By **Team Banda:**
Jonathan Guaman
Cameron Adomako-Adjei
Faculty Advisor: Dr. Celcia Dong

Table of Contents

1. Objective

The goal of this report is to create music using digitally generated signals and analyze the resulting signal in both the time and frequency domains. Musical notes can be represented as sinusoidal waves oscillating at different frequencies, as seen in [1]. The combination of all these different sinusoids results in a complex signal, which we hear as music. This generated musical signal can be analyzed via Fourier analysis, which allows one to observe the individual frequency components that lead to the creation of music.

2. Procedure

2.1. Background

When it comes to generating musical notes using digitally created signals, we first needed to understand how these waveforms are formed. Sound waves are produced when waveforms vibrate the surrounding molecules at audio frequencies. For example, in [2] it states that the human hearing range is from 20 Hz to 20 kHz, meaning humans can hear frequencies as low as 20 Hz and as high as 20 kHz.

The process of how digital computers or devices handle signals begins by digitizing a continuous signal. This is done by sampling it at discrete points in time to create a digital representation of the waveform. Reconstructing the continuous signal from these discrete samples requires following the Nyquist frequency, which means the sampling frequency must be at least twice the highest frequency present in the signal being sampled.

2.2. Musical Notes

When it comes to generating musical notes, the American Standard Pitch Notation (ASPN) is used. ASPN is a method for labeling musical notes based on their pitch and octave. Each note is labelled using the traditional tone names (A through G), followed by a number indicating its octave. The most important note in this system is the *Stuttgart pitch* [4], also known as A4. This note is important because it serves as the reference frequency used to calibrate acoustic equipment. Using the Stuttgart pitch, we can define the following equation:

$$F_N = 440 * 2^{n/12} \tag{1}$$

The equation means that to find the frequency of other musical notes in any octave. The value *n* stands for how many semitones away the note is from the reference frequency. In [5], semitones are the distance between two adjacent notes of the piano, which include the black keys. This can be done by counting how many key are sepating that note from the other note. Below shows an example of how to count semitones.

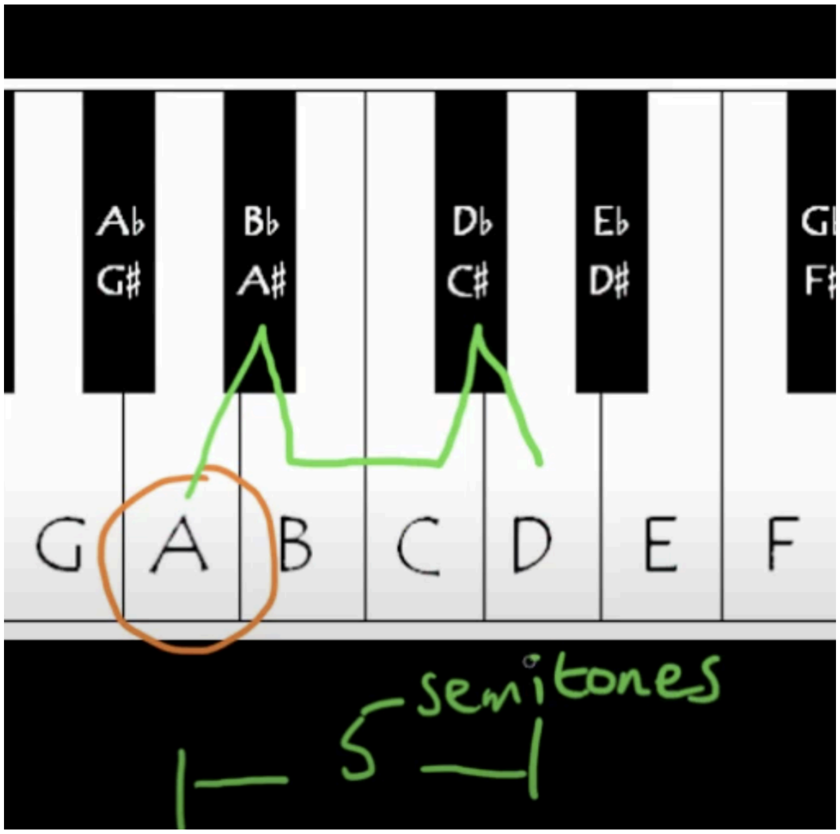

Fig. 2.2.1 - Example of Conting Semitones regarding A and D note [5]

Now that we know the frequency of the note, to digitally create the musical note, we need to generate samples of a sine wave with that frequency in mind. This leads to the following equation:

$$N \ = sin(2\pi F_N t) \tag{2}$$

This equation digitally generates the sine wave that represents the musical note we want. The variable *f* is the frequency of the desired note, and *t* is the duration of the note. The value of *t* relates to the type of musical note we want to create. Musical notes last for different amounts of time, and this duration corresponds to the type of note—for example: whole notes, half notes, quarter notes, eighth notes, and sixteenth notes.

These durations can be determined by dividing 2 by the type of note we want, since 4 beats occur in 2 seconds. Therefore, dividing 2 by the note type gives us how long the note should last.

2.3. Sheet music

Using the equations above, we should be able to create digitalised music notes that come together to form a song. The song we plan to digitally create in MATLAB is "*Adventure of a Lifetime*" by Coldplay. To generate the music, we decided to use the piano sheet music from [3].

To make the sheet easier to read, we used the figure shown below. By using this figure along with equations (1) and (2), we were able to digitally create the chorus of *Adventure of a Lifetime*.

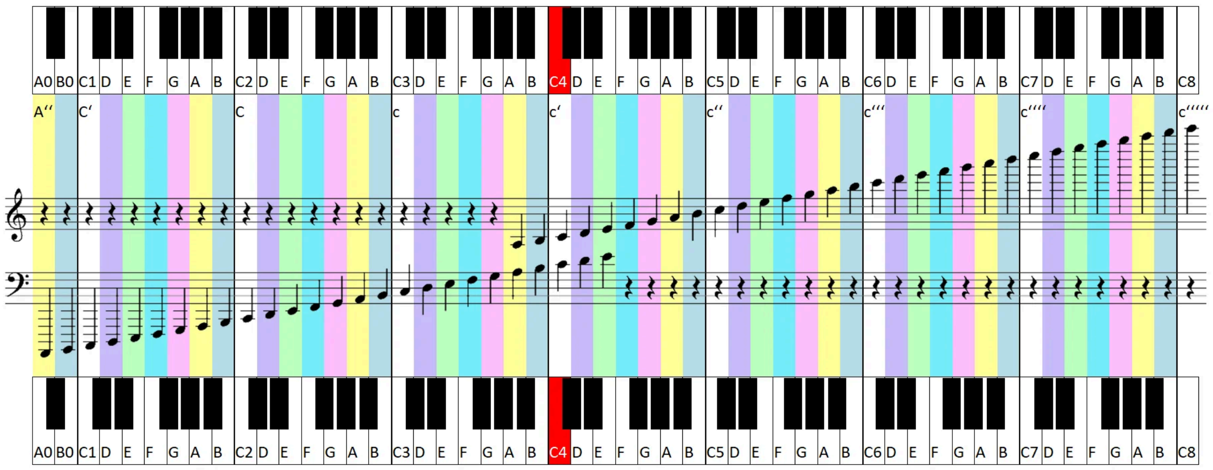
A0 B0 C1 D E F G A B C2 D E F G A B C3 D E F G A B C4 D E F G A B C5 D E F G A B C6 D E F G A B C7 D E F G A B C8
A'' C' C c c' c'' c''' c'''' c'''''

Fig. 2.3.1 - Key for Reading Piano Notes - [6]

2.4. Time Domain

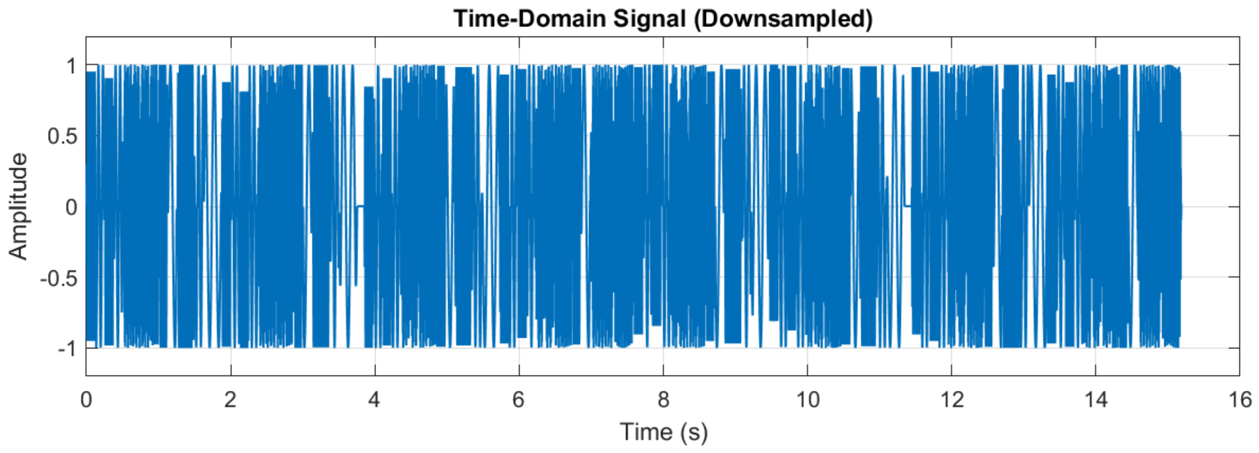

Fig. 2.4.1 - Time Domain Representation of digitally generated music

This graph displays the signal amplitude as a function of time. It is the discrete-time sequence x[n] after all the notes have been generated and windowed. It is the sampled signal produced at a rate of 8000 samples per second. Each note is a finite-duration sinusoid (because the sine wave is windowed). The fading in and out of the waveform shows the envelope shaping operation, which is a time-domain multiplication. The signal as a whole is nonperiodic even though each note is periodic. The shape of the waveform directly reflects the superposition and sequencing of the individual note segments.

2.5. Frequency Domain

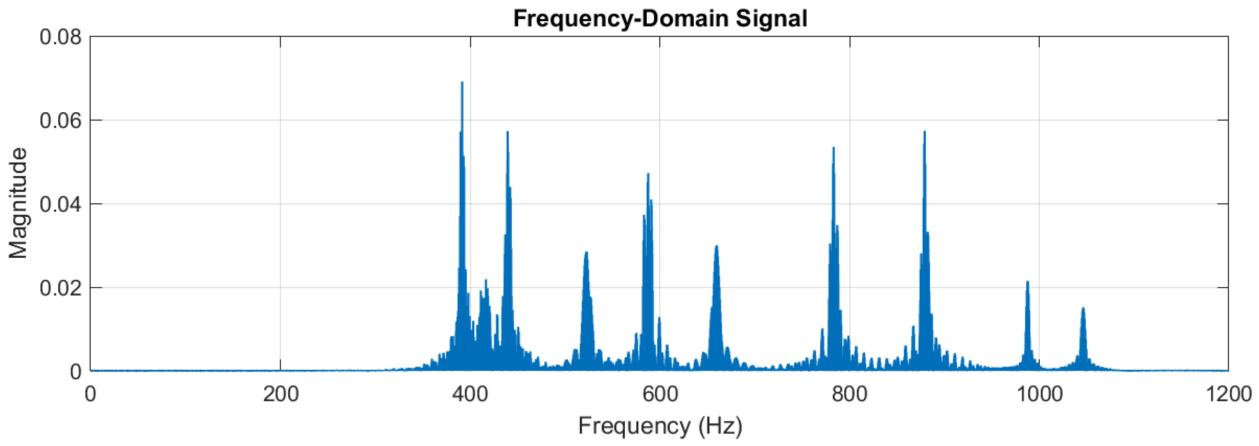

Fig. 2.5.1 - Frequency Domain Representation of digitally generated music.

This graph displays the magnitude spectrum of the entire signal using the Fast Fourier Transform (FFT) and shows how much energy the signal contains at each frequency. Pure sinusoids produce sharp spectral peaks at their fundamental frequencies. Because each note is windowed (faded in /faded out), the peaks are not infinitely narrow; they have a main lobe or peaks and side lobes (small ripples on the left and right of the peaks). These side lobes exist because the . The FFT approximates the discrete-time Fourier transform (DTFT) for a finite-length signal.The plot shows the combination of all frequencies used in the song. Since the system is sampled at 8000 Hz, the[1] Nyquist frequency is 4000 Hz. All musical content falls below 1200 Hz, which is within the allowed bandwidth.

[1]Nyquist frequency- is half the sampling rate. Side note any frequency above the nyquist will distort the signal

2.6. Spectrogram

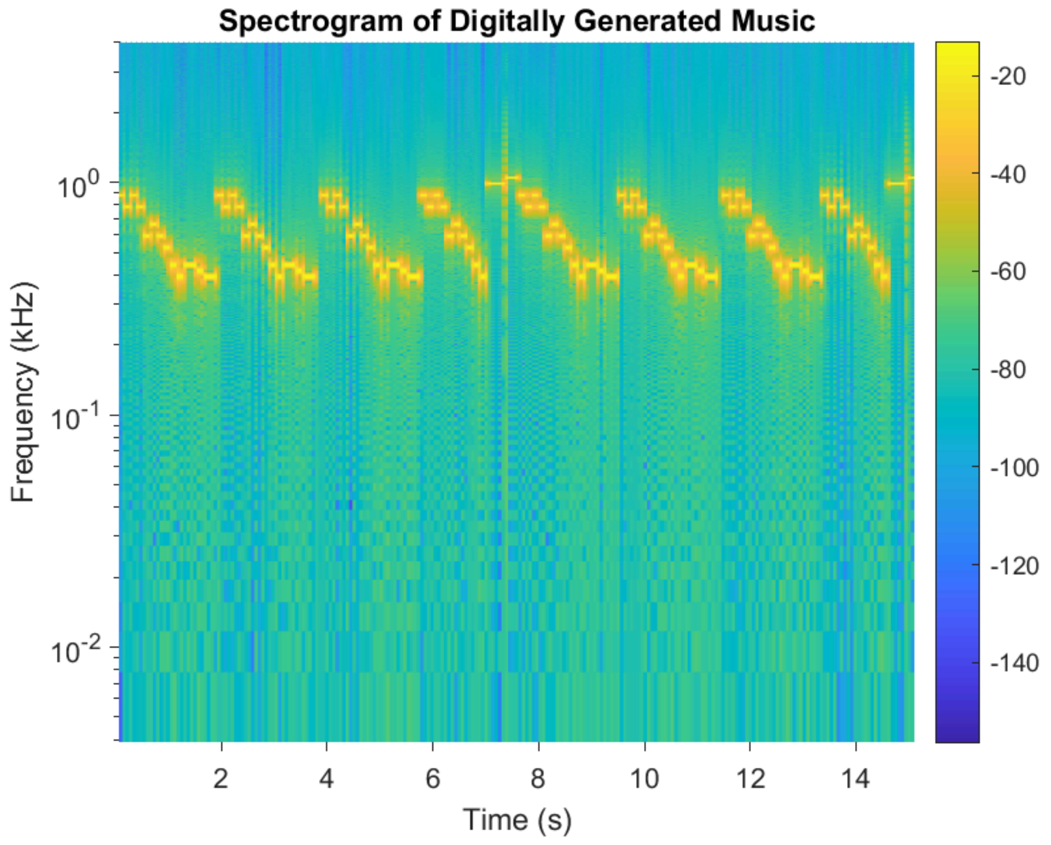

Fig. 2.6.1 - Digitally generated music Spectrogram

The Spectrogram in fig.2.6.1 is a short-time Fourier transform (STFT) that displays how the frequency content of the signal changes over time. The signal is nonstationary, meaning different notes occur at different times. The spectrogram shows frequency vs. time, with color representing magnitude. Horizontal bright lines correspond to the sinusoidal notes. Each block of color corresponds to one finite-duration sinusoid (a musical note). The choice of window length and overlap affects time and frequency resolution (time-frequency trade-off). This graph reveals the structural transitions between musical notes, which cannot be seen in the FFT alone. Through this graph, you can see/visualize when each note occurs, observe how the spectrum evolves, and understand the STFT windowing effects and time-frequency resolution.

3. Project Management

Team Member: Jonathan Guaman
This member is responsible for the research behind digitally creating musical notes in MATLAB by studying music theory and piano note structure. They also worked on generating the chorus of the song in MATLAB. Lastly, they worked on the term report, especially when it comes to the parts of the procedure section.
Team Member: Cameron Adomako-Adjeji
This member is responsible for coding aspects of the project, especially when it comes to the plotting of the generated music. They also worked on the creation of the report and the video on how to run the code.

4. References

[1] Pitch — English Octave-Naming Convention," Dolmetsch Online, Music Theory 1, [Online]. Available: https://www.dolmetsch.com/musictheory1.htm#uspitch
[2]Audiometry," UCSF Health. [Online]. Available: https://www.ucsfhealth.org/medical-tests/audiometry
[3] Adventure of a Lifetime Sheet Music for Piano (Solo) Easy — Coldplay arranged by Joni Lado," MuseScore, [Online]. Available: https://musescore.com/user/23468886/scores/6286127
[4]Apel, Willi (1969). Harvard Dictionary of Music. Harvard University Press. ISBN 9780674375017. Retrieved 2023-01-09 – via Google Books.
[5] Piano Quickie 2: Semitones Explained," *YouTube*, 12 Jan. 2013. [Online]. Available: https://www.youtube.com/watch?v=gOgIzODO9fM. [Accessed: 20 Nov. 2025]
[6] Image post," Reddit. [Online]. Available: https://www.reddit.com/media?url=https%3A%2F%2Fpreview.redd.it%2Fc0fyona0cm281.png%3Fwidth%3D1865%26format%3Dpng%26auto%3Dwebp%26s%3D0560f9ea7a50e0b8ca198595d49f86bc0c520851
[7]"What Is the Nyquist Theorem." *What Is the Nyquist Theorem - MATLAB & Simulink*, MathWorks, **www.mathworks.com/discovery/nyquist-theorem.html.**
[8]"Spectrogram." *Spectrogram Using Short-Time Fourier Transform - MATLAB*, www.mathworks.com/help/signal/ref/spectrogram.html

Appendix

```
% sampling frequency
FS = 8000;
% semitone frequency function
Freq = @(n) 440 * 2^(n/12);
% note generator (with soft envelope)
note = @(freq, dur) softEnvelope( sin(2*pi*freq*(0:1/FS:dur)) );
pauseSeg = zeros(1, FS * 0.1);   % 0.1 sec pause
%Notes base on music sheet
line1 = [note(Freq(12),.125), note(Freq(10),.125), note(Freq(12),.125),
note(Freq(10),.125)];
line2 = [note(Freq(5),.125), note(Freq(7),.125), note(Freq(5),.125),
note(Freq(3),.125)];
line3 = [note(Freq(0),.125), note(Freq(-2),.125)];
line4 = [note(Freq(0),.25), note(Freq(-1),.125), note(Freq(-2),.25)];
line5 = [note(Freq(14),.35), note(Freq(14),.01), note(Freq(15),.25)];
song = [line1,line2,line3,line4];
bridge = [line1,line2,line3,line5];
cs = [song,song,pauseSeg,song,bridge];
chorus = [cs, cs];
sound(chorus, FS);
%TIME + FREQUENCY DOMAIN PLOTS
t = (0:length(chorus)-1) / FS;
%Downsample factor
ds = 20;
t_ds = t(1:ds:end);
chorus_ds = chorus(1:ds:end);
figure('Position',[200 200 900 600]);  % larger window
%TIME DOMAIN
subplot(2,1,1);
plot(t_ds, chorus_ds, 'LineWidth', 1);
xlabel('Time (s)');
ylabel('Amplitude');
title('Time-Domain Signal (Downsampled)');
grid on;
ylim([-1.2 1.2]);
%Frequency Domain
N = length(chorus);
Y = abs(fft(chorus))/N;
f = (0:N-1)*(FS/N);
subplot(2,1,2);
plot(f(1:floor(N/2)), Y(1:floor(N/2)), 'LineWidth', 1);
xlabel('Frequency (Hz)');
```

```
ylabel('Magnitude');
title('Frequency-Domain Signal');
grid on;
xlim([0 1200]);
%Spectrogram and setup
figure;
window = 1024;          % size of FFT window
noverlap = 512;         % overlap between windows
nfft = 2048;            % FFT size for frequency resolution
spectrogram(chorus, window, noverlap, nfft, FS, 'yaxis');
set(gca,'YScale','log')
colorbar;
title('Spectrogram of Digitally Generated Music');
ylabel('Frequency (kHz)');
xlabel('Time (s)');
% function to soften the sound a nd make it less pitchy
function y = softEnvelope(x)
   L = length(x);
   fade = round(L * 0.08);
   env = ones(1, L);
   env(1:fade) = linspace(0,1,fade);
   env(end-fade+1:end) = linspace(1,0,fade);
   y = x .* env;
end
```