[Re] Connectivity reflects coding: a model of voltage-based STDP with homeostasis - Supplementary Material

René Larisch

October 24, 2019

Table 1: Original parameter values out of the adaptive exponential integrate and fire model by Clopath et al. (2010)

Parameter	Values
C, membrane capacitance	281 pF
g_L, leak conductance	30 nS
E_l, resting potential	−70.6 mV
Δ_T, slope factor	2 mV
$V_{T_{rest}}$, threshold potential at rest	−50.4 mV
$\tau_{w_{ad}}$, adaption time constant	144 ms
a, subthreshold adaption	4.0 nS
b, spike triggered adaption	0.805 pA
Isp, spike current after spike	400 pA
τ_z, spike current time constant	40 ms
τ_{V_T}, threshold potential time constant	50 ms
VT_{max}, threshold potential after spike	30.4 mV

Table 2: Original parameter values for the STDP learning rule and for the different experimental setups Clopath et al. (2010)

Experiments	θ_- (mV)	θ_+ (mV)	A_{LTP} (mV^{-1})	A_{LTD} (mV^{-2})	τ_x (ms)	τ_- (ms)	τ_+ (ms)
Visual Cortex	−70.6	−45.3	14×10^{-5}	8×10^{-5}	15	10	7
Somatosensory Cortex	−70.6	−45.3	21×10^{-5}	30×10^{-5}	30	6	5
Hippocampal	−41	−38	38×10^{-5}	2×10^{-5}	16		

1 Original parameter set

Both tables contain the original parameter sets as reported in Clopath et al. (2010) and in Brette and Gerstner (2005). **Tab. 1** shows the parameter set for the adaptive exponential integrate and fire model, and **Tab. 2** shows the parameter for the voltage-based triplet spike timing-dependent plasticity rule for different experimental setups.

2 Influence of random chosen weights on the learning result

For the sake of reproducibility the provided Python scripts contain a fixed seed number. Nevertheless, the task to analyze how the connectivity structure is influenced by activity strength and the task to show the emergent of stable weights on Gaussian curve input use randomly initialized synaptic weights. **Fig. 1** and **Fig. 2** show the results for five simulations with different randomly initialized weights for both tasks.

3 Comparison with the Matlab code

To test the reimplementation in Python with ANNarchy we used the task to learn stable weights on Gaussian-Input. There is an implementation for this task on modelDB. For both, the Matlab code and the Python code, we used the same initial weight matrix and the same input protocol. **Fig. 3** shows the membrane potential for the first 400 ms simulation time for the Matlab code (red line in the top frame), for the Python reimplementation (green line in the middle frame), and the difference between both (blue line in the frame below). We see, that for the first 25 ms the error is zero and at time point 25 ms, the error increases. This shows, that at the beginning, both implementations work equally. To analyze this further, we plotted the average weights to the one post-synaptic neuron in this task (see **Fig. 4**). Please note, that the figure shows the weight values of 400 input stimuli (every stimulus was presented for 100 ms). Again, in both models the curves look very similar, but a small difference exists. Due to the nature of the learning rule (the development of the weight depends on the post-synaptic membrane potential), we assume that a small difference in the membrane potential leads to differences in the weights and as a consequence to a change in the membrane potential and so on. However, we recognize small differences in

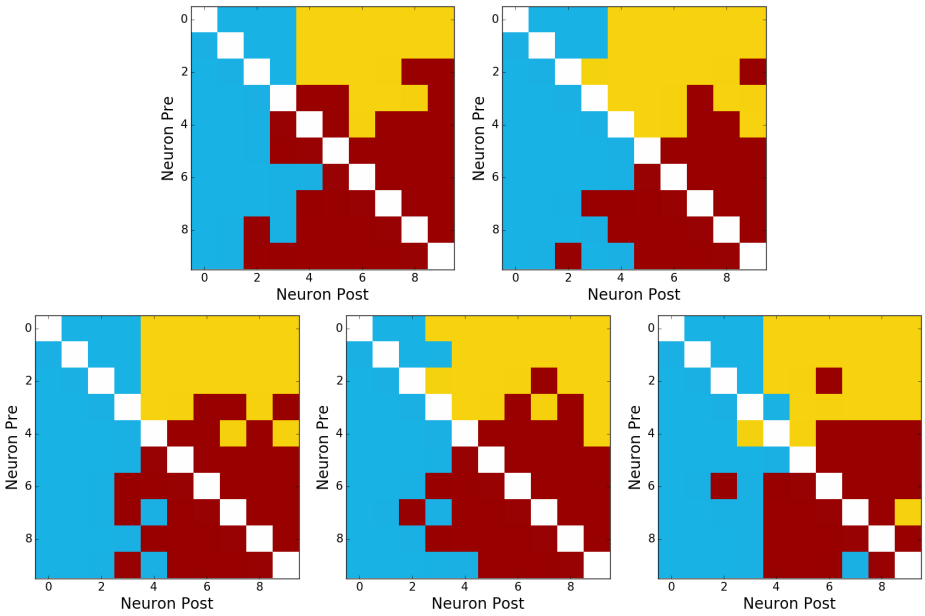

Figure 1: Five examples of how the spiking activity influences the connectivity structure. Initialized with different synaptic weights.

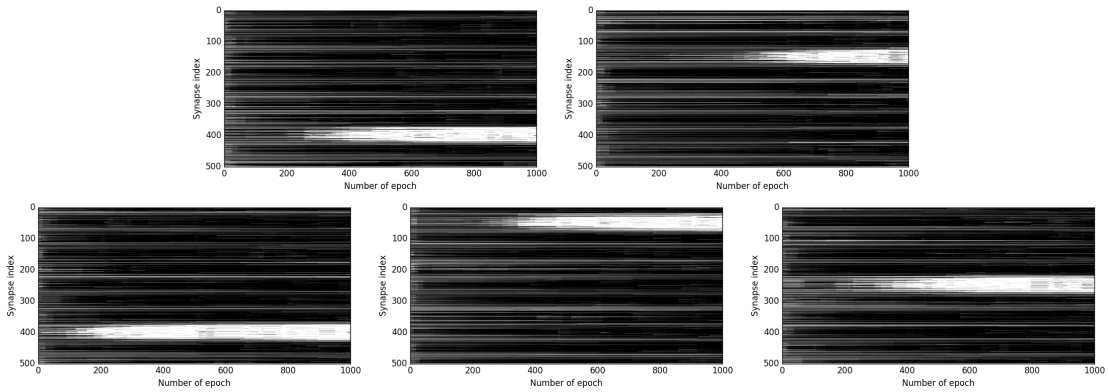

Figure 2: Five examples for the emergence of stable weights by presenting a Gaussian curve input.

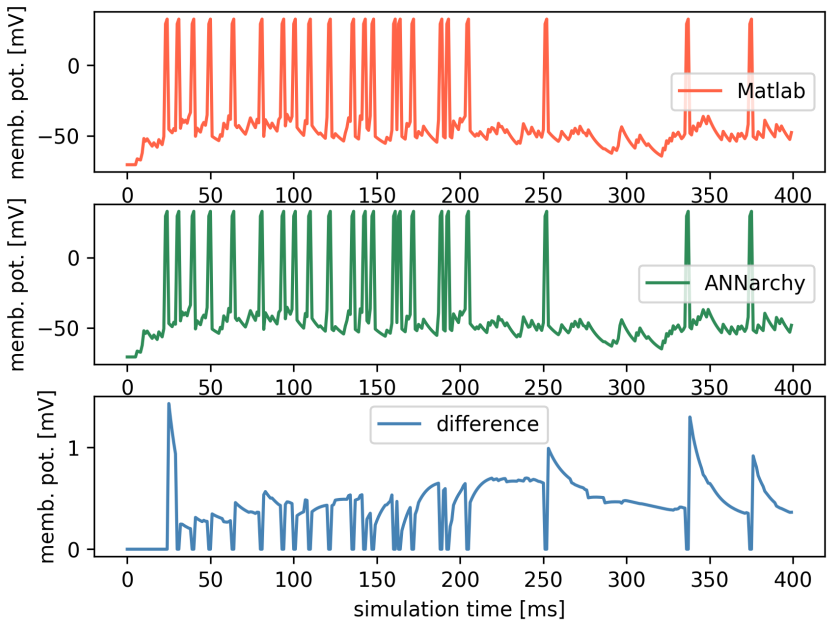

Figure 3: Membrane potential for the Matlab source code from modelDB (top) and the reimplementation in Python with ANNarchy (middle). Although in both implementations the curves look equal, we see a little difference between them (down).

the membrane potential between the two implementations if the synaptic learning was deactivated. We worked out the following main reason: The execution order of the different variables. In the Matlab code, the membrane potential is updated two times in the same calculation step (two calculation steps after a spike). Due to the usage of ANNarchy, we can not change the execution order during the simulation and ANNarchy allows only one update of a variable per calculation step. Additionally, other variables, which are necessary for the membrane potential, have the same issue. We assume, that this leads to small differences in the membrane potential and in the weight change. This difference is accumulating for long simulations (as for the receptive field learning) and make it necessary to change the parameter. Please note, this argumentation only relates to the differences between the Matlab code on modelDB and the Python reimplementation with ANNarchy and not to the correctness of the Clopath et al. (2010) model.

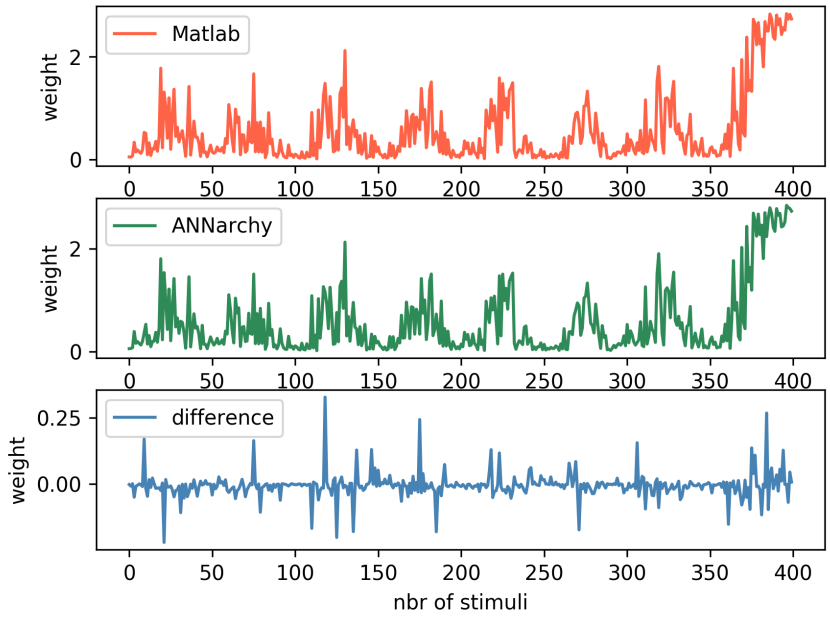

Figure 4: The average synaptic weight for the Matlab source code from modelDB (top) and the reimplementation in Python with ANNarchy (middle). Although in both implementations the curves look equal, we see a little difference between them (down).

References

Brette, R. and Gerstner, W. (2005). Adaptive exponential integrate-and-fire model as an effective description of neuronal activity. *Journal of Neurophysiology*, 94(5):3637–3642. PMID: 16014787.

Clopath, C., Büsing, L., Vasilaki, E., and Gerstner, W. (2010). Connectivity reflects coding: a model of voltage-based STDP with homeostasis. *Nature Neuroscience*, 13(3):344–352.