DETERMINING PROPERTIES OF SYNAPTIC STRUCTURE IN A NEURAL NETWORK THROUGH SPIKE TRAIN ANALYSIS

Evan Brooks, B.A.

Thesis Prepared for the Degree of

MASTER OF ARTS

UNIVERSITY OF NORTH TEXAS

May 2007

Brooks, Evan, Determining Properties of Synaptic Structure in a Neural Network through Spike Train Analysis. Master of Arts (Mathematics), May 2007, 38 pp., 8 tables, 8 illustrations, references, 3 titles.

A “complex” system typically has a relatively large number of dynamically interacting components and tends to exhibit emergent behavior that cannot be explained by analyzing each component separately. A biological neural network is one example of such a system. A multi-agent model of such a network is developed to study the relationships between a network’s structure and its spike train output. Using this model, inferences are made about the synaptic structure of networks through cluster analysis of spike train summary statistics A complexity measure for the network structure is also presented which has a one-to-one correspondence with the standard time series complexity measure sample entropy.

TABLE OF CONTENTS

Page

LIST OF TABLES v

LIST OF FIGURES vi

Chapter

I. INTRODUCTION 1

II. THE PHYSIOLOGY OF A BIOLOGICAL NEURON 3
 1. Structure of a Typical Neuron 3
 2. Process 4
 3. Network Properties 5

III. SIMULATED NEURAL NETWORK MODEL 7
 1. Single Neuron 9
 2. Network Algorithm 10

IV. PROFILING STRUCTURE BY SPIKE TRAIN ANALYSIS 14
 1. Moving Average Process 14
 2. Linking Moving Averages with Network Structure 15
 3. Using a Profile Map to Infer Properties of Network Structure 18

V. CORRESPONDENCE BETWEEN MEASURES OF STRUCTURAL COMPLEXITY AND SPIKE TRAIN COMPLEXITY 21
 1. Complexity Measure for the Spike Train 21
 2. Complexity Measure for the Structure 23

VI. CONCLUSION 29
 1. Determining Structure of Neural Networks by Cluster Analysis Map 29
 2. Network Properties Associated with SampEn Scores of Zero 30

3. Development of a Structural Complexity Measure in the 3-Neuron Case 31

4. Possible Extensions 31

APPENDIX: MATLAB CODE 32

Neural Network Simulator 33

Moving Average Calculator 35

Sample Entropy Calculator 36

LIST OF REFERENCES 38

LIST OF TABLES

Page

1. Neural network simulation parameters, input by user 7
2. Time-dependent quantities for the neural network 8
3. Initial values for time-dependent quantities 9
4. Parameters for example network 11
5. Network parameters used in this paper 13
6. Relative frequency of specific connections by cluster in 3-neuron case 16
7. Relative frequency of specific connections by cluster in 4-neuron case 18
8. Configurations up to isomorphism, 3-neuron case 25
9. Parameters used for calculating structural complexity of 3-neuron network 26

LIST OF FIGURES

Page

1. Culture of spinal tissue growing on 64-electrode recording matrix. Center for Network Neuroscience, UNT.. 5

2. Spike train of simulated 4-neuron network.. 6

3. Graph representation of example network.. 11

4. Spike train for example 3-neuron network 13

5. Moving average points for 3-neuron networks, grouped by cluster.................. 15

6. Subgraph of members of Cluster #4 16

7. Examples of isomorphic neural networks.. 25

8. Comparison of structural complexity and sample sntropy for 3-neuron networks 27

I. INTRODUCTION

"Complexity" is a concept which arises often in system analysis. However, the term is ill-defined and often used subjectively. The general consensus is that a complex system has a relatively large number of dynamically interacting components and tends to exhibit emergent behavior that cannot be explained by analyzing each component separately. Burggren and Monticino (2005) stated that any definition of physiological complexity must take three things into account:

- A lack of high predictability of output
- Sensitivity to initial conditions
- Nonlinear interactions between components

A biological neural network is certainly one example of such a system.

This paper has two main objectives:

- Analyze the time series output of a simulated neural network and make inferences about the underlying synaptic structure that produced it.
- Show a correspondence between a complexity measure on the time series output and a complexity measure on the network's structure.

If a time series of neural outputs is given, one natural question to ask is what is known about the structure of the network that produced it. It is also natural to think the number of neurons and the nature of the connections between them relate to the network's structural complexity, and to think that the more structurally complex a network is, the more complex its output should be. A second question, then, is whether one can establish such a link. The goal of this paper is to present answers to both of these questions.

This paper begins with a review of biological neurons. Then the algorithm used to simulate a biological neural network is described. After this, a method of profiling neural networks by analyzing their spike trains is given, followed by a method of determining properties of a network's structure based on this profile. Next, measures of structural and time series complexity are presented. Finally, a correspondence between the time series and structural complexities is given.

II. THE PHYSIOLOGY OF A BIOLOGICAL NEURON

This section reviews how an individual neuron and a neural network function. In general, the term neuron refers to an element which receives information and subsequently emits information. This paper considers simulations of biological neurons.

1. Structure of a Typical Neuron

A neuron has a cell body surrounded by a semi-permeable membrane. This membrane contains structures called ion pumps that regulate the electric potential in the cell body relative to the space around the cell, providing it with an equilibrium potential. In addition to the cell body, the typical neuron has a structure on one end with branchlike protrusions called dendrites (from the Greek dendros). The dendrites conduct electric impulses from other neurons into the neuron the dendrites belong to. On the other end of the neuron, there is a relatively thicker single protrusion called an axon. The axon is a long, tentacle-like filament that extends from the cell body and is typically shielded by a myelin sheath, which facilitates conduction of electricity down the length of the axon. The myelin sheath often partitions the axon into distinct nodes, lined up one after another. At the end of the axon opposite the cell body, the axon branches out like a fraying rope or tree. This region is called the terminal arborization. It is here that the neuron is connected to other neurons. The gap between the terminal arborization of one neuron and the dendrites of another is called the synapse. Also contained within the terminal arborization are chemicals which provide communication across the synapse when released. These chemicals are collectively referred to as neurotransmitters. There are many types of neurotransmitters, some which excite other neurons and others which inhibit. Information cannot cross the synapse without neurotransmitters (Anderson, 1995).

2. Process

A neuron transmits information by way of electric and chemical impulses to other neurons connected to it. A typical neuron will receive information through its dendrites from other neurons in the form of chemical neurotransmitters. These chemicals will have an effect on the cell body's electric potential, either raising it or reducing it, depending on the type of neurotransmitter. Although the ion pumps in the cell body continuously work to keep the cell in equilibrium, enough input can raise the potential inside the neuron. If this potential is raised high enough across the threshold, the neuron's potential undergoes a dramatic increase for a very brief period of time. This surge has a high enough potential for an electric impulse, called a spike or the action potential, to travel down the axon. The typical cell body can produce spikes at a frequency of 1000 Hz (Anderson, 1995). After producing a series of spikes, a neuron's cell body cannot maintain the potential difference and fatigues, collapsing back to equilibrium potential as it discharges through the cell membrane instead of down the axon (Anderson, 1995). The spikes moving down the axon eventually reach the terminal arborization. Their arrival triggers the release of neurotransmitters across the synaptic gaps into the dendrites of other neurons. The neuron replenishes its stock of neurotransmitters but can be depleted by an overflow of spikes in a short time. In this case, no information can travel across the synapse until the neurotransmitter stock has been replenished.

3. Network Properties

Whenever one neuron fires, the spike it produces will affect all the neurons connected to it. These neurons in turn may fire spikes of their own. The synaptic structure of a network is a map of how the neurons connect to one another. The synapses connect unidirectionally, from the terminal arborization of one neuron into the dendrites of another.

Neural activity can be monitored by putting electrodes into the network or growing a neural culture on an existing electrode matrix, as shown in Figure 1. The electrodes detect and record spikes. In practice, there are many neurons to each electrode. For the simulation in this paper, each neuron is assumed to have its own electrode which "listens" to it.

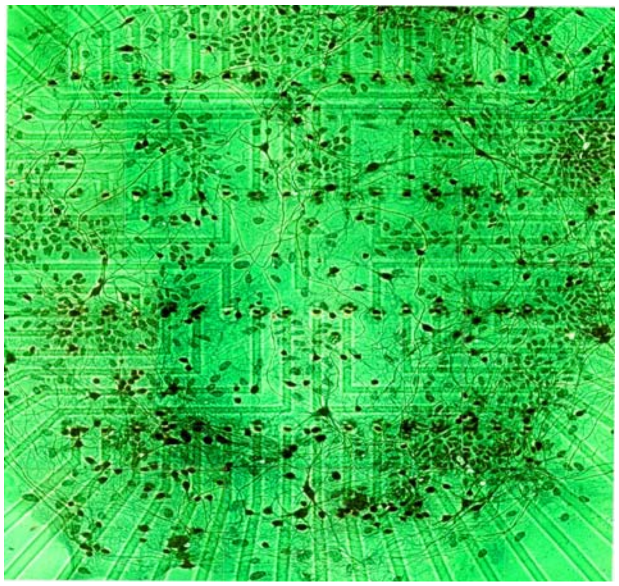
Figure 1. Culture of spinal tissue growing on 64-electrode recording matrix. Center for Network Neuroscience, UNT.

A spike train is a time series graph of the output of a neuron or a set of neurons. One objective of this paper is to analyze the spike train of a simulated neural network and make inferences about the underlying synaptic structure that produced it. The other objective is to show a correspondence between an entropy measure on the spike train and a complexity measure on the structure.

This paper only considers networks with a small number of neurons. While clearly not representing the number of neurons in a highly evolved animal, there do exist creatures whose nervous systems consist of as few as 10 neurons.

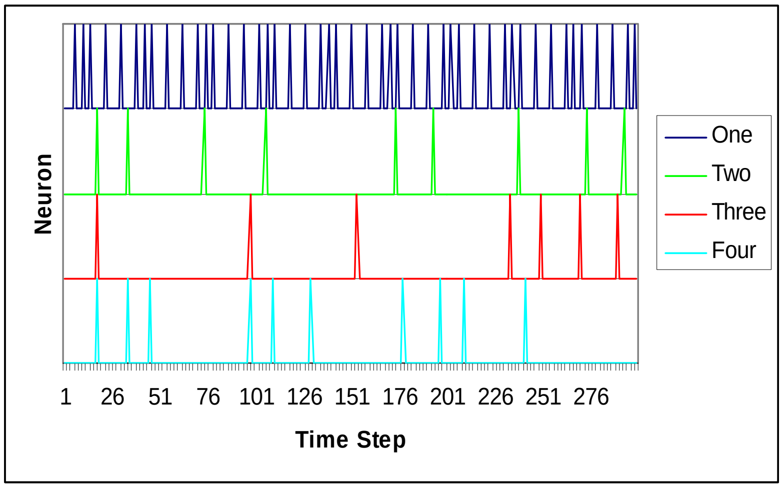

Figure 2. Spike train of simulated 4-neuron network.

III. SIMULATED NEURAL NETWORK MODEL

To simulate a neural network, a multi-agent algorithm processes inputs and outputs of N individual neurons over T discrete time steps and records the spike train output. This algorithm is encoded in Matlab and included in the Appendix of the paper. Note that the network requires an external stimulus to drive it. This is because the neurons only fire upon receiving sufficient stimulus. The algorithm is deterministic, operating on fixed parameters input by the user. Parameters are given in Table 1.

Table 1. Neural network simulation parameters, input by user.

Parameter	**Description**
V	External stimulus applied each time step to one neuron, required to drive the network.
θ	Threshold potential in the cell body. If cell body potential is higher than threshold, a spike is sent down the axon.
λ	Amount of potential removed by ion pumps over each time step.
L	Length of the axon, in nodes. It takes one time step to traverse one node.
C	Maximum capacity for neurotransmitters in the neuron.
k	Cost in neurotransmitter units for sending a spike from the terminal arborization.
R	Rate of neurotransmitter replenishment at each time step.
N	Number of neurons in the network.
$D = (d_{ij})$	N x N matrix representing connections between neurons. $d_{ij} = 1$ if neuron i connects into neuron j, $d_{ij} = 0$ otherwise.

The algorithm also records the time-dependent states of several components of the individual neurons within the network. These values will change as the simulation runs, and can be viewed as functions of time. Examples of these states include cell body potential in each neuron at time *t* and the level of neurotransmitters in each neuron at time *t*. A full list of these quantities is given in Table 2.

Table 2. Time-dependent quantities for the neural network.

Quantity	**Description**
$v_j(t)$	Potential in the cell body of j^{th} neuron at time t before adding inputs from other neurons.
$w_j(t)$	Potential in the cell body of j^{th} neuron at time t after adding inputs from connected neurons.
$A_j(t) = \{a_{1,j}(t), a_{2,j}(t), \ldots a_{L,j}(t)\}$	Firing state of the axon of j^{th} neuron at time t, represented as an array. The i^{th} element represents the state of the i^{th} node, with $a_{1,j}(t)$ being the node closest to the cell body. If the node contains a spike, then $a_{1,j}(t) = 1$. Otherwise, $a_{1,j}(t) = 0$.
$c_j(t)$	Quantity of neurotransmitters present in j^{th} neuron at time t.
$Arb_j(t)$	Spike output of j^{th} neuron at time t. Takes value 1 if a spike is sent, 0 otherwise.
$S(t) = \{Arb_1(t), Arb_2(t), \ldots Arb_N(t)\}$	Spike output of the entire network at time t.

The initial values for the time-dependent parameters for all neurons are listed in Table 3.

Table 3. Initial values for time-dependent quantities.

Quantity	**Initial Value**
$v_j(t)$	$v_j(1) = 0$
$w_j(t)$	$w_j(1) = 0$
$A_j(t)$	$A_j(1) = \{0, 0, \ldots, 0\}$
$c_j(t)$	$c_j(1) = C$
$Arb_j(t)$	$Arb_j(1) = 0$

1. Single Neuron

Each neuron simulated in the network is assumed to have the same parameters. The simulation model for a single neuron proceeds as follows.

Cell Body Potential Algorithm

The cell body potential in the j^{th} neuron at time t is given by $v_j(t)$. At each time step, the algorithm sums up inputs from other neurons connected to the j^{th} neuron. The input from a neuron is 0 if the neuron did not send a spike the previous step and is 1 if it did. Once summed, the inputs are added to $v_j(t)$ to obtain $w_j(t)$, the potential in the cell body at time t after inputs from the other neurons have been added in. If $w_j(t) > \theta$, then the neuron fires a spike and resets the cell body potential to equilibrium. That is, $v_j(t+1)$ is reset to 0. Otherwise, the ion pumps in the cell body reduce the cell potential by a small amount λ, in which case $v_j(t+1) = w_j(t) - \lambda$.

Axon Algorithm

If $w_j(t) > \theta$, then the cell body sends a spike into the axon, so $a_{1,j}(t) = 1$. Otherwise, no spike is sent, so $a_{1,j}(t) = 0$. A spike propagates down the length of the axon one node at a time. It takes one time step for a spike to traverse a node. Thus, for $l = 2, 3, \ldots, L$; set $a_{l,\mathrm{j}}(t) = 1$ iff $a_{\mathrm{i}-1,j}(t\text{-}1) = 1$.

Synapse Algorithm

Recall that the neuron has a given neurotransmitter capacity C and a level of neurotransmitters at time t, $c_j(t)$. It also has a cost, k, for sending the spike out. Recall that if there is a spike at the end of the axon, then $a_{L,j}(t) = 1$. If this is the case, the algorithm checks if there are enough neurotransmitters available for transmission of the spike. If $c_j(t)\text{-}k \geq 0$, then the spike moves out across the synapse. The algorithm accordingly sets $Arb_j(t) = 1$. Otherwise, there are not enough neurotransmitters available, so $Arb_j(t) = 0$. If the spike moves out, then the algorithm deducts the cost from $c_j(t)$ and replenishes the level according to the set parameters. That is, $c_j(t+1) = c_j(t)–k+R$. If $Arb_j(t) = 0$, then there is no cost to deduct, so $c_j(t+1) = c_j(t)+R$.

2. Network Algorithm

Now consider a network of N neurons. Recall that the N x N matrix $D = (d_{ij})$ represents the connective structure of the network. If neuron i connects into neuron j, then $d_{ij} = 1$, otherwise $d_{\mathrm{ij}} = 0$. For example, in a four-neuron network, $d_{14} = 1$ implies that neuron #1 connects into #4. Neurons are assumed not to connect into themselves, so $d_{\mathrm{jj}} = 0$ for all j.

A designated neuron receives an artificial external stimulus every time step to drive the system. This neuron is called the "leader". Without loss of generality the leader is always neuron #1.

Recall that the spike output of the network at time t is given by $S(t) = \{Arb_1(t), Arb_2(t), \ldots Arb_N(t)\}$. Also recall that V represents the amount of external stimulus the leader receives. For the leader, the sum of the inputs at time t is given by the dot product of the first column of D with $S(t\text{-}1)$, plus the external stimulus V. For $j > 1$, the sum of the inputs for the j^{th} neuron at time t is given by the dot product of the j^{th} column of D with $S(t\text{-}1)$.

Plotting $S(t)$ over $1 \leq t \leq T$ yields the spike train. Both objectives of this paper involve analyzing this spike train.

Example

Consider a three neuron network with the following parameters:

Table 4. Parameters for example network.

Parameter	Value
V	1
θ	2
λ	0
L	2
C	2
K	1
R	1

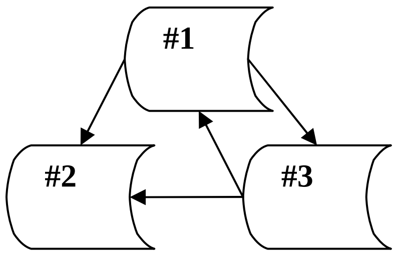

Figure 3. Graph representation of example network.

Suppose $D = \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 1 \\ 1 & 0 & 0 \end{bmatrix}$. When considering a network's structure, it is convenient to view a neural network of *N* neurons as a directed graph with *N* vertices, as shown in Figure 3. In this context, the arcs represent the unidirectional connections between neurons.

Tracking the output vector, *S*(*t*), for the first 10 time steps yields the following results. The spike trains for each neuron are given in Figure 4.

Step 1: The artificial input has not stimulated the leader above the threshold. *S*(1) = {0, 0, 0}.

Step 2: The leader fires a spike, but the spike has not reached the terminal arborization yet. *S*(2) = {0, 0, 0}.

Step 3: The spike is carried across the synapse into neurons #2 and #3. *S*(3) = {1, 0, 0}.

Step 4: The leader has fired another spike down its axon; #2 and #3 have received input. *S*(4) = {0, 0, 0}.

Step 5: Another spike is carried into #2 and #3. *S*(5) = {1, 0, 0}.

Step 6: All three neurons are sending spikes now. *S*(6) = {0, 0, 0}.

Step 7: All three spikes reach the synapses and are carried across. *S*(7) = {1, 1, 1}.

Step 8: The leader has received an input from #3 and from the artificial stimulus, so it fires again. #2 also receives enough inputs to fire again, but #3 does not. *S*(8) = {0, 0, 0}.

Step 9: #1 and #2 fire across the synapse. *S*(9) = {1, 1, 0}.

Step 10: On the next step, #1 will fire again. #3 has one input already, and will fire before #2 fires again. *S*(10) = {0, 0, 0}.

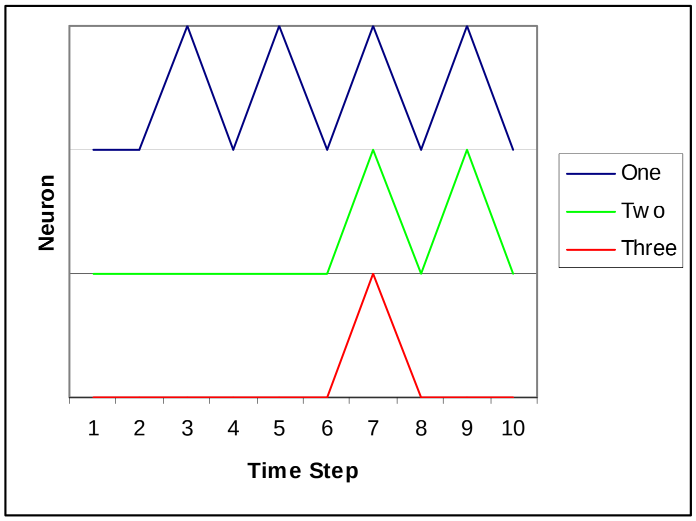

Figure 4. Spike train for example 3-neuron network.

For the remainder of this paper, the parameters used for the neural network simulation are listed in Table 5. These values are chosen because they reflect realistic values in a biological network (Anderson, 1995).

Table 5. Network parameters used in this paper.

Parameter	Value
V	25 (mV)
θ	75 (mV)
λ	3 (mV)
L	2 (nodes)
C	8 (units)
K	4 (units)
R	1 (unit)

Note that the spike train output of a network depends on the values of these parameters. Thus, while the methods in this paper are independent of the choice of these parameters, the specific values in the models are not.

IV. PROFILING STRUCTURE BY SPIKE TRAIN ANALYSIS

The objective of this section is to profile a neural network's structure through simple analysis of its spike train. This allows inferences to be made about structure when given only the spike train. Such profiling is accomplished by calculating a point of moving averages in $[0,1]^N$ for each network and then classifying the networks by grouping these points.

1. Moving Average Process

A simple method of analyzing a spike train is to calculate a "moving average" for the series. This average represents the typical frequency at which each neuron fires. By treating each neuron's moving average as the coordinate of a point in $[0,1]^N$, one can use these points as representations of the networks they are derived from.

Calculating the Moving Average

Given a spike train output, specify a time interval of length m. Let $Arb_j(t)$ be the output of the jth neuron at time t. For $t = 1, 2, .., T\text{-}m$; let $\mu_j(t) = \frac{1}{m}\sum_{k=t}^{t+m} Arb_j(k)$ be the average value in this m-window. Then average all of these averages across the run time. Call this value $M_j = \frac{1}{T-m}\sum_{k=t}^{T-m}\mu_j(k)$. Then $(M_1, M_2, \ldots, M_N)$ is the point in $[0,1]^N$ representing the moving average value for the spike train of the network.

2. Linking Moving Averages with Network Structure

Given a spike train and its moving average point, $M = (M_1, M_2, \ldots, M^N)$, the goal is to determine the structure of the network which produced it. By performing a cluster analysis, one can group together those networks with similarly positioned moving average points. Comparing the networks within each cluster, it turns out that networks within any given cluster have a common subgraph.

Example

Consider the family of 3-neuron networks. For each network, let the network's connectivity matrix be given by $D = (d_{ij})$ and let k be the base 10 value of $d_{12}\, d_{13}\, d_{21}\, d_{23}\, d_{31}\, d_{32}$. Let M_k denote the moving average point of the network indexed by k. Figure 5 plots all 64 moving average points in $[0,1]^3$. Cluster analysis was performed on the points with nine target clusters. That is, the points were grouped together according to their relative proximity. The clusters are indexed by color in Figure 5.

Grouping the networks according to their clusters and taking the average value for each connection within a cluster yields a relative frequency of connections within each cluster, shown in Table

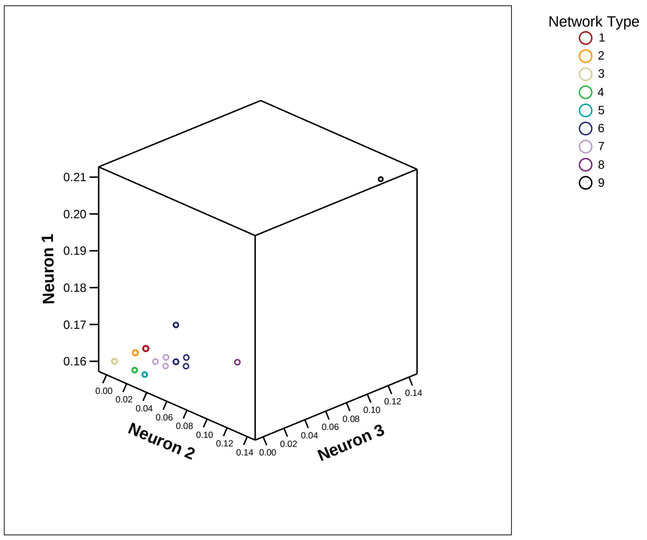

Figure 5. Moving average points for 3-neuron networks, grouped by cluster.

6. Each percentage value in the table represents the proportion of networks in the given cluster and connection. In the context of this example, 100% of the networks in cluster #1 have the leader connected directly to neuron #3.

Table 6. Relative frequency of specific connections by cluster in 3-neuron case.

Cluster	d_{12}	d_{13}	d_{21}	d_{23}	d_{31}	d_{32}
1	0%	100%	50%	50%	100%	50%
2	0%	100%	50%	50%	0%	50%
3	0%	0%	50%	50%	50%	50%
4	100%	0%	0%	50%	50%	50%
5	100%	0%	100%	50%	50%	50%
6	100%	100%	67%	33%	67%	33%
7	100%	100%	0%	33%	0%	33%
8	100%	100%	0%	100%	0%	100%
9	100%	100%	67%	100%	67%	100%

Based on the connection frequencies in Table 6, it is easy to see that the networks within each of the nine clusters have traits in common. This is equivalent to saying that the graphs of each network within a cluster have a common subgraph. For example, the networks in cluster #8 all contain the graph depicted in Figure 6 as a part of their structure.

The fact that network structure can be profiled by analyzing the spike trains also supports the idea that a relationship exists between the complexity of the network's structure and the complexity of its spike train.

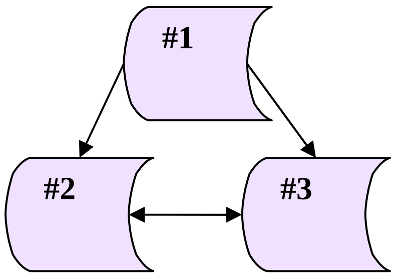

Figure 6. Subgraph of members of Cluster #4.

Example

In the 4-neuron case, define k and M_k as in the previous example, with k representing the 12-bit number $d_{12}\ d_{13}\ d_{14}\ d_{21}\ d_{23}\ d_{24}\ d_{31}\ d_{32}\ d_{34}\ d_{41}\ d_{42}\ d_{43}$. This time, it is not easy to graphically represent the plot of all 4,096 networks in $[0,1]^4$. However, it is still possible to perform a cluster analysis on the points. This process compares the distances between points and groups points together into a preset number of clusters. In this example, that preset number of clusters is 15.

As in the previous example, grouping the networks according to their clusters and taking the average value for each connection within a cluster yields a relative frequency of connections within each cluster. This is shown for four neurons in Table 7. Again, each percentage value in the table represents the proportion of networks in the given cluster and connection.

The bold values in the chart indicate a connection type common to almost all the networks in that particular cluster. In particular, note that most of the bold values are in the first three columns, implying that the first neural connections to be profiled are the connections from the leader. This implies that the first distinguishing feature of a network's structure is the way in which the leader directly connects to the other neurons. By using more clusters, one can profile the networks in greater detail. However, the picture in Table 7 is enough by itself to motivate a method for completing the first objective of the paper. This method is outlined in the next section.

Table 7. Relative frequency of specific connections by cluster in 4-neuron case.

Cluster	d_{12}	d_{13}	d_{14}	d_{21}	d_{23}	d_{24}	d_{31}	d_{32}	d_{34}	d_{41}	d_{42}	d_{43}
1	27%	0%	0%	45%	41%	41%	50%	48%	50%	50%	48%	50%
2	100%	100%	26%	50%	100%	25%	50%	99%	25%	50%	50%	50%
3	87%	87%	88%	44%	80%	77%	47%	78%	81%	46%	75%	81%
4	100%	26%	100%	50%	25%	100%	50%	50%	50%	50%	99%	25%
5	77%	67%	68%	51%	61%	61%	57%	69%	69%	54%	70%	67%
6	73%	79%	77%	58%	69%	77%	52%	77%	67%	54%	77%	72%
7	30%	99%	100%	52%	53%	53%	53%	25%	95%	47%	26%	99%
8	0%	100%	0%	50%	48%	47%	49%	46%	46%	50%	48%	47%
9	100%	9%	43%	60%	35%	57%	48%	56%	50%	49%	77%	37%
10	68%	74%	99%	47%	42%	66%	51%	39%	71%	47%	38%	30%
11	100%	56%	0%	50%	52%	32%	49%	47%	32%	50%	46%	48%
12	84%	94%	47%	51%	58%	34%	53%	31%	37%	50%	50%	59%
13	0%	0%	100%	50%	49%	48%	50%	46%	48%	49%	45%	45%
14	91%	0%	100%	49%	32%	50%	50%	52%	52%	53%	9%	38%
15	0%	99%	96%	50%	50%	51%	48%	36%	34%	52%	28%	39%

3. Using a Profile Map to Infer Properties of Network Structure

By taking a sample of *N*-neuron networks, one can perform a cluster analysis to group the networks together according to their moving average points. In essence, clustering a sample of networks produces a rough "map" of the full family of *N*-neuron networks. In other words, the clusters generated by the process divide $[0,1]^N$ into sections. As shown before, networks in these sections have common structural properties. The higher the number of clusters one chooses, the greater the specificity one gains in the profiling. Thus, the number of clusters acts as a "resolution" parameter for the mapping process. Given a spike train from an *N*-neuron network, one can determine which cluster the network's moving average point is closest to. One can then use this map to predict structural properties of the network in question.

Example

Consider a random sample of 400 4-neuron networks, about 10% of the full family. By calculating the moving average points for the networks in the sample, one can perform a cluster analysis as shown before. In this example, 15 clusters were chosen to provide a rough map. Associating each network with its cluster number gives a map over the 4-neuron networks. Note that some "landmark" networks, such as the blank network or the completely connected network, have extreme-valued moving average points. For example, the moving average point for the blank network will have 0 for every coordinate except the first. Including these networks in sample will improve the map's accuracy by forcing the cluster analysis to accommodate these extreme cases.

Recall that each 4-neuron network has a 12-bit binary expansion given by the number $d_{12}\ d_{13}\ d_{14}\ d_{21}\ d_{23}\ d_{24}\ d_{31}\ d_{32}\ d_{34}\ d_{41}\ d_{42}\ d_{43}$. By averaging each bit in the network's binary expansion within a cluster, one gains a matrix similar to the one in Table 7. The elements in the matrix can be viewed as probabilities. For example, if a network from a given cluster were chosen at random, the element corresponding to that cluster and the element d_{12} represents the probability that the chosen network includes a connection from neuron #1 to neuron #2.

Given a spike train output from a network known to include four neurons, one can calculate its moving average point, call it M. By calculating the Euclidean distance between M and every point in the sample, one can determine which of the sample points is the closest neighbor of M. Then one can predict that M belongs to the same cluster as that nearest point, say cluster j. Then the j^{th} row of the probability matrix predicts the target network's connection matrix, with 1's being a certain connection, 0's being a certain disconnection, and numbers in between representing the likelihood of a connection.

One can describe the accuracy of this map by comparing the predicted certain connections and disconnections to the actual connectivity matrix for the target network. If every certain prediction matches the corresponding value in the target network, the prediction is considered accurate. To test this mapping process, a 400 network sample was used to draw the map and predict the connection matrices of 200 randomly selected networks. The predictions were accurate in 199 of 200 cases.

By inferring structural properties of a neural network based on a given spike train, this mapping method completes the first objective of the paper.

V. CORRESPONDENCE BETWEEN MEASURES OF STRUCTURAL COMPLEXITY AND SPIKE TRAIN COMPLEXITY

The previous section proposed a method for inferring the synaptic structure of a network by analyzing a spike train. The second objective of this paper is to show a relationship between the structural complexity and the spike train complexity. To do this, a measure of complexity for each concept is needed. It is natural to think that the more structurally complex a network is, the more complex its spike train should be. This section presents a complexity measure for the spike train output of the neural network. It then describes common network features that correspond to specific values of the spike train complexity measure. Based on these features, a proposed complexity measure for the network structure in the 3-neuron case is presented. This measure corresponds in a one-to-one fashion with the time series complexity measure.

1. Complexity Measure for the Spike Train

As a multivariable time series, the spike train can be analyzed using the Sample Entropy statistic. This statistic measures the level of aperiodicity in the spike train.

Definition of Sample Entropy

Given a time series $S(t)$ for $1 \leq t \leq T$, call $\{S(k), S(k+1), \ldots, S(k+m)\}$ is called a sequence of length m, for $1 \leq k \leq T\text{-}(m+1)$. For two sequences of length m, $\{S(k), S(k+1),$

…, $S(k+m)$} and {$S(n)$, $S(n+1)$, …, $S(n+m)$}, say they match within a given error tolerance $\varepsilon>0$ if $\sum_{j=0}^{m} \left| S(k+j) - S(n+j) \right| \le \varepsilon$.

For a specified sequence length m and error tolerance ε, the sample entropy of a time series, or SampEn score, is the negative logarithm of the conditional probability that two matching sequences of length m will continue to match if they are extended to sequences of length $m+1$.

Properties of Sample Entropy

Empirically, since the time series is known, the conditional probability that two matching sequences of length m will continue to match if they are extended to sequences of length $m+1$can be calculated easily. Let A be the number of matching sequences of length m, and let B be the number of matching sequences of length $m+1$. Note that B is necessarily no larger than A. Then SampEn(m, ε) = $-\ln(\frac{B}{A})$. Define SampEn(m, ε) = 0 if $B = 0$ to ensure that SampEn is well-defined. Note that if the time series is periodic, then $A = B$, so SampEn(m, ε) = 0. Also note that if $A = 0$, then SampEn(m, ε) = 0 since the conditional probability is trivially 1. Thus time series that are aperiodic to the point of being random also have low SampEn scores, since A is small.

Higher SampEn scores occur when the time series has a structured aperiodicity. This lack of predictability conforms to one of the generally agreed-upon qualities of physiological complexity outlined by Burggren and Monticino (2005). SampEn scores are also known to be relatively robust with respect to the choice of m and with respect to the length of the time series (Richman and Moorman, 2000). This means that one has some freedom in choosing m. It also means that SampEn scores for a 300-step time series are just as valid as SampEn scores for a 1,000-step time series.

The ability of SampEn to measure structured aperiodicity and assign appropriate values to both periodic and completely random time series suggests that it is a valid complexity measure for the time series. The fact that it is relatively independent of the choice of window width and time series length makes it very appealing, computationally. These are the primary reasons SampEn is used to represent the spike train complexity.

2. Complexity Measure for the Structure

The basic assumption in this section is that a more complex structure is associated with a more complex spike train. Given this assumption, the second objective of this paper is really to determine which structural properties in a network add to its "complexity."

When considering a network's structure, it is convenient to view a neural network of N neurons as a directed graph with N vertices. In this context, the arcs represent the synaptic connections between neurons.

Structural Properties of Networks with Extreme SampEn Scores

To date, there have been no measures of graphical complexity established for a graph with a small number of vertices. It is thus necessary to construct such a measure from the structural properties of the network. Intuitively, a complexity measure for the structure of a neural network should include properties such as the number and type of feedback loops, any symmetries in the network between neurons, and a measure of volume in the sense of how many neurons are actively involved.

Empirically, one significant result is that the sample entropy for the network associated with the completely connected graph is 0. This is true regardless of the number of neurons in the network. This implies that a completely connected network, while structurally saturated, is no more "complex" than the blank or completely disconnected network. Another result of note is that networks with strong "symmetry" have low SampEn scores. One example of network symmetry is when one can re-label a pair of non-leader neurons so that the connectivity matrix remains unchanged. The spike trains of symmetric networks tend to look periodic after a small number of time steps. The networks which yield high SampEn scores tend to be the ones where each neuron has a different level of input than its neighbors. This "staggering" effect prevents the network from settling into periodic spike trains.

Graph Isomorphisms

When constructing a structural complexity measure, it is important to note that many network graphs are isomorphic. This fact reduces the number of structures to consider significantly. Recall that two graphs G and H with vertices $V_G = \{v_1,v_2,\ldots v_N\} = V_H$ are isomorphic if there exists a permutation p: $V_G \rightarrow V_H$ such that (v_j, v_k) is an arc in G iff $(p(v_j), p(v_k))$ is an arc in H. In the neural networks in this paper, the leader neuron is the only neuron to receive external stimuli. Since the parameters for each neuron in the network are the same, it follows that two networks are isomorphic if such a permutation exists and it fixes the leader (compare Figure 7a and 7b). Recall that a component of a graph is connected to a vertex if there exists a sequence of arcs from that vertex to the component. In the context of this paper, two networks are also considered isomorphic if the components of each network which are connected to the leader are isomorphic (compare Figure 7a and 7c, or 7b and 7c).

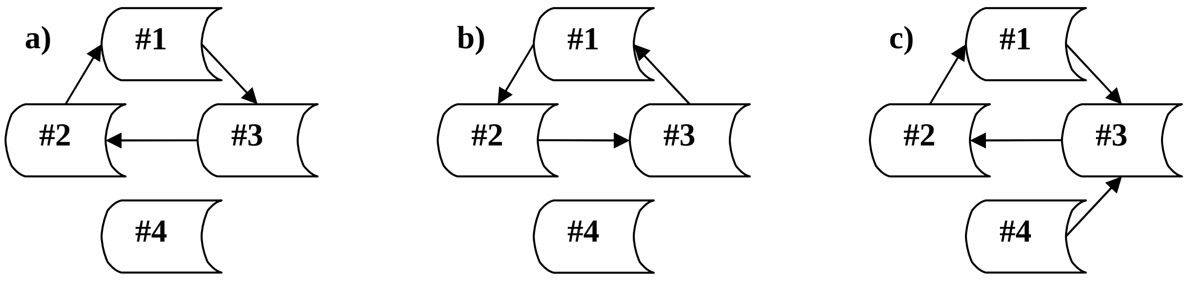

Figure 7. Examples of isomorphic neural networks.

It is relatively easy to show that two networks are isomorphic if their connection matrices are equivalent through row operations.

A Structural Complexity Measure for 3 Neurons

In the 3-neuron case, there are only 19 configurations up to isomorphism. These 19 configurations are listed in Table 8.

Table 8. Configurations up to isomorphism, 3-neuron case.

d_{12}	d_{13}	d_{21}	d_{23}	d_{31}	d_{32}
0	0	0	0	0	0
1	1	0	1	0	1
1	1	1	1	0	1
1	1	1	1	1	1
1	1	1	1	1	0
1	0	0	0	0	0
1	1	0	0	0	0
1	0	0	1	0	0
1	0	0	1	1	0
1	0	0	1	0	1
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	0	0	0
1	0	1	1	1	1
1	0	1	0	0	0
1	0	1	1	0	0
1	0	1	1	0	1
1	1	0	1	0	0
1	1	1	1	0	0

Using this as a basis for calculations, one can define several structural parameters, listed in Table 9. These parameters represent the properties discussed in the beginning of this section, which are intuitively linked to structural complexity.

Table 9. Parameters used for calculating structural complexity of 3-neuron network.

Parameter	Variable	Formula
Number of 1-step feedback loops into the leader	*Loop*1	$Loop1 = (d_{12})(d_{21}) + (d_{13})(d_{31})$
Number of 2-step feedback loops into the leader	*Loop*2	$Loop2 = (d_{12})(d_{23})(d_{31}) + (d_{13})(d_{32})(d_{21})$
Asymmetry between neurons 2 and 3	*Asym*	$Asym = \|d_{23}-d_{32}\|$
Significance level of neuron 2 output	*Sig*2	$Sig2 = d_{12}(1 + 0.5(d_{32})(d_{13}))$
Significance level of neuron 3 output	*Sig*3	$Sig3 = d_{13}(1 + 0.5(d_{21})(d_{23}))$
Degree to which neurons 2 and 3 influence neuron 1 through feedback	*Phase*	$Phase = \text{Max}\{(d_{12})(d_{21}), (d_{13})(d_{31})\} + \text{Max}\{(d_{12})(d_{23})(d_{31})(Sig2), (d_{13})(d_{32})(d_{21})(Sig3)\}$
Saturation of feedback to neuron 1	*Sat*	$Sat = \begin{cases} 0 & Phase = 0 \\ (1+Phase)^{-1} & Phase > 0 \end{cases}$

Then define the structural complexity, *K*, of the network as follows:

$$K = [d_{12} + (d_{13})(Asym) + Sat - (Loop1)(Loop2)(Asym)][\text{sgn}(2.25 - (Sig2)(Sig3))]$$

The first term in the first factor makes intuitive sense for a structural complexity measure, as a network in which the leader connects to one neuron has the potential to be more complex than a blank network. The second term adds to the measure, provided that there is some asymmetry in the way that the follower neurons are connected to each other. The third term adds more to the measure, depending on how strong the volume of feedback to the leader is. Note that as this volume increases, the amount added to the complexity measure decreases. The last term reflects the overall degree of "asymmetric connectedness" in the network. From this term, an asymmetric network has potential for a high complexity measure, but only if the degree of feedback to the leader (in the form of *Loop*1 and *Loop*2) is limited.

Note that the last factor describes the "saturation" of the network. The maximum possible value for *Sig*2 and *Sig*3 is 1.5, so this last factor is zero precisely when the follower neurons are "loud" enough to flood the network. When this happens, the spike train becomes periodic after a few time steps, and the sample entropy drops to zero. Using this measure, a one-to-one correspondence is achieved between structural complexity and SampEn scores for 3-neuron networks, as seen in Figure 8. Note that the point at $K \approx .26$ has a SampEn score of 0.083, while the point at $K = 1$ has a SampEn score of 0.084.

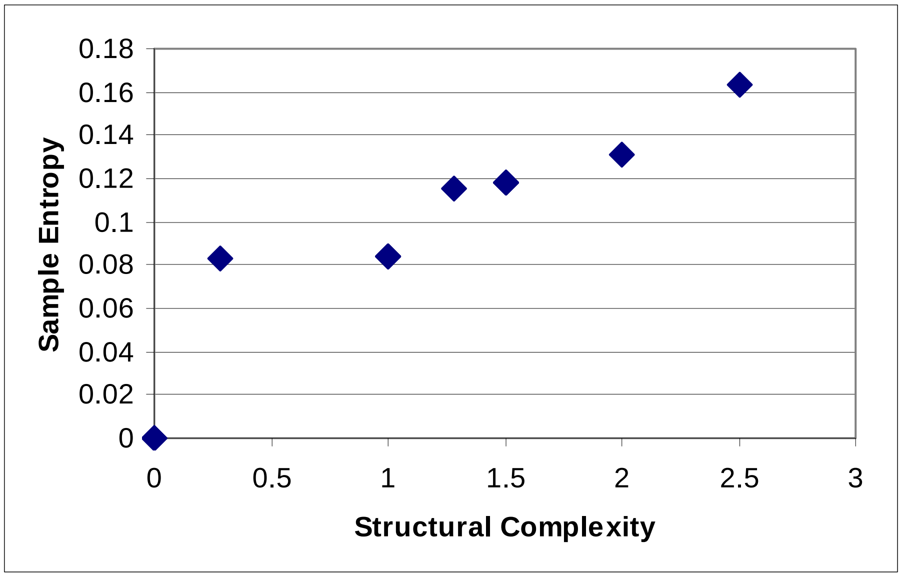

Figure 8. Comparison of structural complexity and sample sntropy for 3-neuron networks.

Given a SampEn score along with the number of neurons and the parameters for each neuron, one can infer information about the network's structure. In the context of this example, a SampEn score of 0 indicates that the network is isomorphic to a blank network or a completely saturated network. A SampEn score of 2 indicates that both neurons two and three are fed directly from the leader and are not symmetric with respect to each other, with no feedback loops to the leader.

Four Neuron Case

A natural question to ask at this point is whether this structure measure can be easily extended to a four-neuron network. Unfortunately, a natural extension preserving the one-to-one correspondence seen in the 3-neuron case does not appear to exist. When going from three neurons to four, it becomes unclear how to describe the network's "symmetry", which was easily described in the 3-neuron case by the variable *Asym* in Table 9. It also becomes unclear how to describe the saturation denoted in the second factor of the 3-neuron structure measure.

VI. CONCLUSION

Recall that this paper has two major objectives:

- Analyze the time series output of a simulated neural network and make inferences about the underlying synaptic structure that produced it.
- Show a correspondence between a complexity measure on the time series output and a complexity measure on the network's structure.

In pursuit of these objectives, the following results were obtained.

- One can profile the neural networks according to a moving average analysis of their spike trains and thus determine the structure of the network which produced it.
- Certain structural properties of the network yield specific sample entropy scores for the spike train.
- Using these properties, one can derive a structure measure to form a one-to-one correspondence with the sample entropy scores. From this correspondence, one can also infer which type of network produced a given spike train.

1. Determining Structure of Neural Networks by Cluster Analysis Map

In Section IV.2, it was shown that by calculating a moving average point for a spike train, one can classify the network that produced it by the point's location in [0,1]N. Recall that if enough clusters are allowed, each network in a cluster will have a common subgraph. It was shown in Section IV.3 that by taking a large enough sample of networks of a given size, one can perform a cluster analysis to produce a rough “map” of the networks in general. In this fashion, finding a spike train’s moving average point allows one to predict which network produced it. The more clusters are defined in the map and the larger the sample, the better the map will be at predicting the network associated with a given spike train. This achieves the first objective of this paper.

2. Network Properties Associated with SampEn Scores of Zero

It was noted in Section V.3 that regardless of the number of neurons in the network, the sample entropy for the network associated with the completely connected graph is always 0. In the context of this paper, the “complexity” of a completely connected network is no more than that of a completely disconnected network, implying that measures of structural complexity must include more than simply summing up the number of inputs or components in the system.

Also in Section V.3, networks with strong “symmetry” exhibit very low SampEn scores. Examples of these symmetric networks include the completely connected network described in the preceding paragraph, as well as networks where one could re-label a pair of neurons so that the connectivity matrix remains unchanged. Networks of this type tend to produce spike trains which move into “lockstep” within a few time steps. The networks which yield high SampEn scores tend to be the ones where each neuron has a different level of input than its neighbors. This “staggering” effect prevents the network from settling into the lockstep described before.

3. Development of a Structural Complexity Measure in the 3-Neuron Case

The insights from the second conclusion allow for the development of a structural measure for the family of 3-neuron networks which corresponds in a one-to-one fashion with the SampEn scores. The measure, shown in Section V.3, takes the number of connections, the symmetry of the network, and the level of feedback saturation into the leader into account. By doing so, it achieves a one-to-one correspondence with the SampEn scores from the 3-neuron family. Aside from justifying the intuitions behind its construction, the measure also offers another way of inferring structural properties when given a spike train output.

4. Possible Extensions

A few natural questions arise at the end of these conclusions. The first is what modifications need to be made to the 3-neuron structural measure from Section V.3 in order to extend it faithfully to the 4-neuron family. Another question is how sensitive is the network simulation to variations between the neurons' parameters. In this paper, each neuron had the same threshold, axon length, and neurotransmitter depletion rate. What would happen if these values were different for each neuron? Would a map like the one described in Section IV.3 still be valid? Questions like these are still open, and answering them may provide more understanding of just what makes a system "complex."

APPENDIX

MATLAB CODE

Neural Network Simulator

```
%This program takes inputs for artificial stimulus, number of neurons, and
%the number of time steps.  It outputs a matrix containing the spike train for the network.
function neuralnetwork(charge,N,T,numb)
X=zeros(N,T);

%Input a connectino matrix here, or load from another program.
D=[0 40 0 0; 0 0 0 40; 40 0 0 0; 40 0 0 0];

%Neruon Parameters/Initial Values.  Note how the program keeps track of the
%entire network at once.
Charge=zeros(N);
Charge(1)=charge;
P=75; %<--Threshold potential
v=zeros(N,T);
d=2;
L=4;
Axon=zeros(N,L);
c=4;
tran=zeros(N,T);
for i=1:1:N
   tran(i,1)=8;
end
C=8;
R=1;
V=zeros(N,T);
arb=zeros(N,T);
Arb=zeros(N,T);
%Time counter
Tau=zeros;
Tau(1)=1;
%Here are the interactions between the neurons.
for t=2:1:T
   Tau(t)=t;
   for n=1:1:N

      %Inputs into the neuron
      Total=0;
      for i=1:1:N
         Total=Total+Arb(i,t-1)*D(i,n);
      end
      v(n,t)=v(n,t-1)+Charge(n)+Total;
      V(n,t)=v(n,t);
```

```
        %determining whether to fire spike
        if v(n,t)>=P
            Axon(n,1)=1;
            v(n,t)=0;
        else v(n,t)=v(n,t)-d;
            if v(n,t)<0
                v(n,t)=0;
            end
        end

        %spike traveling down the axon length
        for i=1:1:L-1
            if Axon(n,L-i)==1
                Axon(n,L-i+1)=1;
                Axon(n,L-i)=0;
            end
        end

        %spike jumping across synapse if enough neurotransmitter
        if Axon(n,L)==0
            Arb(n,t)=0;
        else
            if tran(n,t-1)-c<0
                Arb(n,t)=0;
            else Arb(n,t)=1;
                tran(n,t)=tran(n,t-1)-c;
            end
            Axon(n,L)=0;
        end

        %recharging transmitter
        if tran(n,t-1)<C
            tran(n,t)=tran(n,t-1)+1;
        end
    end
end

save
```

Moving Average Calculator

```
%This program runs the network simulator and computes the moving average
%point.  It is designed to handle many networks together, to provide data
%for the cluster analysis.

function averager(charge, N, T, d, tot)


H=zeros(N,tot);
F=zeros(N,tot);

for take=1:1:tot
   take
   save
   neuralnetwork(charge,N,T,1,take)
   load
   D =D/40 ;
   A=zeros(N,1);
   for i=0:1:T-d
      temp=zeros(N,1);
      for j=1:1:d
         for k=1:1:N
            temp(k,1)=X(k,i+j)+temp(k,1);
         end
      end
      for k=1:1:N
         F(k,i+1)=temp(k,1)/d;
      end
   end
   A=mean(F,2);

   A(1);
   take;
   H(1,1);
   for k=1:1:N
      H(k,take)=A(k);
   end
   H';
   H(1,1);
end

H'

save
```

Sample Entropy Calculator

```
%This is the Sample Entropy algorithm.  It inputs a window width num, error
%tolerances, and a starting point.  The time series is loaded from another
%program.

function sampen(num,ep,r,start)
load

%adjust to size
[N2,N1]=size(X);
A=0;
B=0;

for k=start:1:N1-num-1
  for j=k+1:1:N1-num-1
    tempA=0;
    tempB=0;
    for i=1:1:num
      temp1A=0;
      temp1B=0;
      for phi=1:1:N2
        if X(phi, k+i-1)-X(phi, j+i-1)~=0
          temp1A=temp1A+1;
          temp1B=temp1B+1;
        end
      end
      if temp1A>r
        tempA=tempA+1;
      end
      if temp1B>r
        tempB=tempB+1;
      end
    end
    temp1B=0;
    for phi=1:1:N2
      if X(phi, k+num)-X(phi, j+num)~=0
        temp1B=temp1B+1;
      end
    end
    if temp1B>r
      tempB=tempB+1;
    end
    if tempA<=ep
      A=A+1;
```

```
        end
        if tempB<=ep
            B=B+1;
        end
    end
end

if A==0
    sentropy=0;
elseif B==0
    sentropy=0;
else
    sentropy=-log(B/A);
end
sentropy;

save
```

LIST OF REFERENCES

Anderson, James A. An Introduction to Neural Networks. Third Printing, 1997. 1995, Massachusetts Institute of Technology. Bradford Books, Cambridge, MA.

Burggren, W. W. and Monticino, M. G. Assessing Physiological Complexity. Journal of Experimental Biology, Vol. 208, 3221-3232, 2005.

Richman, Joshua S., and Moorman, J. Randall. “Physiological time-series analysis using approximate entropy and sample entropy.” American Journal of Physiology – Heart and Circulatory Physiology. Vol. 278, Issue 6, H2039-H2049, June 2000.