

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

[MATLAB/Digital Audio Worksheet – Digital Audio > Noise Shaping](#)¹

(You can also try this worksheet using Octave under Linux.)

Introduction:

Noise shaping is a method for lessening the effect of quantization error by spreading the quantization noise over a wider spectrum, moving more of it to higher frequencies where the noise either will not be heard or can be filtered out without loss to the original audio signal. It is implemented by means of a feedback loop that adds the quantization error from one sample to the sample that follows. First-order noise shaping operates according to the following feedback equation.

Let F_in be an array of N digital audio samples that are to be quantized, dithered, and noise shaped, yielding F_out .
For $0 \leq i \leq N - 1$, define the following:
 F_in_i is the i^{th} sample value, not yet quantized.
 D_i is a random dithering value added to the i^{th} sample.
The assignment statement $F_in_i = F_in_i + D_i + cE_{i-1}$
dithers and noise shapes the sample. Subsequently,
 $F_out_i = \lfloor F_in_i \rfloor$ quantizes the sample.
 E_i is the error resulting from quantizing the i^{th} sample
after dithering and noise shaping.
For $i = -1$, $E_i = 0$. Otherwise, $E_i = F_in_i - F_out_i$

Noise shaping is used in conjunction with dithering because noise shaping alone does not decorrelate the noise from the original signal. Dither helps to randomize the noise, and noise shaping moves some of the noise to higher frequencies.

In this worksheet, you'll first try the dithering and noise shaping options in your audio processing program, listen to the results, and examine the frequency components of the waveform. You'll also try your hand at implementing the basic feedback loop for noise shaping and listen to your results.

Exercise 1

Open an audio clip in your audio processing program. Choose one that has a sampling rate of 44.1 kHz and bit depth of 16 bits. You need only a mono file, so if it's stereo you can collapse it down to mono. Listen to the audio file.

¹This material is based on work supported by the National Science Foundation under Grant No. DUE-0340969. This worksheet was written by Jennifer Burg and Michael Boger.

First, you're going to listen to the effects of reducing the bit depth. Reduce the bit depth of your sound clip from 16 to 4 bits, but deselect the "dither" option. Listen to the clip.

Question 1

What does the clip sound like now that you've reduced the bit depth from 16 bits to 4 bits? What does the waveform look like?

Exercise 2

Undo the operation, restoring the clip to 16 bits. Now, reduce the clip again to 4 bits. This time select the "dither" option, but don't select the "noise shaping" option. You can use the triangular pdf (probability density function).

Question 2

What does the clip sound like now that you've reduced the bit depth from 16 bits to 4 bits with dithering? What does the waveform look like?

Exercise 3

Undo the operation, restoring the clip to 16 bits. Now, reduce the clip again to 4 bits, but this time select both the "dither" and the "noise shaping" option. Use a noise shaping algorithm that works for a sampling rate of 44.1 kHz, or choose one at random. (You can experiment with different ones and see how they sound.)

Question 3

What does the clip sound like now that you've reduced the bit depth from 16 bits to 4 bits using both dithering and noise shaping? What does the waveform look like?

Exercise 4

Now you're going to move your sound clips to MATLAB. So that you don't have to deal with such large files, cut out a small segment from the audio file if the file is long. (Between one and five seconds should be enough.) Choose something that has some dynamic range within it. Save the short clip as a *.wav* file. Call it *w.wav*.

Now reduce the bit depth of the short clip to 4 bits per sample. Keep the sampling rate at 44.1 kHz. When you reduce the bit depth, don't select the "dither" option. Save this clip as *w_4bits.wav*.

Go back to *w.wav*. Reduce it to 4 bits again. This time choose the "dither" option and the triangular pdf (probability density function), but *no* noise shaping. Save this version as *w_dith.wav*.

Go back to *w.wav*. Reduce it to 4 bits again. Choose the "dither" option again, but this time use noise shaping. Save this version as *w_shaped.wav*.

Question 4

Listen to the four clips and verify that you can hear the difference you expect.

Exercise 5

Open MATLAB and import your four clips into MATLAB. You can do this with statements like

```
w = wavread('w.wav');
```

Get the error wave associated with each of the reduced-bit-depth audio files. You can do this with statements like

```
quantError = w-w_4bits;
```

Look at and compare the graphs of the three error waves. You can look at them with statements like

```
plot(quantError);
```

Then change the scale of the x and y axes to zoom in on the waves to see detail. When you compare them, make sure that the scale is the same on all three.

Save all three files with names corresponding to the array names in MATLAB. You can do this with statements like

```
wavwrite(quantError, 44100, 'quantError.wav');
```

Question 5

What statements did you use to read in the files, create the error files, plot them, and save them? How would you compare the graphs of the three error waves? Which wave appears to have the highest frequency?

If you're completing this worksheet electronically, your instructor may want you to save the three waveform graphs as *.emf* files and insert them into your answer sheet.

Exercise 6

Go back to your audio processing program and open the three error wave files you created in MATLAB. Look at them in the spectral view of your audio processing program.

Question 6

How would you compare the three error waves in the spectral view? Which appears to have the most energy in the high frequency components?

If you're completing this worksheet electronically, your instructor may want you to do screen captures of the three spectral views and insert them into your answer sheet.

Exercise 7

Go back to MATLAB. Now you're going to try your hand at implementing the first-order feedback loop for noise shaping. Create a 440 Hz sound wave in MATLAB by

generating a “second’s worth” of samples with a sampling rate of 44.1 kHz. Call the array of samples *wA*. You can do this with statements like the following:

```
x = linspace(0, 44100, 44100);  
wA = sin(0.0627*x);
```

Listen to the wave. It should be the note A. You can do this with

```
wavplay(wA, 44100);
```

Look at the values in the array of samples. Notice that they are normalized to values between -1 and 1 .

Question 7

How would you scale these value so that they are between -32768 and 32767 (i.e., 16-bit values)? Do it.

Question 8

Once the values are scaled to 16-bits, how would you requantize them to 4 bits? Do it.

Question 9

How would you compute the error generated from requantizing? Note that we want the error to be on a 16-bit scale. Do the requantizing, putting the error values in an array called *error*.

Exercise 8

Write a MATLAB program that implements the first-order feedback loop for noise shaping.

Hints:

You can use a triangular probability density function for the dithering. An easy way to implement this is to generate two random numbers between -0.5 and 0.5 and sum them. Be sure the dithering value is on the scale of the 4-bit-depth file.

Use vector operations where possible to speed up execution (rather than running the array of sample through a *for* loop and operating on them element-by-element.)

Use your noise shaping algorithm on an audio file that has some dynamic range but isn’t too long – three to five seconds should be long enough for you to hear the differences resulting from requantization with or without noise shaping. Listen to the result, and look at the resulting error waveform in MATLAB.

Question 10

How did you generate the error waveform? What does it look like? How does it compare with the error wave generated when the audio file is requantized without dither or noise shaping?

Question 11

How did you implement your dithering? What values did you use? Explain.

Attach your MATLAB program to your answer sheet.