troller) with the position of up to three of the six axes of control. All these function mappings are created through graphical function editors. The fourth pane edits the sequence of notes that are played when the pad is hit. The sequence is expressed as a normal MusicKit text scorefile. The scorefile format has been enhanced with additional tags that represent all programmable parameters in a pad. It is then possible to externally generate a textual representation of a pad and then load it into PadMaster (for example, a set of pads for a performance might be algorithmically generated and then loaded into the program).

4. PadMaster in performance

PadMaster has been used to compose and perform "Espresso Machine", a piece for PadMaster and Radio Drum, two TG77's and processed electronic cello (Chris Chafe playing his celletto). The piece is an environment for improvisation in which the PadMaster and celletto performers exchange ideas and play with predetermined materials. The piece is composed in three PadMaster Scenes, each with several groups of related materials that are triggered during the performance. One baton is reserved for triggering pads and the other for continuous three dimensional control of the currently performing pads.

The performance of this piece on several occasions has raised several issues. The simultaneous mapping for several pads of baton movement to MIDI continuous controllers is one of the most interesting performance capabilities of the program, but also raises the possibility of serious MIDI bandwidth clogging. The current version of PadMaster dynamically adapts to the changing conditions and adjusts the position sampling frequency to try to reduce the bandwidth used when several pads are playing simultaneously. More work needs to be done in measuring MIDI usage in a more precise way to avoid sending too much information, but at the same time to avoid control lag if the sampling frequency fall to a very low value. There is also a measurable gap in playback when scenes are changed, during which MIDI activity is not updated as the MIDI and graphical routines share the same execution thread.

5. Future developments

PadMaster is currently undergoing a complete rewrite to implement new and improved functionality.

Pads will be resizable so that each scene can have different number and size of pads if necessary. We have found that sometimes it would be desirable to concentrate important or critical performance functions in a few big pads. Resizable pads would also allow for linking performance behavior to the place where the pad is hit, something that is not possible in the current version. Another enhancement to pads will be inheritance, so that multiple pads with related behavior (something we have found common and very useful) could be grouped together, with the common functionality being editable as a group.

The action types of performance pads will be enhanced to allow for inclusion of algorithms and soundfile playback. The algorithms will be able to use a well defined API to enable linking of baton movement to algorithm parameters.

The NextStep operating system includes a remarkably easy to use system to communicate with remote objects (objects that live in other computer[s] but are directly linked to the execution of a local program). That opens the possibility of using remote computers connected through Ethernet as servers for MIDI or soundfile playback. There are also several scheduled refinements for performance such as a completely separate thread for all MIDI interaction so that graphics may lag behind the performance but there will be no delays when switching between scenes.

Another important enhancement will be defining a way to use different MIDI controllers in addition or instead of the Radio Drum (percussion controllers, normal keyboards, MIDI pedals, etc).

References:

[1] Max Mathews, *The Stanford Radio Drum, 1990*

[2] David Jaffe, Julius Smith and others, *The MusicKit "http://ccrma-www/CCRMA/Software/MusicKit/MusicKit.html"*

[3] Carlos Cerana (composer) / Adrian Rodriguez (programmer), *MiniMax, a piece for Radio Drum*

INTERACTIVE COMPOSITION USING MARKOV CHAIN AND BOUNDARY FUNCTIONS

JÔNATAS MANZOLLI * & ADOLFO MAIA JR. **

*Interdisciplinary Nucleus for Sound Studies (NICS)**

*Applied Mathematics Department (IMEEC)***

University of Campinas (UNICAMP)

13081-970 Campinas SP- BRAZIL

Jonatas@dsif.fee.unicamp.br

ABSTRACT

The research presented here reviews the use of stochastic processes on composition; Markov chains were studied as a tool for developing musical structures in the 70s. This paper presents a model based on Probability Vectors and Boundary Functions which are used on iterative processes to generate further sequences of vectors. The resultant numerical structures are mapped to different classes of musical parameters. The text presents this new approach - the musical ideas behind the method as well as the mathematical model. Finally, it describes a graphic-based compositional system for a MS-Windows Environment.

1. INTRODUCTION

One of the composer's main tasks is to make choice and it would be possible to find several moments within a compositional process in which his/her choices are based on chance. Adding to this the Boulez's concept (1986) that there is inside of each composer a *kernel of darkness* and the Cage's concept that *pure chance is ultimately irrational, by which I mean there can be no closed system of explanation that includes it, save for the Universe itself* (Cage, 1961), we have joined ideas which give us a good flavour and an insight for presenting an application of stochastic process to musical composition..

There are several historical references of composers using random methods in their compositions. Even musicians such as Mozart, Haydn and C.P.E. Bach worked with compositional techniques based on chance. As described by Loy (1988), this method was called Mozart's Dice Game (Würfelspiel); it was a music game used for composing minuets and other incidental works. If the strict musical aesthetic of the Classical Era provided a framework for such approach, it would not perhaps be so surprising to find Cage in the twentieth century using chance as a compositional tool. An interesting connection between the world of computer experimental music and the world of chances was established in the first Computer Music piece by Cage-Hiller called HPSCHD (1967-69) as described by Austin (1992). The genesis the compositional model used in HPSCHD was the work of Hiller & Isaacson (1959); their pioneering research was the seed for an model called *rules-driven-by-noise*. The development of this concept subsequently grew in many different directions and many different decision-techniques were related to stochastic process and Markov chains, as reported by Hiller & Isaacson (1959), Xenakis (1971) and Jones (1981).

The first attempt to integrate a graphic interface and Markov Chains in a computer compositional system was the Jam Factory program. It used four generative structures named *players* which were controlled by *Transition Tables*. This system was designed by Zicarelli (1987) and as he said: *an essential part of the Jam Factory algorithm is the probabilistic decision made on every note as to what transition table to use.* The aim of this program was to use a n-th order transition table to affect the degree of variations of an original material or seed material.

From this point, it is possible to present our research plan. We use a Markov process to generate sequences of *Probability Vectors* (or transition tables as described by Zicarelli, 1987) and these vectors, through numerical iterations, determine the trajectories of the musical material. We also add to this model a set of *Boundary Functions* in a way that the iterative process produces a sequence of probability vectors which converges autonomously to stable states. Finally, the computer graphic interface is a tool for controlling the initial parameters, to listen to the convergence process and to choose the best musical result.

The text below is the first report of our research and as such we still have a lot of questions and proposals. Our computer implementation is very simple and we intend to gradually move on the direction of complex compositional systems. We presented a brief review of the mathematical ideas related to stochastic processes. After we introduce a definition of the Boundary Functions and it follows a discussion of the computer implementation in a MS-Windows Environment.

2. MARKOV CHAIN AS A COMPOSITIONAL MACHINE

In this section we introduce the mathematical concepts underlying our approach. A definition of a Stochastic Process implies in two basic concepts*: State and Probability Transition Between States.* It is also a useful mathematical tool to describe these states with parameters so named as *State Variable* and the overall set of states is entitled *State Space*. In music, for example, we have an immense freedom of choosing this state space: *the 12 pitches of the chromatic scale, a set of sounds generated by a computer sound card, a set of sound from nature, a set of instruments and so on.* These state variables determine completely each particular state.

Now, suppose that we can go from a state to another one randomly. In this case we can ask this question: *which is the parameter to do such a transition between states?* This is named as *Probability Between States.* In this way, to run a stochastic process we must assign a transition probability between all elements of the state space. We define this as a *Distribution Probability* for all possible transitions. Any sequence of states can be constructed applying the distribution probability to control the transition process. These sequences are named *Trajectories*. For example, if a musician uses a Gregorian Mode as state space the resultant trajectory is a modal melody or if he/she defines the state space as a chromatic scale the trajectory will be a chromatic sequence.

Mathematically, we can write the following definition for the *State Space* with m elements:

$$\mathbf{S} = \{s_1, s_2, s_3 \ldots . s_m\} \Leftrightarrow \mathbf{S} = \{s_i\}_{i=1\ldots m}$$

and a sequence such as

$$\{s_1{}^1, s_1{}^2, s_2{}^3 \ldots s_i{}^N\} \Leftrightarrow \{s_i^j\} \textit{ with } 1 \leq i \leq m,\ j=1\ldots N \textit{ and } s_i^j \in \mathbf{S}$$

is a *Trajectory*; where repetition of a state is, of course, permitted and N is an arbitrary number which gives the length of the trajectory.

A very useful stochastic process is the so called *Markov Process*. This is a particular case in which the local probability transition depends only on the previous state. In mathematical terms:

$$P(S_{ij}) = (S_i \rightarrow S_j).\ (S_j)$$

Going down to the earth, we specialise our model to elucidate this idea with a simple musical example. Let us take a particular case whose space states are three pitches. We can denote them as $\{\mathbf{A_1}, \mathbf{A_2}, \mathbf{A_3}\}$. It is possible to define the probability transitions as

$$P_{ij} = P(A_i \rightarrow A_j) \text{ with } i,j = 1,2,3$$

and then we can write a set of all these probabilities as a *3x3 Matrix* $\mathrm{P} = [p_{ij}]_{\text{with } i,j=1\ldots3}$. The fundamental property of this transition probability matrix is that the sum each row must be equal to one and using this matrix we can write a multiple trajectory for all elements of the state space.

Joining the ideas above the *Markov Chain* with n-steps can be written as

$$v^{N+1} = \mathrm{P}.v^N \quad i=0,1,2\ldots.N$$

where V^0 is a given initial probability vector (distribution or transition table). In other words the Markov chain is nothing more than a sequence of probability vectors $\{\mathbf{v}^0, \mathbf{v}^1, \mathbf{v}^2 \ldots\ldots \mathbf{v}^N\}$ generated by an iterative process of multiplication of the initial vector $\mathbf{V}^0$ by the matrix P.

To make contact with musical reality, we must construct a rule that transforms a numerical trajectory in a sequence of pitches. It is important to stress that we are using a set of three pitches as an example, but the sequence can be constructed using a set of dynamics, rhythmic figures, sounds, instruments, numerical samples taken from a waveform or any other set of musical material. Since the state space is finite (there are only 03 elements in this example) we can arrange it like a line vector which we name *Vector State*: $\mathbf{A} = [A_1\ A_2\ A_3]$.

In this way all trajectories of elements can be collected in the sequence of state vector:

$$\mathbf{A}^0 = [A_1{}^0\, A_2{}^0\, A_3{}^0] \Rightarrow \mathbf{A}^1 = [A_1{}^1\, A_2{}^1\, A_3{}^1] \Rightarrow \ldots.\ \mathbf{A}^N = [A_1{}^N\, A_2{}^N\, A_3{}^N]$$

This is achieved with some functions we have named *Boundary Functions* (BF). To construct them, let us define a fixed vector

$$\mathbf{K} = [k_1\ k_2\ k_3] \ \textit{satisfying } 0 \leq k_i \leq 1.$$

The boundary functions (BF) are defined as follows:

A. *Cyclical Boundary Function*

Take a probability vector $[v_1{}^N, v_2{}^N, v_3{}^N]$ of the N-th step of a Markov Process and compare its entries with the fixed vector $\mathbf{K}$. Now we impose the boundary condition:

$$\text{if } \ v_i{}^N \leq k_i \quad \Rightarrow \quad A_i{}^{N+1} = A_i{}^N$$

$$\textit{otherwise} \quad \Rightarrow A_i{}^{N+1} = A_{i+1}{}^N$$

In this case, the resultant musical pattern is a single melodic line in which the pitches follow a cyclical order within the three-pitches state space. Of course this is nothing more than one of the infinite possibilities of BF for Markov chains applied to music. Each one of them will generate a different musical behaviour. Below we give two other BFs.

B.Rhythmic Boundary Function

We can repeat the above Markov Process independently, p times say. Putting these p lines in parallel we have a kind of *multi-phonic structure* and extending the three-pitches space state to include *silence*, *e.g.* $[A_1, A_2, A_3, \textit{silence}]$. We can write the following definition:

$$\text{if } v_i^N \le k_i \Rightarrow A_i^{N+1} = A_i^N$$
$$\textit{otherwise} \Rightarrow \textit{silence}$$

This BF, by its own definition, generates a multi-layered sequence of pitches with an implicit rhythmic structure.

C. Parallel Boundary Functions

We can run an arbitrary number of independent Markov process simultaneously, in a way that each one controls a different musical parameter. For example we can have 03 stochastic processes to control pitches, dynamics and rhythmic figures. In this way the overall result is a complex sound pattern which can be understood as a kind of composition. Finally, we would like to stress, once more, that there is an infinite number of these B.F. Their definitions depends only of musical taste and the composer's imagination.

3. COMPOSING USING A GRAPHIC INTERFACE

Our intention is to provide an environment for sound exploration giving to a musician the essential mutability to control his/her creative decisions. As presented above, we use a graphic interface to control the probabilistic decision such as the Jam Factory program (Zicarelli 1987), but we change the probability vectors itself at each step of the process and we imposed Boundary Functions to drive the numerical sequences to stable states or attractors. Thus the compositional idea behind our model is to explore a system in which the musical material is self-organised through an iterative process. Instead of using probability vectors or n-th order tables as the decision criteria, the vector's' numerical behaviour is the generative process.

We designed a graphic interface to control the Markov process and the musical mapping. We decided to implement the *Parallel Boundary Functions Model* presented above with 03 stochastic processes to control pitches represented by MIDI Note Number table, dynamics represented by MIDI Velocity Number table and rhythmic figures input from a MIDI Keyboard and represented by floating points. The system was developed as a three part real time process. The first part we called *Mathematical Section* and it is the controller of the Markov process i.e. the stochastic Matrix and the probability vector. The stochastic matrix has *03 Diagonal Blocks* such that each block controls independently, or parallely, the three musical parameters above cited. Correspondingly, the probability vector is also partitioned in 03 parts on which the 03 blocks acts. This structure of the Mathematical Section is elucidated below in *figure 01*.

The *Musical Section* is a set of numbers with contains the musical State Space represented by MIDI parameters and rhythmic figures i.e. [1..127] for note, [1..127] for velocity, floating point for rhythm. There is also rhythmic pulse, input by the musician, to control the speed of the performance process. These musical parameters are stored in a numerical vector which is also input by the composer using a keyboard or any other MIDI Controller.

The third section is the *Visual Interface* which was developed for a MS-Windows environment. To design this interface, we used the idea of sonic process controlled by *parametric interactive cells*. A cell is defined as a set of controller parameters which is modified in real time. The program runs 12 Markov processes in parallel each of them is a parametric cell. A cell contains a stochastic Matrix, a probability vector and a representation of the musical state space to control the compositional process. This interactive process consists of two steps: a) *the computer initialises a cell using random numbers* and b) *these parameters are changed by the composer using a MIDI controller device*.

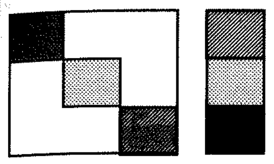


figure 1.0

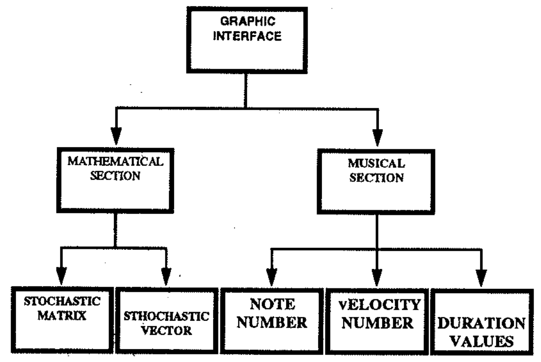
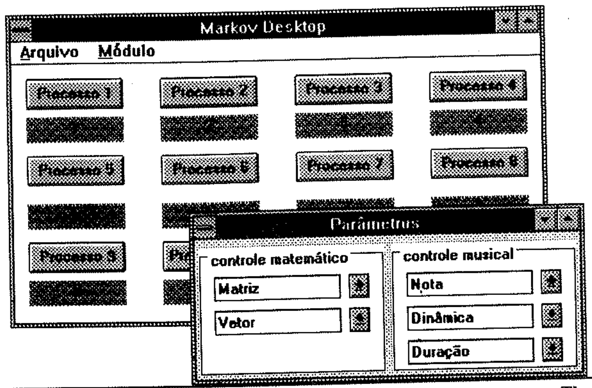


figure 2.0

Figure 1.0 the structure of the Probability Matrix and Probability Vector. Figure 2.0 is a diagram of the compositional system.

The computer controls the compositional process in real-time. The iteration of the Markov process runs at same time the composer listens to the musical material generated by the system. Thus each computer's iteration encapsulates multiplication of the stochastic matrices by the probability vectors and the mapping of the probabilities numbers to the musical State Space. The process is a tool for generating a convergence process and changing, controlling and listening to the musical result. In the *figure 02*, we present a diagram of the compositional system and the graphic interface is presented in *figure 03*.



This is the graphic interface of the for a MS-Windows environment. The top window controls the parametric cells and the right window is used to input control parameters.

4. CONCLUSION

In this work we have emphasised the dynamic aspect of the stochastic processes (in particular of the Markov processes) in contrast to the static concept of probability distribution as worked by several authors such as Xenakis (1970), Jones (1981), etc. We have investigated the applications of our method to a macro-structural approach based on the MIDI event representation. From this point in our research, we are going to construct a micro-structural model in which the Markov process will control waveforms or sound cells such as described by Manzolli (1994). Finally, the use of Stochastic Process in composition can provide many ways of helping the composer's choices. If under Boulez (1986) point of view there is a *kernel of darkness* inside each composer and for Cage (1961) *pure chance is ultimately irrational*, we are still searching for a good representation of this complex and immense world that is the compositional systems created by the composer's own imagination.

REFERENCES

Austin, L. 1992. "An Interview with John Cage and Lejaren Hiller". Computer Music Journal 16(3):15-29.
Boulez, P. 1986. Orientations. Translated by M. Cooper from "Points de Rèpere", 1981. London: Butler & Tanner Ltd, pg 60-67.
Cage, J. 1961. Silence, The MIT Press, Cambridge, Massachusetts.
Hiller, L. & L.Isaacson 1959. Experimental Music, New York:McGraw-Hill Book Company Inc.
Jones, K. 1981. "Composition Applications of Stochastic Processes." Computer Music Journal 5(2):45-61.
Loy, G. 1988. "Composing with Computers a Survey of Some Compositional Formalisms and Music Programming Languages." In Current Directions in Computer Music Research, ed. Mathews, M.V. and J.R. Pierce, Cambridge: The MIT Press, Massachussetts, 1989.
Manzolli, J. 1994. "Non-linear Dynamics as Timbral Constructs." Proceedings of the I Brazilian Symposium on Computer Music, Caxambú,MG.
Xenakis, I. 1971. Formalized Music, Indiana University Press, Bloomington.
Zicarelli, D. 1987. "M and Jam Factory." Computer Music Journal 11(4):13-29.

ACKNOWLEDGEMENTS

We would like to thank the Brazilian Research Counsel (CNPq) for financial support, Microsoft Inc. for providing the basic software used for developing the graphic interface, to the head of the NICS, Prof. Raul do Valle, for allowing us to develop this new approach under the NICS umbrella and our student Marciel Rocha who has been pleased into debugging the compositional program.

THE DEVELOPMENT OF A GESTURE INTERFACES' LABORATORY

JÔNATAS MANZOLLI
Interdisciplinary Nucleus for Sound Studies (NICS)
University of Campinas (UNICAMP)
13081-970 Campinas SP- BRAZIL
Jonatas@dsif.fee.unicamp.br

ABSTRACT

This describes the goals of the Laboratório de Interfaces Gestuais (LIGA) at the Interdisciplinary Nucleus for Sound Studies (NICS-UNICAMP). The aim of LIGA is to build MIDI controllers using digital circuits and various transducers, and to create graphic interfaces for interactive compositional purposes. Three prototypes are presented: two new graphic interfaces developed for MS-Windows Environment and a glove interface which uses the movement of the hands and wrists to produce MIDI events.

1. INTRODUCTION

The desire to use gestures to produce music with electronic devices dates from 1919 when an electronic instrument *Theremin* was invented by the Russian scientist Lev Termen. Music was produced when a performer moved his hands in the neighborhood of antennae to control the amplitude and the pitch of two quasi-sinusoidal oscillators (Glinsky, 1992; Boston, 1989).

More recently, the use of computers as musical instruments was referred to by Boulez (1977) in the following terms: *Oscillators, amplifiers, and computers were not invented in order to create music; however, and particularly in the case of the computer, their functions are so easily generalized, so eminently transformable, that there has been a wish to devise different objectives from the direct one: accidental conjunction will create a mutation.*

The development of the MIDI protocol allowed to use the computer as a controller of musical parameters enabling interactions with musician(s) in real time. However, the MIDI digital data stream differs fundamentally from the analogue approach. For example, using a Theremin a musician acts directly in the sound continuum. In opposition, MIDI devices control only a discrete sequence of sound events called MIDI events or messages. Therefore the development of MIDI-based interfaces to operate closer to the sound level is an important Computer Music research issue. The Steim Foundation in Amsterdam, has conducted significant research in this direction and has developed a series of interfaces (Krefeld, 1990). Composers have designed new instruments (Teiltelbaum, 1984; Beck, 1991). A novel approach to musical organization was presented by Orton (1992): a MIDI-based interactive instrument called *MIDIGRID*. Rowe (1993) discusses and illustrates very well the concepts of interaction between musicians and computer, contains a survey on interactive systems and a description of his own *Cypher*.

The seed for the *Laboratório de Interfaces Gestuais* (LIGA) was planted at the Institute of Sonology (1991-92) when I worked in a join project with the Steim Foundation (Manzolli, 1993) developing interactive gloves with Hall effect sensors and mercury switches for the Sensorlab (Cost 1992). I came back to Brazil with those gloves but without the Sensorlab. This new situation left me to create LIGA. A search for solutions to replace the SensorLab and to use other transducers begun. Another goal was to create graphic environment interfaces.