Seminar Report

NEURON

Manuel Schischkoff
manuel.schischkoff@student.uibk.ac.at

24 July 2015

Supervisors: Thibault Gauthier
Dr. Georg Moser

Abstract

This seminar report is about NEURON, an environment for empirically-based simulations of neurons and networks of neurons. We will see the workflow of NEURON, the digital representation of a neuron and how to create a simulation. Furthermore a discussion of the 99 bottles of beer problem and a comparison to similar modeling and simulation environments is given.

Contents

1 Introduction **1**

2 What is NEURON **1**
2.1 NEURON's digital representation of a neuron 2
2.2 Simulation Explanation 2
2.3 Workflow 3
2.4 Possibilities of NEURON 5
2.5 Usability of NEURON 7
2.6 Computer Scientific Aspects 7

3 99 Bottles of Beer **9**
3.1 Possible Solutions and Problems 9

4 Comparison to Similar Environments **11**

5 Conclusion **13**

1 Introduction

This seminar report about NEURON for the specialisation seminar 99 bottles of beer is about a modeling and simulation environment for neural networks. We will see the abilities of NEURON and its tools to work efficiently, how a neuron and its components are digitally represented and what properties are needed to simulate a neuron. After a closer look to the workflow of NEURON with help of a tutorial example, we will see what is possible with this environment, in which scientific domains NEURON is useful and where it is not. We will not explain here the current research on huge neural networks, but we will give an overview of this modeling and simulation environment and where we can find a model database with important works for further interests. Finally we will discuss how the lyrics of the song 99 Bottles of Beer could be represented by NEURON, which problems occurred during its implementation and compare this environment to similar ones for a better understanding of NEURON's internals.

An install instruction and the download link of NEURON can be found here[1].

2 What is NEURON

NEURON is an environment for modeling and simulating neurons, network of neurons and their behavior under specified circumstances. Michael Hines, John W. Moore and Ted Carnevale developed the first version of NEURON at Yale and Duke in the 1990s [2]

There are a few possibilities of how we can work with NEURON. On the one hand we can use the graphical user interface to build neurons and networks, define their geometry, biophysical properties and the simulation of their behavior. This user interface is very helpful if we are not very familiar in programming. On the other hand there is a programming way to work with NEURON, where we can use different programming languages like C++, Java, Python or Hoc. If we make use of this opportunity we will define the neurons and all the other parts we have mentioned before within an external file and start the NEURON program given the external file as a program argument. In this seminar report we will use Hoc whenever we make use of the programming part to work with NEURON, since Hoc is very C like, most of the tutorials work with it and NEURON initially worked with Hoc[1]. In my opinion the best way to work with NEURON is a combination of the graphical user interface and a Hoc file, since both sides have its advantages. For example whenever we have to create a new neuron, the cell builder of the graphical user interface is very helpful to get a sense of the geometry of the neuron and if we have to define the parameters of the dendrites, which may be the same for every dendrite, it is easier to do so within a loop over the dendrites array within a Hoc file.

Before we can take a closer look to NEURON we have to know what a neuron exactly is and most important how NEURON digitally represents a neuron.

[1] `https://www.neuron.yale.edu/neuron/download`accessed in July 2015
[2] `https://www.neuron.yale.edu/neuron/` accessed in July 2015

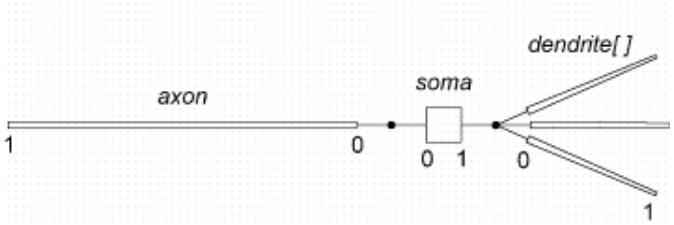

Figure 1: The digital representation of a neuron.

2.1 NEURON's digital representation of a neuron

Neurons are the basis of the nervous system of the brain from living entities. They are responsible for receiving and transmitting informations inside a body. A neuron is build out of several components. The most important components are the soma, the dendrites and an axon. These parts of a neuron are all we need for the digital representation within NEURON to work properly on simulations of neurons and networks of neurons.

The soma is the core of a neuron, it is responsible for keeping the neuron alive and maintains the functionality of a neuron. A dendrite is the frontal extension of the soma. Neurons usually have numerous dendrites building a treelike receptor structure. The dendrites are responsible for receiving information as electrical stimuli from in general other neurons and forwarding them to the cell body. The axon is the rear extension from a neuron. Usually every neuron has only one axon. It is responsible for transmitting a neural signal from the cell body away toward, in general, other neurons [3]. We can see a digital representation of a neuron in Figure 1 [4].

If we work with a single neuron to simulate the behavior of a certain type of neuron, we can ignore the axon and only use dendrites and the soma to get a sense of the functionality of this neuron type. In general a stimulus, which is emitting an electrical signal, makes his way through a neuron from dendrites over the soma to the end of an axon, where the stimulus comes from other neurons. However for simulating just one neuron, the stimulus is just present and comes from nowhere. For the demonstration of the workflow of NEURON we will run through a tutorial, which represents a simulation of a certain neuron type.

2.2 Simulation Explanation

We are going to model and simulate a subthalamic nucleus neuron [5]. This kind of neurons are components of a control system inside the subthalamus. Scientists think that these neurons are responsible for holding muscular response in check and damaging them will result in movement disorder[2]. For simulation issues we need to define some biophysical

[3] `http://psychology.about.com/od/biopsychology/ss/neuronanat.htm` accessed in July 2015
[4] `http://www.neuron.yale.edu/neuron/static/papers/nc97/nc4p1.htm` accessed in July 2015
[5] `http://www.anc.ed.ac.uk/school/neuron/tutorial/tutA.html` accessed in July 2015

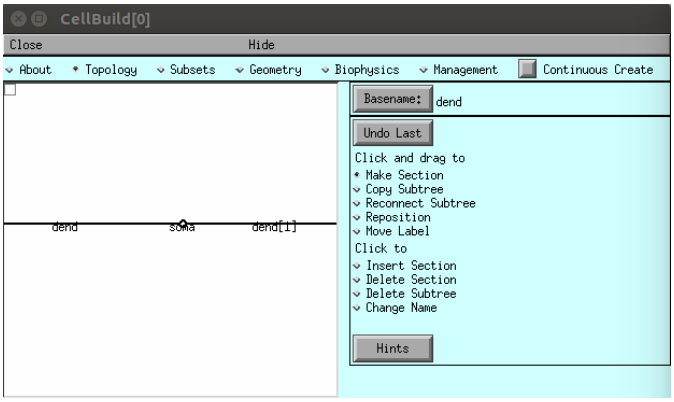

Figure 2: The cell builder of NEURON.

parameters of the neuron. These parameters we have to understand if we want to get an idea of what exactly is happening in nature and since we want to get a sense of the workflow of NEURON, we will not explain all the functionalities of these parameters in this seminar report, because this would lead us too deep into biophysics.

2.3 Workflow

In this section we will see how NEURON works and its common workflow. So we already know that we can use the graphical user interface or an external Hoc file to work with NEURON. Here we want to show both. Therefore we will see some screen-shots of the user interface and some line of codes. The following demonstration was created with help of the tutorial "A NEURON Programming Tutorial" created by Andrew Gillies and David Sterratt [6].

In order to start modeling a neuron we can start the cell builder from the main menu of NEURON. This window will create a new neuron always starting with a soma. Now we can create some dendrites per clicking on make section in the right tab, followed by a click and drag for each dendrite to create them and connect them to the soma. The result will look like in Figure 2 . We can see the corresponding Hoc code in Listing 1, where we just create a soma and an array of dendrites of size 2, in our case. The last 2 lines will connect the dendrites with the soma. Now we have build our neuron, but still it is not useful for anything. We can not make a simulation at this state.

To achieve a useful neuron we need some more steps. Next we define the parameters of the soma of our neuron. To do so we can first click on the geometry tab inside the cell builder and check the boxes for the parameters we want to define. As u can see in Figure 3 on the left side. We are going to define the length of the soma, the diameter and the number of segments. The right side of Figure 3 can be reached by clicking on the biophysics tab inside the cell builder. Here we are going to define some biophysical

[6] `http://www.anc.ed.ac.uk/school/neuron/` accessed in July 2015

```
...
// creates a soma and 2 dendrites inside an array
create soma, dend[2]

// connect things together
connect dend[0](0), soma(0)
connect dend[1](0), soma (1)
...
```

Listing 1: The soma and the 2 dendrites will be created and connected with this Hoc code.

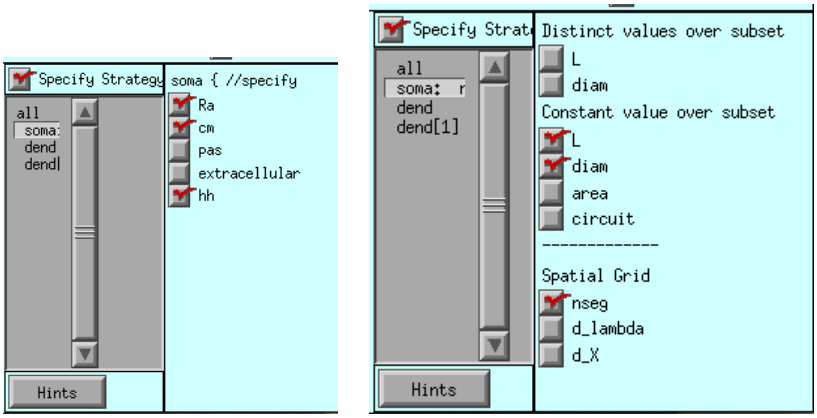

Figure 3: The parameters for the geometry of the soma on the left side and the biophysical parameters on the right side.

parameters like inserting a hodgkin-huxley channel or the electrical resistivity of the soma, which tells us how high an electric impulse has to be, to get recognized by the soma. The Hoc file will be extended as we can see in Listing 2. The parameters of the dendrites needs to get specified too and this will work quite the same as with the soma, as you can see in Listing 3. The only difference here is that we are inserting a membrane instead of a hodgkin-huxley channel.

Finally to create a simulation we need to define a stimulus. To do so we create an object variable called stim, attach it to the soma and create the stimulus of the type IClamp. The IClamp stimulus is emitting some electrical impulses, which the dendrites will detect and forward them to the soma. For the stimulus we have to define parameters like the delay of the impulse, the duration and the amplitude, as we can see in Listing 4. Now we have created a useful and working neuron for our simulation. After we start NEURON with our Hoc file as program argument, we can run the simulation of the stimulus, which is emitting a few electrical impulses over time with given power, by opening the run control window under the tab tools within the main menu. The result of

```
...
// define the soma
soma {
  nseg = 1              // number of segments
  diam = 18.8           // diameter
  L = 18.8              // length
  Ra = 123.0            // electrical resistivity
  insert hh             // hodgkin-huxley channel
  gnabar_hh = 0.25      // sodium channel density
  gl_hh = .0001666
  el_hh = -60.0
}
...
```

Listing 2: Define the geometry and biophysical parameters within a Hoc file.

the simulation can be seen within the graph which should appear automatically and can be modified to look colorful like in Figure 4. We can see that the stimulus starts emitting electrical impulses after a delay of 100 ms, then according to the diameter, the length and the resistivity of the soma and the dendrites we can observe the periodical recurring peaks. Where the duration of a peak corresponds to the combination of the neurons components and its properties. To make this connection more clear, we are going to change some parameters. Within the soma, we change the diameter by increasing it from 18.8 to 280 and the length from 18.8 to 25.8. The diameter from the second dendrite is decreased from 2 to 0.3. The resulting graph looks quite different in voltage changes inside the soma and dendrites as we can see in Figure 5. The differences shows off in bigger breadth of all peaks and therefore the longer duration of them. This observation is simple to explain, since the soma is longer and bigger than before, the impulse needs more time to spread out. The second observation would be the differences of the blue lines, representing the voltage change inside the second dendrite. Here we can see that according to the lower diameter of this dendrite, but with the same resistivity, the voltage changes less than before.

So we have seen how NEURON works and made a small example. However this was a very simple simulation and in the next section we will see what scientists are able to do with NEURON .

2.4 Possibilities of NEURON

Although NEURON is nearly 25 years old, it is still used by scientists for researches. For example M Migliore, F Cavarretta, ML Hines, and GM Shepherd from the Department of Neurobiology, School of Medicine, Yale University USA and Institute of Biophysics, National Research Council, Palermo, Italy and their work on "Distributed organization of a brain microcircuit analyzed by three-dimensional modeling: the olfactory bulb"

```
...
// define dendrites
dend[0] {
  nseg = 5
  diam = 3.18
  L = 701.9
  Ra = 123.0
  insert pas          // membrane
  g_pas = .0001666    // membrane resistance
  e_pas = -60.0       // leakage equilibrium
}

dend[1] {
  nseg = 5
  diam = 2.0
  L = 549.1
  Ra = 123.0
  insert pas
  g_pas = .0001666
  e_pas = -60.0
}
...
```

Listing 3: Define the dendrites.

published in 2014 at Frontiers[3].

The olfactory bulb is a neural structure with extensions to the nasal cavity. It is a receptor for olfactory information and is responsible for transmitting this information to the brain[7].

The work mentioned before concentrates on modeling and simulating this neural structure, for experiments and tests, which should help to understand the functions of brain microcircuits[3]. The project, made with NEURON, can be found in the model database for already implemented models with NEURON. Some examples can be found here[8].

So we can see with NEURON there is a possibility to build small simple neurons and huge networks of neurons and their simulations, where the size of the scope of a network is nearly not limited. Limitations may be the computation power and disk space of the system which is used to create and simulate networks of neurons.

[7] `http://neurosciences.case.edu/faculty/strowbridge/OlfactoryBulb/bulb1.htm` accessed in July 2015

[8] `https://senselab.med.yale.edu/ModelDB/ShowModel.cshtml?model=151681` accessed in July 2015

```
...
// create a stimulus (electron) within the soma
objectvar stim

// create stimulus of type IClamp
soma stim = new IClamp(0.5)

stim.del = 100      // delay
stim.dur = 100      // duration
stim.amp = 0.1      // amplitude
...
```

Listing 4: Define the stimulus of type IClamp within the soma.

2.5 Usability of NEURON

As we have seen until this point of the seminar report, the usability of NEURON concentrates on neuroscience and biophysics and indeed according to the book written for this environment the target users are neuroscience investigators, teachers, students and researchers with background in physical sciences and mathematics[1].

As we can see in the model data base, which is linked to NEURON, there are nearly a thousand already implemented models which are mainly connected to research across-the-board mentioned above. However there are a few models available connected to computer science too[9].

2.6 Computer Scientific Aspects

After some research, it came out that there are connections between NEURON and computer science. As expected, the models, found in the above mentioned model database, concerning computer science, are limited to machine learning with a neural network. In particular, reinforcement learning. For example the work "A Spiking Neural Network Model of Model-Free Reinforcement Learning with High-Dimensional Sensory Input and Perceptual Ambiguity" by Takashi Nakano, Makoto Otsuka, Junichiro Yoshimoto and Kenji Doya[4]. Their work is available in the model data base connected to NEURON here[10]. With their work they try to apply different reinforcement learning frameworks to classic learning problems with neuronal networks[4]. However NEURON might not be the ideal tool for computer scientific interests in neural network for machine learning or artificial intelligence and indeed after some research there can be found many other tools for developing neural networks in a computer scientific senses. Some examples can be

[9] `https://senselab.med.yale.edu/modeldb/ListByModelName.cshtml?c=19&lin=-1` accessed in July 2015

[10] `https://senselab.med.yale.edu/ModelDB/ShowModel.cshtml?model=168143` accessed in July 2015

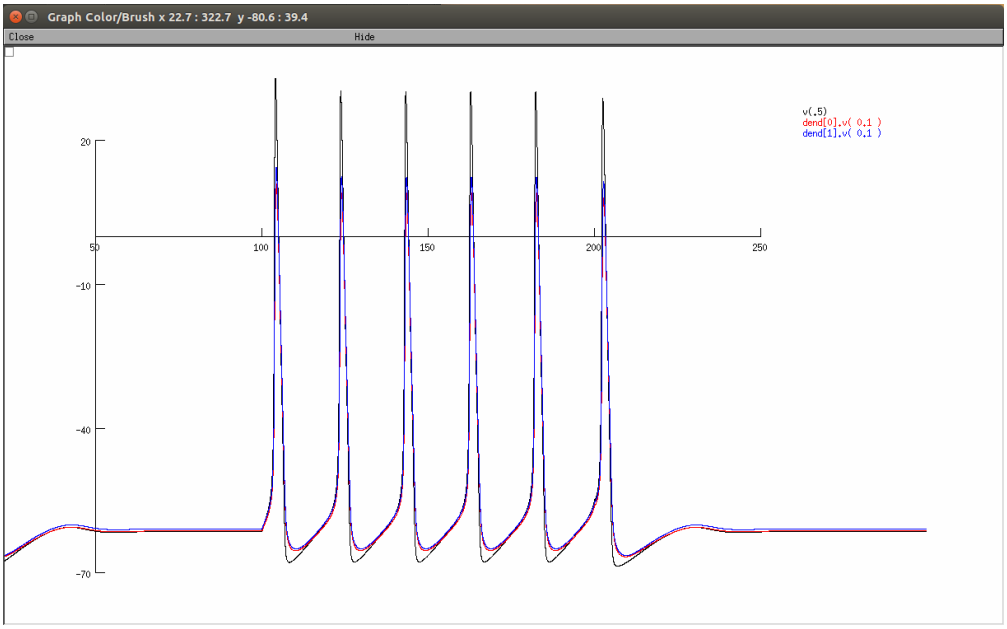

Figure 4: The resulting graph of the simulation. The vertical axis are millivolts and the horizontal one is time in milliseconds. We can see the black line representing the change of voltage within the soma according to the stimulus, as well as the colored lines representing the changes within the dendrites.

found here[11][12].

However, it might be possible to create some logical gates with NEURON as neural network. For example imagine the logical AND, which is only true if both inputs are true and false otherwise. To build such a logical gate with NEURON, we could create a network of 3 neurons, where the first 2 are the input neurons connected to the third one, which represents the output neuron. Now lets assume that from the 2 input neurons some impulses will be emitted and spread out towards to the output neuron with help of their axons. The dendrites from the output neuron will detect these impulses and forward them to the soma, which has an electrical resistivity such high, that the only way to detect an impulse would be if the 2 impulses from the input neurons would arrive at the same time. So if the input neuron 1 is emitting but the second is not, and vice versa, there would not be any detection by the output neuron. Figure 6 should give an idea how the voltage changes inside such a network would look like.

[11] `http://www.neurosolutions.com/index.html` accessed in July 2015
[12] `https://visualstudiomagazine.com/articles/2014/09/01/azure-machine-learning-studio.aspx` accessed in July 2015

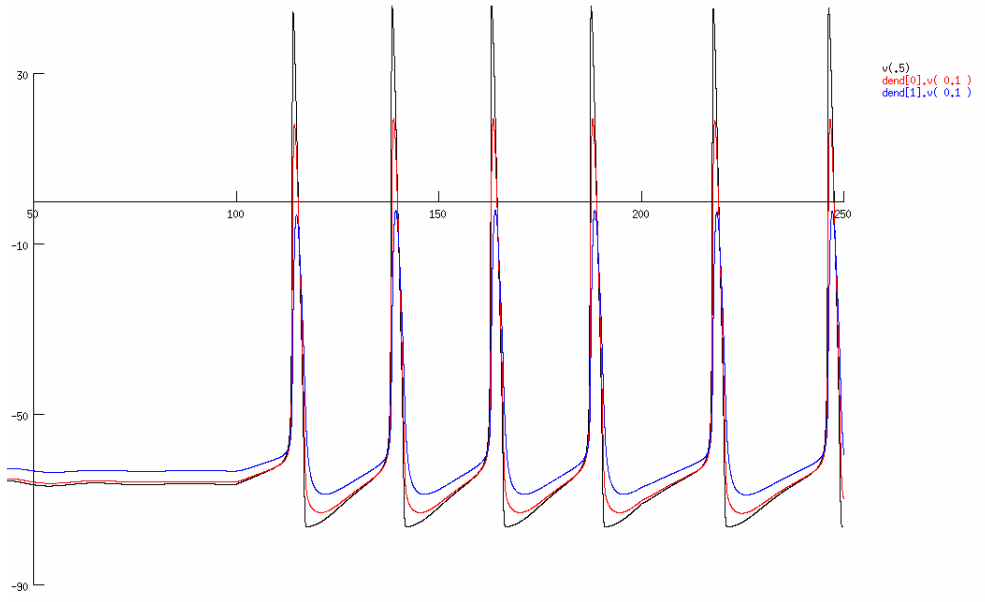

Figure 5: The resulting graph of the simulation according to the changes of some parameters within the soma and the second dendrite.

3 99 Bottles of Beer

In this section we are going to discuss how to implement the lyrics of the 99 bottles of beer song in the sense of this project[13].

We will see some basically possible solutions and the problems which came up. The project mentioned above is about collecting the implementations, which generates the output of the lyrics 99 Bottles of Beer, in as many programming languages as possible, to compare them and get a sense of the syntax of each language. One task of this seminar was to generate these lyrics in the programming language we are responsible for. In this case it is NEURON.

3.1 Possible Solutions and Problems

First of all, NEURON is not a programming language. Thus there is no possibility to write some lines of code which will generate the lyrics, which will lead us to the first problem that occurred.

How to implement a neural network, which generates the lyrics? Although the following suggested solutions are not in the sense of the above mentioned project, it may be still a solution which can be provided within this seminar.

To answer the question mentioned before, a solution could be to generate the lyrics

[13] `http://www.99-bottles-of-beer.net/` accessed in July 2015

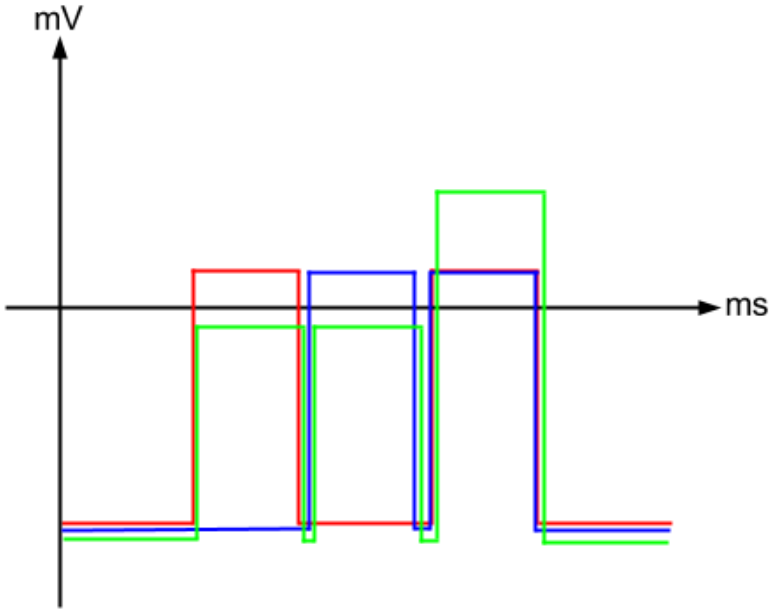

Figure 6: Here we can see how the voltage changes of a neural network representing a logical AND gate could look like. Where the red line would be the input neuron 1 and the blue one input neuron 2. The green line represents the voltage changes of the output neuron. below the horizontal line means that no detection happened and above that the neuron detected the impulses.

with the programming language we are able to use, when we work with NEURON. For example C++, Java, Python or Hoc. Of course we can easily print out the lyrics within a loop to the standard output, but then NEURON would not be involved in any way.

So a solution where we use NEURON would be if we use a network of neurons, where the neurons represents e.g. letters, words, lines or strophes of the lyrics graphically. Their geometry could be formed like letters by the dendrites and connecting neurons from axons of neurons to dendrites of other neurons would do the job of creating a huge network representing the lyrics. However, of course this would not be in the sense of NEURON, since the result would be a network which may simulate nothing useful for any scientific domains, as well as there are no line of codes for the 99 Bottles of Beer project, but it may be still a solution for this task. The problems that occurred during developing such a solution are quite obvious.

First of all we have to build tons of neurons, e.g. with the cell builder, which is quite problematic since the cell builder window is not well enlargeable and the zoom options does not work very well. The next problem would be how to simulate the flow of the

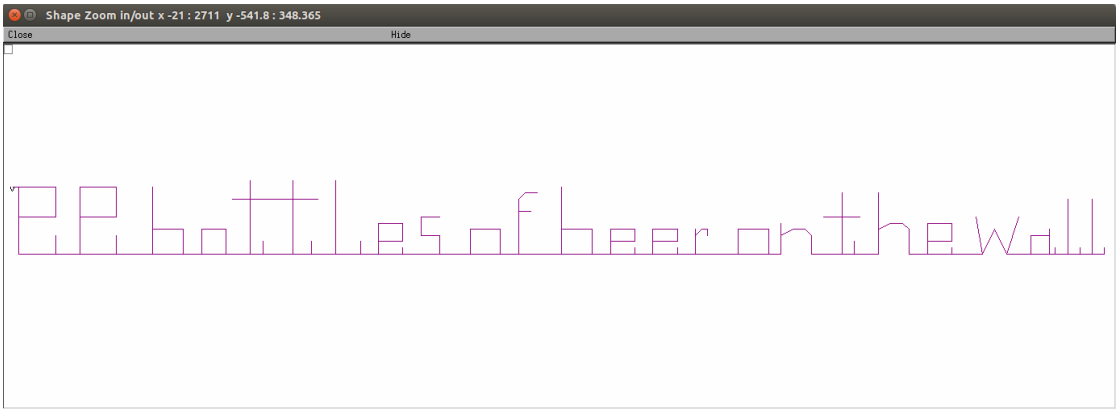

Figure 7: A neuron representing the first sentence of the song 99 Bottles of Beer, by shaping the dendrites to look like letters.

song, e.g. like the iterating part of a corresponding code from 99 to 0. We can not eliminate neurons and create some new ones during a simulation. So we have to build all 100 strophes out of neurons, but still how can we show an electric impulse spreading through such a network?

A resulting voltage graph as we have seen in the section Workflow would be disappointing. However the beginning of such a neural network would look like in Figure 7. There we can see a single neuron with a soma and over 140 dendrites shaping the first sentence of the lyrics.

4 Comparison to Similar Environments

At the end, the last task of this seminar report. The comparison of NEURON to conventional programming languages. Again, since NEURON is not a programming language, but an environment for generating and simulating neural networks, it would be a better task to compare NEURON to something similar. After a while thinking of similar environments, it came up that Android Studio would be a useful pendant to NEURON. Android Studio with the XML files for the graphical layout, the previewer for taking a look at the graphical representation of a model and e.g. the emulator, which comes with Android Studio, to simulate the model will do the job. The reason for choosing Android Studio in combination with XML is, because many people in computer science have already worked with android applications and there are some really good parallels to NEURON working e.g. with a Hoc file. Although we can create a Hoc file for NEURON and a XML file for Android Studio, it works the other way around too. When we are creating a neuron with the cell builder, NEURON will generate a Hoc file where all the i.e. dendrites will be defined as we have placed and connected them with the soma. Android Studio generates XML files when we define the layout of an application

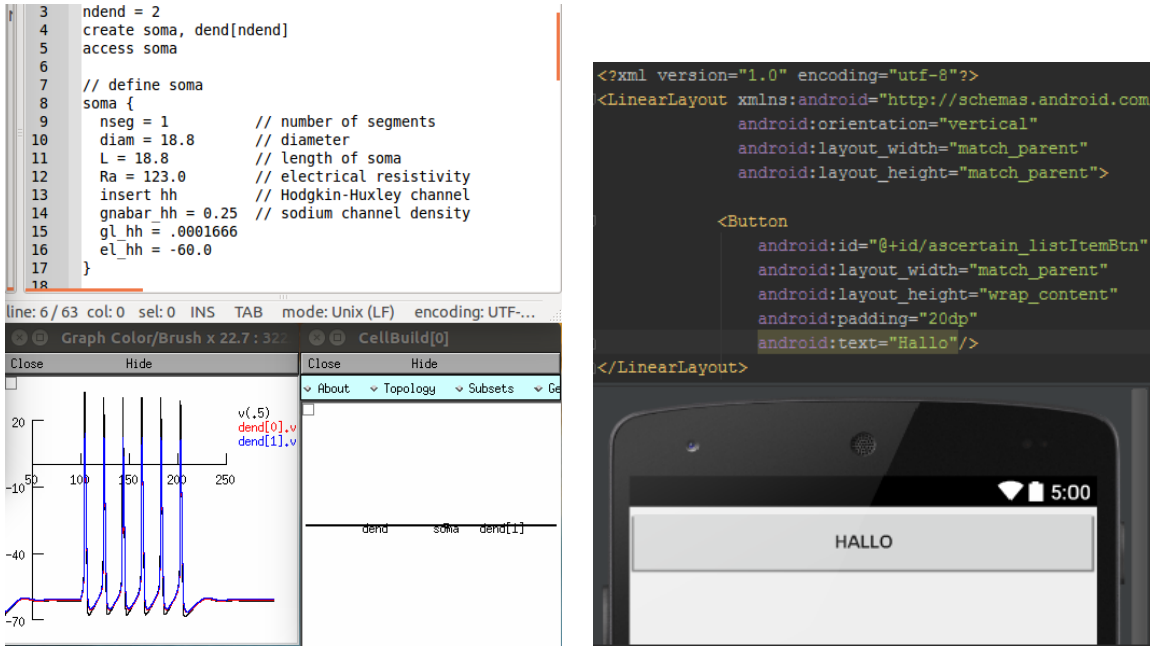

Figure 8: Comparison of NEURON with a Hoc file on the left side and Android Studio with a XML file on the right side.

per drag and drop of its components too. However to see the parallels it is better to define the files for the environments.

First of all we have keywords or tags within a Hoc file like create soma, dendrites or access soma, which NEURON will interpret and make use of it. On the other side we have a XML file with tags like LinearLayout or Button, as we can see in Figure 8. Even more, we have some geometrical parameters for the soma, as we can see in the Hoc file, like the diameter or the length of the soma. The pendant at the XML file would be the layout parameters as the layout width or height for example. Then we have some topic concerning parameters as the electrical resistivity of a soma on the NEURON side and e.g. the name of the button on the Android Studio side. The next part is about the graphical representation of the work which can be produced with these two environments. On the one hand we can see in Figure 8 the cell builder, which shows the geometry of the corresponding neuron. On the other hand the previewer, which comes with Android Studio, shows the graphical representation of the button, created by the corresponding XML file. Finally for simulation issues we can see the voltage graph on the NEURON side and Android Studio is able to simulate e.g. a button click within the emulator for the application. So we can see there are quite good parallels which should clearly show how NEURON works and gives us an idea of the workflow too.

5 Conclusion

In this seminar report we have presented NEURON, an environment for empirically-based simulations of neurons and networks of neurons. First we took a look on what NEURON is and how this environment digitally represents a neuron. Then we have shown the workflow of NEURON on the basis of a simple tutorial example and saw some important work published online where NEURON was involved for simulating a real world neural structure. Computer scientific aspects like reinforcement learning can be implemented with NEURON, but is not the best way to do so. Then we have tried to solve the 99 bottles of beer problem and concluded, that the possible solution would not be in the sense of NEURON or the 99 bottles of beer project mentioned in the corresponding section. In the end we gave a comparison of NEURON to Android Studio to see some parallels like keywords and tags, previewer and simulations, which helps to understand NEURON.

Finally we can say that NEURON is a very useful tool and still in use, although it is about 25 years old. Because NEURON is meant to be a realistic simulator, mainly for biophysicists and neuroscientists, computer scientists do not benefit from this software as much as others do.

References

[1] Nicholas T. Carnevale and Michael L. Hines. *The NEURON Book*. Cambridge University Press, New York, NY, USA, 2006.

[2] Bruce H Robinson MD. *Biomedicine: A Textbook for Practitioners of Acupuncture & Oriental Medicine*. Blue Poppy Press, 2007.

[3] Michele Migliore, Francesco Cavarretta, Michael L Hines, and Gordon GM Shepherd. Distributed organization of a brain microcircuit analysed by three-dimensional modeling: the olfactory bulb. *Frontiers in Computational Neuroscience*, 8(50), 2014.

[4] Takashi Nakano, Makoto Otsuka, Junichiro Yoshimoto, and Kenji Doya. A spiking neural network model of model-free reinforcement learning with high-dimensional sensory input and perceptual ambiguity. *PloS one*, 10(3), 2015.