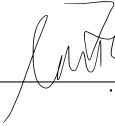


ECE280 - Lab 1: Music Synthesis

Sylvester Johannes Arizie (sna31)

September 28, 2024

I have adhered to the Duke Community Standard in completing this assignment.



Contents

1	Introduction	1
2	Procedure	1
3	Results	1
4	Discussion	1
5	MATLABTM code	2
6	Extension	4

1 Introduction

- I composed a piano version of Twinkle Twinkle Little Stars From Nursery Rhymes as it is a really simple song that makes use of only a subset of keys (notes) and is quite similar in theme with "Mary had A Little Lamb". The simplicity in the original song allowed me the freedom to include enhancements towards producing a better sounding song.
- The signal processing effects that would be explained during this lab include 1. Sampling 2. Generating Discrete Signals using mathematical functions and Generating Continuous signal from this data set 3. Synthesizing music using sinusoidal functions and altering the properties of sinusoids to influence the quality of music produces 4. Volume Variation, Overlapping Tones , Instrument Synthesis, and Audio Processing Effects

2 Procedure

- INSERT EXPLANATION The notes in the music were generated using a combination of the 'Soundsc' function in MATLAB, a set sampling rate, time vectors, frequency vectors and sinusoidal functions. I selected a sampling frequency to match the frequency of the treble clef from my music sheet. I then created an array for the corresponding frequencies of expected notes. Using a time vector, i created vectors to represent the durations of note periods and their corresponding rests. Finally i made a function to generate each note based on its period and frequency. The generated input was stored in a vector which was played at the end via soundsc.
- INSERT EFFECTS For my song, I introduced delay by attenuating the amplitude of the notes generated such that the volume of the sound produced fluctuated with time ,creating the illusion of a "flowing sound". I also introduced harmonics by playing sounds simultaneously. My intuition for this was from the use of both hands to introduce harmonics while playing a piano. To replicate the sound from a piano, I used a sampling frequency that resonates with the C sharp notes of a piano. I also introduced overlapping tones, by combining single notes together.

3 Results

My final song sounded less robotic - although it had feedback, it was less static compared to my original song with a fixed amplitude.

4 Discussion

- Discuss how each effect individually affects the sound.
 - A) Multiple Notes: My use of multiple notes in the song develops rhythm and harmony in the song, I use it to form chords that are more resonant or consonant, reducing the dissonance that exists between single notes played.
 - B) Exponential Decay: The use of exponential decay in the song (through amplitude manipulation) allows for sound modulation - the rise or fall in loudness/amplitude - that allows for volume control. With exponential decay, the volume fluctuates and makes sound produced less statically disorienting. It also creates a sense of depth and spatial dimension
 - C) Harmonics: Harmonics allow for a resonance effect between notes that are 1/2 multiples or whole number multiples of fundamental frequencies, enriching the sound produced and creating the perception of a heightened frequency. I use harmonics by selecting notes that have the same fundamental frequency, making the song hold together without observable discrepancies
 - D) Instrument Synthesis: Given my song mimics a piano, my cosine function uses a frequency of 44100 Hz. Modulating the amplitude of my envelope function by using the exp()) function also helps me achieve the same result.

E) Reverb/Echo: The use of reverb in my song creates the illusion of reflections of the sound from a surface as though the song was being played in a large concert hall.

- What challenges did you encounter, if any, when implementing each effect? How did you resolve the challenges?

The biggest challenge for me was creating a harmonizing sound, and this was because I had initially envisioned playing the different required sounds together instead of first storing them in a vector then playing them together. After including this resolution, I was successful in creating a harmonically resonating sound

- Which effects had the greatest impact on the sound? Which had the least impact? Why?
The effect with the greatest impact on my music would be exponential decay as this drastically changes my song from "static-sounding" to a more natural/ usual sounding song. The effect with the least impact would be the use of echo just because it created a song that was hard to decipher due to the closeness of the notes with each other in the circle of fifths.

- Assuming you had more time, how would you make the song better? Why do you think this would improve your song?

If I had more time, I would try to include synthesis for other instruments, to help build the song and develop the harmonics altogether. This would also make my song stand out more as versions are usually in a single instrument.

- Comments on specific questions.

I. Why is the endpoint $1 - \frac{1}{f_s}$ instead of 1?

The endpoint is stated as $1 - \frac{1}{f_s}$ instead of 1 because of how MATLAB does indexing and this to prevent a form of stack over flow as this bound ensures that our index is reached whereas 1 as an endpoint could generate errors due to $\frac{1}{f_s}$ being a float and thus incrementing improperly

II. Does it matter if you use sin or cos?

It doesn't matter as long as you account for the phase shift differences of 90 degrees when you switch between functions. Otherwise it is easier to use the cosine function as it has already been developed in the pre-lab

III. Why do you need to use vectors of zero amplitude?

Zero vectors are easy to manipulate in that they may represent the presence or absence of a signal, an on or off, to suggest a few. They also allow for accurate timing while maintaining the consistency of timing for the entire structure

IV. What happens when the amplitude signal exceeds the range between -1 and 1?

It is normalized by scaling the entire piece with the maximum value obtained in the sound vector reducing the scaled outputs back to a range between 1 and -1

5 MATLABTM code

- (a) `generatesoundnotes`: The function takes in 2 inputs, the frequency of the note to be generated and the duration for the note. It goes into a conditional loop to check if the value to be generated is a note or a rest period, and performs calculations to return the desired output.

```
%Generating a function that accepts arguments to create a variety of notes  
function result = generatesoundnotes (note_frequency, duration)
```

```
    if note_frequency == 1  
        %soundwave = cos(0.5*pi*frequency*duration);  
        soundwave = 0.5*exp(-1*duration/700).*cos(0.5*pi*note_frequency*duration +  
            (2*pi*duration));
```

```

        else
            %soundwave = cos(2*pi*frequency*duration);
            soundwave = 0.8*exp(-1*duration*note_frequency/90).*
                cos(2*pi*note_frequency*duration);
        end

        result = soundwave;
    end
    % generating and playing sound
    firstrightmeasures = [generatesoundnotes(C, quarter)
        generatesoundnotes(C, quarter) generatesoundnotes(G, quarter)
        generatesoundnotes(G, quarter)
        generatesoundnotes(A, quarter) generatesoundnotes(A, quarter)
        generatesoundnotes(G, quarter) generatesoundnotes(F, quarter)
        generatesoundnotes(F, quarter) generatesoundnotes(G, whole)];

    secondrightmeasures = [generatesoundnotes(E, quarter)
        generatesoundnotes(E, quarter) generatesoundnotes(D, quarter)
        generatesoundnotes(D, quarter) generatesoundnotes(C, quarter)
        generatesoundnotes(G, quarter) generatesoundnotes(G, quarter)
        generatesoundnotes(F, quarter)];

    thirdrightmeasures = [generatesoundnotes(FG, quarter)
        generatesoundnotes(E, quarter) generatesoundnotes(E, quarter)
        generatesoundnotes(D, quarter)
        generatesoundnotes(G, quarter) generatesoundnotes(G, quarter)
        generatesoundnotes(F, quarter) generatesoundnotes(F, quarter)
        generatesoundnotes(E, quarter) generatesoundnotes(E, quarter)];

    fourthrightmeasures = [generatesoundnotes(D, quarter)
        generatesoundnotes(C, quarter) generatesoundnotes(G, quarter)
        generatesoundnotes(G, quarter) generatesoundnotes(A, quarter)
        generatesoundnotes(FG, quarter) generatesoundnotes(F, quarter)
        generatesoundnotes(F, quarter) generatesoundnotes(E, quarter)
        generatesoundnotes(E, quarter) generatesoundnotes(D, quarter)
        generatesoundnotes(C, quarter)];

    firstleftmeasures = [generatesoundnotes(E, quarter)
        generatesoundnotes(E, quarter) generatesoundnotes(E, quarter)
        generatesoundnotes(FG, quarter)
        generatesoundnotes(FG, quarter) generatesoundnotes(E, quarter)
        generatesoundnotes(D, quarter) generatesoundnotes(D, quarter)
        generatesoundnotes(C, quarter) generatesoundnotes(C, quarter)];

    secondleftmeasures =[generatesoundnotes(B, quarter)
        generatesoundnotes(B, quarter) generatesoundnotes(C, quarter)
        generatesoundnotes(E, quarter) generatesoundnotes(E, quarter)
        generatesoundnotes(D, quarter) generatesoundnotes(D, quarter)
        generatesoundnotes(C, quarter) generatesoundnotes(B, quarter)
        generatesoundnotes(E, quarter) generatesoundnotes(E, quarter)];

    thirdleftmeasures =[generatesoundnotes(D, quarter)
        generatesoundnotes(C, quarter) generatesoundnotes(C, quarter)

```

```

generatesoundnotes(G, quarter)
generatesoundnotes(E, quarter) generatesoundnotes(E, quarter)
generatesoundnotes(E, quarter) generatesoundnotes(FG, quarter)
generatesoundnotes(G, quarter) generatesoundnotes(A, quarter)
generatesoundnotes(B, quarter) generatesoundnotes(C, quarter)];

fourthleftmeasures =[generatesoundnotes(D, quarter)
generatesoundnotes(A, quarter) generatesoundnotes(G, quarter)
generatesoundnotes(C, quarter)
generatesoundnotes(G, quarter) generatesoundnotes(FG, quarter)
generatesoundnotes(C, quarter)];

righthand =[firstrightmeasures secontrightmeasures t
hirdrightmeasures
fourthrightmeasures firstrightmeasures
secontrightmeasures thirdrightmeasures
fourthrightmeasures firstleftmeasures
secondleftmeasures thirdleftmeasures
fourthleftmeasures firstleftmeasures
secondleftmeasures thirdleftmeasures
fourthleftmeasures];
%lefthand = [firstleftmeasures secondleftmeasures
thirdleftmeasures fourthleftmeasures firstleftmeasures
secondleftmeasures thirdleftmeasures fourthleftmeasures];
soundsc(righthand, fs);
%soundsc(lefthand, fs);

```

6 Extension