49th IEEE Conference on Decision and Control
December 15-17, 2010
Hilton Atlanta Hotel, Atlanta, GA, USA

Indirect Training of a Spiking Neural Network for Flight Control via Spike-Timing-Dependent Synaptic Plasticity

Greg Foderaro, Craig Henriquez, and Silvia Ferrari

Abstract— **Recently, spiking neural networks (SNNs) have been shown capable of approximating the dynamics of biological neuronal networks, and of being trainable by biologically-plausible learning mechanisms, such as spike-timing-dependent synaptic plasticity. Numerical simulations also support the possibility that they may possess universal function approximation abilities. However the effectiveness of training algorithms to date is far inferior to those of other artificial neural networks. Moreover, they rely on directly manipulating the SNN weights, which may not be feasible in a number of their potential applications. This paper presents a novel indirect training approach to modulate spike-timing-dependent plasticity (STDP) in an action SNN that serves as a flight controller without directly manipulating its weights. A critic SNN is directly trained with a reward-based Hebbian approach to send spike trains to the action SNN, which in turn controls the aircraft and learns via STDP. The approach is demonstrated by training the action SNN to act as a flight controller for stability augmentation. Its performance and dynamics are analyzed before and after training through numerical simulations and Poincaré maps.**

I. Introduction and Motivation

Spiking neural networks (SNNs), such as integrate-and-fire (IF) models [1], and the biophysical Hodgkin-Huxley (HH) model [2], are computational models of biological neurons that consist of systems of differential equations which can reproduce spike dynamics and learning mechanisms observed in real neuronal networks. Although recently SNNs have been shown capable of simulating sigmoidal artificial neural networks (ANNs) [3], the effectiveness of SNN training algorithms is still far removed from that of ANN algorithms such as backpropagation. SNNs offer several advantages over sigmoidal ANNs. Not only are they biologically plausible, but through temporal coding/decoding they can potentially reproduce the computational speeds observed in biological brains. For example, it has been shown that biological neuronal networks can carry out visual pattern analysis with ten-layer neuronal networks in only 100 ms [4], [5]. Moreover, SNNs can be used to build computational models of cortical regions, as shown in [6], because they can simulate the ability of biological neuronal networks to use the precise timing of single spikes to encode information [7], [8].

In an effort to achieve the effectiveness of ANN backpropagation training algorithms and use SNNs for solving complex problems and tasks, SNN backpropagation algorithms have been proposed in [9]. One of the main difficulties to overcome in this research approach is that the signals and errors produced by SNNs are piece-wise continuous, whereas gradient-descent methods typically are not applicable to objective functions that are not continuously differentiable. More importantly, despite numerous research efforts to this date, there is no experimental evidence supporting the existence of biological mechanisms based on the propagation of errors. The work presented in this paper follows an alternate line of research which involves the development of training algorithms that modify the synaptic efficacies through biologically inspired methods including spike-timing-dependent plasticity (STDP) and Hebbian synaptic plasticity. These algorithms have been used to train SNNs to solve simple nonlinear function approximation problems, such as approximating the XOR gate in [10]–[15]. This paper presents a novel supporting training approach that may be used to indirectly train a randomly connected network to serve as a flight controller.

The training approach presented in this paper is motivated by potential SNN applications such as neuroprosthetic devices implanted in the brain to substitute a motor, sensory, or cognitive modality damaged by an injury or disease, and *in-vitro* light-sensitive biological neuronal networks utilized for basic neuroscience research. In these applications, the objective is to stimulate *in-vivo* or *in-vitro* biological neurons to perform a complex function such as processing an auditory signal, or restoring a cognitive function. In neuroprosthetic applications, the artificial device may consist of a micro-electrode array or integrated circuit that stimulates biological neurons via spike trains, but the devices are not capable of directly modifying the neurons' synaptic efficacies, as many SNN training algorithms do. With *in-vitro* neuronal networks, an artificial SNN on a computer can be used to provide input signals to the culture by optically stimulating the neurons with controlled blue light in an attempt to induce plasticity. The resulting network response can be continuously recorded and processed in real time with high spatial and temporal resolution by spike detection and sorting software [16], for the purpose of verifying models and mechanisms by which biological neuronal networks execute the control and storage of information via temporal coding.

In-vitro neuronal networks with random connectivity are produced with a method known as "Banker Cultures", which grows neurons on top of a monolayer of astrocytes [16], [17]. In these networks and in *in-vivo* networks, the actual connectivity and synaptic plasticities are typically unknown

Greg Foderaro and Silvia Ferrari are with the Laboratory for Intelligent Systems and Control (LISC), Department of Mechanical Engineering, Duke University, Durham, NC 27708-0005, {greg.foderaro, sferrari}@duke.edu

Craig Henriquez is with the Department of BiomedicalEngineering, Duke University, Durham, NC 27708-0005, ch@duke.edu

and may change over time as a result of brain plasticity. Therefore, the training approach presented in this paper aims to induce changes in the synaptic efficacies of a randomly connected action SNN by modulating STDP through input spike trains produced by a critic SNN. The critic SNN is trained directly by adjusting its weights through a biologically-plausible algorithm, while the action SNN attempts to control a simulated aircraft with poor short-period flying qualities. Since the action SNN's synaptic strengths cannot be adjusted directly, the critic SNN must learn how to induce the desired plasticity in the action SNN by means of a reward-modulated Hebbian learning algorithm which computes the reward based on the errors between the action SNN output and an optimal control law. Since the SNNs' weights are adjusted online, the approach can be used for adaptive control via SNN.

The novel adaptive control architecture and algorithm are presented in Section III and applied to the SNN architecture described in II. Section V shows the results of the algorithm and includes analysis on the stability and dynamic behavior of the system, and it is demonstrated that the network is capable of learning the aircraft controller computations presented in Section IV.

II. Spiking Neural Network Model

Spiking neural networks seek to provide complete and accurate dynamic models for biological neurons, including support for action potential generation, refractory periods, and post-synaptic potential shaping. When the membrane potential of neuron i, denoted by v_i, exceeds a threshold θ, a spike or action potential in the network occurs and a brief electrical pulse travels unchanged down the axon of the firing neuron to other receiving neurons. If the spike raises the membrane potential of the receiving neuron above its threshold, it too will generate a spike. A neuron will produce spikes that are all of the same form with an amplitude of about 100 mV and a duration of 1 to 2 ms. Because of the similarities between spikes, the information carried on the signals is dependent on the timing and patterns of the spikes, or spike trains. The spikes can then be represented by the Dirac delta function $\delta(t - t_i^k)$, with a singularity at $t = t_i^k$, where i and k are the neuron and spike indices respectively.

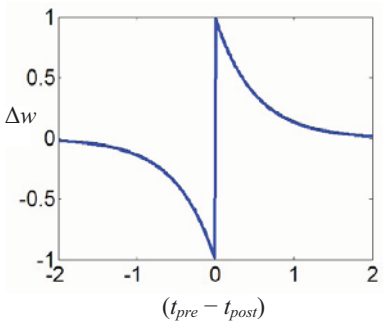

Fig. 1. STDP term as a function of the time delay between the last spike of postsynaptic neuron, and presynaptic neuron.

Early spiking neuron models were vastly simplified and not able to accurately approximate neuron dynamics or reproduce biological learning mechanisms. In this paper, the SNN neurons are modeled according to the Hodgkin-Huxley (HH) formalism with fast sodium and potassium ion channels for the generation of action potentials [2]. HH neurons are a popular choice for SNNs and have been shown to closely represent biological neuron dynamics. The membrane potential, v_m, is governed by

$$c_m \frac{v_m}{\Delta t} = \frac{E_m - V_m}{R_m} - \sum_{c=1}^{N_c} g_c(t)(V_m - E_{rev}^c) + \sum_{s=1}^{N_s} I_s(t) + I_{inject} \quad (1)$$

where the membrane capacity is c_m, the reversal potential of the leak currents is E_m, the membrane resistance is R_m, the number of channels is N_c, the current conductance of channel c is $g_c(t)$, the reversal potential of channel c is E_{rev}^c, the number of incoming synapses is N_s, the current supplied by synapse s is $I_s(t)$, and the injected current is I_{inject} [18]. The membrane potential is found by solving (2), and when v_m exceeds the threshold potential, V_{thresh}, the neuron produces a spike and has v_m reset to a value, v_{reset}, and locked in place during a refractory period, T_{ref}.

In addition to the above deterministic properties, there are also prevalent stochastic qualities in biological neural networks caused by effects such as thermal noise and random variations in neurotransmitters. For this reason, the SNN model includes a zero mean random noise term, I_{noise}, with a variance of 20 nA. I_{noise} is injected into the neuron which causes disturbances in the resting potential. This noise causes a neuron to have a baseline spiking frequency of 8 Hz even when it has no synaptic connections. Biological neurons also display a persistent learning mechanism known as spike-timing-dependent synaptic plasticity, which works to modify the synaptic strengths between neurons. These changes are driven by relationships between adjacently connected neurons where the directions and magnitudes of the changes are dependent on the relative timings of the presynaptic spike arrivals and postsynaptic firings. In the network that undergoes STDP using the method presented, all changes in synaptic strengths occur solely as a result of the STDP mechanism. For each set of neighboring spikes, the weight adjustment is given by

$$\Delta w = Ae^{[(t_{pre} - t_{post})\tau]} \quad (2)$$

As a result, the connection weight increases when the postsynaptic spike follows the presynaptic spike, and the weight decreases when the opposite occurs. The amplitude of the adjustment Δw lessens as the time between the spikes becomes larger, as is illustrated by plotting (2) in Fig. 1.

III. Adaptive Control Architecture and Algorithm

Many approaches have been proposed to train SNNs directly with reward signals or by reward-driven Hebbian

learning and modulation of STDP [10], [14]. However, these methods are not suited to those SNN applications that involve training biological neuronal networks, because biological synaptic efficacies cannot be adjusted directly. This paper presents an approach for controlling a randomly connected action SNN through input signals provided by a critic SNN to modulate biologically-occurring STDP mechanisms, and alter the synaptic efficacies of the action SNN. Let NN_a represent the action SNN of randomly connected SNN of HH neurons, and NN_c denote the critic SNN of feedforward fully-connected HH neurons. In this architecture, schematized in Figure 2, NN_a is treated as a biological network, and NN_c is treated as an artificial network implemented on a computer or integrated circuit. Thus, the synaptic weights of NN_c, defined as w_{ij}, are directly adjustable, while the synaptic efficacies of NN_a can only be modified through a simulated STDP mechanism which is modulated by the input spikes from NN_c. The algorithm presented in this section trains the network by changing the values of w_{ij} inside NN_c, which are assumed to be bounded by a positive constant w_{max} such that $-w_{max} \leq w_{ij} \leq w_{max}, \forall i, j$.

It is assumed that the desired response of NN_a can be specified by a known target function $y = h(x)$, where, in this paper, $y \in \Re^m$ represents the optimal control output for the flight controller, and $x \in \Re^n$ is the target input state. In each SNN, there are n input neurons that correspond to each input state. The NN_a network has m output neurons which relate to the control output variables. As is the case with biological neural networks, SNNs use information coded as spike trains, and therefore the output control signal must be decoded into usable values. Biological neural networks are capable of carrying information based on either firing rates or individual spike timing. In this paper, spike frequencies are used to code continuous signals, and the leaky integrator

$$\hat{y} = \alpha \sum_{t_k \in S_i(T)} e^{\beta(t_k - t)} H(t - t_k) - \gamma \tag{3}$$

is used to decode spike trains and convert them into continuous signals, where, α, β, and γ are user-specified constants, and $H(\cdot)$ is the Heaviside function. Decoding by a leaky integrator is advantageous for use with the flight controller because it is effective in filtering inevitable noise while still allowing the continuous value to change with sufficient speed to match the target function.

The goal of the proposed approach is to induce changes in the synaptic efficacies of NN_a through STDP such that the control output will closely match y for all x. Since the weights within NN_a cannot be adjusted directly, they are manipulated with training signals from NN_c. Connections are made between q pairs of training neurons which serve as outputs in NN_c and inputs for NN_a. To provide feedback signals to the critic, connections are also created between r pairs of neurons from NN_a to NN_c. The information flow through the networks is illustrated in Figure 2. Since it is not known what training signals will produce the desired results in NN_a, the critic must also be trained to match an optimal output signal. However, since the synaptic weights

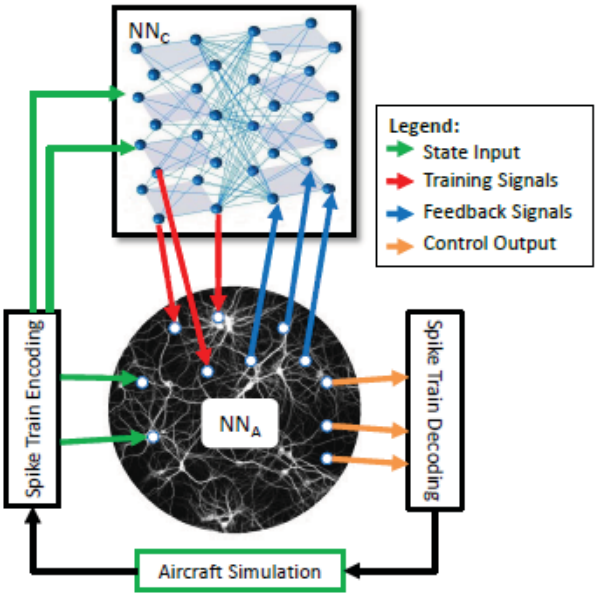

Fig. 2. Adaptive control architecture with critic network NN_c training/controlling action network NN_A to control a simulated aircraft.

of the critic may be adjusted manually, a reward-modulated Hebbian approach can be implemented. Because the target function y is known for all values of x, the controller output error is also known and can be used as a feedback to the critic in the form of an imitated chemical reward, $r(t)$, that decays over time with time constant τ_c. The reward is given by the function

$$r(t) = [b(\hat{y}, y) + r(t - \Delta t)] \cdot e^{-(t - \hat{t}_i)/\tau_c} \tag{4}$$

where $b(\hat{y}, y) = sgn(y - \hat{y})$, which is chosen such that it is positive when the control output is too low and negative when it is too high. The reward is used to determine the incremental weight change according to the rule

$$\Delta w_{ij}(t) = \mu \cdot r(t) \cdot f(\hat{t}_i, \hat{t}_j) \cdot g[w_{ij}(t)] \tag{5}$$

The learning rate, μ, is a constant bounded by $O(w_{max})$.

The STDP term, $f(\hat{t}_i, \hat{t}_j)$, exhibits similar behavior to that of the STDP mechanism acting on the action network, and it is defined as

$$f(\hat{t}_i, \hat{t}_j) = sgn(\hat{t}_i - \hat{t}_j) \cdot e^{-|\hat{t}_i - \hat{t}_j|/\tau_d} \tag{6}$$

The shape of $f(\hat{t}_i, \hat{t}_j)$ is illustrated in Figure 1. When the time delay between neighboring spikes is small, the change of the synaptic strength between the two neurons is great. If the presynaptic neuron is the first to fire, the weight increases, while if the postsynaptic neuron fires first, the weight decreases. The synapses' eligibility, $g[w_{ij}(t)]$, is reproduced by the function,

$$g[w_{ij}(t)] = 1 - c_1 \cdot e^{-c_2 \cdot |w_{ij}(t)|/w_{max}} \tag{7}$$

The eligibility models a phenomenon in biological neural networks where synapses with higher efficacies tend to experience greater changes in weights. The parameters c_1

and c_2 are positive constants.

With this rule, the training of the networks is implemented by discretizing a time period $[0,\ T_{train}]$ by fixed intervals of length Δt, and updating the weights at each timestep such that,

$$w_{ij}(t+\Delta t) = w_{ij}(t) + \Delta w_{ij}(t) \tag{8}$$

The weights are changed starting with the output layer and working backwards through the hidden layers and to the input layer. As the synaptic strengths in the critic are modified, the training signals will evolve such that the effects of STDP will alter the connections in the action network and cause the control output to increasingly match the target function. The learning algorithm is written in pseudocode and is as shown in Algorithm 1.

Algorithm 1 SNN Learning algorithm

```
for t = Δt to T_train do
  for i = N, N − 1, ..., 1 do
    for every pair (j, i) do
      if in action network then
        Δw_ij(t) = A e^{[(t_pre − t_post)τ]}
      else
        Δw_ij(t) = μ · r(t) · f(t̂_i, t̂_j) · g[w_ij(t)]
      end if
      w_ij(t + Δt) = w_ij(t) + Δw_ij(t)
    end for
  end for
  t = t + Δt
end for
```

IV. Aircraft Model and Control

The adaptive controller presented in this paper is trained and tested using a dynamic aircraft model with poor short-period flying qualities taken from [19, p. 369]. For simplicity, the aircraft is assumed to be restricted to longitudinal motion and is described by two state variables, $x = [\alpha\ \ q]^T$, representing the aircraft angle of attack and pitch rate, respectively. The aircraft dynamics are modeled by two linear and coupled differential equations

$$\begin{aligned} \dot{\alpha} &= -0.334\alpha + q - 0.027u \\ \dot{q} &= -2.52\alpha - 0.387q - 2.6u \end{aligned} \tag{9}$$

where u is a scalar control input provided by the elevator deflection. As shown in Figure 3, the above aircraft model suffers from poor short-period flying qualities, as the oscillations in angle-of-attack following an initial disturbance, $\alpha(t = 0) = 5\ deg$, take many seconds to decay [19, p. 369]. In this paper, the action SNN is utilized as a state-feedback controller that provides stability augmentation such that the aircraft will return to steady-level flight with an adequate short-period natural frequency and damping ratio. Using a classical pole-placement approach and the desired short-period flying qualities criteria, as shown in [19, p. 369], the optimal control law,

$$u^* = 2.03\alpha + 1.318q \tag{10}$$

can be determined from (9). For each input state value chosen from a discretized range of pitch rates and angles of attack, the optimal control value, u^*, is computed from (10) and used as a target in the SNN training algorithm presented in Section III, as explained in the following section.

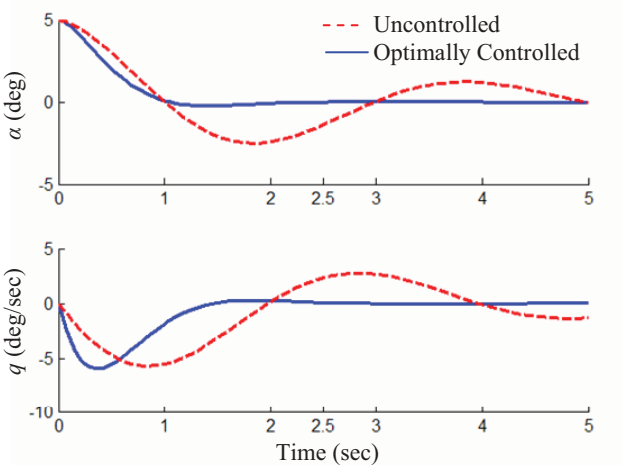

Fig. 3. Time history of aircraft angle of attack, α, and pitch rate q, with and without a controller.

V. Training Simulations and Results

The training architecture and algorithm presented in Section III are implemented to train an action SNN to perform adaptive control of aircraft. The objective is to demonstrate that a so-called action SNN can be trained to solve a benchmark control problem by inducing STDP via a critic SNN, without directly manipulating the SNN synaptic strengths. As a result, the methods presented in this paper could some day be applied to train biological networks via computational or *in-silico* critic SNNs. The simulations are conducted in MATLAB®, using the neural Circuit SIMulator (CSIM) software [18], which utilizes an adaptive time-step Crank-Nicolson integrator to simulate HH SNNs. The aircraft is assumed to operate only under longitudinal flight conditions, and NN_a is used to compute the elevator deflection based on perfect knowledge of the aircraft state x. The SNN controller's effectiveness is tested by giving the system an initial disturbance in the aircraft's angle of attack, and by indirectly training NN_a through NN_c such that the aircraft will return to steady-level flight.

The SNN architecture portrayed in Fig. 2 is implemented in CSIM as follows. The action SNN, denoted by NN_a, is constructed using 50 randomly connected neurons with 2 state input neurons, 16 training input neurons, 6 feedback output neurons, and 1 control output neuron. The critic SNN, denoted by NN_c, consists of 2 state input neurons, 6 feedback input neurons, 2 layers of 20 hidden neurons each, and 16 training output neurons. This architecture can be scaled larger and produce similar results, but if the size is reduced, the performance will suffer. Each neuron is modeled according to the HH dynamics described in Section II. During the implementation of the controller, the signal flow

begins at the flight simulator where the aircraft state values are converted into spike trains and passed to the state input neurons. The critic network sends training signals in the form of spike trains to the action network via the training neurons. The action network returns feedback signals as spikes to the critic network through the feedback neurons, and it outputs a spike train representing the control output.

During the training period, the input state, x, passed to the networks is chosen at random from a list of possible values distributed evenly within a practical range. In this paper, the list used is comprised of $N_{epoch} = 64$ state values, and a new state is selected for each new epoch of $t_{train} = 1$ second until every combination of states has been used. During each epoch, the state input is converted into a spike train that is generated using a Poisson distribution,

$$P(n,t) = e^{-\nu t}\frac{(\nu t)^n}{n!}, \quad n = 0, 1, ... \tag{11}$$

The time interval is the length of the epoch, t_{train}, and the Poisson rate, ν, and the number of spikes, n, are set accordingly for each average spike frequency. The state values are mapped to frequencies with a simple linear relationship where a slower spike rate corresponds to a lower state value and a faster rate communicates a higher state value. The ranges used for the state values and frequencies are $[-7,\ 7]$ for both q and α, and 10 Hz to 100 Hz respectively. The reverse conversion is done when decoding the control signal from the control output spike train. However instead of using a simple linear function, a leaky integrator is utilized which maps a baseline frequency (8 Hz) to -30 and a high frequency (150 Hz) to 30, where the relationship between spike frequencies and the resulting control value is linear between this range. The leaky integrator allows the control value to change smoothly and less erratically than with simple rate coding, where the control value is proportional to the average spike frequency over a preceding user-defined time interval. A sample of the spike train structures is illustrated in Figure 4, where the spikes are shown from the action network's state input, hidden, and control output neurons over a time interval of 1 second.

The two SNNs are trained by implementing the training algorithm described in Section III until the performance of NN_a stops improving significantly or until it is within a desired tolerance. In this paper, the networks have been run through the randomized list of possible input states, which has been varied between $[-7,\ 7]$ for both q and α a total of $h = 30$ times. The total simulated time of training for all runs is $T_{train} = t_{train}N_{epoch}h(\cdot)$. Figure 5 shows the cumulative absolute error between u and u^* over each training run, and it can be seen that it decreases significantly over training. The error continues to decrease if trained for a longer period, thus h can be adjusted depending on the objectives and requirements of the networks.

After training, the SNN controller is tested against the behavior of the aircraft dynamics shown in Section IV when subjected to an initial disturbance of $\alpha(t = 0) = 5$ degrees. The network is not yet able to match the optimal

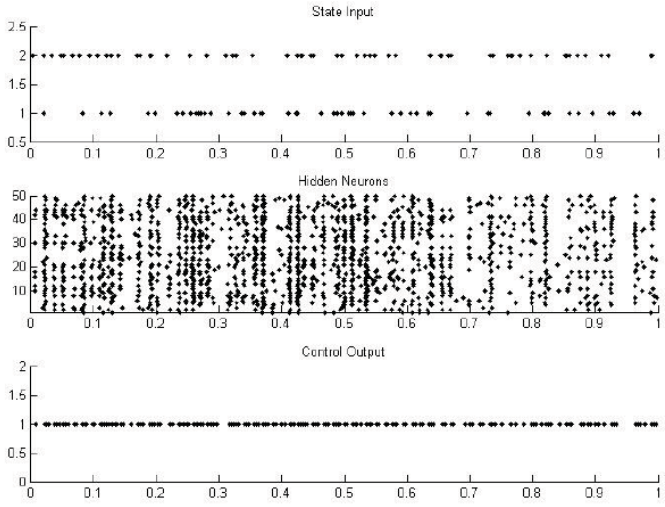

Fig. 4. Spike train sample from the state input, hidden, and control output neurons of the action network over a 1 second time interval.

control output from (10), but it is improved significantly with training. A comparison between the dynamic responses of a trained and untrained network are plotted in Figure 6, and it can be seen that the training lowers the amplitudes of the aircraft pitch oscillations and increases the stability compared to the untrained network. With this configuration, the controller is prevented from improving beyond this level of effectiveness because of the high degree of noise in the networks. This is especially an issue with the randomly-connected action network because it has many recurrent connections. However even with this difficulty, it is observed that the action network is controllable through the critic network. The controller is also determined to not exhibit highly chaotic behavior, as is shown in the Poincaré map in Figure 7. Although the network noise causes the trajectories to behave randomly, the state is seen to approach and remain near an attractor regardless of its initial values.

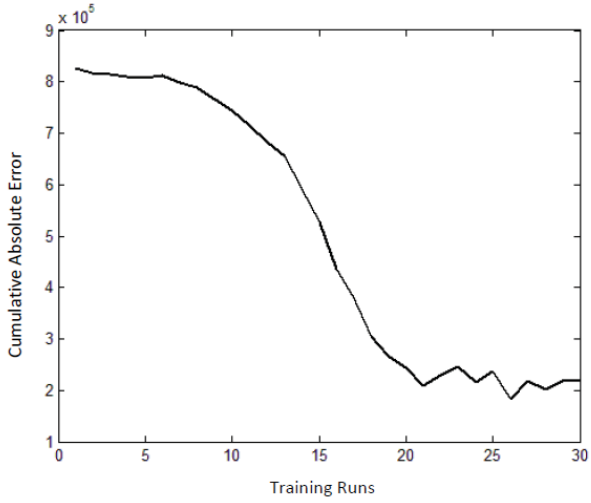

Fig. 5. Cumulative absolute error over each training run

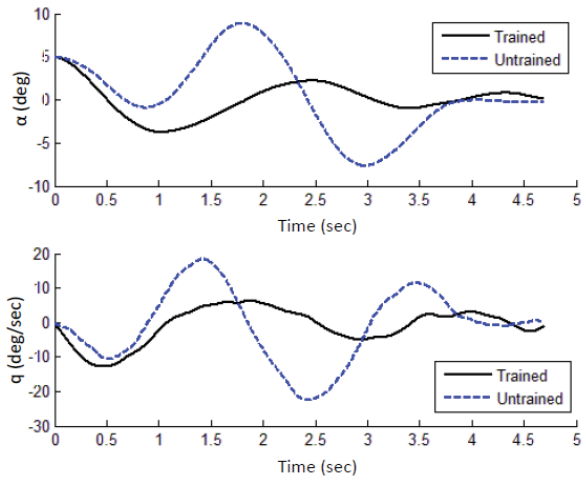

Fig. 6. Comparison of aircraft stability altered by a trained and untrained SNN controller with previously described architecture.

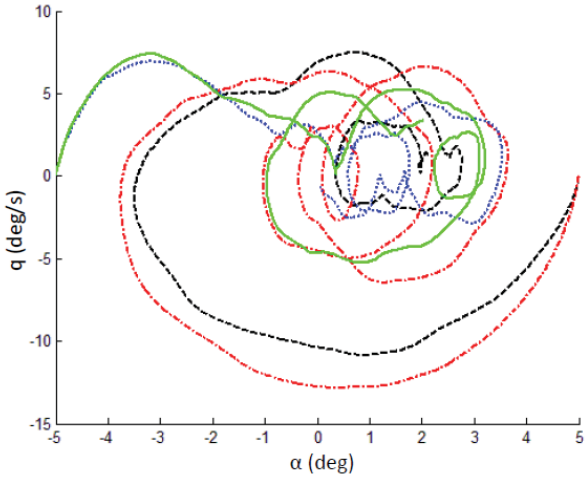

Fig. 7. Poincaré map of aircraft dynamics modified by SNN controller. The plot shows four separate runs with initial conditions, $X = [5,\ 0] and [-5,\ 0]$

To fully understand the effect of the randomly-connected action network on the controller behavior, it is useful to compare it to a similar controller where the action network is a fully-connected feedforward SNN, which has previously been shown capable of approximating functions well [10]. Figure 8 illustrates the significant difference between the two networks, where the feedforward SNN (with two layers of 20 hidden neurons) produces less noise and a more accurate output. This implies that the architecture in Figure 2 may be improved further. Future work will aim to achieve these improvements by optimizing parameters, reexamining the spike train encoding and decoding methods, and investigating the effects of small modifications to the controller architecture.

VI. Conclusion

A novel SNN training algorithm for indirectly teaching a randomly connected network to approximate an optimal flight controller is presented in this paper. The technique

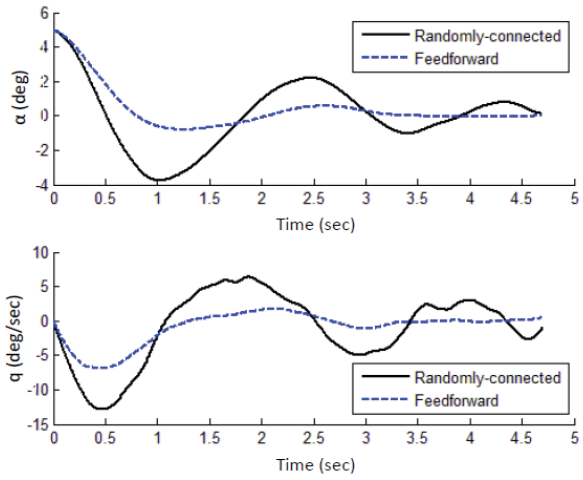

Fig. 8. Comparison of aircraft stability controlled by a SNN architecture with a randomly-connected action network, and a configuration with a fully-connected feedforward action network.

is both biologically-plausible and experimentally viable. A reward-based Hebbian approach is used to directly train a critic SNN, which in turn sends training signals to an action SNN. The reward function imitates a chemical reward that is injected into the network and decays exponentially, and this mechanism can also be implemented *in vitro* by using dopaminergic, adrenergic, and cholinergic signaling pathways. With guidance from the critic, the synaptic efficacies within the action network are modified by implicit spike-timing-dependent plasticity mechanisms, resulting in network training. By employing light-sensitive neurons and controlled light patterns, the effects of the training signals can be applied to biological neuronal networks *in vitro*. The algorithm is demonstrated by training the network to mimic a benchmark flight controller problem, and analysis is provided regarding its dynamic behavior and possible improvements. It has been determined that the action SNN does not exhibit chaotic behavior and can be trained through a critic SNN, but the effectiveness of the trained network as an adaptive flight controller can be improved.

Acknowledgments

This material is based upon work supported by the National Science Foundation under Grant No. 0925407.

References

[1] J. J. B. Jack, D. Nobel, and R. Tsien, *Electric Current Flow in Excitable Cells, 1st ed.* Oxford, UK: Oxford University Press, 1975.
[2] A. L. Hodgkin and A. F. Huxley, "A quantitative description of ion currents and its applications to conductance and excitation in nerve membranes," *Journal of Physiology*, vol. 117, 1952.
[3] W. Maass, "Noisy spiking neurons with temporal coding have more computational power than sigmoidal neurons," *Advances in Neural Information Processing Systems*, vol. 9, 1997.
[4] D. I. Perrett, E. T. Rolls, and W. C. Caan, "Visual neurons responsive to faces in the monkey temporal cortex," *Experimental Brain Research*, vol. 47, 1982.
[5] S. J. Thorpe and M. Imbert, "Biological constraints on connectionist modelling," in *Connectionism in Perspective*, R. Pfeifer, Z. Schreter, F. Fogelman-Soulie, and L. Steels, Eds. Elsevier, 1989.

[6] G. S. Hugh, M. Laubach, M. A. L. Nicolelis, and C. S. Henriquez, "A simulator for the analysis of neuronal ensemble activity: Application to reaching tasks," *Neurocomputing*, vol. 44, 2002.
[7] E. Salinas and T. J. Sejnowski, "Correlated neuronal activity and the flow of neural information," *Nature Reviews - Neuroscience*, vol. 2, 2001.
[8] Z. F. Mainen and T. J. Sejnowski, "Reliability of spike timing in neocortical neurons," *Science*, vol. 268, 1995.
[9] H. Burgsteiner, "Imitation learning with spiking neural networks and real-world devices," *Engineering Applications of Artificial Intelligence*, vol. 19, no. 7, 2006.
[10] S. Ferrari, B. Mehta, G. D. Muro, A. M. VanDongen, and C. Henriquez, "Biologically realizable reward-modulated hebbian training for spiking neural networks," *Proc. International Joint Conference on Neural Networks, Hong Kong*, pp. 1781–1787, 2008.
[11] C. M. A. Pennartz, "Reinforcement learning by hebbian synapses with adaptive thresholds," *Neuroscience*, vol. 81, no. 2, 1997.
[12] R. Legenstein, C. Naeger, and W. Maass, "What can a neuron learn with spike-timing-dependent plasticity?" *Neural Computation*, vol. 17, 2005.
[13] J. P. Pfister, T. Toyoizumi, D. Barber, and W. Gerstner, "Optimal spike-timing-dependent plasticity for precise action potential firing in supervised learning," *Neural Computation*, vol. 18, 2006.
[14] R. V. Florian, "Reinforcement learning through modulation of spike-timing-dependent synaptic plasticity," *Neural Computation*, vol. 19, no. 6, 2007.
[15] S. G. Wysoski, L. Benuskova, and N. Kasabov, "Adaptive learning procedure for a network of spiking neurons and visual pattern recognition," *Advanced Concepts for Intelligent Vision Systems, ACIVS, Antwerp, Lecture Notes in Computer Science*, vol. 4179, 2006.
[16] A. M. VanDongen, "Vandongen laboratory," in *http://www.vandongen-lab.com/*.
[17] T. L. Fletcher, P. Cameron, P. D. Camilli, and G. Banker, "The distribution of synapsin i and synaptophysin in hippocampal neurons developing in culture," *Journal of Neuroscience*, vol. 11, pp. 1617–1626, 1991.
[18] "Neural microcircuits (nmc) software for matlab environment," *http://www.lsm.tugraz.at/index.html*.
[19] R. C. Nelson, *Flight Stability and Automatic Control*. McGraw-Hill, 1998.