Using BrainScaleS-2 in an Undergraduate Lab Course

Ilya Volkov

April 16, 2024

Abstract

Undergraduate physics students at the Heidelberg University have to take the module advanced lab course where they get to know the scientific work in general, and the work of specific groups in particular. In the following, such a course is presented and improved for the Electronic Vision(s) group, which bases its work on neuromorphic engineering. The first goal is to show the capabilities of analog neuromorphic hardware, its limitations and application in order to get undergraduate students interested in this interdisciplinary field. Another goal is to introduce the group itself to students, so that potential candidates for internships and Bachelor's Theses can familiarize themselves with the work done in this group. For this purpose, I have described a simple neuron model and have used it to introduce the LIF-model and an extension of it, the AdEx-model. Applying this theoretical knowledge, students will be given tasks to work with Lu.i, a small circuit board capable of emulating the LIF model, and with BrainScaleS-2, a chip developed in the Electronic Vision(s) group, which implements the AdEx model. In these assignments, the students apply the underlying models of the respective hardware to characterize the components and observe the behavior of theoretical models on real life systems. They are also introduced to working with the `pynn`-package for Python, which is an interface for applications on the BrainScaleS-2 system through EBRAINS. Finally, students are challenged to solve a constraint satisfaction problem in the form of sudoku, applying their knowledge of hardware and software to a real-world problem giving them a glimpse into more advanced applications of neuromorphic hardware.

Contents

1 Introduction

1.1 Biological Overview

The central nervous system is highly complex and responsible for many functions in diverse life forms. From processing sensory information, coordinating movement or having "thoughts", everything is managed by this system. With the aim of understanding how this works, it was possible to identify one important building block of this system: a neuron.

In this work, we will only dip our toes into the deep waters of neuromorphic engineering, limiting ourselves to a few models of neurons and synapses, learning their behaviors and how to model them, and finally taking a look at more advanced applications.

We start with a concept of an **ideal spiking neuron** introduced by Gerstner et al. 2014. This focuses on a limited subset of neuron features and the goal is to introduce the concepts of neuroscience.

A neuron has roughly spoken three major parts: soma, axon and dendrites, which serve different purposes. The soma is the body of the cell and contains the nucleus and most of the protein synthesis takes place here. Attached to it in a branched fashion, are long "wires", which are called dendrites. In short, they receive signals in form of stimulation through chemical processes and transform it into electrical charges, which then travel along the dendrites. The third part of this neuron is a long outgoing "wire" which is initially not branched and is called axon. The tip of an axon is then branched again and connects to the dendrites of other neurons. The connection point between axons and dendrites are called synapses. At the synapse, the axon transforms electrical signals into chemical ones, which are then passed to the dendrites. Commonly, it is referred to as presynaptic cells transmitting signals to postsynaptic cells.

Let's take a closer look at the (physical) behavior of this cell. For this, let's assume, that we could attach a voltmeter to the soma and measure the membrane potential. In the absence of any stimulation, we observe a negative resting voltage of typically $-65\,\mathrm{mV}$ in relation to the extracellular fluid (which is set to $0\,\mathrm{V}$). Therefore, we call this state strongly negative polarized. If we inject stimuli through the dendrites, the membrane potential changes either to a higher or to a lower voltage. This can be formalized as following: For a given time the membrane potential of neuron i is $u_i(t)$. For all times before $0\,\mathrm{s}$ the potential equals $u_i(t < 0\,\mathrm{s}) = u_{\mathrm{rest}}$ which is the state, when the neuron is not stimulated. At $t = 0\,\mathrm{s}$ a stimulation occurs broadly spoken in form of a rising/falling voltage at a dendrite which is triggered by a presynaptic cell.

We define

$$\epsilon_{ij}(t) := u_i(t) - u_{\mathrm{rest}} \tag{1}$$

as the postsynaptic potential (PSP) of the membrane. To indicate the direction of change we define further

$$\epsilon_{ij} = u_i(t) - u_{\mathrm{rest}} \begin{cases} > 0 & \text{excitatory postsynaptic potential (EPSP)} \\ < 0 & \text{inhibitory postsynaptic potential (IPSP).} \end{cases} \tag{2}$$

But the potential of the membrane does not stay at the same level. The voltage decays over time back towards the resting potential u_{rest}.

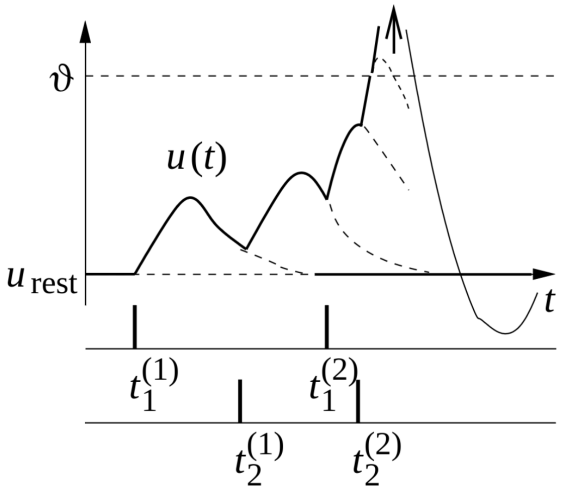


Figure 1: Membrane potential of a neuron stimulated by to presynaptic neurons. On the x-axis is the time while on the y-axis is the membrane potential. At given times $t_i^{(n)}$ the n-th spike from neuron i arrives at our observed neuron. Each spike leads to a rise of the membrane potential u. The dashed line indicated the assumed path if there would not be any additional iputs. In this instance $t_2^{(2)}$ is enough to cross ϑ (threshold) and the neuron fires. This is also called PSP-stacking (image from Gerstner et al. 2014, Chapter 1.2).

Now let's assume there are 2 presynaptic neurons $j = 1, 2$. These neurons emit signal at respective times $t_j^{(m)}$ where m states the m-th signal peak coming from neuron j. At each incoming signal peak charges are deposited at the membrane and the potential of it rises. When many excitatory inputs arrive at the neuron and the membrane potential reaches a critical value, triggering the neuron to fire. Thereby the membrane potential rises rapidly to a peak voltage and this potential then propagates down the axon, which is referred to a spike. The outgoing spikes of a neuron are called a spike train. After such a pulse the membrane potential drops beneath the resting voltage and climbs exponentially back up (For many neuron types it is an exponential behavior (Gerstner et al. 2014)). This is called hyperpolarization or spike-afterpotential. In that time the neuron can't be stimulated. The trajectory of such an event can be observed in fig. 1.

Now we can understand why this model is called an ideal spiking neuron (Gerstner et al. 2014).

1.2 Modeling the Behavior of a Neuron

Leaky integrate-and-fire Model

Neurons are driven by biochemical and bioelectrical principles. Here, we aim to find a model that describes the basic behavior of an ideal spiking neuron without incorporating all biological aspects, while capturing the dynamics described in section 1.1. The following assumptions have to be fulfilled: the spikes always have the same shape, they do not contain any information. All information is coded in the spike timing. Further, the membrane potential abstracts and processes the information. Another feature, that needs to be modelled, is when $u_i(t)$ reaches a critical voltage

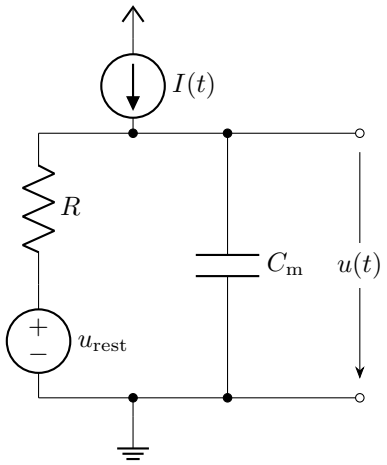


Figure 2: A basic circuit for a neuron model. The capacitance C_m represents the membrane over which the voltage $u(t)$ is measured. Parallel to the capacitance is a resistor which allows outflow of charges. u_rest is voltage source which dictates the state of rest of the capacitance. A time dependent current source $I(t)$ emulates spikes.

threshold ϑ, a spike has to be initialized, causing this neuron "to fire" at a time $t_i^{(m)}$. Such a model, which fulfills the requirements above, is proposed by Lapique 1907 and called leaky integrate-and-fire (LIF).

In this model we assume that the cell membrane is a capacitor with a certain capacitance C_m. When a spike arrives, the membrane potential should rise by roughly the same voltage each spike, which is achievable by depositing a certain charge onto the capacitor using a current source. As previously discussed, the membrane potential decays back to resting potential; therefore, the charge leaks from the membrane, which we emulate with a resistance R. In addition, the neuron is on a resting voltage that is recreated by a voltage source. This completes the basic circuitry for a neuron model as seen in fig. 2.

If we analyze the electrical circuit, we can find a differential equation describing the behavior of the capacitor voltage.

$$\tau_\mathrm{m} \frac{\mathrm{d}u_i(t)}{\mathrm{d}t} = -\left[u_i(t) - u_\mathrm{rest}\right] + R \cdot I_i(t) \tag{3}$$

Here, $\tau_\mathrm{m} = R \cdot C_\mathrm{m}$ is also called the *membrane time constant*, and the index i refers to the i-th neuron. $I_i(t)$ in this equation represents a time dependent current flow onto (excitatory) or away from (inhibitory) the membrane. In neuroscience, this equation, which describes a leaky integrator, is the equation of a passive neuron. Currently, it fulfills the requirement of integrating incoming spikes, but it lacks an active part in the form of a spiking mechanism. For the basic model, we define a threshold value ϑ. When the condition $u_i > \vartheta$ is met, an additional circuit emits a spike that propagates to all connected neurons. At the same time, the potential of the capacitance is drawn to a defined value u_reset and kept at this level for a time period, which is called the "refractory period" τ_r.

Adaptive exponential integrate-and-fire (AdEx) Model

The leaky integrate-and-fire (LIF) model captures only some basic dynamics of a real neuron. Various models aim to enhance this understanding. In Brette et al. 2005 an improved LIF is presented. The main additions are an exponential and an adaptation term.

$$\tau_{\mathrm{m}} \frac{\mathrm{d}u_i}{\mathrm{d}t} = -\left(u_i - u_{\mathrm{rest}}\right) + R\left(I_{\mathrm{stim}} - I_{\mathrm{ad}}\right) + \Delta_{\mathrm{T}} \cdot \exp\left(\frac{u_i - u_{\mathrm{T}}}{\Delta_{\mathrm{T}}}\right) \tag{4}$$

$$\tau_{\mathrm{w}} \frac{\mathrm{d}I_{\mathrm{ad}}}{\mathrm{d}t} = a\left(u_i - u_{\mathrm{rest}}\right) - I_{\mathrm{ad}} \tag{5}$$

The eq. (4) is arranged in such a way that the extension of eq. (3) is easily visible. As new arguments are introduced: I_{ad} which is the adaptation current, the threshold slope factor Δ_{T} and the effective threshold potential u_{T}. Further a second ODE is introduced to describe the dynamics of the adaptation. Therefore, a conductance a and the time constant for the adaptation current τ_{w} is required. Another modification is the reset condition. While previously only the neuron was set to a reset potential, now the adaptation has to be modified as well. The condition is then

$$\text{if } u_i > \vartheta \text{ then } \begin{cases} u_i \to u_{\mathrm{reset}} \\ I_{\mathrm{ad}} \to I_{\mathrm{ad}} + b \end{cases} \tag{6}$$

Here is an additional variable defined, which is the spike triggered adaptation b. Due to the exponential term this model is called Adaptive Exponential Integrate-and-Fire (AdEx). It allows now to describe different neuron types and model more sophisticated behaviors (see section 2.8)

1.3 BrainScaleS-2

One of the current scientific endeavors is to advance our understanding of the brain and apply this knowledge to possible applications to solve real world problems. There are several approaches to learn from the brain: one is by measuring biological neurons and mapping their structures to learn about their role in the nervous system or another is by rebuilding these structures by creating atomic units that imitate observed behaviors. An interdisciplinary scientific field that tackles the latter challenge is called "neuromorphic engineering" that tries to do computing on brain-inspired structures. Recently there are many attempts to implement models of neural systems in analog, digital, mixed-mode hardware or purely in software.

The group Electronic Vision(s) group at Heidelberg University works actively on "silicon brains", including application-specific integrated circuits ASIC that combine analog and digital hardware to emulate neuronal systems.

Hereby the biological processes are represented by physical quantities, such as voltages and currents, that are directly correlated to neural dynamics on a continuous timescale while digital components are used for further processing of events. Scaling the on-silicon components properly, it is possible to evolve the dynamics "[...] at a 1000-fold accelerated time scales compared to the biological time domain [...]" (Pehle et al. 2022) This might sound advantageous but entails it own challenges. Due to the nature of analog computing on a small-scale chip one encounters issues like fixed-pattern noise, which is induced by material properties and manufacturing, as well as temperature

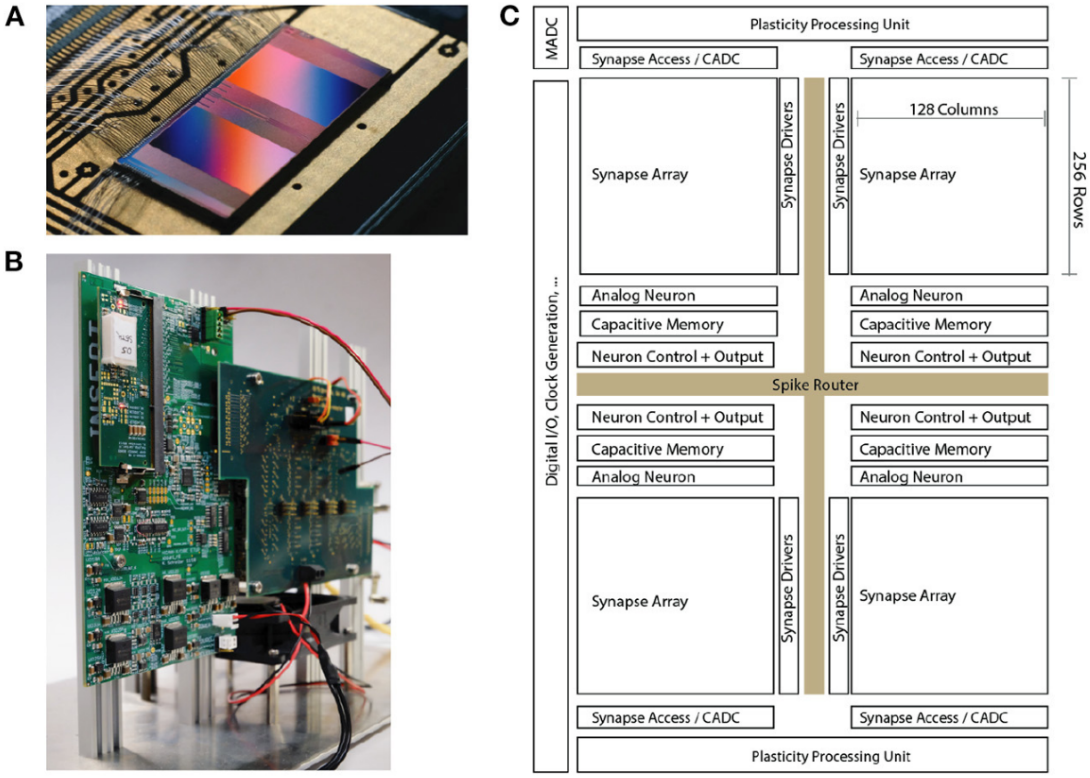


Figure 3: Overview of BrainScaleS-2 from floorplan to experiment setup**(A)** Bonded chip on its carrier board, one can see the two synaptic crossbar arrays. **(B)** Experiment setup with the bonded chip under a plastic cover on a carrier board.
(C) Schematic floorplan of the chip: Two processor cores with access to the synaptic crossbar array are on the top and bottom. The 512 neuron circuits and analog parameter storage are arranged in the middle. The event router routes events generated by the neurons and external events to the synapse drivers and to/from the digital I/O located on the left edge of the chip. (Figure taken from Pehle et al. 2022)

and time-dependent drifts. It is mandatory to have either hardware or software solutions to address these. The basic layout of the BrainScaleS-2 hardware can be seen in fig. 3. For this introduction it is important to get an understanding of the basic features of the chip, leaving out advanced features like e.g. plasticity processing units. On the ASIC there are 512 individual neurons split into 4 quadrants. Each neuron circuit is designed to emulate the previously introduced AdEx equations (eq. (5), eq. (4)). Further the circuitry of each neuron can be configured individually via 80 bits of local configuration static random-access memory (SRAM) and 24 analog parameters which are set by an on-chip digital-to-analog converter with 10-bit resolution. This allows to control for each neuron all potentials and conductance leading to a precisely tuned model dynamics and allowing to compensate for production-induced fixed-pattern deviations. Other parameters, like the membrane capacitance or refractory time τ_{r}, can be directly configured via the locally stored

digital configurations or features like a constant current can be enabled or disabled. For example, it is possible to disconnect the adaptation current and the exponential term from the membrane, reducing the AdEx model to a simple LIF neuron model. Having these possibilities allow studying behaviors of different neurons. For interconnecting all neurons on the chip there are multiple synapse arrays. Each neuron circuit integrates synaptic stimuli from a column of 256 plastic synapses. These synapses feature a time-continuous current injection as modelled in section 1.2 but allow also to operate in a more advanced current-based mode (For more details see Gerstner et al. 2014, Chapter 13.6.3). The total synaptic current

$$I_{\text{syn}} = \sum_i w_i S_i(t) * e^{-t/\tau_{\text{syn}}} \tag{7}$$

hence results from the sum over all associated synapses i, the presynaptic spike trains $S_i(t)$, their respective weights w_i and the synaptic time constant τ_{syn}. The weights are stored locally per synapse in a 6-bit SRAM and modulate the amplitude of an emitted current pulse (Pehle et al. 2022).

All in all, the integrated features described above are a single-die system. To work with this chip a setup is tailored around the requirement of accessibility, called "Cube Setup". This features a BrainScaleS-2-chip under a plastic cover on a carrier board, which holds connections for an analog readout with an oscilloscope and a specialized processor. The latter is an FPGA where experiments are uploaded via a network in form of "playback" programs which contain a configuration of the BrainScaleS-2 chip, execution and data recording whereby the data than is sent back via network to the user.

2 Advanced Lab Course

Undergraduate students of the Heidelberg University have to complete an advanced lab course. The primary goal is to get the students familiar with scientific work and additionally introduce them to research groups. In order to perform experiments during this lab course the students are provided with a script which describes the underlying physics of the setup and contains tasks which are aimed to be done during two complete days or four afternoon sessions. The students perform different tasks related to the group' work to get an understanding of the underlying science. Further they record data that they have to analyze, discuss, and report in form of either a short report (which is comparable to a scientific paper), a lab book or an extended report (which is more comparable to a bachelor thesis report). This familiarizes students to work with lab equipment and correct report their findings. Additionally, they can choose to present their results to an interested group of students during a 15 to 20 minutes presentation.

Providing an advanced lab course is therefore mandatory not only to fulfill a requirement from the administration but to gain attention of students for the scientific work of the group. The course should aim to:

- be doable during a "short" time,
- contain challenging tasks for undergraduate students,
- incorporate measurements that can be analyzed and discussed,
- be related to the scientific work of the group

To simplify the access to the hardware of the Electronic Vision(s) group, the platform EBRAINS will be used. EBRAINS provides a web interface with a Jupyter environment specially configured to work with the backend in the institute. To be more precise, an adapted version of the `pynn`-package is installed to work with the BrainScaleS-2 hardware and access to the quiggeldy service is established. Latter is a user-to-host experiment managing service, that schedules the execution of experiments on available hardware. Although using quiggeldy is not required because the advanced lab course has an individually assigned test setup, it will be used due to simplification reasons. Running experiments in Jupyter requires the students to write python code in a notebook and execute it.

In the following such tasks will be presented and viewed under the previous listed aspects.

2.1 Reworking the introduction

In an earlier iteration of the advanced lab course, the introduction lacked information about the basic neuron mechanics like reaction to stimulation currents and a basic mathematical model to describe it. Therefore, section 1.1 and section 1.2 were integrated into the introduction. They should give more information that fits the tasks and prepare the students more properly. Further information is given about the BrainScaleS-2 system and the hardware setup which is visible in the lab. Content about the `pynn`-package and quiggeldy is given in the notebook introductions.

2.2 Creating Experiments with Lu.i

Newly created tasks focus around a simple LIF-Neuron called Lu.i. The goal of these tasks is to introduce the students to a basic neuron model in a hands-on manner.

f-I Curve

In this task the students will record a f-I curve of a Lu.i using a current source and an oscilloscope. Thereby it is expected that they have some basic knowledge about the work with an oscilloscope from previous lab courses and no further introduction is needed. Additionally, a controllable current source is required to generate influx on the neuron. Therefore, I built a simple circuit as seen in fig. 4.

The goal of the current source is to provide a steady current flow onto the neuron to get stable firing rates while operating.

Now, with a controllable current source it is possible to record an f-I curve of a Lu.i-neuron. This task achieves two goals: At first it introduces the rate-based coding of information which is related to the influx onto the membrane and second it allows to characterize the components of the neuron. It is further possible to set a task to fit the data with a model that is deduced by the students, so they work with eq. (3) getting a better understanding of the model.

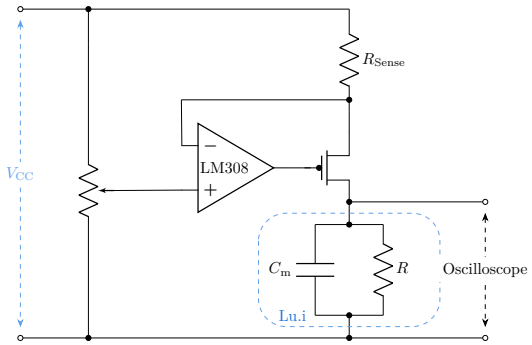


Figure 4: Current source circuit. A basic design for a current source to record a f-I curve with the Lu.i.

For the setup of this task a voltage source with 5 V to power the current source, a modified Lu.i with connections to the membrane and an oscilloscope are required. The task is then to connect the power source to the current source and the outputs of latter to the membrane of the Lu.i. The oscilloscope has to record the membrane voltage over the time, so the students can make several observations:

- The difference between leak over threshold configuration and current induced spiking (which originates in circuitry design).
- Impact of the refractory period τ_{r} and measurement of it (to compare later to fitted data)

To achieve consistent results throughout the different student groups it is convenient to demand a resting potential of 0 V. Having a non-zero resting potential requires another variable in the model and therefore another fit parameter. Tuning the membrane resistor and therefore τ_{m} can be either set to maximum, which would be the easiest configuration to communicate or just let the students set it to an arbitrary value somewhere in the middle of the potentiometer. For simply recording an f-I curve it is not required to know any configured values, but then the task is only to note observed

variable	absolute value	rel. error
τ_m [ms]	1.46 ± 0.25	16.12%
R [kΩ]	45 ± 7	15.56%
C [µF]	32 ± 7	13.64%

Table 1: Fitted values of components for a Lu.i. The values are given with an absolute and a relative error. The capacitance is deducted from the resistor and τ_m as defined in the introduction. As fixed values are set $\vartheta = 812\,\mathrm{mV}$ and $\tau_\mathrm{r} = 12.4\,\mathrm{ms}$

numbers and plot them to obtain a curve, that is starting to rise from a certain current as seen in fig. 5.

An additional task would be to fit a function to the obtained data. Having this task covers requirements for the advanced lab course, namely a more "challenging" exercise because students will have to assume boundary conditions of eq. (3) and understand their observed measurements (one has to account for τ_r and, if the neuron was properly configured, one also can leave out u_rest). Taking these conditions into account and using the separation of variables approach in solving the eq. (3) one obtains

$$\int \frac{\mathrm{d}u}{u - I \cdot R} = -\frac{1}{\tau_\mathrm{m}} \int \mathrm{d}t\,. \tag{8}$$

To calculate the expected firing frequency, one needs to get the total time of charging the neuron to the threshold voltage ϑ. This consists of solving the above equation with the limits 0 to ϑ on the left-hand side and 0 to t_f on the right-hand side. Rearrange the equation according to t_f, you get:

$$t_\mathrm{f} = -\tau_\mathrm{m} \left[\ln\left(1 - \frac{\vartheta}{R \cdot I}\right)\right] \tag{9}$$

It is important to note another condition which arises from the logarithmic part of the equation. The argument of the logarithm has to be greater zero for all times and therefore the condition

$$1 > \frac{\vartheta}{R \cdot I} \implies R \cdot I > \vartheta \tag{10}$$

imposes the initial spike current as well as a required check when applying an optimizer function. For completion of the proposed function for describing the f-I curve an additional term has to be added to t_f which is the refractory time. The final function is then

$$t_\mathrm{f} = -\tau_\mathrm{m} \left[\ln\left(1 - \frac{\vartheta}{R \cdot I}\right)\right] + \tau_\mathrm{r}\,. \tag{11}$$

Using this derived equation, it is possible to fit the data as seen in fig. 5.

Another task is now to extract the components parameters like resistance and capacitance under knowledge of ϑ and I. Adding the schematics of the current Lu.i design student need to show their understanding of the schematics and arguing why their values may or may not be in a certain range. As an example, I obtained through this method the components' values as seen in table 1.

Studying the schematics of the Lu.i it is possible to find the components and compare them to the fitted ones (see table 2).

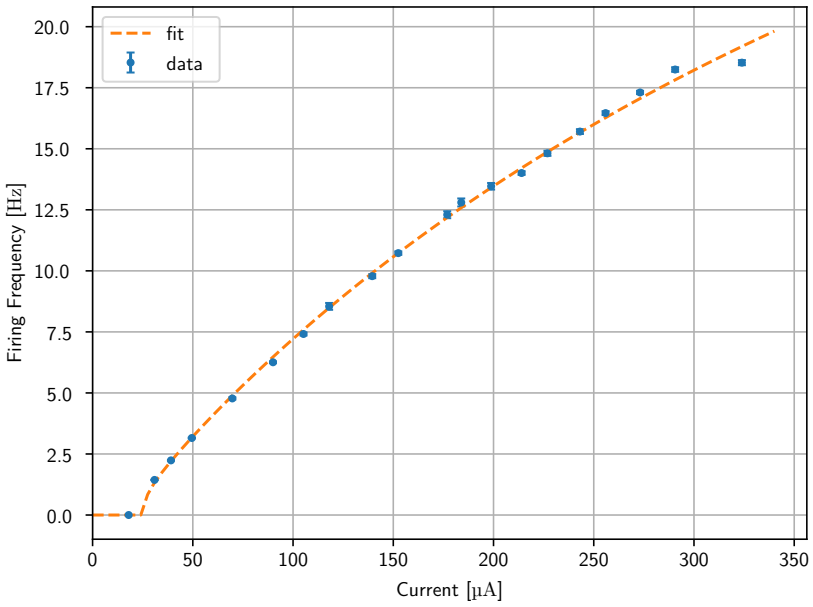


Figure 5: Fitted model to f-I curve of a Lu.i. In blue are data points which were recorded during the experiment, which was set up as described in the task. The blue error bars are used from statistics on the oscilloscope which are standard deviation of the measurement. Note that errors are expected to be small because neurons are deterministic with little noise. Further the derived equation was fitted with *curve_fit* from *scipy.optimize* package.

	fit	theo
R [kΩ]	45 ± 7	50 ± 5
C [µF]	32 ± 7	22 ± 4

Table 2: Comparing components values from fit with schematics. Hereby tolerances for the capacitance and resistor are taken from datasheets and are set to 20% and 10% respectively

2.3 Creating an Environment

Working with the BrainScaleS-2 System requires many Python libraries like `pynn.brainscales2`. To minimize the effort for the student to start working with the chip, an experimental setup was dedicated for the students with an own entry in the quiggeldy scheduler. The connection to the setup is then performed through the EBRAINS web page. There the students start to work with a Jupyter-notebook. Before one of the experiments from the demos (Electronic 2023) can be run, the web surface connects to the hardware through the internet. To do so, it grabs a `.csv` file from a dedicated web page and connected to the stored IP address and port. Thereby it gave a status that it had connected to the setup which name was also stored in the `.csv` file. At the same time, it was possible to change IP address and port without knowing to which hardware setup one was connected in the end. This led to some issues figuring out if the connection was properly established and would lead to confusion under the students, if there were not a differentiation or more precise statement to which setup they are connected. Stating the correct experimental setup

required therefore a small tweak in the code base.

```
    with hxcomm.ManagedConnection() as connection:
        identifier = connection.get_unique_identifier()

    logger.INFO(f'Connection to {identifier} established')
```

With this change after a successful connection to an experimental setup it is stated, which one it is. This should guarantee that the students use the right one and will allow them to perform in situ experiments like measuring the membrane potential with an oscilloscope (see section 2.4).

2.4 Analog Readout

A user, who interacts with the chip through a program, can't observe the physics which underlay the emulation. Although it is possible to trace the membrane potential in software a direct observation on an oscilloscope is much more impressive. In this exercise students are tasked to record an analog output of the chip to visualize the "real" physics. Measuring real quantities and comparing them to the digital units allows calibrating axes of plots and therefore deducing other quantities from observed data (it is also viable to get physical units from software, but it misses out the opportunity to use lab equipment and to make the processes "accessible"). Using lab equipment it is possible to combine knowledge gained from Lu.i experiments like measuring a f-I curve (see section 2.2) and parameterizing components on the chip without any knowledge of the complete system. Further, the analog readout is useful for a synfire-chain demonstration. The latter is an interconnection of neurons in such a way, that their firing activity is self-sustaining and repetitive. Digital measurements are possible and show the firing activity quite well, but are limited by the runtime which might imply that the firing activity stops. Using an oscilloscope proves this self-sustained activity outside a runtime of a program and demonstrates the possibility of a continuous operation of the hardware without any input of signals from a computer or a clock.

2.5 Implementing a simple neuron simulation

Creating a neuromorphic hardware is a tedious process. In this task we want to demonstrate why such hardware helps computational neuromorphic to perform tasks faster. This task is proposed or students who want to write an extended lab report. It is created in such a way that the students are guided through it quite easily. The main goal of this is to introduce some concepts: first to work with simulations. In the Electronic Vision(s) group simulations play an important role to understand what the hardware should do and how it should behave. Therefore, a basic simulation of a LIF neuron is written with the possibility to apply a custom current. The students have to implement a simulation loop which checks for events in form of spikes. To accomplish this task, they learn a basic strategy solving differential equations with Runge-Kutta scheme. As a second goal the task aims to raise an awareness for complications with simulations. It shows that the simulation has a much higher requirement of resources to perform a "simple" task of integrating a LIF-neuron than the emulation on BrainScaleS-2. At the same time, it offers a great potential

for discussion about different hardware scopes. Regarding simulations there has to be made the point, that they can be parallelized (like on GPUs) or accelerated by simply using a more powerful computer. A more extended discussion is provided in the discussion (section 3.4).

The simulations are performed on EBRAINS, whereby the hardware parameters (like CPU) are unknown. For the simulation an optimized solver `solve_ivp` from the `scipy`-package is used with a standard configuration. In the backend the solver uses a Runge-Kutta scheme with an adaptive time stepping. Performing a 10 ms long simulation took about 1.2 s execution time on an average of 1000 runs

Running a basic configuration, 1 `HXNeuron` with no other settings, on the chip in the advanced lab setup required a time of about 1.4 s for 10 ms wall-clock time. But changing from 10 ms to 100 ms results in a significant difference between simulation and emulation. Here the emulation took only 3.5 s while the simulation required 15 s to perform all necessary calculations. An optional task could be made to evaluate these performances based on different wall-clock time scales. In the discussion part will be some points mentioned regarding this difference (section 3.4).

The next task is to create a firing neuron through applying a constant current. Thereby the code has to be modified in such a way that it now applies a constant current on the membrane. Such a curve is seen in fig. 6b. It should be possible to investigate f-I curves and potentially create a simulation to reproduce the results from section 2.2. Performing this task students will observe the required time creating a simulation versus running an emulation. A deeper discussion might also involve existing simulators for neuromorphic computing (like NEST) which are already written, optimized and available but have the same limitations as the self written.

The last task focuses on implementing a time dependent current. Introducing this relates the primary to the second goal of the complete task which was to raise awareness for the complications that arise with more complex features. Neither a propagating spike is implemented nor is this neuron interconnected to others.

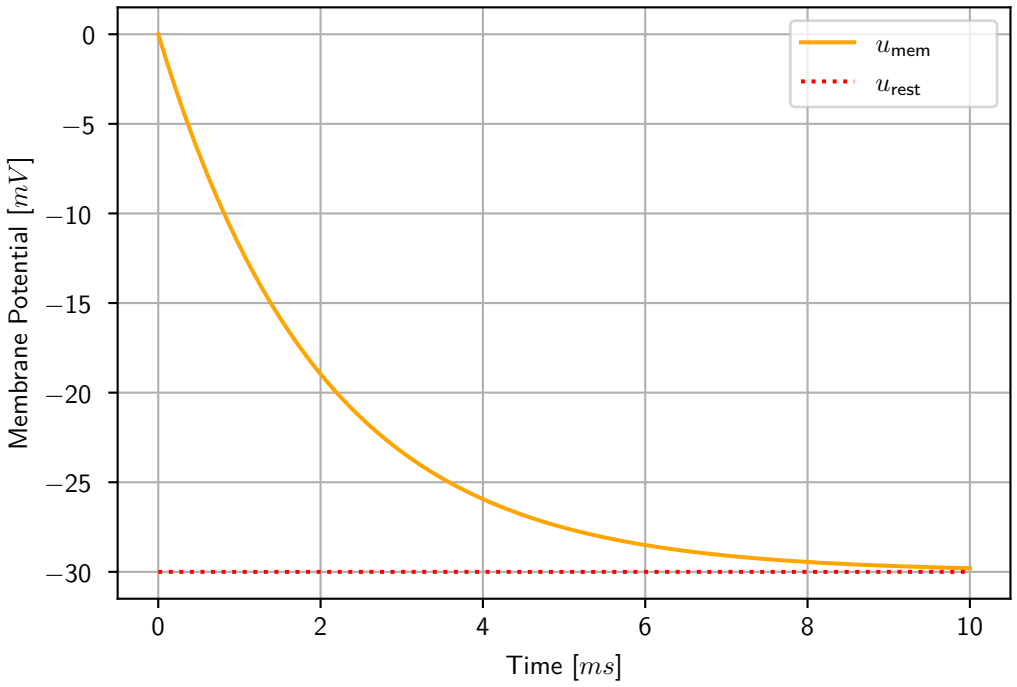
2.6 BrainScaleS-2 f-I Curve

In this task students should record a f-I curve of a BrainScaleS-2 neuron. The goal is to apply the already deduced equation and understand, that it is the same model but with a lot higher firing frequency. Also, it is possible to fit the model onto the recorded data as it was done in fig. 7.

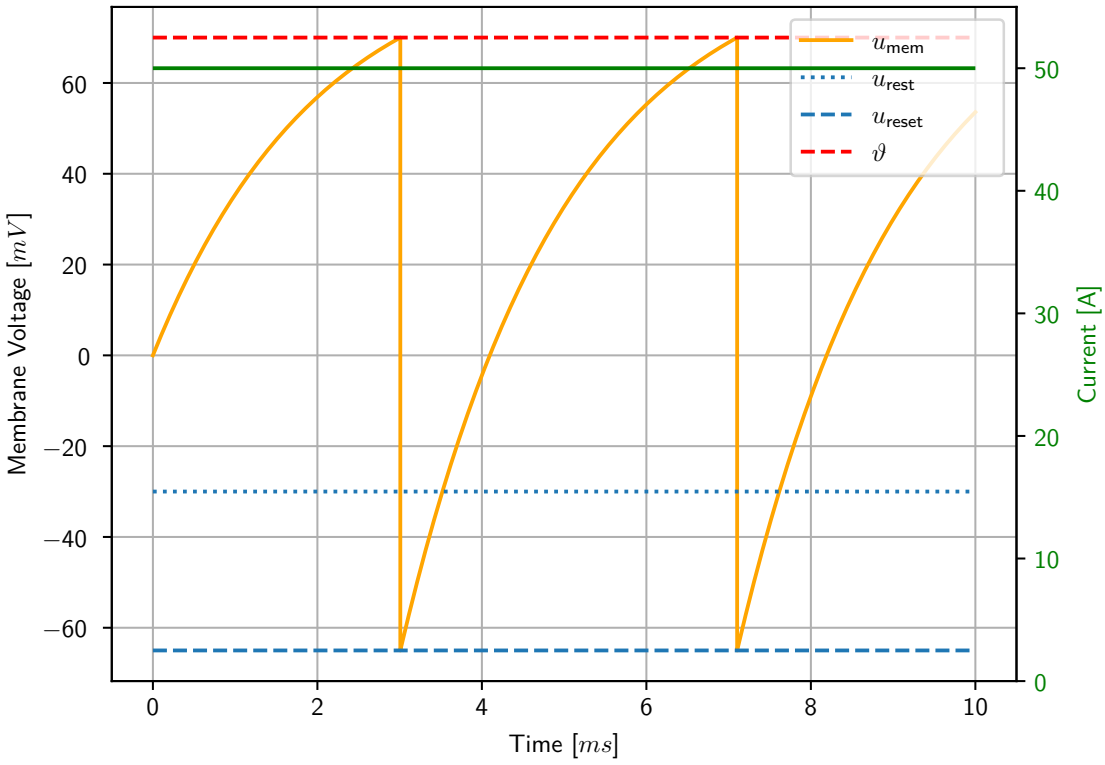
For gaining more information from this fit one would need to calibrate the axes properly (that would be possible but is not yet part of the advanced lab course). In this instance it is not useful to talk about the fitted parameters because they do not carry any information. The measurement lacks units and therefore the fitted parameters are not comparable.

2.7 Demostrating PSP Stacking

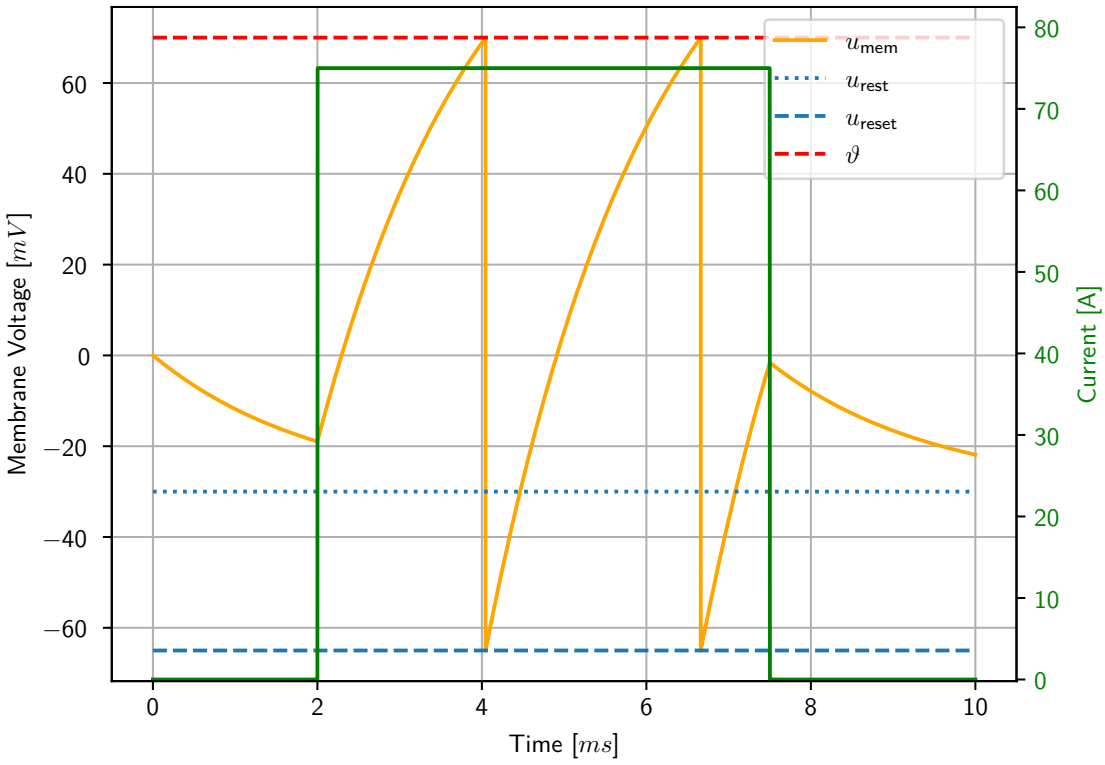
The goal of this task is to look at the behavior of the chip regarding PSP stacking and comparing it to fig. 1. The students should learn hereby the influences of synaptic inputs by toying around with weights, number of presynaptic neurons and different refractory times. This knowledge can later


u_{mem}
u_{rest}
Membrane Potential $[mV]$
Time $[ms]$
0
−5
−10
−15
−20
−25
−30
0
2
4
6
8
10

(a)


u_{mem}
u_{rest}
u_{reset}
ϑ
Membrane Voltage $[mV]$
Current [A]
Time $[ms]$
60
40
20
0
−20
−40
−60
50
40
30
20
10
0
0
2
4
6
8
10

(b)



(c)

Figure 6: A simple LIF-simulator in action for 10 ms runtime. The following times are averages for 1000 runs. In fig. 6a it took 1.41 s to simulate. Here the membrane potential started at 0 V and no input was given. In fig. 6b a constant current was set and to simulate this took 1.58 s. For the right figure a time dependent current was used, which started at 2 ms and stopped at 7.5 ms. It took 1.65 s to simulate.

be used to improve on the Sudoku task and having some discussions about the neuronal inputs and the difficulties with silicon circuits. Such PSP might look like fig. 8.

2.8 Reporting on AdEx notebook

The LIF model captures only the basic behavior of neurons described in section 1.2. In this task the AdEx model (section 1.2) shall be introduced to the students. The goal is to show the impact of the adaptation and exponential term in the equation by creating specialized behaviors like a transient spike under a constant current, which is not possible with the LIF model (as it can be observed in fig. 6c whereby the neuron spikes all the time and is simulated). To do so, the students are presented with a Jupiter notebook created by Billaudelle 2023.

To observe the features of theAdEx equation we need to measure two parameters, namely the

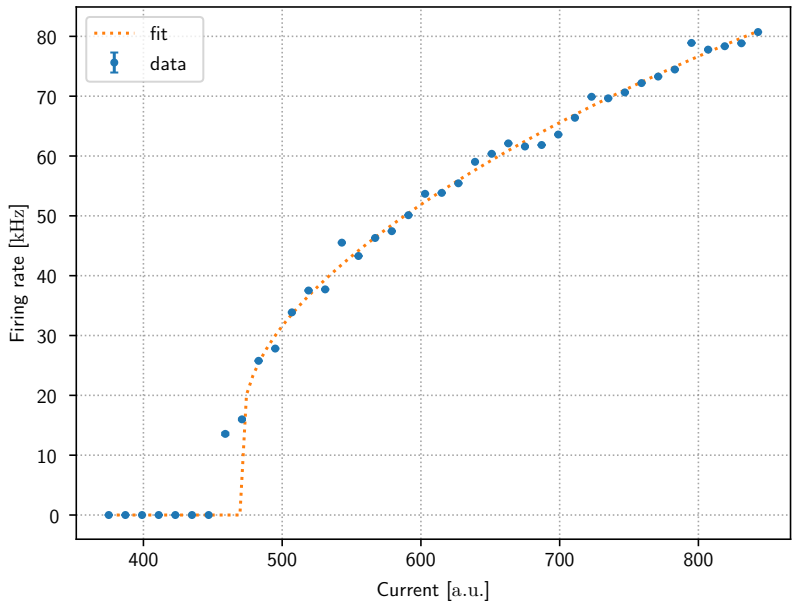


Figure 7: f-I-curve on BrainScaleS-2. This was recorded for one neuron over a runtime of 3 ms. The data point is the mean value of the ISI with standart deviation as error. Errors are small and by the look deterministic. For no recorded spikes in the time frame of the runtime a frequency of 0 was assumed.

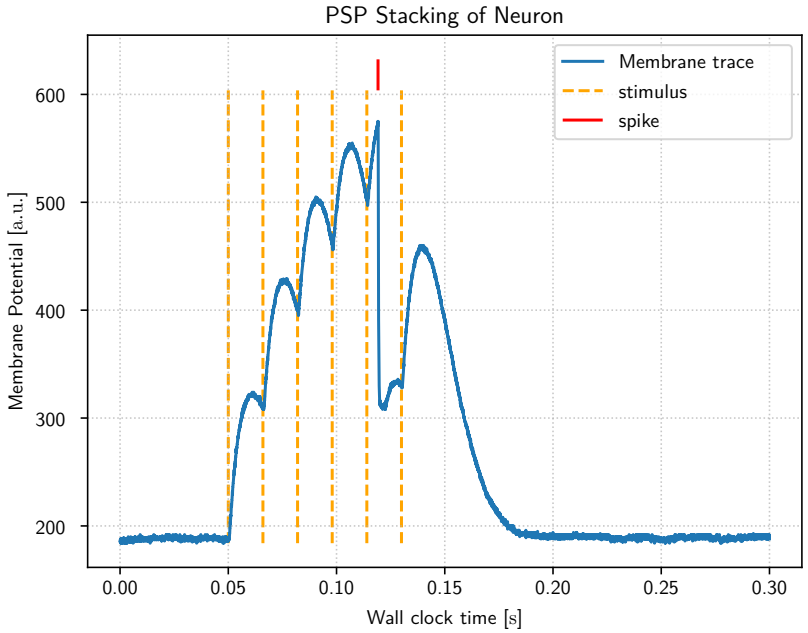


Figure 8: PSP-Stacking observed on BrainScaleS-2. 3 stimulation neurons are connected excitatory to one neuron and fire 6 times in a time range starting at 0.05 ms up to 0.13 ms equidistant. A HXNeuron was used with `leak_v_leak` = 200, `threshold_v_threshold` = 400, `refractory_period_refractory_time` = 50 and `weight` of 63.

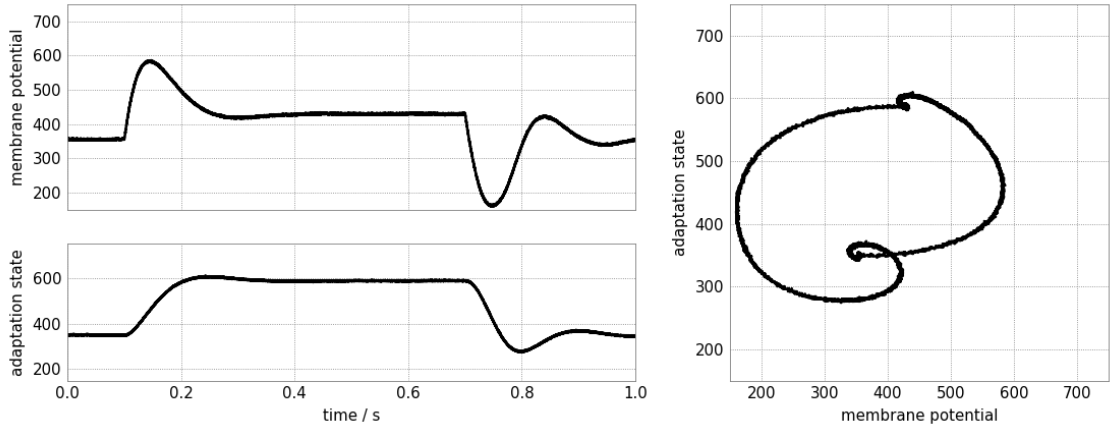


Figure 9: Damped oscillator behavior of an adex neuron. This is achievable with only changing the adaptation bias.

membrane potential and the adaptaion current. This is achieves by creating a population of 2 neurons, whereby one neuron is set as dummy and interconnected in such a way to mirror physical values of the first one. This is required because the hardware can only measure one analog value per neuron. We observe the membrane voltage and spiking for one neuron and for the second one we observe the adaptation-state.

The experiment is than set up to run for 1 ms wall-clock time, activating a current at a time of 0.1 ms and deactivating it at 0.7 ms.

In the first task "Understanding Subthreshold Adaptation" the students are presented with 6 configurable parameters, grouped into 3 different categories. First group **Leak and reset** contains the variables `leak bias` and `leak potential` which can be associated with g_l and u_{rest} in eq. (4) (here g_l is used as $\frac{1}{R}$ and is another representation of the eq. (3)). Next group **Adaptation** has the variables `baseline potential` (w / I_{ad}), `time constant bias` (τ_w) and `a bias` which is a. The last group **Stimulus** contains `stimulus current` which is the magnitude of the current input I_{stim}. The task to correlate variables with the names in the equation should increase the understanding of the impact of the parameters on the resulting behavior.

For this task the primary goal is to see which impact the adaptation term has. By tweaking this parameter, it is possible to obtain a plot like in fig. 9.

It should be well observable that the adaptation term emphasizes "overshoot". This is why the membrane potential rises higher than in that case when there is no adaptation.

The second task, "transient spike", focuses now on the exponential term of the model. With the previously gained knowledge of the adaptation term the student should try to create a transient spike. This will show a new feature of the AdEx model in comparison to LIF and recreate another function of neurons. After setting the parameters correctly, they should observe a transient spike like in fig. 10.

The observations from this experiment, that the students can comment on, are the impact of the exponential term from a certain point (point of no return), when the membrane potential

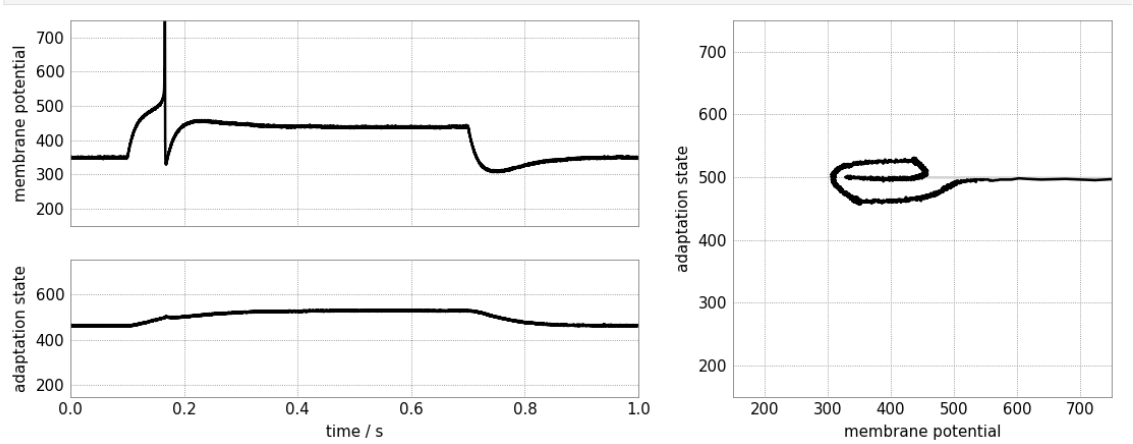


Figure 10: Transient spike on AdEx. Only one spike occurs during stimulation time.

raises independently of the incoming current and secondly the behavior of the adaptation state. Additionally, it would be an interesting task to find some use cases of this feature. The primary goal should be to motivate the student to think about and arguing why it could be useful in either biological sense or in neuromorphic computing. My ideas would be to trigger a single output under a constant input, more precisely speaking detecting a rising edge in a digital signal or reducing power consumption by reducing spikes number, whereby the first is a more computational way of thinking about it while the latter has both, computational as well as biological impacts.

The next task, "spike-frequency adaptation", introduces a new concept, combining both previous introduced terms. This time it is required from the students to adapt both terms to achieve a continuous spiking starting with the stimulation till the end of it. The new feature which is introduced in this task is an adaptation of the spike frequency or a lengthening of inter-spike intervall (ISI) during stimulation. Observing this phenomenon (fig. 11) is the main goal of this task.

Also, as in the task before, it would be nice to add a question in which sense it is useful to have such a feature in biological and computational sense. Good options to mention are applications where priorities and status are relevant to sort. Thereby a newly encountered observable has a higher priority due to higher spiking and can also, as seen in postsynaptic potential (PSP) stacking exercise activate a neuron while at a later point it can not.

In the last exercise, "bursting", another feature of AdEx is introduced and exploited. The students have to find parameters which allow the neuron to fire repeatedly during stimulation in groups of several spikes, called "burstings" (fig. 12).

As previously, additional question about use cases might make sense to motivate the usefulness of this feature. Like in the task before, in this configuration the neuron could trigger under stimulation other neurons and thereby periodically (not like spike-frequency adaptation). Another use case could be to detect single events that than trigger a continuous bursting.

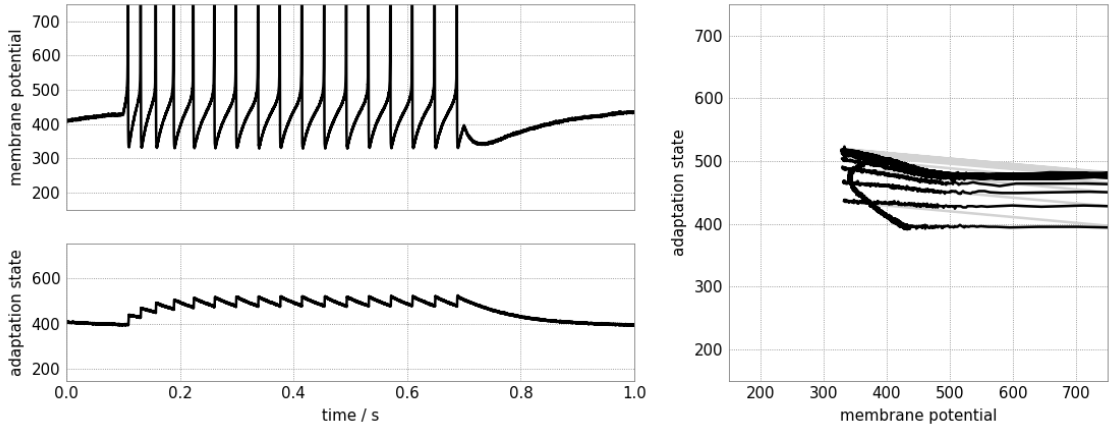


Figure 11: Spike frequency adaptation. Configuring an AdEx neuron in such a way, that it starts firing under stimulation but reduces the frequency during it.

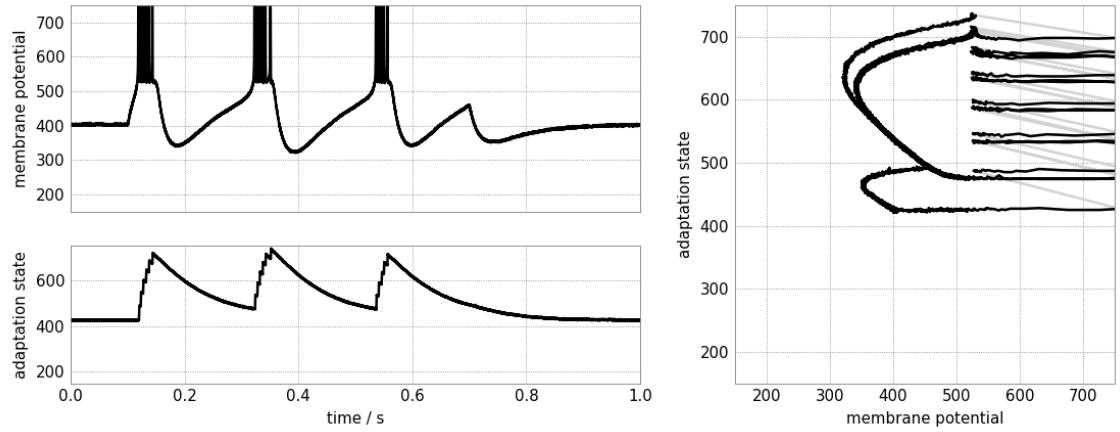


Figure 12: Bursting. Here the AdEx neuron is configured to fire periodically in groups of spikes.

2.9 Cleaning up the Sudoku Task

Previously described tasks highlight key features of the different models and the hardware. This tasks therefore combines this key features into one applied task while being tangible to students.

The main goal is now to use previously gained knowledge to solve this task. This includes an advanced application of the `pynn.brainscales2` API, an understanding of rate-based coding and the interpretation of neuronal dynamics and topology.

For the experiment, a Sudoku with a 4×4 grid is provided, each cell having 4 neurons assigned to it, whereby each neuron represents one possible number of the cell (one neuron equals 1, second equals 2 and so on). Additionally, a random noise source with a Poisson distribution is connected to each neuron individually. Initially, all neurons are excitatorily connected to themselves (a self-excitation).

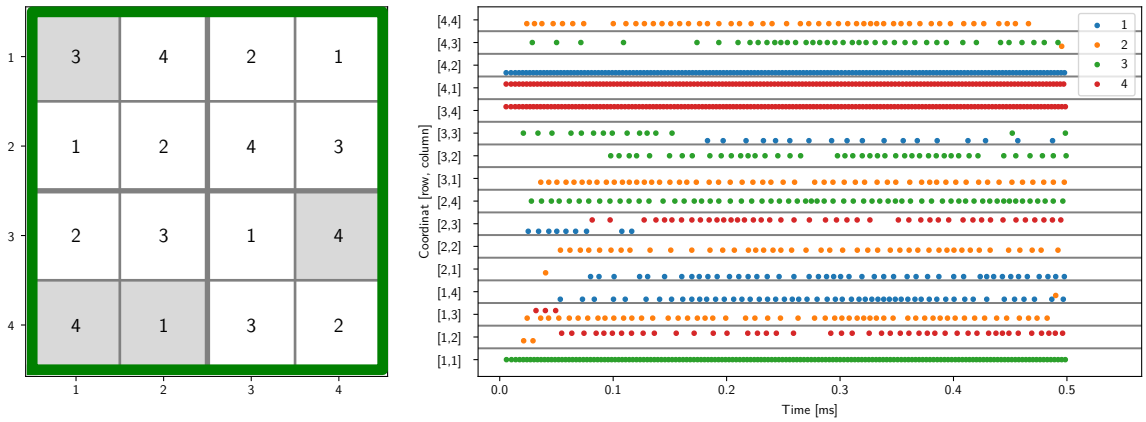


Figure 13: Left-hand figure shows a correctly solved Sudoku with 4 hints (gray colored cells). Right-hand figure represents the 4 neurons of each cell and their spike-trains. It is observable that the hints are continuously spiking.

Further a completed Sudoku is given to the solver, which chooses 8 random cells to be hints. These hints represent correct numbers of the Sudoku and have to be integrated into the network by connecting them to a spiking source, which spikes continuously throughout the runtime of the experiment.

At a later task students should describe, based on their observations, how the network decides, which number is correct. By either looking at the spike patterns (as observed in fig. 13) or investigating the code, they should find, that the neuron with the most spikes is assumed to be the right one.

But before that they have to implement the constraints to the network by writing some code, that interconnects the neurons. These constraints are discussed in the introduction of this experiment and are straight forward, hereby the main difficulty is to describe them correctly in code. If this is successful, the Sudoku should be solved correctly. Now further tasks let them work with this network.

One of these tasks is to let the network run repeatedly without hints. It is expected, that a correct solution is found, and it does not vary between the runs due to fixed-pattern noise. Another task is to try different numbers of hints and see if the solver still can solve the Sudoku in a correct way. Here the goal is to observe the previously described method of determination of a correct solution by looking at the highest spike count. If the determination fails as in fig. 14, the students are required to try to find out why it failed and try to come up with a solution to minimize such failures.

Therefore, they are required to look at different parameters like runtime, connection strength and method of determination in order to find either a sweet spot for the solver or argue which impact which parameter has on the correct solution.

Advanced students might use for argumentation previous tasks like PSP stacking or f-I-curves. As an example, it is observable, that the solver is more likely to find a correct solution the longer the runtime is due to the method of determination of the solution. Having a shorter runtime leads to more failures due to the fact, that the correct neuron did not have enough time to spike. Another option that can be discussed and observed is to change the method of determination to e.g. the

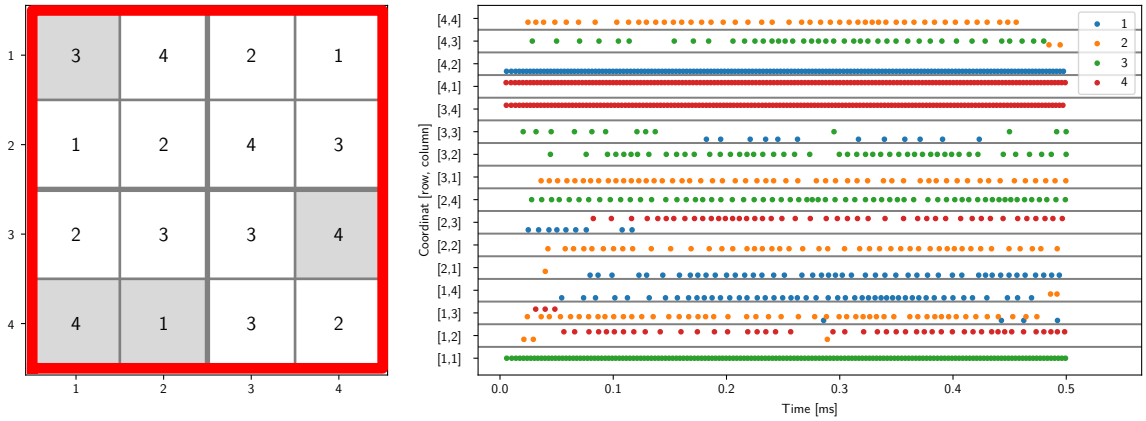


Figure 14: Left-hand figure shows a wrongly solved Sudoku with 2 hints (gray colored cells). The figure on the right hand depicts the corresponding raster plot. Comparing it to the "correct" Sudoku as seen in fig. 13 it becomes obvious, that the difference is in cell [3, 3]: For the correct solution the neuron for the number one spiked 18 and for the number three 15 times. In the wrong case the number one neuron spiked 12 times while the number three neuron spiked 13 times.

last spiking neuron. Doing so might improve the solver to a certain degree, because the neural network tends to favor the correct neurons to spike. At the same time, it happens, that "wrong" neurons fire at the last moment due to either stimulation or fixed-noise. A better strategy might be to observe spikes in the last 10 % of the runtime because there the network had enough time to settle. Although this requires a long enough runtime. Solutions to these issues might be discussed and presented.

3 Discussion

3.1 Difficulties and Opportunities with the Advanced Lab Course

In this work I described tasks which are partially used or can be used in the advanced lab course. All tasks, that are newly created or modified, are required to be tested and seen, if they fit the students, what potential issues arise from either unclear formulations of tasks or from the system itself.

Designing an advanced lab course turns out to be a real challenge due to a few reasons: it has to serve certain requirements like required time and complexity but staying at the same time understandable for students who have not been in touch with the matter. This means, a machine learning task would be an amazing application of the hardware, but it would require an introduction to the theory of it, what by itself is material worth a lecture. Condensing it in a short text is a big hurdle and limits the usability of such tasks. Suppose a great introduction is written the next limiting factor comes in: machine learning tasks require a lot of time to optimize the results which is an issue because the advanced lab course should have a duration of about 2 days. An idea would be to look into an additional advanced lab course focusing only on machine learning tasks and having

as a prerequisite knowledge of the topic or be completely uncoupled of hardware. Surely it would be in the interest of the administration and the group, but it would require staff and time which are limited. Returning to the requirements of the lab course, another one is to make the course interesting and educational for students, where the first point is discussable, and the second point is mandatory. At least a simple approach for the first requirement is to either define a great goal or create mesmerizing moments.

An outlook of this work would be to connect more tasks having a common thread throughout the lab course. What that means is to interlock the tasks in such a way, that each task builds onto the previous task by integrating learned features and applying them all together in a big final exercise.

3.2 Lu.i

The Lu.i task is by itself a nice demonstration of analog computing. With the current design of the exercises the function of the LIF equation is introduced in a comprehensible fashion. It accomplishes the requirement of the advanced lab in multiple points, first it requires an understanding of equations and a derivation of it, a recording of data points with statistics and last combining expected model with real life measurements, though the fitted parameters do not match real components.

If the students set the leak potential to 0 V and do not fit it, the components values can be seen in table 2.

There are further options to expand this task: One could create a spiking network with Lu.is to observe them in real life before one works with BrainScaleS-2 spiking networks. Although this task is highly educational, it might not be suitable for the advanced lab course due to the required time to set it up and lacking measurable quantities. It would be expected to take the students at least 1.5 h to get it done and then there would not be enough time to introduce more advanced concepts on the BrainScaleS-2 system. A further task, which can be proposed, is to build the current source. Thereby, many types of building might be proposed, from soldering it on a PCB up to assembling it on a breadboard. Since it only requires 4 components, it is accessible, although the first proposal would the students require to have knowledge and experience in soldering. This can be handled by either requiring completing another advanced lab course or doing it at their "own" risk. A supply of components and PCBs is also mandatory at the expanse of the group. But this takes time as well, and reduces the length of other tasks which might focus more on neuromorphic engineering. Putting components on a breadboard is in my opinion not really educational due to the beginners' lab course where all physics undergraduates had to build an AM-receiver, therefore it is a "time waste" without conveying new concepts.

3.3 AdEx

The AdEx demo is a good showcase of the BrainScaleS-2 system because it features the underlying model of the chip and uses the "full" potential of the available neuron model. Further it introduces a lot of new building blocks for network architectures that require in standard computing many components, not only "one" neuron. Under the use case of the advanced lab course, it might lack a few important aspects. A minor addition would be short tasks to confront the students

with applications of the introduced modes of the AdEx. Currently, it is good, that it is quite accessibly structured, and the tasks are straight forward. At the same time, this might be too easy for an advanced lab course where student should be tasked to work with scientific problems. Therefore, parts of the notebook should be more focused on not only configuring parameters but analyzing them or thinking further about them. As already suggested in section 2.8 good additions are questions about the application of the different modes in biology as well as in neuromorphic computing.

Another challenge is to find a common thread regarding the advanced lab course. The features that are introduced stand out and are not yet embedded in the flow of the course. It might be useful to investigate interconnections or relations to other demos that are used for the advanced lab course to apply newly gained knowledge onto real problems. I can image that the work from other group members would fit well into this framework at some point. Such a showcase could be logic gates as they were introduced on an old system but this time with more sophistication. The operations would not be rate-based encoded but single spike. In this scope questions like solving time, energy consumption or transistor numbers for different logical operations would be proposed and with enough information answerable comparing neuromorphic with digital hardware.

To come back to what was mentioned before an interconnection to the Sudoku solver could have some potential due to option of fire frequency adaptation in that sense, that the network becomes more flexible in exploring the solution space. This proposal should be discussed further but is at the moment out of scope for this report.

3.4 Simulation of a Neuron

Running a simulation is trivial. But at the same time, it does not cover all required behaviors that are needed, e.g. the simulation code does not include refractory times or hyperpolarization. Also, the code has no intention to create synaptic connections. This is already done with more sophisticated simulators like NEST[1] to name one. Introducing this task to students may be a nice way to outline some pros and cons in building an ASIC as it was done. Nowadays, many problems are solved with simulations, and they are sufficient in their use cases. But they come with their cost. As indicated in this task, running a simple and short simulation of a LIF-neuron is not that time-consuming in comparison to emulating it on the hardware as long as the time frame is short. In that case a simulation takes as long as the execution on EBRAINS. Introducing longer time frames leads to significant longer simulation times while the execution time on the chip is not rising at the same proportion. This should be one major observation. Additionally, the students should observe and note that more complex behaviors come with higher computational cost than running it on the ASIC.

Further an interesting point for discussion is the simulator itself. In this notebook the Runge-Kutta scheme is used, optimized by the `scipy`-package. This integrator uses a more sophisticated integration scheme than the Euler-integration scheme. Taking different integration schemes into account would probably be a little out of scope for students who do not have any experience with

[1] https://www.nest-simulator.org/

simulations. For others, who are familiar with integration schemes and their limitations, it is a perfect argument for their discussion section.

Another thought that can be used in this section for discussion is the EBRAINS stack. To run an experiment the students use a web interface. Their experiments are scheduled through quiggeldy, an experiment scheduler. This manages all users of EBRAINS and schedules their experiments on available hardware. Students can mention the limitations of this system. At the same time, it should be communicated, that the lab setup is dedicated to the advanced lab course and therefore scheduling limitations should be negligible. But the time traversing the stack may have an impact on the experiments and limits the (current) possibilities of running complex tasks due to the initial architecture. If one would like to focus on rate-based coding an additional task would come in handy, simulating a f-I curve based on the integrator introduced in the task before. It would obviously take some time to write the code, but that is not that difficult, especially because one could assume, that only one spike is required. Another potential task could be expanding the neuron model to account for refractory time or try to implement an output to interconnect with other neurons, although this sounds more difficult and time-consuming than students can be tasked to work in the advanced lab course. This task is meant to be suitable for students who are writing an extended report and this implementing a connection would add up nicely with the PSP stacking exercise.

3.5 SudokuTask

The Sudoku notebook closes the advanced lab very well because students can apply all their previously gained knowledge about the `pynn` package and BrainScaleS-2-hardware. Second it is something that is not trivial to solve with commonly available methods. This highlights an advantage of the created chip and of analog computing. Further it requires some thoughts about the implementation in software and analyzing the results to understand where the errors arise from. It is an important concept that is comparable to a Markov chain in the sense that the system can explore many states very quickly in order to find the right one after some time.

Special thanks

I would like to express my special thanks to all the members of the Electronic Vision(s) group and especially to Dr Johannes Schemmel for giving me the opportunity to work on a very exciting project. Not only was I able to gain an insight into academic work, but I was also able to actively help shape and advance it. But I also really enjoyed the time outside of my academic work and I still use the tricks I learned playing table football.

I would also like to thank Yannik, Julian and Sebastian, who have always welcomed my requests with an open door in their office and have thought them through to the last detail and given me good guidance.

Thank you to all those who took the time to read and re-read this work several times and gave very detailed feedback. Without you, this work would have much more ambiguity, complicated wording and inconsistent style. Thank you for teaching me so much!

I'm sure I didn't manage to mention all the people who made me feel so good when I wrote this project. So here are my thanks to all of you guys: Jakob, Philipp, Joscha, Simon, Carl, Tim, Kasper, Erik, Björn, Nils, Andreas, Dan and Jose.

The work carried out in this project used systems, which received funding from the European Union's Horizon 2020 Framework Programme for Research and Innovation under the Specific Grant Agreements Nos. 720270, 785907 and 945539 (Human Brain Project, HBP) and Horizon Europe grant agreement No. 101147319 (EBRAINS 2.0)

Glossary

AdEx Adaptive Exponential Integrate-and-Fire. A more sophisticated model of LIF model with an exponential term to capture fast burst and an adaptation term to account for neuron phenomenons like only one spike for some time or groups of burst. 1, 6–8, 16, 18–20, 23

ASIC application-specific integrated circuit. 6, 7, 24, 26

BrainScaleS-2 Accelerated neuromorphic hardware for emulation of spiking neuronal networks. 1, 6–9, 11, 13, 14, 17, 23, 25, 26

CNS central nervous system. 3

EBRAINS Online plattform to access BrainScaleS-2 hardware. 1, 9, 12, 14, 24

EPSP excitatory postsynaptic potential. 3

FPGA A field programmable gate array is an integrated circuit that can be, in comparison to an ASIC, reprogrammed after manufacturing.. 8

IPSP inhibitory postsynaptic potential. 3

ISI inter-spike intervall. 17, 19

LIF Leaky fire-and-integrate. A basic model to describe neuron dynamics. Consists of a differential equation and a reset condition. 1, 5, 6, 8, 10, 13, 16, 18, 23, 24, 26

Lu.i A simple PCB Board with circuitry to run a LIF Model. It has 3 dendrites which can be configured to be excitatory or inhibitory, tuneable integrator parameters and a spike generation circuit. For more insight github: Lu.i. 1, 10–13, 23

ODE ordinary differential equation. 6

PSP postsynaptic potential. 3, 4, 14, 17, 20, 21, 25

References

Billaudelle, Sebastian (2023). *Complex neuron dynamics with a silicon adaptive exponential integrate-and-fire neuron*. URL: https://github.com/electronicvisions/brainscales2-demos/blob/master/ts_08-adex_complex_dynamics.rst.

Brette, Romain and Wulfram Gerstner (Nov. 2005). "Adaptive Exponential Integrate-and-Fire Model as an Effective Description of Neuronal Activity". In: *Journal of Neurophysiology* 94.5, pp. 3637–3642. ISSN: 0022-3077. DOI: 10.1152/jn.00686.2005. URL: https://journals.physiology.org/doi/full/10.1152/jn.00686.2005 (visited on 12/21/2023).

Electronic, Visions (2023). *brainscales2-demos*. URL: https://github.com/electronicvisions/brainscales2-demos/tree/master. (accessed: 07.11.2023).

Gerstner, Wulfram et al. (2014). *Neuronal dynamics. from single neurons to networks and models of cognition*. eng. Literaturverzeichnis Seite 547-572 ; Hier auch später erschienene, unveränderte Nachdrucke. Cambridge: Cambridge University Press, xi, 577 Seiten. ISBN: 978-1-107-63519-7.

Lapique, L. (1907). "Researches quantatives sur l'excitation electrique des nerfs traitee comme une polarization". In: *Journal of Physiology, Pathology and Genetics* 9. English translation available https://link.springer.com/article/10.1007/s00422-007-0189-6, pp. 620–635. URL: https://cir.nii.ac.jp/crid/1570291225205713280.

Pehle, C. et al. (Feb. 2022). "The BrainScaleS-2 Accelerated Neuromorphic System With Hybrid Plasticity". In: *Front. Neurosci.* 16.795876. DOI: 10.3389/fnins.2022.795876.