JCSM
Journal of Computational Simulation and Modeling
ISSN: 2231-3494 & E-ISSN: 2231-3508, Volume 1, Issue 2, 2011, pp-17-26
Available online at http://www.bioinfo.in/contents.php?id=47

INCORPORATION OF 3D NEURONAL I-V CHARACTERISTICS IN ARTIFICIAL NEURAL NETWORKS: I. ALGORITHM

ALAN W. L. CHIU*
Department of Biomedical Engineering, Louisiana Tech University, Ruston, LA, United States
*Corresponding Author: Email- alanchiu@latech.edu

Received: May 9, 2011; Accepted: June 2, 2011

Abstract – The strategy of brain function replacement therapy is to restore the biological neuronal network by circumventing the damaged tissues with biomimetic neural models through innovative stimulation strategies. The computation capability of the artificial neural networks can be enhanced by incorporating neuronal characteristics. We propose a neural network paradigm using novel activation function that enables both sub-threshold linear dynamics and nonlinear supra-threshold spiking activities with hysteresis. Each neural unit consists of parallel linear (RC) components and nonlinear (activation) components. The overall aim of this paper entails the assembling and assessment of the aforementioned network. First, a novel 3D biologically-inspired activation function, obtained experimentally from rat hippocampal CA1 pyramidal neuron, is given. The activation function maps the current-voltage relationship between the ionic flow and electrical potential traversing through cell membrane. Second, an iterative learning rule is explained and illustrated by calculating the network's synaptic weights as well as the scaling parameters in the activation function. Finally, the signals from a coupled linear system (mimicking sub-threshold activities driven by Gaussian white noise) are used to validate the learning rules. Our preliminary result suggests that the learning algorithm is able to obtain the appropriate synaptic weights and activation scaling factors in less than 10 iterations with mean square error of less than 0.01μV.
Key words – Neural model; Activation function; Sub-threshold activity; Nonlinear; Current; Voltage; Neurons.

I. INTRODUCTION

The connectionist approach of artificial neural network (ANN) to information processing utilizing mathematical and computational representations of interconnected simple neural units was in many ways inspired by biological neural networks (BNNs). In the first generation of ANN, simple networks of binary neural units, each exhibiting either an active or inactive electrophysiological firing or non-firing states, were able to represent logic functions [1]. The state of a neural unit is determined by comparing its weighted input sum from multiple pre-synaptic units to a pre-defined firing threshold. Subsequently, these binary thresholds were replaced by monotonically increasing differentiable continuous activation functions. Using gradient descent learning algorithm such as back-propagation (BP) [2], one can obtain the coupling parameters needed to achieve the desired network output. The analogous biological interpretation of each neural unit output is the average firing rate or the probabilities of generating an action potential (AP). Some probabilistic perspectives of neuronal firing can be incorporated into the ANN framework, reflecting the stochastic aspects of BNNs. It also utilizes the assumption that neurons encode their information in the firing rate of APs [3, 4]. The networks mentioned so-far are static, with the connecting weights emulating long-term memory properties of neurons in which synaptic characteristics are at their steady states. It was also assumed that the activation function is static.

Neural network using individual spikes was proposed [5], allowing the processing of spatial-temporal information. It was a more biologically relevant approach as evidence of temporal coding was observed in the rapid cortical processing rate of visual information [6] and hippocampal pyramidal cells was found to utilize temporal as well as rate coding [7].

A major landmark in the development of neural computation is the emergence of spiking neural network [3, 8, 9]. The spiking neural networks take into account the times of the post-synaptic potentials, represented by a series of prototypical impulse-functions or pre-defined waveforms, similar to the Integrate-and-Fire (IF) model [10]. It was also suggested that spiking neural units with temporal coding in general would have more computational power than traditional sigmoidal units [11]. Other researchers also incorporated different biologically realistic properties into the design of ANNs. One such attempt is the inclusion of dynamic synaptic strength [12] and short-term plasticity [13]. The synaptic strength undergoes dynamic modulation on rapid timescales through mechanisms such as short term facilitation and depression. A new approach to compute the firing thresholds is proposed using curvature methods on the relationship between the transmembrane voltage (v) and its rate of change with time (dv/dt) [14].

We propose a neural network paradigm denoted as pNL utilizing parallel linear (RC) components and nonlinear (activation) components. Sub-threshold linear dynamics and nonlinear supra-threshold spiking activities with hysteresis can be generated using this pNL network. The design specifications for our pNL network include: (a) the artificial neural network model of the hippocampus capable of performing parallel processing on spatial-temporal information under sub-threshold and supra-threshold operating modes; (b) be able to adjust the normal static connective parameters (such as synaptic weights) and adaptively modify the dynamic properties of each neural unit to match the desired application.

In this paper, we introduce how the difference components of the network are assembled. Biologically inspired three dimensional activation function related to the current-voltage characteristics of hippocampal neurons (I-V curve). We envision that this pNL network will be capable of adjusting the excitation threshold and other scaling parameters of the I-V activation curve to match the BNN of interest. Furthermore, a novel learning rule for the synaptic weights and activation threshold is proposed using back-propagation approach similar to that of Swiercz et al. [15] where the synaptic plasticity are modified based on the postsynaptic potentials. In Section II, the design criteria and equations governing the pNL network are given. The learning rules for updating the model parameters are given in Section III. In Section IV, an illustrative example on using pNL to estimate the model parameters of a simple neuronal ensembles undergoing sub-threshold stimulation is given. The system is assumed to obey nonlinear differential equations, consistent with current techniques of using RC circuits to model neuronal membrane behavior below threshold.

II. MODEL DESCRIPTION

A comprehensive neurodynamical system should have the following properties [16]:

1) It should contain state variables (transmembrane potentials, ionic currents and/or membrane recovery variables) are continuous and are described by differential equations.
2) It should contain parallel distributed structure and high degree of freedom for enhanced computational power.
3) It must contain nonlinearity in order to create a realistic neural model.

Most of the conductance-based models stem from the Hodgkin and Huxley (HH) squid axon model [17]. The novelty of HH's work came from the proposition that ion conductance follow some nonlinear gating variables. As a preliminary study, only the transmembrane potentials y(t) measured directly from the somas of the neurons in the BNN are considered. Each neural unit in the model represents can represent a single neuron, as illustrated in Fig. **(1)**. The effect of passive propagation caused by the RC-ladder model of the dendrite [17] is ignored. Each somatic neural unit consists of a linear component and a nonlinear component organized in parallel. The nonlinear (N-) component is characterized by the supra-threshold current-voltage (I-V) activation curve in the generation of APs. It allows for a hysteresis where the repolarization path after the generation of an AP is different from the depolarizing wave front. The linear (L-) component characterized the sub-threshold membrane RC properties. The connections between units are similar to the standard recurrent neural networks with synaptic delay between the outputs.

An illustration of a simple two-cell network used in this paper is shown in Fig. **(2)**. Each neural unit is capable of receiving multiple inputs in the form of post-synaptic current inputs the APs of other neural units or in the form of injected current waveforms. At the axon hillock, the input currents are added after appropriate adjustments from the time-constant and space-constant computation based on the times and locations of synapses and current injection sites. After processing, the post-synaptic neuron generates the output signal that is transmitted to the next neural units. No other communication pathways (gap junction, extra-synaptic and electric field) are considered in this article.

From the Kirchhoff's current law (KCL), we know that the total current flowing into any node of an electric circuit must equal to the total current outflow. Applying the KCL to any output node in our model in Fig. **(2)**, assuming no self-feedback; we obtain the following coupled differential equation:

$$\begin{aligned} C_{sj}\frac{dy_j(t)}{dt} + \frac{1}{R_{sj}}y_j(t) + \gamma_j f_1\left(\alpha_j y_j(t), \frac{\beta_j dy_j(t)}{dt}\right) &= \sum_{i=1}^{I}\omega_{ji}x_i(t) + b_j(t) \\ &+ \sum_{n=1}^{N}\sum_{k\neq j}^{J} r_{jk}^{n} y_k(t-nr) \qquad (1) \end{aligned}$$

The inputs x(t) represent the measurable transmembrane potentials from the pre-synaptic neural units or the external stimuli. The R_s and C_s parameters are the somatic membrane resistance and capacitance as measured in the hippocampal neurons [19]. The bias term b_j can be treated as the weight of a fixed external input of 1 such that $b_j = \omega_{j0}x_0(t)$ where $x_0(t) = 1$. The parameter I is the number of input signals. The parameter J is the number of outputs neurons. The parameter N is the maximum possible number of unit delay feedbacks of the network. In physical terms, the synaptic weights ω_{ji} represent the feed-forward conductance (in the nS range) for unit *j* from input signal *i*. The feedback conductance r_{jk} is the influence of unit *k* to unit *j*.

We modify the differential equations proposed by Izhikevich [20] to represent the I-V characteristics of a hippocampal CA3 neuron. Two possible three-dimensional representations are obtained. The activation function maps the current-voltage relationship between the ionic flow and electrical potential traversing through cell membrane. They were obtained from the intracellular recordings of the CA1 pyramidal neurons [19]. The function f_1 maps the value and the rate of change in the transmembrane potential to the net current entering the cell.

$$i = f_1\left(v, \frac{dv}{dt}\right) \qquad (2)$$

Alternatively, the function f_2 maps the value and the rate of change in current entering the cell to the transmembrane potential.

$$v = f_2\left(i, \frac{di}{dt}\right) \qquad (3)$$

Depending on the types of neurons of interest, the functions f_1 and f_2 are allowed to be scaled in various fashions. Both of these functions contain regions of linear (sub-threshold) I-V relationship as well as nonlinear (AP) dynamics. In this paper, we will simplify the problem by allowing only linear scaling to the activation functions such that excitation threshold of the network can be altered adaptively. The three parameters (α, β, γ) represent scaling factors for transmembrane potential, rate of change of voltage and effective current, respectively.

III. LEARNING RULE

A novel learning algorithm is needed for the proposed pNL network. The learning process involves the optimization of mean square error **E** as a function of each neural output. The gradients of the target signals are only used for the purpose of result validation but not in the actual error computation.

$$\mathbf{E}(\mathrm{y_r}) = \frac{1}{\mathrm{JC}}\sum_{c=1}^{C}\sum_{\mathrm{j}=1}^{\mathrm{J}}(\mathrm{y_r^c} - \mathrm{t_r^c})^2 \qquad (4)$$

where C denotes the number of training cases, J is the number of output units. The target of the jth unit for training case c is symbolized as t_r^c. Currently, the network parameters are updated using a modified gradient descent method as described below. Update of parameters will be performed by iterative training instead of batch learning.

$$\Delta\omega_{\mathrm{rs}}^{\mathrm{new}} = \omega_{\mathrm{rs}}^{\mathrm{old}} - \eta_\omega \frac{\partial\mathbf{E}}{\partial\omega_{\mathrm{rs}}} \qquad (5)$$

$$\Delta \mathrm{r_{rs}^{n}}^{\mathrm{new}} = \mathrm{r_{rs}^{n}}^{\mathrm{old}} - \eta_{\mathrm{r}} \frac{\partial\mathbf{E}}{\partial \mathrm{r_{rs}^n}} \qquad (6)$$

Where η_ω denotes the gradient descent step size for feed-forward weights, η_r represents the gradient descent step size for feedback weights. Using chain rule, we are able to construct the rate of change in mean square error with respect to network parameters

$$\frac{\partial\mathbf{E}}{\partial\omega_{\mathrm{rs}}} = \frac{\partial\mathbf{E}}{\partial y_{\mathrm{r}}}\frac{\partial \mathrm{y_r}}{\partial\omega_{\mathrm{rs}}} \qquad (7)$$

$$\frac{\partial\mathbf{E}}{\partial \mathrm{r_{rs}^n}} = \frac{\partial\mathbf{E}}{\partial \mathrm{y_r}}\frac{\partial \mathrm{y_r}}{\partial \mathrm{r_{rs}^n}} \qquad (8)$$

The gradient of the system output with respect to system parameters is often required for the optimization problem. By differentiating Eq. (1) with respect to the feed-forward synaptic weights (ω_{rs}), we obtain the following expression:

$$\begin{aligned}\mathrm{C_{sr}}\frac{\partial \mathrm{y_r(t)}}{\partial\omega_{\mathrm{rs}}} &+ \frac{1}{\mathrm{R_{sr}}}\frac{\partial \mathrm{y_r(t)}}{\partial\omega_{\mathrm{rs}}} + \alpha_{\mathrm{r}}\gamma_{\mathrm{r}}\mathrm{y_r(t)}\frac{\partial \mathrm{f_1}(\alpha_{\mathrm{r}}\mathrm{y_r}, \beta_{\mathrm{r}}\dot{\mathrm{y}}_{\mathrm{r}})}{\partial\alpha_{\mathrm{r}}\mathrm{y_r}}\frac{\partial \mathrm{y_r(t)}}{\partial\omega_{\mathrm{rs}}}\\ &+ \beta_{\mathrm{r}}\gamma_{\mathrm{r}}\frac{\partial \mathrm{f_1}(\alpha_{\mathrm{r}}\mathrm{y_r}, \beta_{\mathrm{r}}\dot{\mathrm{y}}_{\mathrm{r}})}{\partial\beta_{\mathrm{r}}\dot{\mathrm{y}}_{\mathrm{r}}}\frac{\partial\dot{\mathrm{y}}_{\mathrm{r}}(\mathrm{t})}{\partial\omega_{\mathrm{rs}}}\\ &= \sum_{\mathrm{i}=0}^{\mathrm{I}}\left(\frac{\partial\omega_{\mathrm{ri}}}{\partial\omega_{\mathrm{rs}}}\mathrm{x_i(t)} + \omega_{\mathrm{ri}}\frac{\partial \mathrm{x_i(t)}}{\partial\omega_{\mathrm{rs}}}\right)\\ &+ \sum_{\mathrm{n}=1}^{\mathrm{N}}\sum_{\mathrm{k}\neq \mathrm{r}}^{\mathrm{J}}\left(\frac{\partial \mathrm{r_{rk}^n}}{\partial\omega_{\mathrm{rs}}}\mathrm{y_k}(\mathrm{t}-\mathrm{n}\tau)\right.\\ &\left.+ \mathrm{r_{rk}^n}\frac{\partial \mathrm{y_k}(\mathrm{t}-\mathrm{n}\tau)}{\partial\omega_{\mathrm{rs}}}\right) \qquad (9)\end{aligned}$$

When we differentiate Eq. **(1)** with respect to the feedback weights at each potential time delay, we have:

$$\begin{aligned}\mathrm{C_{sr}}\frac{\partial\dot{\mathrm{y}}_{\mathrm{r}}(\mathrm{t})}{\partial \mathrm{r_{rs}^n}} &+ \frac{1}{\mathrm{R_{sr}}}\frac{\partial \mathrm{y_r(t)}}{\partial \mathrm{r_{rs}^n}} + \alpha_{\mathrm{r}}\gamma_{\mathrm{r}}\frac{\partial \mathrm{f_1}(\alpha_{\mathrm{r}}\mathrm{y_r}, \beta_{\mathrm{r}}\dot{\mathrm{y}}_{\mathrm{r}})}{\partial\alpha_r \mathrm{y_r}}\frac{\partial \mathrm{y_r(t)}}{\partial \mathrm{r_{rs}^n}}\\ &+ \beta_{\mathrm{r}}\gamma_{\mathrm{r}}\frac{\partial \mathrm{f_1}(\alpha_{\mathrm{r}}\mathrm{y_r}, \beta_{\mathrm{r}}\dot{\mathrm{y}}_{\mathrm{r}})}{\partial\beta_{\mathrm{r}}\dot{\mathrm{y}}_{\mathrm{r}}}\frac{\partial\dot{\mathrm{y}}_{\mathrm{r}}(\mathrm{t})}{\partial \mathrm{r_{rs}^n}}\\ &= \sum_{\mathrm{i}=0}^{\mathrm{I}}\left(\frac{\partial\omega_{\mathrm{ri}}}{\partial \mathrm{r_{rs}^n}}\mathrm{x_i(t)} + \omega_{\mathrm{ri}}\frac{\partial \mathrm{x_i(t)}}{\partial \mathrm{r_{rs}^n}}\right)\\ &+ \sum_{\mathrm{k}\neq \mathrm{r}}^{\mathrm{J}}\left(\frac{\partial \mathrm{r_{rk}^n}}{\partial \mathrm{r_{rs}^n}}\mathrm{y_k}(\mathrm{t}-\mathrm{n}\tau)\right.\\ &\left.+ \mathrm{r_{rk}^n}\frac{\partial \mathrm{y_k}(\mathrm{t}-\mathrm{n}\tau)}{\partial \mathrm{r_{rs}^n}}\right) \qquad (10)\end{aligned}$$

Since the feed-forward weights ω_{rs} and the feedback weights r_{rs}^n are independent, many terms can be simplified. Given,

$$\frac{\partial\omega_{\mathrm{ab}}}{\partial\omega_{\mathrm{cd}}} = \begin{cases}1 & \text{if } a = c, b = d\\ 0 & \text{otherwise}\end{cases} \qquad (11)$$

$$\frac{\partial \mathrm{r_{ab}^n}}{\partial \mathrm{r_{cd}^n}} = \begin{cases}1 & \text{if } a = c, b = d, m = n\\ 0 & \text{otherwise}\end{cases} \qquad (12)$$

Since the input vector x is independent of any synaptic weights, the first three terms on the right hand side of the equation can be compacted into a single variable, G^n, with its elements defined as,

$$\mathrm{G_s^n} = \begin{cases}\mathrm{x_s(t)} & \text{if } s = i \text{ for feed} - \text{forward weights,}\\ \mathrm{y_s}(\mathrm{t}-\mathrm{n}\tau) & \text{if } s = k \text{ for feedback weights.}\end{cases} \qquad (13)$$

After the uncorrelated terms have been taken into account, we are ready to solve for output gradient information in the parameter space using the equations,

$$C_{sr}\frac{\partial \dot{y}_r(t)}{\partial \omega_{rs}} + \frac{1}{R_{sr}}\frac{\partial y_r(t)}{\partial \omega_{rs}} + \alpha_r\gamma_r(t)\frac{\partial f_1(\alpha_r y_r,\beta_r\dot{y}_r)}{\partial \alpha_r y_r}\frac{\partial y_r(t)}{\partial \omega_{rs}} + \beta_r\gamma_r\frac{\partial f_1(\alpha_r y_r,\beta_r\dot{y}_r)}{\partial \beta_r\dot{y}_r}\frac{\partial \dot{y}_r(t)}{\partial \omega_{rs}} = x_s(t) \qquad (14)$$

and,

$$C_{sr}\frac{\partial \dot{y}_r(t)}{\partial r^n_{rs}} + \frac{1}{R_{sr}}\frac{\partial y_r(t)}{\partial r^n_{rs}} + \alpha_r\gamma_r\frac{\partial f_1(\alpha_r y_r,\beta_r\dot{y}_r)}{\partial \alpha_r y_r}\frac{\partial y_r(t)}{\partial r^n_{rs}} + \beta_r\gamma_r\frac{\partial f_1(\alpha_r y_r,\beta_r\dot{y}_r)}{\partial \beta_r\dot{y}_r}\frac{\partial \dot{y}_r(t)}{\partial r^n_{rs}} = y_s(t-n\tau) \qquad (15)$$

The derivatives of the output with respect to system parameters ω_{ji} and r^n_{ji} can be extractable by organizing the simplified differential equations into matrix form.

$$\boldsymbol{A} * \dot{\boldsymbol{D}}_{\boldsymbol{F}}(t) = -\boldsymbol{B} * \boldsymbol{D}_{\boldsymbol{F}}(t) - \begin{bmatrix}1\\ \vdots\\ 1\end{bmatrix}\boldsymbol{G} \qquad (16)$$

and,

$$\boldsymbol{A} * \dot{\boldsymbol{D}}^{\boldsymbol{n}}_{\boldsymbol{B}}(t) = -\boldsymbol{B} * \boldsymbol{D}^{\boldsymbol{n}}_{\boldsymbol{B}}(t) - \begin{bmatrix}1\\ \vdots\\ 1\end{bmatrix} \qquad (17)$$

where $\boldsymbol{D}_F(t)$ is the matrix of system output gradient with respect the feed-forward weights denoted in Eq. **(7)**, $\boldsymbol{D}^{\boldsymbol{n}}_{\boldsymbol{B}}(t-n\tau)$ is the matrix of output gradient with respect to the feedback weights in Eq. **(8)** after *n* number of unit delays. Notice that for these two equations, the matrix ***G*** is different according to Eq. **(13)**. More specifically,

$$\boldsymbol{D}_{\boldsymbol{F}}(t) = \begin{bmatrix}
\frac{\partial y_1(t)}{\partial \omega_{11}} & \frac{\partial y_1(t)}{\partial \omega_{12}} & \cdots & \frac{\partial y_1(t)}{\partial \omega_{1i}} & \cdots & \frac{\partial y_1(t)}{\partial \omega_{1I}}\\
\frac{\partial y_2(t)}{\partial \omega_{21}} & \frac{\partial y_2(t)}{\partial \omega_{22}} & \cdots & \frac{\partial y_2(t)}{\partial \omega_{2i}} & \cdots & \frac{\partial y_2(t)}{\partial \omega_{2I}}\\
\vdots & \vdots & & \vdots & & \vdots\\
\frac{\partial y_j(t)}{\partial \omega_{j1}} & \frac{\partial y_j(t)}{\partial \omega_{j2}} & \cdots & \frac{\partial y_j(t)}{\partial \omega_{ji}} & \cdots & \frac{\partial y_j(t)}{\partial \omega_{jI}}\\
\vdots & \vdots & & \vdots & & \vdots\\
\frac{\partial y_J(t)}{\partial \omega_{J1}} & \frac{\partial y_J(t)}{\partial \omega_{J2}} & \cdots & \frac{\partial y_J(t)}{\partial \omega_{Ji}} & \cdots & \frac{\partial y_J(t)}{\partial \omega_{JI}}
\end{bmatrix} \qquad (18)$$

$$\boldsymbol{D}^{\boldsymbol{n}}_{\boldsymbol{B}}(t) = \begin{bmatrix}
0 & \frac{\partial y_1(t)}{\partial r^n_{12}} & \cdots & \frac{\partial y_1(t)}{\partial r^n_{1j}} & \cdots & \frac{\partial y_1(t)}{\partial r^n_{1j}}\\
\frac{\partial y_2(t)}{\partial r^n_{21}} & 0 & \cdots & \frac{\partial y_2(t)}{\partial r^n_{2j}} & \cdots & \frac{\partial y_2(t)}{\partial r^n_{2j}}\\
\vdots & \vdots & & \vdots & & \vdots\\
\frac{\partial y_k(t)}{\partial r^n_{k1}} & \frac{\partial y_k(t)}{\partial r^n_{k2}} & \cdots & 0 & \cdots & \frac{\partial y_k(t)}{\partial r^n_{kj}}\\
\vdots & \vdots & & \vdots & & \vdots\\
\frac{\partial y_K(t)}{\partial r^n_{K1}} & \frac{\partial y_K(t)}{\partial r^n_{K2}} & \cdots & \frac{\partial y_K(t)}{\partial r^n_{Kj}} & \cdots & 0
\end{bmatrix} \qquad (19)$$

$$\dot{\boldsymbol{D}}_{\boldsymbol{F}}(t) = \begin{bmatrix}
\frac{\partial \dot{y}_1(t)}{\partial \omega_{11}} & \frac{\partial \dot{y}_1(t)}{\partial \omega_{12}} & \cdots & \frac{\partial \dot{y}_1(t)}{\partial \omega_{1i}} & \cdots & \frac{\partial \dot{y}_1(t)}{\partial \omega_{1I}}\\
\frac{\partial \dot{y}_2(t)}{\partial \omega_{21}} & \frac{\partial \dot{y}_2(t)}{\partial \omega_{22}} & \cdots & \frac{\partial \dot{y}_2(t)}{\partial \omega_{2i}} & \cdots & \frac{\partial \dot{y}_2(t)}{\partial \omega_{2I}}\\
\vdots & \vdots & & \vdots & & \vdots\\
\frac{\partial \dot{y}_j(t)}{\partial \omega_{j1}} & \frac{\partial \dot{y}_j(t)}{\partial \omega_{j2}} & \cdots & \frac{\partial \dot{y}_j(t)}{\partial \omega_{ji}} & \cdots & \frac{\partial \dot{y}_j(t)}{\partial \omega_{jI}}\\
\vdots & \vdots & & \vdots & & \vdots\\
\frac{\partial \dot{y}_J(t)}{\partial \omega_{J1}} & \frac{\partial \dot{y}_J(t)}{\partial \omega_{J2}} & \cdots & \frac{\partial \dot{y}_J(t)}{\partial \omega_{J1}} & \cdots & \frac{\partial \dot{y}_J(t)}{\partial \omega_{JI}}
\end{bmatrix} \qquad (20)$$

The matrices **A** and **B** only contain non-zero values along their main diagonal elements.

$$diag(\boldsymbol{A}) = \begin{bmatrix}
C_{s1} + \beta_1\gamma_1\frac{\partial f_1(\alpha_{r1}y_1,\beta_1\dot{y}_1)}{\partial \beta_1\dot{y}_1}\\
\vdots\\
C_{sj} + \beta_j\gamma_j\frac{\partial f_1(\alpha_j y_j,\beta_j\dot{y}_j)}{\partial \beta_j\dot{y}_j}\\
\vdots\\
C_{sJ} + \beta_J\gamma_J\frac{\partial f_J(\alpha_J y_J,\beta_J\dot{y}_J)}{\partial \beta_J\dot{y}_J}
\end{bmatrix} \qquad (21)$$

$$diag(\boldsymbol{B}) = \begin{bmatrix}
\frac{1}{R_{s1}} + \alpha_1\gamma_1\frac{\partial f_1(\alpha_1 y_1,\beta_1\dot{y}_1)}{\partial \alpha_1 y_1}\\
\vdots\\
\frac{1}{R_{sj}} + \alpha_j\gamma_j\frac{\partial f_1(\alpha_j y_j,\beta_j\dot{y}_j)}{\partial \alpha_j y_j}\\
\vdots\\
\frac{1}{R_{sJ}} + \alpha_J\gamma_J\frac{\partial f_J(\alpha_J y_J,\beta_J\dot{y}_J)}{\partial \alpha_J y_J}
\end{bmatrix} \qquad (22)$$

$$R = \begin{bmatrix}
0 & r_{12} & r_{13} & r_{1k} & \cdots & r_{1K}\\
r_{21} & 0 & r_{23} & r_{2k} & \ddots & r_{2K}\\
r_{j1} & r_{j2} & 0 & & & r_{jK}\\
\vdots & & & \ddots & & \vdots\\
\vdots & & & & \ddots & r_{J-1,K}\\
r_{J1} & r_{J2} & r_{Jk} & \cdots & r_{J,K-1} & 0
\end{bmatrix} \qquad (23)$$

If the neural units contain self-feedbacks, they can be incorporated in matrix **R** quite readily by replacing the main diagonal elements with non-zero self-feedback weights. By simply solving the matrices from Eq. **(16)** and **(17)**, a general solution of the rate of change can be obtained as a function of time.

$$\mathbf{D}(t)_J = -\mathbf{B}^{-1}\left(\begin{bmatrix}1\\ \vdots\\ 1\end{bmatrix}\boldsymbol{G}\right)\left(1 - e^{-\mathbf{A}^{-1}\mathbf{B}_t}\right) \qquad (24)$$

where $\mathbf{D}$ can be either $\mathbf{D_F}$ or $\mathbf{D^n_B}$ with initial conditions $\mathbf{D_F}(0)$=0 and $\mathbf{D^n_B}(0)$=0. The convergence criteria for system parameters will need to be studied. For the time dependent solution for **D**(t), it should be ensured that the exponents do not approach infinite, i.e. the product $\mathbf{A^{-1}B}$ must be a strictly non-position matrix. The solution **D**(t) can be substituted into Eq. **(7)** and **(8)** to iteratively update the feed-forward and feedback weights.

Another novel approach to this model is that the parameters of the activation function can also be tuned such that excitation threshold can be altered to match either sub-threshold or supra-threshold activities. This capability is realized similarly to the training of the synaptic weights. Only simple linear operations such as expansion and compression of the membrane potential and its rate of change, as well as the current are considered in this paper.

$$\begin{aligned} C_{sr}\frac{\partial \dot{y}_r(t)}{\partial a_r} + \frac{1}{R_{sr}}\frac{\partial y_r(t)}{\partial a_r} + \gamma_r \frac{\partial f_1(\alpha_r y_{r,}\ \beta_r \dot{y}_r)}{\partial \alpha_r y_r}\frac{\partial \alpha_r \gamma_r(t)}{\partial \alpha_r} \\ + \beta_r\gamma_r \frac{\partial f_1(\alpha_r y_{r,}\ \beta_r \dot{y}_r)}{\partial \beta_r \dot{y}_r}\frac{\partial \dot{y}_r(t)}{\partial \alpha_r} \\ = \sum_{i=0}^{I}\left(\frac{\partial \omega_{ri}}{\partial \alpha_r}x_i(t) + \omega_{ri}\frac{\partial x_i(t)}{\partial \alpha_r}\right) \\ + \sum_{n=1}^{N}\sum_{k\neq r}^{J}\left(\frac{\partial r_{rk}^{n}}{\partial \alpha_r}y_k(t-n\tau) \right. \\ \left. + r_{rk}^{n}\frac{\partial y_k(t-n\tau)}{\partial \alpha_r}\right) \qquad (25) \end{aligned}$$

$$\begin{aligned} C_{sr}\frac{\partial \dot{y}_r(t)}{\partial \beta_r} + \frac{1}{R_{sr}}\frac{\partial y_r(t)}{\partial \beta_r} + \alpha_r\gamma_r \frac{\partial f_1(\alpha_r y_{r,}\ \beta_r \dot{y}_r)}{\partial \alpha_r y_r}\frac{\partial y_r(t)}{\partial \beta_r} \\ + \gamma_r \frac{\partial f_1(\alpha_r y_{r,}\ \beta_r \dot{y}_r)}{\partial \beta_r \dot{y}_r}\frac{\partial \beta_r \gamma_r(t)}{\partial \beta_r} \\ = \sum_{i=0}^{I}\left(\frac{\partial \omega_{ri}}{\partial \beta_r}x_i(t) + \omega_{ri}\frac{\partial x_i(t)}{\partial \beta_r}\right) \\ + \sum_{n=1}^{N}\sum_{k\neq r}^{J}\left(\frac{\partial r_{rk}^{n}}{\partial \beta_r}y_k(t-n\tau) \right. \\ \left. + r_{rk}^{n}\frac{\partial y_k(t-n\tau)}{\partial \beta_r}\right) \qquad (26) \end{aligned}$$

$$\begin{aligned} C_{sr}\frac{\partial \dot{y}_r(t)}{\partial \gamma_r} + \frac{1}{R_{sr}}\frac{\partial y_r(t)}{\partial \gamma_r} + \left(\frac{\partial \gamma_r f_1(\alpha_r y_{r,}\ \beta_r \dot{y}_r)}{\partial \gamma_r}\right) = \\ = \sum_{i=0}^{I}\left(\frac{\partial \omega_{ri}}{\partial \gamma_r}x_i(t) + \omega_{ri}\frac{\partial x_i(t)}{\partial \gamma_r}\right) \\ + \sum_{n=1}^{N}\sum_{k\neq r}^{J}\left(\frac{\partial r_{rk}^{n}}{\partial \gamma_r}y_k(t-n\tau) \right. \\ \left. + r_{rk}^{n}\frac{\partial y_k(t-n\tau)}{\partial \gamma_r}\right) \qquad (27) \end{aligned}$$

These equations can be simplified using chain rule and by factoring out the unrelated terms. The partial derivatives of the activation f1 with respect to each of the scaling parameter can be obtained via a look-up table once the **I-**V curve has been measured.

$$\begin{aligned} C_{sr}\frac{\partial \dot{y}_r(t)}{\partial a_r} + \frac{1}{R_{sr}}\frac{\partial y_r(t)}{\partial a_r} + \alpha_r\gamma_r \frac{\partial f_1(\alpha_r y_{r,}\ \beta_r \dot{y}_r)}{\partial \alpha_r y_r}\frac{\partial \gamma_r(t)}{\partial \alpha_r} \\ + \beta_r\gamma_r \frac{\partial f_1(a_r y_{r,}\ \beta_r \dot{y}_r)}{\partial \beta_r \dot{y}_r}\frac{\partial \dot{y}_r(t)}{\partial \alpha_r} \\ = -\frac{\partial f_1(\alpha_r y_{r,}\ \beta_r \dot{y}_r)}{\partial \alpha_r y_r} y_r(t) \qquad (28) \end{aligned}$$

$$\begin{aligned} C_{sr}\frac{\partial \dot{y}_r(t)}{\partial \beta_r} + \frac{1}{R_{sr}}\frac{\partial y_r(t)}{\partial \beta_r} + \alpha_r\gamma_r(t) \frac{\partial f_1(\alpha_r y_{r,}\ \beta_r \dot{y}_r)}{\partial \alpha_r y_r}\frac{\partial y_r(t)}{\partial \beta_r} \\ + \beta_r\gamma_r \frac{\partial f_1(\alpha_r y_{r,}\ \beta_r \dot{y}_r)}{\partial \beta_r \dot{y}_r}\frac{\partial \dot{y}_r(t)}{\partial \beta_r} \\ = -\frac{\partial f_1(\alpha_r y_{r,}\ \beta_r \dot{y}_r)}{\partial \beta_r \dot{y}_r} \dot{y}_r(t) \qquad (29) \end{aligned}$$

$$\begin{aligned} C_{sr}\frac{\partial \dot{y}_r(t)}{\partial \gamma_r} + \frac{1}{R_{sr}}\frac{\partial y_r(t)}{\partial \gamma_r} + \alpha_r\gamma_r(t) \frac{\partial f_1(\alpha_r y_{r,}\ \beta_r \dot{y}_r)}{\partial \alpha_r y_r}\frac{\partial y_r(t)}{\partial \beta_r} \\ + \beta_r\gamma_r \frac{\partial f_1(\alpha_r y_{r,}\ \beta_r \dot{y}_r)}{\partial \beta_r \dot{y}_r}\frac{\partial \dot{y}_r(t)}{\partial \beta_r} \\ = -f_1(\alpha_r y_{r,}\ \beta_r \dot{y}_r) \qquad (30) \end{aligned}$$

Once the rate of change of the error term **E** with respect to each of these scaling parameters are obtained, they can be solved iteratively, the same way as the feed-forward and feedback weights.

IV. ILLUSTRATIVE EXAMPLE

To validate the proposed learning rule, an artificial network undergoing sub-threshold activity is simulated. Sub-threshold activity is assumed to follow simple differential equations. Here we give an example on how the pNL was used to compare the solutions of the differential equations to the network outputs emulating observed transmembrane activities subjected to sub-threshold random inputs after supervised learning. To simulation such behavior, we test a 2 by 2 hippocampus model undergoing sub-threshold stimulation. The inputs *x(t)* are Gaussian white noise signals with standard deviation of 5μV. The output of each neural unit is assumed to follow:

$$\begin{aligned} y_1(t) &= 0.3x_1(t) + 0.2x_2(t) + 0.5y_2(t-\tau) \\ y_2(t) &= 0.4x_1(t) - 0.2x_2(t) - 0.4y_1(t-\tau) \quad (31) \end{aligned}$$

The dynamic range for the state variables for the above differential equation is scaled such that they correspond to the sub-threshold neuronal dynamic range (±10μV). Linear scaling of the activation function *f₁* was also incorporated. Simple gradient descent step size is chosen to be very small (approximately 50pS). The final range of synaptic weights (which may represent the synaptic conductance between neurons) is approximately 2nS. Fig. **(5)** illustrates the result of a successful sub-threshold simulation after 10 training iterations. The 2-cell pNL network is able to reproduce the simulated signals. It can be seen in Fig. **(6)** that the mean square error decreases asymptotically as the number of training iterations goes up.

V. DISCUSSION

Hodgkin-Huxley (HH) [17] and the Integrate-and-Fire (IF) models have been combined [20] to model different cortical neuron types. A set of relatively simple second order differential equation can be coupled to represent network dynamics. Many artificial neural network models have been proposed to mimic behaviors of biological neural networks (BNNs). These models can be categorized into two classes: static and dynamic neural networks. Static neural networks simulate only long-term memory properties of neurons in which input and output are at their steady state. However, static NNs such as McCulloch-Pitts neurons [1] cannot simulate adaptive properties of neurons or short-term memory, which are very important to spatial-temporal pattern recognition. However, synaptic weights are not static. The processes of desensitization and massager replenishment are essential in changing the efficiency of synaptic weights. This paper is inspired by the recent interest in area of neurodynamics in which the subject of neural network is viewed as nonlinear dynamical systems, and standard neural network models are extended to incorporate for the use of temporal processing.

Traditionally, the main emphasis of neurodynamics is placed on the problem of stability. The former mathematical derivation of neurodynamical system operations were heavily relied on the assumption that the system converges to some steady states before new iterations in learning take place. However, our approach is different with the realization that accurate representation of transient dynamics is an important aspect of modeling as well. Our proposed network paradigm is a major deviation from the traditional view of neural network design and modeling. While traditional sigmoidal networks are unable to accommodate temporal information such as refractoriness (which is essential in neuronal behaviors), our highly nonlinear activation function from the I-V measurement allows refractoriness. In addition, recurrent architecture appears to be a more accurate representation of neuron connectivity.
The advantages of a parallel organized nonlinear activation function are numerous. Its characteristics can be summarized into four points. First, it is inspired by biological measurement. The I-V curve is directly measured from the hippocampus [23]; its wave shape is not an arbitrarily defined Dirac delta function [8, 9] intentionally placed at whenever the integrated synaptic current exceeded a hard threshold. The intrinsic parameters of the activation function allows for excitation as well as refractoriness, much similar to real neurons. These characteristics may also be fine-tuned to customize into various neuron types. Second, it is connected in a highly parallel manner. While traditional activation functions are applied in series with the path of data propagation in an ANN, parallel connected nonlinearity makes learning more difficult. Traditional learning paradigms of error propagation can be performed much easier when nonlinearity is organized in a serial manner. With a parallel structure, it is required simultaneous differential equations. Third, the activation function is highly nonlinear. As a result there is no simple analytic expression for the relations between the measured membrane current and voltage, one possible solution to this problem is to incorporate the first derivative information as one additional variable in describing the I-V characteristic. Even with this modification, gradient information remains difficult to compute. Taylor series expansion can be employed around the excitation threshold so that an analytical expression can be obtained. A look-up table approach also aids in simplifying the computation of the gradient descent method. Fourth, it can be adapted by solving for the appropriate scaling factors in the activation function instead of having them held as constants. This has biological significance in that the excitation threshold of a spiking neural unit can be trained dynamically to match the application need.

ACKNOWLEDGMENT
The project is supported by the NIH NCRR INBRE grant P20RR01645 and the Louisiana Board of Regents RCS LEQSF (2007-10)-RD-A-20 grant. The author would also like to thank Mr. Gustavo Gutierrez for his editorial work on the manuscript.

REFERENCES

[1] McCulloch W. and Pitts W.H. (1943) *Bulletin of Mathematical Biophysics*, 5, 115–133.
[2] Rumelhart D., Hinton G. and Williams R. (1986) *Parallel Distributed Processing*, 1, 318–362.
[3] Rieke F., Warland D., de Ruyter van Steveninck R. and Bialek W. (1997) *Spikes: Exploring the Neural Code*, MIT Press.
[4] Silberberg G., Bethge M., Markram H., Pawelzik K. and Tsodyks M. (2004) *Journal of Neurophysiology*, 91:704–709.
[5] Ferster D. and Spruston N. (1995) *Science*, 270: 756–757.
[6] Thorpe S., Delorme A. and Rullen R.V. (2001) *Neural Networks*, 14: 715–726.
[7] Huxter J., Burgess N. and O'Keefe, J. (2003) *Nature*, 425:828–832.
[8] Bohte S.M., Kok J.N. and Poutr´e H.L., (2000) *Proceedings of the European Symposium on Artificial Neural Networks*, 419–425.
[9] Christodoulou C., Bugmann G. and Clarkson T.G. (2002) *Neural Networks*, 15: 891–908.
[10] Geisler C. and Goldberg J.M. (1966) *Biophysical Journal*, 6: 53-69.
[11] Maass W. and Bishop C.M. (1999) *Pulsed Neural Networks*. MIT Press.
[12] Natschlager T., Maass W. and Zador A. (2001) *Network: Computation in Neural Systems*, 12: 75– 87.
[13] Mehrtash N., Jung D., Hellmich H.H., Schoenauer T., Lu V.T. and Klar H. (2003) *IEEE Transactions on Neural Networks*, 14: 980–992.
[14] Sekerli M., Negro C.A.D., Lee R.H. and Butera R.J. (2004) *IEEE Transactions on Biomedical Engineering*, 51:1665–1672.
[15] Swiercz W., Cios K.J., Staley K., Kurgen L., Accurso F. and Sagel S. (2006) *IEEE Transactions on Neural Networks*, 17:94–105.
[16] Haykin S. (2008) *Neural networks and Learning Machines*, Prentice Hall.
[17] Hodgkin L. and Huxley A.F. (1952) *Journal of Physiology*, 117:500-544.

[18] Fu P., Bardakjian B.L., D'Aguanno A. and Carlen P.L. (1989) *IEEE Transactions on Biomedical Engineering*, 36:55–64.
[19] Bardakjian B.L. (1989) *IEEE 11th Annual Conference in EMBS*, 11:1284–1285.
[20] Izhikevich E.M. (2003) *IEEE Transactions on Neural Networks*, 14:1569–1572.
[21] Rall W. (1969) *Biophysical Journal*, 9:1483–1508.
[22] Rall W. (1977) *Handbook of Physiology: the nervous system 1*, American Physiological Society.
[23] Marmarelis V.Z. (1994) *Advanced Methods of Physiological System Modeling*, Plenum Press.

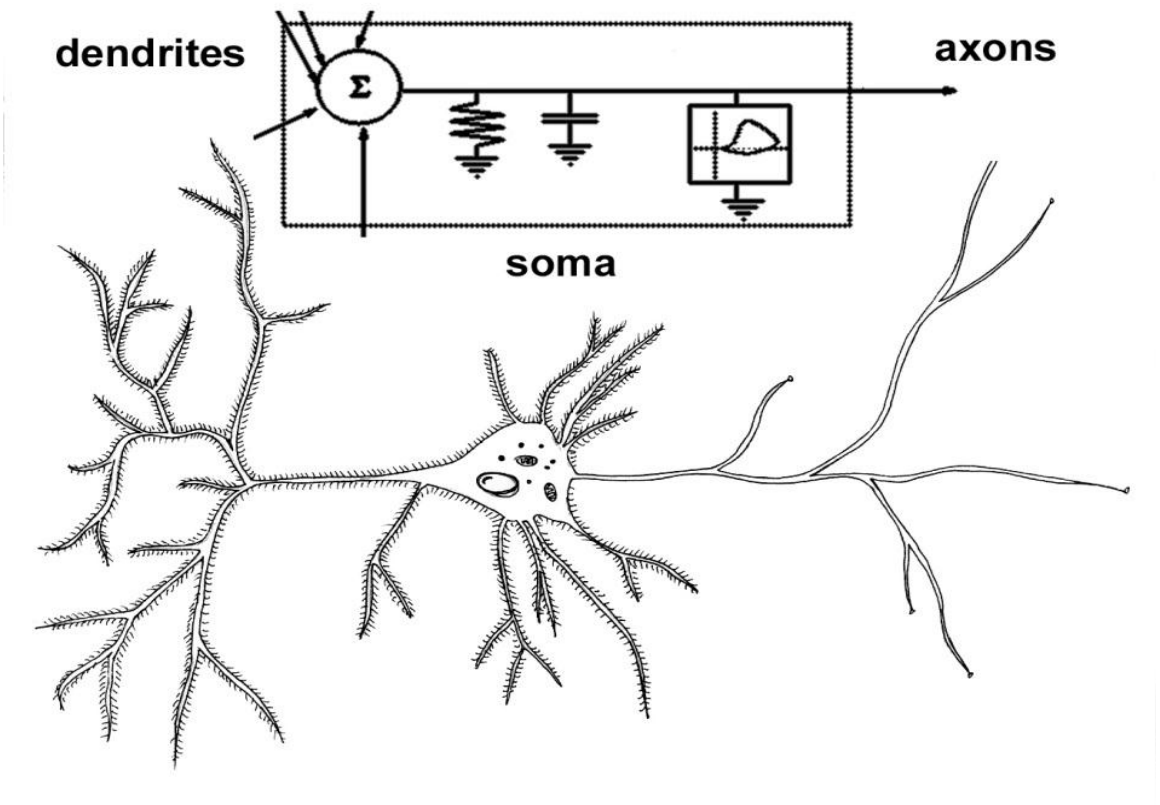

Fig. 1 – A block diagram representation of an artificial neural unit as a somatic model of the hippocampal neuron. The dendrites are not included in this model hence the RC ladder structure is not shown.

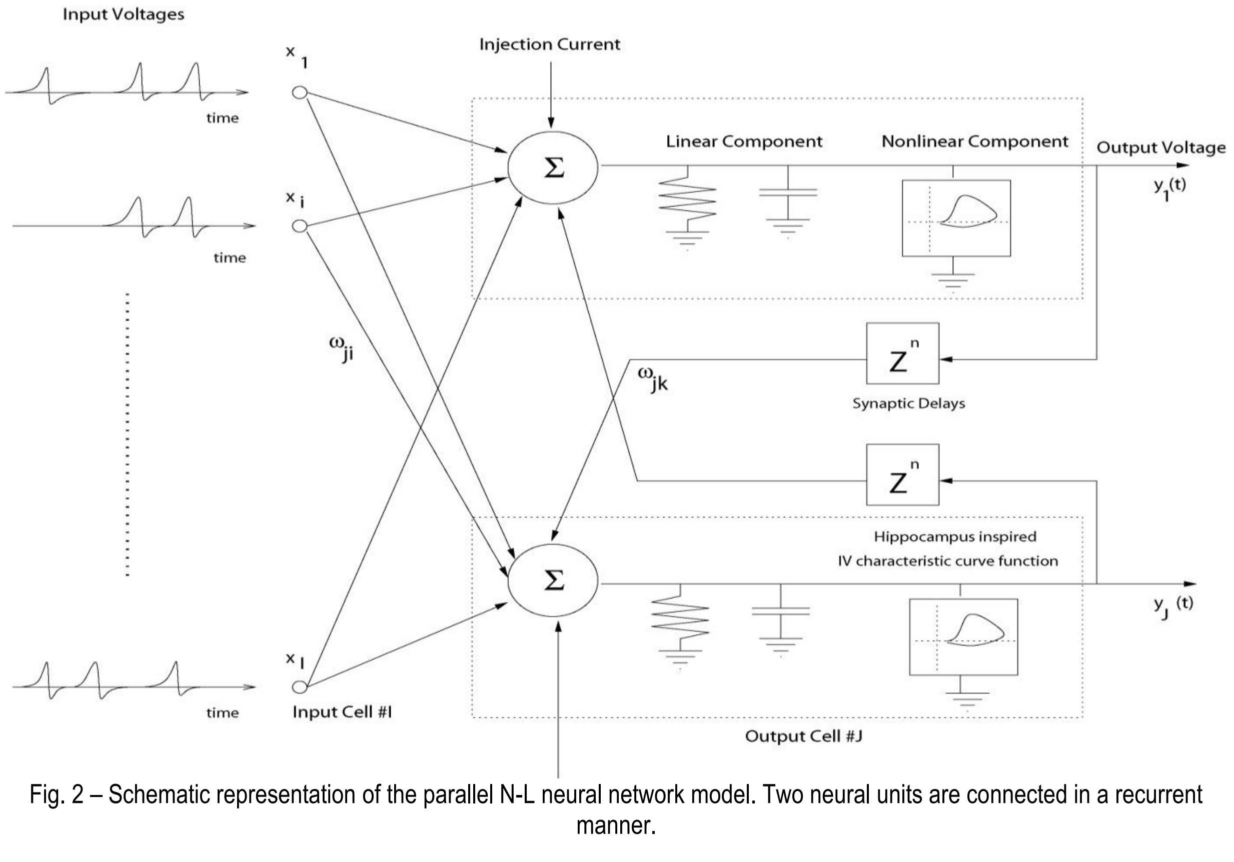

Fig. 2 – Schematic representation of the parallel N-L neural network model. Two neural units are connected in a recurrent manner.

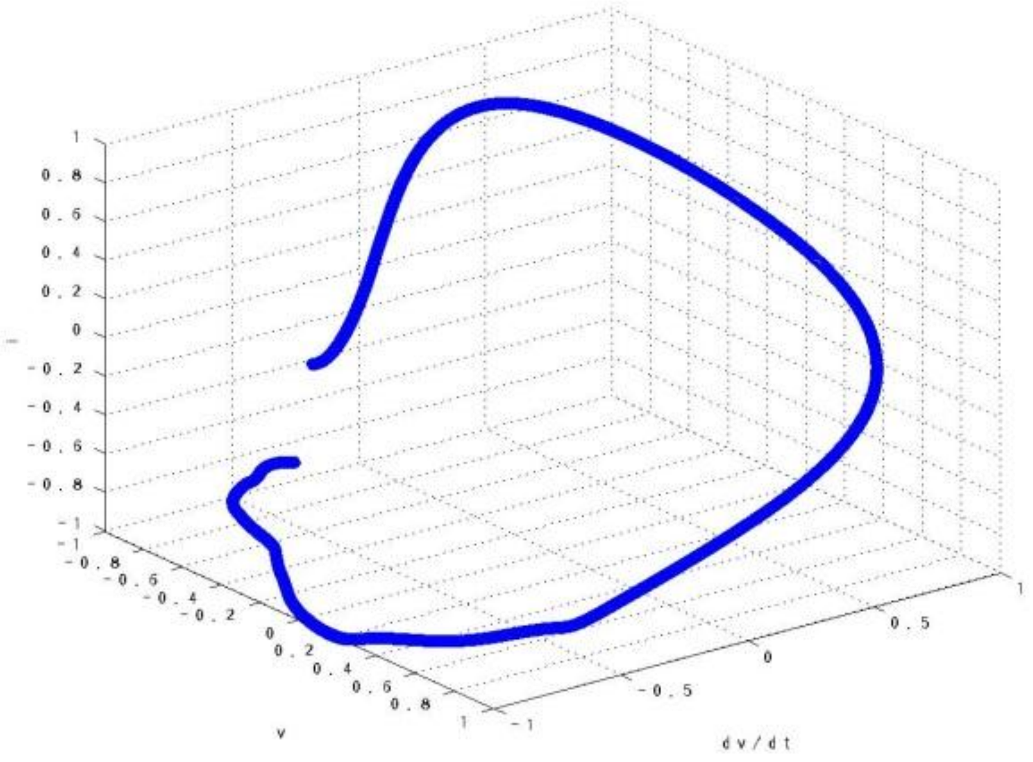

Fig. 3 – Activation function used to compute gradient information, where given the transmembrane voltage at the soma and its derivative, one can obtain an estimate of current flow.

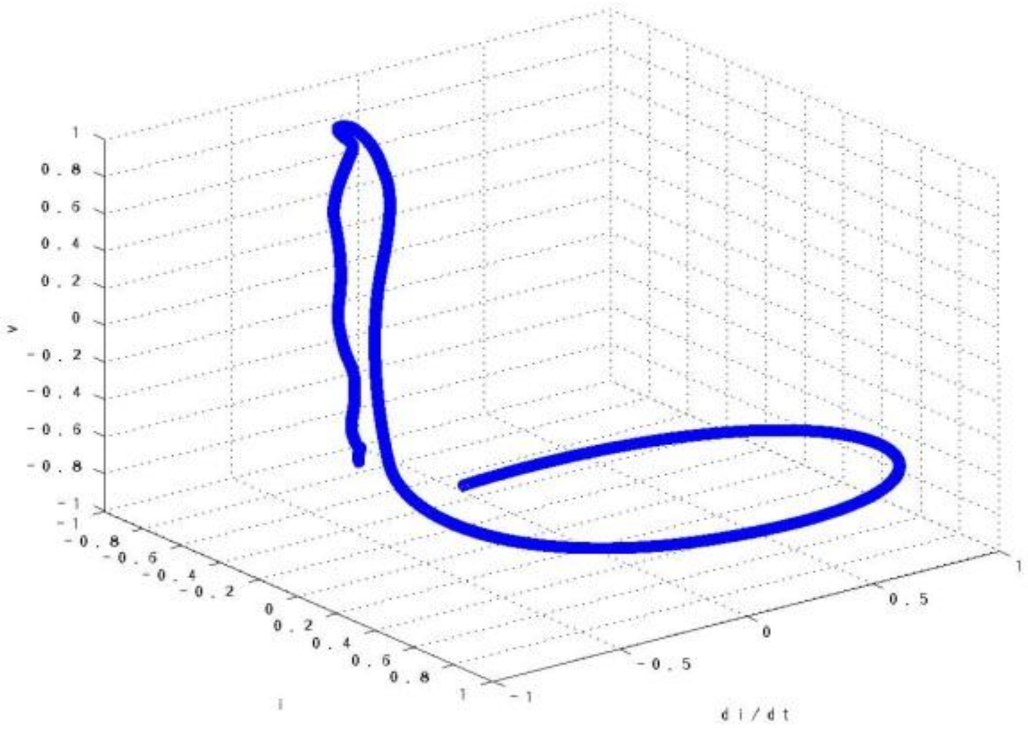

Fig. 4 – Activation function used to compute the final output voltage from the intermediate current sum. Transmembrane voltage can be obtained from the amount and rate of change of total current influx.

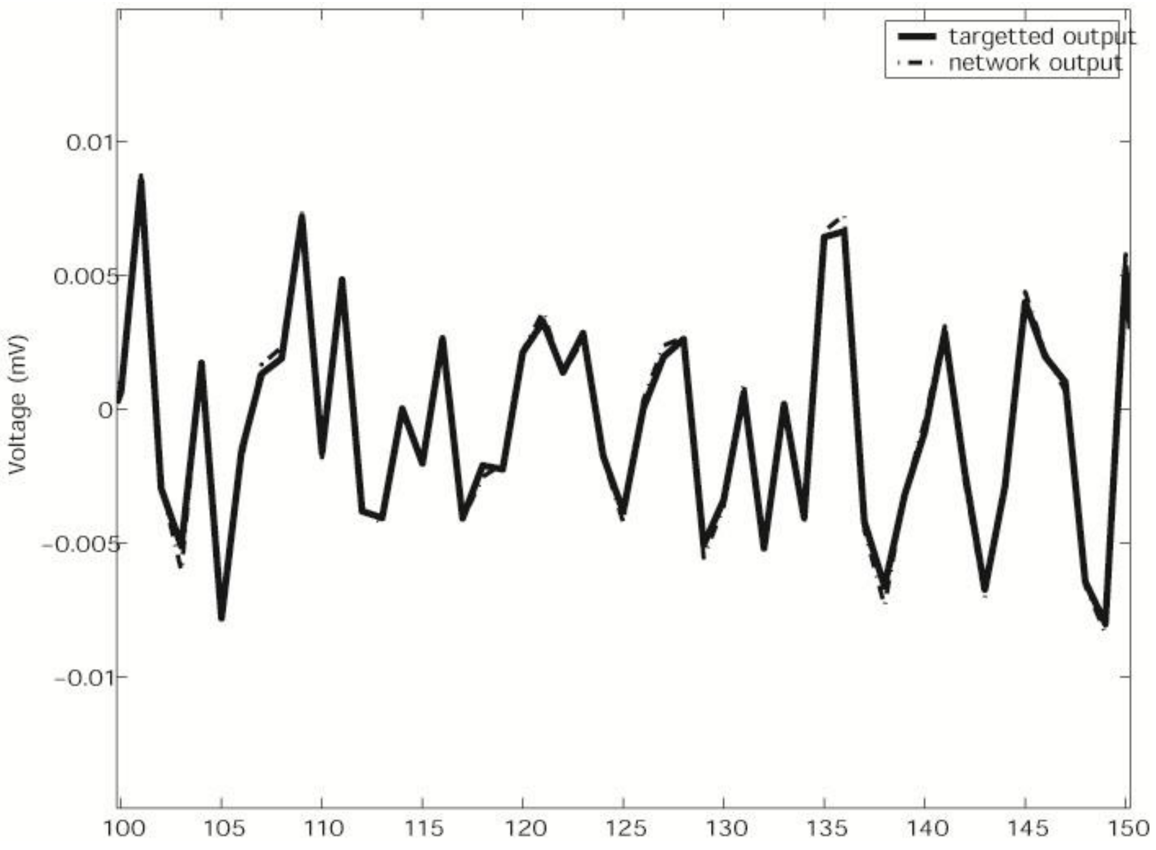

Fig. 5 – Simulation of sub-threshold activities using a hippocampus neurodynamical ANN model. A detailed comparison between the actual output and the simulated data, illustrating that the network is able to simulate sub-threshold activities is shown. The solid line represents the targeted sub-threshold activities obeying simple differential equations. The dashed line is the network output after 10 iterations of training.

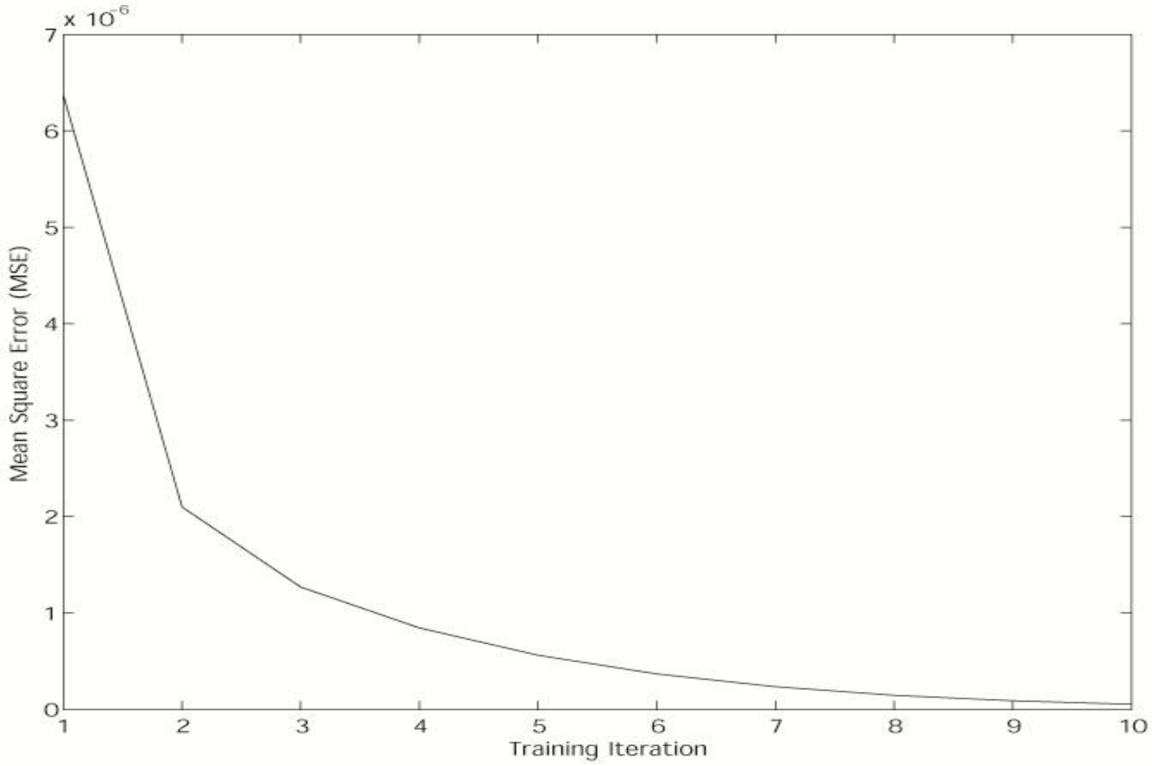

Fig. 6 – Mean Squared Error versus training iterations for sub-threshold activities.