Machine Problem 1

Instructions: Implement Waveform Compression algorithms that we discussed in our CE-DISP2 class – Lecture 3. Use the audio file that is also provided in this MP. As proof of MP1 completion, please submit the ff. before the deadline:

1. A single m-file source code containing the MP1 script. For this script to work in Matlab, all the function definitions should be placed at the end of the m-file, and the function calls should be placed at the beginning of the source code. Please add inline comments to make your code more readable.
2. A PDF document containing explanations of your program design and implementation, answers to the questions and corresponding plots.

Your Matlab program should be able to do the ff.:

Initialization:

1) Reset the Matlab environment, by clearing all variables and closing all open Matlab windows.
2) Read a sound file and load it into the Matlab environment. It should be able to prompt the user to type the name of the file for the input signal. Hint: use audioread() and input() functions. Use the sound file for downloadable from this link, and convert the 2-channel audio into a single-channel one, making it more economical to process.
   a) What is the sampling rate of the input signal?
   b) Try to play the signal by using the sound() function. What happens if you try to play the sound using a lower or higher sampling rate? Why is this so?

Part 1: Implementation of a mu-law compander for the quantized signal. Set mu = 255.

1) Create a function for an N-bit Midtread Quantizer: The function should be able to accept a signal of whatever value of N from the user, and specifications for an input signal's minimum and maximum values. For testing purposes, use N = 3. It should be able to accept a vector data representing a signal and output a quantized signal.
2) Compute the quantization error and the overall SNR for the quantized signal. Print the SNR value, and show the plot of the original, quantized and error signals similar to Slide 21 of Lecture 3. Note the error is computed as $e(\mathrm{n}) = s_{quantized}(\mathrm{n})$ - $s_{original}(\mathrm{n})$.
3) You may use the plot() and the step() functions for visualization. For reference, you should be able to show the graphs of Figure 3.
   a) What is the SNR (in dB) of the original signal…
      i) Using the first formula in Slide 11 of Lecture 3 slides?
      ii) Using the 2nd formula of Slide 11 of Lecture 3 slides?
      iii) Compare the SNR values that you got in 3.a.i and 3.a.ii. What accounts for the difference, if any?
   b) Using SNR (in dB) formula indicated in 3.a.ii, compute the SNR values for N = 3 and N = 5. Which is higher? Why is this so?

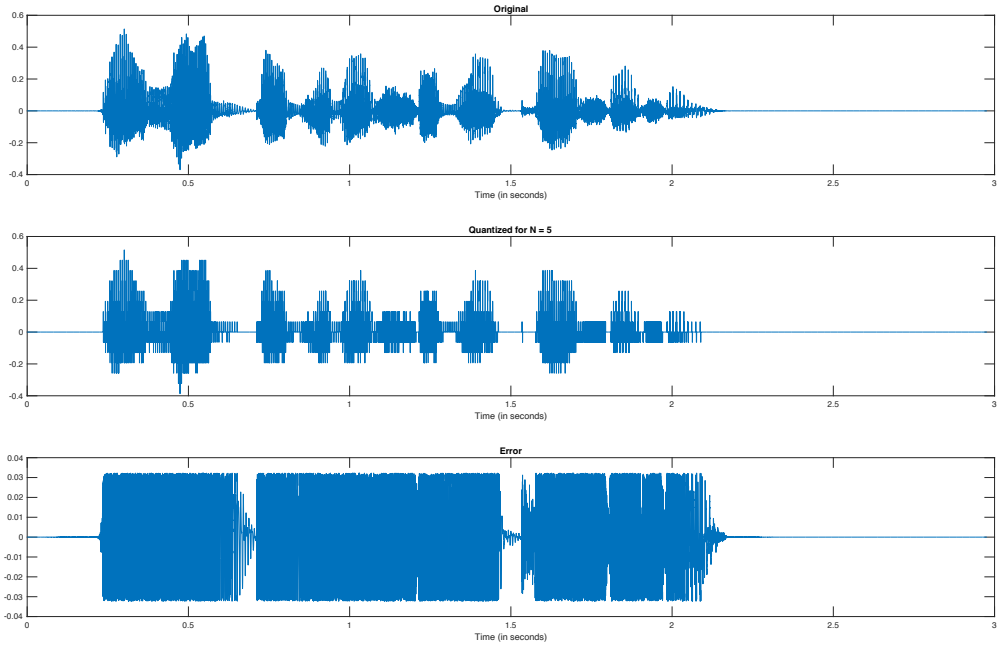

Figure 1. Figures for Part 1 of Machine Problem 1

4) Implement a mu-law compander for the quantized signal. Set mu = 255. Please refer to the block diagram in Figure 2.

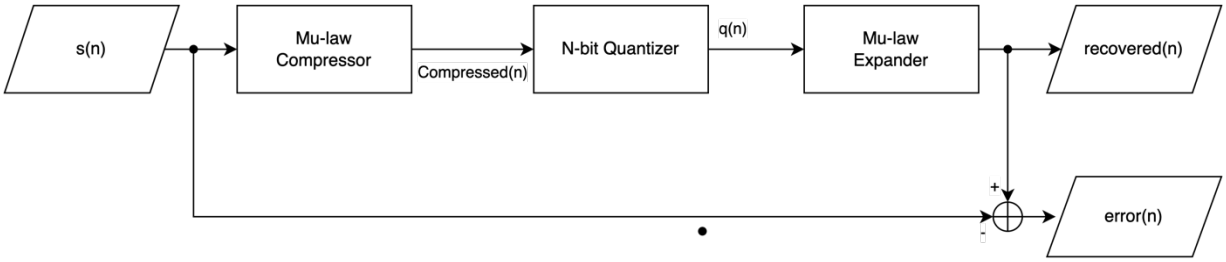

Figure 2. Block Diagram for Part 2 of MP 1

Plot the corresponding signals original signal s(n), the quantized signal q(n), the recovered signal, and the error signal. The plots should be the same as those in Figure 3.

Answer the following questions:

a) What is the SNR (in dB) of the recovered signal for N = 3? What about for N = 5? How do they compare with the SNR values that you obtained for Part 1: 3.b?

b) How do the recovered signals in Part 1 and Part 2 sound like? Which sounds better?

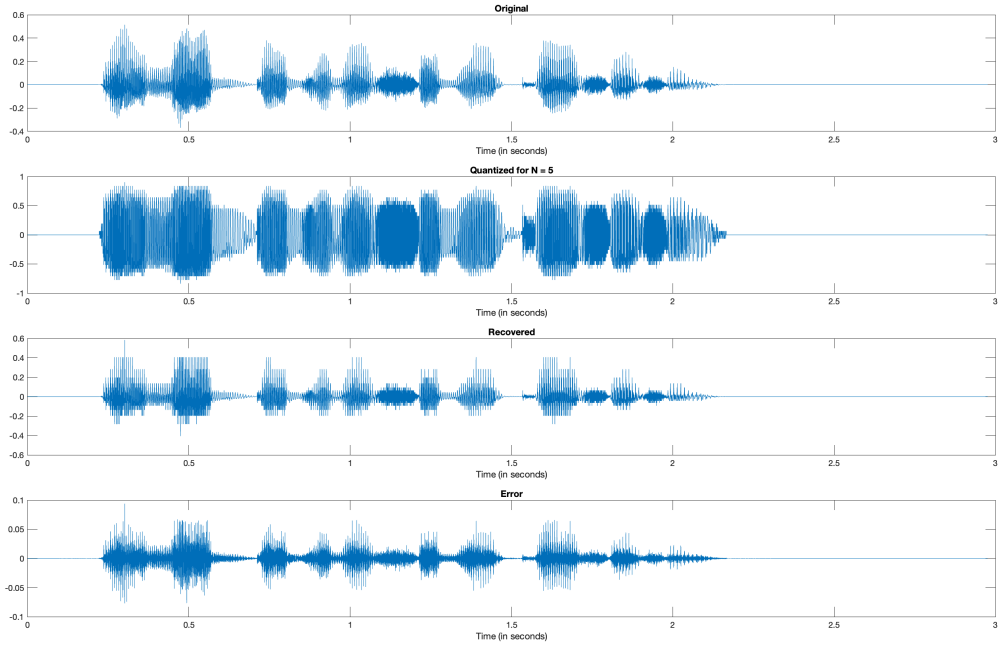

Figure 3. Figures for Part 1 of Machine Problem 1

5) Implement a digital mu-law compander using the same input signal. Refer to Figure 4 for the block diagram.

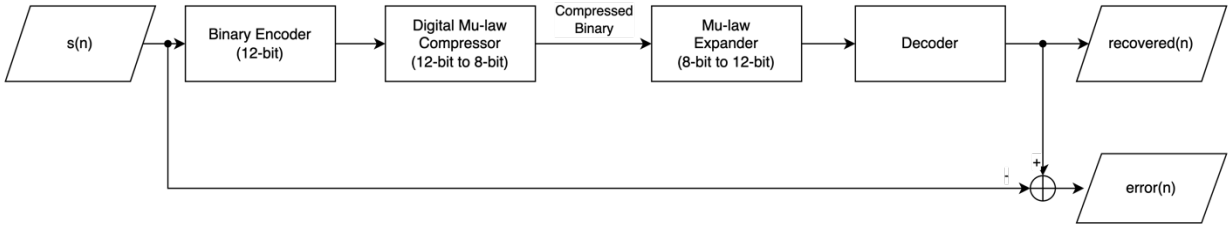

Figure 4. Block diagram of a Digital mu-law compander

a) Implement a binary encoder module that encodes each signal sample into a 12-bit representation (name it as myBinaryEncoder()). In the binary format, use the MSB as a sign bit (1 – positive, 0 – negative) and the rest of the bits for the mantissa. We assume that all samples of s(n) are between -1 and 1, so the mantissa is treated as the fractional part of the number.

i) Refer to the table below for examples. In the table, the first entry $0.75_{10} = 0.11000\ldots_2$

Input	Encoded Binary Pattern[1]
0.75	1110 0000 0000
-0.75	0110 0000 0000
-0.645	0101 0010 1000
0.0125	1000 0001 1001

[1] **The blue bits represent the binary representation of the fractional part. The MSB is the sign bit.**

b) Implement a 12-bit to 8-bit digital mu-law Compressor. Refer to the Lecture 3 slides (slides 22 to 31) for the implementation concepts. Create a function named myMuCompressor() that takes as input a 12-bit format and outputs an 8-bit format.

c) Implement an 8-bit to 12-bit mu-law Expander. Refer to the Lecture 3 slides (slides 22 to 31) for the implementation concepts. Create a function named myMuExpander() that takes as input an 8-bit format, and outputs a 12-bit pattern.

d) Implement an Encoder module (name it as myEncoder()), which converts the 12-bit recovered binary code to corresponding base-10 fractional number.

e) Plot the corresponding signals original signal s(n), the recovered signal, and the error signal. The plots should be similar to those shown in Figure 5.

   i) How much is the signal compressed compared to the original s(n) after encoding it into 12-bit format? How much was it compressed further after encoding to an 8-bit format? What's the effective data compression ratio from s(n) to the compressed binary (i.e. after mu-law compression)?

   ii) What is the SNR of the recovered signal? How does it compare with the SNR of Parts 1 and 2 of this MP?

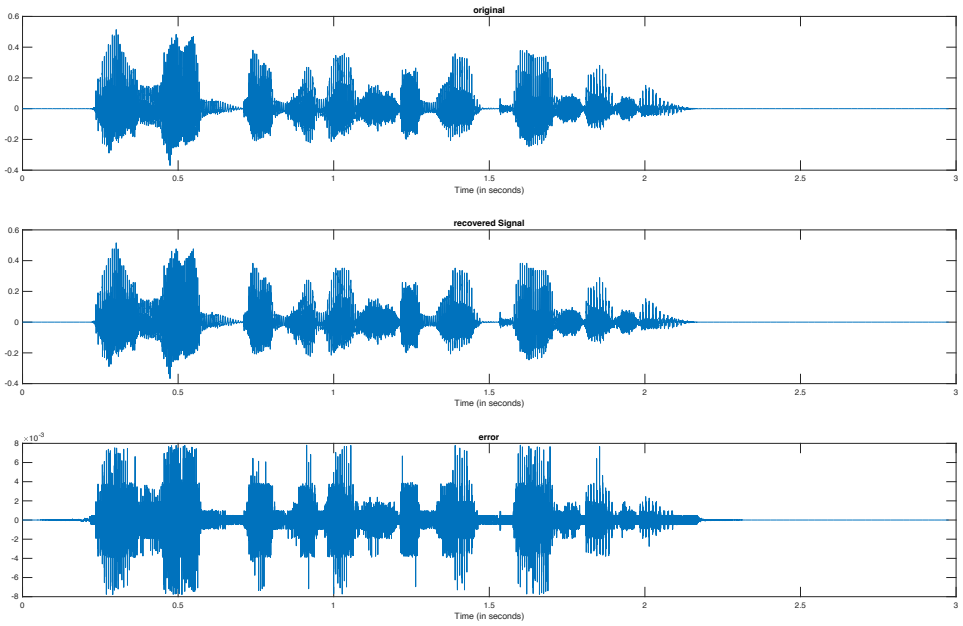

Figure 5. Plots for Part 3 of MP1

==== nothing follows ====