Markov: construção de uma *external* para Puredata

MODALIDADE: COMUNICAÇÃO

Yuri Behr Kimizuka – UDESC/CAPES – Yuribaer@gmail.com
Milton Ulmer – miltonulmer@hotmail.com

Resumo: Esse artigo analisa a oportunidade e as questões práticas e teóricas no desenvolvimento de uma *external*, para Pure Data (PD). Uma *external* é um objeto desenvolvido em outra linguagem de programação. No caso em questão a *external* aqui desenvolvida se destina a realizar operações estocásticas, que podem ser usadas tanto em tempo real quanto para facilitar a geração de dados. Para tanto serão discutidos os fundamentos epistemológicos da criação de um objeto técnico. Os resultados apresentados revelam a utilidade e as dificuldades envolvidas no processo.

Palavras-chave: Pure Data. Estocástica. Objeto tecnológico. *External*.

Markov: building an *external* for Pure Data

Abstract: This article evaluates the practical and theoretical issues in developing an *external* for Pure Data (PD). This *external* intends to perform stochastic operations that can be used either in real time or to facilitate the data generation. In order to accomplish this objective foundations of markovian stochastic and its implications in music will be discussed. The results discloses the benefits and the difficulties involved in the process.

Keywords: Pure Data. Stochastic. Technological object. *External*.

1. Introdução

O Pure Data é um ambiente de programação orientado a fluxo de dados, em tempo real, no qual se pode interagir com a produção e manipulação dos sons. Foi desenvolvido em 1997 por Miller Puckette[1], que já havia anteriormente trabalhado no desenvolvimento da linguagem de programação visual Max. Trata-se de um sistema em tempo real no qual se pode interagir com a produção e manipulação dos sons.

Nesse artigo serão apresentados e problematizados aspectos relacionados à criação de uma *external*, que funciona como um objeto do Pure Data. No ambiente de programação Pure Data não há programas[2], mas *patchs*, "que correspondem, a grosso modo, a caixas de um diagrama de bloco abstrato" e "são conectados em uma rede" (PUCKETTE, 2007, p.17 tradução do autor). Esses *patches* cumprem a função do programa, ou seja, possibilitam ao usuário empreender uma determinada atividade, como por exemplo síntese sonora, e ou processamento de sons. Todo *patch* é construído através da conexões entre os assim denominados objetos, que possuem funções, características e comportamentos específicos. Esses objetos já estão prontos e fazem parte do ambiente de programação, cabendo ao usuário dispor deles como melhor lhe convém.

Entretanto muitos objetos têm sido criadas ao longo do tempo sob a forma de *externals*, e tem por objetivo realizar tarefas as quais contando apenas com os objetos nativos do Pure Data são inviáveis, ou demandam muito esforço. Por exemplo a *external* "*readanysf"*[3], escrita por August Black, possibilita ao usuário utilizar vários formatos de áudio concomitantemente, o que não era possível contando apenas com os objetos nativos.

Desde o início dos nossos trabalhos com o Pure Data sempre houve a necessidade da implementação de procedimentos estocásticos, marcadamente as cadeias de Markov, que é um processo estocástico no qual um dado fenômeno é classificado em estados finitos e discretos. Por estados discretos entende-se que "os eventos num determinado espaço amostral possam ser individualmente distinguidos." (ROY, 2006, p. 334 tradução do autor).

Todavia não havia nenhum objeto nativo voltado para este caso, e empreender uma cadeia de Markov apenas com os objeto pré-existentes é trabalhoso e possui recursos limitados. Por isso houve a necessidade de se criar um objeto dedicado a esse fim.

A estocástica markoviana está presente na música do século XX desde os anos de 1950, introduzida quase ao mesmo tempo – embora com enfoques radicalmente diferentes – por Iannis Xenakis, na França e Lejaren Hiller, nos USA. O uso musical desse processo se tornou cada vez mais presente à medida que os computadores evoluíram. Xenakis, no início do capítulo dedicado a *Música Estocástica Markoviana* diz que "Agora podemos rapidamente generalizar o estudo da composição musical com a ajuda da estocástica." (XENAKIS, 1981, p.60 tradução do autor) De fato hoje isso está cada vez mais presente, já que as cadeias de Markov tem sido usadas tanto para a composição quanto para a análise.

Kerry L Hagan em seu artigo de 2005 *Genetic Analysis of Analogique B* usa o Pure Data para "recriar", como ele mesmo diz, *Analogique B*. No referido trabalho Hagan cria um *patch* destinado a processar os dados que irão resultar na peça de Xenakis. No caso desse *patch* há, evidentemente, uma cadeia de Markov que é implementada servindo-se exclusivamente de objetos já existentes no Pure Data. Embora possível, esta cadeia resulta de múltiplas operações, e pode ser substituída por um único objeto – uma *external*. A diferença fundamental entre o referido trabalho e este, é que Hagan criou um *patch*, enquanto nós através de uma *external*, possibilitamos facilitar a criação dos *patchs* que venha fazer uso das cadeia de Markov.

2. Cadeias de Markov

Cadeia de Markov é um processo estocástico no qual um dado fenômeno é classificado em estados finitos e discretos. Por estados discretos entende-se que "os eventos

num determinado espaço amostral possam ser individualmente distinguidos." (ROY, 2006: p. 334, tradução do autor) Por exemplo, a figura 1 apresenta uma matriz que ilustra as probabilidades de transição de um evento A para um B – os números são décimos de um inteiro, que também podem ser lidos em termos de porcentagem. Nesse caso partindo de estado A probabilidade de mudar para o estado B é de 15%, ao passo que há 85% de probabilidade permanecer no estado A. Por outro lado a partir do estado B a probabilidade de migrar para A é de 60%, e 40% de permanecer em B.

	A	B
A	0.85	0.4
B	0.15	0.6

Figura 1: Matriz de probabilidade de transição.

Tal situação pode também ser representada graficamente, como no exemplo abaixo, em forma de elos de uma corrente, de maneira encadeada, de onde vem o nome cadeia de Markov.

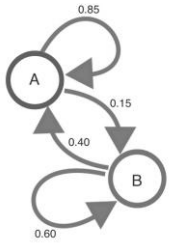


Figura 2: Representação da matriz em forma de diagrama.

A cadeia de Markov possui ainda uma importante propriedade que se apresenta na forma de um conceito de "memória", através do qual o próximo evento passa a depender do anterior. Essa memória, isto é, a maior ou menor probabilidade de transição entre os eventos é abstraída através da noção de antes e depois, visto que neste caso:

> A presença da memória é expressa assim por uma formulação em que um estado antecedente condicione o estado consequente. Para tanto Xenakis faz uso da noção de probabilidade e das cadeias de Markov expressando a estrutura temporal pela formulação da lei de probabilidade. (FERRAZ, 2012: p.67).

No caso específico das cadeias de Markov essa memória pode ainda ser estabelecida em ordens, que é como se classificam essas cadeias. Por exemplo, numa cadeia de Markov de primeira ordem apenas o evento imediatamente interior é considerado. Isso significa que a memória se reporta apenas a esse evento, ao passo que numa cadeia de

segunda ordem são computados os dois eventos anteriores, desse modo a memória vai abarcando cada vez mais dados à medida em que as ordens aumentam. Em consequência disso o processo se torna mais determinístico, ou melhor, tende mais fortemente para um ponto. O que não enfraquece em nada o princípio da cadeia de Markov enquanto processo não determinístico, antes o confirma.

3. Questões teóricas e práticas acerca do desenvolvimento

A primeira questão que surge põe em discussão a maneira de se conceber este objeto *external*. Nesse sentido convém, antes, efetuar uma breve reflexão epistemológica acerca da criação tecnológica. Essa reflexão é oportuna a fim de evitar que a escolha do procedimento markoviano a ser compilada para o sistema seja apenas pragmática.

Simondon dedicou grande parte da sua obra sobre o objeto[4] técnico e suas relações com o humano. Para o filósofo francês o objeto técnico é uma mediação entre o homem e a natureza. (SIMONDON, 2005). É importante ter em mente que o objeto técnico não é um mecanismo separado. Todo objeto técnico decorre de um pensamento, e nele está sempre presente a raiz de sua criação, muito embora a medida que o objeto evolui este se torna cada vez mais independente e individual. A concretização do objeto tecnológico é resultante de um processo que Simondon denomina como individuação, e que é decorrente da solução de um problema.

A partir desse raciocínio, da individuação, é possível deliberar qual caminho a seguir para a elaboração dessa *external*. Duas vias se apresentam, no caso. A primeira é usar uma maneira abstrata de pensamento, ou seja, conceber o processo a partir de seus axiomas e dados. A segunda é uma abordagem mais analógica, no sentido de que opera por analogias, que possa ser compreendida por paradigmas.

A vantagem da concepção abstrata é que por não estar associada a coisa alguma, seus conceitos são livres, e portanto oferece maior liberdade de criação e possibilita características mais flexíveis. Por outro lado esta abstração requer muito mais esforço para empreender a tarefa da criação, no sentido de que a liberdade pode conduzir a uma dispersão de meios. Já na segunda abordagem, por analogia, o existir de um paradigma facilita a conexão imediata entre a ideia da construção e o mundo sensível, mas limita a criação ao conceito no qual esta é análoga.

Como o Pure Data segue um raciocínio que tende a ser mais paradigmático, a nossa opção foi por criar uma *external* baseada num raciocínio semelhante ao do programa

em que será inserida, mas que seja bastante flexível para possibilitar a maior variedade de aplicações possíveis.

4. Objetos e *Externals* no Pure Data

A *external* aqui apresentada, implementa a cadeia de Markov de ordem um, usando uma máquina de estado que tem por base a matriz de transição. Nesta matriz os números das linhas correspondem aos possíveis estados atuais, e os números das coluna aos possíveis estados futuros. Cada posição da matriz contem a probabilidade da transição do estado atual (linha) para o estado futuro (coluna) ocorrer. A cada estímulo, o novo estado é calculado com base no estado atual, e um número aleatório.

Markov possui duas entradas e duas saídas. A primeira entrada recebe o Bang[1], ou seja, o estímulo, e a segunda recebe uma mensagem contendo a matriz de transição. A primeira saída fornece o novo estado calculado a cada estímulo e a segunda saída gera um Bang para o próximo elemento no *patch*, sendo que a tabela de transição pode ser alterada a qualquer instante, e, quantas vezes for necessário. Toda vez que recebe um estímulo, ou seja, uma instrução para mudar de estado, ela consulta a matriz de probabilidade de transição definida pelo usuário e produz um número correspondente ao próximo estado. Esse processo consta de três estágios: primeiro um número aleatório é gerado, segundo este número é comparado com os limites que cada estado está delimitado, e terceiro, de acordo com a faixa este número esteja circunscrito será o estado seguinte.

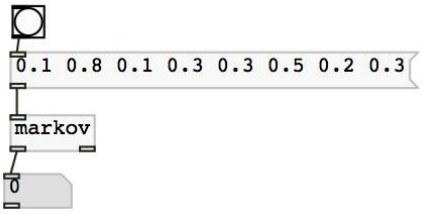


Figura 3: Markov e sua utilização básica.

Dentro do objeto *message* aparece a seguinte sequência de números: 0.1 0.8 0.1 0.3 0.3 0.5 0.2 0.3; essa corresponde aos dados da matriz quando lidos de cima para baixo e da esquerda para a direita. Dessa maneira é possível representar qualquer tipo de matriz de probabilidade de transição.

[1] Bang é o nome de um objeto do Pure Data que tem por objetivo disparar, ou dar início, a uma atividade.

5. Desenvolvimento e linguagem de programação

Desenvolver um novo objeto para o Pure Data é uma tarefa muito diferente de criar um *patch*, uma vez que supõe conhecimentos razoáveis de sistemas e linguagens de programação. Mas não é algo que fuja do âmbito da sonologia, visto que depende apenas do maior ou menor grau de familiaridade que se possui com as ferramentas de programação. Para aqueles que eventualmente desejam programar uma *external* será descrito a seguir, em linhas gerais, o funcionamento básico.

No que diz respeito à programação, uma *external* é uma coleção de rotinas programadas na linguagem C ANSI, que são evocadas pelo Pure Data. A primeira delas markov_setup( ), é usada pelo Pure Data para criar a classe de objetos markov de forma que possa ser usada num *patch*. Também é nesta rotina que são definidas as entradas e o nome das rotinas executadas a cada vez que uma entrada for acionada.

```
void markov_setup(...)
{
          markov_class = class_new(gensym("markov"),
                    (t_newmethod)markov_new,
                    ...);

          class_addbang(markov_class, markov_bang);
          class_addlist(markov_class, markov_list);
}
```

A rotina markov_new( ) é chamada uma vez para cada um dos objetos markov no início da execução de um *patch*, e inicializa os objetos. É nesta rotina que são definidas as saídas.

```
void *markov_new(...)
{
          t_markov *x = (t_markov *)pd_new(markov_class);

          x->f_out = outlet_new(&x->x_obj,&s_float);
          x->b_out = outlet_new(&x->x_obj,&s_bang);
}
```

Para completar a *external* há rotinas que respondem a cada uma das duas entradas.

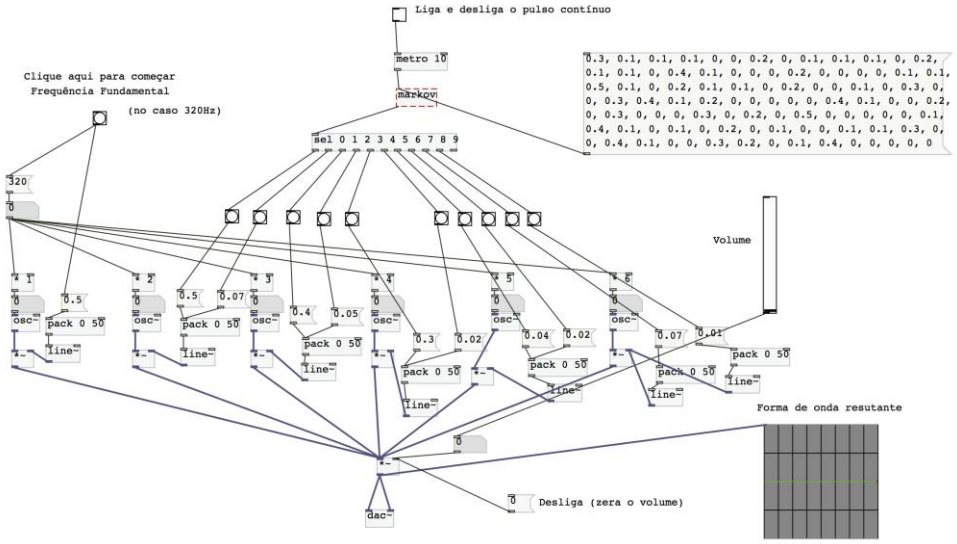
```
void markov_list(...)
{
          //valida e armazena a tabela de transição recebida como
mensagem
}


void markov_bang(...)
```

```
{
        //gera um número aleatório e calcula o novo estado
}
```

6. Possibilidades de utilização

Dentre as inúmeras aplicações possíveis escolhemos demonstrar uma possibilidade de uso na síntese aditiva. Nesse caso as seis primeiras parciais tem a sua intensidade alterada em função de uma cadeia de Markov. Trata-se de um processo de síntese no qual a forma de onda resultante depende da transição entre oito estados.



Exemplo 6: uso da *external markov* na síntese aditiva

7. Considerações Finais

Na realização deste trabalho revelou ser possível criar novos objetos para o Pure Data com resultados satisfatórios do ponto de vista operacional. Após o estudo da estrutura de programação do Pure Data, a external os sistemas LINUX e Windows e Mac Os. O objeto *markov*, após testado, funciona exatamente como um objeto nativo e não foi observada nenhuma falha de operação[5]. O próprio objeto acusa o erro no caso do usuário tentar uma cadeia cuja as linhas não somem 100%.

Implementar a cadeia de Markov dentro do Pure Data possibilita empreender facilmente operações estocásticas necessárias para desenvolver projetos que frequentemente envolvam este tipo de raciocínio, como por exemplo aqueles baseados nas teorias de Xenakis. Por outro lado, também pode ser usado diretamente para a síntese sonora e sistemas de música interativa. Em resumo, muitas são as possibilidades de utilização desse objeto. Certo é, que aqueles que desejarem utilizar este tipo de procedimento nos seus projetos em Pure Data já têm esta ferramenta a disposição.

O êxito na realização do projeto, a adequação do objeto ao conceito e às necessidades a que se destinam inicialmente, demonstram que a implementação da estocástica markoviana no ambiente Pure Data foi plenamente satisfatória. O uso desses processos e os resultados decorrentes, sugerem que novos objetos correlatos devem ainda ser desenvolvidos, o que enseja a compilação de uma biblioteca; que já está em andamento.

Referências:

FERRAZ, Silvio. Três estruturas do tempo em O King de Luciano Berio in *Revista Música.* vol. 13, nº1 agosto de 2012 p.61-95.

LOY, Gareth. *Musimathics, the mathematical fundations of music vol 1*. London: The MIT Press, 2006.

PUCKETTE, Miller. *The Theory and Technique of Electronic Music*. Singapore: World Scientific Press, 2007.

PUCKETTE, Miller. 1988. The Patcher Proceedings, ICMC. San Francisco: *International Computer Music Association*, pp. 420-429. Disponível em: < http://msp.ucsd.edu/publications.html>. Acesso em: 10 fev 2014.

PUCKETTE, Miller. 1996. Pure Data: another integrated computer music environment. Proceedings, Second Intercollege *Computer Music Concerts*, Tachikawa, Japan, pp. 37-41. Disponível em: < http://msp.ucsd.edu/publications.html>. Acesso em: 10 fev 2014.

SIMONDON, Gilbert. *L'Invention dans les techniques.* Cours et Conferences. Édition Établie et Présentée Par Jean-Yves Chateau. Paris: Éditons du Seuil, 2005.

XENAKIS, Iannis. *Musique Formelles*. Paris: Stock, 1981.

Notas

[1] Atualmente professor na área de computação musical da Universidade da California, em San Diego.

[2] Para Puckette o "Pd é um ambiente de implementação, não uma linguagem de especificação." (PUCKETTE, 2007: P.17)

[3] Disponível para download em http://aug.ment.org/readanysf/.

[4] Objeto técnico aqui se refere ao conceito de Simondon e não ao termo "objeto" como são chamados os blocos das linguagens orientadas por fluxo de dados, como no caso do Pure Data.

[5] Foram previstos casos de uso que podem gerar erro, tais como o uso de um numero de estados excessivamente grande, e que estão documentados no arquivo de ajuda.