#### Texas Tech University

#### Department of Electrical and Computer Engineering

#### ECE 4364 Digital Signal Processing

#### Fall 2019 Group D1

# Project 2

Due November 15th before midnight CDT.

This is a group project. Each group should submit to Blackboard exactly two files:

1. A single ZIP file with all the MATLAB programs and data files.
2. A single report document in Word or PDF. Expectations for the report are detailed below.

Submit your solutions to Blackboard by the date and time listed to avoid penalties.

## Project Description

We wish to design a speech audio compression system based on sub-band coding. The speech audio will be split into four different frequency bands of equal bandwidth, and each will be compressed by quantization of the samples. Each band can be quantized by a different amount. Part of your task is to find optimal quantization factors for each band as to maximize the compression, while maintaining reasonable audio quality.

Your project will consist of:

1. An encoder
2. A decoder
3. A quality measurement system

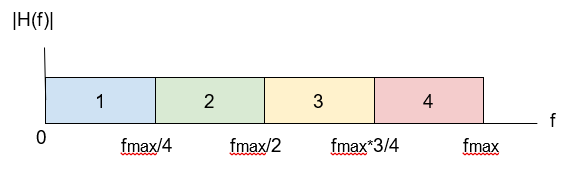
### Test Audio

You will need a test audio file to serve as input to your system. Select an audio file with the following characteristics:

* Sampled at 8000 samples/second
* At least 30 seconds in length
* Containing human speech (not music)

### The Encoder

Your audio encoder will take the 8kHz speech audio as input and split it into four equal-width sub-bands. You may use any method you choose to realize the band splitting. Each of the sub-bands should be sampled at 2000 samples/second.



Then your encoder will quantize each sample in each of the subbands, into a target number of bits. The number of bits per sample can be different for each subband. It is your task to determine a good number of bits per sample to assign to each subband. Please note that using numbers of bits that are a power of 2 will simplify your task of packing them together.

Finally, your encoder will pack the quantized representations of each sample, for all bands, into a single binary file. You may choose any format that you want for the binary file. But the file should not have any padding bits or zeros. It should be a compact file with only the necessary bits to represent the samples.

The ratio between the input file size, and the output file size will be the compression ratio of your encoder.

### The Decoder

Your decoder will take the compressed binary file as input and will reconstruct an audio file at 8000 samples/second. For this you will need to unpack the bits for every sample, and scale them to the original scale. Then the four bands will need to be combined into a single 8kHz signal and saved into an audio file. This output audio file should be sampled at 8000 samples/second and should be intelligible.

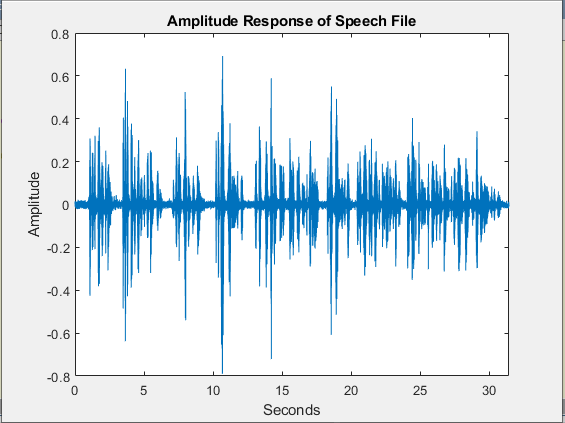
### The Quality Estimator

You are asked to use PESQ as an objective measure of speech quality. You may find the source code for an evaluation version of the PESQ utility at <https://www.itu.int/rec/T-REC-P.862-200511-I!Amd2/en>. PESQ will compare your reconstructed audio with the original one and provide a quality score between 1 and 5. Typical speech codecs used in telephony produce audio with PESQ scores between 3 and 4.

## Procedure

1. Obtain an audio test file (you may record your own)

**For the audio test file, we used a 30 second recording of Fahrenheit 451. The audio was recorded using audacity and is sampled at a sampling frequency of 8 kHz. The Amplitude Response and Frequency Spectrum of this signal are shown below.**

**A close up of a map

Description automatically generated**

1. Implement your encoder and decoder in Matlab
2. Using low compression settings, make sure that you can reconstruct the audio signal with high quality (high PESQ score)
3. Tune your compression settings per-band, for three different configurations:
   1. High-quality. Try to maximize the compression, while maintaining a PESQ score above 4.0.
      1. **With a compression ratio of 1.6 (ratio of 1,1,2,2 respectively on the sub-bands) we were able to maintain a pesq score of around 4.** 
   2. Balanced. Try to maximize the compression while maintaining a PESQ score above 3.0.
      1. **With a compression ratio of 2.7 (ratio of 2,2,4,4 respectively on the sub-bands) we were able to maintain a pesq score of around 3**.
   3. High-compression. Try to maximize the compression while maintaining a PESQ score above 2.0.
      1. **With a compression ratio of 3.8 (ratio of 2,4,4,8 respectively on the sub-bands) we were able to maintain a pesq score of around 2** 
4. Comment about your design choices. How did you decide on them?

**Sub-band Sampling and Filters:**

**For our design, we mainly used the standard FIR() function in MATLAB for sub-band filtering. This filter uses a Hamming Window and was easy to implement for this application. For decimation, we used the for-loop method shown in the subband\_resampling.m file and for upsampling, the MATLAB upsample function was used. We had originally attempted to use the decimate function but found flaws in our result. The results from Homework 6 influenced a lot of the design decisions. An FFT size of 16000 was used to compute the spectrums and is multiplied to match the original spectrum.**

**A screenshot of a cell phone

Description automatically generatedA close up of a map

Description automatically generated**

**Sub-Band 1 Filter Response and corresponding spectrum (0 – 1000 Hz)**

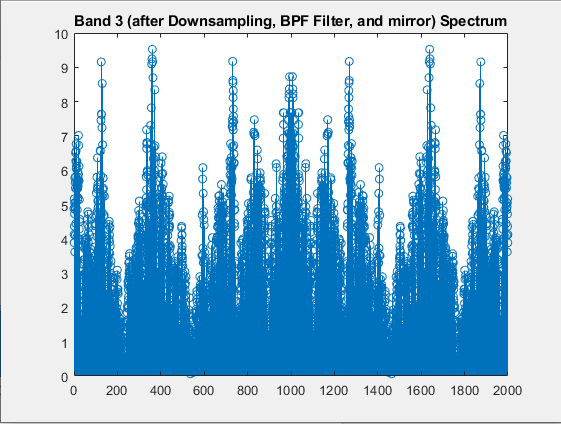
**A close up of a map

Description automatically generatedA screenshot of a cell phone

Description automatically generated**

**Sub-Band 2 Filter Response and corresponding spectrum (1000 – 2000 Hz)**

**A close up of a map

Description automatically generated**

**Sub-Band 3 filter response and corresponding spectrum (2000 – 3000 Hz)**

**A close up of a piece of paper

Description automatically generatedA screenshot of a cell phone

Description automatically generated**

**Sub-Band 4 filter response and corresponding spectrum (3000 – 4000 Hz)**

**Encoder & Decoder:**

**To encode the data, we first changed the data into unsigned fixed-point notation using the number of bits we had available (defined by the compression ratio). This fixed-point value was then written to a binary file along with other data that would have to be known to decode the file. The decoder then (with the help of the reconstruction data at the beginning of the file) reconstructed the four sub-bands and converted them back into floating point notation.**

**Fixed point data was chosen as it is easier to compress into a set number of bits using MATLAB.**

1. Comment on you overall project experience, lessons learned, and ideas for improvement.

**Sub-band Sampling and Filters:**

**One thing we had a lot of issues with was the order of downsampling and upsampling with filters. We had to reference our notes multiple times to ensure we designed the proper Anti-Aliasing and Image-Rejection filters. We continuously tested this with what came from our graphs as well as heavily used the Spectrum Analyzer to compare with the original signal. One thing that could’ve saved us a lot of time was actually taking the time to map out the proper decimated and interpolated order on a piece of paper for each individual sub-band and then model those in MATLAB. We could’ve also used different window designs as well to test the effect before and after compression.**

**Encoder and Decoder:**

**We ended up normalizing the data from -1 to 1 as some signals would be so small that a smaller bit rate the fixed point notation would cause the value to be zero throughout the signal. This did become a problem as a rounding issue caused higher bit rate to become slightly less accurate. We decided that this was worth the tradeoff as it greatly improved the smaller bit rates.**

## Evaluation:

|  |  |
| --- | --- |
| **Item** | **Points** |
| Speech input signal is adequate | 5 |
| Encoder implementation | 20 |
| Decoder implementation | 20 |
| Quantization parameters tuning | 20 |
| Explanation of design choices and project experience | 15 |
| Total | 80 |