Revista Vórtex | Vortex Music Journal | ISSN 2317–9937 | http://vortex.unespar.edu.br/
D.O.I.: ________________________ [Fonte 8 | Cambria | Espaçamento 1.0]

# Making Complex Music with Simple Algorithms, is it Even Possible?

Risto Holopainen
Oslo, Norway

**Abstract:** Algorithmic composition is often limited to score generation, but may also include sound production. All levels from sound synthesis to the generation of a complete composition can be integrated into one monolithic program. A strict separation of the low level of sound synthesis and higher levels otherwise reserved for algorithmic composition is not necessary, information can flow between all levels. An interesting challenge in this kind of thorough algorithmic composition is to generate as complex music as possible with as little code as possible. The challenge has been accepted, successfully or not, in a series of compositions called *Kolmogorov Variations*. We discuss the techniques used in a few of the pieces as well as the promises and perils of this strict approach to algorithmic composition.

**Keywords:** Algorithmic composition, Computational complexity, Dynamic systems, Acoustic processing

Paper's title translated to English; please adapt to a maximum of 90 characters with space. Recebido em 00/00/2021. Aprovado em 00/00/2021. Disponível online: 00/00/2021. Editor: _____. Creative Commons CC-BY [Fonte 8 | Cambria | Espaçamento 1.0]

These days, the word *algorithm* probably rings with associations to deep learning, big data, and those sophisticated systems purporting to be artificially intelligent, all of which are hyped up and popular topics in technologically advanced music. For some people, the word algorithm may evoke even narrower associations to surveillance capitalism and the curated realities of asocial media.

Composers have practised algorithmic composition as a means to transcend the limits of habit and to explore unknown aspects of music, often by delegating choice either to randomness or to some deterministic procedure (Loy 2006, pp. 293-295). Some have focused on style imitation, including a large body of scholarly work. According to Nierhaus (2010) there is a predominance of publications on style imitation by algorithmic composition, as opposed to what he calls ”genuine composition”, partly because the incentive to publish papers is greater among academic researchers than among composers, and because the success or failure of style imitation can be readily evaluated, whereas genuine or original algorithmic composition lacks comparable objective criteria for evaluation.

> Composers publishing material on their algorithmic methods is rather a rare occurrence and may be explained to a certain extent due to the following consideration: Publishing is in general not part of the profession. It is replaced by a composition, whose creative principles – if formalized – are not willingly communicated for obvious reasons. In addition, there are often basic objections to the use of algorithmic principles, because the musical structure becomes comprehensible and the composition therefore cannot claim ”ingenious creativity” any longer. (Nierhaus 2010, p. 263)

To me it is not at all obvious that most composers would want to keep their creative principles secret, on the contrary there is a long list of treatises and papers that deal with technical aspects of composition. Insofar as the musical structure becomes comprehensible because of the use of algorithms, maybe it happens because the algorithm is not very sophisticated. Composing by algorithms, as I will try to illustrate with a few examples, can be every bit as creative as directly writing notes with a pen on a paper, it just involves a few steps of translation before ideas become sound.

Judging from the amount of conferences and research papers, the currently prevailing approach to algorithmic composition seems to take the route of AI and deep learning, which implies a dependence on massive amounts of data in the form of corpora of existing music to be used as a training set. Training neural nets also takes a substantial amount of computation. Maybe it is because of the bleak Beatles pastisches[1] that showcased early landmarks of AI in the service of musical creativity that this approach only seems capable of derivative work. Although I think more original work can and will emerge, at least for me it is preferable to begin composing from scratch, to invent one's own algorithms and to strive beyond mere replication or remixing of already familiar styles of music.

Oftentimes algorithmic composition is limited to score synthesis. Electroacoustic music makes it feasible to integrate the levels of composition and sound synthesis, or contrarily, not even bothering to separate these levels in the first place. Auditory perception certainly works in different ways at different time-scales. Repetitions at a fast rate tend to be perceived as pitch; repetition at slower rates is perceived as a pulse and at even slower rates as recurring motifs or formal sections. Having handles at separate time-scales is therefore very practical, although the boundaries can be fluid. Some processes are applied at the sample rate, other functions are called at slower rates, or only sporadically, but the temporal levels are not necessarily compartmentalised. Information may flow from the fastest levels to the slowest, as well as in the opposite direction. This observation is particularly relevant in systems-oriented approaches.

Much of my work on algorithmic composition is related to dynamic systems where feedback loops generate the evolution of the piece. I often integrate the low level of sound synthesis with the higher level of notes and phrases (if there are any), or the large scale structure of the piece. In the following, I will discuss some of my most recent work on algorithmic composition, namely the *Kolmogorov Variations*, also known as Eleven Hard Pieces.

## 1. A glimpse into some of the techniques used in the pieces

The eleven hard pieces came to be in a process that can be sharply divided into two phases. Since each piece is the output of a computer program, the first and longest phase consisted of a cycle of

1 For example, a song called Daddy's Car. See http://www.flow-machines.com/

programming, generating output, and modifying the program. Each program generates a unique soundfile for each particular composition. Then followed a shorter sound processing and mixing phase in which the pieces were submitted to acoustic processing and various treatments using an analog eurorack modular synthesizer. Although the pieces ended up sounding quite different after these treatments, the form is exactly as it came out of the programs.

Each piece explores certain ideas that may come across from a scrutiny of the code, although they are hardly revealed just by listening to the music. Some of the ideas have to do with representations of time, sound synthesis techniques, or dynamic systems; in other words, these are not primarily musical ideas as such. Of course considerations of the musical structure and expression also drove the development of the programs. I will briefly describe some of the pieces and a few of the techniques used in them.

*Bâtir et Démolir* (*ZAD*) was realised using one program to generate raw material and save it to a soundfile, which is then processed by a second program. The raw material falls into three categories: single sinusoids at different frequencies, stretches of constant amplitude levels (DC offsets), and noisy signals generated by a chaotic map. This material is generated by a ”building” program, which determines the succession, types, and durations of the segments of each material type. A second ”demolishing” program is used to process this material by combining segments using a set of binary operations on the signal and a delayed copy of it, including linear mixing, saturated mixing, ring modulation, taking the minimum and maximum of the two signals. The title is an oblique reference to the sprawling architecture at places like the occupied territory of Notre Dame des Landes in France, which was later to be demolished (ZAD stands for ”zone à défendre”, a zone to be defended).

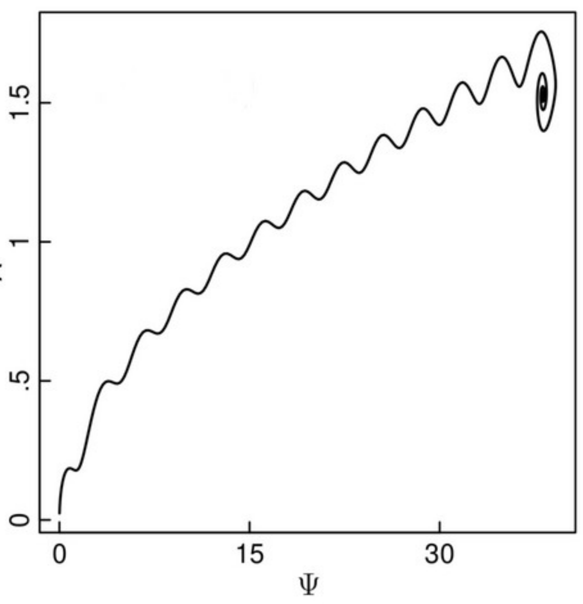
A simple three-dimensional ordinary differential equation is at the core of two pieces, *Megaphone*, and *Auto-detune*. In a somewhat simplified version, the system is given by the equations

$$d\theta_1 / dt = \omega + \delta$$
$$d\theta_2 / dt = \omega - \delta$$
$$d\delta / dt = c|\sin\theta_1 + \sin\theta_2| - \delta,$$

which specifies two phase variables that are mutually detuned by a third variable, and is used by oscillators to generate quasi-periodic signals. I do not know if this particular system has been described in the literature so far, at least it is not included in the rather comprehensive collection of simple chaotic systems by Sprott (2010), even though it seems capable of chaotic behaviour.

Many layers of feedback loops complicate the dynamics, particularly in *Auto-detune*. In contrast, *Megaphone* is relatively simple; the governing equations could be written on a single page. It is an example of a slow-fast system, that is, a system with dynamics on different time-scales, with some degree of interaction between the temporal levels. As it turns out, the dynamics at the slowest time-scale is that of a variable that wanders off in one direction of the state space during the first part of the piece, wiggling slightly on its way, until it reaches the attractor of a stable fixed point and begins spiralling in towards it (as shown in Fig. 1). This process is obvious when listening to the piece as the music slows down and becomes ever more static towards the end. However, it was far from obvious while conceiving the equations for the piece.

FIGURE 1. A plot showing the temporal development of two (out of a total of about 65) variables used in the piece *Megaphone*. The system starts from the initial point (0,0) in the lower left corner, and finally reaches a fixed point.



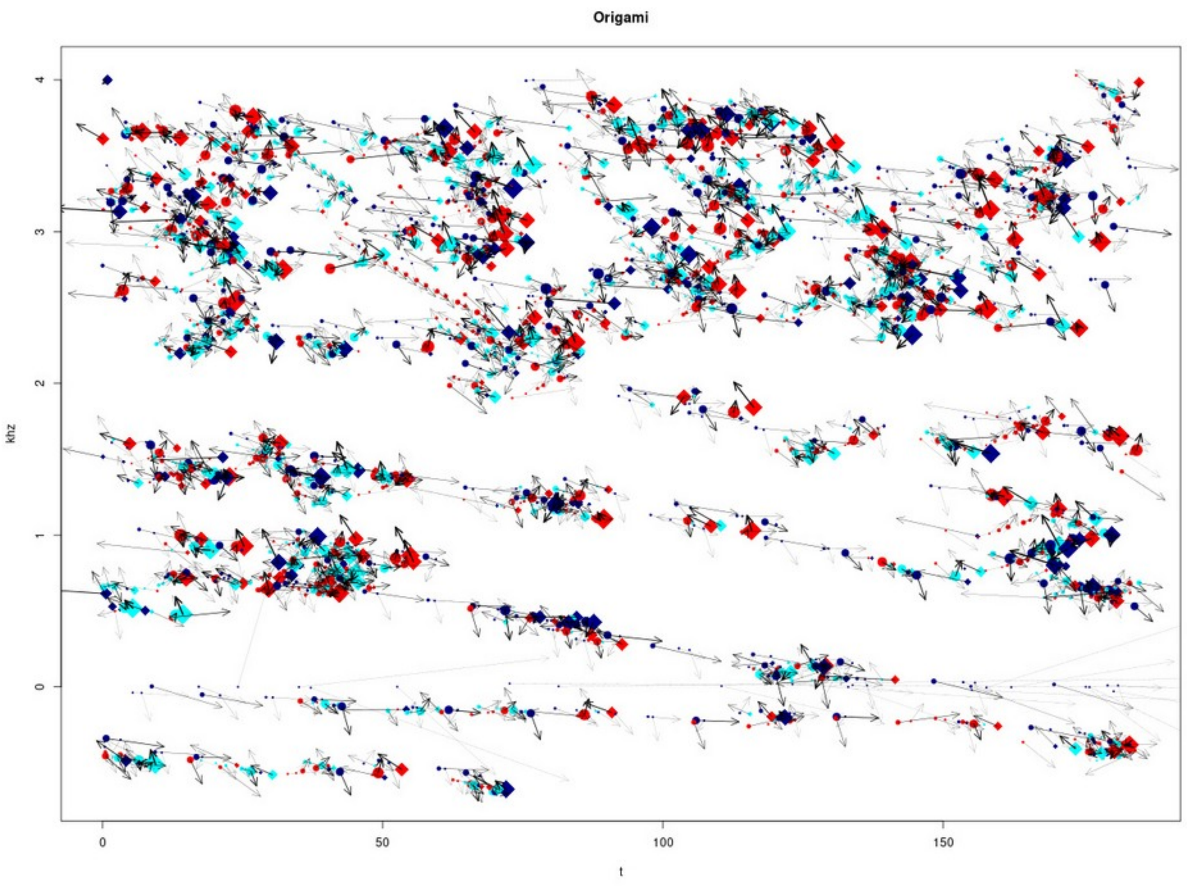
*Speed of Thought* is the only one of the eleven pieces that relies upon stochastic processes. Instead of using the common pseudo-random number generators I decided to explore the inherent indeterminacy of the computer by measuring the time elapsed between certain events. The standard library function clock() returns the current time since the program was started, and by invoking the function twice the duration between two function calls can be measured. Now, since the computer's operating system often handles other processes in the background while executing a program, the timing may differ even between two calls to execute the same piece of code. That indeterminacy is exploited to generate random numbers.

This idiosynchratic source of randomness is used as a gritty noise source, and to update synthesis parameters for pitched sounds that are generated by a simple method of concatenation of line segments into waveforms. As a proof of concept it works, but those interested in pursuing the idea should be warned that it is an extremely slow and inefficient way of generating random numbers. Moreover, there is no guarantee that the random numbers thus generated fit any quality criteria of psudo-random numbers. In fact, the pros and cons of using the computer's clock for random bits has been discussed in the context of cryptography (Schneier, 1996, p.424).

*The Wastebasket of Timbre* borrows its title from a paper by Siedenburg & McAdams (2017). It uses wavetable synthesis to exhibit a collection of more than 20 different timbres in something reminiscent of a serial structure. In fact, one of the points Siedenburg and McAdams make is that one should not confuse timbre as a perceptual attribute of sound with instruments or synthesis algorithms. Acoustic instruments may have different timbre depending on register, dynamics, and articulation. The synthetic and static timbres of this piece lack nuance, rather it is the plethora of different timbres in different voices that contribute to the rich palette. Two tuning systems are used in parallel, just intonation and Pythagorean tuning, both with an added tempered tritone. This piece (as well as the opening piece *Hello World!*) differs from the rest by a more traditional hierarchic separation between sound synthesis, tuning, and the note level.

*Origami* appears as a texture of sinusoids of varying length and articulation. The idea was to break the usual linear flow of time where the program generates the soundfile from beginning to end, from the first sample to the last. Instead, the time may change direction and zigzag its way through the piece. To be precise, it is the correspondence between the sequence in which the piece is generated and the temporal unfolding of the finished piece which is broken. Think of it is as a large score laid out before us. We write by drawing lines in any direction with a pen, without lifting it from the paper. Furthermore, the beginning and the end are joined like a cylinder, so that when the tip of the pen reaches past the end we continue writing from the beginning (Fig. 2).

FIGURE 2. A symbolic score of the piece *Origami* made by plotting some of the data, including onset time and frequency (on the horizontal and vertical axes) as well as amplitude (size), glissando slope and arrows indicating direction of time. Colours represent three different output channels.



The procedures that determine the parameters, which are amplitude, frequency, glissando slope, and writing direction (forwards or backwards in time) are relatively simple. Perhaps one could analyse the program and find a way to implement it such that it writes samples in an ordered succession from beginning to end. If so, the program could run in realtime. But why should it? What differentiates composing from improvisation is that events do not have to be invented in the same order as they appear in the piece of music. The trap of interactive music making is that quite a few interesting processes cannot run in realtime.

*The Quasi-periodic Table* uses sound synthesis by linear ordinary differential equations with quasi-periodic coefficients. Already a two-dimensional linear system with constant coefficients can have oscillating solutions, but here the coefficients vary over time, which makes the system rather difficult to understand. (For the curious, an in-depth analysis can be found in Haken (2004, Part II, ch. 3)). Specifically, the variables live in the space of two-dimensional complex numbers and the system equation is simply

$$dZ/dt = Q_t Z$$

where the four coefficients of the matrix Q are sums of complex exponentials at mutually incommensurate frequencies. In this case the matrix uses only six frequencies, two of its elements being single sinusoids and the other two mixtures of two frequencies each. It is perhaps not obvious that this seemingly simple synthesis model should be capable of rather complex sounds, but for some reason there is intermodulation between the frequencies which produces complex spectra. The piece evolves, although not much, by selecting different values for the coefficients, which are drawn from a collection of constants.

*The Final Countdown* uses three kinds of cross-coupled feedback FM instruments to generate all sound material. The output signal from one of the oscillators is analysed using low level signal descriptors (such as zero crossing rate, local total variation, and amplitude estimates). These signal descriptors influence certain synthesis parameters in a scheme that has been described as a ”feature extractor feedback system” (Holopainen 2012, p. 206), although in this case several other functions also update the parameters. Another technique that has been used in this piece is to occasionally record the output of a signal descriptor, and to read the recorded data cyclically in loops before mapping it to synthesis parameters. The FM systems are quite complex with up to six interacting oscillators. Dynamic systems and chaos is a better context for understanding these systems than anything that has been written about FM theory. The oscillators do not have easily controllable pitch, they may burst into noise cascades or turn silent for no apparent reason. One has to resort to trial and error. The large number of parameters of the FM oscillators makes it hard to map out regions of the parameter space corresponding to stable periodic oscillations, chaos, or oscillation death. This is one of the more complicated of the eleven pieces, both in terms of code size, and because of the nature of the oscillators.

## 2. Sound processing and mixing

While there are no theoretical limits to how sound can be sculpted using digital synthesis and processing, the richness of acoustic or even analog electronic sounds can be hard to obtain. That is one reason why I decided to submit the output of the programs to further processing. Another reason is the detached approach of thorough algorithmic composition where one is almost left with a feeling of having passively witnessed the creation of the music. The sound processing lets in a shred of subjectivity and happenstance, while also linking this to some of my other works. Finally, a third reason for the sound processing is that the source code for the compositions is available for anyone to experiment with, thus making it possible to reconstruct the raw output, or at least something very similar to it. By further processing the output, my version would retain a stamp of originality.

Electronic processing was carried out using analog filters, frequency shifters, ring modulation, spring reverb, flanger effects, and distortion. Mono tracks were processed individually using various eurorack modules, sometimes producing stereo tracks.

Acoustic processing, soundwise the most important treatment, consisted of playing the original soundfiles through exciters attached to vibrating objects, such as a guitar and a drum, as well as ordinary loudspeakers clad with aluminum foil. These instruments and prepared speakers buzz and rattle, they distort the sound and colour it in quite pronounced ways with strong resonances. The somewhat sterile synthetic sound takes on an acoustic appearance. This sound is recorded and used in the final mix together with the analog electronic processed sounds and the original output files.

I use a similar setup of acoustic processing in the live performance work *Neon Meditations* (in collaboration with visual artist Per Hess) for modular synthesizer, coloured neon lights, and colour sensitive light sensors. My first use of acoustic processing goes back to some of the pieces on the album *Signals & Systems* (2015)[2], which also contains some examples of thorough algorithmic

2 See https://ristoid.net/music/signals_systems.html

composition using sound synthesis.

## 3. Computational and musical complexity

Thorough algorithmic composition has its allure, but can be a cul-de-sac. It promises surprising discoveries of unexpected music through the manipulation of structures of interconnections, by search for the right coefficients in equations, by the construction of representations of music or models of sound. One is liberated from the endless doubts associated with the low-level choices that may permeate manual, free, or intuitive composition, whatever the non-algorithmic variety should be called. At least that is how it may appear at first.

We might think that the sound of the finished composition is the only thing that matters for the audience and that the process of making the piece and the thoughts that went into it play no role whatsoever. (If that were the case program notes would never be required.) For listeners who listen for the sake of a pure hedonic pleasure this may be so, but contemporary music and art usually relies on some concept or narrative. The contemporary art scene is full of works that make no sense without an accompanying explanation, a narrative that often cannot be gleaned from the surface of the work (Heinich, 2014, ch. 10). This emphasis on conceptuality may be more pronounced in visual art than in music, although a comparison across the arts is difficult to make.

In a slightly different context, Bayles & Orland write:

> To the viewer, who has little emotional investment in how the work gets done, art made primarily to display technical virtuosity is often beautiful, striking, elegant...and vacant. (...) Compared to other challenges, the ultimate shortcoming of technical problems is not that they're hard, but that they're easy. (Bayles & Orland, 1993, p. 96)

The Kolmogorov Variations certainly face technical problems, even as their raison d'être. An overarching goal was to generate complex music using simple algorithms.

In the 1960's Kolmogorov, along with Chaitin and Solomonoff, independently came up with the concept of algorithmic complexity. An object can be described as a text string, and this text string may be the output of a computer program. Algorithmic complexity measures the complexity of that object by the length of the shortest program that generates its description as its output. It is not a very practical complexity measure, but nevertheless an important theoretical construct.

Trying to adhere to the guiding principle of composing as complex music as possible using as few lines of code as possible is precisely what makes the eleven pieces hard. Unfortunately, those two goals seem to firmly oppose one another[3].

A concise program has low algorithmic complexity, and a lengthy program that cannot be condensed into something shorter has high algorithmic complexity. But the concept has its shortcomings. Consider a program that implements a dynamic system with a bifurcation parameter, such that the output is either a periodic or a chaotic orbit depending on the value of the parameter. Although the program is structurally identical, even a small change of the parameter may produce completely different output. This fact might provide hope that we can find interesting regions in the parameter space of a system implemented by a relatively small program.

Musical complexity is less easily defined. Several dimensions must be taken into account, including melodic, harmonic, rhythmic, and timbral complexity, perhaps some measure of unpredictability, large scale form, or variation over multiple temporal scales (Mauch & Levy, 2011). Musical complexity, understood as a multi-dimensional construct, can be increased by the inclusion of more structure, new rules and exceptions to the rules, functions and interconnections between parts of the program, all of which increase the program's length.

More iterations of a process may also yield more complex output even though the program remains structurally identical. Consider the successive stages of a Koch snowflake or any other fractal generated by iterating a function. Again, the output's complexity does not depend on the program's size, only on the number of iterations.

Brevity of code depends on the programming language used. In audio programming, a MAX/MSP

3 For a possible exception one may consider the case of so called *byte beat*, where single line expressions are turned into sound. These programs consist of a for loop, and the output is some function of the iteration variable, using logic, bitwise operators and arithmetic functions.

or Pd patch looks different from a SuperCollider or Csound instrument, which again looks different from code in Lisp, Python, or C. As we descend towards low level languages the size of the program tends to grow, because higher level languages hide complicated functions in convenient objects. An object that performs an FFT may look innocuous on the canvas of a graphical language, but in fact may encapsulate hundreds of lines of code. What may look like a simple patch in some domain specific high-level language probably is not so simple if one lifts the lid and looks into the functions or objects.

Kolmogorov Variations were written in C++ with a minimum of external libraries. Functions responsible for sound synthesis and other functions responsible for composition properly speaking (when the distinction is relevant) all fit into small programs, sometimes as short as about 250 lines of C++ code. Some programs are significantly longer. As discussed in the introduction, the style imitating branch of algorithmic composition relies on corpora of music to analyse. It bears pointing out that none of the programs used for Kolmogorov Variations take any external input whatsoever. Programs that take an input may generate an output as complex as the input without much ado, although it is a nontrivial matter to generate output that differs substantially, and in interesting ways, from the input. It seems fair to somehow account for the amount of necessary input, as well as the amount of computation, when comparing the complexity of two programs.

In Kolmogorov Variations the code is not always well organised. It is rather the proverbial obfuscated spaghetti code that results from a spontaneous project. Things that would be neatly separated in orderly code want to connect together, and interconnections make for a mess if one is not careful. In at least one case I had a working version of the program and wanted to try some small variation of it. Then it crashed. It was only possible to get the program running again after a lengthy debugging process.

Some distinctive traits may be found in the process of composition-as-programming. It is a process of growth, different from the growth of a handwritten score written from the beginning to the end, and maybe revised a few times. In the algorithmic approach the complete piece may be realised, albeit in a sketchy version, already at an early stage. Further work can address details, refine timbral qualities, articulations, tilt the form of the entire piece, use different criteria for certain decisions, and so on. And ”further work” typically means writing more lines of code. It could also mean erasing a few lines of code, although that tends to happen less often. The process can continue as long as the structure of the program remains more or less comprehensible. At some point a limit is reached and the program turns opaque, inscrutable. Any system that is sufficiently complex will escape our full understanding, this is an insight systems thinkers are well aware of (Meadows 2008, ch. 4). Creating systems that surprise us is in fact an often-stated purpose of music making using complex systems. In algorithmic composition it would be truly surprising to find a really simple program capable of generating highly complex music, but one can still hope for it.

## REFERENCES

BAYLES, David and ORLAND, Ted. *Art & Fear*. Observations on the perils (and rewards) of artmaking. Santa Cruz: The Image Continuum, 1993.

HAKEN, Hermann. *Synergetics*. Introduction and Advanced Topics. Berlin, Heidelberg: Springer, 2004.

HEINICH, Nathalie. *Le paradigme de l'art contemporain*. Structures d'une révolution artistique. Éditions Gallimard, 2014.

HOLOPAINEN, Risto. *Self-organised Sound with Autonomous Instruments*. Aesthetics and experiments. Doctoral thesis, University of Oslo, 2012.

LOY, Gareth. *Musimathics*. The mathematical foundations of music, Vol 1. Cambridge, Massachusetts: The MIT Press, 2006.

MAUCH, Matthias and LEVY, Mark. Structural Change on Multiple Time Scales as a Correlate of Musical Complexity. *Proc. of the 12th International Society for Music Information Retrieval* (ISMIR 2011). Miami, USA, pp. 489-494, 2011.

MEADOWS, Donella. *Thinking in Systems*. A primer. White River Junction, Vermont: Chelsea Green Publishing, 2008.

NIERHAUS, Gerhard. *Algorithmic Composition*. Paradigms of Automated Music Generation. Wien, New York: Springer, 2010.

SCHNEIER, Bruce. *Applied Cryptography, Second Edition*: Protocols, algorithms, and source code in C. New York: Wiley, 1996.

SIEDENBURG, Kai. & McADAMS, Stephen. Four Distinctions for the Auditory “Wastebasket” of Timbre. *Front. Psychol*. 8:1747, 2017.

SPROTT, Julien C. *Elegant Chaos*. Algebraically Simple Chaotic Flows. World Scientific, 2010.