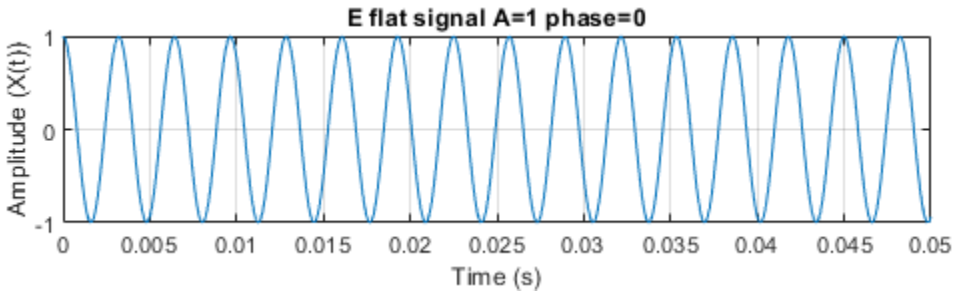
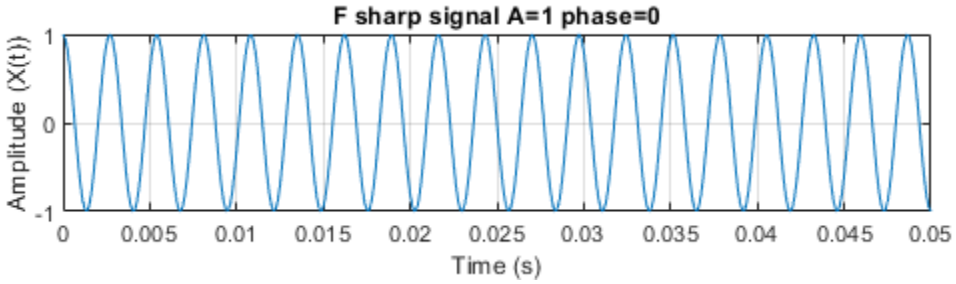
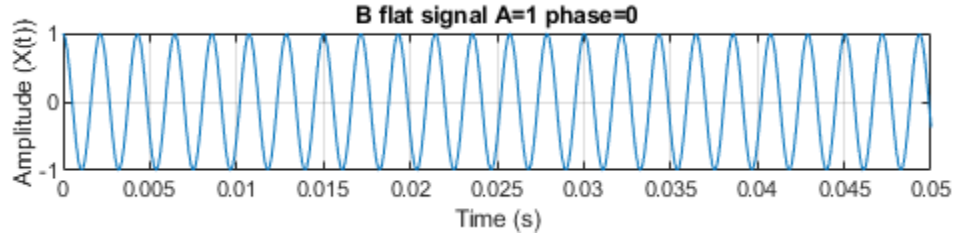
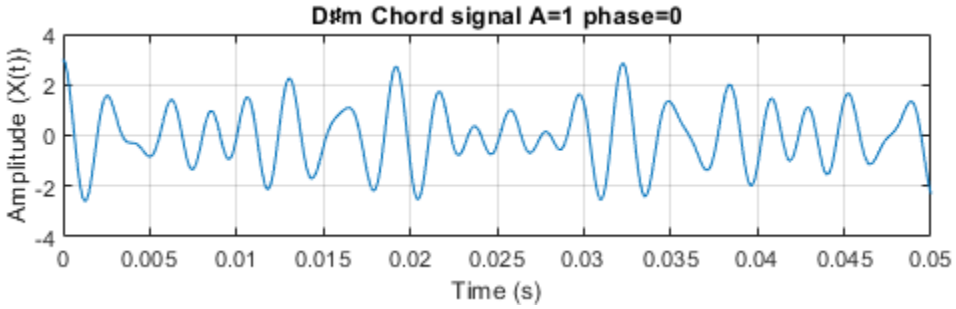
EEE391 - MATLAB Assignment 1: Musical Signal Analysis - Görkem Kadir Solun 22003214

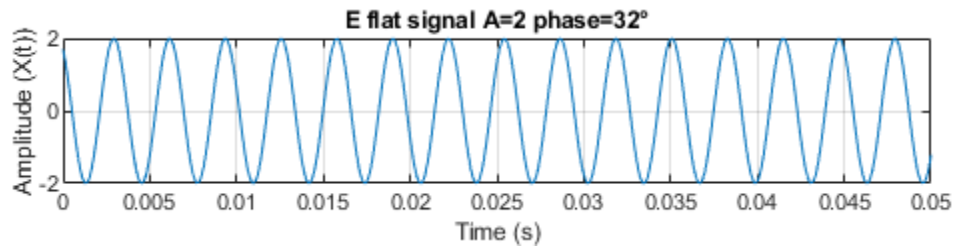
2 Signal Generation

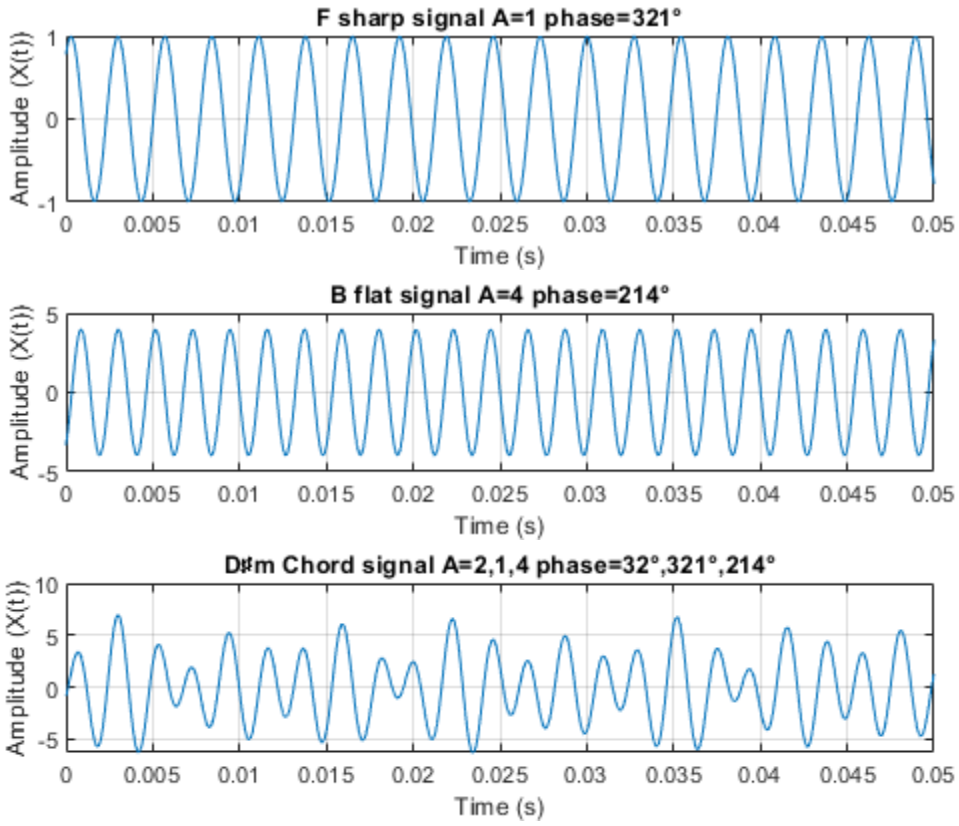












Time and Signal Amplitude: Each plot shows a 0.05-second segment of signals, with time on the x-axis and signal amplitude labeled as x(t) on the y-axis.

Frequencies of Notes: Three notes make up the chord, each with a specific frequency:

E♭ at 311.13 Hz,

F♯ at 369.99 Hz,

B♭ at 466.16 Hz. These frequencies define the pitch of each note in the chord.

The phase, or timing of each note's waveform, plays a key role in the chord's sound. While changing the phase of a single note alone doesn't change its individual sound, the phase relationships between multiple notes affect how the notes combine, shaping the auditory texture of the chord.

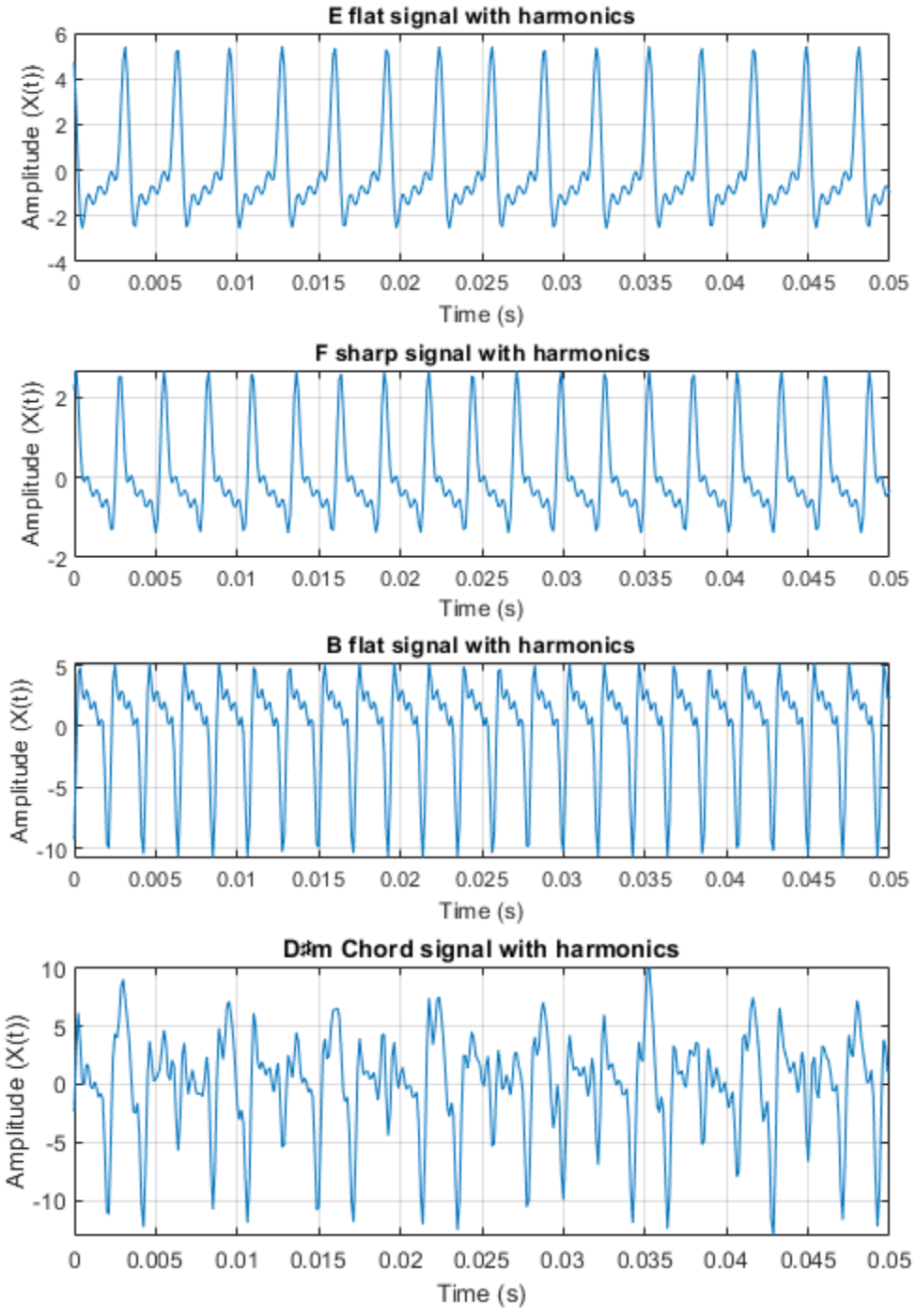
The plots show how the interaction of amplitude and phase leads to interference patterns. Constructive interference (where waves align) amplifies the sound, while destructive interference (where waves cancel) diminishes it. When these signals are combined, they produce a D♯m chord.

The plots vary in loudness depending on the amplitude of each note. Higher amplitudes make certain notes more prominent, affecting the overall tonal balance of the chord.

Adjusting phase settings changes how the notes interfere, resulting in different harmonic textures. These interference patterns affect the chord’s final sound, as constructive and destructive interferences between signals adjust the balance and blend of the chord.

These plots demonstrate how the interplay of frequency, amplitude, and phase among individual notes results in the distinct sound of the D♯m chord, with loudness and harmonic texture shifting based on these factors.

3 Adding Harmonics



Each note (E♭, F♯, and B♭) has its original frequency (fundamental) plus additional harmonics — specifically, the second, third, and fourth harmonics. These harmonics are higher-frequency

versions of the original note, each with decreasing intensity as they go higher. This decrease means each harmonic adds less loudness than the one before it.

Adding harmonics raises the signal's energy and amplitude, increasing the overall loudness of each note within the chord.

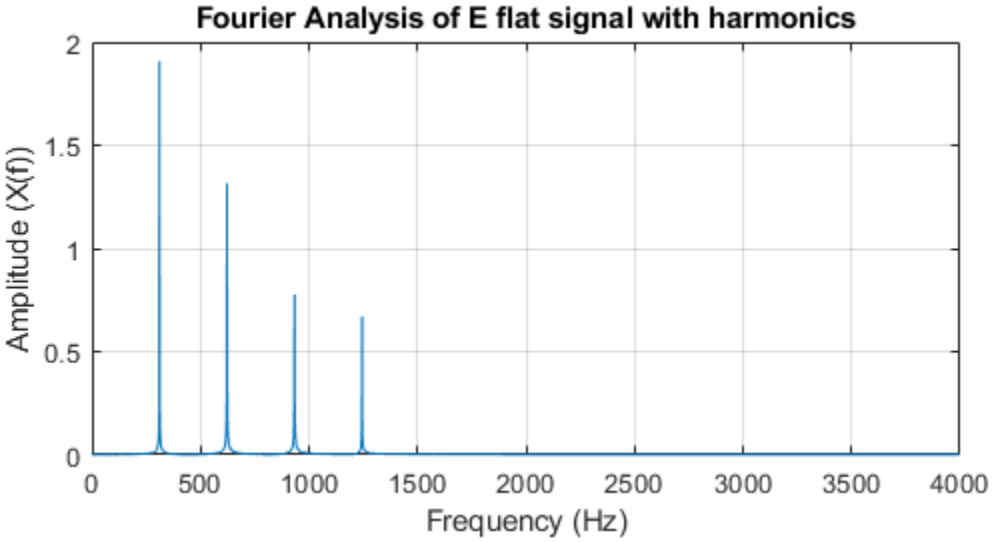
Including these harmonics transforms the sound of the D♯m chord from a simple tone into a more complex and textured one, giving it a richer, fuller quality.

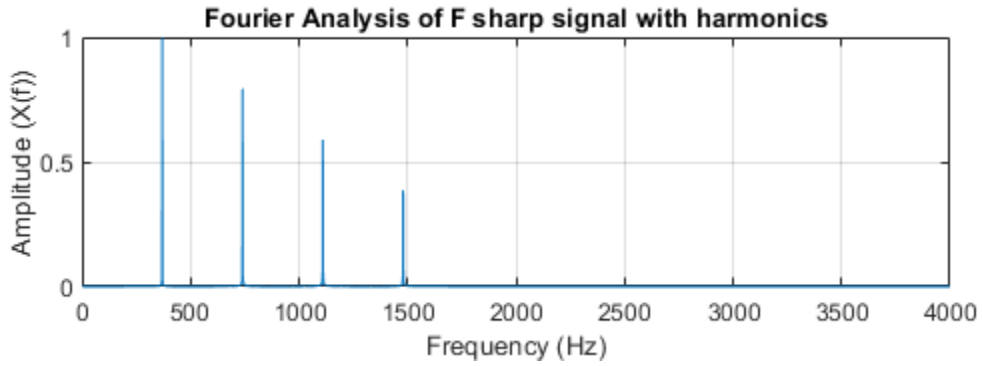
By adding harmonics, the range of frequencies (spectrum) in each note expands. This increases the highest frequency present in the sound, contributing to a broader harmonic range and a more intricate sound profile.

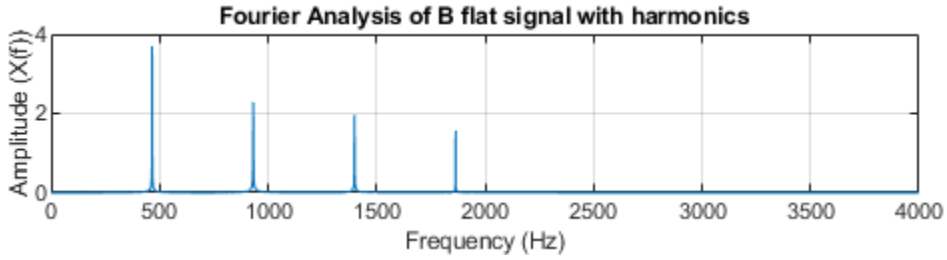
Despite the added complexity, the chord still retains its original timbre — the unique quality that identifies it as a D♯m chord. The harmonics deepen the sound without changing its fundamental identity.

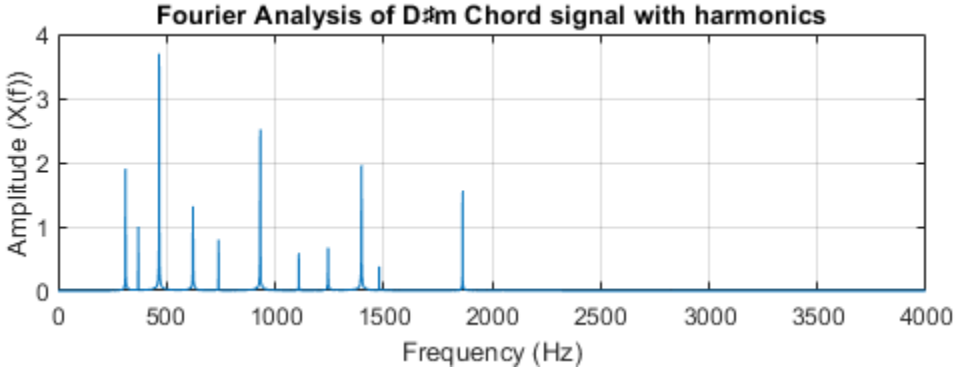
These plots describe how harmonics enhance the D♯m chord by adding depth, fullness, and loudness without altering its essential character.

4 Fourier Analysis









The y-axis of the plot represents magnitude (or amplitude), and the x-axis represents frequency. These axes help show the strength of each frequency component in the sound.

The Fourier transform is applied to the signal, breaking it into frequency components. For each note, the transform reveals peaks at the fundamental frequency (the note's base frequency) and at integer multiples of this frequency, which are the harmonics.

Each note's plot displays additional peaks at its second, third, and fourth harmonics. These peaks appear at frequencies that are integer multiples of the original frequency, reflecting the harmonics added to enrich the note.

The presence of these harmonic peaks in the frequency range broadens the spectrum and creates a richer, more layered sound for the chord, giving it added complexity.

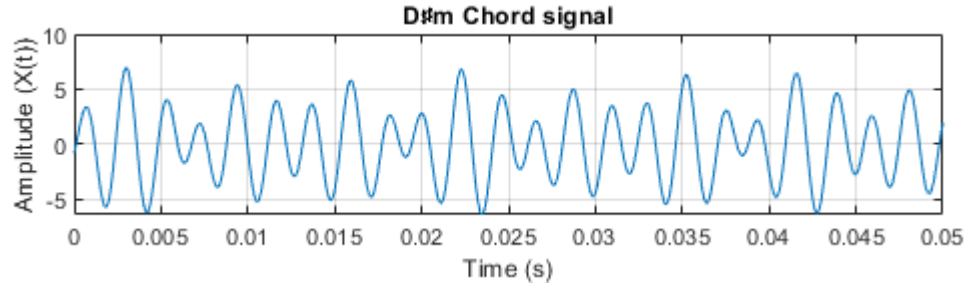
The Fourier transform plot visually maps the sound's structure in magnitude and frequency. Each peak's height (magnitude) represents the strength of that harmonic frequency within the note.

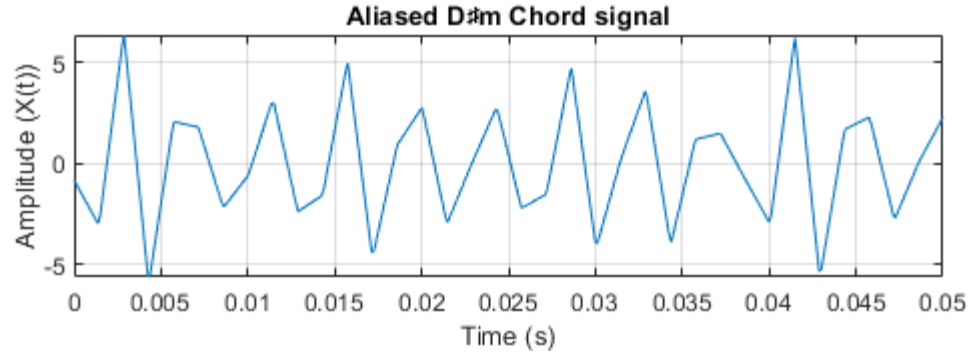
As we go to higher harmonics, their contribution to the sound diminishes, as shown by lower peaks at these frequencies. This reflects how each successive harmonic has a smaller impact on the sound's loudness but still influences the texture and timbre.

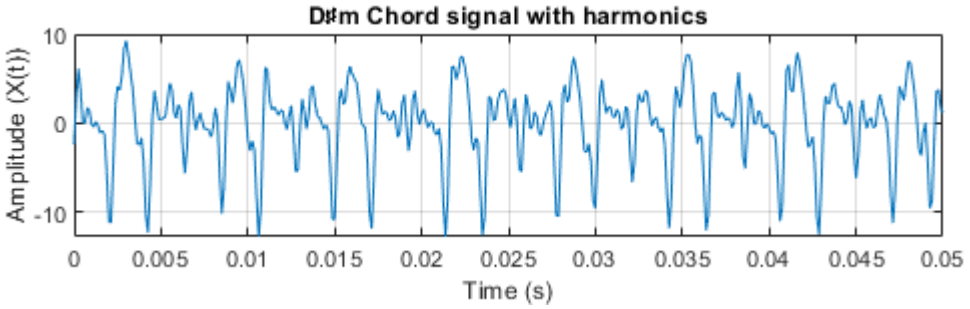
These harmonics add a distinctive quality to the chord, enriching its timbre and making the sound more complex and unique.

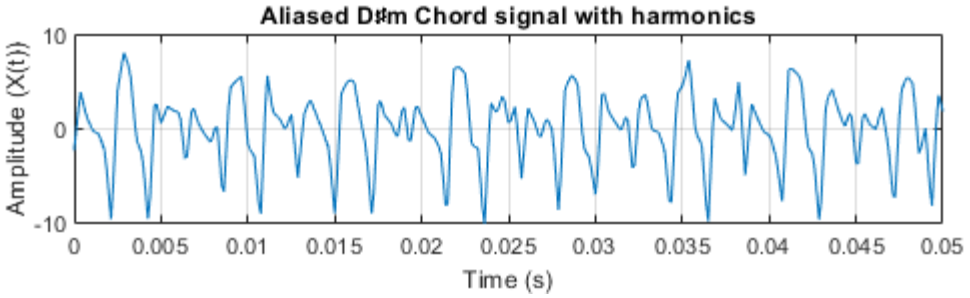
These plots describe how the Fourier transform reveals the frequency structure of the chord, showing how harmonics add complexity and depth to the sound by creating additional frequency peaks that enhance the timbre without overwhelming the fundamental tones.

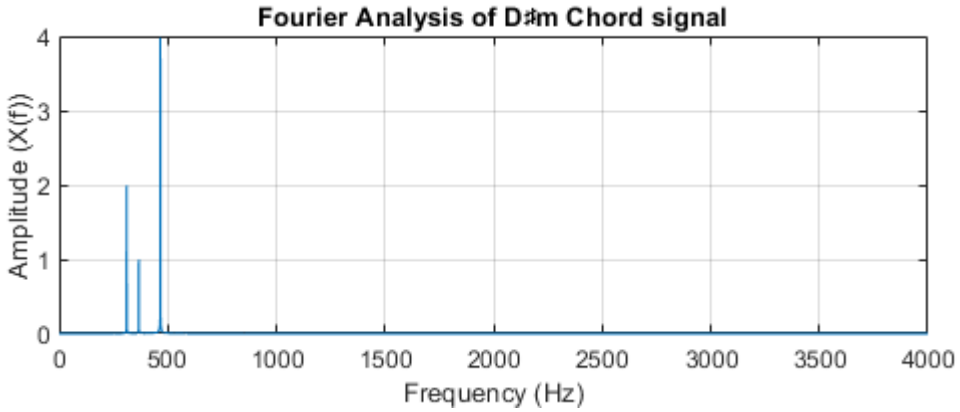
5 Aliasing and Reconstruction

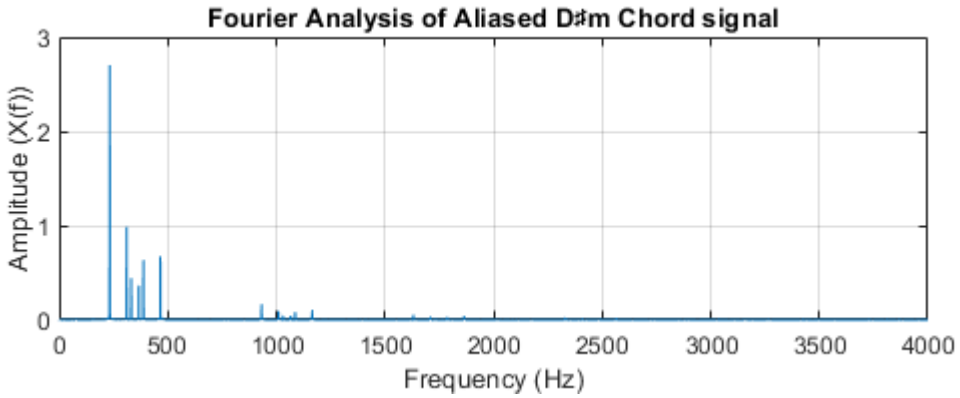


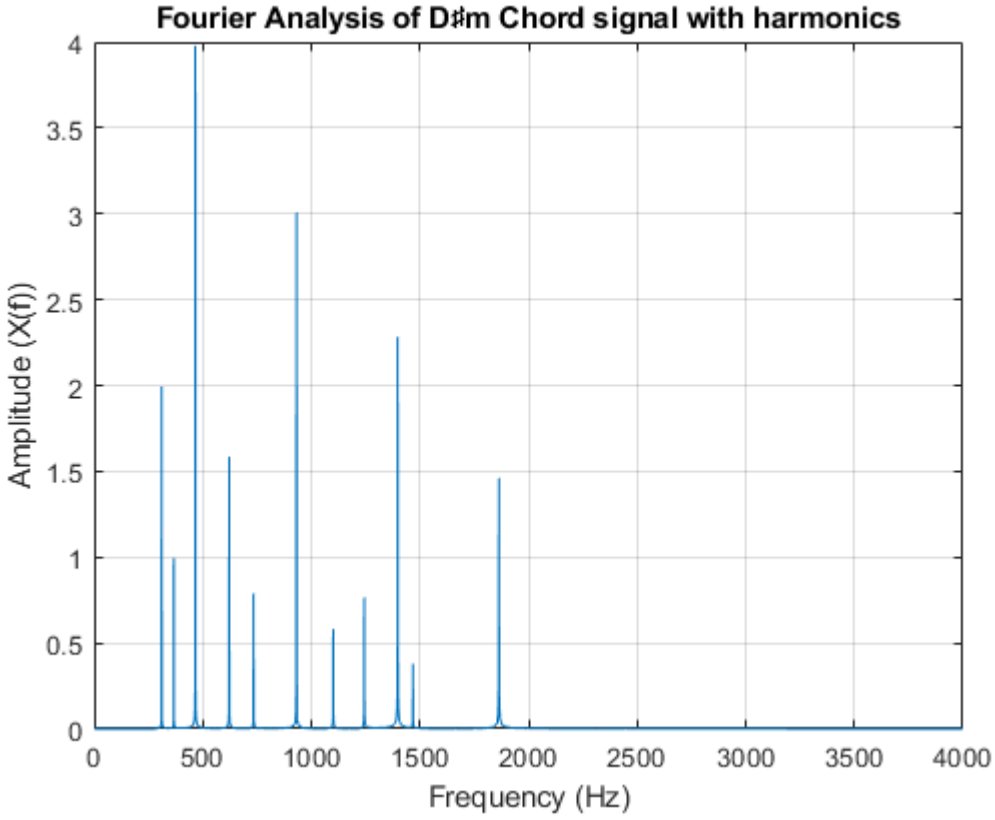


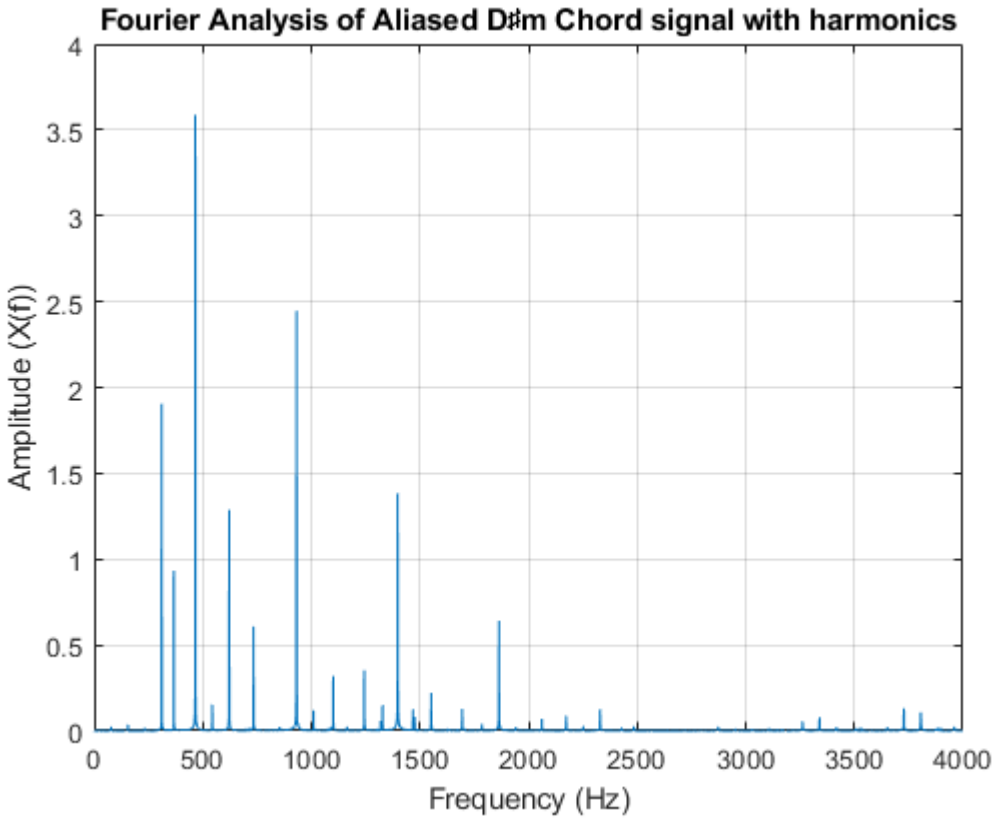





Fourier Analysis of D♯m Chord signal
Amplitude (X(f))
Frequency (Hz)
0
1
2
3
4
0
500
1000
1500
2000
2500
3000
3500
4000


Fourier Analysis of Aliased D♯m Chord signal
Amplitude (X(f))
Frequency (Hz)
0
1
2
3
0
500
1000
1500
2000
2500
3000
3500
4000


Fourier Analysis of D♯m Chord signal with harmonics
Amplitude (X(f))
Frequency (Hz)
0
0.5
1
1.5
2
2.5
3
3.5
4
0
500
1000
1500
2000
2500
3000
3500
4000


Fourier Analysis of Aliased D♯m Chord signal with harmonics
Amplitude (X(f))
Frequency (Hz)
0
0.5
1
1.5
2
2.5
3
3.5
4
0
500
1000
1500
2000
2500
3000
3500
4000

The y-axis represents the magnitude (or strength) of the frequency components, and the x-axis represents the frequency itself.

When the original chord is resampled using the interp1 method (a type of interpolation), aliasing occurs. Aliasing is when higher-frequency components in a signal appear at lower frequencies after resampling, especially if the sampling rate is too low to capture the higher frequencies accurately. This folding back effect places parts of the original high-frequency signal into the lower frequency range.

Due to aliasing, the resampled chord contains unexpected low-frequency components that make it sound noisy and somewhat like the original but with a distorted quality.

When harmonics are added to the resampled (aliased) chord, the aliasing effect becomes even more pronounced. The additional harmonics introduce more high-frequency components, which also get folded into unintended lower frequencies due to aliasing. This increases the sound's distortion and complexity.

Because of these aliasing effects, the final sound is more distorted and noisier than the original chord with harmonics, making it harsher and less clear.

These plots describe how resampling a chord at a lower rate introduces aliasing, creating low-frequency artifacts and noise. Adding harmonics to the aliased chord increases the distortion further, resulting in a complex and noisy sound very different from the original harmonized chord.

Code

```
% Görkem Kadir Solun - 22003214

function plotSignal(signal, t, titleText)
    % Plot the signal
    figure;
    plot(t, signal);
    title(titleText);
    xlabel('Time (s)');
    ylabel('Amplitude (X(t))');
    xlim([0, 0.05]);
    grid on;
    pause(1);
end

function soundSignals(signal1, signal2, signal3, fs)
    % Play the signals
    sound(signal1, fs);
    pause(1);
    sound(signal2, fs);
    pause(1);
    sound(signal3, fs);
```

```
    pause(1);
end

function fourierAnalysis(signal, fs, titleText)
    % Perform Fourier analysis on the signal
    N = length(signal);
    N_2 = floor(N / 2);
    f = fs * (0:N_2) / N;
    Y = fft(signal);
    P2 = abs(Y / N);
    P1 = P2(1:N_2 + 1);
    P1(2:end - 1) = 2 * P1(2:end - 1);

    figure;
    plot(f, P1);
    title(titleText);
    xlabel('Frequency (Hz)');
    ylabel('Amplitude (X(f))');
    xlim([0, fs / 2]);
    grid on;
    pause(1);
end

%
% Part 2 Signal Generation
%

default_fs = 8000;
t = 0:1 / default_fs:1;
E_flat_frequency = 311; % 311.13 Hz
F_sharp_frequency = 367; % 369.99 Hz
B_flat_frequency = 466; % 466.16 Hz

% First generate signals with amplitude 1 and phase 0
E_flat_signal = cos(2 * pi * E_flat_frequency * t);
F_sharp_signal = cos(2 * pi * F_sharp_frequency * t);
B_flat_signal = cos(2 * pi * B_flat_frequency * t);

% Sound the signals
disp('Playing the signals with amplitude 1 and phase 0');
soundSignals(E_flat_signal, F_sharp_signal, B_flat_signal, default_fs);

% Create a chord and sound it
disp('Playing the chord with amplitude 1 and phase 0');
chord = E_flat_signal + F_sharp_signal + B_flat_signal;
sound(chord, default_fs);
```

```
pause(1);

% Plot the signals
plotSignal(E_flat_signal, t, 'E flat signal A=1 phase=0');
plotSignal(F_sharp_signal, t, 'F sharp signal A=1 phase=0');
plotSignal(B_flat_signal, t, 'B flat signal A=1 phase=0');
plotSignal(chord, t, 'D♯m Chord signal A=1 phase=0');

% Now amplify the signals according to my ID's last digits 22003214
E_flat_amplitude = 2;
F_sharp_amplitude = 1;
B_flat_amplitude = 4;

% And phasify the signals according to my ID's last digits 22003214
E_flat_phase = deg2rad(032);
F_sharp_phase = deg2rad(321);
B_flat_phase = deg2rad(214);

% New signals
E_flat_signal = E_flat_amplitude * cos(2 * pi * E_flat_frequency * t + E_flat_phase);
F_sharp_signal = F_sharp_amplitude * cos(2 * pi * F_sharp_frequency * t +
F_sharp_phase);
B_flat_signal = B_flat_amplitude * cos(2 * pi * B_flat_frequency * t + B_flat_phase);

% Sound the signals
disp('Playing the signals with amplitude and phase modifications');
soundSignals(E_flat_signal, F_sharp_signal, B_flat_signal, default_fs);
disp('Playing the chord with amplitude and phase modifications');
chord = E_flat_signal + F_sharp_signal + B_flat_signal;
sound(chord, default_fs);
pause(1);

% Plot the signals
plotSignal(E_flat_signal, t, 'E flat signal A=2 phase=32°');
plotSignal(F_sharp_signal, t, 'F sharp signal A=1 phase=321°');
plotSignal(B_flat_signal, t, 'B flat signal A=4 phase=214°');
plotSignal(chord, t, 'D♯m Chord signal A=2,1,4 phase=32°,321°,214°');

%
% Part 3 Adding Harmonics
%
amplitude_modifiers = [1, 0.8, 0.6, 0.4];
t = 0:1 / default_fs:1;
E_flat_harmonics = zeros(1, length(t));
F_sharp_harmonics = zeros(1, length(t));
B_flat_harmonics = zeros(1, length(t));
```

```
% Add harmonics to the signals
for i = 1:length(amplitude_modifiers)
    E_flat_harmonics = E_flat_harmonics + E_flat_amplitude * amplitude_modifiers(i) *
cos(2 * pi * i * E_flat_frequency * t + E_flat_phase);
    F_sharp_harmonics = F_sharp_harmonics + F_sharp_amplitude *
amplitude_modifiers(i) * cos(2 * pi * i * F_sharp_frequency * t + F_sharp_phase);
    B_flat_harmonics = B_flat_harmonics + B_flat_amplitude * amplitude_modifiers(i) *
cos(2 * pi * i * B_flat_frequency * t + B_flat_phase);
end

chord_harmonics = E_flat_harmonics + F_sharp_harmonics + B_flat_harmonics;

% Sound the signals
disp('Playing the signals with harmonics');
soundSignals(E_flat_harmonics, F_sharp_harmonics, B_flat_harmonics, default_fs);
disp('Playing the chord with harmonics');
sound(chord_harmonics, default_fs);
pause(1);

% Plot the signals
plotSignal(E_flat_harmonics, t, 'E flat signal with harmonics');
plotSignal(F_sharp_harmonics, t, 'F sharp signal with harmonics');
plotSignal(B_flat_harmonics, t, 'B flat signal with harmonics');
plotSignal(chord_harmonics, t, 'D♯m Chord signal with harmonics');

%
% Part 4 Fourier Analysis
%
% Perform Fourier analysis on the signals with harmonics
disp('Performing Fourier analysis on the signals with harmonics');
fourierAnalysis(E_flat_harmonics, default_fs, 'Fourier Analysis of E flat signal with
harmonics');
fourierAnalysis(F_sharp_harmonics, default_fs, 'Fourier Analysis of F sharp signal
with harmonics');
fourierAnalysis(B_flat_harmonics, default_fs, 'Fourier Analysis of B flat signal with
harmonics');
fourierAnalysis(chord_harmonics, default_fs, 'Fourier Analysis of D♯m Chord signal
with harmonics');

%
% Part 5 Aliasing and reconstruction
%
% Intentionally undersample the signals below the Nyquist frequency
% for the highest note in the chord (B flat) and try to reconstruct
% original chord and the second, third and fourth harmonics
```

```
aliased_fs = (B_flat_frequency * 3) / 2;
aliased_t = 0:1 / aliased_fs:1;

% Generate the aliased original chord
aliased_chord = interp1(t, chord, aliased_t, 'linear');
aliased_chord_resampled = interp1(aliased_t, aliased_chord, t, 'linear');

% Generate the aliased harmonics
aliased_harmonic_fs = (B_flat_frequency * 4 * 3) / 2;
aliased_harmonic_t = 0:1 / aliased_harmonic_fs:1;

aliased_chord_harmonics = interp1(t, chord_harmonics, aliased_harmonic_t, 'linear');
aliased_chord_harmonics_resampled = interp1(aliased_harmonic_t,
aliased_chord_harmonics, t, 'linear');

disp('Play the original chord');
sound(chord, default_fs);
pause(1);
disp('Playing the aliased original chord');
sound(aliased_chord_resampled, default_fs);
pause(1);

disp('Playing the chord with harmonics');
sound(chord_harmonics, default_fs);
pause(1);
disp('Playing the aliased chord with harmonics');
sound(aliased_chord_harmonics_resampled, default_fs);
pause(1);

% Plot the signals
disp('Plotting the signals');
plotSignal(chord, t, 'D♯m Chord signal');
plotSignal(aliased_chord_resampled, t, 'Aliased D♯m Chord signal');
plotSignal(chord_harmonics, t, 'D♯m Chord signal with harmonics');
plotSignal(aliased_chord_harmonics_resampled, t, 'Aliased D♯m Chord signal with
harmonics');

% Perform Fourier analysis on the aliased signals
disp('Performing Fourier analysis on the aliased signals');
fourierAnalysis(chord, default_fs, 'Fourier Analysis of D♯m Chord signal');
fourierAnalysis(aliased_chord_resampled, default_fs, 'Fourier Analysis of Aliased D♯m
Chord signal');
fourierAnalysis(chord_harmonics, default_fs, 'Fourier Analysis of D♯m Chord signal
with harmonics');
fourierAnalysis(aliased_chord_harmonics_resampled, default_fs, 'Fourier Analysis of
Aliased D♯m Chord signal with harmonics');
```