SYDE 252

Matlab® Assignment 3

(Due date: 11:30 pm, Nov. 30, 2018)

Instructions:

In this assignment, you will need to fill in your code in designated lines in two Matlab .m files: *SYDE252_Assign3_Script.m* and *myMovePoles.m*. You should also include a Word file, detailing your thought process of completing the questions (see below). When submit your assignment, simply change the two file name into: *SYDE252_Assign3_Script_Group#.m* and *myMovePoles_Group#.m*. The '#' should be your group number. Make sure you also change the function name in the first line of *myMovePoles_Group#.m*, so it matches the file name. You should also make proper changes to *SYDE252_Assign3_Script_Group#.m*, from which *myMovePoles_Group#.m* is called.

Modelling of Speech signals

A rough, yet relatively useful model for the generation process of speech signals is illustrated in the following figure.

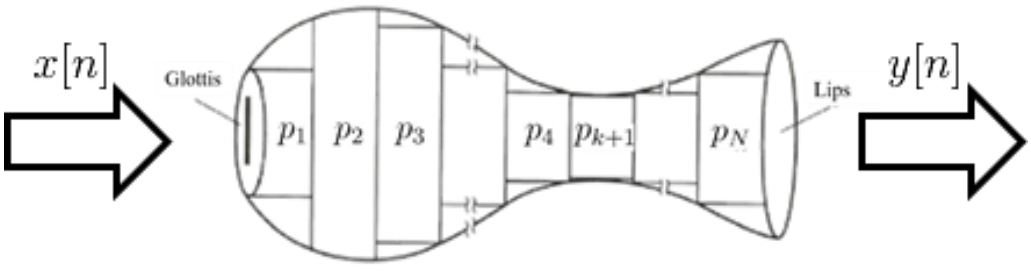

Figure 1 A simplified model of speech signal generation

Signal $x[n]$ models the air flow being pressed through the vocal cord. It is directly related to the words to be spoken. The vocal cord consists of a series of cascaded segments, each of which can be modeled as a first order LTI system, *i.e.* a single-pole system. The output of the vocal cord, in response to the air flow ($x[n]$, the input to the system) pressed through it, is the acoustic signal, modelled by $y[n]$. Since we will be conducting our analysis in discrete-time (sampled signal), the system function of the *i*th segments can be expressed as:

$$H_i(z) = \frac{G_i}{1 - p_k z^{-1}} \tag{1}$$

As such, the system function of the entire vocal cord of *N* segments is:

$$H(z) = \frac{G}{\prod_{i=1}^{N}(1 - p_i z^{-1})} \tag{2}$$

where p_i the pole of the *i*th segment. An equivalent form of the above is:

$$H(z) = \frac{G}{\sum_{i=1}^{N} a_k z^i} \tag{3}$$

where a_k are the coefficient of the polynomial, whose roots are p_i. In this exercise, the order the entire system, *i.e.* the number of segments of vocal cord, is 40. In general, $x[n]$ mainly carries information of the words being spoken, while $H(z)$ carries information regarding the structure of the vocal cord, such as the frequency content of the resulting acoustic signal, and consequently models the speaker's identity. The acoustic signal of each phoneme spoken by a particular person can be very well characterized by a pair of $x[n]$ and $H[z]$. In digital speech signal processing, there are well-established methods to estimate $x[n]$ and $H[z]$, from $y[n]$. This process is called deconvolution. The detail of this

process is beyond the scope of SYDE252. The included Matlab function *getModel.m* is provided for this purpose. This process is usually performed over a short time duration (e.g. 10 ms), during which a phoneme is being spoken and the parameters of $H[z]$ can be considered constant. This estimation process would repeat every 10 ms segment, and a pair of $x[n]$ and $H[z]$ will be obtained for each segment. For reasons beyond the scope of SYDE252, in digital communication systems, it is more efficient to transmit $x[n]$ and the parameters of $H[z]$ (*i.e.* a_k) than transmitting $y[n]$ directly. The only thing one needs to do at the receiving side (your cellphone or skype) is to synthesis $y[n]$, by passing $x[n]$ through a digital filter defined by a_k. This rough version of this synthesis process of is performed by the Matlab function called: *synthVoice.m.*

Please read through the first section of the provided script: *SYDE252_Assign3_Script.m* (line 1-15), run the code, and understand better the above speech processing steps.

Question (1): use your Fourier Transform function in previous assignments to show the spectra of the original audio signal, and the synthesized audio signal. And comment on you observations on the two spectra. Write your answer between line 21-30.

Modifying the speed of the speech, without (significantly) changing the tone of the speech

In the above digital speech analysis and synthesis, $x[n]$ essentially represents 'what is being spoken' and 'at which speed'. In fact, any text-to-speech engine will generate $x[n]$ according the texts to be processed, and then pass it through a predefined speaker model (*i.e.* a set of $H[z]$s). In this exercise, we would perform a simple manipulation of $x[n]$ to change the speed of the speech without changing its tone, at least not significantly.

When we play the audio signal at a half speed (line 34 of *SYDE252_Assign3_Script.m*), the tone is significantly lower (signal has a lower frequency) than the original. Conversely, when we play the audio signal at double the speed (line 36 of *SYDE252_Assign3_Script.m*), the tone is significantly higher (signal has a higher frequency).

Question 2: Explain which property of Fourier Transform can be used to explain this phenomenon. (Write your answer directly between 37-40)

In order to speed up or slow down the speech without changing the tone of the speech, we can modify $x[n]$. If we change the length of $x[n]$, without changing its general 'shape', then we effectively change the speed of the speech. Because $H[z]$ remains the same, the 'speaker', and consequently the tone of the speech will not be changed.

Question 3: Use Matlab's interpolation command *interp1* to generate $x_f[n]$ and $x_s[n]$, for double and half of the original speed, respectively. You need to complete line 42-43.

When $x_f[n]$ and $x_s[n]$ is generated, we can synthesize the corresponding speech (line 47-48). Then you can play the resulted audio signals to check if the desired effects are achieved (line 51-52). Note that you will probably notice that the tone of the audio is still changed although we only changed the length of $x[n]$ (but certainly not as much as line 34 and line 36). This is because the model described in Figure 1 is an extremely simplified model. With a more complex model, one could achieve this speed-changing effect much better.

Modifying the tone of the speech, without changing the speed of the speech

Next we are going to explore another aspect of speech synthesising. In the model described in Figure 1 and the equations, $H[z]$ models mainly the vocal cord properties, hence the

speaker's identity. Of course with our simplified model, we won't be able to use the parameters in $H[z]$ for biometric purposes (speaker identification). However, we are able to modify the parameters of $H[z]$ such that the tone of the speech is changed, while the speed of the speech is the same (using the original $x[n]$).

In our lecture on z-Transform (as well as in Laplace Transform), we see that the positions of the poles not only determine the ROC of the transform, but also have a significant role in determining the frequency characteristics of the corresponding Fourier Transform. For example, moving poles closer to certain frequencies will increase the magnitude of the Fourier Transform at those frequencies. Therefore, to modulate the tone of the speech, we can change the positions of the poles (p_k in Eqn. 3). If we collectively move the positions of all the poles toward higher frequencies (what is the highest frequency in Discrete-time Fourier Transform or z-Transform?), without changing the relative positions of the poles, as well as their magnitudes, we would create a new system with $H_f(z)$. When pass the original $x[n]$ through this new system, a speech signal will have a higher tone, at the same speed of the original audio signal. Conversely, if we collectively move the positions of all the poles closer to low frequencies (what is the lowest frequency in Discrete-time Fourier Transform or z-Transform?), we can generate a speech signal with lower tone at the same speed of the original signal.

Question 4: finish the Matlab function, *myMovePoles.m*. Please pay special attention to the fact that all the poles of the $H[z]$ are either real or conjugate pairs. Therefore, after moving the poles, you would want to make sure they remain either real or conjugate pairs. A simple way to achieve this: do NOT move real poles; only move

complex poles by pure rotation (change only phase angles) and maintain all the conjugate pairs.

Execute line 59-74 of *SYDE252_Assign3_Script.m* to check if your pole-moving function can achieve the desired effect. Again, because of the simplistic model used here, the generated speech is not ‘ideal’. More sophisticated models will be required to generate better, more satisfactory results.