

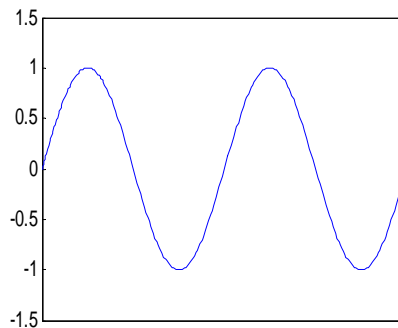
Supplement to Chapter 4 of *The Science of Digital Media* – Digital Audio Representation

Worksheet – Digital Audio > Audio Dithering¹

Modeling Environment: MATLAB and Adobe Audition, Sony Sound Forge, or some other audio processing program

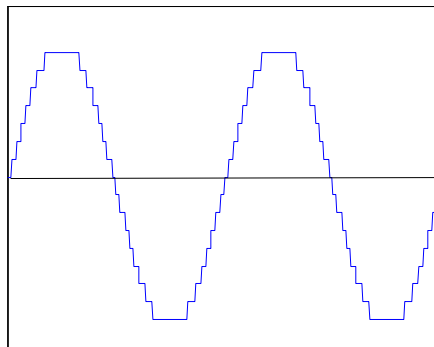
Exercise 1

First, create a sine wave corresponding to the note A, which has a frequency of 440 Hz. Call this sine wave function $f1$. Look at the graph for the sine wave. Scale your graph so that you can see just two cycles of the wave. It should look something like this:



Exercise 2

Adjust a $\sin(x)$ function so that you get 16 quantization levels – 8 non-negative and 8 negative ones. Call your new function $f2$. Create a graph of this function. It should look something like this:



Question 1

What is the bit depth of the figure in Exercise 2?

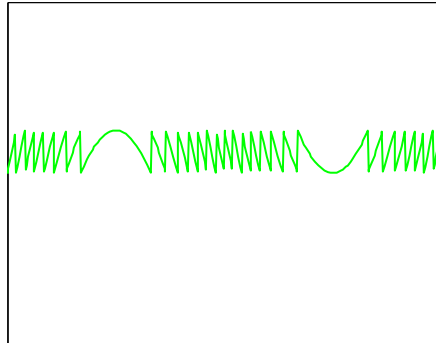
¹This material is based on work supported by the National Science Foundation under Grant No. DUE-0340969. This worksheet was written by Dr. Jennifer Burg (burg@wfu.edu).

Exercise 3

Put the two graphs on the same figure.

Exercise 4

Write a function f_3 that subtracts f_2 from f_1 . Graph this function. It should look something like the wave below. This is your quantization error wave.



Exercise 5

Try to find an environment in which you can create audio files corresponding to f_1 , f_2 , and f_3 and play them. (For example, in an audio processing program, you can generate a pure tone – the note A in this case – and play it. In MATLAB, you can generate vectors of sample values and play them as audio files with *wavplay* in the Windows environment.)

Exercise 6

Find a short mono 16-bit audio clip that has a fairly wide dynamic range, including some low amplitude parts. (If you find an appropriate stereo clip, just resave it as mono.) Call this *clip1*.

Play *clip1*.

Requantize *clip1* at 4 bits per sample without dithering. That's a very low bit depth, and not one that you'd be likely to use, but it will exaggerate the effect we want to hear in this example. Call the requantized audio file *clip2*.

Play *clip2*. Listen to the difference between *clip1* and *clip2*.

Compare the waveforms of *clip1* and *clip2*.

Subtract *clip2* from *clip1* and look at the resulting waveform. Call this waveform *clip3*.

Listen to *clip3*.

Question 2

What does *clip3* represent?

Exercise 7

Go back to *clip1*. Requantize it again, but with dithering. (Don't use noise shaping with the dithering. We'll look at noise shaping in another worksheet.) Call this *clip4*. Play *clip4*. Compare how it sounds to *clip1* and *clip2*. Look at the waveform for *clip4*. Compare it to the waveform for *clip2*.

Question 3

What do you observe about the graphical difference between the waveform for *clip4* and the waveform for *clip2*? Why is there this difference?

Exercise 8

Subtract *clip4* from *clip2*. Call this *clip5*. Listen to *clip5*.

Question 4

What does *clip5* represent?

Exercise 9

We're assuming that in Exercise 6, Exercise 7, and Exercise 8, you were requantizing your audio files in an audio processing program like an audio processing program, both with and without dithering. Now try reduce *clip1* to 4 bits per sample the way you reduced the bit depth in Exercise 1, but this time "by hand" – with a program that you write, or in a modeling environment such as MATLAB,. Again, call this requantized audio clip *clip2*. Dither *clip2* by adding a random value to each sample. Listen to your dithered audio clip. As you do this, keep in mind that the random value you generate between -1 and 1 must be scaled to the bit depth of 4.

Question 5

Does your own version of dithering sound as good as the dithering done in the audio processing program? If not, why do you think it doesn't sound as good? (If it sounds just as good, explain what you did to get such a good effect.)

Question 6

Give examples of at least two probability density functions that are used in audio processing programs.