FH JOANNEUM – University of Applied Sciences
Elektronik and Computer Engineering
Florian Mayer

Applied Signal Processing
Filtering and Extended Logic
Computer Laboratory Block THREE

Please submit a single zip-File (and just use zip) including your documentation and all simulation files, e.g., MATLAB files (*.m). You have to zip (and use only zip) all these files to one single file.

Please make sure that your approaches, procedures and results are clearly presented.
!! For all exercises in which audio examples are played, start with a low volume!!

Example 1: Window Functions

Investigate in the MATLAB functions `hanning, hamming, blackman, blackmanharris, chebwin, fftshift, fft` using the MATLAB help

In order to perform a proper Short-Time Fourier Transform (STFT), we need to investigate in window functions and their properties.

(a) Compare the following window function:

- `hanning`
- `hamming`
- `blackman`
- `blackmanharris`
- `chebwin`

Create and plot the frequency response of all windows. (Please note the correct representation of the axis)

Listing 1: Plot Filter Response Idea

```
M = %%% window length in samples;
n = -M / 2: M / 2-1;
window = %% Add window-type
figure;
plot (n, window);        %%% time domain plot
L = length (window);
NFFT = 1024;     %% number of samples for fft
X1 = fftshift (); %%%% Create a weighted fft of the window function,
    what is fftshift actually doing?
Freq = (-NFFT / 2: NFFT / 2-1) / NFFT;      % Frequency vector
```

```

        Y1 = X1 (1: length (Freq)) / max (X1);
        mag_dB =  10*log10(Y1);         %% dB Calculation
        plot(Freq,mag_dB);

        %% TODO: ADD Proper axis numbering and labeling
```

Example 2: Short-Time Fourier Transform ANALYSIS

Investigate in the MATLAB functions `fft, audioread, resample, pcolor, repmat` using the MATLAB help

For analyzing audio signals, the time domain and the frequency domain representations may be used for different tasks but for some applications we need a representation that jointly represents the time-frequency information. The short-time Fourier transform can be used for this purpose. The STFT consists of the DFTs of portions of the time-domain signal. We will be interested in the STFT of audio signals, so let us first see how to read an audio signal. We will work with .wav files.

(a) For reading in audio, we use the function `audioread`. Let us read one of the speech files provided at moodle. A $25kHz$ signal where a female or male speaker utters a certain test sentence. In order to properly work with this sample, we need to resample the file (downsample) to $16kHz$. Use the command `resample`, how does it work?

(b) As a next step we should implement our own STFT function in order to analyse the behaviour of frequency within a signal over time. Therefore we need to take `fft` results of smaller time snippets and plot them respectively.
Implement the function `[X_stft, vecT, vecF] = stft_ece(x,win,overlap,fs)`, investigate in `stft_ece_help.m` in order to get the big picture. Also discuss with your instructor.

(c) Investigate some time to compute the Time-Frequency representation of some of the provided sound files. Do you notice a difference by altering the window size, window function as well as the number of samples to overlap?

Test you created function in `run_stft.m`!

Example 3: Short-Time Fourier Transform SYNTHESIS

Investigate in the MATLAB functions `ifft, audiowrite, resample, sound, repmat` using the MATLAB help

The STFT representation is useful because it allows us to obtain interesting effects via simple operations on the STFT coefficients. So, when processing a signal with the STFT, the idea is to compute the STFT coefficients, modify the coefficients and then invert the STFT to obtain the modified signal in the time domain. We now investigate some time in order to invert the STFT.

(a) Recall that the columns of your STFT hold the DFTs of the segments of x, modified by a window. For instance, we can obtain samples of x multiplied by a sliding window. Implement the function

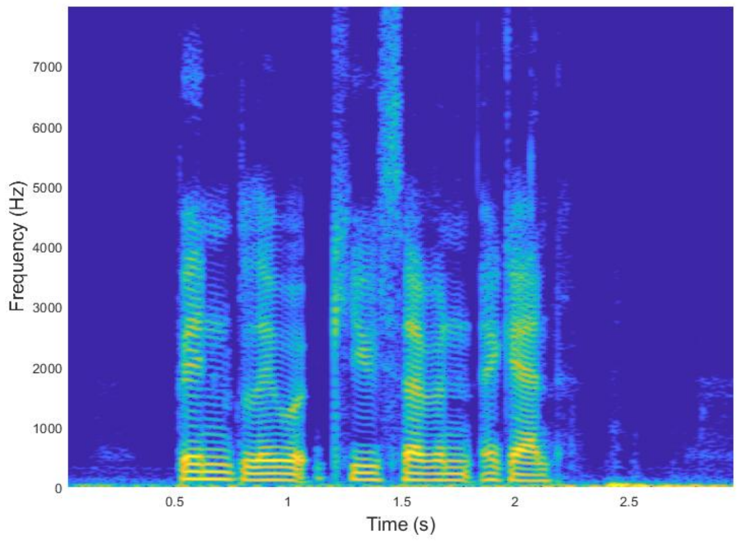

Figure 1: Possible STFT result

`[x_ifft] = istft_ece(X,win,overlap)`, investigate in `istft_ece_help.m` in order to get the big picture of synthesizing the everything back into time domain. Also discuss with your instructor.

(b) Plot and Play your re-created sound file! Do you notice a difference?

Test you created function in `run_stft.m`!

Example 4: The Notch Filter

Investigate in the MATLAB functions `filter, poly, freqz, zplane` using MATLAB help

You can design a filter blocking a single frequency by placing a zero on the unit circle at the desired frequency. One major drawback with placing only a single zero, corresponding to an FIR-filter where all the poles are located at the origin of the z-plane, is that a large area around the zero is significantly attenuated. To achieve a near flat response except close to the frequency we wish to cancel, the poles can be moved out from the origin towards the zero. The closer the poles are to the zeros, the narrower the notch. These filters are called notch filters. The equation for a single-frequency notch filter is as follows:

$$H(z) = \frac{(1 - e^{j\omega_0} z^{-1})(1 - e^{j\omega_0} z^{-1})}{(1 - re^{j\omega_0} z^{-1})(1 - re^{j\omega_0} z^{-1})}, \quad (1)$$

where ω_0 is the normalized frequency to be canceled and r is the radius of the pole. The two numerator terms are the two complex conjugate zeros, and the two denominator terms are the two complex conjugate poles. Filters with multiple notches is designed by multiplying more complex conjugate poles and zeros at the desired frequencies to the numerator and denominator. The radius r controls the width of the notch, and is typically close to and less than 1 so that is is placed just inside the unit circle. The zeros are placed on the same frequencies, but on the unit circle.
From the list of frequencies you wish to cancel, you can create a list of the poles (located near the unit circle at radius r) and the zeros (located on the unit circle):

Set the radius of the poles, r, to a suitable value. Note than the frequency spectrum is symmetric, and this is why the negative frequencies ae also considered in constructing the poles and zeros. The numerator and the denominator polynomials of the filter can now be created and the filter is applied on the sound file x:

Listing 2: Create Poles and Zeros for the desired frequencies of the notch filter

```
        b = poly(z);
        a = poly(p);
        y = filter(b,a,x);
```

(a) Create a Notch filter that tries to cancel out sinusoidal disturbances of `speechPlusOneSinDist.wav` and `speechPlusThreeSinDist.wav`. In order to create your notch filter, analyse the signal using the STFT function. You will get an idea of the characteristics of your frequency components and which frequency you want to filter out.

(b) If the disturbances are not canceled, go back and redesign the filter with new added or adjusted notch frequencies.

(c) Plot the z-plane as well as the magnitude and phase response of your filter.