Oscillation Translation: Kuramoto Odio

Austin Marcus: amarcus6@binghamton.edu
Christian Koertje: ckoertj1@binghamton.edu
Ryan Wing: rwing1@binghamton.edu

May 17, 2022

Abstract

The emergent behavior of how things come together and work as one has been studied under a multitude of models. One such model is the synchronization of many coupled oscillators. The Kuramoto model is a simple dynamical system of oscillators on a network which are nonlinearly coupled to produce emergent behavior. For some simulations it is easy to visually discern the process of synchronization happening, but we wanted to seek a method to hear the synchronization. Sound is simply oscillations of air due to pressure waves, so we explore a couple methods to 'listen' to the simulation. In this paper, we provide a discussion of the model and some simulations along with a couple methods we propose to translate the simulations into audio.

1 Introduction

Oscillators are any system that repeats it's dynamics after some finite time. A simple example is a metronome. When one sets a metronome, they move the pendulum to one side and let it go after which the bar will swing back and for clicking at every beat. Metronomes can be easily coupled by placing them on a board that is allowed to move, say on wheels. Each oscillation will in turn force the others by just a little due to this coupling and the onset of synchronization occurs.

Setting an array of these off in random order results in a mass of clicking that can be hard to deduce much from. With coupling, and a long time frame, the clicking will eventually line up until we are left with essentially one collective click. The transient behavior will be represented by a mass of clicks much like the start of the system so we cannot hear how the synchronization is occuring.

This work studies the coupled oscillator model presented by [1], which is effectively a number of Kuramoto oscillators connected in a regular 2D grid pattern. This model produces interesting transient dynamics when started from a random initial condition, as the oscillators influence each other's phase and effective frequency. Given that the equations governing these dynamics are quite simple, yet the dynamics are intricate and unexpected, this work was

motivated to analyze the model in several ways. First, numerical simulations are explored to characterize dynamic pattern formation. But also, since the phase of the oscillators is depicted by color, other ways of perceiving the dynamics were sought out. Thus, a method to convert the dynamics of the model into an audio representation is developed. Lastly, continuing with this representation, the phase alignment of oscillators is extracted using spectrogram visualizations.

2 Methods

2.1 Kuramoto Model

The model we are working with is known as the Kuramoto model for coupled oscillators. For a network of N oscillators, the model is given by

$$\dot{\theta}_n = \omega_n + K \sum_{m \in U} \sin(\theta_m - \theta_n), \qquad n = 1, 2, \ldots, N \tag{1}$$

where $\theta_n = \theta_n(t)$ represents the phase of oscillator n at time t. The dot notation stands for the derivative with respect to time. On the right hand side of Eq. (1), ω_n is the natural frequency of each oscillator typically taken to be all the same. The last term is the coupling interaction between oscillators θ_m and θ_n with coupling constant K.

Eq. (1) is generalized for any kind of network model as long as U stands for the neighborhood of connected oscillators. In this paper, we focus on the topology for which the oscillators are arranged in a 2-dimensional square lattice such that the neighborhood is defined by the 3x3 Von Neumann grid roughly simulating a spatial interaction. Because of this relation, we can connect the model Eq. (1) to the following nonlocal continuous field model

$$\frac{\partial \theta}{\partial t} = \omega + \int_{\mathcal{R}} K(x - y) \sin(\theta(x - y, t) - \theta(x, t)) dy \tag{2}$$

in which θ and ω are now functions of space. The interaction term is replaced by a nonlocal diffusion term with coupling function $K(\theta)$. Eq. (2) has been studied as a model for brain wave dynamics in the motor cortex [2] showing a wide variety of dynamic pattern formation. We recover the same dynamics as Eq. (1) by letting the interaction radius $\mathcal{R}$ be infinitesimally small such that the diffusive interaction is local. By sticking to the local scenario because of computational intensity, simulating Eq. (1) results in dynamic patterns called spin wheels seen in Figure 1.

2.2 Kuramoto Becomes Odio

Audio is fundamentally very simple. Waves of specific frequencies at specific amplitudes. That's it. A sound source inherently generates both values, which a 'listener' (of whatever type - person, microphone, etc.) receives and interprets. Therefore, after an audio signal is received and recorded, the only parameters available to manipulate are time (pitch, reverb, etc.) and volume (relative, mean, etc.). An oscillator behaves essentially identically to an audio source, so in principal translating their behavior to audio is straightforward. However,

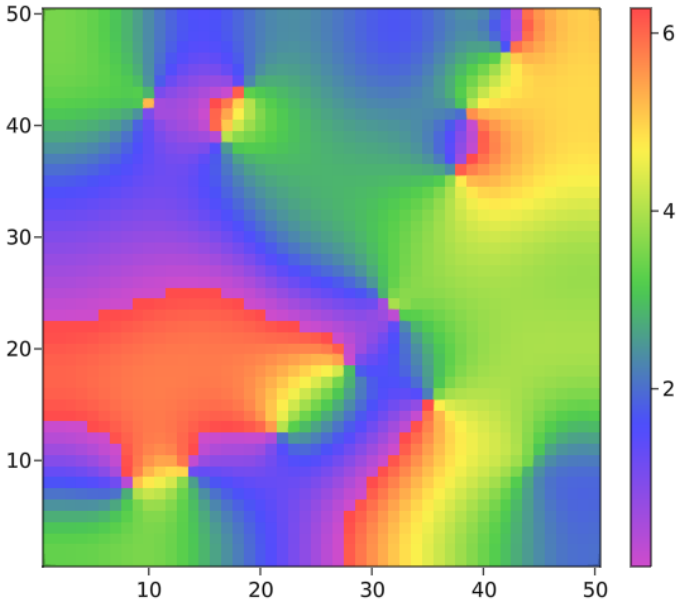

Figure 1: Simulation results with the natural frequencies all the same at 5 rad/sec and free boundary conditions. We can identify by inspection where the spin wheels are located (e.g. at (32, 14)).

audio engineering doesn't typically involve 16,384 coupled 'sound sources' that all need to mixed down to an accurate representation of their collective behavior every 0.01 seconds. This presents a challenge that involves many variables to consider in the design and creation of any optimized solution. We share two methods that each produce a result that seems to correlate reasonably well with the visual representation of the model's behavior. When designing these methods, we wanted to faithfully represent this particular model's behvior, but we also saught to keep our methods not overly specific to *only* this model with the goal of making progress towards a method that might be able to generalize reasonably well to additional models as well. Each of the audio generation methods we implemented are discussed below.

Note: After some debate, we chose to use MIDI as an audio output rather than timeseries audio values. Time-series data (after interpolation) would offer the advantage of being a more smooth, continuous oscillation but would be limited to 'pure tones' i.e. sound with no 'shaping' to it, which humans largely don't enjoy. MIDI has the disadvantage of outputting distinct notes at every timestep and thus acts as though the sound source is starting and stopping as each MIDI note is triggered. However, MIDI offers the advantage of being able to be rendered via any instrument capable of receiving MIDI data as input, and thus more flexible in terms of how the audio ultimately sounds. We chose sanity over continuity.

Also, in MIDI terminology, 'velocity' essentially corresponds to volume when recording a note value. For example, if you press a piano key with a high relative velocity, you get a louder note than a key pressed with low relative velocity.

2.2.1 Quadrant Mean Frequency Method

The first method we implemented to translate the model's values to audio was built primarily as a proof-of-concept to get an initial idea about how audio generated from the Kuramoto model would sound.

Frequency

To try to generate audio that was at least somewhat representative of model behavior while emphasizing simplicity, we adopted an admittedly very coarse-grained method of dividing the model into equal quadrants, calculating the mean value of each quadrant at each time-step, and translating each quadrant's mean to a single MIDI note within a range of all notes in the CMaj scale over two octaves (14 notes). The oscillators' values move through the range(0, 2π), so this was divided into 14 bands of equal width, each corresponding to a note, ascending by pitch.

Velocity

Given the broad range of notes used for frequency variation and our desire for simplicity, we held all velocity values constant at 100. (MIDI velocity uses the range 0-127.) As this produces no relative amplitude differences between notes, frequency is the only variable representing model behavior change in this method.

At every time-step, for each quadrant, a single note is output with static velocity values of 100.

2.2.2 Amplitude Variation of Chords Method

Seeking to improve the granularity of how the model's values were being represented as audio, we designed another method based on the idea of treating the model as a room, split into equal quadrants, with a microphone at the center of each quadrant listening to every oscillator within its respective quadrant.

Frequency

At every time-step, every oscillator's value was classified as one of four notes in a MajC7 chord - C, E, G, B. We used note values for C and E from the third octave and note values for G and B from the fourth octave for greater tonal width, which results in audio that we subjectively deemed easier to differentiate than all notes in a single octave. The oscillators' values move through the range(0, 2π), so this was divided into quartiles corresponding to a note in ascending order.

Velocity

To simulate our microphones in the middle of each quadrant, we generated another 128x128 matrix and then for each quadrant we calculated each index's Euclidean distance from the center of it's respective quadrant, which was then recorded as that index's value. The resulting 'mic map' is shown in Figure 2.

The mic map's values were then used to set velocity values for each corresponding note as determined in the Frequency section above. Thus, we essentially had quadrants of 4,096 instruments each playing one of four notes at varying volumes based on the instrument's proximity to the microphone. All velocity values corresponding to each note are then summed, divided by the total of all values in a 'mic map' quadrant, and then multiplied by 127 to assign a relative velocity value for that note, in that quadrant, at each time-step.

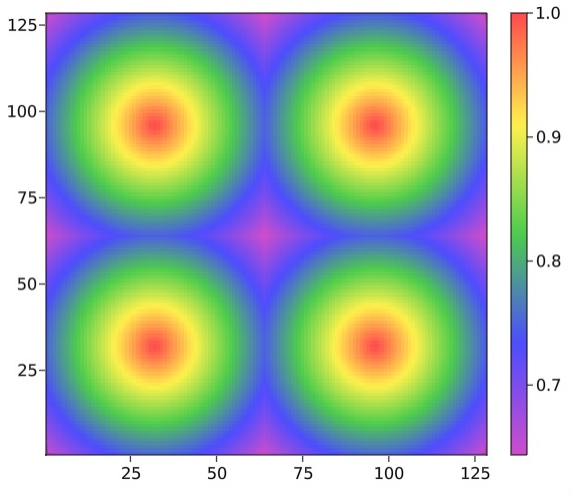

Figure 2: Visualization of 'mic map' matrix simulating microphones placed at the center of each quadrant of the model. Used to vary MIDI velocity values based on proximity to the microphone.

At every time-step, for each quadrant, single C, E, G, and B notes are output with their dynamically assigned relative velocity values.

2.3 Signal Analysis

While the base frequency of the oscillators, ω, does not change over time in the model, the effective frequency at which an oscillator rotates through its phase will change and differ from ω under certain conditions. This is because the oscillators tend to align their phases, which means that at least one oscillators effective frequency must change at least transiently. However, it is hard to observe this dynamic when viewed as point whose color changes in time. Thus, in order to better visualize this transient change in effective frequency as oscillators align phase, two visualization methods from signal analysis were employed.

A spectrogram is a well-known method used to visualize an audio or other type of signal. It works by taking a short-time Fourier transform (STFT) on each chunk of a divided digital signal, which is then plotted for each chunk vertically. Thus, the x-axis is time, the y-axis is frequency, and power at a given frequency is represented by color. These visualizations are an effective way to understand the component frequencies of a time-varying signal. Python's "scipy" library implementation of a spectrogram was used.

In order to observe the *difference* between two signals, the signals were subtracted from each other and then plotted in the time-domain, where the x-axis is time and the y-axis is amplitude. Thus, phase alignment can be seen as a reduction in the amplitude.

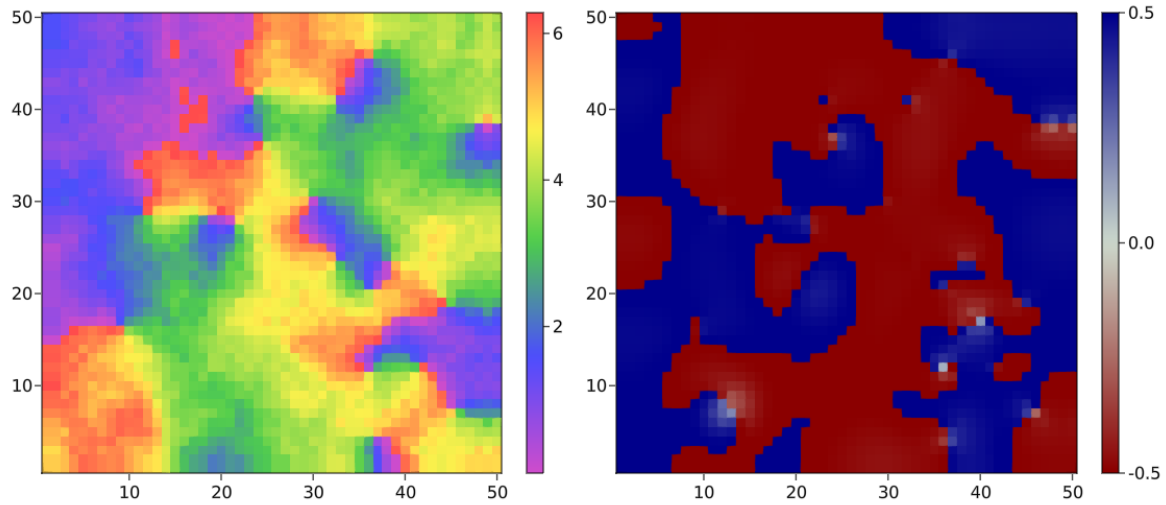

Figure 3: Simulation results with the natural frequencies being distributed around a standard normal distribution with free boundary conditions. The plot on the right is the instantaneous frequency field showing how nearby certain frequencies are to each other.

3 Results

3.1 Simulation

Figure 1 shows our results of the replication of spin wheels in the Complexity Explorable [1]. It is interesting to note that the spin wheels attract to each other, much like oppositely charged particles, and annihilate upon collision. This indicates over long time periods that all spin wheels will converge and total synchronization will be met in this system. It is not known whether any distribution of natural frequencies will sync with this topology, but there is a study based on the basin of synchronization done by [4, 3].

Another interesting simulation results is shown in Figure 3 where the natural frequencies are randomly distributed between positive and negative values. This result does not easily show the existence of spin wheels although it is believed they exist [3]. The inclusion of the instantaneous frequency field is just one way to show the processes of synchronization is occurring even though the dynamics occur on fairly slow time scales.

3.2 Quadrant Mean Frequency Method

This method, for all its simplicity, surprisingly produced a result that seems to correlate reasonably well with the visualization. We were really pleased and encouraged by this type of result from what was only intended as a proof-of-concept version.

3.3 Amplitude Variation of Chords Method

This method increased the amount of computation required to generate note values fairly significantly. The additional computation did result in much more distinct audio representations of each quadrant's behavior.

While we were pleased with the increased granularity of detail in the generated audio, this method also demonstrated that generating 16 notes at every time-step (4 per quadrant), even at varied amplitudes, seems to indicate that more notes per time-step would have a very sharp diminishing returns to scale curve. The audio needs to be helpful to the listener in distinguishing patterns of model data. If the model data is overly compressed into overly-simplistic audio, it risks being unrepresentative. If there are too many simultaneous patterns occurring, the result could simply be overwhelming. There is very clearly a careful balance to strike in the compression ratio of model data:audio representations.

Videos and generated audio for both methods can be accessed at: `https://osf.io/pn8fb/`. See the ".mov" files in each folder. *Each quadrant's audio is panned in stereo space, so headphones are helpful to better hear the spatial representation of the model's behavior.*

3.4 Observing Phase Alignment

In order to observe the phase alignment dynamic, two adjacent oscillators from the 2D grid were examined. Their phase values over time were recorded and visualized using the methods described in Section 2.3. Phase values were converted to a smooth rotating signal by taking the sine of each value. Figure 4 shows the result of subtracting one signal from the other. Spectrograms of both signals were produced and are shown in Figure 5. Note that the dominant frequency has an initial transient before settling down at a stable value. This is depicted by the yellow line in the spectrograms. Paralleling this transient, the amplitude of Figure 4 starts high and decreases to a stable value: this indicates that the oscillators become closer in phase, which makes their waveforms nearly cancel.

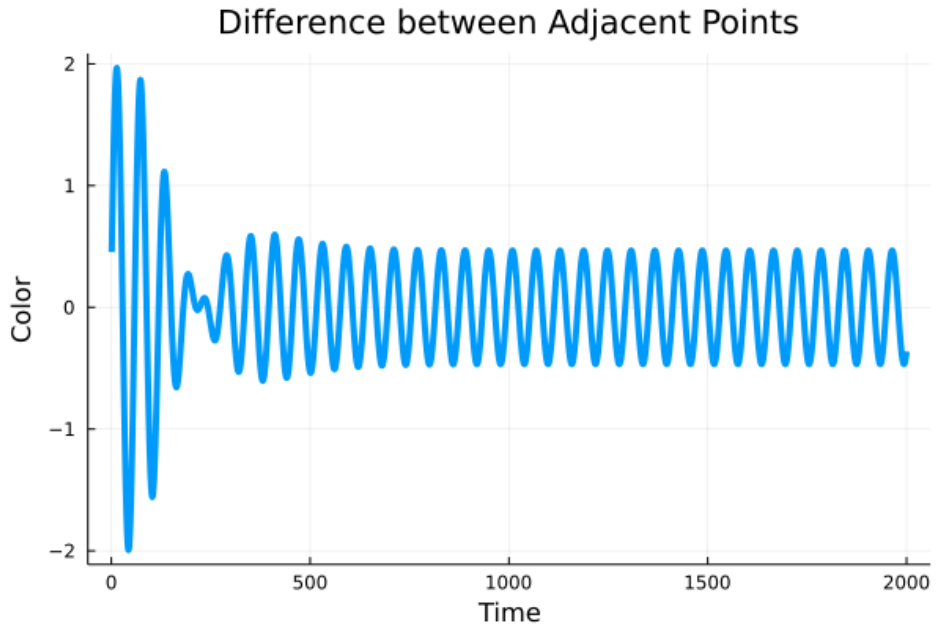

Figure 4: Time-domain plot of difference between signals. After a short transient, the difference converges to an oscillatory attractor.

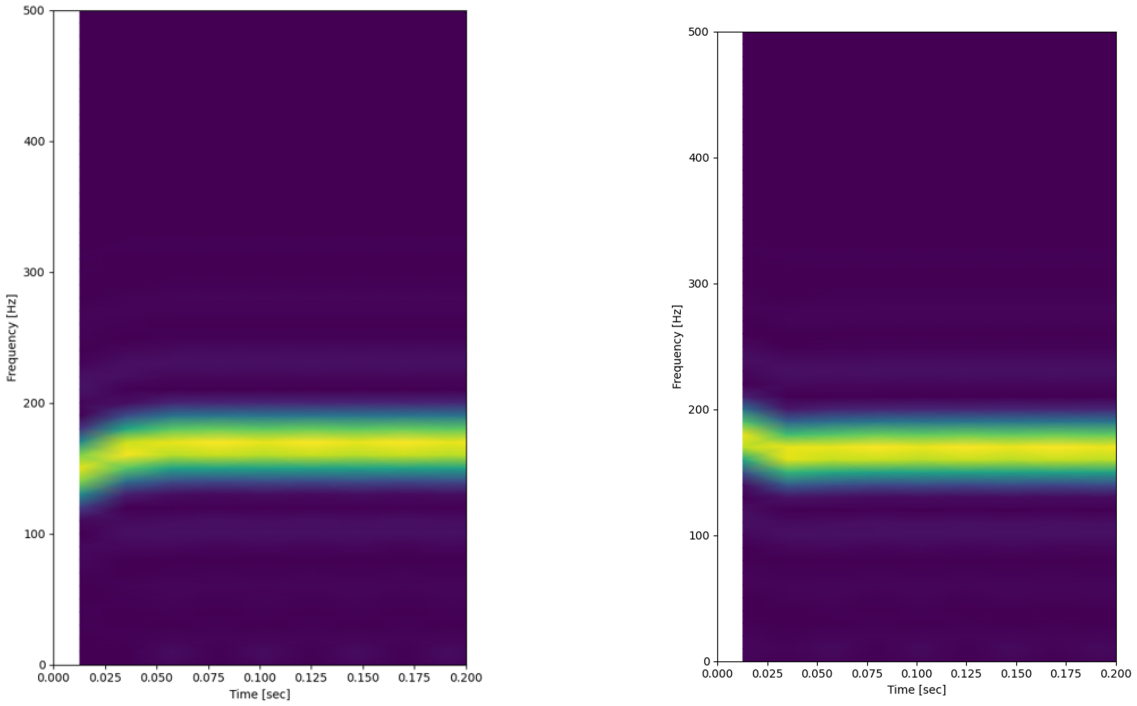

Figure 5: Spectrograms of adjacent oscillators.

4 Discussion

Through computer simulations, we showed how the Kuramoto model Eq. (1) can exhibit nontrivial dynamic pattern formation including spin wheels and synchronization. We then proposed a couple of qualitative methods on how to translate the data into audio, one of which performed slightly better than the other. Lastly, we provided a signal processing analysis to show the behavior of nearby oscillators an how they align their phase.

The audio methods developed are still quite coarse-grained representations of the system's dynamics. The challenge with representing a spatial model with sound is making the spatial detail detectable in sound. While the methods used here do not preserve the location of the dynamics with much precision (despite each quadrant being panned in stereo space to match its relative location in the model), it may be effective in characterizing the system at a larger scale. For instance, in the means method, as the simulation progresses towards a single, pure-tone that gains in relative volume and begins to dominates the soundscape; this corresponds to what can be seen visually, namely, that more oscillators have synced their phase. Similarly, in the chords method, distinct oscillation patterns emerge in each quadrant's audio output. Over time, there is a clear convergence towards a more globally synced oscillation as locally dominant oscillation patterns begin to sync and merge. Thus, larger-scale qualitative dynamics are captured to some degree with the audio methods, but spatial details are not.

One possible method to improve the representational accuracy and spatial precision might be to implement a method that can identify individual 'pinwheel' patterns in the model and dynamically represent them as individual audio sources. This would remove the arbitrary nature of quadrant divisions in the model and likely result in audio that allows for more accurate identification of key sources of interest in the model's behavior.

Lastly, the spectrogram approach to observing the moment of oscillator synchronization complements the audio approach because it shows the dynamics in more detail, but would be difficult to interpret on a larger scale. Thus, this method allows one to "zoom in" on the dynamics. This could be used as an aid to connect audio to an explanation in terms of the variables of the model.

4.1 Future Work

There exists a simple order parameter for the characterization of synchronization known as the phase coherence. It is generally used to study the all-to-all topology, so it would be interested to picture a phase correlation field in a similar light. The continuous field model Eq. (2) can also show a larger variety of dynamic behavior by changing the interaction function. Future studies would be warranted to see how those patterns sound in comparison to the spin wheels.

It would be interesting to explore additional methods for transforming model data to audio representations. Perhaps if there might be a method (or limited set of methods) that could generalize to multiple types of models while still producing meaningful results. That route seems overall preferable to a method that might be highly accurate for a single model, but not applicable to any additional models. (Which is not to say that methods of high-accuracy for individual models would not also be very interesting!) A generalizable method/s might be a valuable tool to incorporate into data visualization packages, especially if audio representations could be activated with simple method calls to generate real-time audio alongside visualizations. This could be a great accessibility feature for people who are visually impaired as well as potentially helpful for anyone as another way to get a better sense for model behavior (pun intended!).

Additionally, our audio generation method does not produce real-time audio that is synced with the visualization, which would be a significant improvement that could be made. Similarly, as the audio is not real-time, this method requires the MIDI files to be read by a synthesis instrument to generate sound, which then must be manually added as an audio track to the visualization video file. This is obviously not ideal and could be greatly improved upon. Finally, the ability to listen to each quadrant's audio in complete isolation from the others is also very helpful in more clearly deducing model behavior. It would be a substantial improvement if a real-time audio generation method included the ability to 'solo' individual channels (corresponding to model quadrants).

A Code

All project code and associated files are available at our project's OSF page (https://osf.io/pn8fb/).

References

[1] Dirk Brockmann. Spin wheels. *https://www.complexity-explorables.org/explorables/spin-wheels/*.

[2] Stewart Heitmann, Pulin Gong, and Michael Breakspear. A computational role for bistability and traveling waves in motor cortex. *Frontiers in computational neuroscience*, 6:67, 2012.

[3] Bertrand Julien Ottino-Loffler. Synchronization unlocked: spirals, zetas, rings, and glasses. 2018.

[4] Daniel A Wiley, Steven H Strogatz, and Michelle Girvan. The size of the sync basin. *Chaos: An Interdisciplinary Journal of Nonlinear Science*, 16(1):015103, 2006.