Digital Signal Processing (18-491/18-691)
Spring Semester, 2025

The 18-691 Final Project
Short-Time Fourier Analysis

Issued: 4/17/25

Due: 4/25/25 at 0100 via Gradescope, although 2-day late submissions will be accepted without penalty

Background: At the beginning of the semester we noted that students enrolled in 18-691 will be required to complete one additional assignment in order to receive graduate credit for the course. This is necessary in order to comply with ECE Department rules that state that courses in multiple tiers with parallel numbers must have at least somewhat different content to justify the parallel numbering.

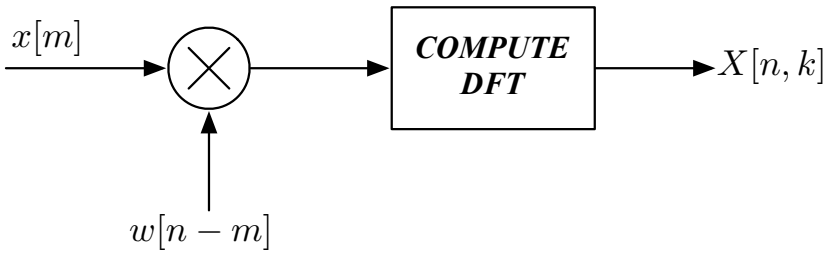
The final assignment for the 18-691 students will be an implementation of short-time Fourier transforms in various configurations. The project is intended to be easy and fun.

The final project will be given a grade equal to one homework assignment. Nevertheless, the grade for the final project will not be dropped from the final grade calculation, no matter how low it is. Each 18-691 student will work on his or her own on this project. The deadline for the final assignment will be April 27.

Introduction: In this assignment you will implement a system that performs short-time Fourier analysis (STFA), which is a key enabling technology for most practical processing of speech and audio. STFA was discussed in class in Lecture 26, and the techniques used for this assignment were introduced in that lecture. The two recommended references (in addition to the class notes and the Powerpoint presentation from Lecture 26) are the notes on short-time Fourier transforms (STFTs) from 18-792 (ADSP) and the chapter on STFTs by Nawab and Quatieri in the Lim and Oppenheim edited text used in ADSP. Both of these references are available online on the Lectures page of the 18-491/691 course website.

As described in the ADSP class notes, STFA is used to enable us to analyze the frequency components of a signal as they evolve over time. In Secs. 3.2 and 3.3 of the ADSP class notes we describe three ways of implementing the STFT: the Fourier transform implementation, the lowpass filter implementation, and the bandpass filter implementation. The STFT $X[n,k]$ of a time function $x[n]$ can be thought of as a matrix of complex coefficients, indicating the instantaneous power of the signal in discrete time frames (indexed by the horizontal axis of the matrix) and frequency

bins (indexed by the vertical axis of the matrix). In this assignment, we will implement different versions of the STFT for different purposes.



Problem C11.1: (Spectrographic display)

(You will work this problem by completing the main file `main_11_1.m` that we provide.)

The **spectrogram** is the major tool used to visualize signals as a simultaneous function of time and frequency. In essence it is a display of the log magnitude of the STFT of a signal, plotted as a function of time and frequency. In this problem you will complete two MATLAB function we provide called `spectrogram_ham_691_dft.m` and `spectrogram_ham_691_bpf.m` that implement a spectrographic representation of a speech waveform using short-time Fourier analysis, displaying them using MATLAB. (You also complete your work in Python if you wish.) You can compare your result to that produced by the MATLAB routines `specgram` or `spectrogram`. Initially you will realize the STFT using the Fourier transform implementation as depicted above.

The speech wave that you will use as input is the utterance "Welcome to DSP-I" that we have used in class from time to time, but downsampled to a sampling rate of 8 kHz. Obtain the waveform `welcome8k.wav` on the available in the `h691_extras` folder on the homework page on the Web.

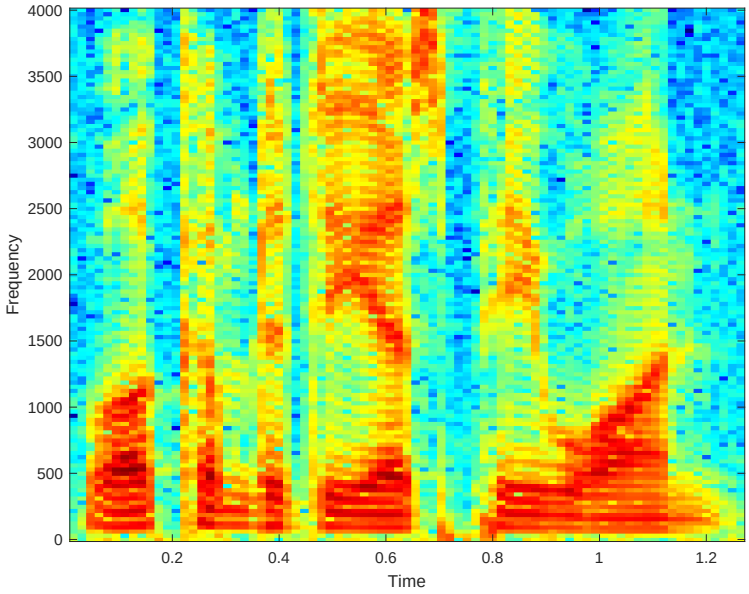
Note: Turn in the MATLAB scripts that you used to obtain your answers in all of the following problems.

(a) Given that you are using Hamming windows and the Fourier transform implementation of the STFT, what is the most efficient choice of the time between frames given the window length that would enable you to completely reconstruct the original time function using the overlap-add (OLA) method)? Explain your reasoning.

(b) What is the minimum DFT size that you would need to use for a window of length N?

(c) What is the window length N that you would need to obtain an effective bandwidth of the analysis window of W Hz? (Recall that the analysis window $w[n]$ can also be considered as a filter, as in the lowpass and bandpass implementations of the STFT.) Express your results in terms of the sampling frequency F_s, and the desired filter bandwidth W.

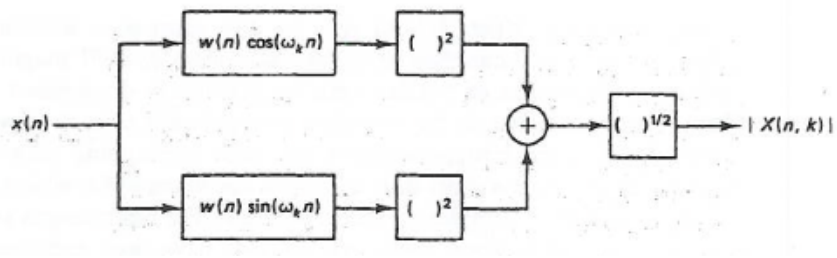
(d) Using your results from parts (a) through (c), calculate the complex STFT coefficients $X[n, k]$ for the "Welcome to DSP-I" utterance separately using effective filter bandwidths (from the windows) of 50 and 350 Hz. State how you achieve these bandwidths using Hamming windows.



(e) Use Hamming windows for your analysis function. Create two vectors, `t` and `f` representing the nominal frame times and center analysis frequencies, respectively. Display your output using the MATLAB commands

```
IMAGESC(t,f,10*log10(abs(X)));
axis xy;
COLORMAP(JET);
```

Your spectrograms should look something like the figure above, although this example was created using a different window length.



(f) Repeat the analyses above, but using the bandpass filter implementation of the STFT using real coefficients, as depicted in the figure above (reprinted from Figure 6.29 in Lim and Oppenheim). Note that the required minimum time between frames for the bandpass STFT implementation will be different from the minimum time between frames for the Fourier transform implementation. (Consider this to be a downsampling problem, with the downsampling ratio determined by the effective cutoff frequency of the window as a lowpass filter, as discussed in Sec. 3.4 of the ADSP class notes.)

Plot the spectrograms you obtain as above, and compare them to your results using the bandpass implementation in part (f) with your results using the Fourier transform implementation in part (e).

What you should turn in on paper:

- A block diagram and description of your systems as you implement them. Be sure to state the critical choices you made in your window length, time between analysis frames, and other design decisions. State how you arrived at these decisions.
- Plots of the two spectrograms of parts (e) and (f) using MATLAB
- Hard copy of your MATLAB scripts

What you will submit electronically on Gradescope (assuming that you are working in MATLAB):

- Your completed files `main_11_1.m`, `spectrogram_ham_691_dft.m`, and `spectrogram_ham_691_bpf.m`.
- Your spectrograms, in .pdf format. Label the files with your name and the suffix "`11_1e_50`", "`11_1e_350`", "`11_1f_50`", and "`11_1f_350`", as in "`rms11_1e50.pdf`".

Problem C11.2: (Signal separation using STFTs)

In this problem we replicate a demo that shows that we can reconstruct intelligible speech from fragments even if only about 25% of the speech is used in the restoration. (This demo also works for five speakers, but breaks down with more speakers.) When applied to automatic speech recognition (ASR), this approach has been called "missing feature recognition." In principle, speech is reconstructed using only the subset of time-frequency elements that are considered to be "good," which in this context means being relatively undistorted by the effects of additive noise, competing speakers, and other degradations.[1] While these approaches can be quite effective for speech enhancement and speech recognition, in real applications it can be quite difficult to determine which time-frequency segments are in actuality the "good" ones. In this homework problem we will circumvent this issue by using so-called "oracle" knowledge, which is to say that the separation will be based on knowledge that would not be available in a realistic scenario.

While the end result of Problem C11.1 was a series of spectrogram displays, in this problem we will reconstruct audio from the STFT coefficients using different methods and with different modifications. In the folder `h691_extras` we provide a mixture of four speech signals (`4speakers.wav`), along with the five signals recorded separately (`Speaker1.wav` through `Speaker4.wav`). The sampling frequency for each of these signals is 16 kHz. We will begin by resynthesizing one of the speech signals in isolation. Subsequently we will separate the mixture by performing four STFTs on the individual speech signals and performing each reconstruction by selecting only those time-frequency elements for which a given speech signal has the greatest power of the four. The four individual speech signals will be reconstructed by performing the ISTFTs using only the subset of

[1]The major contributions of the doctoral thesis of Bhiksha Raj were the introduction of two useful algorithms to perform missing feature restoration. Some of these techniques are reviewed in Raj, B., and Stern, R. M. (2005). "Missing-Feature Methods for Robust Automatic Speech Recognition," *IEEE Signal Processing Magazine*, September 2005.

time-frequency elements for which each signal has the greatest power, setting the other segments to zero.

The speech signals in this problem are all sampled at 16 kHz, and we will be using Hamming windows of duration 20 ms in the analysis and synthesis.

(a) Let us consider first the analysis and resynthesis of one of the signals. In our first pass, we will use the Fourier transform implementation of the STFT, and we will resynthesize the signal using the overlap-add (OLA) method. Begin by using `audioread` to read in the file `Speaker1.wav'`. Determine the window size needed for a 20-ms window duration at 16 kHz, and the subsequent minimum DFT size. Listen to the speech signal using the MATLAB command `soundsc`.

(b) Compute the full STFT $X[n,k]$ of the signal as you had in Problem C11.1 but with different parameter values for the new window length and number of channels. Use the maximum time between frames that is consistent with OLA synthesis with 20-ms Hamming windows.

(c) Compute the ISTFT of $X[n,k]$ using the OLA method as described in the ADSP notes and in the Nawab and Quatieri chapter in Lim and Oppenheim. Listen to the resulting signal using `soundsc`. It should sound exactly like the original signal.

(d) Now read in the audio file `4speakers.wav`, again using the MATLAB command `audioread`, as well as the individual speech signals `Speaker1.wav` through `Speaker4.wav`. Compute four separate STFTs for each of the four individual speech signals. For each value of the frame index n and the frequency bin k, determine which of the speakers (1 through 4) produced the value of $X[n,k]$ with the greatest magnitude. Then resynthesize four separate signals, in which the first resynthesized speech sound is obtained by using the original STFT coefficients only for the subsets of values of n and k for which the STFT of Speaker1 has the greatest magnitude. Set the remaining values of $X[n,k]$ to zero. (Make sure that you make consistent choices for positive and negative frequencies, so that the resulting time function will be real.) Listen to the resulting waveform. It should sound like the original speech waveform with some distortion, but the speech should be quite intelligible. Repeat for Speakers 2 through 4. Save the resynthesized audio waveforms as `Speaker1rf.wav` through `Speaker4rf.wav`.

(e) Repeat part (d) but using the lowpass filter STFT method for analysis, and the filterbank summation (FBS) method for the resynthesis. Note that you will need to use a different frame separation for this combination than you employed for analysis using Fourier transforms and resynthesis using OLA and in part (d). State all your choices and why you made them. Save the resynthesized audio waveforms as `Speaker1rlp.wav` through `Speaker4rlp.wav`.

What you should turn in on paper:

- A block diagram and description of your analysis and synthesis systems as you implement them in parts (b) and (c). Be sure to state the critical choices you made in your window length, time between analysis frames, and other design decisions. State how you arrived at these decisions.
- The spectrogram for the complete resynthesized signal `Speaker1.wav` you obtained in part (b).

- A block diagram and description of your analysis and synthesis systems as you implement them in part (e). Be sure to state the critical choices you made in your window length, time between analysis frames, and other design decisions. State how you arrived at these decisions.
- Spectrograms for the eight resynthesized audio signals from parts (d) and (e).
- Hard copy of your MATLAB scripts

What you will submit electronically on Gradescope (assuming that you are working in MATLAB):

- Your completed file `main_11_2.m` plus any additional MATLAB files you use for the analysis and synthesis.
- Your spectrograms, in `.pdf` format, for each of the four speakers and each of the two analysis/synthesis methods (Fourier transform/OLA and lowpass filtering/FBS). Label the files in a fashion that specifies the speaker and reconstruction method (*e.g.* `Speaker1rf.pdf`, `Speaker2rlp.pdf`, etc.)
- The audio files, in `.wav` format, for each of the four speakers and each of the two analysis/synthesis methods (Fourier transform/OLA and lowpass filtering/FBS). Label the files in a fashion that specifies the speaker and reconstruction method (*e.g.* `Speaker1rf.wav`, `Speaker2rlp.wav`, etc.)