

*By Cameron Yeomans, Emily
Burke, Hunter Moffat, and
Spencer Cameron*

*CALCULATING
RHYTHMIC PATTERN
OUTCOMES
THROUGH MATRIX
MANIPULATION AND
MARKOV CHAINS*



Contents

- Introduction and Musical Background
- Usage of Linear Algebra
- Markov Chains
- Code and program we created



Intro and Musical Background

- In order to understand the project, one must understand the basics of musical composition.
- Musical pieces are composed of bars, bars represent specific a time frame in which notes are played sequentially. Bars are used to measure how many beats are to be played in a certain time frame (usually based off of 4 beats per bar).
- Beats represent the type of note and duration of the note in a given bar. Beats can range anywhere from $\frac{1}{4}$ of a bar all the way to $\frac{1}{32}$ or even $\frac{1}{256}$ of a bar in some cases. The duration of each beat is determined by the time signature of the piece.

Usage of Linear Algebra in our project

Rhythmic Combinations

The image displays six staves of musical notation, each containing a sequence of eighth notes and eighth rests. The notation is presented in a clean, black-and-white style. The first staff begins with a treble clef and a common time signature. The notes and rests are arranged in a way that suggests a specific rhythmic pattern, likely related to the 8-bit binary sequence mentioned in the text. The staves are numbered 5, 10, 15, 20, 25, and 30, indicating the measure numbers.

For our project, we want to generate all possible rhythmic outcomes using matrices.

For our project, we want to generate all possible rhythmic outcomes using matrices. There will be a certain amount of outcomes for the rhythm desired. To accomplish this, we created a program in Python which uses eighth notes and eighth rests in order to replicate the 8 bit binary sequence. This software will allow a user to enter in a certain pattern to start off the rhythm with (or not, if you would like it to be completely random), and then have the computer generate measures of rhythmic patterns based on the user input.

The diagram shows two states, A (blue circle) and B (orange circle), with self-loops and transitions between them. Below, two transition matrices are shown. The 'First-order MC' is a 2x2 matrix with states represented by musical notes. The 'Revised MC' is a 3x3 matrix with states labeled as (N_i, j) .

MARKOV CHAINS

We can represent the output of our program with a Markov chain. Simply put, Markov chains are a set of transitions, which are determined by probability distributions, that must satisfy something called the “Markov property”. The Markov property is a “memoryless property of a stochastic process.”

We can represent the output of our program with a Markov chain. Simply put, Markov chains are a set of transitions, which are determined by probability distributions, that must satisfy something called the “Markov property”. The Markov property is a “memoryless property of a stochastic process.”

Because Markov chains are mathematical systems that track the probabilities of state changes, they can be used to model and predict behavior. For our project, after we generate a musical piece (either randomly, or based off user input), we can then use Markov chains to analyze the probability of each bar inside the musical piece. To do this, we would analyze how often one specific bar is followed by another specific bar. You would turn this into an adjacency matrix. After getting the adjacency matrix you would turn all the adjacencies for that bar into probabilities. We could also modify our program to analyze pre-composed pieces of music.

The Markov chain data from each bar provides many real world uses for our program. One specific use might be to use the data generated by our program to find if there are copyright infringements with a song or musical piece that already exists. This is done by comparing the Markov chain values of what follows “bar A”, in one song to “bar B” in another. If the Markov chains are identical/similar then so are the songs. This feature could be used across many different music sharing mediums like Spotify, Youtube, or Apple

Music. Our program is pretty limited though, because it only produces sequences of one specific note and not every note in the musical scale.

Code / program we created

```
23 rhythmsInBinary = []
24
25 binaryVal = ""
26 for x in range(256):
27     binaryVal = str(bin(x)[2:])
28     while(len(binaryVal) != 8):
29         binaryVal = "0" + binaryVal
30
31     match = True
32     for y in range(len(searchRhythm)):
33         if(searchRhythm[y] == "?"):
34             continue
35         elif((searchRhythm[y] == "+" and binaryVal[y] == "0") or (searchRhythm[y] == "-" and binaryVal[y] == "1")):
36             match = False
37     if(match):
38         rhythmsInBinary.append(binaryVal)
39     match = True
40
41
42 print(rhythmsInBinary)
43
44 rhythmString = ""
45 lilypondString = []
46 for i in range(len(rhythmsInBinary)):
47     rhythmString = rhythmsInBinary[i]
48     for j in range(len(rhythmString)):
49         if(rhythmString[j] == "0"):
50             lilypondString = lilypondString + "r8"
51
```

```
4 import re
5 regex = "[+|\\-|\\?]+$"
6
7 print("Rhythm builder")
8 print("use + to enter an eighth note")
9 print("use - to enter an eighth rest")
10 print("use ? to enter a wildcard")
11 print("input must be 8 chars or less and contain only the chars + and -")
12 error = True
13
14 while(error):
15     searchRhythm = raw_input("Enter a valid rhythm: ")
16     valid = re.search(regex, searchRhythm)
17     if(len(searchRhythm) < 9 and valid):
18         error = False
19     if(searchRhythm == ""):
20         break
21
22
23 rhythmsInBinary = []
24
25 binaryVal = ""
26 for x in range(256):
27     binaryVal = str(bin(x)[2:])
28     while(len(binaryVal) != 8):
29         binaryVal = "0" + binaryVal

```

For our project we wrote a python program that would find every possible rhythm given a starting set of between up to 8 eighth rest or eighth notes.

For our project we wrote a python program that would find every possible rhythm given a starting set of between up to 8 eighth rest or eighth notes. We programmed this in python using an extension called LilyPond.

What we do first is ensure that the rhythm entered is of the right format. We want at most 8 characters, consisting of only '+', '-', and '?'. As you can see below, the '+' represents an eighth note, the '-' an eighth rest, and a '?' can represent either.

The second thing that we must do in our program is next evaluate all the possible rhythmic matches. Our program first finds every possible combination of eighth notes and eighth rests as a binary string. In each binary string a '1' represents a eighth note and a '0' represents a rest. After that, we loop through each possible combination and check if it matches the input given by the user.

Then finally, our program turns the binary strings that matched the rhythmic format into a format that can be written to a LilyPond music file. It then writes out the LilyPond music file containing all the rhythms that match the input string.

Works cited

- Lin, A. (2016, December 01). Generating Music Using Markov Chains. Retrieved from <https://hackernoon.com/generating-music-using-markov-chains-40c3f3f46405>
- Powell, V. (n.d.). Markov Chains explained visually. Retrieved from <http://setosa.io/ev/markov-chains/>
- Soni, D. (2018, March 05). Introduction to Markov Chains. Retrieved from <https://towardsdatascience.com/introduction-to-markov-chains-50da3645a50d>