Algorithmic Composition:
The Music of Mathematics

Carlo Anselmo
AnselmoC18@hsc.edu

May 8, 2018

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.

1 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 perform 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 lives 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 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 designed 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.

2 Theory and Methods

2.1 Pitch

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.

2.1.1 Pitch Patterns

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 $\mathbb{P} = \bigcup_{n=1}^{\infty} \mathbb{Z}^n$, where $\mathbb{Z}$ 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 differs 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.

2.1.2 Pitch Pattern Operations

We outline a number of useful operations we have created in order to manipulate pitch patterns. The first is **transposition**. The transposition of a pitch pattern α of size m by $k \in \mathbb{Z}$ scale steps is denoted by $k+\alpha$, and the addition is component-wise:

$$k + \alpha = (k + \alpha_0, k + \alpha_1, \ldots, k + \alpha_{m-1}).$$

The second is **inversion**. Inversion of α about 0 is given by $-\alpha$, while the inversion about k is $2k - \alpha$.

Third is **rotation**. Rotation is circular, and we denote the rotation of α as $\sigma^k\alpha$, where a positive k represents the number of steps α is rotated to the right, and a negative k signals a left rotation of k steps.

The fourth operation—**reversal**—is denoted by $\overline{\alpha}$, and reversing the order of the elements in α gives

$$\overline{\alpha} = (\alpha_{m-1}, \alpha_{m-2}, \ldots, \alpha_0).$$

Additionally, we utilize two binary operations for pitch patterns, the first being concatenation. Given pitch patterns α and β of length m and n, respectively, the **concatenation** of α followed by β is defined by

$$(\alpha, \beta) = (\alpha_0 \;\; \alpha_1 \;\ldots\; \alpha_{m-1},\; \beta_0 \;\; \beta_1 \;\ldots\; \beta_{n-1}).$$

The second is composition. Let α and β be pitch patterns with lengths m and n, respectively. The **composition** $\alpha\beta$ is the pitch pattern of length mn given by

$$\alpha\beta = (\alpha_0 + \beta,\; \alpha_1 + \beta,\; \ldots, \alpha_{m-1} + \beta)\,.$$

For example, let $\alpha = (2\ 0\ 1)$ and $\beta = (1\ 0)$. We may represent these patterns using the following step diagrams.

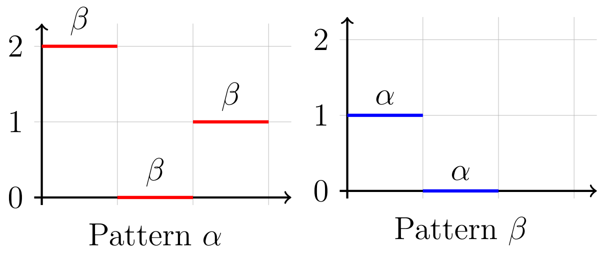


Figure 1: Step diagrams of α and β.

Then

$$\begin{aligned}\alpha\beta &= (\alpha_0 + \beta, \alpha_1 + \beta, \alpha_2 + \beta) \\ &= (2 + (1\ 0), 0 + (1\ 0), 1 + (1\ 0)) \\ &= (3\ 2\ 1\ 0\ 2\ 1)\end{aligned}$$

and

$$\begin{aligned}\beta\alpha &= (\beta_0 + \alpha, \beta_1 + \alpha) \\ &= (1 + (2\ 0\ 1), 0 + (2\ 0\ 1)) \\ &= (3\ 1\ 2\ 2\ 0\ 1).\end{aligned}$$

Note that $\alpha\beta \neq \beta\alpha$. The composition of pitch patterns is often not commutative, meaning $\alpha\beta \neq \beta\alpha$. However, it is associative, such that $(\alpha\beta)\gamma = \alpha(\beta\gamma)$ [2]. We call the pitch pattern (0) the identity element for composition, since $(0)\beta = \beta(0) = \beta$ for all pitch patterns β.

Additionally, note that powers of composition take the following form:

$$\begin{aligned}\alpha &= (\alpha_0,\ \alpha_1, \ldots,\ \alpha_{m-1}) \\ \alpha\alpha &= (\alpha_0 + \alpha,\ \alpha_1 + \alpha, \ldots,\ \alpha_{m-1} + \alpha) \\ \alpha^3 &= ((\alpha\alpha)_0 + \alpha,\ (\alpha\alpha)_1 + \alpha, \ldots,\ (\alpha\alpha)_{m^2-1} + \alpha) \\ &= (\alpha_0^2 + \alpha,\ \alpha_1^2 + \alpha, \ldots,\ \alpha_{m^2-1}^2 + \alpha) \\ \alpha^{n+1} &= \left(\alpha_0^n + \alpha,\ \alpha_1^n + \alpha, \ldots,\ \alpha_{m^n-1}^n + \alpha\right)\end{aligned}$$

The following theorem highlights some properties of composition.

Theorem 2.1. *Let α and β be pitch patterns, and let c, d, j, k, and n be integral constants. Then the following properties of compositions hold:*

1. $(k + \alpha)(j + \beta) = (k + j) + \alpha\beta$.

2. $(k + \alpha)^n = kn + \alpha^n$.

3. $(c\alpha)^n = c\alpha^n$.

4. $(c\alpha)(j+\beta) = j + c\alpha\beta$.

5. $(k+\alpha)(d\beta) = k + d\alpha\beta$.

Proof.

1. We have

$$\begin{aligned}(k+\alpha)(j+\beta) &= ((k+\alpha_0)+(j+\beta),(k+\alpha_1)+(j+\beta),\dots,(k+\alpha_{m-1})+(j+\beta))\\ &= ((k+j)+(\alpha_0+\beta),(k+j)+(\alpha_1+\beta),\dots,(k+j)+(\alpha_{m-1}+\beta))\\ &= (k+j)+(\alpha_0+\beta,\alpha_1+\beta,\dots,\alpha_{m-1}+\beta)\\ &= (k+j)+\alpha\beta.\end{aligned}$$

2. (Base) When $n=1$, $(k+\alpha)^1 = k+\alpha = k(1)+\alpha$.
(Inductive step) We assume $(k+\alpha)^p = kp+\alpha^p$ for some p, and we need to show that $(k+\alpha)^{p+1} = k(p+1)+\alpha^{p+1}$ for all p. Then

$$\begin{aligned}(k+\alpha)^{p+1} &= (k+\alpha)(k+\alpha)^p\\ &= (k+\alpha)(kp+\alpha^p) && \text{(By the inductive assumption)}\\ &= k(p+1)+\alpha^{p+1}. && \text{(By } \mathbf{1.}\text{)}\end{aligned}$$

3. (Base) When $n=1$, $(c\alpha)^1 = c\alpha$.
(Inductive step) We assume $(c\alpha)^p = c\alpha^p$ for some p, and we need to show that $(c\alpha)^{p+1} = c\alpha^{p+1}$ for all p. Then

$$\begin{aligned}(c\alpha)^{p+1} &= \big((c\alpha)^p_0+c\alpha,(c\alpha)^p_1+c\alpha,\dots,(c\alpha)^p_{m^p-1}+c\alpha\big) && \text{(By the note above)}\\ &= \big(c\alpha^p_0+c\alpha,c\alpha^p_1+c\alpha,\dots,c\alpha^p_{m^p-1}+c\alpha\big) && \text{(By the inductive assumption)}\\ &= \big(c(\alpha^p_0+\alpha),c(\alpha^p_1+\alpha),\dots,c(\alpha^p_{m^p-1}+\alpha)\big)\\ &= c\left(\alpha^p_0+\alpha,\alpha^p_1+\alpha,\dots,\alpha^p_{m^p-1}+\alpha\right)\\ &= c\alpha^{p+1}. && \text{(By the note above)}\end{aligned}$$

4. Note that

$$\begin{aligned}(c\alpha)(j+\beta) &= ((c\alpha_0)+(j+\beta),(c\alpha_1)+(j+\beta),\dots,(c\alpha_{m-1})+(j+\beta))\\ &= j+(c\alpha_0+\beta,c\alpha_1+\beta,\dots,c\alpha_{m-1}+\beta)\\ &= j+c\alpha\beta.\end{aligned}$$

5. Observe that

$$\begin{aligned}(k+\alpha)(d\beta) &= ((k+\alpha_0)+(d\beta),(k+\alpha_1)+(d\beta),\dots,(k+\alpha_{m-1})+(d\beta))\\ &= k+(\alpha_0+d\beta,\alpha_1+d\beta,\dots,\alpha_{m-1}+d\beta)\\ &= k+d\alpha\beta.\end{aligned}$$

■

This theorem is by no means exhaustive, but for our purposes, we will utilize only the operations given above when generating pitch patterns.

2.2 Rhythm

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.

2.2.1 Division Schemes

We first define our method of subdividing time into regular intervals, following [3]. A **division scheme** is a division of an interval into $d_{\ell-1}$ equal subintervals, each of which is subdivided into $d_{\ell-2}$ equal subintervals, and so on, concluding with a final subdivision of each interval into d_0 equal subintervals. Division schemes take the form $D = (d_{\ell-1}\ d_{\ell-2} \ldots\ d_1\ d_0)$, where ℓ is the number of levels in the scheme, and the value of d_i 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.

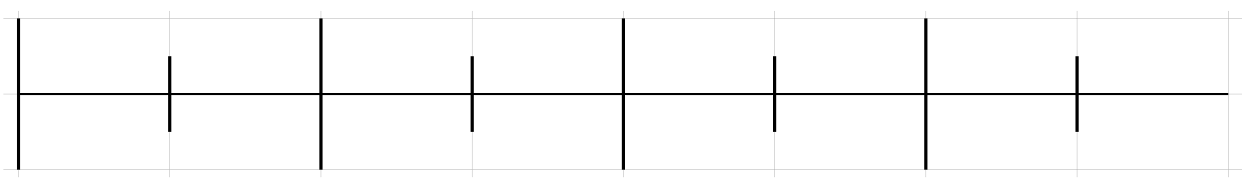


Figure 2: Graphical representation of $D = (4\ 2)$.

If we were to add another level to our scheme so that we would have the new scheme $D = (4\ 4\ 2)$, this would represent four measures of 4/4 time.

2.2.2 Rhythm Patterns

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 are 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.

For example, the rhythm pattern $\rho = (0\ 2\ 3\ 5)$ again represents four notes, although now the notes occur at irregular intervals. Using a division scheme $D = (2\ 3)$, we define the relationship between notes and illustrate them with the following diagram.

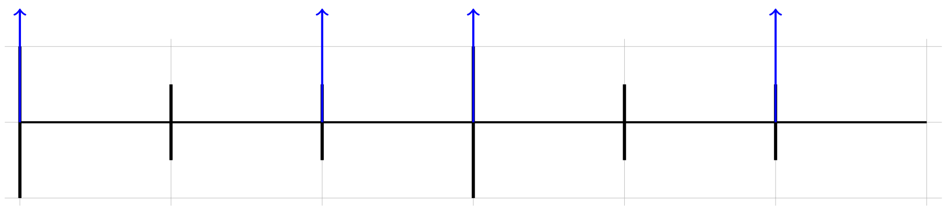


Figure 3: The rhythm pattern (0 2 3 5) with division scheme (2 3).

Here, we say that the first and third notes are each held for two beats, while the second and fourth note are held for one beat each.

Of course, we could also separately assign durations to a given rhythm pattern. Take, for example, the rhythm pattern $\tau = (0\ 3\ 5\ 7)$ with division scheme $D = (4\ 2)$ and durations (2 1 2 1).

Figure 4: The rhythm pattern $\tau = (0\ 3\ 5\ 7)$ with durations (2 1 2 1).

2.2.3 Rhythm Pattern Operations

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).

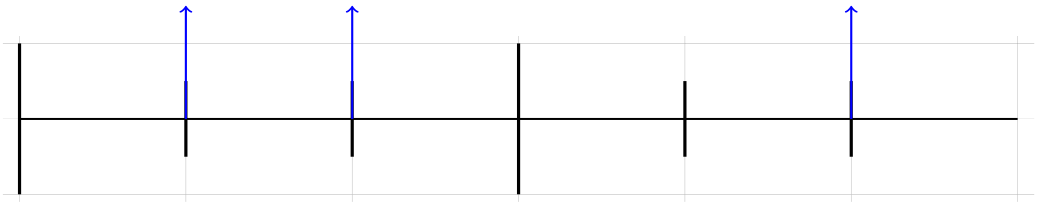


Figure 5a. The rhythm pattern (1 2 5) with division scheme (2 3)

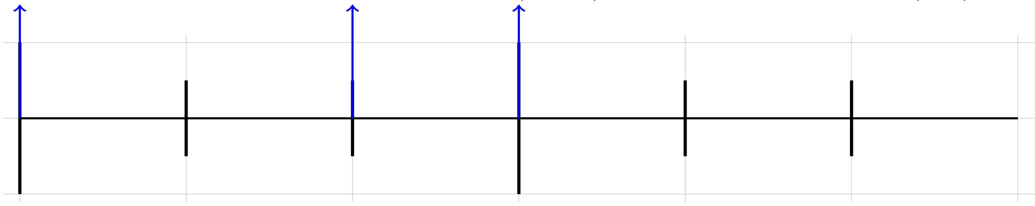


Figure 5b. The resulting rhythm pattern, (0 2 3)

Figure 5. The rotation of a rhythm pattern to the right once.

The next operation is **reversal**, which involves taking a rhythm pattern and writing it in reverse. Note that the reversal operation depends on both the start times and durations of a rhythm pattern.

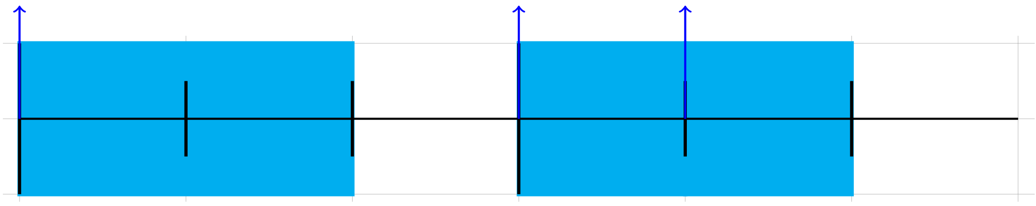


Figure 6a. Rhythm pattern (0 3 4) with durations (2 1 1)

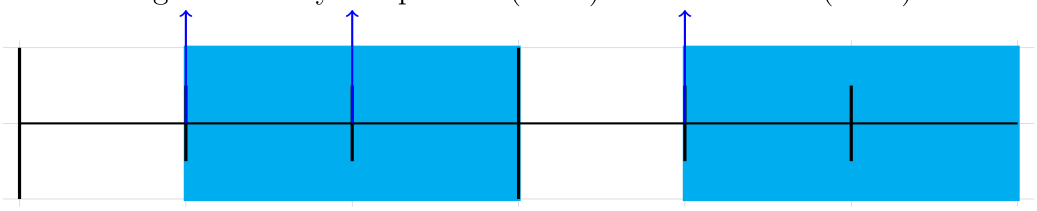


Figure 6b. The reversal of (0 3 4), which is (1 2 4)

Figure 6. The reversal of the rhythm pattern (0 3 4).

The next operation is the **complement**, which involves filling in the "gaps" in an existing rhythm pattern with notes, and removing all prior notes. Unlike reversal, the complement operation depends only on the start times of the rhythm pattern.

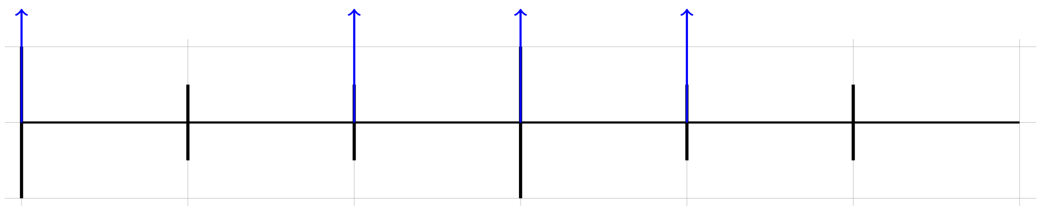


Figure 7a. Rhythm pattern (0 2 3 4) with division scheme (2 3)

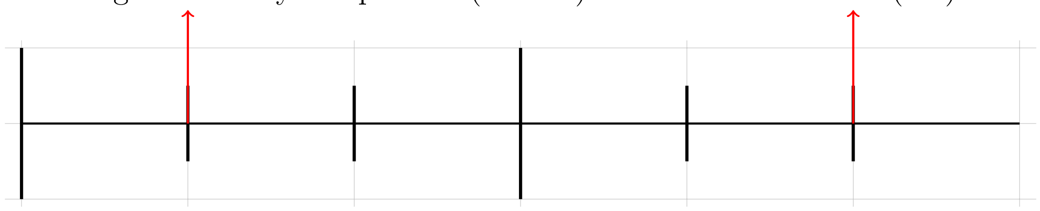


Figure 7b. The complement of (0 2 3 4), which is (1 5)

Figure 7. The complement of the rhythm pattern (0 2 3 4).

Along with the above operations, we also have a number of binary operations. The first is **concatenation**, which functions identically to the concatenation of pitch patterns.

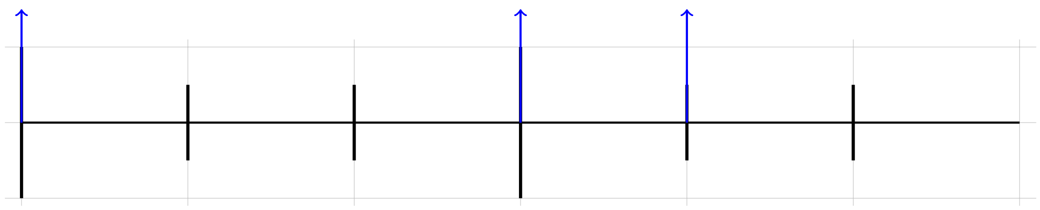


Figure 8a. Rhythm pattern $\tau = (0\ 3\ 4)$ with division scheme (2 3)

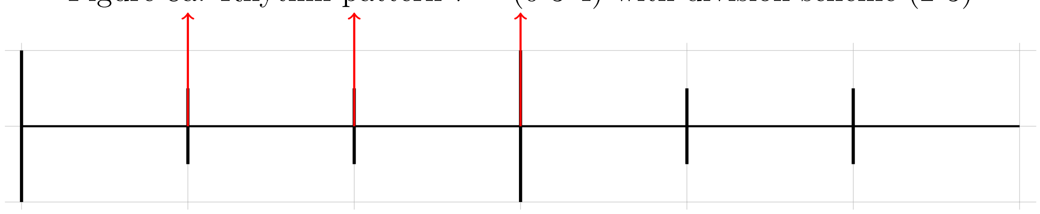


Figure 8b. Rhythm pattern $\rho = (1\ 2\ 3)$ with division scheme (2 3)

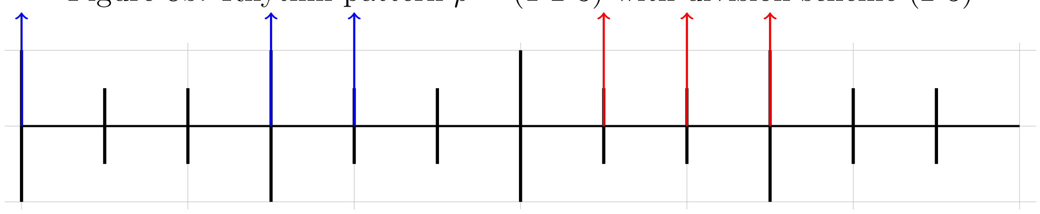


Figure 8c. The result of concatenation, with new division scheme (2 2 3)

Figure 8. The concatenation of two rhythm patterns.

We also have **composition**. As with pitch patterns, composition is a little less straightforward, but we can imagine that the composition of a rhythm pattern with itself involves stretching out said rhythm pattern, then copying this rhythm pattern into each of it's own locations. This rule can be generalized to the composition of any two different rhythm patterns.

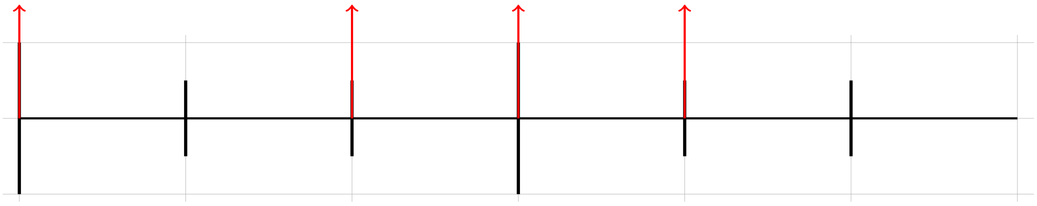


Figure 9a. Rhythm pattern $\tau = (0\ 2\ 3\ 4)$ with division scheme (2 3)

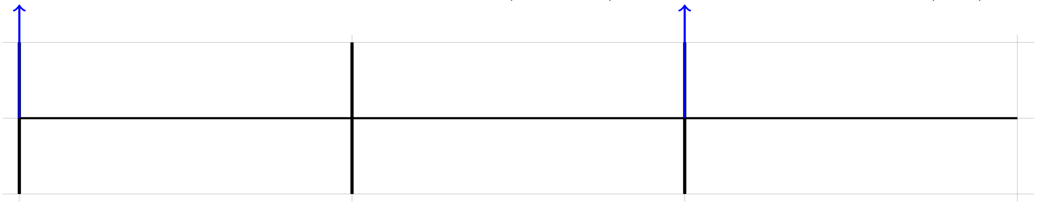


Figure 9b. Rhythm pattern $\rho = (0\ 2)$, with division scheme (3)

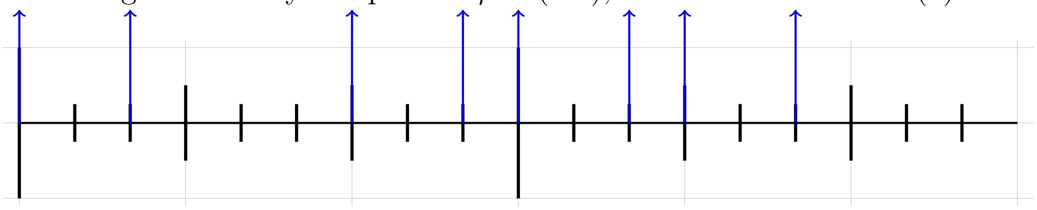


Figure 9c. The composition of the τ and ρ, with division scheme (2 3 3)

Figure 9. The composition of two rhythm patterns.

Finally, we have the merge operation. This operation can be thought of as overlaying one rhythm pattern on top of another, and the resulting rhythm pattern has beats at all locations of its two component rhythm pattern.

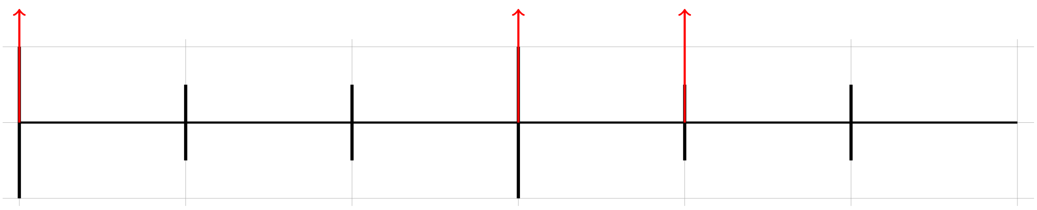


Figure 10a. Rhythm pattern (0 3 4) with division scheme (2 3)

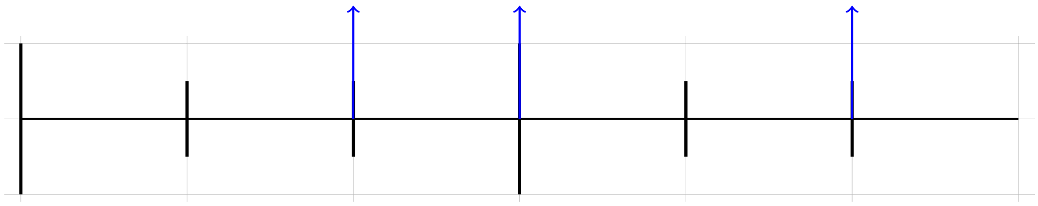


Figure 10b. Rhythm pattern (2 3 5), with division scheme (2 3)

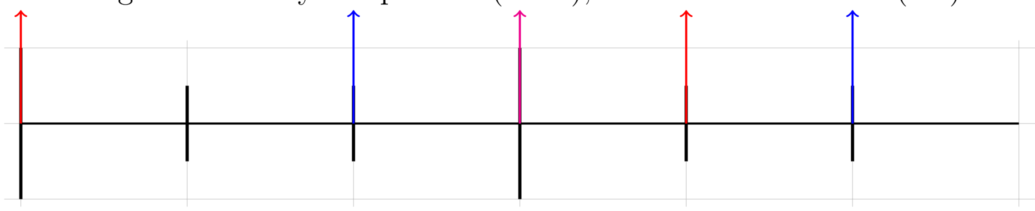


Figure 10c. The merged rhythm pattern, (0 2 3 4 5).

Figure 10. The merging of two rhythm patterns.

2.3 Notating Melody (The Line)

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 mismatch, 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:

2.3.1 Melody Operations

Here, we briefly note that pitch and rhythm patterns are designed in such a way so that any operations that both patterns share 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.

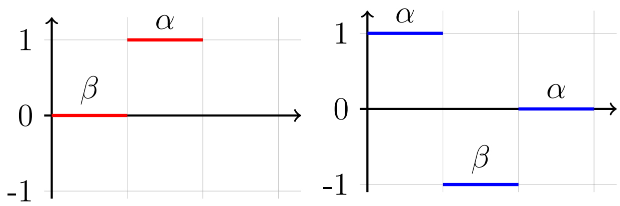
2.4 Substitution Schemes

Substitution schemes are a generalization of composition and the major focus of our research. Though they apply to both pitch and rhythm patterns, this document will only cover substitution schemes with regards to pitch. They are best demonstrated through example.

For instance, let $\alpha = (0\ 1)$ and $\beta = (1\ -1\ 0)$, and define the substitution scheme

$$\begin{aligned} \alpha^{(n+1)} &= \left(\beta^{(n)}, 1+\alpha^{(n)}\right) \\ \beta^{(n+1)} &= \left(1+\alpha^{(n)}, -1+\beta^{(n)}, \alpha^{(n)}\right) \\ \alpha^{(0)} &= \beta^{(0)} = (0) \end{aligned}$$

represented by the following diagrams:



Pattern α Pattern β

Then

$$\begin{aligned} \alpha^{(1)} &= (0\ 1) \\ \alpha^{(2)} &= (1\ -1\ 0\ 1\ 2) \\ \alpha^{(3)} &= (1\ 2\ 0\ -2\ -1\ 0\ 1\ 2\ 0\ 1\ 2\ 3) \end{aligned}$$

and

$$\begin{aligned} \beta^{(1)} &= (1\ -1\ 0) \\ \beta^{(2)} &= (1\ 2\ 0\ -2\ -1\ 0\ 1) \\ \beta^{(3)} &= (2\ 0\ 1\ 2\ 3\ 0\ 1\ -1\ -3\ -2\ -1\ 0\ 1\ -1\ 0\ 1\ 2). \end{aligned}$$

Mathematically, a substitution scheme is simply a mapping from $\mathbb{P}^n$ into itself, where $\mathbb{P}$ is the set of all pitch patterns, and n is a positive integer. Thus, a substitution scheme transforms a vector of n pitch patterns into another vector of n pitch patterns. In this research we have focused on substitution schemes that utilize the basic unary and binary pitch operations defined above. By iterating these schemes we can generate pitch patterns that exhibit interesting structure, both from a musical and a mathematical point of view.

We will now analyze a couple different aspects of substitution schemes. We will first demonstrate how one can solve for the length of a substitution scheme, and then we will prove how to solve for any element of a substitution scheme.

2.4.1 Solving for Length

We can use linear algebra techniques to solve for the lengths of the sequences produced by substitution schemes. Specifically, we can create a matrix representing the number of pitch patterns involved in each row of a substitution scheme, find the eigenvalues of this matrix, and ultimately use diagonalization (see for instance [4]) to solve for the length of a pitch pattern from any iteration of a substitution scheme.

Consider the substitution scheme

$$\begin{aligned} \alpha^{n+1} &= (\alpha^n,\ 2-\beta^n,\ -1+\alpha^n,\ \beta^n) \\ \beta^{n+1} &= (\alpha^n) \\ \alpha^0 = (1\ 3\ 2\ 0) &\qquad \beta^0 = (2\ 0\ 1\ 1). \end{aligned}$$

Then the lengths a_i and b_i of α and β, respectively, of each iterate i of this substitution scheme are

$$\begin{aligned} a_{n+1} &= 2a_n + 2b_n \\ b_{n+1} &= a_n \\ a_0 = 4 &\qquad b_0 = 4. \end{aligned}$$

We may represent this system of equations using matrices.

$$v_{n+1} = \begin{bmatrix} a_{n+1} \\ b_{n+1} \end{bmatrix} = \begin{bmatrix} 2 & 2 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} a_n \\ b_n \end{bmatrix} = Av_n,$$

where $v_0 = \begin{bmatrix} a_0 \\ b_0 \end{bmatrix} = \begin{bmatrix} 4 \\ 4 \end{bmatrix}$. It follows that

$$\begin{aligned} v_n &= Av_{n-1} \\ &= A^2 v_{n-2} \\ &= \dots \\ &= A^n v_0 \end{aligned}$$

for $k \geq 1$. But A is diagonalizable, with $A = PDP^{-1}$, where

$$P = \begin{bmatrix} 1+\sqrt{3} & 1-\sqrt{3} \\ 1 & 1 \end{bmatrix}$$

and

$$D = \begin{bmatrix} 1+\sqrt{3} & 0 \\ 0 & 1-\sqrt{3} \end{bmatrix}.$$

It follows that $A^n = PDP^{-1}$, and we find that

$$a^n = \frac{2\sqrt{3}}{3}\left((1+\sqrt{3})^{n+1} - (1-\sqrt{3})^{n+1} - (1+\sqrt{3})^{n+1}(1-\sqrt{3}) + (1+\sqrt{3})(1-\sqrt{3})^{n+1}\right),$$
$$b^n = \frac{2\sqrt{3}}{3}\left((1+\sqrt{3})^{n} - (1-\sqrt{3})^{n} - (1+\sqrt{3})^{n}(1-\sqrt{3}) + (1+\sqrt{3})(1-\sqrt{3})^{n}\right).$$

Note that the sequence lengths of any substitution schemes of this kind may be analyzed in this way.

2.4.2 Solving for Elements

We will now discuss methods for solving for specific elements of the pitch patterns generated by certain substitution schemes. We begin with perhaps the simplest kind of substitution scheme, iterated composition.

Theorem 2.2. *Let α be a pitch pattern with length $m \geq 2$, and let n be a positive integer. Then for $0 \leq i < m^n$, we have*

$$\alpha_i^n = \sum_{k=0}^{m-1} \alpha_{i(k)}$$

where $i(k)$ is the k^{th} digit in the n-digit base-m expansion of i, including leading zeroes.

Proof. Let α and β be pitch patterns of length m and n respectively, and let $0 \leq i \leq mn-1$. (This n is not the same as the n in the statement of the theorem.) I claim that the i^{th} entry of the composition $\alpha\beta$ is

$$(\alpha\beta)_i = \alpha_q + \beta_r$$

where q and r are the quotient and remainder when i is divided by n. To see this fact, note that by the definition of composition,

$$\alpha\beta = (\alpha_0 + \beta, \alpha_1 + \beta, ..., \alpha_{m-1} + \beta).$$

This composition contains m blocks which consist of copies of β transposed by an amount specified by the entries in α. These blocks have length n and are concatenated together to form $\alpha\beta$ with length mn. Let b be the block number (indexed at zero) that contains the i^{th} entry. Then $b = q$, where q is the quotient when i is divided by n. Since this block contains the indices qn to $(q+1)n-1$, the position of the i^{th} entry is given by $qn \leq i \leq qn + (n-1)$. Let $\rho = i - qn$. Then $0 \leq \rho \leq n-1$. Since $0 \leq r \leq n-1$ where r is the remainder when i is divided by n, then by uniqueness, $\rho = r$. Therefore, to find the i^{th} position in the composition $\alpha\beta$, we location the q^{th} block and count r units into it. In other words,

$$(\alpha\beta)_i = \alpha_q + \beta_r,$$

completing the proof of the claim.

Now, to prove theorem 1, assume that α has length $m \geq 2$. We induct on the exponent in α^n. To prove the theorem is true for $n = 1$, we must show that

$$\alpha_i^1 = \sum_{k=0}^{0} \alpha_{i(k)}.$$

Clearly the left side equals α_i. The right hand side equals

$$\sum_{k=0}^{0} \alpha_{i(k)} = \alpha_{i(0)}.$$

However, $i(0) = i$ since $0 \leq i < m$. Therefore, $\alpha_{i(0)} = \alpha_i$. Now that the base case is established, we will move on to the inductive case.

We will assume that the theorem holds for some positive integer n, and consider α^{n+1}. By claim 0, we have

$$\alpha_i^{n+1} = (\alpha^n \alpha)_i = \alpha_q^n + \alpha_r$$

where q and r are the quotient and remainder when i is divided by m. Thus,

$$i = qm + r$$

where $0 \leq r < m$.

I now claim that r is the lowest-order digit in the base-m expansion of i. Let the base-m expansion of i be

$$i = i_\ell m^\ell + i_{\ell-1} m^{\ell-1} + ... + i_1 m + i_0$$

where $0 \leq i_k \leq m - 1$ for $0 \leq k \leq \ell$. Then

$$i = m\left(i_\ell m^{\ell-1} + i_{\ell-1} m^{\ell-2} + ... + i_1\right) + i_0$$

where $i_\ell m^{\ell-1} + i_{\ell-1} m^{\ell-2} + ... + i_1$ is an integer and $0 \leq i_0 \leq m - 1$. So by the uniqueness of the quotient and remainder, i_0 is the remainder when i is divided by m.

In other words, $i(n) = r$. By the inductive assumption,

$$\alpha_q^n = \sum_{k=0}^{n-1} \alpha_{q(k)}$$

where $q(k)$ is the i^{th} digit in the n-digit base-m expansion of q. Then

$$\begin{aligned}\alpha_i^{n+1} &= \alpha_q^n + \alpha_r \\ &= \alpha_{q(0)} + \alpha_{q(1)} + ... + \alpha_{q(n-1)} + \alpha_r.\end{aligned}$$

Now we must show that $q(k) = i(k)$ for $0 \leq k \leq n-1$.

To this end, I claim that if

$$i = (i_\ell \ i_{\ell-1} \ ... \ i_1 \ i_0)_m$$

and we let $i = qm + r$ where $0 \leq r \leq m-1$, then

$$q = (i_\ell \ i_{\ell-1} \ ... \ i_1)_m.$$

Again,

$$i = i_\ell m^\ell + i_{\ell-1} m^{\ell-1} + ... + i_1 m + i_0$$

where $0 \leq i_k \leq m-1$ for $0 \leq k \leq \ell$. By the division algorithm, there exists unique integers q and r such that

$$i = qm + r$$

where $0 \leq r \leq m-1$. So

$$i_\ell m^\ell + i_{\ell-1} m^{\ell-1} + ... + i_1 m + i_0 = qm + r,$$

but since $0 \leq r \leq m-1$, $i_0 = r$ (by claim 1). So

$$\begin{aligned}i_\ell m^\ell + i_{\ell-1} m^{\ell-1} + ... + i_1 m &= qm \\ i_\ell m^{\ell-1} + i_{\ell-1} m^{\ell-2} + ... + i_1 &= q\end{aligned}$$

as claimed. By the inductive assumption, α_q^n is the sum of the values of the base-m digits of q, and therefore α_i^{n+1} is the sum of the values of the base-m digits of i, completing the induction and the proof. ■

We will now discuss two specific cases of substitution schemes: palindromic and rearrangement schemes.

2.4.3 Palindromic Substitution Schemes

We say that a pitch pattern α of length m is **palindromic** if and only if

$$\alpha = \overline{\alpha}.$$

In terms of indices,

$$\alpha_k = \alpha_{m-1-k}.$$

It turns out that if a pitch pattern is palindromic, then its composition with itself is also palindromic. In order to prove this, we first need two lemmas:

Lemma 2.3. *To reverse a pitch pattern α composed of n blocks, we reverse the order of all blocks, then reverse the elements in each individual block.*

Proof. (By Induction) (Base) When $n = 1$, $\alpha = (\alpha_0 \ \alpha_1 \ \dots \ \alpha_{i-1})$ for an α of length i, and following the lemma, the reverse is indeed $(\alpha_{i-1} \ \alpha_{i-2} \ \dots \ \alpha_0) = \overline{\alpha}$.
(Inductive Step) We assume the reverse of α consisting of p blocks is obtained following the lemma, and we need to show that α of $p+1$ blocks is reversible by the lemma for all patterns with p blocks. We label each block as θ_i, and each can be a pitch pattern of any length. Then for an α of size $p+1$ blocks, we have

$$\begin{aligned}\alpha &= (\theta_0, \ \theta_1, \dots, \ \theta_{p-1}, \ \theta_p) \\ &= (\lambda, \ \theta_p)\end{aligned}$$

where $\lambda = (\theta_0, \dots, \ \theta_{p-1})$ has p blocks. By the inductive assumption,

$$\begin{aligned}\overline{\alpha} &= \overline{(\lambda, \ \theta_p)} \\ &= \left(\overline{\theta}_p, \ \overline{\lambda}\right) && \text{(Inductive assumption when } p = 2) \\ &= \left(\overline{\theta}_p, \ \overline{\theta}_{p-1}, \dots, \ \overline{\theta}_0\right) && \text{(Inductive assumption when } p = n).\end{aligned}$$

∎

Lemma 2.4. *For constant k and pitch pattern α of length m, $\overline{k+\alpha} = k + \overline{\alpha}$.*

Proof. We have

$$k + \alpha = (k + \alpha_0, k + \alpha_1, \dots, k + \alpha_{m-1})$$

and by lemma 2.3

$$\overline{k+\alpha} = (\overline{k + \alpha_{m-1}}, \overline{k + \alpha_{m-2}}, \dots, \overline{k + \alpha_0}).$$

Since k is constant, it follows that its reversal is the same constant, such that $\overline{k+\alpha} = k + \overline{\alpha}$. ∎

And now for our major two claims of this section.

Theorem 2.5. *If a pitch pattern α is palindromic, then α^n is palindromic.*

Proof. (By Induction) (Base) When $n = 1$, $\alpha^n = \alpha^1 = \alpha$, which is palindromic.
(Inductive Step) We assume α^p is palindromic for some p, and we need to show that α^{p+1} is palindromic for all p. Mathematically, we must show

$$\overline{\alpha^{p+1}} = \alpha^{p+1}.$$

Then

$$\begin{aligned}\overline{\alpha^{p+1}} &= \overline{(\alpha_0 + \alpha^p, \alpha_1 + \alpha^p, \dots, \alpha_{m-1} + \alpha^p)} \\ &= (\overline{\alpha_{m-1} + \alpha^p}, \overline{\alpha_{m-2} + \alpha^p}, \dots, \overline{\alpha_0 + \alpha^p}) && \text{(By lemma 2.3)} \\ &= (\alpha_{m-1} + \overline{\alpha^p}, \alpha_{m-2} + \overline{\alpha^p}, \dots, \alpha_0 + \overline{\alpha^p}) && \text{(By lemma 2.4)} \\ &= (\alpha_{m-1} + \alpha^p, \alpha_{m-2} + \alpha^p, \dots, \alpha_0 + \alpha^p) && \text{(Since } \alpha^p \text{ is palindromic)} \\ &= (\alpha_0 + \alpha^p, \alpha_1 + \alpha^p, \dots, \alpha_{m-1} + \alpha^p) && \text{(Since } \alpha \text{ is palindromic)} \\ &= \alpha^{p+1}.\end{aligned}$$

∎

Theorem 2.6. *The substitution scheme*

$$\begin{aligned}\alpha^{(n+1)} &= \left(\alpha_0 + \beta^{(n)}, \alpha_1 + \alpha^{(n)}, \alpha_0 + \beta^{(n)}\right)\\ \beta^{(n+1)} &= \left(\beta_0 + \alpha^{(n)}, \beta_1 + \beta^{(n)}, \beta_0 + \alpha^{(n)}\right)\\ \alpha^{(0)} &= \beta^{(0)} = (0)\end{aligned}$$

is palindromic.

Proof. (By Induction) (Base) When $n = 0$, $\alpha^1 = (\alpha_0\ \alpha_1\ \alpha_0)$ and $\beta^1 = (\beta_0\ \beta_1\ \beta_0)$, which are both palindromic.
(Inductive Step) We assume α^p and β^p are both palindromic for some p, and we need to show that α^{p+1} and β^{p+1} are palindromic for all p. Then

$$\begin{aligned}\overline{\alpha^{p+1}} &= \overline{(\alpha_0 + \beta^{(p)}, \alpha_1 + \alpha^{(p)}, \alpha_0 + \beta^{(p)})}\\ &= \left(\overline{\alpha_0 + \beta^{(p)}}, \overline{\alpha_1 + \alpha^{(p)}}, \overline{\alpha_0 + \beta^{(p)}}\right) && \text{(By lemma 2.3)}\\ &= \left(\alpha_0 + \overline{\beta^{(p)}}, \alpha_1 + \overline{\alpha^{(p)}}, \alpha_0 + \overline{\beta^{(p)}}\right) && \text{(By lemma 2.4)}\\ &= \left(\alpha_0 + \beta^{(p)}, \alpha_1 + \alpha^{(p)}, \alpha_0 + \beta^{(p)}\right) && \text{(Since } \alpha^p,\ \beta^p \text{ palindromic inductively)}\\ &= \alpha^{p+1},\end{aligned}$$

$$\begin{aligned}\overline{\beta^{p+1}} &= \overline{(\beta_0 + \alpha^{(p)}, \beta_1 + \beta^{(p)}, \beta_0 + \alpha^{(p)})}\\ &= \left(\overline{\beta_0 + \alpha^{(p)}}, \overline{\beta_1 + \beta^{(p)}}, \overline{\beta_0 + \alpha^{(p)}}\right) && \text{(By lemma 2.3)}\\ &= \left(\beta_0 + \overline{\alpha^{(p)}}, \beta_1 + \overline{\beta^{(p)}}, \beta_0 + \overline{\alpha^{(p)}}\right) && \text{(By lemma 2.4)}\\ &= \left(\beta_0 + \alpha^{(p)}, \beta_1 + \beta^{(p)}, \beta_0 + \alpha^{(p)}\right) && \text{(Since } \alpha^p,\ \beta^p \text{ palindromic inductively)}\\ &= \beta^{p+1}.\end{aligned}$$

■

We now move on to rearrangement schemes.

2.4.4 Rearrangement Schemes

We define a **rearrangement scheme** as a substitution scheme in which any number of pitch patterns are merely rearranged in each iteration. We claim the following:

Theorem 2.7. *Let $b \geq 2$ be an integer. For $0 \leq \ell \leq b-1$, the substitution scheme*

$$\begin{aligned}\alpha^{\ell,n+1} &= \left(\alpha^{\ell,n}, \alpha^{\ell+1,n}, \ldots, \alpha^{\ell+b-1,n}\right) \quad (1)\\ \alpha^{\ell,0} &= (\ell)\end{aligned}$$

where ℓ is the sequence index and n is the iterate index, is solved by

$$\alpha_k^{\ell,n} = (|k|_b + \ell) \mod b$$

for all n, where k is the position index of each substitution scheme.

To clearly illustrate the substitution schemes in this theorem, let us give a couple of examples. When $b = 2$,

$$\begin{aligned} \alpha^{0,n+1} &= \left(\alpha^{0,n}, \alpha^{1,n}\right) \\ \alpha^{1,n+1} &= \left(\alpha^{1,n}, \alpha^{0,n}\right) \\ \alpha^{0,0} = (0), &\quad \alpha^{1,0} = (1), \end{aligned}$$

and when $b = 3$,

$$\begin{aligned} \alpha^{0,n+1} &= \left(\alpha^{0,n}, \alpha^{1,n}, \alpha^{2,n}\right) \\ \alpha^{1,n+1} &= \left(\alpha^{1,n}, \alpha^{2,n}, \alpha^{0,n}\right) \\ \alpha^{2,n+1} &= \left(\alpha^{2,n}, \alpha^{0,n}, \alpha^{1,n}\right) \\ \alpha^{0,0} = (0), \quad \alpha^{1,0} &= (1), \quad \alpha^{2,0} = (2), \end{aligned}$$

and so on.

To prove this theorem, we first must introduce another lemma:

Lemma 2.8. *Let* $0 \leq j \leq b-1$*. Then if* $j \cdot b^n \leq k \leq (j+1)b^n - 1$*, then* $|k - j \cdot b^n|_b = |k|_b - j$*.*

Proof. The binary expansion of k has $n+1$ digits,

$$j\ b_{n-1}\ b_{n-2} \dots\ b_2\ b_1\ b_0,$$

where the most significant digit is j because $j \cdot b^n \leq k \leq (j+1)b^n - 1$. So

$$\begin{aligned} k - j \cdot b^n &= j\ b_{n-1}\ \dots\ b_1\ b_0 - j\ 0_{n-1}\ \dots\ 0_1\ 0_0 \\ &= 0\ b_{n-1}\ b_{n-2} \dots\ b_2\ b_1\ b_0. \end{aligned}$$

Therefore, the base-b weight of $k - j \cdot b^n$ is j less than the base-b weight of k, and $|k - j \cdot b^n|_b = |k|_b - j$. ■

Now we can prove our theorem.

Proof. (By Induction) (Base) Suppose $n = 0$. We need to show that

$$\alpha_k^{\ell,0} = (|k|_b + \ell) \mod b$$

for $k = 0$. But by **(1)**, $\alpha^{\ell,0} = (\ell)$, so

$$\alpha_0^{\ell,0} = \ell = (|0|_b + \ell) \mod b$$

since $0 \leq \ell \leq b - 1$.

(Inductive Step) We assume

$$\alpha_k^{\ell,p} = (|k|_b + \ell) \mod b$$

for some $p \geq 0$ and all $0 \leq k \leq b^p - 1$. We must show

$$\alpha_k^{\ell,p+1} = (|k|_b + \ell) \mod b$$

for all $0 \leq k \leq b^{p+1} - 1$. Let $0 \leq k \leq b^{p+1} - 1$. Then there exists j such that $0 \leq j \leq b - 1$, and

$$j \cdot b^p \leq k \leq (j+1)b^p - 1.$$

In other words, j is the index of block of length b^p containing k. Then the index of $\alpha_k^{\ell,p+1}$ in the j^{th} block is $k - j \cdot b^n$, since each block is of length b^p. It follows that

$$\begin{aligned}
\alpha_k^{\ell,p+1} &= \alpha_{k-j\cdot b^p}^{\ell+j,p} \\
&= (|k - j \cdot b^p|_b + \ell + j) \mod b && \text{(By the inductive assumption)} \\
&= (|k|_b - j + \ell + j) \mod b && \text{(By lemma 2.8)} \\
&= (|k|_b + \ell) \mod b.
\end{aligned}$$

■

3 Synthesizing Compositions

We now talk about our general approach to synthesizing music using the recursive methods we describe above.

3.1 How to Design Compositions

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" theme as follows:

Figure 11. The Pitch Pattern $\alpha = (2\ 2\ 3\ 4\ 4\ 3\ 2\ 1\ 0\ 0\ 1\ 2\ 2\ 1\ 1)$

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.

3.2 Tweaking the Music

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 disqualify a piece as a pure algorithmic composition. Therefore, we instead present a number of ways to mathematically tweak music so that the process remains algorithmic as well as transparent.

Another aspect of our algorithmic process to consider is that oftentimes, iteratively 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.

3.2.1 Final Transpositions

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.

3.2.2 The Modulo Operation

Another problem that potentially arises after many iterations of a recursively generated pitch pattern is that the difference 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 effort 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.

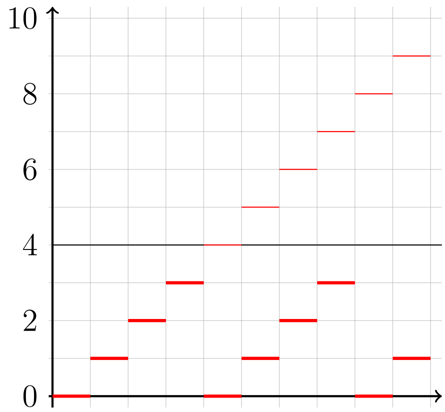


Figure 12. Modulo-4 of a strictly increasing pitch pattern.

To avoid changing the overall texture of a composition, we usually set our 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.

3.2.3 Reflection

We define the reflection of a number x between a and b as

$$f(x) = \begin{cases} z, & z \leq b \\ 2b - z, & z > b \end{cases}$$

where $z = x + \lceil \frac{a-x}{2b-2a} \rceil (2b - 2a)$. Though the definition of a reflection is messy, the concept is not. Take, for example, a pitch pattern $\alpha = (-4\ -3\ -2\ -1\ 0\ 1\ 2\ 3\ 4\ 5)$. Then its reflection between $a = 0$ and $b = 2$ is $(0\ 1\ 2\ 1\ 0\ 1\ 2\ 1\ 0\ 1)$ and appears as follows.

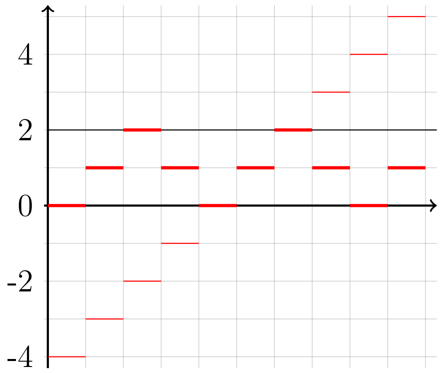


Figure 13. Reflection of α between 0 and 2.

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, regardless 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.

3.2.4 Creating an Ending

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 off a composition in a musically satisfying way.

3.3 Process in Matlab and Lilypond

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 for 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.

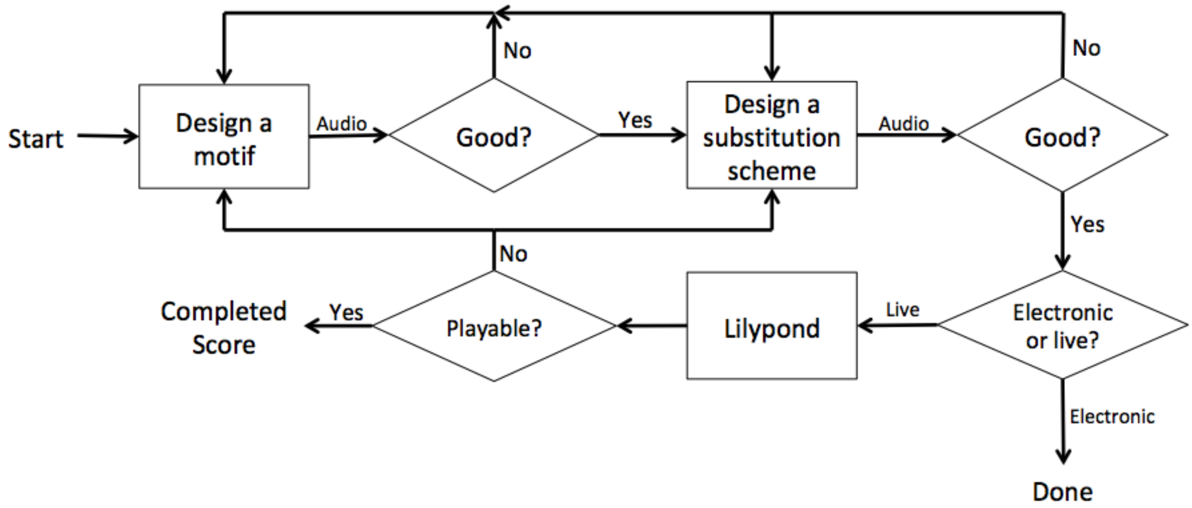


Figure 14. The process of creating an algorithmic composition.

4 My Compositions

I will now demonstrate some of the music that we have created using the processes outlined above.

4.1 Palindromic Composition #3

This is a palindromic piece that we developed before we had the concept of a rhythm pattern. As a result, the two melodic lines are created using the 6^{th} iterate of the substitution scheme, a third part is created using the 5^{th} iterate for both pitch patterns stretched by a factor of three, and the fourth part is the 4^{th} iterate of both pitch patterns stretched by a factor of nine.

$$\alpha^{n+1} = (1 + \beta^n, \alpha^n, 1 + \beta^n)$$
$$\beta^{n+1} = (\alpha^n, 2 + \beta^n, \alpha^n)$$
$$\alpha^0 = \beta^0 = (0)$$

The pitch mapping is $0 \to$ D, $1 \to$ F#, $2 \to$ A, $3 \to$ C#, and all pitches are reflected between 0 and 8.

4.2 March of the Ants

This is an example of a composition that uses a parameter sequence, which changes which operations occur on a pitch pattern over time. The row of the parameter sequence displays which operation will occur, where each parameter is given in its corresponding equation, and the columns represent chronological periods of time. Note that for this composition, we apply a diatonic scale starting at A. Additionally, this piece was designed solely for computer output, so it does not contain any tweaks.

Pitch:

$$\begin{pmatrix}\alpha^{n+1}\\ \beta^{n+1}\end{pmatrix} = \begin{pmatrix}\alpha^n, a \pm \sigma^c \beta^n\\ \beta^n, b \pm \sigma^d \alpha^n\end{pmatrix}$$
$$\alpha^0 = (5\ 2\ 4\ 1\ 3\ 0\ 1\ 2\ 3\ 1\ 2)$$
$$\beta^0 = (5\ 4\ 3\ 2)$$

Rhythm:

Division scheme: (2 2 3 2)

$$\begin{pmatrix}\rho^{n+1}\\ \tau^{n+1}\end{pmatrix} = \begin{pmatrix}\rho^n, \sigma^e \tau^n\\ \tau^n, \sigma^f \rho^n\end{pmatrix}$$

Initial rhythm pattern ρ^0:

event times: (0 4 6 10 12 14 16 18 20 22 23)
durations: (4 1 4 1 2 2 2 1 1 1 1)

Initial rhythm pattern τ^0:

event times: (0 6 12 18)
durations: (6 6 6 6)

Number of inversions: g
Number of iterations per parameter set: h

Parameter sequence:

$$\begin{pmatrix}a\\ b\\ c\\ d\\ e\\ f\\ g\\ h\end{pmatrix} = \begin{pmatrix}0 & 1 & -1 & 2 & 0 & -3 & 0 & -1 & 0\\ 0 & -1 & 1 & 3 & 0 & 2 & 0 & 1 & 0\\ 0 & 0 & 0 & 1 & 0 & -1 & 0 & 2 & 0\\ 0 & 0 & 0 & 1 & 0 & 1 & 0 & -2 & 0\\ 0 & 0 & 0 & 3 & 0 & 8 & 0 & -8 & 0\\ 0 & 0 & 0 & -3 & 0 & -8 & 0 & 8 & 0\\ 0 & 0 & 1 & 2 & 1 & 2 & 2 & 2 & 0\\ 1 & 2 & 2 & 2 & 1 & 2 & 1 & 2 & 1\end{pmatrix}$$

4.3 A Simple Piece

Here is a piece created using a single set of parameters.

$$\begin{pmatrix}\alpha^{n+1}\\ \beta^{n+1}\\ \gamma^{n+1}\\ \delta^{n+1}\end{pmatrix} = \begin{pmatrix}\alpha^n, & \beta^n, & 2-\gamma^n, & 3-\delta^n\\ \beta^n, & 1+\gamma^n, & 2-\delta^n, & -\alpha^n\\ \gamma^n, & -2+\delta^n, & -3+\alpha^n, & -2-\beta^n\\ \delta^n, & \alpha^n, & 1+\beta^n, & -2-\gamma^n\end{pmatrix}$$

$$\alpha^0 = (2\ 3\ 1)$$
$$\beta^0 = (0\ 1\ -1\ 2)$$
$$\gamma^0 = (-2\ 0)$$
$$\delta^0 = (2\ -1)$$

Rhythm:

Division scheme: (3 2 2)

$$\rho^0 = (0\ 4\ 8)$$
$$\tau^0 = (0\ 2\ 6\ 10)$$
$$\upsilon^0 = (0\ 6)$$
$$\phi^0 = (0\ 8)$$

This scheme is iterated twice using a diatonic scale starting at A. The ending concatenates one final measure of a single note held for three beats for each part, and each is assigned pitches in descending order, 4, 2, 6, and 0.

4.4 Longsheng Hike

Substitution Scheme: $\begin{pmatrix}\alpha^{n+1}\\ \beta^{n+1}\\ \gamma^{n+1}\\ \delta^{n+1}\end{pmatrix} =$

$$\begin{pmatrix}\alpha^n, & 2+\beta^n, & 1-\sigma^4\alpha^n, & -1+\beta^n, & 2-\sigma^2\beta^n, & 2+\alpha^n\\ \alpha^n, & \alpha^n, & -2+\sigma^{-4}\beta^n, & 2-\sigma^{-4}\alpha^n, & -2-\sigma^{-2}\alpha^n, & \beta^n\\ \gamma^n, & 1+\delta^n, & 1-\sigma^2\gamma^n, & 1+\sigma^{-4}\delta^n, & 2-\sigma^2\delta^n, & -\gamma^n\\ -4+\delta^n, & -2+\gamma^n, & -3+\sigma^2\delta^n, & -1-\sigma^{-4}\gamma^n, & -2-\sigma^{-2}\gamma^n, & -\delta^n\end{pmatrix}$$

$$\alpha^0 = (0\ 1\ 0\ 1\ 2\ 4\ 2\ 1\ 0\ 1)$$
$$\beta^0 = (0\ 1\ 2\ 4\ 6\ 7\ 5\ 4\ 6)$$
$$\gamma^0 = (0\ 2\ 5\ 2\ 1)$$
$$\delta^0 = (5\ 4\ 3\ 0)$$

Rhythm:

$$\begin{aligned} &\text{Division scheme: } (4\ 2\ 2) \\ \rho^0 &= (0\ 3\ 4\ 6\ 8\ 10\ 12\ 13\ 14\ 15) \\ \tau^0 &= (0\ 1\ 2\ 3\ 4\ 6\ 8\ 10\ 12) \\ \upsilon^0 &= (0\ 4\ 8\ 12\ 14) \\ \phi^0 &= (0\ 7\ 8\ 15) \end{aligned}$$

This scheme is iterated twice using a pentatonic scale starting at A. The first part's pitch pattern is transposed up by two, while the third and fourth part are transposed down by 7. Finally, the first and second pitch patterns are reflected between -5 and 7, the third part is reflected between -11 and -1, and the fourth part is reflected between -11 and 0. After these reflections, the third and fourth parts are transposed up by 5. The ending concatenates one final measure of a single note held for four beats for each part, and each is assigned pitches in descending order, -3, 3, 5, and -5.

Additionally, we have altered the staging of the voices slightly: In the first 6 measures, the second and fourth instruments don't play. In measures 31-36, the third and fourth instruments don't play.

5 Conclusion

We have reported on several techniques we have developed for generating musical compositions algorithmically. We have used sequences of integers called pitch patterns 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 5^{th}, also follow schemes with definitive patterns, in this case following a $(0\ -2\ -1\ -3)$ 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.

Appendix I: Palindromic Composition #3

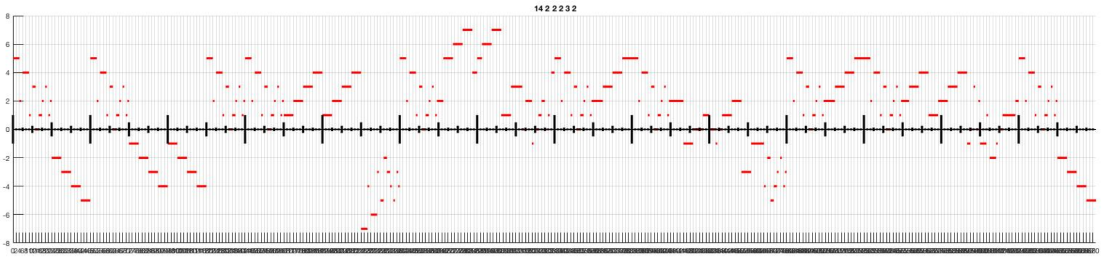
Palindromic Composition #3

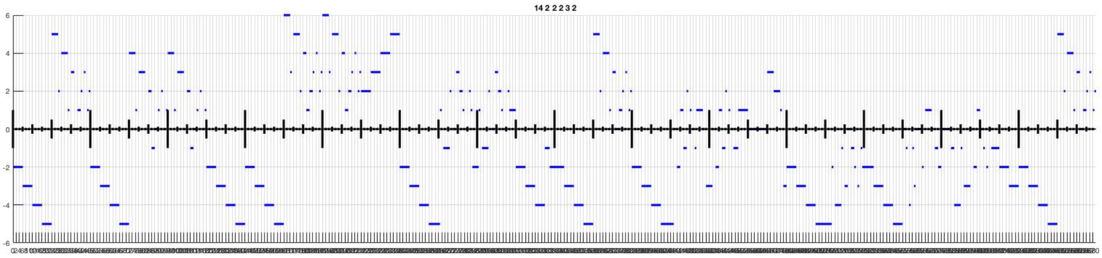
Carlo Anselmo


12
16
20


24

Appendix II: March Of The Ants


14 2 2 2 3 2


14 2 2 2 3 2

Appendix III: A Simple Piece

A Simple Piece

Full Score

Carlo Anselmo

Appendix IV: Longsheng Hike

Longsheng Hike

Full Score

Carlo Anselmo


12
15
18
21


24
27
30
33


35