Tweaking Parameters, Charting Perceptual Spaces

Iván Paz
SEMIMUTICAS / Universitat Politècnica de Catalunya
ivanpaz@cs.upc.edu

Sam Roig Torrubiano
l'ull cec
sam@lullcec.org

ABSTRACT

Live coding builds and conducts the sound by real-time intervention of parametric devices (such as synthesizers). Coding a piece on-the-fly requires to bridge the cognitive gap associated with devices' huge parameter spaces and the possible nonlinear sound variation built-in within them. A possible approach is to have some preselected parameter combinations of which the aural result is known, as a starting point for the performance. However, collecting/memorizing many combinations is time consuming, and using only a few could result in monotony. Therefore, it is convenient to develop models that help to reduce devices' operational complexity with the aim of easing certain musical tasks, such as the automatic creation of variations. Here, a rule-based approach that models the relationships among combinations of parameter values and perceptual categories assigned to them is described. The extracted rules can be used on-the-fly either to simply reproduce the labeled parameter combinations (by calling them by their class as a set of presets), or to obtain new unheard combinations that the model predicts to be consistent with a selected category. The rules are human-readable and describe how parameter combinations relate to perceptual categories. Concrete examples using the system to select material and create the structure of two pieces are presented and discussed.

1. INTRODUCTION

Live coding (Collins et al, 2003; Magnusson 2015; Wang and Cook 2004) intervenes parametric devices to create music and sound (Brown and Sorensen 2009; Rohrhuber and de Campo 2009). By tweaking parameters, the coder explores, categorizes, and selects the appropriate combinations for the different musical contexts. Thus, a piece can be seen as the succession of combinations that creates spatial and temporal structures. Coding a piece on-the-fly requires guiding the sound by changing the parameter settings. This imposes cognitive challenges due to the huge size of devices' parameter spaces, and to the possibility of non-linear sound variation built-in within them. Therefore, it is normal to have some preselected parameter combinations, of which the aural result is known, as a starting point from which to explore during the performance. These combinations chart the parameter space into labeled perceptual categories. However, when only a few combinations are used, the performance can quickly become repetitive and boring for a listener, and again, remembering many combinations imposes practical challenges. This contribution proposes the use of rule learning as a tool to handle this problem.

To clarify what perceptual categories are, consider a system with two variable-frequency oscillators. A possible category would contain all the parameter combinations producing a consonant output (in acoustics a sound with harmonic partials), and another category could be sound with "chaotic" quality. Keep in mind that the perceptual categories are freely defined and assigned by the user. Some issues related with an inconsistent selection of the material are discussed in the conclusions.

It is by means of an aural exploration of the combinations of parameter values that the system's capabilities are learned, and the selection and perceptual categorization of the musical material are performed. This being such a common activity, it seems logical to develop models to aid in reducing the operational complexity of the generators and facilitating certain musical tasks, such as the automatic creation of variations within a part of a piece.

It is worthy to say that there have been excellent examples of methodologies for finding sets of parameters that successfully produce entities with specific perceptual properties. For example, Dahlstedt (2001) and Collins (2002a; 2002b) applied interactive evolution (Dawkins 1986), which uses human evaluation as the fitness function of a genetic algorithm, for system parameter optimization. In Dahlstedt (2001), this technique was applied to sound synthesis and pattern generation tasks, while in Collins (2002a; 2002b), it was used for searching successful sets of arguments controlling algorithmic routines for audio cut procedures.

This work is inspired by these methodologies, and further focuses on creating models of the information that is produced during the aural exploration process (parameter combinations) intended for the following objectives:

1. To provide a structured and interpretable representation of the relationships among parameter values and perceptual categories.

2. To reduce the cognitive and operational complexity of the systems by providing a way to generalize a list of “presets” into a set of rules describing each perceptual category.

3. To extend the compositional capacities by suggesting new unheard (unexplored) parameter combinations that the system predicts to be consistent with a given perceptual category.

An early version of the algorithms described here can be found in Paz et al. (2017). They create rule models of the patterns expressing the relationships among parameter values and categories. Rule models, in contrast with subsymbolic approaches (like neural net classifiers), are human readable and yield interpretable information, which makes them especially attractive for applications in contexts such as live coding, in which the greater the understanding of the system, the better the user possibilities for real-time interaction. Furthermore, it is possible to extend the rule model to unexplored regions of the space expected to be consistent with the selected perceptual categories. The present manuscript describes the current version of the algorithm presented at Paz et al. (2017) providing examples of how the different models have been used. The rest of the manuscript is structured as follows: section 2 introduces the rule extraction algorithms (Rulex and IntRulex), section 3 presents applications in two concrete examples, and finally, section 4 discusses the results and possible further work.

2. RULEX AND INT-RULEX ALGORITHMS

2.1 Rulex Algorithm: Structuring Data through Patterns

The data is captured through an auditory exploration, in which arbitrary perceptual categories are sought and selected. During the exploration, the system parameters are tweaked in turn, until the user decides to store the current combination of parameter values into one of the existing categories (or classes) or into a newly-defined one. An input datum is named a “preset” evoking the way synthesizer configurations are named. Each one is saved as a a list containing the set of parameter values and its associated category.

The only parameter of the Rulex algorithm is the “distance factor” d. It indicates the maximum distance between two combinations of the same category for them to be grouped into a new rule. In this contribution the Hamming distance (number of parameter by parameter differences) was used. In the examples presented here “d” was set equal to 1. Current research is using greater distance factors (and other distance functions).

The Rulex algorithm takes as input a set of “presets”. Then, it searches iteratively for patterns and groups the data among which a pattern is found. Two or more data of the same category exhibit a pattern if they are “located” at a distance factor “d” from each other. The grouped presets are represented by structures called “Strict Rules”. The output of the algorithm is a set of Strict Rules. Next, both concepts are defined.

2.1.1 Preset
A preset is a labeled parameter combination. It is represented in a list in which the first n-1 elements are parameter values and the entry n contains the perceptual category (or class) associated with the parameter combination.
preset = x_1, x_2,. . . , x_{N-1}, x_N

2.1.2 Strict rule
Is a labeled list in which the initial n-1 entries contain sets (whose elements are parameter values), and the entry n contains a perceptual category. It satisfies that all the combinations formed by taking one element from each set, belong to the perceptual category of the rule.
rule = $\{x_1\}$, $\{x_2\}$, . . . ,$\{x_{N-1}\}$, x_N

The rulex pseudocode is shown below. The main functions appear in bold font and are described below.

```
rulex( set of presets P,  set of rules R, distance factor ):
    if R == empty set:
        transform preset_into_rule(first preset) and move it to R
    for preset in P:
        rule1 = transform preset_into_rule(preset)
        for rule2 in R:
            if pattern_found(rule1, rule2):
                create_rule and add newRule to R
        add rule1 to R
    delete redundant rules from R
    return R
```

2.2Functions

1. preset_into_rule
The preset_into_rule function takes a preset and returns a rule in which the first n-1 entries have sets with a single value, corresponding with the respective preset parameter value. For example:
preset = x_1, x_2,. . . , x_{N-1}, x_N
preset_into_rule(preset) = $\{x_1\}$, $\{x_2\}$, . . . ,$\{x_{N-1}\}$, x_N

2. pattern_found
Let set1 and set2 be the sets located at the *jth* entry of rule1 and rule2, where $0 \leq j \leq$ n-1. Note that 0 is the first entry of the rules. Let *difference* count the number of times set1 $\neq$ set2.

The pattern_found function takes two rules as input and checks whether or not they satisfy the following conditions:

class rule1 == class rule2

difference ≤ distance factor = 1

The pattern_found_function returns an array with three entries:

pattern: True, in case both conditions are satisfy. False, otherwise.

unions: A list containing for each *j* set1 ∪ set2

indexes: A list of the *j* indexes for which set1 ≠ set2.

3. create_rule

This function receives the current rule (rule1) being compared and the output of the pattern_found function. As a pattern has been found, a new rule is created. It is formed with the rule1 by overwriting the entries contained at the array of indexes, returned by the pattern_found function, with the unions of the sets of the corresponding entries in rule1 and rule2. The pseudocode is shown below:

```
create_rule(rule1, unions, indexes):
    rule = rule1
    for index in indexes:
        rule[index] = unions[index]
    return rule
```

4. delete_redundant_rules

This function searches in R for those rules that are “contained” in others. rule1 is contained in rule2 if the following conditions are satisfied:

1. class rule1 == class rule2
2. ∀ *j* rule1[j] is subset of rule2[j]

When a rule1 is contained in a rule2, rule1 is eliminated from R.

As a simple example, consider the dataset shown at the top of Table 1. The rules extracted by the Rulex algorithm are shown at the bottom.

Preset	Parameter 1	Parameter 2	category
1	1	2	a
2	1	4	a
3	5	2	a
4	5	4	a
5	2	1	b
6	2	3	b
7	4	1	b
8	4	3	b
$\text{Rule}_{\{1,2,3,4\}}$	{1,5}	{2,4}	a
$\text{Rule}_{\{5,6,7,8\}}$	{2,4}	{1,3}	b

Table 1. Rulex applied to a set of 8 presets. The extracted rules are shown at the bottom of the table. The sub-indexes indicate the instances contained at each rule.

2.3IntRulex: from Points to Intervals

The Rulex algorithm produces a data representation structured by patterns. However, although they are interpretable structures, the Strict Rules represent only points in the space (those formed by taking one element from each set). Therefore, an extension of the coverage of the model for the rules to be able of “guessing” unexplored parameter combinations consistent with their perceptual categories is desirable. This is the function of the IntRulex algorithm. It starts from the rules created by the Rulex and extends its validity from points to intervals in the space. For that, it replaces the sets located at the n-1 parameters of the Strict Rules, by the intervals formed between their minimum and maximum values. By using a heuristic of maximum volume the algorithm then solves the contradictions, i.e., regions assigned by the model to more than one category, that can be created during this process. Next an example is presented.

2.3.1 Interval Rules Example

To provide insight of how the IntRulex works, consider the following set of Strict Rules:

[{2}, {3}, b, 1], [{1, 5}, {2, 4}, a, 1]

As said, the algorithm begins by replacing the sets located at each of the n-1 parameters by the intervals formed between their minimum and maximum values.

[[2], [3], b, 1], [[1, 5], [2, 4], a, 1]

These rules assign to their corresponding category, all combinations formed by taking one value from each interval. In this case a contradiction is created between the valid combinations [2,3,b] and [2,3,a] constructed with the first and second rule respectively. To solve the contradiction, the algorithm proceeds as follows: Note that to solve a contradiction it is sufficient to modify the interval in only one of the parameters. In the example, the contradiction among the rules disappears in any of the following cases:

[[2], [3], b, 1], [[1], [2, 4], a, 1], [[5], [2, 4], a, 1]

[[2], [3], b, 1], [[1, 5], [2], a, 1], [[1, 5], [4], a, 1]

To select which parameter to “break” a criteria of “maximum volume” is taken, i.e., the set of rules covering the largest space is selected (although other heuristics can be implemented). In the example the second set of rules is selected. The resulting rules are called “Interval Rules”. For further references of this process the reader is referred to Paz et al. (2017). Figures 1 and 2 shown the original Interval Rules with the two possible partitions respectively.

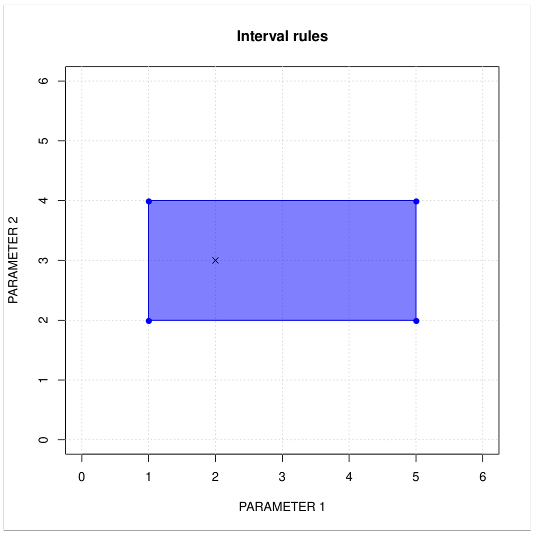

Figure 1. Original Interval Rules [[2], [3], b, 1], [[1, 5], [2, 4], a, 1] created by the IntRulex. It can be seen that a contradiction exists.

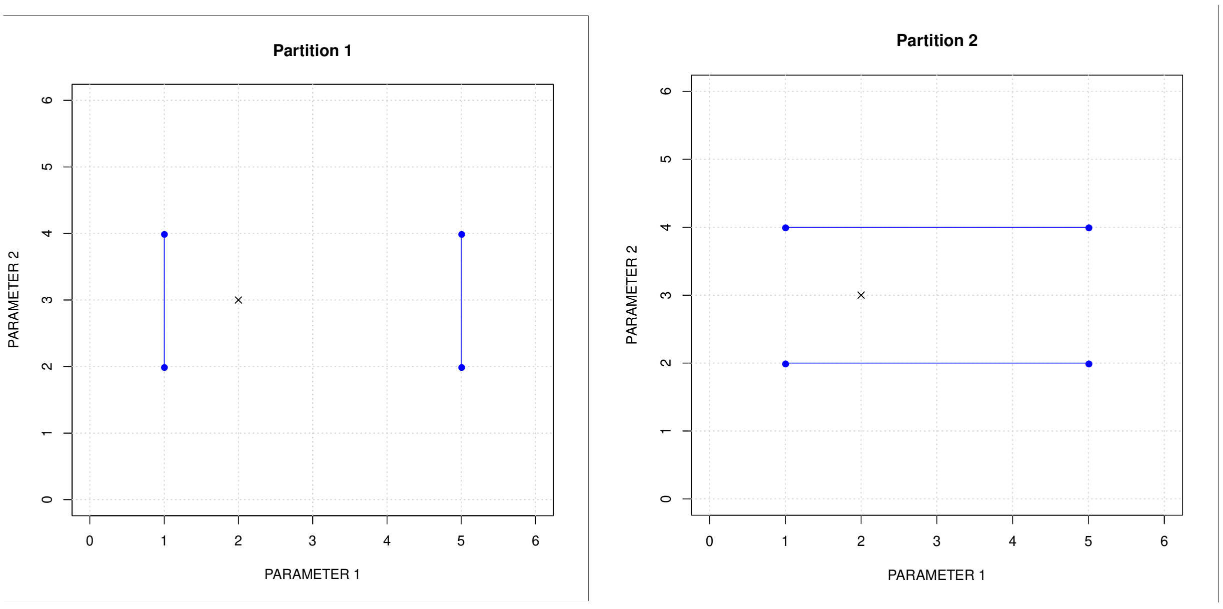

Figure 2. Possible sets of rules that eliminate the contradiction. As a heuristic of maximum volume is used, the second set of rules is selected.

3.EXAMPLES: CREATING STRUCTURE FROM PERCEPTUAL CATEGORIES

In this part two examples of how the Rulex and IntRulex algorithms can be used for the specific tasks of selecting aural material and automating its generation during sections of pieces is presented. The sections are actually defined through the evolution of the perceptual categories, so through perceptual categories selection and labeling, the structural building blocks are also created. The pieces have been lived coded by using the rules to automate the selection of the material for each part, giving the coder the possibility to manipulate in the meanwhile, for example, the signal processing. User tests (for the validation of the rules) conducted with students of the Real Time Interaction Class of the Master's Degree in Sound and Music Computing of the Pompeu Fabra University (winter term of 2016-2017), as well as with computer music composers and audio technology developers can be found in Paz et al. (2017). The examples presented here correspond to the pieces “En Casa” and “Visions of Space” by Iván Paz, which are available at Bohemian drips (2017a; 2017b). These examples were selected since they use Strict and Interval Rules respectively. The Strict Rules are useful for systems in which perceived categories change abruptly when moving away from the classified point. In contrast, when the changes in the perceptual categories are smooth in response to changes in the system parameters, the system is a good candidate for Interval Rules, which generates more variability.

3.1 Example One: En Casa

For “En Casa”, the following SuperCollider (Wilson et al., 2011) node definition (Ndef) was used:

```
Ndef(\EnCasa, {
        arg freq1 = 50, freq2 = 100, amp = 0.2;
        var sig;
        sig = Mix.ar(Saw.ar([freq1, freq1-1, freq2, freq2 + 1], amp));
        sig = sig + Impulse.ar(20, amp/4);
        sig = FreeVerb.ar(sig, 0.33,0.5,0.5,1) * amp;
        sig = Limiter.ar(sig,1);
} );
```

The system parameters are the frequencies (freq1 and freq2 in Hz) and the amplitude (amp). They control the frequencies and amplitudes of the four sawtooth generators. Both frequencies were restricted to the interval [0, 400] and the amplitude took values in [0,1]. To select the data a geometric interface (Paz, 2016) was used. For the selection of material, freq2 was freely tweaked until select 119.563484Hz (for no particular reason but its sound). Then, freq1 was tweaked and all the frequencies in the interval [0, 120.930231] producing "interesting intervals" were searched and labeled with "category 1". The data and the extracted Strict Rule for this category are shown at Table 2 at the top and bottom.

preset	freq1	freq2	amp	category
1	96.744186	119.563484	0.807776	category 1
2	50.232559	119.563484	0.807776	category 1
3	117.209303	119.563484	0.807776	category 1
4	72.558141	119.563484	0.807776	category 1
5	120.930231	119.563484	0.807776	category 1
6	39.069769	119.563484	0.807776	category 1
$\text{rule}_{\{1,2,3,4,5,6\}}$	{96.744186, 50.232559, 117.209303, 72.558141, 120.930231, 39.069769}	{119.563484}	{0.807776}	category 1

Table 2. Rulex applied to a set of 6 presets. The extracted rule is shown at the bottom of the table.

Then, freq1 was tweaked until 178.604650Hz was selected. After that, freq2 was tweaked in the interval [119.563484, 178.604650] and all frequencies producing an interesting interval with freq were selected and categorized as "category 2". The data is shown in Table 3. Note that also 179.561090 was selected.

preset	freq1	freq2	amp	category
7	178.604650	119.563484	0.807776	category 2
8	178.604650	135.675585	0.807776	category 2
9	178.604650	150.674987	0.807776	category 2
10	178.604650	168.215823	0.807776	category 2
11	178.604650	179.561090	0.807776	category 2
12	178.604650	144.728661	0.807776	category 2

Table 3. Presets captured for the second category.

The process was repeated for the four categories that formed the structure of the piece. Note that in Tables 2 and 3 intervals like the unison (freq1=120.930231, freq2=119.563484 and freq1 = 178.604650, freq2 = 179.561090 for categories 1 and 2 respectively), and the perfect fifth (freq1=178.604650, freq2=119.563484 for category 2) are present in the data with small variations consequence of the beats.

3.2 Example Two: Visions of Space

The Ndef used for the second example is shown below. Because the synthesis is somehow messy, it leads to a rather unintuitive perceptual output but we can still make sense of the system and chart its parametric space into meaningful categories by labeling.

```
Ndef(\visions,{
            arg freq, freq1, freq2, numharm, amp;
            var sig;
           sig = Blip.ar(freq, numharm) * amp;
           sig = sig + Blip.ar( [FSinOsc.kr( freq * 2 ),
                   freq + 1, freq/2, freq -1] , numharm * 1.4) !2 * amp;
          sig = sig + Saw.ar([freq1, freq1 - 1, freq1 + 3], amp/9);
          sig = RLPF.ar(sig, SinOsc.kr([0.1, 2], 0, [1700,480], [4000,700,5000])/[20.51,20],
                  SinOsc.ar(0.1,1.5*pi) + 1.05);
          sig = FreeVerb.ar(sig, SinOsc.kr(freq2),0.8, 0.5, 1) !2;
          sig = Limiter.ar(sig,1);
});
```

In this example, three perceptual categories were chosen for the structure of the piece. They were selected to produced an increase in the "impression of chaos" while going from category 1 to 3. The selection was performed by tweaking one parameter at time to favor the formation of rules. The system components are band-limited impulse generators ("Blip"), band-limited sawtooths ("Saw"), resonant low-pass filters ("RLPF"), a reverb effect ("FreeVerb"), a sinusoidal oscillator ("SinOsc"), and an efficient sine wave generator based on a ringing filter ("FSinOsc"). For the components documentation the reader is referred to (SuperCollider Help). The system's parameters are: "numharm" (number of upper harmonics added to the fundamental frequencies of the Blips), "freq", "freq1", and "freq2" (fundamental frequencies of the Blip and Saw) and "amplitude" (amplitude level of the different components). Table 4 shows the extracted Strict and Interval Rules for the first category.

freq	freq1	freq2	numharm	amp	category
Strict Rules					
{242.000009}	{90.636051, 121.821141, 161.278355, 214.226604, 194.188201, 125.672603}	{0}	{50}	{1.7}	chaos 1
{208.930221, 168.511626, 254.860476, 192.395351, 148.302328}	{125.672603}	{0}	{50}	{1.7}	chaos 1
Interval Rules					
[242.000009]	[90.636051, 214.226604]	[0]	[50]	[1.7]	chaos 1
[148.302328, 254.860476]	[125.672603]	[0]	[50]	[1.7]	chaos 1

Table 4. Strict Rules (top) and Interval Rules (bottom) created with the data captured for the perceptual category "chaos 1".

In a mathematical sense, the perceptual categories selected in this example form continuous convex spaces in the dimension of the selected parameters, so Interval Rules can be used as a model of the perceptual spaces with the added advantage of producing more variability within the categories when it is used to generate new material.

4.DISCUSSION AND FURTHER WORK

The presented algorithms provide structured representations of the implicit relationships among parameter settings and specific perceptual categories from labeled data collected during an auditory exploration of a parametric musical system. The produced models can be used to address specific musical tasks. In the examples presented, they are used to to select material and to create musical structure by defining the different parts of two musical pieces. The algorithms were designed to allow different levels of generalization, optionally including unclassified combinations. This approach allows working with architectures in which the changes in the aural perception in response to changes in the parameter values range from smooth to rough. If the perceptual category selected varies a lot in response to small changes in the parameter values, as is the case in the first example, the Strict Rules can be used. If small changes in the selected perceptual category occur due to changes in parameter values, the Interval Rules offer more variability, a more compact model, and the possibility of creating new material without stepping out of the selected categories. The created models can be used as the starting point to more general models. For example, an extension for the IntRulex algorithm that connects the created intervals by using trapezoidal fuzzy membership functions covering all the observed space is presented in Paz et al. (2017).
A possible extension for the exploration process (in which the data is captured) would be to provide facilities for the user to perform a guided exploration of the space.
Needless to say, it is essential for the proper functioning of the system, that the user is consistent in the selection of the combinations placed at each category. Therefore, another possible extension would be to provide the user with an informed mechanism to audit the “outliers” of each perceptual class, in order to review and further refine the inner consistency of each category.

REFERENCES

Bohemian drips. 2017a. Retrieved from https://bohemiandrips.bandcamp.com/track/b2-en-casa

Bohemian drips. 2017b. Retrieved from https://bohemiandrips.bandcamp.com/track/b1-visions-of-space

Brown, A.R., Sorensen, A., 2009. Interacting with Generative Music through Live Coding. Contemp. Music Rev. 28, 17–29.

Collins, N., 2002a. Experiments with a new customisable interactive evolution framework. Organised Sound 7, 267.

Collins, N., 2002b. Interactive evolution of breakbeat cut sequences. Proc. Cybersonics .

Collins, N., McLean, A., Rohrhuber, J., Ward, A., 2003. Live coding in laptop performance. Organised sound 8, 321.

Dahlstedt, P., 2001. Creating and Exploring Huge Parameter Spaces: Interactive Evolution as a Tool for Sound Generation., in: ICMC.

Dawkins, R., 1986. The blind watchmaker: Why the evidence of evolutionreveals a universe without design. WW Norton & Company.

Magnusson, T., 2015. Herding cats: Observing live coding in the wild. Comput. Music J. 38, 8–16.

Paz, I., Nebot, A., Mugica, F., and Romero, E. 2017. “Modeling perceptual categories of parametric musical systems.” Accepted for publication in the Special Issue of Pattern Recognition Letters http://www.sciencedirect.com/science/article/pii/S0167865517302374

Paz, I., 2016. GMuSE: a geometric representation for multi-parameter spaces exploration. Interface Politics 1st International Conference. Publications Gredits. Barcelona p.p. 201-212 ISBN (Ed. Impresa): 978-84-617-5132-7.

SuperCollider Help Retrieved form http://www.gredits.org/wp-content/uploads/2016/11/Publicacions_Gredits_04_V2_web.pdf

Rohrhuber, J., de Campo, A., 2009. Improvising Formalisation: Conversational Programming and Live Coding. New Comput. Paradig. Comput. Music.

Wang, G., Cook, P., 2004. On-the-fly Programming: Using Code As an Expressive Musical Instrument, in: Proc. 2004 Conf. New Interfaces Music. Expr.,pp. 138–143.

Wilson, S., Cottle, D., Collins, N., 2011. The Supercollider Book. MIT Press, Cambridge, MA.