

Utah State University  
ECE 3640  
MATLAB Experiments in Sampling

Computers equipped with sound cards can be used to explore some aspects of sampling and aliasing. In this lab, you will generate signals and explore their effect as they are sampled and played back at various frequencies.

**The basic signal** Enter the following code in MATLAB, and verify that it works as expected:

```
Fs = 8000; % set the sampling rate
T = 1/Fs; % sample interval
tfinal = 4; % length of time
k = 0:tfinal/T; % index vector
f = 440; % signal is at 440 Hz (concert 'A')

sig = 0.2*cos(2*pi*f*k*T); % generate samples of the sinusoidal signal
sound(sig,Fs); % Play the signal (at 8000 samples/sec)
```

From the comments, it should be very clear what is going on. We set up the system parameters, generate samples of the signal, and play it. By default, the `sound` command plays back at 8000 samples/sec.

Try playing around with various amplitudes of the cosine (instead of 0.2); choose an amplitude that is comfortable for you; you will be listening to this signal a lot.

Plot the absolute value of the FFT of `sig` (e.g., something like `plot(abs(fft(sig)))`). Explain what bin values have peaks, and **why**.

**Decimation 1:** The signal can be “downsampled,” taking every other sample, by writing

```
sig2 = sig(1:2:end);
```

(This means index the signal starting from index 1 to the last, but taking only every second sample). Play the signal:

```
sound(sig2,Fs); % play at the original rate
```

For comparison, also play this signal at half the former rate:

```
sound(sig2,Fs/2); % play at reduced rate.
```

What does this sound like? How does the frequency of this signal compare with the first signal? How does `sound(sig,Fs/2)` compare with the original? Answer **Why?** to all these questions! Also, plot the FFT, as before. Explain what bins have peaks, and **why**.

**Other decimations:** Try other decimations, both plotting FFTs and listening to the signals. In particular, downsample by 2,3,5,6,7,8,9,10,11,12, 14, 15. What happens between decimating by 9 and 10? **Why?**

You may want to automate this somewhat by writing the following little MATLAB program to a file and running it:

```
decimatelist = [2 3 5 6 7 8 9 10 11 12 14 15];
for d = decimatelist
    fprintf(1,'Decimating by %d. Press Return to begin\n',d);
    j = input(''); % wait for return key
    sigdec = sig(1:d:end); % decimate the signal
    plot(abs(fft(sigdec)));
    sound(sigdec,Fs); % decimate and play the signal
end % end for loop
```

Describe exactly what frequency is produced by each of these decimation factors. Describe what is happening when the signal begins to decrease in frequency. What do we call this? Account for the behavior of the FFT plots. (Save at least a couple of these to turn in.) Do the peaks change as expected? Do you observe any spectral leakage in these plots?

**Change of playback rate** Now try changing the playback rate. In the PC lab, you can say

```
sound(sig, Fs/2)
```

and it will play the signal back at 4000 samples/second (instead of the default of 8000 samples/sec). What does this sound like? **Why?** Try playback rates of  $F_s$ ,  $1.1 \cdot F_s$ ,  $0.9 \cdot F_s$ ,  $2 \cdot F_s$ ,  $F_s/2$ ,  $F_s/3$ ,  $F_s/4$ . (You may want to write a little script like the one from the previous exercise.) Describe how the produced sound changes with these sampling rates, and **why**.

**Chirp signals** Now we will try a “chirp” signal, a signal that changes its instantaneous frequency with time.

We want a signal that changes its frequency in time, so that at time  $t = 0$  the frequency is  $f_1$  Hz and at time  $t = t_f$  the frequency is  $f_2$  Hz, with linear variation in frequency as a function of time. Such a signal is called a linear chirp.

To set up the data for such a signal, we need to take a look at the relationship between frequency and phase. Consider a signal

$$s(t) = \cos(2\pi f_0 t)$$

The argument to a cosine function is always the dimensionless *phase*. In this case, the of this signal is  $\theta(t) = 2\pi f_0 t$ . Notice that the frequency of the signal can be recovered by

$$\frac{1}{2\pi} \frac{d\theta(t)}{dt} = f_0.$$

In this case, the frequency is constant.

More generally, we might have a phase function which does not vary linearly with time, which leads to a time-varying frequency. In general, for a phase function  $\theta(t)$  we define

$$\frac{1}{2\pi} \frac{d\theta(t)}{dt} = f(t) \tag{1}$$

as the *instantaneous frequency*.

Now let us define how we want our instantaneous frequency. Let  $f(t)$  denote the frequency as a function of time. We want  $f(0) = f_1$  and  $f(t_f) = f_2$ , with linear variation inbetween. We can thus write

$$f(t) = f_1 + \frac{f_2 - f_1}{t_f} t$$

or

$$f(t) = f_1 + mt$$

where  $m = (f_2 - f_1)/t_f$ . Now let us use this in the context of the instantaneous frequency defined in (1). We write

$$\frac{1}{2\pi} \frac{d\theta(t)}{dt} = f_1 + mt$$

or

$$\theta(t) = 2\pi(f_1 t + \frac{1}{2} m t^2)$$

This is the argument to the cosine function what would generate this signal. That is, we would use

$$s(t) = \cos(\theta(t)) = \cos(2\pi t(f_1 + \frac{1}{2} m t)).$$

Notice that the quantity which multiplies the time  $t$  is

$$f_1 + \frac{mt}{2}.$$

In the code below, we call this the frequency, although it is not strictly the instantaneous frequency.

Now we are ready to enter this in MATLAB.

```
Fs = 8000;
T = 1/Fs;
tfinal = 4; % final time

t = 0:T:tfinal; % time function

f1 = 440; % initial frequency in chirp
f2 = 1000; % final frequency in chirp
m = (f2-f1)/tfinal; % chirp slope

f = f1 + m*t/2; % create the "frequency" function

sigch = cos(2*pi * f .* t);
sound(sigch,Fs);
```

The `.*` means “element by element multiply.” Explain what is going on, and why this works.

Now, change the starting and ending frequencies so  $f_1 = 2000$  and  $f_2 = 6000$ . Generate a plot of the frequency and play the signal as before. What is happening? **Explain why.** What is the perceived final frequency? Why does the frequency go up, then go down?

**Aliasing on music.** Now let’s try the effect of aliasing on real music. There is a file on the system known as `handel`, which has a piece of the Hallelujah Chorus. You can load it (into the variable called `y`) and play it by

```
load handel
sound(y,Fs)
```

Try this. To get the effect that aliasing might have, try the following commands:

```
sound(y(1:2:end));
sound(y(1:2:end),Fs/2);
sound(y(1:3:end),Fs/3);
sound(y(1:4:end),Fs/4);
sound(y(1:5:end),Fs/5);
```

Describe the effect that these commands have on the sound, and **why** these effects are happening. (For example, explain why you get the chipmunk choir on the first one.) Why is both decimation (such as `y(1:4:end)`) and sample rate changing (such as `Fs/4`) necessary to keep things pitched correctly.

**Turn in: Your plots, your answers to questions, your observations, and results of any other experiments you may have tried.**

Note: Clear writing and careful reasoning is imperative in this assignment. The technical difficulty is minor, but you must write carefully about what is happening, and why.