AdEx model with two adaptive terms and its restrictions due to hardware limitations of the HICANN chip

Project lab report by
Alexander Kononov

email: *kononov_alexander@hotmail.com*

supervised by
Thomas Pfeil

May 9, 2010

Contents

1 Introduction

1.1 Motivation And Goals

The neuromorphic hardware [1] developed by the research group Electronic Vision(s) [2] at the Kirchhoff Institute for Physics of the University of Heidelberg in cooperation with TU Dresden is planned to be used for neuroscientific experiments by FACETS [3] research collaboration. However, it is not yet fully functional, so future experiments must be currently simulated in software with an intention to port them to the hardware as soon as it is ready to use.

This is why we need a software model which describes the hardware as exactly as possible and incorporates all of its advantages and disadvantages. Creating such a model using up-to-date information known about the hardware will be the first task of this project lab.

It is also of great importance to customize the simulation data according to the hardware demands and restrictions. This could change the outcome of an experiment, or even make certain experiments impossible to run on the hardware.

The second part of this project will be dedicated to analyzing the effects of using input current data with limited time resolution. This is done with an aim to use existing current input circuits for single neuron emulations which is a subject of the Qualitative Single-Neuron Modeling Competition (see below).

1.2 Adaptive Exponential Integrate-and-Fire Model

The hardware chips internally are using a mathematical model, which can describe the neuronal behaviour: predict membrane potential, spike times etc. The model used in hardware is called Adaptive Exponential Integrate-and-Fire Model.

Adaptive Exponential Integrate-and-Fire Model (AdEx) according to Brette and Gerstner [4]

$$C\frac{dV}{dt} = -g_L(V - E_L) + g_L\Delta_T exp\left(\frac{V - V_T}{\Delta_T}\right) - g_e(t)(V - E_e) - g_i(t)(V - E_i) - w \quad (1)$$

$$\tau_w\frac{dw}{dt} = a(V - E_L) - w \quad (2)$$

$$\text{At spike time} : V \to E_L; w \to w + b \quad (3)$$

AdEx is a two-dimensional integrate-and-fire model that provides an effective description of neuronal activity. It takes into account the leakage current, the synaptic currents and external currents. The spike mechanism is modeled as an exponential function, the adaptation term (2) as an exponentially falling current, the synaptic conductances $g_i(t)$, $g_e(t)$ are usually modeled as exponential or alpha-functions[1].

AdEx model predicts correctly the timing of 96% of the spikes (± 2 ms) of a detailed model (according to Hogkin and Huxley) [4], but it is much more simple and allows complicated simulations with far less effort. Variables and parameter are:

[1] Alpha function has the form: $g = g_{max}\frac{t}{\tau}exp\left(\frac{\tau - t}{\tau}\right)$

- V - membrane potential
- C - membrane capacitance
- g_L - leak conductance
- E_L - leak reversal potential
- V_T - spike threshold
- Δ_T - slope factor
- w - adaptation current
- τ_w - adaptation time constant
- a - subthreshold adaptation
- b - spike-triggered adaptation

1.3 NEST

NEST or Neural Simulation Tool [5] is a software package for Linux developed and distributed by the "NEST initiative" – a collaboration of scientists for the development of simulation methods for biologically realistic neuronal networks. This tool is a powerful simulator for single neurons as well as complicated networks with many neurons in various topologies.

NEST is modulary structured. It means that single devices, neuron models or synapse models used by NEST are separate pieces of software written in C++. These modules can be added, altered or changed for use in the whole package according to the demands of certain simulation processes. This feature of NEST is very interesting and will be used in the course of this project to create new models or alter the existing ones for further use in simulations of hardware neurons.

1.4 Python, PyNEST, matplotlib, NumPy

NEST uses its own scripting language called SLI to construct the simulation. This can be a difficulty for many users, due to SLI's stack based syntax. This can be solved by using the extension module called PyNEST [6] which is a Python [7] wrapper around NEST.

Python is a general-purpose programming language, but because of its readability and functionality it is often used as a scripting language, as in our case too. A very useful feature is the ability to import external libraries for using with Python. This way it was possible to use Python scripts for designing all of the simulations in this project lab.

Using PyNEST instead of NEST has the advantage of using a more convenient syntax along with some other useful external packages for Python such as matplotlib [8] or NumPy [9].

Matplotlib is a Python 2D plotting library that can be used in scripts, i.e. one can visualize the data directly from the neural simulation script. This is very useful, as it saves us the time and effort to buffer the data on the hard drive and visualize externally.

NumPy is a fundamental Python package for scientific computing. It is used to create and handle arrays of data, import and export data to/from the hard drive and apply algebraic computing to the data arrays.

1.5 Neuronal Hardware And Its Limitations

The neuromorphic hardware (i.e. the HICANN chips and their wafer-scale analogue) is currently being developed by Electronic Vision(s) Group. With all the benefits in speed, it is also expected to have some limitations, constraining all of the parameters of the AdEx model. The exact relations will be discussed in the corresponding section below.

Also the memory writing rate of the HICANN chips is restricting the maximum resolution of the input current data that can be used. I.e. for the continuous current input there has to be a time grid defined on which the data has to be embedded. Also the grid has to be chosen carefully, because of the limited time resolution.

1.6 Qualitative Single-Neuron Modeling Competition 2009

The data, which is used in the second part of this lab to simulate the behaviour of a biological neuron originates from the neuron competition, which took place in 2009 [10]. The goal of the challenge was to predict spike times with accuracy of ± 2 ms using different neuronal models. The challenge provided biological data for several seconds of current input and membrane voltage responce as training-data which we used for our simulation purposes.

2 Part I: Extending The Existing AdEx Model For NEST

At the beginning of this project lab NEST had already got an AdEx model originally implemented by Marc-Oliver Gewaltig [5]. The program itself is distributed as the source code in C++ and the permission to modify it is given by the authors. The modular structure of the program gives the ability to integrate the modified models immediately without recompiling the entire NEST software.

2.1 Altering AdEx Model For NEST To Match Hardware Givens

The model implemented by Marc-Oliver Gewaltig was an Adaptive Exponential Integrate-and-Fire Model of the form (1) with synaptic conductances shaped as Alpha-Functions. It included one adaptation term w (2),(3). However, the model implemented in hardware has synaptic conductances shaped as exponential functions[2].

It is also possible to interconnect two hardware neurons to use them as one with two different adaptation terms, which could be useful in the latter simulations, so our software model should have two adaptation terms as well (the second one should be set to zero when not used).

2.1.1 Modifying The Form Of Synaptic Conductances

The general idea how the software model works is the following: A general-purpose integrator from the GNU Scientific Library [11] is used. It demands several things as input: A function containing the mathematical model as a system of ordinary differential equations, buffers to store the transitional solving information, current model parameters and state variables (e.g. membrane potential, adaptation current).

At first the model parameters are initialized to their values: membrane potential to reversal potential; synaptic conductances and adaptation terms to zero.

Then they are passed on to the solver (integrator), which uses Runge-Kutta-Fehlberg method to make an integration step of the predefined length. If the spike is emitted ($V > V_{th}$ after the step), then the membrane potential and the adaptation term are corrected accordingly (3). The new state variables are then logged using internal NEST functions and the new state becomes initial condition for the next integration step. This is repeated until the end of simulation. At the end the progression of the relevant variables is stored in memory and can be processed further.

To change the form of synaptic conductances, the differential equations for the alpha functions had to be replaced in the source code by the corresponding equations for exponential functions: $\frac{dg_e}{dt} = -\frac{g_e}{\tau_e}$ and $\frac{dg_i}{dt} = -\frac{g_i}{\tau_i}$. Where τ_e, τ_i are slope factors for excitatory and inhibitory synaptic conductances respectively.

The result can be seen in Figure 1. Both graphs are simulation data of excitatory synaptic conductances with a spike coming in at 10.0 ms. The time constants are equal in both cases. Left is the model before alteration, on the right the alpha-function has been replaced by exponential slope.

[2] $g = g_{max} exp\left(-\frac{t}{\tau}\right)$

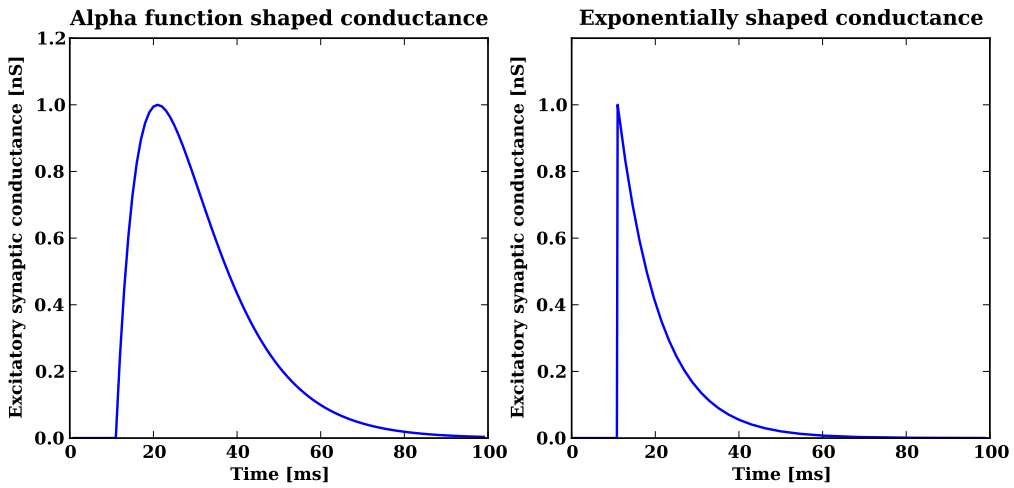


Figure 1: Two forms of synaptic conductances

2.1.2 Implementing A Second Adaptation Term

As mentioned before, one will be able to connect two hardware neurons to act as one with two adaptation terms. To match this in software, the differential equations in the source code had to be changed accordingly, and some new parameters had to be added. The final form of our new equations is the following:

$$C\frac{dV}{dt} = -g_L(V-E_L)+g_L\Delta_T exp\left(\frac{V-V_T}{\Delta_T}\right)-g_e(t)(V-E_e)-g_i(t)(V-E_i)-w_1-w_2 \quad (4)$$

$$\tau_{w_1}\frac{dw_1}{dt} = a_1(V-E_L)-w_1;\ \tau_{w_2}\frac{dw_2}{dt} = a_2(V-E_L)-w_2 \quad (5)$$

$$\text{At spike time}: V \rightarrow E_L;\ w_1 \rightarrow w_1+b_1;\ w_2 \rightarrow w_2+b_2 \quad (6)$$

The idea behind this model expansion is that biological neurons show not only a short-term adaptation of spiking behaviour, but also long-term adaptation, which can be described by our new model [12]. By setting the appropriate time constants τ_{w1}, τ_{w2} and adaptation parameters $a_1,\ a_2,\ b_1,\ b_2$ one can set up the short-term and long-term spiking behaviour of a simulated neuron.

Figure 2 shows example adaptation currents for two cases with equal input currents: on the left only the short-term adaptation is active (time constant $\tau_w = 70$ ms), on the right - both terms are active, and long-term adaptation parameters a and b are set to be (for demonstration purposes) $\frac{1}{3}$ of the short-term ones ($\tau_{w1} = 70$ ms, $\tau_{w2}= 500$ ms). Both subthreshold and spike-triggered adaptation currents now have two decay constants. One can observe this effect considering the same general shape of the curve (mostly characterized by the short-time term) but longer decay times (characterized by the long-time term).

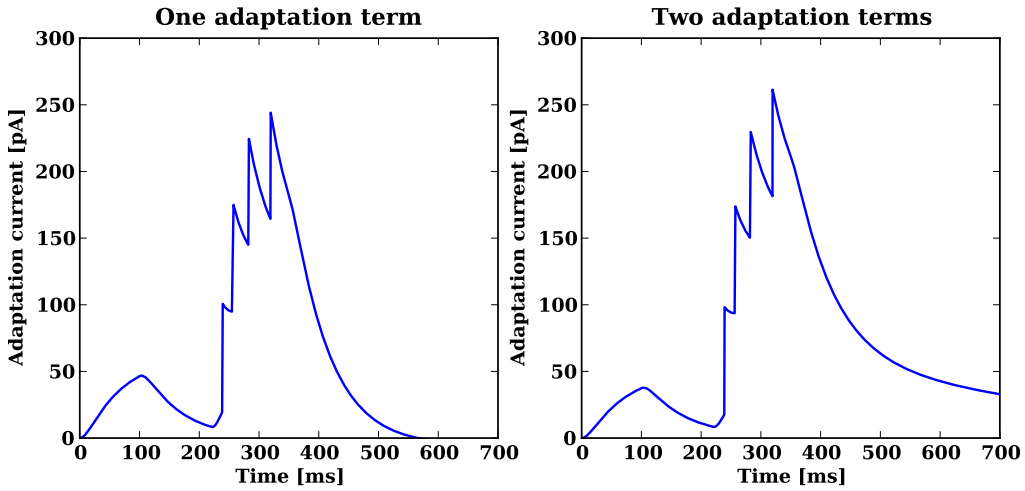


Figure 2: Adaptation currents with one and two adaptation terms

2.1.3 Adding Data Logging Device

NEST uses internally certain devices to log the data produced during simulations. Formerly there had to be a separate device created for every kind of information to be logged. A shortly introduced device called multimeter is intended to solve these problems and ease up the creation of simulations. It can log many values simultaneously. However, the NEST model has to be properly modified for use with multimeter.

In the course of this project lab the AdEx model for NEST was altered to use multimeters to log the values of membrane potential, excitatory/inhibitory conductances and adaptation currents.

2.1.4 Making A Unit Test

As NEST is a modular software, one has to be sure that changes in one module do not affect flawless working of the other modules. This is why one has to introduce a unit test along with a new model. The unit tests are usually run upon a new installation to assure everything is ready to use.

The unit test created for the new model consists of a short simulation using all new features. During this simulation a neuron is stimulated by a DC current and excitatory and inhibitory spikes are received. Membrane potential and spike behaviour are then logged and compared to the expected values. Success is returned at the end of the test if every compared value matches the predicted one.

This way one can keep track of changes in the program and be sure that the simulations still provide correct data if a part of the program is altered (e.g. the integrator routine is changed etc.).

2.2 Applying Hardware Restrictions To The New NEST Model

The HICANN chips are currently being analyzed by simulating the implemented integrated circuits. During these tests all the hardware parameters and their behaviour (e.g. linear, exponential etc.) are analyzed and compared to theoretical ones to assure, that the hardware is capable of emulating biological data properly. The limitations and offsets of single parameters are also determined.

2.2.1 Ranges Of Parameters And Variables

Due to technical limitations (e.g. minimum/maximum supply voltage or current), none of the parameters or variables of the AdEx model can be just set to any value at random. There are only certain ranges in which the varying of the parameters is safe. Setting a parameter outside of its range will cause false results. Moreover, some parameters are interdependent, and there are some completely new effects not considered by the model.

This is why the restrictions have to be implemented in software first - to be able to determine if it is meaningful at all to try to emulate certain data sets in hardware.

The first and most meaningful restriction is concerning the voltage. Due to limited linearity region of certain components, the hardware voltage has a working range of V_{HW}=[0,800 .. 1,200]mV. This means one has to comply with the resulting range of 400 mV during emulations. At the same time one would want the used voltage range to be as wide as possible because the greater the range the lower the noise/signal ratio is. So the idea is to estimate the biological voltage range beforehand and then scale it properly to match the 400 mV in hardware. This introduces the formula:

$$V_{range} = S \cdot (V_{max} - V_{min}) \leq 400mV \tag{7}$$

Where V_{min}, V_{max} are the estimated minimum/maximum biological voltage values, and S is the scaling factor which has to be as high as possible, but at the same time low enough for the voltage range not to exceed 400 mV.

The relations of other parameters are simpler and are presented in Table 1 as they have been implemented in the AdEx model for NEST. Each time the parameter is set externally, its range and interdependences with other parameters are checked to comply with allowed values.

2.2.2 Offset In The Adaptation Term

During the testing of a HICANN chip it was noticed, that the adaptation circuits on the chip deliver constant output voltage even if there is no output voltage expected [13] ($V_m = E_L$). This offset voltage V_{off} was also implemented into AdEx model for NEST to examine if it could affect the results of emulations in the future.

As a result, the differential equations (5) had to be changed to take offset voltage into account. The new form of ODEs is:

$$\tau_{w_i} \frac{dw_i}{dt} = a_i(V + V_{off} - E_L) - w_i \tag{8}$$

Name	**Range/Dependence**	**Note**
V_T	-50 mV	The parameter is fixed in hardware
Δ_T	2 mV	The parameter is fixed in hardware
All voltages	$(V_{max} - V_{min}) \leq \frac{400}{S}$ mV	HW voltages must stay in 400 mV spread
$\frac{C_m}{g_L}$	[5 .. 50] ms	$\frac{C_m}{g_L} = \tau_m$ - Membrane time constant
τ_{w1}, τ_{w2}	[20 .. 200] ms	
$\frac{a_1}{C_m}$, $\frac{a_2}{C_m}$	[40 .. 400] $\frac{\text{S}}{\text{F}}$	$a = \frac{C_m \cdot a_{HW}}{C_{HW} \cdot 10^4}$, C_{HW}=2pF, a_{HW} $\in$[0,4..4]μS
b_1, b_2	[0 .. $\frac{92}{S}$] pA	S - Hardware scaling factor, see (7)
$g_{e,i}$, $E_{e,i}$	-	Yet to be determined

Table 1: Hardware imposed restrictions

This form was eventually implemented into AdEx model for NEST along with other hardware restrictions.

3 Part II: Preparation For Future Neuron Competitions

Because of the enormous emulation speed (10^4 times faster than in-vivo processes) the data flow (for example spike trains) to HICANN chips has to be very high to emulate the real biological processes. However to the date of this report it is unknown whether the input rate for current data to the hardware will be high enough to use the full resolution of biological input data. It means that input data will have to be reduced to some degree (time resolution will have to be lowered).

In this part of the project lab we tried to simulate hardware behaviour using different input data rates to examine the robustness of the model in case of data reduction and see if it will be possible to use real biological data for future emulations, what data alterations will be needed for that and if these alterations can compromise the results of experiments.

3.1 Estimating HICANN Current Input Update Rate

HICANN current memory buffer is designed to work with 5 possible clock frequencies: the highest is 62,5 MHz, the lowest - 62,5/16 $\approx$ 3,9 MHz. These are the frequencies at which input data from a buffer can be applied to the executing circuits. It is yet to be determined how fast this buffer (containing 129 values) can be re-written. One can see, that in case of slow writing rate this buffer is depleted very quickly (33 μs at 3,9 MHz), so it is of crucial importancy to find ways to avoid it, which brings us to reduction of time resolution.

As the actual writing frequency is unknown, we assume that it is equal to the memory clock. The slower the clock - the lower writing rate can be used. Assuming we run emulation at 3,9 MHz, the delay between sequent current values is then $\Delta_{T_{HW}} = \mathbf{256\ ns}$.

In this test we use biological data from Neuronal Challenge 2009 (see above) which has a time resolution of 0,1 ms. If we take into account the 10^4 hardware speed-up factor, 0,1 ms would correspond to $\Delta_{T_{biol}} = \mathbf{10\ ns}$ in hardware time.

As one can see it is not possible to use the raw biological data to make a proper emulation. What one could do is to reduce the time resolution so that the biological current rate is equal to the hardware current rate. In our case we should shrink the raw data by the factor $R = \frac{\Delta_{T_{HW}}}{\Delta_{T_{biol}}} = \mathbf{25{,}6}$. This means, however, that the results could show significant deviation compared to results without data reduction.

3.2 Reducing Biological Data And Analyzing The Effect

To test how far one could reduce the data without compromising the results a simulation in NEST has been written using our enhanced AdEx model.

First of all we used the training data set from Neuronal Challenge 2009 to estimate AdEx model parameters which deliver reasonable results compared to original biological data. Table 2 features these parameters. Synaptic parameters are irrelevant in this case because the data represents a response to an external current with no synaptic input.

C_m	g_L	E_L	Δ_T	V_{th}	V_{reset}	τ_{w_1}	τ_{w_2}	a_1	a_2	b_1	b_2
240pF	13,5nS	-65,8mV	2,2mV	-51,5mV	-51,6mV	98ms	300ms	4,0nS	0,3nS	160pA	30pA

Table 2: Parameters used for simulation

Figure 3 shows a small exerpt comparing our simulation to biological data. It does not exactly reproduce the biology, but one can see that it is reasonably correct in general tendencies and spiking behaviour.

The reduction method is following: We are provided with an input data, which

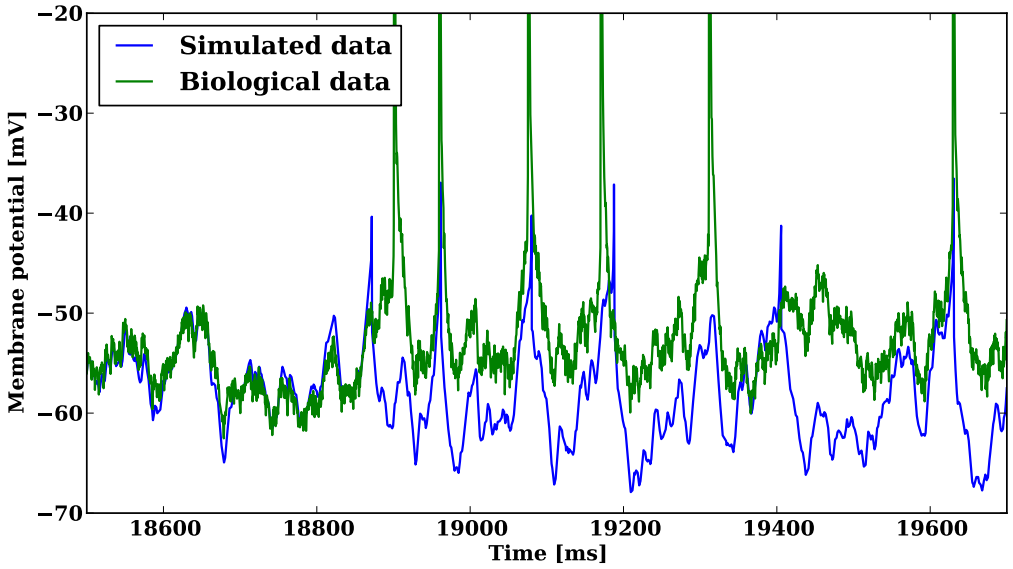


Figure 3: Comparison of biological and simulated data using our parameters

consists of current values and times on which these currents are injected into a neuron. Original time resolution is 0,1 ms. It means every 0,1 ms the input current is changed to a new value. Given a scaling factor R (in our case the desirable value would be 25,6) we calculate a new time grid (i.e. new current value every 2,56 ms). For each of these new time values a mean current value is calculated using the original data ahead of the new time point (i.e. next 2,56 ms). An example of current steps for R=15 compared with original data is given in Figure 4.

As reference for comparison we take the simulation with no reduction. The quality of reduced simulation is rated by evaluating the deviation of spike times. Neuronal Challenge 2009 suggests that spike times are a match if the deviation is less than 2ms. We also took this measure for our evaluation.

After a series of simulations with varying R we were able to construct a plot, which

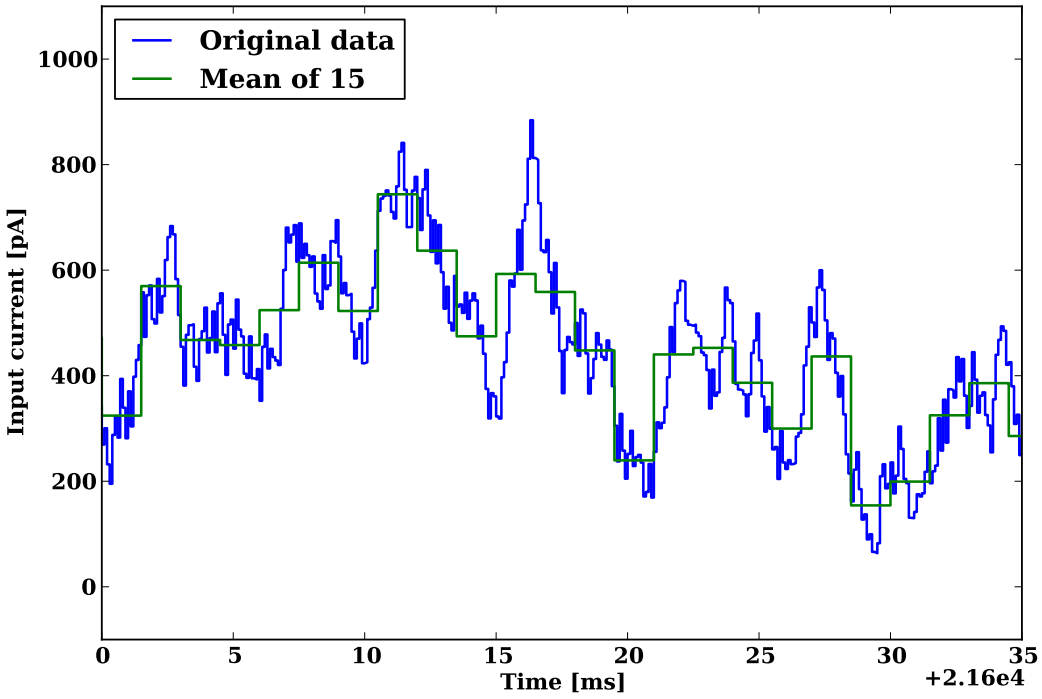


Figure 4: Input current: Unreduced and reduced by factor 15

can be seen in Figure 5.[3] This plot shows the dependence of matching spike percentages on the scaling factor.

The simulation without reduction included 177 spikes, so the statistics in this survey are only meaningful to some degree and a certain error should be considered. The error can be reduced later by making longer simulations.

Regardless of an error, one can clearly see the tendencies of the model behavior if a reduced input data is used. Up to a factor R=5 (equivalent to one current step every 0,5ms) it is not critical to reduce the data. At our desirable factor of R=25,6 the match ratio is still at around 90%, which is fairly good, having in mind that we reduced data volume by 96%!

This insight is rather promising, because we can see now that the model is relatively robust to data reduction and, if the buffer writing rate is increased in the future, the emulation precision will increase substantially. In Figure 5 one can also see the possible memory clock frequencies for the hardware and the corresponding scaling factors (vertical lines). At 62,5 MHz the scaling factor will only be R=1,6 so the reduction of data will have virtually no effect on the quality of emulations.

[3]The plots for time boundaries of 1 ms and 3 ms are included in the appendix for better understanding of tendencies.

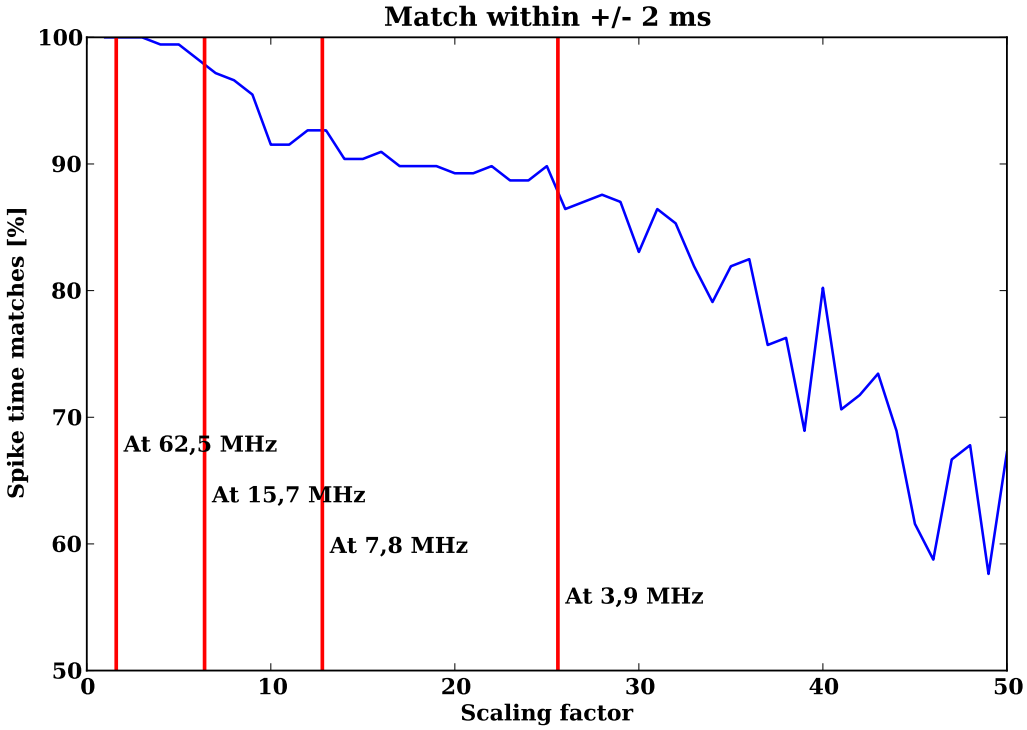


Figure 5: Spike time match dependence on scaling factor

4 Conclusion

To summarize the results of this project lab:

A software model, matching a model implemented in the neuromorphic hardware has been created. The form of synaptic conductances has been changed, a second adaptation term has been added, the voltage offset in hardware adaptation circuits and the limitations of hardware parameters have been taken into account. This model will help us at predicting the behaviour of the hardware, that is still being tested and is not yet fully functional. For example one could use a proper algorithm to fit biological data with our model and extract the fitting parameters. The performance of the model could serve as an estimation of the hardware performance in future neuron modeling competitions, and the resulting parameters could be used later to estimate the corresponding hardware parameters.

To prevent possible problems in future emulations, a test has been performed, analyzing the robustness of the implemented model regarding data reduction. The outcome of this test is overall promising, however the actual buffer update frequency (which is still to be determined) has to be at least 3,9 MHz. At this frequency the current input is distorted significantly, but only $\approx 10\%$ of the spike times do not match. This fact encourages us to participate with our hardware at future neuron modeling competitions using external current input.

References

[1] **J. Schemmel, J. Fieres, and K. Meier:** "Wafer-scale integration of analog neural networks", *Proceedings of the 2008 International Joint Conference on Neural Networks (IJCNN)*, 2008

[2] **Electronic Vision(s) Group:**
Web: `http://www.kip.uni-heidelberg.de/cms/groups/vision/`, 2010

[3] **FACETS:** Fast Analog Computing with Emergent Transient States.
Web: `http://www.facets-project.org/`, 2009

[4] **Romain Brette, Wulfram Gerstner:** "Adaptive Exponential Integrate-and-Fire Model as an Effective Description of Neuronal Activity", *J Neurophysiol*, 2005

[5] **Marc-Oliver Gewaltig, Markus Diesmann:** "NEST (NEural Simulation Tool)", *Scholarpedia*, 2007
Web: `http://www.nest-initiative.uni-freiburg.de/index.php/`

[6] **PyNEST:** NEST Extension module for using with Python.
Web: `http://www.nest-initiative.uni-freiburg.de/index.php/PyNEST`, 2010

[7] **Python:** The Python Programming Language.
Web: `http://www.python.org/`, 2009

[8] **matplotlib:** 2D Python plotting library.
Web: `http://matplotlib.sourceforge.net/`, 2010

[9] **NumPy:** Scientific Computing Tool for Python.
Web: `http://numpy.scipy.org/`, 2010

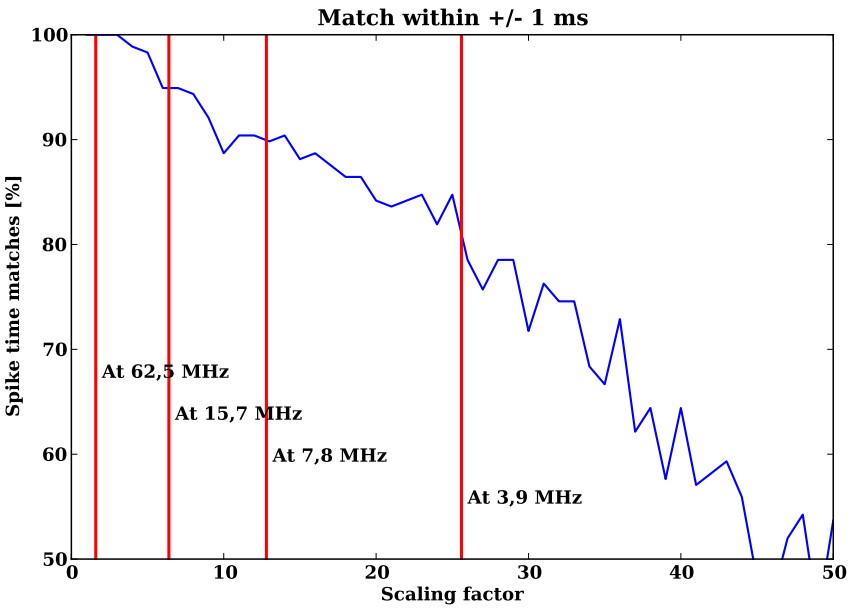
[10] **Neuron Modeling Competition:** Quantitative Single-Neuron Modeling 2009.
Web: `http://www.incf.org/community/competitions/spike-time-prediction/2009`, 2009

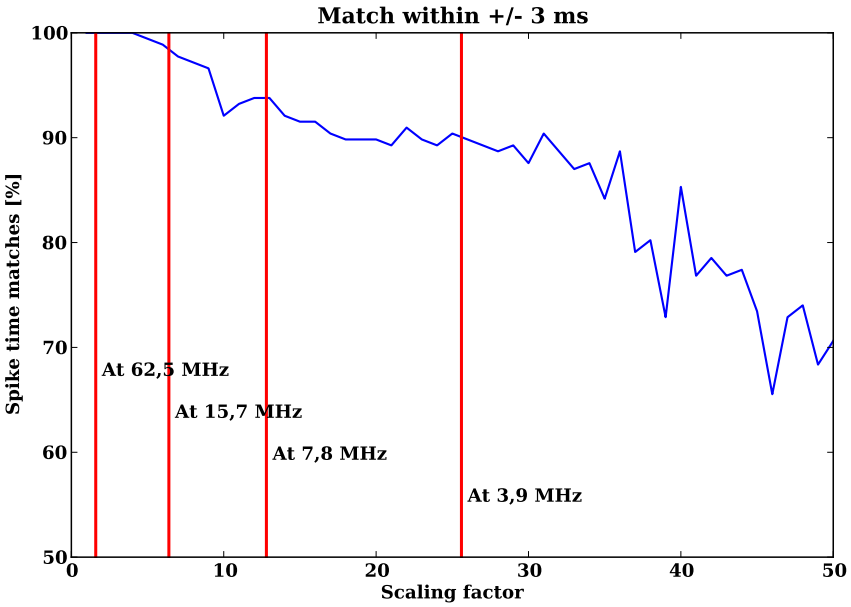
[11] **M. Galassi et. al.:** GNU Scientific Library Reference Manual (3rd Ed.)
Web: `http://www.gnu.org/software/gsl/`, 2009

[12] **Skander Mensi:** Talks at FACETS plenary meeting 2010 in Dresden
Web: `http://facets.kip.uni-heidelberg.de/internal/jss/AttendMeeting?m=displayPresentation&mI=54&mEID=257`, 2010

[13] **Sebastian Millner:** FACETS Hardware Workshop Dresden 2010
Web: `https://facets.kip.uni-heidelberg.de/internal/jss/AttendMeeting?m=displayPresentation&mI=63&mEID=193`, 2010

Appendix


Match within +/- 1 ms
Spike time matches [%]
Scaling factor
At 62,5 MHz
At 15,7 MHz
At 7,8 MHz
At 3,9 MHz
100
90
80
70
60
50
0
10
20
30
40
50


Match within +/- 3 ms
Spike time matches [%]
Scaling factor
At 62,5 MHz
At 15,7 MHz
At 7,8 MHz
At 3,9 MHz
100
90
80
70
60
50
0
10
20
30
40
50