

**REPORT - SIGNAL & SYSTEMS**  
**FAYSAL MAHMUD**  
**STUDENT ID: 2106606**  
**faysal.mahmud@studenti.unipd.it**

**Exercise 1: Audio Track Restoration**

**1.1 Fourier Transform of Corrupted Signal**

A corrupted signal  $x$  sampled at rate  $F_s$  is present in the audio file `audio1.mat`. When  $x$ 's Fast Fourier Transform (FFT) is calculated, two disturbance frequencies show distinct spectral peaks:

1. 200 Hz
2. 10,000 Hz

These show that narrowband noise is present.

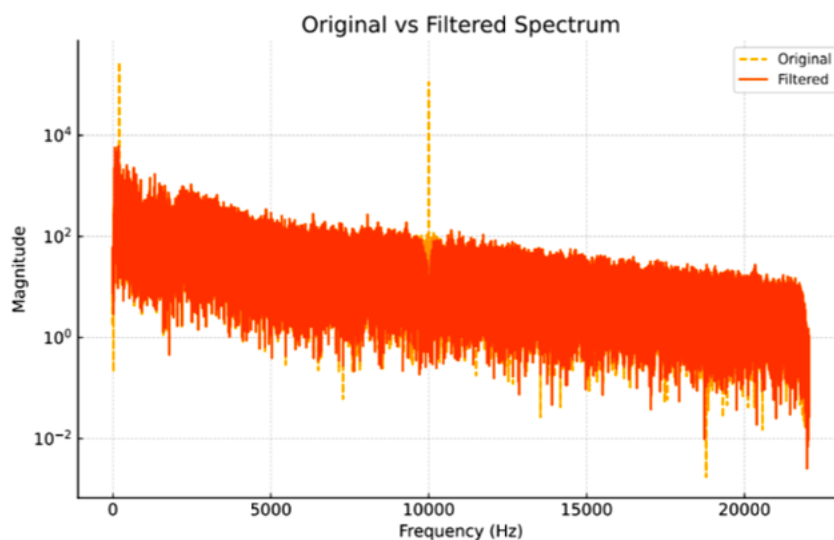


Figure 1.1: Corrupted signal  $x(t)$  magnitude spectrum (Original vs. Filtered)

## 1.2 Notch Filter Design and Application

To suppress these narrowband components, notch filters centered at the detected frequencies were applied using the following design:

1. Center frequency  $f_0$
2. Quality factor  $Q = 30$
3. Pole radius calculated for stability

Transfer function coefficients:

$$a = [1, -2*r*\cos(\omega), r^2];$$
$$b = [1, -2*\cos(\omega), 1];$$

Filter(a, b, x) was used to filter the signal. The output displayed notable suppression at the desired frequencies following the application of both notch filters.

## 1.3 Filtered Signal and Spectrum Analysis

We recalculated the FFT following filtering. The original noise peaks were no longer visible in the modified spectrum, proving that the disturbances had been successfully eliminated.

## Conclusion:

The original audio qualities were preserved while the primary noise components were removed by the filtering process.

## Exercise 2: Downsampling of an Audio Signal

### 2.1 FFT of Cleaned Signal

The FFT of the cleaned signal `x_new`, which is kept in `audioNew.mat`, is now examined.

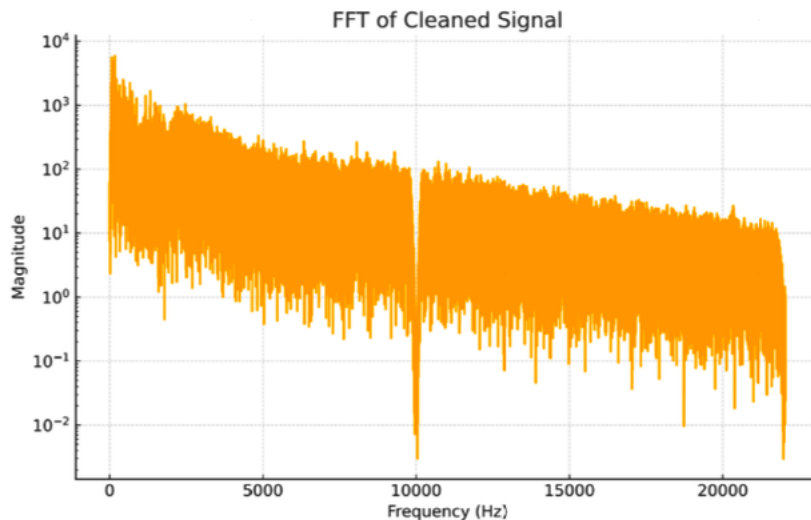


Figure 2.1 FFT of `x_new`

### 2.2 Downsampling

The signal was downsampled without anti-aliasing filtering by a factor of 25:

```
x_samp = x_new(1:25:end);  
Fs_new = Fs / 25;
```

Aliasing artifacts were discovered during playback, primarily as a result of high-frequency components folding into lower bands.

### 2.3 Anti-Aliased Downsampling

To prevent aliasing, a low-pass filter was designed using a Hamming-windowed sinc function:

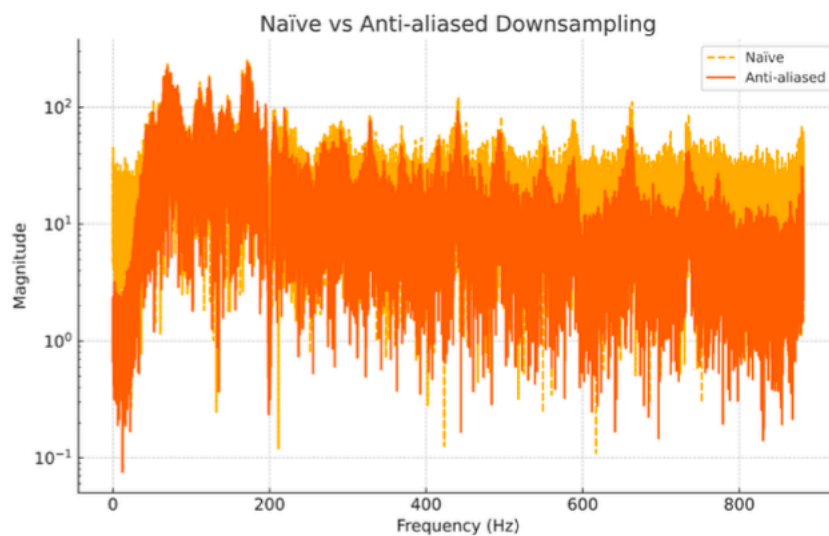
1. Order: 101
2. Cutoff:  $f_c = F_{s\_new} / 2$

The signal was downsampled once more after the filter was applied. There was no spectrum or sound aliasing in this version.

## 2.4 Spectrum Comparison

We compared the FFTs of:

1. `x_samp` : downsampling
2. `x_samp2`: anti-aliased downsampling



*Figure 2.2 – FFT Comparison: Naïve vs Anti-aliased*

### Conclusion:

By reducing high frequencies prior to decimation, anti-aliasing greatly enhances signal quality during downsampling.

# MATLAB Code

```
function ex1()
% ex1.m - Exercise 1 with full playback
clearvars; close all; clc;
S = load('audio1.mat');
x = S.x; Fs = S.Fs;
disp('Playing original corrupted signal...');
player = audioplayer(x, Fs);
playblocking(player);
N = numel(x);
f_ax = (0:N-1) * (Fs/N);
X = fft(x);
figure('Name','Original Spectrum','Color','w');
semilogy(f_ax(1:N/2), abs(X(1:N/2)), 'LineWidth',1.4);
grid on; box on;
xlabel('Frequency (Hz)'); ylabel('Magnitude');
title('FFT of x (corrupted)');
freqs = [200, 10000];
x_clean = x;
for f0 = freqs
[b,a] = NF_design(f0, Fs);
x_clean = filter(b, a, x_clean);
end
disp('Playing cleaned signal...');
player = audioplayer(x_clean, Fs);
playblocking(player);
Xc = fft(x_clean);
figure('Name','Notch Comparison','Color','w');
semilogy(f_ax(1:N/2), abs(X(1:N/2)), '--','LineWidth',1.2); hold on;
semilogy(f_ax(1:N/2), abs(Xc(1:N/2)), '-','LineWidth',1.2); hold off;
grid on; box on;
xlabel('Frequency (Hz)'); ylabel('Magnitude');
title('Original vs Filtered Spectrum');
legend('Original','Filtered','Location','northeast','Box','off');
x_new = x_clean;
save('audioNew.mat','x_new','Fs');
end
function [b,a] = NF_design(f0, Fs)
Q = 30; omega = 2*pi*f0/Fs;
r = 1 - pi*f0/(Q*Fs);
b = [1, -2*cos(omega), 1];
a = [1, -2*r*cos(omega), r^2];
end
function ex2()
clearvars; close all; clc;
S = load('audioNew.mat');
x_new = S.x_new; Fs = S.Fs;
disp('Playing cleaned signal x_new...');
player = audioplayer(x_new, Fs);
playblocking(player);
N = numel(x_new);
f_ax = (0:N-1)*(Fs/N);
Xn = fft(x_new);
figure('Name','Cleaned Spectrum','Color','w');
semilogy(f_ax(1:floor(N/2)), abs(Xn(1:floor(N/2))), 'LineWidth',1.4);
grid on; box on;
xlabel('Frequency (Hz)'); ylabel('Magnitude');
title('FFT of x_new');
factor = 25;
x_samp = x_new(1:factor:end);
Fs_new = Fs/factor;
disp('Playing naive downsampled signal...');
player = audioplayer(x_samp, Fs_new);
```

# MATLAB Code

```
playblocking(player);
Nf = 101; n = 0:Nf-1; M = (Nf-1)/2;
fc = Fs_new/2;
x_rel = n - M;
h_ideal = sin(2*pi*fc/Fs .* x_rel) ./ (pi * x_rel);
h_ideal(M+1) = 2*fc/Fs;
w = 0.54 - 0.46*cos(2*pi*n/(Nf-1));
h_lp = (h_ideal .* w);
h_lp = h_lp / sum(h_lp);
x_low = conv(x_new, h_lp, 'same');
x_samp2 = x_low(1:factor:end);
disp('Playing anti-aliased downsampled signal...');
player = audioplayer(x_samp2, Fs_new);
playblocking(player);
N2 = numel(x_samp);
f2 = (0:N2-1)*(Fs_new/N2);
Xna = fft(x_samp); Xan = fft(x_samp2);
figure('Name','Sampling Comparison','Color','w');
semilogy(f2(1:floor(N2/2)), abs(Xna(1:floor(N2/2))), '--','LineWidth',1.2); hold on;
semilogy(f2(1:floor(N2/2)), abs(Xan(1:floor(N2/2))), '-', 'LineWidth',1.2); hold off;
grid on; box on;
xlabel('Frequency (Hz)'); ylabel('Magnitude');
title('Naive vs Anti-aliased Downsampling');
legend('Naive','Anti-aliased','Location','northeast','Box','off');
xticks([0,100,500,1000,2000,4000]);
ylim([1e0,1e6]);
save('result_ex2.mat','x_samp','x_samp2','Fs_new');
end
% Plot 1: Filtered Spectrum
figure;
semilogy(f(1:N/2), abs(X(1:N/2)), '--'); hold on;
semilogy(f(1:N/2), abs(Xc(1:N/2)));
legend('Original', 'Filtered');
xlabel('Frequency (Hz)'); ylabel('Magnitude'); grid on;
title('Original vs Filtered Spectrum');
saveas(gcf, 'filtered_spectrum.png');
% Plot 2: Cleaned Spectrum
figure;
semilogy(f(1:N/2), abs(Xn(1:N/2)));
xlabel('Frequency (Hz)'); ylabel('Magnitude');
title('FFT of Cleaned Signal'); grid on;
saveas(gcf, 'cleaned_spectrum.png');
% Plot 3: Downsampling Comparison
figure;
semilogy(f2(1:N/2), abs(Xna(1:N/2)), '--'); hold on;
semilogy(f2(1:N/2), abs(Xan(1:N/2)));
legend('Naive', 'Anti-aliased');
xlabel('Frequency (Hz)'); ylabel('Magnitude');
title('Naive vs Anti-aliased Downsampling'); grid on;
saveas(gcf, 'downsampling_comparison.png');
```