Calculating Rhythmic Pattern Outcomes Through Matrix Manipulation and Markov Chains

Linear Algebra 2270-004 Semester Project

By: Cameron Yeomans, Emily Burke, Hunter Moffatt, and Spencer Cameron

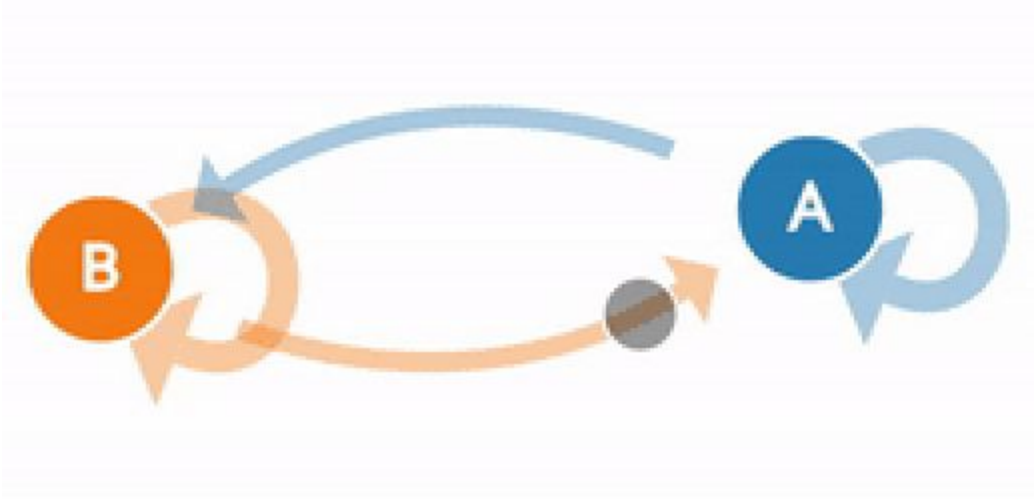
Introduction:

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 ¼ of a bar all the way to 1/32 or even 1/256 of a bar in some cases. The duration of each beat is determined by the time signature of the piece.

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

Relation to Linear Algebra:

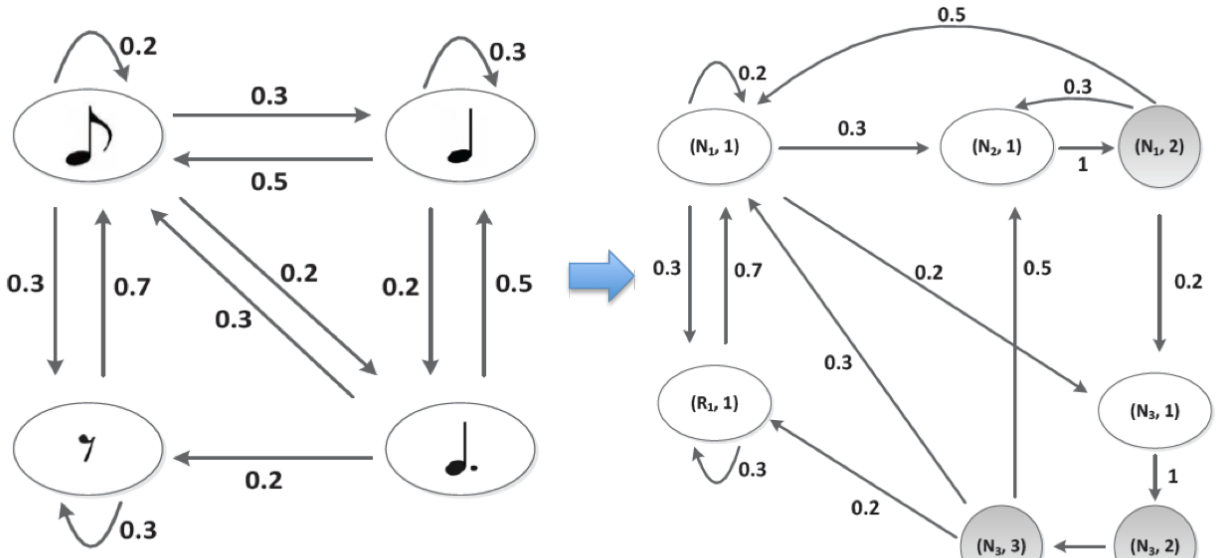
We can represent the output of our program with a Markov chain. Simply put, Markov chains are a set of transitions, which are determined by probability distributions, that must satisfy something called the “Markov property”. The Markov property is a “memoryless property of a stochastic process.”



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.



First-order MC Revised MC

Our program could also be used to produce music from a mathematical approach rather than an artistic one. This could be done by analyzing the Markov chains of popular songs and then using that data to generate a piece based off those chains to create a beat similar to those popular songs. This may seem like a strange approach to creating music, but many electronic music producers such as Aphex Twin produce music using similar methods.

Our Code:

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. A link to its Github repository can be found here: https://github.com/CamYeomans/Rhythm-Analysis.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.

```
3
4   import re
5   regex = "^[\+|\-|\?]+$"
6
7   print("Rhythm builder")
8   print("use + to enter an eigth note")
9   print("use - to enter an eigth 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
```

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.

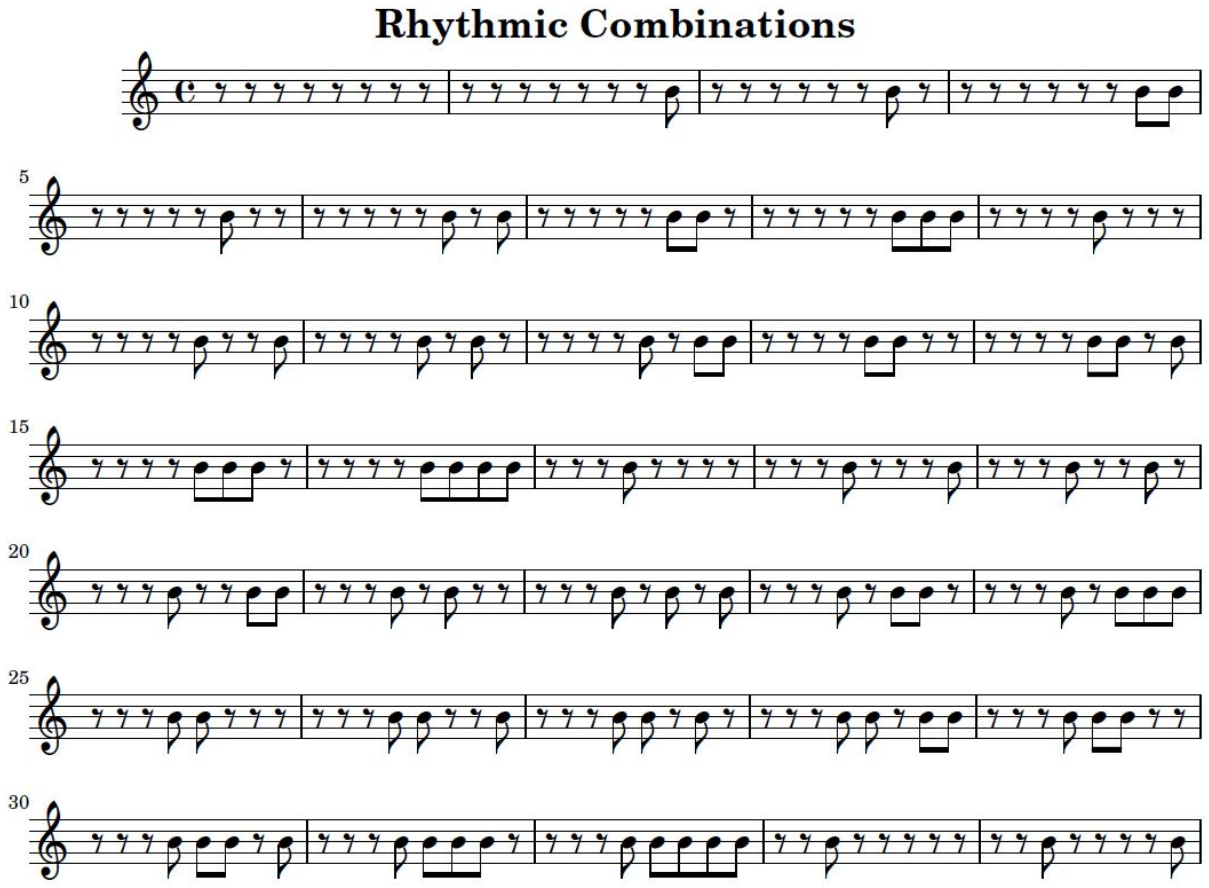
```
rhythmsInBinary = []

binaryVal =""
for x in range(256):
    binaryVal = str(bin(x)[2:])
    while(len(binaryVal)!= 8):
        binaryVal = "0" + binaryVal

    match = True
    for y in range(len(searchRhythm)):
        if(searchRhythm[y] == "?"):
            continue
        elif((searchRhythm[y] == "+" and binaryVal[y] == "0") or (searchRhythm[y] == "-" and binaryVal[y] == "1")):
            match = False
    if(match):
        rhythmsInBinary.append(binaryVal)
    match = True

```

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.

```
rhythmString = ""
lilyPondString = ""
lilyPondArray = []
for i in range(len(rhythmsInBinary)):
    rhythmString = rhythmsInBinary[i]
    for j in range(len(rhythmString)):
        if(rhythmString[j] == "0"):
            lilyPondString = lilyPondString + "r8 "
        else:
            lilyPondString = lilyPondString + "b8 "

    lilyPondArray.append(lilyPondString)
    lilyPondString = ""
print(lilyPondArray[0])


#writes to ly file to be compiled into sheet music
outfile = open('Rhythms.ly','w')

#format syntax
outfile.write("\\header{\n")
outfile.write("\ttitle = \"Rhythmic Combinations\"\n")
outfile.write("}\n")
outfile.write("\\relative c'' {\n")

for i in range(len(lilyPondArray)):
    outfile.write(lilyPondArray[i]+"\n")

outfile.write("}")

```

Some sample output of this program using the LilyPond plugin with eight ‘?’ as an input:

```
Rhythm builder
use + to enter an eigth note
use - to enter an eigth rest
use ? to enter a wildcard
input must be 8 chars or less and contain only the chars + and -
Enter a valid rhythm: ????????
```



One thing that we hoped we could do would be to expand on our existing code to implement different notes of eighth rests/notes. Our code already has a way to examine the rhythm and composition of a given bar, we could just add more parameters to define note types. Once this is done our program could calculate a Markov chain of probabilities for the transition of one specific bar to another.

Bibliography (APA):

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