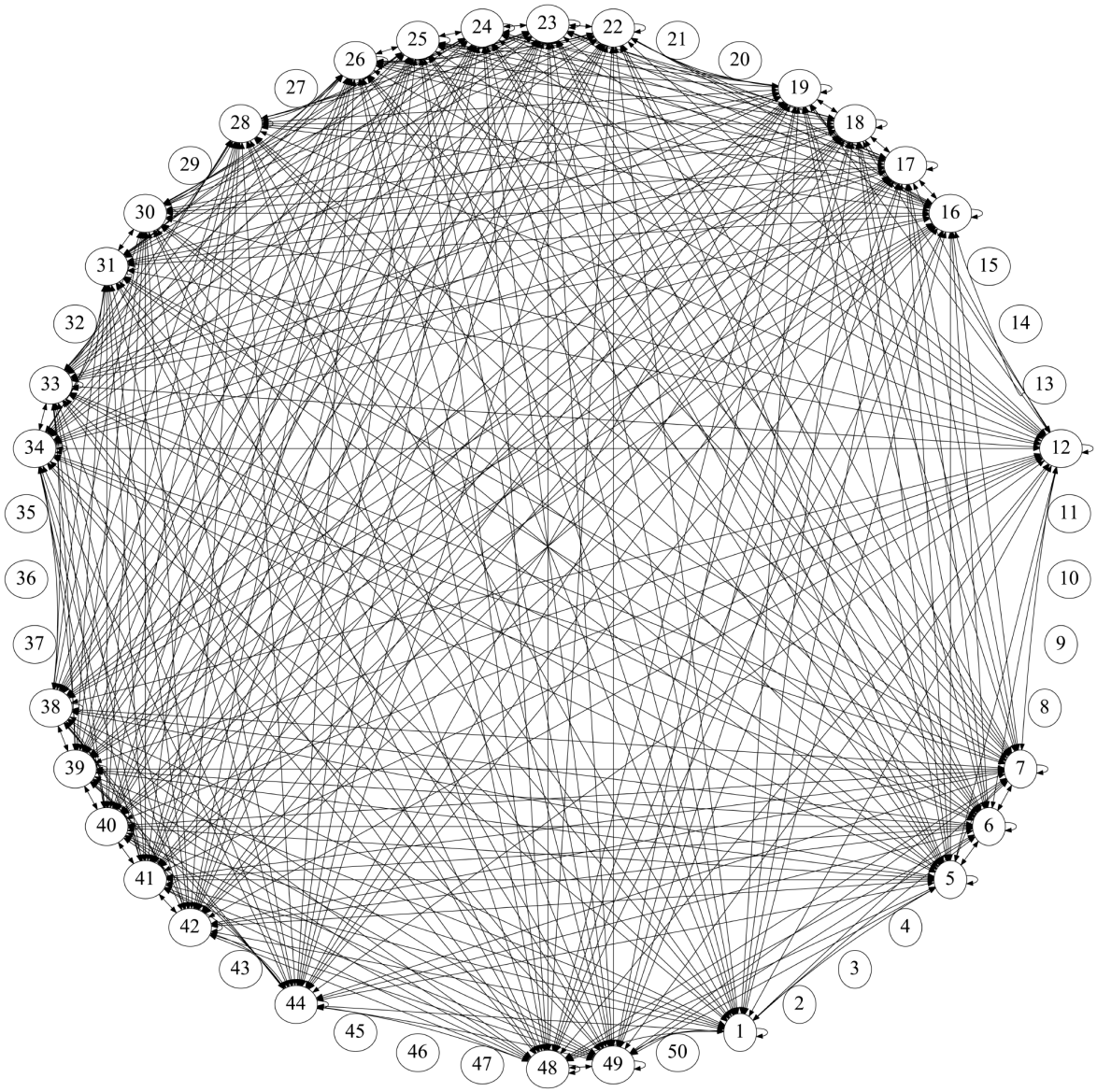
Max Gupta
Columbia University
Junior Spring Thesis in Applied Mathematics
May 2020

Musical Markov Chains: Stochastic Models in Music

Max D. Gupta



With attached files: La Recherche.wav; Martin Garrix Sets 1.2.mps; HMM_Model.m

Credits ***3***

Abstract ***3***

Methodology ***3***

1. Introduction ***5***

1.2 The Traditional Markov Model **5**

1.2 A Simple Compositional Example **6**

2. Traditional Markovian Musical Compositions ***8***

2.1 A Discussion of Xenakis' Analogique A et B **9**

3. Extensions of the Markov Model ***10***

3.1 Kevin Jones: N-th Order Chains and Event Vectors **11**

3.2 Generative Grammars **13**

3.3 Grammars and Markov Chains **17**

3.4 Hidden Markov Models and Related Algorithms for Audio Classification **17**

3.4.1 The Forward Algorithm for Probability Calculation 19

3.4.2 The Viterbi Algorithm for Decoding 20

3.4.3 The Baum-Welch Algorithm for Training 21

4 Musical Generations ***21***

4.1 À La Recherche Du Temps Perdu **21**

4.2 Say My Name – Markov Remake **22**

4.3 Martin Garrix Markov Remake **24**

4.4 HMM Training in MATLAB **24**

5 Discussion ***24***

6 Conclusion ***25***

7 Further Research ***25***

Bibliography ***26***

Appendix A ***28***

Appendix B ***30***

Credits

I owe many thanks and much appreciation to Dr. Mikhail Malt, IRCAM, my advisor on this project, for his continued support, insight, and good humor in instructing me over the course of the last 3 months. As an undergraduate student with no previous experience in algorithmic composition or musical theory whatsoever, I was inspired by Mikhail to continually push my boundaries and to learn at the accelerated rate that I did. While my time in Paris and at IRCAM was cut short because of the COVID-19 global health crisis, I continuously received phenomenal support from IRCAM and Columbia in Paris to support my independent research at a distance. Many thanks to Enyi Koene, my academic advisor; Cathy Collins; and Severine Martin-Hartenstein for facilitating the transition to online learning for all Columbia in Paris students. I'd also like to thank my family and friends for their love and support every day of this journey, none of which goes unrecognized. None of this would have been possible without your support. Thank you.

Abstract

The use of Markov chains in musical composition is highly documented in the community of algorithmic composition and has seen many advances in the years from the 50's through to the 90's. Today, the use of Markov chains has diminished slightly as algorithmic composers have fine-tuned the mathematics of composition to leverage neural nets and machine learning based optimization algorithms. However, Markov chains and many useful extensions of Markov chains, such as generative musical grammars and Hidden Markov Models, can be transformed into powerful musical classification and generation machines in themselves. This paper aims to follow the systematic development of the Markov model architecture, from its conception in 1906 by Andrei Markov, to its first use in music in the 50's, through its main developments in the 80s up until modern uses of the Markov model in contemporary computing technology. We shall first introduce basic Markov theory and its historical use in music through a case study on the composer Iannis Xenakis. We then survey the development of the Markov architecture to incorporate higher order memory systems (K. Jones), linguistics (N. Chomsky), and dynamic programming (Hidden Markov Models). The theories are then brought to life with originally composed works by the author, illustrating the fascinating generative properties of Markov models.

Methodology

The first part of this paper takes a historical perspective, examining 20th century composers and reviewing their techniques through primary and secondary sources (source script and peer review). The second part of the paper takes an architectural point of view to focus on the schematic and technical enlargement of the concept of Markov chains. This section brings in influences from mathematics (dynamic programming) and linguistics to extrapolate on the concept of systematized, generative composition. Finally, the last part of the paper takes a creative approach as I present findings from experimentation using state of the art Markov methods in composition and present original works.

All non-cited diagrams and equations are originally produced using a variety of technical software (noted per diagram). All other diagrams from external sources are appropriately reproduced for the purpose of explanation and cited. It is worthwhile noting that, for someone just beginning their journey in this fascinating field, the topic at hand required many hours of autonomous systems and software training and thus the original productions are more for the purpose of showcasing the power of Markov chains in composition rather than to contribute to the greater musical corpus. It is the author's intention to continue building on technical proficiency to produce higher quality work in the future.

Stochastic techniques constitute a large and growing field in applied mathematics and modern computer science and have been used as a means of data reduction and structuration for complex tasks. Musical composition is such a task. For a computerized music system to process and define a sound, massive sets of data and instructions have to be filtered and processed instantly over a range of acoustic parameters. Specific kinds of stochastic techniques exist to reduce this compositional labor, automate parts of composition, and algorithmically generate novel compositions, all while retaining artistic license. We find this result in the works of Andrei A. Markov (1856-1922), a Russian mathematician intrigued by the analysis and generation of patterned sequences. Markov first applied his method when studying the pattern by which vowels and consonants alternated in a poem by Pushkin, "Eugene Onegin" (Nierhaus, 2008). From its modest poetic beginnings, Markov chains now occupy a central branch of mathematical statistics, engineering, and economics.

1.2 The Traditional Markov Model

In its simplest form, a traditional Markov chain is a very simple idea: a stochastic process, transiting linearly forwards in discrete time steps through a finite (or at most countable) set of states, *without memory*: the next state depends just on the current state, not on any preceding states. We call these 'traditional' chains *1st order chains*. The *order* of a chain is defined as the number of states preceding the current state that determine it, in other words, the number of previous events that influence the current.

Markov chains, or Markov Models as we will refer to them from time to time, are processes that sort events into discrete states and define rules that govern the transitions between these states. The key feature of the traditional Markov chain is the that the transitional rule to attain a subsequent state is determined solely by the nature of the current state, a probability. For example, if *a* and *b* are events defined in the event space of a chain, then the probability of event *a* being directly followed by event *b* is given as:

$$\boldsymbol{P_{ab} = p(E_{i+1} = a \mid E_i = b)}$$

Eq. 1.1: This property is known as the Markov Property, where E is the given event, i is the index of the event within the set of events, and P_{ab} is the probability that event a is followed directly by event b.

The easiest and most common way to represent Markov chains is by use of a matrix of transitional probabilities (MTP) and/or a nodal graph.

$$\begin{bmatrix} E_{11} & \cdots & E_{1j} \\ \vdots & \ddots & \vdots \\ E_{i1} & \cdots & E_{ij} \end{bmatrix}$$

Figure 1.1: The form of the MTP for transitions of events. E_{11} denotes the probability of event 1 going back to event 1, E_{1j} denotes the probability of event 1 going to event j, and so on, for a standard matrix of j columns and i rows. Note that each row and each column have to add up to 1 exactly to complete all transitional possibilities.

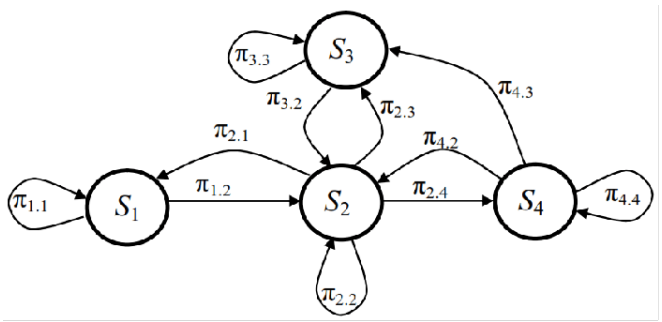


Figure 1.2: The graphical form of an arbitrary Markov chain with 4 states (S_1…S_4) and probabilities of transition represented by π_{ij} for transition from state i to j.

These two representations will be used continuously in different forms throughout this paper, so it is important to make them immediately familiar in their general form. The event space (state space) can contain any finite number of events, which can be related to any finite number of events. An event can and will be defined in a number of ways. For the purposes of musical composition, we denote 'events' as musical notes.

1.2 A Simple Compositional Example

To put the general theory into a musical perspective, we will begin with a simple yet illustrative example. When constructing a Markov chain, we first ask the question: how many events do we want? We'll start with 4 simple musical events. Next, how will we define the events? We'll ascribe 1 musical note per event and choose the 4 notes used in *Mary Had a Little Lamb*: E (event #1), D (#2), C (#3), and G (#4), all natural. Finally, all that is left is to describe the MTP: the relationship between the 4 notes. Below is the original piano score of *Mary Had a Little Lamb*, the transitional matrix that maps the relations of the notes in a 4x4 MTP identical to the format of Figure 1.1.

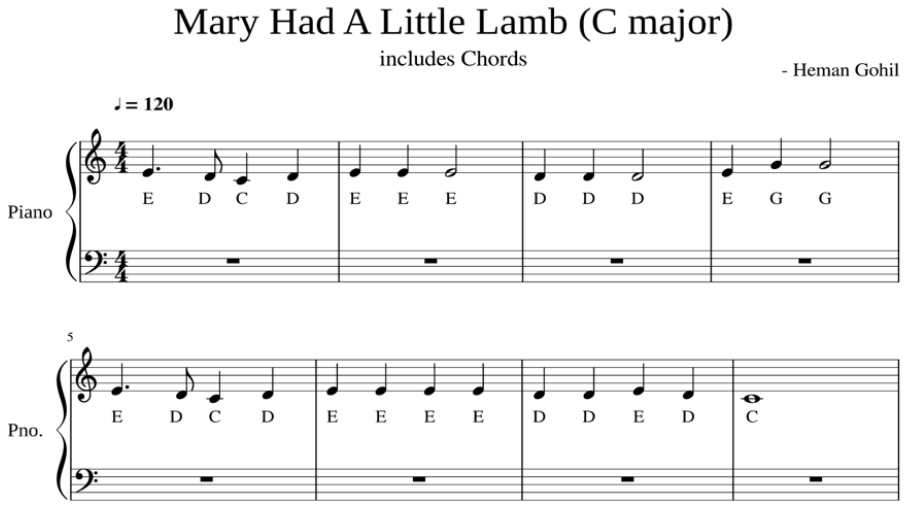


Figure 1.3: Original score of Mary Had A Little Lamb (retrieved from MuseScore)

$$\begin{vmatrix} 5/11 & 5/11 & 0 & 1/11 \\ 4/10 & 3/10 & 3/10 & 0 \\ 0 & 2/3 & 1/3 & 0 \\ 1/2 & 0 & 0 & 1/2 \end{vmatrix}$$

Figure 1.4: The MTP for Mary Had A Little Lamb
NB: all notes in the original score are treated as quarter notes for MTP production.

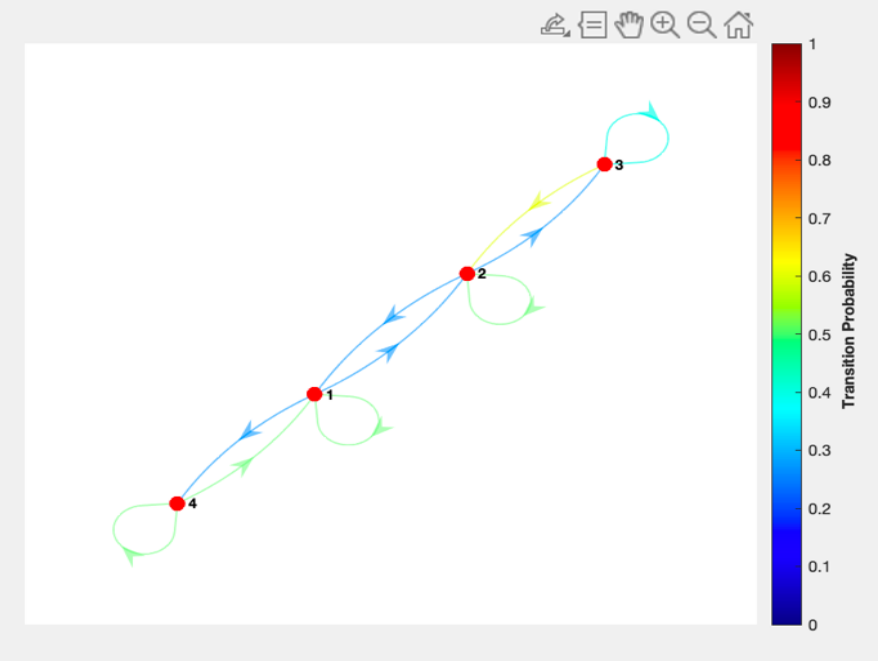


Figure 1.5: Markov chain of Mary Had a Little Lamb generated from Fig 1.4
[constructed in MATLAB]

The methodology used in constructing this chain (illustrated in Figure 1.5, in the image of Figure 1.2) is simple. By looking at the adjacency of notes in the original score (Fig 1.3), we evaluate each note based on the notes that directly follow it. We take all the E's and see that for 11 E naturals, 5 are followed by D natural, 5 are followed by E natural, and 1 is followed by G natural. Thus, 5/11, 5/11, and 1/11 correspond to the transition probabilities in the first row, representing input following E natural (event 1). Observing the set of D naturals (event 2), we see that for 10 D's, we have 3 followed by C natural, 4 by E natural, and 3 repeated to D, corresponding to the 2nd row of transition probabilities. We follow this pattern for the 3rd and 4th events. One observes that the Markov chain can start anywhere on the event space, and thus the possible outcome sequences that are generated are infinite. As an example, we run the chain in MATLAB, starting at E natural, and cycling 25 times to produce the random sequence:

$$X = 2,1,1,1,1,1,1,1,2,2,2,1,2,1,2,2,2,1,1,2,3,2,2,3,3,2$$

Eq. 1.2: 25 step random walk in Figure 1.5, produced in MATLAB Simulation [see Appendix A]

$$X = [D,E,E,E,E,E,E,E,D,D,D,E,D,E,D,D,D,E,E,D,C,D,D,C,C,D]$$

Eq 1.3: Musical note transcription of 25 step random walk

The sequence above is transcribed into musical notation to produce the following note sequence, a Markov chain remix of *Mary Had a Little Lamb*:

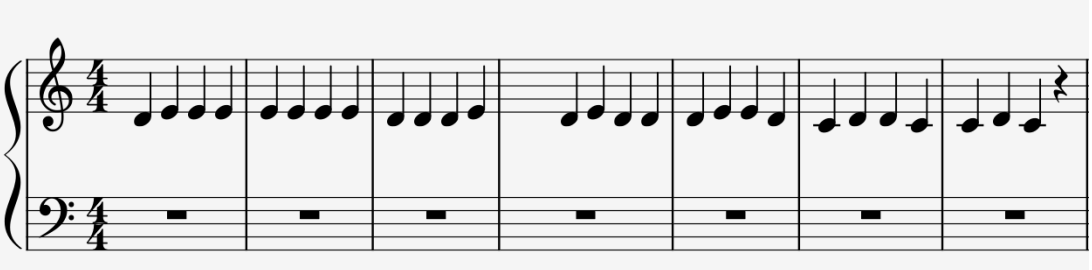


Figure 1.6: Markov Sequence, composed in MuseScore

The melody is of course, incredibly simple, yet surprisingly effective. With some modest refinement, adding complementary notes in the same scale, varying note length, and playing with pitch, we can build a complex, fully self-generating improvisational machine as we will see in section 4.

2. Traditional Markovian Musical Compositions

The use of Markov chains in musical generation is often studied for their intuitive use and the ease with which they generate creative and novel-sounding rhythms automatically. This field of study is now widely known as *algorithmic composition*, the production of music through algorithmic structure in statistics, mathematics, and computer code (Nierhaus, 2008). As we will see, algorithmic composition is an incredibly interdisciplinary, innovative, and fast-growing field. One of the most notable examples is Hiller and Isaacson's 1956 *Illiac Suite*, a composition that was generated using rule systems and Markov chains in 1956 (Sandred et a, 2009). The piece was designed as a series of experiments on formal musical composition. During the following decade, Hiller's work inspired colleagues from the University of Illinois to further experiment with algorithmic composition, using a library of computer subroutines for algorithmic composition called MUSICOMP (Ames, 1987). This library provided a standard implementation of the various methods used by Hiller and others. No further reference to it will be made, however, since it is severely outdated.

There were also several other early and notable examples of Markov chain algorithmic composition, though not so widely cited as Hiller and Isaacson's. *Push Button Bertha*, composed in 1956 around the same time as Hiller's *Illiac Suite*, was a program created by Martin Klein and Douglas Bolitho with the computer DATATRON (Ames, 1987). Algorithmic composition of that era, however, was more akin to mass random digit distributions (DATATRON produced 1000 digits simultaneously, each corresponding to one of eight notes on the chromatic scale), rather than formalized statistics.

Non-scholarly early examples also exist, though they are difficult to assess because of the sparsity of the published material, and the fact that they are mostly not peer reviewed. For example, Pinkerton (1956) described in Scientific American a "Banal Tune-Maker", a simple Markov chain created from a statistical analysis of 39 nursery rhymes. Pinkerton's process in making the *Banal Tune-Maker* offered an innovative new Markov-based structure, moving away from the randomized sputtering of super-computers, and adopting a technique of pitch, frequency, and duration counting and comparison that actually mirrors that which I used in our first compositional example. Pinkerton (1956) would standardize notes into a single octave, transpose them into a code with the 8 chromatic notes and 1-beat rests (*O*) such as *COCCOC/EOEEOE,* and finally count the number of times each note appeared in all 39 nursery rhymes, creating a probability occurrence matrix (i.e. C occurs 16.3% of the time, A 4.5% of

the time, rests 29.7% of the time, etc.). He would then construct a matrix of transition probabilities identical to Figure 1.4 and create "banal tunes" in 6/8 time with alternating eighth notes and rests. Effectively, *Mary Had A Little Lamb* is a cursory representation of this 1956 tune-maker model.

As machines became less expensive, more powerful and in some cases interactive, algorithmic composition slowly took off. Aside from the researchers at the University of Illinois (Hiller's university), there was little continuity in research, and reinventing the wheel in algorithmic composition techniques was common. In this paper, we continue musical research into this most interesting stochastic technique, Markov chains, that launched and serves as the bedrock for many of today's modern and fast-developing approaches to algorithmic composition.

2.1 A Discussion of Xenakis' Analogique A et B

I devote this section to a discussion of an early Markov chain produced work from 1959, *Analogique A et B*, by the French-Greek composer Iannis Xenakis (1922-2001). Xenakis is considered to be one of the pioneers in the implementation of mathematics in music, and successfully utilized tools from stochastic processes, game theory, set theory, cybernetics, and thermodynamics in his compositions. While I rely on the works of Agostino Di Scipio (1997; 2005) and Sam Goree (2017) in this analysis, I also draw from independent analysis via spectral decomposition in Sonic Visualizer (see graphs in Appendix A). Note that the graphs are stand-alone representations of different technical and musical aspects (frequency, melodic range, intensity, and duration) of *Analogique*.

Of all the earliest Markovian pieces, I picked *Analogique A et B* not simply because it is the most highly documented and analyzed work available, but also to highlight problems and dangers of creating music with 1st order Markov chains. Upon a first listening of *Analogique A et B*, one might (correctly) remark on the discordant atonality and lack of any discernible rhythm. As verified by Di Scipio (2005), Xenakis was unable to attain any of these 2nd order sonorities[1] (rhythm and melody) largely due to the limitations of his stochastic models. We observe that Xenakis was largely not at fault for this, since he was limited by the technology of his time (he calculated all transitions by hand), and he had himself predicted he would have needed computers and digital-to-analogue converters (Di Scipio, 1997) to accomplish his task. Instead, the models he constructed were limited to first order models randomly alternating between two groups of intensity, pitch, and density settings shown below:

Table 2.1: Event Spaces for Pitch, Density, and Intensity in Analogique A et B

	Note Intensity	Bar Density	Note Pitch
Event Definitions	$E_1 = pp$ $E_2 = f$ $E_3 = fff$	E_1 = 1 event/half-bar E_2 = 3 events/half-bar E_3 = 9 events/half-bar	$E_1 = \{E_0 \ldots E_1\}$ $E_2 = \{E_1 \ldots D_2\}$ $E_3 = \{D_2..Db_3\}$ $E_4 = \{Db_3 \ldots C_4\}$ $E_5 = \{C_4 \ldots B_4\}$ $E_6 = \{B_4 \ldots A_5\}$
Event Sets	g_0 [I, I, II, III] g_1 [I, II]	d_0 [I, I, II, III] d_1 [I, II, II, III]	f_0 [I, II, V, VI] f_1 [III, IV]

[1] 2nd order sonorities, as labelled by Xenakis in *Formalized Music (1963)* are structures of rhythm and melody formed by 1st order sonorities of individual beats

Each of the groups, (g_0, g_1); (d_0, d_1); (f_0, f_1), constitute what Xenakis labeled musical "screens" in his work *Formalized Music* (1963). These musical screens, representing intensity, density, and pitch groupings respectively, are randomly cycled according to the transition probability matrices:

	X	Y
X	0.2	0.8
Y	0.8	0.2

Figure 2.1a: MTP #1 for intensity, pitch, and density in Analogique (Goree, 2017)

	X	Y
X	0.85	0.4
Y	0.15	0.6

Figure 2.1b: MTP #2 for intensity, pitch, and density in Analogique (Goree, 2017)

Where X and Y are substitutes for (g_0, g_1), (d_0, d_1), and (f_0, f_1). As we can observe, these MTP's draw out a 1st order Markov chain, and thus have a memory size of 1 preceding cell. It is also observed that the matric pairs are identical for all three control parameters. Di Scipio (2005) speculates that this was done to make things less cumbersome for Xenakis to handle by pocket calculator. In doing so, Xenakis also creates internal restraint and a strong similarity in the dimensions of the musical texture. Just as randomly as the intensity veers from pizzicato to fortissimo, for example, the pitch will shoot from low E_0 to high A_5.

The theoretical procedure for the construction of *Analogique A et B* is thus as follow: a) the application of the MTP to select a screen, b) selection of a randomly selected set of values from within the screen, c) translation of values into musical notation. Interestingly enough, however, Di Scipio (2005) notes that, while this process comes as close as possible to mirroring a simple stochastic theorem of music-making, Xenakis actually toyed with the results manually, making only 3/10ths of the piece truly stochastic, and the remaining 7/10ths result from "loosely formalized decisions" that Xenakis made independent of the structural mechanism. It is thus clear that the mechanism proposed here was clearly not intended to be an end in itself, but rather, as affirmed by Di Scipio, "it was meant to create a music as-yet-unheard". Xenakis' compositional model has to be reset and retriggered in order to introduce in the music the changes and contrasts that nourish musical form (Goree, 2017).

Analogique A et B is considered a 'failure', largely by critics and even by Xenakis himself, both in its original intention of becoming the first fully Markov-autotomized piece, and in its executional sound quality (Di Scipio 1997, Di Scipio 2005; Goree, 2017). However, being the first fully documented and comprehensively analyzed piece of Markovian music, Xenakis' *Analogique A et B* is an important landmark in the field. Furthermore, it is remarkable that Xenakis made a coherent and performable piece with Markov calculations done entirely by hand, with no computer access. With *Analogique*, Xenakis exhibited how multiple musical parameters can be independently controlled by individual Markov chains. Structurally, he went onto inspire composers in the 80's to revise his Markovian methods to add higher degrees of functionality.

3. *Extensions of the Markov Model*

As the theory of Markov chains has been used in musical composition for over 60 years, the architecture of the models has developed far beyond the original simplistic Xenakis one-loop chains. As computational power has increased, Markov chains are now able to control both micro-qualities of sound

like timbre, pitch, and duration and macro-qualities like patterning, rhythm, and melody-chunking. The ability to form these compositional '2nd order sonorities', not present in Xenakis' work, comes as a result of increasing the order of the chains, expanding their effective 'memory bank' from one preceding note to dozens. In this section, we examine several examples of the expanded Markov architecture to include higher level memory and broader functions.

3.1 Kevin Jones: N-th Order Chains and Event Vectors

The work of Kevin Jones, PhD, City University of London, is critical in the enlargement of the scope of Markov chains in music. While the works of Lejaren Hiller and Iannis Xenakis were groundbreaking for their novel use of Markov chains, the pieces *Iliac Suite* and *Analogique A et B* exhibited only the beginnings of a system that would develop exponentially in the 80's and 90's. Published in 1980, Kevin Jones' thesis on computer-assisted application of stochastic structuring techniques in musical composition provides many fruitful extensions of the Markov structure to allow its use in higher order computer processes and more beautiful sounding pieces (see Jones 1980).

Jones was fascinated by the Markov architecture and its potential for musical application and sought to expand upon the simplistic 1st order chains using several novel and exciting mechanisms. The simplest and most intuitive extension was increasing the order of the Markov system. A third order Markov chain would, for example, depend on the two preceding musical events. Generally, an *n*-th order chain holds events whose occurrence depends on the preceding (*n*–1) events. This structure of multi-order Markov chains was employed in Jones' 3rd study of *Macricisum*: *Skirtriks*, which is worthy of note and further study for its acoustic quality (Jones 1980).

With each increase in order comes an increase in the memory-bank of the chain. Thus, for an example song with 128 musical events, a 64th order Markov chain would only have two possible outcomes: the first and second halves of the piece. Such a result would result in a simple oscillation between beginning half and end half and would carry no originality. In fact, it would be against copyright laws to produce a song by cutting another original in half and playing halves or thirds randomly. Necessarily, the Markov chains would end up copying and repeating the main melodies. However, we have also seen the cacophonic consequences of the lowest 1st order chains in *Analogique*. The key issue in determining a fitting Markov order for compositional purposes, thus, is determining a degree that produces a spontaneous *and* original result. Ask a statistician about how to better control randomness to produce a balanced dataset and she will always tell you to control for a higher number of parameters in the data analysis. Jones does this by replacing event scalars in Markov chains with *event vectors*. Instead of each musical 'event' consisting of a single entity, we can engineer the event to input multiple parameter values, hence creating a vector. A different Markov chain may be used to control each parameter, or a Markov chain may control more than one parameter. A typical event vector according to Jones includes the following components:

{pitch range, overall pitch, block length, density, intensity, following silence, envelope of elements, harmonic spectrum of elements, spatial position} (Jones, 1981)

Multi-order, event-vector Markov chains create a stochastic process with a higher degree of customizability. We can now control for all musical parameters stochastically, given our Transition Probability Matrix. To visualize, below is a graphical representation of an 8^{th} order Markov chain, controlling multi-note events e_1 to e_8:

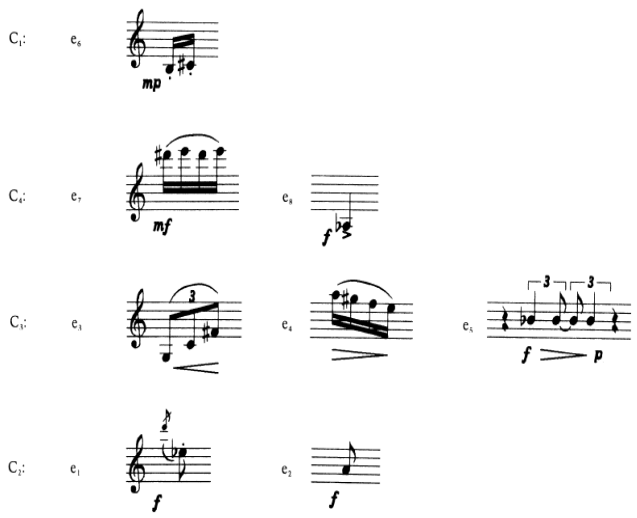


Figure 3.1: 8 Event Vectors Controlled by the adjacent chain [Jones, 1981]

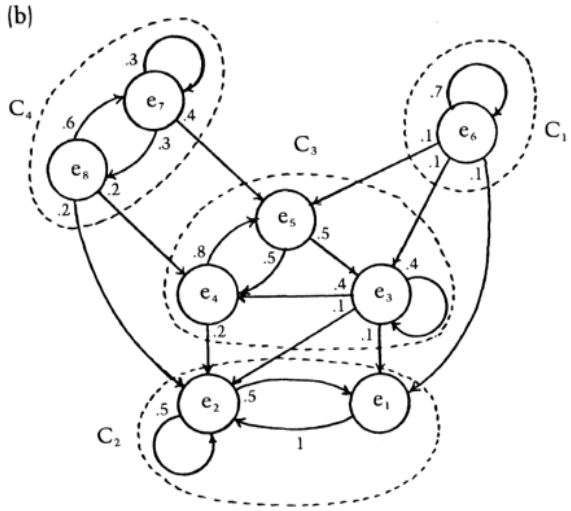


Figure 3.2: Graphical Markov Chain of 8th Order [Kaufman, 1968]

Alternatively, the composer can choose to have a greater degree of flexibility by creating an event vector space for each individual parameter and controlling each with a separate chain. In this case, each musical note is produced as a scalar product of parameter event vectors (2 examples shown below from Jones, 1980). This provides for the highest degree of randomness, since each element of each sound is randomly generated in all of its characteristics: pitch, timbre, duration, and intensity:

$$E_{int} = \langle \text{pp}, \text{pp}, \text{p}, \text{mp}, \text{mf}, \text{f}, \text{ff}, \text{fff} \rangle$$

Figure 3.3a: Event Vector for Intensity

$E_{dur} =$ ⟨ 𝅘𝅥𝅯𝅘𝅥𝅯 , 𝅘𝅥𝅯 , 𝅗𝅥 ⟩

Figure 3.3b: Event Vector for Duration

$$E_{timbre} = \langle \text{flute}, \text{oboe}, \text{violin} \rangle$$

Figure 3.3c: Event Vector for Timbre

$E_{pitch} =$ ⟨ the scale of chromatic notes from C to G' ⟩

Figure 3.3d: Event Vector for Pitch

Or, the composer can choose to predefine notes and alternate between an event vector of these predefined notes, allowing for more control in the desired sound:

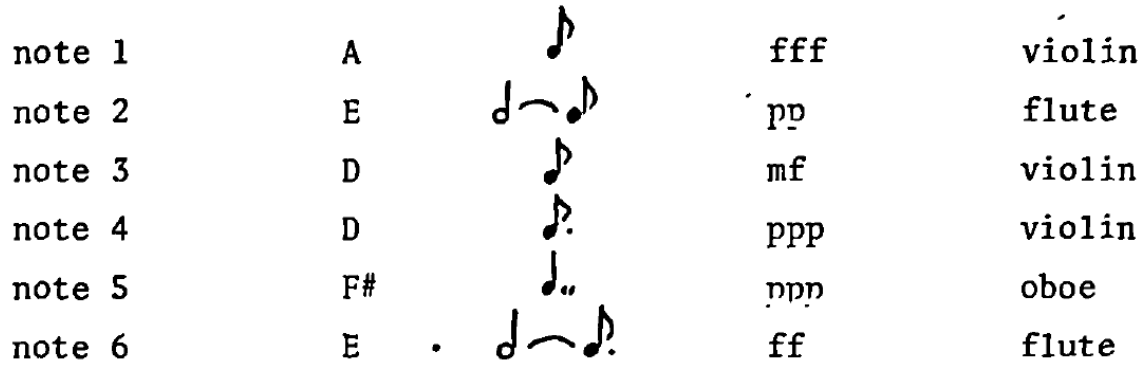


Figure *3.4*: Event Vector Space of Predefined Notes [Jones, 1980]

However, the only rules governing the relations between states and event classes (C_1, C_2, C_3, etc.) are randomly assigned probabilities (seen as decimals in figure 3.2). Such a purely numerical relation cannot easily produce harmonious music. The chords, melodies, and tunes of a song are inherently related in a way that Jones and many other musical theorists find to be close to conversational. *Linguistics*, as it turns out, provides a massive extension of Markov chains using the laws of grammatical structures to form coherent musical phrases. Jones advocates for the branching effect of *space grammars* to be implemented in conjunction with Markov chains (Jones, 1980). Upon my research into the subject of generative grammars, the author found Noam Chomsky's structure and philosophy to be of great interest, with real application to the world of computer music. The field of musical grammars is doubtlessly a highly developed field that merits independent study, however the author's cursory findings are listed below, to build off the work of Jones, who notably describes generative grammars as the most exciting and powerful *extension* of Markov chains (Jones, 1981).

3.2 Generative Grammars

The definition of a formal grammar is a set of production rules, governing the interaction between non-terminal variables and terminal variables. As Jones noted, grammars are natural extensions of stochastic processes and in this section, we examine how to extend stochastic systems to apply to grammars to create an even more powerful generative tool. As we saw with Markov processes in *Analogique A et B*, the audial outcome can often be harsh, discordant, and far from the ordered, principled ideas of traditional music.

Generative grammars offer the benefit of outlining a pre-determined set of rules that govern the interactions of event spaces. These rules are chiefly derived from the study of linguistics. We will apply renowned linguist Noam Chomsky's idea of grammar to musical composition in section 4. The purpose of this section is not to present a comprehensive theory of grammars, but to extend the work presented in Jones' *Compositional Applications of Stochastic Processes (1981)* and structure the Markovian architecture to branch away from the randomness of Xenakis to the order of pleasurable music.

Formally, we will define a generative grammar as follows:

$$G = < V_N, V_T, P, S >$$

Where:

- V_N is a set of *non-terminal* letter variables, the first part of the generative event space

- V_T is a set of *terminal* letter variables, the second part of the generative event space
- P is a set of production rules specifying the combination procedures of terminal and non-terminal variables in the event space
- S is the starting variable that begins the generative sequencing ($S \in V_N$)

The union of sets V_N and V_T is defined as the *alphabet* of the grammar ($A = V_T \cup V_N$). A *string* is any sequence of symbols from the alphabet generated from the alphabet. $A*$ is the set of all possible strings. We will construct a compositional example using the grammar G_1, defined as:

$$G_1 = < V_N, V_T, P, A >$$
$$V_N =< A, B, C >;\ V_T = < a, b, c, d >$$
$$P = 1.\ A \rightarrow aabB$$
$$2.\ B \rightarrow bbC$$
$$3.\ C \rightarrow ccdddD$$
$$4.\ D \rightarrow aabbb$$

For classificational purposes, this is known as a *Chomsky Type 3 finite-state grammar* (Chomsky, 2004), since the sequence terminates finitely with a string of terminal variables each time, as we will see. Using the structure identified previously, we take the grammar starting at A, using production rule 1, production rule 2, and production rule 3 linearly to deconstruct our string, arriving at a set of terminal values:

$$A \rightarrow aaBb \quad by\ production\ rule\ 1$$
$$aaBb \rightarrow aabbbC \quad by\ production\ rule\ 2$$
$$aabbbC \rightarrow aabbbccdddD \quad by\ production\ rule\ 3$$
$$aabbbccdddD \rightarrow aabbbccdddaabb \quad by\ production\ rule\ 4$$

Thus, our grammar G_1 has autonomously produced the string "$aabbbccdddaabbb$". To transcribe this into musical form, we define terminal variables in the same manner that we previously described events in Markov Chains and plot our structure in musical notation. We will assign our first example string to a set of chords that generally resonate well together:

$$a = G\ major\ chord$$
$$b = D\ major\ chord$$
$$c = E\ minor\ chord$$
$$d = B\ minor\ chord$$

Figure 3.5: Mapping the terminal variables of G_1 into musical notes, determined arbitrarily by the author for harmonies

Transposed to musical notation, our grammar produces the resulting melody:

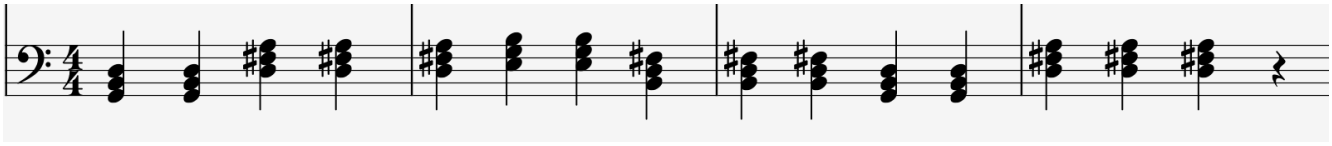


Figure 3.6: String grammar G_1 transposed to musical notation

This generation process operates in precisely the same way as a Markov chain, with the identity of each element in the string being dependent only upon its immediate predecessor. Thus, it should be evident that a Markov chain is *structurally* equivalent to a Type 3 grammar. The main difference is that the grammar can allow for a specific beginning (*S*) and a specific end (last production rule). Furthermore, grammar rules are more easily customizable. We will take a look at a second example of grammar, a *Chomsky Type 3 right-linear grammar*. A right-linear grammar is an infinitely self-generating grammar that has the general production rule: $X \rightarrow yZ$, where $X, Z \epsilon V_N$ and $y \epsilon V_T$. This produces the following generative (right-linear) structure in Figure 3.7:

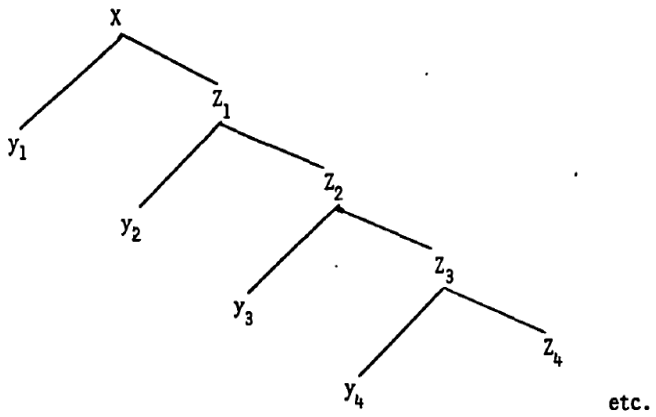


Figure 3.7: Right-Linear grammar structure, generating terminals ad infinitum

Our particular example of a Type 3 right-linear grammar, G_2 will be defined as:

$$G_2 = < V_N, V_T, P, A >$$
$$V_{N2} =< A, B, C >;\ V_{T2} = < a, b, c, d >$$
$$P_2 = 1.\ A \rightarrow aB$$
$$2.\ B \rightarrow daC$$
$$3.\ B \rightarrow abC$$
$$4.\ C \rightarrow cccB$$

Now applying our production rules, P_2, in the following way, we arrive at one of the infinite possible strings, which we map onto a set of 4 musical events, this time singular notes as opposed to chords, and transpose the stringed notes into musical notation in Figure 3.5.

$$A \rightarrow aB \quad by\ production\ rule\ 1$$
$$aB \rightarrow aabC \quad by\ production\ rule\ 3$$
$$aabC \rightarrow aabcccB \quad by\ production\ rule\ 4$$
$$aabcccB \rightarrow aabcccdaC \quad by\ production\ rule\ 2$$
$$aabcccdaC \rightarrow\ aabcccdacccB \quad by\ production\ rule\ 4$$
$$aabcccdacccB \rightarrow aabcccdacccabC \quad by\ production\ rule\ 3$$
$$aabcccdacccabC \rightarrow aabcccdacccabcccB \quad by\ production\ rule\ 4$$

Figure 3.8: The sequential generation of a possible string from the grammatical rules of G_2

$a = A$
$b = B$
$c = F\#$
$d = E$

Figure 3.9: Mapping the terminal variables of G_2 into musical notes, determined arbitrarily by the author for harmonies

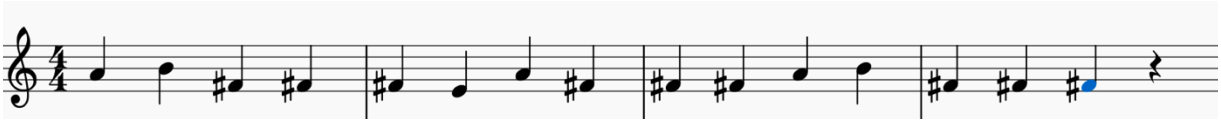


Figure 3.10: String grammar G_2 aabcccdacccabcccB *transposed to musical notation in MuseScore*

We can playfully observe how these grammar sequences, one representing a bass line of chords, and the other representing a tonal melody can be combined. One could easily transform grammar G_1 into an ad infinitum right or left-linear grammar to make the combined sequence below endlessly repeat improvisational mixes as well as the selected tune generated below:

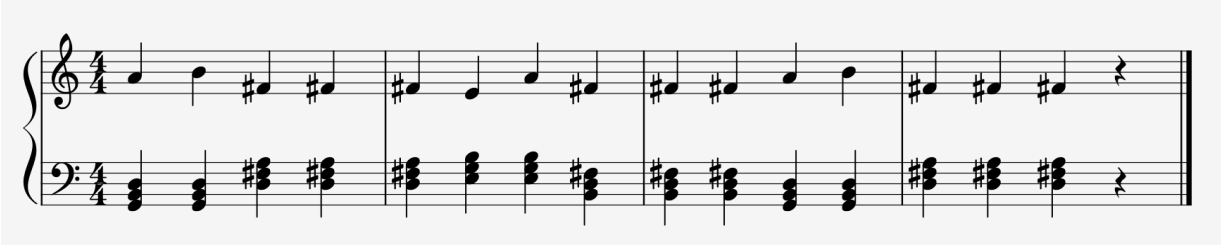


Figure 3.11: Final production of Grammars G_1 and G_2 combined into one musical score

This simple yet elegant 4-measure excerpt is immediately recognizable, harmonious, and, to the author's ears and the few who have heard it, highly pleasing. Although the architecture we used here was comparable in simplicity to that of Xenakis back in 1959, we see how the added specificity of a grammatical structure can add harmony and melody to generative composition, that which was previously difficult to attain in early Markov productions.

More pertinently, grammars also benefit enormously from the ability to become *context free,* meaning they don't depend only on the immediately preceding event, as 1st order Markov chains do. Context free grammars are also able to *self-embed* using recursion not available in Markov chains. For example, a context free grammar can include and process the production rule: $X \rightarrow aXb$ in which the variable X calls itself (an example of grammatical recursion). This property is important in the creation of musical symmetry. For example, a grammar with the production rules is known as a *Type 2 context free grammar* (Chomsky, 2004*)*:

$$1.\ A \rightarrow BAB$$
$$2.\ A \rightarrow a$$
$$3.\ B \rightarrow b$$

Will always have the symmetrical string form $b^n a b^n$ for all positive integers, n. This feature of context free grammars makes them an ideal generative tool for modeling nested, symmetrical properties of

music. This includes not only to traditional aspects of form, but also to the structure of sound textures and timbres (Jones, 1980).

3.3 Grammars and Markov Chains

We have seen how Chomsky Type 3 generative grammars are effectively identical to Markov chains, with the slight difference of being able to choose a start, end, and specific production rules. To contextualize a previous example in grammatical sense, we can map the MTP's of the *Mary Had a Little Lamb remix* in section 1 into grammar G_3, specified as follows (refer back to Fig 1.4):

$$G_3 = < V_N, V_T, P, A >$$
$$V_{N3} =< A, B, C, D >;\ V_{T3} = < a, b, c, d >$$
$$P_3 = A \to aB; A \to aD; B \to bA; B \to bB; B \to bC; C \to cB; C \to cC; D \to dA; D \to dD$$

Our original Markov chain in figure 1.5 is now a grammar, missing only the transitional probabilities, which are now determined by the composer. This example shows the simple translation between Markovian structures: chains and grammars.

We can further specify the grammar G_3 to include a probability distribution over its 9 production rules so that it becomes a *stochastic grammar* (Jones, 1980). This is done simply by adding a distribution D_p over the set of production rules *P*. We will see the use of Markov chains along with stochastic grammars in a novel composition for piano in Section 4.

3.4 Hidden Markov Models and Related Algorithms for Audio Classification

We conclude our discussion of architectural composition with the most modern and technologically applicable extension of the Markov model: Hidden Markov Models (HMM's). Before any further explanation, the author clarifies that while MM's are used in improvisation (seen in section 1) and original generation (section 1, 3), Hidden Markov Models are used primarily in musical classification. They are also the stochastic system with most in common to modern *neural nets* because they are *trainable* on data, as we will see shortly by means of the *Baum-Welch algorithm*, on large audio sample datasets. The field of *audio classification* is wide and intensely studied by machine learning experts, computer scientists, and audio technologists. We are interested in this field for its use in streaming services like Spotify curating song recommendations or Shazam classifying the exact song title from a 5-second recording. The two most commonly applied and effective methods for the task are neural nets and Hidden Markov Models, both subsets of the larger field known as machine learning.

In Section 4, we will examine the construction of a novel audio classification software, built in MATLAB by the author, using Hidden Markov Models and the supporting algorithms that we will summarize below. A note to the reader is made here that as this paper is written for the purpose of original research, focus will be on the explorations in MATLAB rather than on the mathematical theory supporting our findings. However, the related HMM algorithms used for training are indeed beautiful in their efficient construction and we will indulge in a few.

While a Markov chain is used to compute transitional probabilities between sequences of *observable* events, in many cases, event sequences aren't directly observed. They are *hidden*. For example, when listening to a song, you directly observe the notes produced, but the instruments and electronics that

generate the song aren't directly observable. A Hidden Markov Model allows us to create relational probabilities between both observed (like the sound we hear in online streaming apps) and hidden events (like instruments that compose the sounds). The model is constructed as follows:

$$Q = < s_1, s_2, \dots s_N >$$
$$A = < a_{11} \dots a_{ij} \dots a_{NN} >$$
$$O = < o_1, o_2, \dots o_T >$$
$$B = b_i(O_t)$$
$$\pi = < \pi_1, \pi_2, \dots \pi_N >$$

where Q is the set of N event states, A is the transition probability matrix previously studied in section 1 (with a_{ij} representing the probability p(i -> j) such that $\sum_{j=1}^{N} a_{ij} = 1$), and π is the initial probability distribution over the states in Q, π_i being the probability that the Markov chain will start in state *i*. So far, Q, A, and π have been previously seen in regular MM's. HMM's also add the necessary sets to describe the hidden states: *O* is the sequence of T observations each drawn from a vocabulary $V = < v1, v2, \dots vV >$ identical to the vocabulary of grammars previously described, and B is a sequence of *observation likelihoods*, also known as *emission probabilities*, where o_t is an observation from set O generated at state *i.* HMM chains with unspecified observations $(\cdots)$ are given with emission probabilities b_2, b_3, and b_4 in Figure 3.11.

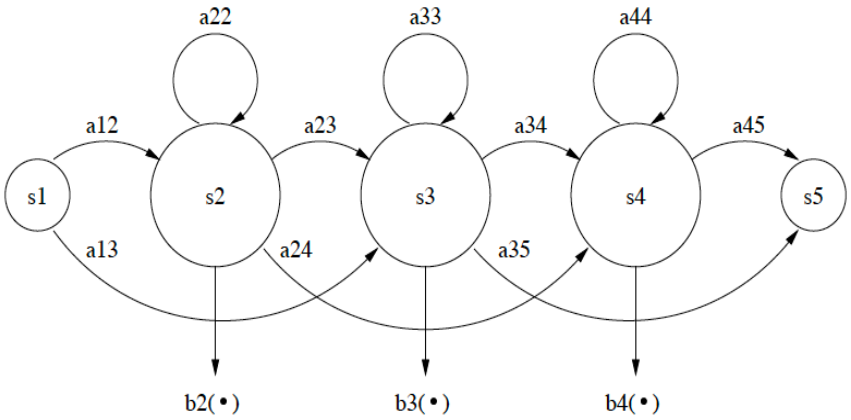


Figure 3.12: A generic HMM graph

As with 1st order MM's, 1st order HMM's also abide by the Markov assumption that the probability of a particular state depends only on the previous state:

$$\boldsymbol{P_{ab} = p(E_{i+1} = a \mid E_i = b)}$$
(Eq. 1.1)

with the added assumption for HMM's that the output observation o_i depends only on the state that produced the observation q_i and not on any other states or observations. An example is as follows: suppose you want to determine Max's mood on a certain day, but you can't directly ask him. Instead, you can observe the type of songs he is listening to on that day and you know what kind of music is *most*

likely to correspond to with each type of mood (happy, energetic, or thoughtful). The music is the observable event and the mood is the hidden event. Now that we have defined the structure, we define the three main types of problems that HMM's are made to address, namely (Sebastian Dalin-Volsing, 2017):

(1) Calculating the probability of a certain output sequence occurring given the HMM
For example, say our HMM states the relation between musical notes (observable set O) and the instruments that generated them (hidden set Q). We hear the notes but are unable to observe the instruments because they are, say, electronically synthesized. We can calculate the probability that a sequence of notes o_i is generated by a certain pairing of instruments q_j. To achieve this, we use what is known as the Forward Algorithm.

(2) Decoding the hidden state sequence Q that produces an observations sequence O
Using our musical HMM previously described, we can precisely identify the pairing of instruments used to generate a musical melody o_i. To achieve this, we use what is known as the Viterbi Algorithm.

(3) Learning A and B from an observation sequence O and the set of states in the HMM
Given our observable sequence O of notes and the set of hidden states Q from the Viterbi algorithm, we can calculate the MTP, A, and the observational likelihoods, B, to build a comprehensive model for how the notes were generated. This is the final step of audio classification. We achieve this by use of the Baum-Welch Algorithm.

3.4.1 The Forward Algorithm for Probability Calculation

For an HMM with *N* hidden states and an observation sequence of *T* observations, there are N^T possible hidden sequences. For effective tasks, where *N* and *T* are large, N^T is usually a very large number and so we cannot hope to compute the total observational likelihood by computing each individual hidden state sequence and summing them (that is, computing each hidden sequence by the step by multiplication shown in Figure 3.13). The forward algorithm replaces this time-consuming, exponential process. The algorithm in use in Problem 1 is best seen diagrammatically. Say we construct an HMM, λ_1 as follows:

$$Q = < H, C >$$
$$O = < (3,1,3) >$$
$$A = \begin{vmatrix} 0.6 & 0.4 \\ 0.5 & 0.5 \end{vmatrix}$$
$$B = \ b_1(o_1) = 0.4;\ b_1(o_2) = 0.2; b_2(o_3) = 0.1$$
$$\pi = [0.8, 0.2]$$

Graphically, this HMM is represented by figure 3.13 (Jurafsky, 2019), where the observational sequence, (3,1,3) is visible at the bottom, generated from the hidden "decisional" layers of Q. Because π gives us two starting points, H or C, there are two branches of hidden layers, each of which we refer to as a "trellis". One can imagine a processing "trolley" of probabilities to make sense of this word. The forward algorithm is a simple summation of the trellis probabilities with $\alpha_t(j)$ representing the probability of being in state j after seeing the first *t* observations. The forward algorithm sums all the probability paths $\alpha_t(j)$ marked in Figure 3.13 to arrive at the final sequence (3,1,3).

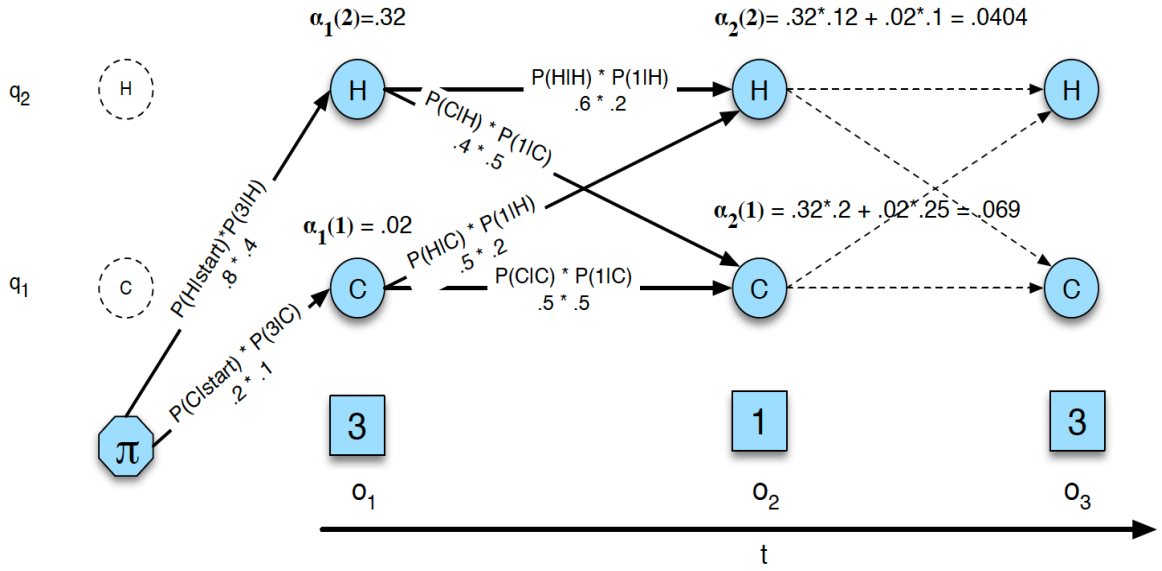


Figure 3.13: A trellis visualization of HMM λ_1 (diagram from Stanford, 2019). The top trellis sums to $\alpha_2(2) = 0.0404$ and the bottom sums to $\alpha_2(1) = 0.069$.

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t)$$

Eq. 3.1 The forward algorithm summation. A full pseudo-code of the algorithm is found in Appendix A. Be careful to note that α_t represents the probability path while a_{ij} denotes the MTP entry in A at (i,j)

$\alpha_t(j)$ gives us the final probability of the sequence (3,1,3) given our HMM λ_1, generated by the forward algorithm. This is useful, the forward algorithm may seem trivial for a small dataset HMM like our example, however, set O can be infinitely large and calculating probabilities individually would be an exponentially increasing time waste. However, knowledge on how likely a melody is doesn't give the whole picture necessary for classification of genre, instrument, or style given the song in question. If we are given a tangible outcome sequence and wish to derive the *exact*, most probable generative sequence, we must look to an important algorithm of slightly higher complexity, the *Viterbi algorithm.*

3.4.2 The Viterbi Algorithm for Decoding

The digital communications revolution of the 20th and 21st century was, in many ways, curated by the Viterbi algorithm, which allows for the dynamic 'decoding' of patterns in speech, genetic sequencing, music, or any other discrete sequence (Durey, 2003). The task of decoding is precisely the problem posed by the Hidden Markov Model: given an observable sequence of events, how can we systematically classify their origins to recognize future similarly generated samples? The Viterbi algorithm operates on the same "trellis" view of a Hidden Markov Model, now that we can calculate the likelihood of set of observations given the model, we want to consider how we can directly observe the most likely sequence of hidden states. We could hope to achieve this by running the forward algorithm for each possible hidden state sequence (HHH, HHC, HCH, etc.) and compute the likelihood of the observation sequence, (3,1,3) given each hidden state sequence. We know now that with N^T many hidden sequences, this would be far too time inefficient. The Viterbi algorithm, like the forward algorithm is another form of dynamic

programming that cuts the exponential processing time. The full pseudo-code of the Viterbi algorithm is provided in the Appendix and is implemented in MATLAB in the adjoining file "*HMM Model.m*".

3.4.3 The Baum-Welch Algorithm for Training

The Baum-Welch Algorithm is the most mathematically challenging of the three algorithms, but also the least necessary to present in its full form, since we can apply a heuristic of an expectation maximization function. We assume the observational set, O, comes from a random process and that the MTP, A and emission probabilities, B are unknown. Because the generative process is random, we start with a randomly generated set of probabilities in a skeleton MTP and emission set B and calculate the probability of the observational set O within the 1st random set. We expect this to be low, since the initial probability distribution set was simply meant to get us started. Now we have two starting parameters, A and B. How do we figure out a more accurate probability for the transitional matrixes? First, we must postulate that transition at every spot in every observed sequence, then see how the probabilities compare to the best probabilities for those observed sequences. We can then use the ratio between old and new location for the updated transition probability. Mathematically, this test and switch method looks prohibitively complex:

$$b_l'(a) = \sum_d \frac{\sum_{t|x_t^d=a} \alpha^d(t,l)\beta^d(t,l)}{P(x^d)}$$

Eq. 3.2: The Baum-Welch Algorithm Trellis Summation (Karpov, 2002)

This is why, for the time being, we stick to our heuristics. The iterative process of switch, evaluate, compare, switch is carried out for all "slots" within the MTP A and the array B, and we evaluate the probability of the observational set O changing using the forward algorithm. We keep repeating the iterative process until the probability stops increasing significantly (until progress in improvement of the learning algorithm has halted). In the MATLAB code of this project, the Baum-Welch algorithm is a simple one-line function, trained on a random data set of 100 and 200 observable sequences. In future experiments, these data sets will be inputted as audio recordings and song samples, so that a fully functional genre-classifier HMM model can be built.

4 Musical Generations

In this section I present several original musical and code-based compositions produced in Ableton Live, MuseScore, and MATLAB with Markov chains to illustrate the theory of the chains previously described. All work in this section is original and the associated files are found in the associated ZIP folder submitted with this paper (Á La Recherche Du Temps Perdu, Say My Name (Markov Remake), Martin Garrix Markov Remake, HMM_Model.m).

Original Markov and Markov-Inspired Compositions

4.1 À La Recherche Du Temps Perdu

A La Recherche du Temps Perdu is a song I composed in Ableton Live 10, inspired by Markov chains. I took on this project as a creative assignment and wanted to experiment with simulating Markovian decisions as a composer. The piece doesn't utilize any formalized Markov chains itself, my compositional method involved looping randomly between three melodies: one upper, one middle, and one lower melody, each of which in turn loops randomly amongst 4-5 notes each, played on a synthesizer piano. The background resonates with an airy synthesizer pad, hollow drums, and spare percussive tones. I named this piece A *La Recherche Du Temps Perdu* for the moodiness I felt composing it in the confinement of home-quarantine.

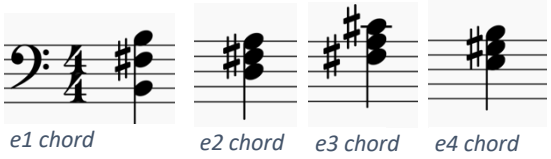
4.2 Say My Name – Markov Remake

This piece is entirely Markov-generated and pulls together concepts discussed in this paper, using event vectors and string grammars to compose a Markov remake of the song *Say My Name* by Odesza, to be performed on piano.

$$\begin{vmatrix} \cdot & e_1 & e_2 & e_3 & e_4 \\ e_1 & 0 & .8 & 0 & .2 \\ e_2 & .1 & .2 & .7 & 0 \\ e_3 & 0 & .2 & .1 & .7 \\ e_4 & .8 & 0 & 0 & .2 \end{vmatrix}$$

Fig 4.1: Chord Matrix – Say My Name

Chords:



$$\begin{vmatrix} \cdot & e_1 & e_2 & e_3 & e_4 & e_5 & e_6 \\ e_1 & .2 & .4 & .4 & 0 & 0 & 0 \\ e_2 & .2 & .2 & .2 & .4 & 0 & 0 \\ e_3 & .3 & .3 & .2 & 0 & .1 & .1 \\ e_4 & 0 & .3 & .1 & .1 & .3 & .2 \\ e_5 & 0 & 0 & 0 & .4 & .2 & .4 \\ e_6 & .2 & .1 & 0 & .2 & .4 & .1 \end{vmatrix}$$

Fig 4.2: Note Matrix – Say My Name

Notes:

e_1 = B; e_2 = C#; e_3 = F_1#; e_4 = A; e_5 = E; e_6 = F_2#

A random chord sequence is generated over 20 steps in MATLAB within the Markov chain of Figure 4.1 to produce the following pattern for the bottom bass line (with 1 standing in for e_1, 2 for e_2, etc.):

$$X_1 = [2,1,2,2,1,2,3,4,1,2,3,3,4,1,2,3,4,1,2,3,4]$$

Likewise, a random chord sequence is generated over a 100 step within the chain of Figure 4.2 to produce the following pattern for the top melody line:

$$X_2 = [3,3,1,2,1,1,1,2,2,3,2,4,2,4,2,4,4,5,4,2,4,6,4,5,6,5,4,2,1,3,1,2,4,5,6,4,5,6,1,3,6,5,4,5,4,4,6,2,2,1,1,3,1,2,3,1,2,1,2,4,2,3,3,2,1,2,4,5,6,5,6,1,1,3,2,1,3,2,1,3,2,4,5,6,5,6,4,2,4,4,6,5,6,1,3,2,3,2,2,4,5,4,5,4]$$

Here we transpose the top melody from X_2 into all eight notes, forming the top line on a piano scale, and the bottom bass chords from X_1 into all half-notes, forming the bottom line to produce the final Markov-generated score in Figure 4.3. The final e_4 chord at the end is stretched over two whole notes for the sake of continuity.

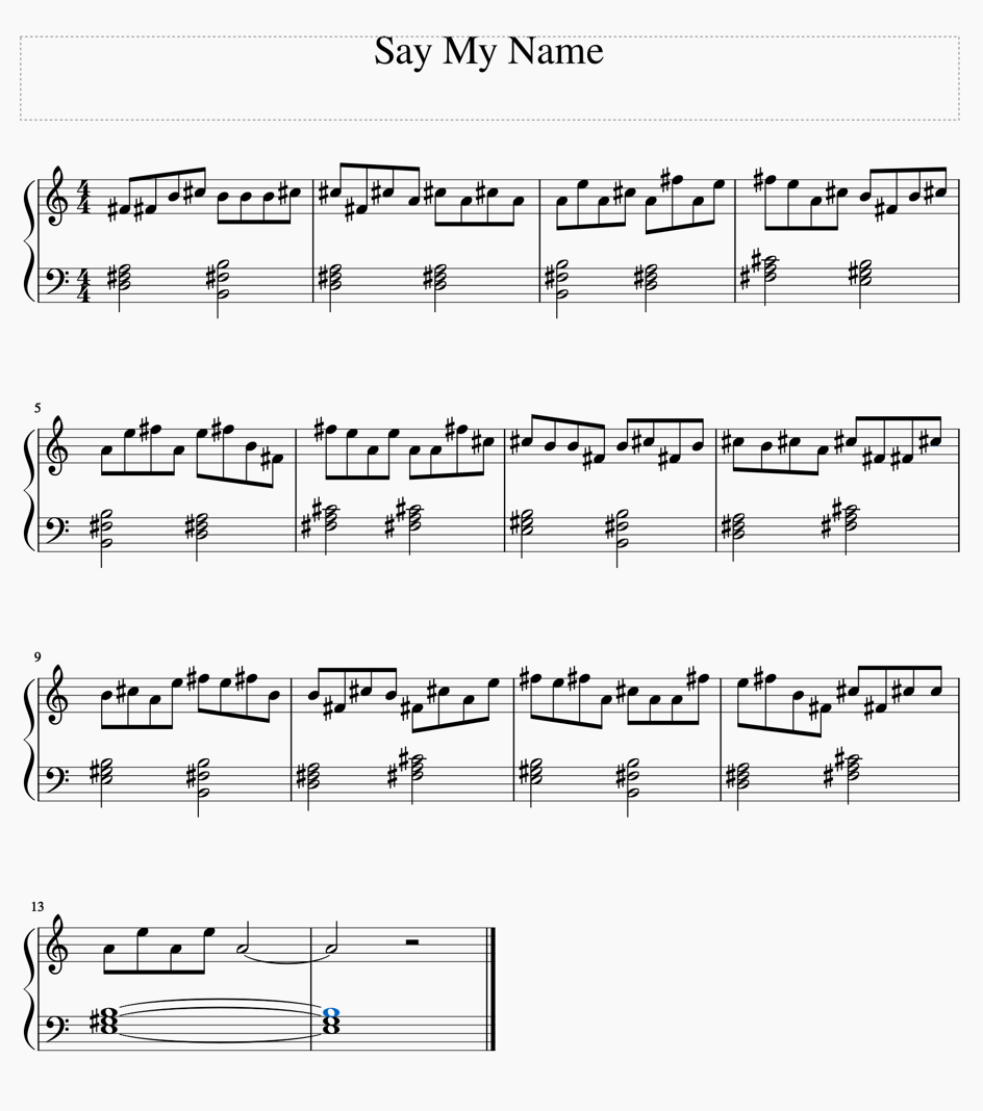


Fig 4.3: Say My Name (Markov Remake) Score (composed in MuseScore)

4.3 Martin Garrix Markov Remake

The transitional decision making of Markov chains inspired me to think of the transitional decisions that DJ's make during live performances. The very basis of DJ'ing involves picking the right songs for the right times and mastering the art of the *transition*. I decided to pursue a personal project and make a Markov chain to simulate the mixing choices of *Martin Garrix*, currently the world's number one attraction in dance and electronic music, and one of my personal favorite artists. I collected song data from 3 of Garrix's hour-long sets and manually cut each of his songs at their transition point. Then, based on the ending melody and note sequence of the previous song, I created a matrix of transition probabilities to simulate a 'live set' of remixes from Martin's sets. Of course, this is not meant to replicate Martin's decision-making process, the genius of which eludes most, but rather to create a model to better understand DJ choices in mixing music. This piece is currently ***in progress*** and so I am sharing a few excerpts from the work so far, attached in the adjoining compressed .ZIP file.

4.4 HMM Training in MATLAB

This mini project applies the theory of Hidden Markov Models discussed in Section 3 used in audio classification. The construction of a full musical genre classifier is a gargantuan task and merits a research paper of its own, so the work presented in this project is a sample of the capabilities of MATLAB (mathematical programming language) to apply the Viterbi, Forward, and Baum-Welch algorithms on musical data (data itself not included). The full MATLAB code is included in Appendix A.

5 Discussion

Markov models are indeed powerful tools for musical composition and, as seen in Section 4, can be incredibly simple and intuitive to use, once the basic syntax has been learnt. Moreover, the model has many important extensions in music, from grammars for musical generation to hidden Markov models for musical classification. Interestingly enough, I found myself studying Markov chains as a result of the natural recurring musical patterns I heard in non-formalized (composer-based) electronic music. Thus, when composing my first piece, *À La Recherche Du Temps Perdu*, I applied the principles of *recurrence* and *randomness* at my discretion. Electronic music is naturally a recurrent field: the basic sounds are composed of compressed, synthesized, and distorted waves, periodic functions which recur over a set time interval. Occasionally, vocals, instruments, and acoustics are transposed onto the raw electronic rhythm, but seeing as I don't have access to these resources at the moment, *À La Recherche Du Temps Perdu* remains purely electronic. The piece is also in its early stages and subject to reworking.

Say My Name, my first purely Markov-produced work, produced surprisingly harmonious results. When compared to another Markovian work like Xenakis' *Analogique A et B*, we note many differences and some similarities. First, while both pieces are indeed traditional, 1st order Markov chains, *Say My Name* makes use of event vectors (chords), while Xenakis' piece uses event scalars (singular notes). Secondly, Xenakis based his compositional process largely off the granular "sound-cloud" concepts he outlined in *Formalized Music (1963)*, while I used only 2 Markov chains. Lastly, while Xenakis created identical Markov MTP's for pitch, intensity, and density, I used different MTP's for melody and bass lines, while treating pitch, intensity, and density as constant.

The Martin Garrix set remake is largely experimental at this stage, but with further work and development, it will be interesting to see how Markov chains can illuminate certain features of transition-based decision-making, especially with discrete data such as songs and speech. I look forward to the continued development of this project.

Finally, our HMM model serves as an implementation of the Viterbi, Baum-Welch, and forward algorithms as discussed in Section 3, as well as the first step in the construction of an audio classification machine. With further development and research, I look forward to building a novel HMM-based classification model that inputs .WAV or .MP3 files to dynamically decode structures and classify artists, styles, and even influences.

6 Conclusion

Conducting research on this project was a fascinating endeavor: what struck me about the use of Markov Models is their simplicity and variety, both in form and in application. In addition to their practical value as computational tools, Markov chains reveal important insights on artistic style, choice, and patterns. It is fascinating to think that such a simple piece of dynamic programming like the Viterbi algorithm can so effectively decode complex signals. With the application of musical grammars previously in composition, and now the application of a speech-processing algorithm to musical processing, we have seen a large overlay of linguistic theory that has been surprisingly vital to our algorithmic composition inquiries. The study of algorithms, language, and music seems intertwined and intersecting at the point of creative generation. While some may classify algorithms at the opposite end of the spectrum from where creativity lies, I refer to the great Noam Chomsky, who so aptly characterizes the goal of my study when he says: *"I think that true creativity means free action within the framework of a system of rules. It is only when you have the combination of freedom and constraint that the question of creativity arises."* (Chomsky, 2004). This is evidenced by the fact that while English follows a strict set of grammatical rules, is confined by a limited alphabet and a limited vocabulary, we are always hearing new sentences and new expressions; in fact, we hardly ever come across the exact same sentence twice. The same is true of music, with its grammar, syntax, and vocabulary of its own. Music is a generative element, and we go through life continually encountering sound and song in novel forms. Change being the only constant, it only makes natural sense to look at music through a statistical viewpoint.

7 Further Research

As a student, I am continually fascinated by the generative and expandable properties of simple algorithms and models that have wide-reaching applicability. I am an admirer and a student of electronic music and architecture as they relate to mathematics for this very reason. Markov chains are inherently beautiful structures that have enormous potential. As mentioned, I am keen on continuing to work toward a fully functioning HMM audio classifier model to build on work I summarized in Section 4. Moreover, I look forward to studying how the application of neural networks in music generation can be enhanced, and how machine learning can be usefully applied in making and analyzing creative artistic works. Furthermore, I am interested in continuing my development in musical production, for I greatly enjoy working with electronics in Ableton. Seeing as the music produced in this project was only my second-ever foray into musical production, I look forward to becoming more well-versed with the techniques and software to continue producing music.

Bibliography

Ames, C. (1987). Automated Composition in Retrospect: 1956-1986. *Leonardo*, 20(2), 169-185. doi:10.2307/1578334

Ames, C. (1989). The Markov Process as a Compositional Model: A Survey and Tutorial. *Leonardo*, 22(2), 175-187. doi:10.2307/1575226

Browne, R. (1973). "Formalized Music: Thought and Mathematics in Composition", by Iannis Xenakis (Book Review). *Notes*, 30(1), 67. doi: 10.2307/896037

Chai, W., & Vercoe, B. (2001). Folk Music Classification using Hidden Markov Models. *Proceedings of International Conference on Artificial Intelligence*, 6(4).

Chomsky, N., & In Otero, C. P. (2004). *Language and politics*.

Dalin-Volsing, S. (2017). Classification of Musical Genres using Hidden Markov Models. [Unpublished Masters' Thesis]. Lund University Student Papers. https://lup.lub.lu.se/student-papers/search/publication/8912380

Di Scipio, A. (1997). The Problem of 2nd-Order Sonorities in Xenakis' Electroacoustic Music. *Organized Sound*, 2(3), 165-178. doi:10.1017/S1355771898009029

Di Scipio, A. (2005). Formalization and Intuition in Analogique A et B. *Definitive Proceedings of the International Symposium Iannis Xenakis (Athens, May 2005)*. http://iannis-xenakis.org/Articles/Di Scipio.pdf

Durey, A. S. (2003). Melody Spotting using Hidden Markov Models [Unpublished Doctoral Dissertation]. Georgia Institute of Technology. https://smartech.gatech.edu/bitstream/handle/1853/5399/durey_adriane_s_200312_phd.pdf

Fernández, J. D., & Vico, F. (2013). AI Methods in Algorithmic Composition: A Comprehensive Survey. *Journal of Artificial Intelligence Research*, 48, 513-582.

Giannakopoulos, T., & Pikrakis, A. (2014). *Introduction to Audio Analysis: A MATLAB® Approach*. Academic Press.

Goree, S. (2017). Structure and Randomness in Iannis Xenakis' Analogique A. [Unpublished Masters' Thesis]. https://samgoree.github.io/assets/Capstone_Thesis.pdf

Hagan, K. L. (2005). Genetic Analysis of Analogique B. *Electroacoustic Music Studies Network, Montreal*. http://www.ems-network.org/IMG/EMS2005-Hagan.pdf

Jones, K. (1980). Computer Assisted Application of Stochastic Structuring Techniques in Musical Composition and Control of Digital Sound Synthesis Systems [Unpublished Doctoral Dissertation]. City University London. https://openaccess.city.ac.uk/id/eprint/7463/

Jones, K. (1981). Compositional Applications of Stochastic Processes. *Computer Music Journal*, 5(2), 45-61. doi:10.2307/3679879

Jurafsky, D., & Martin, J. H. (2019). Speech and Language Processing (draft). October 2019. https://web.stanford.edu/~jurafsky/slp3/

Karpov, I. (2020). Hidden Markov Classification for Musical Genres. [Unpublished Report]. Rice University. http://www.music.mcgill.ca/~ich/classes/mumt611_06/similarity/GenreHMM.pdf

Nierhaus, G. (2008). *Algorithmic Composition: Paradigms of Automated Music Generation*. Springer Vienna.

Pinkerton, R. C. (1956). Information Theory and Melody. *Scientific American*, 194(2), 77-87.

Sandred, Ö., Laurson, M., & Kuuskankare, M. (2009). *Revisiting the Illiac Suite - A Rule-Based Approach to Stochastic Processes*. Sonic Ideas/Ideas Sonicas. 2. 42-46.

Xenakis, Iannis (1971). *Formalized Music Thought and Mathematics in Composition*. Indiana University Press.

Appendix A

Forward Algorithm Pseudo Code

1. Initialization:

$$\alpha_1(j) = \pi_j b_j(o_1) \quad 1 \le j \le N$$

2. Recursion:

$$\alpha_t(j) = \sum_{i=1}^{N} \alpha_{t-1}(i) a_{ij} b_j(o_t); \quad 1 \le j \le N, 1 < t \le T$$

3. Termination:

$$P(O|\lambda) = \sum_{i=1}^{N} \alpha_T(i)$$

Viterbi Algorithm Pseudo Code

1. **Initialization:**

$$\begin{aligned} v_1(j) &= \pi_j b_j(o_1) & 1 \le j \le N \\ bt_1(j) &= 0 & 1 \le j \le N \end{aligned}$$

2. **Recursion**

$$\begin{aligned} v_t(j) &= \max_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t); \quad 1 \le j \le N, 1 < t \le T \\ bt_t(j) &= \operatorname*{argmax}_{i=1}^{N} v_{t-1}(i)\, a_{ij}\, b_j(o_t); \quad 1 \le j \le N, 1 < t \le T \end{aligned}$$

3. **Termination:**

$$\text{The best score:} \quad P* = \max_{i=1}^{N} v_T(i)$$

$$\text{The start of backtrace:} \quad q_T* = \operatorname*{argmax}_{i=1}^{N} v_T(i)$$

Baum-Welch Algorithm Pseudo Code

function FORWARD-BACKWARD(*observations* of len T, *output vocabulary* V, *hidden state set* Q) **returns** *HMM=(A,B)*

initialize A and B

iterate until convergence

E-step

$$\gamma_t(j) = \frac{\alpha_t(j)\beta_t(j)}{\alpha_T(q_F)} \quad \forall\, t \text{ and } j$$

$$\xi_t(i,j) = \frac{\alpha_t(i)\, a_{ij} b_j(o_{t+1})\beta_{t+1}(j)}{\alpha_T(q_F)} \quad \forall\, t,\ i, \text{ and } j$$

M-step

$$\hat{a}_{ij} = \frac{\sum_{t=1}^{T-1} \xi_t(i,j)}{\sum_{t=1}^{T-1}\sum_{k=1}^{N} \xi_t(i,k)}$$

$$\hat{b}_j(v_k) = \frac{\sum_{t=1 s.t.\ O_t=v_k}^{T} \gamma_t(j)}{\sum_{t=1}^{T} \gamma_t(j)}$$

return A, B

Appendix B

Analogique A et B Spectral Analysis

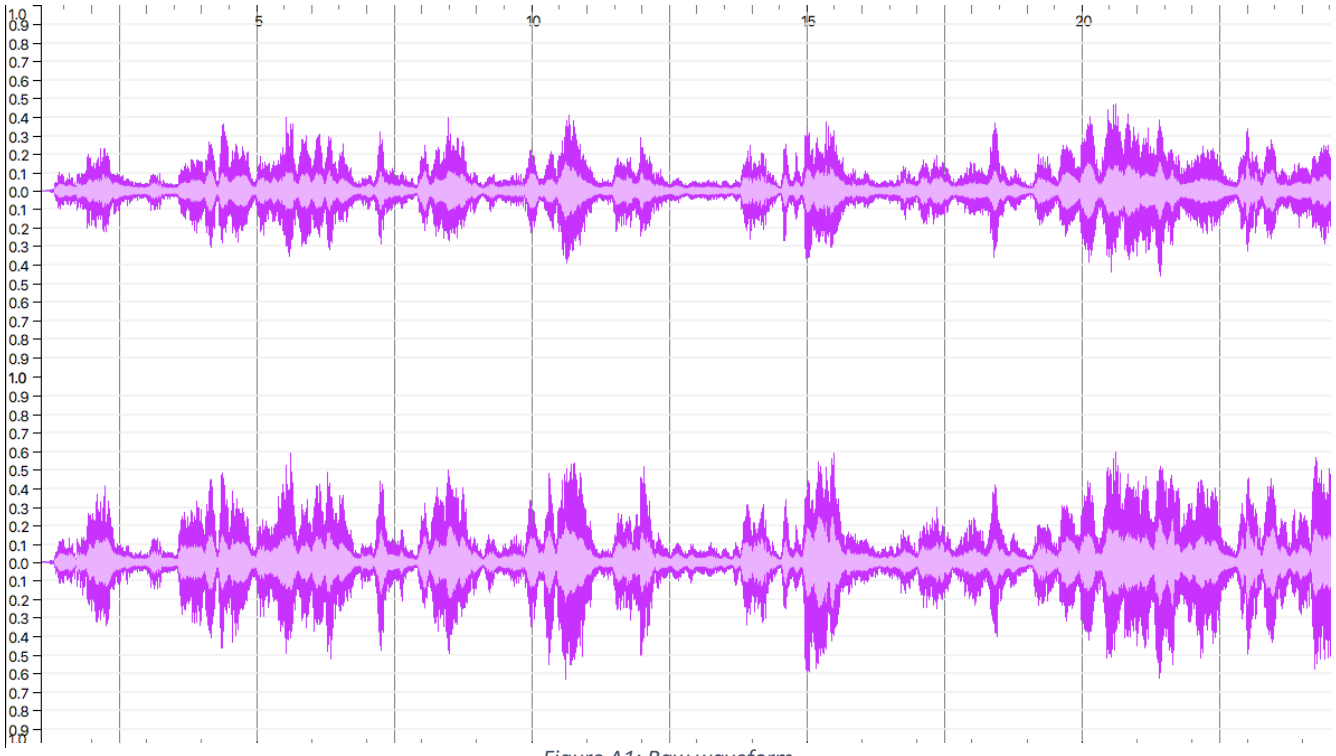


Figure A1: Raw waveform

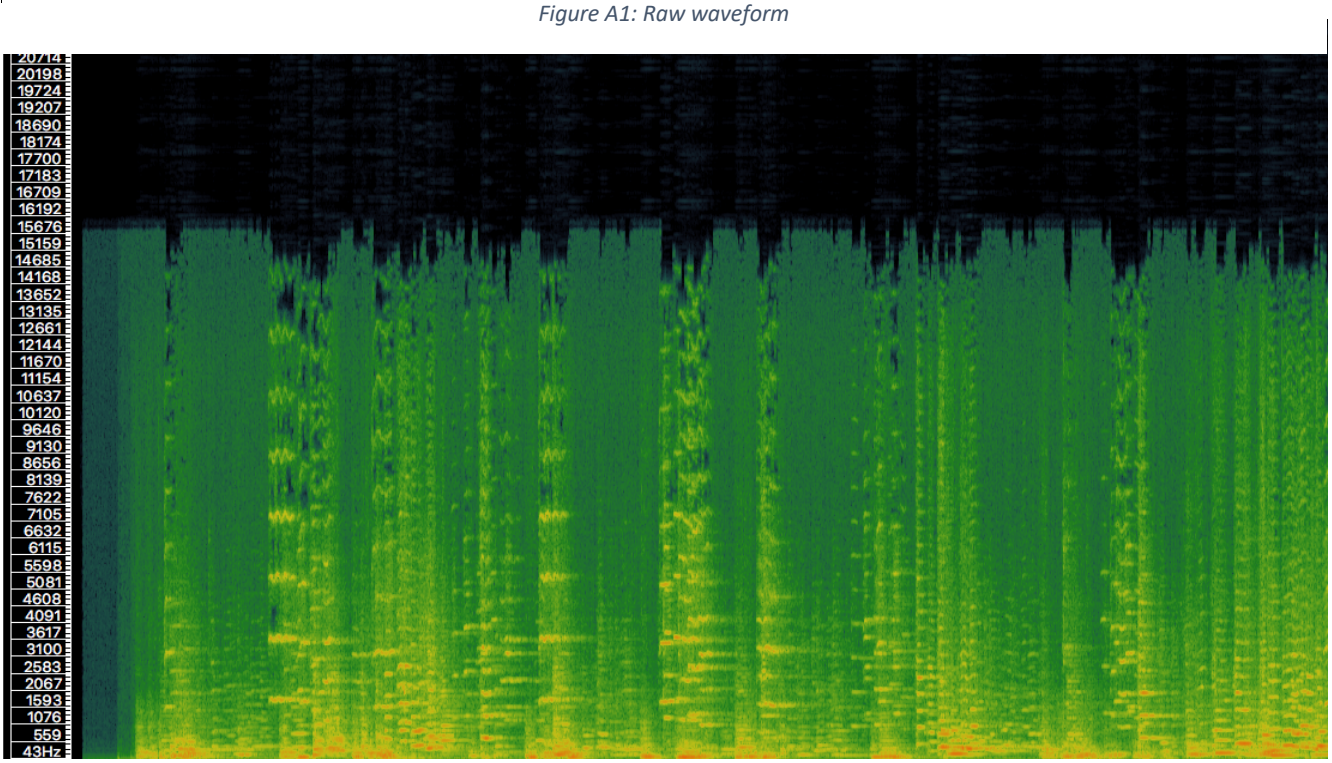


Figure A2: Spectrogram

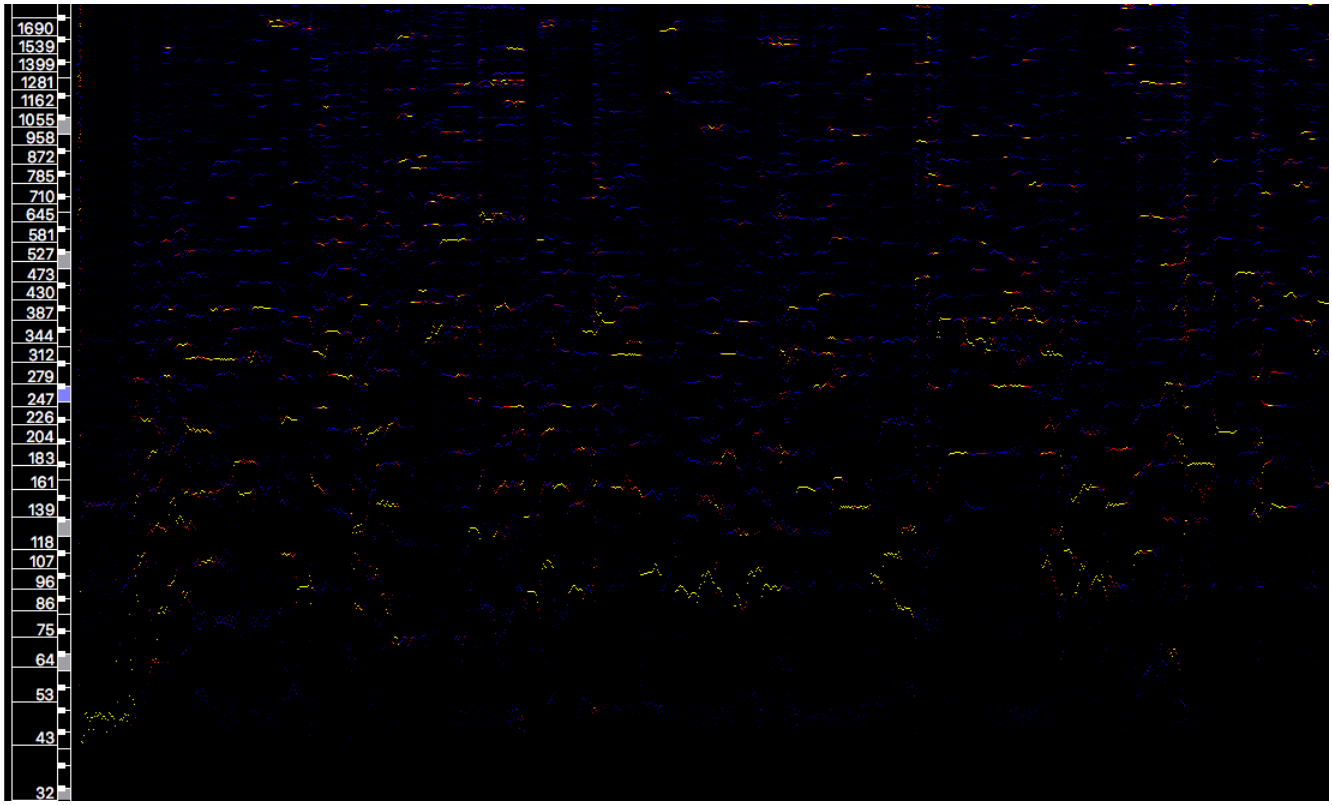


Figure A3: Frequency Spectrogram

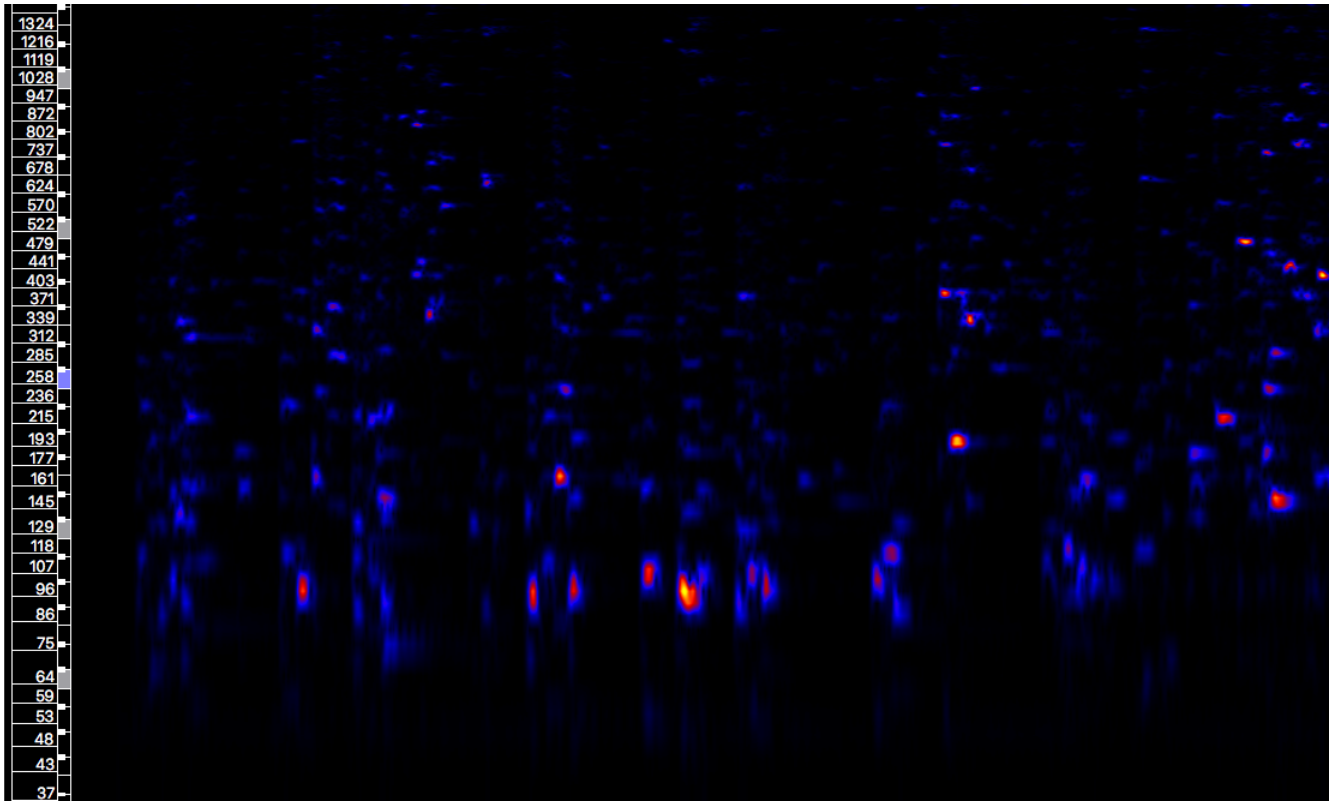


Figure A4: Melodic Range Spectrogram