Signals and Systems

Final Project

Dr. Marco Fabris marco.fabris.1@unipd.it
Dr. Jacopo Pegoraro jacopo.pegoraro@unipd.it
Giorgia Disarò giorgia.disaro@phd.unipd.it

Instructions

This report consists of two exercises in which the concepts of filtering and sampling must be applied to audio signals. In the first exercise, you are required to restore an audio track corrupted by disturbances, while in the second, you are asked to study the effects produced by the sampling of an audio signal. The exercises must be completed using two separate MATLAB scripts, producing clear code with brief comments to aid understanding.

Additionally, a report (maximum 4 pages) in PDF format must be written, briefly describing the work done in the exercises. The report should contain answers and arguments in response to the questions in the assignment. The required plots, as well as any additional ones considered relevant, must be included. All plots must be clear and include grids and any useful information to facilitate their interpretation.

Submission Guidelines

The MATLAB scripts and the report must be submitted through the designated section of the course Moodle page by **11:59 PM on June 15, 2025**, for students intending to take the second partial. For all other students, the submission deadline is **11:59 PM on June 30, 2025**.

A total of **three files** must be uploaded:

- Two MATLAB script files for Exercises 1 and 2, named respectively: `surname_name_studentID_ex1.m` and `surname_name_studentID_ex2.m`
- One PDF file containing the report and all required plots, named: `surname_name_studentID_report.pdf`

Submissions that do not follow these instructions will not be evaluated.

Exercise 1: Audio Track Restoration [1 POINT]

Given the file `audio1.mat`, which contains an audio signal corrupted by disturbances, complete the following tasks:

1. Load the file `audio1.mat` using the `load` command. It contains the signal `x` and its corresponding sampling frequency `Fs`.

2. Listen to the signal `x` using the commands provided in the **Appendix** of this document.

3. Plot the magnitude of the Fourier Transform of the signal `x`, considering that the signal starts at time $t = 0$. At which frequencies does the disturbance occur?

4. To remove the disturbance, you may use band-stop filters capable of attenuating spectral components in a very narrow frequency range (namely, notch filters), depending on their quality factor Q. Use the following function to design the filters:

```
% Design a notch filter
% f0 = center frequency of the filter [Hz]
% Fs = sample rate of the signal to be filtered [Hz]

function NF = NF_design(f0, Fs)

    N = 6;          % filter order
    Q = 30;         % quality factor of the filter

    % Tolerances in passband (deltap) and stopband (deltas)
    deltap = 0.01;
    deltas = 0.001;

    % Tolerances expressed in dB: [A]_dB = 20*log10(A)
    Ap = 20*log10(1 + deltap) - 20*log10(1 - deltap);  % (
        approximate)
    Ast = -20*log10(deltas);

    % Filter specification
    SPEC = 'N,F0,Q,Ap,Ast';
    NF_spec = fdesign.notch(SPEC, N, f0, Q, Ap, Ast, Fs);
    NF = design(NF_spec);

    % Plot of the frequency response
    % The blue line represents the frequency response of the
        designed filter
    fvtool(NF);
end
```

Listing 1: Design of a Notch Filter

where f_0 represents the center frequency of the filter at which the attenuation is maximum, and F_s represents the sampling frequency of the signal to be filtered. Both function arguments are scalars expressed in Hz. Choose the value of the center frequency f_0 appropriately, using the frequency response plot generated by this function to assist you.

To build the filter (or filters, if necessary), you can use the command

```
h = NF_design(f0, Fs);
```

where f_0 is the center frequency and F_s is the sampling frequency of the signal to be filtered. The filtering is then performed using the command:

```
x_filt = filter(h, x).
```

5. Once the disturbance has been removed, listen to the filtered signal and produce a plot comparing the magnitude of the Fourier Transform of the filtered signal x_{filt} with that obtained in point 3. Have you succeeded in completely removing the disturbance? Provide a brief comment.

Exercise 2: Sampling of an Audio Signal [2 POINTS]

Save the filtered signal from the previous exercise, along with the corresponding sampling frequency, in a MATLAB file named `audioNew.mat`. If you were unable to obtain the filtered signal, the file will be provided to you. Given the file `audioNew.mat`, complete the following tasks:

1. Load the file `audioNew.mat` using the `load` command. It contains the signal `x_new` and its sampling frequency F_s.

2. Listen to the signal `x_new` using the commands provided in the **Appendix** of this document.

3. Construct the signal `xsampled` by sampling the signal `x_new` at frequency $F_{s1} = F_s/25\,\text{Hz}$.

4. Listen to the signal `xsampled` and observe the presence of a sonic artifact. Explain a possible cause (and the reason for the perceived difference).

5. To overcome the issue described in the previous step, propose an alternative scheme to obtain a new sampled signal `xsampled_new` at the sampling frequency $F_{s1} = F_s/25\,\text{Hz}$ free from sonic artifacts. A low-pass filter will be useful for this purpose. To this end, compute the Nyquist frequency for the sampled signal, and then use the following function to design the filter:

```matlab
% Design a low-pass filter
% f_stop = stopband frequency of the filter [Hz]
% Fs = sample rate of the signal to be filtered [Hz]

function LPF = LPF_design(f_stop, Fs)
    % Passband frequency
    f_pass = f_stop - 100;

    % Tolerances in passband (deltap) and stopband (deltas)
    deltap = 0.01;
    deltas = 0.001;

    % Tolerances expressed in dB: [A]_dB = 20*log10(A)
    Ap = 20*log10(1 + deltap) - 20*log10(1 - deltap);  % (
        approximate)
    Ast = -20*log10(deltas);

    % Filter implementation
    SPEC = 'Fp,Fst,Ap,Ast';
    LP_spec = fdesign.lowpass(SPEC, f_pass, f_stop, Ap, Ast,
        Fs);
    LPF = design(LP_spec, 'equiripple');
    % Plot of the frequency response
    % The red dotted line represents the tolerance in the
      passband,
    % the transition band and the stopband while the blue
      line represents the
    % frequency response of the designed filter
    fvtool(LPF);
end
```

Listing 2: Design of a Low-Pass Filter

where f_{stop} represents the minimum frequency of the stopband, that is, the frequency beyond which the filter strongly attenuates the spectral components of a signal, and F_s represents the sampling frequency of the signal to be filtered. Both function arguments are scalars expressed in Hz. Justify the choice of the value f_{stop}.
To filter the signal `x_new`, sampled at frequency F_s, using the filter

```
h = LPF_design(fstop, Fs)
```

execute the command

```
x_filt = filter(h, x_new).
```

6. Resample the signal `x_filt` to obtain a new sampled signal `xsampled_new` at the sampling frequency $F_{s1} = F_s/25\,\text{Hz}$.

7. Listen to the signal `xsampled_new`. Are there sonic artifacts after the sampling operation? Is the new sampled signal comparable to the original signal?[1]

8. Produce two plots showing:

   - In the first plot, the magnitude of the Fourier transform of the signals `x_new` and `x_filt`.
   - In the second plot, the magnitude of the Fourier transform of the signals `xsampled` and `xsampled_new`,

   considering that these signals are defined starting from time $t = 0$.
   Briefly comment on the results.

[1]The sonic artifact introduced by "aggressive" downsampling may be subtle and difficult to perceive during playback. We provide an additional audio track named `'audio_jk.mat'`, which may offer clearer evidence of the artifact effects. As an optional task, you could try solving Exercise 2 once more using this audio track.

Appendix

Playing an audio file in MATLAB

Consider the vector sound containing an audio signal sampled at frequency F_s. To play back the audio signal contained in the vector sound, execute the following commands:

```
listening_time = 26;
player = audioplayer(sound, Fs);
play(player);
pause(listening_time);
stop(player);
```

NOTE:

- The function audioplayer creates an audioplayer object for sound playback given an audio signal and its sampling frequency.
- Given an audioplayer object, the functions play and stop start and stop the playback of the audio signal contained in the object, respectively.
- The command pause(listening_time) temporarily suspends any MATLAB activity and resumes execution after e.g. 26 seconds. This time corresponds to the length of the audio tracks provided in this assignment. To reduce the suspension time (i.e., the listening time of an audio track), modify the argument listening_time in the command pause(listening_time) to a non-negative number.
- Using semilogy in place of plot may offer a clearer view of the magnitude of the different Fourier transforms.