# Necessity of STDP and Heterosynaptic Competition in Sequential Firings

Ansel Blumers and Ankan Ganguly

December 15, 2015

**Abstract**

Sequential cognitive processing are driven by sequences of neural activity in various parts of the brain. Hahnloser *et al.* claim that the combination of heterosynaptic competition within single neurons and spike-time-dependent plasticity (STDP) lead the network to sequential activity [3]. In this paper, we will re-examine their claim by investigating the effect of heterosynaptic competition and the necessity of STDP. We build a model which partially replicates the results of [3], and demonstrate that these partial properties remain when we replace STDP by Hebbian learning.

# Contents

## 1 Introduction

Vertebrates often need to process sequential events. Sequential cognitive processing is driven by sequences of neural activity in various parts of the brain. The ubiquity of repeating sequential neural patterns prompts the speculation that the mechanisms driving these patterns may be general across species. One test subject under frequent investigations is the Zebra finch. Zebra finches are songbirds that produce sounds in a recurring fashion. They can produce motifs that last up to 1 second, while single neuron bursts only sustain about 6 ms. This coding of sequential activity has been observed to be sparse [4].

The first model seeking to represent neural sequence generation is categorically referred as synaptic chain networks [5]. The connectivity matrix in such networks is directional  neuronal groups are connected in a single file fashion. However, the network connectivity must be pre-defined by hand, severely slashing its usability.

One attempt to circumvent the drawback was made by Fiete and Hahnloser as described in the article *Spike-Time-Dependent Plasticity and Heterosynaptic Competition Organize Networks to Produce Long Scale-Free Sequences of Neural Activity.* The authors claim that the combination of heterosynaptic competition within single neurons and spike-time-dependent plasticity (STDP) lead the network to sequential activity.

STDP rules modify the synaptic connection so that the repeated timely causal activation of pair of neurons strengthens the connection, while the reverse is weakened [1]. Synchrony among firings arises naturally as a result. Because of this, a neuron that fires frequently will become a hub that modulates the firings in a network level. To form a long sequence, STDP alone is insufficient as suggested by the authors. Heterosynaptic competition was then introduced to supplement STDP. The underlying idea is based on cellular biology and well-documented post-synaptic neurons conserve the total synaptic weight by depressing some input synapses when others undergo potentiation [6]. The authors imposed a slightly different implementation onto their model  heterosynaptic long-term depression (hLTD) .

In this paper, we will re-examine Hahnloser *et al.*'s claim that the combination of STDP and hLTD give rise to the generation of neural sequential firings. We will first investigate how strong of an effect hLTD imposes on the model by tuning relevant parameter. Then will we inspect the necessity of STDP.

## 2 Methods

In this paper, we used the simulated Integrate and Burst (IB) model introduced in [3]. We then implemented STDP learning and Hebbian learning.

### 2.1 Integrate and Burst Model

Our IB model is an implementation of equations described in [3]. We provide a detailed description of those equations here, with more attention paid to the intuition behind these equations. There are $N$ neurons connected in an all-to-all environment. Each neuron $i$ bursts when its membrane potential, $V_i$, hits the threshold $V_\theta$. We assume that every neuron bursts for $T_{burst}$ time. While bursting, each neuron fires four times uniformly over the burst interval before its potential is reset to $V_{reset}$.

When a neuron is not bursting, its potential is governed by a typical conductance based leaky integrate and fire model with built-in inhibition:

$$\tau_V \frac{\mathrm{d}V_i}{\mathrm{d}t} = -g^L(V - V^L) - g_i^E(V - V^E) - g_i^I(V - V^I) \tag{1}$$

The leak conductance, $g^L$, is assumed to be a homogeneous constant input, as is the leak potential $V^L$. Notice that in the absence of excitatory or inhibitory conductance, neuron $i$ will tend to $V^L$. Therefore we can say that the leak potential is also equal to the rest potential.

The excitation potential, $V^E$, and the inhibition potential, $V^I$, act as upper and lower bounds on untethered potential respectively.

The excitation conductance, $g^E$, is defined by the activity in neighboring neurons as well by external random stimulation:

$$g^E = Ws + W_0 b$$

Where $W_{ij}$ is the strength of the synapse from neuron $j$ to neuron $i$, $W_0$ is the conductance strength of external synapses and $b$ is a Poisson random variable with frequency $r_{in}$. $r_{in}$ was an input parameter. In [3] $r_{in}$ was constant, but we chose to assign $r_{in}$ a high value and anneal it to jump-start neural activity. $s_i$ is the activation of neuron $i$. It is incremented each time neuron $i$ fires, and it decays by

$$\frac{\mathrm{d}s}{\mathrm{d}t} = -\tau s$$

The inhibitory conductance, $g^I$, is defined as the sum of the adaptation inhibition, $g^I_{ada}$, and the global inhibition, $g^I_{glob}$. The adaptation inhibition is the internal inhibition generated by the adaptation activation of a neuron $s_i^{ada}$. It is defined by the same equations as $s_i$, but with a slower time-constant $\tau_{ada}$. $g^I_{ada} = A_a s^{ada}$ is proportional to $s^{ada}$, and $g^I_{glob} = \frac{A_g}{N}\sum_{i=1}^N s_i$ is proportional to the average activation of a neuron in the system.

## 2.2 Learning

The purpose of this paper is to replicate and test some of the claims made in [3]. One of the claims made is that the model used by [3] demonstrates how STDP can contribute to the synchronous regular synfiring chains exhibited in Zebra Finch [4]. We implemented STDP, but we also implemented a pure Hebbian learning rule to demonstrate that STDP learning is not necessary for the model to exhibit synchronous regular firing chains.

### 2.2.1 STDP Learning

We followed [3] precisely in our implementation of STDP learning. Define

$$K(t) = \begin{cases} e^{-t/\tau_{STDP}} & \text{if } t > 0 \\ -e^{-t/\tau_{STDP}} & \text{if } t < 0 \\ 0 & \text{otherwise} \end{cases}$$

as the STDP kernel. For every pair of neurons $i, j$, let $t_i < t_j$ be two times when $i$ and $j$ fired respectively. Then According to STDP, the weight matrix element $W_{ij}$ should increase proportional to $K(t_j - t_i)$. Notice that the change in $W$ according to STDP is approximately anti-symmetric. [3] strengthened the nonlinearity of STDP by making STDP growth of a synapse proportional to the strength of that synapse (with an added factor so 0 weight synapses could grow). In particular, the paper defined:

$$\Delta_{ij}^{STDP}(t) = \left(\frac{W_{ij}(t-1)}{w_{max}} + 0.001\right) * \left(x_i(t)K(0)x_j(t) + \sum_{\tau=0}^{t}[x_i(t)K(\tau)x_j(t-\tau) - x_i(t-\tau)K(\tau)x_j(t)]\right) \tag{2}$$

Where $x_i(t)$ is a binary variable taking the value 1 if neuron $i$ fired at time $t$. So, at each time $t$, $\Delta^{STDP}(t)$ is proportional to the change in $W$ with respect to STDP. Our implementation assumed $K(t) \approx 0$ for $t > 4$ms.

However, the key innovation of this paper was to introduce a secondary source of competition. If the total strength of all synapses into or out of a given neuron exceed a soft limit $W_{max}$ after STDP learning, then all such synapses experience heterosynaptic long-term depression (hLTD) proportional to the amount by which the soft limit is exceeded:

$$\theta_i^{col} = \left[\sum_{j=1}^{N}(W_{ij} + \eta\Delta_{ij}^{STDP}) - W_{max}\right]^+ \tag{3}$$

where $\eta$ is the STDP learning step size, and $\theta^{col}$ denotes the amount by which the soft limit has been exceeded. Define $\theta^{row}$ similarly. Then,

$$\frac{\mathrm{d}W_{ij}}{\mathrm{d}t} = \eta\Delta_{ij}^{STDP} - \epsilon(\theta_i^{col} + \theta_j^{row}) \tag{4}$$

where $\epsilon$ is the strength of the soft constraint. This updates weights according to STDP along with a penalty if the soft limit is exceeded. Finally, to ensure that STDP never diverges, we implement a hard limit. If any element $W_{ij}(t) > w_{max}$, where $w_{max}$ is a constant parameter, we manually set $W_{ij}(t)$ to $w_{max}$. Throughout the paper we assume that $w_{max} = W_{max}$.

We actually implemented the results incorrectly based on a typo in [3]. Our implementation used the following equations:

$$\theta_i^{col} = \left[\sum_{j=1}^{N}(W_{ij} + \Delta_{ij}^{STDP}) - W_{max}\right]^+ \tag{5}$$

$$\frac{\mathrm{d}W_{ij}}{\mathrm{d}t} = \eta\Delta_{ij}^{STDP} - \eta\epsilon(\theta_i^{col} + \theta_j^{row}) \tag{6}$$

As demonstrated in sections 3 and 4, this causes some significant issues with convergence. Unfortunately, we noticed the problem too late into the project to redo the simulations.

### 2.2.2 Hebbian Learning

Hebbian plasticity is a general family of learning rules which strengthen synapses between neurons which fire with high correlation. Hebbian learning rules are generally considered incomplete because while they explain how synapses are strengthened, they do not explain how synapses might be weakened.

One way to complete the Hebbian learning rule is with STDP. In STDP learning, highly correlated neurons experience a large change in synaptic strength, but the direction of the change (increase or decrease) depends on the relative timing of firing between the synapses.

However, the soft and hard limits imposed by the model also provide a way for synapse strength to decrease, so it should be possible to implement this model using Hebbian plasticity. To do this, we ran the exact same learning algorithm as implemented for STDP, but we modified the kernel to be

$$K(t) = \begin{cases} e^{-t/\tau_{Heb}} & \text{if } t > 0 \\ 0 & \text{otherwise} \end{cases}$$

With this new kernel,

$$\Delta_{ij}^{Heb}(t) = \left(\frac{W_{ij}(t-1)}{w_{max}} + 0.001\right) * \left(x_i(t)K(0)x_j(t) + \sum_{\tau=0}^{t} x_i(t)K(\tau)x_j(t-\tau)\right) \tag{7}$$

gives us a Hebbian learning rule. We implemented the soft and hard synaptic weight limits as in STDP learning.

## 2.3 Parameter Choices

Our implementation of the IB neuron network with STDP learning matched that used in [3] in many cases. However a few parameters were chosen differently. The table below provides a list of parameters we used:

| Parameters | Values |
|---|---|
| $dt$ | $2 \times 10^{-5}$s |
| $V^E$ | 0V |
| $V_\theta$ | -0.05V |
| $\tau$ | 0.004s |
| $N$ | 50 |
| $\eta$ | varies |
| $A_a$ | 9 |

| Parameters | Values |
|---|---|
| $\tau_V$ | 0.01 F/m$^2$ |
| $V^I$ | -0.07V |
| $V_{reset}$ | -0.055V |
| $r_{in}^{start}$ | 10000Hz |
| $w_{max}$ | 0.14 |
| $\epsilon$ | varies |
| $\tau_{STDP} = \tau_{Heb}$ | 0.02s |

| Parameters | Values |
|---|---|
| $V^L$ | -0.06V |
| $W_0$ | 5S/m$^2$ |
| $T_{burst}$ | 0.006s |
| $r_{in}^{min}$ | 4000Hz, 6000Hz |
| $W_{max}$ | 0.14 |
| $A_g$ | 4 |
| $\tau_{ada}$ | 0.015s |

# 3 Results

We were primarily interested in two major results produced in [3]. First, we were interested the occurrence of long synfire chains with synchronous regular bursting. We also investigated the paper's claim that STDP learning with $w_{max} = W_{max}$ produced scaled permutation matrices.

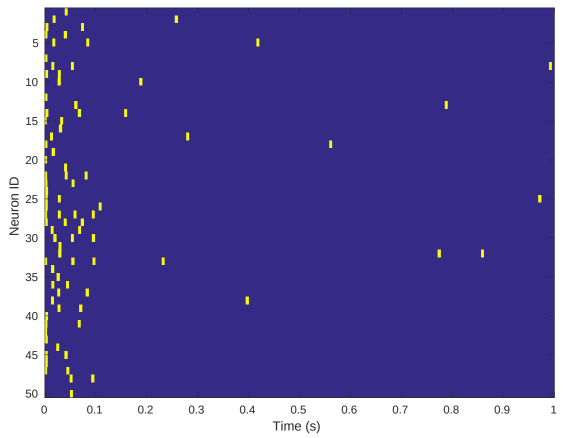
Though we tried to replicate the algorithm described in [3], we were unable to produce a stable system with the provided inputs, so we had to tune the parameters a bit. We eventually found parameters at which burst rates were steady and believable.

With the parameters we found, we were able to observe weaker versions of most of the observations in the paper. We found that weight matrices converged near scaled permutation matrices and we were able to observe some short sequences of firing chains, but we did not see the synchronous regular behavior exhibited in the paper.

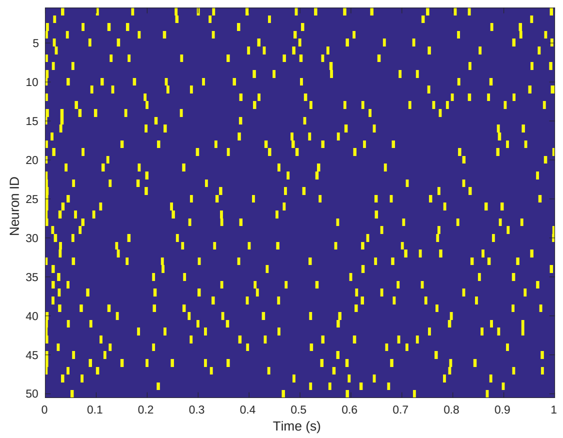
Finally, we replaced STDP learning with Hebbian learning and demonstrated that the weights of a system undergoing Hebbian learning with soft and hard limits converged as closely or better to scaled permutation matrices than they did under STDP learning.

## 3.1 Parameter Tuning

Although our model was constructed to be identical to the model used in [3], we were unable to use the same parameters as the paper. As an example, the paper claimed to set $r_{in}$, external input, at 4 Hz. At 4 Hz, we were unable to get our system to burst at all! We assumed it was a typo (the rest of the parameters were given in terms of ms), and attempted running our simulation at 4000Hz. Even 4000Hz was not enough to get the system started, so we tried simulated annealing. The system began with 10000Hz of stimulation, which decreased steadily to 4000Hz. We did this again, but this time stopped at 6000Hz. The difference was striking (see figure 1).



(a) 4000 Hz



(b) 6000 Hz

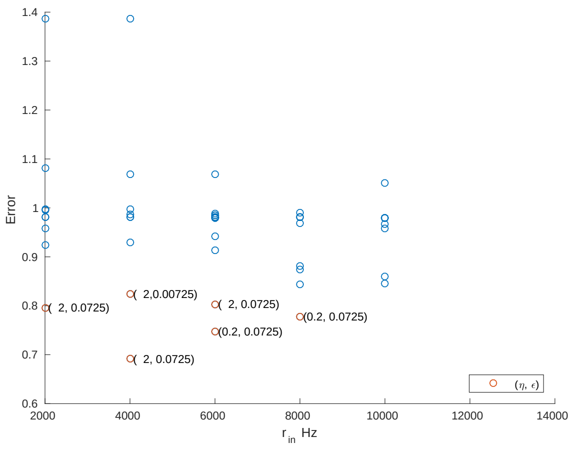
Figure 1: In figure 4 of [3], the authors displayed a burst plot in which each neuron burst every 75 ms with high synchrony and regularity. We did not observe such synchrony in either plot, but the average frequency of firing in (b) was approximately 75 ms. Notice both systems exhibited exaggerated firing in the beginning. This was a consequence of simulated annealing.

Beside $r_{in}$, we spent a lot of time finding appropriate values for $\eta$ and $\epsilon$.

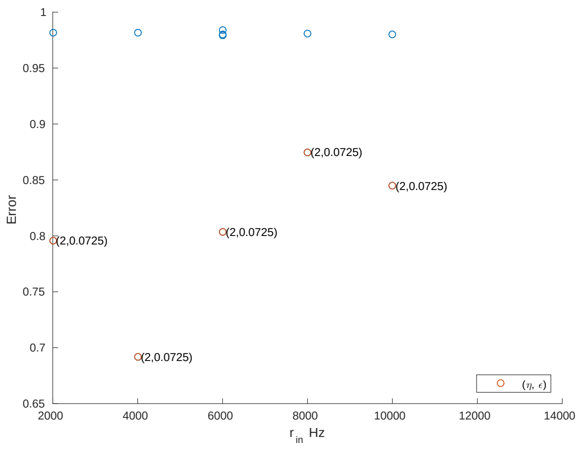
$\eta$ is the learning step-size. If $\eta$ is too small, the weight matrix will not converge in a reasonable amount of time. If it is too large, whenever the weight matrix overshoots the soft limit, it will do so by a large margin, which will prompt large corrective measures, so convergence to a scaled permutation matrix (which is barely under the soft limit), will not be plausible.

$\epsilon$ is the strength of the soft limit. If $\epsilon$ is too small, then when the weight matrix passes the soft limit it will not experience very strong hLTD. This will have the effect of weakening heterosynaptic competition. If $\epsilon$ is too large, then each time the weight matrix passes the soft limit, it will experience very strong hLTD. Because the scaled permutation matrix is on the border of the soft limit, having a large $\epsilon$ will make any scaled permutation matrix an unstable critical point in our simulation.

To evaluate these, we ran the simulation on a large combination of parameter values for $r_{in}, \eta$ and $\epsilon$ (see figure 2a). For each of these, we tested convergence using an error function (see section 3.2). Then we isolated the product $(\eta\epsilon)$ which seemed to yield the best results (see figure 2b).



(a) Errors of all simulations run

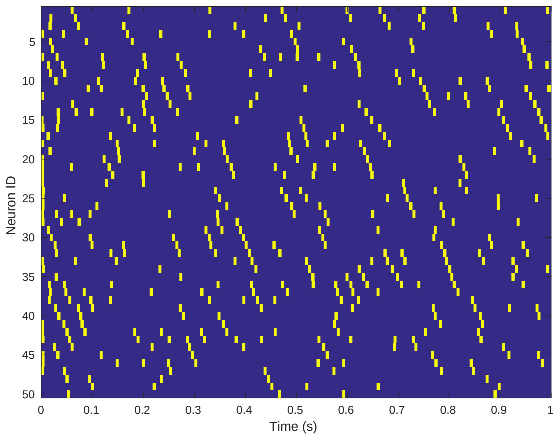


(b) Errors of all simulations with $\eta\epsilon = 0.145$

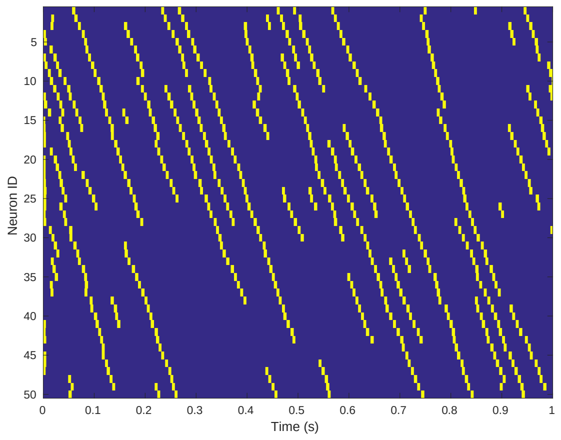
Figure 2: Recall from (6) that in our implementation, $\eta$ and $\eta\epsilon$ are important tuning parameters for the rate of convergence. In all cases, we ran the algorithm for 50000 time steps with $dt = 2 \times 10^{-5}$s. Notice that all error values are very high. (a) This is a plot of every simulation we ran. The best results are labeled. (b) This is a plot of the subset of points from (a) where $\eta\epsilon = 0.145$.

The best two combinations of parameters seemed to be $(r_{in}, \eta, \epsilon)$ =(4000Hz, 2, 0.0725) and (6000Hz, 0.2, 0.0725). For the rest of the paper, we used these parameters unless otherwise noted.

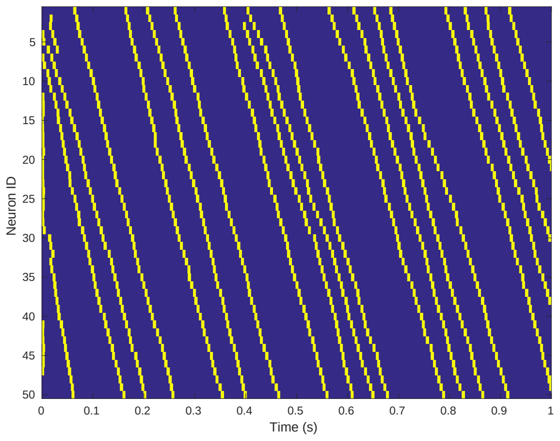
Finally, we tried running a playback sequence with a fixed scaled permutation weight matrix. By symmetry, according to the model if one scaled permutation matrix is a stable point, then all of them must be. Therefore we fixed $W$ to be a scaled permutation matrix. For convenience, we assumed this scaled permutation matrix was a large cycle. We ran it without modification to observe burst patterns for differing values of $w_{max}$ (see figure 3).



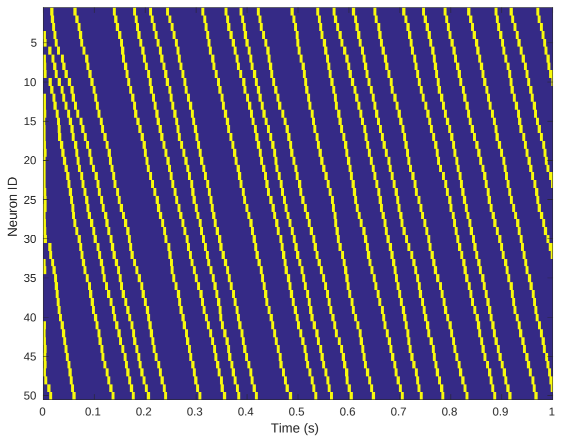
(a) Bursts without learning, $w_{max} = 0.14$.



(b) Bursts without learning, $w_{max} = 0.3$.



(c) Bursts without learning, $w_{max} = 0.5$.



(d) Bursts without learning, $w_{max} = 0.7$.

Figure 3: We ran the simulation for 50000 time steps at $dt = 2 \times 10^{-5}$s with $r_{in} = 6000$Hz. As expected, for high values of $w_{max}$, we witnessed high regularity, synchrony and continuous firing chains. At our default value of $w_{max} = 0.14$ we witnessed a few small firing chains, but much more irregular firing.

At our default $w_{max}$ value, the IB model was unable to maintain the patterns of long sequential firing observed in [3], however larger $w_{max}$ values did display these patterns.

## 3.2 Convergence and Stability

Our IB model with STDP learning very quickly developed a stable bursting rate. Given up to about 4000Hz of external stimulation, this bursting rate was essentially 0, but for more stimulation the bursting rate was consistent. The neurons in the network constructed in figure 4 of [3], on which our paper was based, bursted every 75 ms or so. From figure 4, we can see that this corresponds to an external stimulation of slightly less than 8000Hz in our model.

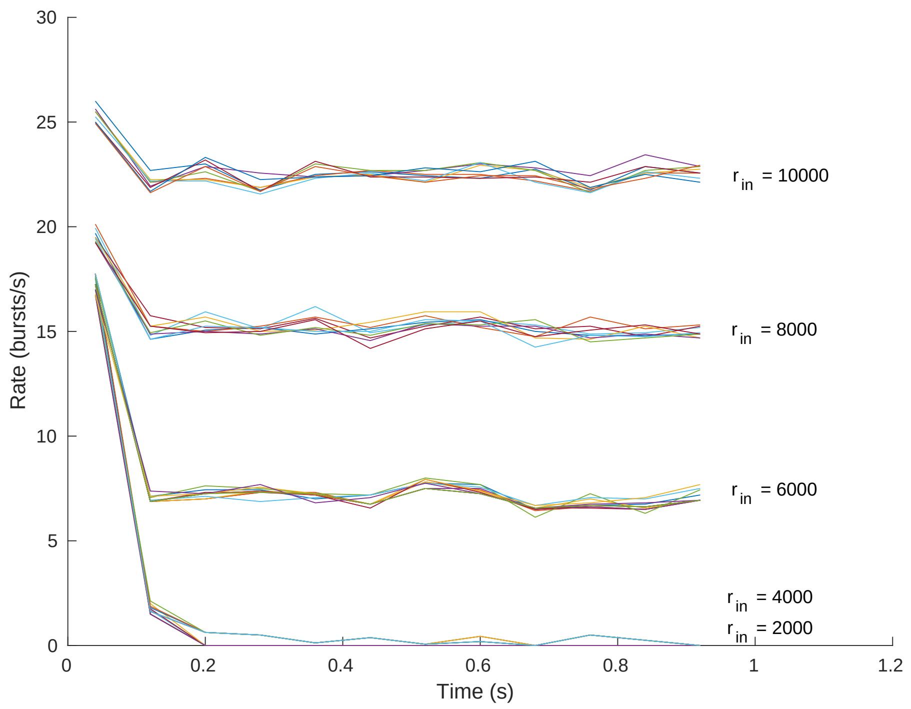
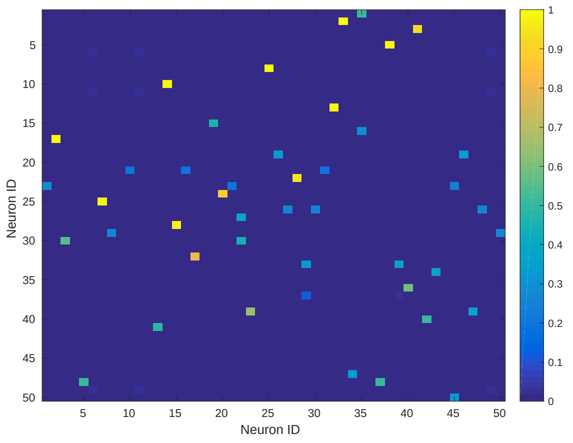
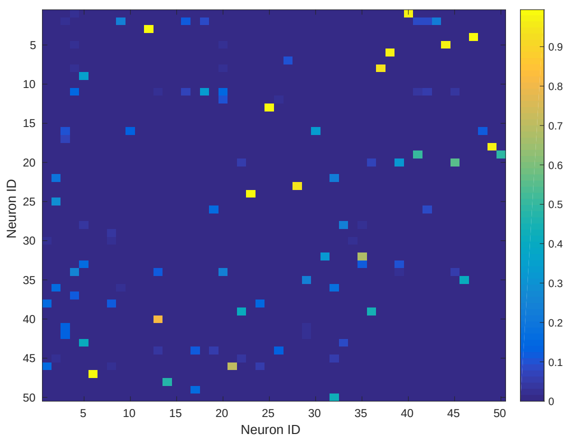


Figure 4: The plot above measures the average firing rate of a neuron in the recurrent network over time over approximately 50 simulations. Notice that the average firing rate converges cleanly to different values dependent only on $r_{in}$.

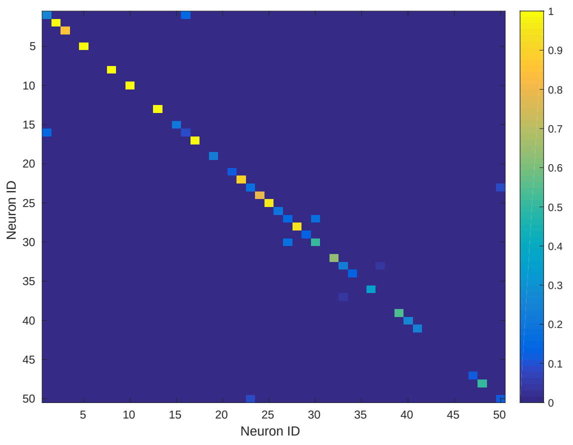
From this we concluded that our base IB model was stable. The next thing we needed to test for was convergence of our weight matrix under STDP with soft and hard limits to a scaled permutation matrix. Recall that all permutation matrices $P$ satisfy $PP^T = I$. So, while we can visually inspect heat maps of $W$ to see if it's close to a scaled permutation matrix, it is easier to look at how $WW^T$ compares to $w_{max}^2 I$ (see figure 5).



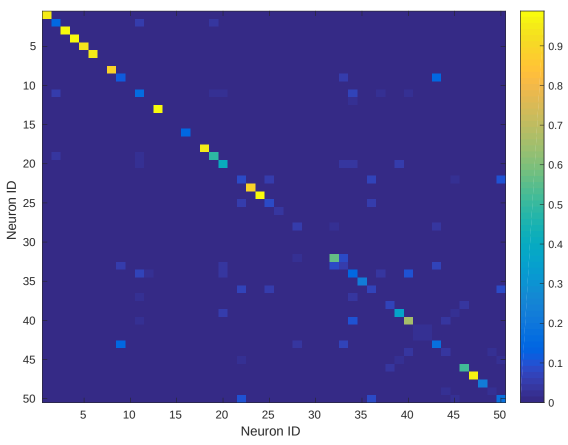
(a) Weight matrix of 4000Hz external input with annealing



(b) Weight matrix of 6000Hz external input with annealing



(c) $WW^T$ of 4000Hz with annealing



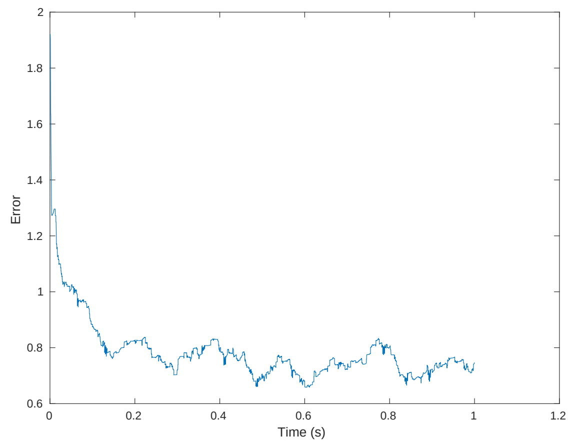
(d) $WW^T$ of 6000Hz with annealing

Figure 5: After running each simulation for 50000 time steps with $dt = 2 \times 10^{-5}$s at 4000Hz and 6000Hz, we plotted the weight matrices, (a) and (b), and for each weight matrix $W$ we plotted $WW^T$, (c) and (d). While neither weight matrix converged perfectly to a scaled permutation matrix, each one looks like it is almost converged to a scaled permutation matrix.

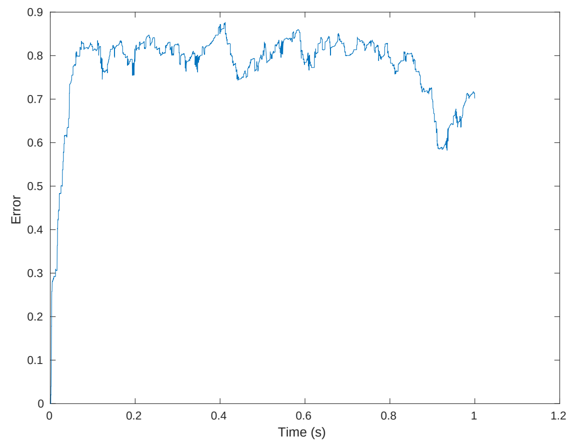
While these figures seemed to confirm that the weight matrices were converging to scaled permutation matrices under STDP, it would be better to get some sense of convergence over time. To do this we introduce the following error function:

$$e(W) = \sum_{i,j=1}^{N} \left| WW^T - w_{max}^2 I \right|_{ij}$$

This is the componentwise absolute difference between $WW^T$ and $PP^T$ for any permutation matrix scaled by $w_{max}$. We can consider $e$ the square distance of $W$ from the set of square roots of $w_{max}^2 I$. Notice that this measurement is actually a little generous, since permutation matrices are not the only square roots of the identity. However, it is simple to compute and if it is nonzero, then we are guaranteed that the weight matrix is not a scaled permutation matrix. See figure 6 for a plot of error over time. Notice that this error is unnormalized by $w_{max}$, but that was not important here because we only used the error function with $w_{max} = 0.14$.



(a) Plot of error in weights over time starting from a full connection weight matrix.



(b) Plot of error in weights over time starting from a scaled permutation weight matrix.

Figure 6: (a) We ran a simulation where initial weights were all-to-all (minus diagonals) and uniform. (b) We ran a simulation where initial weights were already set to a scaled permutation matrix. In both cases the error roughly converged to about 0.7 with a fairly large variance.

What's interesting is that after multiple runs with the same parameters we always converged to the same distance plus or minus some variance. There are a few explanations for this. As will be demonstrated in the next section, our implementation was flawed, and this caused most of the error. However, even with the proper implementation we observed a certain stable amount of error. See section 4 for more details.

## 3.3 Hebbian Learning versus STDP

Consider the following learning rule:

$$\frac{\mathrm{d}W_{ij}}{\mathrm{d}t} = f(W_{ij}, X) \text{ if } \sum_k W_{ik}, \sum_\ell W_{\ell j} \leqslant W_{max}, \text{ otherwise}$$

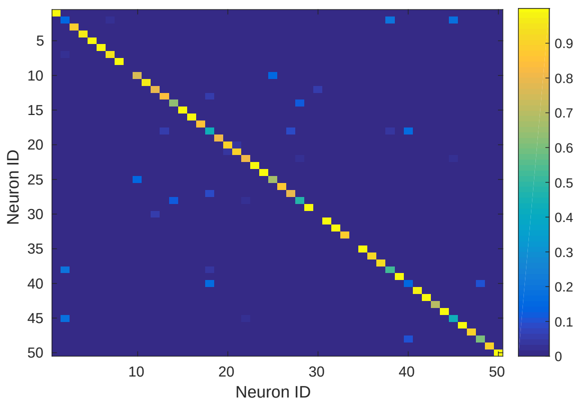
$$\frac{\mathrm{d}W_{ij}}{\mathrm{d}t} = f(W_{ij}, X) - \delta\left(\left(\sum_k W_{ik} - W_{max}\right)^+ + \left(\sum_\ell W_{\ell j} - W_{max}\right)^+\right)$$

$$0 \leqslant W_{ij} \leqslant w_{max} \quad W_{max} = w_{max}$$

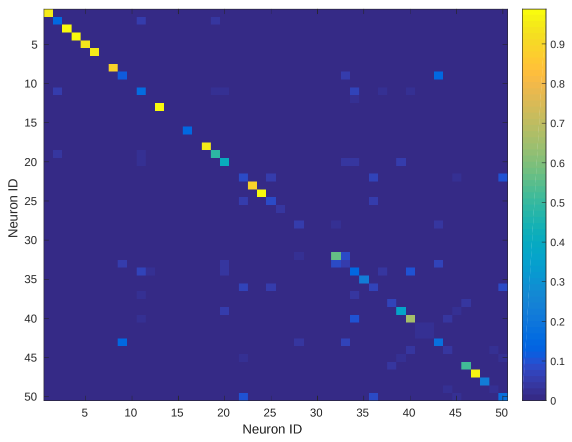
Where $X$ is a random variable and $f(W, X) \geqslant 0$ and it is strictly increasing with respect to $W$ with high probability.

Suppose $W = w_{max}P$ where $P$ is a permutation matrix. Consider $W_Y = W + Y$, where $Y$ is a random matrix such that for all $i$ and $j$, $|Y_{ij}| < \epsilon$ for some small $\epsilon \ll w_{max}$. Fix $i, j$ so that $W_{ij} = w_{max}$. Then there are three cases. First, both the rows and columns of $(W_Y)_{ij}$ satisfy the soft constraint. Then $(W_Y)_{ij}$ will increase until the soft limit is hit. If the soft limit is hit, assuming $f$ increases strictly enough, $(W_Y)_{ij}$ should continue increasing until it hits the hard limit. At that point, all other elements in its rows and columns should tend to 0, or for sufficiently small $\delta$, they may remain slightly positive.

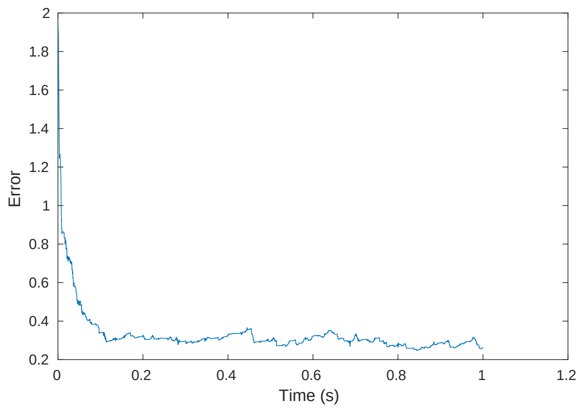
While this treatment was far from rigorous, it does demonstrate that learning rules other than STDP may be combined with hLTD and hard limits to produce scaled permutation weight matrices. In particular, since higher synapse strength between neurons increases synchrony and firing rates, Hebbian plasticity may satisfy the conditions on $f$ to produce scaled permutation weight matrices. To test this hypothesis, we ran a few simulations (see figure 7).



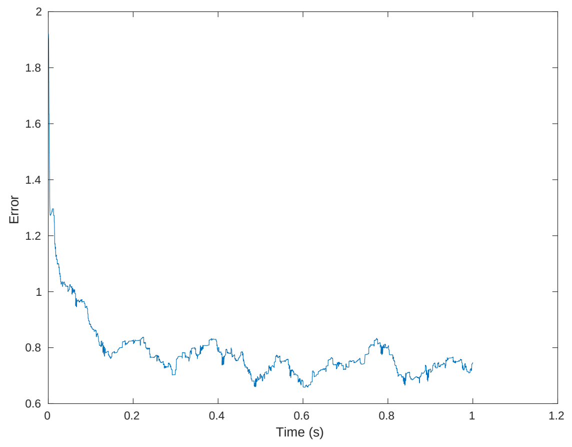
(a) $WW^T$ matrix after Hebbian learning



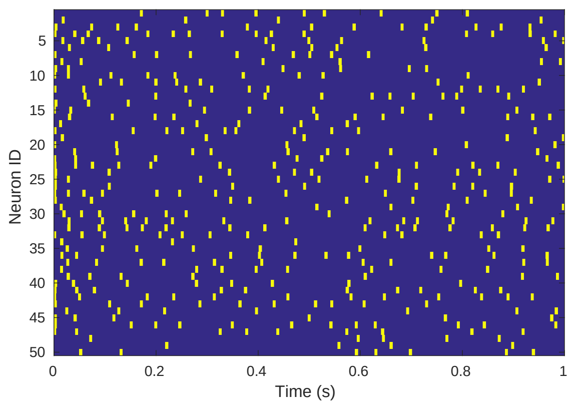
(b) $WW^T$ matrix after STDP learning



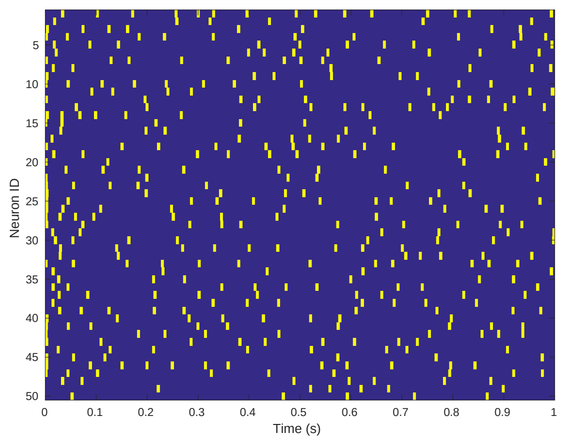
(c) Error over time of Hebbian Learning



(d) Error over time of STDP Learning



(e) Burst history of Hebbian plot



(f) Burst history of STDP plot

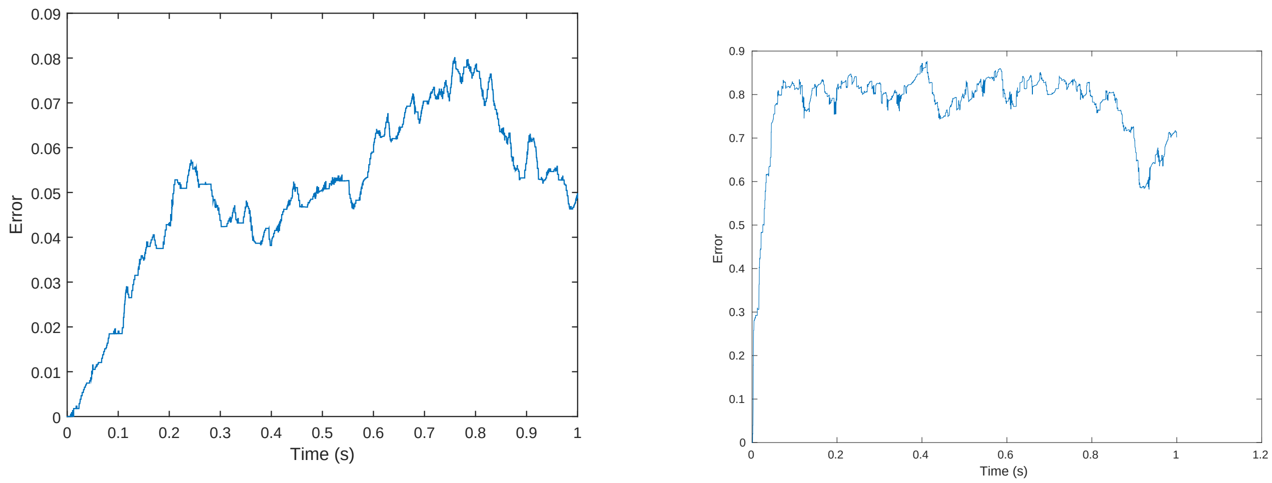
Figure 7: Comparison of STDP and Hebbian learning simulations at 6000Hz of input

From figure 7, it's clear that the weight matrix of the network with Hebbian learning converged more closely to a scaled permutation matrix than the STDP equivalent and with less variability. However, both simulations demonstrated similar levels of synchronized regular burst output and similar occurrence of synfire chains. This provides strong evidence that Hebbian learning with hLTD is sufficient to reproduce the results of [3].

## 4 Discussion

Our model was plagued by a number of small problems which we did not have the time to address. The most severe of these problems was our use of the incorrect equations (5) and (6) presented in [3] instead of equations (3) and (4). Why are these equations wrong? In this model, hLTD depresses all synapses to or from a particular neuron if STDP alone would have pushed in synapses over the soft limit. However, the equations for $\theta^{col}, \theta^{row}$ check to see if $W + \Delta^{STDP}$ increase beyond the soft limit. $W + \Delta^{STDP}$ is a meaningless quantity. Instead, the equations should check to see if $W + \eta\Delta^{STDP}$, which is equal to $W$ updated by STDP alone, is greater than the soft limit. Equation (6) is not technically incorrect, since for different choices of $(\eta, \epsilon)$ it is equivalent to equation (4). However, it introduces unnecessary complications, whereas equation (4) has two independent learning coefficients.

To see the effect of this, observe a graph of the error function starting from a permutation matrix with the correct method vs our implementation (figure 8):



(a) Error over Time, Corrected Method with $r_{in} = 6000Hz$, $\eta = 0.002$, $\epsilon = 0.145$

(b) Error over Time, Incorrect Method with $r_{in} = 6000Hz$.

Figure 8: It is clear that the error from the correct method is less than the error from the incorrect method by a factor of 10! This suggests most of the error seen earlier in the paper was due to a bad implementation, however even the corrected version has a certain constant error.

This implementation problem severely weakens our argument regarding the convergence of weight matrices under Hebbian plasticity. We were unable to amend this because the problem was caught too late for us to redo the simulations.

However, even with the correct implementation, there seems to be a certain amount of structural error.

We also could have spent more time with parameter tuning. Most of our work on this project was spent trying to replicate the results of [3] using their model and parameters, but when the parameters didn't work as expected, we should have constructed our own. Ultimately, this did not prove a huge problem, because we were able to develop a working, stable IB neural network simulation.

The most interesting result of this paper is the hypothesis that learning methods other than STDP may be used to replicate the results of [3]. At the end of the results section, we provided a rough sketch of an argument that such learning rules combined with hLTD may cause the emergence of scaled permutation weight matrices. This argument could be formalized to categorize precisely which learning rules this would apply to.

Another interesting extension of this paper would be to generate permutation matrix behavior in a completely different network. For example, BCM's use of a moving threshold automatically triggers a heterosynaptic long-term depression event similar to that used in this model whenever the weighted total input to a neuron exceeds the threshold [2]. BCM models are best applied to simple rate networks with output rates being linear or sigmoidal functions of the inputs. If such a network could be extended to a recurrent network and used to generate synfiring chains with weights converging to scaled permutation matrices, then this would provide further insight into how competition can be applied to learning to produce such firing patterns.

## 5 Summary

We generated a model similar to that used in [3] in an attempt to replicate the results from the paper. Our model did not work with the parameters provided by the paper, but we were able to generate a working IB model. In addition, while we did not see the synchronous regular firing observed in [3], we were able to show partial convergence of weight matrices to scaled permutation matrices and limited synfiring chain behavior. We were also able to replicate these partial results with Hebbian learning instead of STDP learning. While our model had a number of minor flaws, it provided a nice start to verifying and testing the assertions of [3].

## References

[1] L. F. Abbott and Sacha B. Nelson. Synaptic plasticity: taming the beast. Nat Neurosci.

[2] Leon N. Cooper Elie L. Bienenstock and Paul W. Munro. Theory for the development of neuron selectivity: Orientation specificity and binocular interaction in visual cortex. The Journal of Neuroscience, 2(1):32–48, January 1982.

[3] Ila R. Fiete, Walter Senn, Claude Z.H. Wang, and Richard H.R. Hahnloser. Spike-time-dependent plasticity and heterosynaptic competition organize networks to produce long scale-free sequences of neural activity. Neuron, 65(4):563 – 576, 2010.

[4] Richard H. R. Hahnloser, Alexay A. Kozhevnikov, and Michale S. Fee. An ultra-sparse code underliesthe generation of neural sequences in a songbird. Nature, 419(6902):65–70, Sep 2002.

[5] D. Kleinfeld and H. Sompolinsky. Associative neural network model for the generation of temporal patterns. theory and application to central pattern generators. Biophys J, 54(6):1039–1051, Dec 1988. 3233265[pmid].

[6] Sebastien Royer and Denis Pare. Conservation of total synaptic weight through balanced synaptic depression and potentiation. Nature, 422(6931):518–522, Apr 2003.