Chapter 1

dummy-hack Introduction

Chapter 2

dummy-hack Background

Chapter 3

Baseline

3.1 Phase-synchronization

3.2 Phase- & frequency-synchronization

3.3 K. Nymoen's bi-directional phase-adjustment

3.4 K. Nymoen's middle SA-leveled frequency-adjustment

3.5 System target state: harmonic synchrony

The state of harmonic synchrony is defined [1] as the state in which all agents in the musical collective "fire"/"flash", as described in Subsection 5.1.2, at an even and underlying interval or pulse, a certain number of times in a row. This is not to say all agents will have to "fire"/"flash" simultaneously, as has traditionally been the case for pulse-coupled oscillators []. Exactly how this can look is shown in Section 5.4, especially in Figure 5.1.

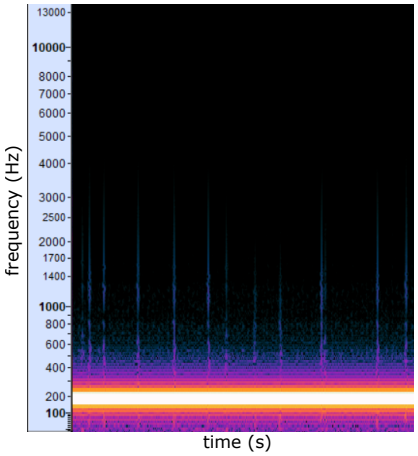
As one is designing and creating an interactive music technology system, one might want to encourage and allow for the playing of various musical instruments at various rhythms/paces, as it might be quite boring if all instruments were played at the exact same measure or pulse. As K. Nymoen et al. [1] reason when discussing their own interactive "Firefly" music-system, as well as coining the term of *harmonic synchrony*:

Temporal components in music tend to appear in an integer-ratio relation to each other (e.g., beats, measures, phrases, or quarter notes, 8ths, 16ths).

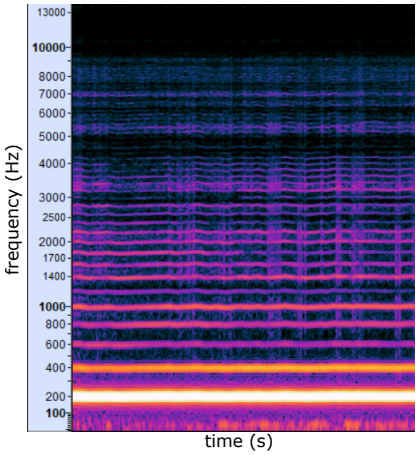
and

Being an interactive music system, people may want their device to synchronize with different subdivisions of a measure (e.g. some play quarter notes while others play 8ths).

Accomodating for these aspects then, K. Nymoen et al. took inspiration for achieving synchronization in a decentralized system from the concept of *harmonics* in the frequency spectrum of a waveform, in that each harmonic wave or overtone has a frequency with an integer-relationship to the fundamental (smallest) frequency. This phenomenon can e.g. be seen in the frequency spectrogram of a humanly hummed G3-tone, depicted in Figure 3.1b, where one can observe the presence of harmonics and overtones having frequencies with integer relationships to the fundamental (smallest) frequency at around 196 Hz.



(a) The frequency spectrogram of the audible waveform being a monotone and purely generated G3-tone at 195.99 Hz [2].



(b) The frequency spectrogram of the audible waveform being a more-or-less monotone but non-pure G3-tone, hummed and recorded by me [], as I tried to repeat the tone in 3.1a with my voice.

Figure 3.1: Frequency spectrograms of two different-sounding waveforms of the same G3-tone at 195.99 Hz. Note the absence and presence of harmonics and overtones in waveform 3.1a and 3.1b respectively, as well as the integer-relationships between the fundamental (lowest) frequency and the harmonics in 3.1b. Frequencies in a harmonically synchronized agent collective will for the first ϕ-problem resemble the frequencies in 3.1a, where all frequencies are equal and constant. Conversely, when frequencies can be heterogenous and unequal, as in the ϕ- & ω-problem, the frequencies in a harmonically synchronized agent collective will rather resemble the frequencies in 3.1b, where these higher frequencies with integer-relationships to the fundamental and lowest frequency can be present.

More accurately then, and inspired by integer-relationships in the frequencies of harmonics and the fundamental frequency in the spectrogram of a waveform, K. Nymoen et al. [1] describe the *legal* frequencies the robots's oscillators in the musical robot collective have to adhere to in order for the oscillator-frequencies to be considered *harmonically synchronized*:

All musical robots i, in a harmonically synchronized state, will have frequencies ω_i which are element in the mathematical set

$$\Omega_{legal}(\omega_0) = \omega_0 \cdot 2^{\mathbb{N}_0} = \{\omega_0, 2\omega_0, 4\omega_0, 8\omega_0, ...\}, \tag{3.1}$$

where ω_0 is the lowest frequency in the robot-collective (or the fundamental frequency if you will), and $\mathbb{N}_0$ are the natural numbers including the number zero. If e.g. the smallest oscillator-frequency in the musical robot collective (ω_0) was equal to 1.5Hz, *legal* oscillator-frequencies the rest of the musical robots in the collective could have would be $\Omega_{legal}(1.5\text{Hz}) = \{1.5\text{Hz}, 3\text{Hz}, 6\text{Hz}, 12\text{Hz}, ...\}$.

Hence, in terms of oscillator-/robot-frequencies ω_i for all robots i, we have a harmonically synchronized robot-collective if (and only if)

$$\omega_i \in \Omega_{legal}(\omega_0), \forall i, \tag{3.2}$$

where we then say all robots in the robot-collective have *legal* frequencies.

This state of *harmonic synchrony* is then the system goal state K. Nymoen et al. achieve using their phase- and frequency-update/-adjustment functions, as explained above in Section 3.3 and 3.4, and is also the target/goal state we want to continue achieving in this thesis, but hopefully with even more *self-aware* methods (which are hopefully quicker and more robust).

3.5.1 Detecting harmonic synchrony

In order to test and evaluate synchronization-performance in their firefly-inspired oscillator-system, K. Nymoen et al. [1] develop a measure used to detect when the system has reached a state of synchrony. Using the firings of the fireflies, some well-defined conditions have to be met in order for the fireflies to be deemed *harmonically synchronized*:

- **Condition 1**: Firing may only happen within a short time-period t_f.
- **Condition 2**: All nodes must have fired at least once during the evaluation period.
- **Condition 3**: Between each t_f, a period t_q without fire events must be equally long k times in a row.

By utilizing transmitted firings/pulses from the robots in our robot-collective, these conditions can be enforced and checked throughout the synchronization-process, in order to detect if the oscillator-network becomes harmonically synchronized.

For getting a better idea of how these conditions being met looks like, see the *performance-measure plot* in Figure 5.1 where the oscillators/robots fulfill the abovementioned requirements right before ending the synchronization process.

These requirements, amongst other illustrations in Nymoen et al.'s paper [1], thus constitutes a blueprint for the design of a performance-/synchrony-measure able to detect the achievement of harmonic synchrony in a decentralized network of "firing"—or pulse-coupled—oscillators. The time having passed from the start of the synchronization-process until the detection of harmonic synchrony will then be defined as the performance-score, indicating how fast or slow the oscillators are at synchronizing.

The exact details of how such a performance-/synchrony-measure is implemented for our musical multi-robot oscillator-network, in the synchronization-simulator, will be given in Section 5.4.

Chapter 4

dubby-hack Tools n stuff

Chapter 5

Implementation

5.1 Simulator setup: the musical multi-robot collective

5.2 Synchronizing oscillator-phases

5.3 Synchronizing oscillator-frequencies

5.4 Performance-/synchrony-measure: detecting harmonic synchrony

The performance-measure will be used in our synchrony-simulator to evaluate and test the multi-robot collective's ability to harmonically synchronize to each other. As mentioned in Subsection 3.5.1, K. Nymoen et al.'s requirements and illustrations [1] for achieving *harmonic synchrony* serve as a blueprint or guide for how to similarly implement our synchrony-/performance-measure. This performance-measure should be able to, during synchronization-simulation, detect if harmonic synchronization has been achieved in our decentralized oscillator-network. The successful triggering of this detection will then in turn terminate the synchronization simulation-run and save to a dataset the time it took to synchronize (the performance-score), in the case of a 'synchronization-success' — an example of which can be seen in Figure 5.1.

The resulting and corresponding performance-scores obtained using this performance-measure will then take values of the simulation-time (in seconds) it takes for the robot-collective, from the start of the synchronization-simulation, to achieve the system target state of *harmonic synchrony*, as specified in Section 3.5.

Now if **Conditions 1-3** from Subsection 3.5.1 are kept, we have harmonic synchrony.

My specific implementation of the performance-/synchrony-measure essentially consists of enforcing all the requirements or rules listed in 3.5.1, given some constant t_f- and k-values (e.g. $80ms$ and 8 respectively [1]). And again—to recall from 3.5.1—t_f is the short time-window within which nodes are allowed

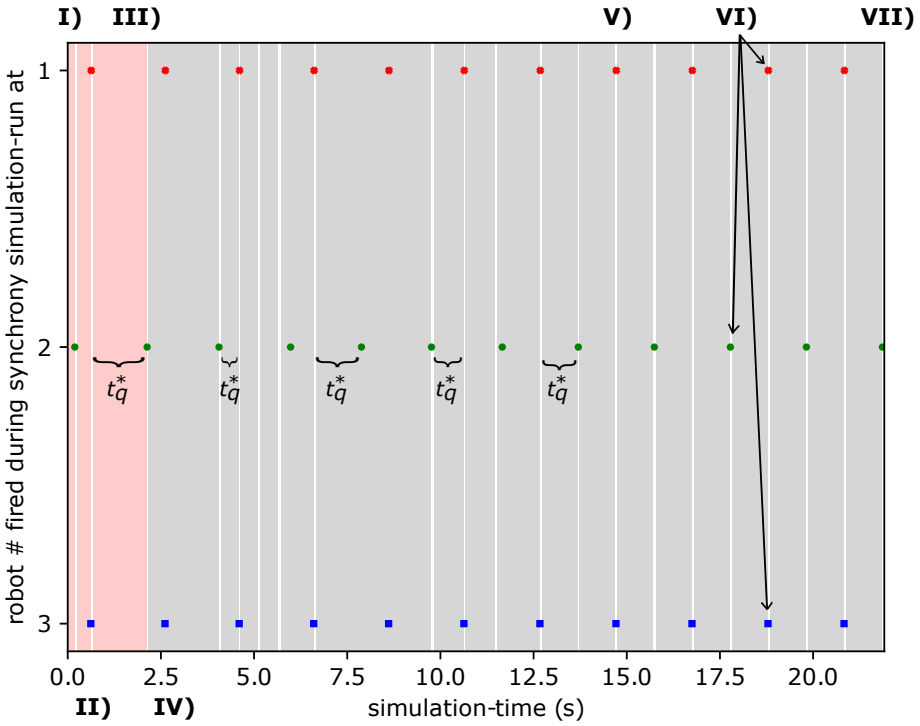


Figure 5.1: **Performance-/synchrony-measure plot**: recording a detection of harmonic synchrony. Initial red area represents the "start-up"-period where no (gray) t_q-value is defined yet. **I)** First robot firing, initiating a process of defining such a "silent" time-window t_q within which no nodes are allowed to fire, and a whole "legal-firing" t_f-window (here shaded in white and only 80ms long, as in [1]) ensues. **II)** At least one robot firing, and with their firing-time(s) giving us the *early* t_q*-defining* time-value (see schema below), as well as triggering second whole t_f-window to ensue. **III)** At least one robot firing after the *early* t_q*-defining* time-value was found, hence (with their firing-time(s)) giving us the *late* t_q*-defining* time-value right before using these when finishing the on-started process of defining the "silent" time-window t_q; triggering a half (due to stabilization purposes by future centering of fire-events) t_f-window to ensue. **IV)** A robot is caught firing illegally during a "silent" t_q-window and hence resets the *towards_k_counter*-variable, as well as (re-) starts a process like in **I)** for (re-) defining a new t_q-value. **V)** Firing after this point only happens during short "legal-firing" t_f-windows, so **Condition 1** (cf. 3.5.1) is held. **VI)** All robots have fired at least once during the evaluation-period, so **Condition 2** is held. **VII)** The robot collective has fired legally and evenly, without resetting t_q, k (equal to 8 in this case) times in a row, so **Condition 3** is fulfilled — and harmonic synchrony is thus achieved and detected after 21.9 seconds of synchronization.

to fire at each beat, and k represents how many times nodes have to fire at even underlying pulses/beats in a row without changing the t_q-period—before becoming harmonically synchronized.

The requirement of firing evenly k times in a row with identical t_q-periods can be—and in fact is in our implementation—enforced by incrementing an integer variable *towards_k_counter* after a 'legal' t_f-window has occured (i.e. one or more nodes fired inbetween the onset and ending of the t_f-window), and conversely by resetting *towards_k_counter* to 0 when an illegally transmitted firing was heard during a 'silent' (or so it was supposed to be at least) t_q-window, hence restarting the synchrony-detection process—as can be seen occuring several times in Figure 5.2.

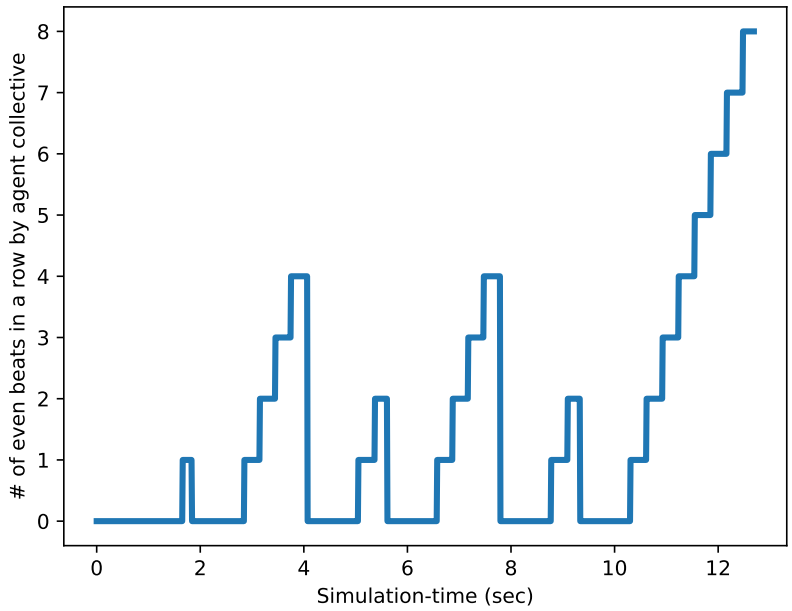


Figure 5.2: A **synchrony-evolution plot**, displaying the temporal recording of the *towards_k_counter*-variable throughout a synchrony simulation-run in Unity. The counter is incremented as the robot-collective fires evenly within 'legal' t_f-windows, and is conversely reset to 0 if illegal firings during 'silent' t_q-windows are heard. Note that in this specific simulation-run above, the agents were on their way to achieve harmonic synchrony five times before the 10th second of the synchronization-simulation already, but since one or more of them fired 'illegally' (i.e. inside a t_q-window), they were consequently 'punished'—or rather deemed 'not synchronized enough yet'—by getting their counter reset to 0. Eventually however, through further phase- & frequency-synchronization, the multi-robot collective was in this case after 12.5 seconds able to achieve harmonic synchrony, when *towards_k_counter* became equal to k, as well as all other requirements for achieving *harmonic synchrony* was met.

Initially, the t_q-period/-window is not initialized, as it entirely depends on the frequencies to which the robot-collective converges to; however, when an illegal firing (i.e. a firing perceived during a t_q-window) occurs—t_q is also then reset itself to a hopefully more correct value, given by the following formula,

which is visually explained further in the schema in Figure 5.3:

$$\begin{aligned} t_q^* &= late\_t_q^*\_defining\_timevalue - early\_t_q^*\_defining\_timevalue - t_f \\ &= median(t_1^{'}, t_2^{'}, ..., t_m^{'}) - median(t_1, t_2, ..., t_n) - t_f, \end{aligned} \tag{5.1}$$

where n fire-events were recorded during the earlier t_f-window, and m fire-events were recorded during the later t_f-window.

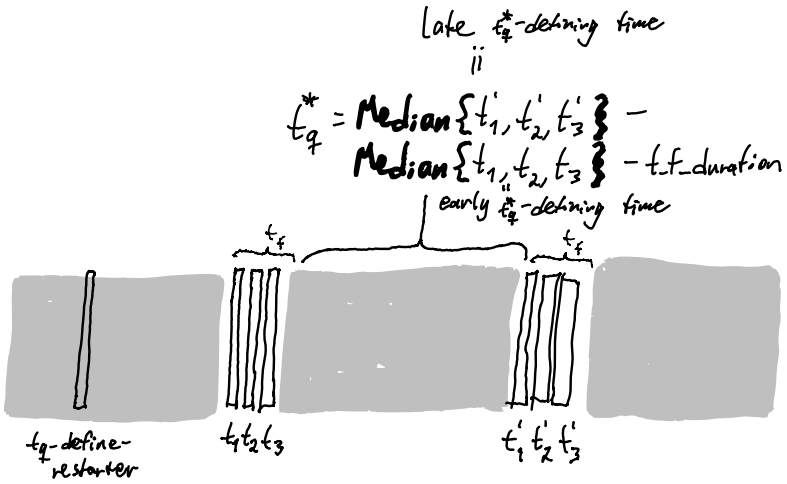


Figure 5.3: Schema: how "silent" t_q-windows are defined. Til Kyrre: Dette var bare grovt, som jeg tenker jeg må lage mer profesjonelt i Inkscape ellernoe.

The specific steps and procedures needed to implement this performance-/synchrony-measure is expounded in detail in Algorithms 1, 2, and **??**.

Algorithm 1: Performance-/synchrony-measure part A (må evt. fylles inn)

Algorithm 2: Performance-/synchrony-measure part B (må evt. fylles inn)

If a certain amount of time, e.g. 5 minutes [1], has gone without the detection of harmonic synchrony occuring, the simulation-run is terminated as a "fail".

Bibliography

[1] Kristian Nymoen et al. "Decentralized harmonic synchronization in mobile music systems". In: *2014 IEEE 6th International Conference on Awareness Science and Technology (iCAST)*. IEEE, 2014, pp. 1–6.

[2] Tomasz P. Szynalski. *Online Tone Generator*. URL: `https://www.szynalski.com/tone-generator/` (visited on 02/02/2022).