Proceedings of the 2nd International Faust Conference (IFC-20), Maison des Sciences de l'Homme Paris Nord, France, December 1-2, 2020

THE EDGE OF CHAOS LIBRARY: A LARGE SET OF FAUST FUNCTIONS FOR THE IMPLEMENTATION OF MUSIC COMPLEX ADAPTIVE SYSTEMS

Dario Sanfilippo

Independent researcher
sanfilippo.dario@gmail.com

ABSTRACT

This paper presents the Edge of Chaos library, a set of Faust functions dedicated to the implementation of music complex adaptive systems through time-variant audio feedback networks. The paper first describes the background and the motivations behind the library, then it discusses the library modules and some of the most relevant functions. The library includes eight modules for information processing, sound processing and generation, stability processing for self-oscillating designs, and mapping strategies. Lastly, the paper discusses future work for the library to transform it into a music complex adaptive systems generator software for live performance. The library is available on Github[1] and it is published under the GNU GPL v2.0 license [2].

1. BACKGROUND

The early studies on complexity are from at least the 1970s with Edgar Morin and V. Rao Vemuri [1, 2], but several others were researching the same field from different directions. For example, Prigogine was investigating dissipative structures, non-equilibrium thermodynamics, and the function of time in biological systems [3]; the theories on autopoiesis by Maturana and Varela [4]; Kauffman and his work on random Boolean networks showing the emergent evolution of their self-organisation [5]; the research on artificial life by Langton [6]. In the 1970s, John H. Holland implemented a computational model of adaptation in evolutionary systems inspired by the work of Rosenblatt [7]. Holland published his early research on genetic algorithms in 1975 [8], and he was a distinguished researcher in the field of adaptation. [9]. Holland also became part of the Santa Fe Institute[3] in 1985, a place where some of the most prominent complex adaptive systems (CASes) thinkers like Melanie Mitchell and James Crutchfield currently work.

CASes are now used in several fields, and they have gained substantial importance during the years. Some of their applications are used to predict and understand complex real-world phenomena through computational models for economic trends or the development of technological progress [10, 11]; the evolution of intelligence [12]; or global behaviours in societies [13]. CASes are also applied in the implementation of artificially intelligent systems like self-repairing software and intelligent anti-virus systems [14]; or robotics and linguistics [15, 16].

Today, Alice Eldridge and Agostino Di Scipio have provided some of the most important contributions in the creative music practice with adaptive systems [17, 18, 19, 20]. Other works based on feedback and cybernetics are discussed in [21]. The author, too, has investigated music complex adaptive systems extensively in his research [22, 23, 24], as well as in collaboration with Di Scipio [25, 26].

Edge of Chaos is a FAUST library dedicated to the implementation of music complex adaptive systems and is the result of the author's 15-year-long research on feedback systems and complex adaptive systems for the generation of music through recursive and time-variant audio networks, particularly following an agent-based modelling approach [27]. The library comprises a large set of functions ranging from standard DSP techniques to original algorithms. These include functions for the extraction of low-level and high-level information, the processing and generation of audio streams through dynamical systems, linear and nonlinear mapping strategies to couple information and sound, as well as energy-preserving functions to guarantee stability in self-oscillating systems. Considering the intrinsic time-variant nature of complex adaptive systems, FAUST was an optimal choice for the implementation of the library due to its stream-oriented paradigm.

2. MODULES IN THE EDGE OF CHAOS LIBRARY

After a quick introduction on the artistic importance of feedback and adaptation in music, we can now present the content of this project. The library consists of a total of ten modules, eight of which are the fundamental ones containing DSP functions, and the remaining ones provide access to the functions environments. The modules are the following:

- allEOC.lib
- auxiliaryEOC.lib
- delaysEOC.lib
- edgeofchaos.lib
- filtersEOC.lib
- informationEOC.lib
- mathsEOC.lib
- oscillatorsEOC.lib
- outformationEOC.lib
- stabilityEOC.lib

2.1. AllEOC.lib and edgeofchaos.lib

AllEOC.lib and *edgeofchaos.lib* are access modules. This first one imports all of the remaining modules by means of the `import` function. The second one is equivalent to FAUST's *stdfaust.lib* and provides access to the remaining modules through a series of environments by means of the `library` function.

[1] https://github.com/dariosanfilippo/edgeofchaos

[2] https://www.gnu.org/licenses/old-licenses/gpl-2.0.en.html

[3] https://www.santafe.edu/about. Accessed on the 29th of August 2019.

2.2. AuxiliaryEOC.lib

AuxiliaryEOC.lib is a small module that contains the functions `dirac`, `step`, and `inspect`. The first two functions are the Dirac delta function and the step function, whereas the last function gives a visual representation of the values of signals, updated at block-size rate. This module is mostly used for testing and debugging and its environment prefix is "au".

2.3. DelaysEOC.lib

The *delaysEOC.lib* module is essentially the same set of delay lines that we can find in FAUST's standard library, except the delay parameter here is set in seconds rather than in samples for a samplerate-independent set of delays. The environment prefix for this module is "d2".

2.4. FiltersEOC.lib

The *filtersEOC.lib* module is dedicated to filters of different kinds. It includes the biquadratic filters designed by Robert Bristow-Johnson as well as some of the filters in [28, 29]. The module also includes a set of integrators, adaptive crossovers, and an excellent analytic filter designed by Olli Niemitalo[4]. The environment prefix is "f2". As an example, below we can see the code for Olli Niemitalo's analytic filter, which will be essential for the calculation of low-level information such as instantaneous amplitude, frequency, and phase:

```
analytic(x) =  real , imaginary
with {
    im_c =
        (   0.4794411608296202665 ,
            0.8762435889504858020 ,
            0.9766029691687165836 8 ,
            0.9974994041220337504 0);
    re_c =
        (   0.1617774170636316621 9 ,
            0.7330690130335572242 ,
            0.9453630196680627984 0 ,
            0.9906005141670404246 0);
    tf(c, y, x) = c * (x + y') - x'';
    imaginary = x' :
        seq(i, 4, tf(ba.take(i + 1, im_c))
            ~ _);
    real = x :
        seq(i, 4, tf(ba.take(i + 1, re_c))
            ~ _);
    };
```

Below, instead, we can see a second-order state-variable filter with zero-delay feedback topology. The filter has the following outputs: lowpass, highpass, bandpass, bandpass (normalised), lowshelf, highshelf, bandshelf, bandstop, peak, and allpass.

```
svf2blti(cf, q, k, in) =   tick
                           ~ ( _ ,
                               _) : (  ! ,
                                       ! ,
                                       _ ,
                                       _ ,
                                       _ ,
                                       _ ,
                                       _ ,
                                       _ ,
                                       _ ,
                                       _)
    with {
        r = m2.div(1, (2 * q));
        wa = (2 * ma.SR) *
         tan(m2.w(cf) / 2);
        g = wa / ma.SR / 2;
        tick(s1, s2) =  u1 ,
                        u2 ,
                        lp ,
                        hp ,
                        bp ,
                        bp_norm ,
                        ls ,
                        hs ,
                        b_shelf ,
                        notch ,
                        peak ,
                        ap
            with {
                u1 = v1 + bp;
                u2 = v2 + lp;
                v1 = hp * g;
                v2 = bp * g;
                hp = m2.div((in - 2 * r
                    * s1 - g * s1 - s2),
                            (1 + 2 * r *
                                g + g *
                               g));
                bp = s1 + v1;
                lp = s2 + v2;
                bp_norm = bp * 2 * r;
                b_shelf = in + k *
                    bp_norm;
                ls = in + k * lp;
                hs = in + k * hp;
                notch = in - bp_norm;
                ap = in - 4 * r * bp;
                peak = lp - hp;
            };
    };
```

[4] dsp.stackexchange.com/questions/37411/iir-hilbert-transformer/59157#59157

2.5. InformationEOC.lib

InformationEOC.lib is one of the most important modules and it contains functions, several of which are original designs, for low-level and high-level information processing. Some of the low-level algorithms include loudness, spectral centroid, roughness, and noisiness. Among the high-level ones we have dynamicity, heterogeneity, and complexity. The module also includes a vast set of peak envelope analysis functions based on both switching or cascaded architectures [30].

Algorithms such as spectral centroid and noisiness follow an

adaptive design. The spectral centroid is a self-regulating mechanism based on a negative feedback loop around a crossover that finds the cut-off at which the power of high and low spectra is approximately equal. Specifically, the system (minimally) oscillates around a balancing spectral point. The noisiness algorithm, instead, calculates the magnitude of the derivative of the zero-crossing rate, while the differentiation period is determined by the spectral centroid of the input signal, providing a noisiness index that follows more closely a perceptual behaviour. Other low-level information processing algorithms include the measurement of roughness and spectral spread, as well as lowest and highest frequency component measurements.

The high-level algorithms are based on principles of dynamical systems and complex systems, as well as recurrence quantification analysis and state-space analysis through absolute average deviation. These principles are applied to low-level information signals, which are then combined nonlinearly. Notably, these algorithms are dynamicity, heterogeneity, and complexity. The dynamicity algorithm detects variations in the low-level information signals. The heterogeneity algorithm is based on average absolute deviation and it is related to the unpredictability of the state-space of the low-level information signals. The complexity measurement, instead, is based on the concept of *edge of chaos* and it detects variations in the heterogeneity of the low-level information signals. These algorithms are discussed in detail in [31].

2.6. MathsEOC.lib

MathsEOC.lib is the largest module in the library with almost 2000 lines of code. The module includes functions for statistics, linear and nonlinear fuzzy logic, interpolators, network topologies, matrixes, linear and nonlinear mapping, windowing functions, hysteresis, angular frequency, and several time constants for exponential decays in one-pole systems. Furthermore, the module includes a set of sequences such as the Fibonacci sequence, to name one of the most famous, that may be deployed in feedback delay networks as a replacement for standard prime or co-prime delays for the investigation of new behaviours.

2.7. OscillatorsEOC.lib

The *oscillatorsEOC.lib* module is mainly for band-limited oscillators of classic waveforms based on integration of band-limited impulse trains (BLIT). Some of the functions implement the systems described in [32], such as the bipolar BLIT with arbitrary duty cycles. Others, instead, follow a different design or have been modified by the author. Particularly, unlike the technique described in by Stilson and Smith where the bipolar BLIT is implemented by summation of a unipolar BLIT with its delayed and inverted copy, the technique showed here uses an even ratio between the frequencies of the sine functions used to generate the sinc function. This results in a correct harmonic content (odd harmonics) for any given BLIT frequencies.

Despite these oscillators provide the most accurate frequency content for band-limited classic oscillators compared to FAUST's band-limited oscillators, the oscillators in the library presented here show irregularities in time-variant configurations due to the integrators. Furthermore, it is worth mentioning that the harmonic content of the oscillators in *oscillatorsEOC.lib* can be varied arbitrarily. Specifically, the variation is locked to the beginning of each cycle in the sinc function to avoid discontinuities, which allows for harmonic modulation as exploration of a new synthesis technique. Below, we can see the code for the bipolar BLIT. Note that the bipolar sinc function `asinc_bi` is in *mathsEOC.lib*.

```
asinc_bi(M, x) =
    m2.if(phase < ma.EPSILON,
        1,
        m2.if(abs(.5 - phase) < ma.EPSILON,
            -1,
            sin(M1 * m2.twopi * phase) /
                (sin(m2.twopi * phase) *
                    M1)))
      with {
          M1 = rint(M) * 2;
          phase = ma.frac(x);
      };
```

```
blit_bi(h, f) = m2.asinc_bi(h1, phase)
    with {
        lim = rint(m2.div(ma.SR, f) / 4);
        h1 = ba.sAndH(trigger, min(lim,h));
        trigger = (ma.signum(f) *
            (phase - phase') < 0);
        phase = os.phasor(1, f);
    };
```

Below, instead, we can see the code for what is arguably the best-performing recursive quadratic oscillator available, designed by Martin Vicanek[5].

```
osc_quad(f) =  tick ~ (_ , _)
    with {
        k1 = tan(ma.PI * f / ma.SR);
        k2 = 2 * k1 / (1 + k1 * k1);
        tick(u, v) =
            omega - k1 *
                (v + k2 * omega) ,
                    v + k2 * omega
            with {
                omega =
                    (u + au.dirac) -
                        k1 * v;
            };
    };
```

2.8. OutformationEOC.lib

This module contains functions for nontrivial transformation and generation of audio streams. Here, we can find standard techniques such as pitch shifting using delay-modulated with overlap-add-to-one delay lines, or time stretching following the same principles. Alternatively, some original functions for audio processing include a zero-crossing-synchronous granulator and recursive nonlinear transfer function. The first algorithm generates continuous streams of non-overlapping grains based on first and second derivative analysis to minimise distortion introduced by grain concatenation. The second one is a recursive design for a waveshaper in which the output determines the transfer function which determines the output, circularly. The module also contains standard granular processing techniques as well as less common ones using

[5] https://vicanek.de/articles/QuadOsc.pdf

non-homogenous overlap-add-to-one strategies. Furthermore, the module includes iteration systems such as elementary cellular automata, that can be used both for formal and timbral processing, as well as chaotic systems such as the Lorenz system, than can be used as a nontrivial oscillator, provided that the output is scaled accordingly.

The author is planning on extending this module with a set of complex oscillators based on self-oscillating and self-modulating nonlinear feedback delay network (FDN). The FDN may follow a similar approach as that found in standard artificial reverberators, that is delay lengths chosen as prime or coprime numbers [33], although particular focus will be given to the exploration of delay lengths based on some of the number sequences in *mathsEOC.lib*. Furthermore, each node in the network would include a more or less articulated nonlinear element – for example, a waveshaper or a frequency modulation unit – so as to realise the paradigm of high-level nonlinear iterated function to achieve complex oscillations.

Below we can find the code for the implementation of the Lorenz system.

```
lorenz(x0, y0, z0, a, b, r, dt) =
    iterate ~ ( _ , _ , _)
    with {
        iterate(x, y, z) =
            x1 + a * (y1 - x1) * dt,
            y1 + (r * x1 - y1 - x1 * z1) *
                dt,
            z1 + (x1 * y1 - b * z1) * dt
            with {
                x1 = x + x0 - x0';
                y1 = y + y0 - y0';
                z1 = z + z0 - z0';
            };
    };
```

2.9. StabilityEOC

Finally, the last library module includes a set of functions for stability processing that can be deployed in self-oscillating systems or any other system requiring control over the boundaries of amplitude values. Depending on the specific applications, it is possible to use different designs ranging from bounded saturators, lookahead limiters, and adaptive self-regulating dynamic processing.

The bounded saturators are mostly based on [28]. These saturators provide different spectral responses and have adjustable upper and lower bounds. Particularly, this approach for stability processing in feedback systems can be particularly powerful when combined with filters. Limiting the amplitude of a signal through saturators results in added harmonics due to waveshaping effects. The iterative process of increases the spectral content can be particularly effective to generate dynamical and rich spectra. On the other hand, if these are combined with filters, two competing positive and negative feedback loops can be established to enhance the complexity of the outcome [23]. A creative use of bounded saturators in FDN systems can be found in [24] in the work *Phase Transitions* (2019), which is also entirely implemented in FAUST[6].

The lookahead design is inspired to a post on limiters by IOhannes Zmölnig[7], although the delay, time constants, and smoothing circuits are different. The general idea for lookahead limiting is to analyse the amplitude profile of the input, delay the input signal itself, and perform an amplitude correction through a curve that is slow enough as not to introduce considerable distortion. The amplitude profile for the system in this module is calculated using a cascaded circuit with attack, hold, and release times. This circuit has the advantage of having a smoother curve in the transition between attack, hold, and release phases, although the effective attack, hold, and release times are interrelated as this design is based on cascaded filters. Particularly, we have a peak holder feeding into a peak envelope feeding into a one-pole lowpass. Considering the $2\pi\tau$ time constant for one-pole lowpass where most of the final value is reached after the attack time, we will only have a hold segment in the resulting envelope function if the hold time is greater than the attack time, and the hold segment will be approximately the hold time minus the attack time. Otherwise, only attack and release segments will result from this cascaded design. Furthermore, the attack time should still be much smaller than the release time to minimally affect the decay. Despite these flaws, this approach results in a cleaner output due to smoother transitions. Alternatively, an envelope function with switching attack, hold, and release sections can be found in *mathsEOC.lib*. With regard to the $2\pi tau$ time constant used for this design, the lookahead delay is chosen as the same as the attack time so that the peaks in the amplitude profile are synchronised with the peaks in the input signal, resulting in a limiter with *brickwall* characteristics.

Finally, another approach to handle stability in self-oscillating systems is to deploy compressor-like units with adaptive designs. Similarly to lookahead limiting, this approach analyses the input signal to extract an amplitude profile and, subsequently, perform a correction to keep amplitude levels within desired ranges. Unlike the lookahead limiting function above where the input is unaffected if below the limiting threshold, this design performs a continuous adjustment increasing or reducing the input gain as an inverse relationship to the envelope curve. Essentially, this design consists of a negative feedback mechanism that keeps the amplitude levels within working bounds. The envelope profile can be calculated, for example, using RMS or peak envelope processors, while the relationship between gain and envelope profile can be nonlinear when using powers to shape the envelope. This technique is particularly interesting as it allows for nontrivial formal developments when using large analysis windows.

3. CONCLUSIONS

We have provided a brief historical background on cybernetics and complexity and we have provided a context for the interactions that these disciplines had with the development of new music. We have also provided musical examples that show the importance of recursive designs, adaptation, and complexity for the investigation of new music. Then, we have introduced the *Edge of Chaos* library, a large set of Faust functions that provide the basic building blocks for users who are interested in developing music CASes. The library is fully-documented and it includes both established and original algorithms for standard and nonconventional sound and music computing. The library consists of eight dedicated modules that can be combined following an agent-based modelling and self-modulating paradigm to realise autonomous or semi-autonomous music systems.

The *Edge of Chaos* library will be maintained and expanded as much as possible. The future developments for the software library

[6] https://rottingsounds.org/threads/auditorium/phase-transitions/

[7] http://iem.at/~zmoelnig/publications/limiter/

will aim at turning it into a CASes generator. Within the FAUST environment, all of the elements in the library would be interfaced through a single command-line operation that can describe the fundamental characteristics of a complex network. Following Kauffman's approach with random Boolean networks [34, 35], CASes can be generated using a single seed that feeds a set of iterated nonlinear functions to output uncorrelated pseudo-random values. These values would then be used to select the nodes in the information processing and sound processing networks of each agent, the relationships between agents, the network topologies, and more. The resulting CASes would then be *black boxes* whose structures and organisations are randomly determined, although the systems would be entirely deterministic in their behaviours. Furthermore, the DSP network that is generated randomly can still be explored by the user using Faust's block diagram generation to gain insights over particular DSP connections that appear to be repeatedly convincing for the creative practice.

4. APPENDIX

4.1. FiltersEOC.lib list of functions

```
analytic,
apbi,
apblti,
biquad,
bpbi,
bp2blti,
bsbi,
cic,
hpbi,
hpblti,
hp1p,
hp1pint,
hp1praw,
hp1p1z,
hp1p1zraw,
int_clip,
int_eu_b,
int_eu_clip,
int_eu_f,
int_trap,
int_trap_clip,
integrator,
leaky,
lpbi,
lpblti,
lp1p,
lp1pint,
lp1praw,
lp1p1z,
lp1p1zraw,
sah_inv,
slew_limiter,
svfblti,
svf2blti,
xover_butt,
xover1p1z,
xover1p1z_ada,
xover1praw,
xover1p1zraw,
xover2p2z.
```

4.2. MathsEOC.lib list of functions

```
aad,
asinc_bi,
asinc_uni,
avg_ari,
avg_ari_w,
avg_geo,
avg_geo_w,
avg_harm,
avg_harm_w,
avg_pow,
avg_quad,
bip,
complement,
dec2bin,
delta,
delta2,
diff,
div,
factorial,
hp_and,
hp_imp,
hp_nand,
hp_nimp,
hp_nor,
hp_nxr,
hp_or,
hp_xor,
interpolate_mn,
if,
ifN,
inv,
line,
line_reset,
map_lin,
map_par,
map_pcw,
map_log,
map_pow,
matrix,
maxN,
minN,
ny,
ph,
primes,
prime_base_pow,
relay_hysteron,
round_pow2,
rt9,
rt19,
rt55,
rt60,
seq_catalan,
seq_fibonacci,
seq_hexagonal,
seq_lazy_caterer,
seq_magic_number,
seq_pentagonal,
```

```
seq_square,
seq_triangular
sd,
sp,
topologies,
top_anti_diag,
top_diagonal,
top_diag_shift,
top_full,
top_tri_low,
top_tri_up,
twopi
uni,
unit_log,
var,
w,
window_hann,
window_sine,
wrap,
y_and,
y_or,
zeropad_up,
zeropad_down,
z_and,
z_imp,
z_nand,
z_nimp,
z_nor,
z_nxr,
z_or,
z_xor.
```

4.3. OutformationEOC.lib list of functions

```
eca,
grains_dl_nhw,
grains_dl_zc,
grains_zc,
lorenz,
pitch_shift,
pole_mod,
rev_fdn_smo,
rev_fdn_pol,
sampler,
ssbm,
time_stretch,
tvtf.
```

5. REFERENCES

[1] Edgar Morin, *La nature de la nature*, vol. 123, Seuil Paris, 1977.

[2] Venkateswararao Vemuri, *Modeling of complex systems: an introduction*, Academic Press, 2014.

[3] Ilya Prigogine and Grégoire Nicolis, "Self-organisation in nonequilibrium systems: towards a dynamics of complexity," in *Bifurcation analysis*, pp. 3–12. Springer, 1985.

[4] Humberto R Maturana and Francisco J Varela, "Autopoiesis: The organization of the living," *Autopoiesis and cognition: The realization of the living*, vol. 42, pp. 59–138, 1980.

[5] Stuart A Kauffman, "Emergent properties in random complex automata," *Physica D: Nonlinear Phenomena*, vol. 10, no. 1-2, pp. 145–156, 1984.

[6] Christopher G Langton, "Studying artificial life with cellular automata," *Physica D: Nonlinear Phenomena*, vol. 22, no. 1-3, pp. 120–149, 1986.

[7] Frank Rosenblatt, "The perceptron: a probabilistic model for information storage and organization in the brain.," *Psychological review*, vol. 65, no. 6, pp. 386, 1958.

[8] H Holland John, "Adaptation in natural and artificial systems," *Ann Arbor: University of Michigan Press*, 1975.

[9] Uri Wilensky and William Rand, *An introduction to agent-based modeling: modeling natural, social, and engineered complex systems with NetLogo*, MIT Press, 2015.

[10] J Doyne Farmer, "Market force, ecology and evolution," *Industrial and Corporate Change*, vol. 11, no. 5, pp. 895–953, 2002.

[11] J Doyne Farmer and François Lafond, "How predictable is technological progress?," *Research Policy*, vol. 45, no. 3, pp. 647–665, 2016.

[12] David C Krakauer, "Darwinian demons, evolutionary complexity, and information maximization," *Chaos: An Interdisciplinary Journal of Nonlinear Science*, vol. 21, no. 3, pp. 037110, 2011.

[13] Marco Lagi, Karla Z Bertrand, and Yaneer Bar-Yam, "The food crises and political instability in north africa and the middle east," *Available at SSRN 1910031*, 2011.

[14] Stephanie Forrest, Steven A Hofmeyr, and Anil Somayaji, "Computer immunology," *Communications of the ACM*, vol. 40, no. 10, pp. 88–97, 1997.

[15] Luc Steels, "The synthetic modeling of language origins," *Evolution of communication*, vol. 1, no. 1, pp. 1–34, 1997.

[16] Luc Steels, "Evolving grounded communication for robots," *Trends in cognitive sciences*, vol. 7, no. 7, pp. 308–312, 2003.

[17] Alice Eldridge, *Collaborating with the behaving machine: simple adaptive dynamical systems for generative and interactive music*, Ph.D. thesis, 2007.

[18] Alice Eldridge, Alan Dorin, and Jon McCormack, "Manipulating artificial ecosystems," in *Workshops on Applications of Evolutionary Computation*. Springer, 2008, pp. 392–401.

[19] Alice Eldridge and Chris Kiefer, "The self-resonating feedback cello: interfacing gestural and generative processes in improvised performance," *Proceedings of New Interfaces for Music Expression 2017*, vol. 2017, pp. 25–29, 2017.

[20] Agostino Di Scipio, "'sound is the interface': from interactive to ecosystemic signal processing," *Organised Sound*, vol. 8, no. 3, pp. 269–277, 2003.

[21] Dario Sanfilippo and Andrea Valle, "Feedback systems: An analytical framework," *Computer Music Journal*, vol. 37, no. 2, pp. 12–27, 2013.

[22] Dario Sanfilippo, "Time-variant infrastructures and dynamical adaptivity for higher degrees of complexity in autonomous music feedback systems: the Order from Noise (2017) project," *Musica/Tecnologia*, vol. 12, no. 1, pp. 119–129, 2018.

[23] Dario Sanfilippo, "Complex adaptation in audio feedback networks for the synthesis of music and sounds," *Computer Music Journal (pending peer-review)*, 2020.

[24] Dario Sanfilippo, *Complex musical behaviours via time-variant audio feedback networks and distributed adaptation: a study of autopoietic infrastructures for real-time performance systems*, Ph.D. thesis, University of Edinburgh, 2020.

[25] Dario Sanfilippo and Agostino Di Scipio, "Environment-mediated coupling of autonomous sound-generating systems in live performance: An overview of the Machine Milieu project," in *Proceedings of the 14th Sound and Music Computing Conference, Espoo, Finland*, 2017, pp. 5–8.

[26] Agostino Di Scipio and Dario Sanfilippo, "Defining ecosystemic agency in live performance. The Machine Milieu project as practice-based research," *Array Journal*, vol. 12, pp. 28–43, 2019.

[27] John H Holland, *Complexity: A very short introduction*, OUP Oxford, 2014.

[28] Vadim Zavalishin, "The art of VA filter design," *Native Instruments*, 2012.

[29] William C Pirkle, *Designing audio effect plug-ins in C++ with digital audio signal processing theory*, Taylor & Francis, 2013.

[30] Udo Zölzer, *Digital audio signal processing*, John Wiley & Sons, 2008.

[31] Dario Sanfilippo, "Time-domain algorithms for low-level and high-level information processing," *Computer Music Journal (pending peer-review)*, 2020.

[32] Timothy Stilson and Julius O Smith III, "Alias-free digital synthesis of classic analog waveforms," in *ICMC*, 1996.

[33] Davide Rocchesso and Julius O Smith, "Circulant and elliptic feedback delay networks for artificial reverberation," *IEEE Transactions on Speech and Audio Processing*, vol. 5, no. 1, pp. 51–63, 1997.

[34] Stuart A Kauffman, "Metabolic stability and epigenesis in randomly constructed genetic nets," *Journal of theoretical biology*, vol. 22, no. 3, pp. 437–467, 1969.

[35] Stuart A Kauffman, *The origins of order: Self-organization and selection in evolution*, Oxford University Press, USA, 1993.