Algorithmic Composition: The Music of Mathematics

Carlo J. Anselmo ‘18 and Marcus Pendergrass

Department of Mathematics, Hampden-Sydney College, Hampden-Sydney, VA 23943

ABSTRACT
We report on several techniques we have developed for generating musical compositions algorithmically. Most of our techniques are based on our idea of a sequence recursion, which is a method for generating finite integer sequences that can represent pitches and rhythms. Our research is two-pronged: we develop the mathematical properties and techniques associated with sequence recursions, and we apply these techniques to the synthesis of new musical compositions. Sequence recursions use basic musical operations such as repetition, transposition, and inversion to iteratively generate integer sequences of increasing length and complexity. These sequences are then mapped into musical structures such as scale or rhythm patterns to produce melodies and accompaniments. We present several examples of musical compositions produced by this process.

INTRODUCTION
In this paper, we report on several techniques we have developed for generating musical compositions algorithmically. These techniques are based primarily on our idea of a sequence recursion, which is a method for generating finite integer sequences that can represent pitches and rhythms. We call these sequences pitch patterns and rhythm patterns, respectively, and we have developed many operations we can per- form on them. Pitch patterns provide a list of integers that are mapped onto pitches, while rhythm patterns designate the start times for notes. Rhythm patterns also have a corresponding object called a division scheme, which dictates the interval of time a rhythm pattern live on. For both pitch and rhythm, some of the operations we have developed mirror basic musical operations such as repetition, transposition, and inversion when we iteratively generate sequences of increasing length and complexity. However, we also have a number of operations that are unusual in standard musical composition and yield interesting and unexpected results. For both pitches and rhythms, these unusual operations are rotation, reversal, concatenation, and composition, and rhythm patterns additionally use the merge operation. Modern musical notation utilizes notes, which are objects that contain information about
both pitch and duration. In this case, pitch and rhythm are intrinsically locked together, but in our research, information about pitch and rhythm are stored in two separate sequences, and we later tie them both together into one object, which we call a melody/line.

Our research is two-pronged: we develop the mathematical properties and techniques associated with sequence recursions, and we apply these techniques to the synthesis of new musical compositions. Our primary tool for sequence recursions is something we call a substitution scheme, and in our research, we have thoroughly explored the uses and properties of these schemes. Substitution schemes take shorter, simpler pitch and rhythm patterns, and transform them into longer and more complicated sequences. We also present two special cases of substitution schemes, called palindromic and rearrangement schemes.

When synthesizing new compositions, we utilize our algorithmic techniques to generate both melody and harmony. We usually only focus on designing melody, because harmony and accompaniments naturally arise out of the melody we create. We also take into consideration a musician’s needs for sheet music, and we have de- signed a number of methods for performing mathematical tweaks on music to make it more playable, including transposition, modulo, and reflection. We present several examples of musical compositions produced by this process.

This primary mathematical focus of this research is on pitch and its related components. Though we do not delve too deeply into the mathematics of rhythm, we have developed enough theory to at least utilize it in the creation of music [3]. Additionally, the two other primary aspects of music, volume and timbre, are selected arbitrarily—however, we could certainly determine these parameters algorithmically as well.

THEORY AND METHODS
We will first discuss pitch and a mathematical way to model it. Musical pitch refers to the quality of sound governed by the rate of vibrations producing it. In terms of human perception, we are talking about the degree of highness or lowness of a tone. Without getting into too much detail, distinct pitches in musical notation are denoted by the letters A to G. Notes increase in pitch as one would expect, such that a C sounds lower than a D, and so on.

Core to this research is the idea that musical pitches can be represented with integers. For example, we could arbitrarily map the integers 0, 1, and 2 to the notes C, C#, and D, respectively. In order to meet these specifications, we combine strings of integers into an object we have created called a pitch patterns, which we directly transform into sequences of musical notes. We denote the set of all pitch patterns by P = 1n=1 Zn, where Z is the set of all integers. Throughout this paper, we will use letters from the first half of the Greek alphabet to denote pitch patterns.

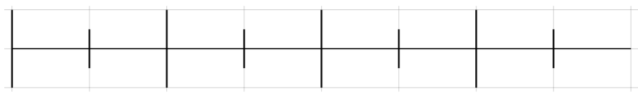
As a side note, an octave is an interval between one musical pitch and another with a frequency that delivers by a power of two. The note that is an octave above a C is still called a C, but we say it is in a higher octave. There are twelve tones between an octave in standard western music, although there are an infinite number of frequencies. In our research, we rely on octaves through our notion of octave equivalence, which we define as the ability to move between different octaves of a pitch without changing the fundamental sound of a musical composition. When faced with a pitch pattern with a large range of integers, we utilize octave equivalence to continue notes as far upwards or downwards as desired.

In addition to pitches, rhythms may also be represented mathematically. In music theory, a beat is the basic unit of time; it is often defined as the numbers a musician would count while performing. A note may last any number of beats, and each note has a name with respect to its duration. In most cases, a whole note refers to a note which lasts four beats. Intuitively, a half note refers to a note with a duration of two beats, a quarter note to one beat, an eighth note to a half a beat, and so on. A measure is a segment of time corresponding to a specific number of beats. Dividing music into measures provides regularity in a musical composition, a regularity that helps analyze music in a more mathematical fashion. A time signature defines how many beats occur in a measure, as well as which note receives the beat. One final aspect of interest for rhythms is the accent, which describes a particular emphasis placed on a note. Generally, greater emphasis is placed on higher-level beats, while greater subdivisions receive a decreasing amount of emphasis.

Though our research does not heavily focus on the mathematical representation of rhythm, we still work with the conceptual properties of rhythm and explore methods for generating new rhythms.

A division scheme is a division of an interval into d`1 equal subintervals, each of which is subdivided into d`2 equal subintervals, and so on, concluding with a final subdivision of each interval into d0 equal subintervals. Division schemes take the form D = (d`1 d`2 . . . d1 d0), where ` is the number of levels in the scheme, and the value of di is the number of subdivisions at level i. Beats are numbered from zero. We typically interpret the levels in divisions schemes as how much emphasis a particular beat receives, with the leftmost level receiving the most emphasis, and the rightmost level receiving the least.

Division schemes are particularly useful because they mirror time signatures in standard Western music notation. For example, the division scheme D = (4 2) represents a division of an interval into four major pieces, each of which is also divided into two pieces. A measure of 4/4 time in Western music has this structure. Note that in this case, we have four major beats, and each has eighth-note resolution. This division scheme can be diagrammed as follows.



As with pitch patterns, we have developed a way to organize rhythms systematically. We define a rhythm pattern as a sequence of strictly increasing positive integers that represent the starting times for a sequence of notes. Unless we specify duration, one note is played up until the starting time of a new note. We denote rhythm patterns with letters from the second half of the Greek alphabet. An example of a perfectly reasonable rhythm pattern is (0 1 2 3), which signifies four notes held for equal durations. The purpose of divisions schemes is to contextualize rhythm patterns in an interval of time, and they are particularly useful at illustrating more complicated rhythm patterns. Division schemes also inform us when the last note of a rhythm pattern ends.

We also have a number of operations we can perform on rhythm patterns. Though we have not developed the mathematical notation for rhythm, we can still accurately describe how each operation occurs. Due to the nature of rhythm patterns, we do not have transposition and inversion operations. Therefore, our first unary operation is rotation. Like with pitch patterns, rotation is circular. For example, the right rotation of the rhythm pattern (1 2 5) with division scheme (2 3) is (0 2 3).

When we put pitch and rhythm together, the result is a string of notes that we perceive as a melody. Similarly, when we lay pitch and rhythm patterns side by side, we have sequences of parameters that can ultimately be transformed into a musical melody. In our research, we refer to the combination of a compatible pitch and rhythm pattern as a line. When we say compatible, we mean that both the pitch and rhythm pattern have the same an equal number of elements so that the resulting musical notes that are created are assigned both pitch and rhythm. This should make sense, because if the length of a given pitch and rhythm pattern mis-match, some notes with either not be assigned a pitch, or in the opposite case, some notes will not have a rhythm.

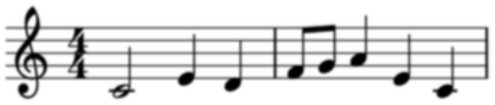
As an example, consider the following set of parameters:

Division scheme: (2 4 2)

Rhythm pattern: (0 4 6 8 9 10 12 14)

Pitch pattern: (0 2 1 3 4 5 2 0) with pitches assigned to notes on the C-Major scale

With these parameters, we create the following melody:



Here, we briefly note that pitch and rhythm patterns are designed in such a way so that any operations that both patterns chare can be applied to both at the same time without causing any problems. This include the unary operations of rotation and reversal and the binary operations concatenation and composition. Additionally, we can perform as many operations on pitch as we desire without worrying about a potential mismatch in the length of the corresponding rhythm pattern. However, unary rhythm operations that alter the number of events in a pattern (e.g. complementation) cannot be applied to a line, because there is no natural way to assign pitches to the altered rhythms.

The foundation for algorithmic musical composition can be approached any number of ways. Traditionally, we begin by building a substitution scheme that is entirely arbitrary and hopefully interesting. We may have an idea of the shape we want a piece to take, but more often, we just design a substitution scheme that is mathematically fascinating, and we let the outcome surprise us. Additionally, while creating our substitution schemes, we want to keep in mind which parameters we will utilize. For example, we may want a scheme that merely transposes and rearranges our input, or we could create a scheme that additionally rotates and inverts our input.

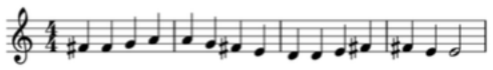
When considering the operations, we have outlined above, we may also decide whether we want each operation to apply to a motif as a whole, or instead to our pitch and rhythm patterns independently. For example, we usually rotate a motif by thinking of it as a sequence of notes, but we could by all means rotate pitch and rhythm independently and then reconstruct our motifs afterwards. The arrangements of operations in our substitution schemes are only limited by the imagination of the composer, and these substitution schemes determine the large-scale structure of a composition.

After we have outlined the operations we wish to perform, we then design a number of motifs to input into our algorithm. How these motifs are designed ultimately determines the texture of a composition, and through these motifs, we may hear a composer's own style emerge. We may also instead choose to mathematically represent a well-known motif, and then mutate said motif with our substitution schemes. For example, we could represent a simplified version of Beethoven's "Ode to Joy

"

the

me



as follows:

We don't often design motifs with harmony in mind, because harmony naturally arises when multiple motifs are played together. When creating motifs, we also tend to create pairs of faster moving parts with higher pitches, and pairs of slower moving parts with lower pitches. Afterwards, we assign a musical instrument to each line in the output—generally, we assign melodic instruments (like the violin) to higher and quicker parts, and bass instruments (like the electric bass or tuba) lower and slower parts.

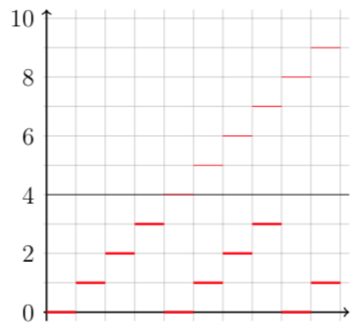
If we want a piece to be played purely through software, we often don't make any final tweaks to our music because software can handle any musical oddity thrown at it. However, if we want our music to be played by a musician on a traditional musical instrument, we often need to make adjustments to the sheet music to meet a musician's demands, particularly when it comes to pitch. We could of course just arbitrarily revise sheet music on a whim, but these revisions would ultimately dis-qualify a piece as a pure algorithmic composition. Therefore, we instead present a number of ways to mathematically tweak music so that the process remains algo- rithmic as well as transparent.

Another aspect of our algorithmic process to consider is that oftentimes, itera- tively generated music ends abruptly. As a result, we may decide to concatenate a final measure or two of new music onto the end of a composition to give it a greater sense of finality.

One of the more obvious problems that occurs after many iterations of a substitution scheme is that all of the elements in the resulting pitch patterns may be rather small or large, resulting in very low or high pitches. If we were to assign one of these parts to a musical instrument, we may want to transpose it up or down to meet a player's needs, often by utilizing octave equivalence. With multiple instruments playing a piece, if too many instruments are playing in the same frequency range, the piece may often sound chaotic and muddy, and will often be quite dissonant. We may reduce this cluttering by transposing lines apart.

Another problem that potentially arises after many iterations of a recursively gen- erated pitch pattern is that the di↵erence between the smallest and largest integers can become quite large. When converted to pitches, extreme values from a pitch pattern in either direction create a variety of problems. In the worst case, a pitch may be so extreme that it falls outside the range of human hearing, or it may reach a point that is at least not pleasing to listen to. For most musical instruments that can only play a limited number of octaves, extreme pitches become unwieldy or even unplayable. With these problems in mind, we utilize a couple of methods to resolve these issues in a mathematically satisfying fashion.

One operation that resolves this problem is the modulo operation. The modulo operation returns the remainder of the division of two integers, and this operation is particularly useful because it allows us to set an upper limit on the elements in a pitch pattern. In an e↵ort to avoid working with the modulo and negative numbers, we usually perform a transposition beforehand. For example, if we want a particular pitch pattern to only have the integers 0 to 3 in it, then we would take the modulo-4 of each number in the pattern.



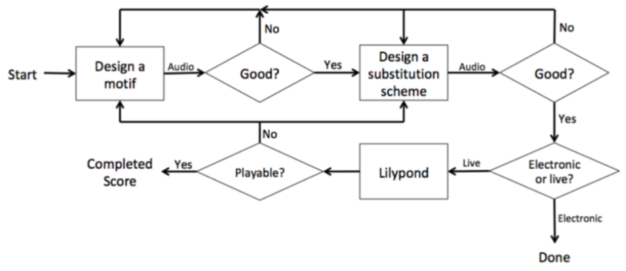
To avoid changing the overall texture of a composition, we usually set us modulo to a pitch that is exactly an octave or two above the base note so that pitches that get moved down simply move down a number of octaves. Then, when pitches become undesirably high, the modulo takes care of this by shifting them all down an octave to keep them manageable.

Our primary concern with this option is that if we set the modulo too low, our notes do not vary too much, and a composition may potentially become monotonous over time. However, if we go in the opposite direction and set the modulo too high, we perceive very obvious jumps in pitch from very high to very low notes. This may be desirable in certain situations, but it is often not. Therefore, we more often utilize the following method.

We prefer using reflections in favor of a mod during composition, because the reflection does not produce any large gaps between pitches at the bounds, regard- less of how large we set the range of pitches allowed. Additionally, the reflection can handle negative integers without any issues. One may reasonably argue that reflection of a pitch pattern is problematic because it may not accurately preserve the texture of a given pitch pattern. Of course, we mainly use a reflection to ensure a melody is playable by a musician, so if accuracy is ever a question, one can opt to instead have computer software produce the music without making any tweaks.

As mentioned before, iteratively generated music often ends abruptly. As a result, we often create very simple endings for our pieces, either by concatenating on an additional measure consisting of a single note for each part, or by constructing a short ending that aligns with the rest of the composition. Either way, creating an ending is a purely creative endeavor, and our primary goal is to tie together a composition in a musically satisfying way.

In our research, we have thoroughly designed and implemented our own code in Matlab and Java to build motifs and design substitution schemes. We use Matlab because it works especially well with matrices and iteration. Matlab also allows the audio output of any pitch pattern that is outputted from a substitution scheme and/or tweaked. If we want to create musical notation of our music for a musician to perform, we use another function we have created to transform numerical output into code readable by Lilypond, a separate programming language used purely for western music notation. The process diagrammed below is one of many ways



to create a composition.

CONCLUSION

We have reported on several techniques we have developed for generating musical compositions algorithmically. We have used sequences of integers called pitch pat- terns and rhythm patterns and their respective operations to generate list of integers that, through iteration, are increasingly long and complex. The primary tool that we have used to iteratively generate these patterns is the substitution scheme, which take shorter, simpler pitch and rhythm patterns, and transform them into longer and more complicated sequences. We have also presented two special cases of substitution schemes. We ultimately use the integers from pitch and rhythm patterns to create musical melodies.

When synthesizing new compositions, we design a number of substitution schemes and short motifs, and our process handles the rest of the details. We have also taken into consideration a musician's needs for sheet music, and we have designed a number of methods for performing mathematical tweaks on music to make it more playable. Finally, we present several examples of musical compositions produced by this process. Future work includes analyzing existing compositions by popular composers to see if they even loosely follow our algorithmically techniques. From our minimal exploration of this area, it seems as though musical compositions rarely follow our precise methods of generating music, but it is worth noting that at least the shape of many different types of music can be expressed using our generative techniques. For example, the majority Prelude in C from The Well-Tempered Clavier by J.S. Bach follows a basic (1 2 3 4 5 3 4 5) scheme until the very last few measures. Parts of other pieces, like the introduction Beethoven's 5th, also follow schemes with definitive patterns, in this case following a (0-1-2-3) scheme.

Other topics that we would like to explore would be a deeper mathematical exploration of rhythm patterns and division schemes. We would also like to implement a number of more complicated methods for generating music, such as "triggers" that alter the music after observing a certain sequence of elements in pitch and rhythm schemes. Finally, we are interested in designing an app that lets the user enter their own motifs, choose a set of operations, and then listen to the output.

REFERENCES

[1] T. Johnson, Self-Similar Melodies, Editions 75, 1996.

[2] M. Pendergrass, Patterns and Schemes for Algorithmic Composition, preprint, 2016.

[3] M. Pendergrass, Musical Structures and Operations, preprint, 2018.

[4] D. C. Lay, Linear Algebra and Its Application, Addison Wesley, 2003.