Neuron-modeled Audio Synthesis

Jeff Snyder
Princeton University Music Department
310 Woolworth Center
Princeton, NJ 08544
josnyder@princeton.edu

Aatish Bhatia
Princeton University Council on Science and Technology
233 Lewis Library
Princeton, NJ 08544
aatishb@princeton.edu

Mike Mulshine
Princeton University Music Department
310 Woolworth Center
Princeton, NJ 08544
mulshine@princeton.edu

ABSTRACT

This paper describes a project to create a software instrument using a biological model of neuron behavior for audio synthesis. The translation of the model to a usable audio synthesis process is described, and a piece for laptop orchestra created using the instrument is discussed.

Author Keywords

neuron, biological model, software, musical instrument

CCS Concepts

•**Applied computing** → **Sound and music computing;** Performing arts; Computational biology;

1. INTRODUCTION

While the majority of audio synthesis methods aim for powerful and clear control over sound parameters, some approaches eschew this control in favor of a more experimental and serendipitous outlook. One example is seen in the barely-controllable instrumental systems of David Tudor, the performance of which he described as "an act of discovery. I try to find out what's there and not to make it do what I want but to, you know, release what's there." [6]

The project described is one path toward this type of musical interaction, in which a biological model is used as a synthesis technique with intent to discover exciting and unexpected sonic possibilities.

2. MOTIVATION FOR NEURON SYNTHESIS

This idea came about when a member of the Princeton Laptop Orchestra (PLOrk), Mitch Nahmias, was explaining how the lab he works at in the Electrical Engineering department used biological models of neurons to create optical computers. He mentioned that, under certain conditions, these neuron models could be coaxed to oscillate, and he mused that it might sound interesting to use this as a basis for audio synthesis. Together, Jeff Snyder and Nahmias created a quick prototype using the simplest possible neuron model in the programming language ChucK[11], and it made sound! We promptly lost that work in a computer crash, and dejectedly set the project aside for another day.

NIME'18, June 3-6, 2018, Blacksburg, Virginia, USA.

Two years later, the concept came up again in discussions with physicist Aatish Bhatia, and he was also curious about how these models might sound. Bhatia and Snyder worked on researching the existing models to explore, and Bhatia handled the mathematics of turning the differential equations into a real-time solvable algorithm. Once they had a working synthesis algorithm, Snyder brought Mike Mulshine into the project to work on integrating the algorithm into a usable instrument.

The expectation was never that this synthesis technique would sound dramatically different from traditional methods, but rather that it would provide an interesting alternative to synthesis methods that have easily understood control parameters (pitch, amplitude, etc.). We hoped that by introducing control parameters relating to physical processes that are unfamiliar, such as "sodium channel activation", a more experimental approach to digital synthesis would be opened up to the user.

3. PRIOR ART

This project is in the tradition of physical modeling synthesis, where mathematical models of real-world systems are simulated for sound creation[3][10]. Much of the research into this area so far has focused on modelling physical systems for which sound production or sound transformation are the typical use cases, such as flute acoustics[2], room reverberation[7], or guitar amplifier simulation[12]. This project falls into a much smaller field: repurposing mathematical models that are not originally intended to describe audio systems for audio purposes. Examples include the use of strange attractors[8] and other creative misuses of scientific research tools.

The prior research with the closest relation to this project is another attempt to use a biological neuron model for synthesis. This is briefly included as part of Nick Collins' paper, Errant Sound Synthesis [1]. Audio implementations of the FitzHugh Nagumo and Termen Wang models of neuron behavior are mentioned and included in the library of SuperCollider code that accompanies the paper. Collins' approach is similar to ours in that he also uses the Euler method to discretize the equations, and includes these algorithms in a publicly available library. We hope that our paper adds more illuminating detail to the process, and a more in-depth discussion of the usefulness of this concept in introducing non-linear control into instruments.

4. ADAPTING THE BIOLOGICAL MODEL FOR AUDIO SYNTHESIS

While looking for an appropriate neuronal spiking model to implement, we researched various alternatives. After reading Izhikevich's review of neural spiking models [5], we decided to implement the Hodgkin-Huxley (HH) model [4].

This is an important classic model in computational neuroscience, comprising of 4 coupled non-linear differential equations, and various biologically inspired input parameters such as the activation and inactivation rates of ion (i.e. sodium and potassium) channels. While the HH model is more computationally expensive than simplified, single-purpose models of neuron spiking, it is also a very general model that can reproduce many different neuronal spiking patterns. It is therefore well suited to our goals of exploring the rich parameter space of neuronal spiking behaviors.

To implement this model, we followed the standard Euler method to discretize the four differential equations. This resulted in a set of four difference equations that determine the membrane voltage V_m of a neuron. The first three equations are similar in form:

$$\begin{aligned} n(t+\Delta t) &= \alpha_n \Delta t + n(t)\big(1-(\alpha_n+\beta_n)\Delta t\big) \\ m(t+\Delta t) &= \alpha_m \Delta t + m(t)\big(1-(\alpha_m+\beta_m)\Delta t\big) \\ h(t+\Delta t) &= \alpha_h \Delta t + h(t)\big(1-(\alpha_h+\beta_h)\Delta t\big) \end{aligned} \quad (1)$$

The variables being updated here are n, m, h, which are quantities between 0 and 1 representing the rate of potassium channel activation, sodium channel activation, and sodium channel inactivation, respectively. The term Δt is a timestep — a smaller timestep results in a more accurate simulation, but is more computationally expensive. The α and β variables are functions of the membrane voltage $V_m(t)$ provided in the original paper by Hodgkin and Huxley [4].

To clean things up a bit, we define the following time constants:

$$\tau_K(t) = \frac{c_m}{g_K n(t)^4} \quad \tau_{Na}(t) = \frac{c_m}{g_{Na} m(t)^3 h} \quad \tau_l = \frac{c_m}{g_l} \quad (2)$$

These time constants depend on the activation/inactivation rates (n, m, and h), as well as on other biologically inspired tunable parameters (g_K, g_{Na}, g_l, c_m). They are used to update the membrane voltage V_m as follows:

$$\begin{aligned} V_m(t+\Delta t) = V_m(t)\big(1-\Delta t(\frac{1}{\tau_K}+\frac{1}{\tau_{Na}}+\frac{1}{\tau_l})\big) \\ +\Delta t\big(\frac{I(t)}{C_m}+\frac{V_k}{\tau_K}+\frac{V_{Na}}{\tau_{Na}}+\frac{V_l}{\tau_l}\big) \end{aligned} \quad (3)$$

There are three new tunable parameters here (V_K, V_{Na}, V_l), as well as an input current $I(t)$ which we are free to vary. Thus, the rate of change of the membrane voltage is governed not only by the input current $I(t)$, but also, via the time constants, on the activation/inactivation rates (n, m, and h), all of which vary over time and, in turn, depend upon the membrane voltage (via the α and β functions). This set of coupled equations thus results in rich feedback loops and non-linear effects, and the dependence of the output voltage on the input parameters is difficult to analytically predict.

To implement this model in code, we execute the following steps:

1. Set the initial membrane voltage $V_m(0)$ and initial values of the activation/inactivation rates $n(0)$, $m(0)$, $h(0)$. We initialized these to zero. Also set the timestep Δt, which we initialized to 1/50.

2. Choose values for the tunable parameters (g_K, g_{Na}, g_l, c_m,V_K, V_{Na}, V_l), and a form for the input current $I(t)$

3. Use equations 1 to update the activation/inactivation rates

4. Use equation 2 to calculate the time constants

5. Use equation 3 to update the membrane voltage. This is the output sound signal.

6. Repeat steps 2-4. The tunable parameters and input current can be varied as needed, allow a user to control the audio synthesis.

Further details of discretizing the HH model (including biologically relevant values of parameters) can be found in various online references [1].

Once we had a working audio synthesis version of the model in ChucK[11], running as a custom class that generates signal per-sample using input parameters from a MIDI device, we set about adapting it to be easier to interface with a variety of platforms. First, we adapted the ChucK code to be a pseudo-object in our OOPS Audio Library[9]. Since we had already integrated our OOPS library into the JUCE framework, this made it simple to export a VST plugin, allowing the neuron synthesis to run inside a DAW, or in Max/MSP using the vst~object. It also allowed us to quickly prototype running the synthesis model on embedded hardware, such as our Genera brain, based on an STM32F7/H7 microcontroller[9].

Figure 1: A waveform of a single neuron spike using the Neuron synthesizer (AC-coupled)

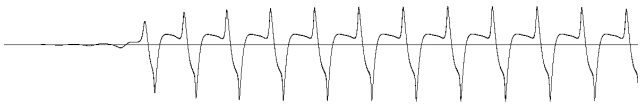

Figure 2: An oscillation waveform created by the Neuron synthesizer (AC-coupled)

5. USE CASE: CONNECTOME, A PIECE USING THE NEURON INSTRUMENT

5.1 Compositional Concept

We wanted to quickly test the Neuron synthesizer's features in a real-world musical setting. Collaboratively with the Princeton Laptop Orchestra (PLOrk), we composed a structured improvisation piece exploring the possibilities of the instrument. This piece is called Connectome, named for the map of neural pathways in the brain. Our goal was to create a piece that was inspired by the functioning of the brain, in which the performers represented individual neurons.

Since neurons pairing with or communicating with other neurons seemed to be a basic metaphor for brain function, we decided to structure the form around "connections" between performers. We decided not to follow the metaphor too directly, to give ourselves flexibility to follow directions that were musically satisfying. In the piece, we used a simple interaction paradigm between players where two players

[1]Introduction to Biological Signal Processing and Computational Neuroscience by Richard B. Wells http://www.mrc.uidaho.edu/ rwells/techdocs/Biological %20Signal%20Processing/Chapter%2003%20The%20Hodgkin-Huxley %20Model.pdf (accessed January 2018)

could "connect" with each other, which meant that they would be listening to each other and playing in a kind of call and response. The form of the piece would allow for players to form and break connections with each other, so that the structure of the connection map would change over the course of the piece.

To facilitate the establishment of connections between players, we added a second instrument besides the Neuron synthesizer to each player's arsenal. We wanted something that would be easily heard in a complex texture, and also visually clear, so that players could look up to see who was making the sound. To satisfy these needs, we chose to use acoustic woodblocks. Each performer has a woodblock and a drumstick at their station, and the signalling of connection/disconnection actions in the piece is controlled by the use of these woodblocks.

While the Neuron synthesis instrument was not capable of precise pitch control, the performers did find ways to coax expressive gestures from the instrument. We focused the improvisation structure around allowing the exploration of these gestures, while still guiding the musical control that was most readily achievable. While it was extremely hard for performers to predict what pitch they would enter on in a gesture, they did have enough control to match a pitch they heard from another performer, or form a consonance with another performer through a glissando action. They could also reliably navigate between the extremes of tone colors from mellow to aggressively acidic.

5.2 Compositional Form

The form of the piece that resulted involved alternating sections of performers pulsing on woodblocks and performing on the Neuron synthesis instrument. One section asked the performers to independently pulse on their woodblock, ignoring other performers. At their discretion, they could choose another performer to make eye contact with and begin to match the phase and tempo of their woodblock pulse with this performer. When these two connected performers had pulsed in phase for a few seconds, they would put down their woodblocks, tell the software who they had connected to (necessary for a visualization component described below), and begin a back-and-forth communication. Since there were an odd number of performers, there was always one performer pulsing on a woodblock without a partner, waiting to connect with someone. However, since the connected partners can disconnect at any time, the identity of this unconnected woodblock player would shift throughout the piece. There were two sections in the piece where a more coordinated group motion was executed. These occurred when a signal from a member of the group who was deemed the "conductor" would cause connected pairs to disconnect and instead join in a group musical gesture on the Neuron synthesis instrument. The first gesture was to slowly lower the pitch of your sound until the whole group landed on a relatively consonant low chord. The second gesture was the inverse, in which all performers coalesce onto an indeterminate high chord. The piece ends after the second gesture, in a coda where all performers start with asynchronous woodblock pulsing and gradually shift tempo and phase until they are all in rhythmic unison, at which point the "conductor" signals the end of the piece with a sharp cutoff.

Interestingly, while the "drift into rhythmic unison" gesture that was chosen as the closing motif is musically effective and provides a satisfying closure to the piece, in the metaphor of neuron behavior, a large group of neurons firing synchronously is a seizure in a real human brain.

5.3 Live Generative Visualization

We wanted the connection metaphor to be as clear as possible to the audience, so we opted to also do some live video visualization. We also wanted this aspect of the performance to be aesthetically interesting on its own, rather than purely informational. Visual artist and game designer Drew Wallace worked to create a live visualization program that was projected behind the performers at the premiere, in which every performer is represented by a blob that connects tendrils to other blobs when the performers make musical connections. Like the audio, the video was biologically inspired, taking visual cues from "Golgi stain" photographs of human brain neurons. To slightly mirror the digital aliasing artifacts in the neuron synthesizer audio, the visualization also included a contrast glitch that created pixelated edges at the boundaries of the neuron representations.

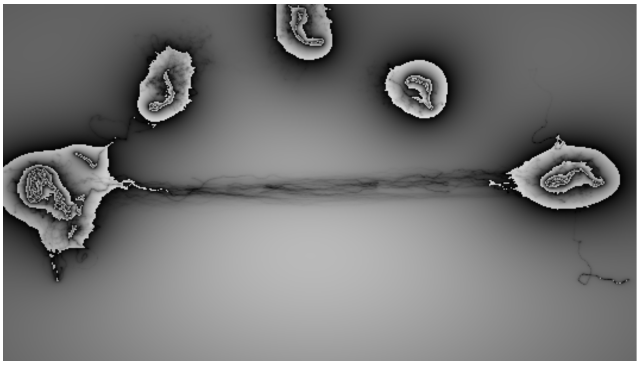

Figure 3: A screenshot from the live generative visualization for Connectome

The live visualization received information from the individual performers' computers via OSC messages over UDP on a wireless network. These messages contained any connection events (such as "/connect 1 5" to draw a connection between players 1 and 5), and a continuous stream of amplitude envelope information from each player (such as "/amp 1 .014"). Using this data, the visualization drew connections between the representations of the players, and scaled the size of each blob to be proportional to the amplitude of the sound made by that player.

6. REFLECTIONS ON THE INSTRUMENT

We needed to decide on an interface for the Neuron synthesis instrument, and in the premiere performance of Connectome we chose a relatively simple solution of a small USB-MIDI controller (Akai MPK2 keyboards and Alesis Trigger Finger controllers were used), connected to a laptop running the VST plugin in Max/MSP. This had the immediate advantages that the instruments were compact and large number of performers could be supplied easily with an instrument, since there were 11 players in the premiere. We already had a large quantity of these USB-MIDI controller types on hand in the ensemble. We mapped the eight parameters of input to the knobs on these controllers, and mapped the "membrane voltage" input to a note-on/note-off event, with different voltage options mapped to different keys on the controller.

Some disadvantages of using these typical MIDI controllers were that they were not particularly visually striking, and they may have suggested interactions to the performers that were not native to the Neural synthesis instrument. For instance, those performers who had the MPK2 keyboards may have expected standard equal-tempered pitch control. One could certainly imagine a more interesting interface to

the software back-end which would be more inspiring to the player and give more information to the audience. Also, creating an embedded version of the instrument could have provided it with a more coherent identity, as opposed to the general-purpose blandness of a laptop computer in front of each performer, especially since the visual feedback of the laptop screen was not used.

The most interesting feature of the instrument turned out to be the effect of the unfamiliar and non-intuitive parameter-to-sound mappings. While there was a dedicated knob for each parameter of the model, the meanings of these knobs were obscure to the users. One represented the "sodium channel activation", another the "sodium channel inactivation", and yet another the "potassium channel". Many of these parameters interacted with each other in ways that were sometimes counterintuitive but often serendipitous and exciting. Most notably, the performers found it surprisingly expressive that the instrument didn't have a "filter cutoff" parameter, but that the interaction of the various parameters could act on the brightness of the sound, while also effecting pitch and other features. One parameter could produce a darken/brighten effect in certain conditions, but also slightly detuned the oscillation and could halt oscillation entirely on other conditions. Informal feedback from the performers included comments that this felt a bit more like playing an acoustic instrument, where many sonic parameters are coupled so as to not be independently controllable. For instance, while palm-muting guitar strings darkens their tone, it also shortens their decay time.

Another feature of the instrument that was particularly interesting was that many of the sounds that we came to consider idiomatic to the instrument were the result of artifacts from pushing the model beyond physically possible parameters, or artifacts of the discretization process. For instance, we allowed performer access to a parameter for step time in the sampling of the system, which gave more direct control over the general pitch range of the oscillation, but also made it much easier to create bizarre effects from aliasing.

One unfortunate effect of the instrument's unpredictability was the fact that the model could be pushed into unstable settings that would cause it to simply stop making sound until a full reset of the parameters was executed and the audio buffer was zeroed out. We dedicated a panic button on the controllers to allow this reset functionality, but it would be ideal to have a better understanding of the conditions under which this happens and prevent them from occurring.

7. FUTURE WORK

The immediate goals for the Neuron synthesizer are to create a more interesting dedicated interface for the instrument and to produce a dedicated embedded version that avoids the need for a computer. It is also important to us that we reduce the instability of the model, or discover what the causes of instability are and limit the input parameters or internal algorithm to prevent catastrophic errors that result in the model refusing to make sound.

Beyond this, we hope to improve the documentation of the Neuron pseudo-object in the OOPS library so that others can more easily incorporate the model into their own code, and release a public version of our VST plugin.

We are working on a virtual analog synthesizer version of the instrument in the VCV Racks format[2], which turns out to have very exciting properties due to the ease of creating feedback loops from the output to the inputs. We plan to release this when it is finished.

We also intend to write more music that takes advantage of this unusual technique, and possibly create a new instrument which allows for selection between several competing biological neuron models to make a kind of meta-neuron synthesizer.

Outside of the specific Neuron synthesizer outlined in this paper, the authors hope to explore other mathematical and biological models as possible sources for sound synthesis.

[2]https://github.com/spiricom/VCVrack

8. CONCLUSIONS

The authors have presented a project that uses a biological model of a neuron as an audio synthesis technique. While this has been attempted before, we intend for this paper to explore the idea more extensively, with a description of the translation process and an example of a musical application of the resulting instrument. We also consider this work to be an argument for the advantages of more closely coupled non-linear control parameters in synthesis, as opposed to isolated linear input parameters that do not interact. Hopefully this work will suggest future developments in the musical applications of physical models of systems not originally intended for sound production.

9. REFERENCES

[1] N. Collins. Errant sound synthesis. In *Proceedings of the International Computer Music Conference*, 2008.

[2] P. Cook. A meta-wind-instrument physical model, and a meta-controller for real time performance control. In *Proceedings of the ICMC*, 1992.

[3] L. Hiller and P. Ruiz. Synthesizing musical sounds by solving the wave equation for vibrating objects: Part 1. *Journal of the Audio Engineering Society*, 19(6):462–470, June 1971.

[4] A. L. Hodgkin and A. F. Huxley. A quantitative description of membrane current and its application to conduction and excitation in nerve. *Journal of Physiology*, 117(4):500–544, 1952.

[5] E. Izhikevich. Which model to use for cortical spiking neurons? In *IEEE Transactions on Neural Networks*, pages 1063–1070, September 2004.

[6] R. Kuivila. Open sources: Words, circuits, and the notation/realization relation in the music of david tudor. *Presented at the Getty Research Institute Symposium, "The Art of David Tudor," in 2001.*

[7] Y. W. Lam. A comparison of three diffuse reflection modeling methods used in room acoustics computer models. *The Journal of the Acoustical Society of America*, 100:2181–2192, 1996.

[8] J. P. Mackenzie. Chaotic predictive modelling of sound. In *Proceedings of the International Computer Music Conference*, pages 49–56, September 1995.

[9] M. Mulshine and J. Snyder. Oops: An audio synthesis library in c for embedded (and other) applications. In *NIME 2017*, pages 460–463. Princeton University Music Department, 2017.

[10] J. O. Smith. *Physical Audio Signal Processing: for Virtual Musical Instruments and Digital Audio Effects*. W3K Publishing, 2010.

[11] G. Wang, P. Cook, and S. Salazar. Chuck: A strongly timed computer music language. *Computer Music Journal*, 13(4):10–29, Winter 2015.

[12] D. T. Yeh. Automated physical modeling of nonlinear audio circuits for real-time audio effects - part ii: Bjt and vacuum tube examples. In *IEEE Transactions on Speech and Audio Processing*, volume 18, March 2011.