A Numerical Approach to Understanding Oscillator Neural Networks

Natalie Klein
Mentored by Jon Wilkins

Networks of coupled oscillators are a form of dynamical network originally inspired by various biological and physical systems. Much existing research focuses on simple, restricted models which are analytically tractable. In order to explore networks which do not fit into the existing models, we have programmed a flexible numerical simulation of oscillator networks. We have used the simulated network as an artificial neural network which uses a genetic algorithm for training. Preliminary results with the current neural network simulation suggest that the genetic algorithm is able to produce and optimize networks which perform differentiation between input frequencies. In future work, we hope to expand the existing numerical model, explore the role of noise during training on robustness, look at the effect of different network architectures, and explore optimization of neural networks for other tasks.

I. Introduction

Oscillator networks are an example of a dynamical network which can exhibit synchronization phenomena, a quality thought to be essential to the functioning of many biological systems. As a result, artificial oscillator networks are inspired largely by existing biological and physical systems. Real networks of neurons exhibit network structures with synchronization and oscillatory phenomena. Macroscopic biological systems such as groups of fireflies also exhibit periodic and synchronizing behavior. In addition, physical systems such as Josephson laser arrays behave much like lattice oscillator networks.

For this reason, artificial oscillator neural networks can be seen as valuable for replicating and possibly even understanding their natural counterparts. In addition to this aim, using the networks as artificial neural networks becomes both practically and theoretically interesting as a machine learning algorithm. These types of artificial oscillatory neural networks have been used in the real world primarily for visual scene recognition, as some evidence suggests that the visual cortex may be very similar in structure and mechanism to oscillator neural networks. Another study used an oscillator neural network to recognize and differentiate double vowel sounds.

Aside from the few published practical, neural network approaches to oscillator networks, much of the published research into general oscillator networks seeks closed-form, analytically tractable descriptions of the dynamics. In almost all cases, this requires the modeler to make simplifying assumptions and place tight restrictions on the network properties. Many of the papers restrict the networks to chain, lattice, or globally-coupled structures, require the oscillators to have nearly identical initial properties, constrain the connection strengths, and forbid asymmetrical connections. This may be detrimental to overall understanding of oscillator networks because some research suggests that unusually-formed networks exhibit quite different behavior from the regularly structured networks usually studied. In addition, much of the literature focuses mainly on synchronization as the only valuable property of the network dynamics. However, even the simple versions of oscillator

neural networks can exhibit much richer behavior than simply global synchronization.

II. Goals

Driven by the idea that there is much more to oscillator neural networks than can be captured in the present analytical models, we seek to produce a flexible numerical simulation of oscillator neural networks which can be used to observe network behavior. Our numerical model is extremely flexible, allowing asymmetric connections, various distributions of asymmetric or symmetric connection strengths and oscillator properties, and any network structure desired. Once this model is established, we aim to extend this model into a neural network framework and to train the neural network on various tasks and observe its performance. The model is numerical and therefore will not be exact, but we believe it will be a close enough approximation to the “ideal” oscillator network to offer valuable observations. An important aspect of the numerical approach is that we are not attempting to numerically solve the phase equations which are usually the form of the analytical models. Instead, we are simply simulating physical oscillators using a modified version of the damped wave equation. Any global behavior will be extracted numerically afterward from the data. Our future goals with the numerical simulation include examining the effect of noise during neural network training on robustness, testing the effect of various distributions of initial network properties, investigating the effect of changing the network topology, and allowing the neural network to change the network topology as it trains to improve performance. We also want to look into applying the neural network to various real-world tasks such as sound recognition or even molecular modeling.

III. Implementation

Computer-based numerical simulations are only able to calculate and store values to certain accuracy and the numerical computation techniques are often slightly inexact. Therefore, some general considerations must be taken when constructing a computer-based numerical simulation. Especially important is the consideration of the effect and quantity of error due to both machine precision and approximations. In order to appropriately account for error, we must consider all sources: machine precision error, error in estimation techniques, and the fact that the network itself could be extremely sensitive to small perturbations in the data. Preliminary measures we took were to use the most accurate estimation techniques available, and to use techniques where we can specify an appropriate maximum error tolerance as needed. The first test was to use input conditions on simple networks where the behavior was easily predictable and to compare the numerical with exact results. We also did some experiments where we ran simulations with very similar, but slightly perturbed, initial conditions to see whether this caused vast error in the results. So far, we have not seen any reason to believe numerical error is a significant factor, but more inquiry into this is needed. We plan to continue testing the similarity of results with varied floating point precision and to test more sets of data that are only slightly perturbed from each other to see how stable the results are.

Because some elements of the simulation draw upon “random” (technically pseudo-random) distributions, another important factor to consider is the necessity of repeatability since a pseudo-random number generator is used. To ensure repeatability and complete control even when using pseudo-random elements, we allow the user to explicitly set the pseudo-random number generator seed value and provide a known default seed value if none is set. All uses of a pseudo-random number generator in the simulation will be seeded exactly

the same so that any given run of the simulation is exactly repeatable.

The details of the numerical simulation can be broken up into separate components: the standalone network-only simulation and the neural network simulation, which includes a training algorithm for which we used a genetic algorithm.

In the network simulation, the main data fields are the node data and the weighted connections matrix. The node data includes a starting position, velocity, and acceleration, as well as a constant natural frequency term and a constant damping term. The connections matrix is not required to be diagonal, which means that asymmetric connections are accepted. Any distribution of connection weights, both positive and negative, are acceptable. When the simulation is run, each node propagates its current position state to each of the nodes it is connected to. This is done for all nodes before any of the next time step's states are calculated. Once all of the states are calculated, they are set and the simulation time is advanced. In this way, the update rule is synchronous and all oscillators move from one time step to the next at the same time.

The time step is determined using the Nyquist-Shannon sampling theorem, which for a wave's frequency, gives an appropriate sampling rate that would give data accurate enough to completely and uniquely reconstruct the wave. Essentially the theorem requires that $T \leqslant \frac{1}{(2f)}$ where *T* is the time step and *f* the frequency. The simulation uses the maximum frequency present in the initial configuration to determine the time step. The number of time steps to evaluate the network to get to a steady state is determined by a tunable parameter which is multiplied by the largest period of oscillation present in the initial configuration. This is a property that needs to be tweaked for various networks until a better, more universal measure may be discovered.

The oscillators themselves are currently modeled as harmonic oscillators following a modified standard wave equation. The differential equation describing node i is:

$$\ddot{x}_i = -a_i x_i - b_i \dot{x}_i + \sum_{j \in N} c_{ij}(x_i - x_j) \quad ,$$

where *N* is the set of nodes connected to node *i*, *c* is the weighted connection strength, *a* is the square of the inherent frequency, and *b* is the damping constant. We used an open-source implementation of the Runge-Kutta-Fehlberg method, which uses approximations of order 4 and 5 and a variable step size to approximate closely the solutions to the wave equations. When the output nodes are evaluated after the system has been run for a specified amount of time, the Fast Fourier Transform is used and appropriately scaled to extract the dominant frequency and amplitude of the output wave. In addition, the beginning third of the data is ignored when calculating the dominant frequency and amplitude to avoid the effect of transient behavior.

To apply the network simulation as a neural network, one needs only to place a few restrictions on which nodes are “input” nodes and which are “output” nodes, on the nature of the connections between the nodes, and provide a training algorithm. A genetic algorithm was chosen as the search algorithm for various reasons. The traditional approaches to back-propagation algorithms require a differentiable function in order to determine the “slope” of the error at a certain point in order to determine how to change the solution to decrease error. This also requires knowing exactly the impact of changing parameters on each node on the node's output. While this is practical and simple for simpler neural networks, it seemed not to be the best solution for oscillatory neural networks. Therefore, we wanted to use a more general solution space search algorithm. A genetic algorithm was chosen because it is suited well to finding one good solution to a problem, even if it is not necessarily the optimum solution. In our current applications, we are simply searching for solutions which fit a fairly

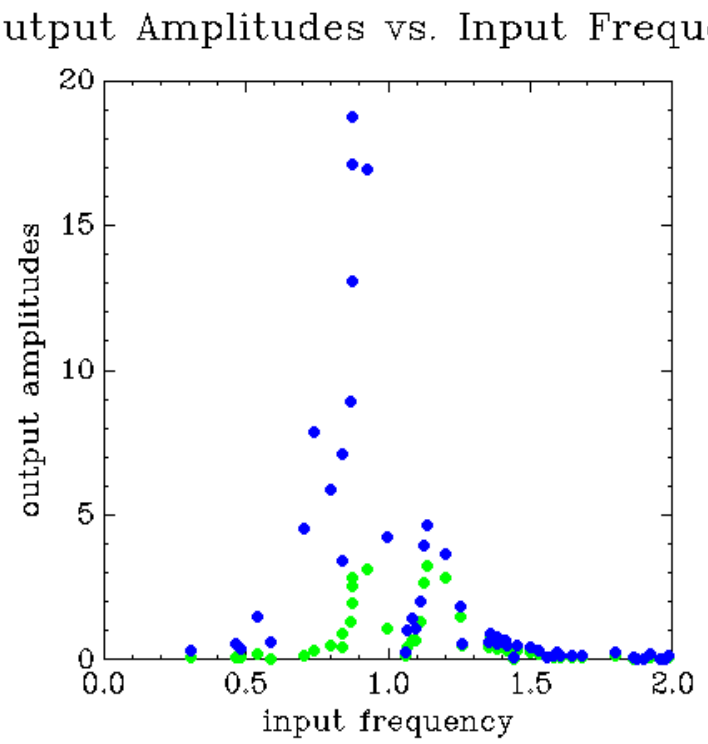
general criteria and not concerned with finding the global optimum. The chromosome used in the genetic algorithm contains the network's node data and connections matrix. The evolutionary part of the algorithm does not use crossover but simply selection, replacement, and mutation. The initial population is chosen by adding random noise to a chromosome that the user has passed in. This allows the genetic algorithm to have a reasonable starting point that can also help direct what kinds of results are desired. When it is time for selection, the genetic algorithm must first determine the fitness of each chromosome. The fitness in the currently developed genetic algorithm is determined by taking each chromosome and creating and running a full simulation of the oscillator network with the data in the chromosome over a range of input values. The current fitness function evaluates the output of each network and looks for the network which produces the most different output over the various input values. This is determined by creating vectors containing the output amplitude data for each output node over the inputs, so that a system with *n* inputs will yield an *n*-dimensional vector for each output node. These vectors are then placed together as columns of a Gram matrix we will call *A*. By evaluating $\left|det\, A^T A\right|^{1/2}$, we get a measure of the volume of the parallelotope (*k*-dimensional parallelogram) formed by the vectors. This volume is maximized both when the amplitudes of the output vectors are large and when the outputs of the various nodes are the most different (or the vectors are closest to orthogonal). Therefore, the chromosomes with the greatest volume in the generated paralellotope are the most fit chromosomes. Once the fitnesses are determined for all the chromosomes in the population, the least fit are replaced by copies of the most fit. Then the entire population undergoes mutations with a certain chance that any given chromosome will mutate. Right now the only data that mutates is the inherent frequency and the connection strength, but we hope to allow greater mutations of the connections and topology in the future. The chance of mutation and the radius of mutation can be controlled by the user. This process is repeated over and over again for a certain number of generations, and then the most fit chromosome is returned.

The simulation offers many opportunities for a user to store any data along the way and to view graphs of the oscillator behavior, the outputs over time, or the fitness over time. More graphing and data manipulation will be added in the future. Another major goal is to distribute the computation using parallel processes which is currently being undertaken. This will allow larger and more intricate networks to be computed in more reasonable time spans.

IV. Results

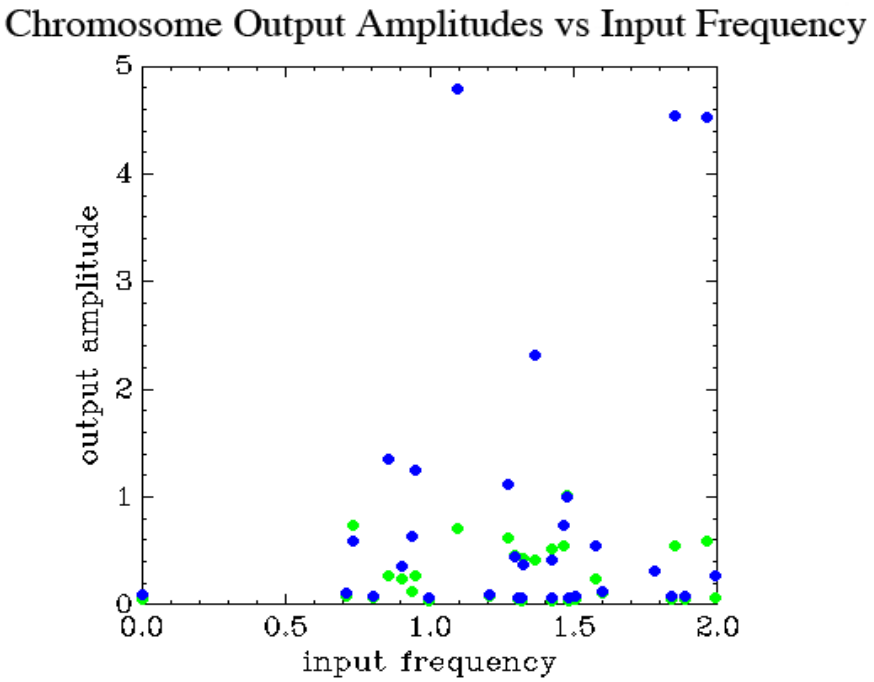
Due to the amount of time required to get up to date on the current research and implement a solid, flexible simulation, the results as of right now are preliminary.

We were able to implement a simulation of an oscillator neural network as described above, though there are many things we hope to improve upon in the future. Our first goal with the current genetic algorithm and fitness function was to look for networks which are good at “frequency differentiation”. These networks will show substantial differences between their output amplitudes as different inputs are given to the network. We started with a 5-node neural network that had one input node, two middle-layer nodes, and two output nodes. By hand, we experimented until we found a network which appeared to exhibit some of the behavior we were looking for. This was the result we found by hand for a “frequency differentiation” network.



The graph shows the amplitudes of the two different output nodes, where one output is designated by blue dots and the other by green dots. Clearly, the blue output node's amplitude is very different from the green node's amplitude around a certain input frequency (approximately between 0.8 and 0.9). This network could be thought of as being able to identify a certain input frequency by triggering this large difference in the output amplitudes at that input frequency.

In order to explore the solutions near this one that might better accomplish the goal of frequency differentiation, we input the above hand-picked network into the genetic algorithm as the starting condition. One of the results is shown in the graph below.



This graph shows a less severe spread between the two output nodes at any one point, but does show a more marked difference between the nodes' amplitudes over the entire interval of input frequencies.

These results lead us to believe that there is rich “frequency differentiation” behavior which can be exhibited by the oscillator neural networks. More exploration and time is needed to discover possibly more interesting or useful cases, but the results we have found so far exhibit the behavior we were looking for to some degree and therefore are promising.

V. Conclusion

Oscillator neural networks are a fascinating concept which offer rich applications and implications in the understanding of biological processes of synchronization. While much research has been done in the past ten years, there is still much to explore. Through this numerical simulation, we will be able to continue to explore possibilities which are not possible through traditional analytical means. In particular, we want to parallelize the simulation to allow more computationally feasible exploration of larger networks and greater solution spaces to search. We hope to explore further the effect of noise on the neural network training, the effect of changing inherent network properties, and the implications of allowing the genetic algorithm to modify the network structure as it searches for a better solution. This paper is meant to serve as a somewhat informal explanation of the methods used to construct the numerical simulation, the preliminary results we have found so far, and the goals we hope to achieve in the future as we continue work on this project.