Analyzing and composing music with algorithms and machine learning

Author: Dimo Chanev
Supervisiors: Emil Kelevedjiev
Zornitsa Dzhenkova

August 14, 2018

Contents

Abstract

The goal of the project is to compare, experiment and prove some mathematical algorithms for analyzing and composing similar pieces of music with random numbers. The algorithms are based on analyzing some relations from given note notations of given melodies, in form of standard MIDI files.

1 Introduction

1.1 Frequency relations (tuning)

The sound is a type of mechanical vibration (waves) so it has the parameters of a normal mechanical wave. However, in the field of music we are most interested in the frequency of the wave, because it makes the variety of tones we hear. But not every sequence of frequencies sounds pleasant to the human. So people developed some algorithms to "syncronize" the sequence of tunes. Here are the most used of them:

1.1.1 Pythagorean tuning

The Pythagorean tuning [5] is the base of almost all tuning systems. It is based on the Pythagorean fractions.

$$f = \frac{i+1}{i};\ i = 1, 2, ..., n$$

Figure 1: Pythagorean fractions

It is assumed that if the frequencies correlate like these numbers, we hear them in harmony and the music will sound pleasant (if we select appropriate sequence of these tones of course).

If we pick a string with given length we know that the frequency of the produced sound depends on the length of the string. So to pick the "perfect" fraction, we have to divide it like the Pythagorean fractions.

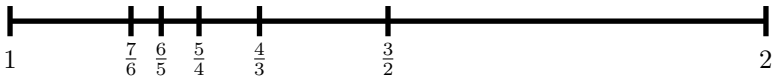


Figure 2: The "pefect" string separation

1.1.2 Equal temperament

Equal temperament is a tuning scale developed by Vincenzo Galilei. His idea is that the frequency ratio between each pair of adjacent notes is the same. In equal temperament tunings, the generating interval is often found by dividing some larger desired interval into a number of smaller equal steps (equal frequency ratios between neighbouring notes). The tones are divided in groups called

octaves. The ratio between the frequencies of last and the first tone of the octave is two.

$$\frac{last\ tune}{first\ tune} = \frac{2}{1}$$

In Western music, the most commonly used tuning system since the 18th century has been the twelve-tone equal temperament (or 12-TET) [3]. The 12-TET divides the octave in 12 eqal parts, all of them equal on logarithmic scale, with ratio $\sqrt[12]{2} = 2^{\frac{1}{12}} \approx 1.05946$. In modern times 12-TET is usually tunned relatively to the standard frequency of 440 Hz, called A4, meaning that the note A is tuned to 440 Hz and all other notes are derived by the logarithmic scale described above.

Equal temperament is an approximation of the Pythagorean scale, which is more convenient for use.

Name	**12-TET**	**Pythagorean scale**
Unision (C)	$2^{\frac{0}{12}} = 1$	$\frac{1}{1} = 1$
Minor second ($C\sharp/D\flat$)	$2^{\frac{1}{12}} \approx 1.05946$	$\frac{16}{15} \approx 1.06666$
Major second (D)	$2^{\frac{2}{12}} \approx 1.12246$	$\frac{9}{8} = 1.125$
Minor third ($D\sharp/E\flat$)	$2^{\frac{3}{12}} \approx 1.18920$	$\frac{6}{5} = 1.2$
Major third (E)	$2^{\frac{4}{12}} \approx 1.25992$	$\frac{5}{4} = 1.25$
Perfect fourth (F)	$2^{\frac{5}{12}} \approx 1.33484$	$\frac{4}{3} \approx 1.33333$
Tritone ($F\sharp/G\flat$)	$2^{\frac{6}{12}} \approx 1.41421$	$\frac{7}{5} = 1.4^*$
Perfect fifth (G)	$2^{\frac{7}{12}} \approx 1.49830$	$\frac{3}{2} = 1.5$
Minor sixth ($G\sharp/A\flat$)	$2^{\frac{8}{12}} \approx 1.58740$	$\frac{8}{5} = 1.6^*$
Major sixth (A)	$2^{\frac{9}{12}} \approx 1.68179$	$\frac{5}{3} \approx 1.66666^*$
Minor seventh ($A\sharp/B\flat$)	$2^{\frac{10}{12}} \approx 1.78179$	$\frac{16}{9} \approx 1.77777^*$
Major seventh (B)	$2^{\frac{11}{12}} \approx 1.88774$	$\frac{15}{8} = 1.875^*$
Octave (C)	$2^{\frac{12}{12}} = 2$	$\frac{2}{1} = 2$

Note: the values with * can't be represented like Pythagorean fractions with decent accuracy but the human ear can't differentiate this (in most cases)

Figure 3: Values of the tones in 12-TET and Pythagorean scale

1.2 MIDI standart

The purpose of MIDI [1] is to define a compact representation of musical sheets, which makes it appropriate for disk storage.

The standard MIDI (SMID) file is divided into chunks. Each chunk has a type and length represented as ASCII and 32 bit number respectively. There are two types of chunks: "MThd"(e.g. Header chunk) and "MTrk"(e.g. Track chunk). There is always one header while there can be more than one tracks.

1.2.1 Header chunk

The header chunk contains data for the whole file. It has fixed length of 6 bytes. The first 2 bytes represent the format of the file:

- $0 \rightarrow$ One track
- $1 \rightarrow$ Multiple tracks but one song
- $2 \rightarrow$ Multiple songs

The second 2 bytes represent the number of tracks. For format 0 it is always 1. And the last two bytes represent how many ticks are in a quarter note.

1.2.2 Track chunks

Each track is composed of events. Each event has delta time – the time (in ticks) which passes after the previous event before executing this event. There are two types of events: MIDI events – the ones which represent the actual music, and META – these which supply additional info like titles, lyrics, etc.

Here are listed the most important events (however, they are useless without the others):

- NOTEON – this event initiates playing with given pitch (frequency) and velocity. The MIDI representation of this event is:
  "1001nnnn 0kkkkkkk 0vvvvvvv" where nnnn is the binary representation of the channel, kkkkkkk - the pitch and vvvvvvv - the velocity
- NOTEOFF - this event stops playing the note with given pitch (frequency) and the intensity of the stop. The MIDI representation of this event is:
  "1000nnnn 0kkkkkkk 0vvvvvvv" where nnnn is the binary representation of the channel, kkkkkkk - the pitch and vvvvvvv - the velocity

Each MIDI event that has to play a note uses this standard for encoding the frequency of the played tune [6]:

$$f = 440 * 2^{\frac{d-69}{12}} \textit{ where } d \textit{ is the note number and } f \textit{ is the } frequency$$

Therefore the note number is:

$$d = 69 + 12 \log_2 \frac{f}{440}$$

1.3 Markov chains

Markov chain [4] is a stochastic model which describes a sequence of events in which the probability of each event depends entirely on the previous event (in some cases the previous n but the table which describes the chain size becomes big quickly because it equals to k^n *where k is the number of events*).

Markov chains are represented as a table in the most cases. Each cell of the table contains the probability of triggering an event from the row/column after the event in the column/row respectively (see Figure 4).

	Next event		
	A	B	C
A	33%	22%	45%
B	81%	9%	10%
C	30%	60%	10%

		Next event		
		A	B	C
A	A	16%	16%	68%
A	B	100%	0%	0%
A	C	12%	75%	13%
B	A	37%	25%	38%
B	B	0%	0%	100%
B	C	100%	0%	0%
C	A	33%	33%	34%
C	B	83%	17%	0%
C	C	100%	0%	0%

Figure 4: Example Markov chains for the string
"AABAACBABACBABACCACAACBAAACBACBBCABAACBA"

1.4 Used algorithms

Each algorithm is based on analysis with Markov chains. Their common block diagram is represented in Figure 5.

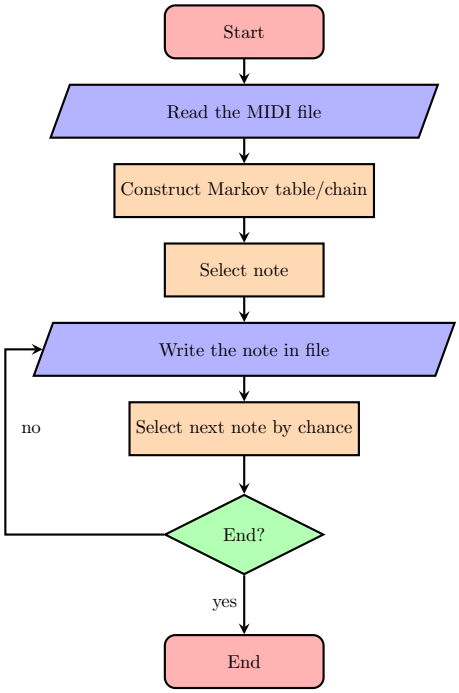


Figure 5: A common block diagram of the algorithms

1.4.1 Algorithm with one tune pitch

This algorithm constructs a Markov table based only on the frequencies of the tones. The size of the table depends on the variety of tunes used. Then it selects starting note and each next note is picked using random number generator and the chances in the Markov chain.

1.4.2 Algorithm with one tune pitch and note duration

This algorithm constructs a Markov table based on the frequencies of the tones and their length. Because of this, the size of the table increases and depends not only on the variety of notes but on the variety of note lengths as well. Then it selects a starting note and each next note is picked using random number generator and the chances in the Markov chain.

1.4.3 Algorithm with two tune pitch

This algorithm constructs a Markov table based only on the frequencies of the tones but it calculates the chance of next event, based on the previous two, so the size of the table increases and is the size of the table in 1.4.1 squared. Then it selects a starting note and each next note is picked using random number generator and the chances in the Markov chain.

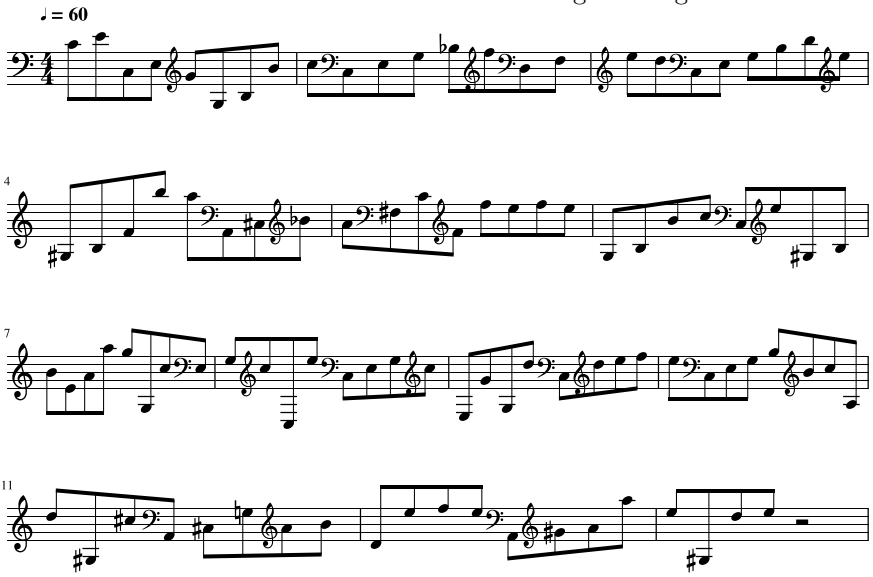
1.4.4 Rust

All the implementations of the algorithms for the project are written in Rust programming language [2]. It is based on C but guarantees memory safety, threads without data races, generics (templates) and the standard library is bigger and has some bindings for functional programming.

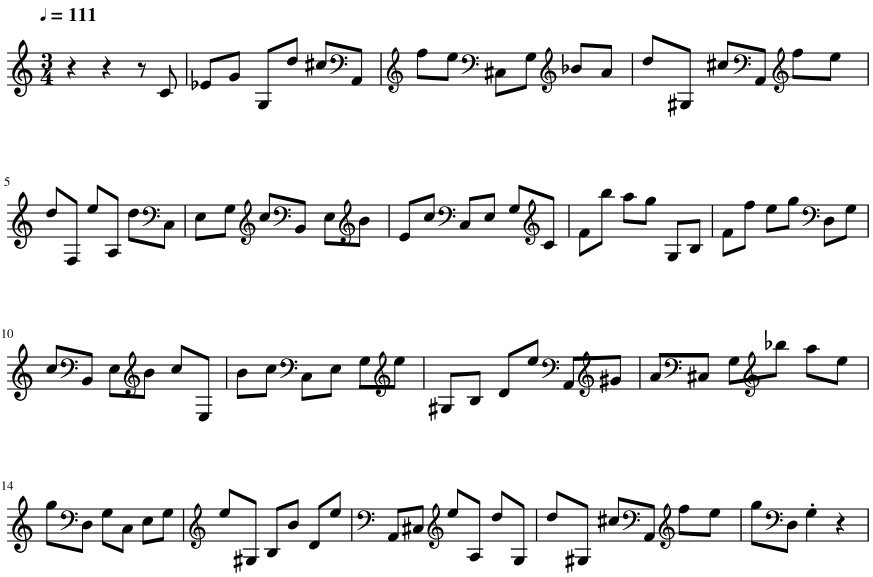
2 Results

Here are given some example results from tests. The tests are run with original music "Noctune op. 9 no. 2" written by Chopin (the note notation is given in Figure 6):

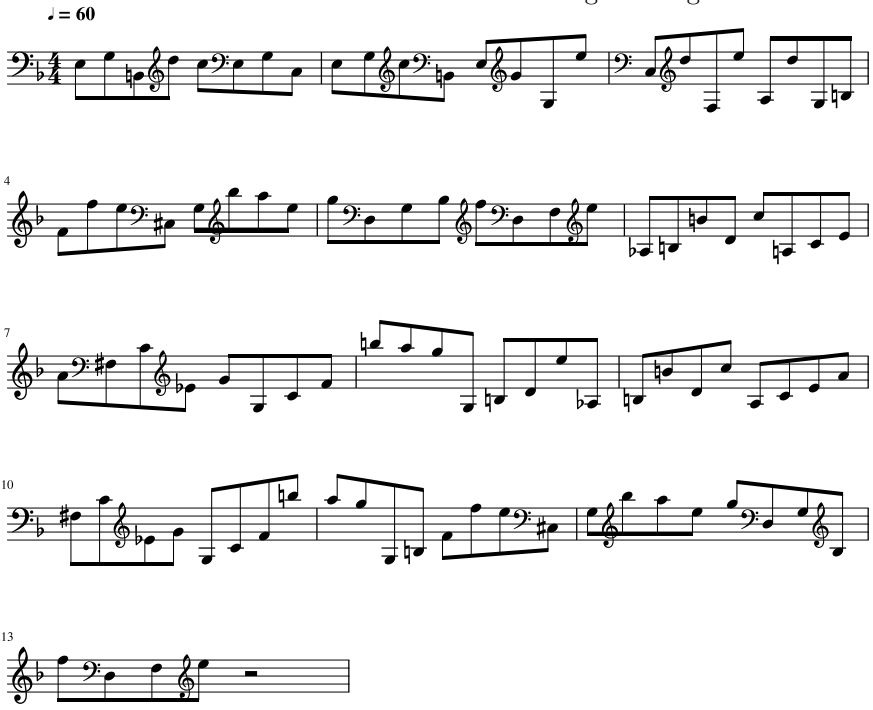
- Below is shown the output of the algorithm that constructs the Markov chain with one note before an event without using the length.



- Below is shown the output of the algorithm that constructs the Markov chain with one note before an event using the length.



- Below is shown the output of the algorithm that constructs the Markov chain with two notes before an event without using the length.



All the sheets are given in the github repo of the project like MIDI files if the reader is interested in hearing how they sound like.

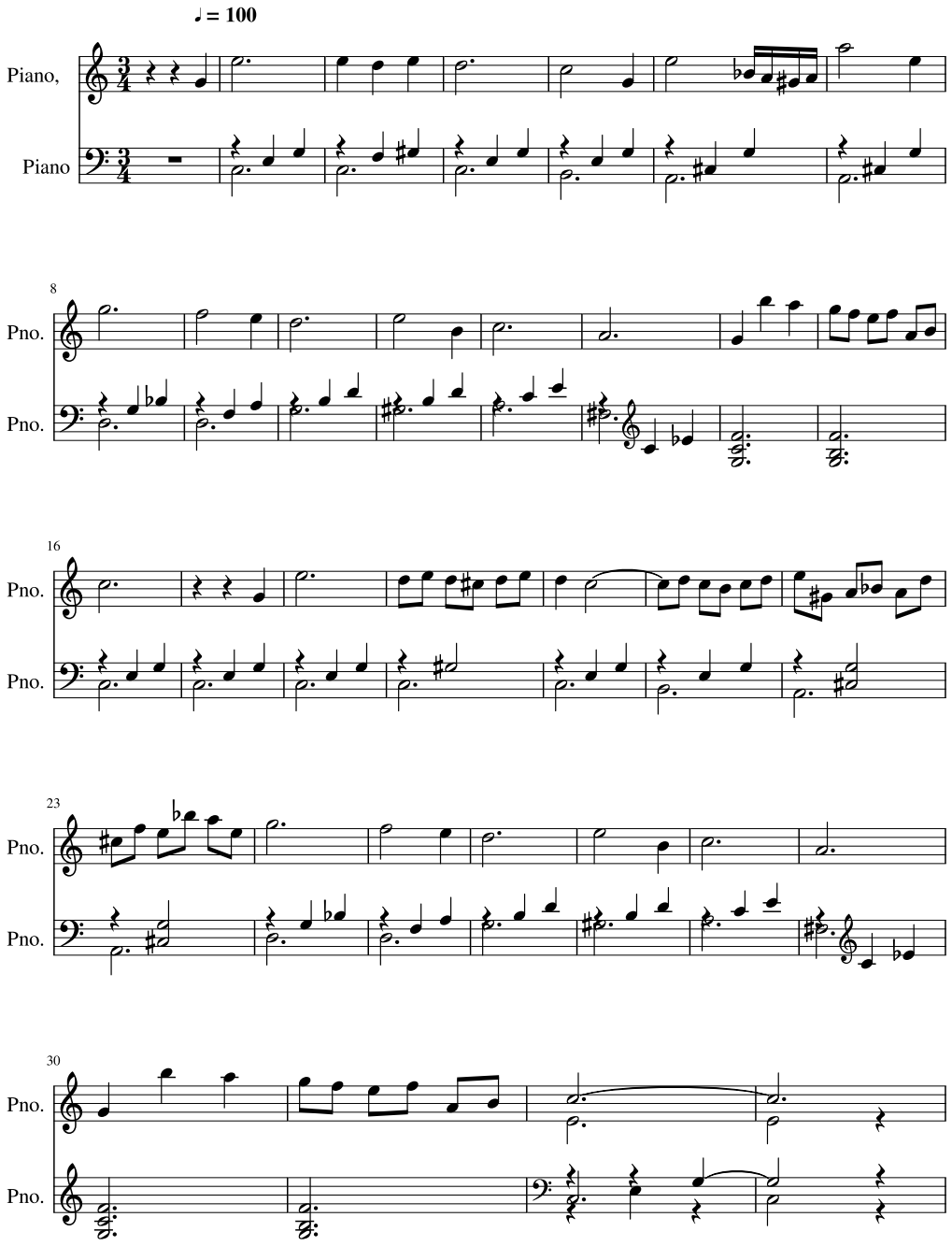


Figure 6: Noctune op. 9 no. 2 by Chopin

3 Future development

The future of the project is aimed at developing more algorithms for composing music and statistical models. Here are listed the planned improvements:

1. Parallelize the algorithms to improve the speed of the construction of the table because on large amount of notes the algorithm that uses the length of the notes is too slow;
2. Implement the algorithm that uses a pair of notes and their lengths to predict the next one;
3. Use Markov chains to check the similarity of two musical pieces;
4. Use neural networks to generate music;
5. Implement other algorithms.

4 Code

All the following links lead to certain branches or commits of the GitHub repository of the project:

- Implelentation of the algorithm that constructs the table using one note for next note prediction
- Implelentation of the algorithm that constructs the table using one note and its length for next note prediction
- Implelentation of the algorithm that constructs the table using two notes for next note prediction

Aknowledgements

- Emil Kelevedjiev - for explaining the theory of Markov chains and introduction to the music programming
- Zornitsa Dzhenkova - evaluating the results of the algorithms and suggesting fixes for the bugs
- HSSIMI - for the opportunity to participate in SRS '18
- Zvezdin Besarabov - help with the structure of the paper
- Pressiana Marinova - 6toto sa pi4ui and for proofread
- Ivaylo Zhelev - 6toto sa pi4ui
- Georgi Gachev - 6toto sa pi4ui

References

[1] David Back. Standard midi-file format spec. 1.1, updated. http://www.music.mcgill.ca/~ich/classes/mumt306/StandardMIDIfileformat.html#BM0_, 1999.

[2] The Rust Project Developers. The rust programming language. https://doc.rust-lang.org/book/second-edition/index.html, 2011.

[3] Joe Monzo. 12-tone equal-temperament. http://tonalsoft.com/enc/number/12edo.aspx.

[4] Craig Stuart Sapp. Digital music programming ii: Markov chains. http://peabody.sapp.org/class/dmp2/lab/markov1/.

[5] Margo Schulter. Pythagorean tuning and medieval polyphony. http://www.medieval.org/emfaq/harmony/pyth.html, 1998.

[6] Joe Wolfe. Note names, midi numbers and frequencies. https://newt.phys.unsw.edu.au/jw/notes.html.