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

Applied Signal Processing
Analysis in Frequency-Domain
Computer Laboratory 2

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: Discrete-Time Fourier Transform - Implementation

(a) Implement the discrete-time Fourier transform `[vX]=computeDTFT(vn,vx,vw)` without using a loop. Consider the general form of the DTFT:

$$X(\omega) = \sum_{n=-\infty}^{\infty} x[n]e^{-j\omega n}$$

(b) Compute the DTFT for the given input signal $x[n] = [1, 5, -1, 6, 3]$ using your function `computeDTFT`. Plot your result. Does the result agree with your expectations? Why or why not?

(c) Compare your function with the MATLAB command `fft` for the signal $x[n]$. Do the results agree? What are the restrictions on n and ω_k? How can you modify the signals or the results so that both calculations agree? You can use the command `norm(vxX1-vx2,2)` to calculate the Euclidean norm between the two results (see the MATLAB reference pages for details).

Example 2: Discrete-Time Fourier Transform - Properties

(a) Investigate in the built-in MATLAB function `freqz, audioread, sound` using the MATLAB help:

- `[H,w] = freqz(b, a, N)`
- `[H,w] = freqz(b, a, N, 'whole')`
- `[H,w] = freqz(b, a, N, fs)`

What is the difference between those function calls?

(b) Download all four animal sounds from the Moodle course site. Use the built-in MATLAB function `audioread()` to load the soundfiles into the workspace properly.

(c) Plot the time signal and the DTFT of each one. Please title your plots and label your axes. For each DTFT, choose N greater than the length of the signal in order to get a smooth plot. Comment on what you see in the DTFT's.

(d) So far you have plotted with the frequency axis in radians, which may be hard to interpret. If you know the sampling rate of the data, however, you can "un-normalize" the frequency axis while keeping the same plot shape. To do this job, we can use `freqz` with different parameters:

```
 [x, Fs, b] = wavread( 'cat.wav' );
[H, F] = freqz( x, 1, length(x), Fs );
```

What is the frequency of the first peak (fundamental frequency) of each animal sound?

Example 3: Simple Three-Band Equalizer

Investigate in the MATLAB functions `freqz, filter, sound, audioread` using the MATLAB help

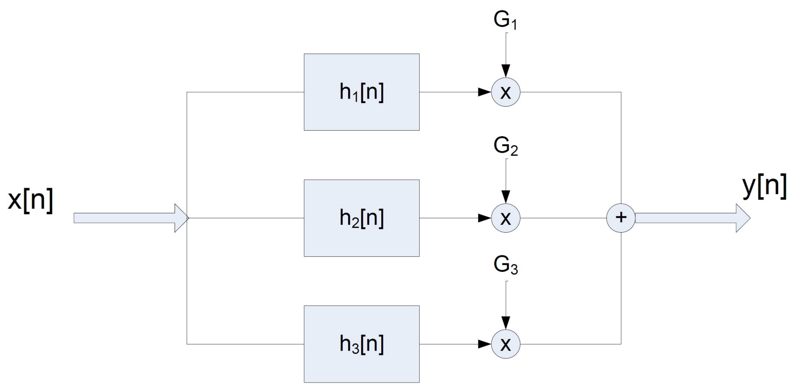
(a) The music `music.wav` file is standard CD quality, sampled at 44.1 KHz with a 16-bit word size for each channel. For your report, answer the following: If the file is 10 seconds long, then how many samples of data will there be? (Hint: the data samples are in stereo.) If we needed to transmit at this CD quality, with no compression, how many bits/second would be needed?

Filter coefficients

Filter One	B1=	0.0495	0.1486	0.1486	0.0495	
	A1=	1.0000	-1.1619	0.6959	-0.1378	
Filter Two	B1=	0.1311	0	-0.2622	0	0.1311
	A1=	1.0000	-0.4824	0.8101	-0.2269	0.2722
Filter Three	B1=	0.0985	-0.2956	0.2956	-0.0985	
	A1=	1.0000	0.5772	0.4218	0.0563	

(b) We have three filters, each defined by the difference-equation coefficients in the **filter coefficients table**. What are the properties of each filter? You will learn how to design them later. For this moment, just investigate which one is LP, HP, BP. Use `freqz` with $Fs = 44100$ and plot the frequency response of each filter and the pole-diagram using `zplane`.

(c) The idea of an equalizer is to filter, weight, and sum. Audio mixing boards do this, though with many more than the three channels we have here. Use the MATLAB code below to finish your own equalizer script. Try different values for $G1$, $G2$, and $G3$ (shown in the block diagram) and listen to the resulting $y[n]$.



Listing 1: Weighted Filter example

```
clear all;clc;close all;
% prepare B,A for each filter here
% load music
[x,Fs,b] = audioread('music.wav'); %Examplified function call
x_left = x(:,1);
x_right = x(:,2);
% filter 1
y1_left = filter(B1,A1,x_left);
y1_right = filter(B1,A1,x_right);
y1 = [y1_left y1_right];
% continue here for filter 2 and 3
% at the end, we need to sum
G1 = 1.1;
G2 = 0.5;
G3 = 2;
y = G1*y1 + G2*y2 + G3*y3;
sound(y,Fs);
```

(d) Compare the filtered audio with the original signal. What does each filter do to the music? Why don't you hear $y1$, $y2$, $y3$ independently as distinct sources?

Example 4: Fast Fourier Transformation - Reverse Engineering

(a) Investigate the `radix_fft_dit.m` script. Identify and note the purpose of key sections: zero-padding, bit-reversal, and FFT computation loop.

(b) Analyze the `run_fft.m` script. Understand how it generates a test signal and how the `radix_fft_dit` function is applied.

(c) Make a minor change in the `radix_fft_dit.m` script (e.g., alter a loop). Observe and document the effects on the FFT output when `run_fft.m` is executed.

(d) Test signal parameters (frequency, amplitude) in `run_fft.m`, does aliasing occur? Analyze how these changes affect the FFT output.

Example 5: The MATLAB FFT, what is the circular convolution?

Investigate in the MATLAB functions `fft, ifft, conv` using the MATLAB help

(a) Generate a signal $x[n]$ with the sampling frequency $fs = 200Hz$ and a sample length of $N = 512$ samples. The signal should include the following frequencies $f = [13, 56, 60]Hz$. Use the function `fft()` to compute the DFT of the signal. Create a proper frequency vector in order to plot a proper frequency spectrum. What do you observe?

(b) Now change the underlying frequencies within the signal to $f = [13, 56, 120]Hz$. What could be observed now? What is actually happening?

(c) Calculate the (as last time) linear convolution of the two given signals $x_1[n]$ and $x_2[n]$ and plot the result. Two signals are given by

$$x_1[n] = 1\delta[n] - 1\delta[n-1] + 3\delta[n-2]$$

and

$$x_2[n] = 2\delta[n] - 1\delta[n-1] + 1\delta[n-2].$$

(d) Calculate the DFT (MATLAB `FFT`) of both signals, multiply the DFT coefficients, and calculate the inverse of the result. Why is the result different from (c)?

(e) Concatenate 2 zeros to the given signals and repeat (c) again. Why does the results now agree with (d)?

Example 6: Feature Extraction - Speech recognition

Investigate in the MATLAB functions `audioread, fft, histogram` using the MATLAB help
In order to illustrate the use of the FFT, we investigate in a small speech recognition algorithm. The main goal is not to teach speech recognition techniques, but to show that the spectrum of a signal provides us useful information of a signal. Our file set contains of 40 'yes' and 'no' files, where male and female speaker utter the phrase. ($Fs = 16000$) The main goal is to implement a function `yes_no.m` that distinguishes between an uttered 'yes' and 'no'.

(a) Compute the power spectrum of a signal using the `fft` compare an uttered 'yes' and 'no' example. What do you notice comparing these two spectra? In order to realize this we could create use the second equation which also is related to parsevals theorem:

Parseval's theorem: $\sum_{n=0}^{+N-1} |x[n]|^2 = \frac{1}{N} \sum_{k=0}^{+N-1} |X[k]|^2$

$$X[k](dB) = 20log(|X[k]|^2)$$

(b) What could be a possible feature in order to differentiate between these sets. One possible feature would be to take the sum of the magnitude of the FFT components that correspond to the frequencies of NO interest and divide it by the sum of the magnitude of the FFT components that corresponds to the frequency band of interest. (HINT: We could pick these proper regions by identifying a suitable range on the x-Axis) Write a function that calculates the ratio between the FONI and the FOI.

(c) Next it is necessary to find a threshold value that separates the feature value for yes from that of no. One way to find an appropriate threshold is to calculate the feature for all of the training files and to examine the histogram for the yes values and the no values. Therefore, for all files we need to store the ratio result in one vector in order to perform the build in function `histogramm` on this vector.
Plot the two histograms and decide a suitable threshold.

(d) Use the provided function `YESNOFeature.m` in order to test your recognition of 'yes' and 'no' with your own recorded voice.

(e) Try to do the same for 'up' and 'down' (OPTIONAL).