Spectrogram Analysis and Applications

# Overview

The frequency characteristics of nonstationary signals vary as a function of time. This requires Fourier analysis of these signals to be localized and time-dependent, resulting in analysis methods known as time-frequency distributions. One elementary time-frequency distribution based on the FFT is the short-time Fourier Transform (STFT). The STFT applied to speech is often called a *speech spectrogram*. Spectrograms are well suited to analyze speech signals with their time-varying narrowband features.

In this project, you will:

* Calculate spectrograms of frequency modulated signals.
* Construct narrowband and wideband spectrograms of speech signals.
* Estimate a speech signal from a modified STFT.
* Modify the time scale of a speech signal.

The project zip file contains the audio files and MATLAB functions which you will need for this project. To access the zip file, click on the link below where you found this file on the web site.

## Spectrogram Definition

The spectrogram of a signal *x[n]* is defined as:



where *ω[n]* is the window used. The window will determine what portion of the signal is used for analysis and controls the frequency resolution of the spectrogram. The parameter *n* denotes the reference position of the window on the signal. Let the window be of length *Nwin* and nonzero only in the interval *0 ≤ k ≤ (Nwin - 1)*. The above equation reduces to:



# Computing Spectrograms in MATLAB

It is easy to construct spectrograms in Matlab since the STFT involves only windowing and the FFT. This project will require extensive use of the Matlab function spectrogram. It is suggested you take some time to fully understand the details of the spectrogram function. This can be done by typing help spectrogram at the Matlab prompt. For a more graphical presentation of the help (including figures) try doc spectrogram. A few important points about the spectrogram function for this project:

* Note that if the spectrogram is defined as it is in the previous section, the first column of the spectrogram generated by matlab is really , not *X0(ω)*.
* The spectrogram function with no output arguments will plot the results to the current figure.
* For real signals *x[n]*, the spectrogram function only calculates or plots a portion of the frequency components (from 0 to π) since the remaining are “redundant”. How are they “redundant”?
* It is common in speech processing to use a “short” time window and to plot frequency on the y axis. When you have loaded the speech signals mentioned below, the command spectrogram(s1,hamming(256),’yaxis’) will produce a good example spectrogram plot. Please use the ‘yaxis’ parameter for all your figures.

The MATLAB function diary may be helpful to keep track of commands when you are experimenting. Create separate M-file(s) to be able to rerun and test parts of your code. Do not turn in a log of all the commands you tried, just the source code that was relevant to solve the different parts of the project. In addition to this source code, please turn in the required plots, keeping in mind that you are expected to generate reasonable graphics.

# Frequency-Modulated Signals

1. Spectrograms display frequency variations as a function of time. In this section, we will attempt to develop some intuition about the STFT by analyzing signals with known frequency variations, specifically linear frequency modulated (FM) chirp signals. The mathematical form of a continuous-time linear FM chirp is:



Generate a discrete-time linear FM chirp signal *x[n]* using a sampling frequency of 5 MHz and letting the chirp rate be μ = 4.0 X 109. Assume that the continuous-time chirp lasts 200 μs. Generate a spectrogram for this signal *Xn(ω)* using 256-point FFT’s, a 256-point triangular window and an overlap of 255 samples between sections. (The MATLAB function triang can be used to construct the triangular window.)

1. Notice the distinct ridge in the time-frequency plot of part (1). Let us attempt to relate this to the functional form of the linear FM chirp signal. If the frequency modulation of *x[n]* is rewritten as , one could interpret *f1(t)* as the “instantaneous frequency” of *x(t)*. Another way to construct an “instantaneous frequency” of *x(t)* is the time derivative of the phase: where *x(t) = cos [φ(t)]*. Calculate both of these definitions of “instantaneous frequency” for the signal in part (1) and determine which definition corresponds to the slope of the ridge in the spectrogram.
2. Sketch the spectrogram if the chirp rate was changed to μ = 1.0 X 1010 . Verify your intuition by defining another chirp signal *x2[n]* with this chirp rate and computing its spectrogram. Compare this spectrogram to the one obtained in part (1) and discuss your conclusions.

# Narrowband and Wideband Spectrograms

The window used to compute the STFT determines the frequency resolution of the analysis. In this section, you will examine the effect of the window on the spectrogram of a voiced speech signal. You will need two speech files in the project zip file: s1.mat and s5.mat. Load in these files by typing load sl and load s5 at the Matlab prompt. Assume these speech signals are sampled at 8 kHz. If you wish to listen to these files, the Matlab function soundsc can be used to autoscale and play signals.

The exercises below depend on the frequency structure of human speech. Voiced human speech is produced by buzzing from the vocal cords (glottal excitation) into the resonant frequencies (formants) of the vocal tract. At high frequency resolution, we can see the individual harmonics of the glottal excitation signal at multiples of its pitch frequency. At low frequency resolution, the individual harmonics are blurred, but the emphasis applied by the vocal tract resonances is more obvious in the wide peaks. As the lips, teeth and tongue change the shape of the vocal tract, the resonant frequencies (formants) change the sound to correspond to various recognizable vowel sounds. Thus, the specific formant frequencies determine the vowel sound perceived by the listener.

1. One application of a spectrogram is to estimate the fundamental frequency as a function of time. This estimation can be performed using a narrowband spectrogram which has fine frequency resolution and poor temporal resolution. Determine the proper spectrogram parameters (FFT length, window length and overlap between sections) to construct narrow band spectrograms of s1.mat and s5.mat. Use a triangular window and FFT lengths that are powers of two. Experiment with different parameter values to obtain spectrograms that enable you to estimate and sketch the fundamental frequency of both speech signals as a function of time.
2. Another application of a spectrogram is to estimate the formant frequencies as a function of time. This estimate can be performed using a wideband spectrogram which has fine temporal resolution and poor frequency resolution. Determine the proper spectrogram parameters (FFT length, window length and overlap between sections) to construct wideband spectrograms of s1.mat and s5.mat. Use a triangular window and FFT lengths that are powers of two. Experiment with different parameter values to obtain spectrograms that enable you to estimate and sketch the formant frequencies of both speech signals as a function of time.

# Modified Short-Time Fourier Transforms

1. The STFT of a signal has a lot of redundant information. Note that an arbitrary function Xn(w) may not be a valid STFT. This can be seen by noting that overlappingportions of the input are used to construct *Xn(ω)* for different values of *n*. Taking the inverse DFT of each *Xi(ω)* may not result in the same values of *x[n]* for different values of *i*, therefore an arbitrary function *Xn(ω)* may not be a valid STFT.

There are different methods to estimate a signal from a modified STFT. In this section, you will estimate the signal by determining the signal *y[n]* which produces the smallest error for the following criterion:



Write a Matlab function to implement this estimation from a modified STFT. To simplify the programming, assume that the STFT that you are processing was created with 1024-point FFT’s and a 256-point rectangular window with segments overlapping by 128 points.

One way to approach this problem is to look at how a STFT is constructed with the parameters above. The first column of the STFT is constructed by taking the first segment of the signal in time and computing its FFT. In this case, that would be the 1024 point FFT of the first 256 points zero-padded to length 1024. The second column of the STFT is constructed by taking the next segment of the signal in time and computing its FFT. Since the segments overlap by 128 points, this would mean the 256 points from 128 to 383 are zero-padded to length 1024, and the 1024 point FFT is computed.

Now, if we want to estimate the signal from a modified STFT, we reverse the concepts above. Using the first column of the modified STFT, we can take an IFFT and the first 256 points should correspond to the time signal from 0 to 255. The IFFT of the second column of the modified STFT can be computed and the first 256 points will correspond to the time signal from 128 to 383. And we can repeat this for the remaining columns. Note that there are now two estimates of the time signal from 128 to 255 since the IFFT’s of the first two columns overlap in time. Because of the error criterion we are using (and the rectangular windows), we can just take the average of the estimates at each point. If we continue doing this, note that except for 128 points at the beginning and the end of *y[n]*, there are two estimates at every point in time. This illustrates the redundant information in the STFT.

The only input variable to this function, besides the modified STFT, will be the number of samples of the spectrogram (note: this could also be determined from the modified spectrogram itself). Also assume that the output signal *y[n]* will be real and that modifications to the STFT are made in a manner such that IDFT of each slice of *Xn(k)* will also be real. Even if conjugate symmetry is satisfied, taking the inverse DFT with the Matlab function ifft may result in a nonzero imaginary component due to precision error. In this case, take the real part of the ifft result to eliminate the precision error. Also, note that the MATLAB function spectrogram will only compute *Xn[k]* for 0 ≤ k ≤ 512 since the original sequence was real, so you will have to reconstruct *Xn[k]* for 513 ≤ k ≤ 1024 to be able to compute a 1024-point inverse DFT.

Test this function by using the unmodified spectrogram of the speech signal in vowels.mat as an input to your program and comparing the output to the original signal. Plot the difference between the input and output.

# Changing the Rate of Speech

1. There are many scenarios where changing the rate of speech may be useful. One example is slowing down speech for beginners trying to learn a new language. Simple schemes that temporally interpolate or decimate the speech signal will change the speech rate but also alter characteristics of the speech such as the fundamental frequency and formant frequencies. The altered speech characteristics can make the speech signal unintelligible. Ideally, one would like to be able to modify the rate of speech while preserving the speech characteristics. The STFT can be used to accomplish this. In this section, you will modify a STFT and then use the function that you wrote in the previous section to obtain a speech signal with a different time scale than the original signal, but with the frequency characteristics of the speech preserved.

First, load in the speech signal from vowels.mat and compute the spectrogram using a 256-point rectangular window with an overlap of 128 points per section, 1024-point FFT’s, and assuming a sampling rate of 8 kHz. Then compress the time scale by a factor of 2 by throwing out every other slice in time, and use the function from the previous section to obtain a faster speech signal with the same frequency content. Plot the speech signal before and after time scale modification. Listen to the signals to verify that your processing is working correctly.