ECE 537 **HW #6** **Fall 2018**

Univ. of Illinois **Due 2 weeks** **Prof. Allen**

Topics of this homework: Spherical waves; Short time Fourier Transform; Speech coding;

1. The general solution to the spherical wave equation is

$$p(r,t) = f(ct-r)/r + g(ct+r)/r.$$

where p is the pressure, r is the radius, c is the speed of sound, and t is time. For the case of harmonic outward traveling waves, the first function may be written

$$f(ct-r)/r = \frac{p_0}{r}\delta\left(t-(r-r_0)/c\right), \text{for } r \geq r_0 \text{ else } 0,$$

which has a Fourier transform (check this!)

$$F(\omega,r) = \frac{p_0}{r}e^{-j(r-r_0)\omega/c},$$

where p_0 is a constant that describes the magnitude of the pressure at $r=r_0$ and $t=0$.

A delta function has the units of its reciprocal argument, thus $\delta(r-r_0)$ has units of $1/r$ since when you integrate it against its argument, you have a dimensionless 1.

Write down a similar formula for the retrograde wave in both the time and frequency domains. For initial conditions assume that the pressure is 1 Pascal [Pa] at 1 meter [m]. For the speed of sound, assume $c = 345$ [m/s].

   (a) Describe and plot the pressure of the retrograde wave as a function of time.

   (b) What is the total energy of the wave over the same time period?

   (c) What is the intensity?

   (d) Consider an inward-bound wave that starts from a 1 [m] radius. What is the solution in this case?

   (e) When the wave reaches 1 mm, give the time, and intensity of the wave.

   (f) When the time reaches 2.8986 [ms], explain what happens.

   (g) Can you think of any applications of this retrograde (receding wave) solution to the wave equation?

2. A 1 cm radius point source having a pressure of 1 [Pa] is placed 1 meter from a rigid wall.

   (a) Write down the solution for this physical condition (i.e., Assume outward traveling wave, in the frequency domain, plus a single image, reflected off a rigid wall, defined by a boundary condition that the normal partial velocity of the wave at the surface of the hard wall is zero.).

   (b) Compute the velocity $U(x,y,z,\omega)$ of the wave field, and evaluate it at the wall. (I suggest that you express your answer in rectangular coordinates.)

   (c) Give an expression for the normal component (normal to the wall) of the velocity $U_x(x=0,\omega)$ of the field. (I am looking for the proof of the fact that the normal velocity is zero, given a single image of the source, of the single wall.)

3. The time-bandwidth product γ is a property of a window, defined as the product of the duration of a window T [s] (non-zero samples), time the bandwidth B [Hz] (twice the bandwidth from 0 Hz to the frequency of the first zero, namely find

$$\gamma = T \times B.$$

Determine B manually (computationally). Let $F_{\text{max}} = 10^4$ [Hz] (i.e., the sample period is $T = 1/2F_{\text{max}}$).

Find the stop-band attenuation δ (in dB) and the time-bandwidth product γ for an L point long

(a) Hamming window, $L = 2^n$ points, for $n = 3, 8, 12$.

(b) $L = 128$ point Kaiser window, with *beta* $= 6$, 8 and 10.5.

(c)

(a) Plot the log-magnitude spectrum (FFT resp. of the window $W(f)$), for $f > 0$. One plot for the Hamming and one for the Kaiser, with the spectra superimposed. Label your plots.

(b) Plot $\gamma(T)$ for each of these windows.

From Matlab, try `help Kaiser` and `help hamming`. To compute the FFT, use the command `W=fft(w)`. The positive frequencies run from $W(1 : NF)$ where $NF = length(w)/2 + 1$.

4. Define an over lap add (OLA) error function:

$$e(t) \equiv \frac{R}{W(0)} \sum_{n=0}^{N-1} w(t - nR) - 1,$$

where $w(t)$ is a window function (i.e., Hamming, hanning or Kaiser window), $W(0) = \sum_n w[n]$ is the sum over $w(t)$, n is an integer running from 0 to $N - 1$ and R will be the *decimation parameter*, which is an integer between 1 and the length of the window L. For this problem let $L = 128$ and $N = 1024$, and take w to be a Kaiser window having $\beta = 10.5$. Let $t = mT$ with $T = 1/(2 \times 10^4)$.

(a) Plot $e(t)$ for $R = L/4$.

(b) Plot $e(t)$ in the range of $L \le t \le N - 1 - L$.

(c) Plot $e(t)$ with $R = L/2$.

(d) Why is $e(t)$ periodic? What is the period? Why?

(e) Explain what is going on. What is happening at the ends of $e(t)$ in the first plot?

5. The idea of this problem is to make a very simple *lossy* speech codec. A codec encodes and then decodes speech, reducing the number of bits for the representation. The idea is to reduce the bit rate as much as possible, without reducing the perceptual quality of the speech. For example, If you have access to the "sox" program (Cygwin under windows or unix/linux), you can convert your voice to gsm with the command: `sox file.wav file.gsm`. I have placed this file on the website with the other speech samples.

You can then listen to the gsm file by converting the gsm file back to a new wave-file with the command: `sox -r 16000 -c 1 file.gsm file-gsm.wav`. The "sox" command `play WhenAllElse8k.gsm` will (should) play the sound.

TO-DO:

(a) First read the wave file into memory using matlab's wavread command
`[s,Fs]=wavread('WhenAllElse8k.wav');`
Verify that $F_s = 8000$ [Hz] and that the speech is read in properly. Plot it, and write it back out in another file, to verify this, for example.

(b) Next strip off and save the sign from the speech waveform, using the Matlab `b16=sign(s)` command, which saves the sign in array `b16`. This counts as one bit/sample.

(c) Next take the magnitude of the speech and take the square root. By taking the square root, the sign-reduced speech is reduced from a 15 bit resolution to 8 bit resolution. This is because the original data was stored as sign and magnitude 16 bit numbers, and taking the square root reduces the dynamic range by 1/2.

(d) We next need to fix the numbers to be integers. This is the *quantizing step.* Scale the speech up by 7 bits (scale by 128) Use the "fix()" command. Then scale back down by 128. Verify that the resulting numbers are less than 1 after you have finished doing this.

(e) Now square the result, and put the sign back on. You should now have a linear speech signal again, but the compressed version should have 1/2 the dynamic range of the signal you started with, plus 1 bit (the sign bit).
We started with 16 bit/sample speech, but you should now have 9 bits/sample.

(f) Make a histogram of `log2(abs(speech))`

(g) Scale the speech so that the maximum value is 32767. Then take the cube root. What is the dynamic range now? How does the reconstructed speech sound?

(h) Repeat the process using more compressive roots. How far can you go with this type of processing (how low can the bit rate be compressed until the speech quality is poor).
Here is the code that I used to do this:

```
%C=2
C=3; %96 dB speech compressed to a 32 dB range

M=15/C; %Number of levels
disp(sprintf('%g bits per speech sample',M+1));

%Encode
b16=sign(s);
S=abs(s);
SM=S.^(1/C); %compress
SM=0.5+fix(SM*M); %scale up, quantize and save for later analysis
sM=SM/M; %scale back down to be less than 1

%Now decode
sM=b16.*(sM.^C);
```

See if you can improve on what I tried.

(i) Make a spectrogram and plot a small piece of the waveform, to show the quantization levels. Tell me what you think it sounds like.

Give brief a discussion of what you found.

The following is the code that I used to quantize the samples:

The speech files for various values of C are on the website. See if you can make higher quality speech samples, at low bit rates.

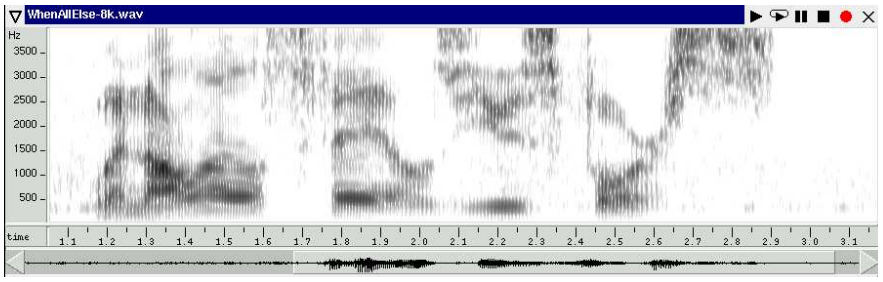

Figure 1: *Spectrograms made with* wavesurfer *of the starting sample of speech (*`WhenAllElse.wav`*) and of the 6 bit quantized speech (*`6bit.wav`*).*

6. Write a program in Matlab that performs overlap-add processing on the speech samples provided

   (http://auditorymodels.org/537/Speech/kp1-g722_64.wav) or

   (http://auditorymodels.org/537/Speech/WhenAllElse8k.wav).

   You will need to use a window function such as the Hamming window. Use a window that has a duration of about 10 ms, and overlap the window by 4:1 (i.e., for every output sample, there will be 4 input samples that contribute).

   (a) Also try 8:1, 2:1 and 1:1 overlap.

   (b) Alternate the sign of the odd frames (each windowed piece is a frame), and repeat the process.

   (c) FFT each frame of speech, conjugate it with Matlab's `conj()` function, then overlap-add (OLA) the pieces. What does `conj()` do to the windowed pieces? How does the resulting speech sound for each of the 4 overlaps?

   For each case listen to the results and report what you hear. Explain what is going on.