7-26-2011

Simulation of Abnormal/Normal Brain States Using the KIV Model

Mark H. Myers

To the University Council:

The Dissertation Committee for Mark Hebron Myers certifies that this is the final approved version of the following electronic dissertation: "Simulation of Abnormal/Normal Brain States Using the KIV Model."

_____

Robert Kozma, Ph.D.
Major Professor

We have read this thesis and recommend its acceptance:

_____

Stan Franklin, Ph.D.

_____

Walter Freeman, M.D.

_____

Max Garzon, Ph.D.

_____

Charles Blaha, Ph.D.

Accepted for the Council:

_____

Karen D. Weddle-West, Ph.D.
Vice Provost for Graduate Programs

Simulation of Abnormal/Normal Brain States Using the KIV Model

by

Mark Hebron Myers

A Dissertation

Submitted in Partial Fulfillment of the

Requirements for the Degree of

Doctor of Philosophy

Major: Computer Science

The University of Memphis

August 2011

Acknowledgments

I would like to thank Dr. Robert Kozma, my mentor, advisor, and teacher, for opening the door for me to the amazing universe of Computational Neurodynamics. The subject matter is vast, complex and wonderful. I am also thankful to Dr. Walter Freeman, MD who has provided me an insight into the field of neurology that has left me truly grateful and humbled. Dr. Charles Blaha, thank you for your insight into the area of electrical brain stimulation and how it can overcome the effects of brain pathologies. Thank you, Dr. Stan Franklin for your wisdom, patience and time. Thank you Dr. Max Garzon, a great professor and master thespian, for not only providing me with the fundamental tools of several disciplines, but for the high bar of excellence you demand from me and your students. Lastly, I would like to thank my family for their support and my parents who have always guided me towards the path of success.

To my advisor:
Give a man a fish and you feed him for a day. Teach a man to fish and you feed him for a lifetime.

- **Chinese Proverb**

To my dissertation committee:
Because of all your teachings, I stand on the shoulders of giants!

- **Sir Isaac Newton**

Lovers and madmen have such seething brains, such shaping fantasies that apprehend more than cool reason ever comprehends.

- **William Shakespeare, A Midsummer Night's Dream, Act V**

The IRB permit number is: E04-199, University of Memphis, June 8, 2004.

ABSTRACT

Myers, Mark Hebron. Ph.D. The University of Memphis. August 2011. Simulation of abnormal/normal brain states using the KIV model. Major Professor: Robert Kozma, Ph. D.

Recent studies have focused on the phenomena of abnormal electrical brain activity which may transition into a debilitating seizure state through the entrainment of large populations of neurons. Starting from the initial epileptogenisis of a small population of abnormally firing neurons, to the mobilization of mesoscopic neuron populations behaving in a synchronous manner, a model has been formulated that captures the initial epileptogenisis to the semi-periodic entrainment of distant neuron populations. The normal non-linear dynamic signal captured through EEG, moves into a semi-periodic state, which can be quantified as the seizure state. Capturing the asynchronous/synchronous behavior of the normal/pathological brain state will be discussed. This model will also demonstrate how electrical stimulation applied to the limbic system restores the seizure state of the brain back to its original normal condition.

Human brain states are modeled using a biologically inspired neural network, the KIV model. The KIV model exhibits the noisy, chaotic attributes found in the limbic system of brains of higher forms of organisms, and in its normal basal state, represents the homogeneous activity of millions of neuron activations. The KIV can exhibit the 'unbalanced state' of neural activity, whereas when a small cluster of abnormal firing neurons starts to exhibit periodic neural firings that eventually entrain all the neurons within the limbic system, the network has moved into the 'seizure' state. These attributes have been found in human EEG recordings and have been duplicated in this model of the brain.

The discussion in this dissertation covers the attributes found in human EEG data and models these attributes. Additionally, this model proposes a methodology to restore the modeled 'seizure' state, and by doing so, proposes a manner for external electrical titration to restore the abnormal seizure state back to a normal chaotic EEG signal state. Quantification measurements of normal, abnormal, and restoration to normal brain states will be exhibited using the following approaches:

- Analysis of human EEG data
- Quantification measurements of brain states.
- Development of models of the different brain states, i.e. fit parameters of the model on individual personal data/history.
- Implementation of quantitative measurements on "restored" simulated seizure state.

TABLE OF CONTENTS

LIST OF FIGURES

Figure **Page**

KEY TO SYMBOLS OR ABBREVIATIONS

AON……...……Anterior Olfactory Nucleus

EEG……...Electroencephalograph

IRB……..Institutional Review Board

IPSP…….……..…...Inhibitory Post-Synaptic Potential

OB…...……...Olfactory Bulb

PC……..…….Prepyriform Cortex

PSD……..Power Spectral Density

STD…….Standard Deviation

VNS…….……....Vagus Nerve Stimulator

CHAPTER 1: INTRODUCTION

Overview

In this dissertation we investigate quantitative measurements that demonstrate how the KIV (K-4) model can be used as a metaphor of the limbic system of the human brain. The brain states of normal/pathological (seizure)/restoration are modeled to further understand the pathological states of the brain and propose a titration therapy through this model.

Spectral analysis of high density EEG arrays from pediatric patients with intractable types of seizures, demonstrate a clear delineation of normal EEG behavior vs. abnormal behavior. The power spectral densities (PSD) of the EEGs have power-law distributions ($1/f^{\alpha}$) in log-base10 displays. Quantitative measurements of electroencephalograph (EEG) characteristics demonstrate that attributes of the normal, chaotically behaving action potentials of mesoscopic groups of neurons can be captured into a non-linear, biologically-inspired neural network model of the limbic system of the brain. This model can be used to study abnormal aspects of action potentials as seen in epileptic seizures when EEG output displays semi-periodic signals that synchronize throughout the cortex.

We model the dynamics of the limbic system through the KIV model, a biologically-motivated computational model of the brain, which can model the normal/abnormal attributes found in mesoscopic neural activity (Kozma & Freeman, 2003). Through internal and external weight adjustments of the nodes/neurons of the chaotic oscillators within the KIV, we can model the interactions of cortical-hippocampal

systems. In this manner, we can demonstrate how our EEG simulator can be used for external electrical conditioning of the brain to reduce abnormal action potentials effects found during the seizure state. During seizure events, organized, semi-periodic electrical discharges develop in the epileptogenic zone and spread, within seconds, over wide areas of the cerebral cortex. Seizure discharges typically last seconds to minutes and are followed by the normal non-linear, chaotic neuron behavior that is captured through an EEG.

In this dissertation, we analyze quantifiable temporal shifts in normal action potentials, as well as ictal synchronization of mesoscopic neuron populations. As normal EEG is enormously varied, manifesting qualitative changes depending on behavioral state, we initially develop methods to quantify brain states using EEG features. Next, we focus on the distinctive features of normal and abnormal states of the brain, respectively. We distinguish ictal behavior from non-ictal behavior, and accurately capture these states in the EEG simulator. Next, we demonstrate the capability of the KIV-based EEG simulator to incorporate electrical stimulation therapies that reduce the effects of the seizure state. In this manner, we demonstrate an EEG simulator that captures normal abnormal states of the limbic system, which can provide a means to test electrical stimulation techniques to restore the normal, chaotic electrical activity of the brain.

Structure of the Dissertation

This dissertation is separated into six parts. Chapter 2 reviews the background of seizure state attributes. It discusses how the interactions of large populations of neurons cause the debilitating seizure state of semi-periodic neuron oscillations.

Chapter 3 introduces quantitative measurement techniques for capturing the brain dynamics in human EEG. Power spectral density analysis, autocorrelation, and phase space analysis are discussed and utilized within the analysis process. The discussion of the analysis in this chapter provides the framework of experimentation with EEG signals in the subsequent chapters.

Chapter 4 provides the background of the KIV model used to simulate normal/abnormal brain EEG signal attributes. The biologically inspired neural network will be discussed to demonstrate how this model can be used as a metaphor for limbic system modeling.

Chapter 5 describes initial research of the underlying causes of petit-mal seizures and the application of this work into the KIII model. This research is used to emulate the epileptogenisis and incorporate this work into the broader KIV network.

Chapter 6 discusses applying the analysis methods discussed in chapter 3 to human EEG data. Quantification of normal/abnormal states is exhibited through analysis methods in order to capture EEG attributes. Additionally, these analysis methods are applied to the KIV model. Comparisons of human EEG/simulated EEG are demonstrated in this chapter.

Chapter 7 demonstrates how simulated seizure treatment of the seizure state is reduced through the application of an external stimulation titration method.

CHAPTER 2: EPILEPTIC SEIZURES

Pathophysiology of Seizure Characteristics

Hughlings Jackson (Jackson, 1878) proposed the classic definition of epileptic seizures as, "…excessive and disorderly discharge of nerve tissue." William Gowers suggested that the brain's constitution or formation was accompanied by an abnormal structure within the brain. The following figure demonstrates seizure activity recorded from an EEG through an electrode (Figure 1). This type of seizure known as localized or focal seizure occurs within a specific area of the brain.

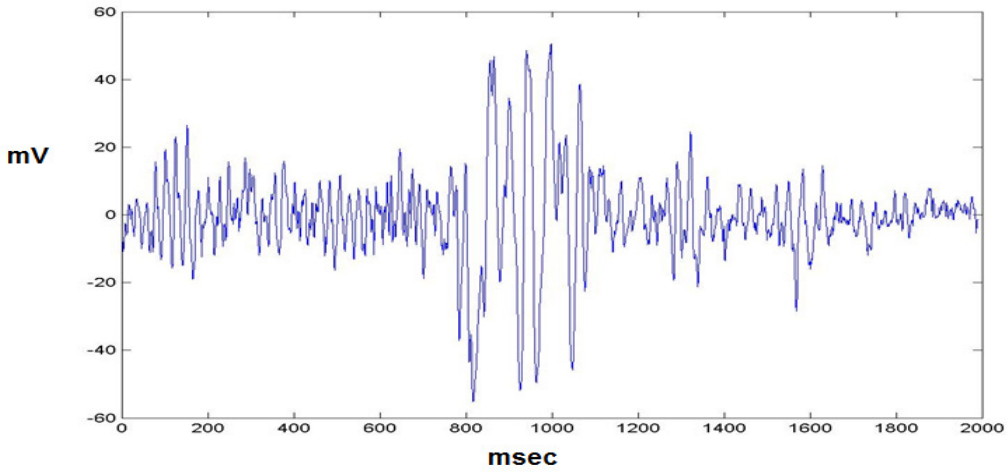


Figure 1. Localized (focal) Seizure.

Recording of electrical activity in the human brain was performed by Hans Berger, E.D. Adrian and B.H.C. Matthews. This technique was quickly applied to the analysis of epilepsy. Two types of seizures were discovered: generalized seizures and partial/focal.

Currently, the most common forms of generalized seizure are tonic/clonic convulsion without aura (grand-mal), petit-mal, and myoclonic jerk. For grand-mal seizures, the tonic phase involves muscle stiffness on both sides of the body, followed by

the clonic phase of muscle jerking (Figure 2). Petit-mal absence is characterized by an individual who loses awareness for a few seconds. EEG patterns consist of spike wave activity occurring at the rate of 3 cycles per second (Figure 3). Myoclonic jerk consists of very rapid symmetrical upward jerk of the arms and head and forward bend of the trunk.

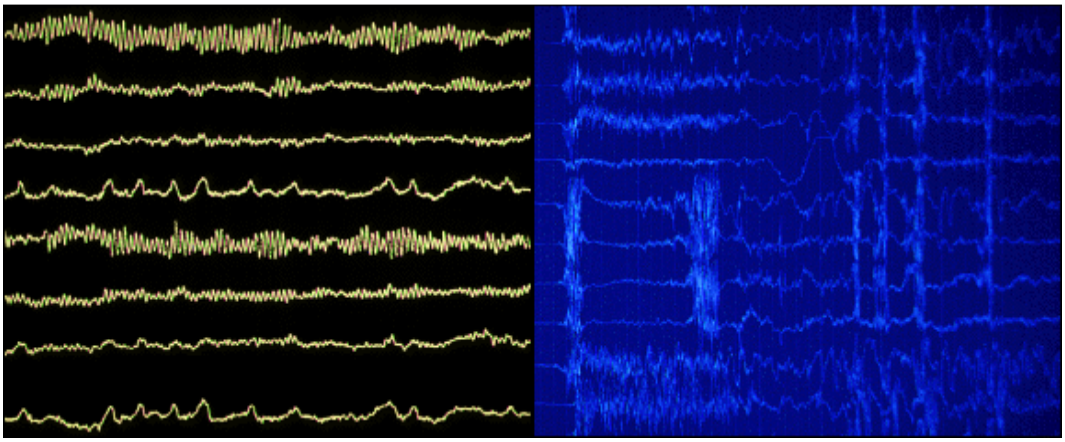


Figure 2. Examples of Grand-mal seizures

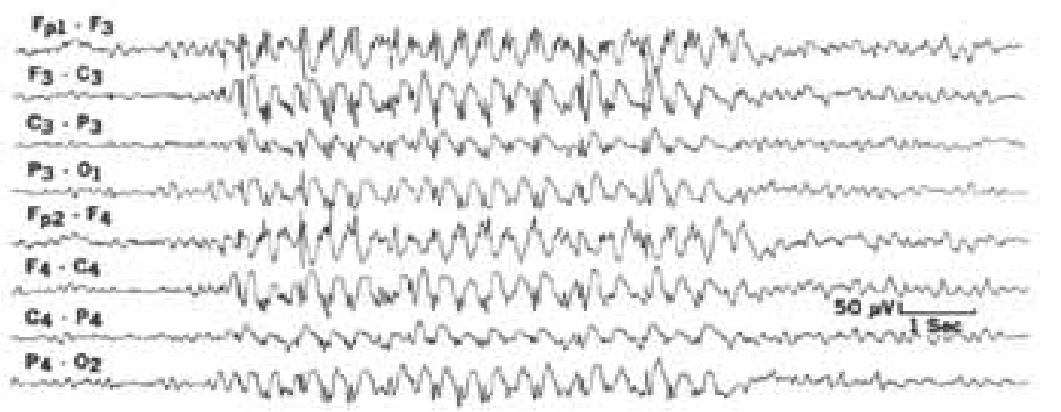


Figure 3. Example of Petit-mal seizure

When the seizure occurs in the motor cortex of the brain, jerking of the muscles occurs on the opposite part of the body, and may spread out to other opposite parts if the

seizure location becomes larger. This is known as a Jacksonian seizure. When electrical discharges pass through the Corpus Callosum to the other side of the hemisphere, consciousness is lost and convulsions occur all over the body. Temporal lobe seizures originate from discharges from the temporal lobe. This type of seizure commences with an alteration of the thought process and perception caused by the seizure. This is the aura. Surgery has been known to cure this type of seizure.

Generalized seizures are formed inherently throughout the structure of the brain where focal seizures involve a specific area of pathology, i.e., tumor, brain damage due to injury, etc. Temporal Lobe seizures may be due to a loss of neurons in the hippocampal area of the brain.

Seizure control involves medication treatment, surgery, diet, and for intractable types of seizures, the vagus nerve stimulation brain implant. The majority of seizure treatment involves medical treatment in the form of antiepileptic drugs where approximately half of the patients diagnosed with epilepsy become seizure-free on their first antiepileptic drug (AED) (Wheless, 2006). Where AEDs and surgery fail to alleviate the patient of the debilitating seizure effects, those living with refractory epilepsy perceive their condition as being life-limiting, which greatly reduces their overall quality of life and potentially leads to a lifetime of dependence on others.

Drug therapy does not always control seizures and can be associated with negative side effects. Additionally, only a minority of patients are candidates for epilepsy surgery. Vagus nerve stimulation (VNS) therapy, approved by the US FDA in 1997, is a treatment option that is effective in reducing refractory types of seizure frequency and severity as well as improving patient quality of life without the pharmacological side

effects associated with traditional antiepileptic drugs (Wheless & Baumgatrner, 2004; Wheless & Maggio, 2002). Vagus nerve stimulation generally refers to several different techniques used to stimulate the vagus nerve, including those in studies in animals where the vagus was accessed through the abdomen and diaphragm. The VNS system has been used a treatment for intractable types of seizures where medication and surgery have not worked. Vagus Nerve Stimulation may also refer to stimulation of the left cervical vagus nerve using a commercial device, the NCP System (Zabara, 1985).

A study featuring 50 patients, median age 13 years, with medically refractory epilepsy were implanted with VNS system. The study showed reductions in total seizures were 42% at one month, 58.2% at three months, and 57.9% at six months. The most common adverse events reported were voice alteration and coughing during stimulation. Other uncommon adverse events included increased drooling and behavioral changes (Frost, Gates, Helmers, Wheless, Levisohn, Tardo & Conry, 2001).

Epilepsy Experiments

In this section we review findings from open-loop stimulation epilepsy experiments in order to understand the effects of electrical stimulation modulation of the brain. As an example, epilepsy of hippocampal origin may enable an implanted electrode where stimulation can be applied chronically (Merill, 2005). In contrast, a noninvasive system may be used to induce antiepileptic changes in superficial cortex.

Clinical studies are often grouped by anatomical targets (Ellis & Stevens, 2008; Halpern, Samadani, Litt, Jaggi & Baltuch, 2008), but target-specific factors need to play a greater role in individualizing electrotherapy strategy and characterizing its mechanisms. Afferent connections throughout the different regions of the brain can influence the

dynamics of neuronal populations and sensitivity to electrical stimulation. As is true of deep brain stimulation for Parkinson's and functional electrical stimulation (FES), if axonal projections are excited in the vicinity of the electrode, the functional target of stimulation is determined by where those axons project, and not solely by the neuronal network near the stimulation electrode(s). Therefore, by analyzing the functional outcome of stimulation is to focus on induced changes in efferent activity in the context of global network function.

In electrotherapy of movement disorders, continuous stimulation during therapy is standard (Johnson, Miocinovic, McIntyre & Vitek, 2008) it is therefore noteworthy that in epilepsy therapy intermittent (ON-OFF) protocols are being explored (Andrade, Zumsteg, Hamani, Hodaie, Sarkissian, Lozano & Wennberg, 2006; Velasco, Velasco, Velasco, Jimenez, Marquez & Rise, 1995). The effects of open-loop stimulation are divided into ON effects, and OFF effects. The distinction between ON and OFF effects is made when the stimulation is intermittent (e.g., repeated patterns of 5 min on, 5 min off): the ON effects occur during a functional stimulation "unit" such as a pulse train, and the OFF effects are observed transiently after each ON functional unit ends.

In hyperexcitable tissue, brain slice studies have indicated whether DC fields suppress or aggravate seizures depends on neuronal geometry relative to the sign and direction of the applied fields (Sunderam, Gluckman, Reato & Bikson, 2010). (Richardson, Gluckman, Weinstein, Glosch, Moon, Gwinn, Gale & Schiff, 2003) demonstrated that a cylindrical electrode placed axially in the hippocampus, similar to the placement of a standard clinical hippocampal depth electrode, creates a radially expanding field that is aligned with the CA3 pyramidal cell neurons and can be used to

polarize them and thereby modulate their activity. Though this study was performed on animals, the finding has a direct clinical translation path. Very-low frequency stimulation through CNS-implanted electrodes has not been applied to humans, in part due to electrochemical safety concerns (Sunderam, Gluckman, Reato & Bikson, 2010).

Electrical Titration Approaches

Vagus nerve stimulation (VNS) devices result in a significant reduction of seizure frequency in many patients when antiepileptic drugs (AED) or resection of the seizure focus localized to one mesial temporal lobe cannot avail the effects of the seizure state (Karceski, 2007). Titration parameters, such as frequency, intensity, pulse width, and duration are used as external input into the KIV simulated limbic system in order to model the interaction of external stimulation into the cortex. The following VNS attributes were found in the literature. Low intensity trains of VNS (100 μA, 30 Hz, 500 μs, 20 second periods) have been found to hyperpolarize pyramidal neurons of rat parietal association cortex (Zagon & Kemeny, 2000). A range of values were utilized starting with the parameters: 1.5 to 5 V; pulse width 90 to 200 μs, stimulation frequency, 5 to 10 cps (Kerrigan et al., 2004). In another study, five patients treated with the VNS system using the parameters: output current 1mA, signal frequency 30 Hz, signal pulse width 500 μs, signal on for 30 seconds, showed increased cortical inhibition associated with stimulation without any evidence of cortical excitability (Marrosu, Santoni & Pulighedda, 2003). Additional experimental results suggested that the sensitivity of epileptic brain to VNS was different during the epileptic process, such that the inhibiting effect of VNS to seizure decreased as the development of seizures occurs over time (Yang, 2007).

An ideal form of electrical seizure control might be a single precisely tuned pulse that resets or annihilates a developing or ongoing seizure. Entrainment of interictal-like bursts with low frequency (1-5Hz) pulse trains in hippocampal slices was shown to interrupt the development of full electrographic seizures causing short-term synaptic depression of excitatory neurotransmission. But clinical tests of low-frequency pulse trains have had mixed results. It is well known that even a single pulse can trigger an electrographic discharge in hyperexcitable tissue (Lesser, 2008). It has also been proposed that high-frequency (50-100 Hz) stimulation can suppress, or at least shorten, an ongoing electrographic seizure. High-frequency stimulation trains may have distinct effects at stimulation onset versus after several minutes of ongoing stimulation, due to adaptive tissue mechanisms (e.g., potassium clearance). Specific high-frequency stimulation waveforms can trigger seizures. Notable is the classic and popular kindling seizure model elicited from high-frequency trains (Betram, 2007). It is clear from the preceding discussion that is not possible to define a unique stimulation protocol that is optimal for every situation, and therefore a patient-tailored stimulation protocol is needed (Osorio, Frei, Sunderam, Giftakis, Bhavaraju, Schaffner & Wilkinson, 2005; Sunderam, Gluckman, Reato & Bikson, 2010).

The EEG Data Set

A high density array of electrodes was placed onto the surface of the scalp of several neurosurgical patients with medically intractable epilepsy, who were a candidate for the Vagus Nerve Stimulator (VNS) surgical treatment. The data for this study was provided by Dr. James W. Wheless, MD, Professor and Chief of Pediatric Neurology and LeBonheur Chair in Pediatric Neurology, University of Tennessee Health Science Center,

and Dr. Don M. Tucker of Electrical Geodesics Inc., (EGI) through their high density EEG system.

Features of normal/abnormal brain activity have been monitored for 60-90 minutes. A high density 256-channel EEG system captured the action potentials of electrical brain activity. Action potentials are captured at a sampling rate of 250 points/second.

Figure 4 demonstrates the locations of normal and abnormal EEG behavior captured in an EEG. These sites will be used for comparative analysis. Figure 5 demonstrates interictal behavior, while Figure 42 magnifies the seizure event in Figure 5. In this display, we see the two disparate channels change from their respective non-linear dynamic state to a lock-step semi-periodic state.

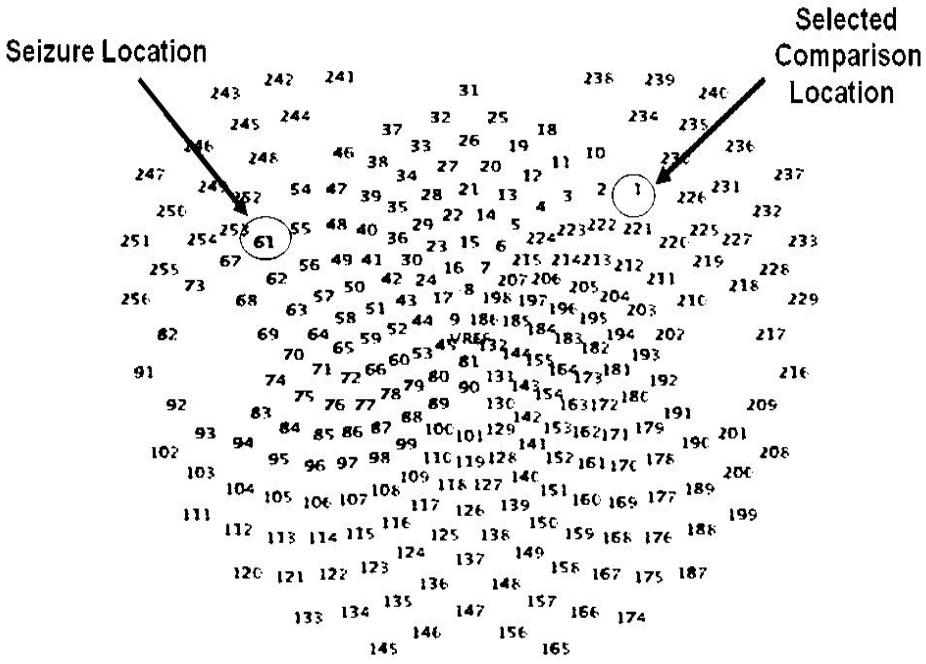


Figure 4. Electrical Geodesics Inc (EGI) 256-Channel EEG System. The display features comparison electrodes used for Phase Lock, Channel 1, reference signal. Channel 61, seizure location.

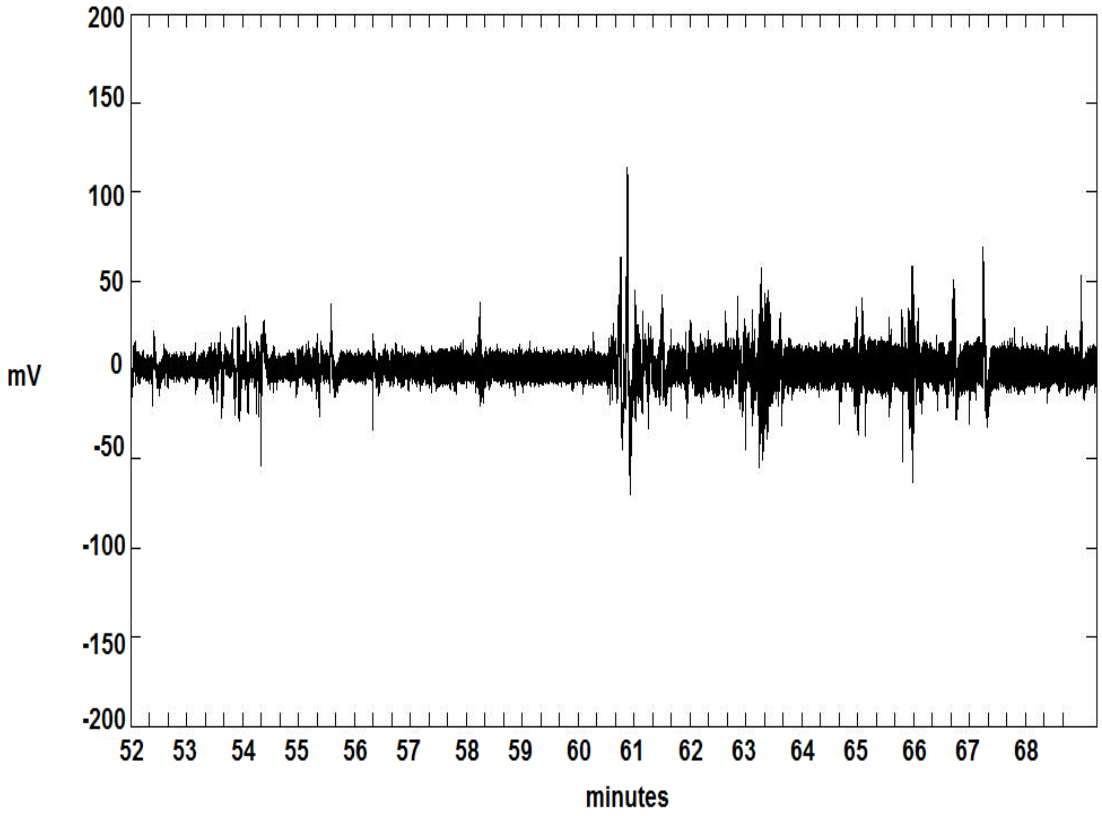


Figure 5. Synchronization between channel 61 and 1 indicates epileptogenisis starting at minute 63.

The analysis of the dataset was done using MATLAB software.

EEG Data Analysis

EEG recordings were captured via a high density 256-channel EGI system from four patients over a 60-90 minute time frame. Each seizure event was verified by the physician and tested against our algorithm. Pre-processing of EEG data consisted of finding the standard deviation (STD) of all 256 channels against each channel, and sorting the result set from highest deviation to lowest. In this manner, we can locate the

epileptogenisis or working channel. There may be several channels with high STD values that correlate to the locus of the seizure site. We also take the channel with lowest STD as our comparison or reference channel. These two channels will be used as input to our algorithm. Those EEG channels that correspond to more artifact behavior, i.e., electrode movement instead of seizure behavior, were discarded.

Pre-processing the datasets involved sorting out pre- and post- VNS data sets, and then separating each large dataset into 7.55 minutes of EEG recordings. This preprocessing of the EEG data enabled faster processing and management of the large datasets, since the EEG data for 256 channels for over 60 minutes would be roughly 4-5 Gigabytes of data. EEG data filtering was accomplished using a Remez filter within the 6- 12 Hz range (Theta – Alpha) in order to isolate the seizure occurrence.

CHAPTER 3: QUANTITATIVE MEASUREMENT TECHNIQUES

Neonatal Seizure Detection Methods – Autocorrelation Approach

There are several prevalent seizure detection methods that focus on the EEG signal properties found in the seizure state. The Gotman approach (Gotman, Flanagan, Zhang & Rosenblatt, 1997) involves breaking up the seizure signal found in infants into different types of rhythmic discharges that correspond to seizure/non-seizure areas. The Gotman detection method is based on the information available in the frequency spectrum of the newborn EEG, obtained through FFT. The Gotman method consists of finding a large peak at the main seizure frequency, accompanied by one or two other main frequencies, and little power elsewhere in the spectrum.

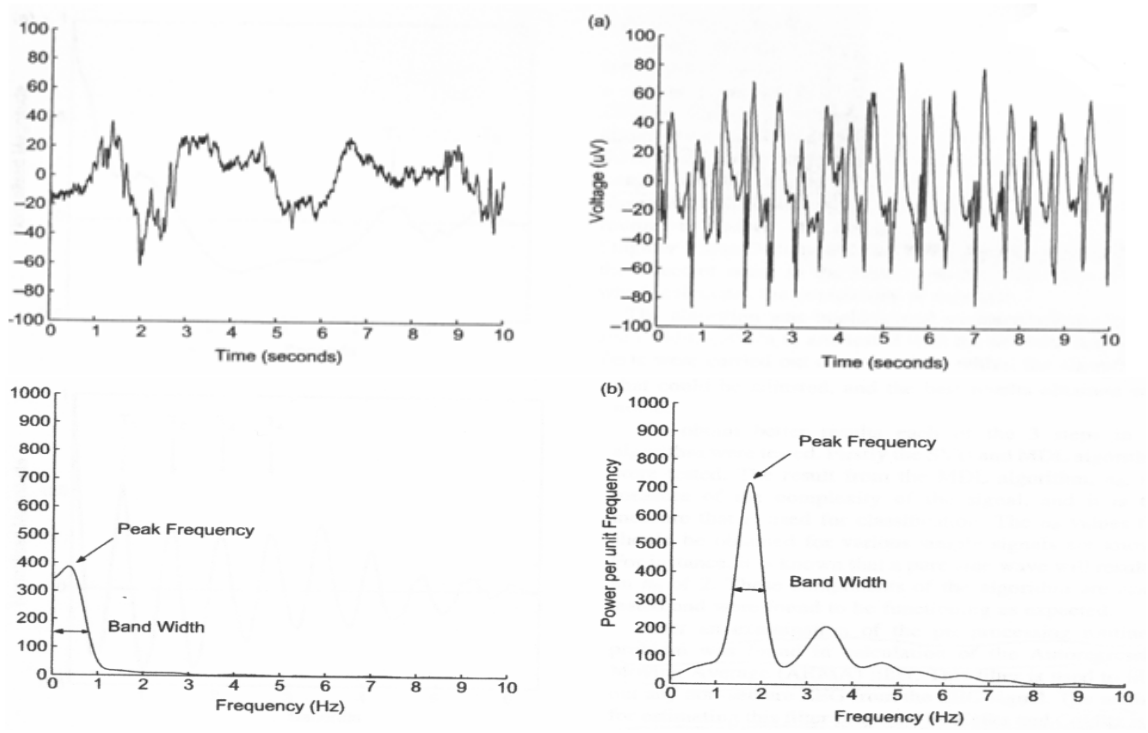


Figure 6. (Left side) Non-Seizure EEG (top), non-Seizure frequency spectrum (bottom). (Right side) Seizure EEG (top), seizure frequency spectrum (bottom).

The autocorrelation function is used to measure 'rhythmicity' or periodic areas found in the signal (Liu, Hahn, Heldt & Coen, 1992). Autocorrelation, the cross-correlation of a signal with a delayed version of itself, is useful for finding repeating patterns in a signal, such as determining the presence of periodic signals in complex signals. The center of each peak in the autocorrelation result is determined by its moment center; the point which halves the area between zero-crossings. Moment center ratios are calculated with respect to the autocorrelation result of the previous calculation. The closer these ratios are to integers, the higher the score that window of the EEG receives.

In the case of a repeating pattern, such as seizure spikes, the autocorrelation results in regularly spaced peaks and troughs as seen in Figure 7. During non-seizure times when the signal is not rhythmic, the autocorrelation results in irregular peaks and troughs (Faul, Boylan, Connolly, Marnane & Lightbody, 2005).

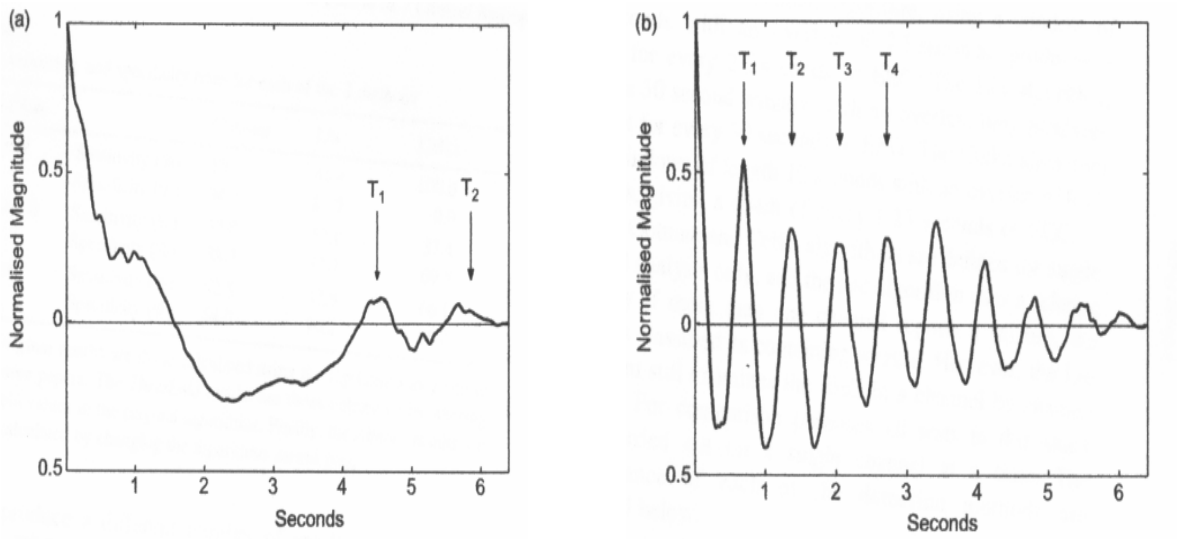


Figure 7. Autocorrelation result of non-seizure (a) and seizure (b) segments (Liu, Hahn, Heldt & Coen, 1992).

Regularity in peaks is the same as having a dominant frequency. In theory, the entropy of the autocorrelation function should be able to quantify how regular the peaks are and therefore the level of rhythmic in the original signal. Regularity in peaks is the same as having a dominant frequency. Autocorrelation is performed on 1 minute windows of EEG data and 16 second windows of simulated EEGs. Low-pass filtering using the Remez filter for filtering 6-12 Hz of the EEG and simulated EEG was performed in order to locate semi-periodic behavior in the signals.

Capturing Non-Linear Dynamic Properties of Epileptic Seizures.

The central concept consists of the idea that seizures represent transitions from a chaotic (preictal) state to an abnormal/periodic (ictal) state and back to a normal (postical) state. Very large biological populations have the potential for chaotic dynamics. Most biological systems are too complex to be easily understood. Largest lyapunov exponents are used to measure chaos-to-order-to chaos transitions (Wolf, Swift, Swinney & Vastano, 1985). Lyapunov exponents will determine if points in a system will eventually diverge, converge or become a neutral fixed point. This number, called the Lyapunov exponent 'λ', is useful for distinguishing among the various types of orbits. It works for discrete as well as continuous systems. When $\lambda > 0$, the orbit is unstable and chaotic. The points in a time series, plotted as phase space diagrams that represent an orbit of the traversing points, no matter how close, will diverge to any arbitrary separation. The points and their time offset values are said to be unstable. This does not preclude any organization as a pattern may emerge. As λ approaches 0, the orbit is a neutral fixed point (or an eventually fixed point). A Lyapunov exponent of zero indicates that the system is in some sort of steady state mode. A physical system with this

exponent is conservative. Take the case of two identical simple harmonic oscillators with different amplitudes. Because the frequency is independent of the amplitude, a phase portrait of the two oscillators would be a pair of concentric circles.

The advantage to using the Lyapunov exponent's method allows us to use a small representation of the population data (~15000 data points). The disadvantage is that results depend critically on the choice of the initial points from the dataset.

The following figure displays a patient's EEG which captures pre-ictal, ictal, and post-ictal states over the time series covering 62-65 minutes.

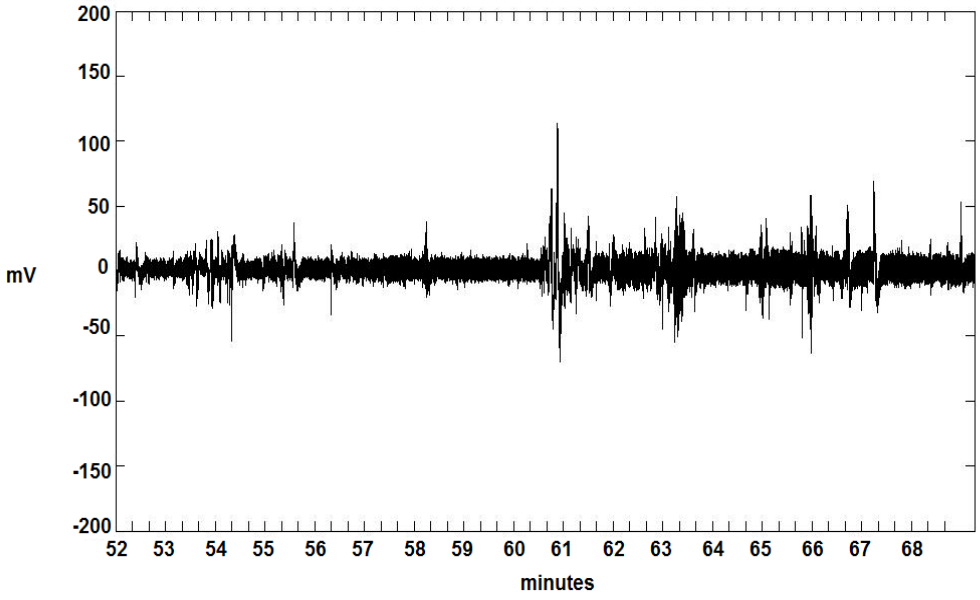


Figure 8. Channels 61 (black) and 1 (grey). Channel 1 is the reference electrode and 61 is the working electrode.

The following diagrams depict EEG time series captured at 250 samples per second for 1 minute intervals. Figures 9, 10, and 11, demonstrate phase space displays of pre-ictal, ictal, and post-ictal behavior, respectively.

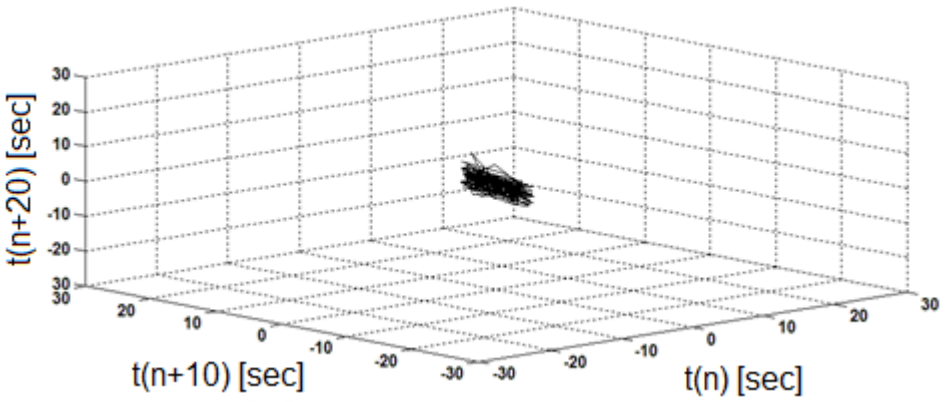


Figure 9. Time Series: 62 – 63 minutes – pre-ictal time series.

Lyapunov $\lambda = 0.2274$

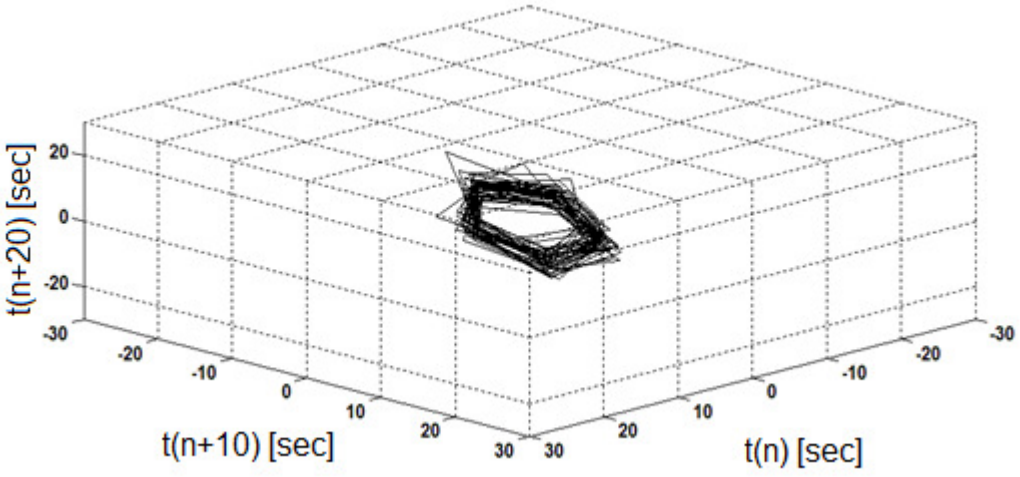


Figure 10. Time Series: 63 – 64 minutes - ictal time series.

Lyapunov $\lambda = 0.0772$

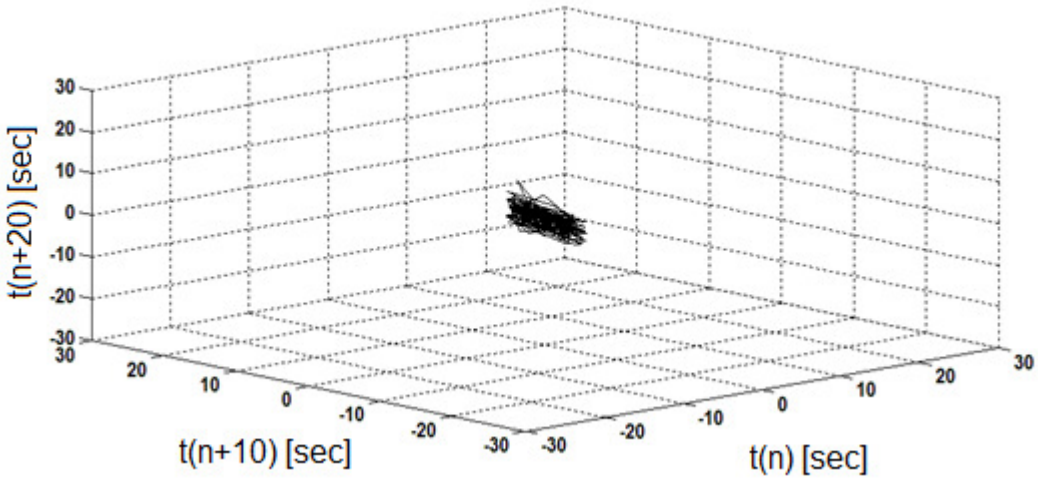


Figure 11. Time Series: 64- 65 minutes – post-ictal time series.
Lyapunov λ = 0.1383

Figure 10 displays phase space that occurs during the ictal period of the time series. The Lyapunov λ value is dramatically decreased compared to the previous λ values and rise again as seen in Figure 11 when the time series moves into a post-ictal state.

Phase Space Analysis

Phase Space representations of the normal and abnormal states of seizure patient's EEGs were taken in order to demonstrate limit cycle and chaotic behavior. Nonlinear analyses have suggested that the seizure state may represent a transition from the normal EEG state to one of increased synchronous activity and that a more orderly state characterizes an epileptic seizure (Velazquez, Cortez, Carter & Wennberg, 2003). Phase diagrams will demonstrate normal/abnormal brain states in order to display the normal EEG in the form of a chaotic attractor vs. the abnormal seizure EEG in the form of a limit cycle. Plots of the EEG time series against a fixed time offset reveal the dynamics of the

EEG signal as the time series may converge, diverge or move in a period motion (Freeman, 1986). Time series offset values (n) were plotted for normal and seizure states, where the x, y, and z axis were plotted for EEG and simulated EEG analysis. The selected offsets enable a clearer depiction of the chaotic attractors found in the normal EEG time series vs. the semi-periodic behavior found in the EEG of the seizure state.

Power Spectral Density Analysis

Power spectral density (PSD), or energy spectral density, is a general concept applied to a signal which may have physical dimensions such as power per Hz, or energy per Hz, etc. It is often called simply the spectrum of the signal. Due to the identified spatio-temporal dynamics of EEG signals, the PSD often exhibits a linearly decreasing behavior over log10 coordinates considering frequency and amplitude of PSD or spectral power. This is called in the literature "power law" or scale-free behavior ($1/f^{\alpha}$), where cognitive processing states varied by the calculated PSD linear regression values $-\alpha$. Power law relation seen in Figure 12 spans the whole range of frequencies from 1 Hz to 100 Hz. Power law behavior is attributed to the brain structural connectivity and dynamical properties.

The temporal power spectral densities (PSD) in coordinates of log power vs. log10 frequency often conform to power-law distributions ($1/f^{\alpha}$) with exponents α varying between 0 and 4 for human EEGs, as displayed in the following Figure 12. (Freeman, Holmes, West & Vanhatalo, 2006b).

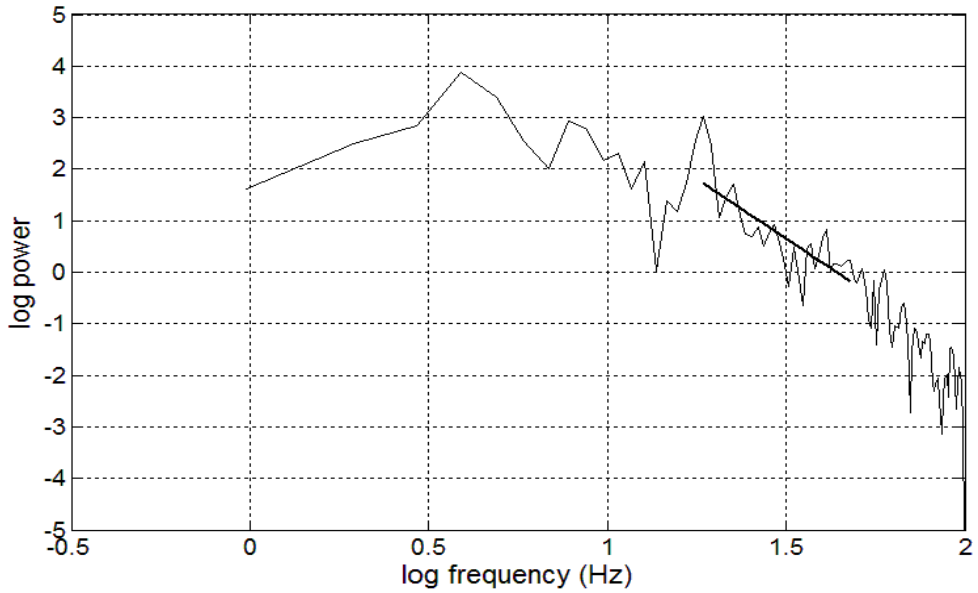


Figure 12. The slope value corresponds to the "power law" or scale-free behavior (1/fα), where cognitive processing states varied by the slope -α. Alpha values of human EEGs during awake and sleep states have been found to be within -2 to -3 (Freeman, 2006a). EEG figures featuring PSD values for EEG data where alpha = -2.16

The power spectral density (PSD) may often conform to a power-law distribution, by taking the linear regression of the PSD in coordinates of log power vs. log frequency (Freeman & Erwin, 2008). These findings are explained with a model of the neural source of the background activity in mutual excitation among pyramidal cells. The dendritic response of a population of interactive excitatory neurons to an impulse input is a rapid exponential rise and a slow exponential decay, which can be fitted with the sum of two exponential terms. When that function is convolved as the kernel with pulses from a Poisson process and summed, the resulting "brown" ($1/f^2$) or "black " ($1/f^3$) noise conforms to the EEG time series and the PSD in rest and sleep states (Freeman & Zhai,

2009). Linear regression ranges are selected within self-stabilized neural beta-gamma range (20-50 Hz) based on analysis of human EEG normal/abnormal states.

Variations in the observed slope are attributed to changes in the level of the background activity that is homeostatically regulated by the refractory periods of the excitatory neurons. Departures in behavior from rest and sleep to action are accompanied by local peaks in the PSD, which manifest emergent nonrandom structure in the EEG. Awake and sleep states will have slope values within 2 and 3 in the beta-gamma range (Freeman & Zhai, 2009).

The stability of the normal background activity is governed by a non-zero point attractor, by which the level is homeostatically regulated. The PSD values involve a range of values where PSD values are initially flat, then decay rapidly throughout the rest of the PSD display. The location of the inflection frequency in PSD of EEG between 1 Hz and 10 Hz, or delta-theta-alpha ranges, indicates typical values between 0.1/s and 10/s for prevailing steady state input levels (Freeman & Zhai, 2009). This is verified by simulating the PSD of the evoked activity in Figure 13 which show the inflection point as located above 10 Hz.

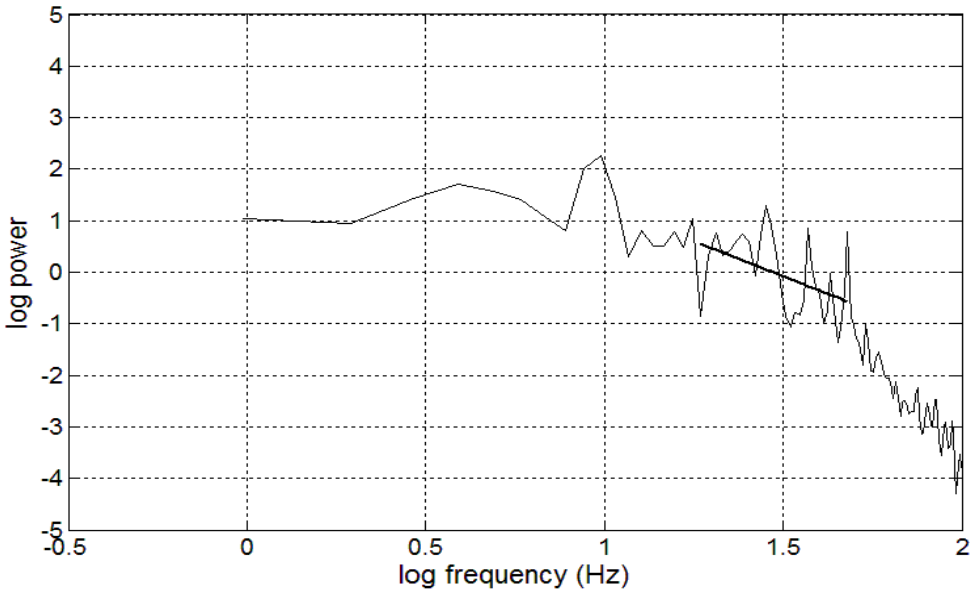


Figure 13. Simulated EEG of normal background activity, where alpha = -2.73.

The PSD of the states of sleep, awake, seizure, and VNS dampened seizure conform to the power law relation, $1/f^{-\alpha}$, where $\alpha = -3$ when the patient is at rest, $\alpha = -3$ to -2 during the awake state, $\alpha = -2$ during cognitive tasks, whereas when $\alpha = -2$, we have what's referred to as brown noise, $1/f^2$, which is found to the PSD of the EEG in animals engaged in intentional behaviors (Freeman, 2006). This change also appears in pre-ictal EEG, so it might offer a clue to understanding the mechanisms of complex partial seizures and predicting their onsets (Freeman, Holmes, West & Vanhatalo, 2006). The calculated PSD linear regression values during the seizure state approaches the theoretical limit of -4 (Freeman & Zhai, 2009).

Entrainment of Mesoscopic Neuron Populations

Another approach to analyzing the seizure state is to view seizures as the result of the progressive recruitment of brain sites in an abnormal hyper-synchronization. Seizures

appear to be bifurcations of a neural network that involves progressive coupling of the focus with normal brain sites. Seizure activity is characterized by recurrent, short-term electrical discharges of the cerebral cortex that result in intermittent disturbances of brain function (Sackellares, Iasemidis, Shiau, Gilmore & Roper, 2000; Schelter et al., 2006). The state between seizures, known as interictal behavior, appears to have minor spiking activity. In seizures of focal onset (focal seizures, partial seizures), the anatomical distribution of the interictal spikes varies, but spikes tend to occur most commonly in the epileptogenic zone and its connections. During the seizure, organized, semi-periodic electrical discharges develop in the epileptogenic zone and spread, within seconds, over widespread areas of cerebral cortex. Seizure discharges typically last seconds to minutes and is followed by the normal non-linear, chaotic neuron behavior that is captured through an EEG.

Network of Coupled Oscillators

Brain models have been created that exhibit epileptic seizures through a network of coupled oscillators that produce seizure like behavior by coupling periodic/abnormal oscillators to the rest of the network. These models have been able to exhibit the disruption of the Correlation-Synchronization-Entrainment patterns observed prior to seizures (Tsakalis, Chakravarthy & Iasemidis, 2005). This model is inspired by Freeman, Kozma, and Werbos (2001). Initially, Rossler-like oscillators with each oscillator i (i = 1, . . ., N) are constructed through the following equations:

$$\frac{dx_i}{dt} = -\omega_i y_i - z_i + b_i + \sum_{\substack{j=1 \\ j \neq i}}^{N} (\varepsilon_{i,j}(x_j - x_i) + u_{i,j}^{I} \qquad (1)$$

$$\frac{dy_i}{dt} = -\omega_i x_i - \alpha y_i, \frac{dz_i}{dt} = \beta_i x_i + z_i(x_i - \gamma_i) \quad (2)$$

where the intrinsic parameters α, β, γ, ω are chosen in the chaotic regime, e.g., 0.4, 0.33, 5, 0.95, respectively. b_i is a small constant bias term, different for each oscillator, which ensures that the origin is not an equilibrium point (in our examples, b_i's have "random" values in [−0.2, 0.2]). □, □' are the generally asymmetric coupling strengths; in this example, □ = □'. When the □ between two oscillators increases, their dynamical behaviors synchronize until they become nearly identical at high values of □. The value, u_{ij} produces internal feedback to the system, PI (Proportional Integral) compensates for network oscillator coupling:

$$u_{i,j}^I = k_{i,j}(x_j - x_i), k_{i,j} = PI_I\{\rho_{i,j} - c_*\} \quad (3)$$

and c_* is a threshold parameter (here taken as $c_* = 0.1$). The PI_I notation signifies that the considered PI feedback is part of the internal network of the "brain". The estimation of the correlation is performed in an exponentially weighted fashion in order to simplify the model's simulation:

$$\rho_{i,j} = m^2_{x_i x_j} / (m_{x_i x_i} m_{x_j x_j}) \quad (4)$$

The author used a time constant of 200 sec (a = 0.005) for his simulations.

$$\dot{m}_{x_i x_j}(t) = -am_{x_i x_j}(t) + ax_i(t)x_j(t) \quad (5)$$

These equations form the following Brain Network of coupled oscillators. The connecting lines indicate non-zero coupling between neighboring oscillators as seen in Figure 13.

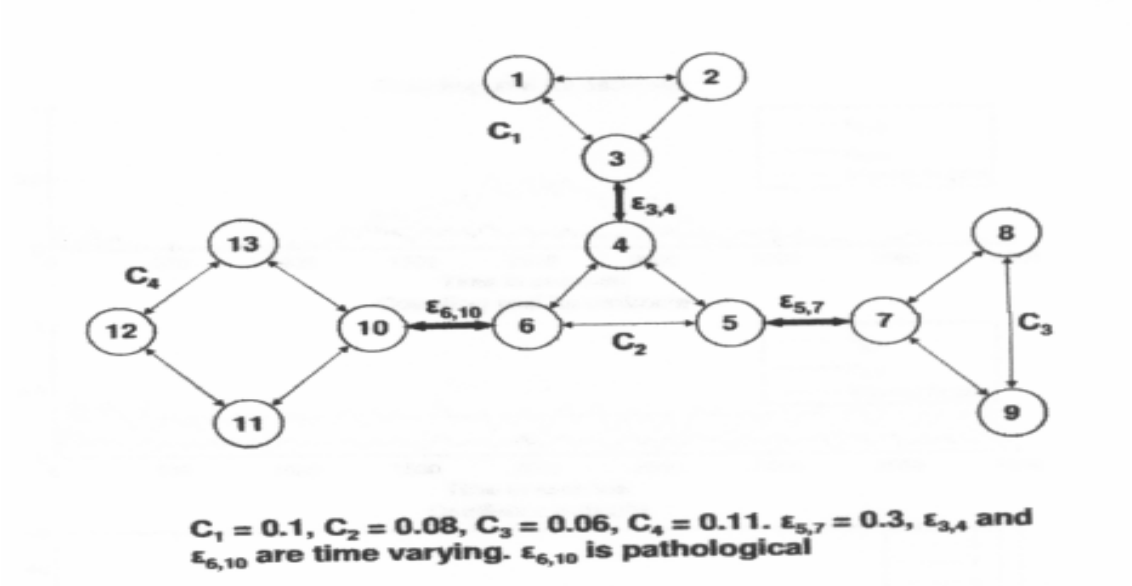


Figure 13. Brain Emulator as a Network of Coupled Oscillators

A sufficient PI value should be able to decorrelate adjoining oscillators in the network and preserve the chaotic behavior of the network. If several oscillators overcome the destabilizing behavior of PI, the attempts to cancel the effects of excessive diffusive coupling ε will not be reduced within the network and the network will turn "epileptic" (Figure 14).

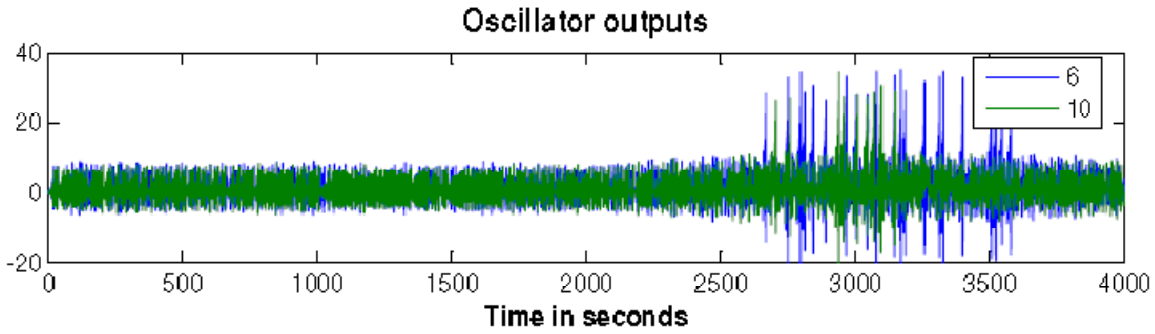


Figure 14. Internal feedback is reduced 10% from that of the "normal" brain (Tsakalis, Chakravarthy & Iasemidis, 2005).

Adding additional feedback control mechanism to the previous equations enables "shock" therapy modeling in order to disrupt the signal from its seizure state. A stimulation method is modeled to aid the PI value in decoupling the oscillators. Chaos is restored (Figure 15)! The Decoupling Control, which is inspired from adaptive control, records the output of the normal state of the brain to maintain the output of the signal through continuous monitoring of the signal.

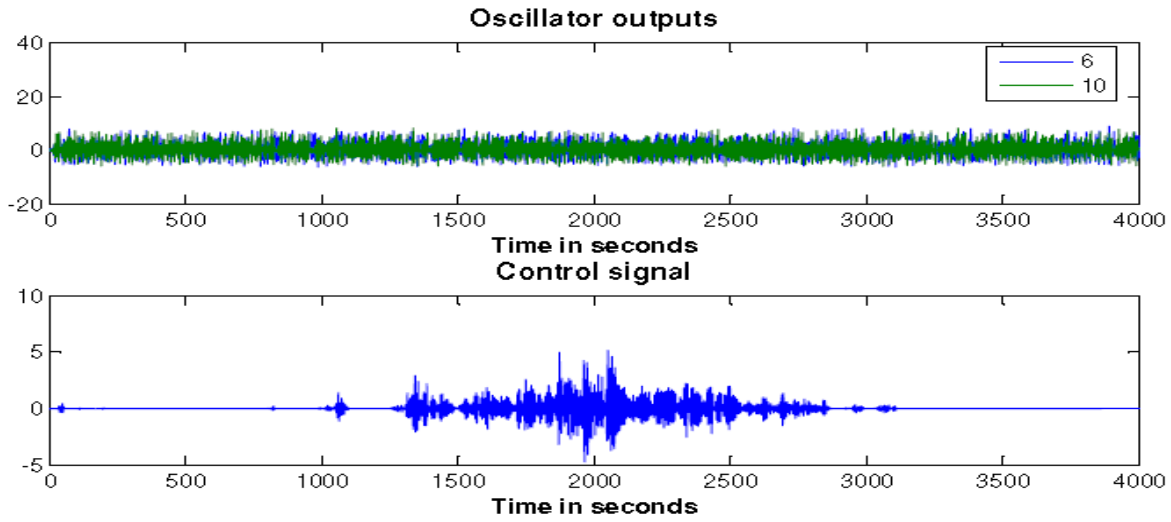


Figure 15. Top display features 'restored' time series from seizure state. The bottom display features the control input signal to block the entrainment of periodic oscillators to control the overall network (Tsakalis, Chakravarthy & Iasemidis, 2005).

Additional analysis of the seizure prediction/classification algorithms is needed in order to compare the strengths/weaknesses of each approach. Additionally, we utilize the KIV model which Tsakalis based his chaotic network approach in order to model normal/abnormal brain states.

CHAPTER 4: EEG SIMULATION USING THE KIV MODEL

Nonlinear dynamics is the framework that is currently being pursued to explain the brain processes. Models based on Dynamic theory have been applied to several biological systems and it provides a better explanation of the phenomenon compared to the traditional connectionist and computational approaches (Bak, 1996; Van Gelder, 1995). Nonlinear Dynamic theory applied to brain modeling is not a completely new approach, but an existing approach in a new theoretical framework. It belongs to the new generation of the biologically plausible and dynamical connectionist models (Kozma, 2007). A model based on this approach, the K models are explained in section 'K Models Overview' of this chapter. The parameterization of the model to simulate EEG signals is discussed in Chapter 6, 'KIV Parameterization'.

EEG analysis done by Freeman, Pritchard, Duke and others, characterizes the brain as a chaotic, nonlinear dynamic system. These properties help explain some of the underlying mechanisms of the brain (Pritchard & Duke, 1995; Skarda & Freeman, 1987). The following section provides an explanation of the nonlinear dynamics in the brain.

Brain as a Chaotic Nonlinear Dynamic System

This section provides an explanation of why the brain is characterized as a "chaotic nonlinear dynamic system".

Brain as a Dynamic System: A dynamic system is one whose state changes over time and is governed by an evolution rule, which captures the way the system changes with time. At a high level, the brain can be characterized as a system having different states like wake, sleep, cognitive, and pathological states. Intuitively, one can say that the

brain changes from one state to another over time. So the brain can be characterized as a dynamic system. The temperal evolutionary rule(s) can be described using differential equations. The behavior of a dynamic system such as the brain can be explained using attractors (Stam, 2005). Basic types of dynamic behavior include fixed point, limit cycle, and strange (chaotic) attractor, examples of which are shown in Figures 18, 21, and 23, respectively.

Nonlinearity in the Brain: The support for characterization of the brain as a nonlinear system can be gleaned from the analysis of brain activity captured by EEG signals. On a more general level, the complete functionality of the brain cannot be obtained by breaking it down into smaller parts and then combining the solution of each of those parts, which is a defining feature of nonlinear systems.

Chaos in Brain: A dynamic approach to understanding the brain seems intuitive, whereas chaos in the brain is far from intuitive. Nonlinearity and chaos go hand in hand and linear systems are never chaotic. Although there is much debate as to the precise description of chaos, a widely accepted property of a chaotic system is defined as "*sensitivity to initial conditions"* (Strogatz, 2000), which means that the slightest change to the initial settings of a system, over time will lead to a result that is far from the result obtained with original settings. The notion that the brain is sensitive to initial conditions flies in the face of the existing knowledge about the brain. The human brain is extremely robust. As an example, humans can recognize a face, even if is slightly different from what they have seen the first time.

Chaos in the brain doesn't fit the usual definition and is not caused by sensitivity to initial conditions. What then, is the source of chaos in the brain? The underlying

mechanisms of the brain are not purely ordered like a computer, neither is it purely random (noisy). It is somewhere in between, made up of some structure and randomness (Erdi, 2008), which is why the brain is said to be on the "edge of chaos". This property of the brain enables it to be robust and adapt to its environment. The randomness inherent in the brain is what makes it chaotic, not the sensitivity to initial conditions (Freeman, 2000b). Skarda and Freeman proposed that the group behavior of the neurons produces chaos, which is essential for carrying out all the functions of the brain such as information processing, creation of memory and knowledge generation (Skarda & Freeman, 1987).

K Models Overview

In this section, the K models, which are used to model the chaotic and nonlinear dynamic nature of the brain, are described. The K models are biologically based hierarchical models which were designed by Dr. Freeman. The hierarchy consists of K0, KI, KII, KIII, and KIV in order of increasing complexity and are described in the following subsections. This model addresses some of the disadvantages of the traditional connectionist models.

K0 set. *The interactions between groups of neurons produce the dynamics seen in the brain. Although an individual neuron is an indispensible component of the brain, the functioning of a single neuron is not enough to explain the complex behavior produced by the brain. Freeman, Kozma and others believe that the complex behavior of the brain can be expalined by the interations of the mesoscopic neuron populations. Hence the basic unit in K models, namely the K0 set, is not a single neuron, but* represents a mesoscopic

neuron population (~104 neurons). This is one of the major differences between traditional connectionist models and the K models.

The dynamics of the K0 set are governed by a point attractor. This means that, over time, a non-interacting population of neurons will converge to the point zero. The K0 set is the basic unit of the K models, upon which the rest of the hierarchy is based. Figure 16a shows the K0 set and Figure 18 is an example of a point attractor landscape.

The dynamics of the K0 set are given by the following second order ODE (ordinary differential equation):

$$\frac{1}{ab}\left[\ \frac{d^2 p(t)}{dt^2} + (a+b)\frac{dp(t)}{dt} + abp(t)\ \right] = X(t) + I(t) \tag{6}$$

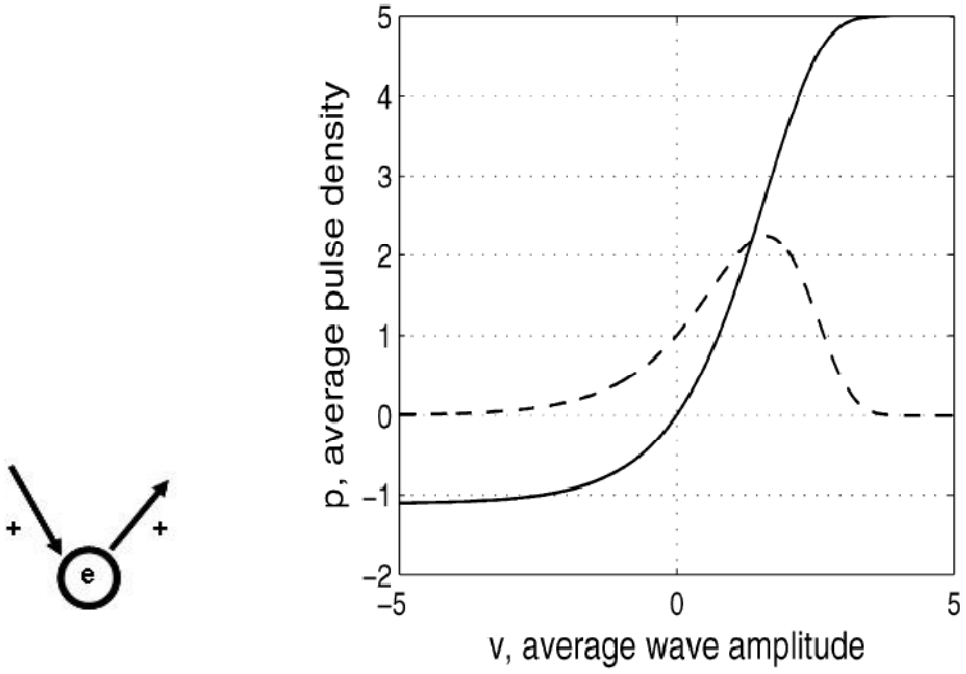
where 'a' and 'b' are time constants determined based on physiological experiments, p(t) is the pulse density at time t and X(t) is the external input and I(t) is the external input at time t.

Dendrites convert the axonal pulse they receive into a wave, which is later converted back into a pulse by the axon. This pulse-wave and wave-pulse is an essential function of the neurons and neuron populations. The pulse to wave transfer function is assumed to be linear, and the wave-pulse transfer function is nonlinear and is given by the following equation (Ilin & Kozma, 2006):

$$p = Q(v) = q_m\left(1 - \exp\frac{1 - \exp\ v}{q_m}\right) \tag{7}$$

where v is the average wave density, p is the average pulse density and q_m is a constant whose value is between 1 and 14. The asymmetric nature of equation 2 changes for different values of q_m. Figure 16b displays the output of equation 7 for $q_m = 5$. The

circle with an ‘e’ is the K0 set or node and ‘e’ stands for excitatory population. The arrow pointing toward the node is the input (received by the dendrites of a neuron) to the K0 set and the arrow pointing away from the node is the output axonal pulse of the K0 set. ‘+’ is used to indicate the input to the K0 set is excitatory and the effect of the output of the K0 population is also excitatory in nature. The asymmetricity of the function depends on the value of the constant q_m.



a) b)

Figure 16. K0 set and nonlinear transfer function. a) The K0 set represents a population of neurons. b) The nonlinear transfer function Q(v) given by Eq. (7) (solid line) and its derivative (dashed line). The maximum of the derivative is shifted to the positive value of the average wave amplitude; $q_m = 5$. (Ilin & Kozma, 2006).

The conversion from pulse density at excitatory synapses is expressed by a positive gain coefficient k_e, and that at inhibitory synapses is expressed by a negative gain coefficient k_i. The action of modulatory neurons is dependent on k_e or k_i. The conversion of the sum of current density is described by a static nonlinear equation (7) in the form of a sigmoid curve that has a single parameter Q_m specifying the slope and maximal asymptote of the curve (Freeman, 1979). The shape of the curve Q_m is determined experimentally from the pulse probability of cortical neurons (Eeckman & Freeman, 1991; Freeman, 1975), Figure 17.

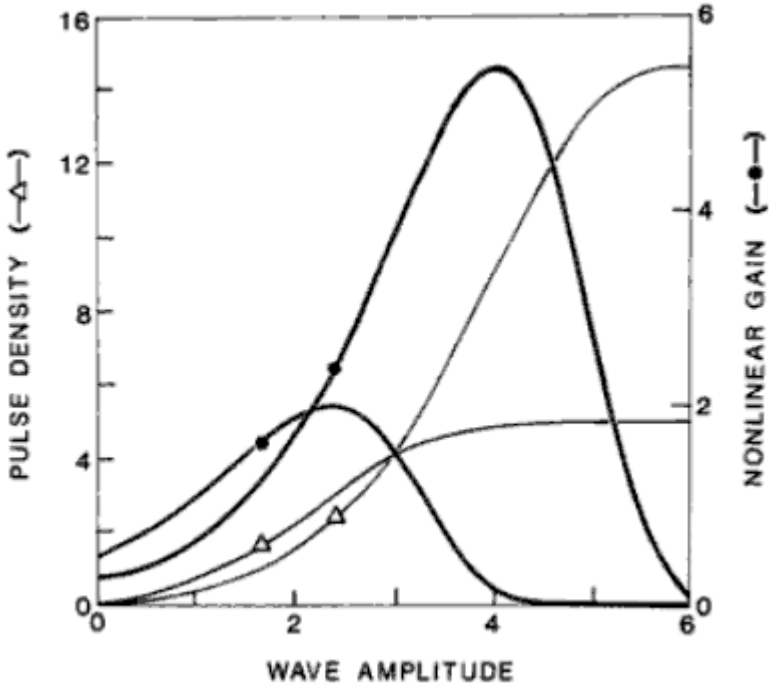


Figure 17. The biologically derived sigmoid curve realting pulse density, p, to wave density, v (light trace) is asymmetric about its rest point (triangle). The steepest point of the slope, dp/dv (dark trace), lies to the excitatory side of the rest point. The steepness is determined be a parameter Q_m, where curves are shown for $Q_m = 2$ in a quiescent subject

and Q_m = 6 in an aroused, motivated subject. Avarage and maximal values in the bulb and pyriform cortex are 5 and 12 respectively (Eeckman, & Freeman, 1991).

KI set. The KI is made up of two neuron populations which are either excitatory (KI_E) or inhibitory population (KI_I). The two populations are connected to each other and a numeric value called weight is assigned to each connection. The dynamics of KI are governed by a non-zero point attractor (Kozma, Aghazarian, Huntsberger, Tunstel & Freeman, 2007; Kozma & Freeman, 2003) (Figure 18).

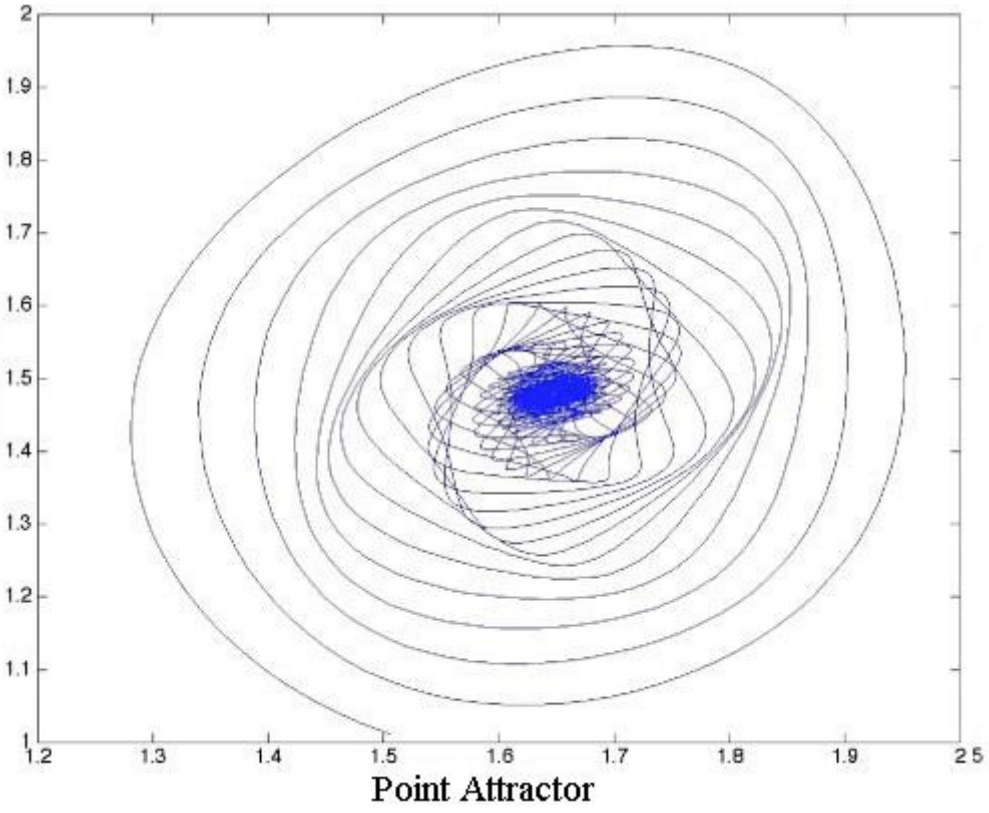
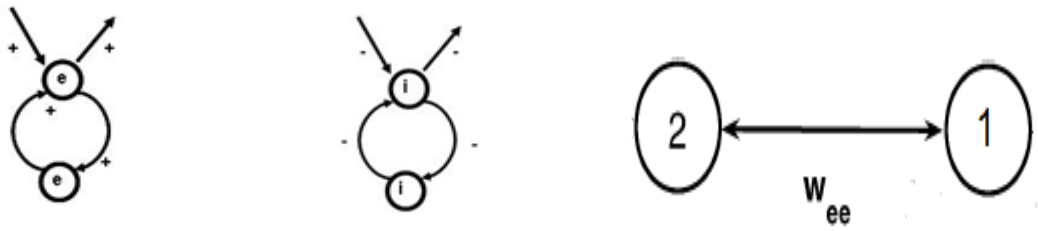


Figure 18. Point Attractor, one of the basic attractor types found in dynamical systems. (Kozma, 2003).

Figures 19 shows the KI_E and KI_I populations respectively. Feedback is introduced in the KI sets. KI sets are also governed by point attractors. The Point Attractor is one of the basic attractor types found in dynamical systems. The K0 and KI are governed by this type of attractor. K0 always converges to the point zero, whereas KI converges to a zero or non-zero point depending on the parameterization of KI (Kozma, 2003). There is only one type of interaction in the KI sets and the feedback is also either exitatory or inhibitory. Figure 17a displays excitatory KI populations and inhibitory KI populations which are constructed by two interacting neuron populations. Generally, the top nodes receive the input and output is also read for the top node. Feedback is introduced in the KI sets.



a) b)

Figure 19. a) Excitatory KI population and Inhibitory KI population are made up of two interacting neuron populations. Simple excitatory KIe set with two K0 units. KIe has a single parameter w_{ee} which represents the level of mutual excitation within the neural population b).

A simple case of two interacting excitatory K0 sets are shown in Fig. 17b. We assume that the pulse density of the first K0 set is transformed into the wave density by the nonlinear function Q(v) (Eq. (7)) and again into the pulse density by the linear

function, represented by the weight w_{ee}. The same holds for the second K0 set. The dynamics is given by the following two second order ODEs:

$$\ddot{y}_1 + \alpha\dot{y}_1 + \beta y_1 = \beta w_{ee}\, Q(y_2), \qquad \ddot{y}_2 + \alpha\dot{y}_2 + \beta y_2 = \beta w_{ee}\, Q(y_1). \tag{8}$$

Here $\alpha = a+b$, and $\beta = ab$. The equilibrium can be found by setting the derivatives to zero, which gives the following equations for the equilibrium values of y_1 and y_2:

$$y_1^* = w_{ee}\, Q(y_2^*), \qquad y_2^* = w_{ee}\, Q(y_1^*). \tag{9}$$

We characterize the equilibria of Eq. (9) depending on gain parameter w_{ee}. For small values of w_{ee}, there is a unique stable zero equilibrium. As we increase w_{ee}, we reach a value w_{LP} when a limit point (LP) appears. The zero equilibrium is stable until a threshold value of $w_{ee} = w_{BP}$, which is a bifurcation point (transcritical bifurcation). Above w_{BP}, the zero equilibrium becomes unstable (Ilin & Kozma, 2006).

KII set. *KII models the interactions between excitatory and inhibitory populations and is made up of interconnected KIE's and KII's (Figure 20). The feedback between the excitatory and inhibitory components produces periodic oscillations which are governed by a limit cycle attractor (Freeman, 2000a; Kozma, Aghazarian, Huntsberger, Tunstela & Freeman, 2007) (Figure 21).*

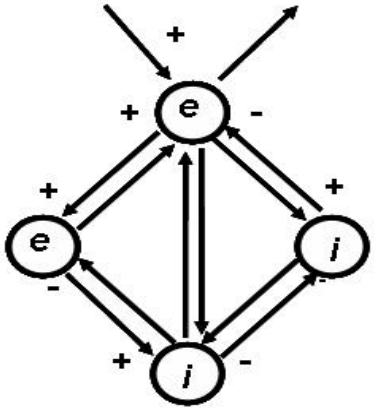


Figure 20. KII sets contain both excitatory and inhibitory feedback, which produces oscillatory behavior.

There are four types of interactions (connections) that are possible in KII sets, namely excitatory-excitatory, excitatory-inhibitory, inhibitory-excitatory and inhibitory-inhibitory. Each connection is assigned a weight. Hence, interactions withn a KII are described by four ODE's.

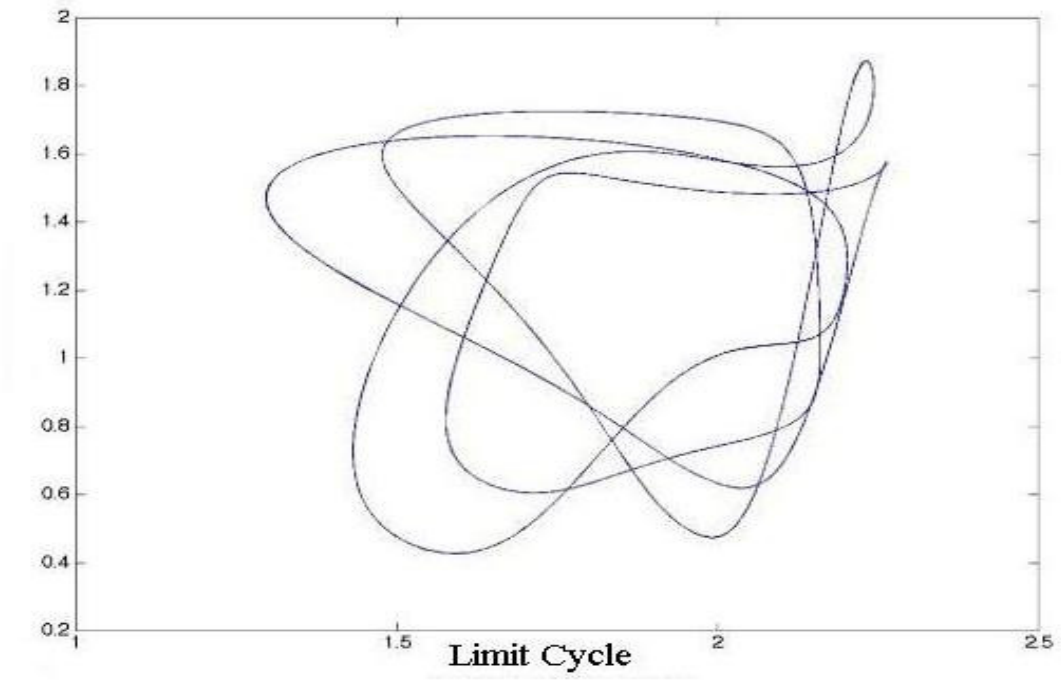


Figure 21. The Limit Cycle, one of the basic attractors types found in dynamical systems (Kozma, 2003).

KIII set. Multiple KII's are connected together to produce a KIII set. Delayed feedback connections are introduced in the KIII sets. The KIII set models sensory systems in the brain such as visual cortex, olfactory systems, and other brain regions. KIII architecture is based on the cortical colums and layers found in the brain. Each column can interact with other columns and the interactions between columns is through lateral connections (blue and pink lines in Figure 22). Figure 22 shows a general architecture of the KIII set. The architecture can be changed by changing the number of layers (rows) and columns and also by changing the connectivity between them. The dashed lines in Figure 22 represent the delayed-feedback loops.

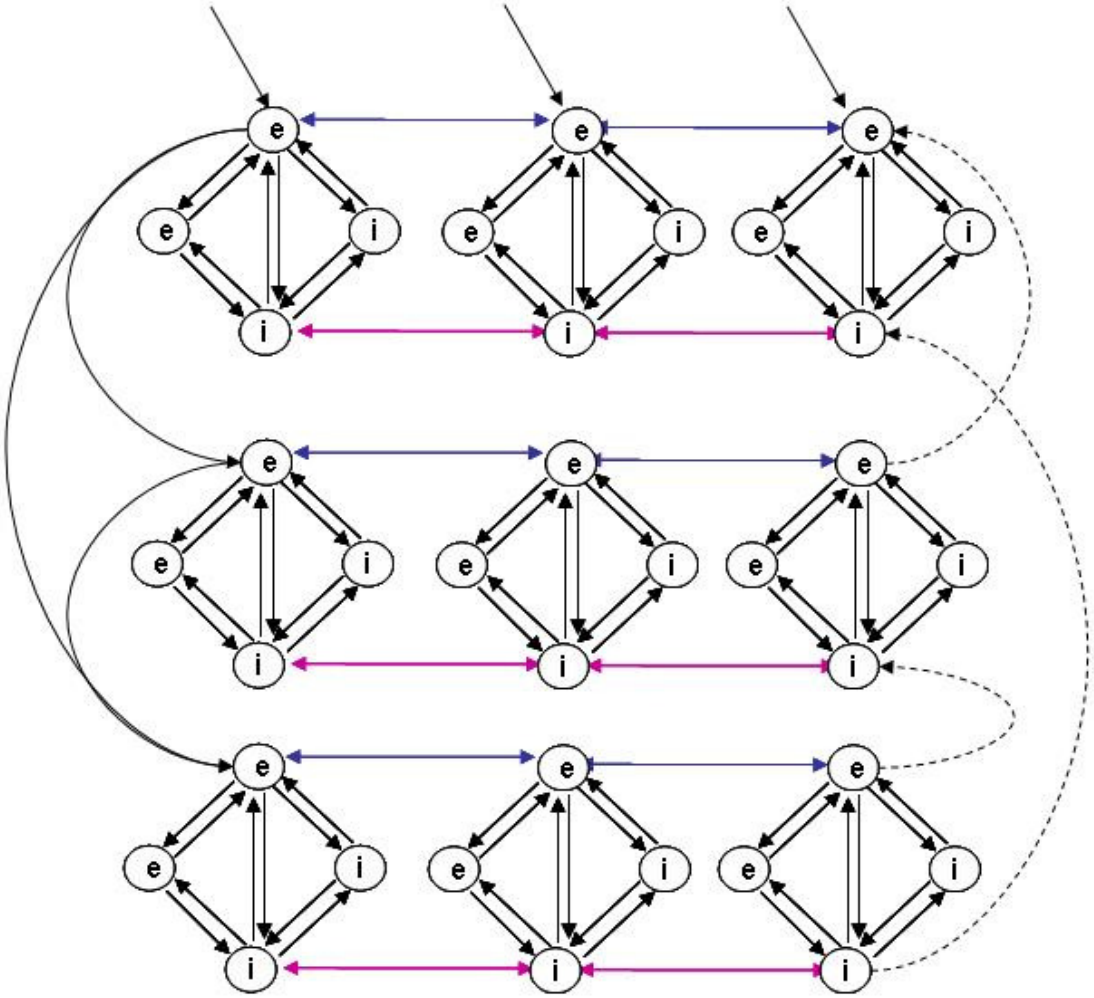


Figure 22. KIII sets. Delayed feedback connections (dashed lines) and lateral weights (blue and pink lines) are introduced in the KIII sets (Kozma, & Freeman, 2003).

KIII sets are governed by a chaotic attractor, which are produced by the interactions between multiple oscillating KII components (Kozma, & Freeman, 2003); see Figure. 23. KIII is used to model various sensory systems in the brain such as the olfactory, visual, and auditory system.

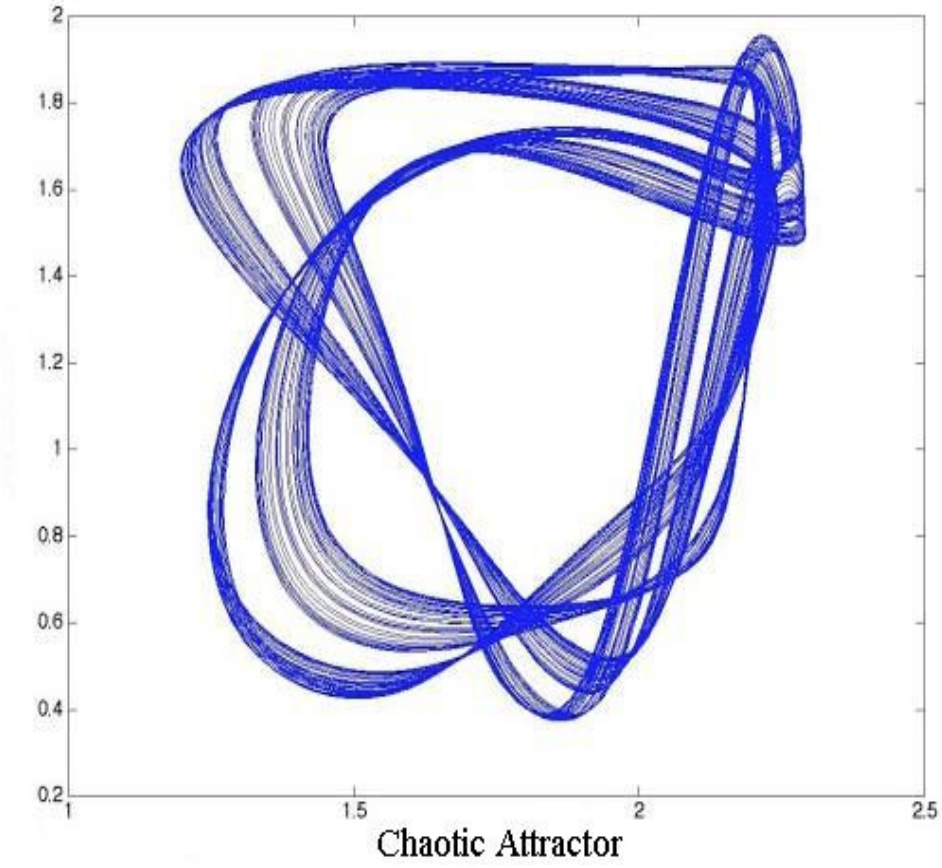


Figure 23. Chaotic Attractor, yet another type of basic attractor type found in dynamical systems. KIII's and KIV's are governed by chaotic attractors, which is also the kind of dynamics observed in healthy brains (Kozma, 2003).

KIV model. The interactions between the various sensory systems are modeled in KIV sets. The KIV consists of multiple KIII's, one for each sensory system needed in the model and a KII or KIII that interacts with each of the KIII's. Knowledge, memory and intentional behavior can be modeled using the KIV (Freeman, 2000; Kozma, Aghazarian, Huntsberger, Tunstela & Freeman, 2007).

The KIV used in this dissertation has two KIII's and a KII. One KIII is for the somatosensory cortex and the other for the hippocampus. The KII is used to model the entorhinal cortex/amygdala. In the human brain, the cortex processes sensory input, while the hippocampus plays an essential part in memory, learning and navigation. The cortex is also associated with several brain disorders such as epilepsy, schizophrenia, Alzheimers etc. One of the functions of the amygdala is to integrate the sensory and spatial information with the internal motivation/needs to reach a decision. It is also essential for long term memory formation.

The architecture of the KIV is shown in Figure 24. The KIII's architecture consists of three layers. Apart from the lateral weights between the KII's within each KIII, lateral connections, WA, WB and WC are introduced, which connect the three components somatosensory cortex, hippocampus and entorhinal cortex/amygdala of the KIV as shown in Figure 24. The signals from the somatosensory cortex KIII and the hippocampus KIII are integrated by the KII entorhinal cortex/amygdala (Kozma & Freeman, 2003; Myers & Kozma, 2007). The KIV used in this dissertation models the dynamics of somatosensory cortex, hippocampus and entorhinal cortex/amygdala, in order to simulate the electrical activity EEG during pathological states of the brain, as seen in Figure 25.

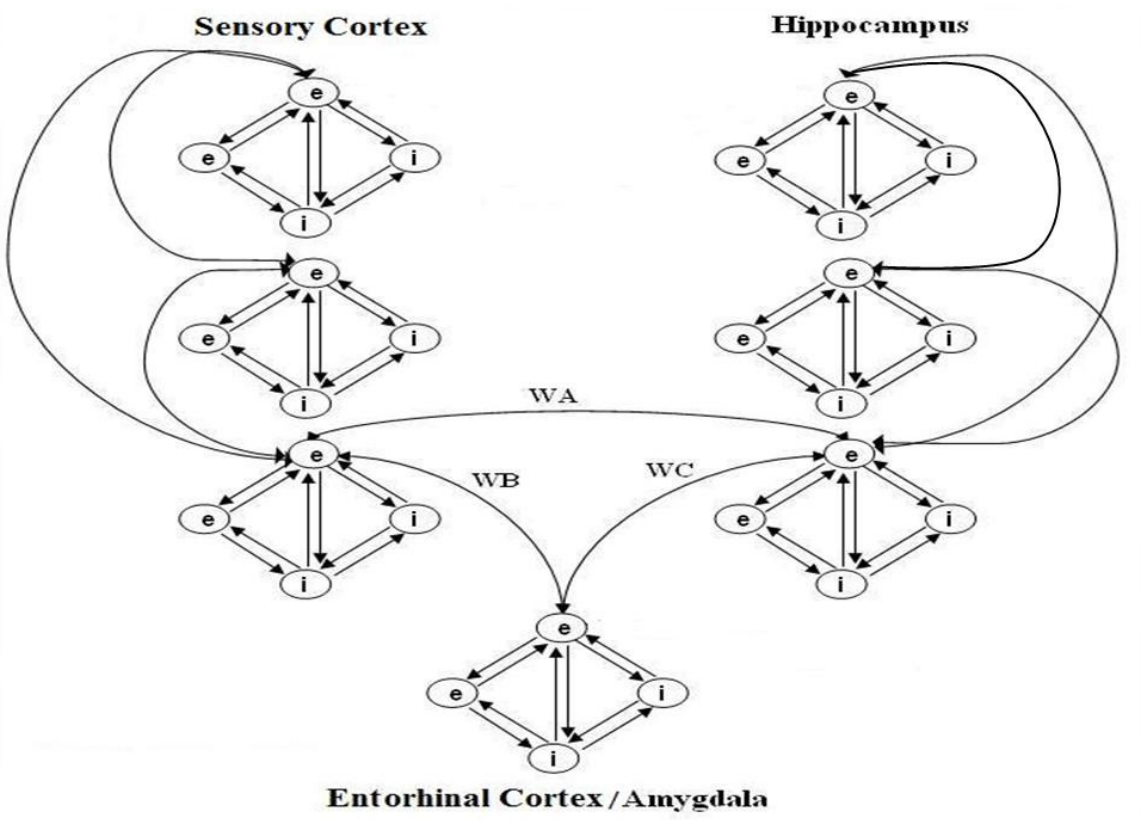


Figure 24. The KIV model of the limbic system, consisting of two KIII's and one KII (Myers et al, 2008).

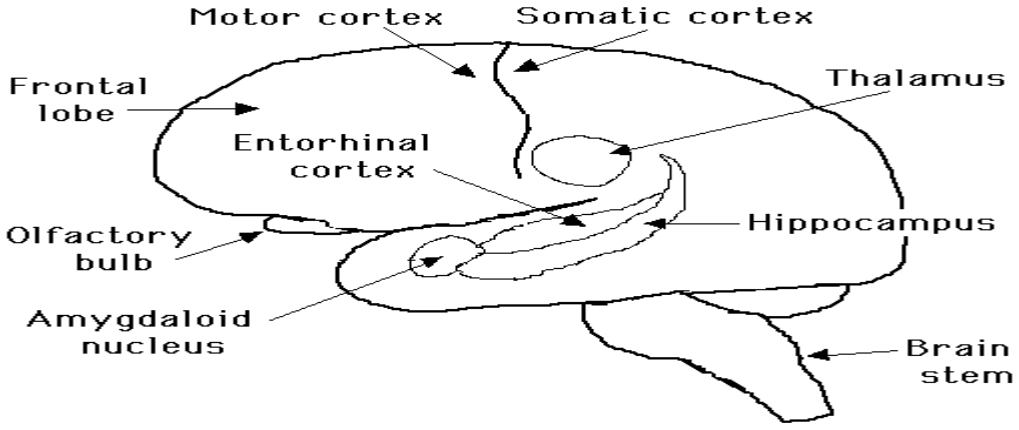


Figure 25. Schematic illustration of the components of the limbic system

There are three layers (L1, L2, L3) of KIIs in both KIII networks. There is one layer in the KII network (L1). The top KII nodes of the 3rd layer of both KIII networks and the top KII nodes of the KII network are shown to illustrate how the 3 networks are updated, Figure 26.

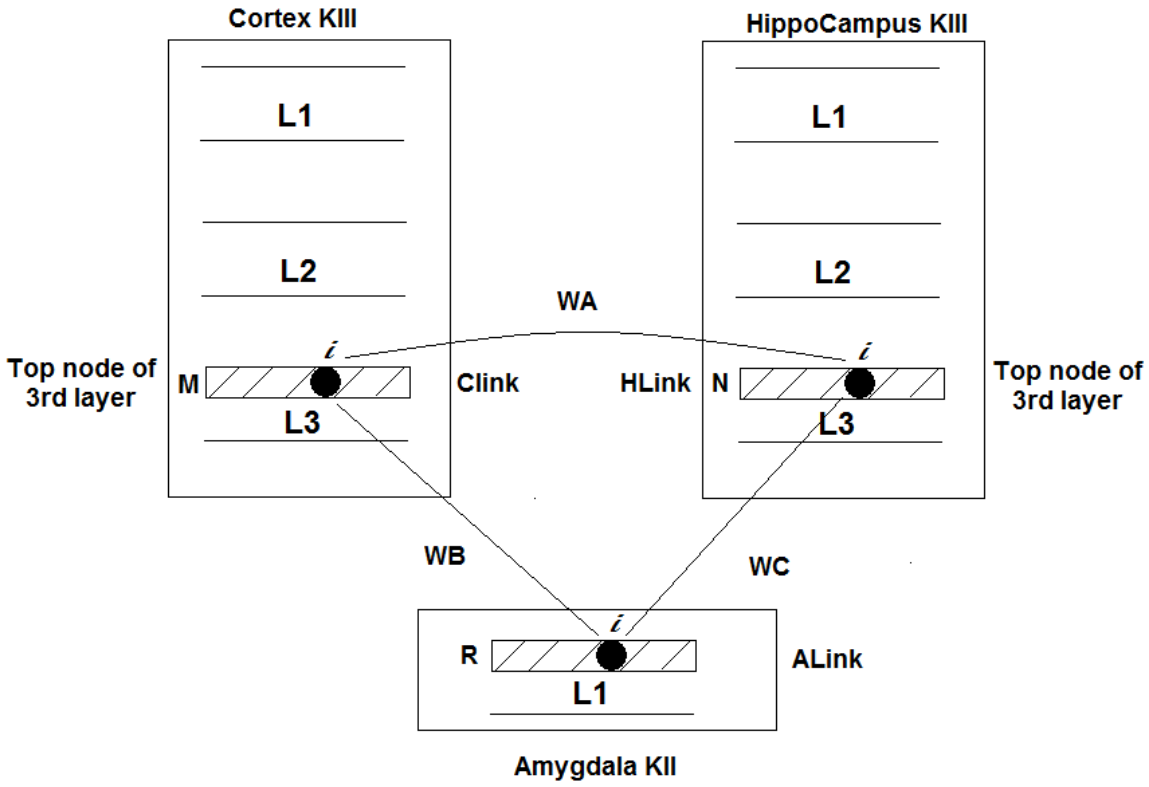


Figure 26. The following KIV schematic features how KIII and KII networks are updated throughout the KIV network.

Clink, HLink, and ALink consists of a matrix of (12, 1) top KII nodes. The values M, N, R equal the matrix size of 12. The term '*i*' refers to the '*i*th' node in the 3 matrices. WA, WB, and WC are external weight values. The KIV network features matrices to be the same size, therefore the weight values are the same, such as WA=WB=WC=0.325. For each time step each node in the KIII 3rd layer is calculated, and then all the nodes in KIII

are updated. The KII network is updated by simple matrix addition between the KIII network, multiplied by the external weights, WB and WC, i.e.:

$$\text{ALink} = (\text{WB} \times \text{CLink}) + (\text{WC} \times \text{HLink}) \quad (9)$$

$$\text{HLink} = (\text{WA} \times \text{CLink}) + (\text{WC} \times \text{ALink}) \quad (10)$$

$$\text{Clink} = (\text{WA} \times \text{HLink}) + (\text{WB} \times \text{ALink}) \quad (11)$$

All three matrices update the nodes in the KIV network via the 2nd ODE.

The external weights are bidirectional, as indicated by the double arrow heads in Figure 24, with the same value and therefore enable feedback between the three networks. The following table lists the parameters of the external and internal weight of the KIV.

Table 1

Network of Coupled Oscillators constants for KIV and Tsakilis networks

WA= WB= WC	KIII parameters (Hippocampus)	KIII parameters (Sensory Cortex)	KII parameters (Entorhinal Cortex/ Amygdala)	KIII Feed-forward weights	KIII Delayed Feed-back weights	KIV 2nd ODE constant	Pulse wave density values	Tsakilis 2nd ODE constants
0.325	kii1=[0.50, 2.20, 2.20, 2.50] kii2=[0.06, 2.25, 2.23, 2.40] kii3=[0.30, 2.30, 2.00, 2.36]	kii4=[0.05, 2.20, 2.20, 2.50] kii5=[0.06, 2.25, 2.23, 2.50] kii6=[0.30, 2.00, 2.10, 2.25]	kii=[2.00, 1.50, 2.00, 1.00]	[0.30, 0.50, 0.50]	[0.60, -0.50, 0.50]	a= .22 b=.72	Qm= 1-14	α=0.40 β=0.33 γ=5.00 ω=0.95

KIII parameterization, including feed-forward and delayed feedback connection weights that join KII objects within the KIII, enables normal background activations to

occur in a chaotic manner. KII parameters provide a periodic signal output, until external weighting (WA/WB/WC) between the two KIII objects and KII object bring the system to output a chaotic signal. KIV 2nd ODE rate constant values, a and b from Eq. 6 have been determined experimentally a = 0.22 ms−1 and b = 0.72 ms−1 (Kozma & Freeman, 2003). The value of constant q_m varies between 1 and 14 for different types of neural populations and for different states of the animal in sleep or being awake and motivated (Ilin & Kozma, 2006). Tsakalis 2nd ODE constant values from Eq, 1-2 (Tsakalis, Chakravarthy & Iasemidis, 2005) have been determined to enable their Rossler oscillator construction to maintain a chaotic state of output, as opposed to the KIV rate constant values which enable the KIV to produce stronger frequency ranges around 40 Hz. A schematic of KIV parameterization values are displayed in Figure 27.

The KIV is a chaotic dynamic memory model which encodes sensory information in the form of periodic spatial-temporal oscillations of non-linear processing elements. The amygdala is linked with both KIIIs as shown in Figure 27 with some weighted matrix. With proper weight selection the KIV neural network can maintain some non-convergent chaotic oscillations among all the components of the system. The activations from Hippocampus and Cortical KIII's are transferred between them through connection weight matrix WA. The activations between Hippocampus and Amygdala are passed

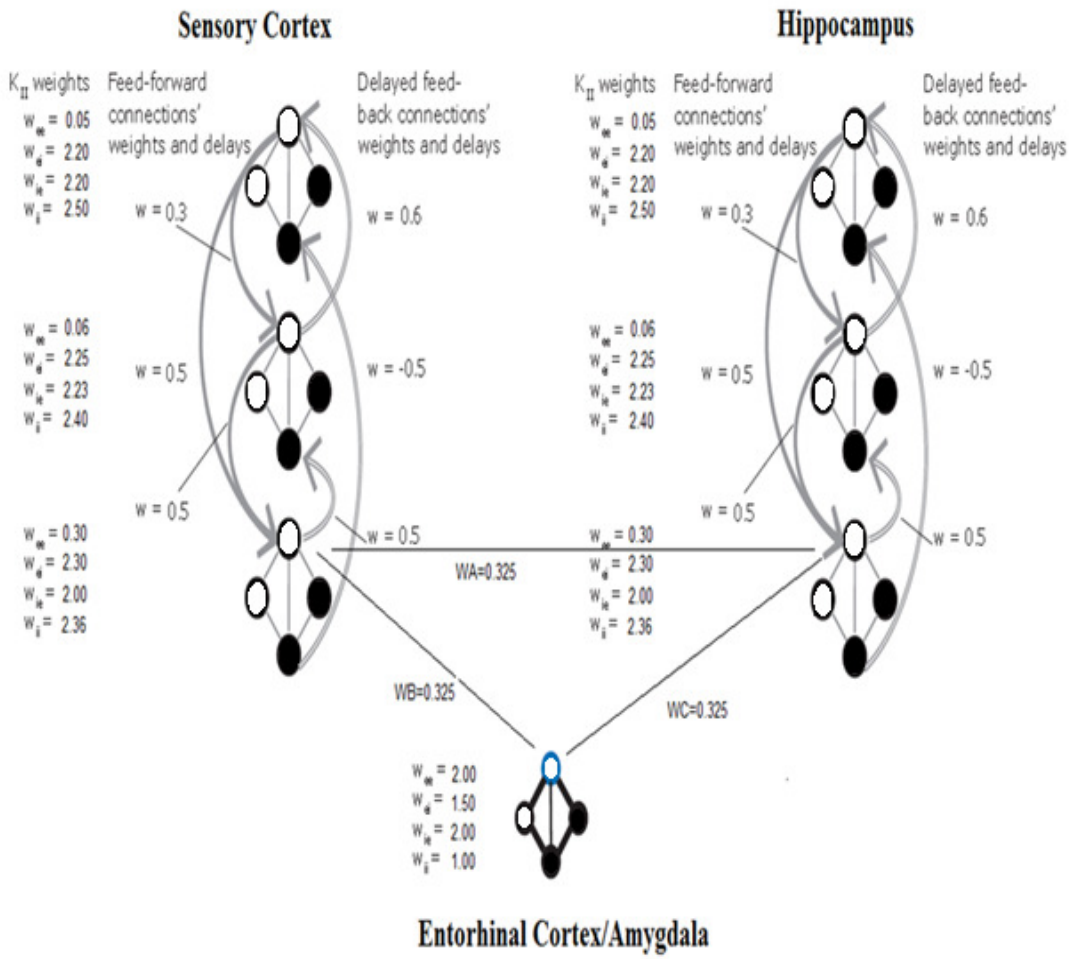


Figure 27. Schematic of KIV model with external and internal weights

through weighted matrix WB. Finally, the activations between Cortex and Amygdala are passed through weight matrix WC, as seen in Figure 27.

The internal weights of the KIII and external weights between the KIII and KII subcomponents provide the manner in which we can model various states of electrical neuronal activity found in a human EEG. Through this model, we can capture those attributes of abnormal/normal EEG data to provide further understanding of the healthy brain development. The output is read from the top KII nodes of the bottom most layers

in the KIV. Three outputs, one each from cortex, hippocampus and amygdala are obtained. The K models are implemented using MatLab. An iterative approach called Runge-Kutta method is used for the approximation of the solution of ordinary differential equations. The sampling rate is set at 0.5ms, in order to capture output with sufficient time resolution.

CHAPTER 5: KIII SIMULATED SEIZURES

Freeman's analysis of electrical stimulation of the lateral olfactory tract (LOT) has been found to induce an epileptiform seizure in the prepyriform cortex (PC) of cats, rats and rabbits (Freeman, 1962). We will initially develop the same simulation of seizure behavior described in Freeman's work (Freeman, 1972) using the KIII model to capture the electrophysiology of stimulated induced seizures.

As the complex sensory dynamics has been observed first in the olfactory bulb, early KIII sets mimic the architecture of the olfactory system (Chang & Freeman, 1998; Kozma & Freeman, 2003). Each layer is a distributed KII set, which consists of interacting neural populations. The simplified KIII set shown here consists of 3 layers of distributed KII sets corresponding to different anatomical parts of the brain: the olfactory bulb (OB), Anterior Olfactory Nucleus (AON), and Prepyriform Cortex (PC). Input signals enter OB through the glomeruli layer (top layer in Figure 28). Excitatory elements of the OB layer correspond to the populations of the secondary dendrites of the mitral cells. The inhibitory populations are the granule cells. Freeman has modeled mutual excitation between the mitral cells and mutual inhibition between the granule cells (Chang & Freeman, 1998). AON consists of excitatory pyramidal and inhibitory stellate cells. OB sends projections to AON and PC. They, in turn, send feedback projections, as shown in Figure 28. The delays in the feedback tract are greater than in other parts of the olfactory system, due to the length of the tract. Therefore, feedback links are modeled by delayed connections as in Figure 28 (Freeman, Chang, Burke, Rose & Badler, 1997). The two main cell types of excitatory and inhibitory cells in the prepyriform cortex are

the superficial pyramidal cell, which is excitatory, and the granule or stellate cell, which is excited by the former and inhibits it.

The KIII model generates chaotic activity with a variety of parameter settings as seen in each layer of the KII, Figure 29. Certain parameter aspects appear to be critical. In each KII set, the strength of mutual inhibition must exceed that of mutual excitation (kii > kee) (Freeman, 1986). By itself, the KII_{OB} set is capable only of a steady state activity ('DC') or oscillation at one frequency, not of chaotic activity. Coupling of the KII_{OB} set with the KII_{AON} set results in multi–frequency oscillation that repeats a complex pattern about every 0.3 s. Introduction of the KII_{PC} set to attain the KIII level deregularizes the 3/s activity (Freeman, 1986).

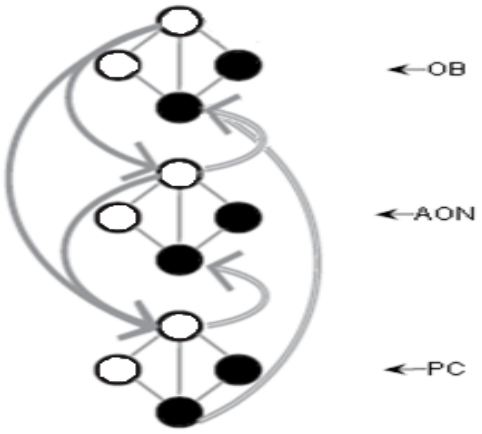


Figure 28. Simplified KIII system and its relationship with the anatomy of the olfactory system. Olfactory Bulb (OB), Anterior Olfactory Nucleus (AON), Prepyriform Cortex (PC). Black and white circles: inhibitory and excitatory populations, respectively.

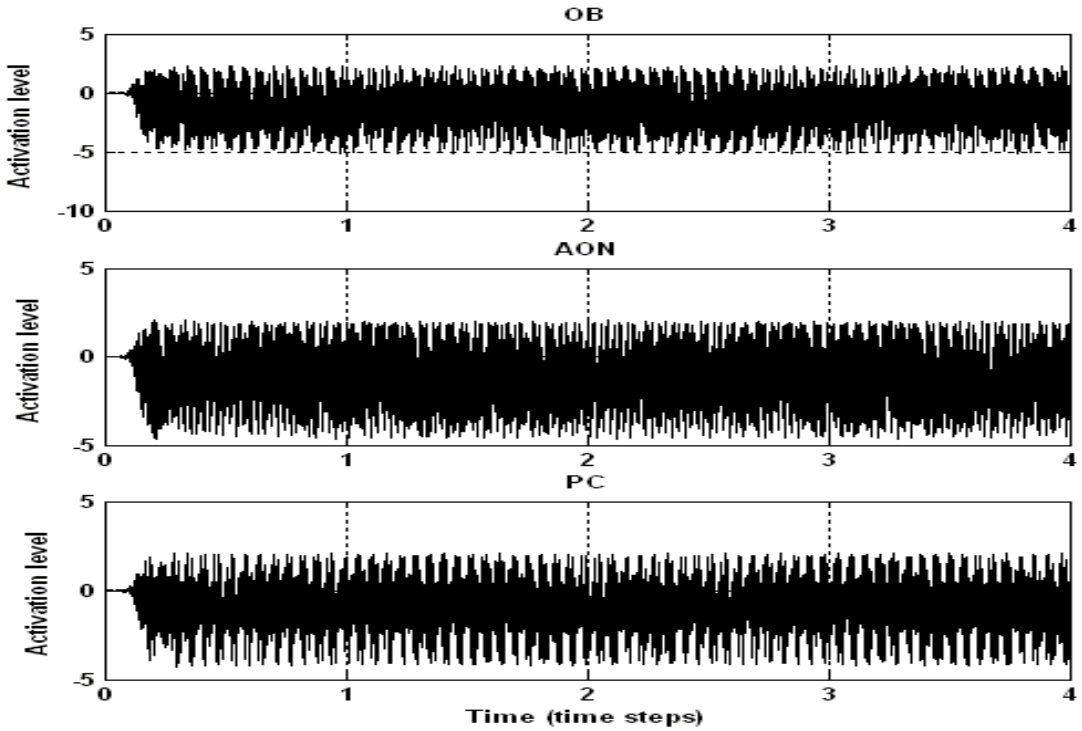


Figure 29. KIII model output of the background EEG as seen in the Olfactory Bulb (OB), Anterior Olfactory Nucleus (AON), Prepyriform Cortex (PC).

Parameterization of the KIII model to output seizure behavior is accomplished in the following manner. Initially, feedback is decreased from KII_{OB} to KII_{PC} and KII_{AON} to KII_{PC}. Next, feedback is decreased from the KII_{PC} to KII_{AON}. Finally, feedback is increased from KII_{AON} to KII_{OB}. The directions of change that are effective serve to reduce the activity of excitatory elements and to increase that of inhibitory elements and produce simulated EEG activity at the 3/s (Figure 31 as seen in the literature in Figure 30) (Freeman, 1986).

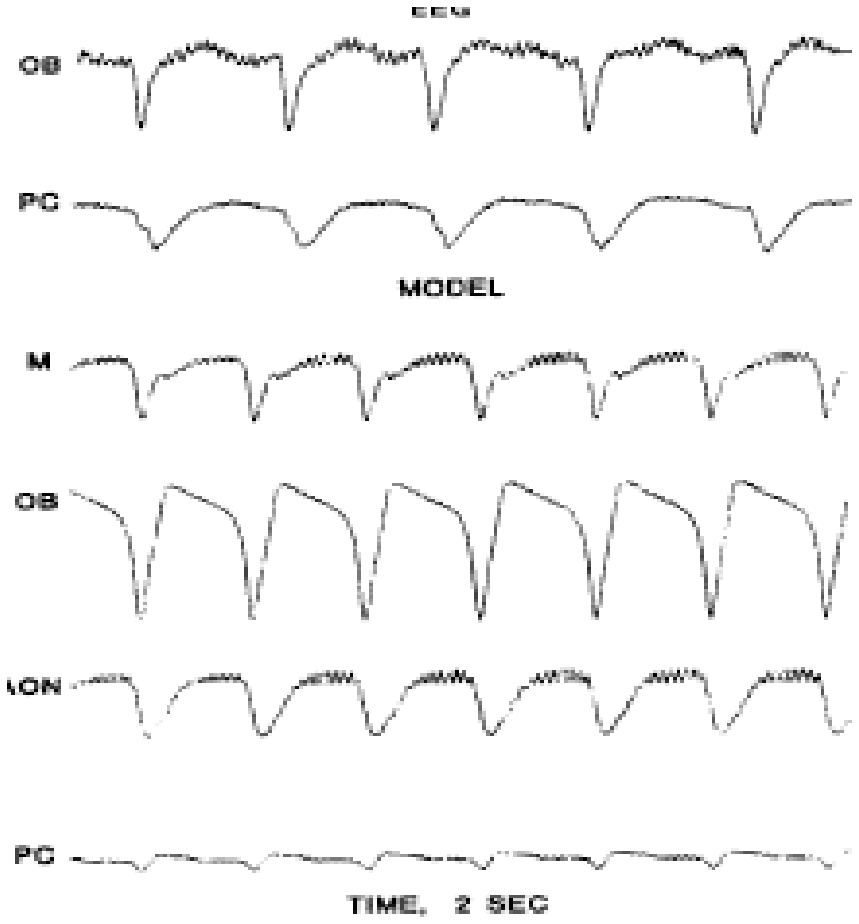


Figure 30. Examples of 2–s time segments of EEGs recorded from a rat during a seizure, comparing these with the outputs of the KIII model (Freeman, 1986).

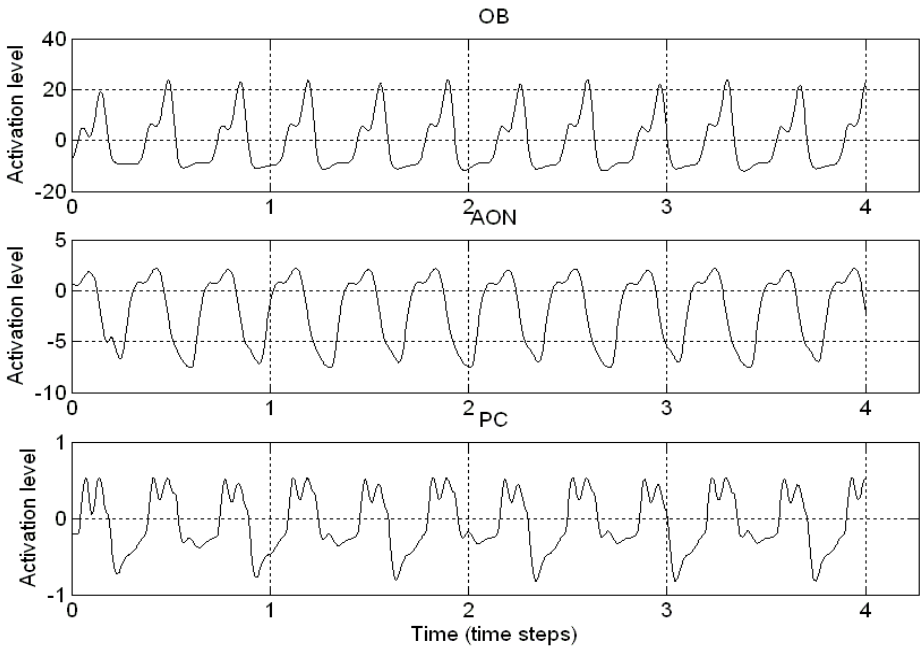


Figure 31. KIII model output of the background EEG to a 3/s spike train resembling the seizure as seen in the Olfactory Bulb (OB), Anterior Olfactory Nucleus (AON), Prepyriform Cortex (PC) caused by the reduction of the activity of excitatory elements and the increasing of inhibitory elements within the KIII.

Runaway inhibitory neurons are not caused by the failure of inhibition to control runaway excitation but in the failure of the afferent excitatory synapses to maintain symmetry that leaves the inhibitory interneurons in a hyperexcited and unstable condition (Freeman, 1986).

Several statistical properties are employed to characterize the KIII model as a model for normal EEG behavior and petit-mal seizures (Freeman, 1972). Power spectral density (PSD), exhibits a linearly decreasing behavior over log-log10 coordinates considering frequency and amplitude of PSD or spectral power, as seen in Figures 32 and 33.

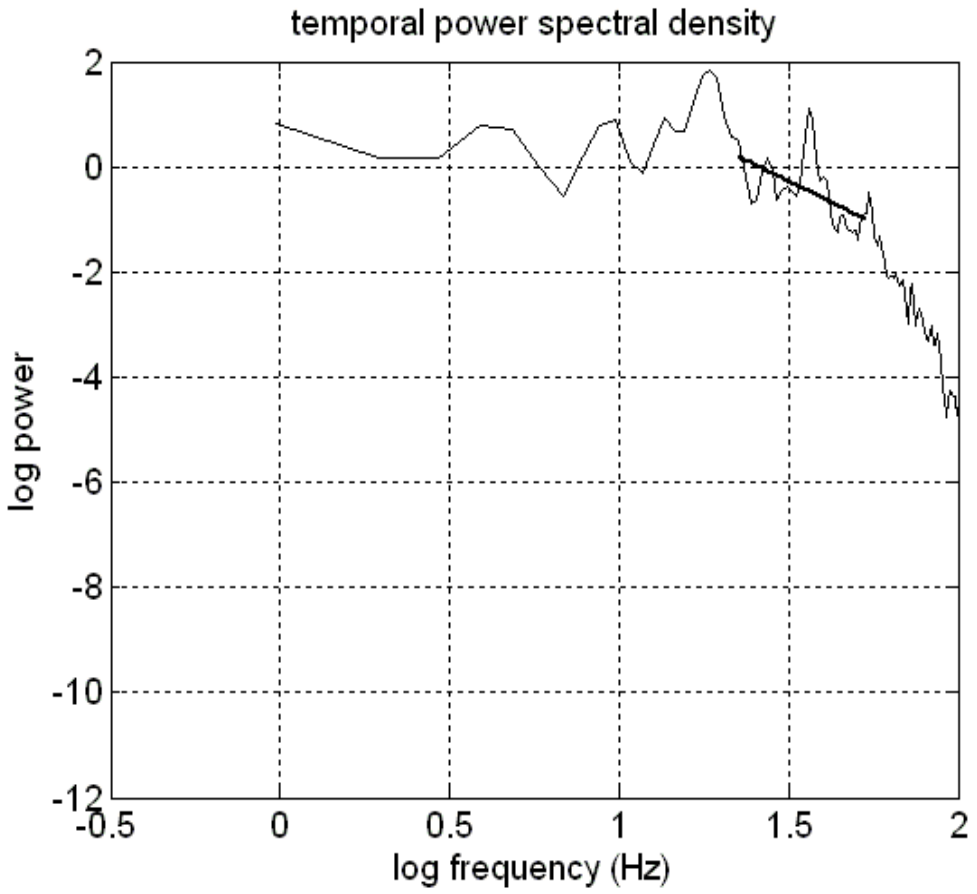


Figure 32. Power Spectral Density display of the KIII model. Log10 graph exhibits strong log power around 20 Hz.

Linear regression calculations provide a slope of the PSD display of 2.75. The slope value corresponds to the "power law" or scale-free behavior ($1/f^{\alpha}$), where cognitive processing states varied by the slope $-\alpha$. Alpha values of Human EEGs during awake and sleep states have been found to be within -2 to -3 (Freeman, Holmes, West & Vanhatalo, 2006).

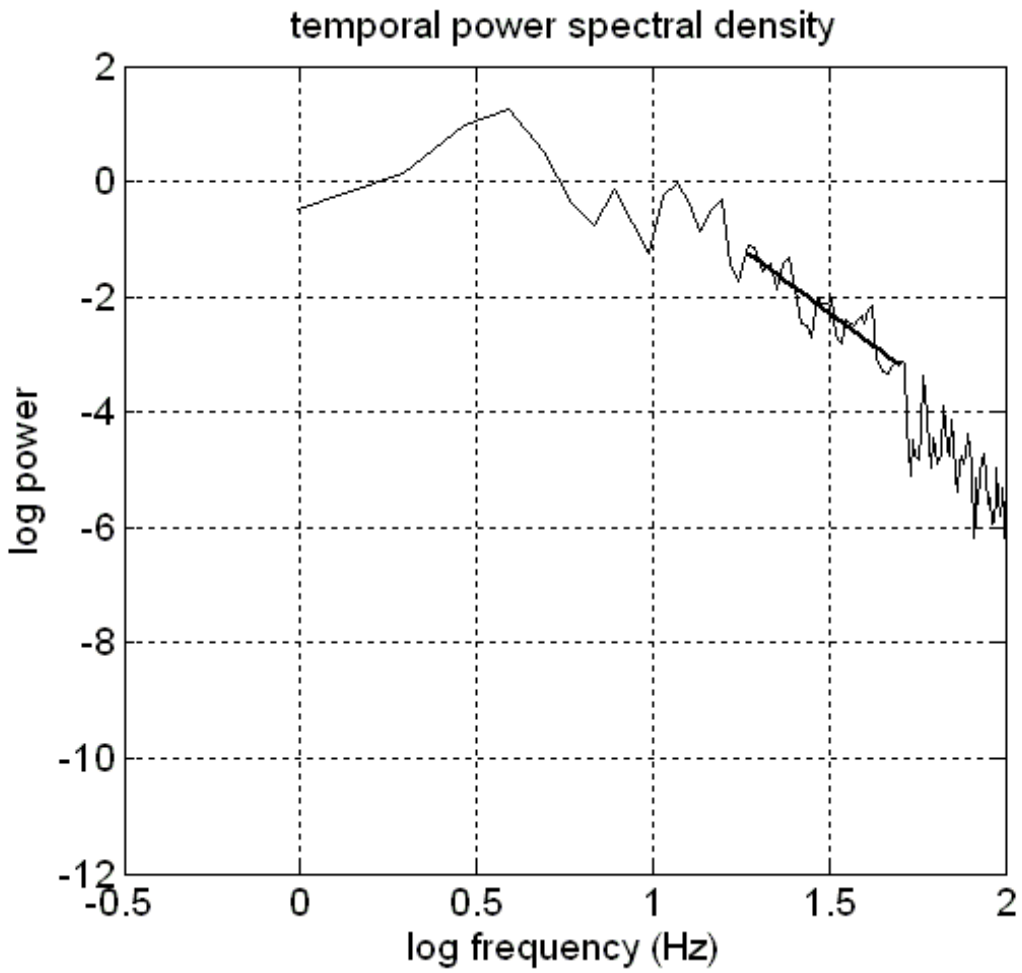


Figure 33. Power Spectral Density display of the KIII model during a simulated seizure. Log_{10} graph exhibits strong log power around 3 Hz. Linear regression calculations provide a slope of the PSD display of 3.98.

The calculated PSD linear regression values during the seizure state approaches the theoretical limit of less than -4 (Freeman & Zhai, 2009).

Autocorrelation displays of the KIII network during the simulated seizure state exhibit monotonously decreasing behavior, indicative of semi-periodic behavior (Wilrich, 2004) (Figure 34).

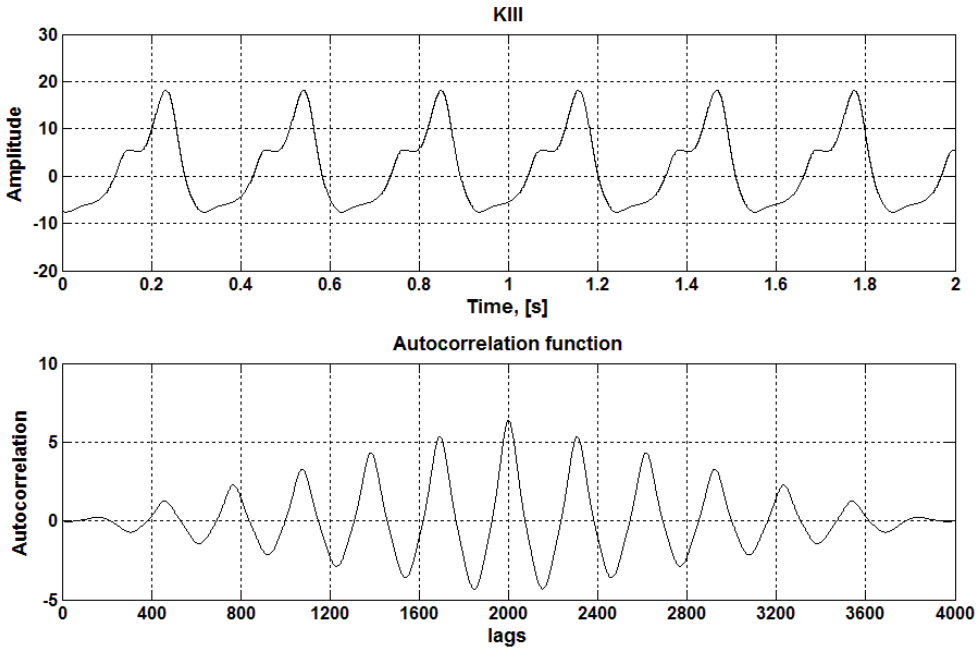


Figure 34. Autocorrelation of KIII simulated seizure signal.

Phase diagrams of the KIII seizure state exhibit semi-periodic behavior as the signal is displayed as a time series offset of twenty points from the original time series. We see in Figure 35 that the signal moves into an elliptical orbit during the simulated seizure event.

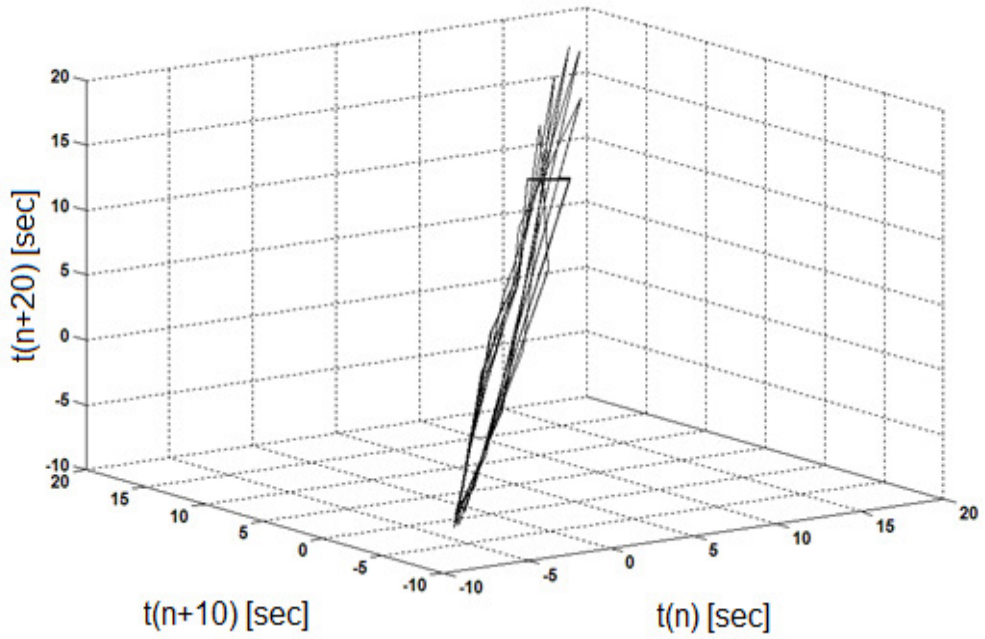


Figure 35. Phase diagram of the KIII simulated seizure state.

In terms of modeling global cortical seizure states, it is noted that spikes seen in the prepyriform cortex suggest that seizure activity has been propagated into other brain regions by the feedback connections from those other parts of the limbic system. They are subject to analysis and simulation with non–linear dynamics (Freeman, 1972). We will utilize the seizure parameterized KIII and incorporate this model into the KIV model to demonstrate seizure propagation to other parts of the limbic system.

CHAPTER 6: A MODEL FOR BRAIN PATHOLOGIES

Epilepsy Modeling

Several neural network models have been developed in modeling the bistable states of normal/abnormal brain behavior: an *interictal* one characterized by a normal, apparently random, steady-state of ongoing activity, and another one that is characterized by the paroxysmal occurrence of a synchronous oscillations (seizure). Lopes da Dilva et al. (2003) developed a model of thalamic and thalamocortical networks in order to demonstrate the behavior of populations of interacting neurons lumped together. In his experimentation, he illustrates the two brain states through the displays of 'normal' and 'seizure' attractors. The model consists of processing three inputs: a one glutamatergic (AMPA) and two γ-aminobutyric acid (GABA)ergic, with the corresponding synaptic transfer functions. The summed activity is the input to a nonlinear transfer function that represents the generation of impulses, including the low-threshold spikes. Stam and Pritchard (1999) applied nonlinear cross prediction (NLCP), to investigate if polymorphic delta activity (PDA) and frontal intermittent rhythmic delta activity (FIRDA) reflect linear or nonlinear brain dynamics. These dynamical properties of PDA and FIRDA could be reproduced by the Lopes da Silva model. PDA and FIRDA reflect a chaotic attractor and a limit cycle attractor, of the normal/abnormal brain states respectively, perturbed by dynamical noise. Another model based on "lumped'' or ''mean field'' neural models features corticothalamic modeling which has been based on the evolution of several dynamical variables within each of these populations. The variables represent the local mean value of a physiological process at position r in these neural systems, averaged over a small patch (~0.3 mm) of surrounding neuropil (Breakspear et al., 2006).

When the coupling between cortical to thalamic increases, feedback increases, whereas the model exhibits tonic-clonic or absence seizure effects. Models of corticothalamic dynamics have been developed that reproduces and unifies many features of EEGs, including the discrete spectral peaks in the slow wave, 'delta', 'theta', 'alpha', and 'beta' bands, seen in waking and sleeping states as well as generalized epilepsies (Robinson et al., 2003). Mean-field equations enabled the modeling of corticothalamic interactions.

Takeshita, Sato & Bahar (2007) models synchronized neural activity between two neurons to demonstrate seizure activity through the increase of extracellular potassium concentration which has been observed during epileptiform activity. In doing so, the model exhibits limit cycle behavior. The addition of noise in to the model enables noise-induced transitions between ''in-phase'' and ''antiphase'' network activities, analogous to the case of intermittent seizure activity. Additional conductance-based neuron modeling occurs in the model by Frohlich, Sejnowski & Bazhenov (2010). Both pyramidal cells (PYs) and fast-spiking inhibitory interneurons (INs) were modeled as two-compartment, conductance based neurons using the classic Hodgkin-Huxley form for ionic currents. The network used in this study consisted of 200 PYs and 40 INs. The network included recurrent excitatory connection between PYs and recurrent feedback inhibition. Prolonged perturbation over several seconds caused sufficiently elevated potassium levels thereby preventing the network from returning to the normal physiological state and moving into clonic activity.

Tsakalis and Chakravarthy (Chakravarthy, Sabesan, Iasemidis & Tsakalis, 2007; Takeshita, Sato & Bahar, 2005) developed a neural mass model, with an internal feedback mechanism to maintain synchronous behavior within normal levels despite

elevated coupling. Normal internal feedback quickly regulates an abnormally high coupling between the neural populations, whereas pathological internal feedback can lead to hypersynchronization and the appearance of seizure-like high amplitude oscillations. Feedback decoupling is introduced as a robust seizure control strategy. An external feedback decoupling controller is introduced to maintain normal synchronous behavior. Other internal feedback models featured closed-loop feedback control systems in epileptic seizures combining methods from seizure prediction and deep brain stimulation (Good, 2009). Periodic stimulation was also performed, with a reduction of seizure frequency in 33% of six rat modeling instances. Autoregressive modeling and neural network based modeling techniques are used to model and simulate electroencephalogram (EEG) signals of normal/abnormal states (Kannathal, 2006). Chaotic invariants like correlation dimension (CD), largest Lyapunov exponent ($lambda_1$, Hurst exponent (H), and Kolmogorov entropy (K) are used to characterize the dynamical properties of the actual and modeled signals. Additional neural network models features functional coupling between cerebral structures through coupling parameters in the model (Wendling, 2001). Seizure activity can be classified on the basis of interactions between medial and lateral neocortical structures.

Cognitive processing models have been developed of the hippocampal region in order to demonstrate when a subpopulation of neurons achieves and maintains a given spatiotemporal pattern, or a given "relatedness in activity," and which as a consequence allows for the identification of a relation between that pattern and an external event. Representations are transient because neuron firing typically is maintained in one spatiotemporal pattern for only hundreds to thousands of milliseconds (restated, the

duration of an identifiable spatiotemporal pattern is typically hundreds to thousands of milliseconds), unless we consider pathological conditions, e.g., rhythmic, cyclical firing characteristic of epilepsy (Berger, Song, Chan & Marmarelis, 2010). In the case of brain pathologies such as epilepsy, the brain model would exhibit the low frequency, high amplitude of the seizure state.

Initial Seizure Modeling

We modify the activity in the KIV model by adjusting the weight connections between the cortex, hippocampus, and amygdala. A small perturbation in initial conditions can greatly affect the evolution of chaotic systems through time, but once those conditions are established, the future of a chaotic system is just as deterministic as a non-chaotic system (Kozma, 2003). By introducing a greater bias to the KII signal we can cause the signal to change its chaotic nature.

Figure 36 displays the basal state A with high dimensional oscillations for a healthy patient. The Lyapunov exponent is 0.10, showing well-developed chaos. Figure 37 exhibits state B with increase inhibitory connection weights. This state simulates epileptic conditions. The trajectory becomes less chaotic, and the Lyapunov exponent is 0.02. Figure 38 shows the case of simulated electrical external treatment, when highly chaotic behavior is restored in the model by applying external noisy bias. This is called state C. The calculated Lyapunov exponent is 0.12 indicating well-developed chaos. State C is similar to the conditions that external stimulation treatment induces on the EEG activity of the seizure patient. Namely, it forces the system back to an average system state (Myers & Kozma, 2007).

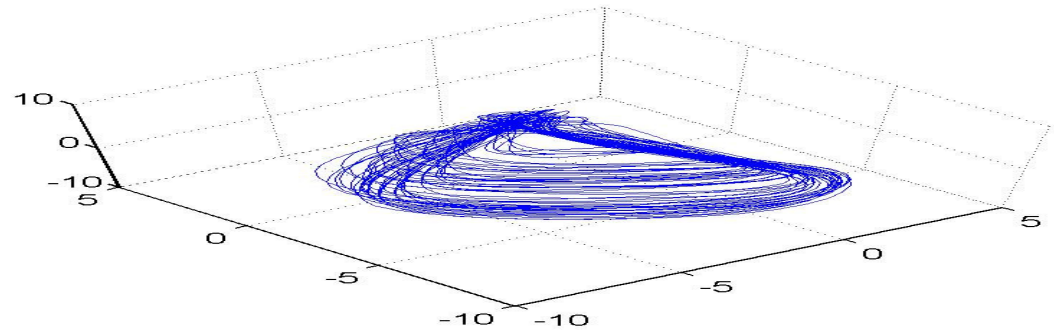


Figure 36. Simulations with KIV model in the basal normal state A. The figure is return plot in the time-delayed phase space.

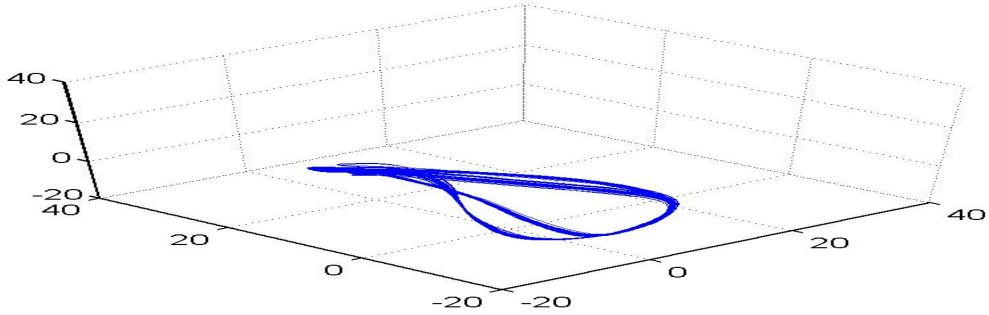


Figure 37. Simulations with KIV model in the simulated seizure state B.

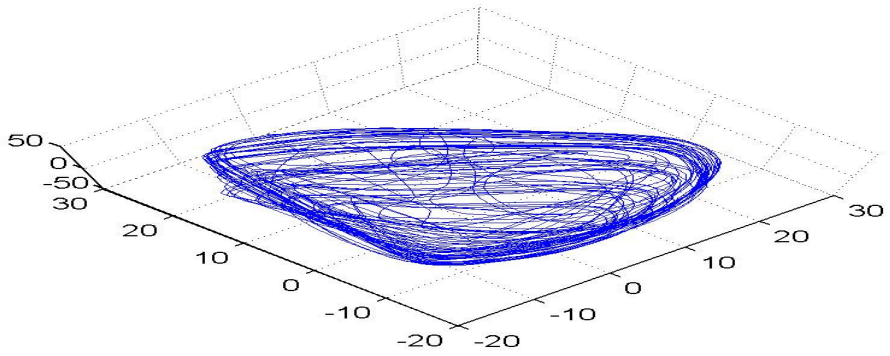


Figure 38. Simulations with KIV model in the simulated VNS treatment state C. Higher complexity chaotic state is restored

KIV Parameterization

The goal of utilizing the KIV in this study is to have the model exhibit normal 'chaotic' EEG behavior, then transition to abnormal semi-perioidic behavior as displayed on EEGs exhibiting seziure type behavior. Additionally, the true strength of the KIV model is not only its capability of exhibiting non-linear dynamic behavior, but to demonstrate how abnormal brain pathologies such as how seizure behavior can be reduced or nullified via external input techniques into the model. In this manner, the KIV can act as a test bed for new electrical stimulation titration methods before those methods are tested on animals and eventually humans. Therefore the KIV must exhibit a 'restoration' state of transitioning back from semi-periodic behavior to normal chaotic behavior. Table 2 features the internal and external KIV network parameters in used to create the 'normal', 'seziure', and 'restore' states.

Table 2

KIV Parameterization for normal->seizure->restore states

State	**WA/WB/ WC**	**KIII Feed-forward /Delayed Feed-back weights**	**External Input into KIV via BSI (KII) Internal Parameters**
'normal'	0.325	[0.30, 0.50, 0.50, 0.50, -0.50, 0.60]	
'seizure'	5.0	[0.30, 0.20, 0.20, 0.30, -0.50, 10.0]	
'seizure->restored'	5.0	[0.30, 0.20, 0.20, 0.30, -0.50, 10.0]	[2.0, 1.5, 2.0, 1.0]

The parameters used in the KIV for normal EEG modeling are the external lateral weighting between the three objects (WA, WB and WC) which were reduced so that each object (the two KIIIs and the single KII) could produce their respective signal ouptut with some small degree of influence from the rest of the network in order for the KIV to exhibit the chaotic normal state as seen in the human and simulated EEG time series (Figures 39 and 40). In order to exhibit the 'normal' chaotic state of human EEGs, the KIV model was adjusted to exhibit the same 'noisy' attributes through the input additive noise throughout the network (Kozma, 2003).

Biological systems exhibit both low-dimensional and high-dimensional chaotic states where both states coexist, and are in perpetual transition between the two extremes (Kaneko, 1990; Tsuda, 1996). Using the architecture of the KIII, the additive noise is implemented to homeostatic regulation of the network within the chaotic attractor regime. In response to external stimulation, the behavior of the signal changes can switch from periodic, quasiperiodic, or aperiodic oscillations to chaotic attractors and vice versa. The additive noise level prevents the dynamics from collapsing to a fixed point, and the system exhibits a nontrivial oscillation (Kozma, 2003). More generally, additive noise in the KIII model increases oscillations in the gamma band, from 20 Hz to about 60 Hz, where this frequency band is found in normal cognitive processing EEG attributes (Freeman, 2000a).

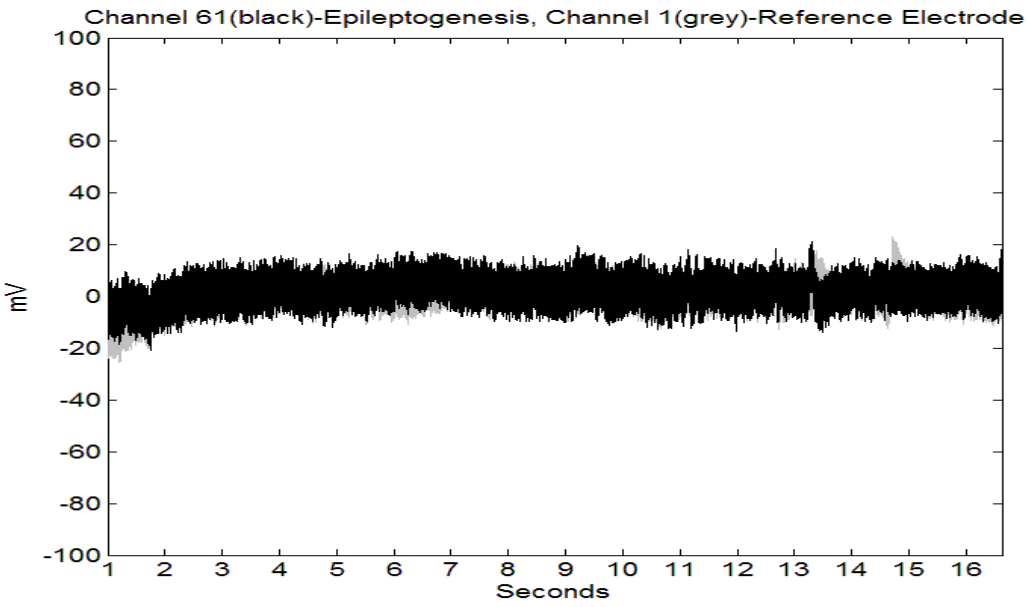


Figure 39. EEG time series featuring 'normal' EEG behavior.

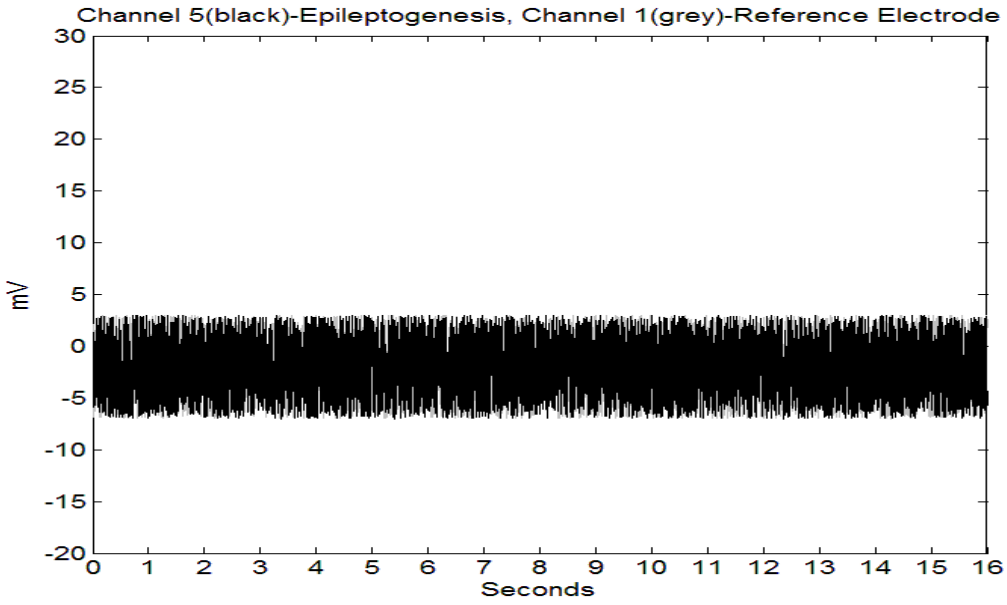


Figure 40. Simulated EEG time series featuring 'normal' EEG behavior.

In order to simulate the epileptogenesis of a localized group of abnormal firings of neurons, the lateral weights between the KII networks within a KIII network are modifed as discussed in section 'KIII Simulated Seizures'. In order for this signal to

propogate throughout the KIV network and into the other objects, the coupling or lateral weights between the networks must be increased. Therefore, the feedback that gets updated into the other networks causes a signal stratification within the KIV network, producing a higher amplitude and semi-periodic signal. The signal is read out from the bottom KII network which represents the neurological equivalent of the amygdala. All the nodes/neurons within the KII network produce synchronous producing signals, i.e., each of the nodes/neurons mirror each others signals. This output exhibits the phenomenum of entrainment found in seizure behavior.

The transition of 'normal' to 'seizure' states is performed by having the KIV initially increase the lateral weights that couple the KIV objects together, and update the internal KII object at a given time period, as seen in human EEG compared to simulated EEG (Figure 41).

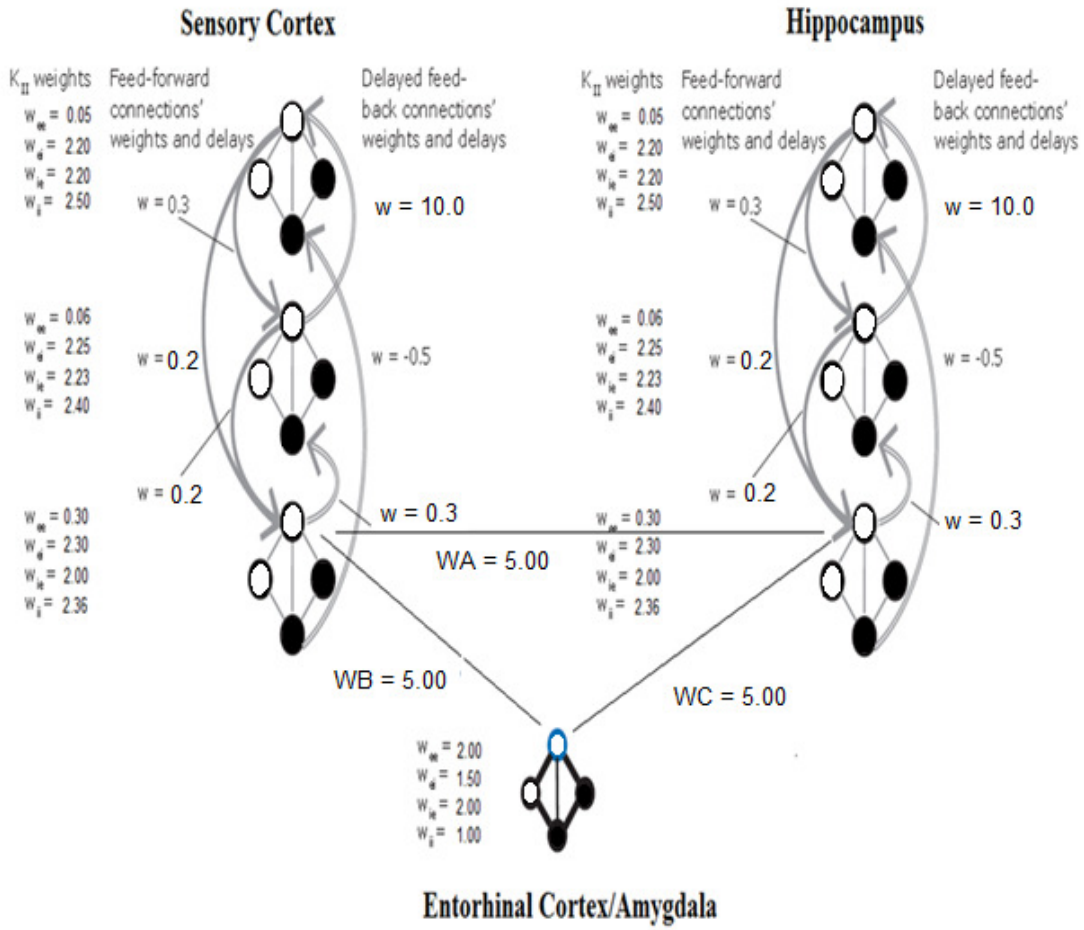


Figure 41. KIV parameterization of seziure state through feed-forward and delayed feedback changes. Additionally, External weight parameters, WA/WB/WC have been increased to tightly couple the three network oscillators.

The 'seizure' state would end when the KIV network reduces the lateral weights to its orginal low weighting and update the the intenal KII object after another set time. 'Random' seizure events would occur by invoking these seizure parameters utilizing random time periods, in order to mimick the seizure event in human EEG seen in Figure 42 and demonstrated in Figure 43.

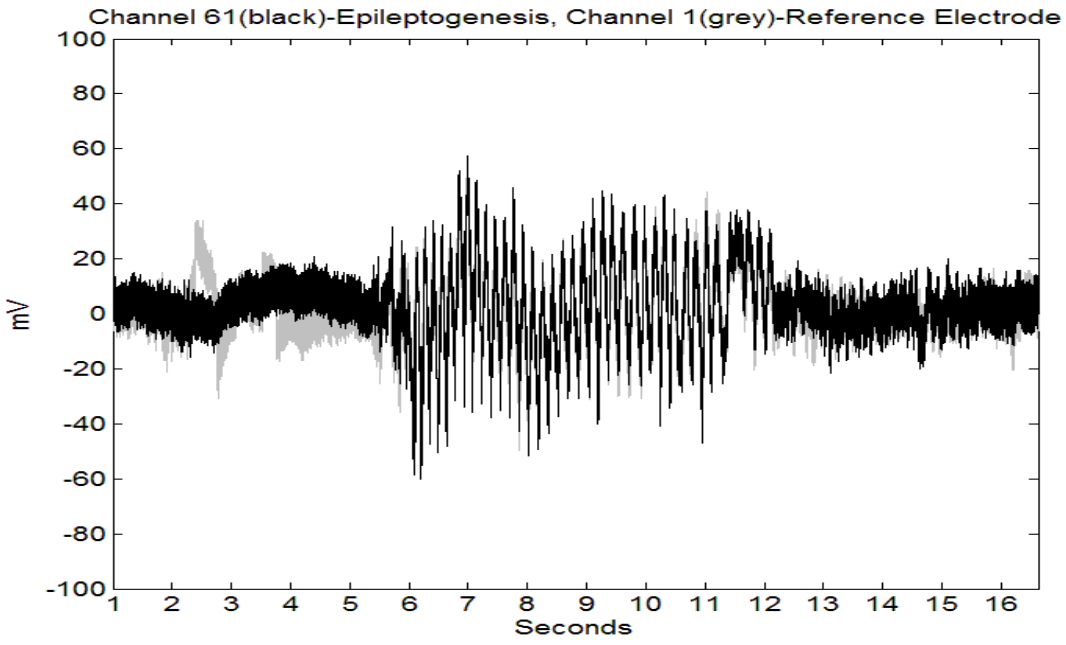


Figure 42. EEG figures depict a seizure event causing entrainment of two disparate neuron populations.

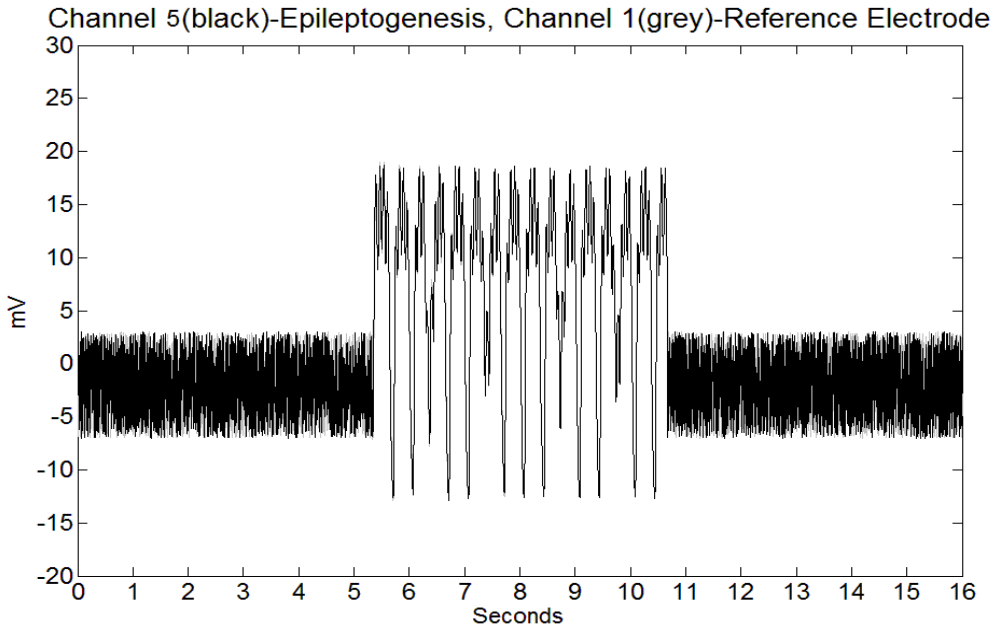
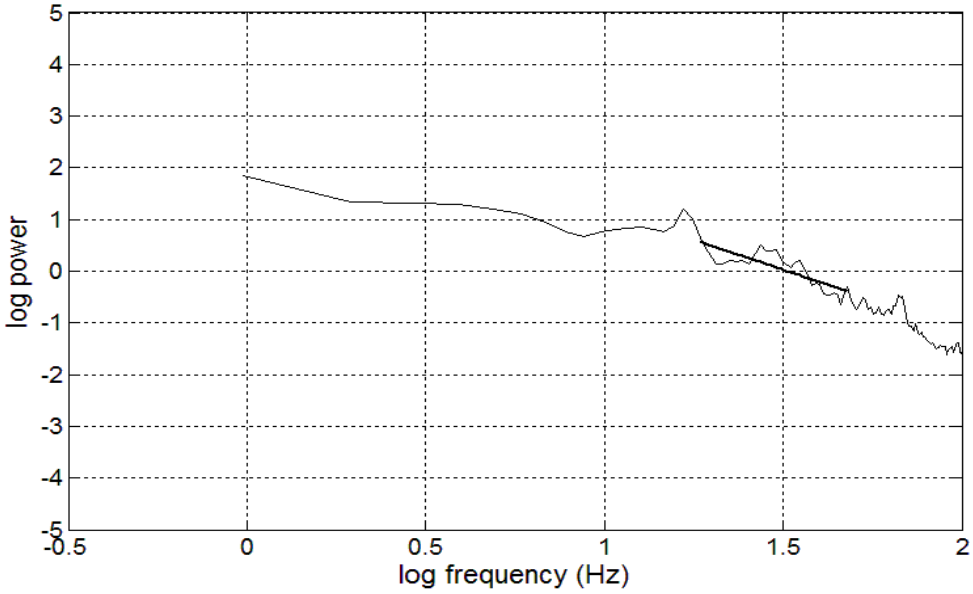


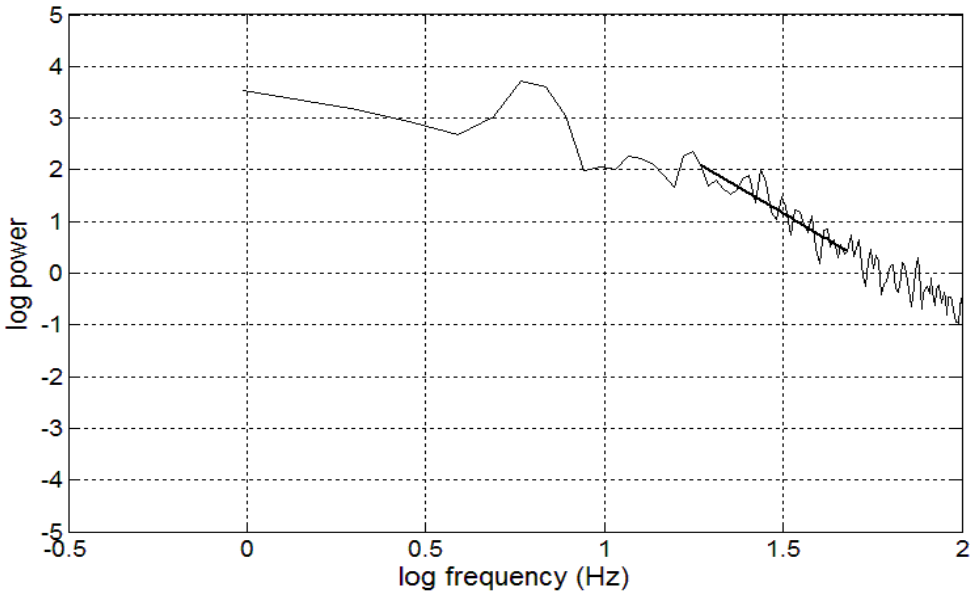
Figure 43. Simulated EEG figures depict a seizure event causing entrainment of two disparate neuron populations.

EEG and Simulated EEG PSD Analysis

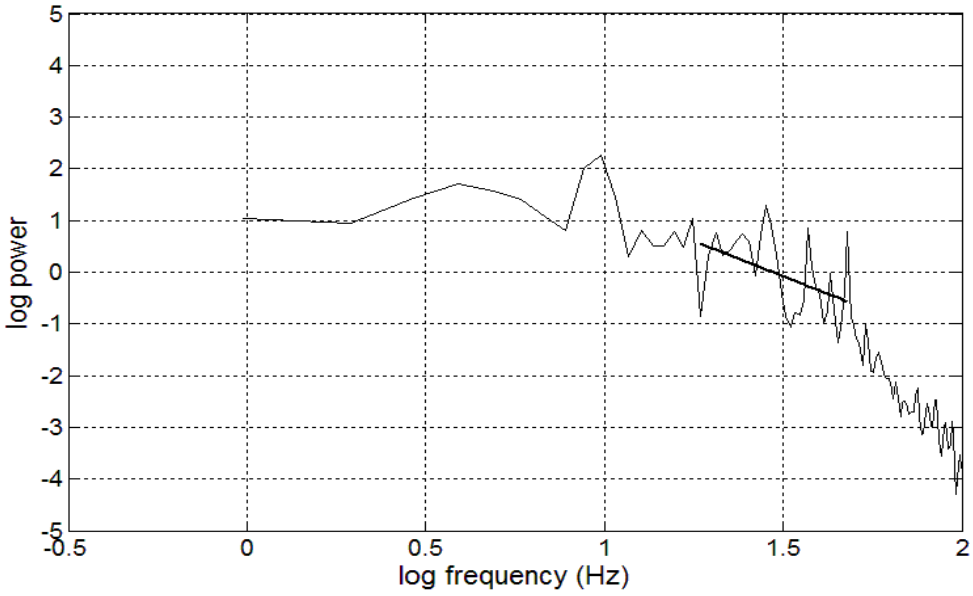
PSD analysis found in Figure 44 (a-d), features PSD displays of EEG and simulated EEG time series, where linear regression was performed in order to find the slope of the PSD values. The slope, -α corresponds to the alpha value in $1/f^{\alpha}$. Simulated EEG PSD alpha values are in the same range of normal EEGs (2-3) and seizure state EEGs (< 4).



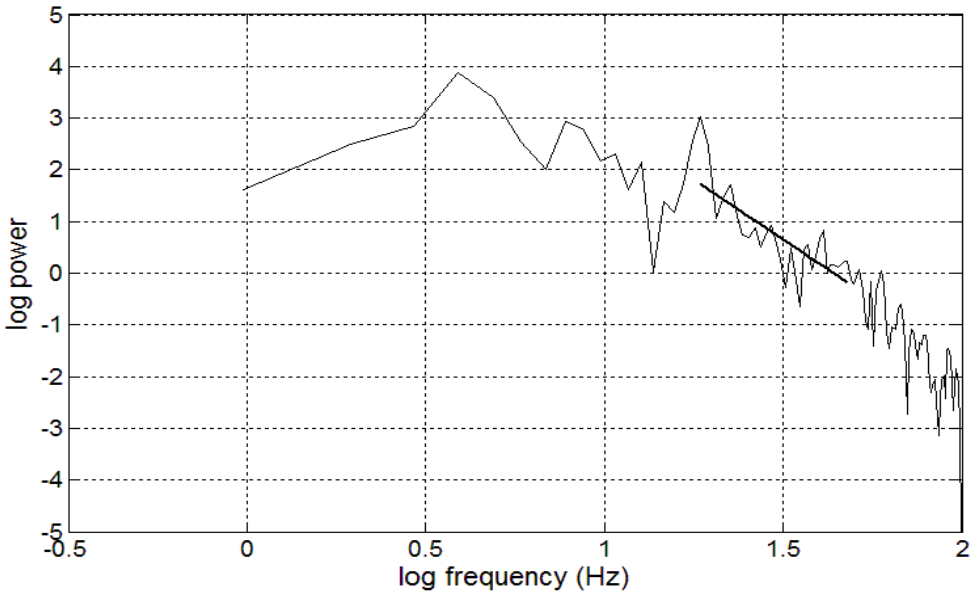
(a)


log power
log frequency (Hz)
5
4
3
2
1
0
-1
-2
-3
-4
-5
-0.5
0
0.5
1
1.5
2

(b)


log power
log frequency (Hz)
5
4
3
2
1
0
-1
-2
-3
-4
-5
-0.5
0
0.5
1
1.5
2

(c)



(d)

Figure 44. EEG figures featuring PSD values for EEG data where alpha = -2.16 (a) and a seizure event, where alpha = -3.68 (b). Simulated EEG of normal, where alpha = -2.73 (c) and seizure state, where alpha = -3.88 (d).

The PSDs of four human patients EEGs were analyzed to confirm the differing alpha values found between the normal and seizure states, as seen in Table 3. Alpha values collected from the slope of the log10 values confirmed the ranges for normal/seizure states, where $\overline{\propto}$ and σ_α denote the average and standard deviation of α, respectively. The PSD values found during seizure activity exhibited high power in both the low and high theta ranges, corresponding to the 3/s wave that dominated the EEG (Freeman, Holmes, West & Vanhatalo, 2006).

Five simulated patients are constructed through the KIV by varying the levels of noise throughout the KIV by 5%, whereas patient A1 is a KIV with full input noise, and

patient A5 has 20% less noise than A1. These simulated 'patient' EEGs illustrate how noise affects the alpha values collected from their respective EEGs. Additionally, the KIV normal and seizure state exhibited the same alpha value range, as seen in Table 3.

Table 3

Power law exponent (-α) of the PSD regression function ($1/f^{\alpha}$) Human EEG Data

	Normal state				Seizure state			
	Patient 1 (25)	Patient 2 (25)	Patient 3 (25)	Patient 4 (25)	Patient 1 (25)	Patient 2 (25)	Patient 3 (25)	Patient 4 (25)
$\bar{\alpha}$	2.38	2.53	2.46	2.51	3.65	3.68	3.72	3.64
σ_{α}	0.24	0.29	0.25	0.24	0.16	0.14	0.11	0.1
Min (-α)	2.02	2.01	2.04	2.15	3.41	3.39	3.54	3.42
Max (-α)	2.91	2.93	2.86	2.89	3.89	3.91	3.88	3.81

Table 4

Power law exponent (-α) of the PSD regression function ($1/f^{\alpha}$) Simulated EEG using KIV Model

	Normal state					Seizure state				
	A1 (25)	A2 (25)	A3 (25)	A4 (25)	A5 (25)	A1 (25)	A2 (25)	A3 (25)	A4 (25)	A5 (25)
$\bar{\alpha}$	2.40	2.40	2.50	2.67	2.70	3.61	3.65	3.74	3.77	3.77
σ_{α}	0.28	0.26	0.24	0.22	0.16	0.18	0.17	0.13	0.11	0.11
Min(-α)	2.04	2.06	2.12	2.32	2.41	3.31	3.33	3.44	3.57	3.60
Max(-α)	2.86	2.88	2.92	2.95	2.96	3.96	3.91	3.95	3.97	3.98

The simulation using noise in the KIV requires temporal filtering to give 1/f amplitude spectra of temporal frequencies. This method of simulation is based on the premise that EEG activity is due to near-white noise generated by immense numbers of interacting pyramidal cells, whose activity episodically undergoes transient increases in spatial coherence (Freeman, 2006). Noteworthy are the steepened slope of 1/f PSD in the seizure state compared with the normal EEG state (Figure 44a), which is simulated in the KIV model (Figure 44c).

PSD values found during the EEG time series and the simulated EEG time series exhibit low standard deviation during the seizure states per patient. This activity may be due to the entrainment of large scale neural population whose power per frequency is limited to the theta range (Figure 44b – human EEG, Figure 44d – simulated EEG). High theta activity causes the slope of the PSD to rise sharply, and diminishes any other brain activity state, i.e., the carrier wave in the rest state, active state and sleep state which seems to be diminished or lost during the seizure ((Freeman, Holmes, West & Vanhatalo, 2006).

The role of noise in the KIV is illustrated in Table 3, whereas the lessening of noise from patient A1 to A5 seems to increase the PSD alpha values from the normal to abnormal KIV time series. The rise of PSD values found in the theta frequency bands of 3-8 Hz causes the slope of the PSD to rise sharply, causing the higher frequency bands to diminish due to the reduction of noise in the system, and therefore moving the system into a lower dimension chaotic state (Kozma, 2003). High standard deviations of the slope of the alpha values are maintained during the simulated normal states, but diminish as noise is reduced from the model. Conversely, lower standard deviation values are

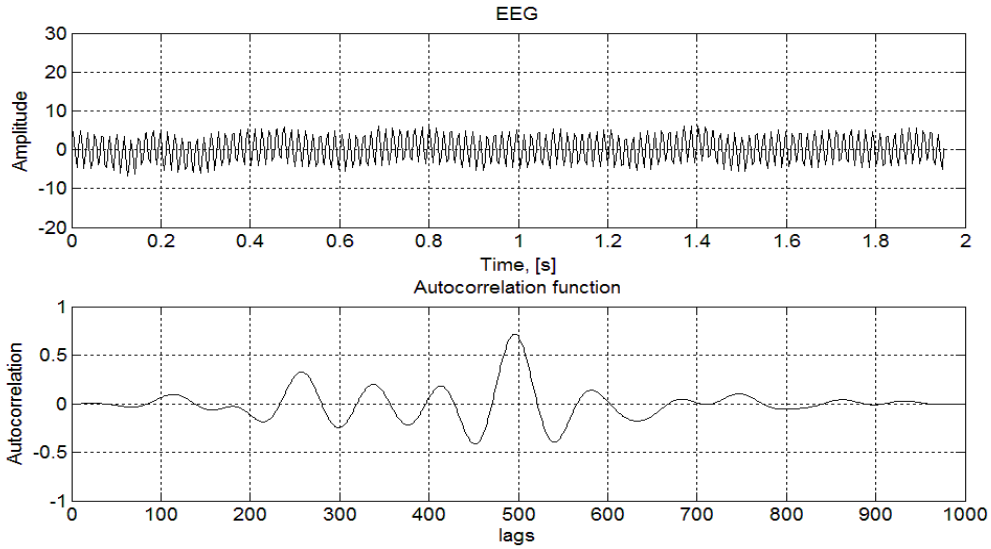
found during the seizure state, increasing as noise is removed from the network, but within a much smaller range.

The PSD in the normal state of human and simulated EEG conforms roughly to 1/f having a slope near –2 (Figure 44a and c). We propose that the normal neural activity manifests a process of self-organized criticality, in which the mean firing rates of neurons are held at a critical value throughout the cortex, homeostatically stabilized locally by their thresholds and refractory periods (Freeman, 2004).

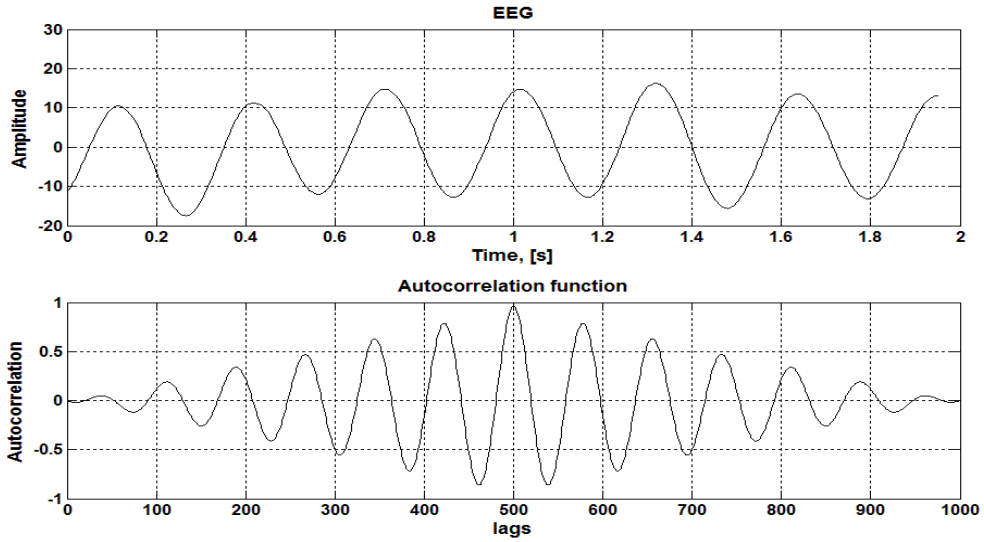
The onset of seizures is characterized by high amplitude slow waves and a shift in the PSD nearer to 1/f with a slope near –4 and loss of the peaks in PSD in the classical ranges (beta – gamma frequency ranges) (Figure 44b and d). The seizures were characterized by high amplitude waves at 3/s showing that the normal shared waveform (the carrier wave in the normal state, i.e., beta–gamma ranges) was diminished or lost during the seizure. The PSD during seizure likewise often differed sharply from the 1/f form of the PSD in the normal states with high power in both the low and high theta ranges, corresponding to the 3/s wave that dominated the EEG (Freeman, 2006). We attribute these properties to sustained loss of large-scale synaptic integration (Varela et al, 2001) that normally smoothes by cooperative interactions the local spatial variations in activity that arise through competitive inhibition and prevents breakdown of large-scale organization of neocortex (Freeman, Holmes, West & Vanhatalo, 2006). When interactions among inhibitory neurons rose to a high intensity with a deficit of excitation, an instability emerged in which by random fluctuations some neurons increased their activity, inhibited their neighbors, and became more excited in regenerative feedback that produced a self-limited explosive discharge of half the population and an inhibitory post-

synaptic potential in the excitatory neurons to which they projected (Freeman, Holmes, West & Vanhatalo, 2006). The seizure states are departures from this sequence into diminished global integration by sustained loss of long-range correlated activity, initially with loss of spectral peaks in the PSD, followed by explosive growth of local activity (Freeman, Holmes, West & Vanhatalo, 2006).

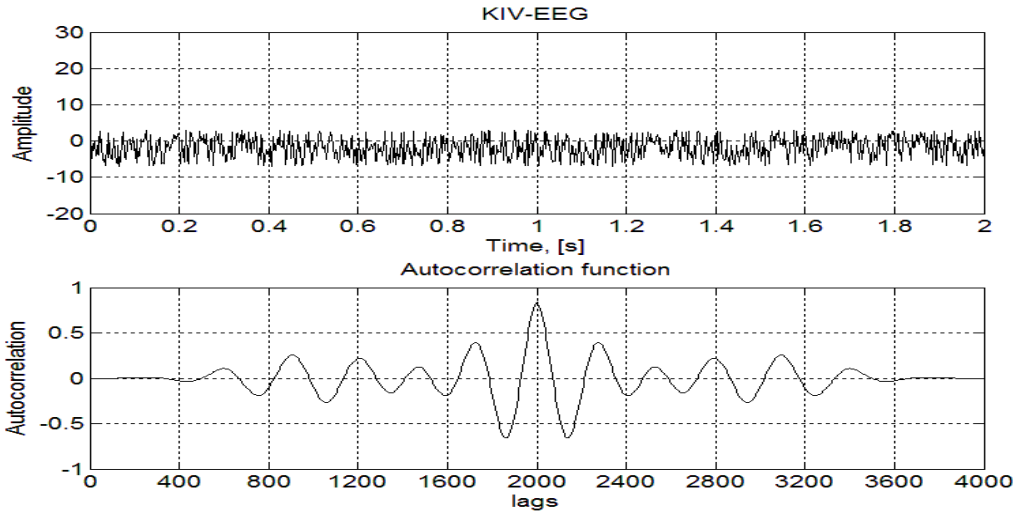
Autocorrelation diagrams depicted in Figure 45(a-d) show the same periodic and non-periodic behavior between seizure and non-seizure states of the simulated EEG.



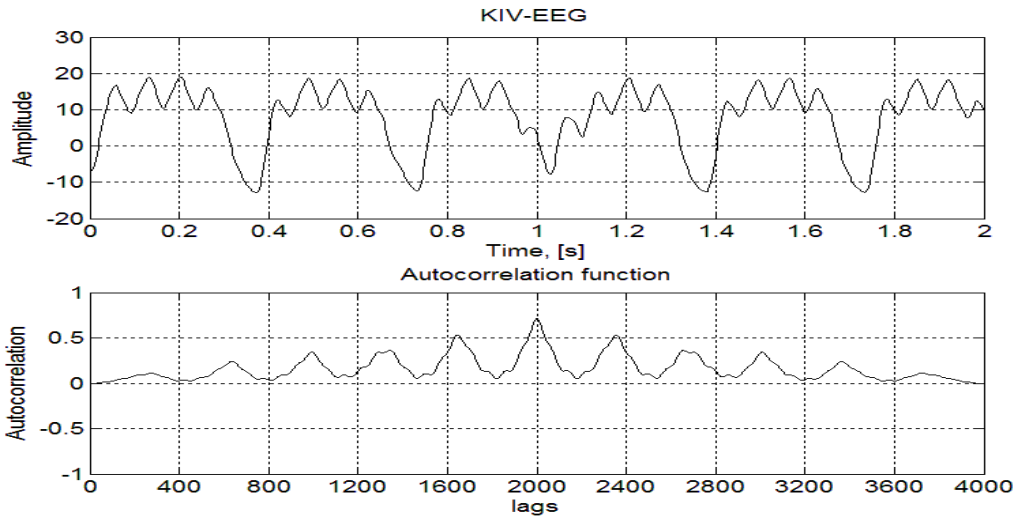
(a)


EEG
Amplitude
Time, [s]
Autocorrelation function
Autocorrelation
lags

(b)


KIV-EEG
Amplitude
Time, [s]
Autocorrelation function
Autocorrelation
lags

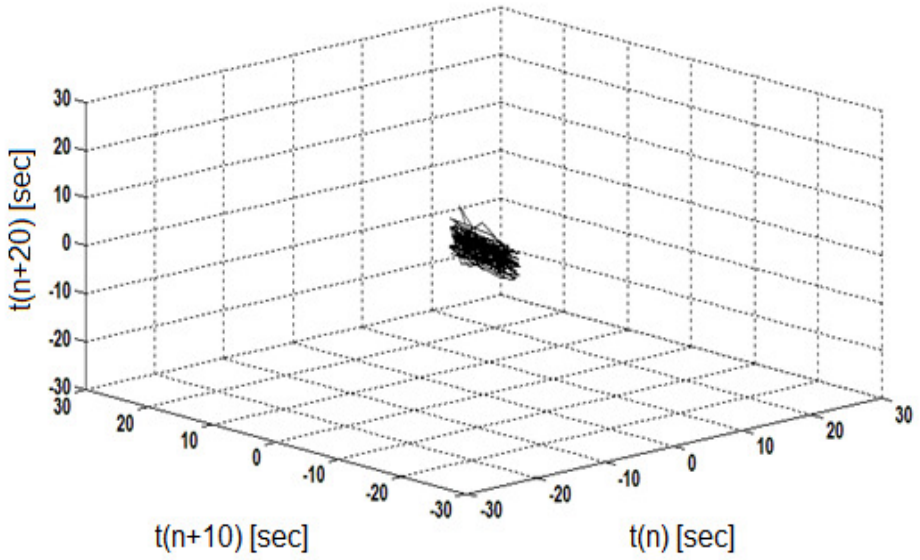
(c)



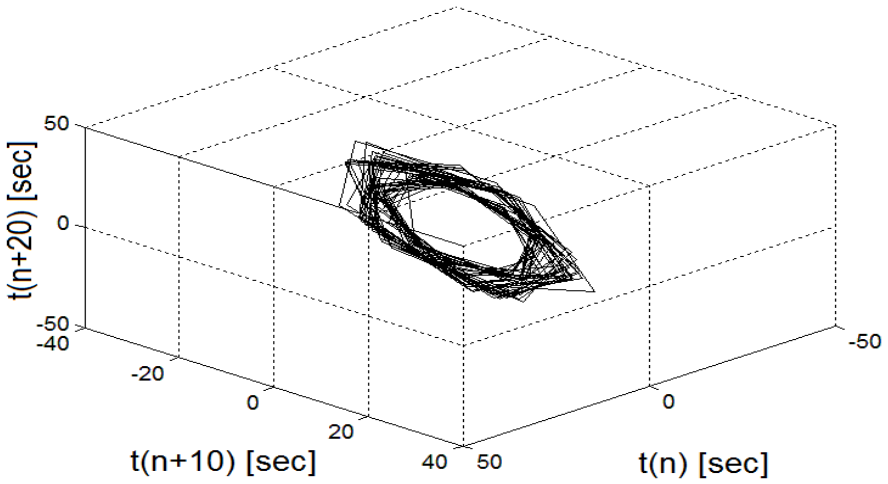
(d)

Figure 45. EEG figures featuring autocorrelation diagrams for EEG data where the normal chaotic state is exhibited in the diagram (a) and a seizure event (b), depicts periodic activity. Simulated EEG of normal, where chaotic values are depicted (c) and the simulated EEG shows periodic activity (d).

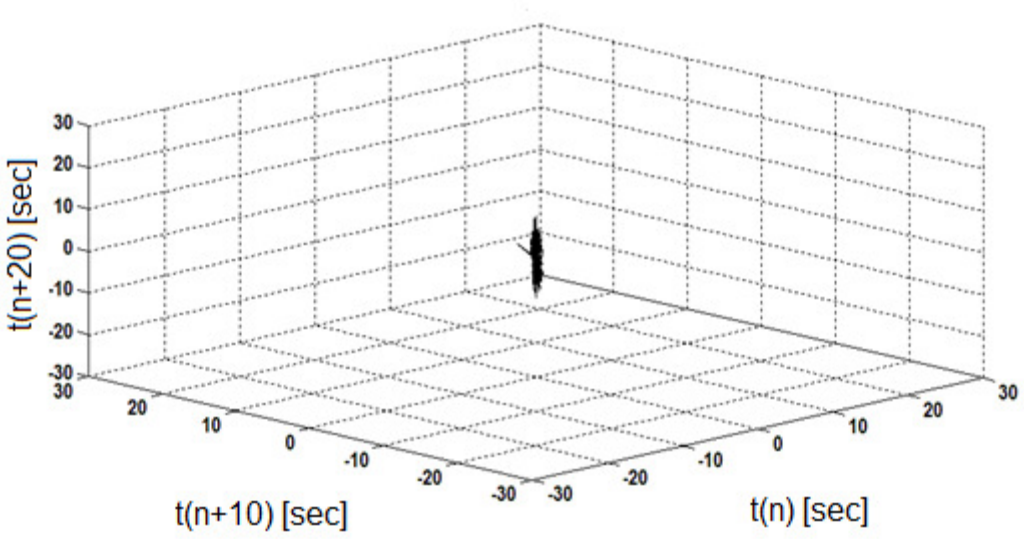
Phase diagrams displayed in Figure 46(a-d) depict the chaotic attractors found in normal EEG and simulated EEG as well as limit cycles found in the EEG and simulated EEGs of the seizure state.


t(n+20) [sec]
30
20
10
0
-10
-20
-30
30
20
10
0
-10
-20
-30
t(n+10) [sec]
-30
-20
-10
0
10
20
30
t(n) [sec]

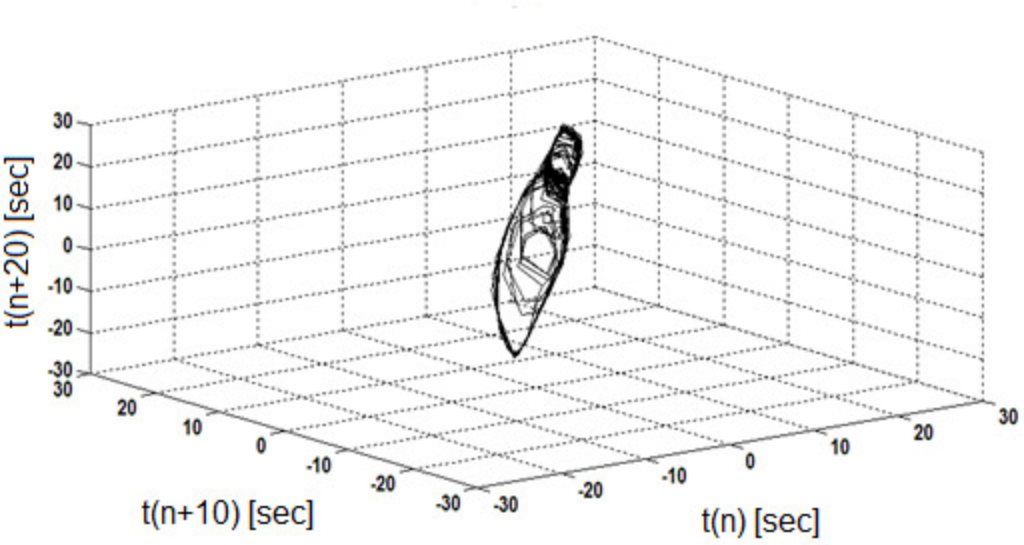
(a)


t(n+20) [sec]
50
0
-50
-40
-20
0
20
40
t(n+10) [sec]
50
0
-50
t(n) [sec]

(b)



(c)



(d)

Figure 46. EEG figures featuring phase diagrams values for EEG data where the normal chaotic attractors are exhibited in the diagram (a) and a seizure event depicts a limit cycle (b). Simulated EEG of normal, where chaotic attractors are depicted (c) and seizure state, exhibits the same limit cycle attributes (d).

CHAPTER 7: SEIZURE REDUCTION IN THE KIV MODEL

The 'seizure to restore' state is accomplished through the input of Brain Stimulator Interface (BSI) object. This object is a modified KII object which enables the restoration of the KIV seizure state back to its normal chaotic attractor output. The de-synchronizing external signal is a KII signal with the original node values. In this manner, we are adding a KII network to the KIV network to overcome the semi-periodic firings of an 'abnormal firing' KII network due to runaway inhibitory neuron hyperexcitation (Figure 47). The modified KII object also consists of an amplitude reduction signal which is a sample of the seizure state time period and increase the values by 1%. This sample is subtracted from the the seizure signal. The BSI added input and the amplitude reduction sampling signal restore the signal back to its normal state (Table 2, 'seizure->restore' state) (Figure 48). The external signal is comprised of two input matrices that break the entrainment of the nodes within the KII and reduce the overall amplitude of the signal. This technique mirrors the approach from (Tsakalis, Chakravarthy & Iasemidis, 2005) Decoupling Control mechanism. Additionally, the theory of superposition applies in amplifying/reducing signals in restoring the seizure state back to its normal state. As found in the literature, low-frequency pulse trains produced a hyperexcited state of the seizure state (Lesser, 2008) as well as a high frequency stimulation train (Betram, 2007). It is through a simulated patient-tailored protocol that reduced the simulated seizure effect (Osorio, Frei, Sunderam, Giftakis, Bhavaraju, Schaffner & Wilkinson, 2005; Sunderam, Gluckman, Reato & Bikson, 2010).

The simulated seizure state is maintained at the same time segment and the model is re-executed, except with an added external input to simulate electrical pacing therapy (BCI object). The KIV architecture involves adding the BCI object to the KII matrix, which joins the two KIII's networks. The external BCI object is subtracted to the KII matrix before the next output KII signal is calculated through the 2nd order ODE. The signal is subtracted to the semi-periodic signal to restore the chaotic attributes of nodes within the KIV network.

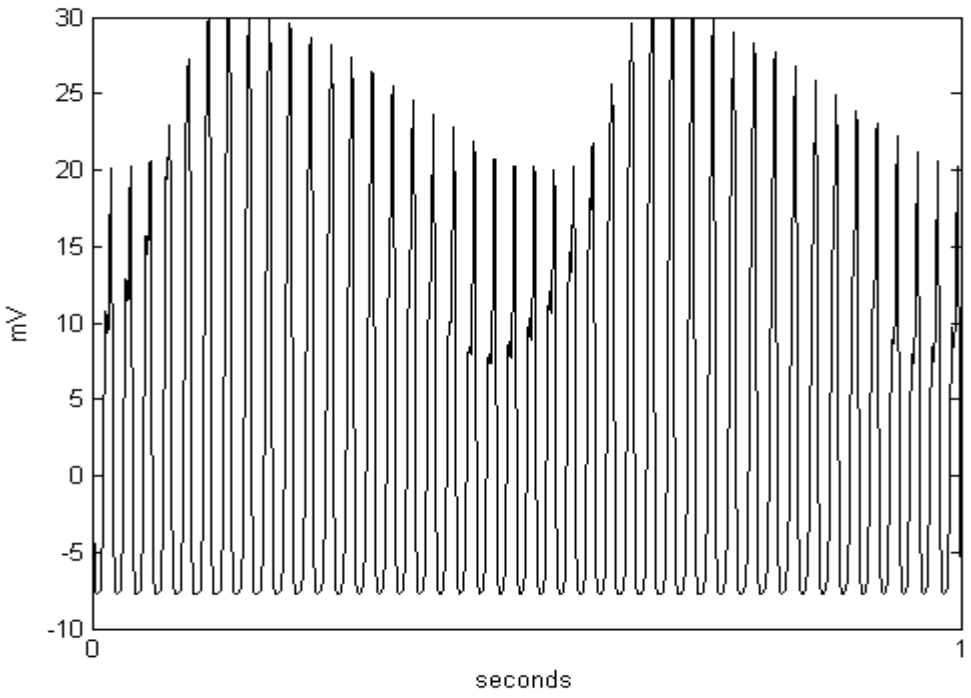


Figure 47. De-synchronization input signal

The external interface to the KIV model is a 42 Hz signal, with a 21 ms pulse width applied over the seizure state time series.

Therefore, the amplitude of the seizure state is reduced to normal neural state values, as shown in Figure 48.

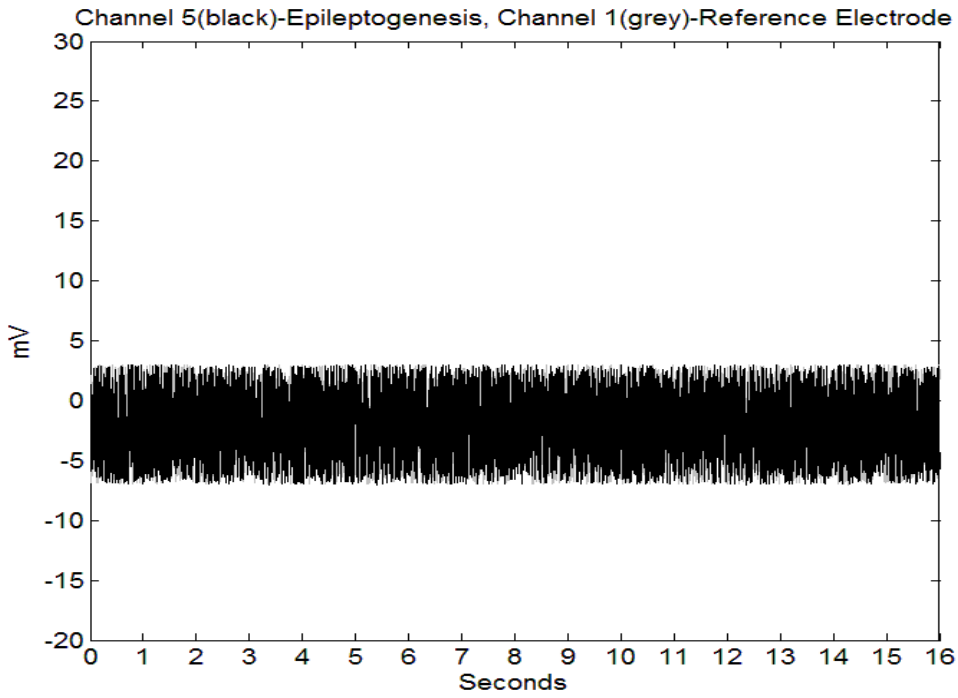


Figure 48. EEG figures on the right side featuring the restored simulated EEG where the normal chaotic state is exhibited in the diagram.

PSD analysis of the KIV model depicts the same low alpha value range as seen in the normal EEG time series (Figure 49).

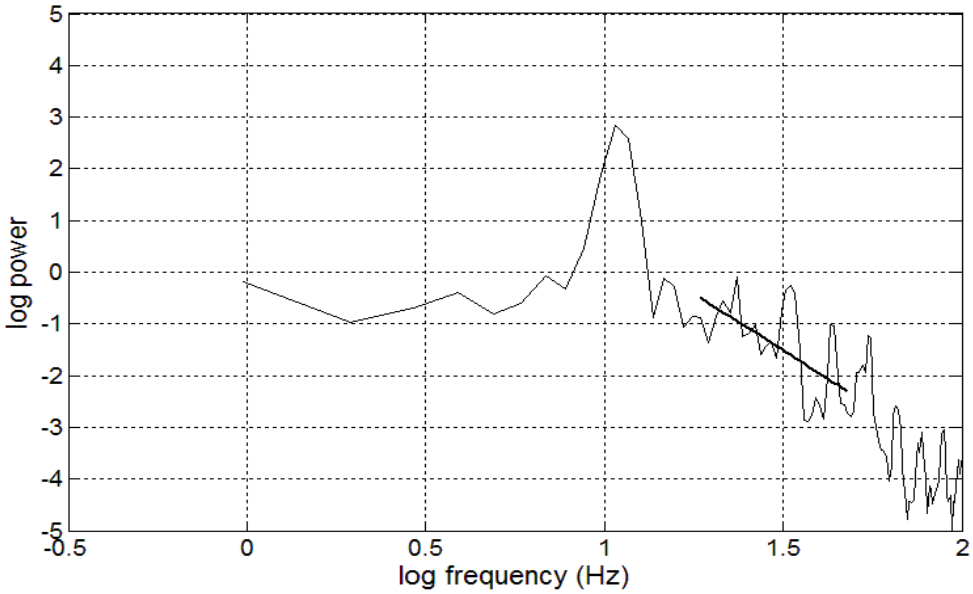


Figure 49. The restored state with alpha = -2.73

The following PSD values (Table 4) show that the restoration state also exhibits small standard deviation across the simulated patient EEGs, while maintaining the alpha range of 2 to 3.

Table 5

Power law exponent (α) function of the PSD regression function ($1/f^{\alpha}$) Simulated EEG using KIV Model

Restore with Titration Signal

	A1 (25)	A2 (25)	A3 (25)	A4 (25)	A5 (25)
$\bar{\alpha}$	2.47	2.52	2.61	2.69	2.77
σ_{α}	0.18	0.17	0.17	0.17	0.16
Min(α)	2.13	2.17	2.28	2.36	2.44
Max(α)	2.65	2.70	2.78	2.89	2.95

Phase diagrams show the same chaotic attractor as found in the normal and simulated EEG as well as autocorrelation displays depicts the restored signal back to its original non-periodic state as seen in Figures 50 and 51, respectively.

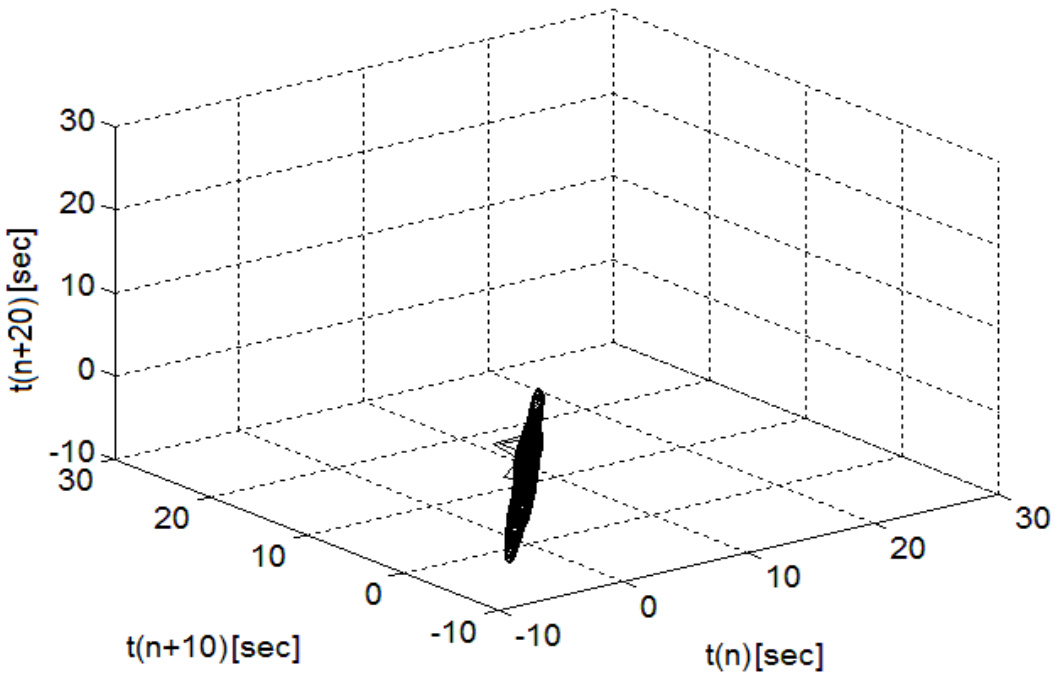


Figure 50. The phase diagram of the restored state of the simulated EEG.

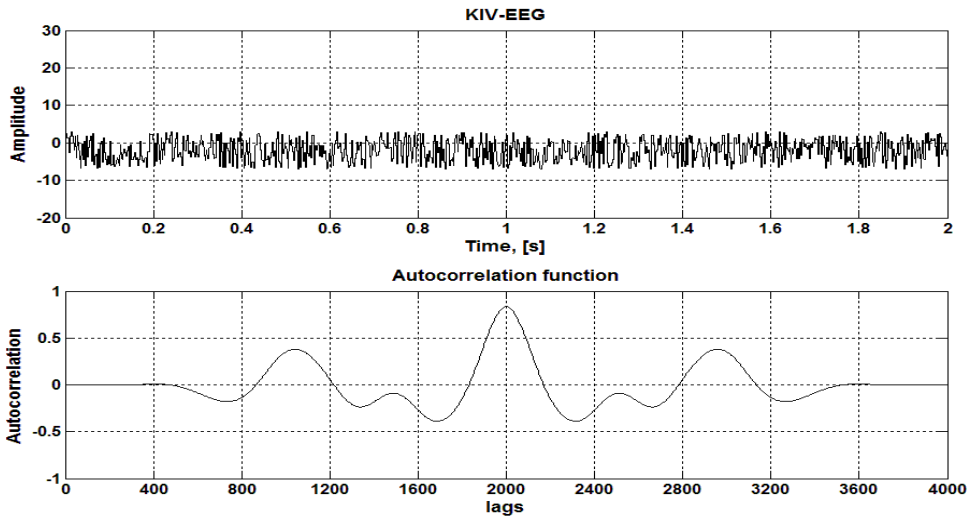


Figure 51. The autocorrelation of the restored state of the simulated EEG.

The external signal that restores the KIV model back to its initial state causes the runaway inhibitory signal to become 'rebalanced', since the external signal may provide the excitatory signal needed to restore the signal. Through the EEG simulator, electrical titration methodologies can be tested, and researched in order to provide an insight to the manner VNS treatment reduces the effects of the seizure state.

External stimulation applied to the KIV may accomplish one of the following seizure reduction realizations:

1) A reduction of runaway inhibitory activations that enable the restoration of the 'normal' state.
2) A 'rebalancing' of excitatory and inhibitory activations that overcome runaway inhibitory activations through the external excitatory input that act as excitatory activations.

Therefore, the amplitude and phase of the seizure state is reduced to normal neural state values.

CHAPTER 8: CONCLUSION AND FUTURE WORK

Conclusion Discussion

The KIV model enables brain pathology simulation and analysis through internal excitatory and inhibitory node tuning of KII networks and external coupling of KIII and KII networks within the KIV. Through this model, electrical titration therapies can be simulated and tested that could lead to a deeper understanding of the underpinnings of mesoscopic pathological neuron behavior.

Initially, we demonstrated seizure behavior through the KIII model to simulate the 'epileptogenesis' process where a local group of neurons exhibit hyperexcitability through the imbalance of excitatory and inhibitory neuron populations. An instability develops in which some inhibitory neurons become more disinhibited (excited) and others more inhibited (less active) to the point where there is a discharge that is manifested in a massive compound inhibitory post-synaptic potential (IPSP) of the excitatory neurons (Freeman, 1972). Several statistical properties were employed to characterize the KIII model as described in the literature (Freeman, 1972). Autocorrelation functions and power spectral analysis were utilized to characterize the 'seizure' state of the KIII EEG. Additionally, phase diagrams demonstrated abnormal brain states of the seizure EEG in the form of a limit cycle.

Attributes of mesoscopic neuron hyperexcitability have the characteristics of increased amplitude and change of phase or frequency of the neuron's activations. KIII model seizure parameterization was implemented in one of the KIII objects of the KIV network in order to model local neuronal hyperexcitability, hence establishing the local

area of seizure behavior or epileptogenesis. By increasing the external weighting of the networks, the local group of neuron's activation behavior can spread throughout the KIV network. The epileptogenesis process is commonly accepted to be closely associated with changes in neuronal synchronization in a network of components that may be spatially distributed (Fisch & Spehlmann, 1999). Therefore, this process demonstrates that seizure precursors may begin locally and then expand spatially, and even "entrain" to other brain structures before reaching the critical mass required to initiate a clinical seizure (Engel, 2007).

The KIV exhibits normal and abnormal seizure states through the KIII and KIV parameterization discussed in this dissertation. Quantification measures were applied to demonstrate biological equivalence between human EEG and simulated EEG. The PSD values in the normal state of human and simulated EEG conforms roughly to 1/f having a slope between 2.02 and 2.93 (Table 2) due to the mean firing rates of neurons being held at a critical value throughout the cortex, homeostatically stabilized locally by their thresholds and refractory periods (Freeman, 2004). The PSD values in the seizure state of human and simulated EEG conforms roughly to 1/f having a slope between 3.41 and 3.91 (Table 3) due to the high power in both the low and high theta ranges, corresponding to the 3/s wave that dominated the EEG. PSD values found during the EEG time series and the simulated EEG time series exhibit low standard deviation during the seizure states per patient. This activity may be due to the entrainment of large scale neural population whose power per frequency is limited to the theta range.

Autocorrelation of the human EEG and simulated EEG output during the seizure state demonstrates regularity in peaks of the time offset or time 'lag' of the EEG time

series. This regularity implies a dominant frequency is present in the EEG time series, which is found in the PSD diagrams to be 3 Hz, therefore conforming to the 3/s waveform found during seizure activity.

Phase diagrams of the EEG time series demonstrate that the normal/abnormal brain states exhibit high and low chaotic dimensionality in the form of a chaotic attractors (Figures 45a and c) vs. the abnormal seizure EEG in the form of a limit cycle (Figures 45b and d). Plots of the EEG time series by a fixed time offset reveal the dynamics of the EEG signal as the time series may converge, diverge or move in aperiodic motion (Freeman, 1986). Chaotic attractors are the manifestation of cortical columns and layers and their noisy neural interactions. KIII's and KIV's are governed by chaotic attractors, which is also the kind of dynamics observed in healthy brains (Kozma, 2003). Conversely, unbalanced feedback between the excitatory and inhibitory components produces periodic oscillations which are governed by a limit cycle attractor, as found in seizure behavior.

Restoration of the KIV seizure state back to the initial chaotic normal state is accomplished through the introduction of an external signal into the KIV network. This external signal is composed of the inverse of the seizure signal at slightly higher amplitude added to a KII signal. When this signal is applied to the KIV as the next action potential is calculated, the output of the KIV displays the restored normal state, as seen in Figure 47. As stated in the previous chapter, external stimulation applied to the KIV may accomplish one of the following seizure reduction realizations; a reduction of runaway inhibitory activations that enable the restoration of the 'normal' state, or a 'rebalancing'

of excitatory and inhibitory activations that overcome runaway inhibitory activations through the external excitatory input that act as excitatory activations.

The restoration state also exhibits small standard deviation across the simulated patient EEGs, while maintaining the alpha range of 2 to 3. Figure 47 shows the restored simulated EEG after an external input has been added to the system. Phase diagrams show the same chaotic attractor as found in the normal and simulated EEG as well as autocorrelation displays depicts the restored signal back to its original non-periodic state.

Through the EEG simulator, electrical titration methodologies can be tested and researched in order to provide an insight to the manner how electrical titration therapy reduces the effects of the seizure state, as well as other brain pathologies.

Areas of Continuing Research

- **Utilize the simulated EEG model to study other brain pathologies, i.e., Parkinson's, Alzheimer's, schizophrenia, depression, etc.** Develop brain models using the KIV in order to capture EEG characteristics to better understand the underpinnings of each disease. Brain stimulation techniques have been applied to patients with Parkinson's as well as depression, therefore brain stimulation techniques can be applied to the KIV to determine which titration method can alleviate the pathological effect on the model. In this manner, electrical titration methods can be optimized per patient.
- **Incorporate additional quantitative measurements to KIV model to further develop the model into a mesoscopic brain simulator.** Additional measurement tools such as Morlet wavelets can find wave packets of differing amplitudes and phases. This study would enable the location of brain pathologies with signature

waveforms that can be adjusted to fit a particular segment in an EEG time series, with various degrees of signal fitting. Also, the incorporation of Short-Term Lyapunov exponents (STLMax) would enable a fine-tuned analysis tool to locate changes of non-linear dynamic artifacts within a time series, such as the changes in the EEG signal as it moves in various degrees of non-linearity to a semi-periodic state.

- **Tune the KIV to produce cognitive processing attributes, such as phase transitions in order to model cognitive processing features along with brain pathologies.** Cognitive processing tools could be applied, such as the application of the analytic amplitude and phase of an EEG signal to locate phase transitions as it relates to consciousness and brain pathologies. Additionally, adaptive filtering tools can be developed such as Knife Edge filters and neural network implementations for enhanced frequency change detection relating to brain state changes of awake, rest and seizure states.
- **Incorporate additional nodes/neurons in the network to approach a closer scale of limbic system modeling.** A KIV model with at least 64 nodes would bring the model closer to the high density EEG monitoring equivalence of human EEGs of the limbic system of the brain. High density EEG recordings of 64 channels could be emulated to capture larger neuron population dynamics, such as cognitive processing and global cortical changes due to simulated brain pathologies. While this model would be computationally expensive, the benefit of this model would enable the study how the unity of conscious perception is brought about by the distributed activities of the central nervous system.

Additionally, studies of how brain pathologies affect conscious perception as seen in Alzheimer and schizophrenia patients could be accomplished. Models of decreased or heightened perception could help explain the differing levels of consciousness.

- **Develop KIV into real-time action potential generator with a controller interface in order to change KIV parameters dynamically so that internal and external parameter adjustments can be made to finely tune EEG output.** Dynamic KIV parameter adjustment would enable closer patient EEG emulation, and also enable external electrical titration therapies to be incorporated into the model to demonstrate different brain pathology reduction techniques. External stimulation amplitude, frequency, pulse duration and width can be adjusted and applied to the simulated EEG in order to provide an optimized patient-based electrical titration pacing remedy.

REFERENCES

Andrade, D.M., Zumsteg, D., Hamani, C., Hodaie, M., Sarkissian, S., Lozano, A.M., & Wennberg, R.A. (2006). Long term follow-up of patients with thalamic deep brain stimulation for epilepsy. *Neurology*, *66(10)*, 1571–1573.

Bak, P. (1996). *How Nature Works: the science of self-organized criticality,* New York: Springer-Verlag.

Berger, T.W., Song, D., Chan, R.H., & Marmarelis, V.Z. (2010). The Neurobiological Basis of Cognition: Identification by Multi-Input, Multi-output Nonlinear Dynamic Modeling: A method is proposed for measuring and modeling human long-term memory formation by mathematical analysis and computer simulation of nerve-cell dynamics. *Proc IEEE Inst Electr Electron Eng, 4:98(3)*, 356-374.

Bertram, E. (2007). The relevance of kindling for human epilepsy. *Epilepsia*, *48(2)*, 65–74.

Bhavaraju N.C., Frei, M.G., & Osario, I. (2006). Analog Seizure Detection and Performance Evaluation. *IEEE Transactions on Biomedical Engineering*, 53, 2.

Breakspear, M., Roberts, J.A., Terry, J.R., Rodrigues, S., Mahant, N., & Robinson, P.A. (2006). A Unifying Explanation of Primary Generalized Seizures Through Nonlinear Brain Modeling and Bifurcation Analysis. *Cerebral Cortex*, *16*, 1296—1313.

Chakravarthy, N., Sabesan, S., Iasemidis, L.D., & Tsakalis, K. (2007). Controlling Synchronization in a Neuron-Level Population Model, Journal: *International Journal of Neural Systems - IJNS* , *17(2)*, 123-138

Chang H.J., Freeman W.J., & Burke B.C. (1998). Optimization of olfactory model in software to give 1/f power spectra reveals numerical instabilities in solution governed by aperiodic (chaotic) attractor. *Neural Networks*, *11*, 449–466.

Critchley, M. (1949). *Sir William Gower, 1845-1915. A Biographical Appreciation*. London, William Heinemann.

Eeckman, F.H., & Freeman, W.J. (1991). Asymmetric sigmoid nonlinearity in the rt olfactory system, Brain Resarch, *557*, 13-21.

Elger C.E., & Lehnertz K. (1998). Seizure prediction by non-linear time series analysis of brain electrical activity. *Eur J Neurosci*, *10*, 786–789.

Ellis, T.L., Stevens, A. (2008). Deep brain stimulation for medically refractory epilepsy. *Neurosurg Focus, 25(3)*, E11.

Engel J., Pedley T.A., Aicardi J., & Dichter M.A. (2007). *Epilepsy: A Comprehensive Textbook, (Vol. 1 2 ed.)* Philadelphia, PA: Lippincott Williams & Wilkins.

Faul, S., Boylan, G., Connolly, S., Marnane, L., & Lightbody, G. (2005). An Evaluation of Automated Neonatal Seizure Detection Methods. *Clinical Neurophysiology*, *116*, 1533-1541.

Fisch B.J., & Spehlmann R. (1999). *Fisch and Spehlmann's EEG primer: basic principles of digital and analog EEG. (3rd ed.)* Amsterdam, The Netherlands: Elsevier.

Freeman, W.J. (1962). Alterations in prepyriform evoked potential in relation to stimulus intensity, *Exp. Neurol*, *6*, 70–84.

Freeman, W.J. (1972). Waves, pulses and the theory of neural masses. *Progr. Theoret. Biol*., *2*, 87–165.

Freeman, W.J. (1975). *Mass Action in the Nervous System*. New York: Academic Press

Freeman, W.J. (1986). Petit Mal Seizure Spikes in Olfactory Bulb and Cortex Caused by Runaway Inhibition After Exhaustion of Excitation. *Brain Research Reviews, 11*, 259–284.

Freeman, W.J. (2000a). *How brains make up their minds.* New York: Columbia University Press.

Freeman, W.J. (2000b). A proposed name for aperiodic brain activity: stochastic chaos. *Neural Networks, 13(1)*, 11-13.

Freeman W.J. (2006). Origin, structure, and role of background EEG activity. Part 4. Neural Frame Simulation. *Neurophysiol.,* 117, 572-589.

Freeman W.J., Holmes M.D., West G.A., & Vanhatalo S. (2006). Dynamics of human neocortex that optimizes its stability and flexibility. *Int J Intell Syst,* 21, 1–21.

Freeman, W.J., Chang, H., Burke, B., Rose, P., & Badler, J. (1997). Taming chaos: Stabilizing aperiodic attractors by noise. *IEEE Transactions on Circuits and Systems*, *44*, 987–996.

Freeman, W.J., Kozma R., & Werbos P. (2001). Biocomplexity: Adaptive behavior in complex stochastic dynamical systems. *BioSystems*, *59*, 109-123.

Freeman, W.J., & Rogers, L.J. (2002). Fine Temporal Resolution of Analytic Phase Reveals Episodic Synchronization by State Differences in Gamma EEGs *J. Neurophysiol*., *87*, 937-945.

Freeman, W.J. (2004). Origin, structure, and role of background EEG activity. Part 2. Amplitude. *Clin. Neurophysiol*, *115*, 2089-2107.

Freeman, W.J., & Zhai, J. (2009) Simulated power spectral density (PSD) of background electrocorticogram (ECoG). *Cognitive Neurodynamics, 3(1)*, 97-103.

Frohlich, F., Sejnowski, T.J., & Bazhenov, M. (2010). Network Bistability Mediates Spontaneous Transitions between Normal and Pathological Brain States. *The Journal of Neuroscience*, *30(32)*, 10734 –10743.

Frost, M., Gates, J., Helmers, S.L., Wheless, J.W., Levisohn, P., Tardo, C., & Conry, J.A. (2001). Vagus nerve stimulation in children with refractory seizures associated with Lennox-Gastaut syndrome. *Epilepsia*, *42(9)*, 1148-1152.

Good, L.B., Sabesan, S., Marsh, S.T., Tsakalis, K., Treiman, D., & Iasemidis L. (2009). Control of synchronization of brain dynamics leads to control of epileptic seizures in rodents. *Int J Neural Syst, 19(3)*, 173-196.

Gotman, J., Flanagan, D., Zhang, J., & Rosenblatt, B. (1997). Automated Seizure Detection in the Newborn: Methods and Initial Evaluation. *Clinical Neurophysiology*, *103*, 356-362.

Gregory, R. L. (1987). *The Oxford Companion to The Mind*. Oxford University Press (USA).

Halpern, C.H., Samadani, U., Litt, B., Jaggi, J.L., & Baltuch, G.H. (2008). Deep Brain Stimulation for Epilepsy. *Neurotherapeutics*, *5*, 59–67.

Ilin, R., & Kozma, R. (2006). Stability of coupled excitatory-inhibitory neural populations & application to control multistable systems. *Phys. Lett. A*, *360*, 66-83.

Jackson, H.J. (1878). On the affections of speech from disease of the brain. *Brain*, *1*, 304-330.

Johnson, M.D., Miocinovic, S., McIntyre, C.C., & Vitek, J.L. (2008). Mechanisms and targets of deep brain stimulation in movement disorders. *Neurotherapeutics*, *5(2)*, 294–308.

Kaneko K. (1990). Clustering, coding, switching, hierarchical ordering, and control in a network of chaotic elements. *Physica D, 41*, 137–172.

Kannathal, N., Puthusserypady, S.K., & Min, L.C. (2006). Elman neural networks for dynamic modeling of epileptic EEG. *Conf Proc IEEE Eng Med Biol Soc*, *1*, 6145-8.

Karceski S. (2007). Electrical stimulation devices in the treatment of epilepsy. *Acta Neurochir* Suppl. *97(2)*, 247-59.

Kerrigan, J.F., Litt, B., Fisher, R.S., Crantoun, S., French, J.A., Blum, D.E., Dichter, M., Shetter, A., Baltuch, G., Jaggi, J., Krone, S., Brodie, M.A., Rise, M., & Graves, N. (2004). Electrical Stimulation of the Anterior Nucleus of the Thalamus for the Treatment of Intractable Epilepsy, *Epilepsia*, *45(4)*, 346-354

Kossoff, E.H, Pyzik, P.L., Rubenstein, J.E., Bergqvist, A.G. C., Buchhalter, J.R., Donner, E.J., Nordli Jr., D.R., & Wheless, J.W. (2007). Combined Ketogenic Diet and Vagus Nerve Stimulation: Rational Polytherapy?, *Epilepsia, 48(1)*, 77–81.

Kozma, R. (2003). On the Constructive Role of Noise in Stabilizing Itinerant Trajectories in Chaotic Dynamical Memories. *Chaos, Special Issue on Chaotic Itinerancy, 13(3)*, 1078-1090.

Kozma, R. (2007). Intentional systems: Review of neurodynamics, modeling, and robotics implementation, *Physics of Life Reviews*, *4(4)*.

Kozma, R., Aghazarian, H., Huntsberger, T., Tunstel, E., & Freeman, WJ. (2007). Computational Aspects of Cognition and Consciousness in Intelligent Devices. *IEEE Computational Intelligence Magazine*, 53-64.

Kozma, R., & Freeman, W.J. (2003). Basic Principles of the KIV Model and its application to the Navigation Problem. *Journal of Integrative Neuroscience*, *2(1)*, 125-146.

Lehnertz, K., & Elger, C.E. (1998). Can epileptic seizures be predicted? Evidence from nonlinear time series analysis of brain electrical activity. *Phys Rev Lett*, *80*, 5019–22.

Lesser, R.P., Lee, H.W., Webber, W.R., Prince, B., Crone, N.E., & Miglioretti, D.L. (2008). Short-term variations in response distribution to cortical stimulation. *Brain*. *131(6)*, 1528–39.

Liu, A., Hahn, J., Heldt, G., & Coen, R. (1992). Detection of Neonatal Seizures through Computerized EEG Analysis. *Clinical Neurophysiology*, *82*, 30-37

Lopes da Silva, F., Blanes, W., Kalitzin, S.N., Parra, J., Suffczynski, P., & Velis, D.N. (2003). Epilepsies as dynamical diseases of brain systems: basic models of the transition between normal and epileptic activity. *Epilepsia*. *44(12)*, 72-83.

Marrosu, F., Santoni, F., & Pulighedda, M. (2003). Correlaton between GABA(A) receptor density and vagus nerve stimulation in individuals with drug resistant partial epilepsy. *Epilepsy Res, 55*, 59-70.

Merrill, D.R., Bikson, M., & Jefferys, J.G. (2005). Electrical stimulation of excitable tissue: design of efficaciousand safe protocols. *J Neurosci Methods*, *141(2)*, 171–98.

Myers, M.H., & Kozma R. (2007). Modeling Brain Electrical Activity Involving Vagus Nerve Stimulation, Proc. Int. Conf. Artificial Neural Networks in Engineering, ANNIE2007, November 11-14, 2007, St. Louis, MO. *Intelligent Engineering Systems Through Artificial Neural Networks*, *17*, 3-8, ASME Press.

Myers, M.H., Kozma R., & Freeman W.J. (2008). Studies in Synchronization in KIV Model. In R. Wang et al. (Eds). *Advances in Cognitive Neurodynamics, ICCN07*, *2*, 207-211, Berlin, Heidelberg: Springer Verlag.

Navarro, V., Le Van Quyen, M., Martinerir, J., Rudrauf, D., Baulac, M., & Menini, C. (2007). Loss of Phase Synchrony in an Animal Model of Partial Status Epilepticus. *Neuroscience, 148*, 304-313.

Osorio, I., Frei, M. G., Sunderam, S., Giftakis, J., Bhavaraju, N. C., Schaffner, S. F., & Wilkinson, S. B. (2005). Automated seizure abatement in humans using electrical stimulation. *Annals of Neurology*, *57*, 258–268.

Pritchard, W.S., & Duke, D.W. (1995). Measuring "Chaos" in the brain: A tutorial review of EEG Dimension Estimation. *Brain and Cognition*, *27*, 353-397.

Richardson, K.A., Gluckman, B.J., Weinstein, S.L., Glosch, C.E., Moon, J.B., Gwinn, R.P., Gale, K., & Schiff, S.J. (2003). In vivo modulation of epileptiform activity with radial hippocampal electric fields. *Epilepsia*, *44*, 768–777.

Robinson, P.A., Rennie, C.J., Rowe, D.L., O'Connor, S.C., Wright, J.J., Gordon, E., & Whitehouse, R.W. (2003). Neurophysical modeling of brain dynamics. *Neuropsychopharmacology*, Suppl 1, S74-9.

Sackellares J.C., Iasemidis L.D., Shiau D.S., Gilmore R.L., & Roper S.N. (2000). Epilepsy—when chaos fails. In K. Lehnertz, J. Arnhold, P. Grassberger, C.E. Elger, (Eds). *Chaos in the Brain*. (pp. 112–33) Singapore: World Scientific.

Schelter, B., Winterhalder, M., Maiwald, T., Brandt, A., Schad, A., Schulze-Bonhage, A., & Timmer, J. (2006). Testing statistical significance of multivariate time series analysis techniques for epileptic seizure prediction, *Chaos*, *16(1),* 013108.

Skarda, C.A., & Freeman, W. J. (1987). How brains make chaos in order to make sense of the world. *Behavioral and Brain Sciences*, *10*, 161-195.

Stam, C.J. (2005). Nonlinear dynamical analysis of EEG and MEG: Review of an emerging field. *Clinical Neurophysiology, 116(10)*, 2266-2301.

Stam, C.J., Pritchard, W.S. (1999). Dynamics underlying rhythmic and non-rhythmic variants of abnormal, waking delta activity. *Int J Psychophysiol, 34(1)*, 5-20.

Strogatz, S.H. (2000). *Nonlinear dynamics and chaos: with applications to physics, biology, chemistry, and engineering*. Cambridge, MA, Westview Press.

Sunderam, S., Gluckman, B., Reato, D., & Bikson, M. (2010). Toward rational design of electrical stimulation strategies for epilepsy control. *Epilepsy Behav, 17(1)*, 6-22.

Takeshita, S., Sato, Y.S., & Bahar S. (2007). Transitions between multistable states as a model of epileptic seizure dynamics, *PHYSICAL REVIEW, E 75*, 051925.

Tsakalis, K., Chakravarthy, N., & Iasemidis, L. (2005). Control of Epileptic Seizures: Models of Chaotic Oscillator Networks, *Proceedings of the 44th IEEE Conference on Decision and Control, and the European Control Conference*, Seville, Spain, December 2005.

Tsuda, I. (1996). A new type of self-organization associated with chaotic dynamics in neural networks, *Int. J. Neural Networks, 7*, 451–459

van Gelder, T., & Port, R. F. (1995). It's about time: an overview of the dynamical approach to cognition. In R. F. Port and T. van Gelder, (Eds) *Mind As Motion: Explorations in the Dynamics of Cognition*, (pp. 1-43) Massachusetts Institute of Technology, Cambridge, MA.

Varela F., Lachaux J-P, Rodriguez E, & Martinerie J. (2001). The brain-web: phase synchronization and large-scale integration. *Nature Rev Neurosci, 2*, 229-239.

Velasco, F., Velasco, M., Velasco, A.L., Jimenez, F., Marquez, I., & Rise, M. (1995). Electrical stimulation of the centromedian thalamic nucleus in control of seizures: long-term studies. *Epilepsia*, *36(1)*, 63–71.

Velazquez J.L., Cortez M.A., Carter S. III, & Wennberg R. (2003). Dynamical regimes underlying epileptiform events: role of instabilities and bifurcations in brain activity. *Physica D, 186*, 205-20.

Wendling, F., & Bartolomei, F. (2001) Modeling EEG signals and interpreting measures of relationship during temporal-lobe seizures: an approach to the study of epileptogenic networks. *Epileptic Disord,* Special Issue: 67-78.

Wheless, J.W., Clarke, D.F., Arzimanoglou, A., & Carpenter, D. (2007). Treatment of pediatric epilepsy: European expert opinion, *Epileptic Disorders, 9(4)*, 353-412.

Wheless J.W. (2006). Intractable epilepsy: A survey of patients and caregivers. *Epilepsy Behav*, *8(4)*, 756-64.

Wheless, J.W., & Baumgartner, J. (2004). Vagus nerve stimulation therapy. *Drugs of Today*. *40(6)*, 501-15.

Wheless, J.W., & Maggio, V. (2002). Vagus nerve stimulation therapy in patients younger than 18 years. *Neurology*, *24*, 59(6 Suppl 4), S21-5.

Wilrich, P. (2004). *Frontiers in statistical quality control*. *7*, 136, Physica-Verlag Heidelberg.

Wolf A., Swift J. Swinney H., & Vastano, J. (1985). Determining Lyapunov exponents from a time series. *Physica D: Nonlinear Phenomena*, *16(3)*, 285-317.

Yang, H.J., Peng, K.R., Hu, S.J., & Liu, Y. (2007) Inhibiting effect of vagal nerve stimulation to seizures in epileptic process of rats. *Neurosci Bull., 23(6)*, 336-40.

Zabara, J. (1985). Time course of seizure control to brief, repetitive stimuli. *Epilepsia, 26*, 518.

Zagon, A., & Kemeny, A.A. (2000). Slow hyperpolarization in cortical neurons: a possible mechanism behind vagus nerve stimulation therapy for refractory epilepsy, *Epilepsia*, *41(11)*, 1382-138.

APPENDIX A : INSTITUTIONAL REVIEW BOARD (IRB)

THE UNIVERSITY OF MEMPHIS

Institutional Review Board

To: Mark Myers Computer Science

From: Chair, Institutional Review Board for the Protection of Human Subjects Administration 315

Subject: Analysis of Phase-Jumps in EEG output Related to Cognitive Processing (E04-199)

Approval Date: June 8, 2004

This is to notify you that the Institutional Review Board has designated the above referenced protocol as exempt from the full federal regulations. This project was reviewed in accordance with all applicable statutes and regulations as well as ethical principles.

When the project is finished or terminated, please complete the attached Notice of Completion and send to the Board in Administration 315.

Approval for this protocol does not expire. However, any change to the protocol must be reviewed and approved by the board prior to implementing the change.

Chair, Institutional Review Board
The University of Memphis

APPENDIX B : KIV SOURCE CODE

The list of files in the KIV files

k2_Klayer.m - constuctor of the KII layer data structure
k2_new.m - constuctor for the KII network
k2_parameters.m - sets parameters of a KII network
k2_run.m - simulates KII network by running it with given inputs
k2_SolveNextStep.m - updates the KII network for one time step of integration with numberic Runge-Kutta method
k3_config.m - creates internal data structure for a KIII network
k3_delay2.m - computes the distributed delayed connections
k3_diag.m - returns diagnostic parameters for a KIII network
k3_filt.m - filters out the internal signal for read-out
k3_formKIImatrix.m - forms connections weight matrix of KII network inside KIII layer
k3_formKlink2.m - forms link between layers structures of KIII network
k3_formKSlink.m - forms link between layers structures of KIII network
k3_formLateralMatrix1.m - forms a lateral connections weight matrix inside a KIII network
k3_habituation.m - computes habituation learning update
k3_hebb.m - computes Hebbian learning update
k3_Klayer.m - creates layer of KII networks in a KIII
k3_Klayer_SolveNextStep2.m - updates one layer in a KIII network for one time step of integration
k3_KSlink.m - creates links between layers in a KIII network
k3_KSLink_Calc.m - calculates delayed links in KIII network
k3_new.m - creates new KIII networks and runs it for warm-up period
k3_parameters.m - sets parameters for KIII network data structure
k3_q.m - computes sigmoid transfer function
k3_run.m - simulates the KIII network for given inputs
k3_SolveNextStep.m - updates KIII network for one time step integration
k3_SolveNodeRungeKutta.m - implements Runge-Kutta numerical method for integration for the network update
k3_train.m - simulates the network for training inputs and makes the training updates of the connections in a KIII network
k4_new.m - updates KIII and KII networks and runs it for warm-up period
k4_k2_solveNextStep.m - updates KII network within KIV network for one time step integration
k4_k3_SolveNextStep.m - updates KIII network within KIV network for one time step integration
k4_Klayer_SolveNextStep2.m - updates one layer in a KIII network for one time step of integration
k4_run.m - updates KIV networks and runs it for active period
k4_train.m - This function returns and k4 network after training with the input from both the k3's

k4_update.m - This function updates the K4 network by updating the cortical hippocampal and amydale links one time step
k4_parameters.m - sets parameters for KIV network data structure
k2_stim.m – a kII network used as an external input to reduce the simulated seizure produced by the KIV network.
autocorr_eeg.m – autocorrelation function used to measure the correlation of a time series against a time offset segment of the same series.
psd_analysis.m – power spectral density measurement function of simulated EEG time series.

k2_Klayer.m

```
function K = k2_Klayer( params, w, wLat)
% returns k-layer object with the following arguments
%
%   params      - parameters object with all needed constants
%   w           - weights matrix of full k-ii
%   wLat        - is the matrix of lateral weights

% set time counter to zero
K.t = 0;

% set buffers for activation and its derivative to zeros
K.A = zeros(4*params.MM, params.buf_size);
K.B = zeros(4*params.MM, params.buf_size);

% set buffer for current input
K.I = zeros(4*params.MM, 1);

% set global layer weight matrix
K.OMEGA = wLat;
for j=1:params.MM
    K.OMEGA((j-1)*4+1:j*4,(j-1)*4+1:j*4) = w;
end
```

k2_new.m

```
function [K] = k2_new(MM)
%
% function k2_new returns new K_II object with
% given size of input layer MM, that is number
% of full k-ii sets in the layer
%
% At this moment all parameters are set to some
% default values used in the lab for k-sets.
%
% The default parameters for K_II sets in are as follows:
%
%   k3_formKIImatrix(1.8, 1, 2, 0.8);   % w_ee, w_ei, w_ie, w_ii
%
% K-set object has the following fields:
%
%             L: [1x1 struct]   - layer object (see k3_Klayer)
```

```
%             t: 400            - current total number of time steps
run
%        params: [struct]       - parameters of the network
%      buf_size: 80             - size of activation buffer in time
steps
%     it_active: 100            - number of time steps for active phase
%      it_relax: 300            - number of time steps for relaxation
%            MM: 10             - size of input
%
% see also k2_run(), k2_parameters(), k3_formKIImatrix, k3_Klayer,
k3_formLateralMatrix1
%
% by bileon, August 2004

% call k2_parameters for all weiths of full k_ii
params = k2_parameters(MM);

%Lateral Connection Matrix
wLat = k3_formLateralMatrix1(params.wLat(1), params.wLat(2),
params.MM);

%internal weights
w = k3_formKIImatrix(params.kii(1), params.kii(2), params.kii(3),
params.kii(4));

% prepare a KLayer
% give it params, k-ii weights matrix and lateral weights matrix
K = k2_Klayer(params, w, wLat);

% store parameter object inside network object
K.params = params;

% prepare input (small perturbation)
var1 = 0.1;

in = [  var1; 0.0; 0.0; 0.0];
in = repmat(in, params.MM, 1);
% in(1:4:end) = 2 * var1 *( rand(params.MM,1) - 1/2 );
%in(1:4:end) = 2 * var1 *( rand(params.MM,1) - 1/2 );

% run network for one active (with input) iteration
for t = 1:1
    K.t = K.t + 1;
    K = k2_SolveNextStep(K, in );
end

% run network for res of parmas.it_warmup iterations with zero input
for t = 2:params.it_warmup
    K.t = K.t + 1;
    K = k2_SolveNextStep(K, zeros(size(in)));
end
```

k2_parameters.m

```
function [params] = k2_parameters(input_size)
% Returns an object that contains parameters of K-II
% network which include:
%
%   it_warmup       - number of time steps for warm up period
%   it_active       - number of time steps for active phase
%   it_relax        - number of time steps for inactive phase
%   dT              - time resolution for RK solver ( sec. / time steps
)
%   K-II weights    - weights w_ee, w_ei, w_ie, w_ii for K-II network
%   wLat            - lateral intra-layer connections weights
% all parameters are packed into 'params' data structure
%
% see also k2_new, k2_run, k3_formKIImatrix
%
% by bileon, August 2004

% size of input
params.MM = input_size;

% time steps for warm up period, active and inactive phase of k3 cycle

% mm
params.it_warmup = 1000;
params.it_active = 1000;
params.it_relax  = 50;
%mm

% params.it_warmup = 10000;
% params.it_active = 10000;
% params.it_relax  = 10000;


% size of internal buffer to store activation history
params.buf_size = max( params.it_active, 70);

% weights of full k-ii
% params.kii = [1.8, 1.0, 2.0, 0.9];
params.kii = [2.0 1.5 2.0 1.0]; % Orig
%params.kii = [0.05, 2.20, 2.20, 3.4];
%params.kii = [0.0 0.0 0.0 0.0];

% time resolution for Runge-Kutta solver
params.dT = 0.5;

% lateral itra-layer connections
% wLat_ee and wLat_ii
params.wLat = [1.5 -1.0];
% params.wLat = [0.15 -0.1];
% params.wLat = [0.0 0.0];

% Constants of differential equation
params.a = 0.22;
params.b = 0.72;
```

```
% Constant in asymetric sigmoid
params.sig = 5;
```

k2_run.m

```
function [K] = k2_run(K, in)
% function k2_run returns K-II object K after
% it has been run for K.it_active time steps with
% each input sample vector from set 'in' (active period)
% followed by K.it_relax time steps of relaxation period
% (zero input) and set of std(activations) vectors
% that is standard deviation of 50 time steps of
% active phase for excitatory units of third layer
% for every input sample.
%
% Input set in is composed of input samples row by row, that
% is in = [ x_1; x_2; ... x_k].
%
% Sigma is composed of vectors that are computed as follows:
%
%   sigma(i,:) = std( activations );
%
% where 'activations' is set of activation for all top K-II
% excitatory units in the third layer.
%
% see also k2_new, k2, k2_train
%
% by bileon, June 2004


if size(in) ~= 0

    sigma = [];

    % iterate over input vectors
    for i = 1 : size(in,1)

        tic;

        % take input from set_train
        inp = repmat(zeros(4,1), K.params.MM,1);
        inp(1:4:end) = in(i,1:K.params.MM);

        % get current network time
        total_t = K.t;

        % run ACTIVE phase for 100 iterations
        for t = 1:K.params.it_active
            K.t = t + total_t;
            K = k2_SolveNextStep(K, inp ); %zeros(size(in))
        end

        % run INACTIVE phase for 300 iterations
```

```
        for t = K.params.it_active + 1 : K.params.it_active +
K.params.it_relax
            K.t = t + total_t;
            K = k2_SolveNextStep(K, zeros(size(inp)));
        end

        ['iteration ' int2str(i) ' of ' int2str(size(in,1)) ' time
elapsed ' num2str(toc)]

    end
else
    [' no input ... nothing done ']
end
```

k2_SolveNextStep.m

```
function K = k3_SolveNextStep2( K, extIn )
%-------------------------------------------------------------------%
%--The University of Memphis, Department of Mathematical Sciences--%
% File:      k3_Klayer_SolveNextStep2
% Author:    Roman Ilin
% Date:      5/2003
% Purpose:   given array of external inputs into the layer, extIn, 4*N
by 1
%            calculates the next time step activations for all KII sets
%-------------------------------------------------------------------%

k = mod(K.t - 1, K.params.buf_size) + 1;

% index in a round robin buffer by modulo K.Hist,
% turn it around if modulo division is zero
if mod( K.t - 1, K.params.buf_size) == 0
    kk = K.params.buf_size;
else
    kk = mod( K.t - 1, K.params.buf_size);
end

in = K.OMEGA*k3_q( K.A(:,kk),K.params.sig );
%in = K.OMEGA*k3_q( K.A(:,mod(K.t - 1, K.Hist) + K.Hist),K.Zig ) /
max(1, K.N - 1) ;
%NOT SURE WHICH WAY IS CORRECT
%in = k3_q(K.OMEGA* K.A(:,mod(K.t - 1, K.Hist) + K.Hist),K.Zig );
K.I = extIn;

% sovle ODE and store current results
[K.A(:, k) , K.B(:, k)] = k3_SolveNodeRungeKutta(K.params.a,
K.params.b, K.params.dT, K.A(:, kk ), K.B(:, kk ), K.I + in  );
```

k3_config.m

```
function K = k3_config( varargin )
% KCONFIG Is a container for KSet configuration info
% The following are the properties which can be set by the user or
% can assume their default values
% Property               Possible Values         Default
Comment
%
% 'general.type'            'kii', 'kiii'           'kii'
type of configuration
% 'general.inputtype'       'file', 'matrix'        'matrix'
source of input
% 'general.history'         1..xx                   40
length of 'history'
% 'general.time'            1..xx                   3200
simulation time, msec
% 'general_dt'              0..xx                   0.5
time step, msec
%
% 'kii.units'               1..xx / integer         1
number of units in KII
% 'kii_schema'              'name'                  ''
schema for single KII
% 'kii.w'                   4 by 4 matrix           0
KII connection weights
% 'kii.lateralw'            4xx by 4xx matrix       0
KII lateral connection weights
% 'kii.filename'            'name'                 'kii_out'
KII output file name
% 'kii.zig'                 0..xx                   5
zigmiod coefficient
% 'kii.A0'                  4xx by 1                []
KII init activations
% 'kii.B0'                  4xx by 1                []
KII  derivatives of init activations
%
% 'kiii.layers'             1..5                    1
Number of KIII layers
%
% 'kiii.layer1.kii.units'   1..xx                   1
Number of Layer1 units
% 'kiii.layer1.kii.schema'  'name'                  ''
layer1 KII schema
% 'kiii.layer1.w'           4 by 4 matrix           0
layer1 KII connection weights
% 'kiii.layer1.lateralw'    4xx by 4xx marix        0
layer1 lateral connections
% 'kiii.layer1.filename'    'name'                  'layer1_out'
output filename
% 'kiii.layer1.zig'         0..x                    5
zigmoid coefficient
% 'kiii.layer1.A0'          4zz by 1 matrix         []
initial activations
```

```
%
% 'kiii.layer2.kii.units'
% 'kiii.layer2.kii.schema'
% 'kiii.layer2.w'
% 'kiii.layer2.lateralw'
% 'kiii.layer2.filename'
% 'kiii.layer2.zig'         0..x                    5
zigmoid coefficient
%
% 'kiii.layer3.kii.units'
% 'kiii.layer3.kii.schema'
% 'kiii.layer3.w'
% 'kiii.layer3.lateralw'
% 'kiii.layer3.filename'
% 'kiii.layer3.zig'         0..x                    5
zigmoid coefficient
%
% 'kiii.layer4.kii.units'
% 'kiii.layer4.kii.schema'
% 'kiii.layer4.w'
% 'kiii.layer4.lateralw'
% 'kiii.layer4.filename'
% 'kiii.layer4.zig'         0..x                    5
zigmoid coefficient
%
% 'kiii.layer5.kii.units'
% 'kiii.layer5.kii.schema'
% 'kiii.layer5.w'
% 'kiii.layer5.lateralw'
% 'kiii.layer5.filename'
% 'kiii.layer5.zig'         0..x                    5
zigmoid coefficient
%
% 'kiii.schema'             'name'                  ''
name of KII connevction schema
%
% 'kiii.klink.1.1'          k3_link2 object           k3_link2( 'null',
0);
% 'kiii.klink.1.2'
% 'kiii.klink.1.3'
% 'kiii.klink.1.4'
% 'kiii.klink.1.5'
%
% 'kiii.klink.2.1'          k3_link2 object           k3_link2( 'null',
0);
% 'kiii.klink.2.2'
% 'kiii.klink.2.3'
% 'kiii.klink.2.4'
% 'kiii.klink.2.5'
%
% 'kiii.klink.3.1'          k3_link2 object           k3_link2( 'null',
0);
% 'kiii.klink.3.2'
% 'kiii.klink.3.3'
% 'kiii.klink.3.4'
% 'kiii.klink.3.5'
```

```
%
% 'kiii.klink.4.1'          k3_link2 object           k3_link2( 'null',
0);
% 'kiii.klink.4.2'
% 'kiii.klink.4.3'
% 'kiii.klink.4.4'
% 'kiii.klink.4.5'
%
% 'kiii.klink.5.1'          k3_link2 object           k3_link2( 'null',
0);
% 'kiii.klink.5.2'
% 'kiii.klink.5.3'
% 'kiii.klink.5.4'
% 'kiii.klink.5.5'


K.General_Type = 'kii';
K.General_InputType = 'matrix';

% K.General_History = 40;
% K.General_Time  = 40*80;
%K.General_dT    = 0.5;

K.KII_Units = 1;
K.KII_Schema = '';
K.KII_W = zeros(4,4);
K.KII_lateralW = zeros(4*K.KII_Units, 4*K.KII_Units);
K.KII_filename = 'kii_out';
K.KII_Zig = 5;
K.KII_A0 = [];
K.KII_B0 = [];

K.KIII_Layers = 1;

K.KIII_Layer1_KII_Units = 1;
K.KIII_Layer1_KII_Schema = '';
K.KIII_Layer1_W = zeros(4,4);
K.KIII_Layer1_lateralW = zeros(4*K.KII_Units, 4*K.KII_Units);
K.KIII_Layer1_filename = 'layer1_out';
K.KIII_Layer1_Zig  = 5;
K.KIII_Layer1_A0  = [];

K.KIII_Layer2_KII_Units = 1;
K.KIII_Layer2_KII_Schema = '';
K.KIII_Layer2_W = zeros(4,4);
K.KIII_Layer2_lateralW = zeros(4*K.KII_Units, 4*K.KII_Units);
K.KIII_Layer2_filename = 'layer2_out';
K.KIII_Layer2_Zig  = 5;

K.KIII_Layer3_KII_Units = 1;
K.KIII_Layer3_KII_Schema = '';
K.KIII_Layer3_W = zeros(4,4);
K.KIII_Layer3_lateralW = zeros(4*K.KII_Units, 4*K.KII_Units);
K.KIII_Layer3_filename = 'layer3_out';
K.KIII_Layer3_Zig  = 5;
```

```
K.KIII_Layer4_KII_Units = 1;
K.KIII_Layer4_KII_Schema = '';
K.KIII_Layer4_W = zeros(4,4);
K.KIII_Layer4_lateralW = zeros(4*K.KII_Units, 4*K.KII_Units);
K.KIII_Layer4_filename = 'layer4_out';
K.KIII_Layer4_Zig  = 5;

K.KIII_Layer5_KII_Units = 1;
K.KIII_Layer5_KII_Schema = '';
K.KIII_Layer5_W = zeros(4,4);
K.KIII_Layer5_lateralW = zeros(4*K.KII_Units, 4*K.KII_Units);
K.KIII_Layer5_filename = 'layer5_out';
K.KIII_Layer5_Zig  = 5;

K.KIII_Schema = '';

K.KIII_Klink_1_1 = k3_link2( 'null', 0, [], []);
K.KIII_Klink_1_2 = k3_link2( 'null', 0, [], []);
K.KIII_Klink_1_3 = k3_link2( 'null', 0, [], []);
K.KIII_Klink_1_4 = k3_link2( 'null', 0, [], []);
K.KIII_Klink_1_5 = k3_link2( 'null', 0, [], []);

K.KIII_Klink_2_1 = k3_link2( 'null', 0, [], []);
K.KIII_Klink_2_2 = k3_link2( 'null', 0, [], []);
K.KIII_Klink_2_3 = k3_link2( 'null', 0, [], []);
K.KIII_Klink_2_4 = k3_link2( 'null', 0, [], []);
K.KIII_Klink_2_5 = k3_link2( 'null', 0, [], []);

K.KIII_Klink_3_1 = k3_link2( 'null', 0, [], []);
K.KIII_Klink_3_2 = k3_link2( 'null', 0, [], []);
K.KIII_Klink_3_3 = k3_link2( 'null', 0, [], []);
K.KIII_Klink_3_4 = k3_link2( 'null', 0, [], []);
K.KIII_Klink_3_5 = k3_link2( 'null', 0, [], []);

K.KIII_Klink_4_1 = k3_link2( 'null', 0, [], []);
K.KIII_Klink_4_2 = k3_link2( 'null', 0, [], []);
K.KIII_Klink_4_3 = k3_link2( 'null', 0, [], []);
K.KIII_Klink_4_4 = k3_link2( 'null', 0, [], []);
K.KIII_Klink_4_5 = k3_link2( 'null', 0, [], []);

K.KIII_Klink_5_1 = k3_link2( 'null', 0, [], []);
K.KIII_Klink_5_2 = k3_link2( 'null', 0, [], []);
K.KIII_Klink_5_3 = k3_link2( 'null', 0, [], []);
K.KIII_Klink_5_4 = k3_link2( 'null', 0, [], []);
K.KIII_Klink_5_5 = k3_link2( 'null', 0, [], []);

K.KIII_KSlink = cell(5,5);  %for now assume max 5 layers

for j=1:2:nargin
    switch lower(varargin{j})
    case 'general.type'
        K.General_Type = varargin{j+1};
    case 'general.inputtype'
        K.General_InputType = varargin{j+1};
```

```
case 'buffer.size'
    K.buf_size = varargin{j+1};
case  'general.time'
    K.General_Time  = varargin{j+1};
case  'general_dt'
    K.General_dT    = varargin{j+1};


case 'kii.units'
    K.KII_Units = varargin{j+1};
case 'kii_schema'
    K.KII_Schema = varargin{j+1};
case 'kii.w'
    K.KII_W = varargin{j+1};
case 'kii.lateralw'
    K.KII_lateralW = varargin{j+1};
case 'kii.filename'
    K.KII_filename = varargin{j+1};
case 'kii.zig'
    K.KII_Zig = varargin{j+1};
case 'kii.a0'
    K.KII_A0 = varargin{j+1};
case 'kii.b0'
    K.KII_B0 = varargin{j+1};


case 'kiii.layers'
    K.KIII_Layers = varargin{j+1};

case 'kiii.layer1.kii.units'
    K.KIII_Layer1_KII_Units = varargin{j+1};
case 'kiii.layer1.kii.schema'
    K.KIII_Layer1_KII_Schema = varargin{j+1};
case 'kiii.layer1.w'
    K.KIII_Layer1_W = varargin{j+1};
case 'kiii.layer1.lateralw'
    K.KIII_Layer1_lateralW = varargin{j+1};
case 'kiii.layer1.filename'
    K.KIII_Layer1_filename = varargin{j+1};
case 'kiii.layer1.zig'
    K.KIII_Layer1_Zig  = varargin{j+1};


case 'kiii.layer2.kii.units'
    K.KIII_Layer2_KII_Units = varargin{j+1};
case 'kiii.layer2.kii.schema'
    K.KIII_Layer2_KII_Schema = varargin{j+1};
case 'kiii.layer2.w'
    K.KIII_Layer2_W = varargin{j+1};
case 'kiii.layer2.lateralw'
    K.KIII_Layer2_lateralW = varargin{j+1};
case 'kiii.layer2.filename'
    K.KIII_Layer2_filename = varargin{j+1};
case 'kiii.layer2.zig'
```

```
    K.KIII_Layer2_Zig  = varargin{j+1};


case 'kiii.layer3.kii.units'
    K.KIII_Layer3_KII_Units = varargin{j+1};
case 'kiii.layer3.kii.schema'
    K.KIII_Layer3_KII_Schema = varargin{j+1};
case 'kiii.layer3.w'
    K.KIII_Layer3_W = varargin{j+1};
case 'kiii.layer3.lateralw'
    K.KIII_Layer3_lateralW = varargin{j+1};
case 'kiii.layer3.filename'
    K.KIII_Layer3_filename = varargin{j+1};
case 'kiii.layer3.zig'
    K.KIII_Layer3_Zig  = varargin{j+1};

case 'kiii.layer4.kii.units'
    K.KIII_Layer4_KII_Units = varargin{j+1};
case 'kiii.layer4.kii.schema'
    K.KIII_Layer4_KII_Schema = varargin{j+1};
case 'kiii.layer4.w'
    K.KIII_Layer4_W = varargin{j+1};
case 'kiii.layer4.lateralw'
    K.KIII_Layer4_lateralW = varargin{j+1};
case 'kiii.layer4.filename'
    K.KIII_Layer4_filename = varargin{j+1};
case 'kiii.layer4.zig'
    K.KIII_Layer4_Zig  = varargin{j+1};

case 'kiii.layer5.kii.units'
    K.KIII_Layer5_KII_Units = varargin{j+1};
case 'kiii.layer5.kii.schema'
    K.KIII_Layer5_KII_Schema = varargin{j+1};
case 'kiii.layer5.w'
    K.KIII_Layer5_W = varargin{j+1};
case 'kiii.layer5.lateralw'
    K.KIII_Layer5_lateralW = varargin{j+1};
case 'kiii.layer5.filename'
    K.KIII_Layer5_filename = varargin{j+1};
case 'kiii.layer5.zig'
    K.KIII_Layer5_Zig  = varargin{j+1};

case 'kiii.schema'
    K.KIII_Schema   = varargin{j+1};

%now use spacial links instead of klinks
case 'kiii.kslink.1.1'
    sz = size(K.KIII_KSlink{1,1}, 2);
    K.KIII_KSlink{1,1}(sz+1) =  varargin{j+1};
case 'kiii.kslink.1.2'
    sz = size(K.KIII_KSlink{1,2}, 2);
    K.KIII_KSlink{1,2}(sz+1) =  varargin{j+1};
case 'kiii.kslink.1.3'
    sz = size(K.KIII_KSlink{1,3}, 2);
    K.KIII_KSlink{1,3}(sz+1) =  varargin{j+1};
case 'kiii.kslink.1.4'
```

```
    sz = size(K.KIII_KSlink{1,4}, 2);
    K.KIII_KSlink{1,4}(sz+1) =  varargin{j+1};
case 'kiii.kslink.1.5'
    sz = size(K.KIII_KSlink{1,5}, 2);
    K.KIII_KSlink{1,5}(sz+1) =  varargin{j+1};


case 'kiii.kslink.2.1'
    sz = size(K.KIII_KSlink{2,1}, 2);
    K.KIII_KSlink{2,1}(sz+1) =  varargin{j+1};
case 'kiii.kslink.2.2'
    sz = size(K.KIII_KSlink{2,2}, 2);
    K.KIII_KSlink{2,2}(sz+1) =  varargin{j+1};
case 'kiii.kslink.2.3'
    sz = size(K.KIII_KSlink{2,3}, 2);
    K.KIII_KSlink{2,3}(sz+1) =  varargin{j+1};
case 'kiii.kslink.2.4'
    sz = size(K.KIII_KSlink{2,4}, 2);
    K.KIII_KSlink{2,4}(sz+1) =  varargin{j+1};
case 'kiii.kslink.2.5'
    sz = size(K.KIII_KSlink{2,5}, 2);
    K.KIII_KSlink{2,5}(sz+1) =  varargin{j+1};


case 'kiii.kslink.3.1'
    sz = size(K.KIII_KSlink{3,1}, 2);
    K.KIII_KSlink{3,1}(sz+1) =  varargin{j+1};
case 'kiii.kslink.3.2'
    sz = size(K.KIII_KSlink{3,2}, 2);
    K.KIII_KSlink{3,2}(sz+1) =  varargin{j+1};
case 'kiii.kslink.3.3'
    sz = size(K.KIII_KSlink{3,3}, 2);
    K.KIII_KSlink{3,3}(sz+1) =  varargin{j+1};
case 'kiii.kslink.3.4'
    sz = size(K.KIII_KSlink{3,4}, 2);
    K.KIII_KSlink{3,4}(sz+1) =  varargin{j+1};
case 'kiii.kslink.3.5'
    sz = size(K.KIII_KSlink{3,5}, 2);
    K.KIII_KSlink{3,5}(sz+1) =  varargin{j+1};

case 'kiii.kslink.4.1'
    sz = size(K.KIII_KSlink{4,1}, 2);
    K.KIII_KSlink{4,1}(sz+1) =  varargin{j+1};
case 'kiii.kslink.4.2'
    sz = size(K.KIII_KSlink{4,2}, 2);
    K.KIII_KSlink{4,2}(sz+1) =  varargin{j+1};
case 'kiii.kslink.4.3'
    sz = size(K.KIII_KSlink{4,3}, 2);
    K.KIII_KSlink{4,3}(sz+1) =  varargin{j+1};
case 'kiii.kslink.4.4'
    sz = size(K.KIII_KSlink{4,4}, 2);
    K.KIII_KSlink{4,4}(sz+1) =  varargin{j+1};
case 'kiii.kslink.4.5'
    sz = size(K.KIII_KSlink{4,5}, 2);
    K.KIII_KSlink{4,5}(sz+1) =  varargin{j+1};
```

```
    case 'kiii.kslink.5.1'
        sz = size(K.KIII_KSlink{5,1}, 2);
        K.KIII_KSlink{5,1}(sz+1) =  varargin{j+1};
    case 'kiii.kslink.5.2'
        sz = size(K.KIII_KSlink{5,2}, 2);
        K.KIII_KSlink{5,2}(sz+1) =  varargin{j+1};
    case 'kiii.kslink.5.3'
        sz = size(K.KIII_KSlink{5,3}, 2);
        K.KIII_KSlink{5,3}(sz+1) =  varargin{j+1};
    case 'kiii.kslink.5.4'
        sz = size(K.KIII_KSlink{5,4}, 2);
        K.KIII_KSlink{5,4}(sz+1) =  varargin{j+1};
    case 'kiii.kslink.5.5'
        sz = size(K.KIII_KSlink{5,5}, 2);
        K.KIII_KSlink{5,5}(sz+1) =  varargin{j+1};

    case 'MM'
        K.MM = varargin{j+1};

    end
end
```

k3_delay2.m

```
function a = KDelay2( K, b, N, M)
% function a = KDelay2( K, b, N, M )
% performs delay opeation between two Klayer's.  Assumes that there is
only one
%   link between the layer's, so that the signals from each (KII) unit
of the
%   sending layer are summed together, transformed using the Klink2's
schema
%   and then sent to each KII unit of the receiving layer.  This
assumtion of course causes
%   amplification of the signals if the numbers of units in the
receiving and sending
%   layers are diferent, so an appropriate amplification coefficient
should be defined
%   in the Klink2 object.
% Arguments:
% Klink2 K, input matrix b, Integer N (from units), Integer M (to
units)
% K         is Klink2 object
% b     is signal from KLayer, consisting of N KII units, 4*N by Hist
% a     returned signal, 1 by 4*M, where M is the number of KII units
in the receiving layer
% method    can be 'average' or 'converge-diverge'

%A. get the delayed (filtered) values
%DD = K.D; % k3_Klink2_get(K, 'D');
a1 = zeros(4,1);
hist = size(K.D,3);

if size(b,2) < hist
       error('KDelay2:The input signal does not have sufficient history
length');
end

r = size(b,1);
method = K.method;

switch method
    case 'average'
        %ONE WAY OF DOING THIS: AVERAGE OF INPUT LAYER FEEDS ALL
ELEMENTS OF THE
        %OUTPUT LAYER
        for l=1:4:r
            temp = zeros(4,1);
            for j=1:4
                temp(j) = sum(diag(reshape(K.D(:, j, :),4,hist) *
b(l:l+3,1:hist)'));
            end
            a1 = a1 + temp;
        end

        %B. build the input for all M units
        a = zeros(M*4,1);
        for k=1:M
            a(4*(k-1)+1:4*k,1) = a1;
```

```
        end
        a = a .* K.Amp;

    case 'converge-diverge'
        % ANOTHER WAY OF DOING IT: NOT AVERAGE FEEDS ALL, BUT NEIGHBORS
FEED INTO
        % NEIGHBORS
        % SUPPOSE, N >= M, then if C = N/M, we can say that the input
KII units with
        % index from floor(C*(i-1)+1) to floor(C*i+1) feed the output
unit i
        % What if N < M?  then we can say that output elements from
        % floor(C*(j-1)+1) to floor(C*j+1) receive from input element j
        % this needs to be developed for the case when N/M or M/N are
not
        % whole numbers.  For now just use the case with whole numbers
        error('KDelay2:"converge-diverge" has not been implemented yet,
SORRY');
    case 'converge-diverge-int'
        %converge-diverge with integer ratio of the number of
input/output or
        %output/input elements
        temp_out = zeros(N,4);  %keep the results of time delay here
        for l=1:4:r
            temp = zeros(1,4);
            for j=1:4
                temp(j) = sum(diag(reshape(K.D(:, j, :),4,hist) *
b(l:l+3,1:hist)'));
            end
            %now temp contains output of unit l
            temp_out((l+3)/4,:) = temp;
        end
        if N > M  %converge
            C = N/M;
            if floor(C)-C ~= 0
                error('KDelay2:the ratio of the number of KII in
input/output must be a whole number');
            end
            for j=1:M
                a(4*(j-1)+1:4*j)=sum(temp_out(C*(j-1)+1:C*j,:));
            end
            a = a .* K.Amp;
        elseif N == M  %equal
            for i=1:N
                a(4*(i-1)+1:4*i)=temp_out(i,:);
            end
            a = a .* K.Amp;
        else%diverge
            C = M/N;
            if floor(C)-C ~= 0
                error('KDelay2:the ratio of the number of KII in
input/output must be a whole number');
            end
            for i=1:N
                a(4*((C*(i-1)+1)-1)+1:4*C*i)=repmat(temp_out(i,:),1,C);
            end
            a = a .* K.Amp;
```

```
        end
    otherwise
        error('KDelay2:Unknown methods passed, has to be "converge-
diverge-int" or "average"');
end
```

k3_diag.m

```
function [pr, nt, bt] = k3_diag(x, q)
%
% returns three number which tell how each
% of the following tests perform on given input x signal
%
%   pr - PSD ratio test: ratio of the power in lower frequency range
%        to the power in the gamma range (5 - 20 Hz / 20-80 Hz)
%   nt - normality test, we want the signal to have normal distribution
%        jbtest is used ?????
%   bt  - balancing test, we want mean value of the signal be in the
%         neighbourhood of the maximum for the derivative of sigm
function
%
% takes in signal [x] and parameter of the sigmoid function [q]
%
% by Roman Ilin, bileon,  September 2004


% balancing test
bt = abs(log(q) - mean(x));

% psd ratio
[y ff] = psd(x);

low_low = 0.05;
low_up = 0.1;

gamma_low = 0.1;
gamma_up = 0.8;

xx = 1: max(size(y));
% lower freq. range summation for power
x_begin = max(find( xx < low_low * max(size(y))));
x_end = max(find( xx < low_up * max(size(y))));

s1 = 0;
for i= x_begin : x_end
    s1 = s1 + ( 1/max(size(y)) )* y(i);
end

% higher freq. range summation for power
x_begin = max(find( xx < gamma_low * max(size(y))));
x_end = max(find( xx < gamma_up * max(size(y))));

s2 = 0;
for i= x_begin : x_end
    s2 = s2 + ( 1/max(size(y)) )* y(i);
end
```

```
pr = s1/s2;


% nt - nothign is done yet
nt = 0;
```

k3_filt.m

```
% k3_filt.m nov-20-99
% Matlab script for filter design and testing
% prepared for INFO411 class Lab work
% by R. Kozma
% March 11, 1997

% create initial DATA FILE
% normally distributed random time series - white noise
%xx=randn(1,20*512);

function xxfil=k3_filt(x)

f=[0 0.08 0.08 0.16 0.16 1];
m=[0 0    1    1    0    0];
b=fir2(70,f,m);

fP3=[];
%plot(xx);
% check visually Gaussianity
%hist(xx,128); pause
% this is quite nice bell curve
% calculate the POWER SPECTRAL DENSITY
% the resolution of the spectrum is 1/512
% note: psd uses the default Hanning vindow
% to avoid aliasing and it takes the average
% of 20 spectra calculated by fft

%[zz yy]=psd(xx,256,1000);
%loglog(yy,zz,'r')
%hold
%pause(0.1)

% now we have plotted the spectrum of xx time series
% it is rather constant in accordance with the 'white'-ness
% it is not completely flat due to statistical errors
% you can smooth it by selecting longer time series
% e.g. using 100 instead of 20 segments in the xx=... line
% FILTER DESIGN
% you can select various filters with matlab
% here we design our own filter using 'fir2' function
% 'fir2' is a general-shape finite impulse response filter
% in systems theory this is related to moving average MA models
% first define the desired shape of the filter

%f=[0 0.08 0.08 0.16 0.16 1];
%m=[0 0    1    1    0    0];
```

```
% f is the vector of the frequency intervals
% where 1 is the Nyuist freq = half the sampling frequency
% m is the vector of the desired magnitudes
% note: size(f) = size(m)
% chech this by plotting
%plot(f,m)
%pause
% create an order 50 fir filter of this shape
% plot the resulting filter-shape
% note it is not 'exactly' the designed one
% you can increase the accuracy using higher order filters

%b=fir2(70,f,m);

%[h, w]=freqz(b,1,512);
%plot(f,m,w/pi,abs(h))
%pause
% now we are ready to FILTER the time series
% and also calculate the spectrum of the filtered signal

xxfil=filter(b,1,x);
%[yfilt ax]=psd(xxfil,256,1000);
%loglog(ax,yfilt);

fP3=[fP3; xxfil];

%end; %(for mm cycle)

% you see that only a narrow band of the spectrum remained
% we will use this procedure for preprocessing speech signals
% and generate Mel scale coefficients over pre-defined frequencies
% PHONEME MEL-SCALE COEFFICIENTS CALCULATION
% introduce Mel-scale central frequencies
%fmel=[86 173 256 430 516 603 689 775 947 1033 1120 1292 1550 1723 1981
...
%2325 2670 3015 3445 3962 4565 5254 6029 6997 8010 9216 11025];
```

k3_formKIImatrix.m

```
function  w  = k3_formKIImatrix( Kee, Kei, Kie, Kii )
%-------------------------------------------------------------------%
%--The University of Memphis, Department of Mathematical Sciences--%
% File:      formKIImatrix
% Author:    Roman Ilin
% Date:      5/25/2003
% Purpose:   returns 4 by 4 matrix for KII set given e, ei, ie, ii
weights
%-------------------------------------------------------------------%
w =     [0     Kee -Kie    -Kie; ...
          Kee   0   -Kie    0; ...
          Kei   Kei 0       -Kii; ...
          Kei  0   -Kii    0];
```

```
% w =     [   0       Kee     -Kie    -Kie; ...
%             Kee     0       -Kie    0; ...
%             Kei     Kei     0       Kii; ...
%             Kei     0       Kii     0];
%
% %w =     [   0       Kee     -Kei    -Kei; ...
%              Kee     0       -Kei    0; ...
%              Kie     Kie     0       -Kii; ...
%              Kie     0       -Kii    0];
```

k3_formKlink2.m

```
function [ K ] = formKlink2(  varargin )
%-------------------------------------------------------------------%
%--The Univrsity of Memphis, Department of Mathematical Sciences--%
% File:      formKlink2
% Author:    Roman Ilin
% Date:      6/17/2003
% Purpose:   Return Klink2 object with given connection schema
%            The input arguments allow to specify each connection's
%             weight  bi.aj.w  and delay bi.aj.n
%            only assume simple delays for now
%-------------------------------------------------------------------%
% CONSTRAINT:  max lenth of delay is 100
%-------------------------------------------------------------------%
length = 70;
D = zeros(4, 4, length+1);
Amp = 0;
connections = zeros(4,4);
method=[];
for j=1:2:nargin
    switch lower(varargin{j})
    case 'b1.a1.w'
        connections(1,1) = varargin{j+1};
    case 'b1.a2.w'
        connections(1,2) = varargin{j+1};
    case 'b1.a3.w'
        connections(1,3) = varargin{j+1};
    case 'b1.a4.w'
        connections(1,4) = varargin{j+1};
    case 'b2.a1.w'
        connections(2,1) = varargin{j+1};
    case 'b2.a2.w'
        connections(2,2) = varargin{j+1};
    case 'b2.a3.w'
        connections(2,3) = varargin{j+1};
    case 'b2.a4.w'
        connections(2,4) = varargin{j+1};
    case 'b3.a1.w'
        connections(3,1) = varargin{j+1};
    case 'b3.a2.w'
        connections(3,2) = varargin{j+1};
    case 'b3.a3.w'
        connections(3,3) = varargin{j+1};
    case 'b3.a4.w'
        connections(3,4) = varargin{j+1};
```

```
    case 'b4.a1.w'
        connections(4,1) = varargin{j+1};
    case 'b4.a2.w'
        connections(4,2) = varargin{j+1};
    case 'b4.a3.w'
        connections(4,3) = varargin{j+1};
    case 'b4.a4.w'
        connections(4,4) = varargin{j+1};
    case 'amp'
        Amp = varargin{j+1};
    case 'method'
        method = varargin{j+1};
    end
end

% Creating Transformation Matrix for Klink2 connecting
% two Klayers
% index 1: sending unit  b
% index 2: receiving unit a
% index 3: filter coefficients
%---------------from node b1----------------
%---------  1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5
% D(1,1,:) = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 .05 0 0 0 0 0 0 0 0]';%
to a1
% D(1,2,:) = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]';% to
a2
% D(1,3,:) = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 .25]';%
to a3
% D(1,4,:) = [0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0]';% to
a4
%----------------------------------------
for j=1:2:nargin
    switch lower(varargin{j})
    case 'b1.a1.n'
        D(1,1,:) =  [zeros(1, varargin{j+1}-1), connections(1,1),
zeros(1, length - varargin{j+1} + 1)]';
    case 'b1.a2.n'
        D(1,2,:) =  [zeros(1, varargin{j+1}-1), connections(1,2),
zeros(1, length - varargin{j+1} + 1)]';
    case 'b1.a3.n'
        D(1,3,:) =  [zeros(1, varargin{j+1}-1), connections(1,3),
zeros(1, length - varargin{j+1} + 1)]';
    case 'b1.a4.n'
        D(1,4,:) =  [zeros(1, varargin{j+1}-1), connections(1,4),
zeros(1, length - varargin{j+1} + 1)]';
    case 'b2.a1.n'
        D(2,1,:) =  [zeros(1, varargin{j+1}-1), connections(2,1),
zeros(1, length - varargin{j+1} + 1)]';
    case 'b2.a2.n'
        D(2,2,:) =  [zeros(1, varargin{j+1}-1), connections(2,2),
zeros(1, length - varargin{j+1} + 1)]';
    case 'b2.a3.n'
        D(2,3,:) =  [zeros(1, varargin{j+1}-1), connections(2,3),
zeros(1, length - varargin{j+1} + 1)]';
    case 'b2.a4.n'
        D(2,4,:) =  [zeros(1, varargin{j+1}-1), connections(2,4),
zeros(1, length - varargin{j+1} + 1)]';
```

```
    case 'b3.a1.n'
        D(3,1,:) =  [zeros(1, varargin{j+1}-1), connections(3,1),
zeros(1, length - varargin{j+1} + 1)]';
    case 'b3.a2.n'
        D(3,2,:) =  [zeros(1, varargin{j+1}-1), connections(3,2),
zeros(1, length - varargin{j+1} + 1)]';
    case 'b3.a3.n'
        D(3,3,:) =  [zeros(1, varargin{j+1}-1), connections(3,3),
zeros(1, length - varargin{j+1} + 1)]';
    case 'b3.a4.n'
        D(3,4,:) =  [zeros(1, varargin{j+1}-1), connections(3,4),
zeros(1, length - varargin{j+1} + 1)]';
    case 'b4.a1.n'
        D(4,1,:) =  [zeros(1, varargin{j+1}-1), connections(4,1),
zeros(1, length - varargin{j+1} + 1)]';
    case 'b4.a2.n'
        D(4,2,:) =  [zeros(1, varargin{j+1}-1), connections(4,2),
zeros(1, length - varargin{j+1} + 1)]';
    case 'b4.a3.n'
        D(4,3,:) =  [zeros(1, varargin{j+1}-1), connections(4,3),
zeros(1, length - varargin{j+1} + 1)]';
    case 'b4.a4.n'
        D(4,4,:) =  [zeros(1, varargin{j+1}-1), connections(4,4),
zeros(1, length - varargin{j+1} + 1)]';
    end
end

K = k3_link2( '', Amp, D, method );
```

k3_formKSlink.m

```
function [ K ] = k3_form_KSlink( varargin )
%-------------------------------------------------------------------%
%--The University of Memphis, Department of Mathematical Sciences--%
% File:      formk3_KSlink
% Author:    Roman Ilin
% Date:      2/15/200
% Purpose:   Return k3_KSlink object
%            The input arguments allow to specify the connections and
%            delays
%-------------------------------------------------------------------%
for j=1:2:nargin
    switch lower(varargin{j})
    case 'inidx'
        inidx = varargin{j+1};
    case 'outidx'
        outIDX = varargin{j+1};
    case 'klink'
        objKlink = varargin{j+1};
    end
end
if mod(size(inidx,2),4) ~= 0
    error('formk3_KSlink: wrong dimension of the input layer, must be
multiple of 4');
end
if mod(size(outIDX,2),4) ~= 0
    error('formk3_KSlink: wrong dimension of the output layer, must be
multiple of 4');
end
K = k3_KSlink( inidx, outIDX, objKlink );
```

k3_formLateralMatrix1.m

```
function w = k3_formLateralMatrix1( we, wi, NN )
%-------------------------------------------------------------------%
%--The University of Memphis, Department of Mathematical Sciences--%
% File:      formLateralMatrix1
% Author:    Roman Ilin
% Date:      5/25/2003
% Purpose:   returns 4*N by 4*N matrix for k3_Klayer lateral
connections
%           this is TYPE 1 connection schema, top and bottom nodes are
connected
%           the weights we and wi are factored by the inverse number of
units
%-------------------------------------------------------------------%
o = [ we 0 0 0; ...
        0 0 0 0; ...
        0 0 wi 0; ...
        0 0 0 0];
if NN > 1
    w = repmat(o, NN, NN) ./ (NN - 1);
else
    w = repmat(o, NN, NN);
end
```

k3_hebb.m

```
function [W]=k3_hebb(Activations,W, lr)
% returns updated weights matrix given activations for all
% units and current weight matrix. Takes 'Activatins' and
% 'W' in as parameters.
%
% The update of the weights 'W' is done based on Hebbian
% learining rule:
%
%   dW(i,j) = alpha * (sigma(i) - mean_s) * ( sigma(j) - mean_s);
%
% where mean_s is average standard deviation over all units'.
%
%
% see also k3_new, k3, k3_train, ke_run
%
%   by bileon, April 2004

% determine size of net
MM = size(W,1);

% learining rate
alpha = lr;

% compute std values for activation channels
sigma_act = std(k3_filt(Activations));

% mean standard deviation of all channels
mean_s = mean(sigma_act);

% compute weight updates and store in matrix
dW = zeros(MM);
for i = 1 : MM
    for j = 1 : MM
        dW(i,j) = alpha * (sigma_act(i) - mean_s) * ( sigma_act(j) -
mean_s);
    end
end

% delete uncorrelated, those <0
dW = max(dW, zeros(MM));

% update matrix
W = W + dW;

% clean main diagonal from near zero values
W = W - diag(diag(W));
```

k3_Klayer.m

```
function K = k3_Klayer( Zig, L, dT, KII_W, KII_schema_name, n, wLat,
IsFileOn, file_name, schema_name, A0, B0, params )
%-------------------------------------------------------------------%
%--The University of Memphis, Department of Mathematical Sciences--%
% File:      Klayer
% Author:    Roman Ilin
% Date:      5/2003
% Purpose:   KLAYER is the constructor for Klayer object
%-------------------------------------------------------------------%
%   K = Klayer(Zig, L, dT, KII_W, KII_schema_name, n, wLat, IsFileOn,
file_name, schema_name, A0, B0 )
% arguments
% Zig                   sigmoid function parameter
% L                     length of history
% dT                        time step, in milliseconds
% KII_W                  is single unit KII's weight matrix, can be
empty if KII_schema_name is
%                       specified
% KII_schema_name        is the name of the schema for KII unit
% n                      is the number of KII units in the layer
% wLat                   is the matrix of lateral weights, can be empty
if schema is specified
% IsFileOn
% file_name
% schema_name            is the name of pre-defined lateral connections
schema
%A0, B0                 the initial values for Activation and
Derivative for each Node, 4*N by 1

% PART 1
%------------------ATTRIBUTES-------------------
K.N = n;
K.FileOutputOn = IsFileOn;
K.FileName = ''; % file_name;
K.fid = 0; %fopen(K.FileName, 'w');

K.t = 0;

K.Zig = Zig;
K.h = dT;
K.Hist = L;

K.A = zeros(4*K.N, K.Hist );
K.B = zeros(4*K.N, K.Hist );
K.I = zeros(4*K.N, 1);
K.OUT = [];

K.swap = 0;
%apply initial conditions if passed in
if ~isempty(A0)
    K.A(:, K.Hist) = A0;
end
if ~isempty(B0)
    K.B(:, K.Hist) = B0;
end
```

```
K.a = params.a;
K.b = params.b;
K.params = params;

%-------------------------------------------------------

%PART 2
%-----------------LATERAL CONNECTIONS------------------
K.Schema = schema_name;
if ~isempty(K.Schema)
    switch lower(K.Schema)
        case 'meanfield_1'  %top elements are all connected
                o1 = [ 0.2 0 0 0; 0 0 0 0; 0 0 0 0; 0 0 0 0];
                o2 = [];
                for j=1:K.N
                    o2 = cat(1, o2, o1);
                end
                K.OMEGA = [];
                for j=1:K.N
                    K.OMEGA = cat(2, K.OMEGA, o2);
                end
        case 'neighbor'

        otherwise
       %throw error
        error('Schema name did not match any existing Lateral conn.
schema, Klayer constructor');

    end
else
    K.OMEGA = wLat;
end
%-------------------------------------------------------

% PART 3
%-----------------KII INNER CONNECTIONS-----------------
K.KII_Schema = KII_schema_name;
if ~isempty(K.KII_Schema)
    switch lower(K.KII_Schema)
    case 'k0'
        [Kee, Kei, Kie, Kii] = weights(0, 0, 0, 0);

    case 'ob'   %zero fixed point
        %[Kee, Kei, Kie, Kii] = weights(1.0, 1.0, 1.0, 0.9);
        [Kee, Kei, Kie, Kii] = weights(0.25, 1.5, 1.5, 1.8);
    case 'aon'  %positive fixed point
        %[Kee, Kei, Kie, Kii] = weights(1.5, 1.0, 1.0, 1.9);
        [Kee, Kei, Kie, Kii] = weights(1.4, 1.1, 1.1, 1.8);

    case 'pc'   %negative fixed point
        %[Kee, Kei, Kie, Kii] = weights(1.0, 1.2, 1.2, 1.7);
        [Kee, Kei, Kie, Kii] = weights(0.25, 1.4, 1.4, 1.8);

        %Parameters from Haizhon's Model
```

```
    case 'h_ob'   %zero fixed point
        [Kee, Kei, Kie, Kii] = weights(1.8, 1, 2, 0.8);
    case 'h_aon'  %positive fixed point
        [Kee, Kei, Kie, Kii] = weights(1.6, 1.6, 1.5, 2);
    case 'h_pc'   %negative fixed point
        [Kee, Kei, Kie, Kii] = weights(1.6, 1.9, 0.2, 1);

    otherwise
        %throw error
        error('Schema name did not match any existing KII schema,
Klayer constructor');
    end
    K.W = [0     Kee -Kie    -Kie; ...
           Kee   0   -Kie    0; ...
           Kei   Kei 0       -Kii; ...
           Kei   0   -Kii    0];

else
    K.W = KII_W;
end
for j=1:n
    K.OMEGA((j-1)*4+1:j*4,(j-1)*4+1:j*4) = K.W;
end
%-------------------------------------------------------------

% PART 4
%-----------------REGISTER CLASS-------------------------
%K = class( K, 'Klayer');



function [ee, ei, ie, ii] = weights(Wee, Wei, Wie, Wii)
ee = Wee; ei = Wei; ie = Wie; ii = Wii;
```

k3_Klayer_SolveNextStep2.m

```
function K = k3_Klayer_SolveNextStep2( K, extIn )
%-------------------------------------------------------------------%
%--The University of Memphis, Department of Mathematical Sciences--%
% File:      k3_Klayer_SolveNextStep2
% Author:    Roman Ilin
% Date:      5/2003
% Purpose:   given array of external inputs into the layer, extIn, 4*N
by 1
%            calculates the next time step activations for all KII sets
%-------------------------------------------------------------------%

k = mod(K.t - 1, K.Hist) + 1;

% index in a round robin buffer by modulo K.Hist,
% turn it around if modulo division is zero
if mod( K.t - 1, K.Hist) == 0
    kk = K.Hist;
else
    kk = mod( K.t - 1, K.Hist);
end
```

```
in = K.OMEGA*k3_q( K.A(:,kk),K.Zig );
%in = K.OMEGA*k3_q( K.A(:,mod(K.t - 1, K.Hist) + K.Hist),K.Zig ) /
max(1, K.N - 1) ;
%NOT SURE WHICH WAY IS CORRECT
%in = k3_q(K.OMEGA* K.A(:,mod(K.t - 1, K.Hist) + K.Hist),K.Zig );

    % N O I S E   A D D E D   H E R E
    % add noise, for each layer take K.params.noise(j) weight of noise
%     noise = zeros(K.params.MM * 4, 1);
%     noise(1:4:end) = K.params.bias + K.params.noise *
randn(K.params.MM,1);
%     in = in + noise;
% mm
     noise = zeros(K.params.MM * 4, 1);
%     noise(1:4:end) = randn(K.params.MM,1);
    % -mm add 4-15-2007
    %in = [ 1; 0; 0; 0; 2; 0; 0; 0; 3; 0; 0; 0];
    %in = [0.122; 0.144; 0.455; 1; 0.898; 0.344; 0.232; 2; 0.232;
0.655; 0.111; 3];
    in = in + noise;
% mm
K.I = extIn;

% sovle ODE and store current results
[K.A(:, k) , K.B(:, k)] = k3_SolveNodeRungeKutta(K.a, K.b, K.h, K.A(:,
kk ), K.B(:, kk ), K.I + in  );
K.A(:, k) = K.A(:, k) + noise;
```

k3_KSlink.m

```
function K = KSLink( inIndexes, outIndexes, link )
%-------------------------------------------------------------------%
%--The Univrsity of Memphis, Department of Mathematical Sciences--%
% File:      KSLink.m
% Author:    Roman Ilin
% Date:      2/15/200
% Purpose:   Constructor
%-------------------------------------------------------------------%
%KSLINK Describes the space-time link between two layers
%  inIdx and outIdx are the arrays of indexes of respectively input and
%  output layers.
%  KLink is the KLnik2 object describing the time-distributed link.
%  inIdx and outIdx arrays select which KII units of the input and
output
%  layer are participating in this link.  Therefore, the indexes must
be
%  the union of intervals 1:4, 5:8, 8:12, etc...
K.inIdx = inIndexes;    %indexes from the input layer
K.outIdx = outIndexes;   %indexes from the output layer
%K.inLayer = intLayerIn;  %index of the input layer, integer
%K.outLayer = intLayerOut; %index of the output layer, integer
K.Klink = link;    %k3_link2 object, to calc the delays
%K = class( K, 'KSLink');
```

k3_KSLink_Calc.m

```
function [ outActivations ] = k3_KSlink_Calc( K, inActivations ,
outsize)
%-------------------------------------------------------------------%
%--The Univrsity of Memphis, Department of Mathematical Sciences--%
% File:      k3_KSlink_Calc
% Author:    Roman Ilin
% Date:      2/15/200
% Purpose:   CALC Space-Time Delay
%-------------------------------------------------------------------%
%  Calculates the delayed signal from one layer to another
%  inActivations is a 2D array of activations from the inbound
% layer, and K.inIdx is the selector of which indexes are "working" for
% this link.  similarly K.outIdx determines which outbound layer
indexes
% are receiving the output.
%  !!! inIdx and outIdx must be the multiples of 4 since we select on
KII
%  unit basis, so if a layer has 10 KII sets in it, the possible
indexes
%  can be [1:4] or [5:8] or [9:12] etc. or any combination of them.
But
%  inIdx = [2:5] is NOT
VALID!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
outActivations = zeros(outsize,1);
outActivations(K.outIdx) = k3_delay2( K.Klink ,
inActivations(K.inIdx,:) , size(K.inIdx,2)/4, size(K.outIdx,2)/4);
```

k3_link2.m

```
function K = k3_link2( Schema_Name, Amp, D, method )
%-------------------------------------------------------------------%
%--The University of Memphis, Department of Mathematical Sciences--%
% File:      Klink2
% Author:    Roman Ilin
% Date:      5/2003
% Purpose:   Constructor for Klink2 object, which represent distributed
%            time delay link between two Klayer's
%-------------------------------------------------------------------%
%function K = Klink2( Schema_Name, Amp )
% Schema_Name       name of the predefined link schema
% Double Amp   - is the global amplification coefficient
% method - "average", "converge-diverge-int", etc.. methods of spacial
% convergence
% Detailed description: -- OLD
% KLink2        KLink2 describes connection between two KII units.
Each unit has 4 nodes, so we have
%           16 connection links, in general.  Each link is described
similar to  a Digital Filter,
%           by its coefficients.  If Ts and Tf and the start and end
time steps then the number of coefficients
%           must be Tf - Ts + 1.  ASSUME that each of 16 channels has
the same Tf and Ts (this is
%           easy to achieve by patching the shorter filters with zero
coefficients.
%
```

```
%           D is the 3-dimensional matrix, 4 by 4 by Tf - Ts + 1.  It
can be thought of as consisting of 16 columns
%           each of the columns is a digital filter applied to its
corresponding channel.
%   !IMPORTANT!   First dimension           - sending KII
%                 Second dimension  - receiving KII
%                 D can either be passed in, or filled internally using
Schema_Name
%   !IMPORTANT!   This class is used by global function KDelay2
%   if isa(Schema_Name, 'KLink2')
%         K = Schema_Name;
%         return;
%   end
K.Amp = Amp;
if isempty(method)
    K.method = 'average';  %default for now is average
else
    K.method = method;
end
K.D = D;
```

k3_new.m

```
function [K] = k3_new(MM)
%
% function k3_new returns new K_III object with
% given size of input layer MM, that is size of
% all three layers.
%
% At this moment all parameters are set to some
% default values used in the lab for k-sets. This
% function calls K-III constructor k3() and gives it
% three layer object prepared by function k3_Klayer().
%
% The default parameters for K_II sets in first, second
% and third layer are as follows:
%
% First layer:    k3_formKIImatrix(1.8, 1, 2, 0.8);   % w_ee, w_ei,
w_ie, w_ii
% Second layer:   k3_formKIImatrix(1.6, 1.6, 1.5, 2); % w_ee, w_ei,
w_ie, w_ii
% Third layer:    k3_formKIImatrix(1.6, 1.9, 0.2, 1); % w_ee, w_ei,
w_ie, w_ii
%
% K-set object has the following fields:
%
%             N: 3              - number of layers (double K-II layers)
%             L: [1x3 struct]   - array of layer objects (see
k3_Klayer)
%        Schema: ''             - schema name for some predifined
shemas
%             D: []             - links object
%             t: 400            - current total number of time steps
run
%        SLINKS: {5x5 cell}     - array of inter-layer links objecst
%      buf_size: 80             - size of activation buffer in time
steps
%     it_active: 100            - number of time steps for active phase
%      it_relax: 300            - number of time steps for relaxation
%            MM: 10             - size of input
%         STEPS: 1000           - total number of time steps (not used)
%
%
% see also k3, k3_train, k3_run, k3_formKIImatrix, k3_Klayer,
k3_formKlink2
%
% by bileon, June 2004

% call k3_parameters for all major parameters of k3
params = k3_parameters(MM);

% set up all parameters and instantiate the network
%number of units in layers
NN1 = params.MM;
NN2 = params.MM;
NN3 = params.MM;

% %input
```

```
% dT = 0.5;

%Lateral Connection Matrix
%connect a1 and a3 in each layer
we1 = params.wLat1(1);  wi1 = params.wLat1(2);
we2 = params.wLat2(1);  wi2 = params.wLat2(2);
we3 = params.wLat3(1);  wi3 = params.wLat3(2);

wLat1 = k3_formLateralMatrix1(we1, wi1, NN1);
wLat2 = k3_formLateralMatrix1(we2, wi2, NN2);
wLat3 = k3_formLateralMatrix1(we3, wi3, NN3);

%internal weights
w1 = k3_formKIImatrix(params.kii1(1), params.kii1(2), params.kii1(3),
params.kii1(4));
w2 = k3_formKIImatrix(params.kii2(1), params.kii2(2), params.kii2(3),
params.kii2(4));
w3 = k3_formKIImatrix(params.kii3(1), params.kii3(2), params.kii3(3),
params.kii3(4));

%klinks
%Weights = [0.15  0.6  0.05 0.25 -0.05 0.3];
Weights = [0.1529  0.5979  0.0518 0.2501 -0.0502 0.2789];
Weights = params.weights;

%klinks
k12 = k3_formKlink2('b1.a1.w', 1 , 'b1.a1.n', 1, 'Amp', Weights(1)/NN1,
'method', 'converge-diverge-int');
k13 = k3_formKlink2('b1.a1.w', 1 , 'b1.a1.n', 1, 'Amp', Weights(2)/NN1,
'method', 'converge-diverge-int');

k21 = k3_formKlink2('b1.a1.w', Weights(3), 'b1.a1.n', 17, 'b1.a3.w',
Weights(4), 'b1.a3.n', 25, 'Amp', 1/NN1, 'method', 'average');
k23 = k3_formKlink2('b1.a1.w', 1, 'b1.a1.n', 1, 'Amp', 0, 'method',
'converge-diverge-int');

k31 = k3_formKlink2('b3.a3.w', Weights(5), 'b3.a3.n', 25, 'Amp', 1/NN1,
'method', 'average');
k32 = k3_formKlink2('b1.a3.w', Weights(6), 'b1.a3.n', 25, 'Amp', 1/NN1,
'method', 'average');

%configuration
conf = k3_config(   'general.type', 'kiii', ...
                    'kiii.layers', 3, ...
                    'kiii.schema', '', ...
                    'general.time' , 500, ...
                    'buffer.size', params.buf_size, ...
                    ...
                    'kiii.layer1.kii.units', NN1, ...
                    'kiii.layer2.kii.units', NN2, ...
                    'kiii.layer3.kii.units', NN3, ...
                    ...
                    'kiii.layer1.kii.schema', '', ...
                    'kiii.layer2.kii.schema', '', ...
                    'kiii.layer3.kii.schema', '', ...
```

```
                    ...
                    'kiii.layer1.lateralw', wLat1, ...
                    'kiii.layer2.lateralw', wLat2, ...
                    'kiii.layer3.lateralw', wLat3, ...
                    ...
                    'kiii.layer1.w', w1, ...
                    'kiii.layer2.w', w2, ...
                    'kiii.layer3.w', w3, ...
                    ...
                    'kiii.kslink.1.2', k3_formKSlink('inIDX',[1:4*MM] ,
'outIDX' , [1:4*MM] ,'klink', k12 ), ...
                    'kiii.kslink.1.3', k3_formKSlink('inIDX',[1:4*MM] ,
'outIDX' , [1:4*MM] ,'klink', k13 ), ...
                    'kiii.kslink.2.1', k3_formKSlink('inIDX',[1:4*MM] ,
'outIDX' , [1:4*MM] ,'klink', k21 ), ...
                    'kiii.kslink.2.3', k3_formKSlink('inIDX',[1:4*MM] ,
'outIDX' , [1:4*MM] ,'klink', k23 ), ...
                    'kiii.kslink.3.1', k3_formKSlink('inIDX',[1:4*MM] ,
'outIDX' , [1:4*MM] ,'klink', k31 ), ...
                    'kiii.kslink.3.2', k3_formKSlink('inIDX',[1:4*MM] ,
'outIDX' , [1:4*MM] ,'klink', k32 ));

conf.MM = MM;

% prepare three KLayers - could pass the whole conf object
params.a = 0.22; % 0.6; % corresponds to frequency 40.021
params.b = 0.72; % 0.6;
L1 = k3_Klayer(conf.KIII_Layer1_Zig, params.buf_size, params.dT, ...
    conf.KIII_Layer1_W, conf.KIII_Layer1_KII_Schema,
conf.KIII_Layer1_KII_Units, ...
    conf.KIII_Layer1_lateralW, 1, conf.KIII_Layer1_filename, '', [],
[], params );


params.a = 0.22; % 0.3; % corresponds to frequency 47.0217
params.b = 0.72; % 0.3;
L2 = k3_Klayer(conf.KIII_Layer1_Zig, params.buf_size, params.dT, ...
    conf.KIII_Layer2_W, conf.KIII_Layer2_KII_Schema,
conf.KIII_Layer2_KII_Units, ...
    conf.KIII_Layer2_lateralW, 1, conf.KIII_Layer2_filename, '', [],
[], params );

params.a = 0.22; % 0.9; % corresponds to frequency 51.0465
params.b = 0.72; % 0.9;
L3 = k3_Klayer(conf.KIII_Layer1_Zig, params.buf_size, params.dT, ...
    conf.KIII_Layer3_W, conf.KIII_Layer3_KII_Schema,
conf.KIII_Layer3_KII_Units, ...
    conf.KIII_Layer3_lateralW, 1, conf.KIII_Layer3_filename, '', [],
[], params );

% prepare K nerwork out of L1 L2 L3 layers
K = k3(3, [L1 L2 L3], [], conf.KIII_KSlink, conf, params);
K.params = params;

% prepare input (small perturbation)
```

```
in = [  1.0; 0; 0; 0]; %hilbert transform phase slip = 1.5

% run network for 400 iterations
for t = 1:1
    K.t = K.t + 1;
    K = k3_SolveNextStep(K, in );
end

for t = 2:params.it_warmup
    K.t = K.t + 1;
    K = k3_SolveNextStep(K, zeros(size(in)));
end
```

k3_parameters.m

```
function [params] = k3_parameters(input_size)
% Returns an object that contains all major
% parameters of K-III network which include:
%
%   it_warmup       - number of time steps for warm up period
%   it_active       - number of time steps for active phase
%   it_relax        - number of time steps for inactive phase
%   dT              - time resolution for RK solver ( sec. / time steps
)
%   alfa            - learning rate for k3_hebb
 %   K-II 1 weights  - weights w_ee, w_ei, w_ie, w_ii for K-II network
in 1st layer
%   K-II 2 weights  - weights w_ee, w_ei, w_ie, w_ii for K-II network
in 2nd layer
%   K-II 3 weights  - weights w_ee, w_ei, w_ie, w_ii for K-II network
in 3rd layer
%   tau's           - time delays for inter-layer connections
%
% all parameters are packed into 'params' data structure
%
% see also k3, k3_new, k3_config
%
% by bileon, June 2004

% size of input
params.MM = input_size;

% time steps for warm up period, active and inactive phase of k3 cycle
params.it_warmup = 1000;
params.it_active = 1000;
params.it_relax  = 50;


% size of internal buffer to store activation history
% mm params.buf_size = max( params.it_active, 70);
params.buf_size = max( params.it_active, 70);

% parameters of k-ii networks for the 1st, 2nd and 3rd layer
params.kii1 = [0.05, 2.20, 2.20, 2.50];
params.kii2 = [0.06, 2.25, 2.23, 2.40];
```

```
params.kii3 = [0.30, 2.00, 2.10, 2.25];
params.kii4 = [0.05, 2.20, 2.20, 2.50];
params.kii5 = [0.06, 2.25, 2.23, 2.50];
params.kii6 = [0.30, 2.00, 2.10, 2.25];

% time resolution for Runge-Kutta solver
params.dT = 0.5;

% learning rate in hebbian learning algorithm
params.alpha = 5;

% delays for inter-layer links in the following order
% tau_12, tau_13, tau_21, tau_23, tau_31, tau_32
params.taus = [1, 1, 17, 1, 25, 25]; %orig

% and weights of corresponding connections
params.weights = [0.3  0.5  0.5 0.6 -0.5 0.5]; %orig weights
%params.weights = [0.2  0.2  10.0 0.6 10.0 10.0]; %seiz weights

% lateral itra-layer connections
% wLat_ee and wLat_ii
params.wLat1 = [0.15 -0.1];
params.wLat2 = [0.2  -0.2];
params.wLat3 = [0.15 -0.1];

% constants of differential equations
params.a = 0.22;
params.b = 0.72;
```

k3_q.m

```
function [ s ] = k3_q( x, Sig )
%------------------------------------------------------------------%
%--The University of Memphis, Department of Mathematical Sciences--%
% File:      Q
% Author:    Roman Ilin
% Date:      5/2003
% Purpose:   Sigmiod function for K Sets
%------------------------------------------------------------------%
%   x   vector input
%   s   vector output
%   Sig is the sigmoid coefficient
s = Sig .* (1 - exp( -1/Sig * (exp(x) -1)));
%s = max(s, -1);
```

k3_run.m

```
function [K, sigma] = k3_run(K, in)
% function k3_run returns K-III object K after
% it has been run for K.it_active time steps with
% each input sample vector from set 'in' (active period)
% followed by K.it_relax time steps of relaxation period
% (zero input) and set of std(activations) vectors
% that is standard deviation of 50 time steps of
% active phase for excitatory units of third layer
% for every input sample.
```

```
%
% Input set in is composed of input samples row by row, that
% is in = [ x_1; x_2; ... x_k].
%
% Sigma is composed of vectors that are computed as follows:
%
%   sigma(i,:) = std( activations );
%
% where 'activations' is set of activation for all top K-II
% excitatory units in the third layer.
%
% see also k3_new, k3, k3_train
%
% by bileon, June 2004


if size(in) ~= 0

    sigma = [];

    % iterate over input vectors
    for i = 1 : size(in,1)

        tic;

        % take input from set_train
        inp = repmat(zeros(4,1), K.MM,1);
        inp(1:4:end) = in(i,1:K.MM);

        % get current network time
        total_t = K.t;

        % run ACTIVE phase for 100 iterations
        for t = 1:K.it_active
            K.t = t + total_t;
            K = k3_SolveNextStep(K, inp ); %zeros(size(in))
        end

        % compute network output
        act = K.L(3).A;

        % get time brake position in buffer
        if mod(K.t, K.buf_size) == 0
            t_temp = K.buf_size;
        else
            t_temp = mod(K.t, K.buf_size);
        end

       % get activation history of third layer
        act = [ K.L(3).A( :, t_temp + 1 : end) K.L(3).A( :, 1:t_temp)];
        act = act(:, max(1, size(act,2) - K.it_active ) : end);

        sigma_temp = std(act(1:4:end, :)');
```

```
        sigma = [ sigma; sigma_temp];


        % run INACTIVE phase for 300 iterations
        for t = K.it_active + 1 : K.it_active + K.it_relax
            K.t = t + total_t;
            K = k3_SolveNextStep(K, zeros(size(inp)));
        end

        ['iteration ' int2str(i) ' of ' int2str(size(in,1)) ' time
elapsed ' num2str(toc)]

    end
else
    [' no input ... nothing done ']
end
```

k3_SolveNextStep.m

```
function K = k3_SolveNextStep( K, ExtInput )
%------------------------------------------------------------------%
%--The Univrsity of Memphis, Department of Mathematical Sciences--%
% File:      KIII_SolveNextStep
% Author:    Roman Ilin
% Date:      5/2003
% Purpose:   Advance KIII set one step
%------------------------------------------------------------------%
%KIII_SOLVENEXTSTEP KIII_SolveNextStep( K, ExtInput )
%ExtInput is the external input for the current time step ,
%coming to Layer 1 of the KIII set.  N by 1 vector
%PROBLEM WITH Q being applied before summation, NEED TO TEST AND FIX,
ALSO
%COMPARE TO HIZHON's CODE
for j=1:K.N
    in = zeros(4* K.L(j).N,1);
    for k=1:K.N
        %here use space links to obtain the inter-layer communication
        if j ~=  k
            % calc signal from layer k to layer j for all communication
            % lines
            % sender2 is the piece of history suitable for KDelay.
            % if statement is used because if k > j, history is already
            % flipped, so the indexing changes
            sig = K.L(k).Zig;
            hist = K.L(k).Hist;
            %index when 'history' starts, needs to be coordinated with
the 'flips' of history
            if k <= j
                s = mod(K.t, hist);
            else
                s = mod(K.t - 1, hist) + 1;
            end

            for p=1:size(K.SLINKS{k,j},2)
                sender2 = [K.L(k).A(:,s:-1:1) K.L(k).A(:,K.buf_size:-
1:s+1)];
```

```
                in = in + k3_KSLink_Calc(K.SLINKS{k,j}(p),
k3_q(sender2, sig), size(in ,1));
            end
        end
    end
    if j == 1
        in = in + ExtInput;
    end

%     % N O I S E   A D D E D   H E R E
%     % add noise, for each layer take K.params.noise(j) weight of
noise
%      noise = zeros(K.params.MM * 4, 1);
%     noise(1:4:end) = K.params.bias(j) + K.params.noise(j) *
rand(K.params.MM,1);
%     in = in + noise;

%mm
       noise = zeros(K.params.MM * 4, 1);
%       noise(1:4:end) = rand(K.params.MM,1);
    %in = [ 0.9553; 0; 0; 1.1144; 0; 0; 0; 0.9762; 0; 0; 0; 0];
%in = [ 1.0011; 0; 0; 0; 1.0011; 0; 0; 0; 1.0011; 0; 0; 0; 1.0011; 0;
0; 0; 1.0011; 0; 0; 0;1.0011; 0; 0; 0;1.0011; 0; 0; 0;1.0011; 0; 0;
0;1.0011; 0; 0; 0;1.0011; 0; 0; 0;];
   noise = [0.00010967; 0; 0; 0.00010967; 0; 0; 0; 0.00010967; 0; 0; 0;
0];
      %in = [0.122; 0.144; 0.455; 1; 0.898; 0.344; 0.232; 2; 0.232;
0.655; 0.111; 3];
   % in = [3.255; 3.255; 3.255; 3.255; 3.255; 3.255; 3.255; 3.255;
3.255; 3.255; 3.255; 3.255];
     % in = [ 1; 0; 0; 0; 2; 0; 0; 0];
       in = in + noise;
%mm

%input for all k layers into jth layer
    K.L(j).t = K.t;
    K.L(j) = k3_Klayer_SolveNextStep2(K.L(j), in);
end
```

k3_SolveNodeRungeKutta.m

```
function  [OutA, OutB] = SolveNodeRungeKutta( a, b, h, PrevA, PrevB,
RHS )
%-------------------------------------------------------------------%
%--The Univrsity of Memphis, Department of Mathematical Sciences--%
% File:      SolveNodeRungeKutta
% Author:    Roman Ilin
% Date:      5/2003
% Purpose:   Solve second order differential equation describing  the
%            the dynamics of a K0 set using Runge-Kutta method
%-------------------------------------------------------------------%
%a, b       constants
%h          step size
%PrevA, PrevB   previous values of A and B: activation and its
derivative
%RHS            Right Hand Side, Input
```

```
%Kozma's Way
k1 = F(a,b,   PrevB
)*h;
l1 = G(a,b,   PrevA                  ,PrevB                          ,
RHS           )*h;

k2 = F(a,b,   PrevB + l1/2
)*h;
l2 = G(a,b,   PrevA + k1/2      ,PrevB + l1/2               , RHS
)*h;

k3 = F(a,b,   PrevB + l2/2
)*h;
l3 = G(a,b,   PrevA + k2/2      ,PrevB + l2/2               , RHS
)*h;

k4 = F(a,b,   PrevB +  l3
)*h;
l4 = G(a,b,   PrevA +  k3       ,PrevB + l3                 , RHS
)*h;

OutA = PrevA     + (k1 + 2*k2 + 2*k3 + k4) / 6;
OutB = PrevB     + (l1 + 2*l2 + 2*l3 + l4) / 6;


%These are the two right-hand side functions, for solving
%two simultaneous equations of first order with Runge Kutta
function [f] = F(a,b, bb)
% returns right hand side of the equation
%               A' = B
%  bb,  aa, input are the values substituted for B and A and I
f = bb;

function [g] = G(a, b, aa, bb, input)
% returns right hand side of the equation
%               B' = -(a+b)*B - abA + abI
%  bb,  aa, input are the values substituted for B and A and I
g = -(a+b)*bb - a*b*aa + a*b*input;
```

k3_train.m

```
function [K] = k3_train(K, set_train)
% function k3_train returns K_III object K after
% it has been trained with samples from set_train
% when for K.it_active time steps input sample vector
% activation is gathered and k3_hebb function called;
% followed by K.it_relax time steps of relaxation period
% (zero input)
%
% Training takes place after active phase for each input sample
% in 'set_train'. Function k3_hebb() is called with lateral weights
% matrix in third layer and activation history of top K-II excitatory
% units for half active period. Weights matrix W = K.L(3).OMEGA and
% activations act = K.L(3).A.
```

```
%
%
% see also k3_new, k3, k3_train, ke_hebb
%
% by bileon, June 2004

for i = 1 : size(set_train,1)

    tic;

    % take input from set_train
    in = repmat(zeros(4,1), K.MM,1);
    in(1:4:end) = set_train(i,:);

    % run ACTIVE phase for K.it_active iterations
    for t = 1:K.it_active
        K.t = K.t + 1;
        K = k3_SolveNextStep(K, in ); %zeros(size(in))
    end

    % run HEBBIAN UPDATE for LEARNING

        % get weights of third layer
        W = K.L(3).OMEGA;

        % get time brake position in buffer
        if mod(K.t, K.buf_size) == 0
            t_temp = K.buf_size;
        else
            t_temp = mod(K.t, K.buf_size);
        end

        % get activation history of third layer
        act = [ K.L(3).A( :, t_temp + 1 : end) K.L(3).A( :, 1:t_temp)];
        act = act(:, end - round( K.it_active / 2 ) : end);

        % call hebb() function with selected weights and activations
        % and update W
        W(1:4:end, 1:4:end) = k3_hebb(act(1:4:end, :)', W(1:4:end,
1:4:end), K.params.alpha);

        % update the k3 model's weights matrix by redefining it
        K.L(3).OMEGA = W;
        K.O_H(i,:,:) = W(1:4:end, 1:4:end);

    % run INACTIVE phase for 300 iterations
    for t = K.it_active + 1 : K.it_active + K.it_relax
        K.t = K.t + 1;
        K = k3_SolveNextStep(K, zeros(size(in)));
    end
     ['iteration ' int2str(i) ' of ' int2str(size(set_train,1)) ' time
elapsed ' num2str(toc)]

end
```

K4_new.m

```
function [K] = k4_new(MC,MH,MM)

%MC = The size of all three layers of the cortex KIII, i.e. the number
of KIIs. Cortex KIII = 3
%MH = The size of all three layers of the hippo KIII, i.e. the number
of KIIs. Hippo KIII = 3
%MM = The size of the one layer of the amy KII,i.e. the number of KIIs.
Amy KII = 1

%creating a cortical kiii network
C = k3_cortex(MC);
%creating a hippocampal kiii network
H = k3_hippo(MH);
%creating a amygdala kii network
A = k2_new(MM);

%creating a brainstem - interface kii network
BS = k2_stim(MM);

params = k4_parameters(MC,MH,MM);

% temp variables
WA = params.WA;
WB = params.WB;
WC = params.WC;
NN1 = params.MC;
NN2 = params.MC;
NN3 = params.MC;
NN1h = params.MH;
NN2h = params.MH;
NN3h = params.MH;
NN1a = params.MM;


% prepare K nerwork out of L1 L2 L3 layers

% crete new k4 object and put C, H, and A into it
K.C = C;
K.H = H;
K.A = A;
K.BS = BS;


% put parameter object into the K4 too
K.params = params;

% global time keeper
K.t = K.C.t; % make sure H and C have the same time counter !!! (Amy
too)


% run network for several iterations
```

```
for t = 1:K.params.it_active
    t

  % Increase external external for seizure activity
  if (t > 2000 & t < 4000)
    %if (t > 2000 & t < 3000)
        K.params.WA = 5;
        K.params.WB = 5;
        K.params.WC = 5;
    elseif (t > 4000)
        K.params.WA = 0.3225;
        K.params.WB = 0.3225;
        K.params.WC = 0.3225;
  end

    % set global time of K4 one step ahead
    K.t = K.t + 1;

    % take Cortical K3, compute links from Hypp., and Amy,  ---------
Cort
    % and update Cort. K3 one time step

    % get current time position in the buffer
    if mod(K.t, K.H.params.buf_size) == 0
        t_temp = K.H.params.buf_size;
    else
        t_temp = mod(K.t, K.H.params.buf_size);
    end
    hypLink = zeros(4,1);
    hypLink = repmat(hypLink,params.MH,1);
    hypLink(1:4:end) = k3_q( K.H.L(3).A(1:4:end, t_temp),
K.H.L(1).Zig)'* K.params.WA ;
    amyLink = zeros(4,1);
    amyLink = repmat(amyLink,params.MM,1);
    amyLink(1:4:end) = k3_q( K.A.A(1:4:end, t_temp), K.H.L(1).Zig)'*
K.params.WB ;
    extIn = zeros(K.params.MC*4,1);

    % update the k3
    K.C.t = K.t; % synch k3 Cort with K4 global timer
    K.C = k4_k3_solveNextStep(K.C, extIn, hypLink, amyLink);


    % take Hippocampal K3, compute links from Cortex., and Amy,  ------
--- Hyp
    % and update Hippo. K3 one time step

    % get current time position in the buffer
    if mod(K.t, K.C.params.buf_size) == 0
        t_temp = K.C.params.buf_size;
    else
        t_temp = mod(K.t, K.C.params.buf_size);
    end
    CorLink = zeros(4,1);
    CorLink = repmat(CorLink,params.MC,1);
```

```
    CorLink(1:4:end) = k3_q( K.C.L(3).A(1:4:end, t_temp),
K.C.L(1).Zig)'* K.params.WA ;
    amyLink = zeros(4,1);
    amyLink = repmat(amyLink,params.MM,1);
    amyLink(1:4:end) = k3_q( K.A.A(1:4:end, t_temp), K.C.L(1).Zig)'*
K.params.WC ;
    extIn = zeros(K.params.MH*4,1);

    % update the k3
    K.H.t = K.t; % synch k3 Hippo with K4 global timer
    K.H = k4_k3_solveNextStep(K.H, extIn, CorLink, amyLink);

    % take Amygdala k2, compute links from Cort K3 and Hyp k3   -------
-- Amy

    %get current time position in the buffer
    if mod(K.t, K.A.params.buf_size) == 0
        t_temp = K.A.params.buf_size;
    else
        t_temp = mod(K.t, K.A.params.buf_size);
    end
    amyLinkC = zeros(4,1);
    amyLinkC = repmat(amyLinkC,params.MC,1);
    amyLinkC(1:4:end) = k3_q(K.C.L(3).A(1:4:end, t_temp),
K.A.params.sig)'* K.params.WB;
    amyLinkH = zeros(4,1);
    amyLinkH = repmat(amyLinkH,params.MH,1);
    amyLinkH(1:4:end) = k3_q(K.H.L(3).A(1:4:end, t_temp),
K.A.params.sig)'* K.params.WC;

    %Update Brain Stimulator object
    stimLink = zeros(4,1);
    stimLink = repmat(stimLink,params.MM,1);
    stimLink(1:4:end) = k3_q(K.BS.A(1:4:end, t_temp), K.A.params.sig)'*
K.params.WC;

    %update k2
    K.A.t = K.t; %synch k2 amygdale with K4 global timer
    % restore section - works
    if (t < 2000)
        amyIn = amyLinkC + amyLinkH;
    elseif (t > 2000 & t < 4000)
    % Apply stimulator to reduce seizure event
        amyIn = amyLinkC + amyLinkH + stimLink;
    end
    %amyIn = amyLinkC + amyLinkH;
    K.A = k2_solveNextStep(K.A, amyIn);

end
```

k4_k2_solveNextStep.m

```
function K = k4_k2_solveNextStep(K, amyLinkC, amyLinkH)

%-------------------------------------------------------------------%
%
```

```
%purpose given the input from cortical and hippocampal
%-----------------------------------------------------------------%

k = mod(K.t - 1, K.params.buf_size) + 1;

% index in a round robin buffer by modulo K.Hist,
% turn it around if modulo division is zero
if mod( K.t - 1, K.params.buf_size) == 0
    kk = K.params.buf_size;
else
    kk = mod( K.t - 1, K.params.buf_size);
end

in = K.OMEGA*k3_q( K.A(:,kk),K.params.sig );
%in = K.OMEGA*k3_q( K.A(:,mod(K.t - 1, K.Hist) + K.Hist),K.Zig ) /
max(1, K.N - 1) ;
%NOT SURE WHICH WAY IS CORRECT
%in = k3_q(K.OMEGA* K.A(:,mod(K.t - 1, K.Hist) + K.Hist),K.Zig );
K.I = amyLinkC + amyLinkH;

% sovle ODE and store current results
[K.A(:, k) , K.B(:, k)] = k3_SolveNodeRungeKutta(K.params.a,
K.params.b, K.params.dT, K.A(:, kk ), K.B(:, kk ), K.I + in  );
```

k4_k3_SolveNextStep.m

```
function K = k4_k3_SolveNextStep(K , ExtIn, otherIn, AmyIn)
% solves RK for one step for the K3 network,K
% given external input, ExtIn, link from the other
% k3 in k4 formation, otherIn, and link from amygdala
% k2, AmyIn
%
% extIn - goes to the top layer
% otherIn and AmyIn - go to the third layer as external input
%
% by bileon, bharat, April 2005

for j=1:K.N

    in = zeros(4* K.L(j).N,1);

    for k=1:K.N
        %here use space links to obtain the inter-layer communication
        if j ~=  k
            % calc signal from layer k to layer j for all communication
            % lines
            % sender2 is the piece of history suitable for KDelay.
            % if statement is used because if k > j, history is already
            % flipped, so the indexing changes
            sig = K.L(k).Zig;
            hist = K.L(k).Hist;
            %index when 'history' starts, needs to be coordinated with
the 'flips' of history
            if k <= j
                s = mod(K.t, hist);
```

```
            else
                s = mod(K.t - 1, hist) + 1;
            end

            for p=1:size(K.SLINKS{k,j},2)
                sender2 = [K.L(k).A(:,s:-1:1) K.L(k).A(:,K.buf_size:-
1:s+1)];
                in = in + k3_KSLink_Calc(K.SLINKS{k,j}(p),
k3_q(sender2, sig), size(in ,1));
            end
        end
    end
    if j == 1
        in = in + ExtIn;
    end
    if j == 3
        in = in + otherIn + AmyIn;
    end

    %input for all k layers into jth layer
    K.L(j).t = K.t;
    K.L(j) = k3_Klayer_SolveNextStep2(K.L(j), in);
end
```

k4_Klayer_SolveNextStep2.m

```
function K = k4_Klayer_SolveNextStep2( K, extIn )

k = mod(K.t - 1, K.Hist) + 1;

% index in a round robin buffer by modulo K.Hist,
% turn it around if modulo division is zero
if mod( K.t - 1, K.Hist) == 0
    kk = K.Hist;
else
    kk = mod( K.t - 1, K.Hist);
end

in = K.OMEGA*k3_q( K.A(:,kk),K.Zig );
%in = K.OMEGA*k3_q( K.A(:,mod(K.t - 1, K.Hist) + K.Hist),K.Zig ) /
max(1, K.N - 1) ;
%NOT SURE WHICH WAY IS CORRECT
%in = k3_q(K.OMEGA* K.A(:,mod(K.t - 1, K.Hist) + K.Hist),K.Zig );

    % N O I S E   A D D E D   H E R E
    % add noise, for each layer take K.params.noise(j) weight of noise
    noise = zeros(K.params.MM * 4, 1);
%    noise(1:4:end) = K.params.bias + K.params.noise *
randn(K.params.MM,1);
    %in = in + noise;

K.I = extIn;

% sovle ODE and store current results
```

```
[K.A(:, k) , K.B(:, k)] = k3_SolveNodeRungeKutta(K.a, K.b, K.h, K.A(:,
kk ), K.B(:, kk ), K.I + in  );
K.A(:, k) = K.A(:, k) + noise;
```

k4_run.m

```
function [K, sigma] = k4_run(K, in_C ,in_H)
if size(in_C) ~= 0

    sigma = [];

    % iterate over input vectors
    for i = 1 : size(in_C,1)

        tic;

        % take input from set_train and set_trainh
        in = repmat(zeros(4,1), K.C.MM,1);
        in(1:4:end) = in_C(i,1:K.C.MM);
        oin = repmat(zeros(4,1), K.H.MM,1);
        oin(1:4:end) = in_H(i,1:K.H.MM);
        % get current network time
        K.t = K.C.t;

        % run ACTIVE phase for 300 iterations
        % this function gives updated k4 model
        for j = 1:K.params.it_active
        % increment the global time by one step
        K.t = K.t + 1;
        K = k4_update(K , in(:,:) , oin(:,:));
        end

        %THE READ OUT

        % get time brake position in buffer
        if mod(K.t, K.C.params.buf_size) == 0
            t_temp = K.C.params.buf_size;
        else
            t_temp = mod(K.t, K.C.params.buf_size);
        end

       % Collect the activation from amygdala
        act = [K.A.A( :,t_temp + 1:end) K.A.A( :, 1:t_temp)];

        %Apply standard deviation
        sigma_temp = std(act(1:4:end, :)');

        sigma = [ sigma; sigma_temp];


        % run INACTIVE phase for 100 iterations
       for j = 1:K.params.it_relax
        % increment the global time by one step
        K.t = K.t + 1;
        K = k4_update(K , zeros(size(in)) , zeros(size(oin)));
```

```
        end


         ['iteration ' int2str(i) ' of ' int2str(size(in_C,1)) ' time
elapsed ' num2str(toc)]

    end
else
    [' no input ... nothing done ']
end
```

k4_train.m

```
function [K] = k4_train(K , in_C , in_H)
% This function returns and k4 network after training with the input
from
% both the k3's.
% when for K.it_active time steps input sample vector
% activation is gathered and k3_hebb function called;
% followed by K.it_relax time steps of relaxation period
% (zero input)
% Training takes place after active phase for each input sample
% in both the k3's training sets. Function k3_hebb() is called with
lateral weights
% matrix in third layer and activation history of top K-II excitatory
% units for half active period. Weights matrix W = K.L(3).OMEGA and
% activations act = K.L(3).A.


for i = 1 : size(in_C,1)

         tic;

        % take input from set_train
        in = repmat(zeros(4,1), K.C.MM,1);
        in(1:4:end) = in_C(i,:);
        oin = repmat(zeros(4,1), K.H.MM,1);
        oin(1:4:end) = in_H(i,:);
        % get current network time
        K.t = K.C.t;

        % run ACTIVE phase for 500 iterations
        % this function gives updated k4 model
         for j = 1:K.params.it_active
        % increment the global time by one step
        K.t = K.t + 1;
        K = k4_update(K , in , oin);
        end

  %Run hebbain update for learining

       % get weights of third layer of both hippocampal and cortical
         WC = K.C.L(3).OMEGA;
```

```
         WH = K.H.L(3).OMEGA;

        % get weights the amygdale


       %W = K.A.OMEGA;

  % get time brake position in buffer
        if mod(K.t, K.C.params.buf_size) == 0
            t_temp = K.C.params.buf_size;
        else
            t_temp = mod(K.t, K.C.params.buf_size);
        end

         % get activation history of third layer
         actc = [ K.C.L(3).A( :, t_temp + 1 : end) K.C.L(3).A( :,
1:t_temp)];

         acth = [ K.H.L(3).A( :, t_temp + 1 : end) K.H.L(3).A( :,
1:t_temp)];

         % get activation from the amygdale

         %act = [ K.A.A( :, t_temp + 1 : end) K.A.A( :, 1:t_temp)];

         % call hebb() function with selected weights and activations
        % and update W

        WC(1:4:end, 1:4:end) = k3_hebb(actc(1:4:end, :)', WC(1:4:end,
1:4:end), K.params.alpha);
        WH(1:4:end, 1:4:end) = k3_hebb(acth(1:4:end, :)', WH(1:4:end,
1:4:end), K.params.alpha);

         % update the k3 model's weights matrix by redefining it
      K.C.L(3).OMEGA = WC;
      K.H.L(3).OMEGA = WH;

 % run INACTIVE phase for 100 iterations
   for j = 1: K.params.it_relax
        % increment the global time by one step
        K.t = K.t + 1;
        K = k4_update(K , zeros(size(in)) , zeros(size(oin)));
    end
end
```

k4_update.m

```
function K = k4_update(K, in_C , in_H)
%This function updates the K4 network by updating the cortical
hippocampal
%and amydale links one time step

% run network for several iterations
```

```
for t = 1:size(in_C , 1)

    % set global time of K4 one step ahead
    K.t = K.t + 1;

    % take Cortical K3, compute links from Hypp., and Amy,  ---------
Cort
    % and update Cort. K3 one time step

    % get current time position in the buffer
    if mod(K.t, K.H.params.buf_size) == 0
        t_temp = K.H.params.buf_size;
    else
        t_temp = mod(K.t, K.H.params.buf_size);
    end
    hypLink = zeros(4,1);
    hypLink = repmat(hypLink,K.params.MH,1);
    hypLink(1:4:end) = k3_q( K.H.L(3).A(1:4:end, t_temp),
K.H.L(1).Zig)'* K.params.WA ;
    amyLink = zeros(4,1);
    amyLink = repmat(amyLink,K.params.MM,1);
    amyLink(1:4:end) = k3_q( K.A.A(1:4:end, t_temp), K.H.L(1).Zig)'*
K.params.WB ;
    extIn = zeros(K.params.MC*4,1);
    extIn = extIn + in_C;

    % update the k3
    K.C.t = K.t; % synch k3 Cort with K4 global timer
    K.C = k4_k3_solveNextStep(K.C, extIn , hypLink, amyLink);


    % take Hippocampal K3, compute links from Cortex., and Amy,  ------
--- Hyp
    % and update Hippo. K3 one time step

    % get current time position in the buffer
    if mod(K.t, K.C.params.buf_size) == 0
        t_temp = K.C.params.buf_size;
    else
        t_temp = mod(K.t, K.C.params.buf_size);
    end
    CorLink = zeros(4,1);
    CorLink = repmat(CorLink,K.params.MC,1);
    CorLink(1:4:end) = k3_q( K.C.L(3).A(1:4:end, t_temp),
K.C.L(1).Zig)'* K.params.WA ;
    amyLink = zeros(4,1);
    amyLink = repmat(amyLink,K.params.MM,1);
    amyLink(1:4:end) = k3_q( K.A.A(1:4:end, t_temp), K.C.L(1).Zig)'*
K.params.WC ;
    extIn = zeros(K.params.MH*4,1);
    extIn = extIn + in_H;

    % update the k3
    K.H.t = K.t; % synch k3 Hippo with K4 global timer
```

```
    K.H = k4_k3_solveNextStep(K.H, extIn, CorLink, amyLink);

    % take Amygdala k2, compute links from Cort K3 and Hyp k3   -------
-- Amy

    %get current time position in the buffer
    if mod(K.t, K.A.params.buf_size) == 0
        t_temp = K.A.params.buf_size;
    else
        t_temp = mod(K.t, K.A.params.buf_size);
    end
    amyLinkC = zeros(4,1);
    amyLinkC = repmat(amyLinkC,K.params.MC,1);
    amyLinkC(1:4:end) = k3_q(K.C.L(3).A(1:4:end, t_temp),
K.A.params.sig)'* K.params.WB;
    amyLinkH = zeros(4,1);
    amyLinkH = repmat(amyLinkH,K.params.MH,1);
    amyLinkH(1:4:end) = k3_q(K.H.L(3).A(1:4:end, t_temp),
K.A.params.sig)'* K.params.WC;

    %update k2
    K.A.t = K.t; %synch k2 amygdale with K4 global timer
    amyIn = amyLinkC + amyLinkH;

    K.A = k2_solveNextStep(K.A, amyIn);

end
```

k4_parameters.m

```
function [params] = k4_parameters(cortex_size,hipp_size,amygdala_size)

% size of input
params.MM = amygdala_size;
params.MC = cortex_size;
params.MH = hipp_size;

% time steps for warm up period, active and inactive phase of k3 cycle
params.it_warmup = 1000;
params.it_active = 1000;
params.it_relax  = 50;

% time resolution for Runge-Kutta solver
params.dT = 0.5;

%learing rate for hebb
params.alpha = 5;

lambda=1.0;

nn=0.3225;

params.WA = lambda*nn;
params.WB = lambda*nn;
params.WC =lambda*nn;
```

k2_stim.m

```
function [K] = k2_stim(MM)
%
% function k2_new returns new K_II object with
% given size of input layer MM, that is number
% of full k-ii sets in the layer
%
% At this moment all parameters are set to some
% default values used in the lab for k-sets.
%
% The default parameters for K_II sets in are as follows:
%
%   k3_formKIImatrix(1.8, 1, 2, 0.8);   % w_ee, w_ei, w_ie, w_ii
%
% K-set object has the following fields:
%
%             L: [1x1 struct]   - layer object (see k3_Klayer)
%             t: 400            - current total number of time steps
run
%        params: [struct]       - parameters of the network
%      buf_size: 80             - size of activation buffer in time
steps
%     it_active: 100            - number of time steps for active phase
%      it_relax: 300            - number of time steps for relaxation
%            MM: 10             - size of input
%
% see also k2_run(), k2_parameters(), k3_formKIImatrix, k3_Klayer,
k3_formLateralMatrix1
%
% by bileon, August 2004

% call k2_parameters for all weiths of full k_ii
params = k2_parameters_stim(MM);

%Lateral Connection Matrix
wLat = k3_formLateralMatrix1(params.wLat(1), params.wLat(2),
params.MM);

%internal weights
w = k3_formKIImatrix(params.kii(1), params.kii(2), params.kii(3),
params.kii(4));

% prepare a KLayer
% give it params, k-ii weights matrix and lateral weights matrix
K = k2_Klayer(params, w, wLat);

% store parameter object inside network object
K.params = params;

% prepare input (small perturbation)
var1 = 0.1;

in = [  var1; 0.0; 0.0; 0.0];
in = repmat(in, params.MM, 1);
```

```
% run network for one active (with input) iteration
for t = 1:1
    K.t = K.t + 1;
    K = k2_SolveNextStep(K, in );
end

% run network for res of parmas.it_warmup iterations with zero input
for t = 2:params.it_warmup
    K.t = K.t + 1;
    K = k2_SolveNextStep(K, zeros(size(in)));
end
```

autocorr_eeg.m

```
N=500; % Number of samples
f1=1; % Frequency of the sinewave
FS=256; % Sampling Frequency
n=0:N-1; % Sample index numbers

%normal EEG
x=y1(100001:100500,1);  % Generate the signal, x(n)

t=[1:N]*(1/FS); % Prepare a time axis
figure;
subplot(2,1,1); % Prepare the figure
plot(t,x,'k');
title('EEG');

xlabel('Time, [s]');
ylabel('Amplitude');
grid;
subplot(2,1,2); % Prepare the figure
Rxx=xcorr(x); % Estimate its autocorrelation
plot(Rxx/200,'k'); % Plot the autocorrelation
grid;
title('Autocorrelation function');
xlabel('lags');
ylabel('Autocorrelation');
```

psd_analysis.m

```
[filename, pathname] = uigetfile( ...
       {'*.txt', 'All Rabbit EEG Files (*.txt)'; ...
           '*.dat', 'EEG Files (*.dat)'; ...
        '*.*',                   'All Files (*.*)'}, ...
        'Pick an EEG file');
if isequal(filename,0)|isequal(pathname,0)
       disp('EEG File not found')
```

```
    else
       disp(['File ', pathname, filename, ' EEG file found'])

end
aa=textread([pathname filename],'','headerlines',2);

a=y1;
numCh = size(a,2)-1;
numData =input('Window size in analysis   :');
Initial_time=input('What is the first point in analysis?   :');

for ch=1:1:numCh
     x(:,ch)=a(Initial_time: numData+Initial_time-1,ch);
end

for chh=1:numCh
     [aa bb]=psd(x(:,chh)-mean(x(:,chh)),512,500);
end

XX=log10(bb(20:55));
YY=log10(aa(20:55));

figure;
plot(log10(bb),log10(aa))
hold

psd_val=polyfit(XX,YY,1);

y_p2=polyval(psd_val,XX);
plot(XX,y_p2,'r');
grid

xlabel('log frequency (Hz)'), ylabel('log power');
title('temporal power spectral density');
```