Time-variant infrastructures and dynamical adaptivity for higher degrees of complexity in autonomous music feedback systems: the Order from noise *(2017) project*

Dario Sanfilippo

Received 26th July 2017
Revision 17th September 2017

Introduction

Order from noise (2017) is a project implementing one of my first prototypes of systems based on the idea of *dynamical adaptivity*, which is one of the most important aspects of my recent research and artistic practice. All my work is based on the idea that autonomous emergent systems (Corning 2002) – more specifically complex adaptive systems (Baranger 2000; Benkirane *et al.* 2002; Kitto 2006; Mitchell 2006; Morin 2007) – are key and substantially essential for achieving formal, performative and technical innovation in the context of live performance and music composition in general. The generally accepted idea according to which there is a 'master-slave' relationship between human and machine is entirely rejected and replaced by the concept of *hybridisation*: a condition where human and machine are inseparable and incessantly cooperate and coevolve to let the performance emerge. Within this framework, the machine is not a subordinated means but an *entity* which shapes the aesthetic and formal development as much as the performer does. In some cases, this idea can be pushed even further with works where the machine is the sole performing entity, as it happens for the piece presented here.

Order from noise (2017) is based on a time-variant feedback delay network containing a set of entangled nonlinear processing algorithms for audio and information signals. The work is an example of autonomous self-performing system and it is realised by feeding the network with one millisecond of background noise from the performance environment. The initial recirculating noise impulse is what entirely determines the formal evolutions of the system which have substantially different long-term developments for each different noise impulse. An approach to present the work live is that of reinitialising the system a number of times for a period of about three-five minutes to show its sensitivity to initial conditions and the long-term divergence between the formal structures. Here, instead, the recording is a non-edited extract from a single performance of the machine (in a two-channel configuration) which took place in studio in April 2017, and which depicts the evolution of the system and its performing capabilities from the beginning – its initialisation – to an arbitrarily chosen ending.

Before discussing the technical details regarding the software implementation, though, it is important to clarify a few concepts in order to briefly express the idea of dynamical adaptivity.

Generally speaking, the term *adaptive* refers to interacting agents that, individually or collectively, can change their state in response to variations in the environment or other interconnected agents. These changes can take place in the short-term, where the state of the agents is temporarily affected, or it can happen in the long-term, with permanent or long-lasting variations in their states.[1] In the field of complex adaptive systems, the term sometimes refers to a more specific behaviour, namely that of systems which are capable of changing their state in response to the environment or context to maintain a particular condition (to survive, for example) or to improve themselves (reaching a goal or target, for example).

Here, I will use the term in a more general sense, referring to systems which are capable of changing their state based on the specific *context* (or *history*, we might say) that they *experience*[2] at any given time. I prefer to use the term "context" rather than "environment" to include systems which are structurally coupled with the environment as well as closed systems which are coupled with themselves without an external environment. In both cases, I am referring to recursive systems, that is systems which provide the context that, circularly, affects their own state.

It is also necessary to make a distinction between *time-invariant* and *time-variant* systems. In simple terms, a time-invariant system is a system which performs the same operation *at all times* (Smith 2008). In the other case, we will have a system whose operation changes over time. Another important distinction, strictly related to the one above, is that between *dynamical* and *adaptive* systems. The output of a dynamical system changes over time, but the internal state of its agents may remain unaffected. On the other hand, an adaptive system implies that its internal state and, very likely, its output, too, change over time. As a practical example, we can consider an analogue mixer with a feedback configuration. Some specific setup of the parameters may result in an output that, to some extent, changes over time, although the parameters of the mixer themselves will be static. This would be a time-invariant system. On the other hand, a simple example of an analogue time-variant and adaptive system could be a voltage-controlled filter with a feedback configuration: the output of the system (the context, in that case) will change the internal state of the filter, which, in turn, will affect the output.

Theories and motivations

While some interesting results can be achieved with dynamical systems, adaptive systems are more likely to generate behaviours which exhibit higher long-term variety

[1] https://www.complexityexplorer.org/explore/glossary. Accessed: 26th of July 2017.

[2] Whether a machine can, in some cases, be considered as something that experiences its context is an important issue involving computational phenomenology, artificial intelligence, cognitive science and other fields which I plan to discuss in another paper.

and complexity. Digital signal processing and audio programming provide very versatile tools for the implementation of time-variant adaptive systems in the domain of sound: ideally, provided that stability is taken into account, all variables in a DSP unit can be driven by sonic signals and can thus vary at sample rate. That way, the generated sounds and the states of the components can affect each other, making the system adaptive and time-variant. Practitioners like Gordon Mumma (Mumma 1967) and Nicolas Collins[3] (in the analogue domain), Agostino Scipio and myself (both individually and as a duo (Di Scipio 2008; Sanfilippo 2013; Sanfilippo & Di Scipio 2017) and others extensively adopt this approach for the implementation of such systems. A typical procedure is that of performing several kinds of analysis operations on the output such as RMS and brightness estimation to obtain infrasonic control signals. We could generally refer to this as *information processing*.[4] These signals, often based on their perceptual characteristics and their relationship with the domains of the variables in the processing units, are mapped to certain ranges and then used to control the state of the components in a large network. (See Di Scipio (2003) for a detailed discussion on this method). Using infrasonic signals to pilot these variables is highly desirable if not necessary, for high-rate, sudden changes in the DSP parameters would produce an output with a continuously large and homogeneous spectral band, so it would not be possible to perceive the state variations in the long-term.

The information and audio processing algorithms implemented, the specific connections between the control signals and the variables, the linear and nonlinear mapping strategies used as well as the network topologies, all these elements determine the *infrastructure* of a system. In a large network, these elements can already provide a high number of configurations and an even larger number of possible states that a system can reach. That, theoretically, could be considered as something that guarantees a good variety and complexity in the long-term behaviour of a system, albeit the practical case tends to be much different from the ideal scenario. In my experience, the realisation of an autonomous music system which exhibits a convincing variety and complexity over a relatively long time span has been something difficult to achieve, even when implementing large and articulated networks. The reason is possibly that one specific infrastructure roughly corresponds to one specific behaviour; that way, high-order patterns and attractors (Gleick 2011) may emerge over time decreasing the global complexity. Variety and complexity are convincing when, in the long-term, there is a non-trivial interplay between order and disorder, redundancy and entropy, sound and silence, repetition and surprise, as well as homogeneity and heterogeneity in the characteristics of the output. Ultimately, these all contribute to creating a behaviour which is expressive and organic: what, to some extent, can be considered a form of musical intelligence.

[3] http://www.nicolascollins.com/texts/peasouphistory.pdf

[4] The term "information processing" can assume different meanings depending on the context in which it is used. Here, I am referring to *any* processing technique used to transform an audio signal into a new signal which will eventually modulate one or more variables in a DSP unit. The question whether a machine performing an analysis on audio signals is *representing* information or if it is actually *generating* it is central for me and Di Scipio and we will address this issue in a different paper.

Theories and techniques

Partly inspired by the interface that I have implemented to perform my *Single-fader versatility* (2016), I thought that the autonomy of a system could have been improved by making its infrastructure time-variant, hence resulting in a *dynamical adaptivity*. Based on what characterises the infrastructure of a system, I decided to build a prototype where the ranges in the mapping functions between control signals and DSP variables change over time with regard to the input sound of each node. The result is a nontrivial interpolation between positive and negative feedback relationships for each signal-variable pairs, which in turn creates an ever-changing set of different adaptive modalities in the system.

The network of this project has six nodes, each of them containing two cascaded units, carrying out the following audio processing algorithms: asynchronous granulation, recursive comb filtering, variable high-pass/low-pass filtering, pulse-width modulation (PWM), resampling and 16th-order feedback delay network (FDN) processing. The entire system has been implemented using the Pure Data Vanilla[5] programming environment and the processing units have been designed so that all parameters can be modulated at sample rate, i.e. using audio signals.

The granulators are an extension of Miller Puckette's pitch shifter found in the Pure Data help patch G09. They use a pair of overlapping reading heads implemented through variable delay lines, which allow for the delay time to be modulated at audio rates and also make it possible to have fractional delays by interpolating samples. In Pure Data, the interpolating algorithm is a four-point cubic function, the same used for its wavetable lookup object used for the implementation of samplers.

A variable delay line can be considered as a model for a rotating tape loop to which a fixed writing head and a moving reading head are attached. The input of the writing head[6] is the signal which is being written on the tape; the input of the reading head is the delay after which that signal will be output. The length of the tape (D) is the maximum possible delay time and is the case where the reading head is just behind the writing head. On the other hand, if the reading head is immediately next to the writing head we will have a zero-delay output. All other positions, determined by the input signal of the reading head, are the possible delays between 0 and D.

Intuitively, if the reading head moves towards the opposite direction of the tape we will have an increase in speed. Conversely, if it moves towards the same direction of the tape we will have a decrease in speed (until the speed of the reading head exceeds that of the tape). The first case is a *pitch transposition* up, the second case is a transposition down. As discussed in (Puckette 2007), if $d[n]$[7] is the input of the reading head, setting the delay in samples, the pitch transposition factor $t[n]$ for delay lines is given by the following formula:

5 http://msp.ucsd.edu/software.html

6 Whenever terms such as "tape" or "head" are used, I am referring to them metaphorically as all the described operations are carried out in the digital domain.

7 As a convention for the mathematical formulae, variables within square brackets are integer numbers, while variables within round brackets are real numbers.

$$t[n] = 1 - (d[n] - d[n-1]).$$

Using this formula, we can calculate the slope of a line which represents the delay variation necessary to perform the desired transposition, although this variation can only take place for a limited period given the finite length of the delay line. A continuous transposition can be achieved using two overlapping reading heads. These heads are 180° out of phase and their cycles are smoothed out using windowing functions to avoid audible discontinuities. In fact, this is the reason why such a design can be used for granular processing: each cycle of a reading head is a portion of sound which is being read, and this portion can be of any duration between 0 and D, thus short enough for granular processing.

The remaining parts of the granulator are the *grain rate* – which, in this design, is linked to the grain duration – and the *time transposition* or *time stretching*. Once the slope for a given pitch transposition has been calculated, keeping the same transposition for different grain rates is only a matter of scaling down the delay variation size to have a constant slope. Time transposition can be implemented by consistently offsetting the grains with regard to the movement of the tape, and we can still use the transposition formula above to calculate the slope of the offset. Besides, this design is consistent with negative transposition factors of both time and pitch which will result, respectively, in exploring the buffer backwards and playing the grains in reverse.

Rocchesso and De Poli (Rocchesso 2003) have described granular processing as

$$y_g[n] = \sum_k A_k g_k[n - l_k],$$

where $y_g[n]$ is the signal resulting from the combination of k grains having their own amplitude (A_k), waveform ($g_k[]$) and temporal location (l_k).

These parameters are time-variant in my system, so we can rewrite the equation as

$$y_g[n] = \sum_k I_{1,k}(x[n])g_k[\mathrm{n} - I_{2,k}(x[n])]$$

where $I_i()$ represents an information processing algorithm (we will discuss it later) that will modulate each time-variant parameter. To be precise, $I_i()$ is itself a time-variant function and that is the essence of the *dynamical adaptivity* idea developed here, although, for the sake of simplicity, it will not be indicated in the mathematical representations. (Note that, although the granulator or other units may have a feedforward structure, the system is still recursive for the overall output will be fed back into its input.) If $w_d[]$ is a windowing function of the form

$$w_d[i] = cos((iT_d - 0.5)\pi),$$

we can complete the description of this unit by adding the pitch and time transposition parameters and rewriting the grain function $g_k[]$ as:

$$g_k[i] = w_d[i]z(s(I_{3,k}(x[i])) + p(I_{4,k}(x[i]))).$$

Here, the *positive sine* windowing function of *d* samples modulates the output of a five-second delay line, *z*()[8], where *s*() and *p*(), which are in turn modulated, provide the information for time and pitch transposition.

Recursive comb filtering, too, is implemented using variable delay lines so that both the feedback delay and the feedback coefficient can be modulated with signals. The algorithm can be described by the following formula:

$$y[n] = x[n] + I_1(x[n])y(n - I_2(x[n])).$$

As we will see later, using coprime delay lengths is a technique used in FDN reverberators to minimise the overlapping of poles (Schroeder 1973). A similar approach is used here to maximise the distribution of resonances in the spectrum and, consequently, the possibility of sonic emergence, though, in this case, the delays of the two cascaded combs are 'pseudo-coprime' numbers: the delay length of the comb filters are simply prime numbers representing a delay in milliseconds raised to a time-variant, non-integer power. If *SR* is the sample-rate, $I_2(x[n])$ can be rewritten as:

$$(prime^{I_2(x[n])}/1000)SR.$$

The third unit is a variable filter, that is a filter that can gradually morph from high-pass to low-pass and vice versa. This is simply realised with three cascaded one-pole IIR systems of the form:

$$y[n] = (1 - |B|)x[n] + By[n - 1].$$

The input coefficient is calculated as the complement of the absolute value of the feedback coefficient to normalise the output of the system, and the feedback coefficient is limited to $-1 \geq B \geq 1$ to keep the system stable. This system will have a resonance at DC (0Hz) for positive values of *B*, and a resonance at Nyquist (*SR*/2) for negative values of *B*. The absolute value of the feedback coefficient determines how close the cutoff of the filter is to its resonance (though the relationship between *B* and the cutoff is nonlinear (Chamberlin 1984)). To get a better understanding, we can consider three special cases: with $B = 1$ we will have a low-pass with a cutoff at 0Hz; with $B = 0$ the input signal will be unaltered; with $B = -1$ we will have a high-pass with a cutoff at Nyquist.

This unit has been designed with an internal negative feedback mechanism: the feedback coefficient *B* is actually the output of the filter itself. This way, the greater the output amplitude, the narrower the passing band of the filter and the more energy will be attenuated. More precisely, *B* is calculated by first multiplying the output of the filter by a factor, writing it on a 20-second variable delay line, raising the output of the

[8] Note that the input of the delay line setting the delay can be a non-integer number as the output is interpolated.

delay line to a power,[9] filtering it with a low-pass at 20Hz to slow down the variation, and finally limiting it using a hyperbolic tangent function. The factor, delay time and power are time-variant parts of the unit, thus we can rewrite *B* as follows:

$$B = tanh(lp_{20}((I_1(x[n])z(I_2x[n]))^{I_3(x[n])})).$$

The PWM unit is also somewhat simple and we only have a pulse train which modulates the input of the unit. The frequency and pulse width of the pulse train are themselves modulated by the information processing of the input signal.

If $pt(f, w)$ represents a pulse train at frequency f with a pulse width of w%, we can describe the unit as:

$$y[n] = x[n]pt(I_1(x[n]), I_2(x[n])).$$

The resampling unit has the same structure as the granulator, in that it also uses two windowed overlapping reading heads from a variable delay line. The main differences are that the size of the frames are greater than 0.1 seconds and that there is no temporal displacement of the frames. Essentially, only the pitch is affected, though the windowing function is raised to a positive power to have narrower or wider window shapes which can result in amplitude modulation effects. In this case, the grain function $g_k[]$ can be rewritten as:

$$g_k[i] = w_d[i]^{I_{1,k}(x[n])}z(s(1) + p(I_{2,k}(x[\mathrm{n}]))).$$

Finally, the last unit in the system is based on a 16th-order FDN, a model often used for artificial reverberation, although the different kind of internal processing algorithms and parameters in my design generate an output that not always resembles that of a reverberator.

An FDN system of order N can be described by the following relations:

$$y[n] = \sum_{i=1}^{N} z_i[n]$$

$$z_i[n] = \sum_{j=1}^{N} B_{i,j}z_j(n - m_i) + x(n - m_i).$$

This is a simplified version of an FDN as there is no direct input and no scaling factor for the outputs of the delay lines. $B_{i,j}$ represents the coefficients of a feedback matrix of type Hadamard (not indicated above for simplicity) that is used to maxim-

[9] The base can be a negative number and the exponent is fractional. This could result in imaginary numbers which are not representable in Pure Data, so the computation is simply realised by taking the absolute value of the input, raising it to the power, and then multiplying the result by the sign of the input. All other power operations described here which imply negative numbers raised to fractional powers are calculated in the same way.

ise the interactions among the recirculating signals and, particularly for the design of reverberators, model the phase cancellations and reinforcements that take place in a room. Low-pass filters are often implemented after the delaying stage of these networks to simulate the faster decay of high-frequency components. In my design, the low-pass has been replaced by the variable filter ($f_{coeff}()$) which, together with $B_{i,j}$ (whose stability threshold is given by $1/\sqrt{N}$) and the lengths of the delay lines (chosen as 'pseudo-coprime') are the time-variant elements of this unit. The second relation above can then be rewritten as:

$$z_i[n] = \sum_{j=1}^{N} I_{i,j}(x[n]) f_{I_j(x[n])}(z_j(n - I_i(x[n])) + x(n - I_i(x[n]))).$$

Even though it was not shown in the mathematical descriptions, the units that implement some internal feedback mechanism also have a lookahead limiting stage in the loop to make sure that the system is stable. These limiters are implemented through peak envelopes using a delay of two milliseconds for the input signal. More precisely, the peak envelopes will decay of -60dBs in 10 seconds and the attack of the limiter is slightly smoothed out by a low-pass filter which is synchronised with the input delay. L is the limiting threshold and, of course, the signal is unaltered (except for a delay) if below that value:

$$y[n] = x[n - [0.002 * SR]]min(1, L/lp_{500}(peakenv_{10}(x[n]))).$$

The global structure of the system, too, is recursive. It is also self-oscillating, and it is therefore necessary to make sure that the system is stable. To achieve this, and to establish a convincing relationship between the emergence of sound and the emergence of non-sound, I implemented a stage of dynamical compressors after the output of the units, before they are fed back into themselves. The compressors use peak envelopes with a 100-second decay and the complement of their output, after being raised to a time-variant positive power, is used to scale down the input:

$$y[n] = x[n](1 - peakenv_{100}(x[n]))^{I(x[n])}.$$

The information processing method used to generate the control signals is different from the one described above: rather than calculating the RMS or brightness, I am using a form of input-dependent nonlinear LFO. This is achieved by simply low-passing the input signal to slow it down with cutoffs as low as ~0.01Hz; the result is then processed using dynamical normalisation to roughly keep the signal within the [-1; 1] range; it is raised to some relatively large power to force it around 0 and finally used to pilot the frequency of a unipolar sawtooth oscillator. The aforementioned function $I(x[n])$ can then be described by the following relations:

$$y[n] = (nT_{SR}f)mod1$$

$$f = norm(x[n], lp_{0.01}(x[n]))^{exp}.$$

The dynamical normaliser is based on RMS estimation and has two inputs, a reference signal and a signal to be normalised according to the reference signal. Of course, the RMS window needs to be large enough depending on how slow the normalised signal is to have a consistent result:

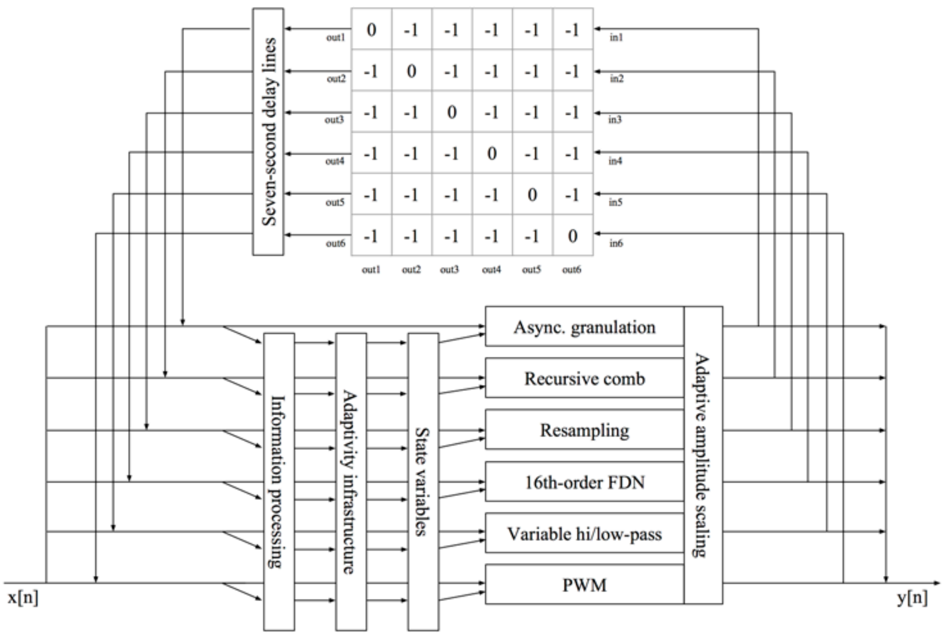
$$norm = x_{norm}[n](RMS(x_{ref}[n])/RMS(x_{norm}[n])).$$

The first difference that we can notice, compared to the information processing techniques based on RMS or brightness, is that, instead of having one and only one value being generated with a specific input, the output of this algorithm will depend on time. Of course, this also introduces some degree of *opacity* in the way the system functions. One reason is the one that I just described: the fact that both input and time will affect the output. The other one is that the operation of low-passing simply averages a signal, which does not exactly have a perceptual correlate such as brightness or loudness. This algorithm, indeed, could be considered as a *black box,* although the system is still entirely deterministic and dependent on its context. In fact, triggering the system with the same initial conditions each time would produce the same stream of samples, while slightly different inputs would result in entirely new formal developments. The reason why I wanted to explore this new design is to enhance the unpredictability of the system by not having an explicit relationship between output and variables. I also wanted to have a low-level mechanism regulating the system (low-pass filtering) so that it would determine its behaviours and structures in a more 'personal' way than one following a human-implemented perceptual model.

In this system, the information processing takes place in two stages: on one hand, it is used to pilot the variables in the DSP units according to specific mapping functions (adaptivity); the upper level in which information processing also takes place is what modulates the mapping functions themselves, resulting in different modes of self-structuring (dynamical adaptivity).

The units are interconnected through seven-second delay lines with a 'quasi-fully-connected' network topology. The information processing algorithms will determine the range of action and the mode of action of the control signals, resulting in a dynamical adaptivity, and they will generate the control signals themselves which will affect the state variables of the units. The overall design of the system can be schematised as shown in the following diagram.

The diagram shows the overall configuration of the system. The input, which is a one-millisecond background noise captured from the performance environment is sent to the information and sound processing modules. The outputs from the information processing modules will determine the specific relationships between information and DSP variables and, ultimately, the specific values which will affect the state variables of the DSP units. The outputs from the DSP units are processed through adaptive amplitude scaling algorithms to make sure that the network is stable and will be fed back into themselves and the information processing units after seven seconds through a unity-amplitude, phase-inverted, quasi-full matrix. The outputs of the DSP units are then summed together to make up one channel. The work



presented here is a stereo version and two coupled systems like the one showed in this figure have been used.

Conclusion

Other implementations which I am planning to explore will make the infrastructures dynamical by reconfiguring the connections among variables and control signals when using perceptually-related analysis algorithms, or by interpolating among different analysis algorithms while keeping the same connections. This could be determined by using a high-level analysis algorithm, namely a complexity index estimator which combines low-level analysis algorithms (RMS, brightness, noisiness, roughness), recurrence quantification analysis, average absolute deviation and nonlinear processing. The idea of *meta information processing*, too, will be explored, which involves control signals affecting the variables in algorithms generating other control signals, thus making them time-variant. The time-variant systems approach could be pushed even further by having *dynamical nodes* and *dynamical topologies*. It means that each node would be morphing through several processing techniques and that the way they are interconnected would be varying over time. In this situation, all characterising elements of a system will be changing, realising the *system of systems* (Morin 1992) paradigm even more profoundly, as well as the shift from sound to *music synthesis*.

References

Baranger, M. (2000) *Chaos, complexity, and entropy*. Cambridge: New England Complex Systems Institute.

Benkirane, R. (2002). *La complexité, vertiges et promesses: 18 histoires de sciences:[entretiens avec Edgar Morin, Ilya Prigogine, Fransisco Varela...]*. Le pommier.

Chamberlin, H. (1984). *Musical Applications of Microprocessor*. Sams.

Corning, P. A. (2002). The re-emergence of "emergence": A venerable concept in search of a theory. *Complexity*, *7*(6), 18-30.

Di Scipio, A. (2003). 'Sound is the interface': from interactive to ecosystemic signal processing. *Organised Sound*, *8*(3), 269-277.

Di Scipio, A. (2008). Émergence du son, son d'émergence: Essai d'épistémologie expérimentale par un compositeur. *Intellectica*, *48*(49), 221-249.

Gleick, J. (2011). *Chaos: Making a new science (Enhanced edition)*. Open Road Media.

Kitto, K. J. (2006). *Modelling and generating complex emergent behaviour*. Flinders University, School of Chemistry, Physics and Earth Sciences..

Mitchell, M. (2006). Complex systems: Network thinking. *Artificial Intelligence*, *170*(18), 1194-1212.

Morin, E. (1992). From the concept of system to the paradigm of complexity. *Journal of social and evolutionary systems*, *15*(4), 371-385.

Morin, E. (2007). Restricted complexity, general complexity. *Science and us: Philosophy and Complexity. Singapore: World Scientific*, 1-25.

Mumma, G. (1967). Creative aspects of live-performance electronic music technology. In *Audio Engineering Society Convention 33*. Audio Engineering Society.

Puckette, M. (2007). *The theory and technique of electronic music*. World Scientific Publishing Co Inc.

Rocchesso, D. (2003). *Introduction to sound processing*. Mondo estremo.

Schroeder, M. R. (1973). Computer models for concert hall acoustics. *American Journal of Physics*, *41*(4), 461-471.

Smith, J. O. (2008). *Introduction to digital filters: with audio applications* (Vol. 2). Julius Smith.

Sanfilippo, D. (2013). Turning perturbation into emergent sound, and sound into perturbation. *Interference: A Journal of Audio Culture*, (3). Available online: http://www.interferencejournal.org/turning-perturbation-into-emergent-sound/.

Sanfilippo, D. and Di Scipio, A. (2017) Environment-mediated coupling of autonomous sound-generating systems in live performance: an overview of the *Machine Milieu* project. Espoo, Finland: *Proceedings of the 14th Sound and Music Computing conference*. 21-27.

One recording of the *Order from noise* (2017) project

http://www.fupress.net/public/journals/18/2017-2018/sdos_iir-1.wav