Navigating Density Topologies: Multi-Scale Markov Chains for Real-Time Rhythmic Co-Improvisation

Matteo Lorito
Conservatorio Statale di Musica "Giuseppe Verdi" di Torino
Via Giuseppe Mazzini, 11, 10123 Torino, Italy
matteo.lorito@conservatoriotorino.eu

Abstract

We present an approach to real-time rhythmic co-improvisation that treats stochastic processes as tools for navigation rather than imitation. Rather than learning to reproduce a performer's sequences, the system constructs a geometric space of rhythmic density and moves through it, producing behaviour structurally informed by the performer without copying their trajectories. The method uses a hierarchy of Markov chains at three temporal scales (30s, 10s, 4s): at each scale, two-dimensional Gaussian convolution transforms observed transition counts into a navigable space where transition probability depends on geometric proximity rather than temporal ordering, and higher scales condition lower ones through Gaussian weighting on density targets. An independent second-order chain governs rhythmic articulation. Learning exclusively from the ongoing performance and requiring no pre-trained corpus, the system drives an external timbral sample navigator in electroacoustic improvisation.

1 INTRODUCTION

Algorithmic co-improvisation systems face a fundamental design question: what should the machine learn from the performer, and what should it do with that knowledge? Systems in the Markov tradition (from Pachet's Continuator (Pachet, 2002) to the Factor Oracle lineage of OMax (Assayag and Dubnov, 2004), Somax2 (Lévy, 2012), and Dicy2 (Nika et al., 2017)) typically learn sequential patterns from the performer's input and generate stylistically consistent continuations. The machine absorbs the performer's vocabulary and grammar, then speaks back in the same language.

This approach is powerful for idiomatic music, where shared stylistic conventions provide a transferable grammar. In electroacoustic practice, however, traditions define orientations of aesthetic inquiry rather than shared vocabularies, and each performance tends to establish its own internal grammar. A system trained on a corpus of diverse improvisations faces a convergence problem: the learned model smooths individual character into aggregate statistical behaviour. This suggests a different starting point. Rather than using stochastic processes for their capacity to learn sequential patterns from data, our system employs them for their intrinsic mathematical properties: in particular, the Markov chain's ability to navigate finite state spaces with explicit, interpretable transition probabilities. This orientation could be described as a structuralist approach to co-improvisation: the machine's musical identity emerges from the geometry of its generative mechanisms, not from an absorbed style.

We propose a system in which learning serves not to acquire the performer's sequential grammar (what follows what) but to construct a navigable density space over the performer's behaviour: a characterisation of what is near what in a space of temporal density classes. The machine navigates this space independently, producing rhythmic propositions that are geometrically coherent with the performer's activity but not imitative of their specific trajectories.

The system operates across three nested temporal scales (30 seconds, 10 seconds, and 4 seconds), each governed by its own first-order Markov chain over discrete density classes. Transition matrices at each level are built from observed state sequences, then smoothed by two-dimensional Gaussian convolution, the operation that transforms sequential co-occurrence counts into the navigable density space. Higher levels bias lower levels through continuous Gaussian weighting on event-count targets, producing a top-down planning cascade from macro-level density arcs to micro-level onset placement. An independent second-order Markov chain governs articulation, and an autoregressive process shapes the temporal distribution of onsets within planning windows.

The system's output (onset times and duration proportions) drives an external timbral sample navigator. Regularity emerges as a continuous parameter, and all generative parameters are interpretable and accessible to the performer.

This paper presents the system's architecture, its design rationale, and reflections from performance practice, situating the work within Markov-based interactive systems and deep learning approaches to rhythm. We argue that density-space navigation occupies a distinct and productive position: generative without being imitative, responsive without being reactive, and autonomous without being opaque.

2 RELATED WORK

Interactive music systems that learn from and respond to live performers have a rich history. We focus here on three lineages most relevant to our work (Markov-based interactive systems, sequence recombination approaches, and deep learning models for rhythm) before discussing the closest prior system to ours.

2.1 Markov-based interactive systems

Pachet's Continuator (Pachet, 2002) established the paradigm of real-time Markov co-improvisation: a variable-order Markov chain built from MIDI note sequences, ca-

pable of generating stylistically consistent continuations. The system's hierarchical reduction functions handle imprecision across multiple representations (pitch, pitch-region, pitch-and-duration), and its real-time operation makes it immediately responsive to a performer's input. Hawryshkewich et al.'s Beatback (Hawryshkewich et al., 2010) applies a similar approach to percussion, learning drum patterns via Markov models in call-response and accompaniment modes; Linskens (2014) apply Markov chains of varying order to melodic improvisation. These systems learn *sequential co-occurrence* (the probability that event B follows event A) and generate by sampling from these learned distributions. Our system shares the real-time Markov framework but departs in two respects: it operates on density classes rather than note-level events, and its Gaussian convolution turns sequential co-occurrence into spatial proximity rather than a reproducible grammar.

2.2 Sequence recombination systems

The Factor Oracle lineage (OMax (Assayag and Dubnov, 2004), Somax2 (Lévy, 2012), ImproteK/Dicy2 (Nika et al., 2017)) captures sequential structure via suffix-based data structures and generates by recombining stored audio or symbolic segments along oracle paths. These systems produce compelling results in idiomatic contexts. However, they do not independently model rhythmic structure: the temporal properties of the output are inherited from the recombined segments rather than generated by a dedicated rhythmic process. Dicy2's temporal scenario mechanism provides some rhythmic guidance through external score alignment, but the rhythmic content itself remains a property of the corpus rather than a generative dimension. The Variable Markov Oracle (Wang et al., 2016) extends this lineage toward guided improvisation, constraining oracle navigation with external query targets, yet rhythm remains inherited from the corpus. Our system addresses rhythm as an independent generative layer: it produces onset times and duration proportions from dedicated density and activity models, decoupled from the timbral content of the samples that are triggered.

2.3 Deep learning approaches to rhythm

Recurrent neural networks have been applied to rhythm generation, from Eck and Schmidhuber's (2002) early LSTM experiments with blues structure to Gillick et al.'s (2019) GrooVAE, a variational autoencoder that learns expressive drumming from the Groove MIDI Dataset. Roberts et al.'s (2018) MusicVAE demonstrates hierarchical latent spaces for capturing long-term musical structure. These systems achieve impressive results within their training domains, and conditioning mechanisms provide genuine adaptability at inference time. However, they rely on structural assumptions difficult to reconcile with our context: a fixed quantisation grid (typically 16th-note) structuring both training and output, categorical instrument roles, and a corpus that defines the stylistic boundaries of generation. Our system imposes no fixed quantisation grid: regularity emerges as a continuous parameter derived from the performer's temporal behaviour, allowing the output to range from quasi-metric to fully irregular within a single performance. It generates abstract onset times rather than instrument-specific patterns.

2.4 Closest prior work

Frieler and Zaddach (2022) present the system most structurally similar to ours: two independent Markov chains with higher-level conditioning of lower levels, applied to jazz improvisation. Our approach differs in three respects: continuous Gaussian kernel conditioning rather than categorical matrix selection, temporal window scales rather than musicological units (bar, phrase, section), and a design for contexts where the idiom is established locally within each performance rather than drawn from an established tradition. More fundamentally, the difference from prior hierarchical stochastic systems is orientational rather than mechanical: techniques such as smoothing and inter-level conditioning are shared, but here they serve to navigate a geometric space of density relationships rather than to reproduce sequential likelihood.

3 THE CO-IMPROVISATION SYSTEM

The rhythmic generator presented in this paper is one component of a larger system for electroacoustic co-improvisation. The complete environment comprises an audio analysis pipeline, the rhythmic generator described here, and a timbral sample navigator currently under development. Research in auditory scene analysis (Bregman, 1990) and timbre perception (McAdams, 2019) demonstrates that listeners organise sound into hierarchical structures: micro-variations of spectrum, amplitude, and envelope aggregate into recognisable objects and gestures, which in turn compose higher-level textures and formal regions. This perceptual hierarchy operates across both timbral and temporal dimensions simultaneously but through partially independent mechanisms: spectral identity and temporal density can be tracked, varied, and perceived separately. Spectromorphology (Smalley, 1997) and Roads's multiscale composition (Roads, 2015) offer related accounts of form as emerging from multiple organisational planes.

This perceptual independence motivates the architectural separation between our timbral and rhythmic systems. The timbral navigator governs *what* the machine sounds like through corpus-based concatenative synthesis with hierarchical navigation. The rhythmic generator governs *when* and *how densely* the machine plays: it produces onset times and sample duration fractions from dedicated density and activity models. The two systems share a common mathematical framework (hierarchical Markov chains, Gaussian kernels, feature-space conditioning) but operate on independent state spaces with independent learning processes.

The coupling between them is minimal and unidirectional: the rhythmic generator sends trigger messages (onset time, duration fraction) to the timbral navigator, which selects and plays a sample at each trigger. Neither system has access to the other's internal state: the rhythmic system is a scheduling layer, not a sound source.

A real-time audio analysis pipeline, built on FluCoMa descriptors (Tremblay et al., 2022), connects the performer to both systems: timbral features (brightness, spectral tilt, spectral flux) feed the timbral navigator's conditioning, while temporal features (event density, temporal centroid, sound/silence ratio, inter-onset interval statistics) feed the rhythmic generator. The performer thus influences both systems through a single acoustic signal, each process responding to different aspects of it.

4 DESIGN RATIONALE

The system's architecture reflects a series of interconnected design choices. This section presents the theoretical and aesthetic motivations; the technical realisation follows in Section 5.

Modelling rhythmic behaviour through event density. In electroacoustic music, rhythmic organisation is often experienced not as metric pattern but as varying density of events across time: passages are perceived as sparse, moderate, or dense, and formal structure emerges from the evolution of this density over multiple temporal scales (Roads, 2015; Smalley, 1997). This approach has precedent in Xenakis's treatment of rhythmic processes as stochastic density functions, where Markov chains govern the distribution of events across time rather than determining individual note sequences (Xenakis, 1992). Our system takes event density as its primary compositional parameter, modelling rhythmic behaviour as navigation through a space of density states. The system comprises three parallel generative processes: (i) a density hierarchy of three nested Markov processes determining *how many* events occur at each temporal scale; (ii) an activity process determining *how long* each triggered sample plays (its duration fraction, from a brief impulse to the sample's full length); and (iii) an autoregressive centroid process shaping *where within each planning window* events concentrate. A fourth element, the regularity parameter, is derived from the performer's inter-onset interval statistics and controls the degree of temporal evenness at the onset placement stage. The following paragraphs motivate the design choices underlying each.

Stochastic processes as structural material. Each stochastic mechanism is chosen for the structural behaviour it produces: finite state spaces (Markov chain), navigable neighbourhood structure (Gaussian kernel), mean-reverting trajectories (autoregressive process).

Discrete density classes. The system quantises continuous density values into discrete classes: 19 at the 4-second scale, 10 at 10 seconds, 6 at 30 seconds. Musically, event density in electroacoustic improvisation is experienced as qualitative regions ("sparse," "moderate," "dense") rather than as precise numerical values. Discrete classes model this categorical perception explicitly: a performer does not distinguish between 3.1 and 3.2 events per second, but does distinguish between "a few isolated events" and "a continuous stream." Continuous mean-reverting processes model gradual density drift well, but they do not naturally capture abrupt density changes: at each step, the most likely next value is always near the current one. Discrete states allow the transition matrix to encode both gradual and sudden transitions within the same row: for instance, a performer's tendency to either maintain the current density or jump sharply to a contrasting one. The class boundaries were determined from preliminary analysis of performer data, positioned to concentrate resolution where density values cluster most frequently (1–6 events per second), with coarser resolution at extremes. Systematic optimisation of these boundaries remains future work. Additionally, discrete states enable the two-dimensional Gaussian convolution that transforms the transition count matrix into a navigable density space, which we call the *density topology*: a neighbourhood structure where proximity indicates similarity of density behaviour rather than sequential relationship (cf. the timbre-space tradition: Wessel, 1979; Bell, 2023).

Multi-scale hierarchy. The three-level hierarchy (30s, 10s, 4s) reflects the multi-scale perception discussed in Section 3, corresponding approximately to Roads's formal, phrase, and gestural temporal scales (Roads, 2015): the 30-second level captures broad formal arcs, the 10-second level captures phrase-scale gestures, and the 4-second level shapes moment-to-moment onset placement. These durations also correspond to ranges identified in music cognition: the 4-second window approximates the perceptual present within which successive events group into a single experienced gesture (Pöppel, 1997; Snyder, 2000), the 10-second window sits within phrase-level short-term memory, and the 30-second window addresses formal relationships engaging long-term memory (Snyder, 2000). The exact durations are practical settings within these perceptual ranges, configurable in implementation; their systematic evaluation remains future work. The mechanism by which higher levels bias lower levels is detailed in Section 5.

The autonomy spectrum. The system's three processes operate at different levels of autonomy relative to the performer. Density and activity are the most autonomous: their Markov processes navigate their respective state spaces independently, informed by the performer's behaviour through the learned transition matrices but not tracking moment-to-moment changes. The temporal centroid occupies a middle ground: an autoregressive process generates smooth gestural trajectories that drift around the performer's observed central tendency, producing independent micro-level phrasing while maintaining a general alignment with the performer's temporal distribution. Regularity is the most performer-dependent: it is derived from the coefficient of variation of the performer's inter-onset intervals, anchoring the machine's temporal evenness to the performer's rhythmic character. An exploration factor (detailed in Section 5.2) preserves the capacity for unexpected departures, preventing the machine from locking into the performer's exact pattern. This gradient, from autonomous navigation to performer-tracking, distributes creative agency across parameters: the machine proposes its own density and activity trajectories while maintaining rhythmic compatibility with the performer's temporal behaviour.

5 SYSTEM ARCHITECTURE

This section describes the technical realisation of the system. Learning and generation operate on independent clocks: learning advances each time the analysis pipeline delivers new data (every second); generation advances on its own internal metro, whose tick length can be configured independently. The two can run separately: observing without generating, or generating from a previously learned matrix.

5.1 Overview

The system produces two output streams. The primary stream assembles onset times and duration proportions every 12 seconds, drawing on three consecutive 4-second planning chunks. A secondary stream generates 0–4 onsets per 30-second cycle for triggering longer samples at a

coarser temporal grain. Both streams are sent to the timbral sample navigator.

Figure 1 shows the complete data flow. Three parallel processes (density, activity, and temporal centroid) each maintains its own state and updates independently. Their outputs are combined at the scheduler to produce the onset stream. Figure 2 shows how these processes unfold over time. On the learning side, the system continuously classifies incoming density values and periodically rebuilds each level's transition matrix from recent observations (every 60, 100, and 120 seconds respectively), each time smoothing by Gaussian convolution. On the generation side, the density hierarchy advances every 30, 10, and 4 seconds respectively; the activity process (which determines what fraction of each triggered sample's duration is played, from a brief impulse to the full length) transitions at irregular intervals governed by persistence durations; and the centroid process updates at each level's generation tick.

5.2 Density hierarchy

Learning. Every second, the system counts the events detected over the preceding 4, 10, and 30 seconds respectively (sliding windows), and classifies each count into a discrete density class. Each of the three levels maintains its own transition matrix, rebuilt independently on its own schedule. The class boundaries are asymmetrically spaced (19 classes at the 4-second scale, 10 at 10 seconds, 6 at 30 seconds), following the rationale of Section 4.

The classified values accumulate into a state history. Periodically, the transition count matrix is rebuilt from recent observations (Figure 2). A small constant (0.1) is added to all cells before counting, ensuring that no transition has zero probability even if it was never observed. The count matrix is then convolved with a two-dimensional Gaussian kernel (9×9 at the 4-second level, 5×5 at 10 seconds, 3×3 at 30 seconds). This convolution spreads each observed transition to its geometric neighbours, replacing the specific record of "from class i, the performer went to class j" with a smooth distribution over the neighbourhood of

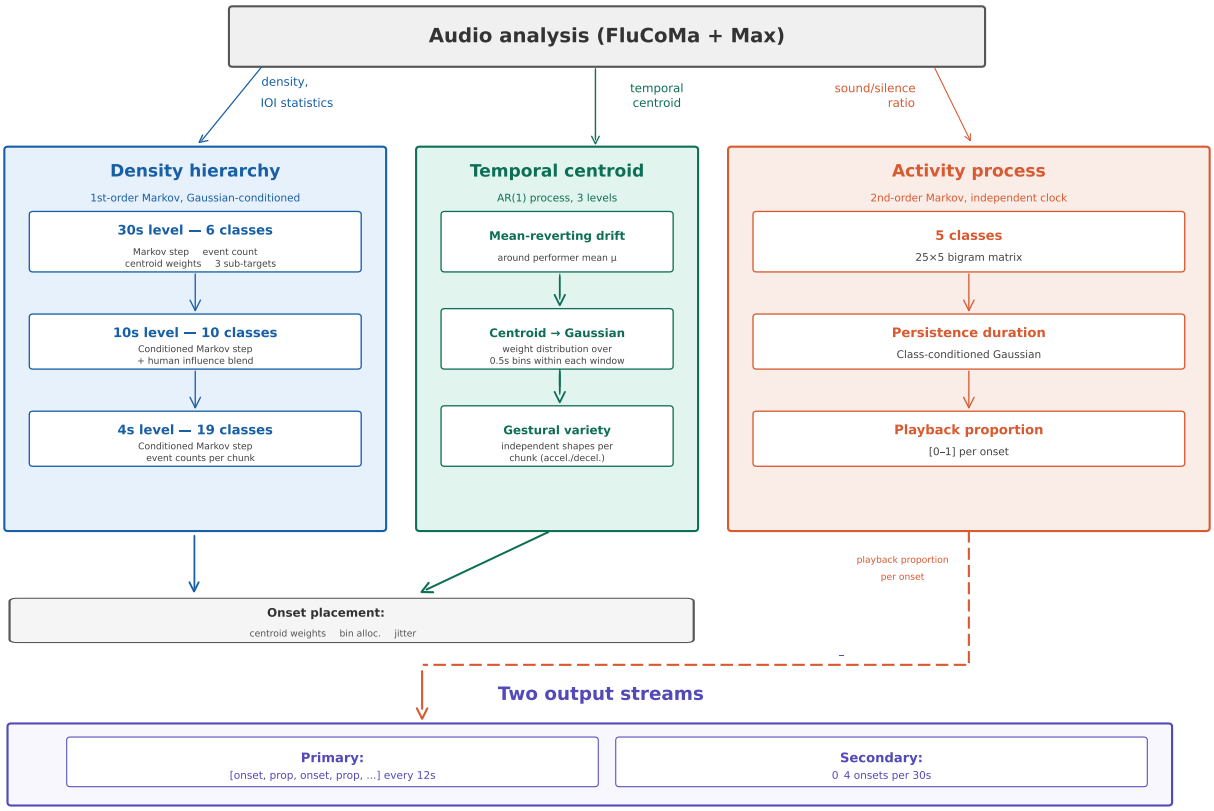

Figure 1: System architecture. Left (blue): the three-level density hierarchy with conditioned Markov navigation. Centre (green): the temporal centroid AR(1) process. Right (coral): the independent activity process. All three converge at the onset placement stage, which outputs two streams, primary (12s blocks) and secondary (30s), to the timbral sample navigator.

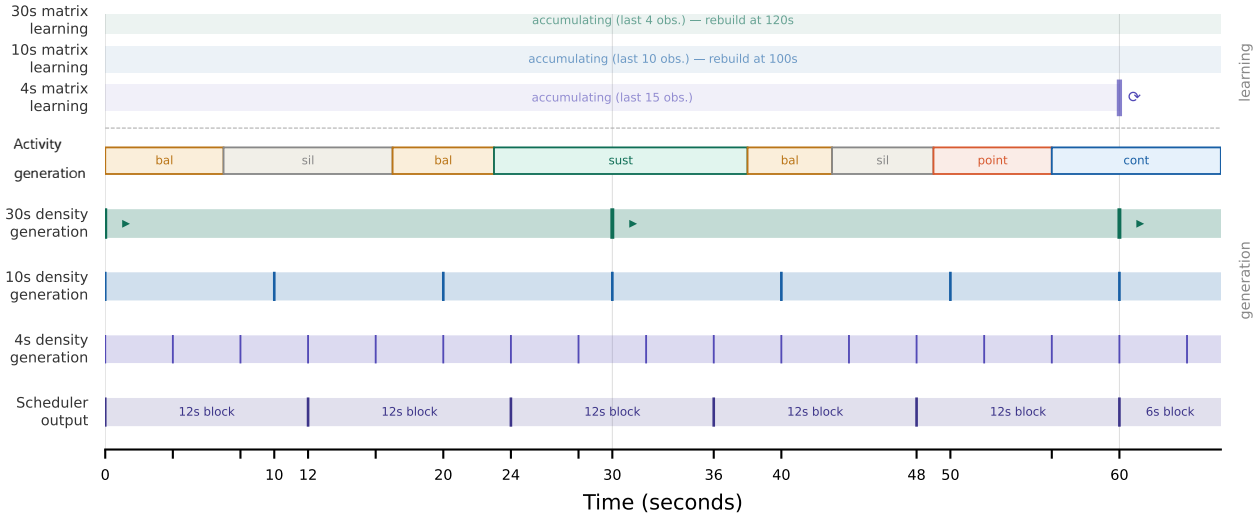

Figure 2: Temporal processes over a 60-second span. Generation processes (bottom): the density hierarchy operates on fixed windows at three scales, the activity process transitions at irregular intervals, and the scheduler assembles 12-second output blocks. Learning processes (top): matrices rebuild on slower independent schedules. The superposition of these asynchronous processes prevents any single periodicity from dominating the output.

j. The convolved matrix is row-normalised to produce a stochastic transition matrix (Figure 3).

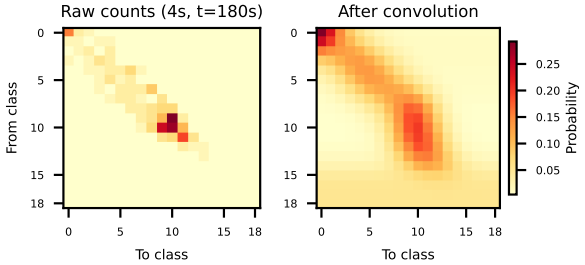

Figure 3: Transition matrix before and after Gaussian convolution (4s level, all 19 classes, from the analysed session at tick 180). The raw matrix is sparse and concentrated in the lower classes; after convolution, probability mass spreads to neighbours, producing the navigable density space.

Generation. Navigation proceeds top-down through the three levels. Figure 5 illustrates the complete process with a concrete numerical example; we walk through each step here.

Step 1 — 30-second planning. Every 30 seconds, the 30-second Markov chain advances: the current row of its transition matrix is sampled to select the next density class. Each class corresponds to a representative event count: the midpoint of the class's density range multiplied by the window duration. For example, class 3 spans approximately 0.6–0.8 events per second, yielding roughly 25 events over 30 seconds.

Step 2 — distributing across 10-second sub-windows. The 25 events are not divided equally. A centroid value (from the AR(1) process, Section 5.4) shapes the distribution: for instance, a centroid of 0.7 (late-concentrated) produces Gaussian weights that allocate more events to the third sub-window, yielding targets of $[5, 8, 12]$.

Step 3 — conditioned 10-second navigation. Each target biases the 10-second Markov chain toward density classes whose event counts are close to the target. The conditioning multiplies the Markov row probability for each candidate state j by a Gaussian kernel centred on the target event count:

$$p'_j = \frac{p_j \cdot \exp\left(-\frac{(e_j - t)^2}{2\sigma^2}\right)}{\sum_k p_k \cdot \exp\left(-\frac{(e_k - t)^2}{2\sigma^2}\right)} \tag{1}$$

where p_j is the Markov transition probability, e_j the event count of class j, t the target from the level above, and σ an adaptive parameter. Figure 4 illustrates this mechanism. The unconditioned Markov row (panel a) is bimodal, with peaks at classes 2 and 7, reflecting a performer who tends to either maintain moderate density or jump to high density. The Gaussian weights (panel b) are centred on target class 3 (corresponding to 12 events over 10 seconds). The conditioned distribution (panel c) shows the result: the near-target peak at class 2 is amplified from 0.22 to 0.47, while the distant peak at class 7 drops from 0.25 to 0.02. The conditioning steers the chain toward the target region without erasing the Markov row's structure entirely. A human influence parameter optionally blends the hierarchical target with the performer's current observed density. An exploration factor (default 3%) occasionally bypasses conditioning entirely, sampling directly from the unconditioned Markov row.

Step 4 — 4-second event counts. The same conditioning operates from the 10-second to the 4-second level. Each 10-second target is distributed across three 4-second chunks (again using centroid weights), and three conditioned 4-second Markov steps produce the final event counts: for example, $[4, 5, 3]$ for one 10-second block. To prevent discontinuities at block boundaries, a carry mechanism prepends onsets from the previous block and stores excess onsets for the next, ensuring the listener perceives a continuous stream.

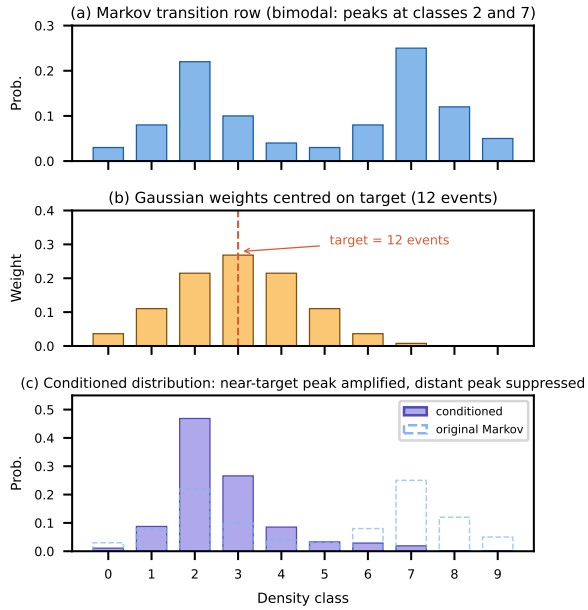

Figure 4: The conditioning mechanism (target = 12 events). (a) Bimodal Markov row with peaks at classes 2 and 7. (b) Gaussian weights centred on the target (12 events, class 3). (c) Conditioned distribution (solid) versus original (dashed): the near-target peak (class 2) is amplified while the distant peak (class 7) is suppressed.

5.3 Activity process

The activity process determines what fraction of each triggered sample's duration is played.

Learning. The sound/silence ratio from the analysis (the proportion of each window containing detected sound versus silence) is classified into five activity classes: near-silent (0–0.15), sparse (0.15–0.35), moderate (0.35–0.55), dense (0.55–0.80), and continuous (0.80–1.0). Every 100 ticks, a second-order transition count matrix is rebuilt: each row corresponds to a pair of consecutive activity classes (25 possible bigrams), each column to one of the 5 destination classes, yielding a 25×5 matrix, which is row-normalised.

Generation. Each class has a characteristic *persistence duration* (how long the system remains in that class before transitioning), drawn from a Gaussian distribution (e.g., near-silent: mean 12s, σ 3s; moderate: mean 6s, σ 1s; continuous: mean 14s, σ 3s). When the persistence counter expires, the second-order row corresponding to the current and previous classes is sampled to select the next class, and a new persistence duration is drawn. At each onset, the current activity class determines the *duration proportion*: the relative duration of the triggered sample, from a brief impulse (near 0) to full length (near 1), drawn uniformly from the class's range (e.g., sparse: 0.15–0.35).

5.4 Temporal centroid

The temporal centroid, a value between 0 and 1 describing where within a time window activity is concentrated (0 = early, 1 = late, 0.5 = evenly spread), serves two roles. First, it mediates between the discrete planning windows and the continuous perceptual experience: by shaping *how* onsets distribute within each window, the centroid provides controllable temporal gestures rather than uniform or purely random placement. Two adjacent windows with complementary centroids produce a continuous acceleration across the boundary. Second, it provides gestural variety, so the machine accelerates or decelerates rather than producing metronomically even output.

Learning. The temporal centroid is observed every second at each level. Running statistics (mean μ) are computed from the performer's centroid history.

Generation. The system generates its own centroid trajectory via an AR(1) process at each level:

$$c_{t+1} = \alpha \cdot c_t + (1 - \alpha) \cdot \mu + \sigma_c \cdot \mathcal{N}(0, 1) \qquad (2)$$

where α controls persistence, μ is the mean-reversion target learned from the performer, and σ_c controls stochastic variation. Soft reflection at $[0.02, 0.98]$ prevents boundary collapse. The human influence parameter blends the generated value with the performer's observed centroid.

5.5 Onset placement

Placing onsets within each 4-second chunk. Each chunk is divided into eight bins of 0.5 seconds. The centroid value for this chunk determines a Gaussian weight distribution over the bins: for example, centroid 0.3 centres the Gaussian at position $0.3 \times 4 = 1.2$ seconds, assigning the highest weights to the first bins so that onsets cluster near the beginning. Centroid 0.7 would concentrate them near the end instead. The event count determines how many onsets to place; these are allocated to bins proportionally to the weights. Within each bin, a regularity parameter, derived from the performer's inter-onset interval coefficient of variation, controls jitter: high regularity places onsets near the bin centre; low regularity scatters them. Continuing the example: 5 events with centroid 0.3 and regularity 0.6 might produce onset times at 0.18, 0.64, 1.22, 1.78, and 2.31 seconds.

Duration proportion. At each onset, the current activity class determines a relative duration for the triggered sample. If the activity process is currently in the "sparse" class (range 0.15–0.35), each onset receives a duration proportion drawn uniformly from that range; a value of 0.22, for example, means the triggered sample plays for 22% of its full length before being cut.

In parallel, the secondary stream generates 0–4 onsets over each 30-second span using the same centroid-weighted mechanism at a coarser grain (six 5-second bins), triggering longer samples that complement the primary stream's rapid articulations.

6 IMPLEMENTATION NOTES

The system is implemented in JavaScript within Max/MSP's v8 engine for the generative logic (Markov navigation, conditioning, onset scheduling). Audio onset detection uses FluCoMa objects (Tremblay et al., 2022); all temporal descriptors (event density, temporal centroid, activity ratio, and inter-onset interval statistics) are computed in native Max objects rather than JavaScript, to avoid the timing unreliability caused by JavaScript's low-priority

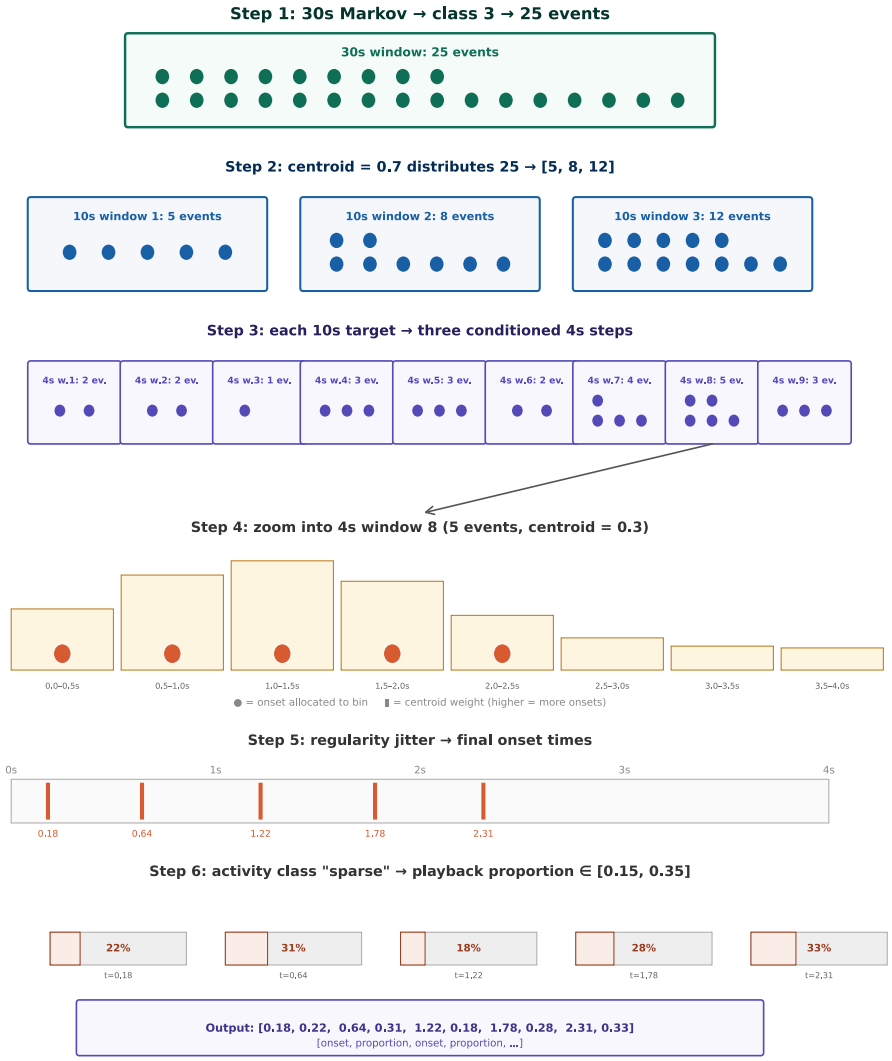

Figure 5: Worked example of one generation cycle. Step 1: the 30-second Markov chain selects a density class (25 events). Step 2: centroid weights distribute these across three 10-second sub-windows ($5 + 8 + 12$). Step 3: conditioned Markov steps select density classes per sub-window. Step 4: event counts per 4-second chunk. Step 5: centroid weights allocate events across 0.5-second bins; regularity jitter produces final onset times. Step 6: the current activity class assigns a duration proportion to each onset.

execution in the Max scheduler. The v8 engine lacks asynchronous execution, so all matrix operations (convolution, normalisation, conditioned sampling) are implemented as tight imperative loops.

The learning and generation processes share global state within a single script. To avoid misalignment between onset-list generation and parsing (which can occur when the v8 processing time exceeds the scheduler's read interval), the system's generation is by design always two generation cycles ahead of the parsing mechanism, ensuring that the scheduler always reads a fully completed onset list. Session persistence is achieved by exporting the current transition matrices and density class states to a JSON file via Max's Dict object; on the next session launch, the system resumes navigation from the saved state, providing continuity across rehearsal sessions. The rhythmic generator's computational cost is negligible: matrix sizes are at most 19×19, and the most expensive operation (2D convolution with a 9×9 kernel) completes in under 1 millisecond.

7 REFLECTIONS FROM PRACTICE

To evaluate the system's behaviour systematically, we analyse 30 sessions in which the system responds to two pre-recorded double bass improvisations by the author (approximately 4 minutes each), played back without real-time interaction.[1] This sacrifices the feedback loop that characterises live performance, but ensures reproducibility and enables direct comparison across runs and parameter settings: the same input produces different outputs only due to the stochastic generation process. For each recording, the system was run 5 times at each of three human influence settings (HI = 0.25, 0.75, and 1.0), yielding 10 sessions per condition. The machine's planned density trajectory (the Markov chain's state at each tick) is compared to the performer's observed density (from the FluCoMa analysis of the input recording).

[1] Audio examples are available at https://github.com/matteo-lorito/AIMC_26_Audio_Examples

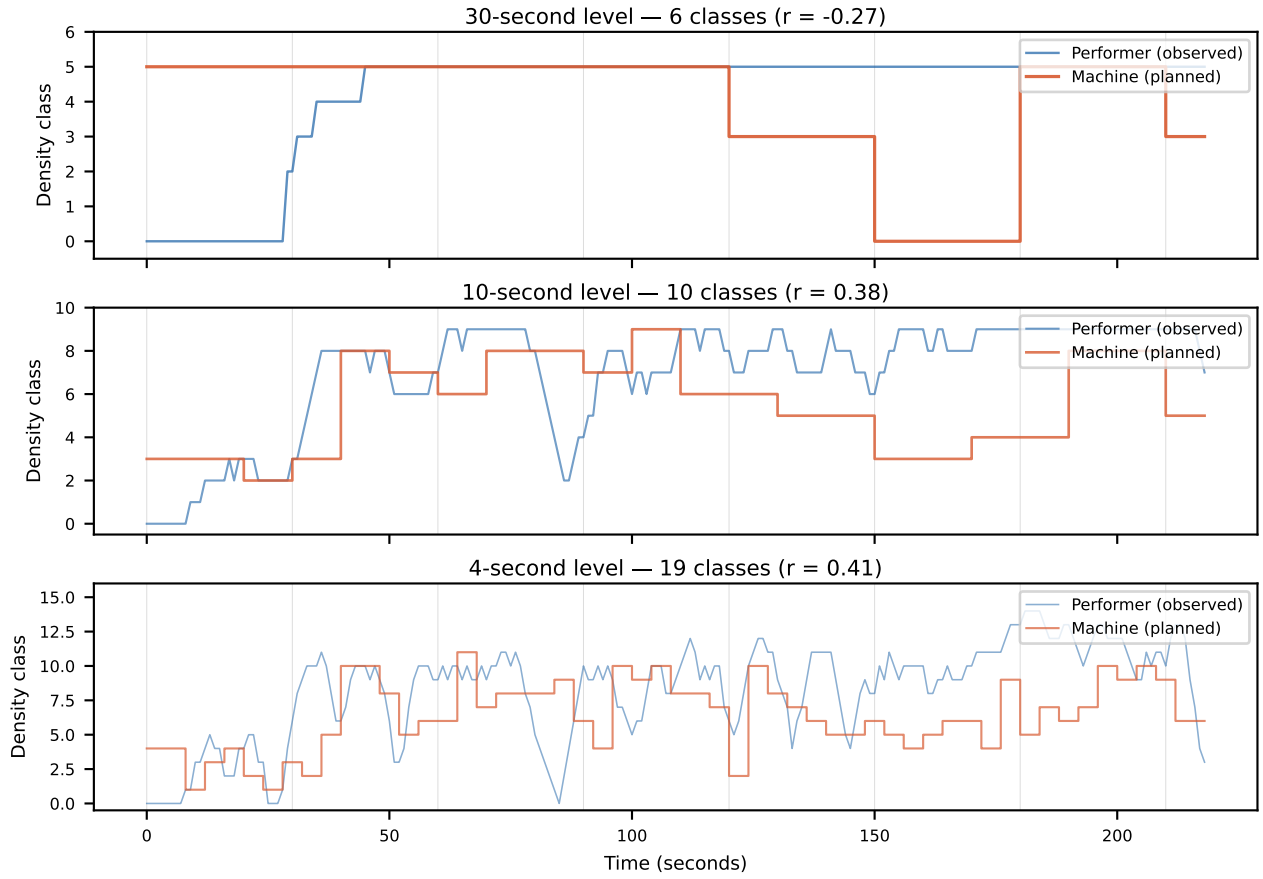

Figure 6: Density trajectories from a representative session (HI = 0.25). Blue: performer's observed density (from FluCoMa analysis). Orange: machine's planned density (Markov chain state). Vertical grey lines mark 30-second boundaries. The two traces share a density region but follow independent trajectories: the machine navigates the performer's density space without imitating the performer's specific path.

Figure 6 shows the performer's observed density and the machine's planned density at all three temporal levels for a representative session at HI = 0.25, the most autonomous setting, which best illustrates the system's independent navigation. Figure 7 summarises the correlation analysis across all 30 sessions. Table 1 reports the aggregate statistics.

Table 1: Aggregate correlation analysis (mean ± SE, $n = 10$ per condition). Statistical significance from Kruskal-Wallis test across three HI levels; Spearman ρ for monotonic trend across all 30 sessions.

Measure	HI=0.25	HI=0.75	HI=1.0	Trend
r(30s)	.35±.13	.23±.11	.29±.11	n.s.
r(10s)	.53±.04	.66±.07	.67±.05	p=.009
r(4s)	.40±.04	.62±.03	.65±.04	p=.001
dir. agree.	.39±.02	.39±.02	.40±.01	n.s.
var. ratio	1.9±.3	2.0±.2	2.6±.4	n.s.

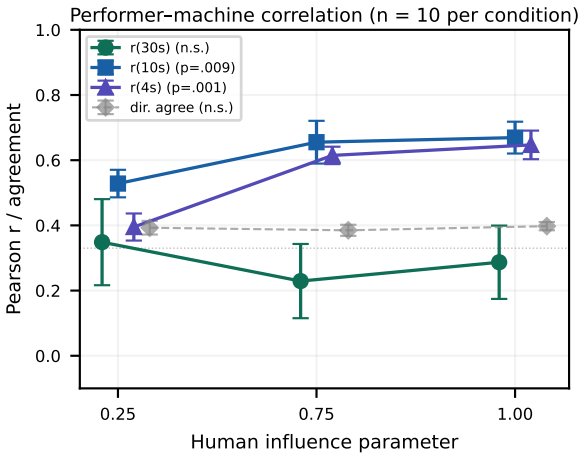

Figure 7: Performer–machine density correlation as a function of the human influence parameter (mean ± SE, $n = 10$ per condition, across two recordings). The 4-second and 10-second correlations increase significantly with HI, while direction agreement remains flat near 0.39 regardless of coupling strength.

The human influence parameter controls density coupling. The correlation between performer and machine density at the 4-second level increases with HI: from $r = 0.40$ at HI = 0.25 to $r = 0.65$ at HI = 1.0. This difference is statistically significant ($p = 0.001$, Kruskal-Wallis test), meaning there is less than a 0.1% probability that such a large difference would arise if the HI parameter had no real effect. The increase is monotonic ($p < 0.001$, Spearman rank correlation): higher HI consistently produces higher correlation. The 10-second level shows the same pattern ($p = 0.009$). The effect saturates: the dif-

ference between HI = 0.25 and HI = 0.75 is significant ($p = 0.002$), while HI = 0.75 and HI = 1.0 do not differ ($p = 0.34$). The HI parameter thus functions as a working autonomy dial.

Micro-level independence is preserved regardless of coupling. To measure whether the machine tracks the performer's moment-to-moment transitions, as opposed to its overall density region, we compute *direction agreement*: at each 4-second tick, both trajectories are classified as rising, falling, or stable, and agreement counts how often their directions match. Imitation would push agreement toward 1.0; chance agreement for independent trajectories is approximately 0.33 (three equally likely directions). The observed values range from 0.39 to 0.40 across all HI conditions, with no significant trend ($p = 0.88$), meaning the data are entirely consistent with no effect of HI on direction agreement. Even at HI = 1.0, where the machine closely follows the performer's density region, the step-by-step path remains independent. This is the core evidence for density-space navigation: the machine shares the performer's density *neighbourhood* but takes its own *trajectory*.

A purely random process would show $r \approx 0$ at all levels; the combination of regional responsiveness with path independence distinguishes density-space navigation from both imitation and randomness.

Hierarchical structure confirmed. The variance of the machine's 4-second density is consistently greater *between* 10-second blocks than *within* them (mean ratio 2.2). A ratio of 1.0 would indicate no phrase-level grouping; the observed ratio confirms that inter-level conditioning creates genuine multi-scale structure. This is visible in Figure 6: broad arcs at 30 seconds, phrase-scale variation at 10 seconds, and moment-to-moment fluctuation at 4 seconds.

Consistent across recordings. The 4-second correlation does not differ significantly between the two recordings at any HI level ($p > 0.09$): the differences between recordings are small enough to be attributable to normal run-to-run variability, confirming generalisation across input material.

Limitations. The 30-second correlation shows large variability across runs (SE $\pm$0.11–0.13) and no significant trend with HI. With only 6–8 Markov steps at this level per session, the measure is underpowered; longer sessions would likely stabilise it. The analysis is based on two recordings from a single performer and one exploration factor setting (0.03). Furthermore, the evaluation uses pre-recorded material without real-time performer–machine interaction; the system's behaviour under genuine feedback conditions remains to be tested. Complementary informal validation comes from performance practice: the complete system (rhythmic generator and timbral navigator) was presented and performed with live musicians at the ICMC 2026 workshop *Dialogues with Improvising Machines: An Embodied Cross-Testing Workshop on Musical Agents*, sustaining extended interactive improvisation. A structured evaluation is planned as the next stage of this research, including additional performers, varied exploration factors, comparison with baseline conditions (e.g., unconditioned single-level Markov or density-matched random generation), and listening studies to assess the perceptual significance of the observed differences.

8 CONCLUSION AND FUTURE WORK

We have presented a system for real-time rhythmic co-improvisation whose central premise is that stochastic processes can be used for navigation rather than imitation. Rather than reproducing a performer's sequential patterns, the machine constructs and traverses a geometric space of density relationships: the Gaussian convolution of transition count matrices, which turns sequential co-occurrence into geometric proximity, is the mechanism that realises this. The multi-scale hierarchy (30s, 10s, 4s) produces coherent trajectories from macro form to micro onset placement, completed by an independent activity process and an autoregressive centroid process. Employing stochastic processes for their mathematical properties rather than their learning capacity, and learning exclusively from the ongoing performance, the system offers an alternative to both corpus-trained deep learning models and sequence-recombination systems: a model of machine musicianship grounded in geometric navigation rather than stylistic reproduction.

Several directions for future development are planned. First, the rhythmic generator will be fully integrated with the timbral sample navigator currently under development, coupling the two output streams to their respective sound sources: the primary stream driving the hierarchical corpus navigator for rapid timbral articulations, and the secondary stream driving a RAVE-based neural synthesis model for sustained textural layers.

Second, systematic experimentation in live performance contexts will test how the system behaves under genuine interactive conditions, where the feedback loop shapes both the density space and its navigation in real time. The fixed-recording analysis isolates the system's generative behaviour, but the musical interest of the system ultimately depends on the emergent dynamics of this closed loop.

Third, corpus design for the timbral navigator (segmentation strategy, feature selection, and their aesthetic consequences) requires further investigation.

Finally, we are exploring the integration of a local language model as an asynchronous meta-controller: tracking the performance's trajectory over longer time spans, formulating strategic goals, and translating these into conditioning targets. This would add reflective, goal-oriented behaviour that memoryless Markov processes cannot provide, while preserving real-time responsiveness.

REFERENCES

Assayag, G. and Dubnov, S. (2004). Using factor oracles for machine improvisation. *Soft Computing*, 8:604–610.

Bell, J. (2023). Maps as scores: "timbre space" representations in corpus-based concatenative synthesis. In *Proceedings of the International Conference on Technologies for Music Notation and Representation – TENOR'2023*, pages 137–146, Boston, Massachusetts, USA. HAL Id: hal-04182706v2.

Bregman, A. (1990). *Auditory Scene Analysis: The Perceptual Organization of Sound*. The MIT Press.

Eck, D. and Schmidhuber, J. (2002). Finding temporal structure in music: Blues improvisation with lstm recurrent networks. In *Proceedings of the 12th IEEE Workshop on Neural Networks for Signal Processing*, pages 747–756.

Frieler, K. and Zaddach, W.-G. (2022). Evaluating an analysis-by-synthesis model for jazz improvisation. *Transactions of the International Society for Music Information Retrieval*, 5(1):20–34.

Gillick, J., Roberts, A., Engel, J., Eck, D., and Bamman, D. (2019). Learning to groove with inverse sequence transformations. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, pages 2269–2279.

Hawryshkewich, A., Pasquier, P., and Eigenfeldt, A. (2010). Beatback: A real-time interactive percussion system for rhythmic practise and exploration. In *Proceedings of the International Conference on New Interfaces for Musical Expression*, pages 100–105, Sydney, Australia.

Lévy, B. (2012). *Principles and Applications of Interactive Corpus-Based Concatenative Synthesis*. PhD thesis, Université Paris 6 – Pierre et Marie Curie.

Linskens, E. (2014). *Music Improvisation using Markov Chains*. PhD thesis, Maastricht University.

McAdams, S. (2019). Timbre as a structuring force in music. In Siedenburg, K., Saitis, C., McAdams, S., Popper, A., and Fay, R., editors, *Timbre: Acoustics, Perception. Springer Handbook of Auditory Research, vol 69*. Springer, Cham.

Nika, J., Deguernel, K., Chemla–Romeu-Santos, A., Vincent, E., and Assayag, G. (2017). Dyci2 agents: merging the "free", "reactive", and "scenario-based" music generation paradigms. In *International Computer Music Conference*, Shangai, China.

Pachet, F. (2002). The continuator: Musical interaction with style. *Journal of New Music Research*, 32(3):333–334.

Pöppel, E. (1997). A hierarchical model of temporal perception. *Trends in Cognitive Sciences*, 1(2):56–61.

Roads, C. (2015). *Composing Electronic Music: A New Aesthetic*. Oxford Academic OUP.

Roberts, A., Engel, J., Raffel, C., Hawthorne, C., and Eck, D. (2018). A hierarchical latent vector model for learning long-term structure in music generation. In *Proceedings of the 35th International Conference on Machine Learning (ICML)*, pages 4364–4373.

Smalley, D. (1997). Spectromorphology: Explaining sound-shapes. *Organised Sound*, 2(2):107–126.

Snyder, B. (2000). *Music and Memory: An Introduction*. The MIT Press.

Tremblay, P., Roma, G., and Green, O. (2022). Enabling programmatic data mining as musicking: The fluid corpus manipulation toolkit. *Computer Music Journal*, 45(2):9–23.

Wang, C.-i., Hsu, J., and Dubnov, S. (2016). Machine improvisation with variable Markov oracle: Toward guided and structured improvisation. *Computers in Entertainment*, 14(3):1–18.

Wessel, D. L. (1979). Timbre space as a musical control structure. *Computer Music Journal*, 3(2):45–52. [Online]. Available: http://www.jstor.org/stable/3680283.

Xenakis, I. (1992). *Formalized Music: Thought and Mathematics in Composition*. Pendragon Press, revised edition.