

# INPUT-OUTPUT MAPPING PERFORMANCE OF LINEAR AND NONLINEAR MODELS FOR ESTIMATING HAND TRAJECTORIES FROM CORTICAL NEURONAL FIRING PATTERNS

Justin C. Sanchez<sup>1</sup>, Sung-Phil Kim<sup>2</sup>, Deniz Erdogmus<sup>2</sup>,  
Yadunandana N. Rao<sup>2</sup>, Jose C. Principe<sup>2</sup>, Johan Wessberg<sup>3</sup>, Miguel Nicolelis<sup>3</sup>  
Departments of <sup>1</sup>Biomedical and <sup>2</sup>Electrical and Computer Engineering  
University of Florida, Gainesville, FL 32611  
[justin, phil, deniz, yadu, principe]@cnel.ufl.edu  
<sup>3</sup>Department of Neurobiology, Duke University, Durham, NC 27710  
[wessberg, nicoleli]@neuro.duke.edu

**Abstract.** Linear and nonlinear (TDNN) models have been shown to estimate hand position using populations of action potentials collected in the pre-motor and motor cortical areas of a primate's brain. One of the applications of this discovery is to restore movement in patients suffering from paralysis. For real-time implementation of this technology, reliable and accurate signal processing models that produce small error variance in the estimated positions are required. In this paper, we compare the mapping performance of the FIR filter, Gamma filter and recurrent neural network (RNN) in the peaks of reaching movements. Each approach has strengths and weaknesses that are compared experimentally. The RNN approach shows very accurate peak position estimations with small error variance.

## INTRODUCTION

Brain-machine interfaces are a developing technology, which might provide a new medium to transfer the intent of an individual to a variety of devices for the purpose of locomotion, enhanced ability, or computer control. Using new multi-electrode recording techniques, signals derived from populations of single neurons could be used to control devices that substitute the movement for individuals with neurologic disorders. Both linear and nonlinear filters [1] have served as tools for studying brain derived signals to estimate hand position in 3-D space.

It still remains unknown which topology, linear or nonlinear, produces estimations of hand trajectories that will be feasible for real-time implementation of a brain-machine interface. The potential advantage of a nonlinear filter as opposed to a linear filter is that it might be able to find a more complex input-output (I/O) mapping that captures the details in the trajectories [2]. We ultimately seek a model that produces high target accuracy with a small variance. The model must also produce the output with a level of dexterity similar to the biologic hand.

The performance of linear and nonlinear (TDNN) models have been compared by Nicolelis and colleagues [1] who demonstrated that firing patterns from ensembles of cortical neurons could successfully predict (in real time) the hand position of a primate. In the prediction procedure, a large array of 100+ microelectrodes was implanted in the pre-motor and motor areas of a primate. Electrode outputs were processed by spike detection and sorting algorithms to determine firings of single neurons. Spike counts in 100 msec windows are then fed to either a 104x10 (104 channels and 10 delays) finite impulse response filter (FIR) trained with least squares (effectively a Wiener filter [3]) or a 1040x15x3 time delay neural network (TDNN) (104 channels with 10 delays each) trained with conjugate gradient to match the x, y, z coordinates of the primate's hand.

Other groups have also demonstrated neural control of devices using linear and nonlinear methods. Neural cursor control using linear filters trained with least-squares has also been attempted by Donoghue et. al. [4]. Chapin and colleagues utilized a recurrent neural network to predict lever pressing from ensembles of rat cortical neurons [5].

In this paper, we aim to specifically determine which model (linear or nonlinear) is best for producing accurate estimations during long excursions in hand trajectory and stationary positions. Secondly, we will evaluate how well the fixed linear and nonlinear models perform over time. We continue the work of Nicolelis and colleagues by substituting the TDNN with a recurrent neural network (RNN) and compare the performance to the FIR and Gamma filters. The choice of the RNN is based upon the fact that it is a nonlinear mapper which requires much fewer parameters than the linear and TDNN models.

## MODELING

*FIR Filters:* In this modeling approach, we assume that there exists a linear mapping between the desired hand position and neuronal firing counts. The estimated hand position output of the linear filter with  $M$  delays (20 for this problem) is given in (1). This equation shows that the delayed versions of firing counts,  $\mathbf{x}(t-i)$ , from 104 neurons are the bases that construct the output signal.

$$y(t) = \sum_{i=0}^{M-1} \mathbf{w}_i^T \mathbf{x}(t-i) \quad (1)$$

The model parameters are updated using the computationally efficient LMS algorithm [6] which utilizes stochastic gradient descent. The linear FIR filter using MSE as the criterion is guaranteed to converge to a single global minimum [3]. Given the input vector  $\mathbf{x}(t)$  and the desired response  $d(t)$ , the weight parameters of the  $M$ -tap FIR filter shown in Fig. 1 are adapted using LMS as given in (2). Here,  $\eta$  is the constant learning rate and  $e(t) = d(t) - \sum_i \mathbf{w}_i^T(t) \mathbf{x}(t-i)$ . Since the adaptation only includes two multiplications and one addition per weight, the computational complexity of LMS is  $O(N)$ , where  $N$  is the number of weights of the FIR filter. Thus, the computational complexity of LMS increases linearly with filter order.

$$\mathbf{w}_i(t+1) = \mathbf{w}_i(t) + \eta e(t) \mathbf{x}(t-i) \quad (2)$$

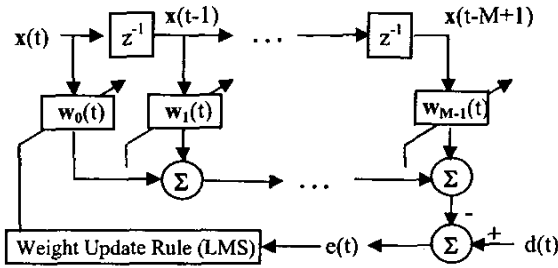


Figure 1: Structure of the FIR filter adapted by the LMS algorithm.

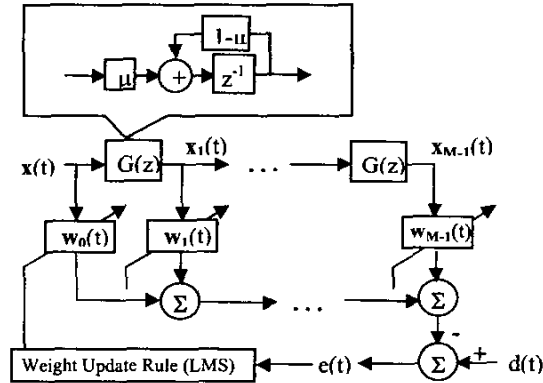


Figure 2: Structure of the Gamma filter adapted by the LMS algorithm.

Since the signal power of the neuronal firing counts can rapidly change, the normalized LMS algorithm can be used to avoid sensitivity to local amplitude and to set a time-varying learning rate that traces time-variant input statistics. The weight update rule for normalized LMS [7] is given in (3) where  $\eta_0$  satisfies  $0 < \eta_0 < 2$ , and  $\gamma$  is a small positive constant.

$$w_i(t+1) = w_i(t) + \frac{\eta_0}{\gamma + \|x(t-i)\|^2} e(t)x(t-i) \quad (3)$$

*Gamma Filters:* The Gamma filter incorporates the desirable features of both FIR and IIR filters by implementing a restricted feedback architecture which uncouples memory depth from filter order [8]. The Gamma filter structure, shown in Fig. 2., allows for a reduction in number of free parameters for an equivalent FIR memory depth. The memory depth of the  $M$ -tap Gamma filter is given by (4) where  $\mu$  is a feedback parameter, which is equivalent to the time resolution  $R$  of the Gamma filter. Hence, there exists a trade-off between memory depth and resolution. The Gamma filter reduces to the adaptive FIR structure when  $\mu=1$ .

$$D_M = M/\mu \quad (4)$$

This memory characteristic is especially useful for this problem since input data has a large dimensionality ( $x(t)$  is a  $104 \times 1$  vector). A 20-tap FIR architecture incorporates 2080 parameters ( $20 \times 104$ ). For simple trajectories, this large number of parameters can add noise to the output and cause poor performance. An equivalent memory depth can be achieved with a 12-tap Gamma filter with  $\mu=0.6$ . This yields a 40% saving in the number of weights from 2080  $\rightarrow$  1248.

Filter weights can be updated in the same manner as the FIR filter given in (2) with the exception of the feedback parameter  $\mu$  of (5) that will not be updated. Since the Gamma filter employs LMS as the adaptation rule, the computational complexity remains as  $O(N)$ .

$$x_k(t) = (1 - \mu)x_k(t-1) + \mu\alpha_{k-1}(t-1), \quad k=1, \dots, M-1 \quad (5)$$

*Recurrent Neural Network:* In this I/O modeling approach, we create a mapping which assumes a nonlinear relationship between the spike trains and hand positions. Here the model is a recurrent multilayer perceptron (MLP) that was proposed in [9]. This network differs from an MLP since it contains feedback connections in its hidden layer. The architecture consists of an input layer with 104 channels, a hidden layer of nonlinear processing elements (PEs), (in this case tanh), and an output layer of linear PEs. Fig. 3 depicts the topology of the recurrent network (RNN) that is used in this study. Each hidden layer PE is connected to every other hidden PE using a unit time delay. We can see in (6) that the state produced at the output of the first hidden layer is a nonlinear function of a weighted combination of the current input and the previous state. The feedback of the state allows for continuous representations on multiple timescales. While both the linear and Gamma (when trained with fixed  $\mu$ ) networks have a fixed memory depth, the recurrent network does not have a constraint on the memory depth and implements an infinite memory by the recurrency in (6). The output layer is a simple linear combination, shown in (7), of the hidden layer states.

$$y^1(t) = f(W^1 x(t) + W^f y^1(t-1)) \quad (6)$$

$$y(t) = W^2 y^1(t) + B \quad (7)$$

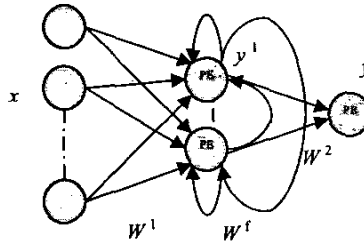


Figure 3. State Recurrent Fully Connected Neural Network

Unlike the linear and Gamma models there is no need to use memory at the input layer. This reduces the number of free parameters in the input layer dramatically (from 2,080 to 104). Memory is created by feeding the states of the hidden PEs among themselves. Each of the hidden PEs outputs can be thought of

as a nonlinear adaptive basis of the input space utilized to project the large dimensionality data. These projections are then linearly combined to form the outputs of the RNN that will predict the desired hand movements.

The MLP from which this topology is derived has been shown to be a universal mapper in  $\mathbb{R}^n$  [10]. The time delay neural network (TDNN) has been also shown to be a universal mapper in myopic functional spaces [11]. Although no theoretical work to prove the universal approximation of the recurrent MLP is known, we expect it to display the same universality because it can be unfolded in a TDNN [12]. Hence this network when properly dimensioned and trained has the power to find the mapping between spike trains and hand positions in 3D space.

Although no analytic solution to solve the nonlinear equation of the RNN is known, adapting its parameters can be achieved using the gradient descent procedure. Since RNNs are recurrent systems, gradients display dependencies over time, and so the common backpropagation algorithm to train neural networks [13] cannot be applied directly. Here we use the Backpropagation Through Time (BPTT) algorithm [14] to train the RNN. In BPTT, the recurrent network is unfolded to create an equivalent feedforward network, with replicated weights, which span a time trajectory. The length of this trajectory is empirically determined. Input data spanning the trajectory is fed to the feedforward network with a random initial state and the PE outputs are stored. An error vector is created at the output and fed (reversed in time) through the dual network to produce local errors. The weights are then updated. Finally, the process begins again for the next trajectory. We used the commercial tool NeuroSolutions to train the RNN [12].

Sufficiently small stepsizes were chosen to promote generalization and ensure stability. Even with small stepsizes, feedback loops in the network can cause outputs to oscillate. Since previous states are used in future state computations, oscillations propagate and cause poor generalization. The network learning must be reinitialized in this situation. Recurrent networks trained with gradient descent methods have difficulty learning time dependencies in long trajectories. Gradients tend to decay exponentially through the trajectory due to the PE nonlinearities [15]. Training RNNs with BPTT also suffers from a high computational complexity. Activations, injected errors, and copies of weights have to be stored over a trajectory length. For one trajectory  $T$  steps long with an  $N$  node network, BPTT requires  $O(N^2T)$  computations and  $O(NT)$  storage [12].

## SIMULATIONS

*Data:* Synchronous, multichannel neuronal spike trains were collected at Duke University using owl monkeys (*Aotus trivirgatus*). Microwire electrodes were implanted in cortical regions with known motor associations [16]. The firing times of single neurons were recorded while the monkey performed a 3-D reaching task. The monkey hand position was also recorded (with a shared time clock) and digitized with a 200Hz sampling rate. The neuronal firings were binned (added) in non-overlapping windows of 100ms, which represents the local firing rate for a neuron. These spike counts were directly used as inputs to the linear, Gamma and

RNN. In order to take the reaction time into account, the spike trains were delayed by 0.23 seconds with respect to the hand position.

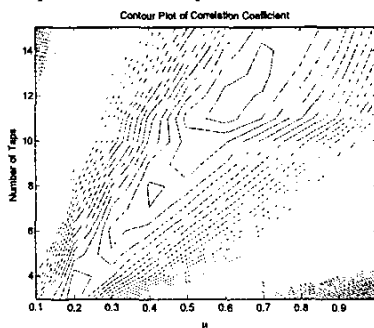


Figure 4. Testing correlation coefficient as a function of the number of filter delays and the feedback parameter  $\mu$ .

*Linear/Gamma:* The spike counts of each of the 104 neurons were used to train both the FIR and Gamma filters to predict the x, y, and z coordinates of the monkey's hand. The FIR topology contained a twenty tap-delay line (2080 weights –  $20 \times 104$ ) while the Gamma implemented an eight tap-delay line with the feedback parameter,  $\mu$ , fixed to 0.6 (1248 weights –  $12 \times 104$ ). The effective memory depth is equivalent for both filters and the optimal parameters were chosen by conducting Monte Carlo simulations over a number of tap delays and the feedback parameters. Results shown in Fig. 4 were obtained by computing the correlation coefficient between the actual and estimated test trajectory. The correlation coefficient gives a measure of how well the actual and estimated trajectories are linearly related. The optimal delay length and  $\mu$ , were the pair that produced the highest correlation coefficient.

The weights of both filters were updated using the LMS rule after the presentation of each exemplar. A training set of 20,000 consecutive bins (2,000 secs) of data was utilized. Forty presentations of the data were required to train the filter weights. Linear filters display a slow rate of convergence when the input data is highly correlated. Neuronal firing patterns collected from electrode arrays are highly correlated in clusters local to the electrodes. Large eigen value spreads in the data cause the performance surface to assume the shape of a valley to give multiple solutions with the same MSE.

In testing, the network parameters were fixed and 3,000 consecutive bins (300 secs) of novel neuronal data were fed in to the filter to predict new hand trajectories. The testing results were evaluated in terms of signal to noise (error) ratio (SNR) between the actual and estimated hand trajectories using a sliding window of 40 samples (4 seconds). The window length of 40 was selected because each movement spans about 4 seconds. The SNR (square of the desired signal divided by the square of the estimation error) gives a measure of the accuracy of estimated position in terms of the error variance. High SNRs are desired since they are produced when the estimated output error variance is small.

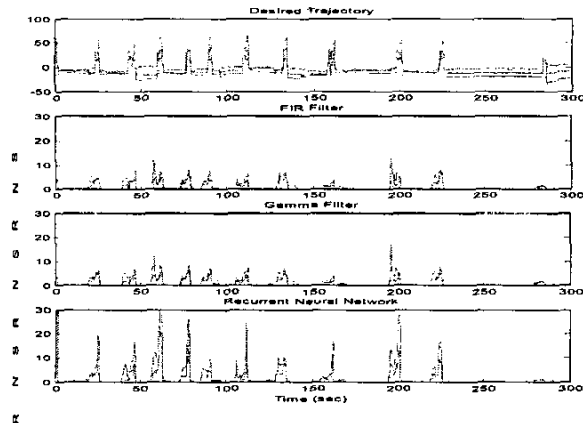


Figure 5. Signal to noise ratio between the actual and estimated hand coordinates.

The  $x$ ,  $y$ , and  $z$  coordinates of hand trajectories during testing are presented in the top subplot of Fig. 5. Also shown in subplots (2) and (3) are the SNRs for the FIR and Gamma filters. Windowed SNRs, averaged over the three coordinate directions, reached values of 7.51 and 8.53 for the FIR and Gamma, respectively. Cumulative SNRs averaged over all coordinate directions and the entire test set (3000 samples, 300 seconds) were 0.86 for the FIR and 0.89 for the Gamma. No decay in the SNR was observed over the entire testing set.

The peaks of the estimated hand trajectory (Gamma filter) superimposed on the actual trajectory are shown in Fig. 6, subplot (1). Three of the six peaks are not captured by the model. Two of the six peaks show an estimated position that reaches the target value and falls off sharply. Only one peak shows accurate estimation by the Gamma filter.

*RNN*: The spike counts of each of the 104 neurons were used to train a RNN (104x5x3) with 5 nonlinear hidden PEs and 3 linear output PEs to predict the  $x$ ,  $y$ , and  $z$  coordinates of the monkey's hand. Each exemplar of data used for the BPTT training algorithm contained a trajectory length of thirty samples (3 secs). Weight updates occurred after the presentation of 10 exemplars (30 secs). A training set of 20,010 consecutive bins (2,001 secs) of data was utilized. In testing, the network parameters were fixed and 3,000 consecutive bins (300 secs) of novel neuronal data were fed in the network to predict new hand trajectories.

The SNR computed using a sliding window of 40 samples (4 seconds) for the RNN is shown in Fig. 5, subplot (4). Windowed SNR calculations, averaged over the three coordinate directions, reached a value of 31.98. This value is significantly larger than both the FIR and Gamma models. The cumulative SNR averaged over all coordinate directions and the entire test set (3000 samples, 300 seconds) was 1.58. The RNN produced an estimated trajectory with a cumulative noise power that is fifty percent smaller than the desired signal. In contrast, both the FIR and Gamma produced outputs with noise power that was fifteen percent larger than the desired signal. No time-dependent decay of the SNR was observed.

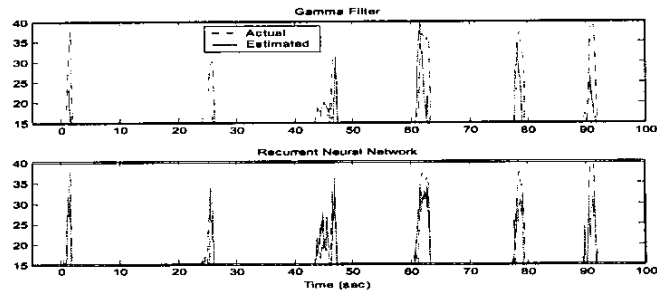


Figure 6. Peaks of Hand Trajectory (Z-Coordinate)

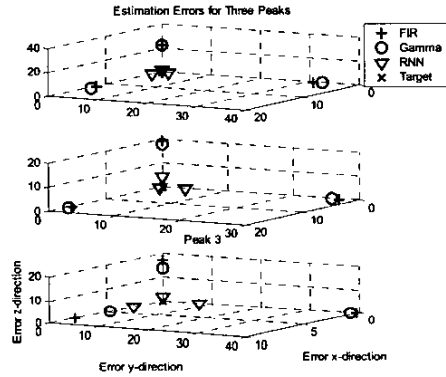


Figure 7. Estimation errors for three peak values. Targets are represented by an x at the origins. The error (mm) in each direction is displayed on the respective axis.

The peaks of the estimated hand trajectory superimposed on the actual trajectory are shown in Fig. 6, subplot (2). Five of the six peaks were captured by the model. None of the estimated peaks display any sharp drop-offs and four of the six are within ten percent error bar around target value. RNN estimation of the peak values is significantly better than the FIR and Gamma filters.

The target accuracy of the RNN is further compared in Fig. 7 that shows the target estimation errors for three peak values. In the figure, the target hand position is represented by an x located at the origin of the coordinate system. The absolute value of the error associated with each direction (x, y, z) is plotted on its respective axis. The RNN errors form a tight cluster of points around the target value, while both the FIR and Gamma filters have large errors in all three directions.

## DISCUSSION AND CONCLUSIONS

In this paper, real neurophysiological data from neuronal spike populations was used to compare linear and nonlinear I/O models to predict 3-D hand positions. The three models utilized differ in the following ways:

- The memory depth of the FIR is limited by the number of delays in the tap-delay line. Longer memory depths result in more free parameters to train. Although the FIR has a single minimum, the convergence speed is affected by a large eigenvalue spread. Training requires a low computational complexity.
- The Gamma filter decouples memory depth from filter order by implementing a restricted feedback architecture. Equivalent memory depths can be achieved with a fewer number of free parameters. Gamma filter training is as computationally efficient as the FIR filter.
- The RNN implements an infinite memory depth on multiple timescales. Memory is moved from the input layer to the hidden layer where the largest decrease in the number of free parameters can be achieved. The RNN architecture used in this paper contained the fewest number of free parameters. RNN is capable of constructing complex nonlinear I/O mappings. BPTT is more computationally complex than standard LMS.

Our results show that during testing, the RNN provides the highest signal to noise ratio for the hand movement in the reaching part of the trajectories. Windowed SNRs were up to four times larger than both the FIR and Gamma filters during the reaching movements. Even with a reduced number of free parameters, the Gamma filter performed only slightly better than the FIR but produced a smaller SNR than the RNN.

We conclude from the SNR plots in Fig. 5 that the part of the trajectory that is more difficult to model (largest error variance) for all three models is when the monkey's hand is at rest close to the body. SNRs in these rest positions drops to zero. We also conclude that there is no noticeable progressive degradation in time of all three models for the period of observation (5 min). This important result shows that models with fixed parameters are capable of producing accurate position estimates over a substantial period of time. Note that the task is rather repetitive and test sets were short.

Accurate target estimation is crucial for the success of this technology. If targets cannot be acquired with fine precision, real-time implementation of a reaching task may not be feasible. The RNN peak estimation showed superior qualities compared to the FIR and Gamma filters. The RNN repeatedly produced peak estimations with errors that formed tight clusters around the target, while the FIR and Gamma had errors with a large spread. The ability to capture the peak values can be attributed to the complex functions produced by nonlinearities in the network. Unlike the linear filters, which occasionally captured a peak, the RNN also maintained the peak value.

The training of the two systems should also be compared. The FIR and Gamma use a very straightforward and well-established training procedure (LMS). We may think that this simple training procedure is an advantage, but the eigenvalue spread of the data causes long convergence times to multiple solutions at a single minima. The RNN has fewer parameters that need to be tuned to the data, but the computational complexity of BPTT is higher. In addition, the trajectory length has to be tuned by hand because it has to include at least one full cycle of the hand movement. The learning rates also have a critical role in the accuracy of the final

model. Improvements in the training of the RNN are in order. For real-time, portable implementation (DSP/FPGA) the RNN poses challenging problems.

We (and other groups) have shown that rather simple procedures are capable of capturing the I/O mappings required to predict hand position from neural spike train populations. Although a lot of work is still required to create reliable and accurate signal processing models, this study can guide us in choosing model classes which provide superior performance.

**Acknowledgements:** This work was supported by a seed grant from the College of Engineering, University of Florida and DARPA.

## REFERENCES

- [1] J. Wessberg, C. R. Stambaugh, J. D. Kralik, P. D. Beck, M. Laubach, J. K. Chapin, J. Kim, S. J. Biggs, M. A. Srinivasan, and a. Nicolelis et, "Real-time prediction of hand trajectory by ensembles of cortical neurons in primates," *Nature*, vol. 408, pp. 361-365, 2000.
- [2] S. Haykin, *Neural networks: a comprehensive foundation*. New York: Toronto: Macmillan; Maxwell Macmillan Canada, 1994.
- [3] S. S. Haykin, *Adaptive filter theory*, 3rd ed. Upper Saddle River, NJ: Prentice-Hall International, 1996.
- [4] M. D. Serruya, N. G. Hatsopoulos, L. Paninski, M. R. Fellows, and J. P. Donoghue, "Brain-machine interface: Instant neural control of a movement signal," *Nature*, vol. 416, pp. 141-142, 2002.
- [5] J. K. Chapin, K. A. Moxon, R. S. Markowitz, and M. A. Nicolelis, "Real-time control of a robot arm using simultaneously recorded neurons in the motor cortex," *Nature Neuroscience*, vol. 2, pp. 664-670, 1999.
- [6] B. Widrow, "Adaptive noise cancelling: principles and applications," *Proceedings of the IEEE*, vol. 63, pp. 1692-1716, 1975.
- [7] J. I. Nagumo and A. Noda, "A learning method for system identification," *IEEE Transactions on Automation and Control*, vol. AC-12, pp. 282-287, 1967.
- [8] B. De Vries and J. C. Principe, "The gamma model: a new neural network model for temporal processing," *Neural Networks*, vol. 5, pp. 565-576, 1993.
- [9] G. V. Puskorius, L. A. Feldkamp, L. I. Davis, and Jr., "Dynamic neural network methods applied to on-vehicle idle speed control," *Proceedings of the IEEE*, vol. 84, pp. 1407-1420, 1996.
- [10] G. Cybenko, "Approximation by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, pp. 303-314, 1989.
- [11] I. W. Sandberg and L. Xu, "Uniform approximation of multidimensional myopic maps," *IEEE Transactions on Circuits and Systems*, vol. 44, pp. 477-485, 1997.
- [12] J. C. Principe, N. R. Euliano, and W. C. Lefebvre, *Neural and adaptive systems: fundamentals through simulations*. New York: Wiley, 2000.
- [13] D. Rumelhart, G. Hinton, and R. Williams, "Learning Internal Representations by Error Back-propagation," *Nature*, vol. 323, pp. 533-536, 1986.
- [14] P. J. Werbos, "Backpropagation Through Time: What It Does and How to Do it," *Proceedings of the IEEE*, vol. 78, pp. 1550-1560, 1990.
- [15] J. F. Kolen and S. C. Kremer, *A field guide to dynamical recurrent networks*. New York: IEEE Press, 2001.
- [16] M. A. Nicolelis, A. A. Ghazanfar, B. M. Faggin, S. Votaw, and L. M. Oliveira, "Reconstructing the engram: simultaneous, multisite, many single neuron recordings," *Neuron*, vol. 18, pp. 529-537, 1997.