

# A Mixed Timing Method for Designing Natural Rhythms in Real-Time Media

*Sofian Audry*

## Abstract

Timing is a core expressive material of real-time media. Practitioners regularly need to generate periodic events at controlled paces such as blinking lights or rhythmic sounds. Since strict periodicity often feels artificial and machinic, a common approach consists in adding randomness to create a more organic, less predictable cadence. However, simple jittering approaches that inject noise directly into the period or frequency of a process provide limited control and can distort the expected timing. This report presents a method that overcomes these limitations by generating irregular but statistically reliable event sequences. Based on the Poisson distribution, it preserves the desired long-run event rate while allowing variability to be modulated precisely, yielding rhythms that feel natural without compromising timing accuracy. We compare several approaches and introduce a mixed Poisson model that offers a continuous, intuitive control over randomness, from stable metronome-like pacing to expressive, burst-like irregularity. Practical implementation on embedded systems with limited computational resources is also presented, demonstrating that the method is expressive and lightweight.

*Keywords:* generative media, interactive media, periodic processes, Poisson distribution, random processes, real-time media.

## Introduction

Real-time media such as interactive installations, live electronic music performances, immersive spaces, video games, and robotic art, frequently rely on temporal repetition to generate effects and shape experiences. Arduino's *Blink* sketch and Max/MSP's *metro* objects are emblematic of this reliance on pacing. From simple toy programs to complex artworks such as audiovisual performances and robotic installations, the structuring of time through periodic events constitutes a fundamental design layer. Temporal control is not merely a technical scheduling issue, but a core expressive concern that mediates perception and interaction (Emmerson 2017).

Perfect periodicity, however, often feels machinic: a perfectly timed ticking may convey an unintended feeling of automation. Human performers, whether in music or dance, typically introduce slight variability in timing, known as *microtiming*, which shapes perception and expressivity (Johnson and Gotham 2023; Gullö and Bromham 2025). In real-time media, the same principle applies: irregularity conveys a sense of aliveness.

When designing such systems, it is often more intuitive to think in terms of *period* or *frequency* rather than in terms of event probability. Artists and designers want to specify actions in temporal units that align with human experience: “once every second,” “about five times per minute,” or “at a regular beat of 120 beats per minute (BPM).” These quantities are directly relatable to everyday rhythms, whether in bodily movement, musical tempo, or cycles in nature. By contrast, reasoning in terms of an abstract probability per step feels less connected to human temporality and harder to control in practice.

Examples abound in digital creative practice: fast blinking lights, audiovisual pulses, breathing-like modulations, robotic behavioral patterns, or evocative sonic events. All rely on temporally controlled repetition, where the balance between regularity and unpredictability defines aesthetic outcomes.

Yet, it is not enough to blindly add randomness. Different contexts ask for different levels of irregularity: sometimes just a gentle fluctuation to make a repetitive process feel more natural, at other times more chaotic processes with sudden bursts and long pauses to create surprise and

contrast. The ability to adjust the level of randomness is therefore essential. What is needed are strategies that let practitioners intuitively specify “about once every  $T$  seconds”, while also shaping how tight or loose the rhythm should feel.

This paper presents different approaches to generating irregularities in periodic processes. We begin by showing two simple ways that are based on period and frequency jittering. Analyzing these processes shows important implicit limitations. This leads us to propose alternative event-generation processes based on the Poisson distribution (Poisson 1837), a probability law designed specifically to represent such random event processes.<sup>1</sup>

## Period Jittering

Let us start with the following Plaquette (Audry and Fredericks 2025) code, which creates a metronome unit that triggers once per second, toggling the built-in LED on and off each time it fires:

```
const float BASE_PERIOD = 1.0f;
Metronome metro(BASE_PERIOD);
DigitalOut led(LED_BUILTIN);
void begin() {}
void step() {
    if (metro)
        led.toggle();
}
```

The most straightforward way to introduce randomness to such a process is simply to add noise to the timer by varying the interval by a random amount at the end of each repetition. Considering a base period  $T_{base}$  and a noise level  $\varepsilon_{base} \in [0, 1)$ , the period  $T_n$  at step  $n$  is chosen by randomly sampling noise  $\varepsilon_n$  from the range  $[-\varepsilon_{base}, \varepsilon_{base}]$  and adding it to the next interval:

$$T_n = T_{base} + \varepsilon_n = (1 + \varepsilon_n)T_{base} \quad (1)$$

---

<sup>1</sup> Siméon Denis Poisson derived this distribution to model wrongful convictions in a given country by counting the number of rare events that occur during a given time interval.

Example with  $\varepsilon_{base} = 0.2$  (20%):

```
const float NOISE = 0.2f;
...
void step() {
  if (metro) {
    metro.period( (1 + randomFloat(-NOISE, +NOISE)) * BASE_PERIOD );
    led.toggle();
  }
}
```

Naive period jittering provides a simple and intuitive way to introduce noise to periodic processes with simple code. The zero-centered noise guarantees that it is unbiased in the periodic domain, meaning that the average interval is equal to the base interval  $T_{base}$ :

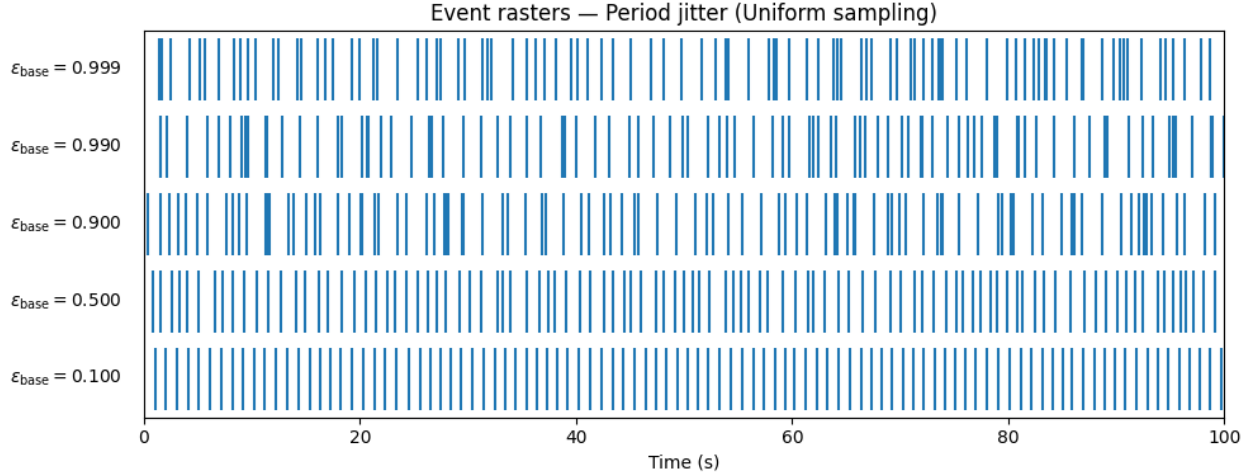
$$\mathbb{E}[T_n] = \mathbb{E}[(1 + \varepsilon_n)T_{base}] = T_{base} \quad (2)$$

Its long-run event rate  $R$ , which corresponds to *how often things happen on average if you watch for a long time*, is also unbiased:

$$R = \lim_{t \rightarrow \infty} \frac{N(t)}{t} = \frac{1}{\mathbb{E}[T_n]} = \frac{1}{T_{base}} = F_{base}. \quad (3)$$

For example, if you run a period jittering process with a  $T_{base}$  of one second, for a duration of 1000 seconds, you should expect to trigger events approximately 1000 times, and the average period will be one second.

However, these bias properties come at a cost: the process constrains variability to a narrow, regular band around a fixed interval bounded superiorly by twice the base period  $T_{base}$ . Indeed, the amount of noise  $\varepsilon_{base}$  needs to be bounded in  $[0, 1)$  so that it never exceeds the oscillation period, which could result in a negative period or frequency. The period remains bounded such that  $T_n \in (0, 2T_{base})$ . This results in relatively monotonic randomness, whereas natural processes often



**Figure 1:** Event raster for period jittering,  $T_{base} = 1$ , uniformly sampled noise at different levels  $\varepsilon_{base}$ . As is shown, higher noise increases the jitter but the period  $T_n$  is bounded between 0 and 2.

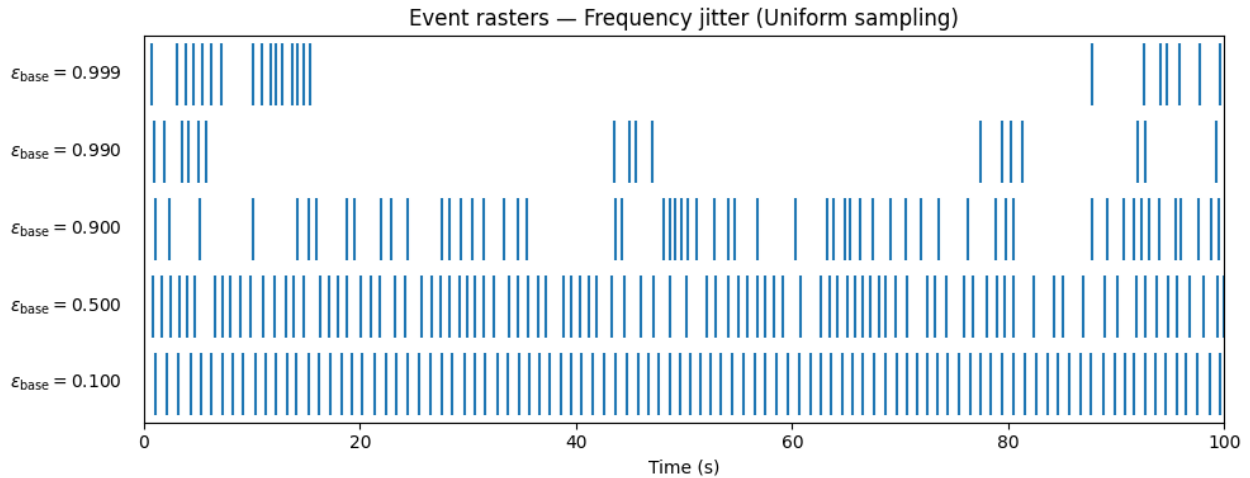
have bursts, pauses, and rare unbounded events (see Fig. 1).

## Frequency Jittering

Naive interval jittering inevitably introduces bias in the conjugate domain: period jittering preserves  $\mathbb{E}(T_n)$  but biases  $\mathbb{E}(F_n)$ , while frequency jittering preserves  $\mathbb{E}(F_n)$  but biases  $\mathbb{E}(T_n)$ . This is a direct consequence of the convex nonlinearity  $T = \frac{1}{F_n}$  and  $F = \frac{1}{T_n}$ . One way to understand this is to consider how, as  $\varepsilon_{base} \rightarrow 1$ , the boundaries of  $T_n$  tend towards  $(0, 2T_{base})$  while the corresponding boundaries of  $F_n$  approach  $(\frac{1}{2T_{base}}, \infty)$ .

This is especially problematic when doing frequency jittering, where the frequency, rather than the period, is randomly modified after each event. In the frequency domain, the update rule for frequency  $F_n$  becomes:

$$F_n = F_{base} + \varepsilon_n = (1 + \varepsilon_n)F_{base} \quad (4)$$



**Figure 2:** Event raster for frequency jittering,  $F_{base} = 1$ , uniformly sampled noise at different levels  $\epsilon_{base}$ . As is shown, although the frequency  $F_n$  is bounded between 0 and 2, the upper bound of the corresponding period  $T_n$  explodes with high levels of noise.

Example with  $\epsilon_{base} = 0.5$  (50%):

```

const float NOISE = 0.5f;
const float BASE_FREQUENCY = 1.0f;
...
void step() {
    if (metro) {
        metro.frequency( (1 + randomFloat(-NOISE, +NOISE)) * BASE_FREQUENCY );
        led.toggle();
    }
}

```

## Poisson Timing

Poisson timing circumvents the limitations of both period and frequency jittering. It picks intervals randomly, not from a uniform range, but from an *exponential distribution* of waiting times. In very simple terms, Poisson timing replaces the bounded uniform jitter with an unbounded distribution of intervals. Events can occasionally cluster (bursts) or be widely spaced (pauses), yet still preserve a specific long-run average rate.

This property makes Poisson timing qualitatively different from naive jittering:

- It does *not* constrain events to a narrow range around the base period.
- It naturally produces variability at different scales, including both micro-irregularities and rare, longer gaps.
- It guarantees that the long-run event rate  $R$  equals the desired base frequency, without introducing bias.

The Poisson process is well-suited to describe many real-world temporal phenomena where events occur irregularly yet with a well-defined average rate:

- **Physical systems:** radioactive decay of atoms; arrival of cosmic rays at a detector; photon counts in a photodiode.
- **Biological systems:** neuronal spikes in the brain; heartbeats with irregular rhythms; random seed dispersal events in plants.
- **Animal and human behavior:** arrival of customers at a service desk; keystrokes while typing; timing of coughs, sneezes, or blinks.
- **Social and urban processes:** cars passing a checkpoint; pedestrians entering a space; phone calls arriving at a call center.

These examples show that the Poisson distribution captures irregular but structured rhythms that occur across scales, from the micro (molecular or neuronal events) to the macro (flows of people or vehicles). For real-time media, this means adopting a model that resonates with patterns already found in the world, providing a perceptually more natural sense of rhythm and flow.

Mathematically, a Poisson process with rate parameter  $\lambda$  (average events per second) is defined such that the probability of observing  $k$  events in a time interval  $t$  is given by:

$$P(N(t) = k) = \frac{(\lambda t)^k}{k!} e^{-\lambda t} \quad (5)$$

From this definition, one can show that the intervals  $T_n$  between events are distributed according to an exponential law with mean  $1/\lambda$ :

$$P(T_n \leq t) = 1 - e^{-\lambda t} \quad (6)$$

which yields the sampling rule:

$$T_n = -\frac{1}{\lambda} \log U \quad (7)$$

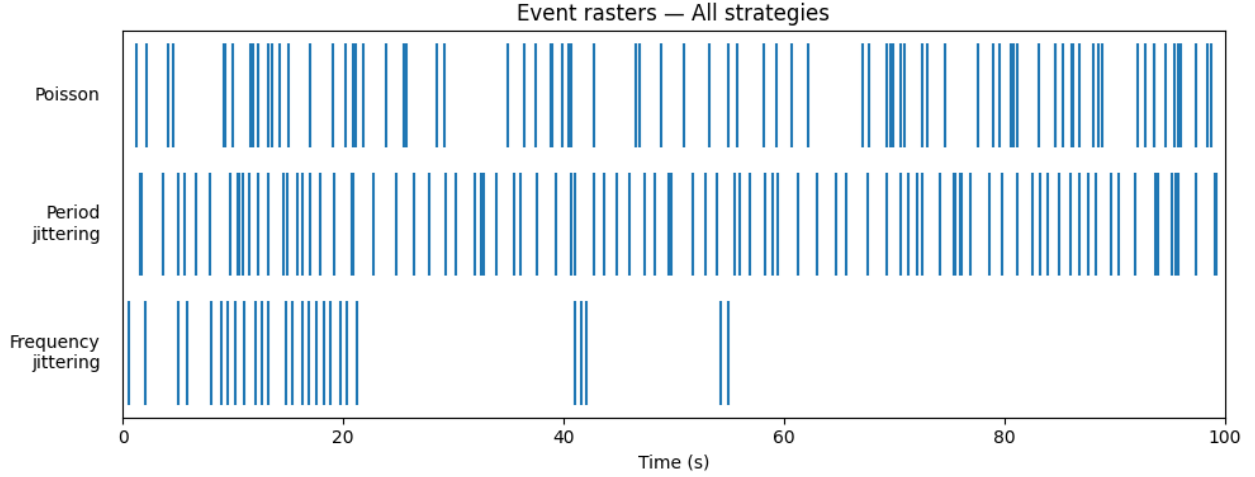
where  $U \sim \mathcal{U}(0, 1)$  is a uniform random number. This is the core mechanism: pick a uniform random number  $U$  between 0 and 1, apply the logarithm transformation, and scale by the base period. The result is a natural, bursty rhythm with the correct long-run event rate.

The critical limitation of period jittering is that it constrains the variability to a narrow, regular band around a fixed period, resulting in a process that remains machine-like, just slightly perturbed. By contrast, Poisson timing yields an exponential distribution of intervals: most events occur near the mean, but occasionally very close or very far apart. These bursts and gaps produce a texture that feels alive, much closer to natural processes. From an artistic standpoint, this qualitative shift is the central reason to adopt Poisson-based methods.

Yet, Poisson distribution does not suffer from the interval explosion of frequency jittering, because probabilities of picking outliers decreases exponentially. Thus, it ensures that the perceived pace matches the target long-run rate  $R = \lambda$  by directly controlling the distribution of  $T_n$  with mean  $1/\lambda$ .

Here is an example implementation of the Poisson timing in Plaquette.

```
void step() {
  if (metro) {
    float u = max(randomFloat(), FLT_MIN); // makes sure u > 0
    metro.period( -BASE_PERIOD * logf(u) );
    led.toggle();
  }
}
```



**Figure 3:** Event raster comparison of Poisson timing with period and frequency jittering,  $F_{base} = 1$ , uniformly sampled noise  $\varepsilon_{base} = 0.999$ .

## Poisson Mix

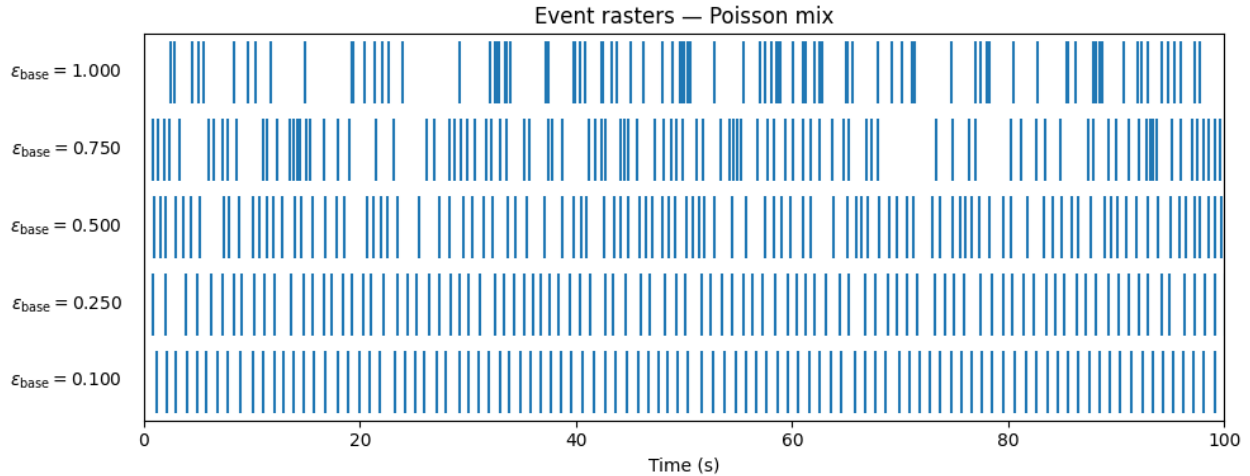
The Poisson distribution has a shortfall: it is not possible to directly control the level of randomness  $\varepsilon_{base}$ . With pure Poisson timing ( $\varepsilon_{base} = 1$ ), inter-arrival intervals  $T_n$  follow the exponential law and may vary wildly, occasionally clustering or leaving long gaps. While this is desirable for naturalistic textures, it may be too irregular for some design contexts.

A simple solution is to mix the exponential distribution with the deterministic base period, using  $\varepsilon_{base}$  as a mixing factor:

$$T_n = (1 - \varepsilon_{base})T_{base} + \varepsilon_{base} \left( -\frac{1}{\lambda} \log U \right) \quad (8)$$

where  $U \sim \mathcal{U}(0, 1)$  is a uniform random variable and  $\lambda = 1/T_{base}$ . This equation can be further simplified:

$$\begin{aligned} T_n &= (1 - \varepsilon_{base})T_{base} - \varepsilon_{base}T_{base} \log U \\ &= T_{base} - \varepsilon_{base}T_{base} - \varepsilon_{base}T_{base} \log U \\ &= T_{base} \left( 1 - \varepsilon_{base}(1 - \log U) \right) \end{aligned} \quad (9)$$



**Figure 4:** Event raster for Poisson mix,  $F_{base} = 1$ , uniformly sampled noise at different levels  $\epsilon_{base}$ .

This interpolation has the following properties:

- at  $\epsilon_{base} = 0$  it yields perfectly periodic timing:  $T_n = T_{base}$ ;
- at  $\epsilon_{base} = 1$  it generates full Poisson timing with exponential intervals;
- for other values  $0 < \epsilon_{base} < 1$ , it creates a continuum between regular pacing and bursty Poisson variability,

In practice, this gives the designer an intuitive parameter to control how natural the rhythm feels: lower values keep events close to a metronome, while higher values approach organic irregularity.

```
void step() {
  if (metro) {
    float u = max(randomFloat(), FLT_MIN); // makes sure u > 0
    metro.period( BASE_PERIOD * (1 - NOISE * (1 - logf(u)) );
    led.toggle();
  }
}
```

This method offers a balance between precise periodicity and natural irregularity, while remaining computationally simple (only requiring one logarithm per event).

## Implementation

Poisson timing was implemented as part of Plaquette 0.8, providing a proof of concept for support of Poisson-based timing on hardware with limited memory and computational capabilities. Two types of interfaces were provided: a *jitter()* function available in oscillator units Metronome and Wave, and a *randomTrigger()* global function for triggering events according to a Poisson distribution.

Support for the *jitter()* function in oscillator units only necessitates two floating point variables: one to hold the random frequency ratio and one for the jittering amount. Computing the next frequency ratio after each cycle requires calling a computationally expensive *logf()* or *log1pf()* function but only happens once per cycle, which is negligible in most real-time media applications for control purposes where the cycle frequency is typically low.

The *randomTrigger(period, interval)* function is reentrant and must be performed in each *step()* call. An alternative version *randomTrigger(period)* is provided that uses the main engine's inter-step interval. The probability of an event occurring in an average period  $T_{base}$  given an interval  $T_{step}$  between each call to the function *randomTrigger()* is equal to  $1 - e^{-T_{step}/T_{base}}$ . We use a 5-th-order Taylor polynomial series approximation to reduce computational costs.

This example shows the use of the *jitter()* function to noisify the period of a metronome. A low-frequency sine wave is used to modulate the amount of jittering:

```
const float BASE_PERIOD = 1.0f;
Metronome metro(BASE_PERIOD);
DigitalOut led(LED_BUILTIN);
Wave lfo(SINE, 20.0f); // LFO with period of 20 seconds

void begin() {}

void step() {
    metro.jitter(lfo);
    if (metro)
        led.toggle();
}
```

The following code uses the *randomTrigger()* function to trigger an event at an average base period, using Poisson distribution.

```
const float BASE_PERIOD = 1.0f;
DigitalOut led(LED_BUILTIN);

void begin() {}

void step() {
    if (randomTrigger(BASE_PERIOD))
        led.toggle();
}
```

## Conclusion

This report examined different strategies for generating temporally irregular yet statistically controlled event sequences in real-time media. We began with naive jittering of period and frequency, showing how these usual go-to methods are simple to implement but limited: period jittering preserves the average interval but constrains variability to a narrow band, while frequency jittering preserves the average frequency but can lead to extreme interval lengths and bias. Both approaches produce rhythms that may feel either too mechanical or unstable.

Poisson timing was then introduced as a better alternative. By drawing intervals from an exponential distribution, Poisson processes allow events to cluster or spread apart naturally while still preserving a desired long-run event rate. This produces textures that feel alive, resonating with patterns observed in physical, biological, and social processes. Finally, we proposed Poisson mix, which interpolates between strict periodicity and full Poisson variability, giving practitioners an intuitive control parameter for shaping the generation of random events.

This study has some limitations. The models were evaluated conceptually and through simulation, but not through formal perceptual and practice-based studies. It remains to be tested how different audiences perceive and differentiate the rhythmic textures produced by these methods in artistic contexts, and how their use affects creative practice. Furthermore, computational performance on

constrained microcontrollers was not benchmarked in detail.

Future work could address these limitations by combining technical evaluations with user studies involving artists and audiences. Integrating Poisson-based timing into higher-level creative frameworks, and comparing its expressive potential with other stochastic models (such as Gaussian noise, renewal processes, or fractal timing), also remain promising directions. By bringing statistical rigor into accessible tools, we hope to enrich the temporal vocabulary available to real-time media practitioners.

## References

- Audry, Sofian and Thomas Ouellet Fredericks (Nov. 10, 2025). “Plaquette: An Object-Oriented Framework for Embedded Signal Processing in Interactive Media”. In: *Journal of Open Source Software* 10.115, p. 9144. ISSN: 2475-9066. DOI: 10.21105/joss.09144. URL: <https://joss.theoj.org/papers/10.21105/joss.09144> (visited on 11/21/2025).
- Emmerson, Simon (Sept. 28, 2017). *Living Electronic Music*. London: Routledge. 216 pp. ISBN: 978-1-351-21786-6. DOI: 10.4324/9781351217866.
- Gullö, Jan-Olof and Gary Bromham (2025). “Manipulating Micro-Rhythm and Micro-Timing in Digital Music Creation, with a Focus on Mixing Music: Three General Perspectives”. In: 2nd Audiovisual Symposium 2024, 6 December, Falun, Sweden. Högskolan Dalarna. URL: <https://urn.kb.se/resolve?urn=urn:nbn:se:knh:diva-6048> (visited on 11/21/2025).
- Johnson, Max and Mark Gotham (2023). “Musical Micro-Timing for Live Coding”. In: *Proceedings of the 24th International Society for Music Information Retrieval Conference, ISMIR 2023, Milan, Italy, November 5-9, 2023*. Ed. by Augusto Sarti et al., pp. 98–105. DOI: 10.5281/ZENODO.10265231. URL: <https://doi.org/10.5281/zenodo.10265231>.
- Poisson, Siméon Denis (1837). *Recherches sur la probabilité des jugements en matière criminelle et en matière civile; précédées des règles générales du calcul des probabilités*. Paris, Bachelier. 444 pp. URL: <http://archive.org/details/recherchessurlap00pois> (visited on 11/21/2025).