# Computational Self-Awareness in Musical Robotic Systems

*Endowing musical robots with self-awareness — an experimental and exploratory study*

David Thorvaldsen

Thesis submitted for the degree of
Master in Informatics: Robotics and Intelligent Systems
60 credits

Institute for Informatics
Faculty of mathematics and natural sciences

UNIVERSITY OF OSLO

Spring 2022

# Computational Self-Awareness in Musical Robotic Systems

*Endowing musical robots with self-awareness — an experimental and exploratory study*

David Thorvaldsen

# Abstract

MSc Thesis/Project Summary.

# Acknowledgements

Tusen takk til alle som har støttet og hjulpet meg. Sånn faktisk.

**Soli Deo Gloria**

# Contents

**1 Baseline** **1**
1.1 System target state: harmonic synchrony . . . . . . . . . . . . . . 1
1.1.1 Detecting harmonic synchrony . . . . . . . . . . . . . . . . 4
1.2 Phase synchronization . . . . . . . . . . . . . . . . . . . . . . . . . . 5
1.2.1 Problem statement . . . . . . . . . . . . . . . . . . . . . . . . 5
1.2.2 Synchronizing phases via phase adjustment . . . . . . . . 5
1.3 Phase and frequency synchronization . . . . . . . . . . . . . . . . . 6
1.3.1 Problem statement . . . . . . . . . . . . . . . . . . . . . . . . 6
1.3.2 Synchronizing frequencies via frequency adjustment . . . 6

# List of Tables

# List of Figures

1.1 Frequency spectrograms illustrating the absence and presence of harmonics and overtones in audible waveforms . . . . . . . . . . . 3

# Chapter 1

# Baseline

**BESKR.:** [ Mellomkapittel om allerede-eksisterende approaches og teori jeg brukte som starting point og som jeg verifiserer og sammenlikner med senere ].

Kristian Nymoen et al. [2] showed how one can, by endowing oscillators with self awareness capabilities amongst other measures, achieve synchronization (*harmonic* at that, cf. 1.1.1) of pulse coupled oscillators's initially unsynchronized phases and frequencies.

As Nymoen designed update functions for synchronizing both phases and frequencies, not only were their firefly-inspired oscillators endowed with self awareness capabilities; they were also designed to adjust their phases and frequencies with (close to) 0 change half-way through the oscillator cycle. These (close to) 0 changes half-way through oscillators's cycles enable individual musical agents to play with different measures (e.g. half or 1/4th as fast) compared to other individuals—being characteristic for musical interaction with several musical individuals—as well as enabling the possibility of achieving the target state of harmonic synchrony (cf. 1.1.1). Their contribution also differs from previous approaches in that Nymoen et al.'s fireflies only need to transmit "fire" signals every other phase climax, not every single phase climax; in other words, synchronization is achieved despite less communication.

First, the relatively new system target state of *harmonic synchrony* will be expounded and explained in 1.1. Then, the aforementioned update functions for both phase ($\phi$) and frequency ($\omega$), explaining the incorporated level of self awareness as well as how a decentralized collective of pulse copuled oscillators can achieve harmonic synchrony, are thus presented in 1.2 and 1.3.

## 1.1 System target state: harmonic synchrony

The state of harmonic synchrony is defined [2] as the state in which all agents in the musical collective "fire" or "flash", as described in Subsection **??**, 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 strict synchronization in pulse-coupled oscillators []. How legal and harmonically synchronized oscillator frequencies and phases look like will be described in this Section.

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. [2] reason while 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 coining a novel state of synchronization for a decentralized system. This novel state arose 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 in a waveform. Such a phenomenon, used as inspiration by Nymoen et al. but not as the basis for any definitions directly, can e.g. be seen in the frequency spectrogram of a humanly hummed G3-tone, depicted in Figure 1.1b. In said example, one can observe the presence of harmonics and overtones having frequencies with integer relationships to the fundamental (smallest) frequency at around 196 Hz, as we see the overtones e.g. has frequencies of around 400Hz, 800Hz, and 1600Hz. We will soon see how such relationships in frequency are used as inpiration for a new definition of a synchronous state; however, the example in Figure 1.1 is only meant to give an idea of where this new state of synchronization stems from, and not as a visual representation of said novel synchrony state.

More accurately then and inspired by integer relationships in waveforms, although not exactly analogus to the example above in Figure 1.1, Nymoen et al. describe *legal* frequencies the oscillator frequencies has to lie within to be considered *harmonically synchronized*:

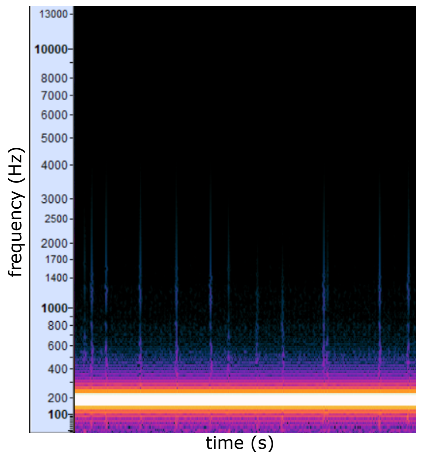
**Legal frequencies definition:**

All musical oscillators $i$, in a harmonically synchronized state, will have frequencies $\omega_i$ which are element in the mathematical set

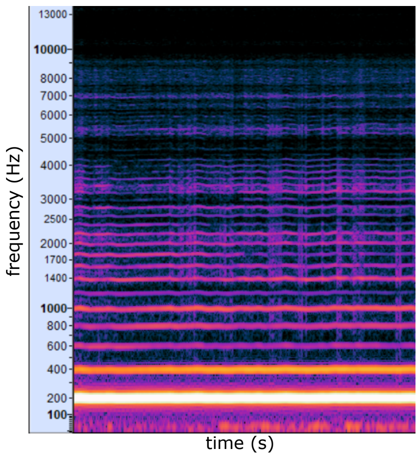
$$\Omega_{legal}(\omega_0) = \omega_0 \cdot 2^{\mathbb{N}_0} = \{\omega_0, 2\omega_0, 4\omega_0, 8\omega_0, ...\}, \tag{1.1}$$

where $\omega_0$ is the lowest frequency in the oscillator 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 oscillator collective ($\omega_0$) was equal to 1.5Hz, *legal* frequencies the rest of the oscillators in the collective could have would be $\Omega_{legal}(1.5\text{Hz}) = \{1.5\text{Hz}, 3\text{Hz}, 6\text{Hz}, 12\text{Hz}, ...\}$.



(a) The frequency spectrogram of the audible waveform being a monotone and purely generated G3-tone at 195.99 Hz [3].



(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 1.1a with my voice.

Figure 1.1: 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 1.1a and 1.1b respectively, as well as the integer-relationships between the fundamental (lowest) frequency and the harmonics in 1.1b. Frequencies in a harmonically synchronized oscillator collective will for the first $\phi$-problem resemble the frequencies in 1.1a, where all frequencies are equal and constant. Conversely, when frequencies can be heterogenous and unequal, as in the $\phi$- & $\omega$-problem, the frequencies in a harmonically synchronized oscillator collective will rather resemble, although not completely correspond to, the frequencies in 1.1b, where higher frequencies with integer relationships to the fundamental and lowest frequency can be present at the same time.

Hence, in terms of frequencies $\omega_i$ for all oscillators $i$ in the oscillator network, we have *harmonically synchronized* and *legal* oscillator frequencies $\omega_i$ if (and only if)

$$\omega_i \in \Omega_{legal}(\omega_0), \forall i. \tag{1.2}$$

Note that the only difference between this definition of *legal oscillator frequencies* and the spectrogram example depicted in Figure 1.1 lies in that for the legal frequencies as defined in (1.1), higher frequencies can only have doubling integer relationships with the lowest fundamendal frequency (e.g. twice or four times as high), whereas the frequencies of the harmonics seen in the humanly hummed G3 tone (1.1b) additionally had all other positive integer relationships (e.g. three or five times as high) with the fundamental frequency.

Now we will see what this entails for the phases $\phi_i$ for all the oscillators in the collective.

**Legal phases definition:**

FYLL INN DEN NYE MATEMATISKE BESKRIVELSEN DU KOM FRAM TIL HER OM DAGEN

This state of *harmonic synchrony* is then the system goal state K. Nymoen et al. achieve using their phase and frequency update functions, as explained above in Section **??** and **??**; and is also the target or goal state we want to continue achieving and experimenting for in this thesis.

### 1.1.1 Detecting harmonic synchrony

In order to test and evaluate synchronization performance in their firefly-inspired oscillator-system, K. Nymoen et al. [2] develop a measurement 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*:

**Harmonic synchronizaton conditions:**

**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 or 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 **Harmonic synchrony detection plot** in Figure **??** where the oscillators fulfill the abovementioned requirements right before ending the synchronization process.

These requirements, amongst other illustrations in Nymoen et al.'s paper [2], thus constitutes a blueprint for the design of a performance or synchrony measurement 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 or synchrony measurement is implemented for our musical multi-robot oscillator-network, in the synchronization-simulator, will be given in Section **??**.

## 1.2 Phase synchronization

### 1.2.1 Problem statement

In this first and more simple synchronization problem, which we call the **phase ($\phi$) synchronization problem**, we assume homogenous and already-synchronized frequencies $\omega_i = 1Hz$ for all oscillators $i$. The oscillator collective hence only has to synchronize phases $\phi_i$ for all oscillators $i$, which are initially uniformly random phase values $\in [0, 1]$. Hence, phase adjustments are needed in order to synchronize phases in the pulse coupled oscillator collective.

### 1.2.2 Synchronizing phases via phase adjustment

#### 1.2.2.1 Mirollo-Strogatz's "standard" phase adjustment

One approach having been used to achieve this in the past is Mirollo-Strogatz's "Standard" phase-adjustment in oscillators [1].

Each oscillator gets a new phase, $\phi(t^+) = P(\phi(t))$, accoring to the **phase update function** (1.3) upon perceiving a "fire"-event from one of the other musical nodes:

$$\phi(t^+) = P(\phi(t)) = (1 + \alpha)\phi(t), \tag{1.3}$$

where $\alpha$ is the pulse coupling constant, denoting the strength between nodes [2], $t^+$ denotes the time-step immediately after a "fire"-event is heard, and $\phi(t)$ is the old frequency of the oscillator at time $t$.

So, if e.g. $\alpha = 0.1$, then a musical oscillator's new and updated phase, immediately after hearing a "fire"-signal from another oscillator, will be equal to $\phi(t^+) = P(\phi(t)) = (1 + 0.1)\phi(t) = 1.1\phi(t)$. 110% of its old phase $\phi(t)$, that is. Hence, and in this way, the oscillator would be "pushed" to fire sooner than it would otherwise (as nodes fire once they have reached phase-climax $\phi(t) = 1$).

#### 1.2.2.2 K. Nymoen's bi-directional phase adjustment

Apart from altering a similar phase adjustment function to the likes of Mirollo-Strogatz's phase adjustment function so that theirs would cooperate well with the system's frequency adjustment in achieving their target goal of *harmonic synchrony* (further explained towards the end of this chapter), K. Nymoen et al. [2] here introduces an example of a bi directional phase adjustment method. The difference between a bi directional method of phase synchronization and a one directional one is that using bi directional phase adjustment, oscillators are simply adjusting each other's phases in an excitatory way; they only "push" other oscillators's phases further or higher (positive updates) when firing themselves, never "holding" or "dragging" them back (negative updates).

Hence, this newer approach to phase adjustment works very similarly to the phase adjustment performed in the "standard" ***Mirollo-Strogatz*** approach presented earlier; the only difference being that now, oscillators update their phases with the slightly more complex **phase update function** (1.4) when hearing a "fire"-event from one of the other musical nodes — allowing for both larger, but also smaller, updated phases compared to the old phases:

$$\phi(t^+) = P(\phi(t)) = \phi(t) - \alpha \cdot sin(2\pi\phi(t)) \cdot |sin(2\pi\phi(t))|, \tag{1.4}$$

where $t^+$ denotes the time-step immediately after a "fire"-event is heard, and $\phi(t)$ is the old frequency of the oscillator at time $t$.

The fact that new and updated phases can both be larger, but also smaller, compared to the old phases, is exactly what's meant by the phase-adjustment being ***bi-directional***, or as the authors call it in the paper as using both excitatory and inhibitory phase couplings between oscillators [2].

The effects then of adjusting phases—upon hearing "fire"-events, according to this newest update-function (1.4)—are that the nodes's updated phases $\phi(t^+)$, compared to their old phases $\phi(t)$, now get decreased if $\phi(t)$ is lower than 0.5, increased if $\phi(t)$ is higher than 0.5, and neither—at least almost—if the phases are close to 0.5. This is due to the negative and positive sign of the sinewave-component in Equation (1.4), as well as the last attenuating factor in it of $|sin(2\pi\phi)| \approx |sin(2\pi\frac{1}{2})| = |sin(\pi)| = |0| = 0$, then if we have $\phi(t) \approx 0.5 = \frac{1}{2}$.

## 1.3 Phase and frequency synchronization

### 1.3.1 Problem statement

In this second and more challenging synchronization problem, which we call the **phase and frequency ($\phi$&$\omega$) synchronization problem**, we no longer assume homogenous and already-synchronized oscillator frequencies like in the first phase ($\phi$) synchronization problem when all frequencies were fixed to 1Hz. Now, the oscillators's frequencies are initialized to uniformly random frequencies $\omega_i$ within a certain minimum initial frequency $\omega_{min}^{init}$ (e.g. 0.5Hz) and maximum initial frequency $\omega_{max}^{init}$ (e.g. 4Hz). Hence, the problem is now more complex; the oscillator collective not only has to synchronize all initially random phases $\phi_i$ as above, but simultaneously, also all initially random frequencies $\omega_i$ for all oscillators $i$ as well. Hence, both phase *and* frequency adjustments are needed in order to synchronize both phases $\phi$ and frequencies $\omega$ in the oscillator collective.

### 1.3.2 Synchronizing frequencies via frequency adjustment

Only one frequency adjustment / update function was used as a starting point for this thesis and eventually implemented in the Unity synchrony simulator. This one frequency synchronization method, as invented by Nymoen et al. [2], is now presented.

**K. Nymoen's frequency adjustment**

This approach to frequency adjustment in pulse coupled oscillators, in order to achieve synchronized frequencies, stands in contrast to previous approaches of synchronization where oscillators frequencies are either equal and fixed, oscillators have to send out a pulse every oscillator cycle, or they do not include any explicit self awareness capabilities. This is not the case for Nymoen's frequency adjustment.

In order to achieve *harmonically synchronized* and "legal" oscillator frequencies (cf. 1.1.1) in the oscillator collective, not only the phases ($\phi$) have to be adjusted and eventually synchronized; frequencies ($\omega$) also have to be synchronized through frequency adjustment.

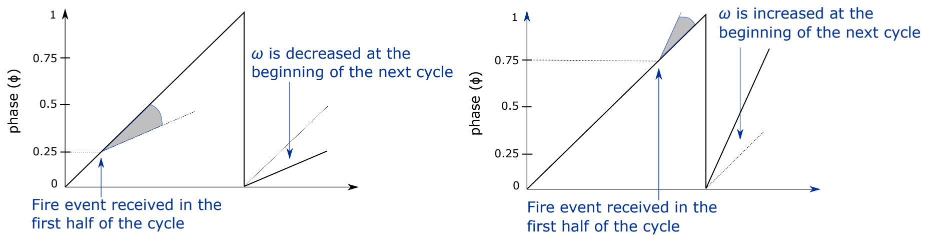


Figure 1.2: An illustration of how frequencies are adjusted after each phase climax (i.e. $\phi = 1$) using Nymoen et al.'s frequency update function. Reproduced from [2].

**BESKR.:** [ OPPRYDDET HITTIL ].

When it comes to the temporality and timing of when these update functions are used and applied; Musical agents's phases get updated/adjusted immediately as "fire"-/"flash"-events are perceived, whereas agents's frequencies do not get updated until the end of their oscillator-cycle (i.e. when having a phase-climax $\phi(t) = 1$). This is also the reason why frequencies are updated discretely, not continuously. So-called H-values however, being "contributions" with which the frequencies are to be updated according to, are immediately calculated and accumulated when agents are perceiving a "fire"-/"flash"-event — and then finally used for frequency-adjustment/-updating at phase-climaxes.

Each oscillator $i$ updates their frequency, on their own phase-climax (i.e. when $\phi_i(t) = 1$), according to the frequency-update function $\omega_i(t^+)$:

$$\omega_i(t^+) = \omega_i(t) \cdot 2^{F(n)}, \tag{1.5}$$

where $t^+$ denotes the time-step immediately after phase-climax, $\omega_i(t)$ is the old frequency of the oscillator at time $t$, and $F(n) \in [-1, 1]$ is a quantity denoting how much and in which direction an oscillator should update its frequency after having received its $n$th "fire"-signal.

The following steps is how the aforementioned $F(n)$ value is found, and can also be refound in Algorithm **??** and **??** in Chapter **??**:

**Step 1: the "in/out-of synch" error measurement / score, $\epsilon(\phi(t))$** Describing the error measurements at the n-th "fire"-event, we introduce an error measurement function.

The error measurement function value (1.6) is calculated immediately by each oscillator $i$, having phase $\phi_i(t)$, when a "fire" event signal from another oscillator $j, j \neq i$ is detected by oscillator $i$ at time $t$.

$$\epsilon(\phi_i(t)) = sin^2(\pi\phi_i(t)) \tag{1.6}$$

As we can see from this error-function, the error-score is close to 0 when the oscillator phase $\phi_i(t)$ is itself close to 0 or 1 (i.e. the oscillator either just fired/flashed, or is about to fire/flash very soon). This implies that if it was only short time ago since we just fired, or conversely if there is only short time left until we will fire, we are not much in error or *out-of-synch*.

The error-score is the largest when an oscillator perceives a “fire”-signal while being half-way through its own phase (i.e. having phase $\phi(t) = 0.5$). We could also then ask ourselves, does not this go against the main/target goal of the system, being *harmonic synchrony* — if agents are allowed to be “half as fast” as each other? We could imagine a completely “legal” and harmonically synchronous scenario where two agents have half and double the frequency of each other. The oscillator with half the frequency of the faster oscillator would then have phase $\phi(t) = 0.5$ when it would hear the faster oscillator “fire”/“flash” — leading to its Error-score $\epsilon(0.5) = sin^2(\pi/2) = 1$, which then makes it seem like the slower oscillator is maximally out of synch, when it is actually perfectly and harmonically synchronized. This calls out for an attenuating mechanism in our frequency update function, in order to “cancel out” this contribution so that perfectly harmonically synchronized agents will not be adjusted further despite their high Error-measurement. As we will see below, exactly such an attenuating mechanism is utilized in our frequency-adjustment method.

This error-measurement/-score forms the basis and fundament for the first component of self-awareness, being the *self-assessed synchrony-score* $s(n)$.

**Step 2: The first self-awareness component, s(n)** This aforementioned self-assessed synchrony-score, $s(n)$, is in fact simply the median of error-scores $\epsilon$.

If we then have a high $s(n)$-score, it tells us that the median of the $k$ last error-scores is high, or in other words that we have mainly high error-scores — indicating that this oscillator is out of synch. Conversely, if we have a low $s(n)$-score, indicating mainly low error-scores for the oscillator — then we have an indication that the oscillator is in synch, hence leading to low error scores, and in turn low $s(n)$-scores.

In other words, each oscillator hence has a way to assess themselves in how much in- or out-of-synch they believe they are compared to the rest of the agents. This is then the first level of self-awareness in the design.

**Step 3: frequency update amplitude- & sign-factor, $\rho(n)$** Describing the amplitude and sign of the frequency-modification of the n-th “fire-event” received. It is used to say something about in which direction, and in how much, the frequency should be adjusted.

$$\rho(\phi) = -sin(2\pi\phi(t)) \in [-1, 1] \tag{1.7}$$

For example, if an oscillator $i$ has phase $\phi_i(t) = 1/4$, it gets a value $\rho(1/4) = -sin(\pi/2) = -1$; meaning, the oscillator’s frequency should be decreased (with the highest amplitude actually) in order to "slow down" to wait for the other nodes. Conversely, if an oscillator $j$ has phase $\phi_j(t) = 3/4$, it gets a value $\rho(3/4) = -sin(3/2\pi) = -(-1) = 1$; meaning, the oscillator’s frequency should be increased (with the highest amplitude) in order to getting "pushed forward" to catch up with the other nodes.

Acts as an attenuating factor, when $\phi(t) \approx 0.5$, in the making of the H-value — supporting the goal of *harmonic synchrony*.

**Step 4: the H-value, and the H(n)-list** The following value, acting as "frequency update contributions", is then as previously mentioned calculated immediately after the oscillator perceives another oscillator's "flashing" signal:

$$H = \rho \cdot s \tag{1.8}$$

Here we then multiply the factor $\rho(n)$ representing how much, as well as in which direction, the oscillator should adjust its frequency, together with a factor $s \in [0, 1]$ of the adjusting oscillator's self-assessed synch-score. This implies that $H \in [0, \rho(n)]$. We hence see that the smallest value $H(n)$ can take for the $n$th "fire"-event is -1, which it does when $\phi = 0.25$ and $s = 1$. The highest value it can take is 1, which it does when $\phi = 0.75$ and $s = 1$. We can also see that even though the self-assessed synch-score $s(n)$ (i.e. the median of error-scores) is high and even the maximum value of 1, thus indicating consistent high error-scores (judging by error-function (1.6)) — the "frequency-update-contribution" $H(n)$ can in the end be cancelled out, as alluded to before, if in fact the amplitude- & sign-factor $\rho(n)$ is equal to 0. Hence, if we have two agents then where the one is twice as fast as the other, and we accept the $H(n)$-value as the "frequency-update-contribution", the slower oscillator which will hear "fire"-events consistently when it has its phase $\phi \approx 0.5$ (if the agents are synchronized) will, even though it gets a high *out-of-synch score* $s \approx 1$, not "be told" to adjust its frequency more by getting a large "frequency-update-contribution", but in fact "be told" not to adjust its frequency more due to the small or cancelled-out "frequency-update-contribution."

To recall, the self-assessed synch-score $s(n)$ tells an adjusting oscillator how in- or out-of-synch it was during the last $m$ perceived "fire"-/"flash"-events — where $s(n) = 0$ signifies a mean of 0 in error-scores, and $s(n) = 1$ signifies a mean of 1 in error-scores. So then if this $H$-value is to be used to adjust the nodes's frequencies with, the frequency will then be adjusted in a certain direction and amount (specified by $\rho(n)$) — given that the oscillator is *enough* "out of synch"/"unsynchronized" (in the case $s(n)$ is considerably larger than 0).

The H-value says something about how much out of phase the oscillator was at the time the oscillator's $n$th "flashing"-signal was perceived (and then followingly how much it should be adjusted, as well as in which direction after having been multiplied together with a sign-factor $\rho(n)$), given then that this H-value also consists of the *self-assessed synch score s(n)* — which again simply was the median of error-scores.

We could look at this $H$-value as representing the direction and amplitude of the frequency adjustment weighted by the need to adjust (due to being out of synch) at the time of hearing "fire"-/"flash"-event $n$. Or in other words, this $H$-value is then the $n$-th contribution with which we want to adjust our frequency with.

Especially interesting cases are when we have $\phi(n) \approx 0.5 \implies \rho(n) \approx \pm 0$, as well as the last $m$ Error-scores $\epsilon(n)$ being close to 0, also leading to $s(n) \approx 0$. In both of these two cases the entire frequency-adjustment contribution $H$ would be cancelled out, due to harmonic synchronization (legally hearing a "fire"-/"flash"-event half-way through ones own phase) in the first case, and due to not being out of synch in the latter (having low Error-Measurements). Cancelling out the frequency adjustment contribution in these cases is then not something bad, but something wanted and something that makes sense. If these $H$-values then are cancelled out or very small, it is indicative of that nodes are already in *harmonic synchrony*, and hence should not be “adjusted away” from this goal state. On the other side, if these $H$-values then are different (e.g. closer to -1 and 1), it is indicative of that nodes are not yet in *harmonic synchrony*, and that they hence should be “adjusted closer” to the goal state.

**The final step: the frequency update function,** $\omega_i(t^+)$ Now, we can pull it all together, for Nymoen et al.’s Frequency Adjustment approach for achieving harmonic synchrony with initially randomized and heterogenous frequencies.

When an oscillator $i$ has a phase-climax ($\phi_i(t) = 1$), it will update/adjust its frequency to the new $\omega_i(t^+)$ accordingly:

$$\omega_i(t^+) = \omega_i(t) \cdot 2^{F(n)}, \tag{1.9}$$

where $t^+$ denotes the time-step immediately after phase-climax, and $F(n)$ is found by:

$$F(n) = \beta \sum_{x=0}^{y-1} \frac{H(n-x)}{y}, \tag{1.10}$$

where $\beta$ is the frequency coupling constant, $y$ is the number of heard/received “fire-event”s from the start of the last oscillator period to the end (i.e. the phase-climax, or *now*) — and the rest of the values are as described above.

This $F(n)$-value then, as we see in Equation (1.10), is a weighted average of all the oscillators’s $H(n)$-values accumulated throughout the oscillator’s last cycle.

# Bibliography

[1] Renato E. Mirollo and Steven H. Strogatz. “Synchronization of pulse-coupled biological oscillators”. In: *SIAM Journal on Applied Mathematics* 50.6 (1990). Publisher: SIAM, pp. 1645–1662.

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

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