UNIVERSITY OF SOUTHAMPTON

Computer-based Musical Composition using a Probabilistic Algorithmic Method

Gary Chapman

Submitted for Examination for the Degree of Doctor of Philosophy

Department of Music, Faculty of Arts

September 2000

UNIVERSITY OF SOUTHAMPTON

ABSTRACT

FACULTY OF ARTS

MUSIC

Doctor of Philosophy

COMPUTER-BASED MUSICAL COMPOSITION USING A PROBABILISTIC ALGORITHMIC METHOD

by Gary Chapman

The idea of using computers for the composition of music based on mathematical algorithms is not new, and the techniques which have been employed are wide-ranging. However, compositional processes which require an understanding of complex mathematical concepts or of computing techniques tend to be inaccessible to those lacking the necessary skills. In others, the relationship between the supplied input data and the resulting musical output is not evident, and they therefore lack the flexibility to meet specific compositional goals. Systems requiring the specification of large sets of musical rules, or which process pre-supplied music, are more regurgitative than creative.

This thesis describes and investigates a probabilistic, Markov chain-based algorithm whose aims are to be conceptually lucid, to require a small number of input parameters, to be capable of a wide range of musical output and to have the flexibility to meet diverse compositional objectives. A computer program has been developed which provides a composing environment through which the algorithm is analysed in depth, its strengths and weaknesses are examined, and its compositional capabilities are explored.

TABLE OF CONTENTS

CONTENTS OF THE ACCOMPANYING COMPACT DISC

The compact disc which accompanies this thesis may be found in a plastic wallet attached to the inside back cover. It can be read only by an IBM™ PC-compatible computer running Windows™ 95, 98 or NT4. The content structure of the compact disc is as follows:

- 4. The Compositional Process
- 5. Analysis of the Algorithm
  - 5.2 Varying Lambda
  - 5.3 Varying Gradient
  - 5.4 Varying Minimum Mean
  - 5.5 Gradients > 1 or < 1
  - 5.6 Varying Note Length
  - 5.7 Varying Note Velocity
  - 5.8 The Reverse Option
  - 5.9 The Reflect Option
- 6. Style Emulation
  - Bach
  - Dance
  - Gagaku
  - Reich
- 7. Compositional Studies
  - Computer Study for Timpani
  - Markov-2
  - Vibrato Study
- Markov Program
  - Markov User Guide
  - Source Code

The four top level directories entitled "Analysis of the Algorithm", "Compositional Studies", "Style Emulation" and "The Compositional Process" contain files which relate directly to the thesis chapters of the same name. Their precise contents will be described later in the relevant chapters. The directory entitled "Markov Program" contains:

> Appendix A. MARKOV PROGRAM USER GUIDE (in Microsoft Word™ 97 format)
> Appendix D. MARKOV PROGRAM SOURCE CODE LISTINGS (in Microsoft Word™ 97 format)

CONTENTS OF THE ACCOMPANYING FLOPPY DISK

The floppy disk which accompanies this thesis may be found in a plastic wallet attached to the inside back cover. It can be read only by an Apple Macintosh™ computer running System 7. The content structure of the floppy disk is as follows:

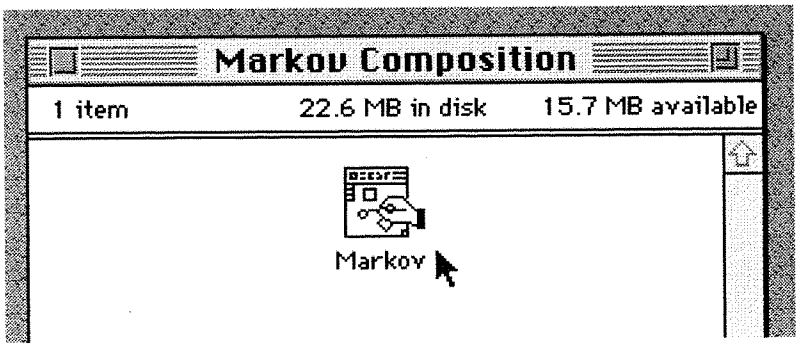
- 4. The Compositional Process
- 5. Analysis of the Algorithm
  - 5.2 Varying Lambda
  - 5.3 Varying Gradient
  - 5.4 Varying Minimum Mean
  - 5.5 Gradients > 1 or < 1
  - 5.6 Varying Note Length
  - 5.7 Varying Note Velocity
  - 5.8 The Reverse Option
  - 5.9 The Reflect Option
- 6. Style Emulation
  - Bach
  - Dance
  - Gagaku
  - Reich
- 7. Compositional Studies
  - Computer Study for Timpani
  - Markov-2
  - Vibrato Study
- Markov Program
  - Markov
  - Markov User Guide

The four directories entitled "Analysis of the Algorithm", "Compositional Studies", "Style Emulation" and "The Compositional Process" contain files which relate directly to the thesis chapters of the same name. Their precise contents will be described later in the relevant chapters. The contents of the directory entitled "Markov Program" are as follows:

> *Markov* executable program
> Appendix A. MARKOV PROGRAM USER GUIDE (in Microsoft Word™ 4.00 format)

MARKOV PROGRAM INSTALLATION INSTRUCTIONS

To install the *Markov* program, simply insert the floppy disk into an Apple Macintosh™ computer and copy the single file, "Markov", to any Macintosh folder, as desired. The *Markov* program is started by double-clicking its application icon :



(in this case the user has placed the icon in a Macintosh folder called "Markov Composition").

The program sends MIDI data to the **modem communications port** on the Apple Macintosh computer. Therefore, in order to be able to hear music generated by the program, a suitable MIDI-compatible sound source (a synthesizer for example) must be connected to the modem communications port.

ACKNOWLEDGEMENTS

I would like to express my sincere gratitude to my supervisors: Ric Graebner, for his unerring help and encouragement during the development, testing and evaluation of the algorithm and computer program, and Nicholas Cook, who proof-read the entire thesis and provided invaluable and illuminating advice regarding its musical and technical content.

My thanks are also due to Ray d'Inverno for verifying the mathematical sections of the thesis.

Chapter 1

Introduction

1.1 THE RESEARCH IN CONTEXT

This research represents a fusion of three distinct, yet interrelated disciplines: Mathematics, Computer Software Engineering and Music.

The ancient Greeks are known to have studied the mathematical principles of sound. Furthermore, the Pythagorean tuning system, developed from around the sixth century B.C., devised a musical scale, based on the proportional relationships between the frequencies of differing pitches, which underlies Western music today.

It is a natural consequence of the awareness of the intrinsic mathematical qualities of musical sound that it should be attempted to build music *from* mathematical systems. The Serial music of the post-war era was based on an inherently simple "music by numbers" process involving the permutation of a twelve-note "tone row". Much of Steve Reich's music of the Sixties and Seventies was similarly "process-oriented", involving phasing techniques and cyclic variations of simple phrases, while Conlon Nancarrow composed canons in which the voices proceed at different speeds, the works being composed directly onto a piano roll for playback on a player piano and achieving complex rhythmic and temporal relationships at speeds beyond the possibilities of human performance.

Such approaches are deterministic in nature but, by taking advantage of the mathematical theory of probability, a degree of randomness and, therefore, unpredictability can be introduced into the resulting music. In the "Musickalisches Würfelspiel", often attributed to Mozart, collections of prescribed phrases are assembled into countless versions of a minuet according to the throw of a dice, whilst similar methods were credited to C.P.E. Bach and Haydn. Stockhausen and Cage have also produced works involving the random ordering of musical sections or of sounds. While such techniques use simple probability to control high-level structures, it is also possible to control probabilistically the musical attributes of the individual notes of a sequence.

However, as the mathematical technique being employed becomes more complex and the structural level at which it works is lowered, so it becomes increasingly impractical to apply it manually due to the amount of computational effort required, and a digital computer becomes a vital tool in the process, with the computer software-encoded computations being performed at very high speed. The computer can be made to introduce probabilistic variation, and furthermore, to do this in a controlled way so that, over and above being simply a labour saving device, the computer is being "creative" on our behalf. Hiller and Isaacson in the late 1950's and later, Xenakis, were pioneers of such techniques.

A branch of probability, Markov theory, enables a degree of relatedness between musical events to be introduced so that their order of occurrence can be probabilistically controlled rather than just their random distribution. Both Hiller and Isaacson's music, and that of Xenakis, employed Markov techniques. However, the vast majority of the work done in the area of Markov-produced music has concentrated primarily on computer-generated variations of pre-supplied music, with any original composition being based on Markov processes with a limited variation in characteristics. The enormous potential for Markov processes to generate explicit and predictable musical event sequences which can be subject to probabilistic variation on a controlled continuum between completely deterministic to completely random, remains largely untapped. This research attempts to correct that omission.

Finally, it is vitally important that musical judgements are applied to the results produced. Ultimately, the question for the composer is "What will be heard when my piece is played?" not "What probability distribution do the pitches satisfy?". The listener will not be directly aware of the latter but the former, through a process of abstraction, is what defines the music's meaning.

1.2 THE AIMS OF THE RESEARCH

1.2.1 General Outline

The main purpose of this research was to develop a mathematical algorithm which may be used to compose musical compositions and which attempts to satisfy a set of stated objectives. This algorithm has been implemented in a computer program facilitating detailed exploration of the algorithm in order to examine how well it satisfies the objectives, to elicit its strengths and weaknesses and to compose new music. Four principal areas of work were involved:-

1. The development of an algorithmic composition process based on the mathematical stochastic process theory of Markov chains.

2. The development of a computer program, *Markov*, which embodies the algorithmic process and provides an appropriate compositional environment via a user-friendly interface.

3. The composition of music using the Markov program, both in 'free' and in stylistically constrained contexts.

4. The analysis of the results.

1.2.2 The Objectives

This section states the objectives against which the algorithm was developed, and discusses the musical reasons why these objectives are considered important (later in this thesis, in Chapter 2, **THE ALGORITHM**, I will revisit them in order to evaluate how well they are satisfied by the algorithm).
The objectives are as follows:-

1) The composer should need no knowledge or understanding of the mathematical techniques employed by the algorithm in order to be able to make effective use of it.

If knowledge of the underlying mathematics is required then the number of composers who can make use of the algorithm is limited to a specialised group: mathematicians with an interest in composing music, and composers with a knowledge of mathematics. The composer should just need to vary the input parameters to the algorithm. What is absolutely vital, however, is that the composer has a clear perception of the ways in which varying the parameters affects the musical output. This may be achieved in any of the following ways:

(i) The algorithm is specified in such a way that the effects of parameter variation are readily apparent.
(ii) The results of a thorough perturbation analysis of the parameters are available in the form of a set of compositional rules or principles relating to parameter values.
(iii) The composer is able to carry out his or her own experimentation with parameter variation and draw reliable conclusions.

In this thesis, Chapter 2, **THE ALGORITHM**, explains the general principles regarding the ways in which the algorithm parameter values control the output, Chapter 4, **ANALYSIS OF THE ALGORITHM**, presents the results of a detailed analysis of the effects of varying the parameters, Chapter 5, **THE COMPOSITIONAL PROCESS**, presents an example of a composer manipulating the algorithm in order to produce a composition and Chapter 6, **STYLE EMULATION**, reveals further compositional principles with regard to achieving specific stylistic objectives.

All this implies that there is a **predictability** about the musical results that will be obtained from the parameter values input. Some algorithms are more **experimental** in that one is not really sure what the results will be in advance and the compositional process is more of a, potentially, exciting voyage of discovery. In fact, the algorithm explored in this thesis permits this approach as well: rather than deciding in advance what parameter values might be appropriate to achieve a particular compositional objective, the composer is also free to assign parameter values more or less at random and see what happens.

2) The composer should be required to specify only a small number of input parameters to the algorithm but a very wide range of musical results should nevertheless be achievable by varying the values of these input parameters.

If there are a large number of input parameters then the algorithm can become unwieldy to use for compositional purposes, not only because it takes time to set up initially, but also because it is more difficult to decide how to modify the parameters subsequently in response to the initial musical output in order to achieve desired changes. By the same token, it is much more difficult to predict the output because there are so many variables to consider. It was an aim of this research that the composer should be involved in a feedback process whereby, having heard the musical results, the parameters can quickly be modified repeatedly so as to converge on a desired result.

On the other hand, limiting the number of input parameters must necessarily result in a sacrificing of flexibility, in terms of the variety of output that can be achieved. It was one of the aims of this research to see just how few parameters one can "get away with" and the results are surprising. Chapter 2, **THE ALGORITHM**, details some strict limitations of the algorithm, Chapter 4, **ANALYSIS OF THE ALGORITHM**, explores the wide range of variation in musical structure which can be achieved and Chapter 6, **STYLE EMULATION**, shows some examples of the wide diversity of musical styles which can be obtained whilst making clear the limitations imposed and any compositional compromises which must be made.

3) The algorithm should be the absolute starting and finishing point in the compositional process.

The composer is neither required to supply any explicit sequences of notes, nor is there any prior mathematical analysis of a supplied piece of music. Some

systems take presupplied music and subject it to transformations, for example, Jacob's "Variations" (Jacob 1996). Other, more mathematically-oriented systems are discussed later in this chapter.

Again, some composers view algorithmic techniques as an aid to composition rather than as a complete solution (McAlpine et al 1999: 19). The intention here is that the use of the algorithm should be "pure", so that it can be assessed solely in terms of its intrinsic ability to produce music, whatever compromises or limitations that might imply.

It should be noted that this is not intended to be **interactive** composition where the composer and computer react to each other in real time. This is a rich and widely studied area of computer-assisted composition (Chadabe 1977, 1983, Chadabe and Myers 1978, Risset 1990, Rowe 1993) which is not under investigation here. Instead, the composer using the Markov program reflects qualitatively on a section of work before deciding what modifications and refinements are required.

4) The music produced by the algorithm should have a sense of evolution.

What is meant here is that the musical events that have occurred will have some degree of **influence** on the musical events that are about to occur. This allows, in particular, for a sense of linear direction in resultant note sequences, typical of Western music.

5) The composer should be able to control the degree, or strength, of this influence.

This means, very importantly, that repeated playings of a piece based on a fixed set of mathematical input parameters **may be different.** Brian Eno, discussing his Koan program, which generates infinitely changing music based on rule sets inputted by the user, said "I really think it's possible that our grandchildren will look at us in wonder and say 'You mean you used to listen to exactly the same thing over and over again?'" (McClellan 1996). The stronger the degree of influence, the more deterministic is the control over the way the piece evolves; the weaker the degree of influence, the more unpredictable the evolution. To put it another way, any of the following situations could apply:-

(i) if the musical event A occurs it is always followed by the musical event B.

(ii) event A is usually followed by event B but may occasionally be followed by events C, D or E.
(iii) event A could be followed by any of a whole set of possible events
(iv) the musical events occur in a totally unpredictable way.

Myhill (1979) discusses the concept of a continuum between random and deterministic music. At one end is complete randomness, but, given the diversity of musical styles and cultures, it is difficult to define what is at the other end. In our case, what lies at the deterministic end is the range of deterministic possibilities afforded by the algorithm, and we may visualise the continuum as a plane through which we may journey towards or away from randomness along an effectively endless multitude of possible paths. Chapter 4, **ANALYSIS OF THE ALGORITHM**, explores the deterministic possibilities in detail, and analyses the results of controlling the degrees of randomness.

6) The composer should not be required to specify musical rules.

One approach to algorithmic composition is to specify a (possibly very large) set of rules, disallowing parallel fifths for example, or requiring each 8 bar phrase to end with a perfect cadence, and then programming the computer to produce music which obeys these rules. However, with this approach one is tending to use the computer as a labour saving device rather than as an instrument of creativity, generating music which meets specific structural goals but which could have been written manually. Of course, it is possible, perhaps inevitable, that the user of the algorithm examined in this thesis will come to it with pre-conceptions of the kinds of musical results which they would consider desirable, but it is then down to the algorithm, or rather the user's control of it, to produce these results.

7) The composer is not required to have any knowledge of computer languages or computer-related logic.

Many computer music languages exist ranging from the various incarnations of Max Mathew's MUSIC language (Dodge and Jerse 1985: 12-15) with which the composer essentially specifies note lists in a numerical format, through to Lisp or C related languages, some of which include facilities for incorporating mathematically generated music, such as Richard Orton's Tabula Vigilans[1] or

[1] The Tabula Vigilans Information Page, contained in the Composers' Desktop Project Website, can be viewed at: **http://www.bath.ac.uk/~masjpf/CDP/tvinfopg.htm**

Tonality Systems' Symbolic Composer for example, and object-oriented languages (Jaffe and Boynton 1989, Pope 1996). However, these languages require a good knowledge of computer programming. Graphical interfaces to music programming languages do exist to make life easier for the composer, such as the powerful and extensively used Max[2] (Rowe 1993: 32-38) among others (Assayag et al 1999), but these nevertheless tend to be structured from a computer logic viewpoint.

Such languages do of course offer, in return for the work involved in constructing compositions with them, enormous flexibility. However, an aim of this research is that, whilst there has been a considerable amount of programming involved in the implementation of the algorithm, it should, from the composer's perspective, be reduced to a set of numerical input values.

It is important, at this point, to make clear a non-objective of this research. The computer program, *Markov*, which I have developed to implement the algorithm, provides a compositional environment in which structured compositions can be created, together with a user-friendly graphical interface to facilitate easy entry and manipulation of parameter values. It is **not**, however, intended to compete with commercially available sequencing and algorithmic composition packages, and advanced graphic and other features which one would expect to find in such packages are not implemented here. To have done so would simply have diverted time and effort away from the main task: the exploration of the algorithm.

1.3 ALGORITHMIC COMPOSITION REVIEW

Simply stated, when composing music with mathematical algorithms, a formulaic system is applied which will generate sequences of numbers. These numbers are then mapped uniquely to values of musical parameters so that the number sequences produced are transformed into note sequences. In its basic form, the process of using a mathematical algorithm for the computer-generated composition of music can be considered to consist of the following stages:

1. The specification of the algorithm in terms of its mathematical formulae and any associated rules.
2. The definition of the process for mapping the numerical output to musical parameter values.

---

[2] An overview of the Max program can be found at the Opcode Systems website, at **http://www.opcode.com/products/max**

3. The coding of the algorithm in a computer program, together with an appropriate interface to allow the composer to specify any input data required by the algorithm.
4. The running of the program to produce musical output.

Stages 1 to 3 refer to the design stage of the process while stage 4 refers to the functional use of the algorithm and will be repeated many times. Chadabe (1983: 22) calls this a "design-then-do" procedure for composing.

To clarify the concepts involved, we begin with a straightforward example of such an approach to composition. This approach, simple in concept and design, is to use a function which varies with time, of the form

$$y = f(t)$$

and map the values of y to a musical parameter, pitch for example, so as to produce music in a time-sequential manner.

This is the principle underlying the UPIC program (Lohner 1986, Xenakis 1996: 150-152), which allows the composer to create a score consisting of a collection of "arcs" using a board on which lines are drawn with a special ball-point pen; each arc describes a pitch-versus-time curve. Thus, for example, a horizontal straight line produces a sound of constant pitch whose duration is determined by the length of the line, while a curve results in a sound whose pitch changes in a continuous manner. Polyphony is achieved by drawing two or more lines which overlap in time. A later version of the program (Xenakis 1992: 329-334) provides a sophisticated mouse-driven interface.

We now proceed with a review of the field of algorithmic composition. This review is not intended to be exhaustive but discusses some of the principal areas of work in algorithmic composition over the last forty years, making reference to the objectives stated above, in an attempt to place the research presented in this thesis in context.

I have classified the algorithmic methods examined into the following categories: fractals and chaos theory, cellular automata, artificial intelligence and stochastic techniques.

1.3.1 Fractals and Chaos Theory

The key property of a fractal curve is that of **self-similarity**. A self-similar structure is one whose parts recursively repeat the whole structure, no matter to which level of detail it is examined, so that the overall characteristics observed at a larger scale are reflected in similar characteristics on a smaller

scale. Fractal shapes have been found to be characteristic of many natural phenomena, for example, the geometry of turbulence in fluids (Mandelbrot 1977: 97-105) and the shapes of coastlines and the relief of the earth's surface (ibid: 256-271).

Chaotic systems, closely related mathematically to fractal geometry, can produce highly structured sequences which have everywhere within them elements of near repetition.

1.3.1.1 Fractional Noises

By treating a sequence of numbers as a waveform and taking its Fourier transform, we obtain the spectrum of the sequence (Moore 1990: 412-413). The spectrum is essentially a graph which breaks the waveform down into its constituent frequencies, the sum of which produces the original waveform. If the spectrum is flat, then the frequencies are evenly distributed and the sequence is purely random. This is termed "white noise". If the shape of the spectrum is proportional to the inverse of the frequency (termed "1/f noise") then the sequence exhibits self-similarity of a fractal nature. Voss and Clarke (1978) analysed the pitch and loudness of examples of many different styles of music, including Bach, Beethoven and the Beatles, and found the spectral shapes of all them to be close to 1/f. 1/f sequences can be generated programmatically (Jones 1984: 84-91, Dodge and Jerse 1985: 290-291). Moore (1990: 442-453) gives examples of waveforms, spectra, and resulting melodies for pitch sequences with spectral curves of $1/f^{ß}$ for various values of ß. The higher the value of ß, the more constrained is the resulting melodic sequence.

Dodge (1988) produced a composition, *Profile*, using 1/f noise. A melodic line was first generated using a 1/f sequence mapped onto a chromatic pitch collection. Then, for each note in the first line, a succession of notes in a second line was generated, using the same 1/f approach. A third line was generated in the same way, in relation to each of the notes in the second line. Bolognesi (1983: 28-31) describes a technique for generating 1/f melodies which makes the hierarchical self-similar structure appear more clearly.

A fractional noise music generating system is relatively simple to set up. However, the lack of controlled predictability makes this approach unsuitable for our purposes.

1.3.1.2 Non-Linear Dynamical Systems

A **Non-linear dynamical system** is a set of one or more equations that are iterated. That is to say that the values obtained from applying the equations are fed back repeatedly into the equations to produce a sequence of values. These values are then mapped to musical parameter values. A simple one-dimensional system has the following form:

$$x_{n+1} = f(x_n)$$

The system is given a starting value, x_0, which is fed into the equation to produce x_1, the second value in the sequence. This is then fed into the equation to produce x_2 and so on.

The sequence of values produced is called the **orbit** of the system and, depending on the values assigned to the parameters of the equations making up the system, can exhibit a variety of different behaviours. For example, it may tend to zero or infinity, or it may settle into a repeating pattern. Of particular interest musically, however, are chaotic orbits, in which the sequence shows unpredictability but also traces of cyclic behaviour, where previously heard patterns will appear to recur but with degrees of variation.

Pressing (1988), for example, explores the equation

$$x_{n+1} = ax_n(1 - x_n)$$

examining the differing output for various values of the parameter a, and also gives examples of higher dimensional systems (that is, consisting of two or more equations) in which each dimension is mapped to a different musical parameter. Such systems allow coordination to be achieved between different parameters. Bidlack (1992), examines a range of chaotic systems, giving pictorial representations of the scores produced by each. Gogins (1991), on the other hand, whilst also making use of dynamical systems, does not apply the values of the orbit directly, but instead derives the **measure** of the set of points making up the orbit. The measure can be interpreted as the **density** of the points in it. The density of a point depends on the number of times it is visited, or "hit", during the generation of the orbit: the more times it is hit, the greater is its density. The resulting score is represented graphically as a two dimensional image with time running left to right on the horizontal axis, pitch running from bottom to top on the vertical axis and the depth of colour of each point varying according to its density. The density is mapped to loudness. Once the score is complete it may be elaborated by applying various transformations, for example, rotation, scaling or translation.

Non-linear dynamical systems have the advantage of requiring only a small number of input parameters whilst being capable of a wide range of output. However, the drawback from our point of view is, again, the lack of control of the output. The chaotic behaviour of the output is extremely difficult to predict in advance, while very small changes in the values of the parameters can result in extremely large variation in output, and the compositional situation is therefore one of experimentation, where the results can nevertheless be surprising and fascinating. An additional disadvantage of Gogins' approach is that it can take an inordinately large number of iterations, possibly millions, for the score to be formed.

1.3.2 Cellular Automata

Cellular automata have been used to model naturally occurring phenomena in a wide range of disciplines including physics, biology and chemistry. A cellular automaton consists of a rectangular array of cells each of which contains a discrete variable quantity. The state of a cellular automaton is determined by the values of all of its cells and the state evolves in step with the ticking of an imaginary clock. A set of rules is applied to determine the value of a cell based on the values of its neighbouring cells. To start the automata, an initial configuration of cells must be specified.

Miranda (1993, 1994) applies two cellular automata in parallel. In the first, called "The Game of Life", each cell can have one of two values, alive (1) or dead (0). It has the following rules:

1. if a cell is dead at time t, it becomes alive at time t+1 if, and only if, exactly 3 of its 8 neighbours are alive at time t
2. if a cell is alive at time t, it becomes dead at time t+1 if, and only if, fewer than 2 or more than 3 neighbours are alive at time t

In the second, called "Demon Cyclic Space", each cell can be in one of n possible states, numbered 0 to n-1. A cell that is in state k at time t dominates any neighbouring cells that are in state k-1, in that their state changes to k at time t+1. A cell that is in state 0 dominates neighbouring cells that are in state n-1.

The Game of Life cellular automaton is used to generate a sequence of trichords. The first note of each trichord is taken from a predefined pitch sequence supplied by the user. The second and third notes are derived by mapping the Game of Life array to a two-dimensional Cartesian coordinate system called a "Neumann Musical Space". Each live cell produces a trichord, the x coordinate of the cell determining the interval of the second note of the

trichord above the first, and the y coordinate determining the interval of the third note above the second. The notes of the trichord do not necessarily occur simultaneously. Instead, the ordering of the three notes is derived algorithmically from the states of the neighbours of the corresponding cell while the precise triggering points and durations are calculated from a user-selected distribution formula[3]. The Demon Cyclic Space cellular automaton, on the other hand, is used to determine the MIDI channel associated with the output associated with a cell. These techniques are implemented in a computer program called CAMUS (**C**ellular **A**utomata **Mus**ic). This system was later developed into CAMUS 3D (McAlpine et al 1999), which uses three-dimensional cellular automata, in order to produce four-note chords, and Markov chains[4] to calculate the note durations.

Again, cellular automata do not satisfy our objectives due to the lack of direct control of the output in terms of input parameter variation. The principal "parameter" is the initial setup of the cell configuration pattern and it is extremely difficult to predict where a particular pattern will lead. Miranda (1993: 14) describes composing with the CAMUS program as being "like the nature of an experimental action: an action the outcome of which is not foreseen", and that, of course, is the appeal of composing in this way.

1.3.3 Artificial Intelligence

An artificially intelligent composition system is one which is able to "learn" the structure of existing musical examples and generalise from them to compose new pieces.

Cope has developed a system called *EMI*, which stands for "Experiments in Musical Intelligence" (Cope 1987, 1991). Using techniques taken from linguistic models, Cope's Schenkerian derived *SPEAC* (Statement, Preparation, Extension, Antecedent, Consequent) system provides a symbolic mechanism for describing hierarchical musical structures and relationships. A **signature** is defined as a set of contiguous interval patterns and Cope's premise is that these signatures represent the essential means by which we recognise the style of a particular composer. The data for two or more works by the same composer are entered and these are then parsed using a pattern matching algorithm to discover the signatures, which are added to a signature **pool.** Durations are pattern matched as well as pitch sequences. The process can now be reversed to generate new material. Rather than make random selections, *ATNs* (Augmented Transition Networks) are employed to make informed, probability-based choices by reviewing previous decisions and selecting an

---

[3] Distribution formulas are discussed in detail in **1.2.4.2** below

[4] Markov chains are discussed in detail in **1.2.4.4** below

option that is appropriate for its predecessors. Cope has used his EMI system to generate musical examples in a range of different styles (Cope 1991: 141-212) including Bach Inventions (ibid: 141-151), a Mozart Sonata (ibid: 154-171) and a Joplin Rag (ibid: 171-173).

Todd (1989) describes a network that can learn aspects of musical structure. The network learns to produce the next note in a sequence based on some memory of past notes. This memory is provided by feedback connections that cycle current network activity back into the network for later use.

Genetic techniques have also been applied to produce variations of music input by the user (Ralley 1995, Burton and Vladimirova 1999). An initial population is seeded with pieces similar to the user's, formed as a result of analysing the user's music. New output is then obtained through a process of mutation and recombination of the members of the population.

Methods which produce variations of music supplied by the user have been criticised for lacking in originality and creativity (Laske 1990). However, as well as providing a supply of new material in a particular style, they can give important insight into the regularities of a style (Loy 1990) and, of course, the supplied music could be in the user's own style. Nonetheless, these techniques do not comply with our third objective, that the algorithm should be the starting point in the composing process, and are therefore not considered further in this thesis.

1.3.4 Stochastic Techniques

A **stochastic process** is a collection of random-variable quantities distributed in space or time (Jones 1981: 45). There are degrees of randomness, however, ranging from complete randomness, where there is no order, to complete determinism. In order to compose music from a stochastic process, a structural framework, or **stochastic generative scheme** (ibid: 45), must be established within which the random behaviour, and thereby the musical output, of the system can be controlled.

1.3.4.1 Rule Based Approach

One way to create a structural framework is to specify a set of rules which the musical output must satisfy, sometimes called a "Random Sieve" (Moore 1990: 413-418). This is the approach adopted by Hiller and Isaacson in the first three movements (or "Experiments") of their ground-breaking work *Illiac Suite for String Quartet* (Hiller and Isaacson 1959). They described the composing of music as the "extraction of order out of a chaotic multitude of available

possibilities" (ibid: 1) and looked to mathematics to aid this task. Their composing process consists of two basic operations:

1. Generate random sequences of integers which are equated to notes of a scale, rhythmic patterns, dynamics, or playing instructions.
2. Screen each integer through a series of arithmetic tests expressing rules of composition. If the integer fails the tests it is rejected, if it passes it is stored until the composition is ready to print.

These operations were coded in a computer program and the printed values were then manually transcribed into standard music notation for live performance.

The rules consisted of **Melodic Rules** (for example, no melodic line may span more than one octave), **Harmonic Rules** (for example, the first and last chords of a melodic line must be based on the tonic triad) and **Combined Rules** (for example, parallel perfect fifths are forbidden).

This is a classic example of the rule-based approach discussed in objective 6. The mathematical mechanisms of the algorithm do not really contribute anything to the nature of the musical output. Instead, the composers have essentially decided everything in advance and are using the computer as a labour-saving device. Also, the system does not lend itself to easy adjustment and refinement of the rules. That is not to detract, of course, from the immeasurable importance of Hiller and Isaacson's pioneering work, which was probably the first serious attempt at using a computer to compose a work of music and has inspired much of the computer-based composition work that has followed (in any case, as we shall see later, the fourth movement of the Illiac Suite does employ mathematical generation techniques). However, this approach does not accord with our stated objectives.

1.3.4.2 Probability Distributions

A more basic stochastic structure consists of specifying a simple probability distribution over a musical event space (Jones 1981: 46). If the probabilities of all events are equal, then they will occur completely at random. Any distribution of probabilities is possible, however, so certain events may be made to be more likely to occur than others (Moore 1990: 418-429, McAlpine et al 1999: 20-22). Lorrain (1980) describes a wide range of probability distributions suitable for music composition and gives procedures for coding them in a computer program. Similarly, Dodge and Jerse (1985: 278-283) provide musical examples produced from a number of different distributions. Xenakis (1992: 131-154) calls

this approach "Free Stochastic Music by Computer" and developed a computer program called SMP, or Stochastic Music Program (Dodge and Jerse 1985: 295-297), in which various musical parameters are chosen from probability distributions or random lookup tables. Again, Koenig's programs Project 1 and Project 2 (Koenig 1970a, 1970b) produce compositions consisting of a number of sections within each of which musical events are generated randomly. By introducing a "repetition check" (Koenig 1970a: 36), whereby a pitch is not allowed to repeat within a particular note sequence, Koenig achieves a technique which he considers to be a more general case of serialism (ibid: 33). The character of the music produced in the various sections is varied by controlling the degree to which repetition checking is applied.

A number of commercial software packages incorporate probability distribution techniques. The Sound Globs program (Scholz 1989, Rothstein 1990), for example, provides a professional graphical user interface with which the composer draws probability distribution curves between fixed ranges of note pitch, loudness, vertical density (polyphony), horizontal density (number of pulses between successive notes) and note duration, while the M computer program (Zicarelli 1987: 19-23), published by Joel Chadabe's Intelligent Music Company, takes a note sequence entered by the user and subjects it to a variety of manipulations, including probabilistic variation.

Finally, Greenhough (1984) describes a system whereby the user specifies a priori probabilities of occurrence for each of the 12 pitches of the equally tempered scale, this level of control being referred to as the p-level. However, in addition, probabilities are specified for the size of the interval between successive pitches (the Δp-level), providing control of the stepwise melodic motion, and for the change of interval (the Δ^2p-level), which effects control of the curvature of the melody. In this way, a wide range of melodic shapes can be obtained.

The basic probability distribution approach satisfies many of our objectives. Conceptually simple, it requires the specification of a relatively small set of parameter values for each musical characteristic. In the case of a mathematically defined distribution, this is just the range of values together with the parameters that specify the distribution. In the case of a probability lookup table, the situation is less straightforward since a number of probability values must be specified manually, but the user interface can help. In addition, some structural parameters may also be required, depending on the compositional framework defined by the computer program in which the algorithm is implemented. The drawback, however, is that while short sequences may give the illusion of a predetermined quality, there will be no evident long-term pattern, and so, in particular, this does not satisfy objective 4, that there should be a sense of evolution. Greenhough's system is an exception

to this since it affords more sophisticated control of melodic structure. However, the amount of data required to specify the p, Δp and $\Delta^2 p$ levels contravenes objective 2, sacrificing predictability and ease of refinement.

1.3.4.3 Stochastic Grammars

A **stochastic grammar** (Jones 1981: 51-60, 1989: 185-195) uses structures derived from linguistics. A formal grammar consists of a set of symbols, an event space, a set of production rules specifying ways in which symbols may be rewritten by combinations of symbols and events, and a starting symbol to begin the generative process. Formal grammars provide a powerful means for representing musical structures (Roads 1979). A stochastic grammar associates probabilities with the rules of generation. As an example, a simple one-dimensional "space grammar" consists of two rules, the first of which causes a splitting in half of the one-dimensional space (interpreted as a time line), while the other terminates the splitting process, resulting in a musical event. The splitting process is applied in a recursive hierarchical fashion to all the subdivisions created by the first rule, causing a further subdivision with probability p_1 or a termination of the splitting process with probability p_2. When all splitting has terminated the generation is complete (note that the generation proceeds in a top-down fashion rather than in time-sequential order). The character of the resulting music is altered by adjusting the probabilities. For example, if the one-dimensional space grammar is applied to the rhythmic structure, increasing p_1 will cause splitting to continue to a greater depth, resulting in shorter note durations and a faster rhythm. Increasing p2 causes the splitting process to terminate earlier so that notes will tend to be longer. This procedure can be extended to further dimensions. For example, with a two-dimensional space grammar, one axis can be associated with time, as above, while the other can be associated with pitch. An additional rule is required to produce splitting in the pitch dimension. Increasing the probability associated with this rule will favour simultaneous note activity. Further dimensions may be added to control intensity and timbre.

Stochastic grammars satisfy many of our stated objectives. The set of input parameters is fairly small and the output is adjusted simply by varying the probabilities assigned to the rules, with a wide variation in musical characteristics being obtainable. The examples given above are simple ones and greater variety and complexity can be achieved from quite modest extensions to the rule sets. However, the top-down nature of the generation process means that the resulting music does not possess an evolutionary quality, in contravention of objective 4.

1.3.4.4 Markov Chains

The Markov chain (Feller 1964: i.338-395, Freedman 1983), formulated in 1906 by the mathematician Andrei Andreevich Markov to distil tendencies in spelling in written Russian, embodies the concept of **conditional probability**. That is to say, the event which is next to occur depends in a probabilistic way on one or more past events.

A given Markov chain is defined by its **transition matrix** which specifies the probabilities controlling the evolution of events. Consider the following example of a transition matrix:

		NEXT PITCH		
		C	D	E
PREVIOUS	C	0	0.7	0.3
PITCH	D	0.1	0.5	0.4
	E	1.0	0	0

This is a matrix for producing a simple three note melody consisting of the pitches C, D and E. The figures specify the probability of the pitch of the next note given the pitch of the previous note. For example, if a C occurs, there is a 0.7 (or 70%) chance that the next note will be a D and a 0.3 (or 30%) chance that the next note will be an E - two consecutive C's never occur because the associated transition probability is 0. If an E occurs, the next note is always C since this transition probability is 1. To start the sequence off we must provide a starting note, which can be either chosen or generated randomly. We can separately control various musical parameters (pitch, note length, dynamic and so forth) in this way. The transition probabilities may be user-specified or they may be derived by analysing the transition frequencies exhibited by an existing piece of music (Dodge and Jerse 1985: 285-288, Moore 1990: 430-439).

The above example is of a **first-order** Markov chain, because the next event depends only on the preceding event. However, higher order chains are possible. For example, in a second-order Markov chain the next event depends on the **two** preceding events, in a third-order chain on the **three** preceding events, and so on. In addition, the event space may consist of single musical parameter values, as in the above example, or longer, pre-composed, musical fragments (Jones 1981: 48).

Markov chains have been applied by different composers in a variety of ways. Xenakis (1960: 86 et seq, 1992: 43-52), for example, views sound as an

integration of sonic grains, each having a duration, frequency and intensity. By dividing time into a succession of small time slices we obtain an evolutionary sequence of "screens". A matrix of transition probabilities can now be formed to describe the transition from one screen to the next (Xenakis 1992: 69-109). Here, the screens form the states of the Markov chain.

As mentioned earlier, the transition matrix may be obtained from an analysis of existing music. Hiller and Baker's 1963 work *Computer Cantata* (Hiller and Baker 1964), for example, makes extensive use of transition tables derived from an analysis of an excerpt from the second movement of Charles Ives' *Three Places in New England.* The Jam Factory computer program (Zicarelli 1987: 23-27), on the other hand, builds transition tables from music entered by the user.

The transition matrix does not have to remain fixed throughout the course of the work, however. The probability values may, for example, be modified continually. In the fourth movement of Hiller and Isaacson's *Illiac Suite* (Hiller and Isaacson 1959: 131-147), the four part musical structure is seen as a random flight of four trajectories, characterised in terms of Markov chain processes, and the transition probabilities are regularly adjusted automatically, causing a shift from a fixed tonality to a freer, more random, texture. Similarly, Ames (1989: 183-184) describes a work developed in collaboration with John Myhill which employs transition probabilities that evolve gradually under strict parametric control. Another approach is to choose one from a number of alternative transition matrices at each stage in the event generation process. In Zicarelli's aforementioned Jam Factory program, for example, transition matrices are held for each of orders one to four and an automatic choice is made, at each note, as to which of the four tables to use based on a user-controlled probability distribution. Zicarelli's experience suggests that "70-80% order two with the rest divided between orders one and three will blend 'mistakes' with recognisable phrases from the source material in a satisfying manner".

One difficulty with Markov chains is that the transition matrices can be very large and therefore occupy a significant amount of computer memory. Lyon (1995) tackles this issue by applying Petri Net techniques to composition in order to achieve computation of Markov processes with significant reductions in the amount of space needed to store the transition matrix in the computer.

The Markov chain approach provides an evolutionary structure as required by objective 4 and indeed is the technique chosen for use in this research, with the event space consisting of single musical parameter values, rather than pre-composed fragments, in accordance with objective 3: that the composer should not have to supply any preformed musical material. However, as mentioned above, one of the major problems in applying Markov

chains to the composition of music is the size of the transition matrix since, for a first-order Markov chain, the composer is required to supply a two-dimensional matrix of probability values whose size varies with the square of the number of possible parameter values. To produce a melody drawn from 10 different pitches, for example, requires a matrix of 100 numbers, and the problem grows exponentially as the order of the Markov chain increases. This is not just a problem from the programming perspective. It is also impractical for the composer to be expected to supply the probabilities explicitly. As we have seen, one way around this problem is to derive the matrices from pre-supplied music but this contravenes objective 3.

In this research, therefore, a technique has been developed, called the **Diagonal Line Method**, which enables a wide range of possibly very large Markov chain transition matrices to be generated, based on a very small set of input parameter values supplied by the user.

* * * *

In this introductory chapter, I have established the objectives behind the development of a mathematical algorithm to be used as the basis for computer-generated musical composition and identified the mathematical techniques which underlie this algorithm together with the reasons why they have been chosen in preference to alternative methods.

In the remainder of this thesis, I will provide a full explanation of the mathematical details of the algorithm, explain its implementation in a composing environment though a computer program, *Markov*, and explore, in detail, its capabilities and limitations.

Firstly, in Chapter 2, **THE ALGORITHM**, the mathematical details of the algorithm are presented, together with the compositional reasons why it was developed in this way. Then, in Chapter 3, **THE MARKOV PROGRAM**, the structure and functionality of the *Markov* computer program are described. Chapter 4, **THE COMPOSITIONAL PROCESS**, explores the compositional environment afforded by the program, including a step by step example of a composition being developed. Next, in Chapter 5, **ANALYSIS OF THE ALGORITHM**, a rigorous study is carried out of the ways in which varying the algorithm's input parameters affects the musical output. Now, in the final two chapters of the thesis, the lessons learned in Chapters 4 and 5 can be applied to the composition of music: Chapter 6, **STYLE EMULATION**, investigates the ability of the algorithm to produce music based on various given styles and Chapter 7, **COMPOSITIONAL STUDIES**, describes some original compositional

studies of my own which were produced using the *Markov* program. There are four appendices: Appendix A, **MARKOV PROGRAM USER GUIDE** (separate copies are held on the accompanying compact disc and floppy disk), provides a comprehensive user manual for the *Markov* program, Appendix B, **MARKOV PROGRAM SCORES**, contains the program-generated parametric "scores" for the pieces discussed in Chapters 6 and 7, Appendix C, **BACH SCORES IN STAFF NOTATION**, contains scores in traditional staff notation for various program-generated realisations of a piece of Bach Harpsichord music, one of the styles studied in Chapter 6, and Appendix D, **MARKOV PROGRAM SOURCE CODE LISTINGS** (held on the accompanying compact disc), provides the complete source code listings for the *Markov* computer program.

Chapter 2

The Algorithm

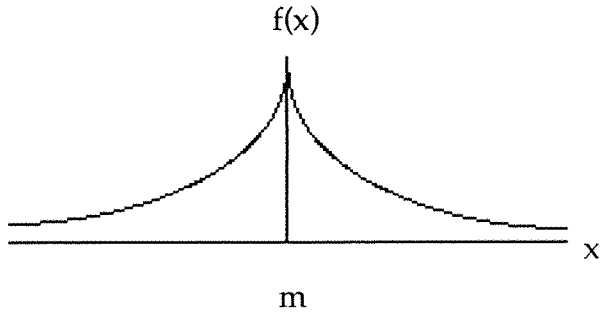
2.1 INTRODUCTION

In the **Introduction** chapter to this thesis, I explained that, due to the desire for the music resulting from the algorithm to have an evolutionary structure, Markov chains have been chosen as the underlying mathematical method (see Section **1.2.4.4**). It was seen there, however, that a major problem with the use of the Markov chain in the composition of music is the size of the transition matrix. Therefore, what is required is a way of generating the matrices according to some pre-specified mathematical rules. The mathematical algorithm developed for the *Markov* program provides such a method, and is described in detail in this chapter.

2.2. GENERATING THE ROWS OF THE TRANSITION MATRIX

2.2.1 The Bilateral Exponential Function

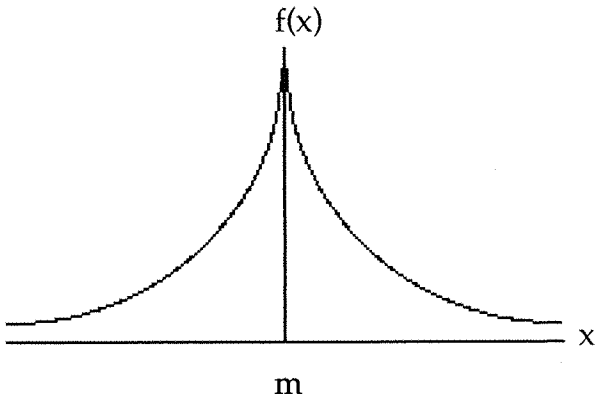
In order to avoid the need to specify each of the individual probability values which make up the rows of the transition matrix, the **complete** set of probability values for any particular row is generated from a **single** function which, as we shall see, has only two parameters and will generate the values for one complete row of the matrix no matter how large it is. Thus, for, say, a 25 x 25 matrix, only the two parameter values for the single function are required to produce the probability values for one row rather than 25.
The function used is the **bilateral exponential** function:



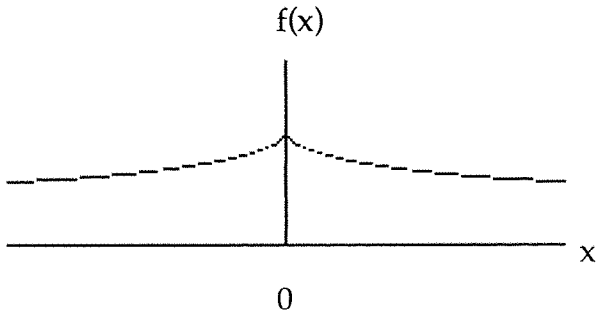
This function has the formula

$$f(x) = \frac{1}{2} \lambda e^{-\lambda |x-m|}$$

where m is the midpoint, or **mean**. The parameter λ provides the degree of probabilistic control required (Objective 5) by controlling the **compactness** of the function - the larger the value of λ the more tightly packed the function is around the mean:

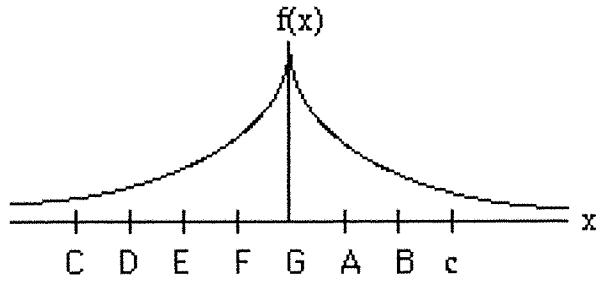


The smaller the value of λ the more spread is the function:



The **bilateral exponential** function was chosen because its single-peaked nature, together with the wide variations in spread which are achievable through the λ parameter provide, when used in conjunction with the Diagonal Line Method (see 2.2.2 below), a powerful mechanism for note sequence generation.

The following example uses eight notes of a C major scale[1] from C to c, centring around a mean pitch value of G:



[1] While a diatonic major scale is used in the illustrative examples throughout this chapter, any pitch collection can be used when composing with the algorithm.

Thus, for a high λ value the probabilities[2] that, say, G is followed by either C, D, E, F, G, A, B or c might be:

C	D	E	F	G	A	B	c
0.01	0.025	0.05	0.17	0.5	0.17	0.05	0.025

while for a low λ value they might be:

C	D	E	F	G	A	B	c
0.08	0.09	0.11	0.15	0.22	0.15	0.11	0.09

Thus, in line with objective **5**, if λ is very high then G will almost certainly be followed by G, if λ is quite high then G will most likely be followed by G but may sometimes be followed by F or A, and as λ becomes lower it becomes increasingly likely that any of the pitches could follow G. This behaviour is quantified more precisely in Section **5.2** where musical results are explored for various specific λ values. It is also shown that for sufficiently high λ,the behaviour becomes completely deterministic e.g. G is **always** followed by G[3].

All that the composer needs to supply now, for each of the possible values in the range, are the corresponding values of λ and the mean, m, thereby completely specifying all rows of the transition matrix. However, this still requires two parameter values (λ and m) for each row of the transition matrix and this is far too many too satisfy objective **2** (for example, a 25 x 25 transition matrix requires 50 parameter values). This is cut by half by assuming the **same** λ value for all rows of the matrix. To reduce the number of required parameters still further, an extension to the algorithm is required which enables the means for **all** rows of the matrix to be generated from a **single** function. The **Diagonal Line Method**, described in the next section, provides such an extension.

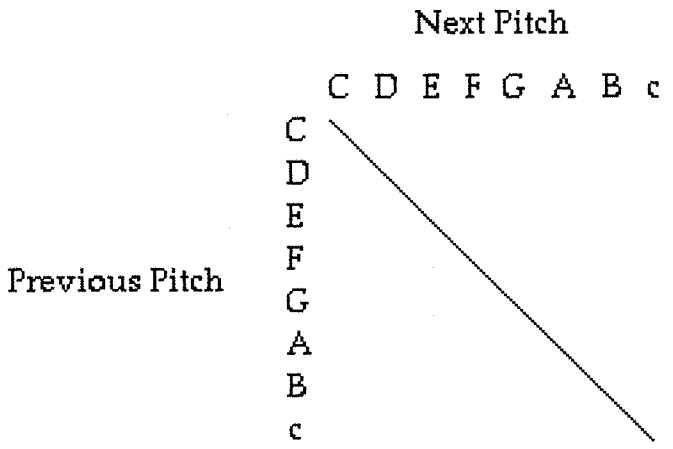
---

[2] To be precise, since the various note parameter ranges are **discrete** valued (discrete pitches, discrete note lengths and so on), the value of the transition probability for each of the possible parameter values in one row of the transition matrix is given by the **area** under the bilateral exponential curve centred around that value.

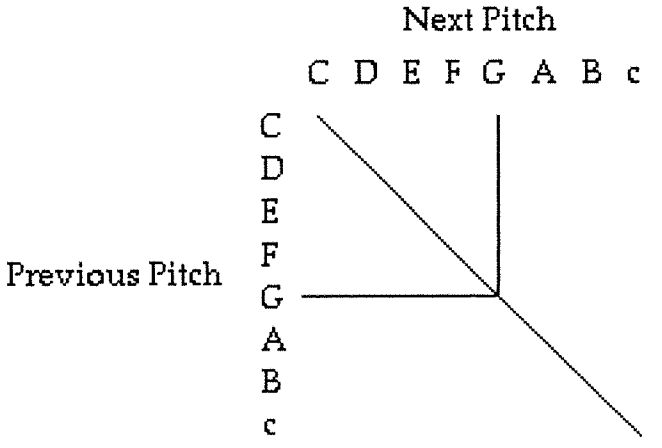
[3] For the purpose of calculation, pitch values are represented as their corresponding MIDI pitch numbers (e.g. middle C = 60) so as to provide a sequential numerical range of values.

2.2.2 The Diagonal Line Method

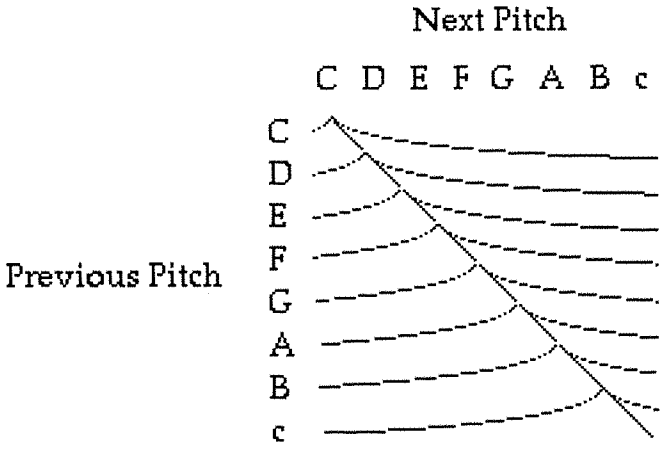
Consider the following diagram:



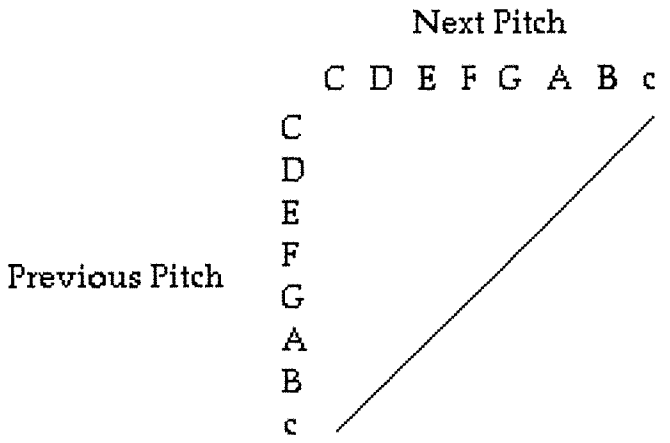
The possible values of the pitch of the previous note are shown on the vertical axis and the possible values of the pitch of the next note to be played are shown on the horizontal axis. A 45^0 downward sloping diagonal line has been specified. To determine the mean of the bilateral exponential function for any row of the transition matrix, draw a line horizontally from the previous pitch until it meets the diagonal line, then draw a line vertically upwards to meet the "Next Pitch" axis and read off the value. For example, if the previous pitch is G:



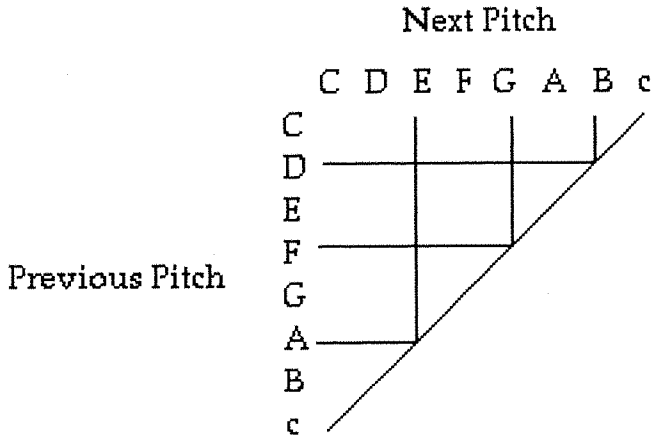
then the pitch of the next note will tend to be G also, and it is not difficult to see, for this particular diagonal line, that for any previous pitch, the pitch of the next note will tend to stay the same (the higher the value of λ, the stronger will be this tendency, while the lower the value of λ, the more "drift" there will be). Thus, the diagonal line defines the following **family** of bilateral exponential curves, one for each row of the transition matrix:



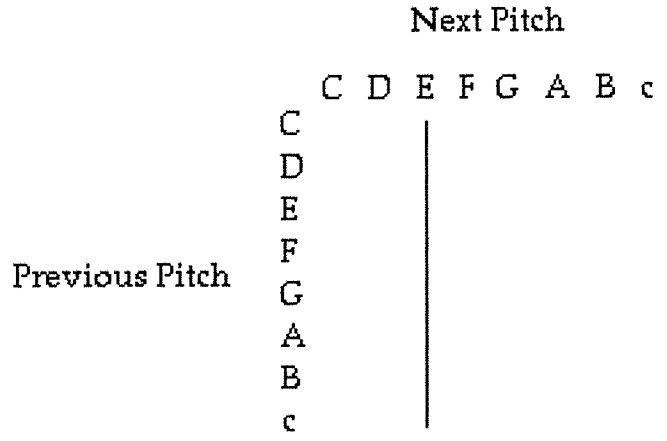
(this diagram should be interpreted in three dimensions, with the curves rising upward from the matrix in accordance with the value of λ).
Now consider another example:



Here, the diagonal is a 45^0 upward sloping line. Now, pitch values towards the extremes of the range will tend to jump to the opposite end of the range while pitches in the middle of the range will tend to stay the same; for example, a D will tend to jump up to a B, an A will tend to jump down to an E and an F will tend to jump only one note up to a G:



Finally, consider the following:



Here, no matter what the value of the previous pitch, the value of the next pitch will tend to be E; that is, the melody will be **centred** around E.

In general, **any** diagonal may be used. For the purposes of this discussion, we shall consider only diagonals from 45^0 downward to 45^0 upward sloping, but other cases will be considered in Section **2.3** below.

The above discussion uses pitch as an example but the same Diagonal Line Method is used to control note length, dynamic, and other characteristics, vibrato for example.

2.2.3 Generating Note Sequences

Each note parameter (pitch, length, dynamic and so on) has its own transition matrix and is calculated separately so as to fully determine all the parameter values for the next note in the sequence.

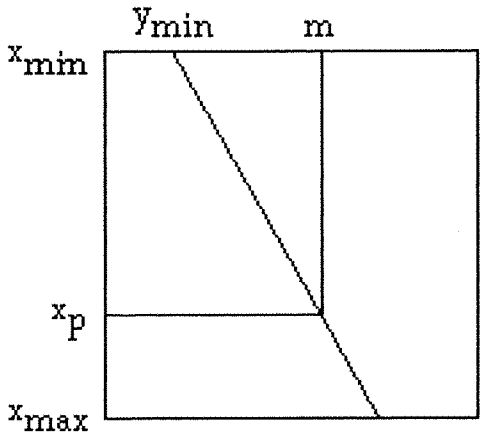
For each parameter, supposing that the value of the parameter for the **previous** note is x_p, the 2-step procedure to generate the parameter value, x_n, for the **next** note in the sequence is as follows:-

1. Calculate the mean of the bilateral exponential function for the row corresponding to x_p using the Diagonal Line Method

2. Calculate the parameter value, x_n, for the next note based on the transition probabilities given by the bilateral exponential function for the row corresponding to x_p.

Step 1 - Calculating the mean of the bilateral exponential function corresponding to x_p

Let $[x_{min}, x_{max}]$ be the range of possible parameter values, y_{min} the start-point value of the diagonal line (referred to in this work as the **Minimum Mean**), g the gradient of the diagonal line and m the mean to be calculated:



Then, using the formula for a straight line, m is given by:

$$m = g(x_p - x_{min}) + y_{min} \qquad (1)$$

where g is given by[4]:

$$\frac{\text{distance covered by line on horizontal axis}}{\text{distance covered by line on vertical axis}}$$

[4] Usually, the gradient of a straight line is given by the inverse of the formula given here, However, due to the nature of the transition matrix, the "x-axis" is the vertical axis in the diagrams shown here and the "y-axis" is the horizontal axis, contrary to Cartesian geometry.

Step 2 - Calculating the parameter value for the next note

We wish to generate a parameter value, x_n, for the next note, which obeys the transition probabilities given by the bilateral exponential function for the row corresponding to x_p. That it is to say, although we do not necessarily know in advance exactly what the value of x_n will be, we want it to be the case that:

(i) A value with a higher transition probability is more likely to occur than a value with a lower transition probability.
(ii) The likelihood of occurrence of a value increases in proportion to its transition probability .

For instance, if the probability of a transition from, say, pitch E to pitch F is 0.5 while to pitch G it is 0.25, it should be twice as likely that E is followed by F than that it is followed by G. To achieve this, Monte Carlo techniques are used (Naylor et al 1966). Monte Carlo exploits the ability of computers to generate very large sequences of uniformly distributed random real numbers from a specified range[5] (by **uniformly distributed** we mean that all values in the range are equally likely to occur: for example, if a fair dice, as opposed to a loaded one, is rolled repeatedly a large number of times, we would expect the occurrences of the values 1 to 6 to be uniformly distributed). The random numbers generated are then adapted for the problem at hand.

Here, a random real number N is first generated from the range 0 to 1. Then, to apply Monte Carlo, we need a mapping which will produce a corresponding value of x_n which satisfies our probability requirements. To achieve this we use the **cumulative distribution**, F(x), for the bilateral exponential function, where

$$F(x) = Pr(x_n \leq x)$$

In other words, F(x) is the probability that x_n is less than or equal to x (note that $0 \leq F(x) \leq 1$). For example, the values of F(x) for a fair dice would be:

$$F(1) = 1/6$$
$$F(2) = 2/6 = 1/3$$

[5] To be precise, the numbers generated by the computer are **pseudorandom**: they require a starting value, or **seed**, which is fed into a recursive formula to generate the sequence. The same seed used on two separate occasions would produce exactly the same sequence, so a common technique, which is employed by the *Markov* program, is to calculate the seed value from the current date and time to the nearest second, thus guaranteeing that no two runs of the program produce the same seed. Strictly speaking, the sequence is deterministic rather than random because it is generated from a precise recursive formula, but the sequence of numbers satisfies statistical tests for randomness and is therefore appropriate for use in Monte Carlo applications.

$F(3) = 3/6 = 1/2$
$F(4) = 4/6 = 2/3$
$F(5) = 5/6$
$F(6) = 1$

We then find the value, x_n, of x for which F(x) is equal to the generated random number N (Naylor et al 1966: 68-73); that is,

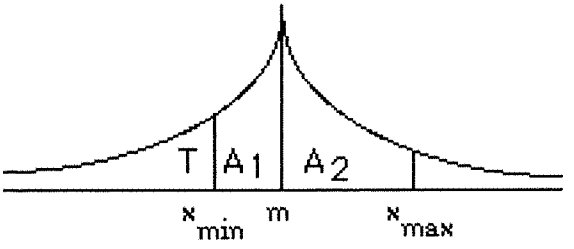
$$F(x_n) = N$$

If we can solve this equation then we are able to calculate the required next value, x_n.
Now, for the bilateral exponential function, F(x) is given by

$$F(x) = 1/2\, e^{\lambda(x - m)} \qquad x - m \leq 0$$

$$F(x) = 1 - 1/2\, e^{-\lambda(x - m)} \qquad x - m > 0$$

However, the bilateral exponential function has infinite tails (that is, left and right hand ends) whereas the note parameter values have a finite range. Therefore it is necessary to **normalise** the function so that the area under the curve over the parameter value range equals unity.
Let T = the area of the left hand tail and $A = A_1 + A_2$ = the area over the parameter value range:



Then, after normalisation, x_n is given by:

$$(F(x_n) - T)/A = N$$

Now,

$$T = \int_{-\infty}^{x_{min}} 1/2\,\lambda e^{\lambda(x-m)}\,dx$$

$$= 1/2\left[e^{\lambda(x-m)}\right]_{-\infty}^{x_{min}}$$

$$= 1/2\,e^{\lambda(x_{min}-m)}$$

$$A_2 = \int_{m}^{x_{max}} 1/2\,\lambda e^{-\lambda(x-m)}\,dx$$

$$= -1/2\left[e^{-\lambda(x-m)}\right]_{m}^{x_{max}}$$

$$= 1/2 - 1/2\,e^{-\lambda(x_{max}-m)}$$

$$= 1/2\,(1 - e^{-\lambda(x_{max}-m)})$$

$$A_1 = 1/2 - T$$

$$= 1/2 - 1/2\,e^{\lambda(x_{min}-m)}$$

$$= 1/2\,(1 - e^{\lambda(x_{min}-m)})$$

and

$$A = A_1 + A_2$$

$$= 1 - 1/2\,(e^{-\lambda(x_{max}-m)} + e^{\lambda(x_{min}-m)})$$

To calculate x_n, we must distinguish between two separate cases:

1. $N \leq A_1/A$ that is, $x_n \leq m$

2. $N > A_1/A$ that is, $x_n > m$

If $N \leq A_1/A$, then

$$(F(x_n) - T)/A = N$$

$$<=> 1/2\, e^{\lambda(x_n - m)} - T = NA$$

$$<=> e^{\lambda(x_n - m)} = 2(NA + T)$$

$$<=> \lambda(x_n - m) = \ln 2(NA + T)$$

$$<=> \mathbf{x_n = (\lambda m + \ln 2(NA + T))/\lambda} \qquad \textbf{(2)}$$

If $N > A_1/A$, then

$$(F(x_n) - T)/A = N$$

$$<=> 1 - 1/2\, e^{-\lambda(x_n - m)} - T = NA$$

$$<=> 1 - 1/2\, e^{-\lambda(x_n - m)} = NA + T$$

$$<=> -1/2\, e^{-\lambda(x_n - m)} = NA + T - 1$$

$$<=> e^{-\lambda(x_n - m)} = -2(NA + T - 1)$$

$$<=> -\lambda(x_n - m) = \ln -2(NA + T - 1)$$

$$<=> \mathbf{x_n = (\lambda m - \ln -2(NA + T - 1))/\lambda} \qquad \textbf{(3)}$$

Since the musical parameters are discrete valued, the value of x_n thus obtained is now rounded to the nearest discrete value.

2.3. EXTENSIONS TO THE DIAGONAL LINE METHOD

2.3.1 Introduction

All the diagonal lines considered thus far have met the bottom edge of the coordinate rectangle before, or at the same time as, meeting the right-hand edge; that is, they have all satisfied the condition

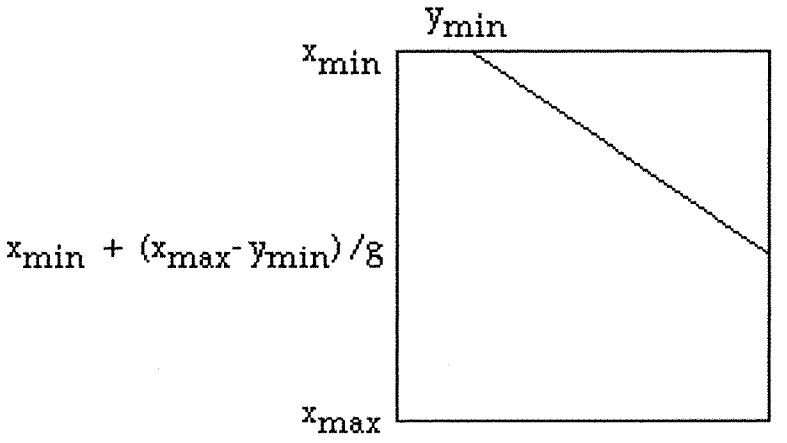
$$g(x_{max} - x_{min}) + y_{min} \leq x_{max}$$

and in the early stages of development of the algorithm, the diagonal line was restricted to this family. However, although this somewhat limited set of lines produces a very wide variety of musical results, it was realised that a large set of diagonal lines was being excluded which could potentially produce yet more variation in output.

Lines not yet considered are those which meet the right-hand edge of the coordinate rectangle before meeting the bottom edge; that is, those for which

$$g(x_{max} - x_{min}) + y_{min} > x_{max}$$

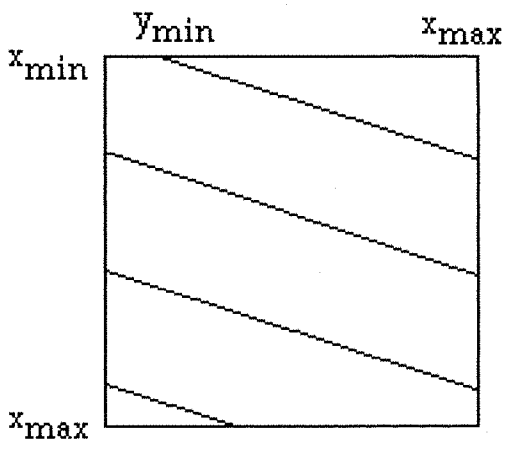
The problem with these lines is that it is impossible to calculate values of the next mean for certain previous parameter values (specifically those greater than $x_{min} + (x_{max} - y_{min}) / g$) because a horizontal line drawn from them does not meet the diagonal line, as in the following:-



Therefore, we need to find a way to continue the line from where it meets the right hand edge so that it encompasses the complete range of x_p values. The algorithm offers the composer a choice of two alternatives for dealing with this situation, and these are described in detail in the next two sections.

2.3.2 Wraparound

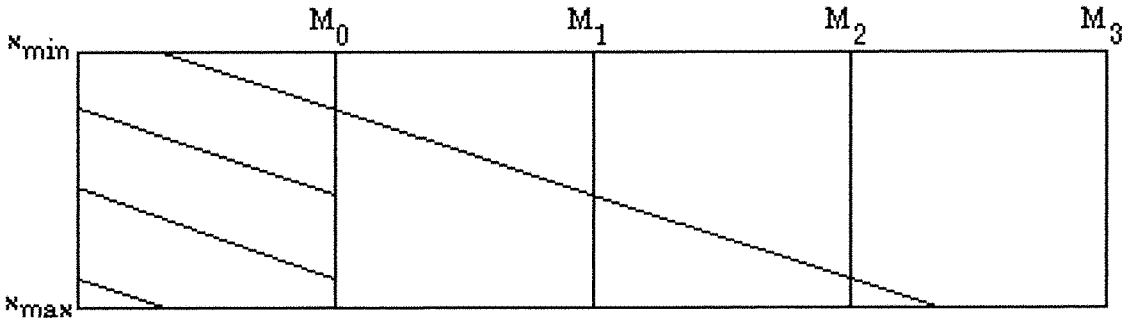
One possibility is to employ **wraparound**, so that the line **re-emerges** at the left-hand edge:-



In order to correctly calculate the required next mean value given the previous parameter value, we need to **remap** values for which the straight line formula **(1)**:-

$$\mathbf{m = g(x_p - x_{min}) + y_{min}}$$

yields values of m outside the range x_{min} to x_{max}[6].
For a positive gradient, g, redraw the diagonal line as follows:-

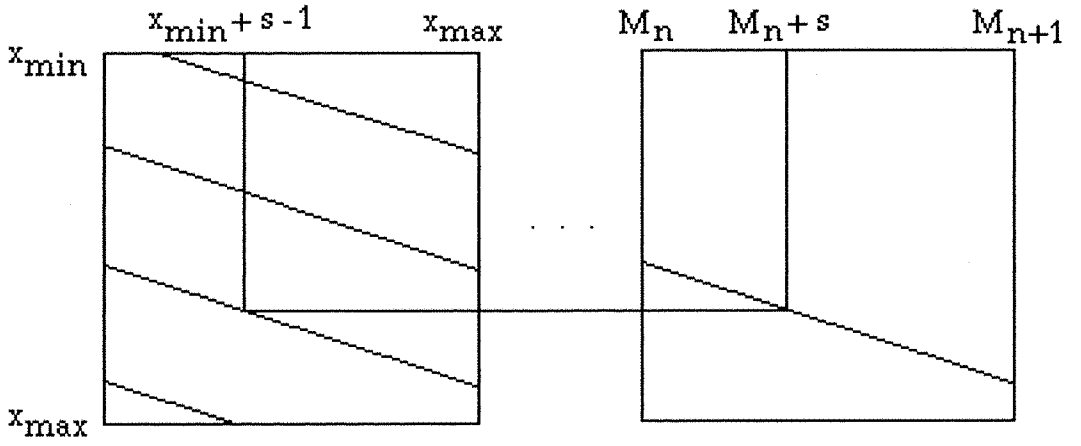


6 Since the parameter values are discrete and the methods that follow assume that the result, m, of this formula is integer, m should be rounded to the nearest integer. It can therefore be assumed that wherever the expression $m = g(x_p - x_{min}) + y_{min}$ appears as a term in the formulae that follow, its value is integer.

where

$$M_0 = x_{max}$$
$$M_1 = x_{max} + (x_{max} - x_{min} + 1)$$
$$M_2 = x_{max} + 2(x_{max} - x_{min} + 1)$$
$$M_3 = x_{max} + 3(x_{max} - x_{min} + 1)$$

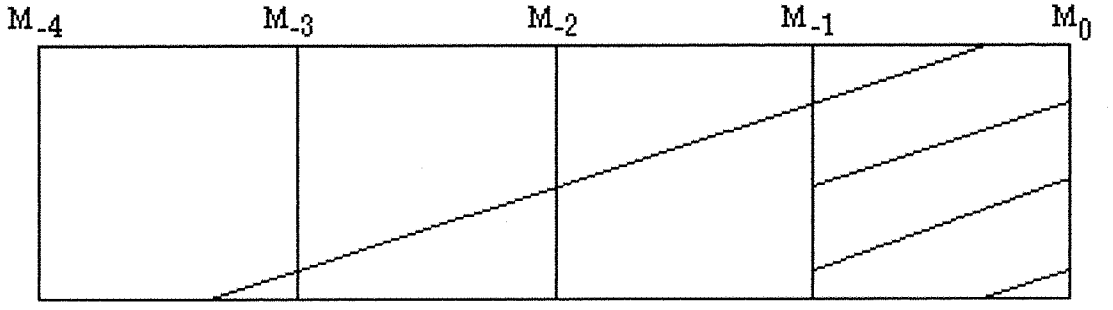
Now consider any region $M_n < m \leq M_{n+1}$ ($M_1 < m \leq M_2$ above, for example):-



Then, letting $m=M_n + s$, where $0 < s \leq x_{max} - x_{min} + 1$, and letting the mapped value of m equal m', m maps to $x_{min} + s - 1$; that is,

$$m = x_{max} + n(x_{max} - x_{min} + 1) + s \ \textit{maps to}\ m' = x_{min} + s - 1, \qquad \textbf{(4)}$$

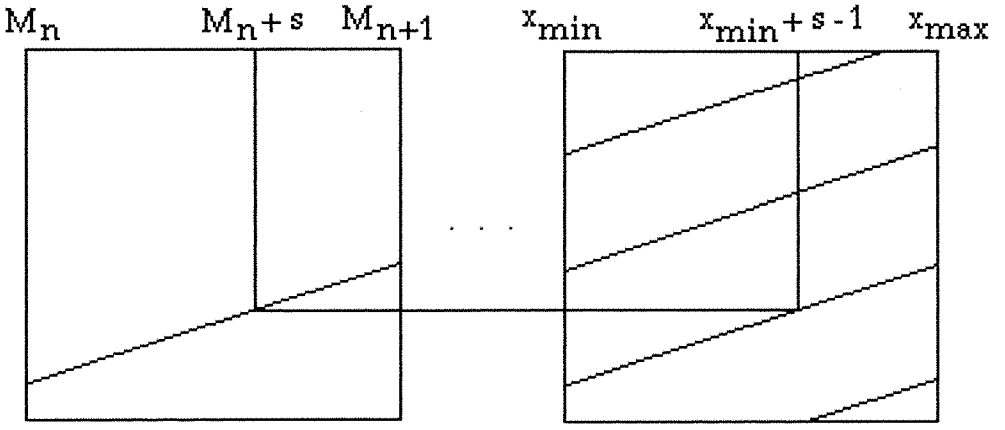
For $g < 0$, redraw the diagonal line as follows:-



where

$$M_{-1} = x_{max} - (x_{max} - x_{min} + 1)$$
$$M_{-2} = x_{max} - 2(x_{max} - x_{min} + 1)$$
$$M_{-3} = x_{max} - 3(x_{max} - x_{min} + 1)$$

Now consider any region $M_n < m \leq M_{n+1}$, $n < 0$ ($M_{-3} < m \leq M_{-2}$ above, for example):-



Again, as in **(4)**,

$$m = x_{max} + n(x_{max} - x_{min} + 1) + s \ \textit{maps to} \ m' = x_{min} + s - 1$$

and the complete mapping can be represented in the following listing:-

m	m'
.	.
.	.
.	.
$x_{max} - 3(x_{max} - x_{min} + 1) + (x_{max} - x_{min})$	$x_{max} - 1$
$x_{max} - 3(x_{max} - x_{min} + 1) + (x_{max} - x_{min} + 1)$	x_{max}
$x_{max} - 2(x_{max} - x_{min} + 1) + 1$	x_{min}
$x_{max} - 2(x_{max} - x_{min} + 1) + 2$	$x_{min} + 1$
.	.
.	.
.	.
$x_{max} - 2(x_{max} - x_{min} + 1) + (x_{max} - x_{min})$	$x_{max} - 1$
$x_{max} - 2(x_{max} - x_{min} + 1) + (x_{max} - x_{min} + 1)$	x_{max}
$x_{max} - (x_{max} - x_{min} + 1) + 1$	x_{min}
$x_{max} - (x_{max} - x_{min} + 1) + 2$	$x_{min} + 1$
.	.
.	.
.	.
$x_{min} - 1$	x_{max}
x_{min}	x_{min}

$x_{min} + 1$	$x_{min} + 1$
.	.
.	.
.	.
$x_{max} - 1$	$x_{max} - 1$
x_{max}	x_{max}
$x_{max} + 1$	x_{min}
$x_{max} + 2$	$x_{min} + 1$
.	.
.	.
.	.
$x_{max} + (x_{max} - x_{min})$	$x_{max} - 1$
$x_{max} + (x_{max} - x_{min} + 1)$	x_{max}
$x_{max} + (x_{max} - x_{min} + 1) + 1$	x_{min}
$x_{max} + (x_{max} - x_{min} + 1) + 2$	$x_{min} + 1$
.	.
.	.
.	.
$x_{max} + (x_{max} - x_{min} + 1) + (x_{max} - x_{min})$	$x_{max} - 1$
$x_{max} + (x_{max} - x_{min} + 1) + (x_{max} - x_{min} + 1)$	x_{max}
$x_{max} + 2(x_{max} - x_{min} + 1) + 1$	x_{min}
$x_{max} + 2(x_{max} - x_{min} + 1) + 2$	$x_{min} + 1$
.	.
.	.
.	.
$x_{max} + (n - 1)(x_{max} - x_{min} + 1) + (x_{max} - x_{min})$	$x_{max} - 1$
$x_{max} + (n - 1)(x_{max} - x_{min} + 1) + (x_{max} - x_{min} + 1)$	x_{max}
$x_{max} + n(x_{max} - x_{min} + 1) + 1$	x_{min}
$x_{max} + n(x_{max} - x_{min} + 1) + 2$	$x_{min} + 1$
.	.
.	.
.	.

The two cases $m \geq x_{min}$ and $m < x_{min}$ must be handled separately.
For $m \geq x_{min}$, let

$$m_1 = m - x_{min} = x_{max} + n(x_{max} - x_{min} + 1) + s - x_{min}$$

$$= (n + 1)(x_{max} - x_{min} + 1) + s - 1$$

Let

$$m_2 = m_1 \bmod (x_{max} - x_{min} + 1) = s - 1$$

Finally, let

$$m_3 = m_2 + x_{min} = x_{min} + s - 1$$

which is the mapping required in **(4)**. Thus, combining the above three transformations, we can deduce that the general wraparound formula for $m \geq x_{min}$ is

$$\mathbf{m' = [g(x_p - x_{min}) + y_{min} - x_{min}] \bmod (x_{max} - x_{min} + 1) + x_{min}}$$

For $m < x_{min}$ (and therefore $n < 0$), let

$$m_1 = m - x_{max} = x_{max} + n(x_{max} - x_{min} + 1) + s - x_{max}$$

$$= n(x_{max} - x_{min} + 1) + s$$

$$= (n + 1)(x_{max} - x_{min} + 1) + s - (x_{max} - x_{min} + 1)^7$$

Let

$$m_2 = m_1 \bmod (x_{max} - x_{min} + 1) = s - (x_{max} - x_{min} + 1)$$

Finally, let

$$m_3 = m_2 + x_{max} = x_{min} + s - 1$$

which is the mapping required in **(4)**. Thus, combining the above three transformations, we can deduce that the general wraparound formula for $m < x_{min}$ is

$$\mathbf{m' = [g(x_p - x_{min}) + y_{min} - x_{max}] \bmod (x_{max} - x_{min} + 1) + x_{max}}$$

and, combining the results for $g > 0$ and $m < x_{min}$ we obtain the complete wraparound result:-

---

[7] This last adjustment is necessary because, since the value is negative and we are about to use modulo arithmetic, we must ensure that the remainder is non-positive

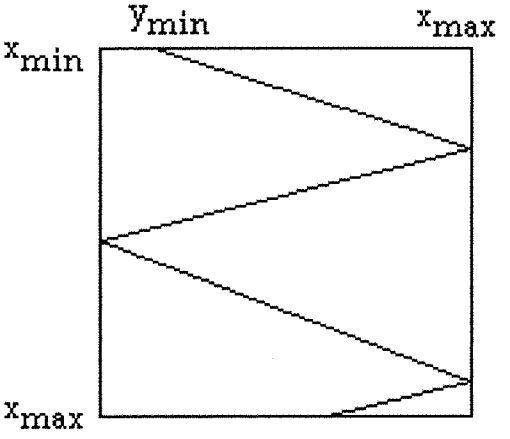
$$m' = \{ [g(x_p - x_{min}) + y_{min} - x_{min}] \bmod (x_{max} - x_{min} + 1) + x_{min}, \; m \geq x_{min}$$
$$\{ [g(x_p - x_{min}) + y_{min} - x_{max}] \bmod (x_{max} - x_{min} + 1) + x_{max}, \; m < x_{min}$$

(5)

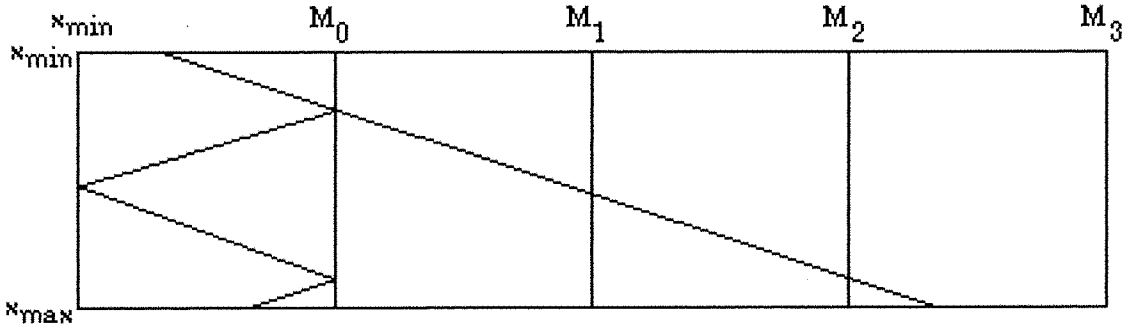
The value of m' thus obtained is now used to calculate the next parameter value, x_n, by applying formulae **(2)** and **(3)**, as described in Section **2.2.3** above.

2.3.3 Reflection

An alternative possibility is **reflection**. Here, diagonal lines which meet the left- or right-hand edge of the coordinate rectangle **reflect** from it, as in the following:-



Again, we need to find the required mapping for values outside the range x_{min} to x_{max}.
For a positive gradient, g, redraw the diagonal line as follows:-



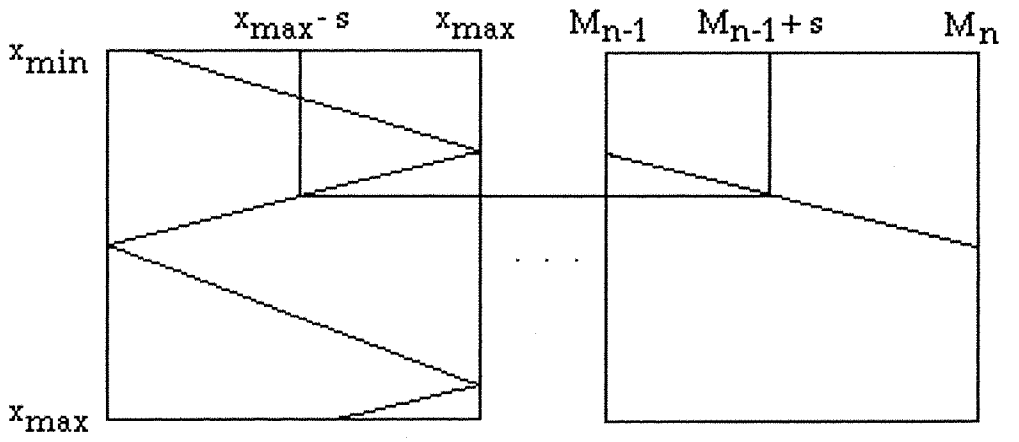
where

$$M_0 = x_{max} = x_{min} + (x_{max} - x_{min})$$
$$M_1 = x_{min} + 2(x_{max} - x_{min})$$
$$M_2 = x_{min} + 3(x_{max} - x_{min})$$
$$M_3 = x_{min} + 4(x_{max} - x_{min})$$

and in general

$$M_n = x_{min} + (n+1)(x_{max} - x_{min})$$

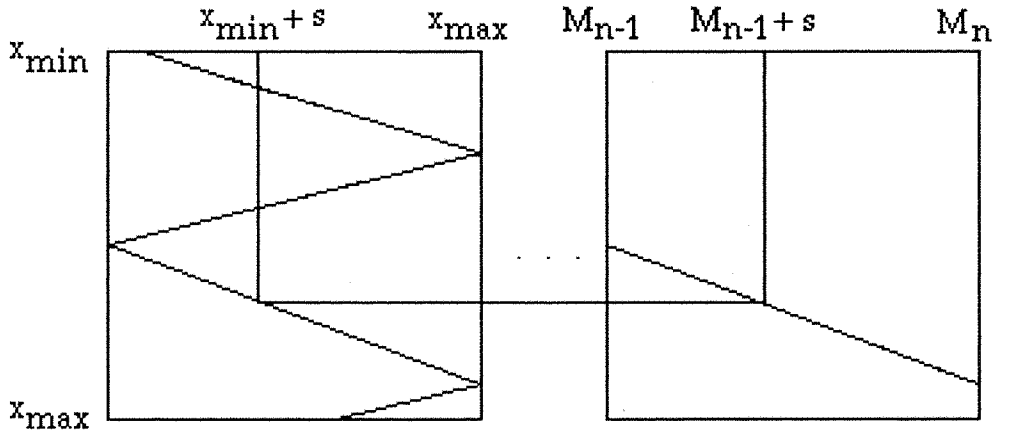
First, consider regions where $M_{n-1} < m \leq M_n$, n odd ($M_0 < m \leq M_1$ above, for example):-



Then, letting $m = M_{n-1} + s$, where $0 < s \leq x_{max} - x_{min}$, m maps to $x_{max} - s$; that is,

$$m = x_{min} + n(x_{max} - x_{min}) + s \textit{ maps to } m' = x_{max} - s \qquad \textbf{(6)}$$

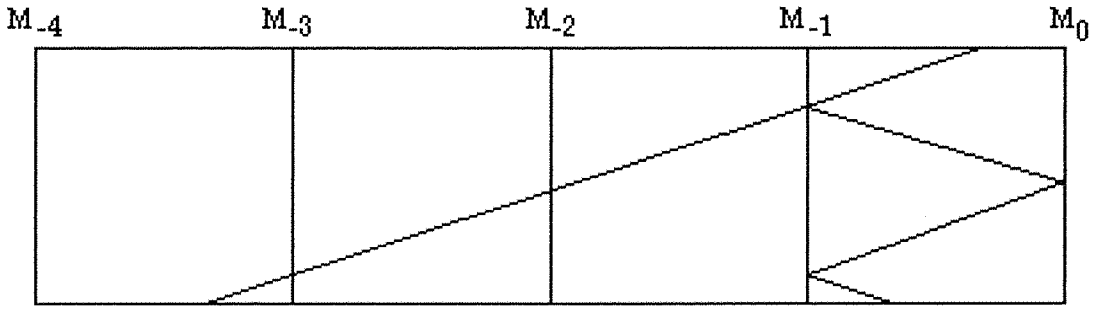
Next, consider regions where $M_{n-1} < m \leq M_n$, n even ($M_1 < m \leq M_2$ above, for example):-



In this case,

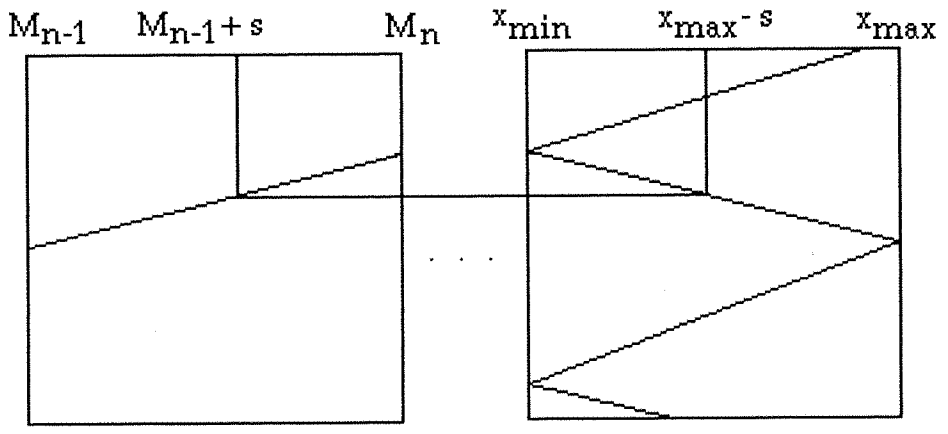
$$m = x_{min} + n(x_{max} - x_{min}) + s \; maps \; \text{to} \; m' = x_{min} + s \qquad (7)$$

For $g < 0$, redraw the diagonal line as follows:-



$M_{-1} = x_{min}$
$M_{-2} = x_{min} - (x_{max} - x_{min})$
$M_{-3} = x_{min} - 2(x_{max} - x_{min})$
$M_{-4} = x_{min} - 3(x_{max} - x_{min})$

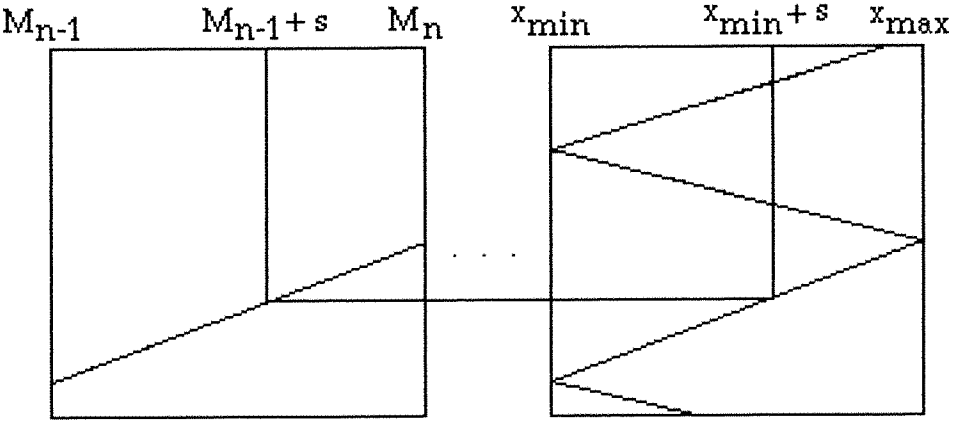
First, consider regions where $M_{n-1} < m \leq M_n$, n odd ($M_{-2} < m \leq M_{-1}$ above, for example):-



Again, as in **(6)**,

$$m = x_{min} + n(x_{max} - x_{min}) + s \textit{ maps to } m' = x_{max} - s$$

Finally, consider regions where $M_{n-1} < m \leq M_n$, n even ($M_1 < m \leq M_2$ above, for example):-



Again, as in **(7)**,

$$m = x_{min} + n(x_{max} - x_{min}) + s \textit{ maps to } m' = x_{min} + s$$

and the general mapping can be listed as follows:-

m	m '
.	.
.	.
.	.
$x_{min} - 3(x_{max} - x_{min}) + (x_{max} - x_{min} - 1)$	$x_{min} + 1$
$x_{min} - 2(x_{max} - x_{min})$	x_{min}
$x_{min} - 2(x_{max} - x_{min}) + 1$	$x_{min} + 1$

$x_{min} - 2(x_{max} - x_{min}) + 2$	$x_{min} + 2$
.	.
.	.
.	.
$x_{min} - 2(x_{max} - x_{min}) + (x_{max} - x_{min} - 1)$	$x_{max} - 1$
$x_{min} - (x_{max} - x_{min})$	x_{max}
$x_{min} - (x_{max} - x_{min}) + 1$	$x_{max} - 1$
$x_{min} - (x_{max} - x_{min}) + 2$	$x_{max} - 2$
.	.
.	.
.	.
$x_{min} - 1$	$x_{min} + 1$
x_{min}	x_{min}
$x_{min} + 1$	$x_{min} + 1$
.	.
.	.
.	.
$x_{max} - 1$	$x_{max} - 1$
$x_{min} + (x_{max} - x_{min})$	x_{max}
$x_{min} + (x_{max} - x_{min}) + 1$	$x_{max} - 1$
$x_{min} + (x_{max} - x_{min}) + 2$	$x_{max} - 2$
.	.
.	.
.	.
$x_{min} + (x_{max} - x_{min}) + (x_{max} - x_{min} - 1)$	$x_{min} + 1$
$x_{min} + 2(x_{max} - x_{min})$	x_{min}
$x_{min} + 2(x_{max} - x_{min}) + 1$	$x_{min} + 1$
$x_{min} + 2(x_{max} - x_{min}) + 2$	$x_{min} + 2$
.	.
.	.
.	.
$x_{min} + 2(x_{max} - x_{min}) + (x_{max} - x_{min} - 1)$	$x_{max} - 1$
$x_{min} + 3(x_{max} - x_{min})$	x_{max}
$x_{min} + 3(x_{max} - x_{min}) + 1$	$x_{max} - 1$
$x_{min} + 3(x_{max} - x_{min}) + 2$	$x_{max} - 2$
.	.
.	.
.	.

For $m \geq x_{min}$, n odd, let

$$m_1 = m - x_{min} = n(x_{max} - x_{min}) + s$$

Let

$$m_2 = m_1 \bmod (x_{max} - x_{min}) = s$$

Finally, let

$$m_3 = x_{max} - m_2 = x_{max} - s$$

which is the mapping required in **(6)**. Thus, combining the above three transformations, we can deduce that the general reflection formula, for $m \geq x_{min}$, n odd, is

$$\mathbf{m' = x_{max} - [g(x_p - x_{min}) + y_{min} - x_{min}] \bmod (x_{max} - x_{min})}$$

For $m \geq x_{min}$, n even, let

$$m_1 = m - x_{min} = n(x_{max} - x_{min}) + s$$

Let

$$m_2 = m_1 \bmod (x_{max} - x_{min}) = s$$

Finally, let

$$m_3 = x_{min} + m_2 = x_{min} + s$$

which is the mapping required in **(7)**. Thus, combining the above three transformations, we can deduce that the general reflection formula, for $m \geq x_{min}$, n even, is

$$\mathbf{m' = x_{min} + [g(x_p - x_{min}) + y_{min} - x_{min}] \bmod (x_{max} - x_{min})}$$

For $m < x_{min}$, n odd, let

$$\begin{aligned} m_1 = m - x_{min} &= n(x_{max} - x_{min}) + s \\ &= (n+1)(x_{max} - x_{min}) + s - (x_{max} - x_{min}) \end{aligned}$$

Let

$$m_2 = m_1 \bmod (x_{max} - x_{min}) = s - (x_{max} - x_{min})$$

Finally, let

$$m_3 = x_{min} - m_2 = x_{max} - s$$

which is the mapping required in **(6)**. Thus, combining the above three transformations, we can deduce that the general reflection formula, for $m < x_{min}$, n odd, is

$$\mathbf{m' = x_{min} - [g(x_p - x_{min}) + y_{min} - x_{min}] \bmod (x_{max} - x_{min})}$$

For $m < x_{min}$, n even, let

$$\begin{aligned} m_1 &= m - x_{min} = n(x_{max} - x_{min}) + s \\ &= (n+1)(x_{max} - x_{min}) + s - (x_{max} - x_{min}) \end{aligned}$$

Let

$$m_2 = m_1 \bmod (x_{max} - x_{min}) = s - (x_{max} - x_{min})$$

Finally, let

$$m_3 = x_{max} + m_2 = x_{min} + s$$

which is the mapping required in **(7)**. Thus, combining the above three transformations, we can deduce that the general reflection formula, for $m < x_{min}$, n even, is

$$\mathbf{m' = x_{max} + [g(x_p - x_{min}) + y_{min} - x_{min}] \bmod (x_{max} - x_{min})}$$

We can determine whether n is odd or even as follows:-

Let $r = [(g(x_p - x_{min}) + y_{min} - x_{min}) \text{ div } (x_{max} - x_{min})] \bmod 2$

Then:-

$r = 0, m \geq x_{min} \Rightarrow$ n even
$r = 1, m \geq x_{min} \Rightarrow$ n odd

$r = 0, m < x_{min} \Rightarrow n$ odd

$r = 1, m < x_{min} \Rightarrow n$ even

Thus, the general reflection formula is as follows:-

$$m' = \begin{cases} \{ x_{min} + [g(x_p - x_{min}) + y_{min} - x_{min}] \bmod (x_{max} - x_{min}), & m \geq x_{min}, r = 0 \\ \{ x_{max} - [g(x_p - x_{min}) + y_{min} - x_{min}] \bmod (x_{max} - x_{min}), & m \geq x_{min}, r = 1 \\ \{ x_{min} - [g(x_p - x_{min}) + y_{min} - x_{min}] \bmod (x_{max} - x_{min}), & m < x_{min}, r = 0 \\ \{ x_{max} + [g(x_p - x_{min}) + y_{min} - x_{min}] \bmod (x_{max} - x_{min}), & m < x_{min}, r = 1 \end{cases}$$

(8)

The value of m' thus obtained is now used to calculate the next parameter value, x_n, by applying formulae **(2)** and **(3)**, as described in Section **2.3** above.

2.3.4 Reverse

Experiments with the algorithm show that, for certain Minimum Mean values used in conjunction with Diagonal Line Wraparound, a tendency for a parameter to move in a certain direction (for a melody to tend to rise, for example) occurs (see Conditions **9** below and Section **5.4**). However, in such cases, when a parameter reaches one end of the possible range of values it then jumps to the opposite end before continuing; for example, when a rising melody reaches the upper limit of the pitch range it jumps back down to the lower limit before beginning to rise again. While this might be a desired effect, it was felt that it would be appropriate to also provide the composer with the option for a rising parameter to begin falling again when it reaches the upper limit of the range, and for a falling parameter to begin rising again when it reaches the lower limit.

The purpose of Reverse is to effect a **change in direction of movement** of a musical parameter when it reaches its upper or lower limit; for example a melody which is tending to rise in pitch begins to fall when it reaches the maximum pitch value in its range, a melody which is tending to increase in tempo (due to note lengths becoming shorter) begins to decrease in tempo when it reaches the shortest note length in its range, and so on.

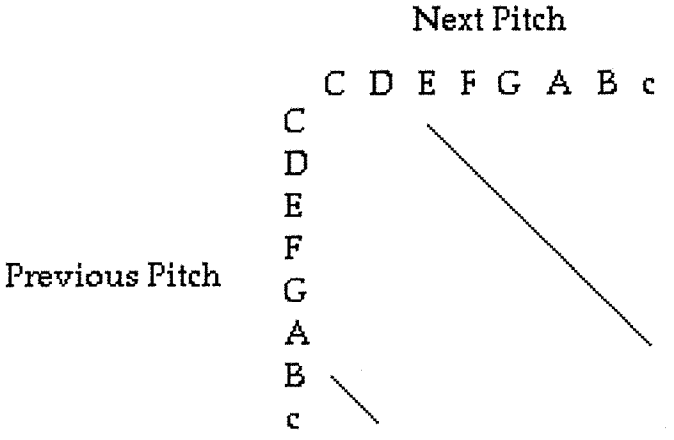
The problem is to be able to detect when a tendency to move in a particular direction is present and, if it is, when the parameter value has

reached the upper or lower limit of its range. The intention here is that the tendency to move should be by **design**, not by random accident, so that the composer can introduce reverse as an intentional controlling factor.

Now, a tendency to move in a particular direction occurs if the following conditions are all true:-

1. The gradient, g, of the diagonal line is positive
2. The minimum mean, y_{min}, is greater than x_{min} **(9)**
3. Wraparound is in effect

For example, in the following situation, the melody will tend to rise (C to E, E to G, G to B)



To check whether the upper or lower limit has been reached, it is not sufficient to check whether the parameter value equals the maximum possible value (in the case of a rising tendency), or the minimum possible value (in the case of a falling tendency), because this may not actually happen since the parameter may overshoot the maximum or minimum value. For example, in the above example, B tends to jump back down to the lower C, rather than moving up to the higher C which is the maximum possible value. Instead we apply the following test, which detects overshoot:-

> If the value is tending to **rise**, and the next mean is **less than** the previous parameter value ($m' < x_p$), then change direction
> If the value is tending to **fall**, and the next mean is **greater than** the previous parameter value ($m' > x_p$), then change direction

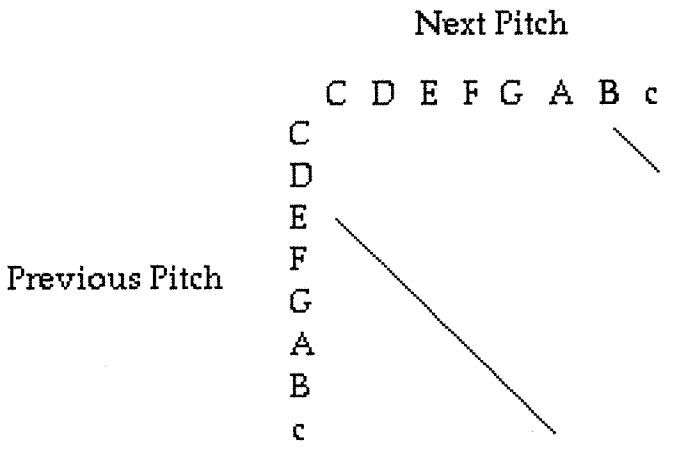
Now, if the conditions in **(9)** are satisfied, the value will tend to rise if the minimum mean, y_{min}, is less than the midpoint of the range, and will tend to fall if y_{min} is greater than the midpoint of the range; that is,

$y_{min} \leq (x_{max} - x_{min})/2$ => tendency to rise
$y_{min} > (x_{max} - x_{min})/2$ => tendency to fall

A change in direction is brought about by leaving the gradient, g, unchanged but modifying the value of the minimum mean, y_{min}, as follows:- if y'_{min} is the new minimum mean, then

$$y'_{min} = x_{max} - y_{min} + x_{min} + 1$$

For example, in the above case, where y_{min} is the pitch E and the pitch is tending to rise, applying the above change of direction formula gives a new y_{min} of B:-



Now, the upper C tends to move to A, A to F, F to D, which is a falling tendency, and an exact reversal of the previous upward movement.
Thus, summarising the above results, in order to apply the reverse effect we calculate a new minimum mean value, y'_{min}, according to the following rules:-

IF $g > 0$ **AND** $y_{min} > x_{min}$ **AND** Wraparound is on **THEN**

IF $[y_{min} \leq (x_{max} - x_{min})/2$ AND $m' < x_p]$ **OR**
$[y_{min} > (x_{max} - x_{min})/2$ AND $m' > x_p]$ **THEN**

$$y'_{min} = x_{max} - y_{min} + x_{min} + 1 \qquad \textbf{(10)}$$

2.4. SUMMARY

2.4.1 Satisfying the Objectives

In the **Introduction** chapter to this thesis, a set of objectives was stated (Section **1.1.2**) which the algorithm was to attempt to satisfy. How well does this algorithm meet those objectives?

1) The composer needs no understanding of the mathematics behind the algorithm in order to use it. What is required is a feel for how different diagonal lines, and the parameters associated with them, affect the musical output. This is explored in detail in Chapter 5, **ANALYSIS OF THE ALGORITHM**, but also, very importantly, this understanding is gained from the experience of experimenting with the algorithm through the *Markov* program, through a process of continually modifying and refining the parameter values to obtain a desired musical result.

2) For each musical parameter (pitch, note length, dynamic and so on), the following algorithm input values are required:-

the Minimum value of the parameter range
the Maximum value of the parameter range
the Minimum Mean value for the associated Diagonal Line
the Gradient of the Diagonal Line
the λ value for the bilateral exponential function

This is a total of just 5 input values. Additionally, the program offers the composer the option of either providing a starting value for the musical parameter (the first pitch of the note sequence, for example) or allowing it to be generated randomly from the associated range, so that there is a possible 6th input value if the composer chooses to provide the starting value explicitly (see Appendix A, Sections **3.3.2.** to **3.3.9**). In addition to this very small set of numerical input values, the composer makes three either/or choices. Two of these correspond to the Diagonal Line extensions described in Section 3 above:

should the Diagonal Line Wraparound (the default) or Reflect?
should the Reverse option be used?

The third choice concerns the possible successive repetition of a note parameter value. During experimentation with the algorithm, it was found that, particularly for high values of λ, sequences of repeated values of a note

parameter will occur. It was felt, certainly in the case of pitch, that the composer may not want this to occur, so the program allows the composer the option of disallowing two successive occurrences of the same value (see Appendix A, Sections **3.3.2.** to **3.3.4**). What actually happens is that if the algorithm does produce a value which is exactly the same as the previous value, then either the next higher or next lower value is used instead, this choice being made by the program at random.

Thus, in summary, for each musical parameter, the total amount of data which the composer has to provide to the algorithm consists of a maximum of just 6 input values plus 3 either/or choices.

3) The algorithm does not analyse any provided music in order to decide its parameter values, nor does the user provide any explicit note sequences (other than, possibly, a starting value, as discussed above). Instead, the algorithm, together with its input parameter values as provided by the composer, is a pure starting point for the production of music.

4) The use of Markov methods provides a sense of evolution of the music produced, with each note occurrence having a direct influence on the note that occurs next.

5) The use of the bilateral exponential function provides the composer with control of the degree of influence a note occurrence has on the note to occur next, through the parameter λ. The higher the value of λ the stronger is the degree of influence, the lower the value of λ the weaker the degree of influence.

6) The composer is not required to provide any musical rules which the music produced by the algorithm must obey. Actually, a very small compromise has been made here since the Reverse option and the choice to disallow two successive repetitions could be seen as simple rules.

2.4.2 Limitations

The intentional simplicity of the algorithm must necessarily impose some limitations on the variety of output which can be achieved.

Firstly, the shape of the bilateral exponential curve, having a **single** peak value, dictates that for any value of a note parameter, there is only **one** value which is the most likely to occur next. Thus, we cannot say, for example, that a pitch of C could be followed either a D or a G with equally high probability, with

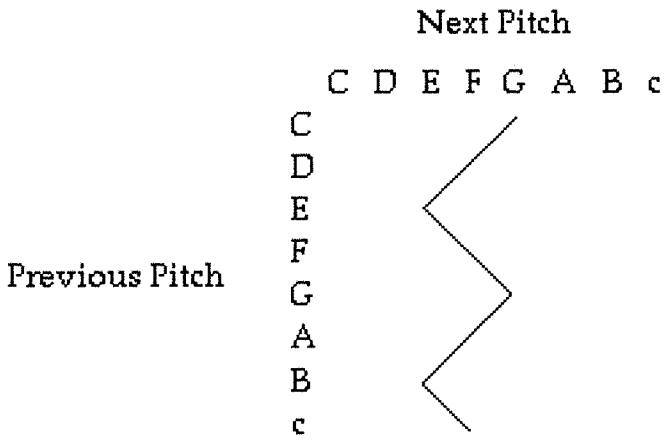
pitches other than those two being less likely. To put it another way, we cannot achieve a row of transition probabilities like the following:-

C	D	E	F	G	A	B	c
0.08	0.3	0.08	0.08	0.3	0.08	0.05	0.03

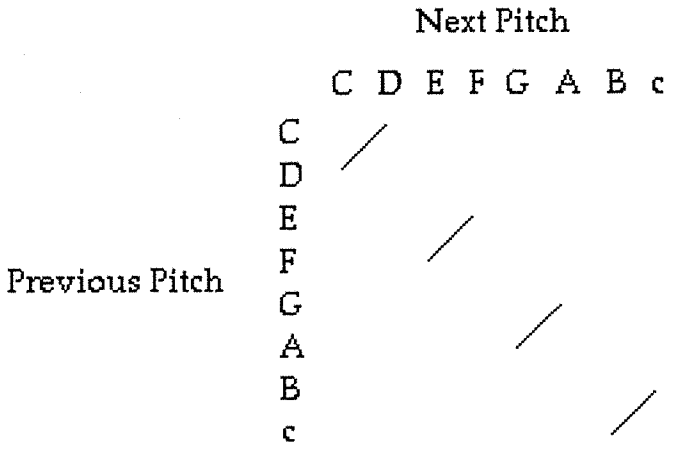
This could by achieved by generating the probabilities using a function with two (or more) peaks, for example:-

However, this more complex function now requires the composer to make separate decisions concerning each of the peaks, needs correspondingly more input parameter values and generally begins to become more inaccessible to non-mathematicians. In addition, once the composer becomes involved in the compositional process it becomes much less obvious how to modify parameters in order to achieve a desired result. True, by keeping to the single-peaked bilateral exponential function it is not possible to produce music which can make distinct "branching" decisions as it evolves (for example, if pitch C has been played then play either D or G next) but, for sufficiently low λ, each of various alternatives could, probabilistically, nevertheless occur. Furthermore, for certain gradient values, a musical parameter can exhibit distinct, alternative modes of behaviour which it "jumps" between so that branching effects are still possible (see Sections **5.5** and **5.9** for a much more detailed analysis).

Secondly, basing the mean generating process solely on a straight line rather than on more complex lines, or even curves, prevents many types of behaviour from occurring, as in the following example:



where E, F and G tend to be followed by another E, or another F, or another G respectively, while the lower C tends to rise by 4 steps to G and B tends to rise by 2 steps to E, or even the following:



where the melody will tend to move between different alternating pairs of pitches, C and D, E and F, G and A, B and c. However, allowing this degree of generality once again requires considerably more input parameters, it becomes much more difficult for the composer to relate changes in musical output to parametric changes and, actually, it is possible to produce sufficiently similar sequence patterns from an appropriately chosen single straight line, particularly when the gradient is greater than 1 or less than -1, so that the difference would not be readily discernible. Also, a point is reached where the composer is beginning to explicitly control the note sequence, in violation of objective 3.

Chapter 3

The Markov Program

3.1 INTRODUCTION

The *Markov* computer program is written in the Pascal programming language on an Apple Macintosh computer and was developed using the Symantec THINK Pascal (Version 4.0) development environment package (Symantec Corporation 1990)[1]. This package provides an integrated environment within which the programmer can create, edit, compile, link and run Pascal programs. There are on-line debugging tools to facilitate problem solving, and separate program files may be organised into a single project.

The program generates music by calculating note sequences according to the rules of the algorithm (see Chapter 2). These sequences are then sent to an external playback device, connected to the computer, using the MIDI standard (International MIDI Association 1988, Loy 1985). Specifically, the sequences are converted into individual MIDI **events** (play a middle C, for example) each of which consists of a short stream of bytes of data. These events are sent to the MIDI interface of the computer, in time-sequential order, from where they are passed to the playback device. My system consists of an Apple Macintosh LC computer connected to a Roland JV-30 synthesizer.

The communication between the *Markov* program and the MIDI interface is achieved using a set of MIDI command library routines provided by Altech Systems' MIDIPascal (Version 3.0) (Altech Systems 1990). These allow the program to send MIDI standard data to either the Apple Macintosh modem communications port or printer port. Routines are available which, for example, allow the size of the MIDI output buffer to be set (see Appendix A, Section **5**), and which transmit a MIDI event - each MIDI event is timestamped allowing precise control of the sequencing of musical events.

De Furia and Scacciaferro's book, **MIDI Programming for the Macintosh** (De Furia and Scacciaferro 1988), was an excellent source of advice and guidance in the use of MIDI programming techniques.

3.2 THE PROGRAM STRUCTURE

3.2.1 Introduction

The *Markov* program is divided into four main sections:-

The **Initialisation Section** sets up the MIDI interface and builds the program's pull down menus.

---

[1] Pascal was chosen simply because this is a language with which I am particularly familiar.

The **Composition Section** allows the composer to input the values of the various parameters via a series of input dialog boxes and stores this data in the Composition File (see Appendix A, Section **4**).
The **Playback Section** reads the data from the Composition File, applies the Diagonal Line algorithm to calculate the values of the musical parameters and sends the corresponding MIDI events to the MIDI interface, resulting in the live playback of the composition on the MIDI device(s) connected to the interface.
The **Termination Section** closes down the MIDI interface and terminates the program.
These four sections are now described in more detail.

3.2.2 The Initialisation Section

The Initialisation Section first calls a MIDIPascal routine to activate the MIDI output port and set the sizes, in bytes, of the input and output buffers[2]. Since the *Markov* program only transmits MIDI data and does not process incoming MIDI data, the size of the input buffer is irrelevant and is set here to a nominal value of 100 bytes. The output buffer size is set to 5000 bytes. In fact, an important feature of the *Markov* program is that it allows the composer to control the size of the output buffer (see Appendix A, Section **5**). However, MIDIPascal does not allow a programmer to change the output buffer size once set, so the *Markov* program provides the composer with size control by limiting the amount of data **stored** in the output buffer, while the size of the output buffer itself remains **fixed** at 5000 bytes[3] which therefore represents the **maximum** amount of data which can be stored in the output buffer.

Next, a series of Apple Macintosh system routines is invoked which create the various *Markov* program pull down menus:- the **File** Menu (see Appendix A, Section **4**), the **MIDI** Menu (see Appendix A, Section **5**) and the **Compose** Menu (see Appendix A, Section **3**). Actually, all these routines do is place the menu titles ("File", "MIDI" and "Compose") on the Apple Macintosh menu bar in the required order and add the corresponding pull down commands (for example, the Compose pull down menu has the commands

---

[2] Timestamped MIDI events are held in the output buffer in time-sequential order and are transmitted to the MIDI interface for playback when their event time arrives.
MIDI events **received** from the MIDI interface, generated from a keyboard for example, are held in the input buffer from where a program can process them.
The input and output buffers are managed by MIDIPascal.

[3] Although this figure was chosen arbitrarily, the intention is to balance the requirement to optimise the use of computer memory against the occasional need for the composer to store large amounts of data in the Output Buffer. 5000 should be large enough for most practical purposes. For the reasons why the program allows the composer to control the Output Buffer's size see Appendix A, Section 5.1

"Edit", "Section Sequence", "Tempo" and "Play"); the functionality behind these commands is implemented elsewhere in the program.
Also in this section, the Text window is drawn. This window is used to display information to the composer. For example, a summary of the contents of the Composition File (see Appendix A, Section **4.5**).

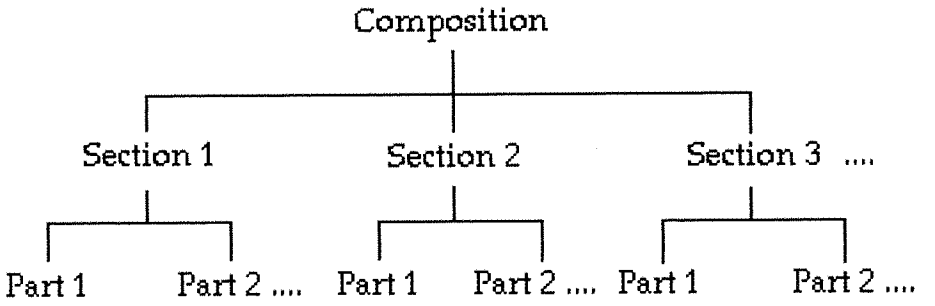
The program now waits for the composer to select a Menu Command, and processes it accordingly (for a full description of all the Menu Commands, see Appendix A, Sections **3,4** and **5**). Generally speaking, a composer will now either:

a) begin to create a new composition by selecting **Compose - Edit**
b) open an existing Composition File using **File - Open** and then modify it by selecting **Compose - Edit**
c) open an existing Composition File using **File - Open** and then play it back by selecting **Compose - Play**

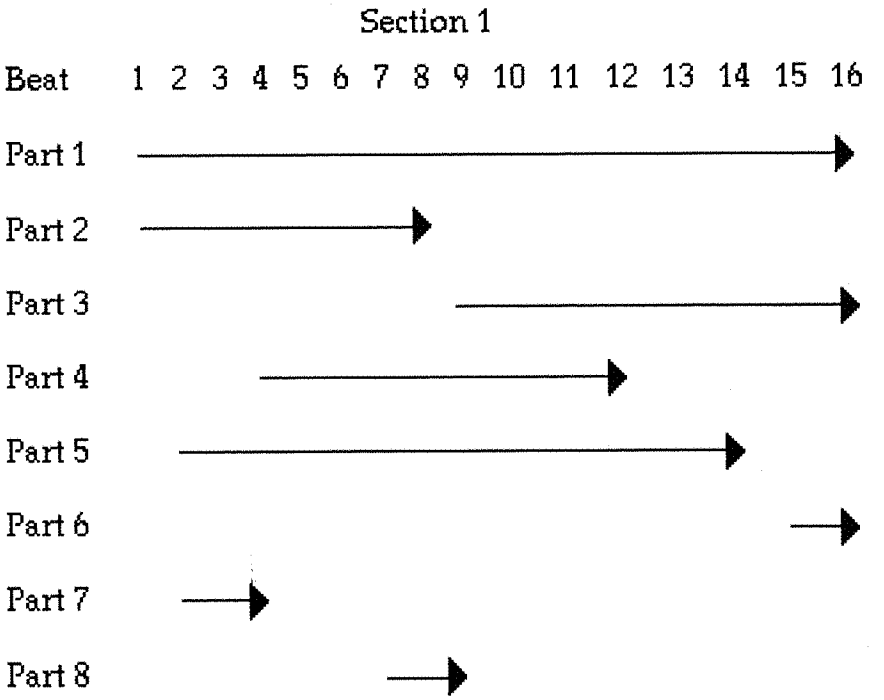
If a or b occurs, the program enters the Composition Section (see **3.2.3** below), while if c occurs the program immediately enters the Playback Section (see **3.2.4** below).

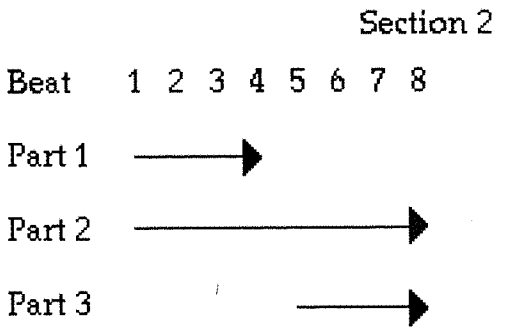
3.2.3 The Composition Section

The Composition Section provides the graphical user interface through which the composer enters, and may subsequently modify, the input data for all musical parameters (pitch, note length and so on), which the diagonal line method requires in order to produce the resulting musical output. A composition has a Section/Part structure, whereby a single, linear, note sequence is called a **Part** and one or more Parts may be grouped together to form a **Section**. A complete composition consists of a number of non-overlapping, sequentially played, Sections. The Parts within each section may overlap however, allowing polyphony to be achieved. Thus, a composition has the following structure:



The period of time over which a Part is to be played is specified by the beat on which it starts, counted from the beginning of its Section, and the beat on which it ends. Parts may overlap, for example:





The composer enters data for all the Sections which make up the composition, and for all the Parts within those Sections, by issuing pull down menu commands and then entering the data in the appropriate fields within the resulting series of input dialog boxes. The graphical user interface is described in full detail in the Markov Program User Guide (see Appendix A). The various dialog boxes themselves were designed and built using the Apple Macintosh Resource Editor, ResEdit (Version 2.1.1) (Alley and Strange 1994).

This section has two subsections:

(i) The Parameter Initialisation subsection, which sets the default values of all the various input data parameters, which the composer can subsequently change if desired (see Appendix A, Section **6**, for a full specification of the default values).

(ii) The Menu Command Processing loop which waits for the composer to select a menu command and then responds appropriately. The data entered by the composer in the resulting input dialog boxes is stored sequentially in the Composition File, Part by Part, Section by Section. This loop repeats continuously until either the composer selects **Compose - Play**, at which point the loop terminates and the Playback Section begins (see **3.2.4** below), or chooses to exit the program altogether, by selecting **File - Exit**, in which case the program passes directly to the Termination Section (see **3.2.5** below).

3.2.4 The Playback Section

This section begins by obtaining, from the Composition File, the Section Sequence specified by the user (see Appendix A, Section **3.7**), then calculates the MIDI clock speed using the composer-supplied tempo (in beats per minute) and calls a MIDIPascal routine to set the MIDI clock speed, which in turn determines the speed at which the composition is played back.

The core of the Playback Section is a main loop which processes one Section at a time, reading in the input data (from the Composition File) for all the Parts in that Section, calculating the musical parameters of the note sequence which will make up that Section and then sending the corresponding MIDI events to the MIDI interface. Each Section is processed as follows:-

1. Calculate the Parameters for the First Notes to be Played in Each Part

For each Part in the section, all the musical parameters (pitch, length, velocity, vibrato depth, vibrato rate, volume, pitch bend and release) for the first note in each of the Parts that make up the Section are calculated. If the composer has

explicitly set the starting value of a particular parameter (see Appendix A, Sections **3.3.2** to **3.3.9**) then that value is used and no actual calculation is necessary, otherwise the first value is generated at random from the possible range of values specified by the composer.

2. Map the Values of the First Notes to be Played

The composer is able to select the possible values of a note parameter (see Appendix A, Sections **3.3.2** to **3.3.4**). For example, the possible pitches could be specified as 60, 64, 67, 72[4] (a one octave C-major arpeggio). However, it would be a very inefficient use of disk space if each of these selected values was stored separately in the Composition File. Instead, before being written to the Composition File, the selected values are mapped to a **sequential** range of integers starting at 1 (for example, the above selection would be mapped to 1, 2, 3, 4) so only the **maximum** value of this range need be stored (since the range always starts at 1). The selected values themselves are stored in 8 separate 16 bit integers, each representing 16 values of the range 0-127 (since the MIDI values of all musical parameters lie in this range) as follows:-

integer 1	0 to 15
integer 2	16 to 31
integer 3	32 to 47
integer 4	48 to 63
integer 5	64 to 79
integer 6	80 to 95
integer 7	96 to 111
integer 8	112 to 127

with each bit of these integers set to either 1 or 0 depending on whether or not the corresponding value has, or has not, been selected. Thus, in the above example, integer 4 would have bit 13 set to 1, representing the value 60, with all other bits set to 0, while integer 5 would have bits 1, 4, and 9 set to 1, representing the values 64, 67 and 72, with all other bits set to 0:-

integer 4:-

Bit	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Value	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0
Param Val	48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63

[4] According to the MIDI standard, each pitch is given a unique number between 0 and 127. 60 corresponds to middle C.

integer 5:-

Bit	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
Value	1	0	0	1	0	0	0	0	1	0	0	0	0	0	0	0
Param Val	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79

Thus, when written as a binary number with the bit order reversed, integer 4 has the value 0001000000000000 = 4096, so integer 4 would be set to 4096, and integer 5 has the value 0000000100001001 = 265, so integer 5 would be set to 265, while integers 1, 2, 3, 6, 7 and 8 would be zero. This means that all the information relating to the selected values, of which there may be up to 128 different values, is stored in just 9 integers (the maximum value of the sequential range plus the 8 selection integers).

Before playback, the 9 integers stored must be mapped back to the actual parameter values they represent. The mapping operation performed here uses bit manipulation techniques to map the first value of a particular parameter value, which is generated from the **sequential** range, to its actual **selected** parameter value. For example, in the case described above, a first pitch of 2 would be mapped to the actual selected value of 64, while a first pitch of 4 would be mapped to the actual selected value of 72.

3. Calculate the Note On Event Times for the First Notes to be Played in Each Part

The times of occurrence of each of the first notes in each Part of the Section are calculated: the first note for each Part will occur either precisely at the beginning of the Part or after a random delay, depending on whether the composer has checked the Random Entry checkbox (see Appendix A, Section **3.3.10.1**). These times, together with the parameter values of the associated notes, are held in a special **MIDI Event Array**. At any time, the program only stores the data relevant to the **next** MIDI event, Note On or Note Off (Loy 1985: 13-15), in each Part. As soon as a MIDI event for a particular part has been output, all the data pertaining to the **next** MIDI event to occur in that Part are calculated and stored in the MIDI Event Array - if the event just output was a Note On, then the corresponding Note Off data is calculated, whereas if the event just output was a Note Off then the Note On data for the **next note** in that Part is calculated. Thus at any particular time the MIDI Event Array will be storing a mixture of Note On and Note Off event data, but only **one** event for each Part in the Section. The fact that the MIDI Event Array just stores the data for next event to occur in each Part, only calculating event data as it needs it, is key to the sequencing functionality of the *Markov* program.

4. Transfer the Note On Event for the First Note to be Played in the Section to the MIDI Output Buffer

Once the parameters for all of the first notes in each Part of the Section have been calculated, the times of occurrence of each of these notes are compared to determine which is the earliest; that is, which note is to occur **first** in the Section. The appropriate MIDIPascal routine calls are then made to send a Note On event of the required pitch and velocity for that note, plus the required settings for vibrato depth, vibrato rate, volume, pitch bend and release (if the user has changed these from the default settings), to the MIDI Output Buffer.

5. Calculate the Note Off Event Time for the First Note to be Played in the Section

As soon as the Note On event for the first note in the Section has been output, its length is used to calculate the time at which its Note Off Event is to occur. The data relevant to this Note Off event is stored in the MIDI Event Array.

6. Transfer the Next MIDI Event to the MIDI Output Buffer

The times of occurrence of each of the events currently held in the MIDI Event Array are compared to determine which is the next event to occur. The appropriate MIDIPascal routine calls are then made to send that event to the MIDI Output Buffer. As discussed above, as soon as that event has been sent, the data for the next event in the corresponding Part is calculated and stored in the MIDI Event Array. If this is a Note On Event for the **second or subsequent** notes in that Part, then the Diagonal Line Method now comes into play[5]. The way the *Markov* program handles this is described below.

7. Calculating the Parameters for the Second and Subsequent Notes to be Played in Each Part

As soon as a Note Off Event has been sent for a particular part, the **previous** values of each of the musical parameters (pitch, for example), relating to the most recent Note On Event for that Part, are used to calculate the **next** values of each of those parameters which will then form the data for the next Note On Event for that Part, as follows:

a) The Diagonal Line for the parameter in question is used to determine the mean, of the bilateral exponential distribution, for the next value of that parameter, corresponding to the previous value (see Section **2.2.3**).

---

[5] As discussed in detail in Section **2.2.3** , the Diagonal Line Method uses the **previous** value of a note parameter to calculate the **next** value. Thus is it not until the **first** parameter values have been calculated that the Diagonal Line method can now be applied to continue generating the sequence.

b) A random number is generated and the Monte Carlo method is used to generate the next value of that parameter (see Section **2.2.3**).

Steps a and b are repeated for all the musical parameters of the note (pitch, velocity, length and so on) and the resulting data is stored in the MIDI Event Array from where the corresponding MIDI events will subsequently be sent to the MIDI Output Buffer, once the time of occurrence of the note dictates that it is the next note to be played in this Section.

8. Continue Generating All the Notes in this Section

Steps 6 and 7 are continuously repeated until all the notes in this Section have been generated - this occurs as soon as a note is generated for which the time of occurrence of its Note Off Event is equal to, or later then, the Section end time (if the calculated time of occurrence of the Note Off Event is **later** than the end time of the Part then it is changed to be **equal** to the end time so that the Section lengths are exactly as specified by the composer).

The diagram in Figure 3.1 at the end of this chapter summarises the complete process for generating all the notes in a section. This diagram depicts the first 9 steps (labelled a to i) in the generation of the notes for a 3 part section. "t" refers to time in milliseconds. The arrowed lines in the diagram have the following meanings:-

↓ in the MIDI Event Array indicates that the associated event at this step has remained in the same position as at the previous step. For example, the Note 1 On Event for Part 1 remains in the same position in the MIDI Event Array at steps a) and b)

↓ in the MIDI Output Buffer indicates that the associated event at this step has remained in the same position in the MIDI Output Buffer as at the previous step. For example, the Note 1 Off Event for Part 1 remains in the same position in the MIDI Output Buffer at steps d) and e).

↘ in the MIDI Output Buffer indicates that the associated event at this step has moved forward one position in the MIDI Output Buffer from the previous step. For example, the Note 1 Off Event for Part 1 moves forward one position in the MIDI Output Buffer from step e) to step f).

The processes occurring in each of the 9 steps in the diagram are as follows:-

a) The parameter values and the Note On Event times for first note in each of the 3 Parts are generated and stored in the Midi Event Array. The parameter values consist of the pitch and velocity of each note. Pitch and Velocity values are passed in a single MIDI Event.
Here, it is assumed that the composer has chosen not to vary vibrato, volume, pitch bend or release as this would require the transmission of further MIDI Events and make the example difficult to follow.
The start times of the first notes in each of the 3 Parts are 500, 0 and 1000 milliseconds respectively. The first note in Part 2 is the first note to be played in the section, its start time, 0, being the lowest of the three, so its data is passed to the MIDI Output Buffer.
A start time of 0 means that the note is to be played right at the beginning of the Section so its MIDI Note On Event is sent to the MIDI Interface and the note begins to play (it will continue to play until the corresponding MIDI Note Off Event is transmitted).

b) The Note Off Event time for the first note in Part 2 is calculated and the relevant data replaces its Note On Event data in the MIDI Event Array. In this case the note is due to stop after 600 milliseconds.
The Note On Event for the first note in Part 1 will be the next event to occur, its event time being 500 milliseconds, so its data is passed to the MIDI Output Buffer. It will remain in the MIDI Output Buffer until 500 milliseconds have elapsed.

c) The Note Off Event time for the first note in Part 1 is calculated and the relevant data replaces its Note On Event data in the MIDI Event Array. In this case the note is due to stop after 1000 milliseconds.
The Note Off Event for the first note in Part 2 will be the next event to occur, its event time being 600 milliseconds, so its data is passed to the MIDI Output Buffer. There are now 2 events queued up in the MIDI Output Buffer, stored in the order in which they are to occur.
d) The Note On Event time for the second note in Part 2 is calculated and the relevant data replaces the Note Off Event data for first note in Part 2 in the MIDI Event Array. In this case the note is due to start after 2000 milliseconds.

The Note Off Event for the first note in Part 1 and the Note On Event for the first note in Part 3 are the next events to occur, at exactly the same time: 1000 milliseconds. In this situation the program sends the data for the lowest numbered Part, 1 in this case, to the MIDI Output Buffer first. However, when the two are events are subsequently transmitted, they will sent immediately one after the other, and the speed of the MIDI interface (31,250 bits per second) is such that the listener hears the events as being simultaneous. There are now 3 events queued up in the MIDI Output Buffer, stored in the order in which they are to occur.

e) The Note On Event time for the second note in Part 1 is calculated and the relevant data replaces the Note Off Event data for first note in Part 1 in the MIDI Event Array. In this case the note is due to start after 1500 milliseconds.
The data for the Note On Event for the first note in Part 3 is passed to the MIDI Output Buffer.
500 milliseconds have now elapsed so the MIDI Note On Event for the first note in Part 1 is sent to the MIDI Interface and the note begins to play.

f) The Note Off Event time for the first note in Part 3 is calculated and the relevant data replaces its Note On Event data in the MIDI Event Array. In this case the note is due to stop after 1600 milliseconds.
The data for the Note On Event for the second note in Part 1 is passed to the MIDI Output Buffer. Note that since an event was transmitted in step e), all the other events in the MIDI Output Buffer have moved forward one place in the queue.
600 milliseconds have now elapsed so the MIDI Note Off Event for the first note in Part 2 is sent to the MIDI Interface and the note ceases to play.

g) The Note Off Event time for the second note in Part 1 is calculated and the relevant data replaces its Note On Event data in the MIDI Event Array. In this case the note is due to stop after 1600 milliseconds. The data for this event is immediately passed to the MIDI Output Buffer: this event and the event for Part 3 are both due to occur at 1600 milliseconds so, as discussed previously, the data for the lower numbered Part is passed first.

1000 milliseconds have now elapsed so the MIDI Note Off Event for the first note in Part 1 is sent to the MIDI Interface and the note ceases to play.

h) The next event in the MIDI Output Buffer, the Note On Event for the first note in Part 3, is also due to occur at 1000 milliseconds so it is sent immediately to the MIDI interface and the note begins to play.
The Note On Event time for the third note in Part 1 is calculated and the relevant data replaces the Note Off data, for the second note, in the MIDI Event Array. In this case the note is due to start after 1600 milliseconds so once again the data for this event is immediately passed to the MIDI Output Buffer.

i) The Note Off Event time for the third note in Part 1 is calculated and the relevant data replaces its Note On Event data in the MIDI Event Array. In this case the note is due to stop after 1800 milliseconds.
The data for the Note Off Event for the first note in Part 3 is passed to the MIDI Output Buffer.

9. Continue Generating the Notes for All Sections
Steps 1 to 8 are repeated in turn for all the Sections that make up the Composition, as determined by the Section Sequence specified by the Composer (see Appendix A, Section 3.7).

10. Wait for the MIDI Output Buffer to Empty
It is an important feature of the program that it is always **calculating ahead**; that is, generating notes **in advance** of when they actually occur. The MIDI Events associated with each note are stored in the MIDI Output buffer. Each event is timestamped and is sent from the buffer to the MIDI interface once its time of occurrence arrives - the transfer of events from the buffer to the interface is managed by MIDIPascal. Once all the notes for the final Section of the composition have been sent to the MIDI Output Buffer, the program repeatedly calls a MIDIPascal function which indicates how much data is remaining in the buffer. When the amount of data reaches zero, the program knows that the final note of the composition has been played, and the Playback Section is therefore complete. What happens next depends on the action taken by the composer. If the composer chooses to modify the composition then the program will re-enter the Composition Section (see **3.2.3** above). If the

composer chooses to exit the program completely, then the Termination Section begins.

3.2.5 The Termination Section

The Termination Section first closes the Composition File that the composer has been working with, and then calls a MIDIPascal routine to shutdown MIDIPascal.

3.2.6 The "Score"

The **File - Export** command (see Appendix A, Section **4.6**) allows the composer to export a comprehensive specification of the entire contents of the Composition File to an ASCII text file for subsequent examination and printing. This forms the "score" for the composition; it describes the overall composition structure and gives full details of the algorithm input parameter values. The following is an example of the opening segment from a program score:

```
COMPOSITION FILE: Hard Disk:Music:Projects:Markov Composition:choir
================

Section  Parts  Min Length  Min Total  Max Length  Max Total
-------------------------------------------------------------
    1       2         100        100         100        100
    2       3          10        110         200        300
    3       1         300        410         300        600

SECTION SEQUENCE:
   1   1   2   1   3   1

Tempo =  60 bpm   MIDI Buffer Size =   1000 Bytes

SECTION:   1
============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
TRANSPOSE        0         0           0                1.000000  0.500000

Part No     MIDI Channel     Starting Beat    Ending Beat
-------------------------------------------------------
    1              1                  1            100
    2              1                 10             50

Sect  1 Part  1, Chan  1, Patch   1, Pan  64, BEATS      1 to      100
===============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           60        72          60               1.000000  2.500000
LENGTH (/  2)    1         2           1               1.000000  0.500000
```

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

The initial portion provides summary information: the name and location of the Composition File, the overall Section/Part structure of the composition, the sequence of Sections that forms the composition (see Appendix A, Section **3.7**), the tempo (see Appendix A, Section **3.6**) and the MIDI buffer size setting (see Appendix A, Section **5.2**). There then follows a Section by Section breakdown, each Section headed with the word "SECTION" followed by the Section number. The Section information begins with the Section sequence transposition settings (see Appendix A, Section **3.2.1.4**) and a summary of the Part structure, followed by the complete set of algorithm parameter values, and option settings, for each Part that makes up the Section, each Part under its own "Sect ... Part ..." heading.

Program score extracts will be given throughout the remainder of this thesis in support of musical examples under discussion.

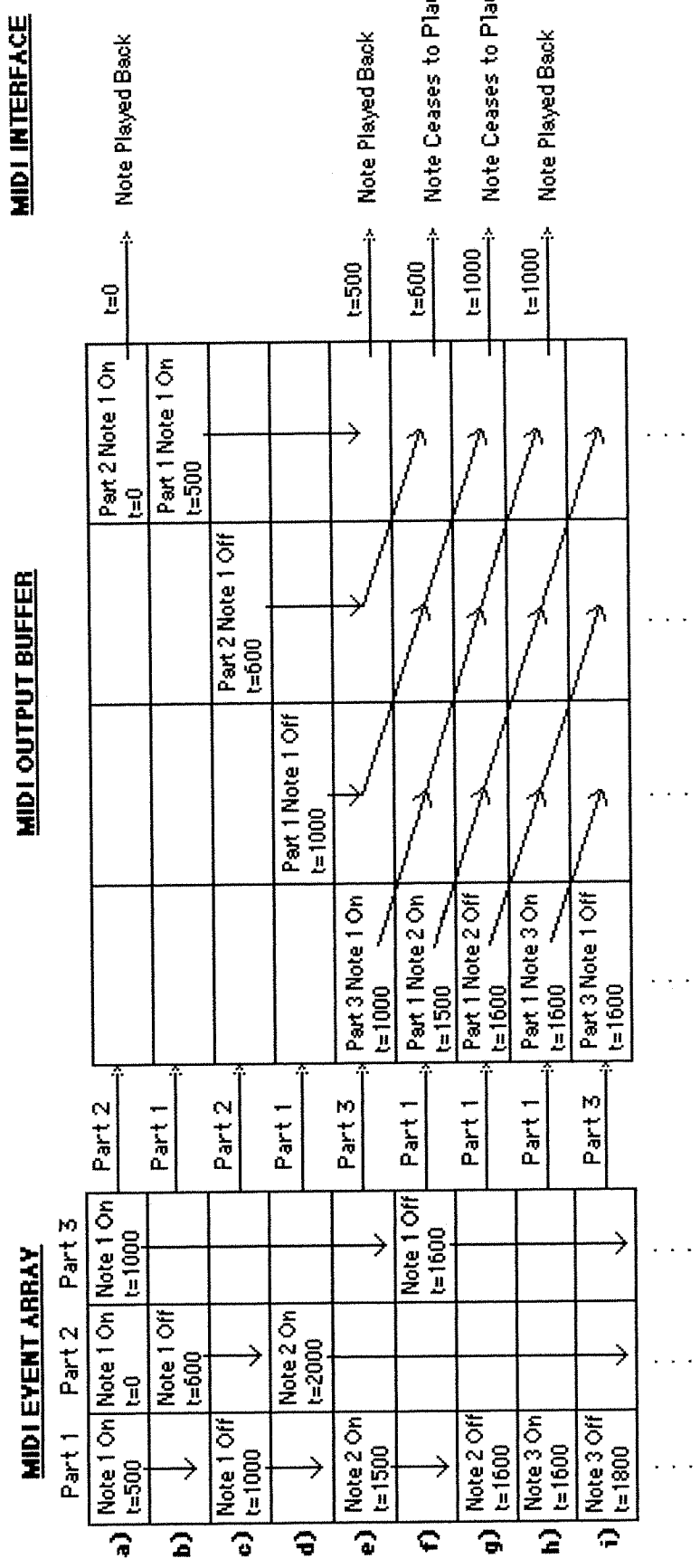


Figure 3.1 The Note Generation Process

Chapter 4

The Compositional Process

4.1 INTRODUCTION

The purpose of this chapter is to show how the compositional process, using the *Markov* computer program, actually works. One of the principal aims of the program is that the composer should become involved in a continual feedback process, trying out some initial parameter values and then refining them based on what he or she hears, so that the composition gradually converges to a desired result. This chapter shows the first 11 stages of an evolving composition, starting with a one line melody based on a simple set of parameter values and then gradually evolving into a three part piece, including a bass part, with a three chord harmonic structure and with rhythmic structure and control of melodic contour.

It is very important to realise that this particular composition is in no way representative of the style of music the program can produce. As will be shown later, the program can produce an extremely wide variety of styles and it just happens that the composition examined here evolved in this particular way. Nor is it supposed to be in any way a finished piece. The intention is to show the first few stages of a composer exploring the compositional capabilities of the program.

The compact disc which accompanies this thesis contains the composition at each of its 11 stages of evolution as MIDI files, for playback from any software application capable of playing standard MIDI files (Microsoft Windows™ Media Player for example). They are contained in the directory called "The Compositional Process" and there are 11 files in all, entitled "1.mid" through to "11.mid" corresponding to the 11 stages. The floppy disk contains them as *Markov* program Composition Files for playback from the *Markov* program. They are contained in the directory called "The Compositional Process" and are called "1" through to "11".

4.2 AN EVOLVING COMPOSITION

Stage 1

As a starting point, a two-octave chromatic scale starting at middle C is chosen. The gradient and λ values are left as the defaults of 1 and 0.5 respectively, so this should result in a fairly gently meandering melody (see Section **5.3.2**, Example 1). The tempo is set to 60 beats per minute and all note lengths are 1/8 of a beat. The Patch value is set to 1, which is the MIDI value for the Piano instrument. The Part length is 30 beats.

The complete score at this stage is as follows (note the Pitch Minimum, Maximum, Minimum Mean, Gradient and λ[1] values):

```
COMPOSITION FILE: Hard Disk:Music:Compositional Process:1
================

Section  Parts  Min Length  Min Total  Max Length  Max Total
-------------------------------------------------------------
    1       1         30         30          30         30

SECTION SEQUENCE:
   1

Tempo =  60 bpm   MIDI Buffer Size =    100 Bytes

SECTION:   1
============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
TRANSPOSE        0         0           0              1.000000  0.500000

Part No     MIDI Channel     Starting Beat    Ending Beat
-------------------------------------------------------
    1             1                  1             30

Sect  1 Part  1, Chan  1, Patch   1, Pan  64, BEATS      1 to      30
===============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           60        84          60              1.000000  0.500000
LENGTH (/  8)    1         1           1              1.000000  0.500000

            REPEAT  | SELECT  |  REVERSE  | REFLECT  | RANDOM ENTRY
            ------------------------------------------------------
PITCH         YES   |   NO    |    NO     |   NO     |
LENGTH        YES   |   NO    |    NO     |   NO     |      NO
VELOCITY      YES   |   NO    |    NO     |   NO     |
```

Stage 2

The composer decides that the melody should vary up and down less quickly. Increasing the λ value will achieve this. A λ value of 2 is tried.

Stage 3

Increasing the λ value has had the desired effect but the composer feels that the sequences of repeating notes, sometimes as many as 10, are disconcerting. The **Disallow Repeats** option is therefore selected for the Pitch Parameter (see Appendix A, Section **3.3.2**).

---

[1] These are shown under the heading "Lambda" in the score.

Stage 4

The use of the chromatic scale means that the melody has an atonal nature and the composer now decides to change the feel by using a major scale instead. The **Select** mechanism is therefore used (see Appendix A, Section **3.4**) to specify the MIDI pitch values 60 62 64 65 67 69 71 72 74 76 77 79 81 83 and 84 which represents a 2-octave C major scale starting at middle C.

The section of the score showing the relevant parameter values at this stage is as follows (note the Pitch λ value, the Pitch REPEAT and SELECT settings and the Pitch Selections):

```
Sect  1 Part  1, Chan  1, Patch   1, Pan  64, BEATS      1 to      30
===============
Parameter      Min       Max       MinMean    Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           60        84          60              1.000000  2.000000
LENGTH (/  8)    1         1           1              1.000000  0.500000

             REPEAT  | SELECT  |  REVERSE  | REFLECT  | RANDOM ENTRY
             ------------------------------------------------------
PITCH          NO    |   YES   |     NO    |    NO    |
LENGTH        YES    |    NO   |     NO    |    NO    |       NO
VELOCITY      YES    |    NO   |     NO    |    NO    |

PITCH SELECTIONS:
  60   62   64   65   67   69   71   72   74   76   77   79   81   83   84
```

Stage 5

The composer is now happier with the melodic aspect but, although repeated note patterns in the melody create perceived rhythmic units, there is currently no algorithmic control of rhythm because all the note lengths are the same. Therefore, the Minimum Note Length is now set to be 1/8 and the maximum to be 4/8. This will produce Note Lengths from all the values in that range; that is, 1/8, 2/8, 3/8 and 4/8 (demisemiquaver, semiquaver, dotted semiquaver and quaver).

Stage 6

The melody now has a sense of rhythm but the composer feels that the inclusion of the 3/8 Note Length (a dotted semiquaver) produces too much syncopation. The Select mechanism is therefore used to specify just the Note Lengths 1/8, 2/8 and 4/8 (demisemiquaver, semiquaver and quaver).

Here is the relevant section of the score (note the Length parameter settings[2] and the Length Selections):

```
Sect  1 Part  1, Chan  1, Patch   1, Pan  64, BEATS      1 to      30
===============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           60        84          60                1.000000  2.000000
LENGTH (/  8)    1         4           1                1.000000  0.500000

             REPEAT   | SELECT   |  REVERSE   | REFLECT   | RANDOM ENTRY
             ---------------------------------------------------------
PITCH          NO     |   YES    |     NO     |    NO     |
LENGTH        YES     |   YES    |     NO     |    NO     |       NO
VELOCITY      YES     |    NO    |     NO     |    NO     |

PITCH SELECTIONS:
  60  62  64  65  67  69  71  72  74  76  77  79  81  83  84
LENGTH SELECTIONS:
   1   2   4
```

Stage 7

The piece currently consists of just a solo melody and the composer would now like to add more interest by introducing a second melodic line. A second Part is therefore added with identical parameter values to the first one - this can be done very quickly using the Copy and Paste mechanism (see Appendix A, Sections **3.3.13** and **3.3.14**). The MIDI channel is then changed for the second Part (see Appendix A, Section **3.3.1**) and the Pan setting modified (see Appendix A, Section **3.3.12**) for both the first and second Parts to 24 and 104 respectively so as to produce left/right stereo separation.

Stage 8

A third, bass, Part is now added. The intention is for this to be a "walking" bass line. The composer selects the Pitch values 24 28 31 36 40 43 48, a two-octave C-major arpeggio, with repeats disallowed, and all Note Lengths are 1/4 beat (semiquaver). The Patch value is set to 33, the MIDI value for Acoustic Bass, and the Pan value to 64 so that the sound is centrally positioned. Finally, using the **Fix Starting Value At** option (see Appendix A, Section **3.3.2**), the starting value is set to be 24, a C natural, so that the bass line begins on the root of the scale and therefore feels "grounded".

---

[2] The "/ 8" which appears after the word "LENGTH" in the score indicates that the length values specified have 8 as their fractional demoninator. For example, the minimum length is 1/8.

Here is the section of the score showing the parameter values for this new Part:

```
Sect  1 Part  3, Chan  3, Patch  33, Pan  64, BEATS      1 to      30
==============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           24        48          24         24     1.000000  2.000000
LENGTH (/  4)    1         1           1                 1.000000  0.500000

             REPEAT   | SELECT   |  REVERSE   | REFLECT   | RANDOM ENTRY
             ------------------------------------------------------------
PITCH          NO     |   YES    |     NO     |    NO     |
LENGTH        YES     |    NO    |     NO     |    NO     |       NO
VELOCITY      YES     |    NO    |     NO     |    NO     |

PITCH SELECTIONS:
  24   28   31   36   40   43   48
```

Stage 9

The composition is now beginning to take shape but there is no sense of harmonic movement. The composer decides to repeat the Section 16 times, using the **Section Sequence** option (see Appendix A, Section 3.7) while using the **Section Transposition** mechanism (see Appendix A, Section 3.2.1.4) to produce chord changes between the successive playings of the Section. The Transposition values 0, 5 and 7 are selected, which correspond to tonic, subdominant and dominant respectively, so as to produce a sense of I - IV - V chord structure. The starting Transposition value is set to 0 so that the piece begins on the root chord. Repeats are disallowed so that there will always be a change of chord each time the Section repeats. The length of all three Parts, and therefore of the Section, is currently 30 beats, which the composer feels is too long to wait for each chord change so the **Section Length** Parameters (see Appendix A, Section 3.2.1.3) are set such that the length of each repetition of the Section is 4 beats only.

The following extract from the score shows the Section Sequence, the Transposition Parameters, the Length Parameters and the Transposition Selections:

```
COMPOSITION FILE: Hard Disk:Music:Compositional Process:9
================

Section   Parts   Min Length   Min Total   Max Length   Max Total
-----------------------------------------------------------------
    1       3          4           4           4           4

SECTION SEQUENCE:
   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1

Tempo =  60 bpm   MIDI Buffer Size =    100 Bytes
```

```
SECTION:   1
===========
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
TRANSPOSE        0         7           0          0     1.000000  0.500000
LENGTH           4         4           4                1.000000  0.500000
SECTION TRANSPOSITION SELECTIONS:
   0   5   7
```

Stage 10

The parameter values for the two main melodic Parts are identical so their structural characteristics are the same, and the composer feels it would be more interesting if they were to move in different ways. In addition, the fact that it is possible for the two Parts to be in the same register for a short while produces a disconcertingly large number of discords. The Minimum Mean Pitch value of the first Part is therefore set to 62, producing a tendency for the melody of this Part to move upwards (see Section **5.4.2**, Examples 1 and 2), and the Minimum Mean Pitch value of the second Part to 84, producing a tendency for downwards movement (see Section **5.4.2**, Examples 11 and 12). The λ Pitch value for each Part is set to 2 so that the movement tendency is quite strong while still allowing some variation (see Section **5.2.2**, Example 2).

The following extracts from the score show the Pitch parameter values for the two Parts:

```
Sect  1 Part  1, Chan  1, Patch   1, Pan  24, BEATS      1 to      30
===============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           60        84          62         60     1.000000  2.000000

Sect  1 Part  2, Chan  2, Patch   1, Pan 104, BEATS      1 to      30
===============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           60        84          84         84     1.000000  2.000000
```

Stage 11

All three Parts consist of continuous note sequences, with no rests. The composer now decides that rests should be introduced into each of the Parts. This is done by allowing notes of zero Velocity to be generated. If the program generates a note of a certain length at zero Velocity then a rest will occur of that length. The Select mechanism is therefore used to specify that only the Velocities 0 and 127 (the maximum) can occur. Rests should not, however, occur so frequently that the flow of the melodic lines is broken so a Minimum

Mean Velocity of 127 is chosen, with a gradient of zero. This means that the Velocity values will tend to be 127 and the frequency with which rests occur can be controlled with the λ value: the higher the value λ the less frequently rests will occur. Here, the composer tries a λ value of 2.

Finally, the complete score for the composition is now as follows:-

```
COMPOSITION FILE: Hard Disk:Music:Compositional Process:11
================

Section  Parts  Min Length  Min Total  Max Length  Max Total
-------------------------------------------------------------
   1       3         4          4           4          4

SECTION SEQUENCE:
   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1   1

Tempo = 200 bpm   MIDI Buffer Size =    100 Bytes

SECTION:   1
============
Parameter      Min      Max      MinMean    Start     Grad      Lambda
-------------------------------------------------------------------------
TRANSPOSE        0        7          0         0    1.000000  0.500000
LENGTH           4        4          4              1.000000  0.500000

SECTION TRANSPOSITION SELECTIONS:
   0   5   7

Part No     MIDI Channel     Starting Beat    Ending Beat
-----------------------------------------------------------
   1             1                 1              30
   2             2                 1              30
   3             3                 1              30

Sect  1 Part  1, Chan  1, Patch   1, Pan  24, BEATS      1 to      30
===============
Parameter      Min      Max      MinMean    Start     Grad      Lambda
-------------------------------------------------------------------------
PITCH           60       84         62        60    1.000000 20.000000
LENGTH (/  8)    1        4          1              1.000000  0.500000
VELOCITY         0      127        127              0.000000  2.000000

            REPEAT  | SELECT  |  REVERSE  | REFLECT  | RANDOM ENTRY
            ------------------------------------------------------
PITCH         NO    |  YES    |     NO    |    NO    |
LENGTH        YES   |  YES    |     NO    |    NO    |       NO
VELOCITY      YES   |  YES    |     NO    |    NO    |

PITCH SELECTIONS:
  60  62  64  65  67  69  71  72  74  76  77  79  81  83  84
LENGTH SELECTIONS:
   1   2   4
VELOCITY SELECTIONS:
   0 127

Sect  1 Part  2, Chan  2, Patch   1, Pan 104, BEATS      1 to      30
===============
Parameter      Min      Max      MinMean    Start     Grad      Lambda
-------------------------------------------------------------------------
PITCH           60       84         84        84    1.000000 20.000000
LENGTH (/  8)    1        4          1              1.000000  0.500000
VELOCITY         0      127        127              0.000000  2.000000
```

```
            REPEAT  | SELECT  |  REVERSE  | REFLECT  | RANDOM ENTRY
            ------------------------------------------------------
PITCH         NO    |   YES   |     NO    |    NO    |
LENGTH        YES   |   YES   |     NO    |    NO    |       NO
VELOCITY      YES   |   YES   |     NO    |    NO    |

PITCH SELECTIONS:
  60  62  64  65  67  69  71  72  74  76  77  79  81  83  84
LENGTH SELECTIONS:
   1   2   4
VELOCITY SELECTIONS:
   0 127

Sect  1 Part  3, Chan  3, Patch  33, Pan  64, BEATS      1 to      30
===============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           24        48         24         24     1.000000  2.000000
LENGTH (/  4)    1         1          1                 1.000000  0.500000
VELOCITY         0       127        127                0.000000  2.000000

            REPEAT  | SELECT  |  REVERSE  | REFLECT  | RANDOM ENTRY
            ------------------------------------------------------
PITCH         NO    |   YES   |     NO    |    NO    |
LENGTH        YES   |    NO   |     NO    |    NO    |       NO
VELOCITY      YES   |   YES   |     NO    |    NO    |

PITCH SELECTIONS:
  24  28  31  36  40  43  48
VELOCITY SELECTIONS:
   0 127
```

Chapter 5

Analysis of the Algorithm

5.1 INTRODUCTION

This analysis investigates the effects of the various algorithm parameters on the musical results produced. Specifically, each parameter is successively varied while the others are kept constant, and the various results are compared so that conclusions can be drawn regarding the effect of that parameter. The objective is that an understanding of the range of musical possibilities of the algorithm can be gained.

For each parameter value example, a short passage of music has been recorded in a MIDI file. These MIDI files may be found on the compact disc which accompanies this thesis in the directory called "Analysis of the Algorithm". The floppy disk contains the examples as *Markov* program Composition Files for playback from the *Markov* program, again in the directory called "Analysis of the Algorithm". Full details of the filenames and locations are given later in this chapter, in the relevant sections.

Before MIDI recording, each passage was first played back by the *Markov* program five times to ensure that, across each of the five different realisations, there was no difference in the overall musical nature of the result. The passage was then played back a sixth time and the result recorded. In no case was there any significant variation across the six realisations - indeed, the various parameter values were chosen to try and ensure that this would be the case, so that valid conclusions could be drawn about the effect of the parameter under analysis.

Pitch values are referred to using the "Cn" notation:-

C5	=	middle C	= MIDI value 60
C6	=	C one octave above middle C	= MIDI value 72
C4	=	C one octave below middle C	= MIDI value 48
D5	=	D above middle C	= MIDI value 62
B4	=	B below middle C	= MIDI value 59

and so on.

5.2 VARYING λ

5.2.1 Introduction

The purpose of this section is to investigate the effect of varying the parameter λ (see Section **2.2.1**) while all other parameter values are kept constant. Specifically, the fixed parameter values are as follows:-

Pitch values:	a four-octave C major scale centred on middle C (that is, C3 to C7)
Starting Pitch:	60 (= middle C, or C5)
Tempo:	60 beats per minute
Note Length:	1/8 (that is, 8 notes per beat, or 8 notes per second in this case)
Gradient:	0
Minimum Mean:	60 (= middle C, or C5)
Passage Length:	32 beats (32 seconds)

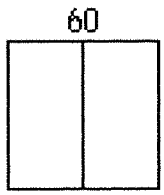
The range of pitch values was chosen as being wide enough for the full effect of the parameter to be appreciated, but narrow enough to discourage a diversity of unusual events from occurring more frequently than the event which is normal for the given parameter value. Even though the normal event would still be the largest single category, the perception of this would be obscured by using too large a range: instead the intention is achieve an appropriate balance between the ordinary and the unusual so that correct conclusions can be drawn from those unusual progressions which do occur.

A major scale (C major in this case) was chosen so that any tendency for the melody to move in a particular way is more readily apparent to a tonally oriented listener.

The note length and tempo was chosen so that the melody is fast enough that the relationship between successive notes, together with any higher level structure, can easily be heard and not so fast that it just becomes a blur of notes.

Each passage starts at middle C to provide a consistent beginning in an attempt to prevent inappropriate conclusions being drawn from different passages starting at widely differing pitches.

Finally, a Gradient of 0 and a Minimum Mean of 60 will produce a melody which **tends** to be centred evenly about middle C - the corresponding diagonal line is as follows:



- and we can therefore deduce the effect of different λ values by studying the degree by which the melody varies from middle C. What we expect is that the higher the value of λ the more the melody will tend to consist entirely of middle C pitches, while the lower the value of λ the more spread around

middle C the pitches will become until eventually all sense of centredness disappears.

The files on the accompanying compact disc and floppy disk which support the examples in the following analysis may be found in the directory called "Analysis of the Algorithm", subdirectory "Varying Lambda". The files are named numerically corresponding to the numbered examples below.

5.2.2 Analysis

Varying λ, Example 1

$\lambda = 5.$

The melody consists of successive repetitions of middle C with occasional variations of one pitch up or down (in this example there were 13 such variations). We can therefore deduce that a λ value of 5 produces a very strong tendency for the melody to behave according to the associated diagonal line. In this example, the result is very uninteresting musically although there may be situations where this is the desired effect; for example, for a backing to a melody or where the other diagonal line parameters produce a melody with a tendency to move in a specific way (see Section **5.4** below). Note that while this λ value produces a very strong tendency, for a sufficiently high λ value the melody will be completely deterministic with no variation from the predicted behaviour (see **5.2.3** below).

Varying λ, Example 2

$\lambda = 2.$

The melody is still very strongly centred on, with frequent repetitions of, middle C but with much more frequent variations up and down and by one or two pitches; that is, the melody consists entirely of the pitches A4, B4, C5, D5 and E5. C5 is the by far the most frequent, B4 and D5 the next most frequent, and there were 7 A4's and 5 E5's out of a total of 256 notes.

Varying λ, Example 3

$\lambda = 1.$

The melody is still very strongly centred on, with frequent repetitions of, middle C but the degree of movement away from middle C to other parts of the scale very occasionally implies harmonic movement: this is because more arpeggiation of notes occurs, thus occasionally implying particular chords. For example, a short sequence of B3 D4 F4 implies a Dominant 7th chord which is then followed by a return to the root chord implied by repetitions of middle C. Apart from the occurrence of one C below middle C (C4) and one A above middle C (A5), the melody lies in the range F below middle C (F4) to F above middle C (F5).

Varying λ, Example 4

$\lambda = 0.5$.

At this value of λ, there are no longer frequent successive repetitions of middle C so the feeling of "centredness" is not as strong: instead, the absence of repeated middle C's combined with a preponderance of pitches around middle C creates a tension in which the listener desires a **return** to middle C. There is also more note arpeggiation, resulting in a greater sense of implied harmonic movement. The melody ranges almost entirely over the two octaves centred on middle C. The occurrence of short sequences of pitches in different registers is beginning to produce rhythmic effects.

Varying λ, Example 5

$\lambda = 0.2$.

The feeling of "centredness" about middle C has all but disappeared and indeed pitches occur across the full four-octave range, but the pitches are in fact still centred on middle C (middle C still accounts for 10% of the pitch values which occur even though there are 29 possible pitch values) and indeed the activity in the middle C register can be heard as an independent melodic part, with the occurrence of low and high pitches infrequent so that the high and low pitches actually stand out as separate melodic parts from the main melody by virtue of their intervallic distance from it. This separation of parts, and the irregularity with which notes occur in those parts, is now more clearly resulting in rhythms which break up what was, in Examples 1 to 3, a consistent pulse.

Varying λ, Example 6

$\lambda = 0.1$.

The centredness about middle C is no longer apparent, but it is still there as a study of the frequencies of occurrence of each of the 29 pitches reveals:

C3 2 (1%)	**C4** 9 (4%)	**C5** 17 (7%)	**C6** 6 (2%)
D3 5 (2%)	**D4** 4 (2%)	**D5** 12 (5%)	**D6** 13 (5%)
E3 2 (1%)	**E4** 11 (4%)	**E5** 17 (7%)	**E6** 10 (4%)
F3 3 (1%)	**F4** 11 (4%)	**F5** 16 (6%)	**F6** 10 (4%)
G3 5 (2%)	**G4** 7 (3%)	**G5** 3 (1%)	**G6** 6 (2%)
A3 11 (4%)	**A4** 17 (7%)	**A5** 12 (5%)	**A6** 2 (1%)
B3 8 (3%)	**B4** 17 (7%)	**B5** 9 (4%)	**B6** 4 (2%)
			C7 0 (0%)

Note the higher frequencies of occurrence of the pitches around middle C (C5), in the range A4 to F5 for example. The high and low pitches stand out as separate melodic parts just as in the previous example but because there is more variation from middle C these parts contain more notes and so sound faster. The notes in this example are noticeably more arpeggiated and so there is an even greater impression of harmonic movement.

Varying λ, Example 7

$\lambda = 0.05$.

The frequencies of occurrence of the pitches in this example are as follows:

C3 3 (1%)	**C4** 10 (4%)	**C5** 14 (2%)	**C6** 8 (3%)
D3 5 (2%)	**D4** 8 (3%)	**D5** 10 (4%)	**D6** 5 (2%)
E3 5 (2%)	**E4** 8 (3%)	**E5** 9 (4%)	**E6** 5 (2%)
F3 8 (3%)	**F4** 13 (5%)	**F5** 12 (5%)	**F6** 9 (4%)
G3 10 (4%)	**G4** 11 (4%)	**G5** 16 (6%)	**G6** 14 (5%)
A3 6 (2%)	**A4** 6 (2%)	**A5** 10 (4%)	**A6** 7 (3%)
B3 10 (4%)	**B4** 13 (5%)	**B5** 14 (5%)	**B6** 5 (2%)
			C7 2 (1%)

Although the pitches around middle C still occur more frequently this is not nearly as marked as in the previous example. This passage does not sound that much different to the previous one except that there is possibly a slightly clearer perception of a middle part, together with high and low part, due to a more balanced spread of pitches across the range.

Varying λ, Example 8

$\lambda = 0.005$.

The frequencies of occurrence in this example are as follows:

C3 8 (3%)	**C4** 8 (3%)	**C5** 9 (4%)	**C6** 3 (1%)
D3 6 (2%)	**D4** 6 (2%)	**D5** 13 (5%)	**D6** 10 (4%)
E3 7 (3%)	**E4** 16 (6%)	**E5** 12 (5%)	**E6** 7 (3%)
F3 9 (4%)	**F4** 10 (4%)	**F5** 8 (3%)	**F6** 5 (2%)
G3 11 (4%)	**G4** 10 (4%)	**G5** 12 (5%)	**G6** 6 (2%)
A3 5 (2%)	**A4** 11 (4%)	**A5** 10 (4%)	**A6** 13 (5%)
B3 10 (4%)	**B4** 11 (4%)	**B5** 8 (3%)	**B6** 8 (3%)
			C7 4 (2%)

Now the frequencies of occurrence are fairly evenly spread across the range and are occurring more or less at random (the fact that some pitches have quite high occurrences - E4 and A6 for example - is due to random variation: it is more likely that this will happen than that the frequencies will be perfectly balanced across the range).
However, this does not sound markedly different to the previous two examples: one still hears three separate parts, in the low, middle and high part of the range, and it must therefore be deduced that this is how pitches occurring at random within a four-octave major scale tend to be heard. The most likely explanation for this is that within a purely random sequence, apparently ordered subsequences will nevertheless occur (Bennett 1998: 167-170). Thus, for example, several consecutive pitches may occur in the same register before the pitches move to a different register, resulting in the perception of separate parts.

Varying λ, Example 9

Four consecutive parts, $\lambda = 0.005$, $\lambda = 0.0005$, $\lambda = 0.00005$ and $\lambda = 0.000005$ respectively, of 8 beats each.

This final example is included to show that there is no audible difference when λ is raised above 0.005 - the four parts join together seamlessly.

5.2.3 Additional Information

Although not included on the compact disc, passages were produced with $\lambda = 20$ and $\lambda = 10$. A five minute passage with $\lambda = 20$ produced no variations at all; that is, continuously repeated middle C's. A one minute passage with $\lambda = 10$ produced continuously repeated middle C's apart from six notes which varied up or down by one pitch.

5.2.4 Conclusions

1. The higher the value of λ the greater is the tendency for the musical result to deterministically obey that predicted by the diagonal line (in this case continuously repeated middle C's), the lower the value of λ of the more the result will tend to vary from that predicted by the diagonal line.

2. A λ value of 20 produces a completely deterministic result, according to the diagonal line.

3. λ values greater than 1 produce strongly deterministic results, the degree of determinism increasing, of course, the higher the value of λ.

4. λ values less than 0.005 produce random results, with no diagonal line influence.

5. A λ value of 0.5 tends to produce a "middle ground" result, where the diagonal line influence is still evident but there are some noticeable variations from the predicted behaviour.

It should be noted that the effect of λ on the nature of the output is not linear. For example, increasing λ by 0.5 from 0.5 to 1 significantly increases the degree of determinism in the output whereas increasing λ by 0.5 from 5 to 5.5 has a barely discernible effect. In fact, λ is a **logarithmically** scaled parameter: λ must be **doubled** from 5 to 10 to achieve the same impact as doubling λ from 0.5 to 1.0. On the other hand, as noted above, this logarithmic scaling only applies within the range 0.005 to 20.

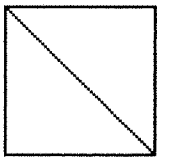
5.3 VARYING GRADIENT

5.3.1 Introduction

The purpose of this section is to investigate the effect of varying the Gradient (see Section **2.2.3**) parameter while all other parameter values are kept constant. For the first five examples, the fixed parameter values are as follows:

Pitch values:	a four-octave C major scale centred on middle C (that is, C3 to C7)
Starting Pitch:	60 (= C5, the mid-point of the pitch range)
λ:	0.5
Tempo:	60 beats per minute
Note Length:	1/8 (that is, 8 notes per beat, or 8 notes per second in this case)
Minimum Mean:	36 (= C3, the lowest pitch of the range)
Passage Length:	32 beats (32 seconds)

Having a Minimum Mean of 36, the lowest pitch of the range, means a diagonal line starting at the top left of the rectangle and sloping downwards - for a gradient of 1, for example, the diagonal line is as follows:



In the first five examples of this analysis, the gradient will be successively lowered. A λ value of 0.5 has been chosen because, as we saw in the previous chapter, this results in the diagonal line effect being evident while still allowing some variation from it.

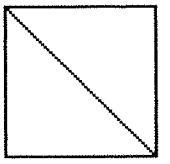
The files on the accompanying compact disc and floppy disk which support the examples in the following analysis may be found in the directory called "Analysis of the Algorithm", subdirectory "Varying Gradient". The files are named numerically corresponding to the numbered examples below.

5.3.2 Analysis

Varying Gradient, Example 1

Gradient = 1.

The diagonal line is as follows:

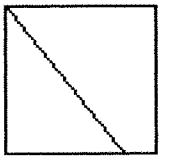


Here, the tendency is for any pitch to repeat itself, but the λ value of 0.5 allows a degree of variation and so the result is a melody which meanders fairly gently around the four-octave pitch range.

Varying Gradient, Example 2

Gradient = 0.75.

The diagonal line is as follows:

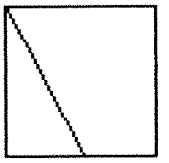


Here, any pitch, apart from the lowest pitch of the range, tends to be followed by a next pitch of a lower value, so the melody is being continuously, but in this example fairly gently, "pulled down" towards the bottom of the pitch range as if on a piece of elastic, with a tendency for any rise in pitch to be followed by a falling melodic part. Note that the amount by which the next pitch tends to be lower is proportional to the previous pitch rather than being an absolute number of pitch values. In this example, a pitch value of the maximum in the range tends to be followed by a pitch value 25% of the range lower, one octave in this case. This percentage value falls proportionately as the pitch lowers: a pitch at the middle of the range tends to fall by 12% of the range and the lowest value in the range tends to be followed by a pitch of the same value. Almost the entire melody lies in the first two octaves of the range.

Varying Gradient, Example 3

Gradient = 0.5.

The diagonal line is as follows:

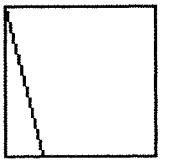


The "pulling down" effect has become more marked. Almost the entire melody lies in the first octave of the range.

Varying Gradient, Example 4

Gradient = 0.25.

The diagonal line is as follows:

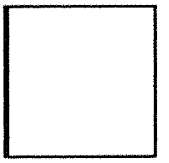


Now, almost the entire melody lies in the range C3 to G3, with any higher pitches being heard as a separate part.

Varying Gradient, Example 5

Gradient = 0.

The diagonal line is as follows:



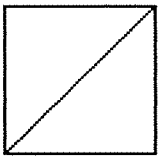
that it is to say, a vertical line at the minimum value of the range. This example is very similar to the previous example. Now, most of the melody lies in the range C3 to F3, and again higher pitches are heard as a separate part.

For the next five examples, since the gradient is negative the Minimum Mean is set to 84 (= C7, the highest pitch in the range). The other parameter values remain unchanged.

Varying Gradient, Example 6

Gradient = -1.

The diagonal line is as follows:

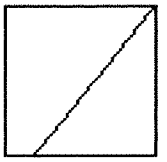


In this case, pitches towards one extreme of the pitch range tend to be followed by pitches at the other extreme, while pitches towards the centre of the range tend to be followed by pitches also towards the centre. Thus, the melody consists of periods of leaping from high pitches to low pitches and back again interspersed with periods of more gently varying pitch sequences towards the centre of the range (in fact, any pitch tends to be followed by its **complement**. For example, a pitch one quarter of the way up the range from the lowest pitch tends to be followed by a pitch one quarter of the way down from the highest pitch) and now there is a part separation effect. Overall, the melody could be said to be "balanced" about middle C, the centre of the range.

Varying Gradient, Example 7

Gradient = -0.75.

The diagonal line is as follows:

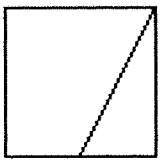


There is the same tendency, as in the previous example, for periods of leaping from high to low interspersed with more level periods, but now the melody is concentrated mainly in the upper 3 octaves. Also, the balance point has moved up. In fact, the diagonal line is such that D5 tends to be followed by F5, and F5 tends to be followed by D5 so it is about this part of the range that the melody is balanced and in this part of the range where the more level periods tend to occur.

Varying Gradient, Example 8

Gradient = -0.5.

The diagonal line is as follows:

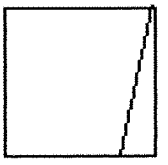


The melody is now concentrated mainly in the upper 2 octaves. Since the distance between the high and low pitches is therefore smaller, the leaping periods, although they still occur, are less pronounced. The melody is now balanced about A5 (in fact, the diagonal line is such that A5 tends to be followed by A5).

Varying Gradient, Example 9

Gradient = -0.25.

The diagonal line is as follows:

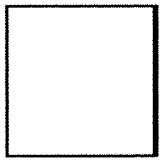


The melody is now concentrated mainly in the upper octave. The leaping periods are therefore even less pronounced, but are there. Since D6 tends to be followed by E6, and E6 by D6, it is about these pitches that the melody is balanced.

Varying Gradient, Example 10

Gradient = 0.

The diagonal line is as follows:



that it is to say, a vertical line at the maximum value of the range. Now there is a tendency for the melody to be drawn to C7, the highest pitch in the range. This is effectively the inverse of Example 5 but here, although there are a number of notes that are lower in pitch than most of the melody, at these high pitches they do not tend to be heard as a separate melodic part to the same extent as Example 5. This is probably because these pitches are too high to constitute what we normally think of as a bass line. This is borne out by the following example, which is the same as the current example but transposed down by 2 octaves:

Varying Gradient, Example 11

Gradient = 0 , Pitch values transposed down by 2 octaves.

This passage has exactly the same characteristics as the previous example but now the lower pitches tend to be heard more clearly as a separate melodic part.

5.3.3 Conclusions

With a Minimum Mean equal to the lowest pitch in the range and a λ value of 0.5:

1. A gradient of 1 produces a melody which meanders about the entire pitch range.

2. A positive gradient of less than one produces a melody which is "pulled down" towards the lower end of the range, with higher pitches being followed by falling melodic parts. This effect becomes more pronounced as the gradient is reduced.

3. For a gradient of 0.25 or less, the melody is highly concentrated in the lower pitches, with higher pitches being heard as a separate melodic part.

With a Minimum Mean equal to the highest pitch in the range and a λ value of 0.5:

1. A gradient of -1 produces a melody containing periods of leaping alternately from high to low pitches and back, interspersed with more stable periods where the pitches lie towards the centre of the range.

2. As the gradient increases from -1 towards zero the leaping effect becomes less pronounced as the pitches become more and more concentrated in the upper pitches of the range.

3. For a gradient of 0, most pitches lie close to the top of the range with occasional lower pitches occurring.

5.4 VARYING MINIMUM MEAN

5.4.1 Introduction

The purpose of this section is to investigate the effect of varying the Minimum Mean parameter (see Section **2.2.3**) while all other parameter values are kept constant. The fixed parameter values are as follows:

Pitch values:	a four-octave C major scale centred on middle C (that is, C3 to C7)
Starting Pitch:	36 (= C3, the lowest pitch of the range)
Tempo:	60 beats per minute
Gradient:	1
Note Length:	1/8 (that is, 8 notes per beat, or 8 notes per second in this case)
Passage Length:	32 beats (32 seconds)
Wraparound:	ON
Reverse:	OFF

As far as the λ parameter is concerned, for each Minimum Mean there are **two** example passages, one with $\lambda = 20$ and one with $\lambda = 0.5$. This is because varying the Minimum Mean produces melodies with a definite tendency to move in a certain way and so hearing the passage first with $\lambda = 20$, which produces completely deterministic results (see Section **5.2.3** above), provides a better control example against which the $\lambda = 0.5$ divergence can be appreciated. The $\lambda = 20$ examples are only 16 beats long, rather than 32, because the deterministic nature of the passage is quickly discernible.

The files on the accompanying compact disc and floppy disk which support the examples in the following analysis may be found in the directory

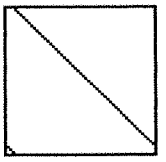
called "Analysis of the Algorithm", subdirectory "Varying Minimum Mean". The files are named numerically corresponding to the numbered examples below.

5.4.2 Analysis

Varying Minimum Mean, Example 1

Minimum Mean = 38, $\lambda = 20$.

The diagonal line is as follows:



(note that the diagonal line re-emerges towards the bottom left-hand corner - this is because the Wraparound[1] effect is turned on).

This example shows the precise rising tendency in Example 2 that follows. The melody is simply repeating four-octave C major scales. Note that the Minimum Mean of 38 is the **second** pitch of the range (D3); that is, **one pitch above** the first pitch of the range.

Varying Minimum Mean, Example 2

Minimum Mean = 38, $\lambda = 0.5$.

While there are some descending sequences of pitches the overall trend is for the melody to rise. When the melody reaches the top of the range (C7), any further attempt to rise results in the melody returning immediately to the bottom of the range - this is a result of the line wrapping around together with the Reverse effect (see Section **2.3.4**) being turned off. The melody therefore consists of repeated rising sequences. At this value of λ, the length of each rising sequence varies considerably - the shortest is 30 notes in length while the longest is 70.

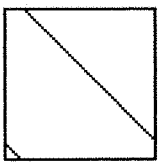
---

[1] See Section **2.3.2.**

Varying Minimum Mean, Example 3

Minimum Mean = 40, $\lambda = 20$.

The diagonal line is as follows:



This example shows the precise rising tendency in Example 4 that follows. The melody consists of the **alternate** pitches of the C major scale; that is, the 1st, 3rd, 5th and so on. Note that the Minimum Mean of 40 is the 3rd pitch of the range; that is, **two pitches above** the first pitch of the range. In fact, the repeated rising sequences alternate between sequences beginning on C3, the first pitch of the range, and sequences beginning on D3, the second pitch of the range. This is because there are an odd number of pitches in the range, so a rising sequence beginning on C3 ends on C7, the top pitch of the range, which is then followed by D3. Had there been an even number of pitches in the range, then each rising sequence would consist of pitches 1, 3, 5 and so on, while the even numbered pitches (2, 4, 6 and so on) would not be heard (since the starting note has been set to the lowest pitch in the range).

Varying Minimum Mean, Example 4

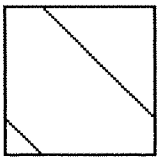
Minimum Mean = 40, $\lambda = 0.5$.

The melody again consists of repeated rising sequences but they are much shorter than in Example 2; that is, the melody rises more quickly. The lengths of the repeating sequences are still subject to wide variation, the shortest being eight notes in length and the longest 28.

Varying Minimum Mean, Example 5

Minimum Mean = 48, $\lambda = 20$.

The diagonal line is as follows:



This example shows the precise tendency in Example 6 that follows. The Minimum Mean of 48 is seven pitches, in the C major scale, above the first pitch in the range, and the first rising sequence consists of pitches 1 8 15 22 29 of the four-octave C major scale; that is, each successive pitch is seven above the previous pitch. The highest pitch in the range, pitch 29, is now followed by pitches 7 14 21 28. This is then followed by 6 13 20 27, 5 12 19 26, 4 11 18 25, 3 10 17 24, 2 9 16 23, before returning again to the beginning of the range. after which the sequence repeats. Thus, within the overall sequence are four separate **falling** C major scales and this is how the overall sequence is heard even though it is constructed from four or five note **rising** sequences.

Varying Minimum Mean, Example 6

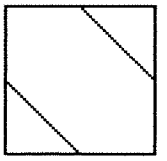
Minimum Mean = 48, $\lambda = 0.5$.

The melody consists of very short rising sequences, typically three to five pitches in length. Since the sequences are so short, the low pitches of each sequence tend to be heard as a slower part within the actual melody, as do the high pitches of each sequence and, to a lesser degree, the middle pitches.

Varying Minimum Mean, Example 7

Minimum Mean = 60, $\lambda = 20$.

The diagonal line is as follows:



This example shows the precise tendency in Example 8 that follows. The Minimum Mean of 60 is 14 pitches, in the C major scale, above the first pitch in the range, and the first rising sequence consists of pitches 1 15 29 of the 4 octave C major scale; that is, each successive pitch is 14 above the previous pitch. The 29th pitch, being the highest pitch in the range, is now followed by pitches 14

28, 13 27, 12 26, 11 25, 10 24, 9 23, 8 22, 7 21, 6 20, 5 19, 4 18, 3 17, 2 16, before returning again to the beginning of the range, after which the sequence repeats. Thus, within the overall sequence are two separate **falling** C major scales and this is how the overall sequence is heard even though it is constructed from two or three note **rising** sequences.

Varying Minimum Mean, Example 8

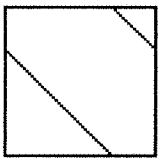
Minimum Mean = 60, $\lambda = 0.5$.

The melody consists of almost entirely of pairs of notes which are wide apart in pitch, and is therefore heard as two separate parts, a low one and a high one. Very occasionally, two consecutive pitches in the same register occur, but this does not affect the overall impression of a two part structure.

Varying Minimum Mean, Example 9

Minimum Mean = 72, $\lambda = 20$.

The diagonal line is as follows:



This example shows the precise tendency in Example 10 that follows. The Minimum Mean of 72 is 21 pitches, in the C major scale, above the first pitch in the range. Now, since the Minimum Mean is well above the mid point of the pitch range, the melody quickly reaches the top of the range and therefore falls. This makes the behaviour less readily discernible. The actual sequence, grouped according to the descending three or four note sequences of which it is constructed, consists of the pitches 1, 22 14 6, 27 19 11 3, 24 16 8, 29 21 13 5, 26 18 10 2, 23 15 7, 28 20 12 4, 25 17 9 of the four-octave C major scale. An upper part of 22 27 24 29 26 23 28 25 is readily heard. The distinction between the middle and lower parts is somewhat blurred as some notes are ambiguously heard as being in either the middle or lower parts, because for some three note sequences for which the first pitch is in the middle register and the third is in the lower register, the middle of the three pitches could be heard as belonging to the first

or the third. Thus, although middle and lower parts are definitely discernible, they may be heard differently on repeated playings of the exact same sequence.

Varying Minimum Mean, Example 10

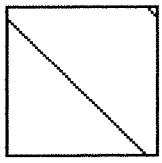
Minimum Mean = 72, $\lambda = 0.5$.

The melody consists mostly of short falling sequences of three or four notes. As the pitches of each sequence are widely spaced, this tends to be heard as three separate parts, high, medium and low.

Varying Minimum Mean, Example 11

Minimum Mean = 83, $\lambda = 20$.

The diagonal line is as follows:



This example shows the precise falling tendency in Example 12 that follows. The melody consists of the **alternate** pitches of the descending C major scale. Note that the Minimum Mean of 83 is 27 pitches above the lowest pitch of the range or **1 pitch below** the highest pitch of the range. In fact, the repeated falling sequences alternate between sequences beginning on B6, the highest but one pitch of the range, and sequences beginning on C7, the highest pitch of the range. This is because there are an odd number of pitches in the range, so a falling sequence beginning on B6 ends on D3, the second pitch of the range, which is then followed by C7 (27 pitches above D3).

Varying Minimum Mean, Example 12

Minimum Mean = 83, $\lambda = 0.5$.

While there are occasional rising sequences of pitches the overall trend is for the melody to fall. When the melody reaches the bottom of the range (C3), any further attempt to rise results in the melody returning immediately to the top

of the range - this is a result of the line wrapping around, together with the Reverse effect being turned off.

5.4.3 Conclusions

With a gradient of 1, Wraparound turned on and Reverse turned off:

1. A Minimum Mean whose value is anything other than the lowest pitch of the range produces a melody which tends to move in one specific direction, either up or down. The degree of this tendency is of course dependent on the value of λ.

2. A Minimum Mean of less than the mid point of the range produces a melody which tends to rise: the higher the value of the Minimum Mean the faster the melody rises. When melody reaches the highest point in the range it returns to the bottom of the range (because the Wraparound effect is on and the Reverse effect is off), so the melody consists of consecutive rising sequences. The tendency is for the melody to rise in steps equal to the number of pitches that the Minimum Mean is above the lowest pitch in the range. As the Minimum Mean rises towards the mid point of the range, the rising sequences contain fewer notes and these notes are wider apart in pitch, so separate melodic parts begin to be heard within the overall sequence.

3. A Minimum Mean greater than the mid point of the range produces a melody which tends to fall: the higher the value of the Minimum Mean the slower the melody falls. When the melody reaches the lowest point in the range it returns to the top of the range (because the Wraparound effect is on and the Reverse effect is off), so the melody consists of consecutive falling sequences. As the Minimum Mean rises just above the mid point, the melody consists of falling sequences of few notes which are wide apart in pitch, so separate melodic parts are heard within the overall sequence. As the Minimum Mean rises towards the top of the range, the falling sequences contain more notes and the impression of separate melodic parts begins to disappear. The tendency is for the melody to fall in steps equal to **one greater** than the number of pitches that the Minimum Mean is below the highest pitch in the range.

5.5 GRADIENTS GREATER THAN 1 OR LESS THAN -1

5.5.1 Introduction

When the gradient is greater than 1 or less than -1, the results for different gradients become very varied and more difficult to predict, with an apparently small change in gradient producing a significantly different result. What is consistent about the note sequences produced is that they tend, after a short transient sequence, to settle to a repeating sequence of pitches, this tendency, of course, increasing as λ increases: to hear the precise sequence λ must be set to a high value (20, for example - see Section **5.2.3** above) so that the result is completely deterministic. The number of notes in the sequence can be anything, from a single pitch to the entire length of the range - that is, a single attack, or such as to bring every pitch in the range into play - and this is one aspect which varies enormously for quite similar gradient values. Each pitch value in the sequence is unique; that is, no pitch is repeated within each repetition of the sequence. Moreover, for the same gradient, a different sequence may be produced from a different starting pitch value: in this case, the different sequences are mutually exclusive; that is, they contain no pitches in common. This mutual exclusivity is not at all surprising: for example, if a starting pitch of 36 produced the repeating sequence 36 37 38 39, and a starting pitch of 40 produced a different repeating sequence then this sequence could not possibly contain any of the pitches 36 37 38 or 39 because if it contained, say, 36, then this would be followed by 37 38 39 and would therefore not be a different sequence.

A selection of examples is given below, to illustrate the variation which can be achieved. Four different gradients are examined: 1.8, 4, 3.5 and -2. For Gradient = 1.8, one example is given with $\lambda = 20$. This is because, as discussed in more detail below, for this Gradient value the **same** sequence is achieved for **all** starting values (apart from 36, which results in a repeating single pitch sequence). For Gradient = 4, **two** examples are given with $\lambda = 20$, with different starting values in order to demonstrate the **two** possible sequences which occur for this Gradient. For the same reason, for Gradient = 3.5, three examples are given with $\lambda = 20$, with different starting values, while for Gradient = -2, one example is given with $\lambda = 20$, because the same sequence is achieved for all starting values (apart from 69, which results in a repeating single pitch sequence). For the first six examples, the fixed parameter values are:-

Pitch values:	a four-octave C major scale centred on middle C (that is, C3 to C7)
Minimum Mean:	36 (C3)
Tempo:	60 beats per minute

Note Length:	1/8 (that is, 8 notes per beat, or 8 notes per second in this case)
Wraparound:	ON
Reverse:	OFF

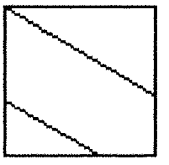
The files on the accompanying compact disc and floppy disk which support the examples in the following analysis may be found in the directory called "Analysis of the Algorithm", subdirectory "Gradients Greater Than 1 or Less Than 1". The files are named numerically corresponding to the numbered examples below.

5.5.2 Analysis

Gradients Greater Than 1 or Less Than -1, Example 1

Gradient = 1.8, $\lambda = 20$, Starting Pitch = 38.

The diagonal line is as follows:-



With this high value of λ, the precise sequence is achieved. After a short transient sequence of, in this case, rising bass notes, the melody settles to a repeating sequence of seven notes. Experimentation shows that, after differing initial transient sequences, the **same** seven note sequence pertains no matter what the starting pitch, unless the starting pitch is 36 (C3), in which case the sequence is just repeated C3's (which is what one would expect given that the Minimum Mean of the diagonal line is 36).

Gradients Greater Than 1 or Less Than -1, Example 2

Gradient = 1.8, $\lambda = 2$, Starting Pitch = 38.

This is the same gradient as the previous example but the λ value of 2 means that some variation about the tendency will occur. What are heard are fragments of the seven pitch sequence of differing lengths, interspersed with

the rising transient bass sequence, sequences varying randomly but shaped around the seven note sequence, and occasional repeating C3 sequences.

Gradients Greater Than 1 or Less Than -1, Example 3

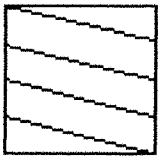
Gradient = 1.8, $\lambda = 0.5$, Starting Pitch = 38.

This is the same gradient as the previous two examples but at this value of λ the melody is allowed to vary to the extent that, although it has similar characteristics to the previous example, the influence of the seven note sequence is much weaker. Rising bass sequences are still evident however.

Gradients Greater Than 1 or Less Than -1, Example 4

Gradient = 4, $\lambda = 20$, Starting Pitch = 38.

The diagonal line is as follows:-



The melody consists of a repeating 14 note sequence, with no initial transient sequence.

Gradients Greater Than 1 or Less Than -1, Example 5

Gradient = 4, $\lambda = 20$, Starting Pitch = 40.

This is the same as the previous example but with a different starting pitch. The melody also consists of a repeating 14 note sequence but this sequence has no pitches in common with the previous sequence. Note that these two 14 note sequences cover 28 of the 29 pitches in the specified range. The missing pitch is C3 (36) - if the starting pitch was C3 then the resulting melody would consist of repeating C3's.

Gradients Greater Than 1 or Less Than -1, Example 6

Gradient = 4, λ = 4, Starting Pitch = 38.

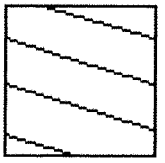
This is the same as example 5 but this value of λ allows a degree of variation while imposing a fairly strong tendency. The result is interwoven fragments, of varying lengths, of each of the 14 note sequences in the previous two examples.

For the next four examples, the fixed parameter values are as above but the Minimum Mean is now 47 (B3).

Gradients Greater Than 1 or Less Than -1, Example 7

Gradient = 3.5, λ = 20, Minimum Mean = 47, Starting Pitch = 36.

The diagonal line is as follows:-



After a short transient sequence, the melody settles to a repeating five note sequence. The pitch sequence forms an arpeggiated Dm chord.

Gradients Greater Than 1 or Less Than -1, Example 8

Gradient = 3.5, λ = 20, Minimum Mean = 47, Starting Pitch = 41.

Here, the parameter values are as in the previous example but with a different starting pitch, The melody consists of a different, mutually exclusive, five note sequence. This sequence also forms an arpeggiated Dm chord but with the F3 in the bass implying a first inversion.

Gradients Greater Than 1 or Less Than -1, Example 9

Gradient = 3.5, λ = 20, Minimum Mean = 47, Starting Pitch = 43.

The parameter values are the same as in the previous two examples but with yet another starting pitch. The melody consists of yet a third different, mutually

exclusive five note sequence. This pitch sequence forms an arpeggiated Em chord, with the G3 in the bass implying a first inversion. Experimentation shows that, no matter what the starting pitch, the melody always settles to one of these three five note sequences except for a starting pitch of 52 (E4), which produces a repeating sequence of E4's.

Gradients Greater Than 1 or Less Than -1, Example 10

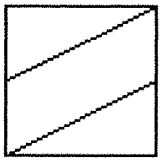
Gradient = 3.5, $\lambda = 4$, Minimum Mean = 47, Starting Pitch = 36.

This is the same as the previous three examples, with the same starting pitch as in example 7, but this value of λ allows a degree of variation while imposing a fairly strong tendency. The result is interwoven fragments, of varying lengths, of each of the five note sequences in the previous three examples, together with one short repeating E4 sequence and occasional short sequences of pitches outside of the three five note sequences. Alternating movement between the chords of Dm and Em can be heard.

Gradients Greater Than 1 or Less Than -1, Example 11

Gradient = -2, $\lambda = 20$, Minimum Mean = 84, Starting Pitch = 36.

The diagonal line is as follows:-



Here, the melody consists of a repeating sequence of 28 notes, and this is in fact the case for any starting pitch except 69 (A5), which results in a repeating sequence of A5's. There is also a slight feeling of a repeating I - IV - V chord progression in the key of C major.

Gradients Greater Than 1 or Less Than -1, Example 12

Gradient = -2, $\lambda = 4$, Minimum Mean = 84, Starting Pitch = 36.
This is the same as the previous example but this value of λ allows a degree of variation while imposing a fairly strong tendency. The result is interwoven

fragments, of varying lengths, from anywhere across the 28 note sequence, together with one short repeating A5 sequence.

5.6 VARYING NOTE LENGTH

5.6.1 Introduction

Exactly the same processes which have been examined thus far in relation to note pitch are equally at work with regard to any other parameter. Therefore, the analysis of note length variation which follows is not as exhaustive as the preceding analysis of note pitch variation: instead, representative examples are presented. However, it is important to observe that whereas melodies can be produced across a wide range of pitches (in the above examples, 29 different pitches across a four-octave C major scale), the number of different note lengths, if the melody is to sound rhythmical, is bound to be comparatively small. If a melody consists of 29 different note lengths then no rhythm will be established because the notes cannot be grouped such that a pulse emerges, whereas if a melody consists only of, say, semiquavers, quavers, crotchets and dotted crotchets then groupings of notes adding up to implied bars of equal lengths will tend to occur so that a pulse will be felt. There is, therefore, much less scope for the algorithm to produce a wide variety of rhythmical patterns to the same extent that different melodic patterns were produced in the previous examples if the number of different note lengths from which it can select is small. Having said that, of the 11 examples presented below, the first five examples in this section use all possible note lengths obtained from dividing a crotchet into 32 parts; that is, 1/32, 2/32, 3/32, ... , 31/32, 32/32 (=1). While this will not produce a **recognisable** rhythm, it is equally important to understand that this is no less musically valid and that interesting rhythmical effects can be produced. The last six examples use a much smaller number of note lengths.

The files on the accompanying compact disc and floppy disk which support the examples in the following analysis may be found in the directory called "Analysis of the Algorithm", subdirectory "Varying Note Length". The files are named numerically corresponding to the numbered examples below.

5.6.2 Analysis

For all the examples in this section, the fixed parameter values are:-

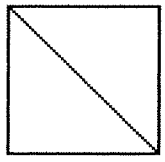
Pitch values: a four-octave C major scale centred on middle C (that is, C3 to C7)

Minimum Mean Pitch:	36 (= C3, the lowest pitch of the range)
Pitch Gradient:	1
Pitch λ:	0.5
Starting Pitch:	36 (= C3, the lowest pitch of the range)
Tempo:	120 beats per minute
Pitch Wraparound:	ON
Pitch Reverse:	OFF
Length Wraparound:	ON
Length Reverse:	OFF

Varying Note Length, Example 1

Note Lengths:	1/32, 2/32, 3/32, ... , 31/32, 32/32 of 1 beat
Minimum Mean Length:	1/32
Length Gradient:	1
Length λ:	0.5
Starting Length:	1/32
Passage Length:	64 beats (32 seconds)

The diagonal line is as follows:-

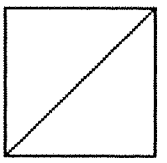


This produces a sequence of notes whose lengths meander gently up and down the 32 length range. The result is alternate periods of accelerando and rallentando.

Varying Note Length, Example 2

Note Lengths:	1/32, 2/32, 3/32, ... , 31/32, 32/32 of 1 beat
Minimum Mean Length:	32/32 (1 beat)
Length Gradient:	-1
Length λ:	0.5
Starting Length:	1/32
Passage Length:	64 beats (32 seconds)

The diagonal line is as follows:

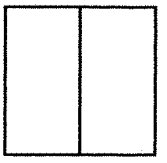


Lengths at one extreme of the range tend to be followed by lengths at the other extreme, while lengths towards the centre of the range tend to be followed by lengths also towards the centre. The result is sequences consisting of alternating short and long notes interspersed with sequences of notes of more even length.

Varying Note Length, Example 3

Note Lengths:	1/32, 2/32, 3/32, ... , 31/32, 32/32 of 1 beat
Minimum Mean Length:	16/32 (1/2 beat)
Length Gradient:	0
Length λ:	0.5
Starting Length:	16/32 (1/2 beat)
Passage Length:	48 beats (24 seconds)

The diagonal line is as follows:-

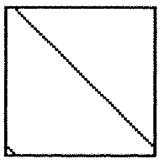


The notes centre around one quaver's length, with deviations up and down. The general effect is of the player keeping time very badly!

Varying Note Length, Example 4

Note Lengths:	1/32, 2/32, 3/32, ... , 31/32, 32/32 of 1 beat
Minimum Mean Length:	2/32 (1/16 beat)
Length Gradient:	1
Length λ:	1
Starting Length:	1/32
Passage Length:	64 beats (32 seconds)

The diagonal line is as follows:

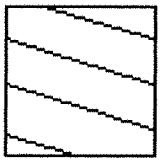


As previously observed, the result of a Minimum Mean slightly higher than the Minimum value of the range is a sequence of parameter values which tend to rise until the Maximum value is reached, at which point, since Wraparound is on, the sequence returns again to the Minimum before rising again. In the case of note lengths, the result is a melody consisting of consecutive rallentando sequences. If the Minimum Mean was slightly lower than the Maximum value of the range then the result would be consecutive accelerando sequences.

Varying Note Length, Example 5

Note Lengths:	1/32, 2/32, 3/32, ... , 31/32, 32/32 of 1 beat
Minimum Mean Length:	1/32
Length Gradient:	3.5
Length λ:	20
Starting Length:	11/32
Passage Length:	48 beats (24 seconds)

The diagonal line is as follows:-



As previously observed, Gradients bigger than 1 tend to produce repeating patterns of parameter values. Here, λ has been set to 20 so that the precise pattern is obtained. In this case the repeating pattern of note lengths is a nine note sequence as follows:

11, 4, 12, 8, 26, 25, 21, 7, 22

Although each note length in the nine note sequence is unique, the difference between some of them is not discernible; for example, 11/32 and 12/32, 25/32 and 26/32.

Varying Note Length, Example 6

Note Lengths:	1/4 beat, 1/2 beat and 1 beat
Minimum Mean Length:	1/4 beat
Length Gradient:	1
Length λ:	0.5
Starting Length:	1 beat
Passage Length:	64 beats (32 seconds)

In this example, only semiquavers, quavers and crotchets have been selected. As discussed in the introduction to this section, this makes it much more likely that a rhythmical pulse will be felt. A steady backing crotchet drum beat has been added to provide a reference. Most of the time, the melody seems to be "in time" but occasionally it seems to slip "out of synch" with the drum beat. This happens when the pulse of the melody is, for more than just a few notes, off the drum beat by a semiquaver which, unlike being off by a quaver, is disconcerting to the listener. This effect is confirmed by the following example, which allows only quavers and crotchets.

Varying Note Length, Example 7

Note Lengths:	1/2 beat and 1 beat
Minimum Mean Length:	1/2 beat
Length Gradient:	1
Length λ:	0.5
Starting Length:	1 beat
Passage Length:	64 beats (32 seconds)

Here, although a syncopation effect occurs when the pulse of the melody is off the drum beat by a quaver, a sense of a constant pulse is nevertheless maintained throughout the melody.

Varying Note Length, Example 8

Note Lengths:	1/4 beat, 1/2 beat and 1 beat
Minimum Mean Length:	1/4 beat
Length Gradient:	1
Length λ:	3
Starting Length:	1 beat

Passage Length:	64 beats (32 seconds)

This is the same as example 6 but the higher value λ of 3 means that the lengths will vary more slowly. There are, therefore, longer sequences of repeating semiquavers, quavers or crotchets. This also means that when the melody falls off the pulse by a semiquaver, it disconcertingly stays so for longer.

Varying Note Length, Example 9

Note Lengths:	1 beat, 2 beats and 4 beats
Minimum Mean Length:	1
Length Gradient:	1
Length λ:	3
Starting Length:	4 beats
Passage Length:	64 beats (32 seconds)

This is, comparatively speaking, the same as example 8 but the effect of employing longer note lengths is to remove the disconcerting "out of synch" sequences.

Varying Note Length, Example 10

Note Lengths:	1/4 beat, 1/2 beat, 1 beat and 2 beats
Minimum Mean Length:	1/4 beat
Length Gradient:	0
Length λ:	1
Starting Length:	1/4 beat
Passage Length:	64 beats (32 seconds)

Here, the gradient of 0, together with a Minimum Mean Length of a semiquaver, means that more of the lengths will tend to be semiquavers with the interjection of some quavers, much fewer crotchets and very occasional minims. The resulting melody maintains the sense of a steady pulse because, even though it may sometimes fall off the pulse by a semiquaver, another semiquaver soon occurs to bring it back in synch.

Varying Note Length, Example 11

Note Lengths:	1/4 beat, 1/2 beat, 1 beat and 2 beats
Minimum Mean Length:	2 beats
Length Gradient:	0
Length λ:	1
Starting Length:	2 beats
Passage Length:	64 beats (32 seconds)

Here, the gradient of 0, together with a Minimum Mean Length of a minim, means that more of the lengths will tend to be minims with the interjection of some crotchets, much fewer quavers and very occasional semiquavers. Now, the pulse becomes lost because once a semiquaver occurs to put the melody out of synch, it stays so for a long period.

5.7 VARYING NOTE VELOCITY

5.7.1 Introduction

Exactly the same processes which have been examined thus far in relation to note pitch are equally at work with regard to any other parameter. Therefore, the analysis of note velocity variation which follows is not as exhaustive as the preceding analysis of note pitch variation: instead, representative examples are presented.

The files on the accompanying compact disc and floppy disk which support the examples in the following analysis may be found in the directory called "Analysis of the Algorithm", subdirectory "Varying Note Velocity". The files are named numerically corresponding to the numbered examples below.

5.7.2 Analysis

For all the examples in this section, the fixed parameter values are:-

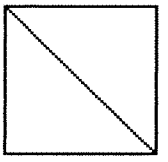
Pitch values:	a four-octave C major scale centred on middle C (that is, C3 to C7)
Minimum Mean Pitch:	36 (= C3, the lowest pitch of the range)
Pitch Gradient:	1
Pitch λ:	0.5
Starting Pitch:	36 (= C3, the lowest pitch of the range)

Tempo:	60 beats per minute
Note Length:	1/8 (that is, 8 notes per beat, or 8 notes per second in this case)
Velocity Range:	50 - 127
Pitch Wraparound:	ON
Pitch Reverse:	OFF
Length Wraparound:	ON
Length Reverse:	OFF
Passage Length:	48 beats (= 48 seconds)

Varying Note Velocity, Example 1

Minimum Mean Velocity:	50 (lowest value in range)
Velocity Gradient:	1
Velocity λ:	0.05
Starting Velocity:	127 (highest value in range)

The diagonal line is as follows:-

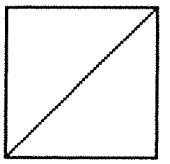


The note velocity meanders up and down the range. The result is a melody containing alternating periods of rising and falling dynamic, of various lengths.

Varying Note Velocity, Example 2

Minimum Mean Velocity:	127 (highest value in range)
Velocity Gradient:	-1
Velocity λ:	0.5
Starting Velocity:	127 (highest value in range)

The diagonal line is as follows:

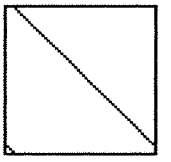


Velocities at one extreme of the range tend to be followed by velocities at the other extreme, while velocities towards the centre of the range tend to be followed by lengths also towards the centre. The result is sequences consisting of alternating loud and soft notes interspersed with sequences of notes of more even dynamic towards the centre of the range. When the leap from loud to soft is very wide, the loud notes are heard as a slower melodic line of half the tempo of the actual generated sequence, the soft notes being barely discernible.

Varying Note Velocity, Example 3

Minimum Mean Velocity:	51
Velocity Gradient:	1
Velocity λ:	0.5
Starting Velocity:	51

The diagonal line is as follows:

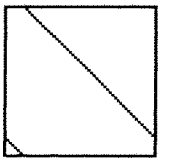


As discussed in previous sections, minimum mean values slightly above the lowest value in the range produce sequences of parameter values which tend to rise. Here, the result is successive sequences of notes which gradually rise to a crescendo.

Varying Note Velocity, Example 4

Minimum Mean Velocity:	60
Velocity Gradient:	1
Velocity λ:	0.5
Starting Velocity:	50

The diagonal line is as follows:

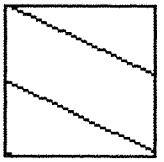


Here, the minimum mean velocity is ten above the lowest value in the range so the consecutive sequences build much more rapidly to a crescendo than in the previous example. Each sequence is, on average, eight notes in length.

Varying Note Velocity, Example 5

Minimum Mean Velocity:	51
Velocity Gradient:	2
Velocity λ:	20
Starting Velocity:	51

The diagonal line is as follows:



As previously observed, Gradients bigger than 1 tend to produce repeating patterns of parameter values. Here, λ has been set to 20 so that the precise pattern is obtained. The result is repeating sequences of 12 different velocity values: 51, 53, 57, 65, 81, 113, 99, 71, 93, 59, 69, 89. The values of 113, 93 and 89, since they are followed by either two or five notes of comparatively smaller velocity, produce stresses which imply a definite triple metre.

5.8 THE REVERSE OPTION

5.8.1 Introduction

The purpose of the Reverse Option (see Section **2.3.4**) is to effect a reverse in direction of a parameter when that parameter is tending to move in a particular direction and reaches the upper or lower limit of its range of values. For example, a melody which is tending to rise will begin to fall when it reaches the top of the pitch range and a melody which is tending to fall will begin to rise when it reaches the bottom of the pitch range.

As previously observed, a tendency to move in a particular direction occurs when the Minimum Mean is other than the lowest value in the range and the Gradient is 1, the speed of movement being higher the further the Minimum Mean is from the lowest value in the range.

Three examples are given below, showing the effect of the Reverse Option on Pitch, Note Length and Note Velocity. Each is the same as an example from a previous section, in which a tendency to move in a particular direction was evident, but with Reverse Option turned on for the relevant parameter

The files on the accompanying compact disc and floppy disk which support the examples in the following analysis may be found in the directory called "Analysis of the Algorithm", subdirectory "The Reverse Option". The files are named numerically corresponding to the numbered examples below.

5.8.2 Analysis

The Reverse Option, Example 1

Pitch values:	a four-octave C major scale centred on middle C (that is, C3 to C7)
Starting Pitch:	36 (= C3, the lowest pitch of the range)
Minimum Mean Pitch:	38
Pitch Gradient:	1
Pitch λ:	0.5
Note Length:	1/8 (that is, 8 notes per beat, or 8 notes per second in this case)
Tempo:	60 beats per minute
Passage Length:	32 beats (32 seconds)
Pitch Wraparound:	ON
Pitch Reverse:	ON

This is the same as Example 2 in Section **5.4.2** above but with the Pitch Reverse Option turned on.

Initially, while there are some descending sequences of pitches the overall trend is for the melody to rise. When the melody reaches the top of the range (C7), it starts to fall and, while there are some rising sequences the overall trend is for the melody to continue to fall until it reaches the lowest pitch in the range (C3), at which point it begins to rise again. The melody therefore consists of alternate rising and falling sequences spanning the entire pitch range.

The Reverse Option, Example 2

Pitch values:	a four-octave C major scale centred on middle C (that is, C3 to C7)
Minimum Mean Pitch:	36 (= C3, the lowest pitch of the range)
Pitch Gradient:	1
Pitch λ:	0.5
Starting Pitch:	36 (= C3, the lowest pitch of the range)
Tempo:	120 beats per minute
Note Lengths:	1/32, 2/32, 3/32, ... , 31/32, 32/32 of 1 beat
Minimum Mean Length:	2/32 (1/16 beat)
Length Gradient:	1
Length λ:	1
Starting Length:	1/32
Passage Length:	64 beats (32 seconds)
Pitch Wraparound:	ON
Pitch Reverse:	OFF
Length Wraparound:	ON
Length Reverse:	ON

This is the same as Example 4 in Section **5.6.2** above but with the Note Length Reverse Option turned on.

The melody begins with a rallentando sequence which continues to slow until the note length reaches the top of the note length range (1 beat, the longest note of the range), at which point it begins to quicken again. It continues to quicken until the note length reaches the bottom of the note length range (1/32 beat, the shortest note of the range), when it begins to slow again. The melody therefore consists of alternate rallentando and accelerando sequences spanning the entire note length range.

The Reverse Option, Example 3

Pitch values:	a four-octave C major scale centred on middle C (that is, C3 to C7)
Minimum Mean Pitch:	36 (= C3, the lowest pitch of the range)
Pitch Gradient:	1
Pitch λ:	0.5
Starting Pitch:	36 (= C3, the lowest pitch of the range)
Tempo:	60 beats per minute

Note Length:	1/8 (that is, 8 notes per beat, or 8 notes per second in this case)
Velocity Range:	50 - 127
Minimum Mean Velocity:	51
Velocity Gradient:	1
Velocity λ:	0.5
Starting Velocity:	51
Pitch Wraparound:	ON
Pitch Reverse:	OFF
Length Wraparound:	ON
Length Reverse:	OFF
Velocity Wraparound:	ON
Velocity Reverse:	ON
Passage Length:	48 beats (= 48 seconds)

This the same as Example 3 in Section **5.7.2** above but with the Velocity Reverse Option turned on.

Initially, the melody gradually rises to a crescendo until the note velocity reaches the top of the note velocity range (127, the loudest note in the range), at which point the velocity begins to fall, resulting in a diminuendo sequence, until the velocity reaches the bottom of the note velocity range (50, the quietest note in the range), when the velocity begins to rise again. The melody therefore consists of alternate crescendo and diminuendo sequences spanning the entire note velocity range.

5.9 THE REFLECT OPTION

5.9.1 Introduction

The Reflect option (see Section **2.3.3**) only has an effect if the Gradient is greater than 1 or less than -1, because for these Gradients the diagonal line will hit the right or left hand edge (or both) of its enclosing rectangle and therefore be reflected from it. As seen in Section **5.5** above, when the Wraparound, rather than the Reflect, option is on, such Gradients tend to produce mutually-exclusive repeating sequences whose values depend on the starting value. It turns out that the same behaviour occurs when the Reflect option is on. However, the repeating sequences produced by the Reflect option often manifest two important behavioural differences to those produced by the Wraparound option:-

1. While Wraparound-produced repeating sequences tend, apart from single note sequences, to be of the same length (Examples 4 and 5 in Section 5.5 above produce two different 14 note sequences, and examples 7, 8 and 9 produce three different five note sequences), Reflect-produced repeating sequences can be of widely differing lengths (the example below produces two 1-note sequences, two 3-note sequences and a 6-note sequence).

2. While Wraparound-produced repeating sequences tend to cover much of the overall parameter range, and to overlap, it is possible for Reflect-produced sequences to occur in a narrow sub-range within the overall range and to be non-overlapping with any of the other sequences (the example below produces a high E-minor triad which is in a different register to the other repeating sequences).

The file on the accompanying compact disc and floppy disk which supports the example in the following analysis may be found in the directory called "Analysis of the Algorithm", subdirectory "The Reflect Option". The file is named numerically corresponding to the numbered example below.

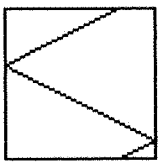
5.9.2 Analysis

The Reflect Option, Example 1

The parameter values are as follows:-

Pitch values:	a four-octave C major scale centred on middle C (that is, C3 to C7)
Minimum Mean Pitch:	72 (C6)
Pitch Gradient:	-2
Pitch λ:	3
Starting Pitch:	74 (D6)
Tempo:	60 beats per minute
Note Length:	1/8 (that is, 8 notes per beat, or 8 notes per second in this case)
Wraparound:	OFF
Reflect	ON
Reverse:	OFF
Passage Length:	48 beats (= 48 seconds)

The diagonal line is as follows:-



Experimentation with a Pitch λ value of 20, and with different starting pitch values throughout the whole range, shows that the possible repeating sequences are:

C6 (single repeated pitch)
A5 F5 B4 A3 G4 D3
F3 D5 E4
C4 (single repeated pitch)
B6 E6 G6

The example here has a Pitch λ value of 3 so the result is interwoven fragments, of varying lengths, of each of the possible sequences. For this particular realisation, the passage spends the first half of its length around the high three note sequence (B6 E6 G6) before moving down to spend most of the second half around the lower sequences.

Chapter 6

Style Emulation

6.1 INTRODUCTION

In this chapter, we shall investigate how well the *Markov* program algorithm may be used to generate music based on existing, known styles.

Four different, intentionally diverse, styles have been examined: Steve Reich Phase Music, Gagaku, Bach Harpsichord and Dance Music. They have been chosen in order to demonstrate:-

(i) the flexibility of the algorithm
(ii) important compositional techniques used when constructing pieces from the algorithm
(iii) the strengths and limitations of the algorithm when attempting to reproduce a given style

The first style, Steve Reich Phase Music, is itself based on a systematic compositional method and it is therefore no surprise that the algorithm is quite successful here. The "phasing" technique inherent in this music is reproducible exactly and the main limitation is the lack of freedom the composer has when constructing the melody.

The second style, Gagaku, is included as an example of a non-western style. This style, being generally harmonically less tightly structured than western styles, turns out to be quite well suited to the algorithm although the tuning system, being different to the western equal-tempered scale, causes some problems due to the limitations of the MIDI standard.

The third style, Bach Harpsichord music, has been chosen as an example of a style of music which has a very tight melodic and harmonic structure. It is not surprising that this type of music is difficult to reproduce given the small amount of input data supplied to each Part that makes up a composition. To produce a new piece reproducing such a style **exactly** would require a thorough analysis of the given style in order to formulate a complete, and possibly large and complex, set of rules specifying the stylistic behaviour. The computer can then produce a composition which obeys all these rules (see Section **1.2.4.1**) and the result is another piece in the same style. This process is in itself not, however, particularly creative. The Diagonal Line algorithm, on the other hand, allows the composer to introduce a degree of determinism such that intended stylistic features are apparent but which nevertheless leaves open the possibility for unexpected musical events to occur. In this attempt to emulate Bach harpsichord music, relatively successful results are obtained by using the algorithm to model one bar at a time, but the need to model the piece in such small, bar-sized sequences exposes a very important limitation of the algorithm.

The final style, Dance Music, is included as an example of a present-day, "popular" style. Here, the ability of the algorithm to generate precise rhythmic structures is demonstrated and, since much of the music in this style is itself often computer generated using a systematic approach, successful results are obtained. It will be seen, however, that a disconcertingly large amount of algorithmic effort is required to achieve very simple melodic sequence structures.

Thus, what is demonstrated in this chapter is that such a deceptively simple algorithm is capable of producing an extremely wide range of stylistic output and that in many instances precise, deterministic compositional elements are obtainable, whilst, equally importantly, in many other instances the algorithm is not so well suited and compositional compromises must be made.

The particular styles investigated here have been chosen not as specific compositional goals in themselves but more to demonstrate both the enormous flexibility of the algorithm **and** its limitations. The chosen styles are not as important as the compositional process at work: that is, starting from a set of stylistic objectives, how does the composer manipulate the parameters of the algorithm such that a resulting composition behaves appropriately? This process is key to the algorithmic approach employed here. At the same time, whilst it is not under consideration here, the "wind it up, let it go and see what happens" approach is also possible and represents another, completely different, way of composing with this algorithm.

The approach that has been adopted here is to take original recorded examples of the chosen styles (and, where possible, an original score) and, in a part informed and part subjective manner, identify key stylistic features. The *Markov* program is then used to construct a short piece, in which the mathematical parameters of the various parts are set so as to approximate as nearly as possible the previously identified features[1]. Extracts from the parametric score (see Section **3.2.6** and Appendix A, Section **4.6**) produced by the *Markov* program are introduced regularly to show the relevant parameter settings.

In the case of the Steve Reich Phase Music and Dance Music styles, the pieces produced from the *Markov* program are constructed in a very precise manner and, consequently, each playing of the composition produces exactly the same result. For the Gagaku and Bach harpsichord styles, the probabilistic nature of the algorithm comes into play and each playing of the composition

---

[1] Note that since the algorithm is, as discussed above, not rule based, these features are relatively simple ones and do not include, for example, permissible cadence patterns, or precise melodic structures. It is therefore surprising how successful the resulting music can be given the intrinsic simplicity of the principles from which is it is constructed.

generally produces a different result. It will therefore very likely be the case that some renditions may, in reference to original examples of the style in question, be more convincing than others. This is nothing to fear: on the contrary, it adds to the creativity of the process since it allows the mathematical system to introduce variations on the original style while nevertheless maintaining the overall character.

It should not be overlooked that, in addition to the choices of parameter values, certain fundamentally important musical choices are made, particularly scale and timbre. If, for example, a chromatic scale was chosen when the original style was based on a melodic minor scale, or a trombone timbre was selected when the original work under investigation was a piano piece, then it is highly unlikely that the musical result from the algorithm would be in any way convincing no matter what the choice of mathematical parameters. However, while such choices are necessary they are absolutely not sufficient. The investigations in Chapter 5 show in great detail the enormous musical variations in output which are obtained from relatively small changes in the values of algorithm parameters, and the lessons learned there are put into practice here in making informed initial choices of parameter values. What are given in the current study are the final choices of the parameters. These were arrived at by a continuous process of refinement, the *Markov* program being used to facilitate this process in a manner explored in detail in Chapter 4, until a result was obtained which, **in the opinion of the composer,** satisfied the original objectives. This, again, is key to the compositional approach which this algorithm affords, but it should be emphasised that the assessment of the success, or otherwise, of the algorithm in emulating the musical styles in question is based on empirical, subjective judgements of the results by ear.

The compact disc which accompanies this thesis contains audio file encapsulations of renderings of each of the pieces produced by the *Markov* program; for the Gagaku style, three separate renderings of each are included while for the Bach harpsichord style, six are included. For the Gagaku and Reich styles, audio files containing a short extract from original recordings of performances in these styles are also included. The floppy disk contains the examples as *Markov* program Composition Files for playback from the *Markov* program. Full details of the filenames and locations are given later in this chapter, in the relevant sections.

6.2 STEVE REICH PHASE MUSIC

6.2.1 Introduction

Steve Reich discovered the process of "Phase" music by accident in 1965 while playing with tape loops of the recorded voice of a Pentecostal preacher. He allowed the two tape loops to gradually move completely out of phase with one another and then slowly move back into unison. The result was the work **It's Gonna Rain**. He then applied this technique to live instrumental music. In **Piano Phase** (1967), two pianists play a short repeating pattern of notes, one playing steadily while the other gradually increases his or her tempo so as to slowly move ahead of the other. This process is continued until both players are back in unison, at which point the pattern is changed and the phasing process begins again.

The example produced by the program is not based on Piano Phase specifically but rather on the general features of Reich's phase pieces, although piano has been chosen as the instrument. It demonstrates how the program can produce completely deterministic results rather than probabilistic ones. In this case, as previously stated, exactly the same piece results every time the composition is replayed.

The accompanying compact disc contains a short extract from Piano Phase, taken from the compact disc *Steve Reich, Early Works*, Elektra/Asylum/Nonesuch Records, 1987, performed by Nurit Tilles and Edmund Niemann (compact disc number 7559-79169-2). It is in a WAV-formatted audio file called "Reich Original.wav" and may be played by any application capable of playing WAV files (Microsoft Windows™ Media Player for example). The compact disc also contains the piece produced by the program in a file called "Reich.wav". The floppy disk contains the program-generated piece as a *Markov* program Composition File called "Reich" for playback from the *Markov* program. All files may be found in the "Style Emulation" directory.

6.2.2 Identifying the Key Elements

There are two key stylistic features:-
a) A short single part piano line, repeated by two separate piano players
b) The two separate piano parts gradually move out of phase before eventually coming together again.

6.2.3 Constructing the Piece Using the Algorithm

As discussed in detail in Section **5.5.2**, repeating sequences are produced by gradients of greater than 1 or less than -1 with λ set high enough to produce a deterministic result. The sequence which results from a particular choice of parameter values is difficult to predict and the sequence used for the *Markov* program Phase Music piece was arrived at after trial and error experimentation.

The relevant section of the score relating to the sequence of pitches is as follows:-

```
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           60        80          62         60     2.000000 20.000000

PITCH SELECTIONS:
  60   62   63   65   67   68   71   72   74   75   77   79   80
```

Note that the pitch selections correspond to 13 notes of a C harmonic minor scale starting from middle C. A gradient of 2 with a Minimum Mean of 62 produces a particular 12 pitch repeating pattern which was felt to be appropriate to the task in hand. The λ value is 20 which, as we have seen, produces a deterministic result where each note in the sequence obeys precisely the prediction of the Diagonal Line algorithm. The exact sequence of MIDI pitch values produced is 60, 62, 65, 72, 63, 68, 79, 77, 74, 64, 75, 71 corresponding to C5, D5, F5, C6, E♭5, G#5, G6, F6, D6, G5, E♭6, B5. This sequence results from setting the starting pitch to 60 (C5). The pitches produced are 12 of the 13 actually selected, the pitch value 80 (G#6) not occurring at all. In fact, a pitch of 80 would be followed by a continuous repeating sequence of 80's and this partitioning of the full set of selected pitches into mutually exclusive subsets (two in this case) depending on the starting value is a common feature of the algorithm for gradients of greater than 1 or less than -1. The reader is again referred to Section **5.5.2**, for a detailed analysis.

As noted previously, the algorithm rather limits the composer in the choice of melody which can be constructed deterministically. It is not possible to produce **any** chosen melody: indeed, the algorithm is not designed for this purpose, and it would be very surprising if a such a mathematical algorithm could achieve this. One specific limiting feature of the deterministic melodic sequences which can be produced by the algorithm, for example, is that, for each repetition of the sequence, each pitch occurs once and once only.

The phase shift effect is produced by setting the note length for the first part to one whole beat and the second to 190/192 of a beat[2]. Thus the second part gradually moves ahead of the first because its note length is very slightly shorter.

The respective relevant sections of the score for the two parts are as follows (the overall tempo being set to 400 beats per minute):

Parameter	Min	Max	MinMean	Start	Grad	Lambda
LENGTH (/192)	192	192	192		1.000000	0.500000

and

Parameter	Min	Max	MinMean	Start	Grad	Lambda
LENGTH (/192)	190	190	190		1.000000	0.500000

The length of the section must be calculated so that it ends at the point when the two parts come back into exact alignment. Now, Part 2 advances by 2/192 = 1/96 of a beat for every note played so after Part 2 has played 96 notes it will have advanced by one whole beat. The time it takes Part 2 to play 96 notes is 96 x 190/192 = 95 whole beats. Thus it will take 95 x 12 = 1140 beats for Part 2 to have advanced by 12 whole beats and therefore be back into exact alignment with Part 1. Therefore, the length parameters of each of the two Parts are set to begin on beat 1 and end on beat 1140. If desired, a new Section can now begin for which different Pitch parameters are specified in order to produce a different sequence. The complete *Markov* program score for the piece can be seen in Appendix B, Section **B.1.1**.

6.2.4 Discussion of the Results

This piece, having been constructed in a totally deterministic way so as to capture the key elements of Reich's work, is extremely convincing. One small point to make is that two live performers, having moved out of phase such that

[2] The chosen value of 192 for the beat length fraction is not arbitrary. The number of ticks per beat has been set in the program to 192 which the authors of Altech Systems' MIDIPascal recommend as it allows tempos of up to 300 beats per minute before interrupts begin to occur too rapidly for slower Apple Macintosh computers to cope. Therefore, choosing 192 as the beat fraction means that calculations of the lengths of subdivisions of a whole beat (190/192 in this case) will be accurate. Had a note length of, say, 99/100 been chosen then rounding errors would occur because the program would have to convert this to a fraction of 192, which be not be exactly the same value.

one was, for example, exactly one beat ahead of the other, would probably linger in this state for a short while before continuing to move out of phase, whereas the rate of phase shift between the two program generated parts is uniform throughout.

6.3 GAGAKU - JAPANESE COURT MUSIC

6.3.1 Introduction

The term **Gagaku**, meaning literally "tasteful" or "correct" music, encompasses all the traditional court music of Japan. Of this music, a subcategory, **Togaku**, refers to music primarily of Chinese origin which was arranged and standardised in Japan and dates from the late 8th century.
Reference was made to live recordings of Togaku Music on the compact disc *Gagaku: Court Music of Japan*, JVC World Sounds, 1994 (compact disc number VICG-5354).
The instruments used in these performances are:

Ryuteki, a high pitched bamboo flute
Hichiriki, a lower pitched double-reed instrument
Kakko, a double-headed barrel drum
Tsuridaiko, a large shallow barrel-shaped drum

The intention is to use the *Markov* program to create a short piece consisting of two sections, inspired by two separate pieces on the compact disc, as follows:-

1. An introduction formed by a short Ryuteki solo, backed by occasional Kakko and Tsuridaiko.
2. A main body consisting of a continuous drone of Sho over which Hichiriki begin to enter, first a solo and then several together.

The Togaku note system consists of seven fundamental modes, or **Choshi**. Each mode consists of a seven-note series. The examples here use the **Hyojo** mode, which approximates to the pitches E, F#, G, A, B, C# and D.

The Togaku repertory is classified both according to the kind of movement and by rhythmic type. The three classes of movement are **Jo**, a prelude or introduction, **Ha**, "breaking away", and **Kyu**, "rapid" or "hurried".

The examples here are of the Jo movement, which typically has a very slow tempo bordering on free rhythm[3].

Two short extracts from the aforementioned live recordings may be found on the accompanying compact disc in WAV-formatted audio files called "Gagaku Original 1.wav" and "Gagaku Original 2.wav". Three separate program-generated realisations are contained in the files "Gagaku1.wav", "Gagaku2.wav" and "Gagaku3.wav". The floppy disk contains the program-generated piece as a *Markov* program Composition File called "Gagaku" for playback from the *Markov* program. All files may be found in the "Style Emulation" directory.

6.3.2 Identifying the Key Elements

The following key stylistic features are identified in the two sections:-

Section 1

a) The solo has a slow 4/4 rhythm
b) The solo consists of quavers, crotchets, minims and semibreves.
c) Longer note lengths are more frequent than shorter ones
d) The solo contains occasional rests
e) The solo is based on the Hyojo mode
f) The melodic line moves fairly gently, usually moving up or down one pitch value at a time but with occasional larger jumps occurring
g) There are very few successive repetitions of the same pitch
h) The player frequently "bends" the notes
i) From about a third the way through the section, single Kakko drum beats occasionally enter
j) From about two thirds the way through the section, single Tsuridaiko drum beats occasionally enter, but with lower frequency than the Kakko
k) Towards the end of the section, a short, accelerating Kakko drum sequence is played

[3] The other movements are rhythmically stricter. For example, the rhythm **Haya Yahyoshi** consists of 16 metrical units, each of which consists of eight four-beat units and begins with a strong Tsuridaiko stroke.

Section 2

a) Sho notes form a continuous backing
b) The successive Sho notes are long and of varying length
c) The different notes come from the Hyojo mode
d) The volume of each Sho note rises and falls
e) A solo Hichiriki enters playing notes from the Hyojo mode
f) After a short while this is joined by a chorus of other Hichiriki all playing notes from a similar register so that discords occur
g) All Hichiriki notes are "bent" frequently
h) The note lengths vary such that the attack points are often offset
i) There is no obvious even rhythm

6.3.3 Constructing the Piece Using the Algorithm

Before going into the precise details of the algorithm parameter settings used in the construction of the piece, it is helpful, when considering the separate realisations of the piece produced by the program, to have a general awareness of how much is fixed and how much is open to variation by the algorithm. For this piece, the details are as follows:-

Section 1

Ryuteki solo:	The length of this solo is fixed at 1'12" but is open to considerable variation in pitch and rhythm on successive playings.
Kakko drum:	The time period over which these beats may occur is fixed at 0'24" to 1'0" but the precise attack points are subject to considerable variation. The same single pitch is used throughout.
Tsuridaiko drum:	The time period over which these beats may occur is fixed at 0'48" to 1'12" but, while occurring less frequently than the Kakko, the precise attack points are again subject to considerable variation. The same single pitch is used throughout.
Accelerating Kakko:	Lasts from 1' to 1'18". Although the precise attack points will vary on successive playings, no difference is readily discernible so this is essentially fixed.

Section 2

Sho backing:	There is a short crescendo of Sho notes from 1'12" to 1'18" which is fixed. Thereafter, the Sho notes continue until the end of the piece. The attack points and rising and falling volume of the notes are subject to considerable variation but the combination of these effects across three Sho instruments playing together averages out to the extent that there is no readily identifiable difference between successive playings.
Solo Hichiriki:	Subject to considerable variation.
Hichiriki chorus:	Subject to considerable variation. Although the overall effect sounds similar on successive playings there can be a noticeable difference in register.

The features described in Section **6.3.2** above are incorporated into the piece to be produced by the program by manipulating the algorithm parameters as follows:-

Section 1

To achieve quavers, crotchets, minims and semibreves, the beat fraction is set to 2, and the selection mechanism is used such that only note lengths of 1, 2, 4 and 8 are allowed, corresponding to quavers, crotchets, minims and semibreves respectively.

To control the frequency of occurrence of note lengths such that longer note lengths are more frequent, the Minimum Mean is set to the maximum of the range of note lengths, 8/2 (that is, a semibreve) with a gradient of 0. This will make note lengths towards the top of the range (the longer lengths) more likely to occur than those towards the lower end of the range (the shorter lengths) (see Section **5.3.2**, Example 10 for a discussion of this effect, but in relation to pitch). However, a λ value of 0.5 allows a reasonable degree of variation so that shorter note lengths will occur from time to time.

The relevant extract from the score is as follows:-

```
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
LENGTH (/  2)   1         8          8                  0.000000  0.500000

LENGTH SELECTIONS:
1   2   4   8
```

To achieve rests, the selection values of note velocity are set so that only 127 (maximum velocity) and 0 (no sound) are allowed. When velocities of 0 are generated, a rest will occur, lasting until such time as a velocity of 127 is next generated.

Since only occasional rests are desired, the Minimum Mean Velocity is set to 127 with a gradient of 0. A λ setting of 2 gives a fairly high tendency for velocities of 127 to be generated rather than velocities of 0, thus achieving occasional rests.

The relevant extract from the score is as follows:-

```
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
VELOCITY         0       127         127       127     0.000000  2.000000

VELOCITY SELECTIONS:
   0 127
```

To achieve the Hyojo mode, the following MIDI pitch values are selected: 76, 78, 79, 81, 83, 85, 86 and 88.

To achieve a gently moving melody, the Minimum Mean is set to 76, the lowest value of the pitch range, with a gradient of 1 and a λ value of 0.5 (see Section **5.3.2**, Example 1).

Since successive repetitions of the same pitch value are infrequent, it was decided to prevent successive repetitions of the same pitch occurring altogether, by selecting the **Disallow Repeats** option in the Pitch Parameter Values dialog box (see Appendix A, Section **3.3.2**).

A MIDI Patch value of 73 has been chosen, corresponding to Piccolo.

The complete section of the score corresponding to the solo part is as follows:-

```
Sect  1 Part  1, Chan  1, Patch  73, Pan  64, BEATS      1 to      60
===============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           76        88          76                1.000000  0.500000
LENGTH (/  2)    1         8           8                0.000000  0.500000
VELOCITY         0       127         127       127     0.000000  2.000000

             REPEAT   | SELECT   |  REVERSE   | REFLECT   | RANDOM ENTRY
             ------------------------------------------------------------
PITCH          NO     |   YES    |     NO     |    NO     |
LENGTH         YES    |   YES    |     NO     |    NO     |       NO
VELOCITY       YES    |   YES    |     NO     |    NO     |

PITCH SELECTIONS:
  76  78  79  81  83  85  86  88
LENGTH SELECTIONS:
   1   2   4   8
VELOCITY SELECTIONS:
   0 127
```

The bending of notes is achieved by using the Pitch Bend parameter. However, it is no use varying the pitch bend of the notes of the solo line itself, because this would just mean that the pitch bend value would vary on a note by note basis[4], whereas what is required here is that the pitch bend varies while a **single** note is playing. To achieve this effect, pitch bend changes must be sent from a different Part but to the **same** MIDI channel as the solo line. The program allows pitch bend values to be set in the range 0 to 127: 0 corresponds to full bend downwards and 127 corresponds to full bend upwards. For most MIDI devices, this implies, by default, a pitch bend range of one tone below to one tone above true pitch[5].

Here, the pitch bend parameter range is set to 32 to 96, giving a range from one semitone below to one semitone above true pitch. A Minimum Mean of 32 and a gradient of 1 with a λ value of 0.7 give a fairly gentle movement up and down the pitch bend range, which is necessary so that the pitch bends sound fairly smooth rather than jerky. The very first pitch bend value generated is 64 so that the first note played starts at true pitch. From then on, the pitch bend will move above and below true pitch but the average value should be around 64 so that the overall sense of modal identity is not lost.

The relevant extract from the score is as follows:-

```
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH BEND      32        96          32         64     1.000000  0.700000
```

All note lengths are 1/8 so that the pitch bend changes fairly quickly (8 times a beat) thus giving the desired pitch bend effect. Only zero velocities are allowed so that no Note On or Note Off events will be generated, just Pitch Bend Change events. The Pitch settings are therefore irrelevant and have been left at the program default values of 60.

The complete section of the score corresponding to the pitch bend change part is as follows:-

---

[4] Note that this could be a valid compositional objective (to achieve microtonal intervals for example) but is not appropriate here.

[5] If the pitch bend range has been changed on the MIDI device then the actual pitch bend values implied by the values in the range 0 to 127, as set in the Markov program, will change proportionately. The program could have allowed the user to set the pitch bend range on the MIDI device by setting parameter values in the program itself but it was decided not to implement this in order to save on disk and memory usage and to limit the amount of data the user has to manipulate. The user can still change the Pitch Bend Range setting on the MIDI device itself if required, but note that since the program allows only 128 Pitch Bend settings, the wider the Pitch Bend Range the worse the resolution becomes. However, 128 settings over a two tone range is perfectly adequate.

```
Sect  1 Part  2, Chan  1, Patch  73, Pan  64, BEATS      1 to      60
==============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           60        60          60                1.000000  0.500000
LENGTH (/  8)    1         1           1                1.000000  0.500000
VELOCITY         0         0           0                1.000000  0.500000
PITCH BEND      32        96          32        64      1.000000  0.700000

             REPEAT   | SELECT   |  REVERSE  | REFLECT  | RANDOM ENTRY
             ---------------------------------------------------------
PITCH          YES    |   NO     |    NO     |   NO     |
LENGTH         YES    |   NO     |    NO     |   NO     |      NO
VELOCITY       YES    |   NO     |    NO     |   NO     |
```

Notice that this part lasts exactly the same length of time as the solo part (that is, 60 beats) and is sent to the same MIDI Channel, 1. It is also important that the Patch and Pan settings are the same so that this part does not interfere with the solo part in any way, other than changing the pitch bend.

For the occasional Kakko drum, a part is added, starting at beat 20, for which only 0 and 127 velocities are allowed. A Minimum Mean value of 0 with a Gradient of 0 and a fairly high λ value of 5 means that 0 velocities predominate and thus the drum only enters occasionally at such times as velocities of 127 do occur.

The relevant extract from the score is as follows:-

```
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
VELOCITY         0       127           0       127     0.000000  5.000000
```

The instrument chosen is Timpani (MIDI Patch 48), with a single pitch of A#. This part is of course sent to a different MIDI Channel (2).

The complete section of the score corresponding to this part is as follows:-

```
Sect  1 Part  3, Chan  2, Patch  48, Pan  64, BEATS     20 to      50
==============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           68        68          68                1.000000  0.500000
LENGTH (/  1)    1         1           1                1.000000  0.500000
VELOCITY         0       127           0       127     0.000000  5.000000

             REPEAT   | SELECT   |  REVERSE  | REFLECT  | RANDOM ENTRY
             ---------------------------------------------------------
PITCH          YES    |   NO     |    NO     |   NO     |
LENGTH         YES    |   NO     |    NO     |   NO     |      NO
VELOCITY       YES    |  YES     |    NO     |   NO     |

VELOCITY SELECTIONS:
   0 127
```

The Tsuridaiko drum part is constructed in a very similar way except that the λ value for the velocity is set to 6 instead of 5. This means that zero velocities are more predominant; that is, the drum enters less frequently. The instrument chosen is Taiko (MIDI Patch 117)

The complete section of the score corresponding to this part is as follows:-

```
Sect  1 Part  5, Chan  3, Patch 117, Pan  64, BEATS     40 to      60
==============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           67        67          67                1.000000  0.500000
LENGTH (/  1)    1         1           1                1.000000  0.500000
VELOCITY         0       127           0       127      0.000000  6.000000

             REPEAT   | SELECT   |  REVERSE  | REFLECT  | RANDOM ENTRY
             ------------------------------------------------------
PITCH          YES    |   NO     |     NO    |    NO    |
LENGTH         YES    |   NO     |     NO    |    NO    |       NO
VELOCITY       YES    |   YES    |     NO    |    NO    |

VELOCITY SELECTIONS:
   0 127
```

The remaining part of this first section is the accelerating Kakko drum sequence towards the end. To achieve an accelerando sequence we must produce a sequence of notes whose lengths are successively shorter. This is done by setting the Minimum Mean to the lowest value in the range (the shortest length) and having a gradient of less than one so that the values are "pulled down" towards the lower end of the range (see Section **5.3.2**, Example 2, for a discussion of this effect, but in relation to pitch). The lower the value of λ the more gradual will be the pull down effect. Here, length values are chosen across the complete range from 5/64 of a beat to 64/64 of a beat (that is, a whole beat). The starting length is set to 64/64 so that the length starts at one beat and gradually shortens thus producing the desired accelerando effect. A gradient of 0.8, being fairly close to one, means that the size of the initial jumps down in length from the starting length of one beat are fairly small, while a λ value of 2 gives a fairly strong tendency so that the lengths steadily shorten and, very importantly, once they reach the lower end of the range they tend to stay there.

The relevant extract from the score is as follows:-

```
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
LENGTH (/ 64)    5        64           5         64      0.800000  2.000000
```

This part can be sent on the same MIDI Channel as the previous Kakko drum part since it does not overlap it. It lasts for the final 10 beats of this first section.

The velocity of this part is set to 100 rather than the possible maximum of 127 because rapidly repeating drum beats (in "machine gun" fashion) tend to drown out other instruments.

The complete section of the score corresponding to this part is as follows:-

```
Sect  1 Part  4, Chan  2, Patch  48, Pan  64, BEATS     51 to      60
==============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           68        68          68                1.000000  0.500000
LENGTH (/ 64)    5        64           5         64     0.800000  2.000000
VELOCITY       100       100         100                1.000000  0.500000

             REPEAT   | SELECT   |  REVERSE   | REFLECT   | RANDOM ENTRY
             ------------------------------------------------------------
PITCH          YES    |    NO    |     NO     |    NO     |
LENGTH         YES    |    NO    |     NO     |    NO     |       NO
VELOCITY       YES    |    NO    |     NO     |    NO     |
```

Section 2

In order to provide continuity between this section and the first, the rapidly repeating Kakko drum continues for the first 5 beats.

To form the Sho backing, three separate parts have been used, playing the notes B, D and E. Each in fact just plays one long note for 125 beats, the entire length of the section. The first of these has the following score:-

```
Sect  2 Part  1, Chan  1, Patch 112, Pan  64, BEATS      1 to     125
==============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           71        71          71                1.000000  0.500000
LENGTH (/  1) 125       125         125                1.000000  0.500000
```

The other two Parts are exactly the same except that they play MIDI pitch values 74 and 76 and are sent to different MIDI Channels.

To achieve the rising and falling volume, a separate part sends volume change events to the same MIDI channel. In fact, each of the three main Sho parts has 2 separate volume change parts associated with it, the first of which forms a short crescendo as the instrument first enters, while the second controls the rising and falling volume thereafter. For the short crescendo, the full volume range of 0 to 127 is specified but a Minimum Mean of 5 with a gradient of less than 1 (0.961 in this case[6]) means that, from the specified starting value of 0, the volume values will tend to be pulled up towards the maximum value

[6] This value has been calculated specifically so that the line ends **exactly** at the maximum value of the range. This ensures that no Wraparound effect occurs so that once the volume reaches the top of the range it tends to stay there and doesn't jump back down to the lower end.

giving the required crescendo effect. A λ value of 2 makes this tendency fairly strong.

The relevant extract from the score is as follows:-

Parameter	Min	Max	MinMean	Start	Grad	Lambda
VOLUME	0	127	5	0	0.961000	2.000000

Note lengths are all 1/8 of a beat so that the volume rises quickly, and the velocity is set to zero so that no Note On or Note Off events are sent, just the volume change events.

The complete section of the score corresponding to this part is as follows:-

Sect 2 Part 2, Chan 1, Patch 112, Pan 64, BEATS 1 to 5

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	71	71	71		1.000000	0.500000
LENGTH (/ 8)	1	1	1		1.000000	0.500000
VELOCITY	0	0	0		1.000000	0.500000
VOLUME	0	127	5	0	0.961000	2.000000

This short crescendo lasts for the first 5 beats of the section. The second volume change part then takes over, lasting from beat 6 until the end of the section. It allows the volume to meander up and down between the MIDI volume values of 32 and 127.

The relevant extract from the score is as follows:-

Parameter	Min	Max	MinMean	Start	Grad	Lambda
VOLUME	32	127	32	90	1.000000	0.200000

The quite low λ value of 0.2 means that large volume jumps sometimes occur and, when this jump is from a low value to a high one, it gives the impression of a new attack point, even though only one long continuous note is actually being played, while the comparative rarity of such jumps means that the notes tend to be long and of varying length. This technique has been used here as it gives a much softer attack than playing a fresh note.

The complete section of the score corresponding to this part is as follows:-

Sect 2 Part 5, Chan 1, Patch 112, Pan 64, BEATS 6 to 120

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	71	71	71		1.000000	0.500000
LENGTH (/ 2)	1	1	1		1.000000	0.500000
VELOCITY	0	0	0		1.000000	0.500000
VOLUME	32	127	32	90	1.000000	0.200000

A similar pair of volume change parts is used to control the volume of the other two Sho parts.

The solo Hichiriki enters at beat 30 and lasts for 30 beats. Pitch selections are from the Hyojo mode and are allowed to meander over the range of pitches but a λ value of 2 means that jumps in pitch tend to be small. Successive repetitions of the same pitch have been disallowed.

To prevent an even rhythm occurring, note length selections are 4, 8, 9, 10, 11, 12, 13, 14, 15, 16 with a beat fraction of 4. Therefore, in addition to crotchets (4/4), semibreves (8/4) and breves (16/4), uneven note lengths occur, 13/4 for example.

Velocities of 0 and 127 only are allowed but a Minimum Mean Velocity of 127 and a zero gradient with a fairly high λ value of 3 mean that rests are infrequent.

The complete section of the score corresponding to this part is as follows:-

```
Sect  2 Part 11, Chan  5, Patch  71, Pan  40, BEATS     30 to      59
===============
Parameter      Min       Max       MinMean    Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           67        81          67        71     1.000000  2.000000
LENGTH (/  4)    4        16           4                1.000000  0.500000
VELOCITY         0       127         127               0.000000  3.000000

             REPEAT  | SELECT  |  REVERSE  | REFLECT  | RANDOM ENTRY
             ------------------------------------------------------
PITCH          NO    |   YES   |     NO    |    NO    |
LENGTH        YES    |   YES   |     NO    |    NO    |       NO
VELOCITY      YES    |   YES   |     NO    |    NO    |

PITCH SELECTIONS:
  67  69  71  73  74  76  79  81
LENGTH SELECTIONS:
   4   8   9  10  11  12  13  14  15  16
VELOCITY SELECTIONS:
   0 127
```

At the conclusion of the solo, three Hichiriki then enter, with their starting beats offset by one beat (they start at beats 60, 61 and 62 respectively). The parameter settings for these are very similar to the solo Hichiriki part but with two important differences:

(i) A smaller set of pitches has been selected so that the three parts tend to stay in the same register

(ii) The Length gradient is set to -1 so that longer note lengths tend to be followed by shorter ones (see Section **5.6.2**, Example 2)

The complete section of the score for the first of these three parts is as follows:

```
Sect  2 Part 17, Chan  5, Patch  71, Pan  40, BEATS     60 to     120
==============
Parameter      Min       Max       MinMean    Start     Grad      Lambda
-------------------------------------------------------------------------
PITCH           69        78          69        71    1.000000  2.000000
LENGTH (/  4)    4        16           4              -1.000000  2.000000
VELOCITY         0       127         127              0.000000  3.000000
            REPEAT  | SELECT  |  REVERSE  | REFLECT  | RANDOM ENTRY
            ------------------------------------------------------
PITCH         NO    |   YES   |     NO    |    NO    |
LENGTH        YES   |   YES   |     NO    |    NO    |       NO
VELOCITY      YES   |   YES   |     NO    |    NO    |

PITCH SELECTIONS:
  69  71  73  74  76  78
LENGTH SELECTIONS:
   4   8   9  10  11  12  13  14  15  16
VELOCITY SELECTIONS:
   0 127
```

The other two Parts are the same but they are sent to a different MIDI Channel and they have different Pan values in order to achieve separation of the three parts. Each of these parts is timed to end five beats before the end of the section so that the whole piece concludes with just the Sho notes playing.

The complete *Markov* program score for the piece can be seen in Appendix B, Section **B.1.2.**

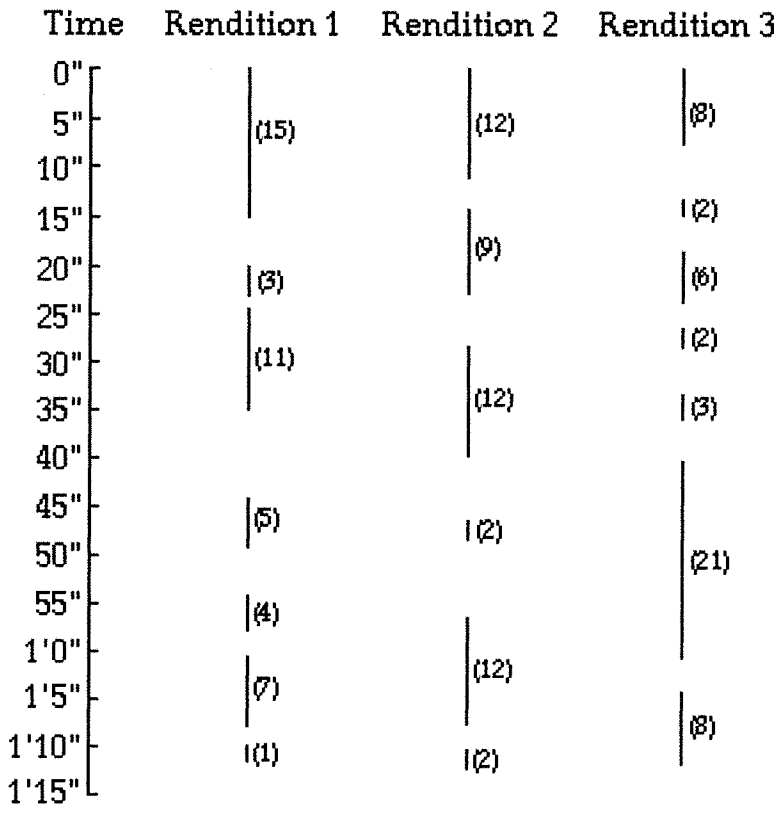
6.3.4 Discussion of the Results

The program-generated realisations of this piece generally work well although some realisations are more convincing than others. For the initial Ryuteki solo, should three or more successive notes from the mode occur in sequence (up or down), E, F#, G for example, then this can sound unconvincing. This event will tend to be infrequent due to the underlying probabilities and, over the three renditions generated by the program, it occurs a total of 4 times across all of the three 1'12" periods that the solo lasts, as follows:

Rendition 1:	D C# B	7" to 15"	
Rendition 2:	A B C#	15" to 23"	
Rendition 3:	D C# B	2" to 8"	and
	E F# G	46" to 55"	

Also unconvincing, but again infrequent due to the probabilistic construction, are long sequences of notes in the Ryuteki solo without a rest. The following diagram shows the periods of continuous note activity in the three renditions - the vertical lines show the start and end points of each of the periods, read against the vertical Time axis, while the figures in brackets

indicate the lengths of each of the sequences in seconds (for example, Rendition 1 begins with a 15 second continuous note sequence lasting from 0' to 15'):-



Notice that the two longest periods of note activity without a rest are 15 seconds, in Rendition 1, and 21 seconds, in Rendition 3, all others being 12 seconds or less. Only the 21 second sequence sounds disconcertingly long when listening to the pieces. In general, however, the algorithm is effective in producing such improvisational sequences.

The Kakko and Tsuridaiko drum attacks are only ineffective if they occur very frequently but this again is probabilistically extremely unlikely and does not happen at all in any of the three renditions produced here. The attack counts for the three Renditions are as follows:-

Kakko Drum: 3, 5 and 5 respectively (time period = 24" to 1'0")
Tsuridaiko Drum: 2, 1 and 2 respectively (time period = 48" to 1'12")

The accelerating Kakko drum sequence at the end of the first section, being constructed in a fairly deterministic way, is always effective.

In the second section, the Sho parts work well. The offset attacks of the Hichiriki, with their discordant effects, also work well and it is only when their registers all drift apart that they become unconvincing. Again, these parts have been constructed such that this is unlikely. Rendition 2 is very effective in this regard, with the three Hichiriki parts moving together, apart from a brief period

from 2'50" to 2'55" when one of the parts is noticeably higher than the other two, so that the required discordant effects are successfully obtained. Renditions 1 and 3 are less successful, with two of the three parts staying together but the other tending to move away from the other two: in Rendition 1 one part becomes much lower than the other two from 2'57" and remains so for the remainder of the piece, while in Rendition 3, from 2'47", one part persistently enters in a lower register than the other two.

The least convincing aspect is the pitch bend effect, used both for the Ryuteki solo in the first section and the Hichiriki parts in the second section. The main problem is that the bends will occur randomly at any point during the playing of a note whereas the live player will always tend to commence the bending of a note at its initial point of attack. This effect occurs quite frequently during all three renditions. Two more readily apparent examples, occurring in the Ryuteki solo, are in Rendition 2, where a note lasting from 7" to 12" suddenly bends upwards after 3", and in Rendition 3, where a note lasting from 57" to 1'0" suddenly bends downwards after 2". The program could achieve the required degree of synchronisation by using a deterministic note length sequence, such that the attack points of the notes forming the solo are completely fixed and the pitch bends are made to occur at the start of each note, but then the essential freedom of the solo would be lost because the algorithm would be allowed no probabilistic variation. Also, occasionally, the pitch bend can cause the notes to drift away from the true pitch so that the sense of modal identity is lost. Since, probabilistically, the average pitch bend tends to be zero, this is fairly rare. The most noticeable occurrence of this effect is in Rendition 2 at 27" where, after a period of silence, the Ryuteki solo enters with A# followed by G# and thereafter tends to stay a semitone above the correct pitch values for the remainder of the solo. Minor occurrences are evident in Rendition 1 at 44", where a G# is played and in Rendition 3 at 18" where an A# is played, followed by a G#, but in both these cases the drift is quickly corrected. For all that, the pitch bend effect nevertheless adds an essential character to the piece.

6.4 BACH HARPSICHORD MUSIC

6.4.1 Introduction

In this attempt to emulate a piece of Bach harpsichord music, there are two principal objectives:

1. To demonstrate the ability of the algorithm to introduce probabilistic and deterministic control at a fine level of detail, and the compositional techniques which can be employed to achieve this.
2. To show the strengths and weaknesses of the algorithm in emulating a style of music which has a precise harmonic and melodic structure.

It must be stated straight away that to allow the algorithm to produce long note sequences and expect that the necessary structural precision will occur is just not feasible. By the probabilistic nature of the algorithm it is possible that short, convincing passages may occur from time to time, but not with a sufficiently high frequency for the results to be anywhere near satisfactory. The likelihood of convincing results diminishes even further if there is to be harmony and counterpoint between separate musical lines.

To reproduce the character of a piece of Bach harpsichord music, the piece must be constructed from phrases consisting of the correct number of equal length bars and there must be a definite harmonic structure, as the piece moves through precise chord sequences. In addition, each chord will last for a precise number of bars, or number of beats within a bar. Further, definite melodic shapes are an essential part of the character of the music.

The approach adopted here, therefore, is to algorithmically control the piece on a **bar by bar** basis.

The intention is to use the program to model the first 16 bars[7] of the Courante from the Bach Suite BWV 813. Reference was made to an original score (Bach 1980) and also to a recording taken from the compact disc *Little Notebook for Anna-Magdalena Bach (selections)*, Analekta Fleurs de Lys, 1995, performed by Luc Beauséjour (compact disc number FL 2 3064), for the purpose of providing a means to compare aurally the results produced by the program with an actual performance of the original piece.

In order to maintain the harmonic structure, and to keep the precise melodic shaping in a small number of important phrases, a relatively high degree of determinism is used in the construction of the version produced from the algorithm. This means that there is a high probability that, as long as the key stylistic elements of the original style have been reliably identified, successive renderings produced by the algorithm will be faithful to the original style, and this is indeed the case here. However, if the degree of determinism is too high then there will be little variation between successive renderings, and, while this would not necessarily be an invalid use of the algorithm, it would defeat the object here, which is to be able to produce many different versions from the same single set of algorithm parameters, all in the desired style.

---

[7] The first section of the original Courante is actually 24 bars in length rather than 16 but, since 16 bars are more than sufficient to satisfy the objectives of this exploration, only the first 16 bars have been studied here.

Therefore, what has been attempted here is to achieve a balance between determinism and probabilistic variation such that the degree of determinism is sufficiently high to achieve stylistic consistency, whilst the degree of probabilistic variation is high enough that significant variations occur between successive renderings. In this way, the aim is to explore the extent to which this algorithm can reproduce melodic elaboration within a constrained tonal and harmonic framework.

The accompanying compact disc contains six separate program-generated realisations in WAV-formatted audio files called "Bach1.wav" through to "Bach6.wav" and the corresponding scores, in staff notation, for the six realisations can be found in Appendix C, where the original score is also given. The floppy disk contains the program-generated piece as a *Markov* program Composition File called "Bach" for playback from the *Markov* program. All files may be found in the "Style Emulation" directory.

6.4.2 Identifying the Key Elements

For this example, a careful study of the original score is essential so that reliable deductions can be drawn as to how to control the parameters of the algorithm so as to reproduce the style as faithfully as possible. In this section, the key findings of the analysis of the original score are discussed. The key elements identified are lettered below and will be referred to later in this chapter when the method of construction of the piece is discussed.

a. Time Signature
The piece is in 3/4 time.

b. Harmonic Structure
It begins in C minor and, at the end of the 16 bars under investigation, modulates to E^{b} major (the Courante as a whole returns to C minor).
The harmonic structure of the 16 bars, as implemented in the *Markov* program emulation[8], is as follows:-

[8] This is a deliberate over-simplification of the harmonic structure of the original piece in order to create a framework within which the algorithm is able to generate a degree of embellishment.

Bar	Chord(s)
1	c
2	c
3	f
4	G
5	c
6	f
7	B^b
8	E^b
9	E^b
10	f
11	g
12	A^b
13	B^b
14	E^b
15	E^b (1 crotchets), B^b (2 crotchets)
16	E^b

c. Beginning of the Piece

The first bar is preceded by a three note lead-in: G A^b F.

d. Texture

There are two separate musical parts, a melody line and a bass line.

e. The Bass Line

The bass line provides a solid harmonic structure. It consists mainly of crotchets, which form arpeggiations of the underlying chord, apart from four bars where quavers are introduced in order to provide a counterpoint to the melody.

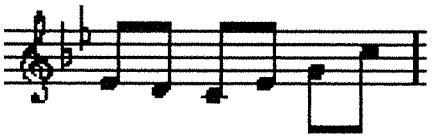
f. The Melody

The melody consists almost entirely of quavers apart from three crotchets which occur at phrase ends. Two or more consecutive occurrences of the same pitch **never** occur.

In order to investigate the way in which the construction of the melody supports the harmonic structure, two simple statistical analyses have been carried out:

Analysis I. For each of the chords making up the chord sequence which forms the harmonic structure of the piece, this analysis counts the relative frequencies

of the diatonic notes in the melody for the duration of the chord, numbering the notes from the root of the chord, the root being note 1. For example, the melody in bar 2 of the original score is as follows:



The underlying chord is C minor, so, since C (note 1) occurs twice[9], D (note 2) occurs once, E^{b} (note 3) occurs twice and G (note 5) occurs once, this bar will contribute the following amounts to the overall totals for Analysis I:

1	2
2	1
3	2
4	0
5	1
6	0
7	0

Analysis II. This analysis counts of the total number of times that each of the numbered diatonic notes occurs as the **first** note of the sequence of notes in the melody occurring against the chord (usually the first note of the bar, apart from bar 15 where two chords occur). In the above example, the first note is E^{b} so this bar will add one to the total for note 3 in Analysis II. The complete results for the 16 bars are as follows:

Analysis I

Note	No of Occurrences	%
1	18	21.95
2	5	6.10
3	15	18.29
4	3	3.66
5	30	36.59
6	2	2.44
7^{b}	8	9.77
7	1	1.22

[9] No attempt is made in this analysis to distinguish between notes of the same pitch value which are an octave apart, but, of course, when the reproduction of the piece is structured using the *Markov* program, both pitches will be included in the selected pitch set where appropriate.

Analysis II

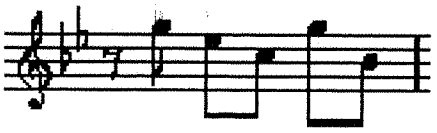
Note	No of Occurrences	%
1	0	0.00
2	1	5.88
3	6	35.29
4	1	5.88
5	9	52.94
6	0	0.00
7	0	0.00

From Analysis I, we can see that fully 76% of the notes are numbers 1, 3 or 5 in relation to the underlying chord; that is, they are from the corresponding tonic triad. A further 10% are the flattened 7th - this, of course, is not part of the diatonic scale but forms part of the 7th chord corresponding to the underlying chord. Of the five note 2's which occur, three are within the first three bars while a fourth forms part of a five note step by step descending sequence at the very end of the 16 bars.

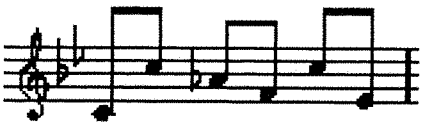
From Analysis II, we can see that more than half of the first notes of the sequences corresponding to the underlying chords are number 5 in relation to the chord, while the bulk of the remainder are number 3.
As we shall see in the next section, these results will be a very important factor in the decisions made regarding the algorithm parameter settings.

g. Melodic Shaping

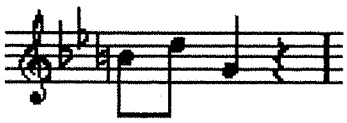
The melody in bars 5, 7 and 9 of the original score has the following precise, arpeggiated, shape (this example is bar 5, based on a C minor chord):



that is, a quaver rest followed by degrees 5, 3, 1, 5 (an arpeggiated tonic triad) and finishing with the flattened 7th, while bars 10, 11, 12 and 13 have the following almost identical shape (this example is bar 10, based on an F minor chord):



which has the initial rest replaced by the 5th degree an octave below the second note in the bar. In the same vein, bars 4, 6 and 8 have the following shape (this example is bar 4, based on a G major chord):



which is another arpeggiated tonic triad shape, these bars occurring at phrase ends. The remainder of each of these three bars is completed by a three quaver contrapuntal line in the bass.

These melodic shapes explain the preponderance notes 1, 3 and 5 in the results of Analysis I above, and of notes 3 and 5 in Analysis II.

h. Ornamentation

The score has little notated ornamentation other than a small number of trills and turns (for the 16 bars studied here there is one trill). However, interpretations of this style of music generally introduce embellishments and the performer on the recording referred to here adds trills and turns liberally.

6.4.3 Constructing the Piece Using the Algorithm

The piece is built in two Sections, one for the first eight bars and another for the second eight bars. This division into Sections has been done to avoid exceeding the maximum number of Parts[10] per Section allowed by the *Markov* program (20 Parts). This has been necessitated by the fine level of control being used: usually a separate Part for each bar and occasionally a Part for a subdivision of a bar, so that Parts are being used more frequently than in the previous examples[11]. The following two tables show, for each bar of the piece, which Section/Part in the *Markov* program realisation implements it[12]:

---

[10] The reader is reminded that "Part", in this context, is a specific construct within the *Markov* program, which allows the composer to specify parameter settings to produce a single, monophonic, line. Polyphony is achieved by having two or more Parts play simultaneously.

[11] Nevertheless, it can be convenient to logically divide a piece into Sections even though some Sections may consist of significantly less than 20 Parts, particularly as the Section Sequence feature can then be used to produce a specific form e.g. Rondo (ABACA).

[12] The values in the Right Hand and Left Hand columns refer to the Part numbers within the Section. Thus, for example, the right hand part in bar 4 is implemented by Part 6 in Section 1.

Section 1

Bar	Right Hand	Left Hand
1	2,3	11
2	4	12
3	5	13
4	6	14
5	7	15
6	8	16
7	9	17
8	10	18,19

Section 2

Bar	Right Hand	Left Hand
9	1	8
10	2	9
11	3	10
12	4	11
13	5	12
14	6	13,14
15	7,15	14
16	16,17	14

The tempo is set to 384 beats per minute and, for convenience, each beat will be one quaver.

The strategy adopted here is to fix the bass completely, copying the original score, in order to provide a solid harmonic structure. The algorithm is then used to generate varying treble lines, on successive playings, against this fixed bass. While most of the treble line is allowed to vary, a small number of bars have been reproduced exactly so as to maintain the original sense of phrasing. Specifically, bars 4, 6, 8, 15 and 16, together with the three note introductory sequence, have been fixed. The remaining 11 bars have been allowed to vary, subject to some degree of shaping control via the algorithm parameter settings, as explained in detail below. In the musical scores of the pieces produced by the program given in Appendix C, those passages which are variable are clearly marked with rectangular outlines. It is obviously a limitation of the algorithm that, in attempting to generate music within the compositional context of such a style, it is only possible to exploit the variation-generating qualities of the algorithm to a relatively small degree. On the other hand, it is a strength of the algorithm that, by building the piece a bar at a time,

it is capable, by virtue of the diagonal line mechanism, of reproducing exactly the required sequences despite its probabilistic nature.
The remainder of this section refers to the lettered key elements identified in the previous section and discusses their implementation using the parameters of the algorithm.

a. Time Signature

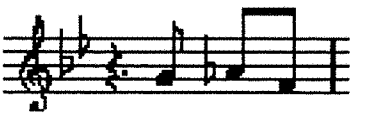
The division of the melody into equal length arpeggiations of distinct chords, and the precisely structured bass line, both discussed in detail below, imply a firm 3/4 time signature.

b. Harmonic Structure

The harmonic structure is maintained by making appropriate pitch selections so as to imply the appropriate chord, again discussed in detail below.

c. Beginning of the Piece

The original work begins with the following introductory three note sequence:



It was decided to reproduce this introductory sequence precisely so as to give the piece a sound beginning. The sequence has a single Part all to itself. The required pitch selections have been made (MIDI values 65, 67 and 68). The Minimum Mean value has been set to 67 (G). Since this is **second** pitch in the range of three values, there will be a step by step rising tendency (see Section **5.4.2**, Example 1); that is, F G A^{b}. A λ value of 20 **guarantees** that this will occur. With Wraparound turned on, when A^{b} is reached the sequence will return to F and begin to rise again. Setting the starting pitch to G (67), the note length to 1 (= a quaver) and the length of the Part to 3 beats, so that only three quavers will occur, produces the required three note pattern. This technique is used elsewhere in the piece when similar three note patterns are required. The ability to obtain this type of patterning is a direct consequence of the application of modulo arithmetic, inherent in the diagonal line algorithm.

The relevant extract from the score is as follows:

```
Sect  1 Part  1, Chan  1, Patch   7, Pan  64, BEATS      1 to       3
==============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           65        68          67         67     1.000000 20.000000
LENGTH (/  1)    1         1           1                 1.000000  0.500000
```

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	NO	YES	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

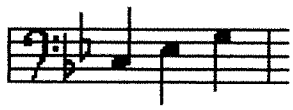
PITCH SELECTIONS:
65 67 68

d. Texture

The melody/bass line separation occurs naturally by virtue of the fact that the Parts which produce the melody are built from pitch selections from a higher register than those Parts which produce the bass line. In addition however, the melody Parts are sent to a separate MIDI channel (1) from the bass Parts (2). While this is not strictly necessary, it does have the advantage that the notation software used here to produce the scores (Emagic Logic™ 4.1) can be instructed to separate the treble and bass clefs according to the different MIDI channels. It would also allow stereo panning to be applied to the two MIDI channels to reinforce the separation although this has not been done here.

e. The Bass Line

In order to provide a solid harmonic foundation to the piece, the bass line has been constructed in an entirely deterministic manner, so as to be the same as the original score. Thus, for example, bar 1:



is formed by selecting the three pitches in question, setting the Minimum Mean Pitch to be the second pitch, with a λ value of 20, so as to produce a definite stepwise rising pattern, setting the note lengths to 2 (that is, 2 quavers = 1 crotchet) and fixing the starting pitch to be C (48), the relevant extract from the score being:

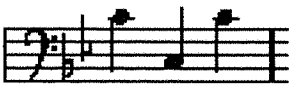
Sect 1 Part 11, Chan 2, Patch 7, Pan 64, BEATS 4 to 9

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	55	51	48	1.000000	20.000000
LENGTH (/ 1)	2	2	2		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	NO	YES	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

PITCH SELECTIONS:
48 51 55

whilst bar 2:



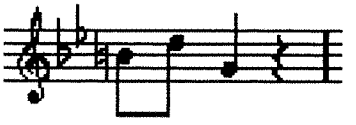
is achieved very simply by selecting the two C pitches, setting the starting pitch to be the higher C and disallowing repeats, the relevant extract from the score being:

```
Sect  1 Part 12, Chan  2, Patch   7, Pan  64, BEATS     10 to      15
===============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           48        60          48         60     1.000000  0.500000
LENGTH (/  1)    2         2           2                1.000000  0.500000

             REPEAT   | SELECT   |  REVERSE   | REFLECT   | RANDOM ENTRY
             ------------------------------------------------------------
PITCH          NO     |   YES    |     NO     |    NO     |
LENGTH        YES     |    NO    |     NO     |    NO     |       NO
VELOCITY      YES     |    NO    |     NO     |    NO     |

PITCH SELECTIONS:
  48  60
```

f. The Melody

Since, in the original, two or more consecutive occurrences of the same pitch never occur, all the Parts which form the melody have been set to disallow pitch repetitions.

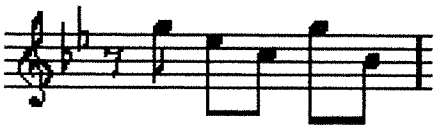
Throughout the 16 bars, all but three of the notes are quavers, the remaining three being crotchets which occur at phrase ends, for example, bar 4 of the original:



However, it is not possible to set algorithm parameters for note lengths in a single Part such that precisely two quavers occur followed by one crotchet - indeed, by its very nature, the algorithm is not designed to be able to produce arbitrarily chosen melodic or rhythmic sequences. To achieve the above rhythm, it would be necessary to use 2 Parts for this bar, one for the two quavers and one for the crotchet, at which point the compositional process becomes unacceptably laborious. Therefore, here all note lengths have been set to one (=

a quaver) for all the melodic Parts, so the program replaces the crotchet with a quaver.

Analysis I above shows that, in the original, 76% of the notes are numbers 1, 3 or 5 counted from the root of the underlying chord while a further 10% are the flattened 7th. Therefore, the pitch selection mechanism provide by the *Markov* program is used to reflect this. Thus, for example, for bar 5, which in the original is as follows:



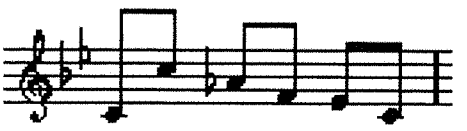
just the pitches corresponding to the C minor triad, plus the flattened 7th, are selected (MIDI values 70, 72, 75 and 79). It was also noted that of the five notes 2's which occur, three are within the first three bars and so the set of selected pitches for these bars has been expanded to include this pitch as well.

Analysis II above shows that more than half of the first notes of the sequences corresponding to the underlying chords are note 5, while the bulk of the remainder are note 3. Since the *Markov* program allows the composer to specify the starting pitch in a Part, this has been used to reflect the starting pitches in the bars of the original. Thus, for example, for bar 5 the starting pitch is set to G (MIDI value 79).

g. Melodic Shaping

As previously discussed, eight of the bars in the original have a precise arpeggiated shape, bar 10 for example:

In this attempt to emulate the style of the original, the intention is to maintain something of the character whilst allowing the probabilistic nature of the algorithm to introduce a degree of variation so that each piece produced is different. What has been done here is to set the diagonal line parameters so that this bar **tends** to have the following shape[13]:



[13] Note that this is a personal compositional decision taken by the author which it was felt was appropriate and vindicated by the results.

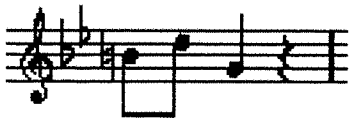
To achieve this, the corresponding five pitches are selected and the Minimum Mean pitch is set to the highest pitch value, with a gradient of one, so as to achieve a stepwise descending tendency through the selected pitch values (see Section **5.4.2**, Example 12). The starting pitch value is set to the lower C (60), and since Wraparound, rather than Reflect, is in effect, the second pitch played will tend to be highest pitch in the range, after which it will tend to descend stepwise, as required. A λ value of 2 provides a quite strong tendency for this melodic shape to occur while still allowing a degree of variation.

The relevant extract from the score for bar 10 is as follows:-

```
Sect  2 Part  2, Chan  1, Patch   7, Pan  64, BEATS      7 to      12
==============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           60        72          72         60     1.000000  2.000000
LENGTH (/  1)    1         1           1                1.000000  0.500000

             REPEAT   | SELECT   |  REVERSE   | REFLECT   | RANDOM ENTRY
             ---------------------------------------------------------
PITCH          NO     |   YES    |     NO     |    NO     |
LENGTH        YES     |    NO    |     NO     |    NO     |       NO
VELOCITY      YES     |    NO    |     NO     |    NO     |

PITCH SELECTIONS:
  60   63   65   68   72
```

The same technique has been employed for the other bars which have a similar melodic shape in the original but with, of course, different pitch selections to reflect the underlying chord.

For those three bars of the original with the following shape occurring at phrase ends, bar 4 for example:



it was decided to reproduce these bars exactly, so as to maintain the original sense of phrasing. This shaping is achieved simply by selecting the three pitches in question, setting the diagonal line parameters to produce a stepwise rising sequence, as shown in the discussion of the bass line construction above, specifying the starting pitch to be the middle of the three values and setting λ to 20. The Wraparound effect will result in the highest of the three pitches being immediately followed by the lowest pitch.

For bar 6, for example, the relevant extract from the score is as follows:

```
Sect  1 Part  8, Chan  1, Patch   7, Pan  64, BEATS     34 to      36
==============
Parameter      Min       Max       MinMean    Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           65        72          68         68     1.000000 20.000000
LENGTH (/  1)    1         1           1                 1.000000  0.500000

             REPEAT   | SELECT   |  REVERSE  | REFLECT  | RANDOM ENTRY
             ---------------------------------------------------------
PITCH          NO     |   YES    |     NO    |    NO    |
LENGTH        YES     |    NO    |     NO    |    NO    |       NO
VELOCITY      YES     |    NO    |     NO    |    NO    |

PITCH SELECTIONS:
  65  68  72
```

For the remaining bars, the melody has been allowed to vary more freely, with λ values of 0.5, apart from the six note step by step descending sequence at the end (formed from the last three quavers of bar 15 and the first three quavers of bar 16) for which the parameters have been set so as to reproduce it exactly.

h. Ornamentation

Although the algorithm is capable of recreating ornamentation, liberal use of these effects does tend to use up Parts quickly, so, although the performer on the recording referred to here adds a number of turns, just one turn has been added to the piece produced by the program, in bar 16, to demonstrate the technique. The turn occurs at a note whose pitch is G, so this is replaced with three quick notes, G A^b G, adding to one quaver's length in total (by setting the length of the Part to 1 beat and the note length to 1/3 beat). The relevant extract from the score is as follows:

```
Sect  2 Part 17, Chan  1, Patch   7, Pan  64, BEATS     43 to      43
==============
Parameter      Min       Max       MinMean    Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           67        68          67         67     1.000000 20.000000
LENGTH (/  3)    1         1           1                 1.000000  0.500000

             REPEAT   | SELECT   |  REVERSE  | REFLECT  | RANDOM ENTRY
             ---------------------------------------------------------
PITCH          NO     |   YES    |     NO    |    NO    |
LENGTH        YES     |    NO    |     NO    |    NO    |       NO
VELOCITY      YES     |    NO    |     NO    |    NO    |

PITCH SELECTIONS:
  67  68
```

The complete rendition is formed by using the Section Sequence feature (see Appendix A, Section 3.7) to play Section 1 followed by Section 2.

The complete *Markov* program score for the piece can be seen in Appendix B, Section **B.1.3**.

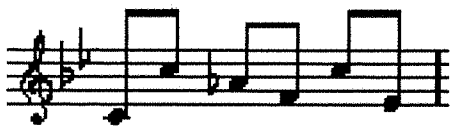
6.4.4 Discussion of the Results

The bass line, having been constructed deterministically so as to be identical to the original, provides a sound harmonic structure to the piece. Indeed, as is evident from the above explanations of the techniques used to construct it, the algorithm lends itself well to this "walking" style bass line. In the melody line, the various phrases are built mainly on appropriate triads, possibly with the flattened seventh added, so that the required chord is usually implied, and this, together with the structured bass line, means that, in general, the correct chord progression is achieved. There are, however, occasional exceptions to this. These exceptions occur when a pitch sequence happens to occur which implies a different chord to that intended. Two examples of this effect are given below:-

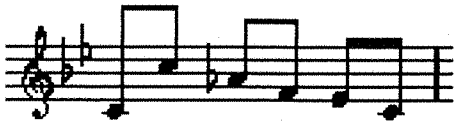
(i) In Rendition 3, Bar 1, the occurrence of F followed by C tends to imply an F minor chord rather than the intended C minor chord.
(ii) In Rendition 6, Bar 14, the pitch sequence C G C G tends to imply a C minor chord rather than the intended E^b major.

For obvious reasons, this effect does not occur when the pitch selections are taken just from the required chord. It is only when the melody line is allowed more freedom, through the introduction of pitches other than those contained in the chord, that the possibility of this effect arises. However, its occurrence is infrequent and the effect is one of an occasional "glitch" rather than a serious deviation from the correct harmonic progression.

For the melody, recall that for the eight bars which, in the original, had the following shape:

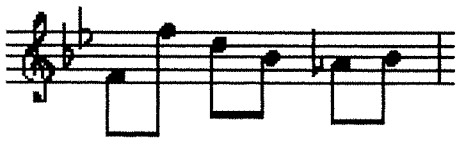


the parameter values were set so as to **tend** to produce:

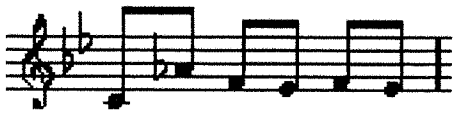


In the renditions which resulted from the program, a variety of different shapes occurred. Mathematically, 1024 different shapes could result for each such bar but since the parameters have been set to produce a tendency for a particular shape rather than allowing the pitches to occur at random, variants of the

second shape above are more likely to occur. Thus not only did the "controlling" shape itself occur (Rendition 6 Bar 11), but so also did



(Rendition 1 Bar 13)

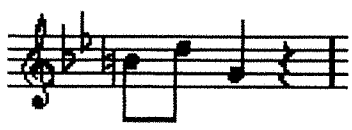


(Rendition 4 Bar 10)

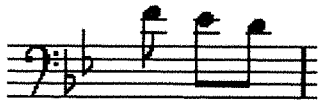
and many more. One limitation of the algorithm, however, is that it is not possible for the **same** shape to be maintained across a chord progression through a sequence of bars, unlike the original.

A further limitation is that is not possible to control what happens at the junction between consecutive Parts or, in this case, bars, so that the melody may not always flow smoothly through consecutive bars. Thus, for example, even though successive repetitions of the same pitch have been disallowed in the parameter settings for each bar, it is possible the last pitch in one bar may be the same as the first pitch in the following bar. However, due mainly to the melodic shaping control introduced in the parameter settings, this occurs just twice through all three renditions, Rendition 3 from bars 1 to 2 and Rendition 5 from bars 2 to 3. It is also possible that a disconcertingly large jump in pitch may occur from the last pitch in one bar to the first pitch in the following bar, for example the jump from C down to $E^{\flat}$ in Rendition 1 bars 1 to 2 and the jump from F down to G in Rendition 3 from bars 13 to 14. Again, these occur as very occasional "glitches".

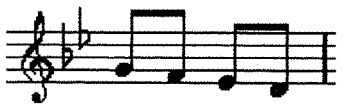
As discussed earlier, the algorithm, by its very nature, is not able to produce arbitrary rhythmic structures. Thus, for example, the following, bar 5, of the original:



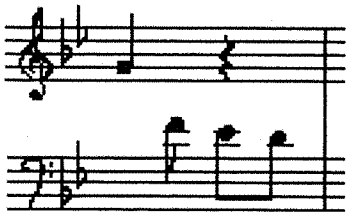
is approximated as three quavers. This tends to lessen the effect of the counterpoint in the bass line. For example, the crotchet in the original above is followed in the bass line by:



but because the piece produced by the algorithm replaces the crotchet in the melody line with a quaver this tends to be heard as:



rather than:



Finally, in addition to the limitations discussed above, the amount of labour involved in constructing this short piece from such small sequences is prohibitive; one need only compare, by referring to the *Markov* program scores in Appendix B, the amount of parametric data required compared to the Steve Reich Phase Music piece to appreciate this. It must be conceded, therefore, that the algorithm is not really appropriate for modelling such styles.

6.5 DANCE MUSIC

6.5.1 Introduction

This style of music, around which the present day "Club" dance culture is built, is characterised by simple, repetitive rhythms overlaid with short melodic fragments. There are a number of genres, for example: "Drum and Bass", which is predominantly percussive, "Ibiza", which has a Latin feel and "Trance", which has a more "dreamlike" atmosphere through the use of synthesized orchestral and electronic sounds. The piece produced here falls into the latter category. It is an original composition but inspiration was taken from the compact disc *Trance Mix '99 - A Spiritual Journey Through Time and Space,* mixed by Richard Evans at Wise Buddah, Virgin Records Ltd., 1999 (compact disc number 7243 8 48332 2 0).

This style demonstrates, in particular, the ease with which the algorithm can be used to built percussive rhythmical structures, emulating a "drum machine".

The accompanying compact disc contains the program-generated piece as a WAV-formatted file called "Dance.wav". The floppy disk contains the piece as a *Markov* program Composition File called "Dance" for playback from the *Markov* program. Both files may be found in the "Style Emulation" directory.

6.5.2 Identifying the Key Elements

a) Percussive Rhythms

A small number of percussion instruments are used in this style of music, usually a Bass Kick Drum and Closed Hi-Hat, and possibly a Hand Clap sound, a Snare Drum and a Cymbal. Each of these sounds repeats at a strict beat interval, usually every 1, 2 or 4 beats, with their respective attack points offset so as to build a layered percussive rhythmic structure.

b) Melodic Structure

Melodic lines typically consist of short, simple fragments, typically no more than 32 beats in length, which repeat continuously, possibly with some small variation, pitch level for example.

c) Bass Line

Bass lines are also very simple, usually a short fragment which repeats continuously and often just a single repeating bass note.

d) Breaks

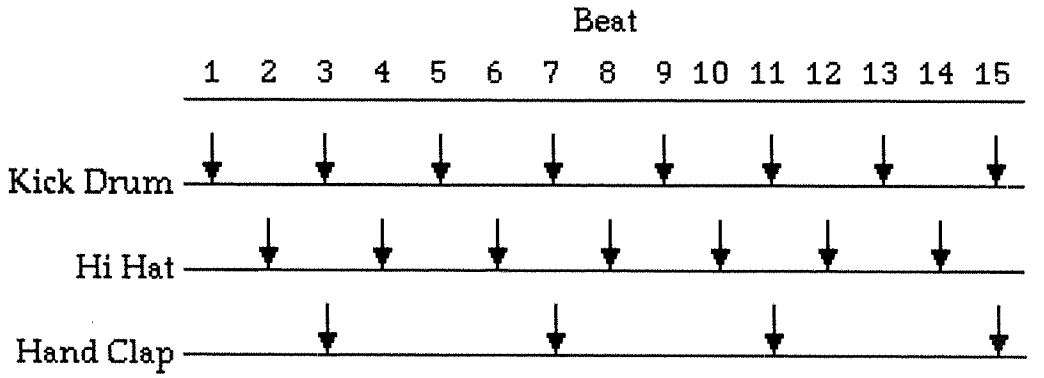
"Breaks" are definite changes which take place regularly and usually consists of the addition and/or removal of a melodic line, percussion instrument or sampled sound (vocal or instrumental), or a change in the melodic or bass lines. Breaks occur strictly at a multiple of 8 beats, usually every 16, 32 or 64 beats. A common feature is for a short, rapid crescendo snare drum sequence to occur leading up to a break.

It is the unrelenting succession of breaks which bring about a progressive musical development, possibly over a period of a number of hours.

6.5.3 Constructing the Piece Using the Algorithm

a) Percussive Rhythms

In the piece produced by the algorithm, three percussion sounds are used - Kick Drum, Hi Hat and Hand Clap - according to the following pattern:-



This is achieved by having a separate Part for each sound with, as specified by the MIDI standard for Drum Set sounds, each being sent to MIDI channel 10 with the pitch set according to the required sound (36 for Kick Drum, 42 for Hi Hat and 39 for Hand Clap). In addition, the initial starting times for each of the three sounds are offset so as to build up the percussion progressively, and staggered according to the required pattern as shown in the above diagram, the Hi Hat beginning at beat 34, the Hand Clap at beat 67 and the Kick Drum at beat 129.

For each Part, it is simply a case of setting the pitch to the required value, setting the note length to be the number of beats between each occurrence of the sound and specifying the appropriate starting beat. Thus, for example, the Hand Clap sound occurs at an interval of 4 beats so the note length for this Part is fixed at 4 beats[14].

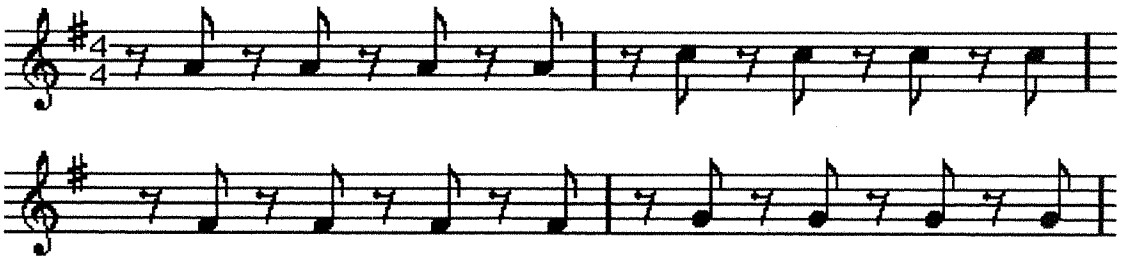
The relevant extract from the score is as follows:-

```
Sect  1 Part  3, Chan 10, Patch   0, Pan  64, BEATS     67 to     248
===============
Parameter      Min       Max       MinMean    Start     Grad      Lambda
-------------------------------------------------------------------------
PITCH           39        39          39               1.000000  0.500000
LENGTH (/  1)   4         4           4                1.000000  0.500000
```

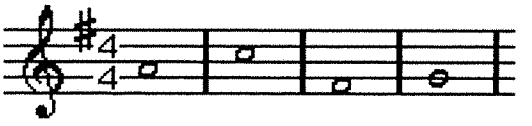
[14] Strictly speaking, this is musically incorrect since it implies that each note will last for a full 4 beats and only finish just before the next one begins. However, since Drum Set sounds are very short with no sustain, this is not an issue - they will last for as long as the sound itself lasts, regardless of the note length setting, provided it is at least as long as the duration of the sound of course.

b) Melodic Structure

There are two melodic fragments. The first, which uses the "Saw Wave" sound, is as follows:-



As has been discussed before, using the algorithm to construct precise melodies is usually impossible, as it is not designed for this purpose. However, in this instance the melody above can be generated from two simultaneous Parts. The first Part plays one **continuous** note in each bar:-



Since this melody has a simple stepwise movement it is easily generated as we have seen previously, the relevant extract from the score being as follows:-

```
Sect  1 Part 10, Chan  3, Patch  82, Pan  64, BEATS    257 to     448
==============
Parameter      Min       Max       MinMean    Start     Grad      Lambda
-------------------------------------------------------------------------
PITCH           66        72          67        69    1.000000 20.000000
LENGTH (/  1)    8         8           8              1.000000  0.500000
VOLUME           0         0           0              1.000000  0.500000

             REPEAT  | SELECT  |  REVERSE  | REFLECT  | RANDOM ENTRY
             ------------------------------------------------------
PITCH          YES   |   YES   |    NO     |   NO     |
LENGTH         YES   |    NO   |    NO     |   NO     |      NO
VELOCITY       YES   |    NO   |    NO     |   NO     |

PITCH SELECTIONS:
  66  67  69  72
```

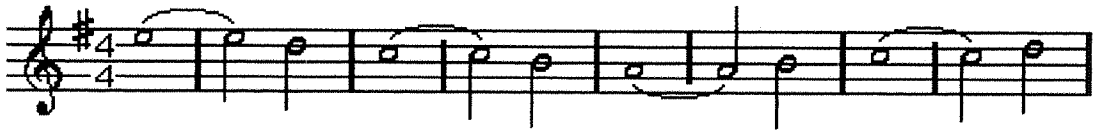
The second Part sends only **MIDI volume change** events to the **same** MIDI channel as the first Part, such that the volume is zero on quavers 1, 3, 5 and 7 of each bar and non-zero on quavers 2, 4, 6 and 8, this being achieved by causing the volume to alternate between, in this case, 0 and 80:-

```
Sect  1 Part 11, Chan  3, Patch   0, Pan  64, BEATS    257 to     448
==============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           60        60          60                1.000000  0.500000
LENGTH (/  1)    1         1           1                1.000000  0.500000
VELOCITY         0         0           0                1.000000  0.500000
VOLUME           0        80          80          0   -1.000000 20.000000
```

Thus, the note only actually sounds every other quaver as required. In fact, only the first note of each bar is a genuine attack point but because the Saw Wave sound has infinite sustain the results are successful.

This melody is joined subsequently by a similarly constructed one which harmonises at intervals of a third.

The second melodic fragment, played on Synthesized Strings, is as follows:-



This stepwise pitch movement is, again, easily constructed and the rhythm is obtained simply by selecting just the two note lengths involved and disallowing note length repeats:

```
Sect  2 Part  6, Chan  3, Patch  52, Pan  64, BEATS      1 to     128
==============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           57        64          64          64    1.000000 20.000000
LENGTH (/  1)    4        12           4          12    1.000000  0.500000
             REPEAT   | SELECT   |  REVERSE  | REFLECT  | RANDOM ENTRY
             ---------------------------------------------------------
PITCH          YES    |   YES    |    YES    |    NO    |
LENGTH          NO    |   YES    |     NO    |    NO    |       NO
VELOCITY       YES    |    NO    |     NO    |    NO    |

PITCH SELECTIONS:
  57  59  60  62  64
LENGTH SELECTIONS:
   4  12
```

c) Bass Line

The bass line is a single repeating note, occurring off the main beat. As with the percussion Parts, it is simply constructed by fixing the pitch and note length to the required values (MIDI pitch 28 and 2 beats respectively) and choosing an appropriate sound.

The relevant score extract is as follows:-

```
Sect  1 Part  8, Chan  1, Patch  25, Pan  64, BEATS    130 to     248
==============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           28        28          28                1.000000  0.500000
LENGTH (/  1)    2         2           2                1.000000  0.500000
RELEASE         16        16          16                1.000000  0.500000
```

Note that the Release has been set to a low value so that each bass note cuts off quickly, thereby giving a "snappy" feel, rather than allowing it to sustain for a full 2 beats.

d) Breaks

Breaks are easily built into the piece just by setting the various Part start and end times to precisely the right points. The Break structure of the piece is as follows:

Break	Beat No	Beat Interval[15]	Event(s)
	1		Piece commences with a continuously repeating Saw Wave note
1	17	16	Hi Hat enters
2	33	16	Hand Clap enters
3	65	32	Bass and Kick Drum enter
4	129	64	1st melodic fragment enters[16]
5	161	32	Harmonising melodic fragment enters
6	193	32	Bass and Percussion drop out 1st melodic fragment drops out
7	225	32	Bass and Percussion re-enter, 2nd melodic fragment enters
8	289	64	Bass and Percussion drop out 2nd melodic fragment drops out
9	305	16	Crescendo Snare Drum roll ends piece

[15] The number of beats per minute for this piece is set to 274 in order to achieve the resolution necessary to build the rhythmic structure and, strictly speaking, the beat intervals, as set in the composition, are double the values shown here. However, the values shown in this table feel more natural when counting.

[16] In fact, the Bass and most of the percussion drop out for 4 beats before this break. However, this just forms a short lead in to the break rather than constituting a break in itself.

Breaks 3 and 7 are preceded by a short crescendo Snare Drum roll. These, and the crescendo roll which ends the piece, are achieved by setting the note velocity parameters for the associated Part such that the velocity begins at zero and quickly increases linearly over the required period (see Section **5.7.2**, Example 3):-

```
Sect  2 Part  8, Chan 10, Patch   0, Pan  64, BEATS    145 to     160
==============
Parameter      Min       Max       MinMean    Start     Grad      Lambda
-------------------------------------------------------------------------
PITCH           38        38          38              1.000000  0.500000
LENGTH (/  2)    1         1           1              1.000000  0.500000
VELOCITY         0       127           3         0    1.000000 20.000000
```

The complete *Markov* program score for the piece can be seen in Appendix B, Section **B.1.4**.

6.5.4 Discussion of the Results

The percussive rhythms are, as has been shown above, easily constructed and entirely convincing.

Melodically, the algorithm is constraining as, in general, it is impossible to produce any given melody. However, due to the relative melodic simplicity of this style of music, it is nevertheless possible to produce a wide range of melodic structures which are in keeping with the style and therefore do not betray any compositional compromise. It should be noted, however, that very simple sequences still require a full set of parametric data, the size of which may be considered to be large in comparison to the simplicity of output obtained. For example, the single-note repeating bass line used in this piece requires 15 parameter values, and, in general, it is a weakness of the algorithm that although the amount of input data required is relatively very small when applied to producing complex results, it is disproportionately large when applied to producing simple ones.

An important limiting factor in this piece is the MIDI sound set, which, being intentionally very general, does not allow for the inclusion of the sorts of synthesized sounds which are often used in this music and which will have been constructed on a synthesizer. Here, Saw Wave and Synthesised Strings have been used but there are few other MIDI sounds which would be appropriate. Indeed, the sound used for the bass line is in fact Nylon-strung guitar, this being more convincing than the MIDI bass sounds!

6.6 SUMMARY

This chapter has explored the ability of the algorithm, within the compositional environment afforded by the *Markov* program, to generate music which meets specific stylistic objectives. This exploration has ranged from completely deterministic musical structures, in the case of Reich Phase Music and Dance Music, through probabilistic variation within a tightly controlled framework, as in Bach Harpsichord, to partially controlled, but relatively free, variation in the case of Gagaku.

The ability of the algorithm to achieve precise melodic sequences, albeit somewhat limited in variety, and specific rhythmic patterns makes it well suited to process-oriented music such as the Reich Phase Music and to the rhythmically structured nature of Dance music, and successful results are obtainable from an amount of input data which, in the former example is very small in total but which, in the latter, is large considering the simplicity of the musical content. The strictly horizontal nature of the sequences produced by the algorithm means, however, that any vertical structure must be built in "manually". In the case of music with a tight harmonic structure such as Bach Harpsichord, this requires fixing certain subsequences, and controlling the general shape of others, to try to ensure that this harmonic structure is achieved, and here this control was applied mostly on a bar by bar basis. Thus, the total amount of data required is increased significantly since, although the same small set of parameter value data is required, it cumulates bar on bar. On the other hand, judicious use of the diagonal line parameters shows the algorithm to be very capable of achieving the short melodic patterns inherent in Bach's style, and once the parameters have been set, repeated renditions in the style are obtainable without the need for any rules to be supplied specifically. Instead, any "rules" are implicit in the parameter settings. The amount of algorithmic effort required to produce the Bach piece is, however, prohibitive. For music with a rather freer harmonic structure, such as Gagaku, much more successful results may be obtained from comparatively long, probabilistically varying, sequences.

Many of the lessons learned in Chapter 5 have been put into practice here, not only the control of probabilistic variation in note sequences, but also, for example, achieving accelerando and crescendo, as well as making the most of the kinds of deterministic sequences which are possible. The techniques have also been applied in unexpected ways to obtain results of which one would not, at first, have thought the algorithm to be capable, for example: phase shift between two separate lines, random bending of the pitch of notes in a sequence, rising and falling volume of a continuously played note, rests within a deterministic note sequence and turns.

Limitations of the algorithm have been revealed and discussed, both specifically and with regard to "errors", in relation to the style in question, which may be manifested. The *Markov* program, clearly, does not provide a multi-purpose compositional environment, but the unique qualities of the algorithm provide a surprisingly large armoury of techniques through which compositions which attempt to meet desired stylistic goals may be realised.

Chapter 7

Compositional Studies

7.1 INTRODUCTION

This chapter describes three compositional studies of my own which were produced using the *Markov* computer program. Each composition exploits different aspects of the composing algorithm. The first, *Markov-2*, is based on two specific diagonal lines which are used to control both pitch and rhythm. This piece is, in a sense, definitive in that the way in which the diagonal lines are affecting the musical output is readily apparent. The second,*Vibrato Study*, focusses on the compositional possibilities afforded by algorithmic control of note vibrato. The final piece, *Computer Study for Timpani*, concentrates primarily on the control of dynamics.

The use of the term "studies" in the chapter title is deliberate. The intention behind these pieces is to explore specific aspects of the algorithm in a pure manner. Basic, "definitive", diagonal line types have purposely been chosen so that the algorithmic process at work may be evident in the realisation of the pieces.

The general structure of each of the pieces is described, relevant extracts from the program score are given, and subjective evaluative comments are made. The complete program score listings can be found in Appendix B.

The compact disc which accompanies this thesis contains realisations of these pieces in audio file format. The floppy disk contains them in composition file format for playback by the *Markov* program. Full details of the filenames and locations are given later in this chapter, in the relevant sections.

7.2 MARKOV-2

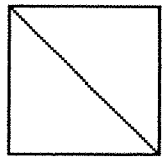
7.2.1 Description

Markov-2 is written for two violins playing pizzicato throughout and at constant dynamic. The piece is written in three movements, each being one minute in length. The pitch and note length ranges stay the same throughout (and are wide), the variation in movements being achieved by varying the diagonal lines for each of these two parameters.

Here, there is a wide variety between different realisations generated by the program but, nevertheless, a clear stylistic similarity. To illustrate this, the compact disc contains three separate realisations. These may be found in the files "Markov-2 (1).wav", "Markov-2 (2).wav" and "Markov-2 (3).wav". The floppy disk contains the *Markov* Composition File, called "Markov-2", for playback by the *Markov* program. All files are held in the "Compositional Studies" directory.

First Movement

For the first movement, for each of the two violin parts, both pitch and note length are controlled by the following diagonal line:



As discussed in Section **5.3.2**, Example 1, this results in a meandering movement, up and down. Therefore, the pitch repeatedly rises and falls, while the rhythm repeatedly speeds up and slows down. The higher the value of λ, the more constrained is the degree of movement. Here, the λ value is set to 0.5, resulting in a fairly gentle up and down motion but with occasional larger jumps occurring. The program score for each of the two violin parts is as follows:

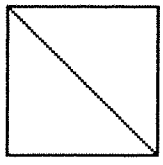
Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	83	48		1.000000	0.500000
LENGTH (/ 32)	1	32	1		1.000000	0.500000

The two parts are sent to different MIDI channels and the Pan values are set so as to achieve stereo separation.

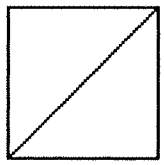
The meandering of the pitch and rhythm is clearly heard and provides a direct aural interpretation of the diagonal line parameter settings. The two parts are moving completely independently of one another but frequently give the illusion of deliberate contrapuntal design. This is typical of the way that various degrees of synchronisation may occur by chance.

Second Movement

For the second movement, the pitch is controlled by the same diagonal line:



but the note length is controlled by a diagonal line sloping in the opposite direction:



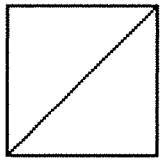
The relevant extract from the score is as follows:

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	83	48		1.000000	0.500000
LENGTH (/ 32)	1	32	1		-1.000000	0.500000

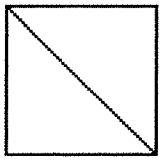
Now, while the pitch moves in the same way as in the first movement, there are periods when the note lengths alternate in length between short and long, resulting in a jerky rhythm, interspersed with periods of more even note length (see Section **5.6.2**, Example 2). The controlling effect of the parameter settings is, again, aurally readily apparent.

Third Movement

For the third, and final, movement, the pitch is now controlled by an upward sloping diagonal:



while the note length is controlled by the same downward sloping diagonal as in the first movement:



The relevant extract from the score is as follows:

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	83	48		-1.000000	0.500000
LENGTH (/ 32)	1	32	1		1.000000	0.500000

Now, there are periods when the pitch alternates between high and low, interspersed with periods of more gentle pitch movement (see Section **5.3.2**, Example 6). An additional effect is that, during periods when both parts are

alternating between high and low pitch, the two lines combine so that the listener hears two much more stable lines, one high and one low.

7.2.2 Evaluation

This piece can thought of as "space filling", in that there is little sense of temporal motion; each of the movements just "is". The separate movements have no beginning or end and, no matter how long they were to last, the listener could arrive at any point and leave at any point and the same qualitative impression would be made. The use of pizzicato violin gives an initial, fleeting illusion that the performance could be human but that sense is quickly dashed as soon as the tempo rises well above what could be humanly achieved, and from then on the piece is transparently mechanistic. The repeatedly falling and rising tempo in the first and third movements provides tension and release, with the resolution of tension unpredictable and often unexpected. A feeling of the lines continually trying to "catch themselves up" provides moments of humour whilst, simultaneously, the atonal nature of the work gives it a sense of unease. Overall, this piece could be said to appeal to the listener on an *intellectual* level.

7.3 VIBRATO STUDY

7.3.1 Description

This piece explores the use of varying vibrato rate as a process of compositional interest. All sounds are the "Voice Ooh" on the Roland JV30.

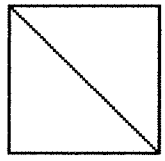
The fact that vibrato is very heavily used, and that the depth of vibrato is fixed at its maximum possible value means that, rather than simply adding a tremulous quality to a note, it becomes a sonic effect in its own right. This effect tends to give the piece an electro-acoustic feel.

Here, again, variations in successive program generations are definitely present but not readily discernible. One rendering only, therefore, is included on the compact disc. It may be found in the file "Vibrato Study.wav". The floppy disk contains the *Markov* Composition File, called "Vibrato Study", for playback by the *Markov* program. Both files are held in the "Compositional Studies" directory.

The piece comprises four main sections:

First Section

A continuous low pitched drone forms a backing over which voices enter infrequently and then die away. Each separate voice entry has a constant but randomly assigned vibrato rate, the rates being selected from a wide range (i.e. slow to fast) according to the following diagonal line:



but with a relatively low λ value of 0.05 so that wide variations of vibrato rate may occur between successive entries. The program score for one of these entries is as follows (notice that the release parameter has been fixed at a relatively high value so that the notes gradually die away rather than terminate abruptly):

```
Sect  1 Part  4, Chan  1, Patch  54, Pan   4, BEATS     13 to      18
==============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           48        84          60               1.000000  0.500000
LENGTH (/  1)    6         6           6               1.000000  0.500000
VIBDEPTH       114       114         114               1.000000  0.500000
VIBRATE         14       114          14               1.000000  0.050000
RELEASE        104       104         104               1.000000  0.500000
```

At first the voices occur singly, later they occur in pairs.

Second Section

The same drone is used as a backing but now the voices enter much more frequently, still with randomly assigned vibrato rates.

Third Section

The drone is replaced by a backing of three voices at different pitches. The first voice has a constant vibrato rate of one cycle per crotchet, the second voice a rate of one cycle per quaver and the third voice a rate of one cycle per semiquaver, so that the vibrato produces a constant rhythmical backing[1]. Over

[1]The precise MIDI vibrato rate values required to produce these rhythms (31, 41 and 61 respectively) were found by a process of trial and error.

this backing, two voices sing individual parts, their note lengths being either crotchets or quavers, in the case of the first part, and either quavers or semiquavers, in the case of the second part. The dynamic level of each of these is two parts is allowed to vary. The program score for one of these two parts is as follows:

```
Sect  3 Part  5, Chan  9, Patch  54, Pan  94, BEATS     61 to     120
==============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           60        84          60              1.000000  0.500000
LENGTH (/  4)    1         2           1              1.000000  0.500000
VELOCITY        40        90          40              1.000000  0.100000
```

Fourth Section

The last short section has 8 different voices at different fixed pitches and (out of phase) vibrato rates, entering one by one so as to build up to a final crescendo.

7.3.2 Evaluation

This piece, in direct contrast to the previous one, can be thought of as "time filling" in that there is a definite sense of forward temporal motion, of a journey taking place. The imagery created of the landscape of this journey, could, by virtue of the "atmospheric" nature of the sounds used, be that of an inhospitable landscape, perhaps otherworldly. Once again, as in the last piece, the initial notion of human performance created, in this case, by the use of a vocal instrument is soon replaced by a contradictory, but mysterious rather than cold, mechanistic feel through the repeated use of sustained, deep vibrato and continuous backing drones. Overall, this piece could be said to invoke an *emotional* response from the listener.

7.4 COMPUTER STUDY FOR TIMPANI

7.4.1 Description

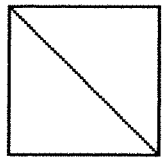
Computer Study for Timpani is built from four separate timpani parts. There is very little rhythmic variation and, apart from the final section, very little variation in pitch either. Instead, the principal parameter of musical interest is dynamic variation.

With this piece, although quite wide variations between successive program generations can occur, the relatively tight overall parametric control means that these variations are not readily perceived by the listener. Only one realisation, therefore, is included on the compact disc. It may be found in the file "Computer Study for Timpani.wav". The floppy disk contains the *Markov* Composition File, called "Computer Study for Timpani", for playback by the *Markov* program. Both files are held in the "Compositional Studies" directory.

The piece is formed from four main sections:

First Section

The opening section consists of short passages of timpani, played at a fairly slow rhythm and with rising and falling dynamic. The first two such passages have their own fixed pitch, followed by two passages where some pitch variation is allowed. For these first four passages, a solo timpani is playing but for the final passage, two timpani play together. All parameters are controlled by the following diagonal line:



The velocity is allowed to vary across the full MIDI range (1 to 127) and a relatively low λ value of 0.05 means that large jumps frequently occur. The pitch and note length (λ=0.5), however, vary more gently. The program score for the first of these passages is as follows:

```
Sect  1 Part  1, Chan  1, Patch  48, Pan   0, BEATS      1 to      30
==============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           48        48          48                1.000000  0.500000
LENGTH (/ 16)    1        16           1                1.000000  0.500000
VELOCITY         1       127           1                1.000000  0.050000
```

This, and the following, section demonstrates how, in contrast to the *Markov*-2 piece, a composition may be built up from a number of parts in a tightly structured way, rather than allowing long periods of probabilistic freedom.

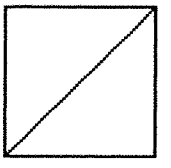
Second Section

This section again consists of short, but much more rapid, passages. Now, the variation in dynamic is tightly controlled. The section is built from four groups of passages, beginning with a group of two passages which occur in quick succession, followed by a similar group of three passages, then a similar group of four, and finally a group of two played simultaneously. For each group, the dynamic level is allowed to vary only within a narrow range, starting softly and then increasing with each group and reaching a crescendo with the final group. These four groups are repeated. The relevant extracts from the program score for the first of these groups is as follows:

```
Sect  6 Part  1, Chan  1, Patch  48, Pan   0, BEATS      7 to      10
==============
Parameter      Min       Max       MinMean    Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           48        60          48              1.000000  0.500000
LENGTH (/ 90)   10        15          10              1.000000  0.500000
VELOCITY        10        32          10              1.000000  0.050000

Sect  6 Part  2, Chan  2, Patch  48, Pan 127, BEATS     11 to      12
==============
Parameter      Min       Max       MinMean    Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           60        72          60              1.000000  0.500000
LENGTH (/ 90)   10        15          10              1.000000  0.500000
VELOCITY        10        32          10              1.000000  0.050000
```

Third Section

In this longer section, four timpani play together, each at its own fixed pitch and note length. The dynamics, however, are allowed to vary according to the following diagonal line:



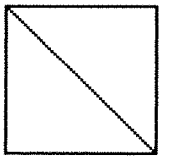
so that loud notes tend to be followed by soft ones, and vice versa. This results in pulses of sound, occurring independently in each of the four parts, so that a (randomly generated) rhythm is implied. The program score for one of these parts is as follows:

```
Sect 14 Part  1, Chan  1, Patch  48, Pan   0, BEATS      7 to     126
==============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           48        48          48                1.000000  0.500000
LENGTH (/120)   50        50          50                1.000000  0.500000
VELOCITY        50       127         127               -1.000000  0.050000
```

As the section progresses, the pitch levels of each of the parts gradually rise, step by step.

Fourth Section

The final section, which forms the finale to the piece, is very similar to the previous section but now the pitches begin to vary across a wide range according to the following diagonal line:



The program score for one of the parts is as follows:

```
Sect 14 Part  9, Chan  1, Patch  48, Pan   0, BEATS    247 to     366
==============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           48        71          48         69     1.000000  0.500000
LENGTH (/120)   50        50          50                1.000000  0.500000
VELOCITY        50       127         127               -1.000000  0.050000
```

7.4.2 Evaluation

This piece can be thought of as both "space-filling" and "time-filling"; from within the apparently static drumming rhythms a sense of evolution emerges through the use of changing timpani pitches. This work, in direct contrast to the previous two, has a distinct human quality. A strength of this piece is its ability to visually stimulate: the listener can readily visualise a live performance, of both drumming and, perhaps, dance, since the piece's rhythmic energy and dynamism creates a sense of spatial motion. Overall, this piece could be said to appeal to the listener on a *physical* level.

CONCLUSION

This research was born from a set of ideals regarding the composition of music based on a mathematical algorithm. The underlying philosophy was "numbers in, music out". That is to say, "if I want to achieve such and such a musical result, then these are the numbers I must feed into the algorithm". Composition could never be reduced to such a simple situation, of course, but from this philosophy began the search for an algorithm which was simple enough that its underlying processes were accessible to the composer without having to understand the details of the mathematics involved, but which was capable of a wide range of musical output which could predictably be controlled by the composer. Out of these ideals grew a set of specific objectives.

From the wide range of algorithmic composition techniques which have been developed over the last forty years, I chose the Markov chain as the underlying process for this research. Relatively simple in concept, it provides a model for the composition of music whose basis is probabilistic evolution, the entire character of which is embodied in the distribution of numbers in a square grid, the transition matrix. What was required was a way for the composer to produce this, potentially very large, matrix quickly and simply and to have some understanding of the character of the process thus created. The diagonal line method provides such a way. Five numerical parameter values are required for each musical attribute, four of which define the diagonal line, and consequently the structural character of the note sequences which will result, and the fifth specifies the probabilistic degree of freedom in relation to that structure.

In order to be able to explore the musical capabilities of the algorithm, I developed a computer program, *Markov*. This program is essentially a MIDI sequencer which provides a user interface for the composer to enter the algorithm's parameter values and to define the overall structural framework within which sequences are generated. What the program also provides is a feedback environment, whereby the composer may move progressively towards a desired musical result. Initial experimentation with the program suggested to me that, for musical reasons, some compromises to the "pure" application of the algorithm were necessary. Specifically, two simple rules were added: the option to prevent two successive repeats of the same value of a note attribute in a sequence, and the ability to fix any of the attribute values for the initial note in a sequence.

An extensive, documented, analysis of the relationship between the musical output and the values of the input parameters, provided what is, in effect, a "cookbook" of compositional techniques. This study showed the

surprising diversity of output obtainable from the algorithm, ranging from constrained, structured sequences to free, aleatoric ones, with many possibilities in between. The controlled dependence of the output on the input values was clearly demonstrated.

The algorithm was now applied to the task of emulating given musical styles. Four different styles were attempted. These styles were chosen for their diversity and as a challenge to the algorithm, not because they were judged in advance to be suited to any idiosyncratic qualities of the algorithm. Indeed, important limitations became apparent, particularly when attempting to achieve rigid harmonic and melodic structures, where it was necessary to compromise by applying the algorithm to very short note sequences. On the other hand, the results were surprisingly successful in many important aspects and the flexibility of the algorithm, within the composing environment offered by the program, was readily apparent.

Finally, I have presented some compositional studies of my own, which attempt to explore specific aspects of the algorithm in a creative manner, and in keeping with my own compositional style. These enable the mathematical processes at work to be manifested in the music and therefore, ultimately, demystified.

Appendix B

Markov Program Scores

B.1.1 Steve Reich Phase Music

```
COMPOSITION FILE: Hard Disk:Music:reich
================

Section  Parts  Min Length  Min Total  Max Length  Max Total
-------------------------------------------------------------
    1      2       1140       1140        1140       1140

SECTION SEQUENCE:
   1

Tempo = 400 bpm   MIDI Buffer Size =    200 Bytes

SECTION:   1
============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
TRANSPOSE        0         0           0                1.000000  0.500000

Part No      MIDI Channel      Starting Beat     Ending Beat
-------------------------------------------------------------
    1             1                   1            1140
    2             2                   1            1140

Sect  1 Part  1, Chan  1, Patch   1, Pan  24, BEATS      1 to    1140
===============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           60        80          62         60     2.000000 20.000000
LENGTH (/192) 192       192         192                1.000000  0.500000

            REPEAT  | SELECT  |  REVERSE  | REFLECT  | RANDOM ENTRY
            ------------------------------------------------------
PITCH         YES   |   YES   |     NO    |    NO    |
LENGTH        YES   |    NO   |     NO    |    NO    |      NO
VELOCITY      YES   |    NO   |     NO    |    NO    |

PITCH SELECTIONS:
  60  62  63  65  67  68  71  72  74  75  77  79  80

Sect  1 Part  2, Chan  2, Patch   1, Pan 104, BEATS      1 to    1140
===============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           60        80          62         60     2.000000 20.000000
LENGTH (/192) 190       190         190                1.000000  0.500000

            REPEAT  | SELECT  |  REVERSE  | REFLECT  | RANDOM ENTRY
            ------------------------------------------------------
PITCH         YES   |   YES   |     NO    |    NO    |
LENGTH        YES   |    NO   |     NO    |    NO    |      NO
VELOCITY      YES   |    NO   |     NO    |    NO    |

PITCH SELECTIONS:
  60  62  63  65  67  68  71  72  74  75  77  79  80
```

B.1.2 Gagaku - Japanese Court Music

COMPOSITION FILE: Hard Disk:Music:gagaku

Section	Parts	Min Length	Min Total	Max Length	Max Total
1	5	60	60	60	60
2	17	125	185	125	185

SECTION SEQUENCE:
1 2

Tempo = 50 bpm MIDI Buffer Size = 700 Bytes

SECTION: 1

Parameter	Min	Max	MinMean	Start	Grad	Lambda
TRANSPOSE	0	0	0		1.000000	0.500000

Part No	MIDI Channel	Starting Beat	Ending Beat
1	1	1	60
2	1	1	60
3	2	20	50
4	2	51	60
5	3	40	60

Sect 1 Part 1, Chan 1, Patch 73, Pan 64, BEATS 1 to 60

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	76	88	76		1.000000	0.500000
LENGTH (/ 2)	1	8	8		0.000000	0.500000
VELOCITY	0	127	127	127	0.000000	2.000000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	NO	YES	NO	NO	
LENGTH	YES	YES	NO	NO	NO
VELOCITY	YES	YES	NO	NO	

PITCH SELECTIONS:
76 78 79 81 83 84 86 88
LENGTH SELECTIONS:
1 2 4 8
VELOCITY SELECTIONS:
0 127

Sect 1 Part 2, Chan 1, Patch 73, Pan 64, BEATS 1 to 60

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	60	60	60		1.000000	0.500000
LENGTH (/ 8)	1	1	1		1.000000	0.500000
VELOCITY	0	0	0		1.000000	0.500000
PITCH BEND	32	96	32	64	1.000000	0.700000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 1 Part 3, Chan 2, Patch 48, Pan 64, BEATS 20 to 50

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	68	68	68		1.000000	0.500000
LENGTH (/ 1)	1	1	1		1.000000	0.500000
VELOCITY	0	127	0	127	0.000000	5.000000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	YES	NO	NO	

VELOCITY SELECTIONS:
0 127

Sect 1 Part 4, Chan 2, Patch 48, Pan 64, BEATS 51 to 60

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	68	68	68		1.000000	0.500000
LENGTH (/ 64)	5	64	5	64	0.800000	2.000000
VELOCITY	100	100	100		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 1 Part 5, Chan 3, Patch 117, Pan 64, BEATS 40 to 60

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	67	67	67		1.000000	0.500000
LENGTH (/ 1)	1	1	1		1.000000	0.500000
VELOCITY	0	127	0	127	0.000000	6.000000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	YES	NO	NO	

VELOCITY SELECTIONS:
0 127

SECTION: 2

Parameter	Min	Max	MinMean	Start	Grad	Lambda
TRANSPOSE	0	0	0		1.000000	0.500000

Part No	MIDI Channel	Starting Beat	Ending Beat
1	1	1	125
2	1	1	5
3	2	1	125
4	2	1	5
5	1	6	120
6	2	6	120
7	3	1	5
8	4	1	125
9	4	1	5
10	4	6	120

11	5	30	59
12	5	30	120
13	6	61	120
14	6	61	120
15	7	62	120
16	7	62	120
17	5	60	120

Sect 2 Part 1, Chan 1, Patch 112, Pan 64, BEATS 1 to 125

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	71	71	71		1.000000	0.500000
LENGTH (/ 1)	125	125	125		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	YES	NO	NO	NO
VELOCITY	NO	YES	NO	NO	

LENGTH SELECTIONS:
125
VELOCITY SELECTIONS:
127

Sect 2 Part 2, Chan 1, Patch 112, Pan 64, BEATS 1 to 5

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	71	71	71		1.000000	0.500000
LENGTH (/ 8)	1	1	1		1.000000	0.500000
VELOCITY	0	0	0		1.000000	0.500000
VOLUME	0	127	5	0	0.961000	2.000000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 2 Part 3, Chan 2, Patch 112, Pan 64, BEATS 1 to 125

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	74	74	74		1.000000	0.500000
LENGTH (/ 1)	125	125	125		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	YES	NO	NO	NO
VELOCITY	NO	YES	NO	NO	

LENGTH SELECTIONS:
125
VELOCITY SELECTIONS:
127

Sect 2 Part 4, Chan 2, Patch 112, Pan 64, BEATS 1 to 5

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	74	74	74		1.000000	0.500000
LENGTH (/ 8)	1	1	1		0.800000	2.000000
VELOCITY	0	0	0		1.000000	0.500000
VOLUME	0	127	5	0	0.961000	2.000000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 2 Part 5, Chan 1, Patch 112, Pan 64, BEATS 6 to 120

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	71	71	71		1.000000	0.500000
LENGTH (/ 2)	1	1	1		1.000000	0.500000
VELOCITY	0	0	0		1.000000	0.500000
VOLUME	32	127	32	90	1.000000	0.200000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 2 Part 6, Chan 2, Patch 112, Pan 64, BEATS 6 to 120

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	74	74	74		1.000000	0.500000
LENGTH (/ 2)	1	1	1		1.000000	0.500000
VELOCITY	0	0	0		1.000000	0.500000
VOLUME	32	127	32	90	1.000000	0.200000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 2 Part 7, Chan 3, Patch 48, Pan 64, BEATS 1 to 5

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	68	68	68		1.000000	0.500000
LENGTH (/ 64)	2	64	6	5	0.000000	2.000000
VELOCITY	100	100	100	0	1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 2 Part 8, Chan 4, Patch 112, Pan 64, BEATS 1 to 125

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	76	76	76		1.000000	0.500000
LENGTH (/ 1)	125	125	125		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	YES	NO	NO	NO
VELOCITY	NO	YES	NO	NO	

```
LENGTH SELECTIONS:
 125
VELOCITY SELECTIONS:
 127
```

Sect 2 Part 9, Chan 4, Patch 112, Pan 64, BEATS 1 to 5

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	76	76	76		1.000000	0.500000
LENGTH (/ 8)	1	1	1		1.000000	0.500000
VELOCITY	0	0	0		1.000000	0.500000
VOLUME	0	127	5	0	0.961000	2.000000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 2 Part 10, Chan 4, Patch 112, Pan 64, BEATS 6 to 120

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	76	76	76		1.000000	0.500000
LENGTH (/ 2)	1	1	1		1.000000	0.500000
VELOCITY	0	0	0		1.000000	0.500000
VOLUME	32	127	32	90	1.000000	0.200000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 2 Part 11, Chan 5, Patch 71, Pan 40, BEATS 30 to 59

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	67	81	67	71	1.000000	2.000000
LENGTH (/ 4)	4	16	4		1.000000	0.500000
VELOCITY	0	127	127		0.000000	3.000000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	NO	YES	NO	NO	
LENGTH	YES	YES	NO	NO	NO
VELOCITY	YES	YES	NO	NO	

```
PITCH SELECTIONS:
  67  69  71  72  74  76  79  81
LENGTH SELECTIONS:
   4   8   9  10  11  12  13  14  15  16
VELOCITY SELECTIONS:
   0 127
```

Sect 2 Part 12, Chan 5, Patch 71, Pan 40, BEATS 30 to 120

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	60	60	60		1.000000	0.500000
LENGTH (/ 8)	1	1	1		1.000000	0.500000
VELOCITY	0	0	0		1.000000	0.500000
PITCH BEND	48	80	48	64	1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 2 Part 13, Chan 6, Patch 71, Pan 88, BEATS 61 to 120

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	69	78	69	71	1.000000	2.000000
LENGTH (/ 4)	4	16	4		-1.000000	2.000000
VELOCITY	0	127	127		0.000000	3.000000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	NO	YES	NO	NO	
LENGTH	YES	YES	NO	NO	NO
VELOCITY	YES	YES	NO	NO	

PITCH SELECTIONS:
69 71 72 74 76 78
LENGTH SELECTIONS:
4 8 9 10 11 12 13 14 15 16
VELOCITY SELECTIONS:
0 127

Sect 2 Part 14, Chan 6, Patch 71, Pan 88, BEATS 61 to 120

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	60	60	60		1.000000	0.500000
LENGTH (/ 8)	1	1	1		1.000000	0.500000
VELOCITY	0	0	0		1.000000	0.500000
PITCH BEND	48	80	48	64	1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 2 Part 15, Chan 7, Patch 71, Pan 64, BEATS 62 to 120

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	69	78	69	71	1.000000	2.000000
LENGTH (/ 4)	4	16	4		-1.000000	2.000000
VELOCITY	0	127	127		0.000000	3.000000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	NO	YES	NO	NO	
LENGTH	YES	YES	NO	NO	NO
VELOCITY	YES	YES	NO	NO	

PITCH SELECTIONS:
69 71 72 74 76 78
LENGTH SELECTIONS:
4 8 9 10 11 12 13 14 15 16
VELOCITY SELECTIONS:
0 127

Sect 2 Part 16, Chan 7, Patch 71, Pan 64, BEATS 62 to 120

Parameter	Min	Max	MinMean	Start	Grad	Lambda

```
PITCH          60       60          60              1.000000  0.500000
LENGTH (/  8)   1        1           1              1.000000  0.500000
VELOCITY        0        0           0              1.000000  0.500000
PITCH BEND     48       80          48        64    1.000000  0.500000

            REPEAT  | SELECT  |  REVERSE  | REFLECT  | RANDOM ENTRY
            ------------------------------------------------------
PITCH         YES   |   NO    |    NO     |   NO     |
LENGTH        YES   |   NO    |    NO     |   NO     |      NO
VELOCITY      YES   |   NO    |    NO     |   NO     |
```

Sect 2 Part 17, Chan 5, Patch 71, Pan 40, BEATS 60 to 120

```
Parameter      Min      Max       MinMean    Start     Grad      Lambda
-------------------------------------------------------------------------
PITCH           69       78          69        71    1.000000  2.000000
LENGTH (/  4)    4       16           4             -1.000000  2.000000
VELOCITY         0      127         127             0.000000  3.000000

            REPEAT  | SELECT  |  REVERSE  | REFLECT  | RANDOM ENTRY
            ------------------------------------------------------
PITCH         NO    |   YES   |    NO     |   NO     |
LENGTH        YES   |   YES   |    NO     |   NO     |      NO
VELOCITY      YES   |   YES   |    NO     |   NO     |

PITCH SELECTIONS:
  69  71  72  74  76  78
LENGTH SELECTIONS:
   4   8   9  10  11  12  13  14  15  16
VELOCITY SELECTIONS:
   0 127
```

B.1.3 Bach Harpsichord Music

```
COMPOSITION FILE: Hard Disk:Music:Bach
================
Section  Parts  Min Length  Min Total  Max Length  Max Total
-------------------------------------------------------------
    1      19         51         51          51         51
    2      17         46         97          46         97
SECTION SEQUENCE:

Tempo = 384 bpm   MIDI Buffer Size =    200 Bytes

SECTION:   1
============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
TRANSPOSE        0         0           0                1.000000  0.500000

Part No     MIDI Channel     Starting Beat    Ending Beat
-----------------------------------------------------------
    1              1                1              3
    2              1                4              7
    3              1                8              9
    4              1               10             15
    5              1               16             21
    6              1               22             24
    7              1               29             33
    8              1               34             36
    9              1               41             45
   10              1               46             48
   11              2                4              9
   12              2               10             15
   13              2               16             21
   14              2               22             27
   15              2               28             33
   16              2               34             39
   17              2               40             45
   18              2               46             49
   19              2               50             51

Sect  1 Part  1, Chan  1, Patch   7, Pan  64, BEATS      1 to       3
===============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           65        68          67         67     1.000000 20.000000
LENGTH (/  1)    1         1           1                 1.000000  0.500000

            REPEAT   | SELECT   |  REVERSE   | REFLECT   | RANDOM ENTRY
            ---------------------------------------------------------
PITCH         NO     |   YES    |     NO     |    NO     |
LENGTH       YES     |    NO    |     NO     |    NO     |       NO
VELOCITY     YES     |    NO    |     NO     |    NO     |

PITCH SELECTIONS:
  65  67  68

Sect  1 Part  2, Chan  1, Patch   7, Pan  64, BEATS      4 to       7
===============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           65        72          65         67     1.000000  0.500000
LENGTH (/  1)    1         1           1                 1.000000  0.500000

            REPEAT   | SELECT   |  REVERSE   | REFLECT   | RANDOM ENTRY
            ---------------------------------------------------------
PITCH         NO     |   YES    |     NO     |    NO     |
```

```
LENGTH        YES   |    NO   |    NO   |    NO   |       NO
VELOCITY      YES   |    NO   |    NO   |    NO   |

PITCH SELECTIONS:
  65  67  72

Sect  1 Part  3, Chan  1, Patch   7, Pan  64, BEATS      8 to       9
===============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           62        63          62         63     1.000000  0.500000
LENGTH (/  1)    1         1           1                 1.000000  0.500000

             REPEAT  | SELECT  | REVERSE  | REFLECT  | RANDOM ENTRY
             ---------------------------------------------------
PITCH          NO    |   YES   |    NO    |    NO    |
LENGTH        YES    |    NO   |    NO    |    NO    |       NO
VELOCITY      YES    |    NO   |    NO    |    NO    |

PITCH SELECTIONS:
  62  63

Sect  1 Part  4, Chan  1, Patch   7, Pan  64, BEATS     10 to      15
===============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           60        67          60         63     1.000000  0.500000
LENGTH (/  1)    1         1           1                 1.000000  0.500000

             REPEAT  | SELECT  | REVERSE  | REFLECT  | RANDOM ENTRY
             ---------------------------------------------------
PITCH          NO    |   YES   |    NO    |    NO    |
LENGTH        YES    |    NO   |    NO    |    NO    |       NO
VELOCITY      YES    |    NO   |    NO    |    NO    |

PITCH SELECTIONS:
  60  62  63  65  67

Sect  1 Part  5, Chan  1, Patch   7, Pan  64, BEATS     16 to      21
===============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           65        72          65         67     1.000000  0.500000
LENGTH (/  1)    1         1           1                 1.000000  0.500000

             REPEAT  | SELECT  | REVERSE  | REFLECT  | RANDOM ENTRY
             ---------------------------------------------------
PITCH          NO    |   YES   |    NO    |    NO    |
LENGTH        YES    |    NO   |    NO    |    NO    |       NO
VELOCITY      YES    |    NO   |    NO    |    NO    |

PITCH SELECTIONS:
  65  67  68  70  72

Sect  1 Part  6, Chan  1, Patch   7, Pan  64, BEATS     22 to      24
===============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           67        74          71         71     1.000000 20.000000
LENGTH (/  1)    1         1           1                 1.000000  0.500000

             REPEAT  | SELECT  | REVERSE  | REFLECT  | RANDOM ENTRY
             ---------------------------------------------------
PITCH         YES    |   YES   |    NO    |    NO    |
LENGTH        YES    |    NO   |    NO    |    NO    |       NO
VELOCITY      YES    |    NO   |    NO    |    NO    |

PITCH SELECTIONS:
```

```
  67  71  74

Sect  1 Part  7, Chan  1, Patch   7, Pan  64, BEATS     29 to      33
===============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           70        79          79         79     1.000000  2.000000
LENGTH (/  1)    1         1           1                 1.000000  0.500000

             REPEAT   | SELECT   |  REVERSE  | REFLECT  | RANDOM ENTRY
             ---------------------------------------------------------
PITCH          NO     |   YES    |    NO     |   NO     |
LENGTH        YES     |    NO    |    NO     |   NO     |       NO
VELOCITY      YES     |    NO    |    NO     |   NO     |

PITCH SELECTIONS:
  70  72  75  79

Sect  1 Part  8, Chan  1, Patch   7, Pan  64, BEATS     34 to      36
===============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           65        72          68         68     1.000000 20.000000
LENGTH (/  1)    1         1           1                 1.000000  0.500000

             REPEAT   | SELECT   |  REVERSE  | REFLECT  | RANDOM ENTRY
             ---------------------------------------------------------
PITCH          NO     |   YES    |    NO     |   NO     |
LENGTH        YES     |    NO    |    NO     |   NO     |       NO
VELOCITY      YES     |    NO    |    NO     |   NO     |

PITCH SELECTIONS:
  65  68  72

Sect  1 Part  9, Chan  1, Patch   7, Pan  64, BEATS     41 to      45
===============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           68        77          77         77     1.000000  2.000000
LENGTH (/  1)    1         1           1                 1.000000  0.500000

             REPEAT   | SELECT   |  REVERSE  | REFLECT  | RANDOM ENTRY
             ---------------------------------------------------------
PITCH          NO     |   YES    |    NO     |   NO     |
LENGTH        YES     |    NO    |    NO     |   NO     |       NO
VELOCITY      YES     |    NO    |    NO     |   NO     |

PITCH SELECTIONS:
  68  70  74  77

Sect  1 Part 10, Chan  1, Patch   7, Pan  64, BEATS     46 to      48
===============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           63        70          67         67     1.000000 20.000000
LENGTH (/  1)    1         1           1                 1.000000  0.500000

             REPEAT   | SELECT   |  REVERSE  | REFLECT  | RANDOM ENTRY
             ---------------------------------------------------------
PITCH         YES     |   YES    |    NO     |   NO     |
LENGTH        YES     |   YES    |    NO     |   NO     |       NO
VELOCITY      YES     |    NO    |    NO     |   NO     |

PITCH SELECTIONS:
  63  67  70
LENGTH SELECTIONS:
   1
```

```
Sect  1 Part 11, Chan  2, Patch   7, Pan  64, BEATS      4 to       9
===============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           48        55         51         48     1.000000 20.000000
LENGTH (/  1)    2         2          2                 1.000000  0.500000

             REPEAT   | SELECT   |  REVERSE  | REFLECT  | RANDOM ENTRY
             ---------------------------------------------------------
PITCH          NO     |   YES    |    NO     |   NO     |
LENGTH         YES    |   NO     |    NO     |   NO     |       NO
VELOCITY       YES    |   NO     |    NO     |   NO     |

PITCH SELECTIONS:
  48  51  55

Sect  1 Part 12, Chan  2, Patch   7, Pan  64, BEATS     10 to      15
===============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           48        60         48         60     1.000000  0.500000
LENGTH (/  1)    2         2          2                 1.000000  0.500000

             REPEAT   | SELECT   |  REVERSE  | REFLECT  | RANDOM ENTRY
             ---------------------------------------------------------
PITCH          NO     |   YES    |    NO     |   NO     |
LENGTH         YES    |   NO     |    NO     |   NO     |       NO
VELOCITY       YES    |   NO     |    NO     |   NO     |

PITCH SELECTIONS:
  48  60

Sect  1 Part 13, Chan  2, Patch   7, Pan  64, BEATS     16 to      21
===============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           56        58         56         58     1.000000  0.500000
LENGTH (/  1)    2         4          4          2     0.000000 20.000000

             REPEAT   | SELECT   |  REVERSE  | REFLECT  | RANDOM ENTRY
             ---------------------------------------------------------
PITCH          NO     |   YES    |    NO     |   NO     |
LENGTH         NO     |   YES    |    NO     |   NO     |       NO
VELOCITY       YES    |   NO     |    NO     |   NO     |

PITCH SELECTIONS:
  56  58
LENGTH SELECTIONS:
   2   4

Sect  1 Part 14, Chan  2, Patch   7, Pan  64, BEATS     22 to      27
===============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           55        65         65         55     1.000000 20.000000
LENGTH (/  1)    1         3          1          3     0.000000 20.000000

             REPEAT   | SELECT   |  REVERSE  | REFLECT  | RANDOM ENTRY
             ---------------------------------------------------------
PITCH          NO     |   YES    |    NO     |   NO     |
LENGTH         YES    |   YES    |    NO     |   NO     |       NO
VELOCITY       YES    |   NO     |    NO     |   NO     |

PITCH SELECTIONS:
  55  62  63  65
LENGTH SELECTIONS:
   1   3
```

```
Sect  1 Part 15, Chan  2, Patch   7, Pan  64, BEATS     28 to      33
==============
Parameter      Min       Max       MinMean    Start     Grad      Lambda
-------------------------------------------------------------------------
PITCH           60        63         60        63     1.000000  0.500000
LENGTH (/  1)    2         2          2                1.000000  0.500000

            REPEAT   | SELECT   |  REVERSE   | REFLECT   | RANDOM ENTRY
            ---------------------------------------------------------
PITCH         NO     |   YES    |     NO     |    NO     |
LENGTH       YES     |    NO    |     NO     |    NO     |       NO
VELOCITY     YES     |    NO    |     NO     |    NO     |

PITCH SELECTIONS:
  60  63

Sect  1 Part 16, Chan  2, Patch   7, Pan  64, BEATS     34 to      39
==============
Parameter      Min       Max       MinMean    Start     Grad      Lambda
-------------------------------------------------------------------------
PITCH           53        63         63        53     1.000000 20.000000
LENGTH (/  1)    1         3          1         3     0.000000 20.000000

            REPEAT   | SELECT   |  REVERSE   | REFLECT   | RANDOM ENTRY
            ---------------------------------------------------------
PITCH         NO     |   YES    |     NO     |    NO     |
LENGTH       YES     |   YES    |     NO     |    NO     |       NO
VELOCITY     YES     |    NO    |     NO     |    NO     |

PITCH SELECTIONS:
  53  60  62  63
LENGTH SELECTIONS:
   1   3

Sect  1 Part 17, Chan  2, Patch   7, Pan  64, BEATS     40 to      45
==============
Parameter      Min       Max       MinMean    Start     Grad      Lambda
-------------------------------------------------------------------------
PITCH           58        62         58        62     1.000000  0.500000
LENGTH (/  1)    2         2          2                1.000000  0.500000

            REPEAT   | SELECT   |  REVERSE   | REFLECT   | RANDOM ENTRY
            ---------------------------------------------------------
PITCH         NO     |   YES    |     NO     |    NO     |
LENGTH       YES     |    NO    |     NO     |    NO     |       NO
VELOCITY     YES     |    NO    |     NO     |    NO     |

PITCH SELECTIONS:
  58  62

Sect  1 Part 18, Chan  2, Patch   7, Pan  64, BEATS     46 to      49
==============
Parameter      Min       Max       MinMean    Start     Grad      Lambda
-------------------------------------------------------------------------
PITCH           51        55         51                1.000000 20.000000
LENGTH (/  1)    1         3          1         3     0.000000 20.000000

            REPEAT   | SELECT   |  REVERSE   | REFLECT   | RANDOM ENTRY
            ---------------------------------------------------------
PITCH         NO     |   YES    |     NO     |    NO     |
LENGTH       YES     |   YES    |     NO     |    NO     |       NO
VELOCITY     YES     |    NO    |     NO     |    NO     |

PITCH SELECTIONS:
  51  55
LENGTH SELECTIONS:
   1   3
```

Sect 1 Part 19, Chan 2, Patch 7, Pan 64, BEATS 50 to 51

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	53	56	53	53	1.000000	20.000000
LENGTH (/ 1)	1	1	1		0.000000	20.000000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	NO	YES	NO	NO	
LENGTH	YES	YES	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

PITCH SELECTIONS:
53 56
LENGTH SELECTIONS:
1

SECTION: 2

Parameter	Min	Max	MinMean	Start	Grad	Lambda
TRANSPOSE	0	0	0		1.000000	0.500000

Part No	MIDI Channel	Starting Beat	Ending Beat
1	1	2	6
2	1	7	12
3	1	13	18
4	1	19	24
5	1	25	30
6	1	31	36
7	1	37	39
8	2	1	6
9	2	7	12
10	2	13	18
11	2	19	24
12	2	25	30
13	2	31	34
14	2	35	44
15	1	40	42
16	1	43	43
17	1	44	46

Sect 2 Part 1, Chan 1, Patch 7, Pan 64, BEATS 2 to 6

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	61	70	70	70	1.000000	2.000000
LENGTH (/ 1)	1	1	1		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	NO	YES	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

PITCH SELECTIONS:
61 63 67 70

Sect 2 Part 2, Chan 1, Patch 7, Pan 64, BEATS 7 to 12

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	60	72	72	60	1.000000	2.000000
LENGTH (/ 1)	1	1	1		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	NO	YES	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

PITCH SELECTIONS:
60 63 65 68 72

Sect 2 Part 3, Chan 1, Patch 7, Pan 64, BEATS 13 to 18

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	62	74	74	62	1.000000	2.000000
LENGTH (/ 1)	1	1	1		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	NO	YES	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

PITCH SELECTIONS:
62 65 67 70 74

Sect 2 Part 4, Chan 1, Patch 7, Pan 64, BEATS 19 to 24

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	63	75	75	63	1.000000	2.000000
LENGTH (/ 1)	1	1	1		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	NO	YES	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

PITCH SELECTIONS:
63 67 68 72 75

Sect 2 Part 5, Chan 1, Patch 7, Pan 64, BEATS 25 to 30

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	65	77	77	65	1.000000	2.000000
LENGTH (/ 1)	1	1	1		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	NO	YES	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

PITCH SELECTIONS:
65 68 70 74 77

Sect 2 Part 6, Chan 1, Patch 7, Pan 64, BEATS 31 to 36

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	67	75	68	67	1.000000	2.000000
LENGTH (/ 1)	1	1	1		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	NO	YES	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

```
PITCH SELECTIONS:
  67  68  70  72  75

Sect  2 Part  7, Chan  1, Patch   7, Pan  64, BEATS     37 to      39
===============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           65        75         74         74     1.000000 20.000000
LENGTH (/  1)    1         1          1                 1.000000  0.500000

             REPEAT   | SELECT   |  REVERSE   | REFLECT  | RANDOM ENTRY
             ---------------------------------------------------------
PITCH          NO     |   YES    |     NO     |    NO    |
LENGTH        YES     |    NO    |     NO     |    NO    |       NO
VELOCITY      YES     |    NO    |     NO     |    NO    |

PITCH SELECTIONS:
  65  74  75

Sect  2 Part  8, Chan  2, Patch   7, Pan  64, BEATS      1 to       6
===============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           51        55         51         55     1.000000  0.500000
LENGTH (/  1)    2         2          2                 1.000000  0.500000

             REPEAT   | SELECT   |  REVERSE   | REFLECT  | RANDOM ENTRY
             ---------------------------------------------------------
PITCH          NO     |   YES    |     NO     |    NO    |
LENGTH        YES     |    NO    |     NO     |    NO    |       NO
VELOCITY      YES     |    NO    |     NO     |    NO    |

PITCH SELECTIONS:
  51  55

Sect  2 Part  9, Chan  2, Patch   7, Pan  64, BEATS      7 to      12
===============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           53        56         53         56     1.000000  0.500000
LENGTH (/  1)    2         2          2                 1.000000  0.500000

             REPEAT   | SELECT   |  REVERSE   | REFLECT  | RANDOM ENTRY
             ---------------------------------------------------------
PITCH          NO     |   YES    |     NO     |    NO    |
LENGTH        YES     |    NO    |     NO     |    NO    |       NO
VELOCITY      YES     |    NO    |     NO     |    NO    |

PITCH SELECTIONS:
  53  56

Sect  2 Part 10, Chan  2, Patch   7, Pan  64, BEATS     13 to      18
===============
Parameter      Min       Max       MinMean     Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           55        58         55         58     1.000000  0.500000
LENGTH (/  1)    2         2          2                 1.000000  0.500000

             REPEAT   | SELECT   |  REVERSE   | REFLECT  | RANDOM ENTRY
             ---------------------------------------------------------
PITCH          NO     |   YES    |     NO     |    NO    |
LENGTH        YES     |    NO    |     NO     |    NO    |       NO
VELOCITY      YES     |    NO    |     NO     |    NO    |

PITCH SELECTIONS:
  55  58
```

Sect 2 Part 11, Chan 2, Patch 7, Pan 64, BEATS 19 to 24

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	56	60	56	60	1.000000	0.500000
LENGTH (/ 1)	2	2	2		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	NO	YES	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

PITCH SELECTIONS:
56 60

Sect 2 Part 12, Chan 2, Patch 7, Pan 64, BEATS 25 to 30

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	58	62	58	62	1.000000	0.500000
LENGTH (/ 1)	2	2	2		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	NO	YES	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

PITCH SELECTIONS:
58 62

Sect 2 Part 13, Chan 2, Patch 7, Pan 64, BEATS 31 to 34

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	51	63	53	63	1.000000	20.000000
LENGTH (/ 1)	1	2	1	2	0.000000	20.000000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	YES	NO	NO	
LENGTH	YES	YES	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

PITCH SELECTIONS:
51 53 63
LENGTH SELECTIONS:
1 2

Sect 2 Part 14, Chan 2, Patch 7, Pan 64, BEATS 35 to 44

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	46	58	51	55	1.000000	20.000000
LENGTH (/ 1)	2	2	2		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	YES	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

PITCH SELECTIONS:
46 51 55 56 58

Sect 2 Part 15, Chan 1, Patch 7, Pan 64, BEATS 40 to 42

```
Parameter      Min       Max       MinMean    Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           68        72          72        72     1.000000 20.000000
LENGTH (/  1)    1         1           1                1.000000  0.500000

             REPEAT  | SELECT  |  REVERSE  | REFLECT  | RANDOM ENTRY
             ------------------------------------------------------
PITCH          NO    |  YES    |    NO     |   NO     |
LENGTH        YES    |   NO    |    NO     |   NO     |       NO
VELOCITY      YES    |   NO    |    NO     |   NO     |

PITCH SELECTIONS:
  68  70  72

Sect  2 Part 16, Chan  1, Patch   7, Pan  64, BEATS     43 to       43
===============
Parameter      Min       Max       MinMean    Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           67        68          67        67     1.000000 20.000000
LENGTH (/  3)    1         1           1                1.000000  0.500000

             REPEAT  | SELECT  |  REVERSE  | REFLECT  | RANDOM ENTRY
             ------------------------------------------------------
PITCH          NO    |  YES    |    NO     |   NO     |
LENGTH        YES    |   NO    |    NO     |   NO     |       NO
VELOCITY      YES    |   NO    |    NO     |   NO     |

PITCH SELECTIONS:
  67  68

Sect  2 Part 17, Chan  1, Patch   7, Pan  64, BEATS     44 to       46
===============
Parameter      Min       Max       MinMean    Start      Grad      Lambda
-------------------------------------------------------------------------
PITCH           63        65          63        65     1.000000  0.500000
LENGTH (/  1)    1         2           1         1     1.000000  0.500000

             REPEAT  | SELECT  |  REVERSE  | REFLECT  | RANDOM ENTRY
             ------------------------------------------------------
PITCH          NO    |  YES    |    NO     |   NO     |
LENGTH         NO    |  YES    |    NO     |   NO     |       NO
VELOCITY      YES    |   NO    |    NO     |   NO     |

PITCH SELECTIONS:
  63  65
LENGTH SELECTIONS:
   1   2
```

B.1.4 Dance Music

COMPOSITION FILE: Hard Disk:Music:dance

Section	Parts	Min Length	Min Total	Max Length	Max Total
1	16	448	448	448	448
2	9	161	609	161	609

SECTION SEQUENCE:
1 2

Tempo = 274 bpm MIDI Buffer Size = 2000 Bytes

SECTION: 1

Parameter	Min	Max	MinMean	Start	Grad	Lambda
TRANSPOSE	0	0	0		1.000000	0.500000

Part No	MIDI Channel	Starting Beat	Ending Beat
1	2	1	448
2	10	34	384
3	10	67	248
4	10	259	384
5	10	113	128
6	10	129	248
7	10	257	384
8	1	130	248
9	1	258	384
10	3	257	448
11	3	257	448
12	5	321	448
13	5	321	448
14	4	417	448
15	4	417	448
16	10	433	448

Sect 1 Part 1, Chan 2, Patch 82, Pan 64, BEATS 1 to 448

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	52	52	52		1.000000	0.500000
LENGTH (/ 2)	1	2	1	1	1.000000	0.500000
VOLUME	90	90	90		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	NO	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 1 Part 2, Chan 10, Patch 0, Pan 64, BEATS 34 to 384

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	42	42	42		1.000000	0.500000
LENGTH (/ 1)	2	2	2		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO

VELOCITY	YES	NO	NO	NO

Sect 1 Part 3, Chan 10, Patch 0, Pan 64, BEATS 67 to 248

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	39	39	39		1.000000	0.500000
LENGTH (/ 1)	4	4	4		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 1 Part 4, Chan 10, Patch 0, Pan 64, BEATS 259 to 384

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	39	39	39		1.000000	0.500000
LENGTH (/ 1)	4	4	4		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 1 Part 5, Chan 10, Patch 0, Pan 64, BEATS 113 to 128

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	38	38	38		1.000000	0.500000
LENGTH (/ 2)	1	1	1		1.000000	0.500000
VELOCITY	0	127	3	0	1.000000	20.000000
RELEASE	16	16	16		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	YES	NO	

Sect 1 Part 6, Chan 10, Patch 0, Pan 64, BEATS 129 to 248

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	36	36	36		1.000000	0.500000
LENGTH (/ 1)	2	2	2		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 1 Part 7, Chan 10, Patch 0, Pan 64, BEATS 257 to 384

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	36	36	36		1.000000	0.500000
LENGTH (/ 1)	2	2	2		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 1 Part 8, Chan 1, Patch 25, Pan 64, BEATS 130 to 248

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	28	28	28		1.000000	0.500000
LENGTH (/ 1)	2	2	2		1.000000	0.500000
RELEASE	16	16	16		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 1 Part 9, Chan 1, Patch 25, Pan 64, BEATS 258 to 384

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	28	28	28		1.000000	0.500000
LENGTH (/ 1)	2	2	2		1.000000	0.500000
RELEASE	16	16	16		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 1 Part 10, Chan 3, Patch 82, Pan 64, BEATS 257 to 448

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	66	72	67	69	1.000000	20.000000
LENGTH (/ 1)	8	8	8		1.000000	0.500000
VOLUME	0	0	0		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	YES	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

PITCH SELECTIONS:
66 67 69 72

Sect 1 Part 11, Chan 3, Patch 0, Pan 64, BEATS 257 to 448

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	60	60	60		1.000000	0.500000
LENGTH (/ 1)	1	1	1		1.000000	0.500000
VELOCITY	0	0	0		1.000000	0.500000
VOLUME	0	80	80	0	-1.000000	20.000000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 1 Part 12, Chan 5, Patch 82, Pan 64, BEATS 321 to 448

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	69	76	71	72	1.000000	20.000000
LENGTH (/ 1)	8	8	8		1.000000	0.500000
VOLUME	0	0	0		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	YES	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

PITCH SELECTIONS:
69 71 72 76

Sect 1 Part 13, Chan 5, Patch 0, Pan 64, BEATS 321 to 448

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	60	60	60		1.000000	0.500000
LENGTH (/ 1)	1	1	1		1.000000	0.500000
VELOCITY	0	0	0		1.000000	0.500000
VOLUME	0	80	80	0	-1.000000	20.000000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 1 Part 14, Chan 4, Patch 52, Pan 64, BEATS 417 to 448

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	64	64	64		1.000000	0.500000
LENGTH (/ 1)	32	32	32		1.000000	0.500000
VOLUME	0	0	0		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 1 Part 15, Chan 4, Patch 0, Pan 64, BEATS 417 to 448

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	60	60	60		1.000000	0.500000
LENGTH (/ 1)	1	1	1		1.000000	0.500000
VELOCITY	0	0	0		1.000000	0.500000
VOLUME	0	127	4	0	1.000000	20.000000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 1 Part 16, Chan 10, Patch 0, Pan 64, BEATS 433 to 448

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	38	38	38		1.000000	0.500000
LENGTH (/ 2)	1	1	1		1.000000	0.500000
VELOCITY	0	127	3	0	1.000000	20.000000
RELEASE	16	16	16		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	YES	NO	

SECTION: 2

Parameter	Min	Max	MinMean	Start	Grad	Lambda
TRANSPOSE	0	0	0		1.000000	0.500000

Part No	MIDI Channel	Starting Beat	Ending Beat
1	2	1	160
2	10	2	128
3	10	3	128
4	10	1	128
5	1	2	128
6	3	1	128
7	4	1	128
8	10	145	160
9	10	161	161

Sect 2 Part 1, Chan 2, Patch 82, Pan 64, BEATS 1 to 160

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	52	52	52		1.000000	0.500000
LENGTH (/ 2)	1	2	1	1	1.000000	0.500000
VOLUME	90	90	90		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	NO	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 2 Part 2, Chan 10, Patch 0, Pan 64, BEATS 2 to 128

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	42	42	42		1.000000	0.500000
LENGTH (/ 1)	2	2	2		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 2 Part 3, Chan 10, Patch 0, Pan 64, BEATS 3 to 128

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	39	39	39		1.000000	0.500000
LENGTH (/ 1)	4	4	4		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 2 Part 4, Chan 10, Patch 0, Pan 64, BEATS 1 to 128

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	36	36	36		1.000000	0.500000
LENGTH (/ 1)	2	2	2		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 2 Part 5, Chan 1, Patch 25, Pan 64, BEATS 2 to 128

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	28	28	28		1.000000	0.500000
LENGTH (/ 1)	2	2	2		1.000000	0.500000
RELEASE	16	16	16		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 2 Part 6, Chan 3, Patch 52, Pan 64, BEATS 1 to 128

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	57	64	64	64	1.000000	20.000000
LENGTH (/ 1)	4	12	4	12	1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	YES	YES	NO	
LENGTH	NO	YES	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

PITCH SELECTIONS:
57 59 60 62 64
LENGTH SELECTIONS:
4 12

Sect 2 Part 7, Chan 4, Patch 52, Pan 64, BEATS 1 to 128

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	52	59	59	59	1.000000	20.000000
LENGTH (/ 1)	16	16	16		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	YES	YES	NO	
LENGTH	NO	YES	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

PITCH SELECTIONS:

```
  52  55  59
LENGTH SELECTIONS:
  16
```

Sect 2 Part 8, Chan 10, Patch 0, Pan 64, BEATS 145 to 160

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	38	38	38		1.000000	0.500000
LENGTH (/ 2)	1	1	1		1.000000	0.500000
VELOCITY	0	127	3	0	1.000000	20.000000
RELEASE	16	16	16		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	YES	NO	

Sect 2 Part 9, Chan 10, Patch 0, Pan 64, BEATS 161 to 161

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	36	36	36		1.000000	0.500000
LENGTH (/ 1)	2	2	2		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

B.2.1 Markov-2

COMPOSITION FILE: Hard Disk:Music:markov-2

Section	Parts	Min Length	Min Total	Max Length	Max Total
1	2	160	160	160	160
2	2	160	320	160	320
3	2	160	480	160	480

SECTION SEQUENCE:
1 2 3

Tempo = 160 bpm MIDI Buffer Size = 1000 Bytes

SECTION: 1

Parameter	Min	Max	MinMean	Start	Grad	Lambda
TRANSPOSE	0	0	0		1.000000	0.500000

Part No	MIDI Channel	Starting Beat	Ending Beat
1	1	1	160
2	2	17	160

Sect 1 Part 1, Chan 1, Patch 46, Pan 24, BEATS 1 to 160

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	83	48		1.000000	0.500000
LENGTH (/ 32)	1	32	1		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	YES
VELOCITY	YES	NO	NO	NO	

Sect 1 Part 2, Chan 2, Patch 46, Pan 104, BEATS 17 to 160

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	83	48		1.000000	0.500000
LENGTH (/ 32)	1	32	1		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	YES
VELOCITY	YES	NO	NO	NO	

SECTION: 2

Parameter	Min	Max	MinMean	Start	Grad	Lambda
TRANSPOSE	0	0	0		1.000000	0.500000

Part No	MIDI Channel	Starting Beat	Ending Beat

Part No	MIDI Channel	Starting Beat	Ending Beat
1	1	9	160
2	2	25	160

Sect 2 Part 1, Chan 1, Patch 46, Pan 24, BEATS 9 to 160

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	83	48		1.000000	0.500000
LENGTH (/ 32)	1	32	1		-1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	YES
VELOCITY	YES	NO	NO	NO	

Sect 2 Part 2, Chan 2, Patch 46, Pan 104, BEATS 25 to 160

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	83	48		1.000000	0.500000
LENGTH (/ 32)	1	32	1		-1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	YES
VELOCITY	YES	NO	NO	NO	

SECTION: 3

Parameter	Min	Max	MinMean	Start	Grad	Lambda
TRANSPOSE	0	0	0		1.000000	0.500000

Part No	MIDI Channel	Starting Beat	Ending Beat
1	1	9	160
2	2	25	160

Sect 3 Part 1, Chan 1, Patch 46, Pan 24, BEATS 9 to 160

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	83	48		-1.000000	0.500000
LENGTH (/ 32)	1	32	1		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	YES
VELOCITY	YES	NO	NO	NO	

Sect 3 Part 2, Chan 2, Patch 46, Pan 104, BEATS 25 to 160

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	83	48		-1.000000	0.500000
LENGTH (/ 32)	1	32	1		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	YES

VELOCITY	YES	NO	NO	NO

B.2.2 Computer Study for Timpani

COMPOSITION FILE: Hard Disk:Music:computer_study_for_timpani

Section	Parts	Min Length	Min Total	Max Length	Max Total
1	1	30	30	30	30
2	1	30	60	30	60
3	1	30	90	30	90
4	1	30	120	30	120
5	2	37	157	37	157
6	2	12	169	12	169
7	3	9	178	9	178
8	4	10	188	10	188
9	2	7	195	7	195
10	2	13	208	13	208
11	3	10	218	10	218
12	4	9	227	9	227
13	2	7	234	7	234
14	12	366	600	366	600

SECTION SEQUENCE:
1 2 3 4 5 6 7 8 9 10 11 12 13 14

Tempo = 150 bpm MIDI Buffer Size = 5000 Bytes

SECTION: 1

Parameter	Min	Max	MinMean	Start	Grad	Lambda
TRANSPOSE	0	0	0		1.000000	0.500000

Part No	MIDI Channel	Starting Beat	Ending Beat
1	1	1	30

Sect 1 Part 1, Chan 1, Patch 48, Pan 0, BEATS 1 to 30

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	48	48		1.000000	0.500000
LENGTH (/ 16)	1	16	1		1.000000	0.500000
VELOCITY	1	127	1		1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

SECTION: 2

Parameter	Min	Max	MinMean	Start	Grad	Lambda
TRANSPOSE	0	0	0		1.000000	0.500000

Part No	MIDI Channel	Starting Beat	Ending Beat
1	2	5	30

Sect 2 Part 1, Chan 2, Patch 48, Pan 127, BEATS 5 to 30

Parameter	Min	Max	MinMean	Start	Grad	Lambda

PITCH	63	63	63	1.000000	0.500000
LENGTH (/ 16)	1	16	1	1.000000	0.500000
VELOCITY	1	127	1	1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

SECTION: 3

Parameter	Min	Max	MinMean	Start	Grad	Lambda
TRANSPOSE	0	0	0		1.000000	0.500000

Part No	MIDI Channel	Starting Beat	Ending Beat
1	1	5	30

Sect 3 Part 1, Chan 1, Patch 48, Pan 0, BEATS 5 to 30

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	60	48		1.000000	0.500000
LENGTH (/ 16)	1	16	1		1.000000	0.500000
VELOCITY	1	127	1		1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

SECTION: 4

Parameter	Min	Max	MinMean	Start	Grad	Lambda
TRANSPOSE	0	0	0		1.000000	0.500000

Part No	MIDI Channel	Starting Beat	Ending Beat
1	2	5	30

Sect 4 Part 1, Chan 2, Patch 48, Pan 127, BEATS 5 to 30

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	60	72	60		1.000000	0.500000
LENGTH (/ 16)	1	16	1		1.000000	0.500000
VELOCITY	1	127	1		1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

SECTION: 5

Parameter	Min	Max	MinMean	Start	Grad	Lambda
TRANSPOSE	0	0	0		1.000000	0.500000

Part No	MIDI Channel	Starting Beat	Ending Beat

1	1	5	37
2	2	5	37

Sect 5 Part 1, Chan 1, Patch 48, Pan 0, BEATS 5 to 37

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	60	48		1.000000	0.500000
LENGTH (/ 16)	1	16	1		1.000000	0.500000
VELOCITY	1	127	1		1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 5 Part 2, Chan 2, Patch 48, Pan 127, BEATS 5 to 37

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	60	72	60		1.000000	0.500000
LENGTH (/ 16)	1	16	1		1.000000	0.500000
VELOCITY	1	127	1		1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

SECTION: 6

Parameter	Min	Max	MinMean	Start	Grad	Lambda
TRANSPOSE	0	0	0		1.000000	0.500000

Part No	MIDI Channel	Starting Beat	Ending Beat
1	1	7	10
2	2	11	12

Sect 6 Part 1, Chan 1, Patch 48, Pan 0, BEATS 7 to 10

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	60	48		1.000000	0.500000
LENGTH (/ 90)	10	15	10		1.000000	0.500000
VELOCITY	10	32	10		1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 6 Part 2, Chan 2, Patch 48, Pan 127, BEATS 11 to 12

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	60	72	60		1.000000	0.500000
LENGTH (/ 90)	10	15	10		1.000000	0.500000
VELOCITY	10	32	10		1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

SECTION: 7

Parameter	Min	Max	MinMean	Start	Grad	Lambda
TRANSPOSE	0	0	0		1.000000	0.500000

Part No	MIDI Channel	Starting Beat	Ending Beat
1	1	2	5
2	2	6	7
3	1	8	9

Sect 7 Part 1, Chan 1, Patch 48, Pan 0, BEATS 2 to 5

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	60	48		1.000000	0.500000
LENGTH (/ 90)	10	15	10		1.000000	0.500000
VELOCITY	33	64	33		1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 7 Part 2, Chan 2, Patch 48, Pan 127, BEATS 6 to 7

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	60	72	60		1.000000	0.500000
LENGTH (/ 90)	10	15	10		1.000000	0.500000
VELOCITY	33	64	33		1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 7 Part 3, Chan 1, Patch 48, Pan 0, BEATS 8 to 9

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	60	48		1.000000	0.500000
LENGTH (/ 90)	10	15	10		1.000000	0.500000
VELOCITY	33	64	33		1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

SECTION: 8

Parameter	Min	Max	MinMean	Start	Grad	Lambda

TRANSPOSE 0 0 0 1.000000 0.500000

Part No	MIDI Channel	Starting Beat	Ending Beat
1	1	3	4
2	2	5	6
3	1	7	8
4	2	9	10

Sect 8 Part 1, Chan 1, Patch 48, Pan 0, BEATS 3 to 4

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	60	48		1.000000	0.500000
LENGTH (/ 90)	10	15	10		1.000000	0.500000
VELOCITY	65	96	65		1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 8 Part 2, Chan 2, Patch 48, Pan 127, BEATS 5 to 6

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	60	72	60		1.000000	0.500000
LENGTH (/ 90)	10	15	10		1.000000	0.500000
VELOCITY	65	96	65		1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 8 Part 3, Chan 1, Patch 48, Pan 0, BEATS 7 to 8

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	60	48		1.000000	0.500000
LENGTH (/ 90)	10	15	10		1.000000	0.500000
VELOCITY	65	96	65		1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 8 Part 4, Chan 2, Patch 48, Pan 127, BEATS 9 to 10

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	60	72	60		1.000000	0.500000
LENGTH (/ 90)	10	15	10		1.000000	0.500000
VELOCITY	65	96	65		1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

SECTION: 9

Parameter	Min	Max	MinMean	Start	Grad	Lambda
TRANSPOSE	0	0	0		1.000000	0.500000

Part No	MIDI Channel	Starting Beat	Ending Beat
1	1	3	7
2	2	3	7

Sect 9 Part 1, Chan 1, Patch 48, Pan 0, BEATS 3 to 7

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	60	48		1.000000	0.500000
LENGTH (/ 90)	10	15	10		1.000000	0.500000
VELOCITY	97	127	97		1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 9 Part 2, Chan 2, Patch 48, Pan 127, BEATS 3 to 7

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	60	72	60		1.000000	0.500000
LENGTH (/ 90)	10	15	10		1.000000	0.500000
VELOCITY	97	127	97		1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

SECTION: 10

Parameter	Min	Max	MinMean	Start	Grad	Lambda
TRANSPOSE	0	0	0		1.000000	0.500000

Part No	MIDI Channel	Starting Beat	Ending Beat
1	1	7	11
2	2	12	13

Sect 10 Part 1, Chan 1, Patch 48, Pan 0, BEATS 7 to 11

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	60	48		1.000000	0.500000
LENGTH (/ 90)	10	15	10		1.000000	0.500000
VELOCITY	10	32	10		1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 10 Part 2, Chan 2, Patch 48, Pan 127, BEATS 12 to 13

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	60	72	60		1.000000	0.500000
LENGTH (/ 90)	10	15	10		1.000000	0.500000
VELOCITY	10	32	10		1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

SECTION: 11

Parameter	Min	Max	MinMean	Start	Grad	Lambda
TRANSPOSE	0	0	0		1.000000	0.500000

Part No	MIDI Channel	Starting Beat	Ending Beat
1	1	2	6
2	2	7	8
3	1	9	10

Sect 11 Part 1, Chan 1, Patch 48, Pan 0, BEATS 2 to 6

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	60	48		1.000000	0.500000
LENGTH (/ 90)	10	15	10		1.000000	0.500000
VELOCITY	33	64	33		1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 11 Part 2, Chan 2, Patch 48, Pan 127, BEATS 7 to 8

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	60	72	60		1.000000	0.500000
LENGTH (/ 90)	10	15	10		1.000000	0.500000
VELOCITY	33	64	33		1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 11 Part 3, Chan 1, Patch 48, Pan 0, BEATS 9 to 10

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	60	48		1.000000	0.500000
LENGTH (/ 90)	10	15	10		1.000000	0.500000
VELOCITY	33	64	33		1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	

LENGTH	YES	NO	NO	NO		NO
VELOCITY	YES	NO	NO	NO		

SECTION: 12

Parameter	Min	Max	MinMean	Start	Grad	Lambda
TRANSPOSE	0	0	0		1.000000	0.500000

Part No	MIDI Channel	Starting Beat	Ending Beat
1	1	2	3
2	2	4	5
3	1	6	7
4	2	8	9

Sect 12 Part 1, Chan 1, Patch 48, Pan 0, BEATS 2 to 3

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	60	48		1.000000	0.500000
LENGTH (/ 90)	10	15	10		1.000000	0.500000
VELOCITY	65	96	65		1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 12 Part 2, Chan 2, Patch 48, Pan 127, BEATS 4 to 5

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	60	72	60		1.000000	0.500000
LENGTH (/ 90)	10	15	10		1.000000	0.500000
VELOCITY	65	96	65		1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 12 Part 3, Chan 1, Patch 48, Pan 0, BEATS 6 to 7

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	60	48		1.000000	0.500000
LENGTH (/ 90)	10	15	10		1.000000	0.500000
VELOCITY	65	96	65		1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 12 Part 4, Chan 2, Patch 48, Pan 127, BEATS 8 to 9

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	60	72	60		1.000000	0.500000
LENGTH (/ 90)	10	15	10		1.000000	0.500000

VELOCITY	65	96	65	1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

SECTION: 13

Parameter	Min	Max	MinMean	Start	Grad	Lambda
TRANSPOSE	0	0	0		1.000000	0.500000

Part No	MIDI Channel	Starting Beat	Ending Beat
1	1	3	7
2	2	3	7

Sect 13 Part 1, Chan 1, Patch 48, Pan 0, BEATS 3 to 7

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	60	48		1.000000	0.500000
LENGTH (/ 90)	10	15	10		1.000000	0.500000
VELOCITY	97	127	97		1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 13 Part 2, Chan 2, Patch 48, Pan 127, BEATS 3 to 7

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	60	72	60		1.000000	0.500000
LENGTH (/ 90)	10	15	10		1.000000	0.500000
VELOCITY	97	127	97		1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

SECTION: 14

Parameter	Min	Max	MinMean	Start	Grad	Lambda
TRANSPOSE	0	0	0		1.000000	0.500000

Part No	MIDI Channel	Starting Beat	Ending Beat
1	1	7	126
2	2	37	156
3	3	60	179
4	4	90	209
5	1	127	246
6	2	157	276
7	3	180	299
8	4	210	329
9	1	247	366
10	2	277	366

11	3	300	366
12	4	330	366

Sect 14 Part 1, Chan 1, Patch 48, Pan 0, BEATS 7 to 126

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	48	48		1.000000	0.500000
LENGTH (/120)	50	50	50		1.000000	0.500000
VELOCITY	50	127	127		-1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 14 Part 2, Chan 2, Patch 48, Pan 127, BEATS 37 to 156

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	57	57	57		1.000000	0.500000
LENGTH (/120)	50	50	50		1.000000	0.500000
VELOCITY	50	127	127		-1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 14 Part 3, Chan 3, Patch 48, Pan 42, BEATS 60 to 179

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	54	54	54		1.000000	0.500000
LENGTH (/120)	50	50	50		1.000000	0.500000
VELOCITY	50	127	127		-1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 14 Part 4, Chan 4, Patch 48, Pan 84, BEATS 90 to 209

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	51	51	51		1.000000	0.500000
LENGTH (/120)	50	50	50		1.000000	0.500000
VELOCITY	50	127	127		-1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 14 Part 5, Chan 1, Patch 48, Pan 0, BEATS 127 to 246

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	69	69	69		1.000000	0.500000

LENGTH (/120)	50	50	50	1.000000	0.500000
VELOCITY	50	127	127	-1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 14 Part 6, Chan 2, Patch 48, Pan 127, BEATS 157 to 276

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	66	66	66		1.000000	0.500000
LENGTH (/120)	50	50	50		1.000000	0.500000
VELOCITY	50	127	127		-1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 14 Part 7, Chan 3, Patch 48, Pan 42, BEATS 180 to 299

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	63	63	63		1.000000	0.500000
LENGTH (/120)	50	50	50		1.000000	0.500000
VELOCITY	50	127	127		-1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 14 Part 8, Chan 4, Patch 48, Pan 84, BEATS 210 to 329

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	60	60	60		1.000000	0.500000
LENGTH (/120)	50	50	50		1.000000	0.500000
VELOCITY	50	127	127		-1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 14 Part 9, Chan 1, Patch 48, Pan 0, BEATS 247 to 366

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	71	48	69	1.000000	0.500000
LENGTH (/120)	50	50	50		1.000000	0.500000
VELOCITY	50	127	127		-1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 14 Part 10, Chan 2, Patch 48, Pan 127, BEATS 277 to 366

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	71	48	66	1.000000	0.500000
LENGTH (/120)	50	50	50		1.000000	0.500000
VELOCITY	50	127	127		-1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 14 Part 11, Chan 3, Patch 48, Pan 42, BEATS 300 to 366

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	71	48	63	1.000000	0.500000
LENGTH (/120)	50	50	50		1.000000	0.500000
VELOCITY	50	127	127		-1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 14 Part 12, Chan 4, Patch 48, Pan 84, BEATS 330 to 366

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	71	48	60	1.000000	0.500000
LENGTH (/120)	50	50	50		1.000000	0.500000
VELOCITY	50	127	127		-1.000000	0.050000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

B.2.3 Vibrato Study

COMPOSITION FILE: Hard Disk:Music:vibrato_study

Section	Parts	Min Length	Min Total	Max Length	Max Total
1	11	82	82	82	82
2	7	90	172	90	172
3	9	150	322	150	322

SECTION SEQUENCE:
1 2 3

Tempo = 60 bpm MIDI Buffer Size = 100 Bytes

SECTION: 1

Parameter	Min	Max	MinMean	Start	Grad	Lambda
TRANSPOSE	0	0	0		1.000000	0.500000

Part No	MIDI Channel	Starting Beat	Ending Beat
1	5	1	80
2	6	1	80
3	7	1	80
4	1	13	18
5	2	25	30
6	3	37	42
7	1	49	54
8	1	67	72
9	2	67	72
10	3	79	82
11	4	79	82

Sect 1 Part 1, Chan 5, Patch 54, Pan 0, BEATS 1 to 80

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	38	38	38		1.000000	0.500000
LENGTH (/ 1)	80	80	80		1.000000	0.500000
RELEASE	104	104	104		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 1 Part 2, Chan 6, Patch 54, Pan 64, BEATS 1 to 80

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	45	45	45		1.000000	0.500000
LENGTH (/ 1)	80	80	80		1.000000	0.500000
RELEASE	104	104	104		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 1 Part 3, Chan 7, Patch 54, Pan 127, BEATS 1 to 80

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	52	52	52		1.000000	0.500000
LENGTH (/ 1)	80	80	80		1.000000	0.500000
RELEASE	104	104	104		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 1 Part 4, Chan 1, Patch 54, Pan 4, BEATS 13 to 18

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	84	60		1.000000	0.500000
LENGTH (/ 1)	6	6	6		1.000000	0.500000
VIBDEPTH	114	114	114		1.000000	0.500000
VIBRATE	14	114	14		1.000000	0.050000
RELEASE	104	104	104		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 1 Part 5, Chan 2, Patch 54, Pan 44, BEATS 25 to 30

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	84	60		1.000000	0.500000
LENGTH (/ 1)	6	6	6		1.000000	0.500000
VIBDEPTH	114	114	114		1.000000	0.500000
VIBRATE	14	114	14		1.000000	0.050000
RELEASE	104	104	104		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 1 Part 6, Chan 3, Patch 54, Pan 84, BEATS 37 to 42

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	84	60		1.000000	0.500000
LENGTH (/ 1)	6	6	6		1.000000	0.500000
VIBDEPTH	114	114	114		1.000000	0.500000
VIBRATE	14	114	14		1.000000	0.050000
RELEASE	104	104	104		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 1 Part 7, Chan 1, Patch 54, Pan 124, BEATS 49 to 54

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	84	60		1.000000	0.500000
LENGTH (/ 1)	6	6	6		1.000000	0.500000
VIBDEPTH	114	114	114		1.000000	0.500000
VIBRATE	14	114	14		1.000000	0.050000
RELEASE	104	104	104		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 1 Part 8, Chan 1, Patch 54, Pan 4, BEATS 67 to 72

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	84	60		1.000000	0.500000
LENGTH (/ 1)	6	6	6		1.000000	0.500000
VIBDEPTH	114	114	114		1.000000	0.500000
VIBRATE	14	114	14		1.000000	0.050000
RELEASE	104	104	104		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 1 Part 9, Chan 2, Patch 54, Pan 84, BEATS 67 to 72

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	84	60		1.000000	0.500000
LENGTH (/ 1)	6	6	6		1.000000	0.500000
VIBDEPTH	114	114	114		1.000000	0.500000
VIBRATE	14	114	14		1.000000	0.050000
RELEASE	104	104	104		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 1 Part 10, Chan 3, Patch 54, Pan 44, BEATS 79 to 82

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	84	60		1.000000	0.500000
LENGTH (/ 1)	6	6	6		1.000000	0.500000
VIBDEPTH	114	114	114		1.000000	0.500000
VIBRATE	23	23	23		1.000000	0.050000
RELEASE	104	104	104		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 1 Part 11, Chan 4, Patch 54, Pan 124, BEATS 79 to 82

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	84	60		1.000000	0.500000
LENGTH (/ 1)	6	6	6		1.000000	0.500000
VIBDEPTH	114	114	114		1.000000	0.500000
VIBRATE	23	23	23		1.000000	0.050000
RELEASE	104	104	104		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

SECTION: 2

Parameter	Min	Max	MinMean	Start	Grad	Lambda
TRANSPOSE	0	0	0		1.000000	0.500000

Part No	MIDI Channel	Starting Beat	Ending Beat
1	5	1	90
2	6	1	90
3	7	1	90
4	1	7	90
5	2	13	90
6	3	19	90
7	4	25	90

Sect 2 Part 1, Chan 5, Patch 54, Pan 0, BEATS 1 to 90

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	38	38	38		1.000000	0.500000
LENGTH (/ 1)	90	90	90		1.000000	0.500000
RELEASE	104	104	104		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 2 Part 2, Chan 6, Patch 54, Pan 64, BEATS 1 to 90

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	45	45	45		1.000000	0.500000
LENGTH (/ 1)	90	90	90		1.000000	0.500000
RELEASE	104	104	104		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 2 Part 3, Chan 7, Patch 54, Pan 127, BEATS 1 to 90

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	52	52	52		1.000000	0.500000
LENGTH (/ 1)	90	90	90		1.000000	0.500000
RELEASE	104	104	104		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 2 Part 4, Chan 1, Patch 54, Pan 4, BEATS 7 to 90

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	84	60		1.000000	0.500000
LENGTH (/ 1)	6	8	6		1.000000	0.500000
VIBDEPTH	114	114	114		1.000000	0.500000
VIBRATE	14	114	14		1.000000	0.050000
RELEASE	104	104	104		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 2 Part 5, Chan 2, Patch 54, Pan 44, BEATS 13 to 90

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	84	60		1.000000	0.500000
LENGTH (/ 1)	6	8	6		1.000000	0.500000
VIBDEPTH	114	114	114		1.000000	0.500000
VIBRATE	14	114	14		1.000000	0.050000
RELEASE	104	104	104		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 2 Part 6, Chan 3, Patch 54, Pan 84, BEATS 19 to 90

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	84	60		1.000000	0.500000
LENGTH (/ 1)	6	8	6		1.000000	0.500000
VIBDEPTH	114	114	114		1.000000	0.500000
VIBRATE	14	114	14		1.000000	0.050000
RELEASE	104	104	104		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 2 Part 7, Chan 4, Patch 54, Pan 84, BEATS 25 to 90

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	84	60		1.000000	0.500000
LENGTH (/ 1)	6	8	6		1.000000	0.500000
VIBDEPTH	114	114	114		1.000000	0.500000
VIBRATE	14	114	14		1.000000	0.050000
RELEASE	104	104	104		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

SECTION: 3

Parameter	Min	Max	MinMean	Start	Grad	Lambda
TRANSPOSE	0	0	0		1.000000	0.500000

Part No	MIDI Channel	Starting Beat	Ending Beat
1	5	3	150
2	6	19	150
3	7	37	150
4	8	55	120
5	9	61	120
6	1	127	150
7	2	130	150
8	3	133	150
9	4	136	150

Sect 3 Part 1, Chan 5, Patch 54, Pan 0, BEATS 3 to 150

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	36	47	36		1.000000	0.500000
LENGTH (/ 1)	148	148	148		1.000000	0.500000
VIBDEPTH	114	114	114		1.000000	0.500000
VIBRATE	31	31	31		1.000000	0.050000
RELEASE	104	104	104		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 3 Part 2, Chan 6, Patch 54, Pan 64, BEATS 19 to 150

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	59	48		1.000000	0.500000
LENGTH (/ 1)	132	132	132		1.000000	0.500000
VELOCITY	100	100	100		1.000000	0.500000
VIBDEPTH	114	114	114		1.000000	0.500000
VIBRATE	41	41	41		1.000000	0.050000
RELEASE	104	104	104		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 3 Part 3, Chan 7, Patch 54, Pan 127, BEATS 37 to 150

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	60	71	60		1.000000	0.500000
LENGTH (/ 1)	114	114	114		1.000000	0.500000
VELOCITY	90	90	90		1.000000	0.500000

VIBDEPTH	114	114	114	1.000000	0.500000
VIBRATE	61	61	61	1.000000	0.050000
RELEASE	104	104	104	1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 3 Part 4, Chan 8, Patch 54, Pan 34, BEATS 55 to 120

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	60	84	60		1.000000	0.500000
LENGTH (/ 2)	1	2	1		1.000000	0.500000
VELOCITY	40	90	40		1.000000	0.100000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 3 Part 5, Chan 9, Patch 54, Pan 94, BEATS 61 to 120

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	60	84	60		1.000000	0.500000
LENGTH (/ 4)	1	2	1		1.000000	0.500000
VELOCITY	40	90	40		1.000000	0.100000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 3 Part 6, Chan 1, Patch 54, Pan 4, BEATS 127 to 150

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	36	47	36		1.000000	0.500000
LENGTH (/ 1)	89	89	89		1.000000	0.500000
VIBDEPTH	114	114	114		1.000000	0.500000
VIBRATE	36	36	36		1.000000	0.500000
RELEASE	104	104	104		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 3 Part 7, Chan 2, Patch 54, Pan 44, BEATS 130 to 150

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	48	59	48		1.000000	0.500000
LENGTH (/ 1)	72	72	72		1.000000	0.500000
VELOCITY	100	100	100		1.000000	0.500000
VIBDEPTH	114	114	114		1.000000	0.500000
VIBRATE	46	46	46		1.000000	0.500000
RELEASE	104	104	104		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 3 Part 8, Chan 3, Patch 54, Pan 84, BEATS 133 to 150

Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	60	71	60		1.000000	0.500000
LENGTH (/ 1)	54	54	54		1.000000	0.500000
VELOCITY	90	90	90		1.000000	0.500000
VIBDEPTH	114	114	114		1.000000	0.500000
VIBRATE	51	51	51		1.000000	0.500000
RELEASE	104	104	104		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

Sect 3 Part 9, Chan 4, Patch 54, Pan 124, BEATS 136 to 150

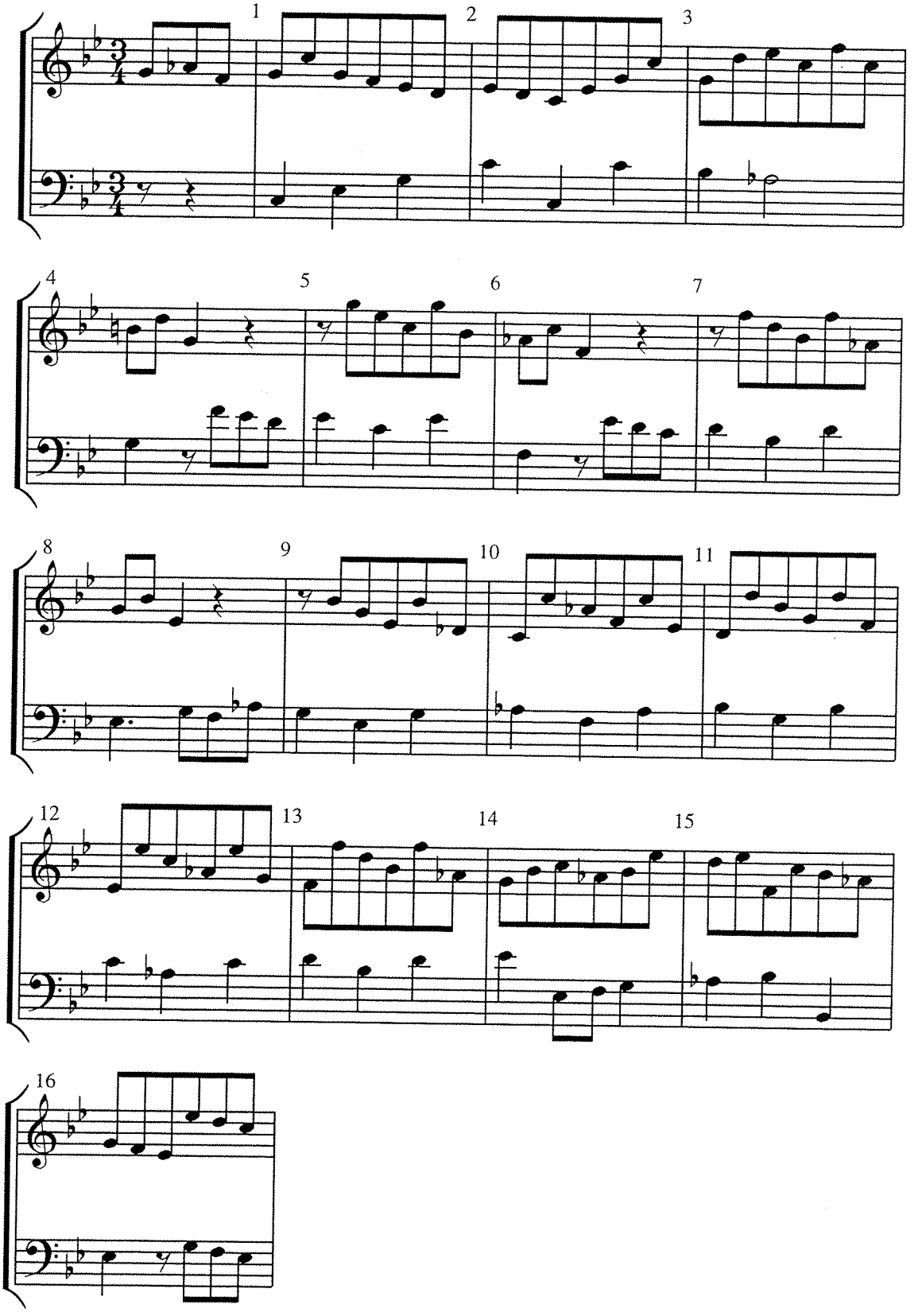
Parameter	Min	Max	MinMean	Start	Grad	Lambda
PITCH	72	83	72		1.000000	0.500000
LENGTH (/ 1)	54	54	54		1.000000	0.500000
VELOCITY	90	90	90		1.000000	0.500000
VIBDEPTH	114	114	114		1.000000	0.500000
VIBRATE	56	56	56		1.000000	0.500000
RELEASE	104	104	104		1.000000	0.500000

	REPEAT	SELECT	REVERSE	REFLECT	RANDOM ENTRY
PITCH	YES	NO	NO	NO	
LENGTH	YES	NO	NO	NO	NO
VELOCITY	YES	NO	NO	NO	

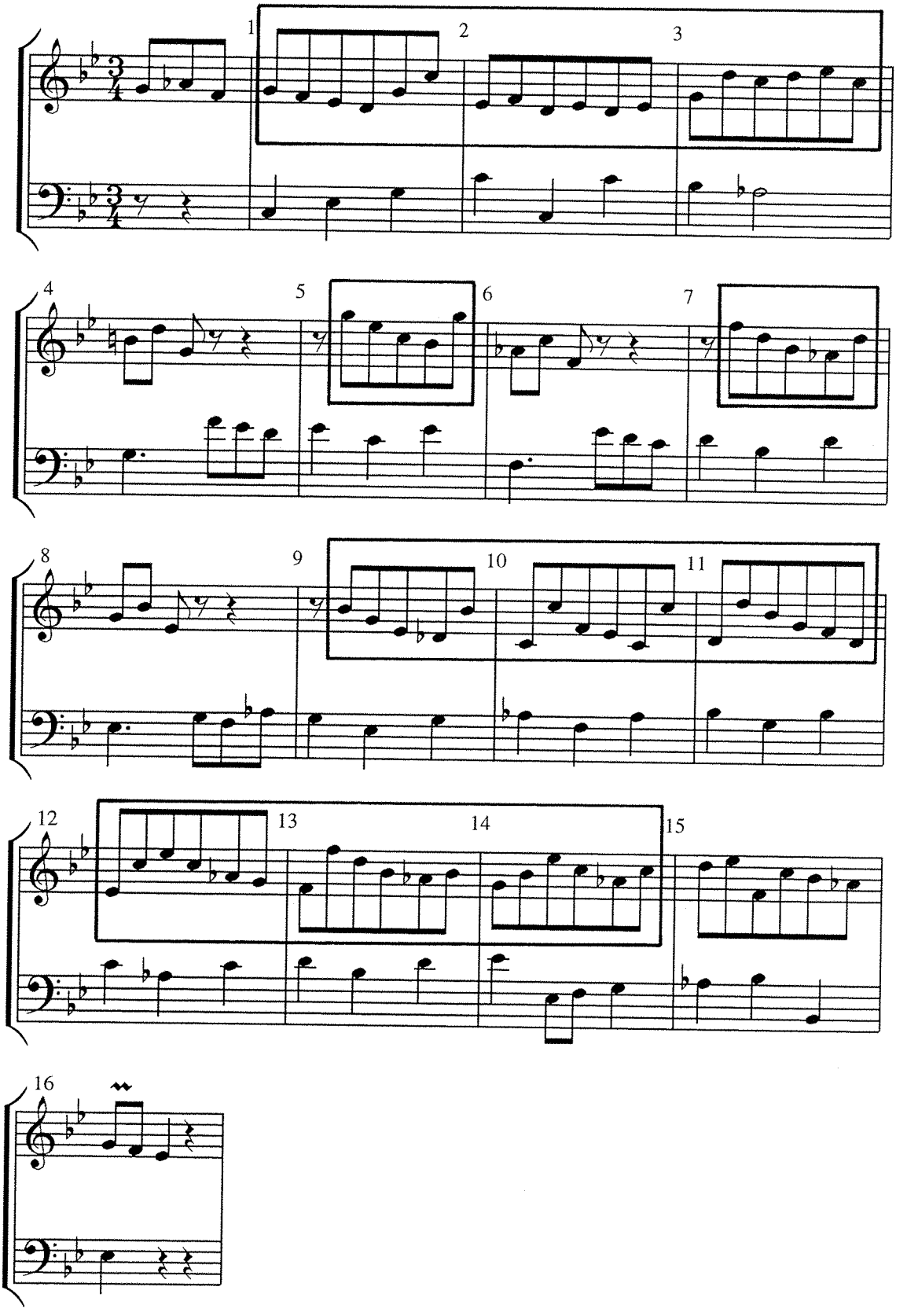
Appendix C

Bach Scores in Staff Notation

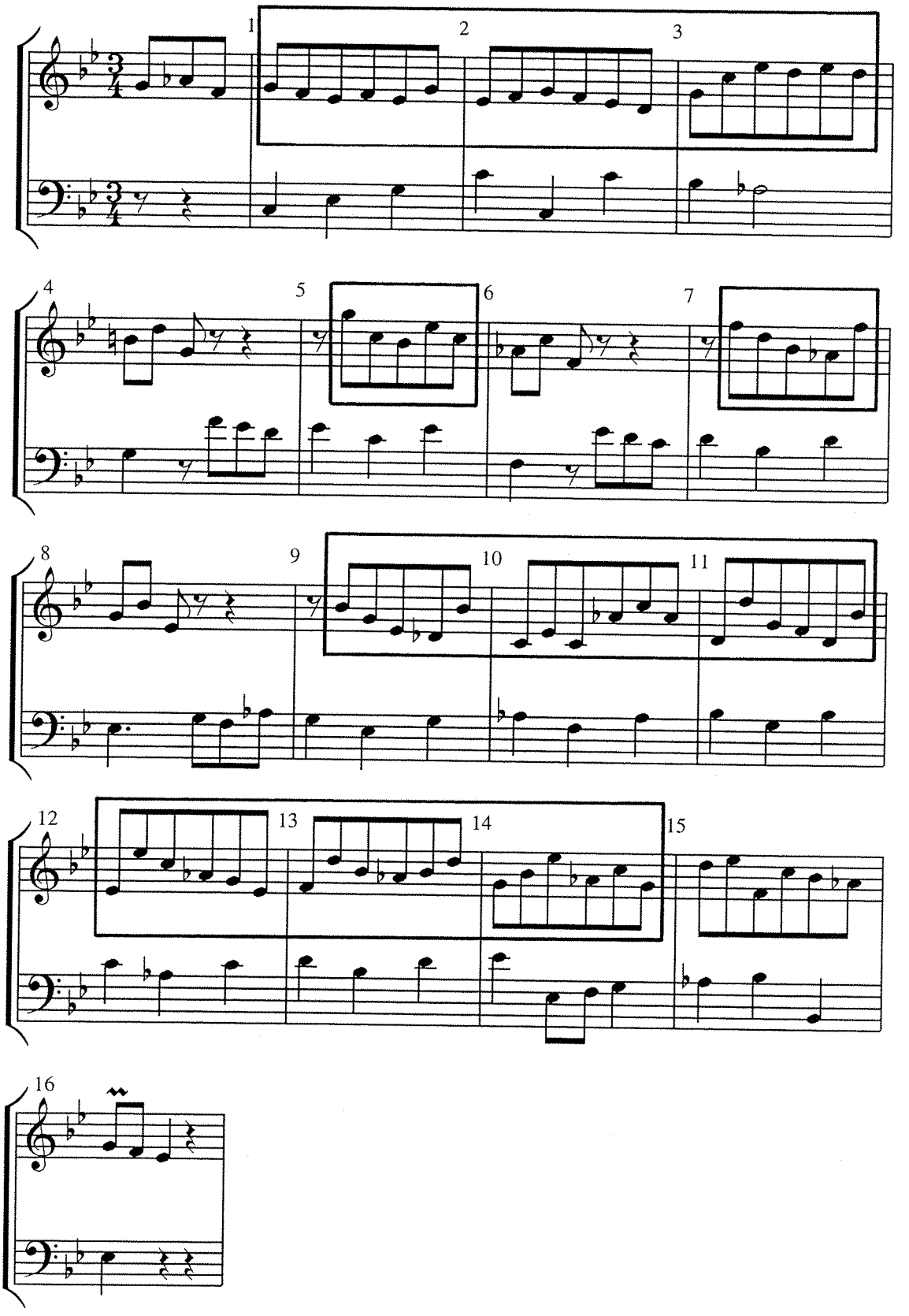
Bach - Original Score



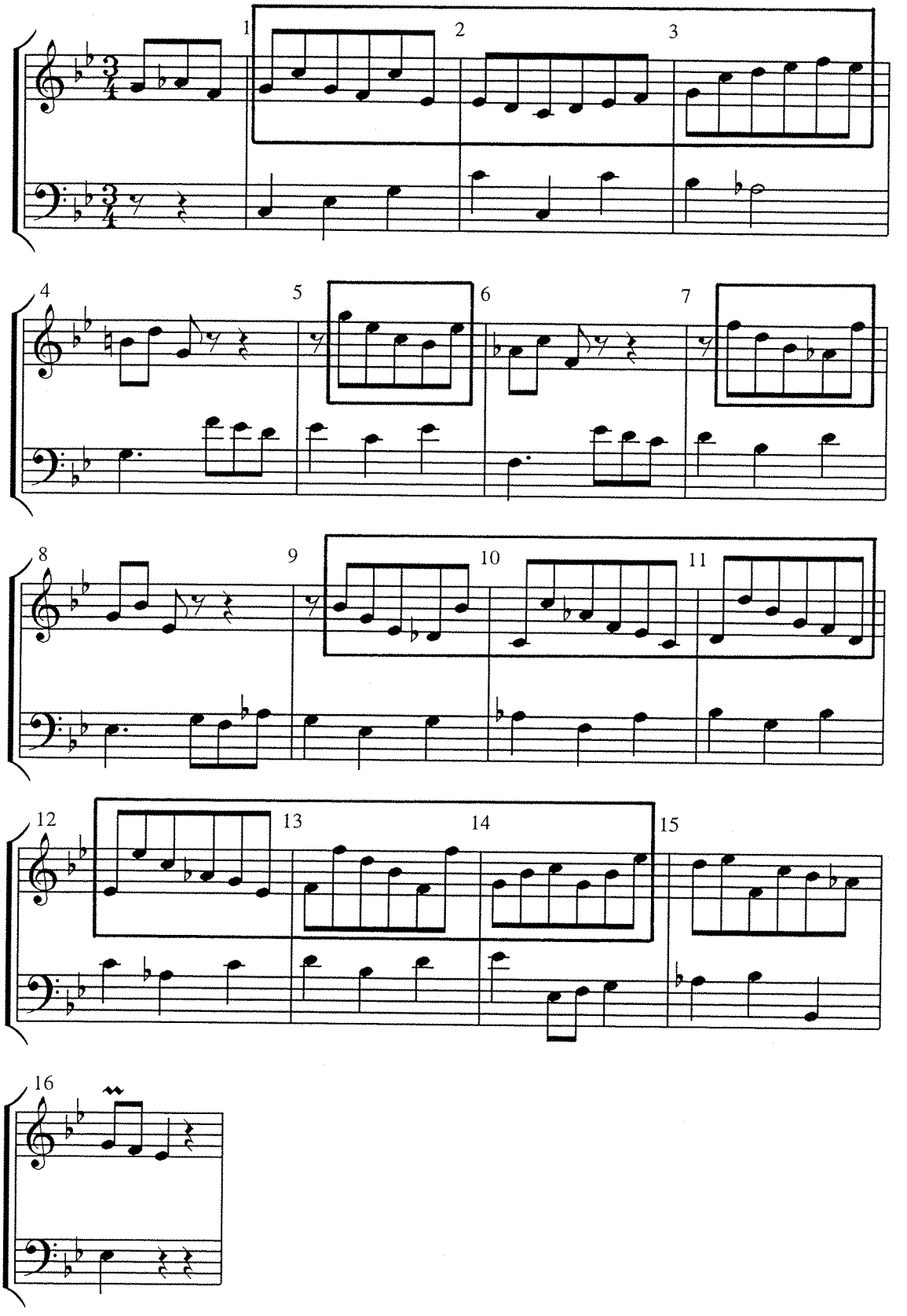
Bach1



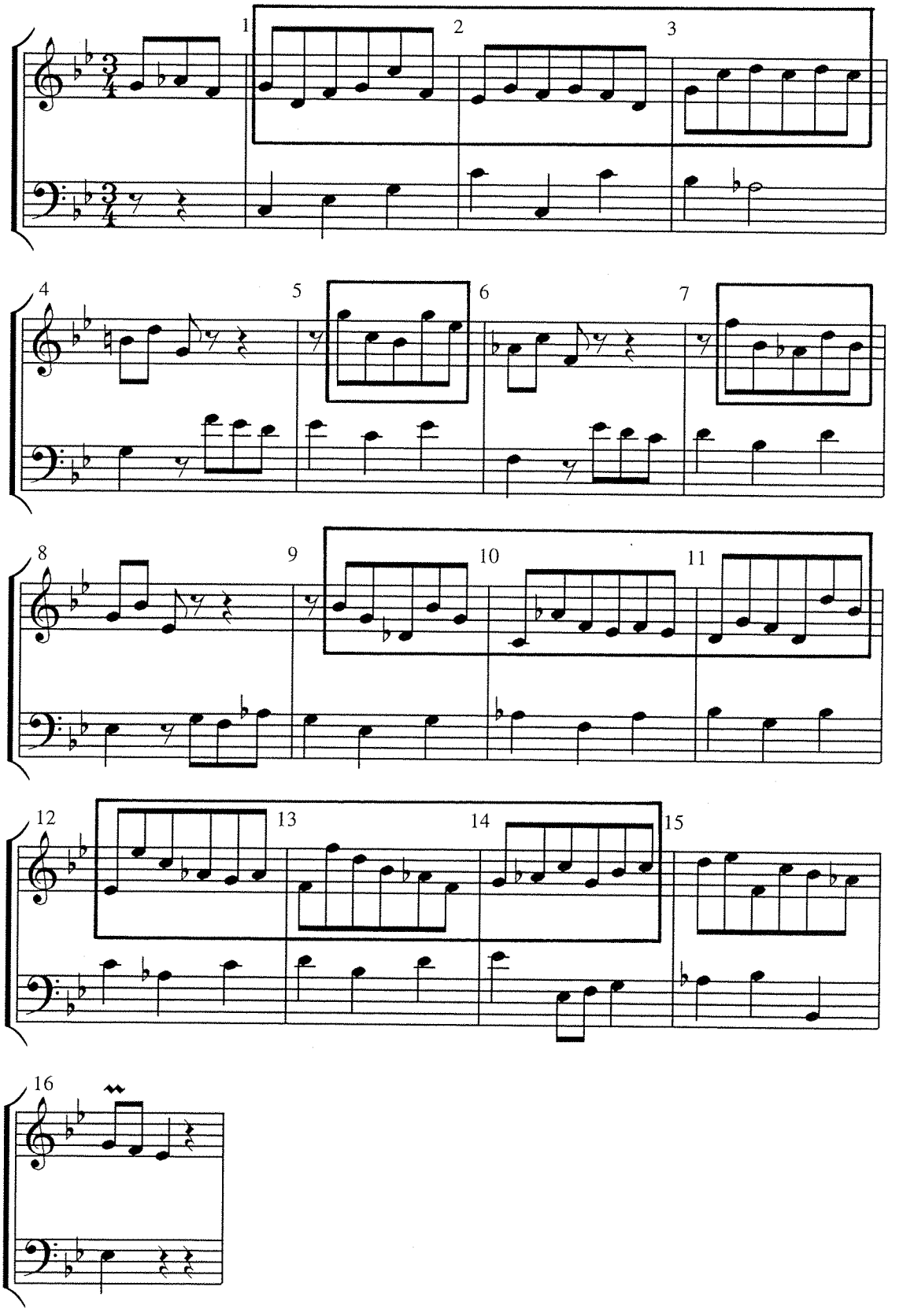
Bach2



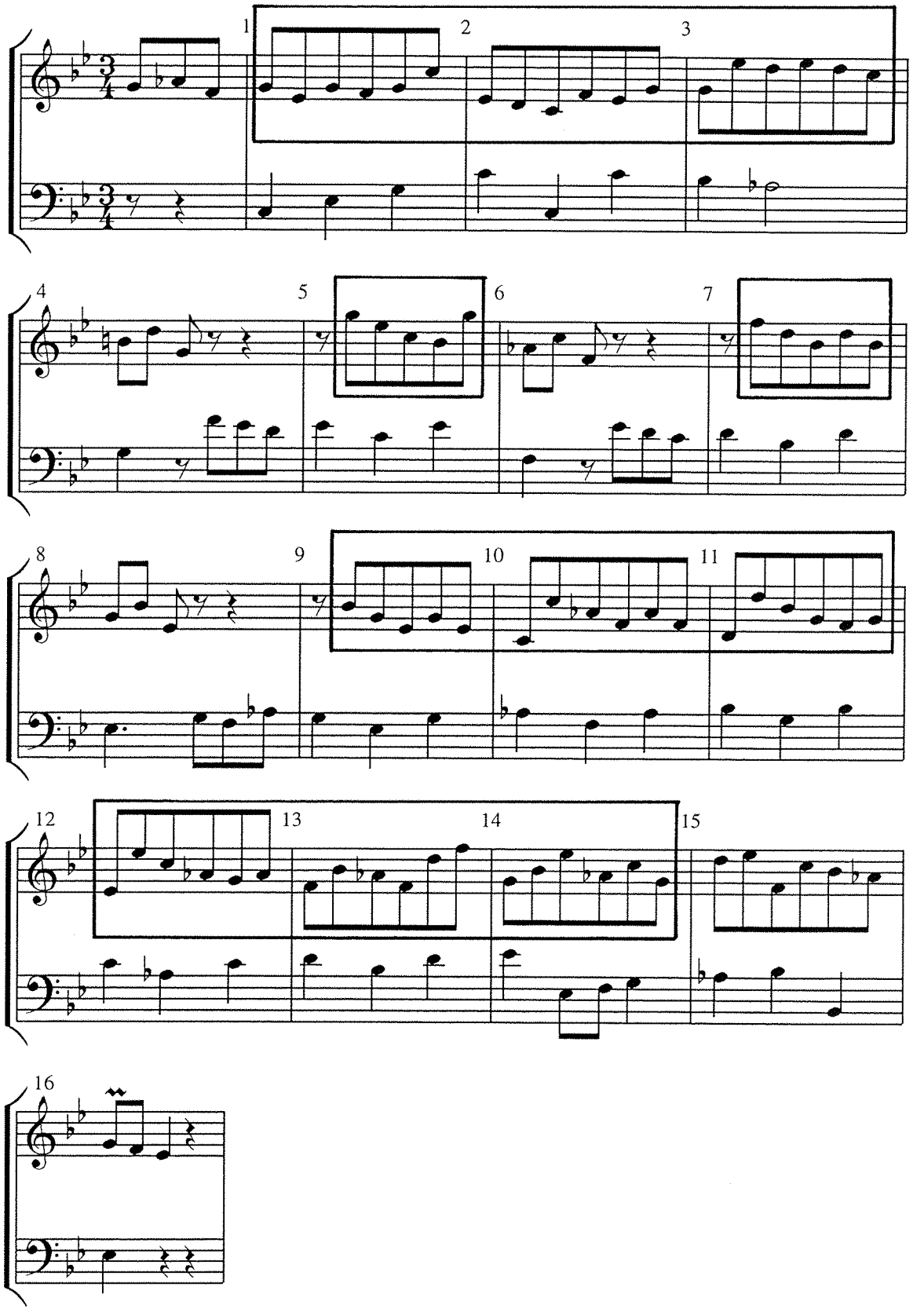
Bach3



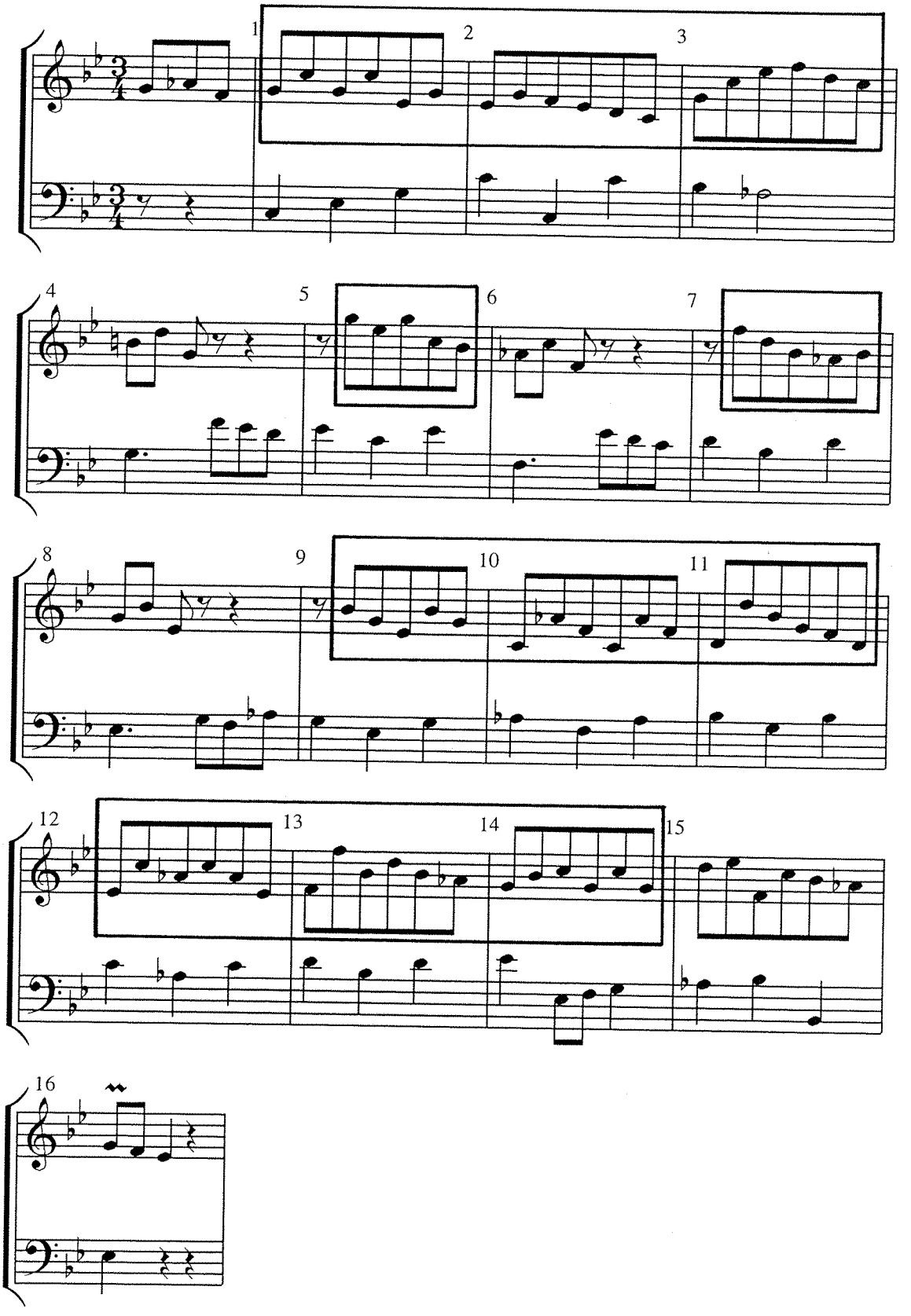
Bach4



Bach5



Bach 6



GLOSSARY

Algorithm The specification of a sequence of unambiguously defined steps which lead to the accomplishment of a task. The specification normally lends itself to the encoding of the algorithm in a computer program.

Fourier Transform The mathematical technique for converting the time-domain representation of a waveform or numerical sequence into its frequency-domain representation.

Function A formula, or group of formulas, which expresses how one quantity depends on the values of one or more other quantities.

Input Parameter One of the values fed into a function in order to obtain the output quantity.

Markov Chain A process in which the probability of occurrence of an event is conditional on the occurrence of one or more past events.

MIDI (Musical Instrument Digital Interface) A communication standard, developed and adopted by manufacturers of electronic musical instruments, which allows instruments and computers to be connected together so that they may share control information.

Object-oriented Programming A method of computer programming in which a program is defined in terms of objects. Objects are in turn defined in terms of the data they contain and the actions that may be performed on them. Objects interact with each other during the operation of the program.

Part The lowest-level building block of a composition created using the *Markov* program. It most commonly consists of a linear, monophonic sequence of notes but may also be used to send MIDI control data, volume or pitch bend changes for example, to the same MIDI channel as another Part, so as to affect the note sequence being produced by that Part. Parts are combined to form Sections.

Perturbation Analysis The systematic study of the effect on the behaviour of a mathematical process of changes to the values of its input parameters.

Petri Net The representation of a Markov process by means of a graph, and an associated table, achieving significant reductions in the amount of data needed to define the transition matrix.

Probability Distribution A formula, or set of values, which expresses the likelihood of occurrence of a value of a random variable. In the case of a continuous random variable, this is the likelihood that it will lie within a particular range of values.

Probability Lookup Table A table of values which expresses the relative frequencies of occurrence of a two or more discrete random events.

Random Characteristic of a process which may produce different results from an identical set of controlling conditions.

Random Variable A discrete or continuous variable quantity which assumes a value as a result of a random process.

Section The top-level building block of a composition created using the *Markov* program. A complete composition consists of a sequence of one or more non-overlapping Sections. Sections in turn consist of one or more simultaneous or overlapping Parts.

Serialism A system of composition based on the transformation of fixed orderings of all twelve notes of the chromatic scale.

Source Code Listing The line by line list of instructions, structured according to the rules of the computer language being used, which form a computer program.

Stochastic Process A system of time-varying random variable quantities.

Transition Matrix A square array of values which define the conditional probabilities of a Markov chain.

Transition Probability An individual probability value, within a Markov transition matrix, specifying the probability of an event occurring conditional on a previous event.

BIBLIOGRAPHY

Alley, Peter and Strange, Carolyn 1994 *ResEdit Complete*, Reading, MA: Addison Wesley

Altech Systems 1990 *MIDI Command Library Programmer's Guide Version 3*, Shreveport, LA: Altech Systems

Ames, Charles 1989 "The Markov Process as a Compositional Model: A Survey and Tutorial", *Leonardo* 22(2): 175-187

Assayag, Gérard et al 1999 "Computer-Assisted Composition at IRCAM: From PatchWork to OpenMusic", *Computer Music Journal* 23(3): 59-72

Bach, Johann Sebastian 1980 *Die sechs Französischen Suiten*, Neue Ausgabe Sämtlicher Werke, Serie V: Klavier- und Lautenwerke, Band 8, Kassel: Bärenreiter

Bennett, Deborah J 1998 *Randomness*, Cambridge, MA: Harvard University Press

Bidlack, Rick 1992 "Chaotic Systems as Simple (but Complex) Compositional Algorithms", *Computer Music Journal* 16(3): 33-47

Bolognesi, Tommaso 1983 "Automatic Composition: Experiments with Self-Similar Music", *Computer Music Journal* 7(1): 25-36

Burton, Anthony R and Vladimirova, Tanya 1999 "Generation of Musical Sequences with Genetic Techniques", *Computer Music Journal* 23(4): 59-73

Chadabe, Joel 1977 "Some Reflections on the Nature of the Landscape within which Computer Music Systems are Defined", *Computer Music Journal* 1(3): 5-11

Chadabe, Joel and Meyers, Roger 1978 "An Introduction to the Play Program", *Computer Music Journal* 2(1): 12-18

Chadabe, Joel 1983 "Interactive Composing: An Overview", *Computer Music Journal* 8(1): 22-27

Cope, David 1987 "An Expert System for Computer-assisted Composition", *Computer Music Journal* 11(4): 30-39

Cope, David 1991 *Computers and Musical Style*, Madison, WI: A-R Editions Inc.

De Furia, Steve and Scacciaferro, Joe 1988 *MIDI Programming for the Macintosh*, Redwood City, CA: M&T Books

Dodge, Charles and Jerse, Thomas A 1985 *Computer Music: Synthesis, Composition and Performance*, New York: Schirmer Books

Dodge, Charles 1988 "Profile: A Musical Fractal", *Computer Music Journal* 12(3)

Feller, William 1964 *An Introduction to Probability Theory and Its Applications* (2 vols.), New York: John Wiley

Freedman, David 1983 *Markov Chains*, New York: Springer-Verlag

Gogins, Michael 1991 "Iterated Functions Systems Music", *Computer Music Journal* 15(1): 40-48

Greenhough, Michael 1984 "A Real-Time, Stochastic Melody Generating System", *Proceedings of the Institute of Acoustics* 6(1) : 47-53

Hiller, Lejaren A and Baker, Robert 1964 "Computer Cantata: A Study in Compositional Method", *Perspectives of New Music* 3: 62-89

Hiller, Lejaren A and Isaacson, Leonard M 1959 *Experimental Music*, New York: McGraw Hill

International MIDI Association 1988 *MIDI Musical Instrument Digital Interface Specification 1.0*, Los Angeles, CA: International MIDI Association

Jacob, Bruce L 1996 "Algorithmic Composition as a Model of Creativity", *Organised Sound* 1(3): 157-165

Jaffe, David and Boynton, Lee 1989 "An Overview of the Sound and Music Kits for the NeXT Computer", *Computer Music Journal* 13(2): 48-55

Jones, Kevin 1981 "Compositional Applications of Stochastic Processes", *Computer Music Journal* 5(2): 45-61

Jones, Kevin 1984 *Exploring Music with the BBC Micro and Electron*, London: Pitman

Jones, Kevin 1989 "Generative Models in Computer-assisted Musical Composition", *Contemporary Music Review* 3: 177-196

Koenig, Gottfried M 1970a "Project 1", *Electronic Music Reports* 2: 32-44

Koenig, Gottfried M 1970b "Project 2", *Electronic Music Reports* 3: 1-15

Laske, Otto 1990 "Letters: Connectionist Composition", *Computer Music Journal* 14(2):11-12

Lohner, Henning 1986 "The UPIC System: A User's Report", *Computer Music Journal* 10(4):42-55

Lorrain, Denis 1980 "A Panoply of Stochastic Cannons", *Computer Music Journal* 4(1): 53-81

Loy, D Gareth 1985 "Musicians Make a Standard: The MIDI Phenomenon", *Computer Music Journal* 9(4):13

Loy, D Gareth 1990 "Letters: Connectionist Composition", *Computer Music Journal* 14(2):13

Lyon, Douglas 1995 "Using Stochastic Petri Nets for Real-Time Nth-order Stochastic Composition", *Computer Music Journal* 19(4): 13-22

Mandelbrot, Benoit B 1977 *The Fractal Geometry of Nature*, San Francisco: W.H. Freeman

McAlpine, Kenneth, Miranda, Eduardo and Hoggar, Stuart 1999 "Making Music with Algorithms: A Case Study System", *Computer Music Journal* 23(2): 19-30

McClellan, Jim 1996 "This is the future", *The Observer "Preview" Magazine*, 12 May 1996: 46

Miranda, Eduardo 1993 "Cellular Automata Music: An Interdisciplinary Project", *Interface* 22: 3-21

Miranda, Eduardo 1994 "Music Composition using Cellular Automata", *Languages of Design* 2: 105-117

Moore, F Richard 1990 *Elements of Computer Music,* Englewood Cliffs, NJ: Prentice-Hall

Myhill, John 1979 "Controlled Indeterminacy: A First Step towards a Semi-Stochastic Music Language", *Computer Music Journal* 3(3): 12-15

Naylor, T H, Balintfy, J L, Burdick, D S and Chu, K 1966 *Computer Simulation Techniques,* New York: John Wiley

Pope, Stephen Travis 1996 "Object-oriented Music Representation", *Organised Sound* 1(1): 55-68

Pressing, Jeff 1988 "Nonlinear Maps as Generators of Musical Design", *Computer Music Journal* 12(2): 35-45

Ralley, David 1995 "Genetic Algorithms as a Tool for Melodic Development", in *Proceedings of the 1995 International Computer Music Conference,* San Francisco: International Computer Music Association: 501-502

Risset, Jean-Claude 1990 "From Piano to Computer to Piano", in *Proceedings of the 1990 International Computer Music Conference,* Glasgow: International Computer Music Association: 15-19

Roads, Curtis 1979 "Grammars as Representations for Music", *Computer Music Journal* 3(1): 48-55

Rothstein, J 1990 "Twelve-Tone Systems Sound Globs Algorithmic Composition Software", *Computer Music Journal* 14(2): 83-85

Rowe, Robert 1993 *Interactive Music Systems: Machine Listening and Composing,* Cambridge, Mass: MIT Press

Scholz, Carter 1989 "Sound Globs Algorithmic Composition for the IBM-PC", *Keyboard* June 1989:135-137

Symantec Corporation 1990 *THINK Pascal User Manual,* Cupertino, CA: Symantec Corporation

Todd, Peter M 1989 "A Connectionist Approach to Algorithmic Composition", *Computer Music Journal* 13(4): 27-43

Todd, Peter M 1990 "Letters: Connectionist Composition", *Computer Music Journal* 14(2):12-13

Voss, R F and Clarke, J 1978 "1/f Noise in Music: Music from 1/f Noise", *Journal of the Acoustical Society of America* 63(1): 258-263

Xenakis, Iannis 1960 "Elements of Stochastic Music", *Gravesaner Blatter* 18: 84-105

Xenakis, Iannis 1992 *Formalised Music*, Revised Edition, Stuyvesant, NY: Pendragon Press

Xenakis, Iannis 1996 "Determinacy and Indeterminacy", *Organised Sound* 1(3): 143-155

Zicarelli, David 1987 "M and Jam Factory", *Computer Music Journal* 11(4): 13-29