STDP, Oscillations, and Network Architectures

E.V.Lubenov

April 17, 2003

1 Introduction

Information processing in biological networks occurs through interactions between discrete spatially distributed processing elements—neurons. The biophysical study of neurons suggests that the basic computation neurons implement consists of *integrate* and *fire* steps. In the integration step the neuron weighs and sums its inputs and if the result exceeds a fixed threshold generates, i.e. fires, a discrete unitary output event called an *action potential* or *spike*. In this simplified framework the nature of the computation is determined by the structure of the weights through which individual elements communicate. The modification of the connection weight matrix as a function of experience is believed to be the physical basis for learning and memory. Recent evidence suggests that the rules for weight modification are determined by the timing of the input and output spikes, a discovery that launched the field of spike timing dependent plasticity (STDP). Another line of research has shown that the excitability of individual elements is modulated by various oscillatory modes of different spatio-temporal structure.

2 Project Description

2.1 Motivating Idea

Over small timescales (ms) oscillatory modes might act to impose structure in the polling of individual elements, thus directing information flow in the network and determining computation. Over longer timescales (hrs) the interaction between oscillatory modes and STDP learning rules might impose structure in the connection weight matrix, thus establishing specific computing architectures co-embedded in the recurrent network.

2.2 Project Objectives

1. Implementation of a parallel distributed *integrate and fire* neural network simulator supporting oscillatory modes and connection weight modification through STDP rules.

   The network dynamics can be expressed as a set of ordinary differential equations (ODE) and therefore the core ingredient of the network simulator is an ODE solver. Essentially all methods for numerically solving initial value problems rely on successive evaluations of the derivative $\mathbf{y}' = F(t, \mathbf{y})$. The function $F(\cdot)$ is the obvious target for exploiting parallelism. How well $F(\cdot)$ lends itself to parallel evaluation will depend on the extent to which some form of domain decomposition can be applied to the problem at hand. Therefore the first step in meeting the project objectives involves formulating the ODEs governing the evolution of the network and exploring their suitable decomposition.

   The next step involves the choice of implementation platform. The possibilities include using Matlab*P with `mm` mode and compiling C/MPI code taking advantage of available parallel libraries.

2. Investigation of the network architectures emerging in the interactions between oscillatory modes and STDP rules.

   The first order effect to be characterized is the structure in the connection weight matrix that emerges under oscillatory modes of different frequencies and intensities. Higher order effects include sensitivity analysis of the emergent structure to model parameter variations and overall model complexity.

3 Methods

3.1 Pyramidal Cell Models

Neural cells posses both passive and active properties that affect the dynamics of the membrane potantial. The morphology of the neuron and the magnitude of the membrane capacitance and leak conductance determine the passive properties, while the distribution and characteristics of the different voltage-gated channels account for the active properties. For the purposes of the present study we explore two classes of models: a passive single compartment *integrate-and-fire* model and an active two compartmental *Pinsky-Rinzel* model. The latter represents a minimal biophysical model capable of displaying the phenomenon of complex bursting, a regime

of action potential generation thought to play a key role in synaptic weight modification.

3.1.1 Passive Integrate-and-Fire Model

Each neuron j is modeled as a single compartment with membrane potential V. Each compartment has an excitatory g_{ex} and an inhibitory g_{in} conductance expressed relative to the the leak conductance. The evolution of the membrane potential is given by

$$\tau_m \frac{dV}{dt} = V_{rest} - V + g_{ex}(t)(E_{ex} - V) + g_{in}(t)(E_{in} - V) \quad (1)$$

$$\tau_{ex} \frac{dg_{ex}}{dt} = -g_{ex} \quad (2)$$

$$\tau_{in} \frac{dg_{in}}{dt} = -g_{in} \quad (3)$$

Three special cases arise in the evolution of the equations above. When the membrane voltage reaches threshold V_{th} at times τ_{ji} an action potential is generated and V is reset to V_{reset}. When the l^{th} action potential arrives at the k^{th} synapse at time t_{kl} the corresponding excitatory or inhibitory conductance is incremented by the peak synaptic conductance $\overline{g}_j k$.

$V(t)$	$\rightarrow$	V_{reset}	when $V(t) = V_{th}$ at $t = \tau_{ji}$
$g_{ex}(t)$	$\rightarrow$	$g_{ex}(t) + \overline{g}_{jk}(t)$	upon input at exc synapse k at t_{kl}
$g_{in}(t)$	$\rightarrow$	$g_{in}(t) + \overline{g}_{jm}(t)$	upon input at inh synapse m at t_{mn}

Synaptic plasticity is introduced in the model through rules that modify $\overline{g}_{jk}$. For a model neuron that has N_j^e plastic synapses there is a corresponding number of functions $P_{jk}(t)$ that control the increments of $\overline{g}_{jk}$ as well as one function $M_j(t)$ that controls the decrements according to

$$\tau_- \frac{dM_j}{dt} = -M_j \quad (4)$$

$$\tau_+ \frac{dP_{j1}}{dt} = -P_{j1} \quad (5)$$

$$\vdots \quad (6)$$

$$\tau_+ \frac{dP_{jN_j^e}}{dt} = -P_{jN_j^e} \quad (7)$$

together with the rules

$$\begin{array}{lcll} M_j(t) & \rightarrow & M_j(t) - A_- & \text{when neuron } j \text{ fires a spike} \\ P_{jk}(t) & \rightarrow & P_{jk}(t) + A_+ & \text{when neuron } j \text{ gets input at synapse } k. \end{array}$$

Finally the peak synaptic conductances themselves are modified according to the following rules. Notice that synapses are somewhat counterintuitively *strengthened* upon output of the postsynaptic neuron and *weakened* upon presynaptic input.

$$\begin{array}{lcll} \overline{g}_{jk} & \rightarrow & \max(0, \quad \overline{g}_{jk} + \overline{g}_{max} M_j(t)) & \text{upon input at synapse } k \\ \overline{g}_{jk} & \rightarrow & \min(\overline{g}_{max}, \quad \overline{g}_{jk} + \overline{g}_{max} P_j k(t)) & \text{upon output of neuron } j. \end{array}$$

The model parameters are as in Song, Miller, and Abbott (2000).

$$\begin{array}{llllll} V_{rest} & = & -70\ (mV) & \tau_m & = & 20 \\ V_{thresh} & = & -54\ (mV) & V_{peak} & = & 0\ (mV) \\ E_{ex} & = & 0\ (mV) & \tau_{ex} & = & 5 \\ E_{in} & = & -70\ (mV) & \tau_{in} & = & 5 \\ A_+ & = & 0.005 & \tau_+ & = & 20 \\ A_- & = & 1.05 * A_+ & \tau_- & = & 20. \end{array} \qquad \begin{array}{lll} V_{reset} & = & -60\ (mV) \\ g_{max} & = & 0.015 \\ g_{in} & = & 0.05 \end{array}$$

3.1.2 Active Pinsky-Rinzel Model

The Pinsky-Rinzel model incorporates a dendritic and a somatic compartment. Each compartment is endowed with the same passive conductance I_{leak}, but with different active conductances. In particular the somatic compartment is equiped with the standard Hodgkin-Huxley voltage-gated Na^+ (I_{Na}) and K^+ (I_K) currents responsible for spike-generation, while the dendritic compartment has a persistent Na^+ (I_{NaP}) ans a slow K^+ (I_{KS}) currents that extend the single spike into a burst and terminate the burst, respectively. The following equations control the evolution of the somatic (V_s) and dendritic (V_d) membrane potentials.

- Compartment equivalent circuit equations

$$C_m \frac{dV_s}{dt} = -I_{Na} - I_K - I_{leak} - \frac{g_c}{p}(V_s - V_d) + I_{soma} \tag{8}$$

$$C_m \frac{dV_d}{dt} = -I_{NaP} - I_{KS} - I_{leak} - \frac{g_c}{(1-p)}(V_d - V_s) + I_{dendrite} \tag{9}$$

  With the exception of I_{leak} all currents in the above equations are voltage-gated and their evolution is thus controled by the evolution of different gate variables, as shown below.

- Current equations

$$
\begin{aligned}
I_{Na} &= \overline{g}_{Na} m_{\infty}^3 h (V_s - E_{Na}) && (10)\\
I_K &= \overline{g}_K n^4 (V_s - E_K) && (11)\\
I_{NaP} &= \overline{g}_{NaP} l_{\infty}^3 (V_d - E_{Na}) && (12)\\
I_{KS} &= \overline{g}_{KS} q (V_d - E_K) && (13)\\
I_{leak} &= g_{leak} (V - E_{leak}) && (14)
\end{aligned}
$$

- Dynamic gate equations

$$
\begin{aligned}
\frac{dh}{dt} &= \phi_h [\alpha_h(V_s)(1-h) - \beta_h(V_s) h] && (15)\\
\frac{dn}{dt} &= \phi_n [\alpha_n(V_s)(1-n) - \beta_n(V_s) n] && (16)\\
\frac{dq}{dt} &= \phi_q \Big[\frac{q_{\infty}(V_d) - q}{\tau_q(V_d)} \Big] && (17)
\end{aligned}
$$

- Gate steady state and time constant equations

$$
\begin{aligned}
m_{\infty}(V_s) &= \frac{V_s + 31}{V_s + 31 + 40(e^{-\frac{V_s+56}{18}} - e^{-\frac{14V_s+559}{90}})} && (18)\\
l_{\infty}(V_d) &= \frac{1}{1 + e^{-\frac{V_d+57.7}{7.7}}} && (19)\\
q_{\infty}(V_d) &= \frac{1}{1 + e^{-\frac{V_d+35}{6.5}}} && (20)\\
\tau_q(V_d) &= \frac{200}{e^{-\frac{V_d+55}{30}} + e^{\frac{V_d+55}{30}}} && (21)
\end{aligned}
$$

- Rate constant equations

$$
\begin{aligned}
\alpha_h(V_s) &= 0.07 e^{-\frac{V_s+47}{20}} && (22)\\
\beta_h(V_s) &= \frac{1}{e^{-\frac{V_s+17}{10}} + 1} && (23)\\
\alpha_n(V_s) &= -0.01 \frac{V_s + 34}{e^{-\frac{V_s+34}{10}} - 1} && (24)\\
\beta_n(V_s) &= 0.125 e^{-\frac{V_s+44}{80}} && (25)
\end{aligned}
$$

The parameters in the model are as in Kepecs and Wang (2000).

- Membrane capacitance and compartment coupling parameter

$$C_m = 1 \ (\mu F/cm^2) \quad p = 0.15$$

- Equilibrium potentials and conductances

$$\begin{array}{lllllllll} E_{leak} & = & -65 \ (mV) & g_{leak} & = & 0.18 \ (mS/cm^2) & g_c & = & 1 \ (mS/cm^2) \\ E_{Na} & = & +55 \ (mV) & \overline{g}_{Na} & = & 55 \ (mS/cm^2) & \overline{g}_{NaP} & = & 0.12 \ (mS/cm^2) \\ E_K & = & -90 \ (mV) & \overline{g}_K & = & 20 \ (mS/cm^2) & \overline{g}_{KS} & = & 0.7 \ (mS/cm^2) \end{array}$$

- Dynamic gate temperature scaling factors

$$\phi_h = 3.33 \quad \phi_n = 3.33 \quad \phi_q = 1$$

3.2 Model Implementation Prototypes

So far we have implemented and explored several model prototypes. They have all been coded in serial Matlab. The goal was to identify and contrast the strengths and weaknesses of several possible approaches before committing to a full blown implementation. All but one prototypes use the matlab ode suite to solve an intial value problem (IVP). The algorithmic variations we have explored so far concern the type of ode solver used and absolute error tolerances.

3.2.1 Piecewise Integration

As described in the previous section at times of input and output the derivatives of the integrate-and-fire variables transiently become unbounded (as the variables get instantaneously reset). This means that the system cannot be continuously integrated over the entire period of interest. Instead the system is integrated up until a given singular point, then the final state of the system is updated to reflect the instantantaneous variable change, and the system is integrated again with initial conditions equal to the last updated state. In this piecewise fashion the system can be integrated over the entire time span of interest.

3.2.2 System Order

If we are interested in the behavior of N neurons each receiving N_m inputs via plastic synapses what is the order of the system we need to integrate? A first look at the integrate-and-fire model equations would suggest that for

each neuron j we will need V, g_{ex}, g_{in}, M_j, P_{ji}, $\ldots$, P_{jN_m} for a total of $4+N_m$ variables, yielding a grand total of $N(N_m+4)$. Of course except for V all other equations are trivial to solve between singular points, since the variables simply decay exponentially from the initial condition with their corresponding time constant. Thus if u represents any such variable, then for a period of time starting at a singular point t_0 up until the next singular point we can write

$$u(t) = u(t_0)e^{-\frac{t-t_0}{\tau_u}}.$$

This together with the addative nature of the update rules yields closed form solutions. These solutions can also be derived by rewriting the original unforced differential equations with a forcing term consisting of a sum of weighted impulses at the appropriate singular points. In both cases

$$g_{ex}(t) = \sum_{k=1}^{N_m} \sum_{t_{kl}<t} \overline{g}_{jk}(t_{kl}) e^{-\frac{t-t_{kl}}{\tau_{ex}}} \tag{26}$$

$$M_j(t) = -A_- \sum_{\tau_{ji}<t} e^{-\frac{t-\tau_{ji}}{\tau_-}} \tag{27}$$

$$P_{jk}(t) = A_+ \sum_{t_{kl}<t} e^{-\frac{t-t_{kl}}{\tau_+}}. \tag{28}$$

Using the above relations we can also write an approximate closed formula for the synaptic weights. The equation below is only approximately correct because it assumes that all inputs and outputs result in actual modification, while the update rules state that no modifications are made when the synaptic weights are at g_max and are to be increased, or are at 0 and are to be decreased

$$\overline{g}_{jk}(t) \approx g_{max}\Big[A_+ \sum_{\tau_{ji}<t} \sum_{t_{kl}<\tau_{ji}} e^{-\frac{\tau_{ji}-t_{kl}}{\tau_+}} - A_- \sum_{t_{kl}<t} \sum_{\tau_{ji}<t_{kl}} e^{-\frac{t_{kl}-\tau_{ji}}{\tau_-}}\Big].$$

So for each neuron, starting at singular point t_0 up until singular point t_1, we integrate a single equation

$$\tau_m \frac{dV}{dt} = V_{rest} - V + g_{ex}(t_0)e^{-\frac{t-t_0}{\tau_{ex}}}(E_{ex}-V) + g_{in}(t_0)e^{-\frac{t-t_0}{\tau_{in}}}(E_{in}-V).$$

3.2.3 Outer Loop Iterations

As described earlier the system is intergrated in a piecewise fashion between singular points. The routines of the ode solver thus constitute an *inner*

loop that is inside an *outer* loop iterating over all singuler points, i.e. times of inputs and outputs. If there are $N_e = 1000$ external inputs coming at a mean rate of $R_e = 40$ (Hz) and $N = 100$ neurons producing outputs at a mean rate of $R = 40$ (Hz) then a period of $t = 100$ (s) will require $t(N_e R_e + NR) = 4,400,000$ outer loop iterations. This number will multiply the time taken by the inner loop and therefore it becomes crucial to optimize the performance of the ode solver.

3.2.4 Problem Stiffness

To obtain efficiency the ode solver routine must be suited to the stiffness of the problem. Stiffness occurs when different elements of a system evolve with vastly different time constants. Since all neurons in the model have the same membrane time constant τ_m that has the same order of magnitude as the excitatory τ_{ex} and inhibitory τ_{in} conductance time constants one should not expect stiffness. In practice stiffness can be detected if variable step solvers (not meant for stiff problems) make the integration step size very small compared to the rate of change (curvature) in the solution. We therefore explicitly integrated a model system under identical input and initial conditions with all the ode solvers available in Matlab under three different levels of absolute tolerance control. The results are presented in Figure 1. As you can see both in terms of time and number of steps taken the explicit Runge-Kutta (4,5) formula (Dormand-Prince pair) implemented by `ode45` performs worst, while explicit Runge-Kutta (2,3) pair of Bogacki and Shampine implemented by `ode23` performs best, indicating the presence of moderate stiffness.

3.3 Model Verification

In this section we simply show the results of integrating both the passive integrate-and-fire model and the Pinsky-Rinzel under conditions reported in the literature. The purpose of these runs was to verify the correctness of the equations and implementation and for later comparison with the final version of the program.

3.4 Remainig Tasks

- Full Scale Parallel Implementation

  Parallelizing a neural network simulation should be straightforward given that the system that is being modeled is inherently parallel and

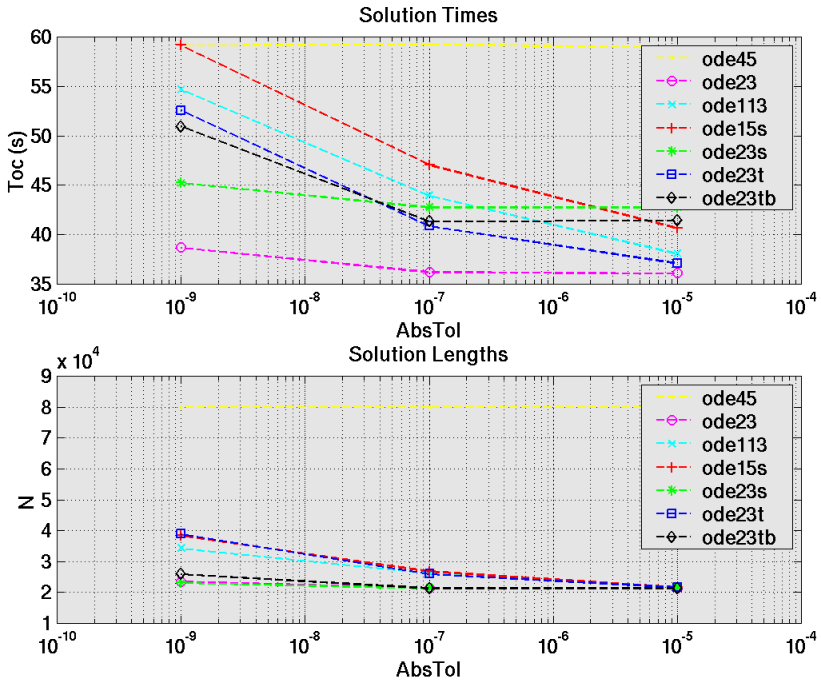

Figure 1: **Model Stiffness Indicators**

A system containing a single neuron receiving 1000 excitatory and 200 inhibitory inputs firing at 20 Hz and 10 Hz respectively was integrated for a period of 100 μs using different Matlab ode solvers. **top** solution times for integrating the system and **bottom** number of steps taken by the solver. According to both criteria ode45 performs worst and ode23 best indicating the possible presence of moderate stiffness in the system.

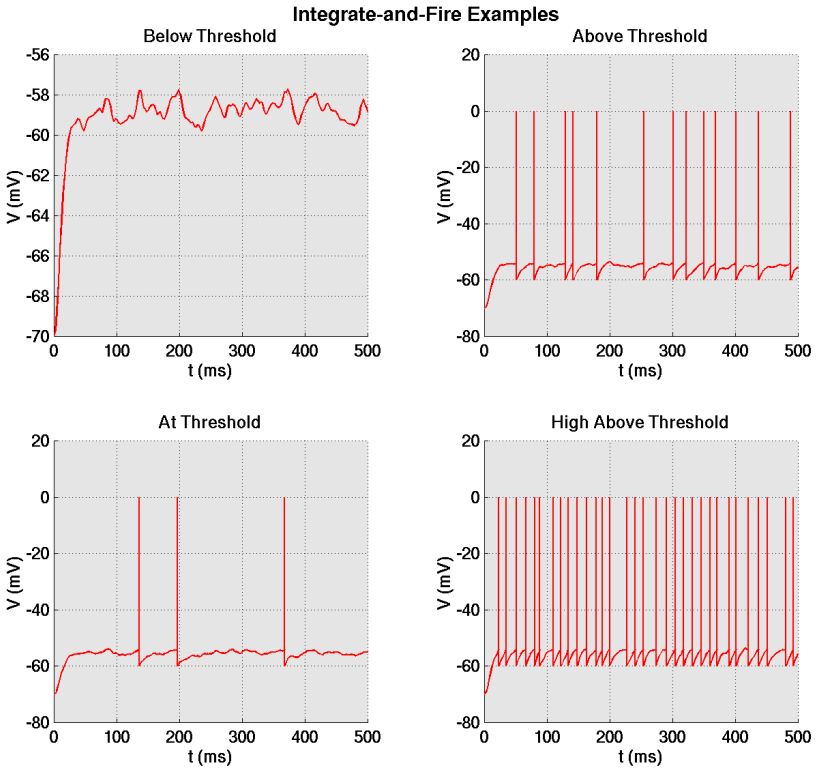

Figure 2: **Integrate-and-Fire Example Output**
A system containing a single neuron receiving 1000 excitatory and 200 inhibitory inputs firing at 20 Hz and 10 Hz respectively was integrated for a period of 500 μs. All four panels correspond to the same input pattern received through a progressively scaled synpatic weight matrix. Four patterns of activity are seen: subthreshold potential fluctutations with no spiking (top left), very sparse spiking (bottom left), denser spiking (top right), and high-rate regular spiking (bottom right).

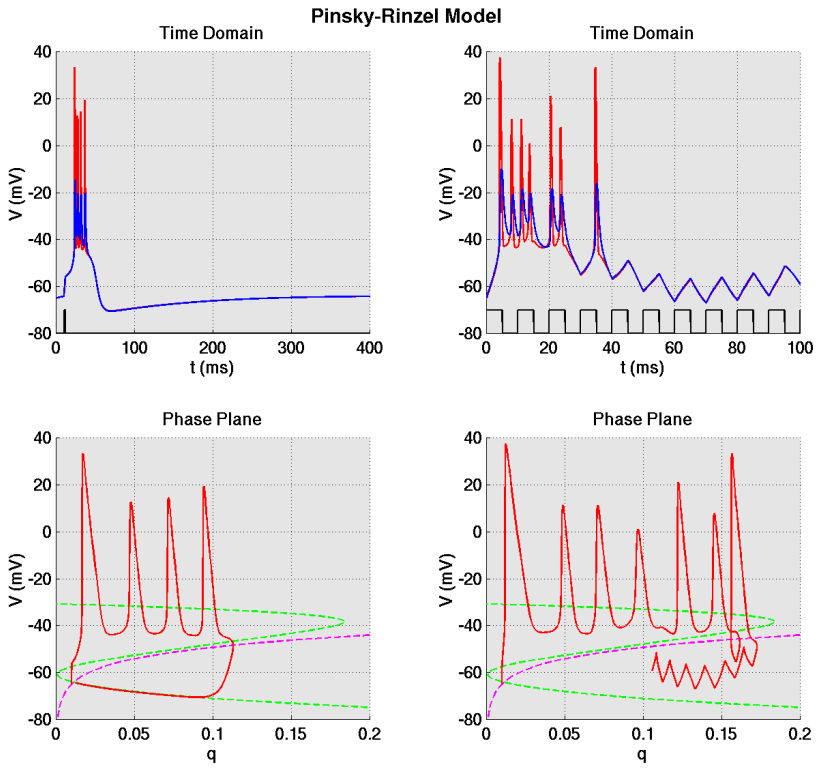

Figure 3: **Pinsky-Rinzel Model Output**

A system containing a single neuron receiving a dendritic current injection was integrated for 400 μs and 100 μs. Top two panels show the potential at the soma (red) and the dendrite (blue) under single spike input (black) to the left and periodic forcing (black) to the right. Notice the complex spiking that is not seen in the passive integrate-and-fire model. The bottom panels show the same evolution ploted in the phase plane to reveal the dynamics generating the bursts. The nullclines of q (magenta) and V_s (green) are also shown. See Kepecs and Wang (2000) for more details.

distributed. At the same time however there is a stringent synchronization requirement, since there is "real time" cross talk between individual neurons that will be integrated on different processors. We plan to explore two parallel implementation options. The first involves distributing the model neurons across the processors explicitly and integrating, while synchronizing and exchanging messages after each time step. This will involve writing some parallel ode solver routines (i.e. for agreeing on a suitable step size to be used by all processors etc.) as well as spike passing interface. The second option is close to the approach taken for the Matlab prototype routines and involves the use of a parallel ode solver library, such as the one provided with SUNDIALS (formerly known as PVODE). This might be more suitable for the Pinsky-Rinzel model, that will likely benefit from a more sophisticated ode solver.

- STDP in Networks of Neurons

  What happens to the synaptic weight matrix of a neuronal network whose elements obey STDP rules?

- STDP in Bursting Neurons

  While the effect of STDP on the synaptic weight distribution has been explored in the context of integrate-and-fire model, there is little known about the interactions between spike-timing dependent plasticity and complex spiking. Therefore it is worth characterizing a hybrid model, i.e. active Pinsky-Rinzel conductances together with modifiable synaptic weights.