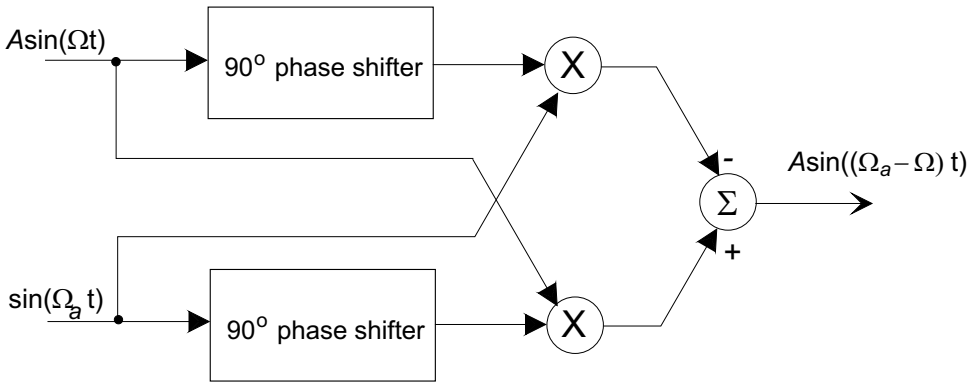


Thus, a spectrum as shown on the left will be translated to that shown on the right. The decoder will restore the correct frequency relationships.

Consider a component $A\sin(\Omega t)$, we want to translate it to $A\sin\left(\left(\Omega_a - \Omega\right)t\right)$. From elementary trigonometry

$$\begin{aligned} A\sin\left(\left(\Omega_a - \Omega\right)t\right) &= A\left(\sin(\Omega_a t)\cos(\Omega t) - \cos(\Omega_a t)\sin(\Omega t)\right) \\ &= A\sin(\Omega_a t)\sin(\Omega t + \pi/2) - A\sin(\Omega_a t + \pi/2)\sin(\Omega t) \end{aligned}$$

which leads to the following time-domain processing to shift a single frequency component.

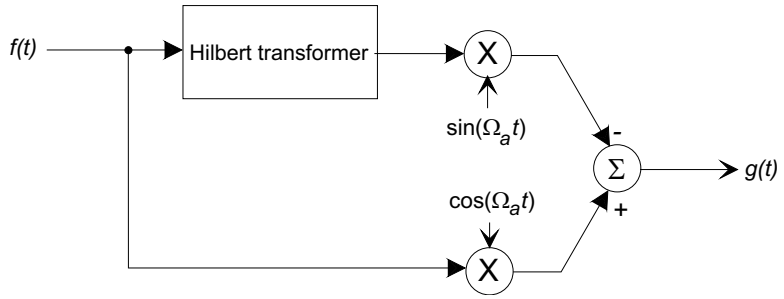


A $\pi/2$ phase-shifter is easy to build for a single frequency, such as $\sin(\Omega_a t)$ but when the input signal, such as a speech signal, contains more than just a single component each spectral component must be shifted by 90°. This requires an all-pass filter with a constant phase shift, which is a difficult design task for a continuous filter. Approximations as active or passive filters are available.

The Hilbert transformer is an all-pass filter with a transfer function

$$H(j\Omega) = \begin{cases} j & \Omega > 0 \\ -j & \Omega < 0, \end{cases}$$

which is exactly what we want. The following shows the scrambler implemented with a Hilbert transformer



Then $G(j\Omega) = F(j(\Omega_a - \Omega)$.

(a) Practical implementations of Hilbert transformers are approximations. For example, suppose an implementation has ripple in its passband so that at a particular frequency Ω_o the transfer function is

$$\begin{aligned} |H(j\Omega_o)| &= 1 + \delta \\ \angle H(j\Omega_o) &= -\frac{\pi}{2} \end{aligned}$$

where δ is a perturbation from the ideal response. Find the output $g(t)$ when the input is $f(t) = A\sin(\Omega_o t)$. Have any "spurious" spectral components been introduced? (You can do this with simple trigonometric identities.)

(b) Your task is to design, implement, and test an audio scrambler and descrambler using this encoding technique:

- You should write a pair of MATLAB functions as follows:

```
y_scramble   = encode(f_audio, Fs, Fa)
y_descramble = decode(y_scramble, Fs, Fa)
```

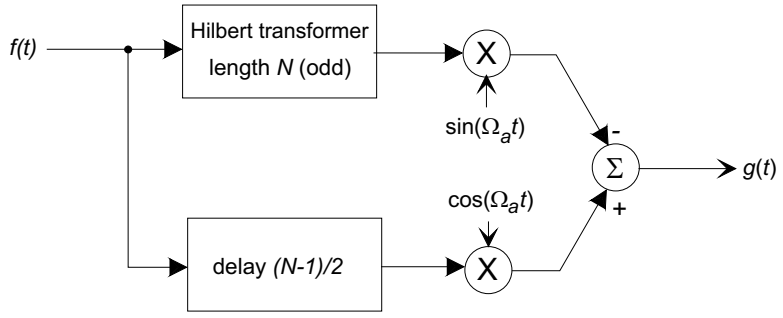
where `f_audio` is a MATLAB array containing the audio file to be processed, `Fs` is the sampling frequency (Hz) of the data in `f_audio`, and `Fa` is the frequency (Hz) around which the spectrum is reflected. The first function should take encode the file, the second restore the scrambled version to the original.

- We will supply you with two audio files (.wav) that you can download and use. One is an ordinary audio file (`PS8Raw.wav`) that is to be to be scrambled and descrambled. The second is one that we have already scrambled (`PS8Scrambled.wav`). These may be imported into your MATLAB workspace using the `wavread()` function

```
[f_audio,Fs,Nbits]   = wavread('myfile')
```

see the MATLAB help.

- You should use the Parks-McClellan design function `firpm()` to design a Hilbert transformer that covers the frequency span 300 – 3000 Hz. (The help on `firpm()` has an example) Be aware that the quality of you scrambling will be affected by the pass-band ripple that you allow.
- Don't forget that the FIR Hilbert transformer is a causal linear-phase system, and that it MUST have an odd length impulse response. You will need to add an equivalent delay filter in the other arm. Design an appropriate FIR impulse response.



- The audio arrays will be large. Use `fftfilt()` to do the actual filtering operations.
- Let's standardize on a reflection frequency $F_a = 3500$ Hz.
- To test your scrambler, load the audio file, then compute and plot its magnitude spectrum. Then scramble the file and plot its magnitude spectrum. Make sure it is what you expect. If there are any spurious components, try to find why they are there, and fix them.

You should submit the following:

(a) Listings of your two functions.

(b) Details of your Hilbert transformer design, including plots of the pass-band ripple, and details of the phase response.

(c) A summary of your approach, including the compromises you made in any design choices.

(d) Plots of the magnitude spectra of

1. The audio file in `PS8Raw.wav`.
2. Your scrambled version of `PS8Raw.wav`.

3. The result of descrambling your scrambled version of `PS8Raw.wav`.

Use `fftshift()` on the spectra to make them more readable.

(e) You should upload three `.wav` files to the 2.161 MIT Server site in the homework section:

1. Your scrambled version of `PS8Scrambled.wav`.
2. Your descrambled version of `PS8RawScrambled.wav`.
3. The result of scrambling then descrambling `PS8Raw.wav`.

Be sure to name the files so that they are clearly identified as yours...

We will be available to help if you run into trouble.

Problem 2: An analog integrator has a transfer function $H(s) = 1/s$. Use the bilinear transform to find (a) the discrete time transfer function $H(z)$, and (b) the difference equation for this form of digital integrator.

Plot the frequency response functions of the resulting digital integrator.

Problem 3: A continuous band-pass filter is described by the transfer function

$$H(s) = \frac{s}{s^2 + 2s + 1}.$$

Derive a recursive computing formula (difference equation) for (1) a step-invariant, (2) a root matching, and (c) a bilinear transform digital filter based on this system. Assume $\Delta T = 0.1$s.

Problem 4: A digital filter is to be designed to the following specifications:

Passband	0-10 kHz
Minimum power gain at Ω_c	0.5
Start of stop-band	11.75 kHz
Maximum power gain at Ω_r	0.1

(a) What is the order of the continuous filter if a Chebyshev design is used.

(b) Prewarp the critical frequencies to find the equivalent frequencies for use in the bilinear transform with a sampling frequency of 50,000 samples/sec.

(c) Use MATLAB to design a continuous Chebyshev Type 1 filter with the prewarped crical frequencies. Then use the `bilinear()` function to compute the discrete time transfer function from the continuous prototype.

(d) Make frequency response and pole-zero plots for your resulting filter.

(e) Use MATLAB to convert the prototype design to a band-pass digital filter with a passband of 5-15 kHz.