LEAST SQUARES AUDIO AND SPEECH COMPRESSION
LINEAR PREDICTIVE CODING (LPC)

Background on LPC Lossy Compression of Speech Signals

Speech, and other audio, signals represented by sample data $Y^N = \{y(n), n = 1, 2, \cdots, N\}$, are often compressed by being quantized to a low bit rate during data transmission in order to obtain faster data transfer rates. Quantization induces distortion in the signal, so this form of compression is said to be "lossy." Unfortunately, *direct* quantization of the speech signal, resulting in the quantized signal $y_q(n)$, may degrade the quality of the speech signal to an unacceptable degree.

One way to reduce the degradation due to quantization is to first fit an *autoregressive* (aka, *all pole*; aka, *infinite impulse response* (IIR)) digital filter model to the original audio signal $y(n)$ of the form

$$y(n) = \underbrace{\sum_{k=1}^{\ell} a(k)y(n-k)}_{\hat{y}(n)} + e(n) \tag{1}$$

The signal $\hat{y}(n)$ is a *linear prediction* of $y(n)$ given the ℓ previous time samples $y(n-1), \cdots, y(n-\ell)$.[1]

Note that we can rewrite (1) as the moving average (aka, *all zero*; aka, *finite impulse response* (FIR)) digital filter model

$$e(n) = y - \hat{y}(n) = y(n) - \sum_{k=1}^{\ell} a(k)y(n-k) \tag{2}$$

where $e(n)$ is known as the (one step ahead) *prediction error* or the *residual error* (or just the *residual*).

Equation (2) shows that given the filter coefficients $a(k)$ *one can easily compute the residuals* $e(n)$ via an FIR moving average digital filtering of the audio signal $y(n)$. On the other hand, equation (1) shows that *one can easily reconstruct (recover) the audio signal* $y(n)$ via an autoreqressive (AR) filtering of the residuals $e(n)$.[2] Thus the audio signal and the residuals contain exactly the same information.

The computation of the residuals $e(n)$ corresponds to an encoding of the sound signal $y(n)$. This procedure is known as *linear predictive coding* (LPC). If the AR linear model (2) is "good enough", the residual (linear prediction) errors should be "small" and we expect that the residuals $e(n)$ will have a much smaller dynamic range than the original signal

[1]The quantity ℓ is called the *model order*. For this project you are to use the value $\ell = 10$.

[2]Assuming that one has values of the initial conditions $y(0), y(-1), \cdots, y(-\ell+1)$. More on this later.

$y(n)$.[3] Note that to know an LPC encoding of $y(n)$ means that one knows *both* the filter coefficients, $a(k)$, *and* the residuals $e(n)$.

More generally, one could have a time-varying model where the filter coefficients, $a(n, k)$ depend upon both n and k. Such models are generally intractable for data-driven learning purposes and one tries to work in domains where the so-called *stationary model* shown in eq. (1) is valid. In order to ensure a reasonably good approximation to the *stationarity assumption,* it will be necessary to break the entire set of audio data Y^N into smaller blocks of data such that within each individual block the stationarity assumption will be assumed to hold. However, within each block a different set of filter coefficients values will be the case. A key goal of this assignment is to learn the filter coefficients, $a(k)$, within each block by determining data-block dependent least squares estimates of the digital filter coefficients. This should produce a good fit of the model (2) to the signal $y(n)$ by learning the filter coefficients which force the sum-of-squares of the residual error $e(n)$ to be as small as possible within each separate block.

As noted above, the residuals $e(n)$ will generally have a much smaller dynamic range than the original audio signal values if the model (2) does provide a good fit to the signal $y(n)$. The speech signal will vary (after normalization) from -1 to 1 while the residuals are mostly smaller than ± 0.1 (relative to the signal magnitude). To minimize the degradation effects of quantization, instead of quantizing the audio signal $y(n)$ one can quantize the residuals $e(n)$, then transmit only the quantized residuals, $e_q(n)$, the coefficients for the model (1), and the quantization rates. Reconstructing the signal from the model and the quantized residuals, $e_q(n)$, to obtain an estimate of the original speech signal, $\hat{y}(n)$, one should have a smaller error Mean Squared-Error (MSE), $\text{MSE} = \frac{1}{N}\sum_n (y(n) - \hat{y}(n))^2$, than if the directly quantized signal $y_q(n)$ was sent, $\text{MSE} = \frac{1}{N}\sum_n (y(n) - y_q(n))^2$.

Assignment Description In Matlab, one may process the signals and listen to them as well. The assignment is to obtain speech data from the internet and to use Matlab to calculate least-squares estimates of the coefficients, $a(k)$, of the model (1) and residuals, $e(n)$, for a given speech signal sample data set $Y^N = \{y(n), n = 1, \cdots, N\}$, then calculate the difference in the MSE for the directly quantized speech signal $y_q(n)$ and the signal $\hat{y}(n)$ reconstructed from the quantized residuals. You are to listen to the original signal, $y(n)$, the directly quantized signal, $y_q(n)$, and the reconstructed signal $\hat{y}(n)$ and described any subjectly perceived differences.

Because the stationary linear model (1) cannot accurately represent the *entire* speech signal, you need to process the signal in blocks of 160 sample data points and calculate model coefficients and quantization levels separately for each block. This means that you will have to decompose the entire data sample into sequential, non-overlapping blocks of 160, process each block separately (to compute $y_q(n)$ and $\hat{y}(n)$ *within* each separate block)

[3]Indeed, if the linear model were perfect, the residuals $e(n)$ would be identically zero! The point is that for a very good fit of the linear model most of the "energy" in the signal $y(n)$ must be due to the linear dynamical part of the model, which corresponds to modeling, say, the resonant response of the human voice box, musical instruments, etc. Thus if most of the behavior of the signal $y(n)$ is due to the dynamics expressed by the linear filtering, and less to the residuals $e(n)$, then quantizing the residuals $e(n)$ rather than the original signal $y(n)$ should do "less damage" to $y(n)$.

and then reassemble the blocks to get the *entire* set of signal samples $y_q(n)$ and $\hat{y}(n)$, for $n = 1, \cdots, N$.

Procedural Details

Step 1. Download a speech signal from the Web. Search engines allow you to limit your search to just speech files. These typically have identifying extensions, such as *.wav*. (You can use any speech format that you want–this is just an example). For a .wav file, you can load the speech signal using the Matlab command *wavread*,

```
>>    y = wavread('file.wav');
```

The speech signal may start with some nonzero *constant* value such as -1 before the actual speech starts. In the array containing the data y, reset these values to 0 up to the point where the values in y begin to vary. You can then listen to the speech signal using the *sound* command in Matlab,

```
>>    sound(y);
```

Step 2. For each separate block of 160 sample points, calculate the quantized speech signal $y_q(n)$ for a specified number of quantization quantization levels, $L = 2^r$, where r is the quantization rate in bits–per–symbol.[4] Once a quantization interval value, q, has been determined, this can be computed via a single line of Matlab code:

```
>>    yq = round(y/q)*q;
```

One also needs to ensure that y_q does not exceed maximum and minimum threshold values. (I.e., quantization has to occur within some prescribed range of sound file values.)

The key, therefore, is to 1) find reasonably effective values of the quantization interval value q and, 2) determine threshold values for y. This is actually a nontrivial task and is discussed at length in graduate courses in speech coding. Because this is an undergraduate course, we need to find more accessible (but less effective) quantization methods.

Perhaps the most naive and easiest is to simply take

```
>>    q = (max(y)-min(y))/(L-1);
```

which will automatically limit the values of the sound file y to between round(min(y)/q)*q and round(max(y)/q)*q.

Unfortunately, *this approach is not robust and will break down* if y has a few unusually large values (so-called *outlier values*).

A modification of the above procedure that is more robust to the outlier effect is to *preprocess* the vector y as follows.

First, find the mean and standard deviation of the values for each separate block of the file. Then truncate values of y within each block which are far from the mean for the block (this will definitely kill the outliers, but will unfortunately kill some good values too). Do this

[4]Keep the number of quantization levels, L, constant over all the blocks. Note that you will have to break up your sound file into blocks of 160 sample points, quantize each separate block of data, and then reassemble the blocks into a single file that you can listen to.

as follows: letting σ be the standard deviation of y within the block, threshold (truncate) those values of y within the block that exceed the range $\pm\alpha \cdot \sigma$ from the mean to the upper and lower threshold values as the case may be. This produces a thresholded vector of data within the block, block y_{thresh}. Now we can apply the procedure describe above:

```
>>    q = (max(ythresh)-min(ythresh))/(L-1);
>>    yq = round(ythresh/q)*q;
```

Note that $\alpha > 0$ is a free parameter whose value you will have to set via trial-and-error. $\alpha = 1$ means that you are truncating y-values that are farther than $\pm$ one standard deviations from the mean, $\alpha = 1.5$ truncates sound file values that exceed $\pm$ one and a half standard deviations, $\alpha = 2$ truncates values that exceed $\pm$ two standard deviations, etc.

Once you have quantized the sound file, you should listen to the quantized speech signal. Try it for quantization rates r = 1, 2, 3, 4, 5, 6, 7, 8, (and various values of α) and listen to the difference in quality. You can calculate the (empirical) MSE by simple matrix multiplication,[5]

```
>>    MSE = (y-yq)'*(y-yq)/N;
```

Report the MSE for various values of α and $r = 1, \cdots, 8$. Is there any relationship between MSE (a *mathematical* measure of distortion) and your *subjective perceptual opinion* of quality based on listening to the sound?[6]

Step 3. Taking $\ell = 10$ (i.e., take the model order to be 10),[7] estimate the filter coefficients $a(k), k = 1, \cdots, 10$ for each separate block of 160 data speech data points using the least squares optimization technique discussed in class by setting an $Aa = b$ linear inverse problem. The Matlab command *toeplitz* might be useful in constructing the matrix A. Type 'help toeplitz' in Matlab to see how it is used. You will have to use the last 10 elements of the previous block (beginning with the second block) in order to construct the A matrix needed to estimate the vector of filter coefficients a.[8] The the first block can be initialized with zeros.

Step 4. Since there will not be an exact solution to $Aa = b$, you can calculate the residual error within each block as,

```
>>    e = b - A*a;
```

Step 5. Write a small FIR digital filter program that computes the residuals *directly* from eq. (2) and the audio data $y(n)$ using the filter coefficients learned in Step 3. For each data block, form the MSE between these residuals and the residuals computed in Step 4. *This MSE should be identically zero.*

[5]You can computer the overall MSE for the entire (very large) sound file by computing the MSE for each separate block and then averaging the per-block MSEs over all of the blocks.

[6]One criticism of the MSE measure of distortion is that it is an arbitrary mathematical criterion which might not correspond to human subjective judgement.

[7]Why $\ell = 10$? Good question! Maybe other values are better. (E.g., smaller values will make the least-squares problem more computationally tractable.) In practice one would have to determine the value that seems to work best, balancing a variety of metrics. There are frequency domain issues pertaining to the spectral content and statistical nature of speech and music that could be mentioned here, but instead the student is encouraged to peruse the literature.

[8]In order to obtain the initial condition values $y(0), y(-1), \cdots, y(-9)$ needed in (1).

Step 6. Write a small AR digital filter program that reconstructs the audio data directly from eq. (1) and the residuals computed in Steps 4 and 5 above using the filter coefficients learned in Step 3.[9] For each data block, form the MSE between the original audio signal samples and the reconstructed audio signal samples. *This MSE should be identically zero.*

Step 7. Quantize the *residuals* using the same quantization rates and procedure described in Step 2. Here, the problem of outliers is very important so the choice of a value of α for residual quantization will be very critical.

Reconstruct an estimate of the original audio signal samples from the quantized residuals using the AR digital filter you wrote in Step 6. Listen to the signal $\hat{y}$ constructed for the various quantization rates and compare to the original signal y and the directly quantized signal y_q.

Step 8. Calculate the MSEs for the reconstructed (quantization-rate dependent) signals $\hat{y}(n)$ and compare it with the previously calculated MSEs for the directly quantized signal. You should plot the two MSEs, corresponding to the two different errors $(y(n) - y_q(n))$ and $(y(n) - \hat{y}(n))$, as a function of quantization rate to compare their sizes. In your written description of the project you should discuss your results objectively (e.g., is the MSE smaller for the signals reconstructed from the quantized residuals?) and subjectively (e.g., is the perceived quality of speech better?) Remember to discuss and compare the results for different quantization levels.

NOTE: Because squared-error is a mathematical measure on discrepancy, it does not necessarily conform to what a human would perceive as better in the sense of intelligibility or quality of reconstructed sound.[10] For this reason for every quantization level of y and e, you need to not just compare the MSE, but you must provide a comparison based on your subjective measure of the quality of the perceived (listened to) reconstructed signal. Furthermore, if you can say that method a at quantization level b is subjectively equivalent to method c at quantization level d, that is very useful for determining which form of compression, if any, is superior in terms of channel bandwidth utilization.

Step 9. In reality, *quantized* filter coefficients, $a_q(k)$, are sent and received. Therefore, quantize the vector of filter coefficients determined in Step 3. Again, you will need to play with the value of α until you find a reasonably effective one. Note that large values of α favors non-distortion of the large coeffcents, while small values of α favors non-distortion of the small coefficients. Setting α very large amounts to letting the largest coefficients set the quantization level. Let $r = 4, 8, 9$ and quantize both $a(k)$ and $e(n)$ *after* $e(n)$ has been computed. (Note that you are quantizing the filter coefficients $a(k)$ in a per block manner, just like you quantized $yq(n)$ and $e(n)$ per block in the previous steps.)

Repeat Steps 7 and 8, replacing the coefficients, $a(k)$, by the quantized coefficients, $a_q(k)$. Remember to report what values of quantization rate *did* allow you to reconstruct the signal adequately and what values *didn't*, and your give your opinion of why some values didn't

[9] As you sequentially reconstruct the blocks, save the last 10 reconstructed audio samples in each block to use as initial conditions in the subsequent block.

[10] In fact it is an interesting line of inquiry to ask if there is a mathematical measure that conforms directly to human estimation of quality.

work (this is just to get you to think about the problem, so don't worry about getting answer right or wrong).

Do not be surprised if the quantized AR filter is unstable (in which case you will not be able to do this Step). Stability of AR filters is a very difficult subject. Unlike FIR filters which are always guaranteed to be stable (which is one reason FIR filters are liked so much by the signal processing and communications engineering communities). An advanced course in speech and audio compression describes how to obtain stable discretized AR digital filters.