Improvising Jazz With Markov Chains

Yuval Marom

This report is submitted as partial fulfilment
of the requirements for the Honours Programme of the
Department of Computer Science
The University of Western Australia
1997

Abstract

Markov chains provide a simple way to model music. In this dissertation, I identify some important musical aspects of a jazz improvisation and present several techniques, based on Markov chains, that model these aspects to produce simulations based on human-generated input.

Markov chains are stochastic processes that develop in time in a manner controlled by probabilistic laws and they deal with the interdependence of events. They are capable of capturing global as well as internal, local structures of a particular sequence of states. Because of these properties, Markov chains are suitable to model a jazz improvisation.

Parallels exist between the activities undertaken by a human musician during a jazz improvisation, and the basic mechanisms of Markov chains. The results of this project are musically and statistically satisfying.

Keywords: computer music, Markov chains, simulations, model fitting
CR Categories: G.3, I.2.0, J.5

Acknowledgements

I would like to thank the following people:

My supervisor from Mathematics, Adrian Baddeley, for his constant input and enthusiasm, which have kept me going all year.

My supervisor from Computer Science, C. P. Tsang, for showing interest throughout the year, and always having new ideas (that I haven't thought of previously!)

Leigh Smith, for providing me with some directions early on in the year when I needed it most. Without his endless bibliography (literature and software) this would have taken a lot more than a year.

My friend Julian Bolleter, and fellow Honours student David Allen-Williams, for sharing their musical talent by playing some keyboard jazz for some of my experiments.

The members of *Compmuse* — the UWA Computer Music Research Group — for showing interest every Tuesday at 12:30 (and at other times too, of course).

My parents, Murray and Irit Aitkin, for collectively being my third supervisor. Their ever present support, and especially at thesis-writing time, was invaluable.

And last but definitely not least, my sister Dorit, who took control of things when I was stressing (quite a frequent event). Much appreciated!

Contents

List of Tables

List of Figures

CHAPTER 1

Introduction

Automated composition is a field of computer music that deals with the problem of designing a procedure that will imitate the compositional process of a human composer. As in other forms of art, such as sculpture and painting, the study of automated composition seeks to find out whether the process of artistic creation is mathematical in nature. The answer to this need not be a simple yes or no. The problem can therefore be reformulated as one of finding the extent of the existence of mathematics in the process of artistic creation.

Musical composition is one of many aspects of human behavior and so conclusions drawn from attempts to model it may be important in the study of Artificial Intelligence and Cognitive Science. From a practical point of view, a model of any music can be used to generate music automatically — uses of this might be found in films, television, and elsewhere.

The words "composition" and "improvisation" will be interchanged throughout this dissertation. Improvisation is simply a sub-class of composition. I do not intend to go into full details of the musical differences between these two activities, but a brief discussion is presented in Chapter 2. It will become apparent when I present the techniques that "improvisation" is a more precise description of the process we are trying to automate.

As the title of the project suggests, the musical application used here is jazz. This gives us another indication that we are dealing with improvisation, as jazz is mostly improvised.

1.1 Objective

We can identify at least two main approaches for generating computer-composed music. One approach is a "self-contained" composing procedure that creates the music solely through its algorithms. In other words, a procedure that does not depend on any input is used to *synthesize* new, truly original, music. The second approach is a procedure that creates the music by following some guidelines, or characteristics, that are inherent in an existing piece of music. In contrast with the first approach, this approach can be regarded as trying to *reproduce* music, or style.

The work presented in this dissertation follows the second approach. The objective of this project is, therefore, to generate simulations based on existing jazz pieces. In statistical terms, we start with existing data, fit a model, and then use the model to simulate new data.

1.2 Musical Considerations

Before trying to formalize musical composition, one must first find out what are the main factors that affect the process of composing. Then, one must find a model that will deal with as many of these factors as possible at the same time. A computer program that simply reproduces a particular piece of music is obviously not an adequate one, because this will not be regarded as an original composition. A model of composition in most musical genres, and particularly jazz, must consider two issues:

1. It must capture the style of the composer. If the model cannot capture style, then we have to question its appropriateness. Style is what distinguishes one composer from another, one time-period from another, one genre from another etc. It is something that must be inherited by the model.

2. It should allow for spontaneous actions. It can be argued that the extent to which listeners find a particular piece of music interesting is significantly affected by the piece's unpredictability (though they are not always aware of it). This point is important if we wish the result to be original, and to maintain audience interest.

Notice that the first consideration requires a copying mechanism, whereas the second one requires a degree of randomness, and these conflict somewhat. The model, therefore, must also include a mechanism for determining and controlling the trade-off between the two. These two considerations can be thought of as the factors that affect the process of composition: a composer makes decisions about where the music will go by following particular guidelines according to his/her style, and also by considering new, original, and unanticipated possibilities.

It is relatively easy to find a simple solution that models one of the above two properties. In this dissertation I present a tool, called a *Markov chain*, for modeling both properties at the same time. Markov chains are used to model temporal dependence between events, or states, of processes. Such processes are called *stochastic processes*. It is my view that these kinds of processes describe some of the characteristics of a jazz improvisation, such as the ones mentioned above.

Furthermore, Markov chains present a direct way to model and control the trade-off discussed earlier.

This project can therefore be regarded as an experiment, where the hypothesis under consideration is that Markov chains provide a suitable mathematical model for composing jazz.

1.3 Approaches

Since we are taking the general approach of fitting a model to real data, it is important to recognize the effect of the model's complexity on the ability to fit it to the data, as well as the amount of data required.

At one end of the scale, a highly realistic model would be very complicated, hard to fit, and would require a large amount of input data in order to produce sensible output. An example of this is to consider the n attributes of a jazz improvisation, consider each state of the process to have n dimensions, corresponding to the attributes, and fit a single Markov chain to the states. A simulation based on this would produce a very realistic piece of music, but in the fitting stage, we would need a very large data set in order to avoid reproducing it, or big parts of it, in the simulation stage.

At the other end of the scale, the simplest model would be trivial to fit, would require an insignificant amount of data, but would result in a nearly random simulation, unrelated to the original music. The extreme example of this is a purely random simulation: this would not require any data at all, and would use a simple random number generator to choose the states.

We see that by increasing the complexity of the algorithm, we increase the degree of realism achieved, but also increase the amount of input required, and reduce the ability to apply the algorithm in practice. A decision must be made as to which of these will have greater weight in the design of the procedures.

In this project we design procedures that model the given music as closely as possible, but maintain feasibility in the sense that only "reasonable" amounts of data are needed, and results are relatively easily obtained. In other words, design decisions that simplify the model are taken at the expense of obtaining a more realistic result.

This dissertation presents two different approaches that are described in detail in Chapter 3. They lie somewhere in between the two extremes discussed above. The first approach uses a single Markov chain to model each of a number of attributes of a jazz improvisation, and then combines the different chains into one model that describes the compositional process as a whole. These attributes are pitch, duration, volume, and note/rest events. I have labelled this approach *Simple Markov Chains* (Section 3.3) in the sense that each Markov chain models its intended attribute alone. That is, there is no external factor affecting the development of the model, not even a connection between the different attributes. This is the main limitation of the approach. It assumes independence between the different attributes, as the modeling of each is performed separately.

The second approach is an extension of the first one, where there is an underlying sequence of states controlling the behavior of the Markov chains, which I therefore call *Controlled Markov Chains* (Section 3.4). These states can be thought of as some kinds of external conditions that have a global effect on the improvisation.

CHAPTER 2

Review of the Literature

2.1 Markov Chains as Stochastic Processes

Signal modelling can be described as deterministic and statistical [13], though the two are by no means mutually exclusive. In deterministic modelling we might refer to characterization of a signal with known pre-determined properties. In statistical modelling we might refer to stochastic processes, such as Gaussian, Poisson, and Markov processes [13], to characterize a signal that is unpredictable, to some extent.

The theory of stochastic processes applies to fields such as statistical physics, population growth, communication and control theory, weather forecasting, time series analysis [15], and speech processing [13]. These are processes that develop in time in a manner controlled by probabilistic laws. They deal with the behavior of a collection of random variables and their interdependence [15].

A Markov process is a stochastic process whose future state is only dependent on the current state, and not on any previous states. A Markov chain is a Markov process operating over a finite set of states. It is used to model real life situations such as: queue servicing, population growth, system maintenance (operation, breakdown, and repair of a system), and written language (text analysis) [15].

In fact, a specific application of Markov chains to text analysis has provided the inspiration for this project. I am referring to a program called "Babble" [7] that was written by one of my supervisors, Adrian Baddeley. This program takes a text file as input and fits a Markov chain to the words or characters in the text, as specified by the user. It then simulates the Markov chain to give a new sequence of words or characters. One can find many parallels of text and music, and so it seems that this technique could be applied to music. One such parallel is to treat the states of the Markov chain as notes, rather than words (or characters).

2.2 Automated Composition

2.2.1 History

Many recorded attempts at automating the process of composition exist throughout musical history. However, these efforts have met with continuing resistance from music critics until the latter half of the current century. Charles Ames has conducted an extensive survey of the wide range of techniques that have been employed from 1956 to 1986 [4] in attempts to capture those characteristics that make a particular musical piece so unique. He argues that prior to this period, researchers have not been able to even address the fundamental assumptions about the nature of creativity and the aesthetic purposes of composition, and hence it is not surprising that such efforts have met with constant criticisms.

Arguably the most influential contributor to the field of automated composition has been Iannis Xenakis who argued the need for non-deterministic approaches. He published his ideas in 1971 in a book called "Formalized Music" [19], which is frequently cited in the literature.

2.2.2 Stochastic Techniques for Computer-Music Composition

People have the misconception that statistics is purely about random processes. Jones [12] dismisses critics' concerns that stochastic techniques lack controllability in composition, and argues that in some cases they can produce results that sound far more "ordered" than deterministic systems.

Stochastic techniques have the distinct advantage over other compositional approaches in that they can produce unanticipated possibilities and "fuzzy edges" [12]. This has the effect of "humanizing" computer-generated music, which is frequently criticized for sounding too artificial.

The uses of Markov chains in automated composition have been surveyed in another article by Ames [5]. The majority of them start with a model and try to create a particular style of music, by setting the parameters of the model. As mentioned in the previous chapter, the approach of this project is different. We start with real music, build a model based on it by estimating the parameters from the data, and then try to re-create the style through the new composition. Furthermore, in this project I investigate different uses of the Markov chain, and try to extend its traditional, simplistic implementation.

2.2.3 Composing Programs

Many musical programs exist, and many more will be written, which include composing facilities that require full user interaction, no user interaction, and every level of interaction in between. Several of these programs also include Markov chain facilities for applying standard Markov chains to musical input [1, 2].

It is therefore not the intention of this project to create another such piece of software for creating "Markovian music". The approach of this project is more an experimental one. The usefulness of the application of Markov chains to musical composition is examined in detail, and assessed with both musical and statistical conclusions.

2.3 The Psychology of Improvisation

Pressing [16] acknowledges the scientific importance of the study of improvisation. He claims that improvisation presents us with a "clear version" of how we think, encoded in behavior. Consequently, he believes that this topic should be studied by both psychologists and Artificial Intelligence researchers.

Bailey [8] has interviewed a number of experienced musicians. They claim that the process of improvisation is *not* a completely ad hoc activity that does not need preparation or consideration. In the introduction, I mentioned the need to consider two important factors of an improvisation: style and originality. Pressing [16] states that the psychological sources of these two factors are learned and improvised behavior, respectively.

In a later article, Pressing claims that one of the things that distinguish improvisation from fixed performance (of a composition) is that the former is an iterative cognitive process, where the last component in the process forms the design of the next one [17]. Furthermore, he identifies memory as one of the most important mental factors of musical composition. The figure below presents Pressing's perception of short and long term memory in the process of musical creation. These ideas form the basic principle of the Markov chain. We can immediately draw a parallel of memory to Markov chains, and moreover, a way to distinguish between short and long term memory, namely the order of the chain.

Figure 2.1: Pressing's perception of memory as a mental factor of musical creation

CHAPTER 3

Methods

3.1 Mathematical Background

Definition: A kth order Markov chain (MC) is a stochastic process $\{X_i|i = 1\dots n\}$ with observed values $\{x_i|i = 1\dots n\}$, which operates over a finite set of discrete states, such that

$$\begin{aligned}&\text{Prob}(X_i = x_i|X_{i-1} = x_{i-1}, ..., X_1 = x_1)\\&\quad = \text{Prob}(X_i = x_i|X_{i-1} = x_{i-1}, ..., X_{i-k} = x_{i-k}).\end{aligned}$$

That is, the state of the chain at time i is dependent only on the previous k states [15].

To clarify this dependence property, let us look at a simple case: a first order Markov chain. The state of the chain at any time is dependent only on its previous state:

$$\text{Prob}(X_i = x_i|X_{i-1} = x_{i-1}, ..., X_1 = x_1) = \text{Prob}(X_i = x_i|X_{i-1} = x_{i-1}).$$

A *homogeneous* Markov chain is one where the probabilities above do not depend on time — they are homogeneous with time. The Markov chains used in this project are taken to be homogeneous.

As mentioned in the definition, we are dealing with a sequence of discrete states, coming from a finite state space. We consider transitions between states, and associate a probability for each unique transition. From the definition we know that the probability that the chain will “jump” from state i to state j can be determined through our knowledge of the chain's previous k states only. The transition probabilities are stored in a *transition probability matrix*.

One way of estimating the transition probabilities from the data is to use the observed frequencies of the transitions. This technique is demonstrated below for a first order Markov chain with state space consisting of the letters of the English alphabet.

If m_A is the total number of transitions originating at state A, and of those transitions, n_A terminate at state A, n_B terminate at state B, and so on, then the probability of a transition from A to A is n_A/m_A, the probability of a transition from A to B is n_B/m_A, and so on. Therefore the process of estimating the transition probabilities involves counting transitions to evaluate m_A, n_A, n_B, etc.

For example, the sequence

A B A C C B A B

results in the estimated transition probability matrix

	A	B	C
A	0	$\frac{2}{3}$	$\frac{1}{3}$
B	1	0	0
C	0	$\frac{1}{2}$	$\frac{1}{2}$

Similarly, the estimated transition probability matrix for a second order Markov chain of the above sequence would be

	A	B	C
(A,B)	1	0	0
(B,A)	0	$\frac{1}{2}$	$\frac{1}{2}$
(A,C)	0	0	1
(C,C)	0	1	0
(C,B)	1	0	0

The process of estimating transition probabilities is called "fitting a Markov chain to the data", or "fitting a model to the data". Other fitting procedures exist but the technique described above is the one used in this project. Note the following constraints on the transition probability matrix resulting from applying this technique:

1. each entry must be between 0 and 1
2. each row must sum to 1.

Once we have fitted the Markov chain, we can use it to generate (simulate) new state transitions by using a random number generator that follows the distribution given by the transition probabilities. Thus, we can obtain a simulated sequence of states whose distribution of transitions conforms to that of the original data.

It is important to note that the process described above is wholly data-driven. The generation of the transition probability matrix, and hence its simulation, is based on local patterns in the data (for example, only states in the input state space can be produced in the simulation). The simulation of the Markov chain will replicate those local patterns in the original data, where this is affected by the order of the chain. On a more global scale, however, the simulation is a new, different sequence of states where, again, the extent of this is affected by the order of the chain. A high order Markov chain would capture large sized local patterns in the data and hence produce a simulation that is globally similar to the original data. In comparison, a lower order Markov chain would capture smaller sized local patterns and produce a simulation that is globally less similar to the data.

3.2 Implementation

Before starting to analyze music, one needs to be able to represent it. Since we wish to automate the procedure, the computer must be able to recognize the representation we use. The Musical Instrument Digital Interface (MIDI) format is one such way of storing and manipulating music in a computer.

The first thing we need to do is convert the music into a form that we can manipulate, in order to extract the information we need. There exist many software packages for handling and manipulating MIDI files. I used a program called `MF2T` (MIDI File to Text) [18] which converts a MIDI file to a text file. It is written in the C programming language and runs under the Unix operating system. A typical output of this program is given in Figure 3.1. This output is very convenient for data extraction.

The first line indicates the beginning of the file. The last piece of information on that line tells us that the duration of the time signature is 120. The third line tells us what the time signature is, in this case 4/4. Hence, the duration of a whole note (4/4) is 120. The fourth and fifth lines contain information about other parameters not important to us here. "`MTrk`" indicates the beginning of the track and each line in the output below it represents information for one note event. The first column gives a time stamp of the event; the second column indicates whether it was a note-on or note-off; the third column indicates the pitch of the note; and the final fourth column indicates the velocity with which the note was played. From this information we can deduce the duration of each note by comparing note-on and note-off times; we can deduce pitch and velocity directly from the output; and we can deduce rests also by comparing note-on and note-off times.

Now that we have transformed the music into a suitable form, we can feed it to the fitting procedure which produces a fitted model. The model will produce a simulation that we can then transform back to MIDI format by the reverse program to `MF2T`, called `T2MF` (Text to MIDI File). This process is shown in Figure 3.2 and it identifies the fitting procedure and the simulation procedure. The dotted region signifies where the overall procedure is dependent on the approach we are taking, as described in the following two sub-sections.

```
MFile 0 1 120
0 Meta Copyright ‘‘Copyright 1997 by Yuval Marom’’
0 TimeSig 4/4 24 8
0 Par ch=1 c=7 v=100
0 Par ch=1 c=10 v=64
MTrk
0 On n=f6 v=78
30 Off n=f6 v=0
30 On n=g6 v=85
90 Off n=g6 v=0
90 On n=g6 v=82
105 Off n=g6 v=0
105 On n=b6 v=79
120 Off n=b6 v=0
120 On n=c7 v=92
150 Off n=c7 v=0
180 On n=a6 v=86
225 Off n=a6 v=0
240 On n=g6 v=70
360 Off n=g6 v=0
TrkEnd
```

Figure 3.1: Typical output from MF2T, the MIDI-File-to-Text converter

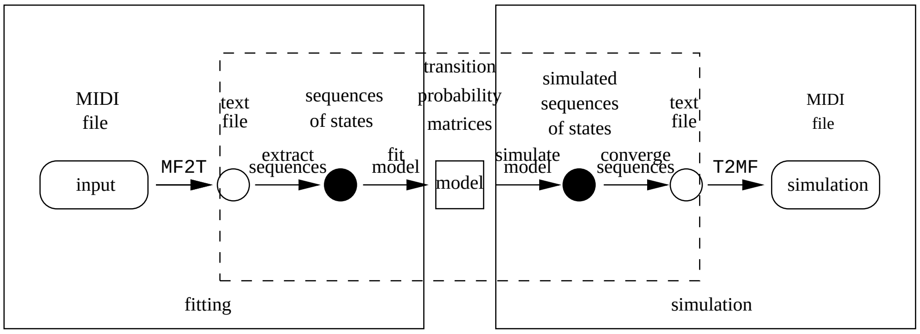


Figure 3.2: An overview of the methodology. The left part is the fitting stage; the right part is the simulation stage. The dotted region is dependent on the approach.

The two approaches use the information from MF2T in a very similar fashion, with minor differences. I have written functions in the C programming language, which use the output from MF2T to extract the sequences described in the following sections. For example, a sequence of pitches extracted from the output shown in Figure 3.1 would be ‘f6’, ‘g6’, ‘g6’, ‘b6’, ‘c7’, ‘a6’, ‘g6’ (the character represents the pitch; the number represents the octave number).

Once these sequences are created, we can fit Markov chains to them and simulate, as described in the previous section. I have written functions in `S-Plus`* that achieve both these tasks.

The simulated sequences need to be converged back into one single file, with lines of note-event information as shown in Figure 3.1, which will serve as input to `T2MF`. I have written functions in `S`-Plus that perform this task by combining information from several sequences (pitch, duration, etc.).

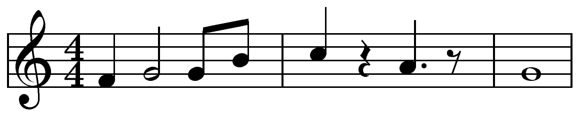
The two approaches use slightly different `C` and `S-Plus` code to perform the specific tasks. The algorithms are described briefly in the sections corresponding to these approaches. All the algorithms have been written by the author except where mentioned otherwise.

3.3 Simple Markov Chains

In this section I describe the first approach employed in this project. The various stages of the development of the model are presented first, and the final model is then discussed.

I refer to the various Markov chains in this section as "simple" in the sense that they are at the top level of the fitting and simulation process. That is, there is no external factor affecting the behaviour of these chains. Furthermore, there is no interaction between the various sub-models of the final model. The issue of interaction is dealt with briefly towards the end of this section, where I remark that one could argue that an interaction does exist.

A small example is used to illustrate the ideas in this section and it is shown in Figure 3.3. It represents the same piece of music that is depicted in Figure 3.1.



Pitch:	F	G	G	B	C		A		G
Duration:	$\frac{1}{4}$	$\frac{1}{2}$	$\frac{1}{8}$	$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{4}$	$\frac{3}{8}$	$\frac{1}{8}$	1
Velocity:	78	85	82	79	92		86		70

Figure 3.3: An example with smallest note duration $\frac{1}{8}$. The pitch and velocities of the notes, and the durations of the notes and rests, are listed. This excerpt corresponds to the same piece of music depicted in Figure 3.1

*`S`-Plus [3, 9] is a statistical, data-analysis, programming language

3.3.1 The Initial Model — Pitch Only

This initial model is very simplistic. The music is treated as a sequence of notes with different pitches, but with equal durations. A rest is interpreted simply as another type of note. That is, one sequence of states is extracted from the music, and the state space consists of the possible pitches, with an extra state added to represent a rest. The duration of each note (and rest) is equal to the smallest note-duration (which can be pre-determined). So any event (note or rest) with duration longer than the smallest note-duration is complemented with consecutive rests to make up that duration. A Markov chain is then fitted to the extracted sequence of states.

The data extracted from the example in Figure 3.3 for this initial version of the model are given as the sequence of states shown in Figure 3.4.

Pitch:	F → R → G → R → R → R → G → B → C → R → R → R → A → R → R → R → G → R → R → R → R → R → R → R

Figure 3.4: The "Pitch Only" approach interprets the excerpt in Figure 3.3 as a sequence of notes with different pitch. A rest is interpreted as a note with pitch "R". The states are assumed to have length equal to the smallest note-duration ($\frac{1}{8}$), so a "long" note is represented by a "short" note followed by rests, and similarly for a "long" rest.

This approach has two limitations:

1. There is no distinction between a note and a rest, resulting in a highly unrealistic implementation, and
2. The simulated music is very monotone and has many "gaps", due to the equal note-durations.

The following two subsections, respectively, discuss how the model is improved to deal with these limitations.

3.3.2 The Improved Model — Notes/Rests

The distinction between a note and a rest is achieved by extracting a sequence of notes and rests (ignoring actual pitch) from the musical excerpt. That is, the state space of the extracted sequence has two states: 'N' and 'R' ('Note' and 'Rest'). A Markov chain is then fitted to this sequence of states.

This is a modification of the initial model, so the "pitch" Markov chain fitted for the initial model is also fitted here, in addition to the note/rest chain. Now, however, we are truly dealing with pitch since a pitch is only assigned to a note event. In other words, the "pitch" chain skips rests. As before, the durations of notes and rests are constant (equal to the smallest note-duration).

The data extracted from the example in Figure 3.3 for the improved model are given as the two sequences of states shown in Figure 3.5.

$$\begin{array}{rl} \text{Notes/Rests:} & \mathrm{N} \to \mathrm{N} \to \mathrm{N} \to \mathrm{N} \to \mathrm{N} \to \mathrm{R} \to \mathrm{R} \to \mathrm{N} \to \mathrm{R} \to \mathrm{N} \\ \text{Pitch:} & \mathrm{F} \to \mathrm{G} \to \mathrm{G} \to \mathrm{B} \to \mathrm{C} \quad \longrightarrow \quad \mathrm{A} \quad \longrightarrow \quad \mathrm{G} \end{array}$$

Figure 3.5: The improved, "Notes/Rests" approach interprets the example of Figure 3.3 as two sequences: a Note/Rests (N/R) sequence, and a pitch sequence. A pitch is only assigned to note events.

The limitation of this approach, as can be seen from Figure 3.5, is that the pitch-sequence does not "hear" the breaks (rests) in the music. That is, a transition exists between notes that are really separated by rests. The question here is whether this is really a limitation. That is, should such a transition exist or not, and does this depend on the length of the separating rest? The model from here on ignores this issue (but if an approach is devised that will consider it, then it should certainly take the second part of the question into account).

3.3.3 Extensions of the Improved Model — Duration & Volume

Like pitch, durations and volumes are assigned only to note events. The duration of rests is treated as before, and rests do not have volume associated with them. Now we have two new sequences of states, in addition to the two described in the previous sub-sections, and hence two new Markov chains.

The state space of the duration sequence consists of multiples of the smallest note-duration. As mentioned in section 3.2, volumes are represented by velocities with which notes are played. These velocities can be any positive integer, so in order to reduce the size of the state space, they are rounded off to multiples of five.

The data extracted from the example in Figure 3.3 for the extended model are given as the sequences of states shown in Figure 3.6.

Note that the smallest note-duration serves as a time stamp. That is, the various events discussed so far are interpreted at each multiple of the smallest note-duration. As the time stamp decreases (that is we use smaller note-durations), the overall representation of the music in the model becomes more realistic.

3.3.4 The Final Model

The final, extended version of the approach is shown in Figure 3.7. The second path (from the top) of Figure 3.7 alone corresponds to the procedure for the ini-

Notes/Rests:	N	→ N	→ N	→ N	→ N	→ R	→ R	→ N	→ R	→ N
Pitch:	F	→ G	→ G	→ B	→ C		⟶	A	⟶	G
Duration:	$\frac{1}{4}$	→ $\frac{1}{2}$	→ $\frac{1}{8}$	→ $\frac{1}{8}$	→ $\frac{1}{4}$		⟶	$\frac{3}{8}$	⟶	1
Volume:	80	→ 85	→ 80	→ 80	→ 90		⟶	85	⟶	70

Figure 3.6: The final, extended version of the approach interprets the example of Figure 3.3 as four sequences: the first two are the same as the ones interpreted by the improved approach (see Figure 3.5). The third is a duration sequence whose states are multiples of the smallest note-duration (in this case $\frac{1}{8}$). The fourth is a velocity sequence whose states are integers rounded off to multiples of five. Durations and velocities are only assigned to note events

tial model (Section 3.3.1), and the top two paths of Figure 3.7 correspond to the procedure for the improved model (Section 3.3.2).

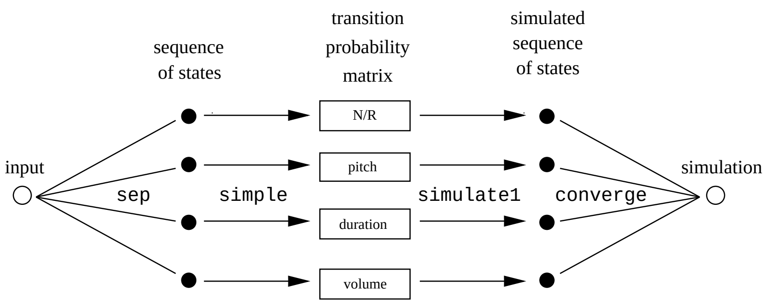


Figure 3.7: The Simple Markov Chain approach. This figure corresponds to the dotted region in Figure 3.2 with this approach

3.3.5 Interactions

The representation of the simulation process shown in Figure 3.7 clearly indicates that there are no interactions between the various aspects of the music discussed so far. Once the information is extracted from the data and separated into the four parts, each part is dealt with separately.

The musical meaning of this is that the decision of what to play next (and how), regarding a particular aspect, is not affected by anything that currently characterizes the music, except for that aspect. For example, the pitch "selector" cannot "hear" current patterns of note and rest runs, does not know the duration of the note about to be played, or how loud (or soft) it will be played. These are only some examples of the interactions between the aspects of the music. Of course, there are many more.

It can be argued that the note/rest states do have an effect on the pitch, duration, and volume states. The note/rest Markov chain determines whether a note or rest is played. Since pitch, duration, and volume (velocity) are only assigned to note events, the note/rest chain has this control over the generation of their states. However, this control is very minimal (if existent), due to the fact that each of the pitch, duration, and volume chains treats the whole piece simply as a collection of consecutive notes. These chains do not have any information of any breaks in the music (due to rests). Therefore, the only piece of information coming from the note/rest simulation that is "seen" by these chains is the overall number of notes played.

3.3.6 Algorithms

The C functions are used within S-Plus using the S-Plus static loading facility. That is, a C function can be called from within S-Plus, and parameter passing is achieved via pointers (hence the C functions do not have a "main" section). Memory must be allocated within S-Plus.

The C function `sep` is used to *separate* the data output by `MF2T` (see Figure 3.1) into four sequences of states as in the example shown in Figure 3.6. This is a monophonic implementation and so simultaneous notes are not accepted (only the first one is dealt with, the rest are ignored). The algorithm reads each line in the output file in turn and deals with the information contained on that line. If the line represents a *note-on*, the information regarding that note is recorded (time, pitch, and velocity); if a note has already been playing for some time (that is, a note that has started in the "past"), then that note is now finished, and its duration is determined. If the line represents a *note-off*, the duration of the last note played is determined. The note/rest, pitch, duration, and velocity information is passed back to the S-Plus calling function through four pointers.

The S-Plus function `simple` is used to build the four *simple* models using the four sequences created by `sep`. It uses another S-Plus function called `transmatrix`, which counts the transitions of a given Markov chain, according to a given order, and returns a *transition matrix*. Probability matrices are then calculated using the count matrices.

The S-Plus function `simulate1` *simulates* a given Markov chain model, for a given simulation length. For each of the sub-models (note/rest, pitch, duration, and velocity) it starts with a random initial state; at each iteration, the entry of the current state in the transition probability matrix is used to determine the next state. A built-in S-Plus function, `sample`, is used here. This function takes a *sample* of entries that have probabilities associated with them, and uses a random number generator together with this distribution of probabilities to select one of the entries. In effect, this is where the simulation takes place.

3.4 Controlled Markov Chains

The main limitation of the *Simple Markov Chains* approach is the lack of interactions between the aspects of the improvisation, as discussed in the previous section. It seems intuitively clear that in a realistic situation some kind of dependence does exist between the various musical events. This is the motivation for developing this new approach. It deals with an external source of control that affects each aspect of the music, thereby introducing a connection between these aspects.

A side effect of applying this technique is that we now have a model of how an improviser is affected by external sources, which seems like an obvious consideration with which we could have begun. Most musical performances (and certainly jazz) consist of a group of people playing different instruments; a solo improvisation, although it is *solo*, is usually accompanied by other instruments (see more on this later).

3.4.1 Hidden Markov Models

A *Hidden Markov Model* [10, 13] consists of a simple Markov chain $S_X = \{X_i | i = 1 \dots n\}$ as outlined in Chapter 3.1, plus an additional, underlying, *hidden* Markov chain $S_Y = \{Y_i | i = 1 \dots n\}$. As before, we are interested in the states of the simple Markov chain. In this scenario, however, these states are functions of the states of the hidden chain. That is,

$$X_i = f(Y_i), i = 1 \dots n. \tag{3.1}$$

The Markov chain S_Y is said to be hidden because we cannot observe its states directly [10]. The opposite is true for the simple chain S_X, and it is therefore called the observed chain. Note that S_X is a Markov chain conditional on S_Y.

Figure 3.8 presents a graphical representation of this idea for a simple case of four states ($n = 4$).

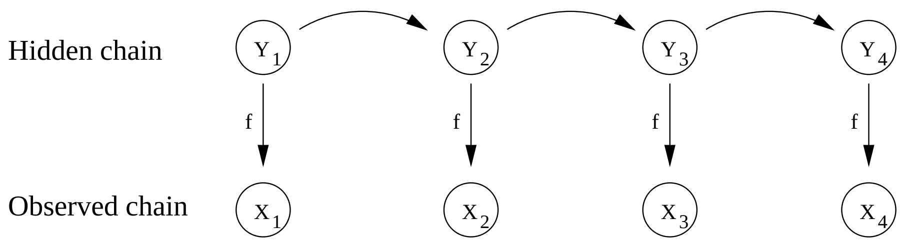


Figure 3.8: A Hidden Markov Model

From the point of view of our aim to model human behavior in the process of composition, this approach seems promising. The hidden states of the process could correspond to some abstract state-of-mind, perhaps, such as mood and feeling.

The current state which the improviser is in will affect what he/she is playing. We can model the behavior of the improviser under each of these abstract states, and we can also model the transitions between the abstract states themselves.

As encouraging as this technique may seem, we encounter a setback before we can even try to implement it. Our general approach is to start with existing data and then fit a model (see Section 1). However, the hidden states are not observable (hidden!), and so this creates a problem. We identify four possible solutions:

1. Devise a different approach where we do not use an existing musical piece: build the model by choosing some hidden states and "composing" the transition probabilities;

2. Use an existing piece, and estimate what the hidden states are;

3. Use an existing piece, and assert what the hidden states are; or

4. Use an existing piece and relax the restriction that the hidden states are unobservable: use some information from the music that can be observed and is believed to have an external effect on the improvisation.

The first situation is a departure from our general approach (and aim). The second is the ideal solution. However, in practice it is extremely hard to estimate any information about hidden states. The third situation is also very hard to implement, and would not guarantee an accurate model. The fourth situation seems like a worth-while simplification of the technique. It can be regarded as being a step before the Hidden Markov Model, and hence any lessons learned from it could be used in the extension to the Hidden Markov Model.

The difference between the fourth situation and the Hidden Markov Model is that the states of the underlying chain are not hidden. This leads to what I call *Controlled Markov Chains.*

3.4.2 Controlled Markov Chains

The introduction of this new type of Markov chain into the model requires clarification of the terminology that will be used to distinguish it from the Markov chains we have used to this point. I will refer to the states of the chain S_Y as *underlying states*, rather than *hidden states* (cf Figure 3.8), for obvious reasons; I will refer to the states of the chain S_X as *simple states*, rather than *observed states* (cf Figure 3.8) because these states are observations of the simple Markov chains discussed in the previous section. Furthermore, the word "observed" is ambiguous as the underlying states are also observed.

We can now estimate

1. the transitions between the underlying states, and

2. the transitions between the simple states within each underlying state.

The second point means that for each of the underlying states, we develop a separate simple Markov chain model as described in Section 3.3. If there are m distinct underlying states, then this approach would produce m transition probability matrices. In the simulation, the current state of the underlying chain determines which of these matrices will be used to produce the simple states. In other words, the underling chain *controls* each of the simple chains.

I will demonstrate the controllability of the simple chains by the underlying chain with the aid of a small example, shown in Figure 3.9. In this example, there is one simple Markov chain, representing note/rest events ('N', 'R'); the underlying chain has two states, '1' and '2' (examples of actual musical events are discussed later).

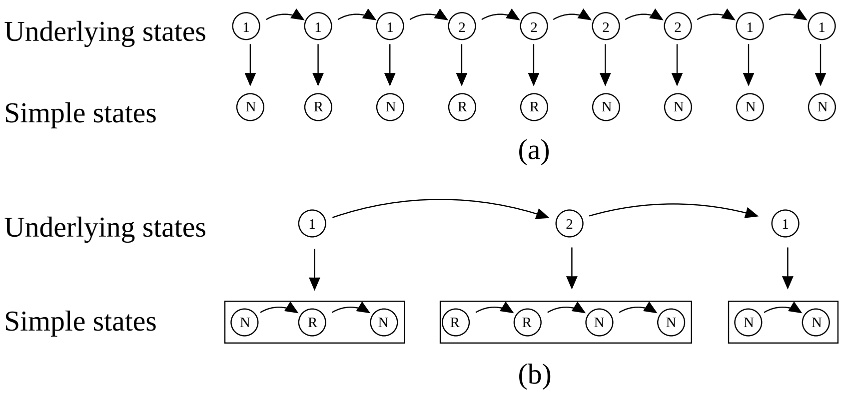


Figure 3.9: Two ways to conceptualize the relationship between the underlying and simple states: (a) each simple state is a direct consequence of an underlying state, and (b) a sequence of simple states is a direct consequence of an underlying state, determined by the duration of that underlying state.

Figure 3.9 presents two ways of conceptualizing the technique. In the first situation (a), we consider the state of the simple chain to be a direct consequence of the underlying chain at each time step. In other words, the state of the underlying chain is checked at each time step to determine how the next simple state will be chosen.

The second situation (b) is a simplification of the first one. Here, some knowledge about the duration of time in which the underlying chain stays at a particular state is known. We now consider the state of the simple chain to be a direct consequence of the underlying chain over this time period. In other words, we only have to refer back to the underlying chain at the end of each time period. In our example, the underlying chain is in state '1' for three time steps, it then moves to state '2' where it stays for four time steps, and finally moves back to state '1' where it stays for two time steps.

The estimated first-order† transition probability matrix for the underlying states is

	1	2
1	$\frac{3}{4}$	$\frac{1}{4}$
2	$\frac{1}{4}$	$\frac{3}{4}$

The estimated first-order transition probability matrix for the simple states controlled by underlying state '1' is

	N	R
N	$\frac{1}{2}$	$\frac{1}{2}$
R	1	0

and the estimated first-order transition probability matrix for the simple states controlled by underlying state '2' is

	N	R
N	1	0
R	$\frac{1}{2}$	$\frac{1}{2}$

Note that for each underlying state, the transition probability matrix is created by gathering the information from all the simple states it controls. This means that an iterative approach must be adopted: whenever we encounter simple states controlled by an underlying state already seen, we simply modify the transition matrix associated with that underlying state to include the new transition information. The transition matrices of all underlying states are updated until all the simple states have been accounted for.

The second situation of the example in Figure 3.9 is preferred for implementation issues such as computation time, amount of storage required, and algorithmic complexity. Of course, this situation depends on the availability of the information regarding the durations of time in which the underlying chain stays at particular states. This is not an over-simplification of the technique as this information can be easily obtained in the following ways:

- One could simulate all the states of the underlying chain first, using transition probabilities, and only then simulate the states of the simple chain(s). Once all the underlying states are known, these durations can be calculated;

†I am using first-order calculations for simplicity

- One could simulate these durations, based on either an empirical distribution calculated from the data, or some other distribution;

- One could fix these durations or, alternatively, use the same durations that occur in the original data.

Note that the first two approaches require that the information regarding the underlying states be estimated from the data, whereas the third one does not.

3.4.3 The Final Model

The previous two sub-sections presented us with several design decisions for the model. I will now summarize the final model by presenting the final decisions taken.

The first decision was to use a non-hidden Markov chain in place of a hidden Markov chain. This decision was discussed in Section 3.4.1.

The second decision concerned obtaining information about the underlying states (presented at the end of the previous sub-section — Section 3.4.2). Before I discuss the chosen approach, I would like to remind the reader that we are trying to build a model of the improvisation. Our focus is still on the simple states (as in the *Simple Markov Chains* approach), which represent the musical events of the improvisation; the underlying states only represent external events (see next sub-section) that affect the improvisation. What is important here is, therefore, the way the underlying states affect the simple states, and not the actual behavior of the underlying states.

I have therefore chosen to use the actual underlying states from the data, in the simulation of the simple states. That is, we do not build a model of the underlying states, and hence we do not simulate them. This naturally reduces the computational complexity.

We can therefore reformulate the procedure as follows:

- we have four simple Markov chains (X_i), as described in Section 3.3;

- we have an observed, underlying sequence of states (Y_i) that is believed to have an external effect on the simple chains;

- the simple chains are "controlled" by the underlying sequence because each chain is conditional on the underlying sequence. That is, for each kth order simple Markov chain we now have

$$\begin{aligned}&\text{Prob}(X_i = x_i | Y_i = y_i, X_{i-1} = x_{i-1}, ..., X_1 = x_1)\\ &\quad = \text{Prob}(X_i = x_i | Y_i = y_i, X_{i-1} = x_{i-1}, ..., X_{i-k} = x_{i-k}).\end{aligned}$$

In other words, (X_i) is a Markov chain given (Y_i).

The process of extracting the information from the data and building the model is very similar to the one used with the *Simple Markov Chains* approach. In fact, we can even use the diagram in Figure 3.7 here. The biggest difference is in the middle part of the diagram — the transition probability matrices. Each one of these is now a set of matrices, as we have seen in the example above (Section 3.4.2). The number of elements in each such set is equal to the number of distinct underlying states.

Another difference (which cannot be seen in Figure 3.7) is that the extracted sequences and simulated sequences have markers to identify where the underlying sequence switches from one state to another. We use these markers to determine the duration of each underlying state.

The last difference is in the code used to perform the various tasks. This is discussed at the end of this section.

3.4.4 Musical Controllers

A jazz improvisation is usually part of an ensemble, and so we do not have to look far to find external sources that affect a solo improvisation. The improviser is playing a solo in the sense that the other instruments provide accompaniment; they help the improviser create the improvisation. We could probably pick any one of these accompanying instruments to be our controlling factors. It is also our intuition that rhythm is a strong factor in the creation of any music.

For these reasons I have decided to pick the drum track as the *musical controller* for the model. The drums form the most dominant part of the rhythm section. We want to find out if the drums "drive" the improvisation.

In each of the examples I use, I have identified distinct drum patterns which change the overall "mood", or "feeling", of the piece. Each of these patterns forms an underlying state in our model. In the simulation, I use exactly the same drum track over the simulated improvisation, as discussed in the previous sub-section.

3.4.5 Algorithms

The function names, parallel to `sep`, `simple`, and `simulate1` — used in the Simple Markov Chains approach (see Figure 3.7 and Section 3.3.6), which are used in the Controlled Markov Chains approach, are `readtracks`, `underlying`, and `simulate2`, respectively.

The output from `MF2T` is slightly different in this scenario as the music is grouped into tracks, corresponding to the underlying states. For each underlying state there is one track, as shown in Figure B.1 — where there are three underlying states and hence three tracks. The values of the underlying tracks are superimposed by the user on the second line (in the example of Figure B.1 they are "1", "2", and "1"). The C function `readtracks` is used to read the information from these tracks. It is similar to `sep` but takes into the account the presence of underlying states, in the form of multi tracks.

The S-Plus function `underlying` is similar to `sep`. It builds a separate model for each *underlying* state. This function also uses `transmatrix` (like `sep`) but in addition, it uses another S-Plus function `editmatrix` to modify a matrix corresponding to an underlying state that has already been seen and hence for which a matrix already exists. This happens when an underlying state is re-visited and we use the music under it to modify the model built when the state was previously visited.

The S-Plus function `simulate2` is similar to `simulate1`. In fact, it performs the tasks in `simulate1` for each underlying state. The underlying states that appear in the original music are used here, in the simulation.

The function that is used to converge all the information back into one file is the same as the one used by the Simple Markov Chains approach — `converge`.

CHAPTER 4

Results

In Chapter 3 I presented a model and discussed its development. Two distinct parts of the development process were identified and allocated separate sections, and separate model names (see Sections 3.3 and 3.4). We note that each modification of the model simply added a new feature to it, rather than created a completely different design. This chapter therefore deals with the statistical significance of these modifications, and the comparison of simulations produced by the different versions of the model, based on some pre-determined measures that are presented in the first section.

It is important to note that comparing single simulations is not very informative. This is due to the stochastic nature of the Markov chain. If we simulate repeatedly, each simulation might be completely different from the one before it. We cannot, therefore, draw conclusions based on a single simulation. A more useful procedure would be to simulate many times to create *samples* of simulations, and draw conclusions regarding these samples. However, in some situations, comparisons of single simulations provide a reasonable approximation to general properties. In other words, in some cases we can reasonably assume that the particular simulation provides a representation that is general enough for our purposes.

The second and third sections deal with the contribution of each of the two approaches — *Simple Markov Chains* and *Controlled Markov Chains* — to the final version of the model.

In the introduction (Chapter 1), I claimed that the Markov chain approach to modeling music provides a mechanism for determining and controlling a tradeoff. This mechanism is the order of the Markov chain. It determines the extent to which the generated music copies the music being modeled (style imitation), and the extent of randomness it exhibits (spontaneous action). The final section of this chapter deals with this inherent factor that is affecting the model throughout its development.

4.1 Statistical Measures of Musical Resemblance

Since we are using transition probability matrices that are calculated from the data to simulate new data, we would like to devise some measures of comparison that do not relate directly to these transitions.

The most obvious measure we can use to describe a piece of music, and one that has been used in other studies [11, 14], is the distribution of pitch. Since we are also modeling other aspects of the music, it is equally important to examine the distribution of each of these (duration, volume, and notes and rests).

Another measure we can use is the distribution of runs of similar, consecutive states. The specific measures we will use are runs of notes and runs of rests. The reason why these measures are informative is that rest runs essentially correspond to the overall rest-durations, which are not described by any other measure we are using; and note runs correspond to the lengths of "phrases" in the music (but they do not give any local information of these "phrases"). For example, the sequence of states shown in Figure 4.1 has note runs represented by the set {1,2,4,1,2,1,2} and rest runs represented by the set {1,4,2,1,1,1}. We can calculate distributions of both of these.

N R N N R R R R N N N N R R N R N N R N R N N

Figure 4.1: Runs of notes and rests

4.2 The Treatment of Notes and Rests

In Section 3.3 we saw that the model was modified three times. However, the major modification was the first one, regarding the treatment of the note/rest relationship. In this section we are concerned with comparing the simulated note/rest information from the two versions of the model.

For clarity, I will call the first version of the model the *naive* version. This is because in this situation, a rest is interpreted as a type of pitch and consequently notes and rests are not properly distinguished, resulting in a (relatively) naive implementation. I will call the second version of the model the *advanced* version.

The test data that I use in this section come from a jazz piece called "Israel", which I obtained from "Jean-Pierre Amringer's Piano-Jazz MIDI Page" [6]. It is quite a "sparse" piece and is played on the piano. The first two bars of this piece are shown in Figure 4.2. The full excerpt can be found in Appendix C. All the Markov chains used here are second-order*.

*See the last section of this chapter for a discussion on the significance of the order of the chain. The reason I am using second order chains with this example is that they achieve the

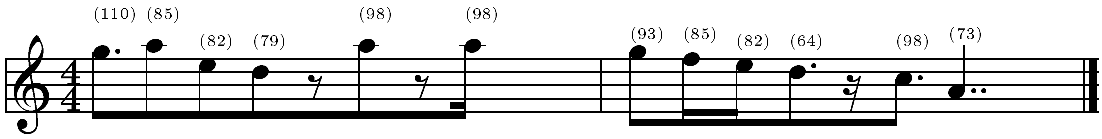


Figure 4.2: "Israel" — first two bars. Numbers in brackets indicate note-velocities. Smallest note-duration is $\frac{1}{16}$.

The sequences of states extracted by the naive and advanced approaches are shown in Figures 4.3 and 4.4, respectively.

Pitch:	g6	R	R	a6	R	e6	R	d6	R	R	R	a6	R	R	R	a6
	g6	R	f6	e6	d6	R	R	R	c6	R	R	a5	R	R	R	R

Figure 4.3: The (partial) interpretation of "Israel" (Figure 4.2) by the naive approach. Numbers indicate octave number.

N/R:	N	N	N	N	R	R	N	R	R	N	N	N	N	N
	R	N	N											
Pitch:	g6	a6	e6	d6	a6	a6	g6	f6	e6	d6	c6	a5		
Duration:	90	60	60	60	60	30	60	30	30	90	90	210		
Volume:	110	80	80	70	90	90	90	80	80	60	90	70		

Figure 4.4: The (partial) interpretation of "Israel" (Figure 4.2) by the advanced approach. A sixteenth note corresponds to a duration of 30.

By comparing the "naive" representation (Figure 4.3) and the "advanced" representation (Figure 4.4) with Figure 4.2, it becomes obvious that the "advanced" representation is more realistic. Therefore, this representation is used to calculate measures of the original data. In other words, when I use the word "original" below, I am referring to the given music, in the "advanced" representation. Obviously, comparisons of the simulations from the naive and advanced models with the "original" data will therefore favour the advanced model. However, we are interested in seeing how the naive model performs relative to the advanced one.

The partial simulations of the naive and advanced models are shown in Figures 4.5 and 4.6, respectively. Full simulations can be found in Appendix C.

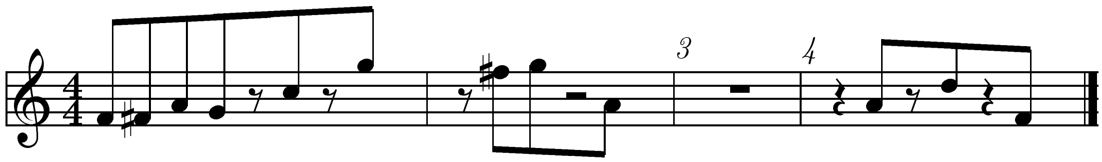


Figure 4.5: Partial simulation of the naive model, based on "Israel". All notes are sixteenth notes, with velocity 90 (an arbitrary choice).

The frequencies of notes and rests from the full data sets are shown in Table 4.1. The naive model simulates a higher frequency of rests (lower frequency of

most musically interesting result.

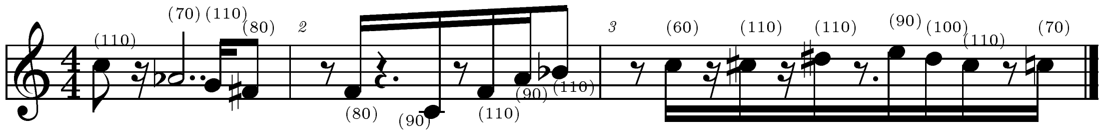


Figure 4.6: Partial simulation of the advanced model, based on "Israel". Numbers in brackets indicate note-velocities

notes) than the advanced model. This is expected due to the fact that the naive approach considers all its states to have durations equal to the smallest note-duration, and consequently interprets longer events as having consecutive sequences of rests, as can be clearly seen in Figure 4.3 (and also discussed in Section 3.3.1).

	% Notes	% Rests
Original	42.0	57.9
Naive Version	28.4	71.6
Advanced Version	43.5	56.5

Table 4.1: Percentages of notes and rests from the original music, and from the simulations of the naive and advanced versions.

Another reason why the naive approach produces such different percentages of notes and rests than the advanced approach can be explained by first recalling that it considers a rest as a type of pitch (See Figure 4.3). If, for example, we are dealing with one octave, then there are 12 possible pitch values, and so the state space consists of those 12 values plus an additional state representing a rest. The (partial) transition probability matrix estimated from our full musical test data, corresponding to one octave, is (figures rounded off to two decimal places):

	C	C#	D	D#	E	F	F#	G	G#	A	A#	B	R
C							.						.
C#							.						.
.							.						.
.							.						.
.							.						.
R	0.02	0.01	0.02	0.02	0.03	0.02	0.01	0.02	0.01	0.02	0.01	0.00	0.81

Due to the presence of "gaps" in the music, the sequence of pitch states includes long sequences of consecutive rests. Therefore, the probability of a transition from a rest to any pitch is much smaller than the probability of staying in a rest state, and so once the chain enters a rest state, it stays there for a relatively long time.

Compare this with the advanced approach, which treats a rest as a state coming from a state space consisting of only two states (compared with 13 states considered by the naive approach in the situation of one octave). The (partial) transition probability matrix estimated for this model is:

	N	R
N	...	...
R	0.36	0.64

The probability of a transition from a rest to a note includes all the small probabilities of transitions from a rest to any pitch in the naive approach. In other words, the probability of leaving a rest state is now higher, producing a smaller percentage of rests.

Figures 4.7 and 4.8 present plots of the distribution of runs of notes and rests, respectively. Note that note runs here consist of any note. That is, there is no distinction between notes with different pitch, duration, and volume. Other possible measures might be distributions of runs of each of these.

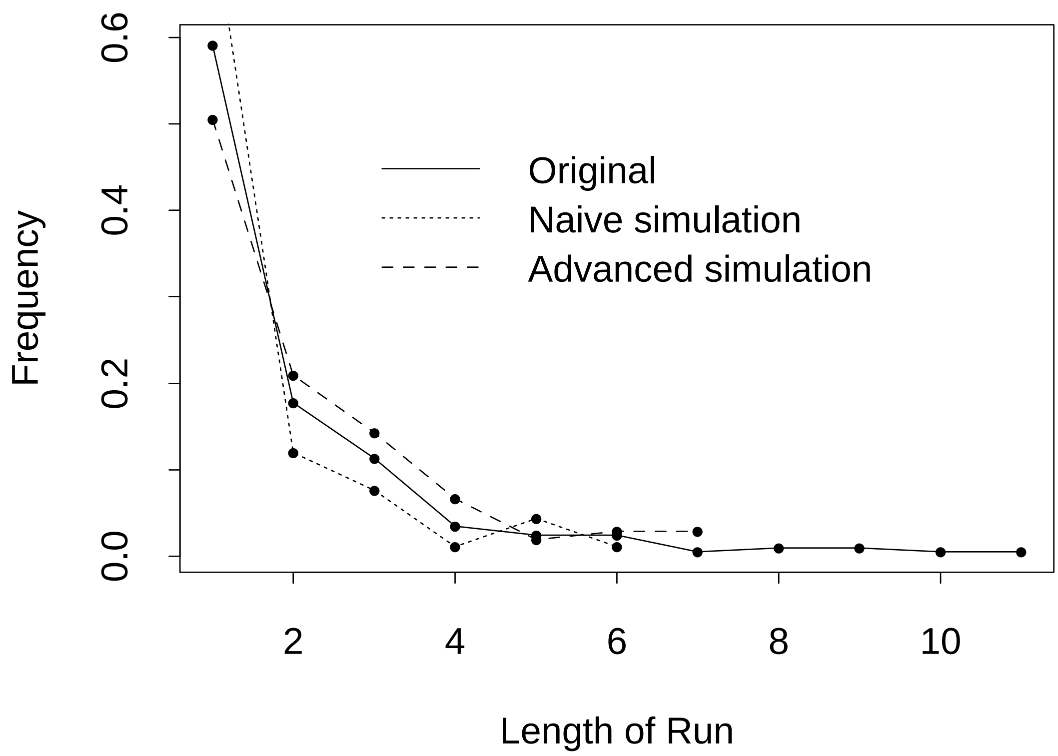


Figure 4.7: Plot of Note Runs.

The three curves in Figure 4.7 are quite similar in shape. An interesting point to note from this plot is that neither version simulates note runs of length greater than seven, which exist in the original data. In this respect, both versions are limited in imitating the original music.

The differences between the two versions of the model are more notable in the distributions of rest runs (Figure 4.8). For short rest runs, of length less than

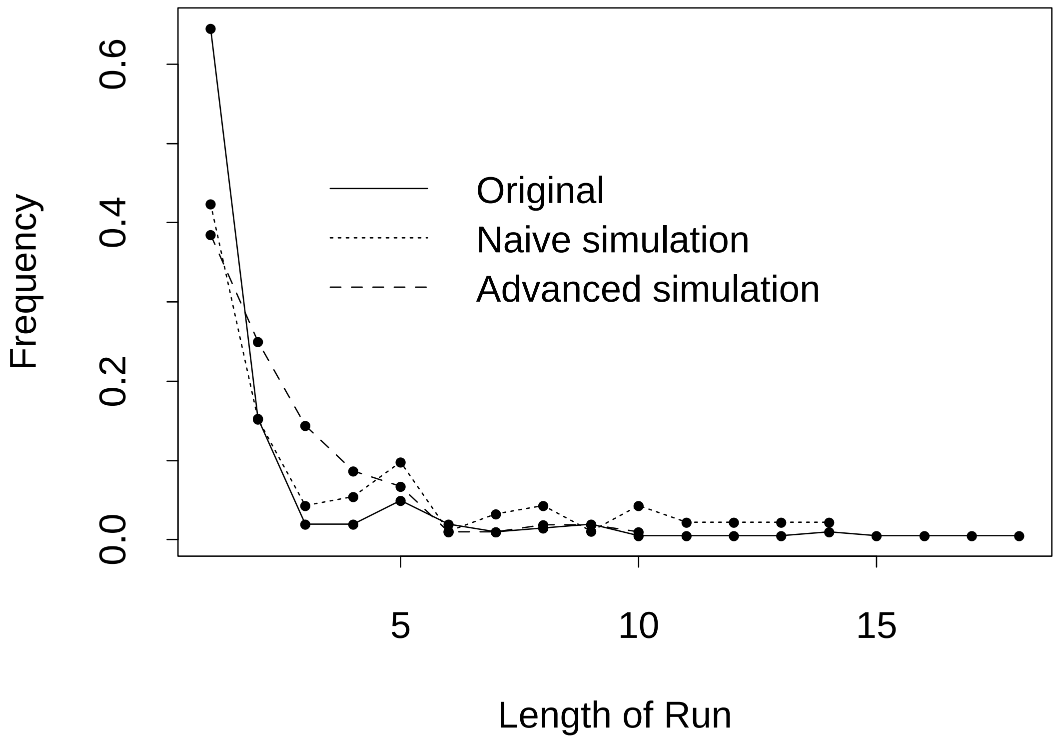


Figure 4.8: Plot of Rest Runs.

five, the naive version seems to model the original data better than the advanced version. However, for longer rest runs of length five to ten, the advanced version seems to model the original data better. The advanced version does not produce any rest runs of length greater than 10, whereas the naive version produces them with quite a high frequency, compared to the original. In fact, the accumulated frequencies of runs of length greater than 10, shown in Table 4.2, indicate that the naive version produces significantly more long-runs than the advanced version and the original sequence. The explanation for this could be attributed to the reasons discussed above.

	%
Original	4.9
Naive Version	15.2
Advanced Version	2.9

Table 4.2: Percentages of rest runs of length greater than 10.

To summarize, we have seen that both approaches are limited, in different ways, in capturing note/rest information from the original data. However, the advanced version is clearly superior and so the extension to the *Controlled Markov Chains* approach, discussed in Chapter 3.4, is therefore based on this approach.

4.3 The Significance of the Underlying States

The modification that was described in detail in Section 3.4 introduced a new level of control into the model. Prior to this modification, the simulation of the music depended solely on the simple Markov chains that dealt with their respective aspect of the music. The purpose of this section is therefore to present some results that might help us determine whether the addition of this new source of dependency significantly affects the model. In this section, I use the word "simple" as shorthand for "Simple Markov Chains", and "controlled" for "Controlled Markov Chains".

As a first comparison between the two models, I will use a statistical technique called the *likelihood ratio test*. Briefly, this procedure involves the following, for each simple chain:

- set the null hypothesis, H_0, to be that the observed sequence is a Markov chain *not dependent* on the underlying states, and the alternative hypothesis, H_1, to be that the observed sequence is a Markov chain *given* the underlying states;
- calculate the likelihood of the sequence under each of H_0 and H_1, using the transition probabilities of each model;
- the ratio of these likelihoods has a known distribution (related to χ^2, under H_0) that we can use to determine whether to accept or reject H_0.

For more details see Appendix A.

The test data that I will use come from an improvisation played by my friend Julian Bolleter. It is called "Jules 1" and is played on the piano. It is accompanied by a drum track, which has two distinct drum patterns — a slow one and a fast one ("1" and "2", say) — that alternate twice (that is, the underlying sequence is "1" "2" "1" "2"). The drum track serves as the controlling signal. The first bar of the music controlled by each drum pattern is shown in Figure 4.9. The full music can be found in Appendix C. As indicated by Figure 4.9 and the full excerpt, the slow drum pattern is on for 20 bars; the fast drum pattern then comes on for 20 bars (bars 21 to 41); the slow drum pattern then comes back on for 21 bars (bars 42 to 63; and finally the fast drum pattern is on for 22 bars (bars 64 to 86).

The simple approach, of course, ignores information from the controlling signal and interprets the musical events as one continuous sequence. The partial musical excerpt of "Jules 1", shown in Figure 4.9, is interpreted by the simple approach as the sequence of states shown in Figure 4.10.

The controlled approach, on the other hand, interprets sets of sequences: one for each underlying state. This is shown in Figure 4.11, which identifies four *tracks*, one for each underlying state. Note the tracks 1 and 3 are controlled by under-

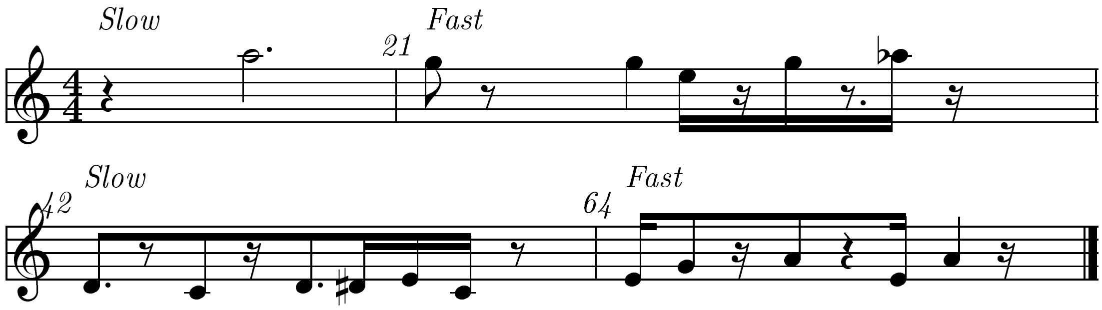


Figure 4.9: "Jules1" — first bar under each underlying drum pattern. "Slow" and "Fast" indicate which underlying drum pattern is currently playing. See Appendix C for full excerpt.

N/R:	R	R	R	R	N	...	N	R	R	N	N	R
	N	R	R	R	...	N	R	R	N	R	N	N
	N	N	R	R	...	N	R	N	N	N	R	R
	N	R	R	R	...							
Pitch:	a6	...	g6	g6	e6	g6	g#6	...	d5	c5	d5	d#5
	e5	c5	...	a5	g5	e5	g5	a5	...			
Duration:	360	...	60	120	30	30	30	...	90	60	90	30
	30	30	...	30	150	30	30	60	...			
Volume:	100	...	120	125	80	70	70	...	125	110	95	110
	100	110	...	125	85	125	90	110	...			

Figure 4.10: The interpretation of "Jules 1" (Figure 4.9) by the simple approach. The number associated with a pitch indicates the octave number. Smallest note-duration is $\frac{1}{16}$, which corresponds to a duration of 30.

lying state "1" (corresponding to the slow drum pattern), and so they both contribute to the creation of its transition probability matrix. Similarly, tracks 2 and 4 are controlled by underlying state "2" (corresponding to the fast drum pattern).

Our model consists of four sub-models, fitted to each of the note/rest, pitch, duration, and volume sequences extracted from the data. It follows that we calculate four likelihood values under H_0, and four under H_1. We perform four likelihood ratio tests. The p-values of the four tests for "Jules 1" are all significant, suggesting that all four chains are dependent on the underlying states. In other words, there is significant evidence from the data that the underlying states have an effect on each of the simple Markov chains.

I will now compare simulations produced by the two models with the original data, using the measures discussed in Section 4.1. The simulations of the simple and controlled models are partially shown in Figures 4.12 and 4.13, respectively. For full excerpts see Appendix C.

Although the two models are statistically different, it is hard to musically notice

N/R:	1:	R	R	R	R	N	…						
	2:	N	R	R	N	N	R	N	R	R	R	…	
	3:	N	R	R	N	R	N	N	N	N	R	R	…
	4:	N	N	R	N	R	R	R	R	N	N	R	…
Pitch:	1:	a6	…										
	2:	g6	g6	e6	g6	g#6	…						
	3:	d5	c5	d5	d#5	e5	c5	…					
	4:	e5	g5	a5	e5	a5	…						
Duration:	1:	360	…										
	2:	60	120	30	30	30	…						
	3:	90	60	90	30	30	30	…					
	4:	30	60	60	30	120	…						
Volume:	1:	100	…										
	2:	120	125	80	70	70	…						
	3:	125	110	95	110	100	110	…					
	4:	125	85	125	90	110	…						

Figure 4.11: The interpretation of "Jules 1" (Figure 4.9) by the controlled approach. The number associated with a pitch indicates the octave number. Smallest note-duration is $\frac{1}{16}$, which corresponds to a duration of 30.

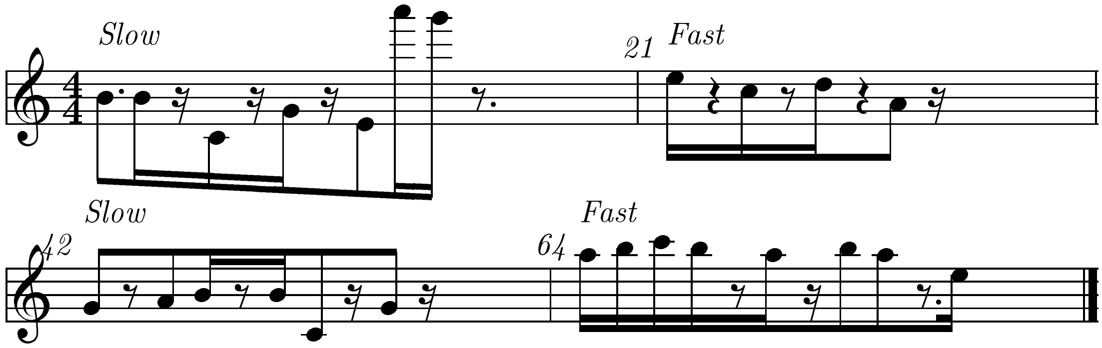


Figure 4.12: Partial simulation of the simple model, based on "Jules 1". The first bar under each underlying drum pattern is shown. "Slow" and "Fast" indicate which underlying drum pattern is currently playing.

distinct differences in their simulations. The reason for this is that the duration of each underlying state (the duration of each drum pattern) is quite large. In other words, the simple chains are controlled by each underlying state for quite a long time, and consequently the effect of a change from one underlying state to the next is not very big. However, at the points where the controlling signal changes its state, one does get an impression of some kind of structure being enforced. Therefore, a more dense controlling signal (that is, changes occurring more frequently) would have more control over the improvisation and indeed a bigger musical effect on the listener.

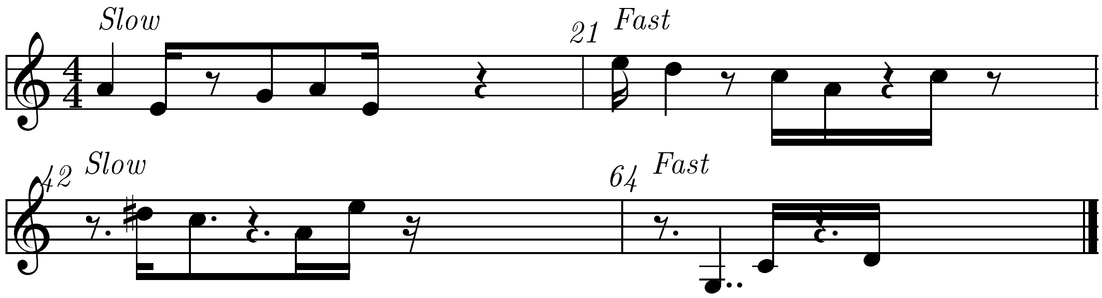


Figure 4.13: Partial simulation of the controlled model, based on "Jules1". The first bar under each underlying drum pattern is shown. "Slow" and "Fast" indicate which underlying drum pattern is currently playing.

Plots of various distributions of the original and the simulations can be found in Appendix B.2. We can see from these plots that the particular simulations of both models captured duration and volume characteristics extremely well. Pitch characteristics were captured reasonably well by both simulations, but not as well as duration and volume. The plots of note and rest runs show that the simulations of both models capture note runs better than rest runs.

4.4 The Significance of the Order of the Chain

As mentioned at the beginning of this chapter, the order of the Markov chain is a property that is affecting the behavior of the chain at all times. In this section we are interested in examining the behavior of the model, subject to different orders. The same musical piece that was used in Section 3.5 is used here ("Israel"), and the *Simple Markov Chains* approach is taken for its simplicity.

Distributions of simulations produced by Markov chains of various orders were compared, using the measures discussed at the beginning of this chapter. These distributions did not provide any evidence that the order of the chain affects the overall resemblance of the simulations with the original music.

However, when we actually listen to the simulated music we notice distinct differences when different orders are used. Small orders produce more random simulations, and high orders produce larger patterns that appear in the original music. This suggests that the statistical measures we have used might not be appropriate to describe musical resemblance. More precisely, the measures we have used are more appropriate for describing the overall, general shape of the music — this is the nature of distributions, which is what these measures are. The differences we can hear arise from more local subtleties in the music that cannot be described by distributions.

We therefore appeal to a different testing procedure that was used in the previous section — the likelihood ratio test. The advantage of using this test is that it is performed on the actual data, rather than the simulations, which therefore gives a more “stable” conclusion. I will test the likelihood of the data when fitted to two Markov chains with consecutive values of order. The purpose of this is to determine whether fitting a kth order Markov chain rather than a $(k-1)$th order Markov chain significantly affects the model. More specifically, we want to know the order of the chain beyond which the fitted model does not change significantly. The results of these tests are presented in Table 4.3. We interpret an entry, k, in Table 4.3 as follows: “changing the order of the chain from i to $i+1$ significantly affects the fitted model, for $1 \leq i < k$; changing the order of the chain from k to $k+1$ does not significantly affect the model”.

	last significant order
Note/Rest	1
Pitch	17
Duration	15
Volume	8

Table 4.3: Converging values of the order of the chain

Another measure we can use, which will help to demonstrate the convergence properties of the order of the chains, is the number of unique k-tuples in the data, where k is the order of the chain. The reason this measure is informative is that the process of fitting the model involves counting transitions, where each transition consists of a source in the form a k-tuple of states and a target in the form of one state (see example in Section 3.1). Therefore, the number of unique k-tuples will affect the total number of unique transitions (in fact, it is an upper bound for it) and hence the size of the fitted model.

Figure 4.14 presents a plot of the number of unique k-tuples of states in each of the note/rest, pitch, duration, and volume sequences. Note that the length of the note/rest sequence is 984 states, and the length of each of the pitch, duration, and volume sequences is 413 states.

The orders at which the pitch, duration, and volume chains converge in Figure 4.14 agree with the figures presented in Table 4.3 regarding the statistical significance of the orders.

The note/rest chain behaves differently from the other three chains. We can see from Figure 4.14 that the size of the note/rest model does not increase by much between consecutive orders, resulting in a non-significant result in each test of the testing procedure above. However, recall that this testing procedure tests Markov chains of consecutive orders. Therefore, the results for the note/rest chain imply that consecutive changes to the order of the chain do not have a significant effect on the fitted model, but they do not say anything about non-consecutive changes to the order. A separate procedure is therefore adopted for this case, where the chains with orders two to 20 are all compared with the first-

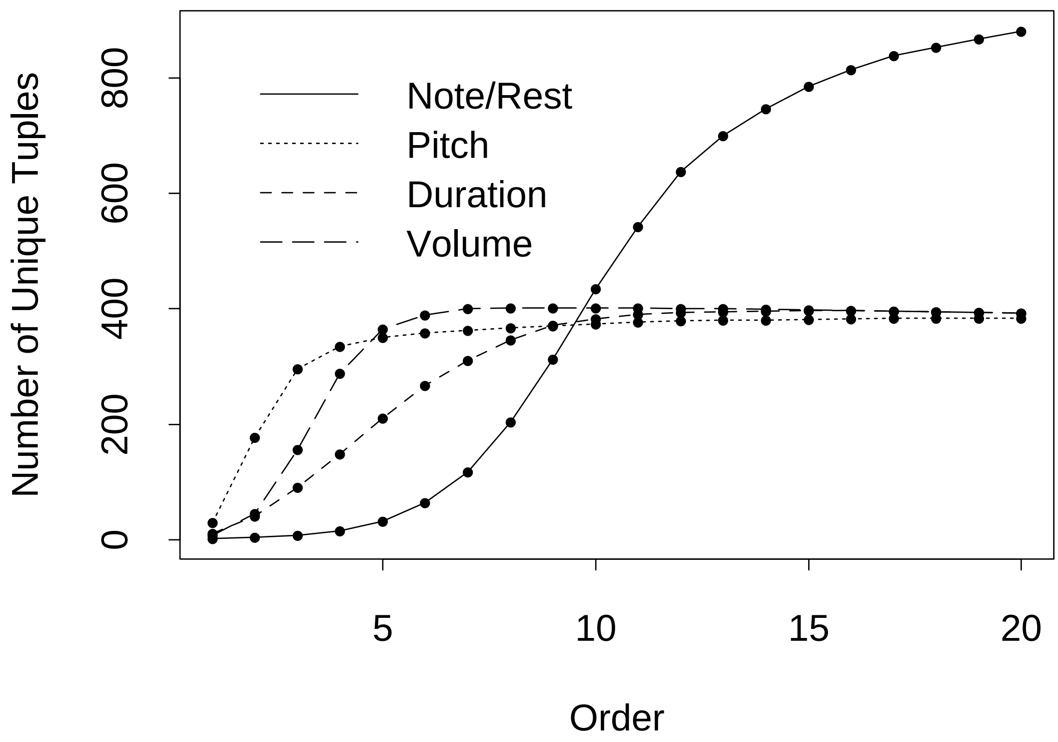


Figure 4.14: Convergence of the number of unique tuples in the data. The length of a tuple is equal to the order of the chain. The length of the note/rest sequence is 984 states, and the length of each of the pitch, duration, and volume sequences is 413 states.

order chain in hope to find a point where a significant change occurs. However, the results of this procedure are all non-significant.

The fact that the order of the note/rest chain does not affect its fitted model may be attributed to the fact that the state space of this chain consists of only two elements, compared to the other three chains whose state space consist of at least 10 elements. With such a small number of states, there is not much room for variations, and an order of one is high enough to capture the essence of the note/rest aspect of the music.

CHAPTER 5

Conclusion and Discussion

We have seen some important developments of the model. The first was the treatment of notes and rests. While this proved to be a significant improvement, it also exposed the fact that this issue needs to be further developed. Our notion of what is a note event and what is a rest event is currently too simplistic. These events "drive" the music and so their representation needs to be complex enough to be realistic. For example, rests need to be distinguished as short rests ("breath of air"), long rests (for the purpose of "resting"), and possibly rests with durations that lie between these two. This raises the idea that durations must be directly associated with rest events, and it seems that notes should also receive this treatment.

The second development of the model was the addition of a controlling mechanism. This gave the music a better sense of structure, which was missing before. It also increased the applicability of the approach, putting it in the context of an ensemble rather than a solo improvisation. Possibly the most important improvement caused by this development is the degree of dependence introduced between the aspects of the music that cannot be assumed to operate independently.

Although statistically the music was affected by the controlling signal, the degree of musical dependence introduced was only slight, suggesting that a more dense controlling signal (that is, more underlying states, and more frequent control) might be required. Alternatively, a different type of controlling signal, corresponding to rhythm, for example, or a chord progression, could be used. In any case, the appropriateness and applicability of this technique pave the way for further extensions based on this idea.

We have seen that the order of the chain significantly affects the fitted model, up to a certain converging point. This convergence is dependent on the actual size of the data, and the size of the state space of the various chains. A specific example was used to show that chains with different state spaces, operating over the same sequence, converge at different points. We have also seen that any order is generally capable of producing simulations with overall, global resemblance to the original music.

Generally, it has been shown in this dissertation that the Markov chain technique is capable of capturing the global characteristics of the improvisation. A limitation of this project is that we have not tested local resemblances, which is statistically harder to do. However, as previously mentioned, when one listens to the simulations produced by these approaches, one can definitely hear local patterns that characterize the original music. Furthermore, one can hear distinct differences in the simulations when chains with different orders are used.

Thus one could conclude from this experiment that some kind of mathematical structure does exist in the process of improvisation. The Markov chain is a simple mathematical model of an important aspect of human behavior: decision-making. We have seen how Markov chains can be used to model decision making affected by memory alone (through the *Simple Markov Chains* approach), and by external conditions as well (through the *Controlled Markov Chains* approach).

From a practical point of view we have developed a method, whereby we can fit a model to a given music *once*, and then use it to simulate *repeatedly*, producing a new and original piece of music each time. This is advantageous in areas that require continuous music, where human intervention is undesirable, such as telephone hold services, restaurants, hotel lobbies, shops, elevators, etc. However, the implementation needs to be converted to real time before the techniques can be used in these areas.

The music produced by the techniques discussed in this dissertation can be categorized as "pleasant". Naturally, the "human touch" is missing, but this is to be expected in the automation of any task, by definition. Listeners who have heard the results of this project have often mistaken the simulated music for the "real" music. The techniques have been successful at capturing and reproducing the musical elements that make a jazz improvisation exciting and unpredictable.

References

1. CommonMusic. An object oriented music composition environment implemented in Common Lisp. Runs on NeXT, Macintosh, SGI, SUN, and I386. `http://www.eecs.umich.edu/~mrozek/nextmusic/CommonMusic.html`.

2. KeyKit. A programming language and graphical user interface for MIDI. Runs on Windows 95 and Linux. `http://www.nosuch.com/keykit`.

3. S-Plus. A statistical, data-analysis, programming language. `http://www.mathsoft.com/splus`.

4. Charles Ames. Automated composition in retrospect: 1956–1986. *Leonardo*, 20(2):169–185, 1987.

5. Charles Ames. The Markov process as a compositional model: A survey and tutorial. *Leonardo*, 22(2):175–187, 1989.

6. Jean-Pierre Amringer. Jean-Pierre Amringer's piano-jazz MIDI page. Available through the World Wide Web at: `http://www.worldnet.fr/~jpa/midi.shtml`.

7. Adrian Baddeley. Babble, 1988. A program that fits a Markov chain to words or characters in a text file, and simulates new text. E-mail: `adrian@maths.uwa.edu.au`.

8. Derek Bailey. *Improvisation. Its Nature and Practice in Music.* Derek Bailey, 1980.

9. R.A. Becker, J.M. Chambers, and A.R. Wilks. *The NEW S Language.* 1988.

10. Robert J. Elliott, Lakhdar Aggoun, and John B. Moore. *Hidden Markov Models. Estimation and Control.* Springer-Verlag New York, Inc., New York, 1995.

11. Topi Jarvinen. Tonal hierarchies in jazz improvisation. *Music Perception*, 12(4):415–437, 1995.

12. Kevin Jones. Compositional applications of stochastic processes. *Computer Music Journal*, 5(2):381–397, 1981.

13. David H. Kil and Frances B. Shin. Hidden markov models. In *Pattern Recognition and Prediction With Applications To Signal Characterization*, chapter 5, pages 155–86. American Institute of Physics Press, New York, 1996.

14. C. L. Kruhhansl. *Cognitive Foundation of Musical Pitch.* Oxford University Press, New York, 1990.

15. Emanuel Parzen. *Stochastic Processes.* Holden-Day, Inc., San Francisco, 1962.

16. Jeff Pressing. Cognitive processes in improvisation. In W. R. Crozier and A. J. Chapman, editors, *Cognitive Processes in the Perception of Art*, pages 345–63. Elsevier Science Publishers B.V., Amsterdam, 1984.

17. Jeff Pressing. Improvisation: Methods and models. In John A. Sloboda, editor, *Generative Processes in Music: The Psychology of Performance, Improvisation and Composition*, chapter 7, pages 129–78. Oxford University Press, New York, 1988.

18. Piet van Oostrum. MF2T for Unix. Dept. of Computer Science, Utrecht University. P.O. Box 80.089, 2508 TB Utrecht, The Netherlands. E-mail: `piet@cs.ruu.nl`. A C-program for converting MIDI files to text files.

19. Iannis Xenakis. *Formalised Music.* Indiana University Press, Bloomington, Ind, 1971.

APPENDIX A

Statistical Testing

Likelihood Ratio Test

We use the following statistical procedure to determine the significance of adding a controlling signal to a *Simple Markov Chain* model. This procedure is used in Chapter 4 (Sections 4.3 and 4.4).

Let the modelled music be represented by the sequence of states

$$X_1 X_2 .... X_N$$

Set the hypothesis as follows:

H_0: music being modelled is a ("simple") Markov chain not dependent on the controlling signal

H_1: music being modelled is a Markov chain given the controlling signal

Reformulate them in terms of models:

Model$_0$: music is described by a single Markov chain with transition probability matrix (P_{ij})

Model$_1$: music is described by m Markov chains, $(P_{ij}^{(0)})$, $(P_{ij}^{(1)})$, $\ldots$, $(P_{ij}^{(m)})$, corresponding to m controlling states.

Define the likelihood of Model$_0$ as:

$$L((P_{ij})) = P_{X_1 X_2} P_{X_2 X_3} \ldots P_{X_{N-1} X_N}$$

If we estimate P_{ij} by $\hat{P}_{ij}$ (from the data), then set the (maximum) likelihood, l_0, of Model$_0$, as

$$l_0 = L((\hat{P}_{ij}))$$

Similarly, the maximum likelihood of Model$_1$ is:

$$L((P_{ij}^{(0)}), (P_{ij}^{(1)}), \ldots, (P_{ij}^{(m)})) = P_{X_1X_2}^{(0)} P_{X_2X_3}^{(0)} \ldots P_{X_hX_i}^{(k)} P_{X_iX_j}^{(k)} \ldots P_{X_{N-1}X_N}^{(m)}$$

and we estimate

$$(P_{ij}^{(0)}), (P_{ij}^{(1)}), \ldots, (P_{ij}^{(m)})$$

by

$$(\hat{P}_{ij}^{(0)}), (\hat{P}_{ij}^{(1)}), \ldots, (\hat{P}_{ij}^{(m)}).$$

(from the data) and set the (maximum) likelihood, l_1, of Model$_1$ as:

$$l_1 = L((\hat{P}_{ij}^{(0)}), (\hat{P}_{ij}^{(1)}), \ldots, (\hat{P}_{ij}^{(m)}))$$

The theory says that under H_0, two times the log of the ratio of l_0 and l_1 has asymptotically a chi-squared distribution. That is

$$2\log(l_1/l_0) \sim \chi^2_\nu$$

where ν is the degrees of freedom, given by the difference in the number of parameters in the two models.

APPENDIX B

Figures

B.1 Multi-track output from MF2T

```
2
1 2 1
MFile 1 4 120
MTrk
0 Meta Copyright "Copyright \xa9 1997 by Yuval Marom"
0 TimeSig 4/4 24 8
TrkEnd
MTrk
0 Par ch=5 c=7 v=125
0 Par ch=5 c=10 v=70
420 On 64 108
480 On 64 0
480 On 69 101
TrkEnd
MTrk
0 Par ch=5 c=7 v=125
0 Par ch=5 c=10 v=70
0 On 69 117
150 On 69 0
180 On 73 104
240 On 73 0
TrkEnd
MTrk
0 Par ch=5 c=7 v=125
0 Par ch=5 c=10 v=70
0 On 57 100
60 On 57 0
60 On 59 100
120 On 59 0
TrkEnd
```

Figure B.1: Multi-track output from MF2T corresponding to the presence of underlying states. The first two lines are super-imposed by the user. The number on the first line is the number of distinct underlying states. The sequence of numbers on the second line indicate under which underling state the tracks are.

B.2 Distribution Plots

Various distributions of “Jules 1” and its simulations, discussed in Section 4.3, are presented here.

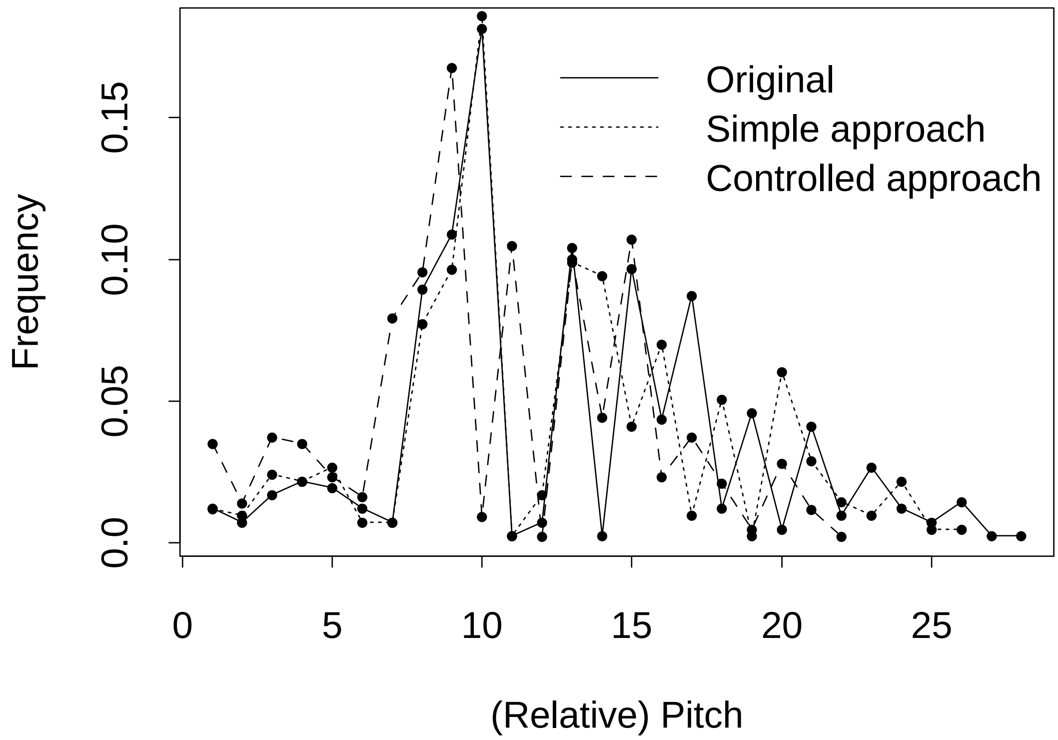


Figure B.2: Distribution of pitch in the original data, and the simulations of the simple and controlled models

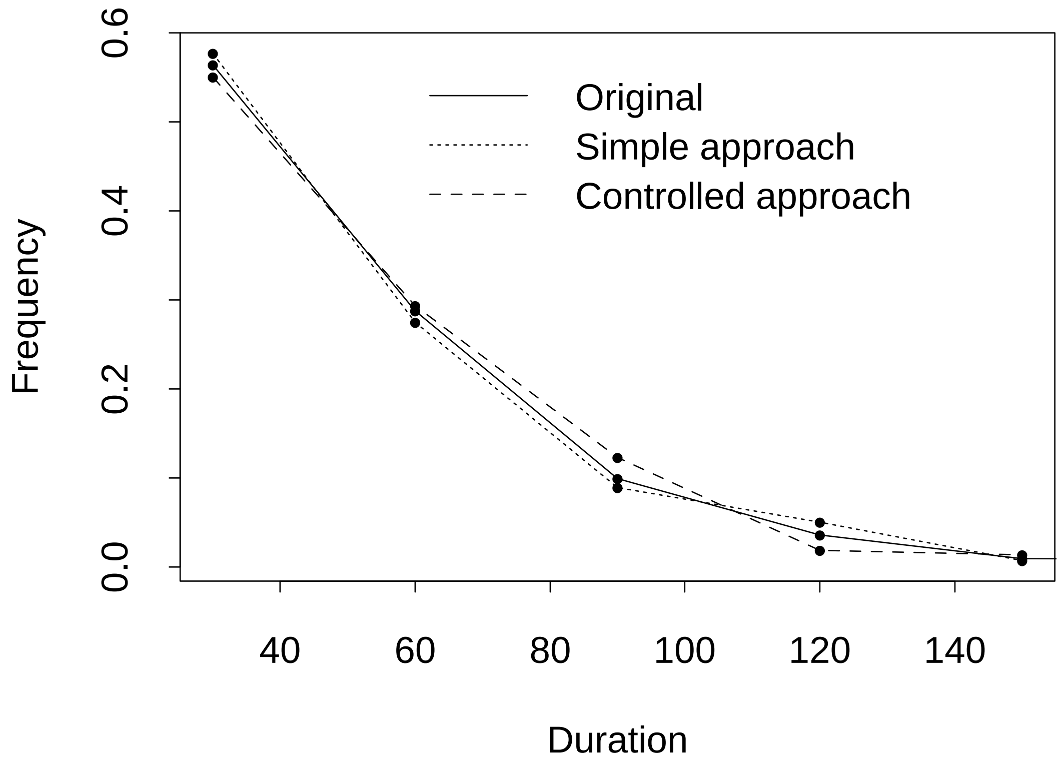


Figure B.3: Distribution of duration in the original data, and the simulations of the simple and controlled models. A quarter note has duration 30

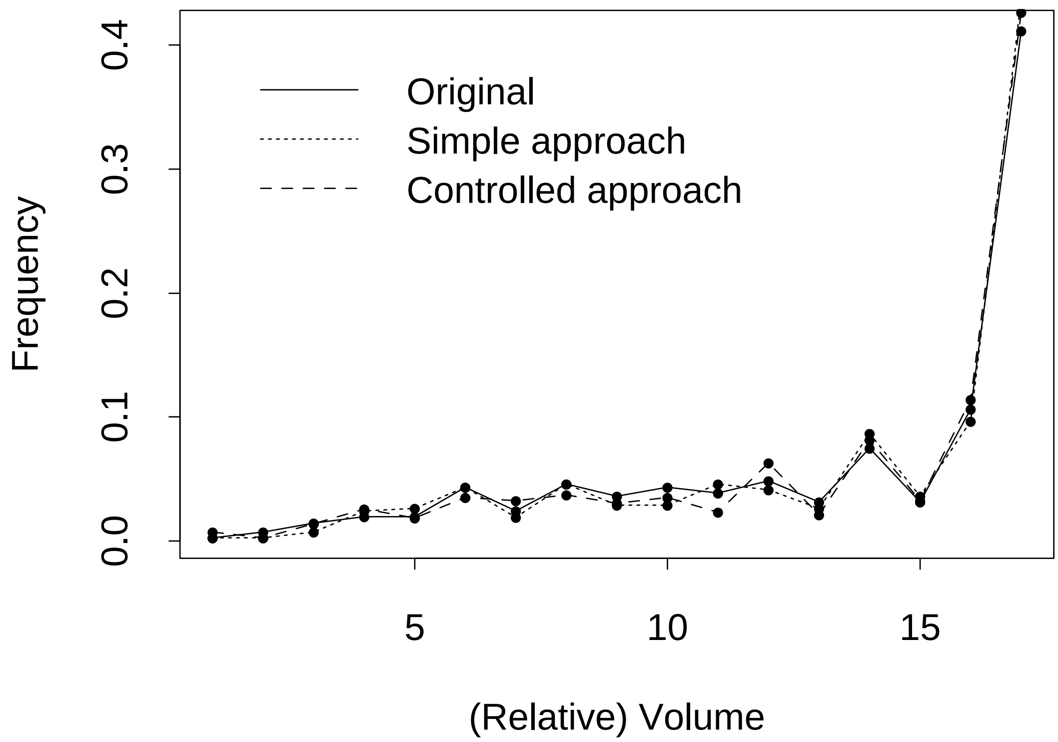


Figure B.4: Distribution of volume in the original data, and the simulations of the simple and controlled models

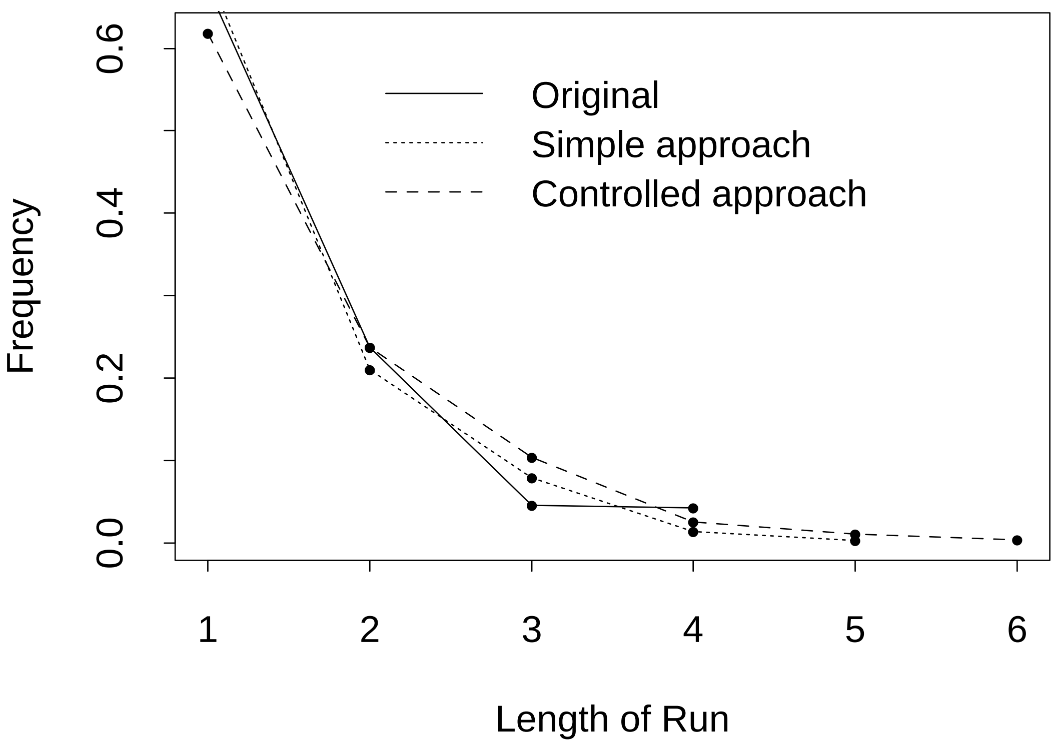


Figure B.5: Distribution of note runs in the original data, and the simulations of the simple and controlled models

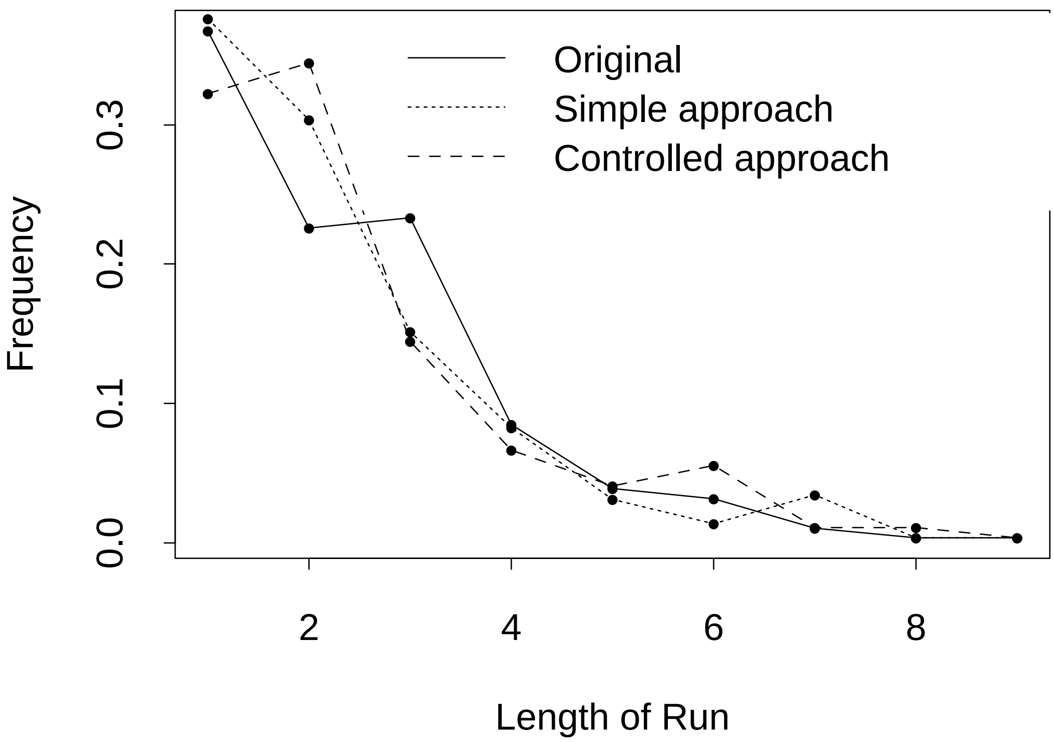


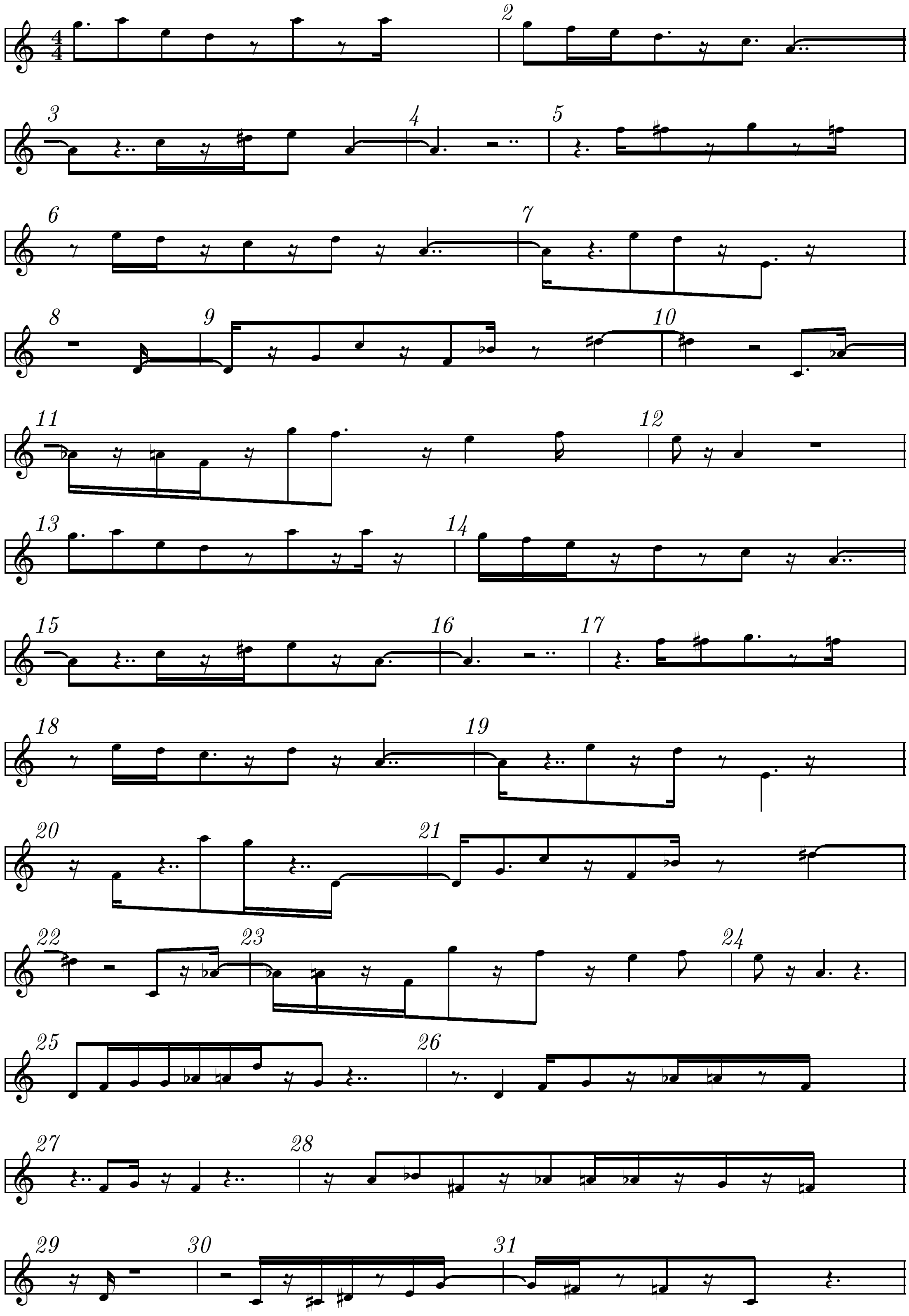
Figure B.6: Distribution of rest runs in the original data, and the simulations of the simple and controlled models

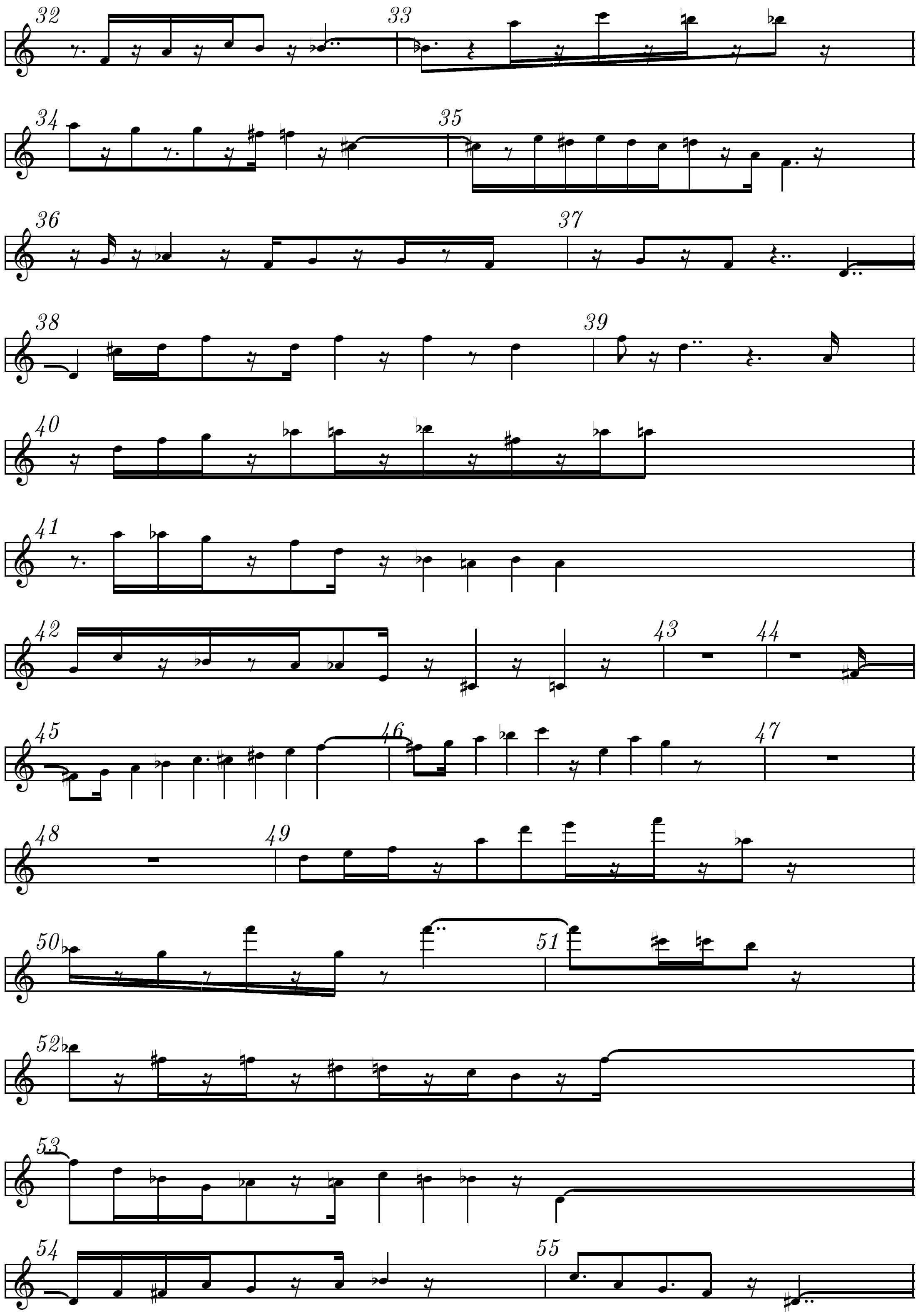
APPENDIX C

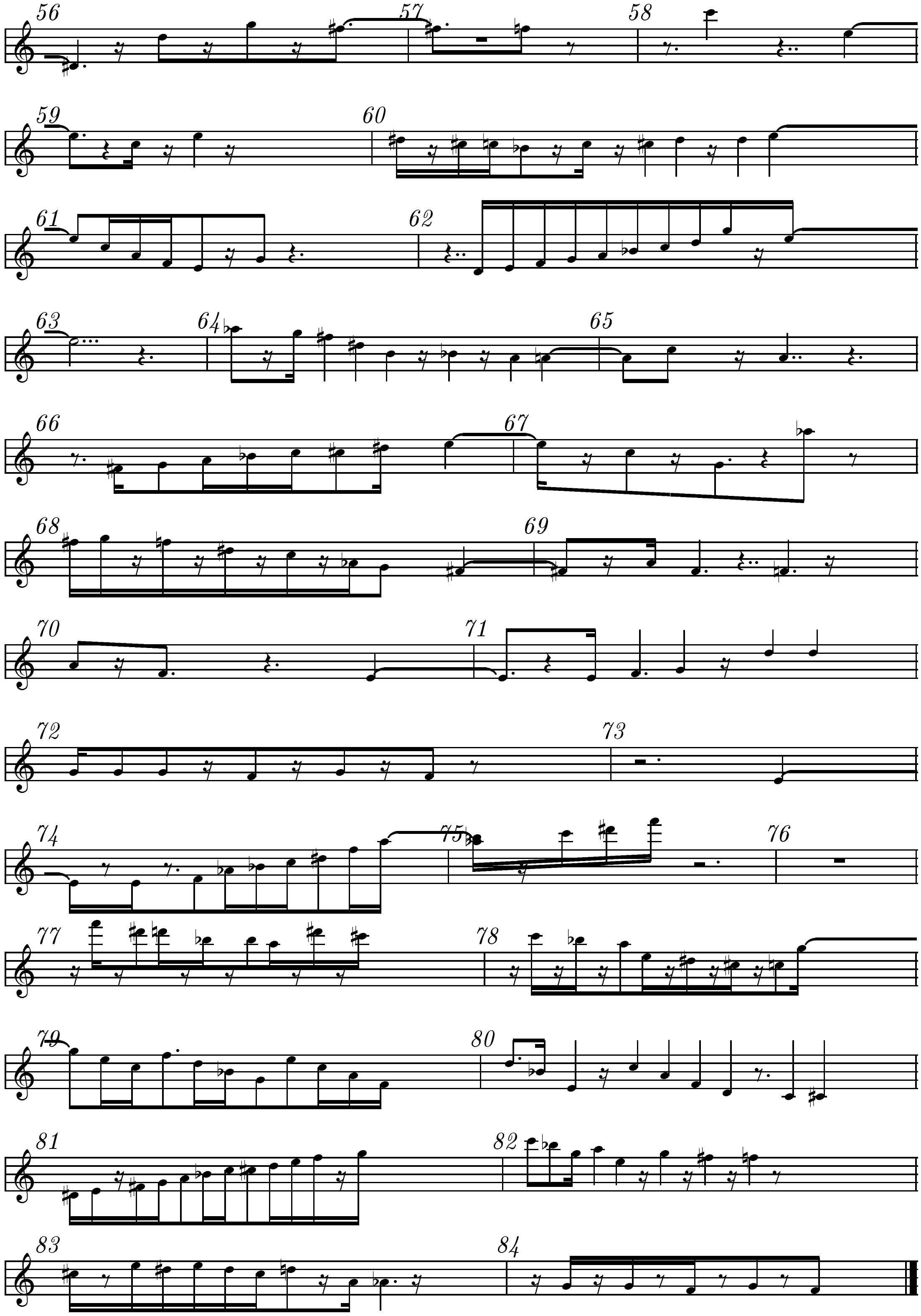
Musical Excerpts

The full musical excerpts of all the examples and simulations used in this dissertation are presented in the following pages.

Israel





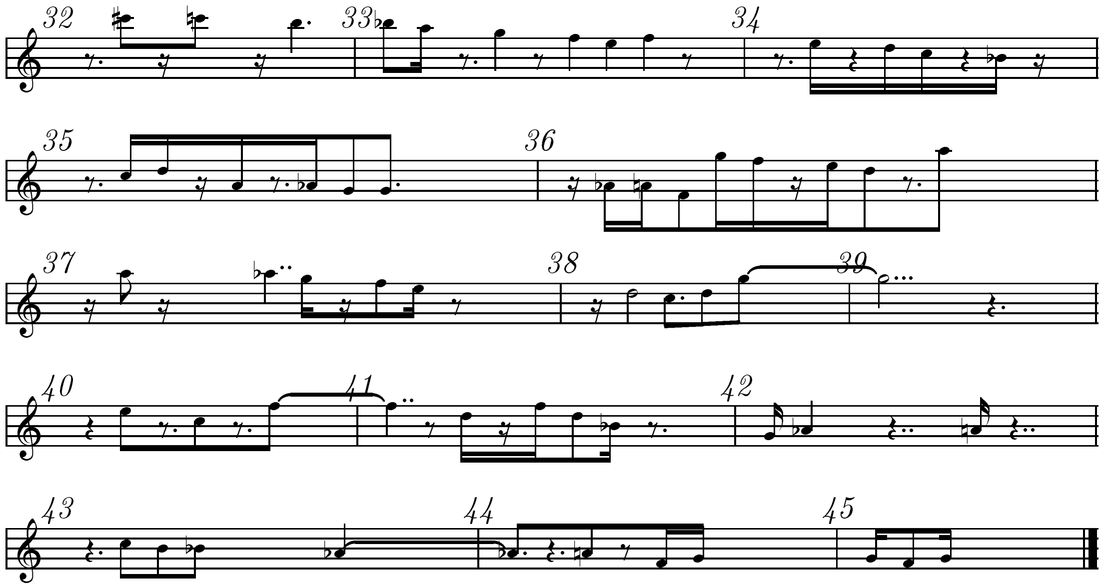


"Naive" Simulation of Israel

by Yuval Marom and S-Plus

"Advanced" Simulation of Israel

by Yuval Marom and S-Plus



Jules 1

by Julian Bolleter


29
30
31
32
33
34
35
36
37
38
39
40
41
42
Slow Drum Pattern
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57


58
59
60
61
62
63
64
Fast Drum Pattern
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86

“Simple” Simulation of Jules 1

by Yuval Marom and S-Plus


Slow Drum Pattern


57
58
59
60
61
62
63
64
Fast Drum Pattern
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85

“Controlled” Simulation of Jules 1

by Yuval Marom and S-Plus


Slow Drum Pattern

Fast Drum Pattern

APPENDIX D

Original Honours Proposal

Title:	Stochastic Techniques for Computer-Music Composition
Author:	Yuval Marom
Supervisors:	Assoc. Prof. C.P. Tsang (Computer Science)
	Prof. Adrian Baddeley (Mathematics)

D.1 Background

Music is a form of human artistic creation. The concept of *creation* an on going issue in Artificial Intelligence and specifically in Computer Music. In particular: can the computer imitate it? Xenakis (1971) suggested that the creative thought of man gives birth to "mental machines", which are merely sets of *constraints* and *choices*. Some of these mechanisms can be expressed in mathematical terms [1, Chapter 1].

Xenakis claims that musical composition offers a collection of sequences that it wishes to be causal, but strict, deterministic causality needs to be replaced by a more general causality, by a probabilistic logic. It was through musical necessity that the laws of calculus of probabilities entered composition.

Choices play a significant role in the process of musical composition and this is where statistics come into the picture. Granted a basic hypothesis, probability theory provides us with choosing tools. Statistics provide quantitative tools for treating an issue previously dealt with solely by qualitative means: the issue of *balance* in a sequence of music [2]. Balance refers to the relative stresses accorded to different musical resources and its formal representation is the statistical distribution [3].

Stochastic processes are processes that develop in time in a manner controlled by probabilistic laws. They deal with the behavior of a collection of random variables and their interdependence [4].

The use of statistics in the field of music composition remains worth of attention. More common techniques that have of been employed in this area are machine learning ones, such as Neural Nets and rule-based systems.

D.2 Aim

The aim of this project is to test whether stochastic processes can imitate the mental processes of a human composer, without considering the psychology associated with such processes. This project can therefore be seen as an experiment, where different stochastic models will be considered and the level of success of each one in generating "good" music will be examined.

Results from this project could be extended to other arts and other activities performed by humans that require creative thought and which we wish to automate. The outcome of this project could have significant implications to Artificial Intelligence and Cognitive Science.

D.3 Methods

The application used in this project will be Jazz. The reasons for this are:

- It is relatively simple. Even the simplest Jazz pieces are very good.
- In composing such music the composer includes:
    1. Some randomness (with regards to pitch, intensity etc')
    2. Some choices (for the same dimensions as in 1 above)
    3. Repetitions
    4. Patterns

  And hopefully we can model all of the above together using stochastic techniques.

The project will consist of three parts, where the third part will depend on the success of the second one:

Initial study of the music

To be able to choose which models are appropriate and how their parameters should be set, some musical research needs to be done. The musical style and its characteristics should be well known. This will be achieved by reading about jazz, listening to some standard pieces of jazz and maybe even talking to musicians.

Modeling & simulating - Analysis by synthesis

The main emphasis of this project will be to choose stochastic models which best describe the different characteristics of jazz. Simulations of these models will then be performed to generate (compose) sequences of jazz.

The most common stochastic models are Markov Chains (MC's) and consequently most of the project time will be devoted to analyzing their different variations. Some examples are: simple MC's, k-dependent MC's, "state incorporation", hidden Markov models, and random walks.

This part will involve two stages:

1. Initially, simple models will be considered to get a feel for the appropriateness of the stochastic techniques in the compositional process. The tasks of observing the behavior of these models and adjusting their parameters as required will be easier when simple models are used.

2. Once the details about the parameters of the models are determined to our satisfaction (ie results lie within the bounds of acceptable music), more complicated models, which might be more realistic, will be considered. This stage will also involve some fine-tuning of the parameters according to the performances of the simulations.

The statistical software used will be *Splus* and either a PC or a Sun running X-Windows will be used to run it.

The output data from the simulations will need to be converted to midi format - a small program will be created which performs this. A midi-interface package, such as *Common Music* and *Cakewalk*, will be used to play the resulting midi file through a Roland U220 sound module.

There needs to be a methodology to test the results of this experiment. The simplest, and probably best, way of doing this is to use a human's opinion — after all, music is for the enjoyment of humans. Objectiveness is very important here, so the author will try to be as objective as possible and will also use a panel of critiques, who will give their opinions as to "how good" the results sound.

Fitting the models to real data

The previous part of the project will be considered a success if adequate models are developed. That is, if the simulations based on these models produce good results. Depending on this success, and if time permits, an attempt will be made to fit the models to real data coming from real jazz pieces.

The collection of such data will be achieved either by:

- Inviting a jazz musician to play some music, or

- Using some benchmark jazz recordings.

The purpose here is to further strengthen the adequacy of the result and see how well they can model real situations.

References

1. Iannis Xenakis. *Formalised Music*. Indiana University Press, Bloomington, Ind, 1971.
2. Charles Ames. A catalog of statistical distributions: Techniques for transforming random, deterministic and chaotic sequences. *Leonardo Music Journal*, 1(1):55–70, 1991.
3. Charles Ames. Statistical and compositional balance. *Perspectives of New Music*, 28(1):80, 1990.
4. Emanuel Parzen. *Stochastic Processes*. Holden-Day, Inc., San Francisco, 1962.