Table of Contents


```
close all
```

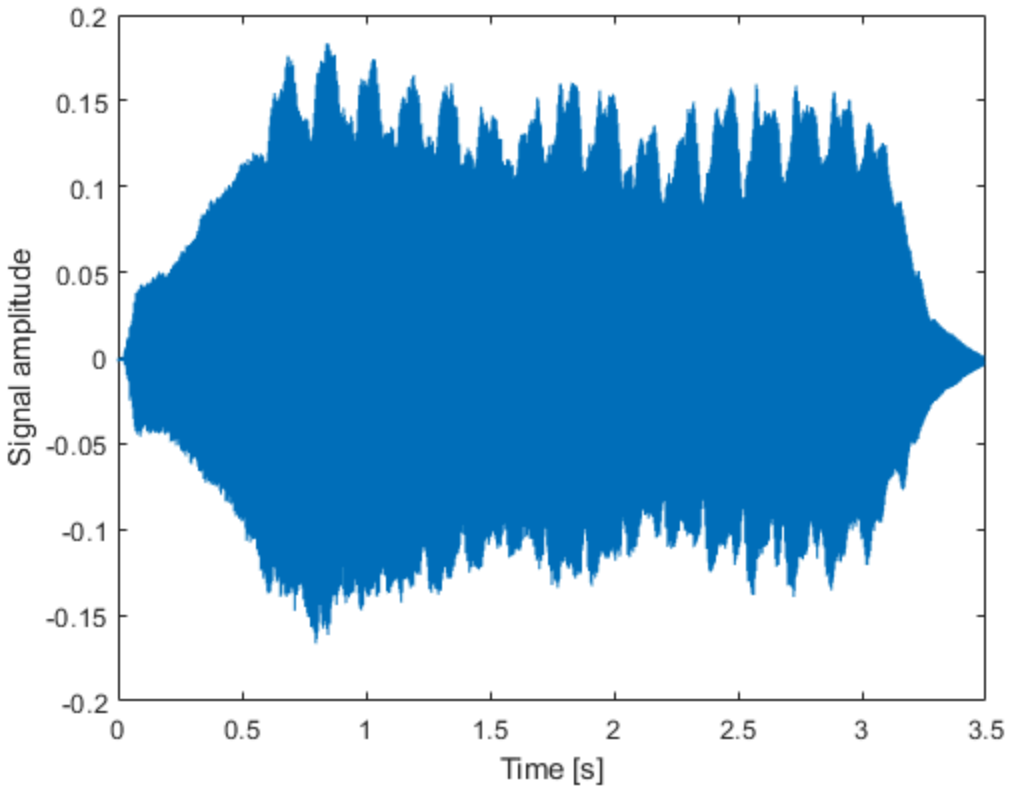
Section 2.2 Time-Domain Analysis

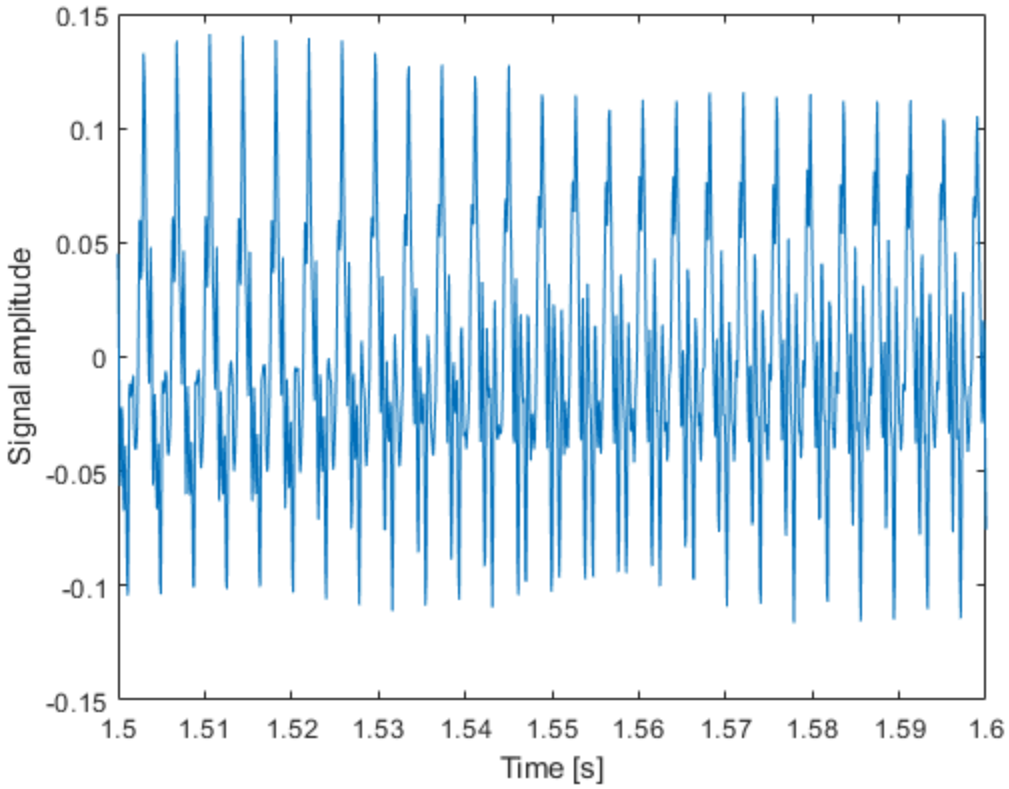
Read the contents of the audio file

```
waveFilename = 'violin-C4.wav';
[instrumentSound, fs] = audioread(waveFilename);

% Play back the recording with automatic scaling
soundsc(instrumentSound, fs);

%a)
    % Plot the waveform in the time domain
N = length(instrumentSound);
Ts = 1/fs;
Tmax = (N-1)*Ts;
t = 0 : Ts : Tmax;
figure;
plot(t, instrumentSound);
xlabel('Time [s]');
ylabel('Signal amplitude');

%b)
    % Mean = -0.0016 Hz
mean(instrumentSound)

%c)
    % i.
figure;
plot(t, instrumentSound);
xlabel('Time [s]');
ylabel('Signal amplitude');
xlim( [1.5 1.6] );

    % ii.
% Compute Fund period
    % Right tip: 1.50295
    % Left tip: 1.599
    % 25 period
fundPeriod = (1.599 - 1.50295)/25; % 1 fund period
fundFrequency = 1/ fundPeriod;
```

```
C4funcFrequency = 261.63; % fund Frequency of C4: 261.63 Hz

Diff_C4toEstimate = C4funcFrequency - fundFrequency;
% The different between 'C4' note with the estimate is 1.3489 Hz, which is
% close enough.

% d) What is the sampling rate of the original recording the

% What is the sampling rate of recording in the file ‘violin-C4.wav’?
    % Audio CD = 44100 Hz
    % Sampling rates = 11025 Hz

% What’s the ratio between these sampling rates?
    % 4-1 ratio compare to 44100 Hz

% -------------- Done with 2.2 ---------------- %
```

```
ans =

   -0.0016
```



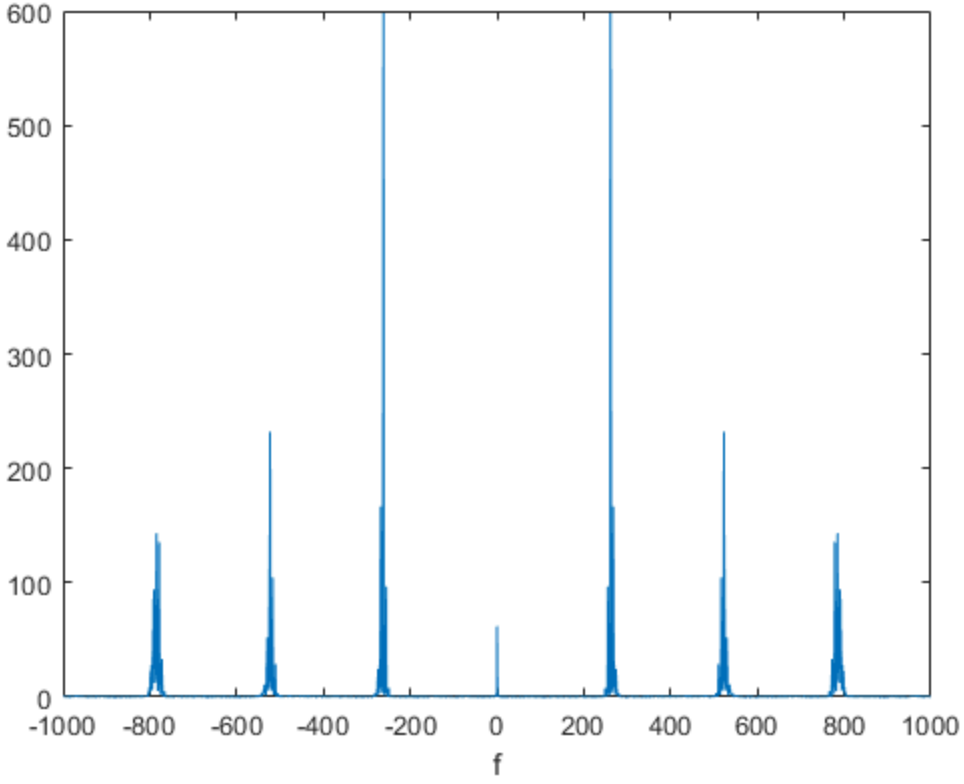


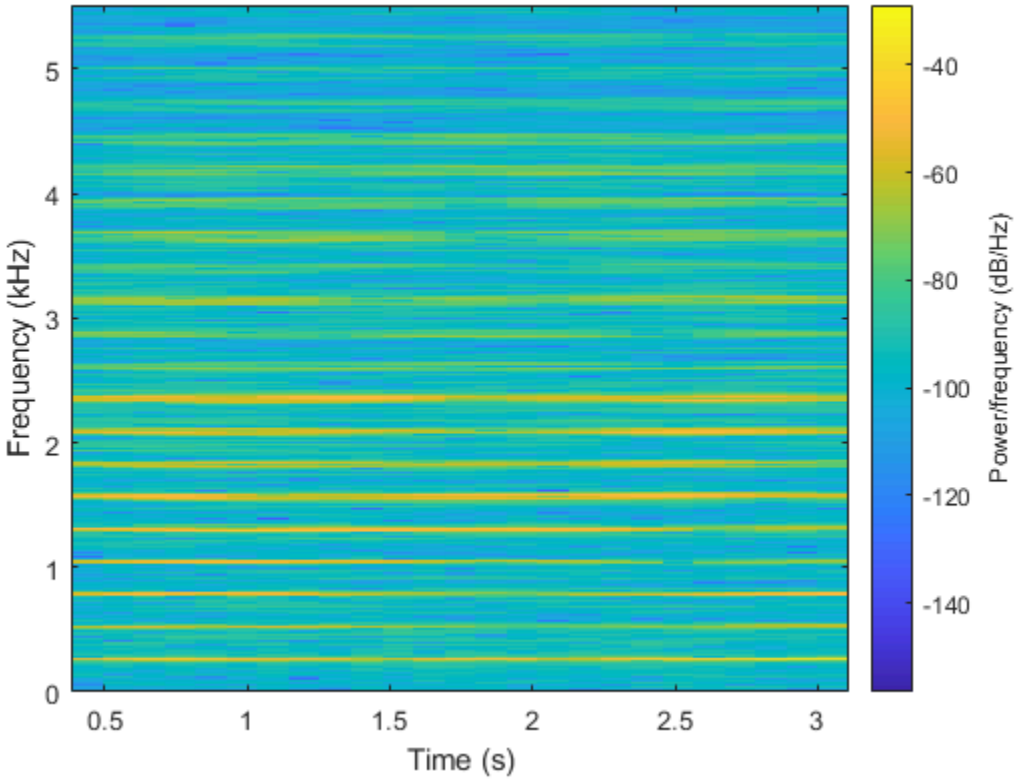
Section 2.3 Frequency and Time-Frequency Analysis

```
% Read the contents of the audio file
waveFilename = 'violin-C4.wav';
[instrumentSound, fs] = audioread(waveFilename);

% Play back the recording with automatic scaling
% soundsc(instrumentSound, fs);

%a)
    % Plot the magnitude of the frequency content using a discrete-time
 version of the Fourier series
fourierSeriesCoeffs = fft(instrumentSound);
N = length(instrumentSound);
freqResolution = fs / N;
ff = (-fs/2) : freqResolution : (fs/2)-freqResolution;
figure;
plot(ff, abs(fftshift(fourierSeriesCoeffs)));
xlabel('f');
xlim( [-1000 1000] );

%b)
    % Explain how you determined the gain, frequency, and phase values,
```

```
    % including any MATLAB code you ve written.

positive_frequencies = ff(ff >= 0);
% Find the maximum magnitude and its index
[max_magnitude, max_index] = max(abs(fftshift(fourierSeriesCoeffs)));
    % This form of max returns both the maximum value (max_val) and its
    % index (max_idx) in the array. If there are multiple occurrences
    % of the maximum value, it returns the index of the first occurrence.

% Determine the gain, frequency, and phase
peak_gain = max_magnitude;
peak_frequency = positive_frequencies(max_index);
peak_phase = angle(fftshift(fourierSeriesCoeffs(max_index)));

% Display the results
fprintf('Gain: %.2f\n', peak_gain);
fprintf('Frequency: %.2f Hz\n', peak_frequency);
fprintf('Phase: %.2f radians\n\n', peak_phase);

%c)
    % Plot the spectrogram
figure;
blockSize = round(N/4);
overlap = round(0.875 * blockSize);
spectrogram(instrumentSound, blockSize, overlap, blockSize, fs, 'yaxis');

% d)
    % From the spectrogram, for what range of time is the principal frequency
 present?
    % My response: Based on the plot, we concluded that it last the entire
 time frame

% ---------------- Done with 2.3 ---------------- %
```

Gain: 598.21
Frequency: 5251.05 Hz
Phase: -3.12 radians


600
500
400
300
200
100
0
-1000
-800
-600
-400
-200
0
200
400
600
800
1000
f


Frequency (kHz)
0
1
2
3
4
5
0.5
1
1.5
2
2.5
3
Time (s)
-40
-60
-80
-100
-120
-140
Power/frequency (dB/Hz)

Section 3.0 Synthesizing the Sound

a) Identify the indices of the largest peaks (adjust as needed)

```
indices_two_largest = [914, 37587]; % Adjust these indices
% fk2 = 522.9 Hz -> 1826 + 1 = 1827
% fk-2 = -522.9 Hz -> 36674 = 1 = 36675
indices_four_largest = [914, 37588, 1827, 36675]; % Add two more indices as
 needed

% Extract the corresponding complex coefficients
coefficients_two_largest = fourierSeriesCoeffs(indices_two_largest);
coefficients_four_largest = fourierSeriesCoeffs(indices_four_largest);

% Create a new vector with all zeros
synthesized_signal_two_largest = zeros(size(fourierSeriesCoeffs));
synthesized_signal_four_largest = zeros(size(fourierSeriesCoeffs));

% Set the values at the identified indices to the complex coefficients
synthesized_signal_two_largest(indices_two_largest) =
 coefficients_two_largest;
synthesized_signal_four_largest(indices_four_largest) =
 coefficients_four_largest;

% Inverse Fourier Transform to synthesize the audio signals
audio_signal_two_largest = ifft(synthesized_signal_two_largest);
audio_signal_four_largest = ifft(synthesized_signal_four_largest);

% Play the synthesized audio signals
%soundsc(real(audio_signal_two_largest), fs);
%pause(length(audio_signal_two_largest) / fs); % Pause for the duration of the
 audio
%soundsc(real(audio_signal_four_largest), fs);

% Describe each synthesized audio sound like:
% The synthesize of two largest peak sounds like a parabola, it increasing,
    % then smoothly decreasing overtime.
% The synthesize of four largest peak sounds richer with higher pitch,
    % volume and more consistent

% ------------------------------------------------ %
%b)
fprintf('--------Section 3b)--------\n');
% Nkeep must be even to have an equal number of negative and positive freq.
% Personal Note: # Nkeep > 100 will broke this
Nkeep = 2;
Nseries = length(fourierSeriesCoeffs);

% Needs fourierSeriesCoeffs vector computed in Section 2.3 above
synthSoundCoeffs = zeros(Nseries, 1);
fourierSeriesCoeffsAbs = abs(fourierSeriesCoeffs);

% Initialize array to store peak frequencies for this case
```

```
peak_frequencies = zeros(Nkeep, 1);

% Find the Nkeep strongest positive and negative frequency components
for n = 1:Nkeep
    [ak, k] = max(fourierSeriesCoeffsAbs);
    synthSoundCoeffs(k) = fourierSeriesCoeffs(k);
    fourierSeriesCoeffsAbs(k) = 0;

    % Calculate the corresponding peak frequency
    peak_frequencies(n) = ff(k); % Convert index to frequency

end

% Convert Fourier series coefficients to time domain using inverse FFT
synthSound = ifft(synthSoundCoeffs);

% Print peak frequencies for this case
fprintf('Keeping %d largest peaks:\n', Nkeep);
fprintf('Peak Frequencies: %s\n', num2str(peak_frequencies'));
fprintf('\n');

% Play the synthesized audio and describe what you hear
%soundsc(synthSound, fs);

% i. Keep 2, 4, 6, and 8 of the largest peaks and describe what you hear.
    % Keep 2: The sound is high pitch and stay constant over time
    % Keep 4: The sound start at lower pitch, increasing then decresing to
 lower pitch over time
    % Keep 6: The sound seem to start at lower pitch than the "4 peaks".
 However, it still increasing then decresing to lower pitch over time
    % Keep 8: The sound is very similar to the "6 peaks", but a bit smoother
 while increasing and decreasing over time.

% ii. Give the peak frequencies of the peaks for each case of keeping 2, 4,
 and 6 largest peaks.
    % Keep 2: -5251.05        5251.05
    % Keep 4: -5251.05        5251.05     -5250.7636      5250.7636
    % Keep 6: -5251.05        5251.05     -5250.7636      5250.7636
 -5250.4773      5250.4773

% ------------------------------------------------- %
% c)
fprintf('--------Section 3c)--------\n');

%i.
% Given peak frequency (e.g., 261.45 Hz)
peak_frequency = 261.45; % Adjust as needed

% Given masking range (e.g., +/- 7.5 Hz)
masking_range = 7.5; % Adjust as needed

% Compute the indices kmin and kmax based on frequency resolution
freq_resolution = fs / N; % Frequency resolution for FFT
kmin = round((peak_frequency - masking_range) / freq_resolution);
```

```
kmax = round((peak_frequency + masking_range) / freq_resolution);


%ii.
% Initialize the number of peaks to keep (e.g., 2, 4, 6, 8)
Nkeep_values = [2, 4, 6, 8];

for i = 1:length(Nkeep_values)
    Nkeep = Nkeep_values(i);

    % Reset coefficients, absolute values, and peak frequencies
    synthSoundCoeffs = zeros(Nseries, 1);
    fourierSeriesCoeffsAbs = abs(fourierSeriesCoeffs);
    peak_frequencies = zeros(Nkeep, 1);

    % Find and keep the Nkeep strongest positive and negative frequency
 components
    for n = 1:Nkeep
        [ak, k] = max(fourierSeriesCoeffsAbs);
        peak_frequencies(n) = ff(k); % Store peak frequency

        % Compute kmin and kmax for frequency masking
        kmin = round((peak_frequencies(n) - masking_range) / freq_resolution);
        kmax = round((peak_frequencies(n) + masking_range) / freq_resolution);

        % Zero out frequencies within the masking range
        fourierSeriesCoeffsAbs(max(Nseries, kmin):min(Nseries, kmax)) = 0;

        % Keep the current peak by setting its coefficient
        synthSoundCoeffs(k) = fourierSeriesCoeffs(k);

        % Zero out the processed peak so that the next peak can be found
        fourierSeriesCoeffsAbs(k) = 0;
    end

    % Convert Fourier series coefficients to the time domain using inverse FFT
    synthSound = ifft(synthSoundCoeffs);

% Describe what I hear:
    % Keep 2: The sound is high pitch and stay constant over time
    % Keep 4: The sound start at lower pitch, increasing then decresing to
 lower pitch over time
    % Keep 6: The sound seem to start at lower pitch than the "4 peaks".
 However, it still increasing then decresing to lower pitch over time
    % Keep 8: The sound is very similar to the "6 peaks", but a bit smoother
 when increasing and decreasing over time.

%iii.
    % Print peak frequencies for this case
    fprintf('Keeping %d largest peaks:\n', Nkeep);
    fprintf('Peak Frequencies: %s\n', num2str(peak_frequencies'));
    fprintf('\n');

    % Play the synthesized audio and describe what you hear
```

```
    soundsc(real(synthSound), fs);
    pause(length(synthSound) / fs); % Pause for the duration of the audio
end
```

```
--------Section 3b)--------
Keeping 2 largest peaks:
Peak Frequencies: -5251.05        5251.05

--------Section 3c)--------
Keeping 2 largest peaks:
Peak Frequencies: -5251.05        5251.05

Keeping 4 largest peaks:
Peak Frequencies: -5251.05        5251.05     -5250.7636      5250.7636

Keeping 6 largest peaks:
Peak Frequencies: -5251.05        5251.05     -5250.7636      5250.7636
 -5250.4773      5250.4773

Keeping 8 largest peaks:
Peak Frequencies: -5251.05        5251.05     -5250.7636      5250.7636
 -5250.4773      5250.4773     -5251.3364      5251.3364
```

Published with MATLAB® R2023a