ECSE 436 **McGill University**

Prof. Bajcsy **Fall 2016**

Laboratory Assignment 2 (parts 1 & 2)

Due: with Part 3

Reading: Lecture and lab/tutorial notes, handouts on Altera FPGA signal processing boards, digital signal processing and coding. Related material in the reference textbook by Lin and Costello (available online) and Proakis, Manolakis Digital Signal Processing text.

1. FPGA Signal Processing Boards

Go over the demo/example described in the FPGA handout and start implementing this demo on your signal processing board. In both cases, demonstrate your implementation to the TA's/instructor, then submit your scripts and description of observations with your final write-up.

(a) Implement encoder of the discussed rate ½ convolutional code used in celllar phones.

(b) Implement a simple echo system that plays back, in real-time, a signal that is corrupted by its equal strength echo delayed by 0.01, 0.1, 0.2 and 0.5 seconds, respectively.

2. Viterbi Decoding Algorithm for the Hamming (8,4,4) Code

A Hamming codeword **c =** (c1, c2, c3, c4, c5, c6, c7, c8) has been transmitted over an AWGN channel with noise mean being 0 and a positive variance. The received vector **r = c + noise** is observed to be **r =** (0.54, - 0.12, 1.32, 0.41, 0.63, 1.25, 0.37, -0.02).

(a1) Using Matlab, the code book of the (8,4,4) code and material discussed in the lecture/lab, determine the closest of all 16 codewords to the received **r**. Submit your script and the answer.

(a2) How many floating point operations (additions, multiplications and comparisons) does the approach from part (a) use to decode a codeword?

(b1) Design an optimized trellis diagram for this code that has at most 4 states per stage.

(b2) Redo part (a1) in Matlab by using Viterbi decoding algorithm for the given Hamming code using its 4-state optimized trellis. Determine and implement the appropriate edge weights for individual trellis edges, then use the survivor path approach to determine your answer. Submit your stage-by-stage distance matrices and results of each stage survivor search.

(b3) Redo part **(a2)** for the improved decoder from part (b2)

(c1) Use Matlab to implement and test the Viterbi decoder from part (b2) for general **r**.

(c2) Use your Viterbi decoder from part (c1) to simulate bit error rate performance of the (8,4,4) Hamming code on the additive Gaussian noise channel. Use 'randn' function to generate Gaussian noise samples and set the noise variance so that the SNR ranges between 0 dB and 10 dB, with a step of 1 dB. Submit your BER vs. SNR plot using 'semilogy' scale as well as the script/s used to obtain it. (**HINTS:** Calculate the errors only for the message bits and simulate each SNR value until you obtain 100 decoding errors or until you run 10,000 packets, whichever is less.)

3. Speech Processing and Acoustics: Convolution and the Impulse Response

An overall effect of a room on any given sound signal *x[n]* can be modeled as a linear time-invariant system and can be completely described by the room's impulse response *h[n]*. As mentioned in class, you can measure impulse response of any specific room by producing a very short sound-burst (an impulse such as a gun-shot or a clap) at the source and then measuring the resulting signal at the microphone which picks up not only the direct line-of-sight contribution from the sound source, but also echoes produced by reflections off walls and room objects. File "h.wav" contains the impulse responses h[n] for a specific room and source/microphone setup.

(a) Using Matlab, plot the impulse response for the channel and explain how the delay of the channel (i.e., the time the sound to travels from source to microphone) can be computed by looking at the impulse response plot. What is the distance of the microphone from the source of impulse ?

(b) The file "speech.wav" is a voice recording from an *anechoic* chamber (a studio with little or no echo), with the microphone very close to the voice. Convolve this recording with the impulse response h[n] and listen to the result before and after convolution. Explain what you hear.

(c) Repeat part (b) for an impulse response h'[n] = h[n] + h[n-3000] where the delay is in number samples. Again, explain what you hear.

HINTS: In Matlab, x=audioread('speech.wav') will read audio file named "speech.wav" into the vector "x". Because a computer can only store a discrete sequence of numbers, each file is a vector $v[n]$ sampling the original response $h(t)$ so that $v[n] = h(nT)$ for indices $n = 0,1,2$ etc. In this experiment the sampling rate for all recordings is $T = 1/16000$ seconds. Convolution can be implemented with the "conv" command and you can listen to any vector of data by using the command sound(v,16000)which plays back the vector $v[n]$ at a rate of 16000 samples per second.

(d) Describe how you can play the signal backwards in time and discuss what you hear.

(e) Play the signal at different playback speeds (13000, 14500, 17000, 18500, 20000) and describe how the signal quality/intelligibility changes.

(f) Observe the effect of aliasing - sub-sample the signal (2:1, 3:1, 4:1, 5:1 and 10:1), then play it back appropriately. Describe what you hear and sketch an appropriate signal spectrum plot, illustrating what is happening.

(g) Quantization is another signal processing operation that is performed (e.g., to save on data storage needs) but it can affect significantly quality of the play-back signal. Quantize your signal using appropriate uniform quantizer that has 16, 8, 4, 2 and 1 bits. Play the quantized signals back and discuss the impact of quantization on the quality of the sound.

4. Real-time Audio Processing on FPGA Signal Processing Boards

Redo the following cases from the previous question in real time. In each of the following cases, submit all implemented scripts, describe what you observed and demonstrate your implementation.

(a) Implement a real-time system to play back audio quantized to 8, 4, 3, 2 1 bits per sample.

(b) To demonstrate the effect of aliasing, replay the incoming signal sub-sampled by a factor 2:1, 4:1 and 8:1. (Hint: Use zero-order hold approach OR try to change the output sampling rate.)

(d) Implement a module generating additive Gaussian noise samples using addition of 6 (or 12) independent, uniformly generated samples from interval [-1,1]. (Hint: Use provided uniform generator module.)

(e) Using histogram method in Matlab, verify that the noise generated by the approach in part (d) is in fact *close* to being Gaussian distributed.

(f) What is the exact mean and variance of noise from part **(d)**? What is the reason that the generated noise is near-Gaussian distributed, even though it comes from uniform noise samples? When can the used the Gaussian approximation become a serious problem in practice? Fully justify your individual answers.

(g) Replay a real-time audio signal corrupted by Gaussian noise at SNR = 40 dB, 30 dB, 20 dB, 10 dB, 0 dB and -10 dB. Describe how you adjusted the noise power and what you hear in each case.

5. Signal Processing of Binary Erasures

(a) Construct an example of 4 linear equations about 4 unknowns which contains only 0 or 1 coefficients and right hand side terms, so that the real and "mod 2" binary solutions differ.

(b) Can you modify your example in such a way, that both the real and binary solutions contain only 0 or 1, but are still different? Fully justify your answer.

(c) What can you conclude from your part (b) about properly implementing an erasure decoder for binary error control code in Matlab?

(d) In Matlab, implement a binary Gaussian elimination method that can solve any system of 4 equations about 4 unknowns using "mod 2" arithmetics. Submit your implementation with the final write-up and demonstrate it to the TA's/instructor.