TOWARDS A MODEL OF BASIC INTENTIONAL SYSTEMS: NONLINEAR DYNAMICS FOR PERCEPTION MEMORY AND ACTION IN AUTONOMOUS ADAPTIVE AGENTS

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Memphis

Derek Shawn Harter

May, 2004

Acknowledgments

I would like to take this moment to thank a number of people, without whose help and support this research would not have been possible. First I must acknowledge my own personal Cognitive Science trifecta, Robert Kozma, Stan Franklin and Art Graesser. They represent a wonderful breadth and depth of knowledge in almost all aspects of Cognitive Science, AI and Psychology, and this work would been much less than it is without their insights and high standards. I especially want to thank Robert Kozma, my committee chair and advisor. His patience and encouragement in my rougher moments were crucial to my success.

Second I would like to acknowledge the students and members of both the Computational Neurodynamics Lab, and the Institute for Intelligent Systems. The CND and IIS have been places of exciting and dynamic research during my time here, and I have learned more that I can imagine from my interactions, presentations, symposium, etc. with the members of these research communities at the University of Memphis.

Third and more personally I must thank Shulan for her faith in me and her patience in dealing with me during the more trying times, and her friendship and support. Also my parents for supporting whole-heartedly my desires both when growing up and in returning to graduate school.

Last but not least I must needs acknowledge the wonderful work of Dr. Walter J. Freeman. His original insights into the dynamics of neural populations is of course the main inspiration and jumping off point for much of this research.

The work presented in this thesis was supported in part by NASA Intelligent Systems Research Grant (NCC-2-1244).

Abstract

Biological brains are extremely good at generating flexible behavior and managing limited resources of time and materials in demanding environments. Our artificial systems have yet to match the capabilities of even very simple biological brains. We are beginning to understand that biological brains belong to the class of nonlinear dynamical systems, and that nonlinear dynamics play a fundamental role in the self-organization of the cognitive mechanisms that produce such flexible and adaptive behavior. Bottom-up approaches to understanding intelligence, such as embodied and situated cognition, have provided some insights into how neural structures organize into low-level cognitive mechanisms. Biological organisms of a certain complexity, such as mammals, are intentional systems. All of their actions are goal-oriented and are focused on achieving goals and sub-goals that satisfy the needs and desires of the organism. We are beginning to understand some of these mechanisms of self-organization in neural populations that lead to rational behavior, especially in the perceptual and memory systems of brains. Our current task in understanding intelligent behavior from a complex systems perspective is to bridge the gap in our knowledge of how such nonlinear mechanisms might produce higher-level cognitive mechanisms.

This dissertation contributes to our understanding of how nonlinear dynamical processes are involved in the mechanisms of cognition, and especially how neural populations self-organize into perceptual and memory producing systems. In this work I develop an engineering simplification of Freeman's K-set neural population model. This is a discrete time, deterministic model capable of producing the spectral and temporal characteristics of the original K-set biological population model, and useful for real-time simulations with autonomous agents (KA-sets). I use the KA model to demonstrate the formation of amplitude modulation (AM) patterns in an autonomous agent. The aperiodic dynamics of the simulation replicate those known to exist in mammalian perceptual systems. I demonstrate how such AM patterns are a model of the formation of cognitive maps and place cells in the hippocampus. I also use the KA model to demonstrate the formation of action selection mechanisms for autonomous agents in an appetitive/aversive task based on aperiodic dynamics.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

The study of complex systems and nonlinear dynamics has provided us with new conceptual and theoretical tools that enable us to categorize and understand complex patterns of behavior in natural systems. Such patterns seem to be universal. They show up in physical processes such as weather formation and fluid dynamics; in biological systems such as ecosystems, developmental biology and evolutionary processes; and in human social and economic systems, such as city formation, traffic patterns and economic markets.

We are beginning to understand that the complex processes observed in the brain are also of this category of self-organizing nonlinear dynamical systems. It has been shown by Freeman and associates (Freeman, 1975; Skarda & Freeman, 1987; Freeman, 1991) that aperiodic dynamics are essential to the development of perceptual mechanisms in biological brains. Nonlinear dynamics are being used to model developmental process (Thelen & Smith, 1994), neurophysiological dynamics (Kelso, 1995) and other phenomena of neural population dynamics (Schiff et al., 1994; Nunez, 2000; Tsuda, 2001; Freeman, 2003; Kozma, Freeman, & Erdi, 2003). We are gaining an understanding of some of the processes by which nonequilibrium, nonlinear thermodynamic systems transform flows of energy and material into complex, self-reproducing, self-organizing structures. We have observed some of the mechanisms by which this self-organization occurs among different types of self-organizing systems: mesh and hierarchy dynamics, autocatalytic loops, competitive processes such as predator/prey and parasite/host arms races, cooperative processes such as species symbiosis, etc. There is much yet to be done on identifying and studying these processes in neural populations and understanding how they produce cognitive mechanisms.

Intelligent behavior is characterized by the flexible and creative pursuit of endogenously organized goals. The archetypal form of intentional behavior is an act of observation through time and space, by which information is sought for the guidance of future actions. Intentional behavior is a goal-directed activity whereby perceptions are sought and actions performed in service of the needs of the organism. Research in cognitive science has been conducted on some of the pieces of intentional behavior, such as mechanisms for focus of attention, goal-formation and expectation. However, many of these pieces have yet to appear together in complete models of cognitive agents, and further the mechanisms by which such processes self-organize are not well understood.

The idea of intentionality provides a framework that helps us to identify the components of cognition, from a top-down perspective, that are necessary for the production of intelligent behavior. Intentional organisms have desires and drives that need to be fulfilled, and goals and sub-goals are pursued by the organism in the service of satisfying its intrinsic drives. Intentional systems are built upon more reactive behavioral systems. Intentionality is greatly aided by the development of long-term memory that can learn to recognize increasingly more complex and abstract spatial-temporal patterns and act accordingly. Intentional organisms also demonstrate the importance of anticipation, expectations, situational awareness and orientation. Intentional organisms are constantly generating expectations of stimuli from their context and current goals. Violations of expectations lead to learning situations for the organism. Focus of attention and orientation are key characteristics of intentional behavior.

In a sense traditional AI aims to explain intelligence at this level of goals, knowledge and actions. Traditional, symbolic approaches also proceed from an assumption of rationality on the part of intelligent agents. This boils down to the simple idea that an intelligent agent is a rational agent, and will always perform actions that it perceives to be in its best interest (Newell, 1990). Symbolic approaches hypothesize that the ability to manipulate symbols is necessary and sufficient to producing general intelligent behavior capable of this type of rational decision making. This view has been criticized on several grounds and a number of fundamental problems have been found including the frame problem, the symbol grounding problem and the problem of situatedness.

Against this traditional background, new approaches have emerged to the explanation of intelligent behavior, including embodied and situated cognition, dynamical cognition and complex systems approaches (Franklin, 1995; Port & van Gelder, 1995; Hendriks-Jansen, 1996; Clark, 1997; Franklin & Graesser, 1997; Pfeifer & Scheier, 1998). Proponents of this view have argued that explanations of cognition can be found without relying on internal symbolic representations.

One of the problems we currently face in understanding intelligence is that both approaches seem right in some fundamental ways, and both have areas of weakness and failure. Symbolic approaches seem better suited to explaining higher level cognition, such as planning and logical reasoning. But symbolic approaches have proven brittle and difficult to ground their systems in the real-world. Situated and embodied cognition, on the other hands, seem well suited to understanding some of the things we think of as simpler aspects of intelligence, such as moving around in physical environments and learning motor skills. It has proven more difficult, however, to scale up these new approaches to explain more higher-level cognitive processes such as memory and planning.

It could be argued that the traditional approaches account for some but not all of the high-level properties of intentional behavior. A fundamental goal of studying the neuronal systems in biological brains that are involved in intentional behavior is to understand how neuronal processes support the higher-level cognitive processes that we identify with intentionality such as memory, orientation and attention. One of the bridging concepts that might enable us to explain these higher-level cognitive processes in terms of neural functions may be found in the study of complex systems and the application of aperiodic dynamics to developing such cognitive mechanisms.

The purpose of this research is to better understand the uses that complex, aperiodic dynamics might play in cognitive mechanisms so we can better bridge the gap in our understanding of how neuronal processes support cognitive mechanisms. In particular, the focus of this dissertation is on the development of a simplified neural population model, based on Freeman's K-sets, that allows for large scale neuronal simulations with autonomous agents. Such a simplification has many advantages. It is more efficient which allows for larger models of nonlinear cognitive processes to be explored. The discrete nature of the model is similar to standard artificial neural network (ANN) equations, making the results more comparable to the body of research on standard and recurrent ANNs. The discrete model is fast enough to be used in real-time robotic applications with relatively large-scale neural simulations. And it is more easily understood and explained which should make it easier to use as a tool when studying nonlinear cognition.

The discrete neural population model developed in this dissertation has been used in experiments with autonomous agents showing the uses of aperiodic dynamics in forming memory and learning conditioned responses. I use the discrete neural model to demonstrate the formation of amplitude modulation (AM) patterns in an autonomous agent in response to environmental stimuli. The AM patterns form a simple type of place cell in a simulated hippocampus, that remembers and identifies locations in the agents environment. I also demonstrate the formation of learned conditioned responses by the model in order to discriminate between desirable and undesirable stimuli from a distance. This work is moving towards building intentional systems by demonstrating the uses of aperiodic dynamics in cognitive mechanisms in autonomous agents. In the next section we will look more closely at some other systems in AI and cognitive science that are attempting to bridge the gap between neural mechanisms and higher level cognition. We will then describe the major results of this dissertation, including the discrete neural population modeled for use in autonomous agents, and the experiments carried out demonstrating its effectiveness in developing cognitive mechanisms using aperiodic dynamics.

Chapter 2

Review

2.1 Nonconvergent Dynamics and Cognitive Systems

What are the mechanisms by which biological organisms, including human beings, produce general intelligent actions in order to survive, reproduce and thrive in their environment? This, in some form, is one of the basic questions that lies at the heart of cognitive science. Various ideas have been put forward as possible answers to this question. Metaphors of cognition have been inspired from the advanced technologies of the times, including hydraulic, phone switchboard and computer metaphors (Von Neumann, 1958). Inspiration has been sought from the realms of formal logic as the means of general intelligent action. Others have looked to abstracted models of how neural tissue functions as possibly holding the key insights to the production of intelligent behavior.

Historically the study of psychology has been focused on discovering the correlations and laws of human and animal intelligent behavior, as a way to begin to understand the underlying mechanisms. Another way of saying this is that Psychology has been interested in studying the *what* questions of intelligent behavior. *What* are the capacities of long- and short-term memory? *What* are the learning gains observed in different tutoring and lecturing strategies? Cognitive science, and especially the areas of AI and cognitive modeling, have on the other hand been focused on studying the structures and mechanisms used by people and animals to support general intelligent behavior, and sometimes specifically what mechanisms could account for the properties observed by psychological studies. This approach to studying intelligent behavior can be thought of as asking the *how* questions. *How* can a collection of simple processors store, recognize and complete patterns? *How* does brain architecture result in emotion?

Alan Turing, with his famous and eponymous *Turing Test* suggested that intelligence is a matter of behavior or behavioral capacity (Turing, 1950; Haugeland, 1997). Whether a system has a mind, and how intelligent that mind is, is determined by what it can and cannot do. In the *Turing Test*, therefore, language ability and capacity is proposed as the behavior by which we can determine if a system has a general intelligent ability at or above the level of human beings. Turing did not claim that a system would not be intelligent if it could not pass his test; only that it **would** be intelligent if it could. Therefore the *Turing Test* provides a sufficient condition for detecting the presence of general human level intelligence (though not a necessary one).

Intelligent behavior really lies on a continuum, from simple tropic behaviors of single celled organisms to tool and language use of human beings. We could imagine defining levels or categories of intelligence, each with an appropriate Turing-like behavioral test that could be used to determine inclusion of a system in a category. Such tests would necessarily not be of language use, but could test things like memory capacity (short-, and long-term), opportunistic vs. goal orientation and problem solving in unique situations. Such tests would give us the ability to state the necessary capacities and behaviors that need to be present to include a system in a certain level of intelligence.

Being able to define and detect levels of intelligence through outward behavioral characteristics is an important piece in the study of intelligent behavior. However, since the rejection of behaviorism and the rise of AI and cognitive science, we have not only been interested in the necessary behavioral characteristics that let us detect intelligence, but also in the types of internal mechanisms and processes that might be necessary and sufficient to produce such observed behaviors. The opening up of the study of intelligence to include internal mechanisms has allowed us to attack the problem both from the outside in, and from the inside out.

Once we begin studying the possible mechanisms of intelligence, it is natural to ask if there is any simplest set or category of mechanism that is both sufficient and necessary for the production of general intelligent behavior. Given that there are different levels of intelligence, are different mechanisms needed to achieve these different levels, or can the same mechanisms be used, only expanded to do more. Previous attempts to define such conditions have focused on things like the ability of formal logic like symbol manipulations to perform tasks we usually think of as intelligent, like playing chess or planning a sequence of tasks to perform a goal (Newell, 1980, 1990). Another major movement has focused on the power of simple non-linear processing units to remember, recognize and complete patterns (Werbos, 1974; Rumelhart, McClelland, & The PDP Research Group, 1986).

Another way of discovering the constraints of intelligence, besides psychological experimentation, is to directly observe the workings of the only known systems that are capable of general intelligent behavior, biological brains. These types of direct measuring of neurological functioning through such methods as EEG recordings and brain imaging techniques have provided us with valuable further constraints on the possible sufficient and necessary dynamics involved in cognition. Such understanding led directly to early connectionist modeling results, and is leading us even further in new directions.

Biological brains are awash in complex, nonconvergent dynamics. Such complex dynamics have usually been abstracted away in connectionist models, with the assumption that they are not necessary to the production of intelligent behavior. However, new ideas in nonlinear dynamical systems theories, both inside and outside of cognitive science, have begun to understand the possible important roles that aperiodic dynamics, such as chaos, may play in self-organizing systems.

Some researchers in dynamical cognition and neurodynamics have speculated on the possibilities that more complex, chaotic like dynamics may play in the role of adaptive behavior (Skarda & Freeman, 1987; Freeman, 1999b; Kozma & Freeman, 1999; Freeman, Kozma, & Werbos, 2000; Kozma & Freeman, 2000, 2001a). Chaotic dynamics have been observed in the formation of perceptual states of the olfactory sense in rabbits (Skarda & Freeman, 1987). Skarda and Freeman have speculated that chaos may play a fundamental role in the formation of perceptual meanings. Chaos provides the right blend of stability and flexibility needed by the system. Essentially, Skarda and Freeman believe that the normal background activity of neural systems is a chaotic state. In the perceptual systems, input from the sensors perturbs the neuronal ensembles from the chaotic background, and the result is that the system transitions into a new attractor that represents the meaning of the sensory input, given the context of the state of the organism and its environment. But the normal chaotic background state is not like noise. Noise cannot be easily stopped and started, whereas chaos can essentially switch immediately from one attractor to another. This type of dynamics may be a key property in the flexible production of behavior in biological organisms.

The possible importance and uses of aperiodic dynamics to intelligence have not yet been fully explored in cognitive science. In the next sections we demonstrate why aperiodic dynamics may be important to intelligent behavior, and define the necessary and sufficient conditions for general intelligent behavior if it is true that such dynamics play a crucial role in cognition. We will then present one such model capable of producing aperiodic dynamics for use in perceptual, memory and behavior producing systems.

2.2 Theories and Conditions of Cognition

2.2.1 Symbolic Systems

The symbolic approach to cognition can best be seen in Newell and Simon's *physical-symbol system hypothesis* (Newell & Simon, 1972, 1976; Newell, 1980, 1990). A physical-symbol system is a physical device that contains a set of interpretable and combinable items (symbols) and a set of processes that can operate on the items (copying, conjoining, creating, and destroying them according to instructions) (Newell & Simon, 1976, p. 86). The physical-symbol system hypothesis states that a physical symbol system has the necessary and sufficient means for general intelligent action (Newell & Simon, 1976, p. 87). This is a strong empirical claim on the nature of intelligence. It states that any system that manipulates symbols is sufficient for producing intelligent behavior, and further that all intelligent systems are necessarily implementations of physical-symbol systems.

In practical terms, the types of syntactic manipulation of symbols found in formal logic and formal linguistic systems typifies this view of cognition. In this viewpoint, external events and perceptions are transduced into inner symbols to represent the state of the world. This inner symbolic code stores and represents all of the system's long-term knowledge. Actions take place through the logical manipulation of these symbols to discover solutions for the current problems presented by the environment. Problem solving takes the form of a search through a problem space of symbols, and the search is performed by the logical manipulation of the symbols through stated operations (copying, conjoining, etc.). These solutions are implemented by forming plans and sending commands to the motor system to execute the plans in order to solve the problem. In the symbolic viewpoint, intelligence is typified by and resides at the level of deliberative thought. Modern examples of systems that fall within this paradigm include SOAR (Laird, Newell, & Rosenbloom, 1987) and ACT-R (Anderson, Silverstein, Ritz, & Jones, 1977).

Discrete symbolic systems do have a competitor within the symbolic paradigm. These models in general use probabilistic declarative structures and are often referred to as gradient models. They are motivated by psychological findings that membership in human categories is often not black and white. People have ideas on the degree to which a certain example belongs in a category, and they have notions of the prototypical member of a category. For example, a robin might be many peoples quintessential idea of a member of the 'bird' category, while a penguin has some decidedly unbirdlike characteristics (swims, doesn't fly) which makes it seem not 100% part of the category of 'bird'. Discrete symbolic systems have been criticized as unhumanlike in this regard when trying to form perceptual categories though see (Miller & Laird, 1996), for an attempt to allow discrete symbolic symbols to display graded responses.

Symbolic systems are often equated with the machine metaphor of mind. In this viewpoint of cognition, the brain is seen in some sense as a computer. The physical brain represents the hardware of the system, and the mind represents the software. The machine metaphor is a very attractive position for many reasons. It explains how the mind connects with and controls the body, the old mind-body problem, in a way that does not resort to a form of dualism.

The symbolic approach works well as a model of cognition, and is capable of modeling many impressive examples of intelligent behavior in AI. However, challenges to this viewpoint of cognition have appeared, both as practical criticisms of the performance of such systems and more philosophical challenges to the physical-symbol system hypothesis.

On the practical side, symbolic models are notoriously inflexible and difficult to scale up from small and constrained environments to real world problems. If symbolic systems are both necessary and sufficient for intelligent behavior, why do we seem to have such problems in producing the flexibility of behavior exhibited by biological organisms?

The inability of symbolic systems to cope with such problems has lead many to a new viewpoint of cognition. When one views cognition as mainly working on the level of deliberative thought, then the hard problems of intelligence appear to be those such as logic and language use. From this viewpoint, the abilities of organisms to orient themselves spatio-temporally, form perceptual categories and develop basic motor skills seem to be easy problems that can be immediately solved once basic systems exist to take care of the harder problems of deliberative thought. But if the physical-symbol system hypothesis does not hold and deliberative thought is not the basic level where intelligence resides, then this viewpoint may be exactly backwards. Those abilities that are so easily dismissed as simple because all children learn them with seeming effortlessness are instead seen as complex and essential to cognition. Perhaps it has taken most of the time of evolution to solve these basic features of intentional activity, and language and logic are phylogenetically more recent and comparatively easy to solve once the proper base of spatio-temporal skills is in place to support them.

2.2.2 Connectionist Systems

A connectionist view of cognition provides an alternative theory of mind to the symbolic approach. The connectionist approach to cognition has existed for as long as the symbolic approach. However, symbolic viewpoints of cognition have dominated the field of cognitive science until a resurgence of interest in connectionist models in the mid '80s.

The connectionist approach differs from the symbolic paradigm in almost all major dimensions. Connectionist models offer a subsymbolic paradigm, where representations are built from the changing contributions of processing units that represent features below the normal level of human symbolic features. Connectionist models emphasize parallel processing, while symbolic systems tend to process information in a serial fashion. Connectionist representations are distributed over many units, while cognitivist symbols are static localized structures. Connectionist models offer many attractive features when compared with standard symbolic approaches. They have a level of biological plausibility absent in symbolic models that allows for easier visualization of how brains might process information. Parallel distributed representations are robust, and flexible. They allow for pattern completion and generalization performance comparable to biological organisms. They are capable of adaptive learning. In short, connectionist models are an attractive alternative model of cognition.

The connectionist hypothesis might be stated as: large-scale parallelism of (relatively simple) non-linear processing units doing local processing and producing distributed representations are necessary and sufficient to the production of general intelligent behavior.

First Generation

Clark (2001) categorizes modern connectionism into three generations. The first-generation of connectionism, that began with the perceptron and the work of the cyberneticists (McCulloch & Pitts, 1943; Rosenblatt, 1958), was revived in the mid '80s with the PDP research groups work (among others) on parallel distributed processing (Rumelhart et al., 1986). First-generation connectionist systems were typified by a multi-layer architecture (usually composed of two or three layers) with strictly feed-forward connections. Back-propagation learning rules have been especially successful in the proliferation of these models (Werbos, 1974). Such architectures are very familiar to practitioners of AI and Neural Network research. These connectionist models of cognition are very attractive and important for many reasons. They are biologically plausible models with some of the flexibility of pattern-recognition and generalization exhibited by biological organisms.

Second Generation

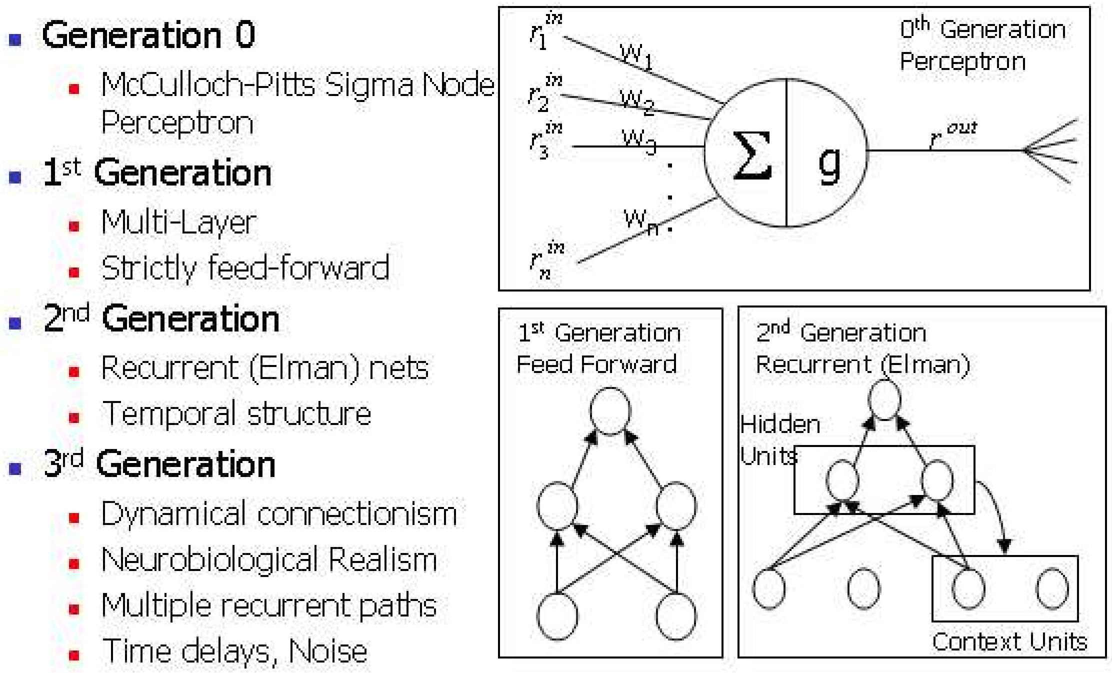
Second-generation connectionism began to appear in the early '90s. Second-generation connectionism extends first-generation networks to begin to deal effectively with dynamic spatio-temporal events. First-generation networks displayed no real capacity to deal with time or order in the environment. Second-generation connectionist systems added recurrent connections to the networks in order to expand these capabilities (Elman, 1990, 1991). Recurrent connections are connections that connect later layers in the network with earlier layers. So second-generation connectionist networks are no longer strictly feed-forward, they contain recurrent connections. The addition of recurrent connections allows for previous states of the network to affect decisions about the current input. In essence, recurrent connections provide a type of short-term memory that allows for the categorization of patterns extended in time across the inputs of the network. This ability to deal with spatio-temporally extended patterns in time is an important addition to the capabilities of connectionist systems.

Third Generation

Third-generation connectionism is the most recent extension of the connectionist paradigm. This generation of models is typified by even more complex dynamic and time involving properties. These models use more complex, and biologically inspired architectures, along with various recurrent and hard-coded connections. So, for example, rather than the typical three layers of first and second generations, third-generation networks may have many areas that represent and reflect architectures and subsystems of biological brains. Because of the increasing emphasis on dynamic and time properties, third-generation connectionism has also been called dynamic connectionism. See Figure 2.1 for a summary of Clark's classification of connectionist generations.

2.2.3 Biological Goal Management Mechanisms from Ethology

The cognitive mechanisms involved in the processes of goal formation, selection and prioritization have received little attention when compared with other cognitive processes such as perception and memory. However, it can be argued that the fundamental properties of what we label as intelligent actions in biological organisms flow from and are in the service of such goals. Goal mechanisms would seem to play a fundamental role in intelligent, intentional behavior, but how animals and people manage multiple, conflicting, complex needs remains obscure. We will now discuss insights that complex systems theories bring to understanding the emergent properties of goal mechanisms in biological organisms. We will place this discussion in the historical context of various schools of thought in cognitive science, outside of the symbolic and connectionist paradigms, on how such mechanisms operate in people and animals and how they may be modeled.



- **Generation 0**
  - McCulloch-Pitts Sigma Node Perceptron
- **1st Generation**
  - Multi-Layer
  - Strictly feed-forward
- **2nd Generation**
  - Recurrent (Elman) nets
  - Temporal structure
- **3rd Generation**
  - Dynamical connectionism
  - Neurobiological Realism
  - Multiple recurrent paths
  - Time delays, Noise

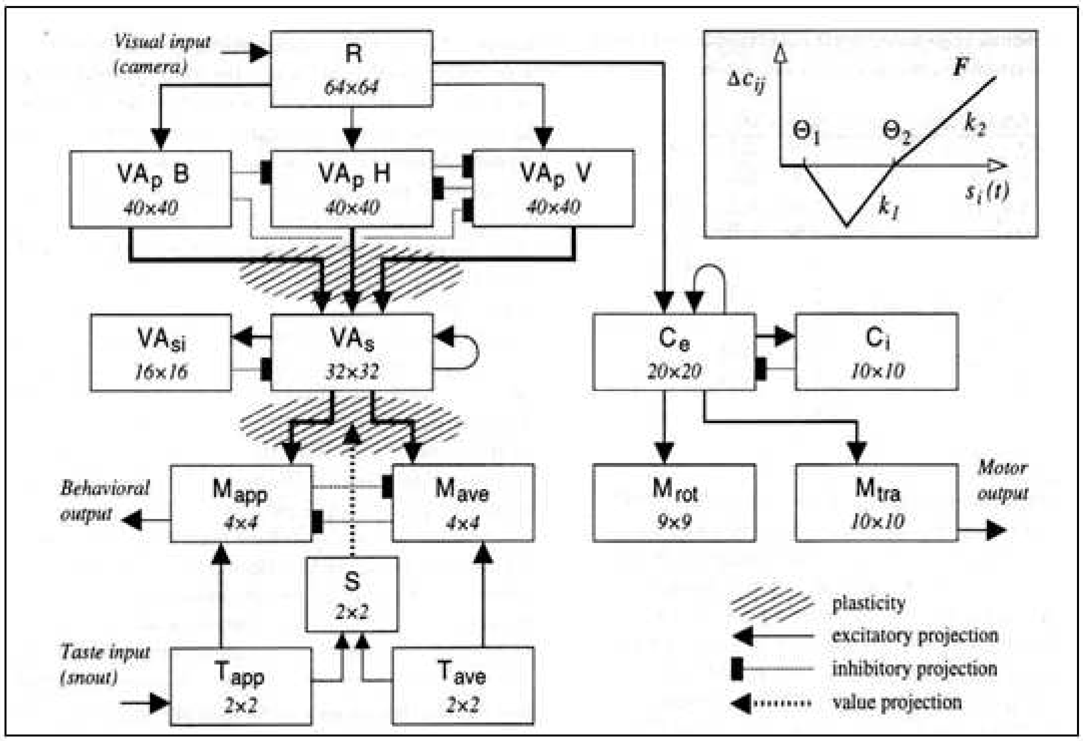


Figure 2.1: Clark's (2001) classification of generations of connectionist systems. Connectionist models have been moving towards more biological architectures, recurrent connections and time-varying dynamics

Ethology is a branch of science concerned with the study and understanding of behavior in animals. Researchers in ethology tend to focus on instinctual behavior patterns, those developed over phylogenetic time scales. Such behaviors are phylogenetic adaptations of a species to its environmental niche, and thus reflect the evolutionarily *learned* history of the species. This is often contrasted with those scientists interested in the development of behaviors learned during the lifetime of a single organism, in other words on an ontogenetic time scale. People who consider themselves developmental psychologists and animal behavioral researchers are interested in learning processes on this time scale.

Complex systems theory has shown that these types of learning, ontogenetic and phylogenetic, may have many characteristics in common that depend on the emergent properties of collective components guided by a historical process. The difference between the two is really one of time scale, phylogenetic over time scales of species, and ontogenetic over time scales of individuals; and of the nature of the collective components being subjected to competitive and selectional pressures. As we will discuss later on, one of the goals of research in complex systems theory is identifying these deep mechanisms common to organization of systems across different time scales.

Even though the behavior patterns studied by ethologists are innate, this does not mean that they are simple reflex actions. Instinctual behavior patterns are very complex, and cover things like nest building in birds, spider web building patterns, and predator hunting patterns just to name a few. Ethologists since the creation of their discipline have been proposing models of the goal-oriented behavioral patterns they observe. Ethological models of behavior, such as those proposed by Tinbergen (1951), Lorenz (1957) and Baerends (1970), are usually conceived of as hierarchical structures of superordinate and subordinate behavioral patterns.

Tinbergen (1951) proposed a popular model of what he called behavior centers, which is a hierarchical structure meant to explain how behavior patterns are organized, prioritized and managed by an organism. For example, think of a common but complex animal behavior like hunting behavior in many mammalian predators such as wolves and cats:

> Every predator hunts in basically the same way. It starts with "search," which turns into "eye-stalk" when a potential meal is found. Once close enough, "chase" begins. "Grab-bite" brings dinner down, and "kill-bite" finishes the job (Coppinger & Coppinger, 2001).

This is usually conceived of as a temporal sequence of distinct behavior patterns, executed one after the other. Many mechanisms may trigger the switching from one pattern to the next in the sequence, although within a level usually we think of environmental cues as being the primary mechanism for such switching, such as the visual spotting and identification of prey which causes "search" to cease and "eye-stalk" to commence. Of course such behavioral patterns are much more complex than simple fixed sequences. For example the appearance of a bigger predator in the middle of the hunt pattern would cause the sequence to be abandoned in favor of a more pressing higher level goal, that of fleeing in order to survive. We will discuss these complexities next.

Figure 2.2 is a possible model of the mammalian "hunt" behavior pattern using Tinbergen's behavior center concept. In this model, the "hunt" behavior is superordinate to the lower level behaviors that can be triggered in the service of fulfilling the hunt goal. The "hunt" behavior might for example be grouped with other higher level goals of a similar nature, such as "body-temperature-maintenance", "water-finding", etc. These might in turn all be conceived as being subordinate to a higher-level goal such as 'day-to-day survival. In turn their could be other such hierarchical groups concerned with different aspects of behavior, such as procreation.

The factors that influence the selection of a particular goal are of course varied and not exactly known. Tinbergen as well as others usually concede there are multiple factors that must go into the decision, both internal and external, and that this process is a very complex blending of multiple constraints and needs. In Tinbergen's behavior centers model, the goals on a particular level are in competition with one another, and only one

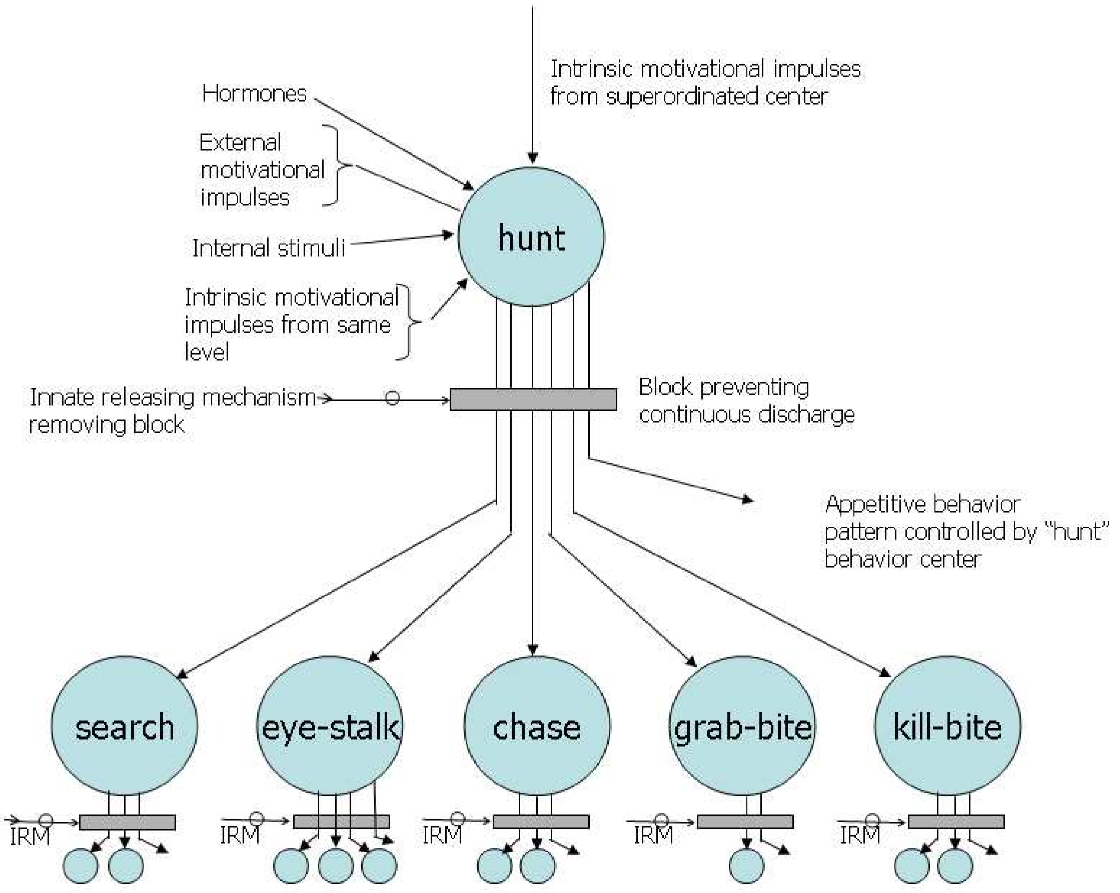

Intrinsic motivational impulses from superordinated center
Hormones
External motivational impulses
Internal stimuli
Intrinsic motivational impulses from same level
hunt
Innate releasing mechanism removing block
Block preventing continuous discharge
Appetitive behavior pattern controlled by "hunt" behavior center
search
eye-stalk
chase
grab-bite
kill-bite
IRM
IRM
IRM
IRM
IRM

Figure 2.2: Example of Tinbergen's representation of a appetitive goal structure, in this case for the "Hunt" appetitive goal. Figure based on Tinbergen (1951, p. 124).

will be active at any given moment. When a goal is the current focus of the organism, it causes what Tinbergen refers to as the release of a block (the innate releasing mechanism or IRM), such that subordinate goal/behavior patterns can become active and try and fulfill the superordinate goal. The release of a block allows the set of behavior patterns on the subordinate level to participate in their own competition. In the "hunt" example, the selection of "search", "eye-stalk", etc. would depend on many factors such as memory of past successful hunting locations; auditory, visual and olfactory cues and so on.

Notice that Tinbergen's model does have mechanisms for dealing with the interruption of behavioral patterns due to priority or opportunity. In the hypothetical example of the predator encountering a bigger predator during its hunt, the idea is that external motivational impulses and internal stimuli would suddenly make a "flee" goal become more active than the current "hunt" goal. In this case, the block preventing discharge would be reactivated for the "hunt" behavior patterns, and innate releasing mechanisms would release the block for "flee" behaviors. Once the danger is avoided ("flee" is satisfied or appetitive in the terminology), the behavior centers model would predict that previously high priority goals would reassert themselves (the predator is still hungry, for example), the result being like a return from the interrupted activity and a reinstantiation of the interrupted behavior pattern. This mechanism would work the same for interrupts due to opportunity, such as passing a river and deciding to take a drink while hunting.

Hierarchical models from ethology are very useful as explanations of some aspects of behavior pattern generation in biological organisms. Many of the insights derived from such models are still considered correct. Ethological models do, however, leave many questions unanswered that are being tackled today through the use of computational modeling and complex systems theory. Ethological models such as Tinbergen's mostly address the questions of how such mechanisms operate. They are a bit more vague on how such mechanisms come to exist in the first place, though of course evolutionary mechanisms are considered the primary explanation for the development of such systems in biological brains. Also, without concrete computational models, such models from ethology leave many questions unanswered on how perception, memory and internal sensors can provide stimuli that drive the mechanisms, and how such stimuli are combined and acted upon.

2.2.4 Subsumption Hierarchies and Emergent Behavior

One obvious step in the development of hierarchical goal management models is their translation and implementation into computational models in autonomous and robotic agents. Computational implementations allow us to develop much more precise ideas on the details of how such mechanisms might operate. Of course, the beginnings of ethology as a science predated the tools (computers) necessary to build such computational models. But there are many current research thrusts interested in biologically inspired models of such behavior pattern generation that owe much to the early hierarchical models of ethology. Hierarchical models of behavior have been used extensively as action selection mechanisms for autonomous agents (Blumberg, 1994; Digney, 1996; Huntsberger, 2001; Tunstel, 2001; Harter & Kozma, 2004h)

One of the missing pieces in such hierarchical models seems to have been an understanding of the roles that emergent processes play in their functioning. One of the first models to emphasize the processes of emergent behavioral control was Brooks' subsumption hierarchy (Brooks, 1990, 1993). The subsumption architecture emphasized many new concepts in the design of behavior producing systems, such as embedding or situating the agent in the environment, tight connection of perception to action, and emergent behavioral control.

Figure 2.3 shows an example of a subsumption architecture for a robot built by Brooks. This particular example has three levels. The bottom layer corresponds to an object-avoidance behavior pattern, the middle layer to simple non-directed wandering behavior, and the top layer to a more systematic exploration of the environment. Brooks'

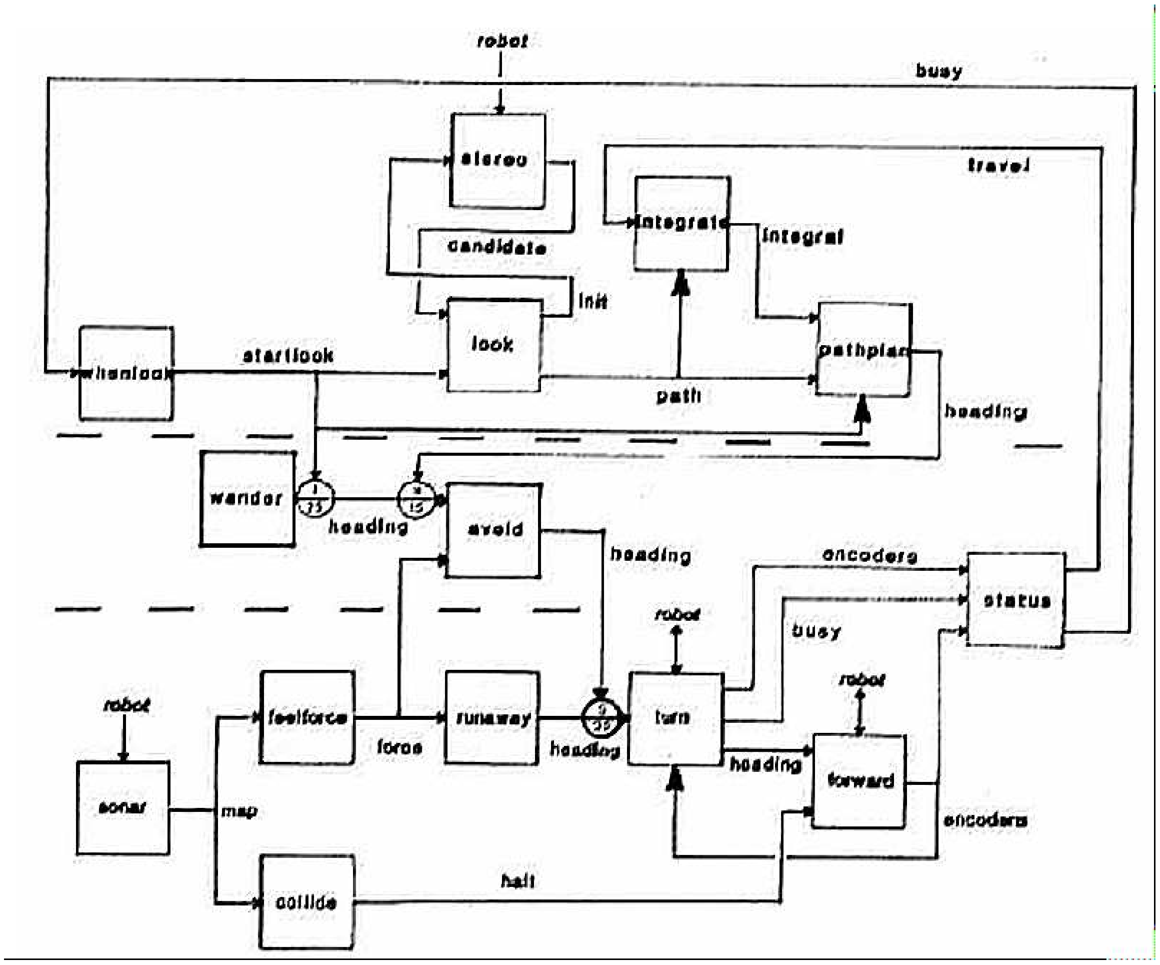


Figure 2.3: An example subsumption hierarchy that implements object avoidance (bottom layer), wander (middle layer), and explore (top layer). Figure taken from Brooks (1986, p. 7).

subsumption hierarchy has many common features with Tinbergen's model of behavior centers in biological organisms. It divides behavioral patterns into hierarchical levels. Also centers at any layer can receive external and internal stimulation as well as intrinsic motivational pulses from the same and superordinate levels.

Brooks computational model of behavior provides many concrete answers to some of the questions that the ethological models raised. For example, centers in the subsumption model are implemented as finite state automata. Simple signals can be passed to and from other centers that act as pieces of information that affect the state transitions of the automata within a center. Thus we see an example of how multiple stimuli can be combined to effect the performance of a center.

The subsumption method of robot behavior generation design provided us with many new insights on some of the short-comings of the earlier ethological models (not to mention symbolic based approaches to designing action selection mechanisms). The major achievement of the subsumption hierarchy was to demonstrate that the collective emergence of behavior from relatively independent and simple behavior patterns is an important concept in behavior producing mechanisms that offers many advantages of flexibility and adaptability. As can be seen from Figure 2.3, centers on a common level often have no easily defined, linear type relation. It is through the direct coupling of the centers to external and internal stimuli that much of the behavioral complexity is produced. Also, the centers in the subsumption architecture can operate in many combinations, such that a combination of many small behaviors in complex sequences produces (emerges) very complex higher level behaviors. Brooks' argument for the importance of emergent mechanisms was mostly aimed at the classical symbolic approaches to cognition. But his emphasis on the power of emergent processes can also be seen as providing some of the missing pieces to the ethological views of behavior pattern generation.

Brooks subsumption approach is ultimately an engineering approach to developing control mechanisms for behavior in autonomous systems. Its basic thrust is how to design independent, heterogeneous behavioral components that can produce sophisticated, and sometimes surprising, emergent higher level behaviors. When applied to ethological models, the implications are clear. While our human intuition might be tempted to identify clear behavioral patterns that have fairly linear relationships, the development of such patterns by evolution is probably much closer to the model of emergent patterns being selected and evolved for by evolutionary pressures on the species. Further, emergent patterns among independent and somewhat redundant behavior patterns can offer many advantages in terms of flexibility and subtlety in fulfilling the needs of the organism. However, the subsumption architecture ultimately only guides engineers in how to design such emergence. The designs for emergence that engineers can develop will tend to be relatively limited in complexity. What is needed is an understanding of how evolutionary and developmental mechanisms organize such systems without a human engineer involved in the process.

2.2.5 Developmental Systems Approaches to Self-Organization

At the other end of the spectrum from studying instinctual behavior are those interested in studying learning processes in animals and humans. Although ontogenetic and phylogenetic processes have historically been studied as separate disciplines, there is a growing awareness of the commonality of many of the mechanisms of these on a deep level (Oyama, 1985; Thelen, 1995). Oyama especially has argued for the artificiality of the nature/nurture divide and called for a clearer recognition of the common processes among both time scales.

Thelen and Smith (1994), Thelen (1995) are among a new group of developmental psychologists that view developmental systems in terms of the emergence of dynamics in complex systems. They envision the development of behavior in cognitive systems as an ontogenetic landscape of stable and unstable attractors and repellors. As the body of the organism changes, new opportunities for behavior are created and destroyed.

Development is seen as a reduction of the degrees of freedom of the system as useful patterns for solving problems are discovered. As stable solutions to problems develop, these in turn change the ontogenetic landscape, opening up new opportunities for some behaviors, and closing off opportunities for others. Development is the discovery of stable patterns of behavior, given the current constraints of the body and the environment.

The view of behavior organization from a developmental perspective in ontogenetic time frames is important because it allows us to begin to study not simply how the mechanisms of behavior production operate, but also how they are formed and developed. Thelen and Smith, among others, have begun the process of seeing the development of behavior as an intrinsically coupled process, where the natural dynamics of the system interact with the constraints of the environment and body. They have begun to use the methods and language of dynamical systems such as attractors and bifurcations to understand these tightly coupled developmental processes.

Viewing development as an interaction of the dynamics of organism and environment allows for certain predictions when one or the other is changed. For example, development of locomotive behaviors, such as crawling, walking and running in human children, might be viewed very differently if the development took place in an atypical environment, such as in a low gravity environment on the moon. A dynamical perspective of development would recognize that there are different solutions to the problem of locomotion afforded by the low gravity environment. Therefore, a child developing in such an environment would not be expected to go through the typical stages of development (crawling, walking, etc.). Instead, different modes would be discovered that are reflective of the capabilities of the developing body and the low gravity environment (bounce walking, etc.).

2.2.6 Complex Systems Approaches to Behavior Pattern Development

So far we have identified two major structural/functional properties that seem to operate in behavior pattern generation. One of these is the stratification of behavior patterns into hierarchies. Hierarchies are usually conceived of as being composed of homogeneous elements. For example a military hierarchy sorts people into homogeneous ranks which are joined together through a chain of command. Similarly Tinbergen's and Brook's models of behavior generation have such homogeneous collections of elements collected together into control levels. However, in Brooks case, the levels are not so rigid, nor are the components in a sense quite homogeneous. The other type of structure is where a collection of heterogeneous components are able to negotiate their needs in a complex way among themselves. The collective behavior of such a heterogeneous group can display emergent properties. In a sense, Brooks' collection of (heterogeneous) finite state automata, each concentrating on a simple task, are an example of this type of heterogeneous collection. We will call this other type of property a meshwork (De Landa, 1997).

These two structures have very different properties and purposes. However, it is relatively hard to find pure cases of one or the other in complex systems that we observe around us. Almost always we observe a mix, a meshwork of hierarchies or a hierarchy of meshworks, or even more complicated interactions. For example, biological species can be thought of as an example of a homogeneous collection of elements (more or less depending on the genetic variation in the species gene pool). Environmental ecosystems are classic examples of meshworks of these homogeneous elements. Species interact with each other in many complicated ways. Ecosystems depend on the development of self-sustaining loops of interacting species with each species occupying an environmental niche in the ecosystem. This meshwork of hierarchies forms a complicated food web linking together a wide variety of animals and plants. Incidentally, the more redundancy and variability in an ecosystem, the more resilient and flexible it tends to be in the face of environmental changes, disasters and other challenges to its existence.

Complex systems theories of these relationships may also be applicable to the processes of behavior production and goal formation in biological organisms. The point is that we now think of behavior generation no longer as a simple hierarchical organization, nor as a single meshwork. Goal formation and organization appears, as in many other complex systems, as a combination of both hierarchies and meshworks of elements. It therefore shares more in common with processes that we observe in the formation and maintenance of ecosystems, than in other systems that are strictly or mostly of one type or the other.

It is worthy to note that not only competitive processes operate in ecosystems. Predator/prey and parasite/host evolutionary arms races are important driving forces in evolutionary processes. But also cooperative processes exist such as symbiosis. Also, as hinted at previously, the self-sustaining loops of matter and energy that occur in ecosystems are important pieces to the puzzle of understanding how they work. These complex autocatalytic loops seem to organize naturally in complex systems with far from equilibrium dynamics. All of these processes, among others known and still unknown, are also present in the goal formation and behavior generation mechanisms of biological brains.

2.3 Nonlinear Dynamics and Cognitive Systems: The Fourth Generation

Biological brains exhibit aperiodic oscillations with a much more rich dynamical behavior than fixed-point and limit-cycle approximation allows. Early connectionist systems captured some of the flavor of neuronal functioning, but abstracted away much of this rich dynamical behavior in favor of simple fixed-point dynamics (Kohonen, 1972; Anderson et al., 1977; Grossberg, 1980; Hopfield, 1982). Second and third generation systems recapture some of the more complex dynamics because of recurrent connections and specialized architectures, but many are still parameterized to ultimately settle down to fixed-point attractors. The question of what use, if any, aperiodic dynamics may play in cognition has largely been ignored, or its possible significance unrealized. The exploration of nonconvergent dynamics in cognitive processes may constitute the fourth generation of connectionist thought in its evolution towards capturing more of the dynamics and functioning of biological brains. In this section we will argue that, far from being unnecessary noise of no use in cognition, aperiodic dynamics are necessary for general intelligent behavior.

2.3.1 Nonlinear Dynamics in Science

The study of nonlinear dynamics has blossomed in all areas of science in the past decades for many reasons. Nonlinear dynamics provide new conceptual and theoretical tools that allow us to understand and examine complex phenomena that we have never been able to tackle before. Nonlinear dynamics seem to show up everywhere, in physical systems like electrical circuits, lasers, optical and chemical systems. But we especially see its ubiquity all around us in the biological world, from fractal growth patterns in biological development and city formation to the self-organizing characteristics of population models, and the importance in regulating healthy biological rhythms such as the beating of the heart.

The study of nonlinear dynamics is concerned with systems whose time evolution equations are nonlinear. But why is nonlinearity an important property of systems? Simply put, in a linear system if you change a parameter or perturb the system, the amplitude and frequency of the system may change, but the qualitative nature of the behavior will remain the same. For example, perturbing an oscillating pendulum does not change the fact that it continues to oscillate with a regular harmonic motion. In a nonlinear system, however, this fact does not hold. Perturbations of the system can cause a qualitative change in the behavior of the dynamics. A small perturbation can cause sudden and dramatic changes in both the qualitative and quantitative behavior of the system (Hilborn, 1994).

This property of nonlinear systems is important in at least two ways. A nonlinear system exhibiting aperiodic dynamics is constantly generating and exploring unique areas of its possible behaviors. This generation of diversity can be a very important property in producing flexible, intelligent behavior. Also, random environmental perturbations can be handled by a nonlinear process, where they would be disastrous to a linear one. For example, a random perturbation to a portion of beating heat muscle fiber that was governed by a linear oscillation would cause the fiber to beat in a new period, possibly conflicting with other areas and leading to fibrillation. Nonlinear processes governed by an attractor can become entrained to one another, and even in the face of random perturbations they will eventually settle back down to beat in synchrony. Nonlinear processes are able to handle random environmental perturbations, but still maintain coherent dynamical state. Linear processes are destroyed by randomness.

Really, nonlinear dynamics are the rule not the exception in systems observed in nature. Almost all real systems are nonlinear at least to some extent. Classical physics and science has been surprisingly successful in using idealized linear approximations to model many physical processes. The range of phenomena that could not be handled successfully by linear approximation was however much greater. But some phenomena at least are beginning to be understood through the application of nonlinear dynamics. To quote a famous mathematician Stanislaus Ulam, “Calling the subject nonlinear dynamics is like calling zoology 'nonelephant studies' ” (Gleick, 1987). The study of linear systems restricts science to only a very small portion of the vast range of possible dynamics. The complex behavior of the brain, we would argue, is one such phenomena where the application of nonlinear dynamical modeling can yield greater understanding.

In a sense, symbolic systems seem to belong in the classical vein of using a linear approximation to model a phenomenon. In this case the systematic application of formal logic to the tasks of problem solving and memory. But these systems suffer the same weaknesses that all linear approximations share. Unexpected situations or perturbations cannot be handled by such models. We will next look at some of the evidence for nonlinear dynamics in the functioning of perceptual neurological functioning.

2.3.2 Nonconvergent Dynamics for Perception

In their influential paper, Skarda and Freeman (1987) argued that chaos, as an emergent property of intrinsically unstable neural masses, is very important to brain dynamics. In experiments carried out on the olfactory system of trained rabbits, Freeman was able to demonstrate the presence of chaotic dynamics in EEG recordings and mathematical models. In these experiments, Freeman and his associates conditioned rabbits to recognize smells, and to respond with particular behaviors for particular smells (e.g. to lick or chew). They performed EEG recordings of the activity in the olfactory bulb, before and after training for the smells.

The EEG recordings revealed that in fact, chaotic dynamics (as shown by the observed strange attractors) represented the normal state when the animal was attentive, in the absence of a stimulus. These patterns underwent a dramatic (nonlinear) transition when a familiar stimulus was presented and the animal displayed recognition of a previously stored memory (through a behavioral response). The pattern of activity changed, very rapidly, in response to the stimulus in both space and time. The new dynamical pattern was much more regular and ordered (very much like a limit cycle, though still chaotic of a low dimensional order). The spatial pattern of this activity represented a well defined structure that was unique for each type of odor that was perceptually significant to the animal (e.g. conditioned to recognize). Figure 2.4 shows an example of such a recorded pattern after recognition of a stimuli of the EEG signals and the associated contour map. In this figure after recognition, all of the EEG waves are firing in phase, with a common frequency (which Freeman called the carrier wave). The pattern of recognition is encoded in the heights (amplitude modulations) of the individual areas. The amplitude patterns, though regular, are not exact limit cycles and exhibit low dimensional chaos. In other words, different learned stimuli were stored as a spatio-temporal pattern of neural activity, and the strange attractor characteristic of the attention state (before recognition) was replace by a new, more ordered attractor related to the recognition process. Each

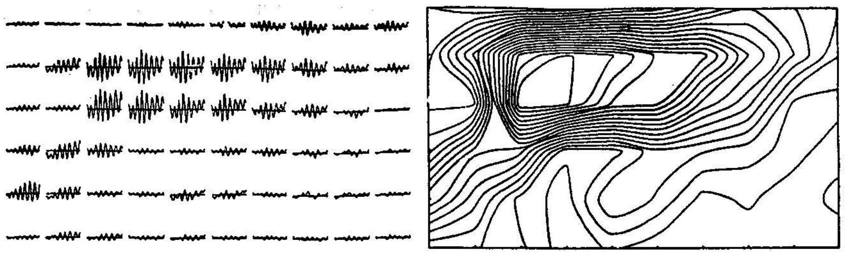


Figure 2.4: EEG carrier wave patterns (left) and contour map (right) of olfactory cortex activity in response to a recognized smell stimulus (from Freeman, 1991, p. 80)

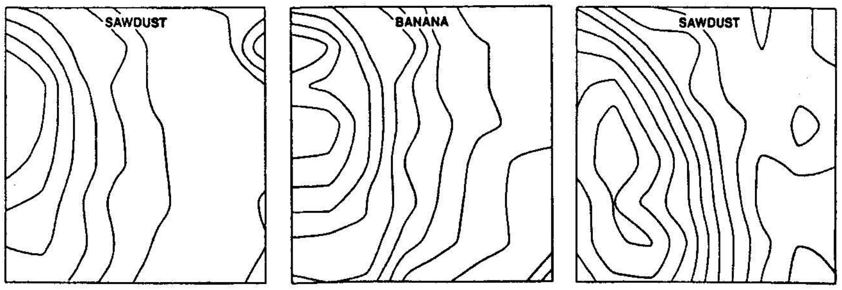


Figure 2.5: Change in contour maps of olfactory bulb activity with the introduction of a new smell stimulus (from Freeman, 1991, p. 81)

(strange) attractor was thus shown to be linked to the behavior the system settles into when it is under the influence of a particular familiar input odorant.

Figure 2.5 shows the effects on the spatial attractor pattern due to learning. Every time a new odor was learned by the animal, all of the existing attractor patterns changed. In this figure the contour pattern of activity for sawdust is shown (before learning the banana odor), for the banana odor, and then again for sawdust. Notice that the spatial pattern for sawdust no longer resembles its previous pattern. Whenever an odor becomes meaningful in some way, changes in the synaptic connections between neurons in different parts of the olfactory cortex take place. Just as in the Hopfield model and other neural networks, these changes are able to create another attractor, and all other attractors are modified as a result of this learning. However, in real brains, the attractors of perceptual meaning are not simple point attractors, but are specific strange attractors.

Freeman suggests that "an act of perception consists of an explosive leap of the dynamic system from the basin of one (high dimensional, in the attentive state) chaotic attractor to another (low dimensional state of recognition) (Freeman, 1991). These results suggest that the brain maintains many chaotic attractors, one for each odorant an animal or human being can discriminate. Freeman and Skarda speculate on many reasons why these chaotic dynamics may be advantageous for perceptual categorization. For one, chaotic activity continually produces novel activity patterns which can provide a source of flexibility in the individual. But since chaos is a ordered state, such flexibility is under control. As Kelso remarks, such fluctuations continuously probe the system, allowing it to feel its stability and providing opportunities to discover new patterns (Kelso, 1995). Another advantage of chaos is that it allows for very rapid switching between attractors, which random activity is not able to do. Freeman also proposed that such patterns are crucial to the development of nerve cell assemblies. For example high dimensional chaos may provide a neutral pattern of correlation activity so that learning does not occur during the attentive state. Only upon collapse of activity to more ordered regions do regular phase synchronizations occur between neural areas, which allow for Hebbian synaptic changes to reliably occur.

2.3.3 Necessary and Sufficient Conditions of Nonconvergent Dynamical Viewpoint

Aperiodic dynamics play a significant role in the organization of perceptual mechanisms in biological organisms. The presence of self-organizing critical states have also been detected in other brain systems. These observations have led to the hypothesis that such dynamics are ubiquitous in brains, and are necessary to the flexible organization of biological behavior. Symbolic systems provide little insight into how they may be connected with an environment and generatively construct knowledge about the world they experience. Looking at symbolic systems as models of biological cognition, they are also silent on why such aperiodic dynamics appear in biological brains. Classical connectionist systems have yet to explore the uses of aperiodic dynamics in memory and action.

These observations of the possible significance of nonconvergent dynamics in brains has led us to speculate on the necessary and sufficient conditions they suggest. Specifically:

- Complex, nonconvergent dynamics are necessary to the production of general intelligent behavior.
- An embodied system with appropriate environmental/sensory coupling and internal structural systems for handling the "what", "where", "why" and "how" functions of the agent are necessary to the production of general intelligent behavior.
- The exploitation of nonconvergent dynamics by and within such an appropriately embodied system are necessary and sufficient for producing general intelligent behavior.

In essence we have proposed two conditions for the production of general intelligent behavior. Aperiodic dynamics characteristic of critical states are necessary for the flexible self-organization of memory and behavior. The dynamics of the brain are strongly coupled with their environment. The interaction of brain dynamics with the environmental system produces behavior. We will explore these issues further in the next section, where we describe one such model of cognition.

2.4 A Necessary and Sufficient Model for the Production of General Intelligent Behavior

The discovery that brain dynamics exhibit chaotic features, and further, that these dynamics may be necessary for the self-organization of intelligent behavior has a profound impact on the types of theories we should propose to explain cognition. In this section we present one such model that is capable of meeting the necessary conditions for the production of general intelligent behavior outlined above.

2.4.1 K Set Primer and History

The K-set hierarchy, developed by Walter J. Freeman and associates (Freeman, 1975; Skarda & Freeman, 1987; Freeman, 1991, 1999b), is both a model of neural population dynamics and a description of the architectures used by biological brains for various functional purposes. The lowest level of the hierarchy, the K0 set, provides a basic unit that models the dynamics of a local population of tens of thousands of neurons. The dynamics of the K0 set are described by a second order ordinary differential equation feeding into an asymmetric sigmoid function:

$$\alpha\beta\frac{d^2a_i(t)}{dt^2} + (\alpha+\beta)\frac{da_i(t)}{dt} + a_i(t) = net_i(t) \tag{2.1}$$

$$net_i(t) = \sum_j w_{ij}o_j(t) \tag{2.2}$$

$$o_j(t) = \epsilon\{1 - exp[\frac{-(e^{a_j(t)} - 1)}{\epsilon}]\} \tag{2.3}$$

This equation was determined by measuring the electrical responses of isolated neural populations to stimulation and other conditions. The α and β parameters are time constants that were determined through such physiological experiments. $a_i(t)$ is the pulse density of the modeled neural population, in other words the average number of neurons that are pulsing in the population at any given point in time. $o_j(t)$ is a nonlinear asymmetric sigmoid function describing the influence of incoming activation.

A K0 unit models the dynamics of an isolated neural population. From the basic K0 unit can be built up architectures that capture the observed dynamics of increasingly larger functional brain areas. The KI models excitatory-inhibitory feedback populations. KII models interacting excitatory-inhibitory populations and correspond to organized

Table 2.1: Characterization of the hierarchy of K sets.

Type	Structure	Inherent dynamics	Examples in brain*
K0	Single unit	Nonlinear I/O function	All higher level K sets are composed of K0 units
KI	Populations of excitatory or inhibitory units	Fixed point convergence to zero or nonzero value	PG, DG, BG, BS
KII	Interacting populations of excitatory and inhibitory units	Periodic, limit cycle oscillations; frequency in the gamma band	OB, AON, PC, CA1, CA3, CA2, HT, BG, BS, Amygdala
KIII	Several interacting KII and KI sets	Aperiodic, chaotic oscillations	Cortex, Hippocampal Formation, Midline Forebrain
KIV	Interacting KIII sets	Spatio-temporal dynamics with global phase transitions (itinerancy)	Hemisphere-wide cooperation of cortical, HF and MF areas coordinated by the Amygdala

* Notations: PG - periglomerular; OB - olfactory bulb; AON - anterior olfactory nucleus; PC - prepyriform cortex; HF - hippocampal formation; DG - dentate gyrus; CA1, CA2, CA3 - curnu ammonis sections of the hippocampus; MF - midline forebrain; BG - basal ganglia; HT - hypothalamus; DB - diagonal band; SP - septum

brain regions such as the olfactory bulb (OB) or the prepyriform cortex (PC). KIII combine 3 or more KII populations to model functional brain areas such as cortex or hippocampus, and are capable of aperiodic dynamics of the type observed in these regions to, for example, derive meaning from perceptual senses. KIV is a model of the basic limbic system, combining KIII sets, to model the ”what” (perceptual), ”where” (hippocampal orientation memory), ”why” (forebrain value system) and ”how” (motor control) of a basic embodied biological agent. The KIV level theoretically is sufficient for the production of general intelligent behavior such as that observed in reptiles and simple mammals. Table 2.1 summarizes the K-set hierarchy described above.

The KI set provides a form of positive feedback. In a KI set, two or more excitatory populations are connected to one another. Their mutual excitation allows for the populations to maintain their activation levels at a steady non-zero state. The KII set allows for a form of negative feedback to be used by neural populations. In a KII, inhibitory populations are connected with excitatory populations. As the excitatory population becomes more active it begins to more actively stimulate the inhibitory population. The inhibitory population, in turn, then inhibits the excitatory population and through a process of negative feedback maintains the dynamics of the system within certain bounds. Negative feedback is a well known process of homeostatic control and is used extensively in artificial systems such as a thermostat to maintain a constant room temperature. Negative feedback is characterized by damped oscillatory or cyclic behavior. Positive and negative feedback are well known properties of biological systems and were first explored extensively for control of systems by cyberneticists (Wiener, 1965).

In the original K model, the purpose of the KIII set was to model the chaotic dynamics observed in rat and rabbit olfactory systems (Freeman, 1987; Shimoide, Greenspon, & Freeman, 1993; Freeman & Shimoide, 1994). KII are capable of oscillatory behavior, as described above. When three or more oscillating systems (KII) of different frequencies are connected through positive and negative feedback, the incommensurate frequencies can result in aperiodic dynamics. The dynamics of the KIII are produced in just this manner, by connecting three or more KII units of differing frequencies together. The KIII set was not only capable of producing time series similar to those observed in the olfactory systems under varying conditions of stimulation and arousal, but also of replicating power spectrum distributions characteristics of biological and natural systems in critical states (Bak, Tang, & Wiesenfeld, 1987; Solé & Goodwin, 2000).

The power spectrum is a measure of the power of a particular signal (or time series as for example that obtained from an EEG recording of a biological brain) at varying frequencies. The typical power spectrum of a rat EEG (see Figure 2.6, top) shows a central peak in the 20-80 Hz range, and a $1/f^{\alpha}$ form of the slope. The measured slope of the power spectrum varies around $\alpha = -2.0$. $1/f^{\alpha}$ type power spectra are abundant in nature and are characteristic of critical states, between order and randomness, at which chaotic processes operate. Power spectra of biological brains have been observed to vary from $\alpha = -1.0$ to $\alpha = -3.0$. The atypical part of the experimental EEG spectra is the central peak, indicating stronger oscillatory behavior in the γ frequencies. This central peak in the 20-60 Hz range is known as the γ frequency band, and is associated with cognitive processes in biological brains. The K-models are capable of replicating the power spectra of biological EEG signals, as shown in Figure 2.6, bottom (Freeman & Shimoide, 1994; Freeman, 1995; Harter & Kozma, 2004d).

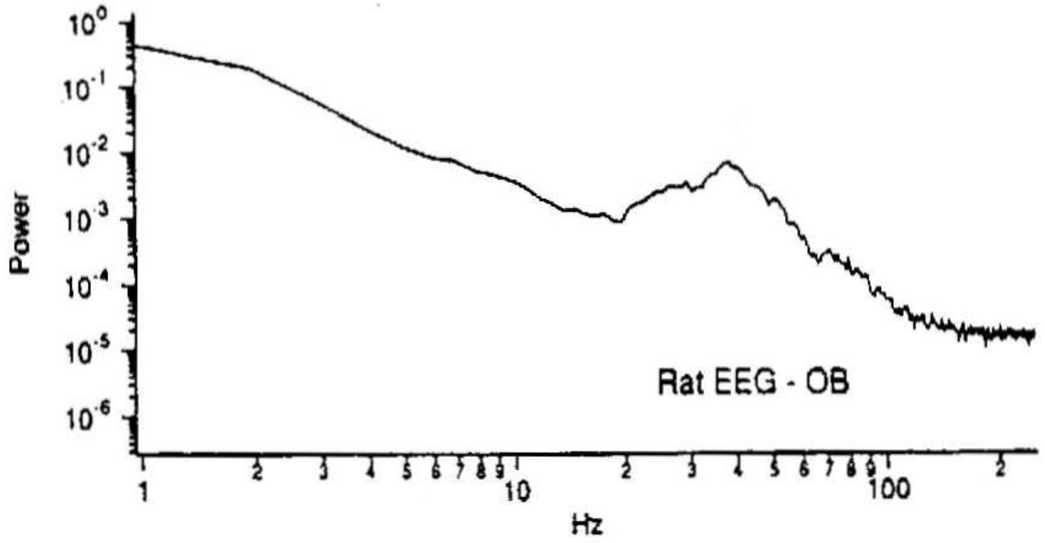
The KIII sets have been shown to be capable of organizing perceptual categories in the fashion observed in biological perceptual systems. The KIII used as such a pattern classifier is very robust and compares well with more standard methods of pattern classification (Kozma & Freeman, 2001a).

The development of the K-set hierarchy from K0 to KIII is necessary for modeling and explaining the dynamics observed in perceptual processes of biological brains. Not only does it involve positive and negative feedback mechanisms for homeostatic control, but also utilizes aperiodic dynamics for flexible generation and recognition of patterns. Though work through the KIII set have focused on perceptual category formation, there is evidence that the basic units of dynamics developed by the KIII for perception are also used in the organization of memory and behavior dynamics. The expansion of the KIII set to model a complete agent, involving the “What” “Where” “Why” and “How” systems of condition two will be described in the next section.

2.4.2 Limbic System and Intentional Behavior

Intentional behavior, in the words of Freeman (2000), is:

> ... an act of observation through time and space, by which information is sought for the guidance of future action. Sequences of such actions constitute



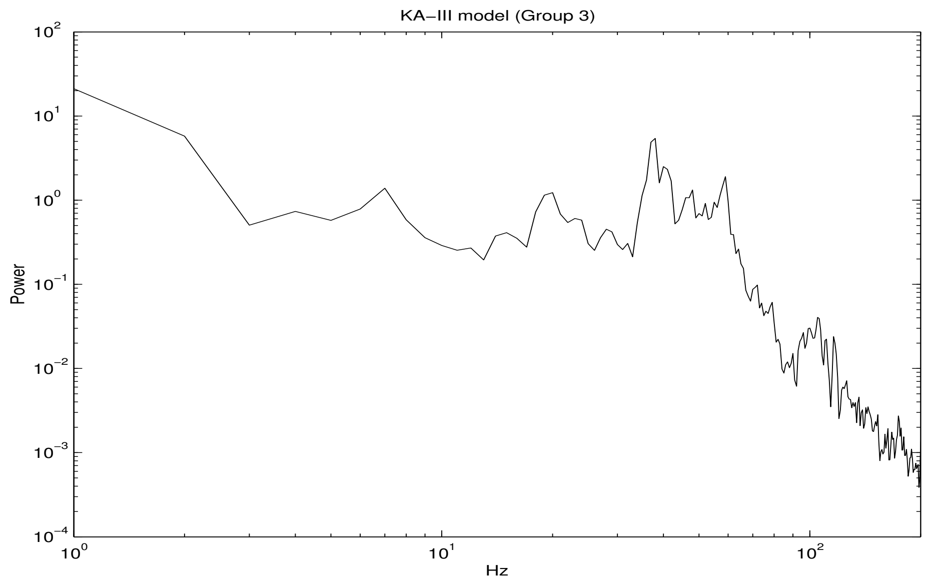


Figure 2.6: The power spectrum of a rat Olfactory Bulb EEG is simulated with the K-III model. The calculated "$1/f^{\alpha}$" slope of the EEG and model is approximately -2.0. Rat OB data from (Kay and Freeman, 1999), KA power spectrum from (Harter, 2003)

> the key desired property of free-roving, semi-autonomous devices ... Intentionality consists of the neurodynamics by which images are created of future states as goals, of command sequences by which to act in pursuit of goals, of predicted changes in sensory input resulting from intended actions by which to evaluate performance, and modification of the device by itself for learning from the consequences of its intended actions.

Intentionality is a result of the endogenous (e.g. internally generated) construction and direction of behavior into the world. We see it in all biological organisms that select their own goals, balance their activities to satisfice multiple, and sometimes conflicting needs, and learn from experience statistical regularities of their environment that are exploitable for survival. Intentional behavior has very much to do with the coordination of all parts of the body, into focused activity. This type of perceptual awareness and coordination is consistent with the concepts of situated and embodied cognition. As noted before, the successful understanding of intentionality is believed by some to be a more fundamental way of understanding cognition as a whole, and upon which more deliberative and logical reasoning skills of humans are built.

The concept of intentional behavior has to do with how the biological organism dynamically organizes and constructs goal states and generates behavior to approach, evaluate and satisfy those goals. In a more traditional autonomous agent view, this boils down to solving the action selection problem, but in a way that does not depend on hard-coding the goals and desires into the organism. Instead through normal developmental progressions, such goal states need to be discovered, constructed, and hierarchically organized. Baars, among others, has postulated a hierarchy of goal contexts that provide a focus for attention and action (Baars, 1988). However, little has been proposed on how such a goal hierarchy comes to develop in an organism. Thelen and Smith's concept of the ontogenetic landscape (Thelen & Smith, 1994) provides the beginnings of a metaphorical representation of how a goal hierarchy develops. They believe that skills are developed by the successive formation and dissolution of attractor dynamics. Development is seen, by them, as the hierarchical organization and construction of the ontogenetic landscape

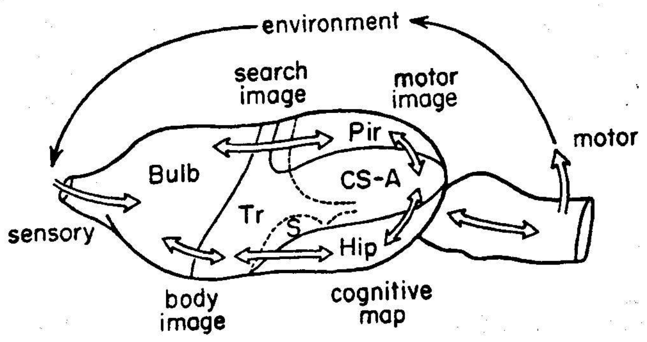


Figure 2.7: A schematic representation of the basic vertebrate limbic system. There are three major divisions, sensory areas, cortex, and motor areas, along with a hippocampus for cognitive maps and other types of long-term memory (from Freeman, 2001)

in the service of the needs and desires of the organism. This formation of behavioral sequence patterns are what are used to integrate and direct intentional behavior.

Brain scientists have known that the minimal nervous system that is capable of supporting the basics of intentional behavior is the limbic system. Phylogenetically, the development of the basic limbic system first appears in amphibians, such as the salamander. This system is comprised of the phylogenetically oldest parts of the forebrain (involved in interoception and goal formation), along with the paleocortex and the deeper lying motor nuclei, as well as some form of a primitive hippocampus. Figure 2.7 shows a schematic illustration of a prototypical vertebrate limbic system.

The model of the basic limbic system presented here provides a starting framework upon which to develop models of the formation of hierarchical goal-state dynamics. This basic architecture, along with principles of self-organization and chaotic neurodynamics, provides a framework for the development of intentional behavior in autonomous agents and a better understanding of such mechanisms in real brains.

Intentionality is a particular *level* of intelligent behavior. The category of intentional behavior can be recognized by such behaviors as: goal directed activity but not overly so; performing opportunistic behavior when such opportunities arise; orientation in the environment directed towards the salient goals; basic memory such as episodic and cognitive maps; and hierarchical balancing of multiple sometimes conflicting goals.

The architecture of the limbic system, with its four functional areas, is a necessary condition for achieving a basic level of intentional behavior. Intentional behavior is simpler than general human level intelligence, but it is a fundamental prerequisite for a system to have true intentionality before general human level intelligence can be achieved.

Some simple long-term memory is required for intentional behavior to occur. Without an episodic memory, systems can be stuck in a loop of repetitive, unproductive behaviors, such as observed with wasps and other insects that do not posses the full architecture needed for intentional behavior. Episodic memory allows the cognitive system to recognize such loops (as boredom) and break out of the unproductive behavior by trying something different.

2.4.3 KIV Model

As described previously, the purpose of the KIII set is to model the aperiodic dynamics observed in the sensory systems of biological brains, and to begin to understand how such dynamics may take part in the formation of meaning for the organism. While perception is an important component in the production of intelligent behavior, it is only a small part of the whole. One insight of the embodiment movement is that studying pieces of the cognitive puzzle (perception, memory, etc.) may in many cases miss important points on how behavior emerges from the pieces working together as a whole in a complete autonomous agent (Freeman, Burke, & Holmes, 2003).

The KIV architecture is a model of what biologists believe may be the simplest neurological structure capable of basic intentional behavior, the limbic system (Kozma et al., 2003; Kozma & Freeman, 2003). The limbic system is composed of four basic functional areas: sensory/perceptual areas, memory and orientation, value system (needs and goals) and motor systems. These four areas can roughly be described as the "What" "Where"

"Why" and "How" functions respectively. Figure 2.8 is a schematic representation of the KIV model of the limbic system.

The hypothesis captured in the KIV model is that the same types of aperiodic dynamics that have been shown to be crucial to the formation of meaning in perceptual systems are also necessary for the formation of memory and motor maps, as well as the hierarchical organization of competing goals. Therefore, at the heart of the sensory, memory and valence system lies a KIII set, which is capable of producing the requisite aperiodic dynamics. We will describe each of the four areas in detail next.

What (Sensory/Perception)

As discussed previously, aperiodic dynamics have been observed in rabbit sensory systems, and are believed to play an important part in the formation of meanings. In the KIV architecture, external signals (exteroception) arrive from the environment (Figure 2.8 top). Each sensory channel is mapped into meanings of interest to the organism, through a process of the formation of chaotic attractors due to learning. In the figure, the OB (olfactory bulb), PC (prepyriform cortex) and AON (anterior olfactory nucleus) are three groups of KII sets that form a sensory KIII. In a full model, each sense would be handled by one or more KIII groups of its own (not shown in figure). The formation and recognition of salient meanings in the environment provide the animal with a sense of "What" important things are in its immediate environment that can or should be dealt with.

Where (Orientation, Memory)

A primitive hippocampus is the center of more long-term memory and orientation functions in simple limbic systems. In the KIV architecture, the formation of cognitive maps of the environment, and the determination of the orientation of the organism in its environment (both locally and globally) is taken care of in the hippocampus. The orientation

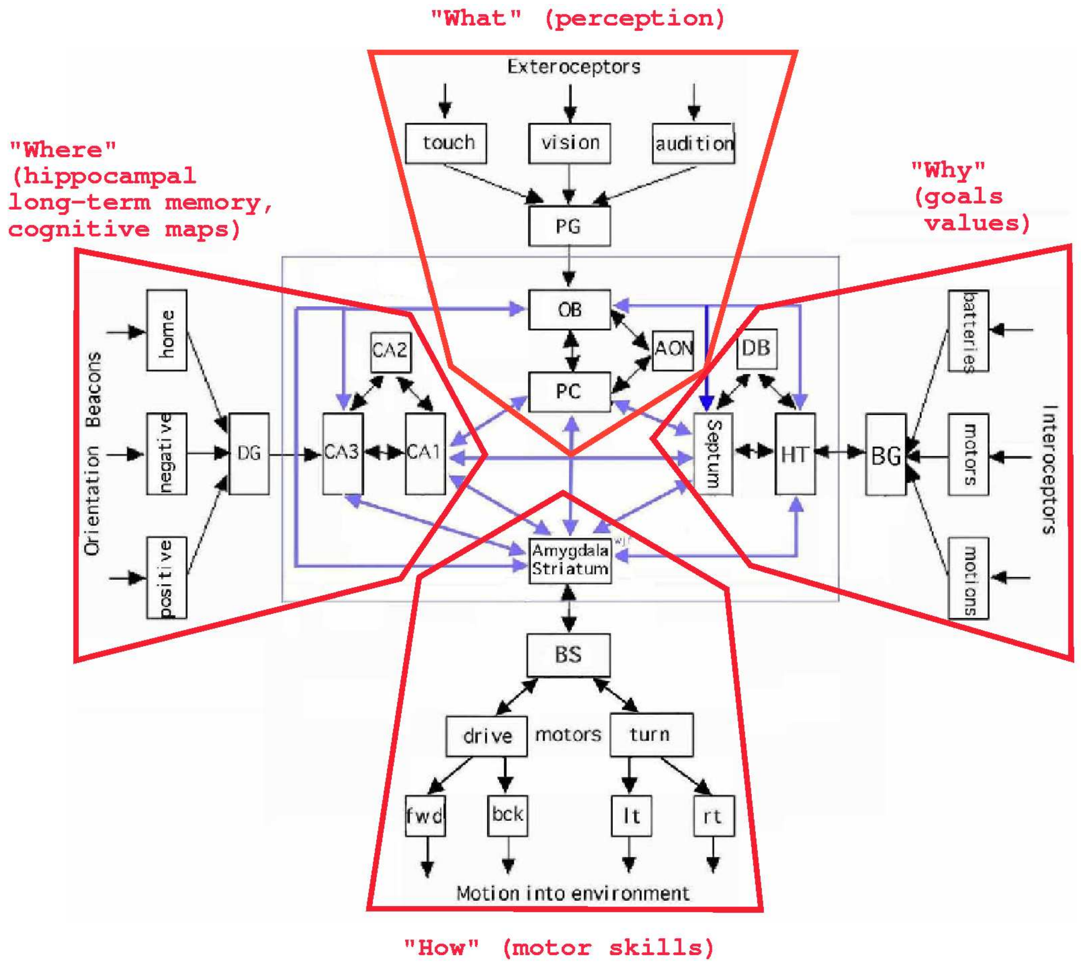


Figure 2.8: The KIV architecture, an embodied (biologically inspired) model that is capable of the hypothesized necessary conditions of aperiodic computational neurodynamics. See Table 2.1 for a description of the brain area labels. Based on figure from (Kozma, 2003)

function of the hippocampus is depicted in the KIV model (Figure 2.8 left) as receiving orientation signals from the environment. The three CA regions (CA1, CA2 and CA3) form a KIII set that is responsible for the formation of memories of the environment of the organism. The formation of cognitive maps, and so called place cells, in the hippocampus, is performed in the CA KIII set, and takes advantage of the flexibility of aperiodic dynamics to form such representations.

Why (Goals, Drives, Value Systems)

Figure 2.8 on the right composes the value system of the KIV architecture, and mediates the production of behavior to guide the organism in completing goals and tasks to satisfy its needs. This system keeps track of the reasons "Why" the organism is doing what it is doing. In the valence system, internal signals that monitor basic needs (such as food, or avoiding damage) are fed into the system (Figure 2.8 right). Another KII forms the heart of the system for forming and balancing a goal landscape of the organism (HT, DB and septum).

How (Motor Actions)

The motor system (Figure 2.8 bottom) is responsible for directing actual effectors for "How" the organism will carry out behaviors in pursuit of its goals. The amygdala provides the goal-oriented direction for the motor system, that is superimposed on local tactile and other protective reflexes.

2.4.4 Learning Mechanisms

Four types of learning are used in the KIII components to form perceptual categories and other dynamics (Kozma & Freeman, 2001a). These types of learning include Hebbian type reinforcement, among others. Though not indicated in our necessary and sufficient conditions, these mechanisms are important pieces in the puzzle in how such complex aperiodic dynamics come to represent meanings for a biological organism. The four types of learning are:

- Continual short-term learning of environmental cues at high rates during exploration.
- Intermediate-term habituation to ambiguous, irrelevant, noisy inputs.
- Long-term reinforcement learning at critical situations (aversive or appetitive) using Hebbian mechanisms.
- Renormalization as needed to maintain homeostatic stability in all units.

Continual short-term learning provides snapshots of actual sensory states at any given time instant. It provides the information for not only snapshots of actual sensory states, but also sequences of learning such present states and the recent past. This represents the short-term memory function of the organism, and provides a buffer of events that have recently occurred. These events can be passed to more permanent storage when salient events occur to the organism or in the environment to trigger such processes.

Habituation is a diminished response to sensory input. If a sensory signal is repeatedly encountered but it is never associated with any meaningful event to the organism, it gradually becomes detuned by the short-term sensors and essentially ignored. This is the process by which we, for example, tune out such things as environmental noise like the air conditioner.

Hebbian learning is the process by which salient perceptual events become meaningfully associated with internal goals of the valence system. This type of learning is episodic, not continuous, and it produces more long-term effects. Hebbian learning is a well known mechanism that involves the strengthening of associations between units that tend to become active together. Through this type of reinforcement, environmental cues, such as a ringing bell or a particular smell, can come to be associated with salient events, such as the appearance of a food reward.

The above learning methods implement incremental changes in the KIII components. As KIIIs are highly nonlinear units, learning effects may result in undesirable destabilization of the dynamics. Homeostatic regulation is a way of achieving stability by maintaining local balances using renormalization.

Chapter 3

Statement of Research Objectives

In our review we covered the main research thrusts that have contributed to our current understanding of intelligence and cognition in terms of complex dynamical systems. The study of complex systems has given us many new tools in understanding the self-organizing properties of non-equilibrium thermodynamic systems. It seems clear, at least to many researchers, that the brain is an example of such a complex system. If this view is correct then many of the properties that we describe as intelligence, intentionality and cognition may be the result of such self-organizing properties. Therefore, a better understanding of intelligent behavior can be obtained by gaining a better understanding of these self-organizing principles by which brains operate.

This basic premise on the nature of brains as complex systems has lead to many new approaches and insights on the mechanisms of intelligence. We have some ideas of how action selection mechanisms might be developed, such as Brooks' subsumption architecture, that use emergent mechanisms to explain intelligent behavior. There remains a large gap, however, in our understanding of these processes in terms of neural dynamics. In particular, we don't yet have a strong grasp of how these competitive and cooperative processes operate among neuronal groups to organize patterns of behavior. This is especially true in light of some of the findings that aperiodic dynamics play a very important role in the flexible and robust performance of biological perceptual systems.

The objective of this research is to explore the following ideas. Current work in third generation dynamical connectionist and autonomous agent research (Matarić, 1991; Edelman et al., 1992; Verschure, Kröse, & Pfeifer, 1992; Matarić, 1995; Verschure, Wray, Sporns, Tononi, & Edelman, 1995; Almássy, Edelman, & Sporns, 1998; Sporns, Almássy, & Edelman, 1999; Verschure & Voegtlin, 1999) has begun to emphasize the importance of exploring whole (complete) systems that are embodied and embedded in an environment (Franklin, 1995; Steels & Brooks, 1995; Pfeifer & Scheier, 1998). Such agents are capable of exhibiting many interesting properties up to now only seen in biological development, such as the self-organization of behavior, self-sufficiency, embodiment and adaptivity and action-oriented representations (Hendriks-Jansen, 1996; Clark, 1997; Pfeifer & Scheier, 1998). This research extends such results by exploring the roles of aperiodic dynamics in the formation of perceptual and behavioral patterns and how such patterns may be organized into a basic intentional system (Freeman, 1999b, 1999a, 2000).

The idea of the brain not as a static manipulator of symbols but as fundamentally a system of self-organizing, pattern-forming dynamic change over time is a relatively recent one in Cognitive Science (Thelen & Smith, 1994; Kelso, 1995; Port & van Gelder, 1995). Within this framework of viewing patterns of brain activity as a dynamical system, there naturally arises the question of which types of patterns are exhibited by and important to brain function. Both point and limit cycle attractor dynamics have been used in dynamic models of cognition (Abraham, Abraham, Shaw, & Garfinkel, 1990; Port & van Gelder, 1995). However, the question naturally arises of weather or not chaotic dynamics plays an important role in the organization of behavior in biological brains. One of the objectives of this research is the development of a simplified model of the K-set equations. Our simplification is a discrete model of the type used in standard artificial neural network (ANN) research. Because of the similarity of the simplified K-set model and ANN equations, we are better able to compare the two bodies of research. The discrete equations of the simplified model are more mathematically tractable, which makes them easier to understand in terms of their basic properties.

The work of Freeman and others (Skarda & Freeman, 1987; Tsuda & Yamaguchi, 1998; Kozma & Freeman, 1999; Freeman, 1999b; Freeman & Kozma, 2000; Freeman et al., 2000; Kozma & Freeman, 2000, 2001a; Kozma, Harter, & Franklin, 2001; Tsuda, 2001) has shown that chaotic dynamics do occur in the perceptual categorization of stimuli in biological brains. Further this work has speculated on the possible roles that chaos may play in developing patterns of behavior. Far from being viewed as inconvenient noise, or an epiphenomenon of brain functioning, chaos has been proposed as a necessary feature of mesoscopic organization for the development of complex adaptive behavior.

The primary role that chaos may play, according to Skarda and Freeman, is as a controlled source of noise that allows for continual access to previously learned sensory patterns (Skarda & Freeman, 1987). This type of continual and rapid access provides a source of great flexibility in the adaptive behavior of biological organisms. If true, the absence of chaotic dynamics may explain in part some of the lack of flexibility displayed by previous symbolic and connectionist models of memory and behavior. Chaotic dynamics provides the ability for rapid convergence into a perceptual attractor. Just as the chaotic flow of people in an airport can instantaneously change at the announcement of gate changes, so can the patterns of chaotic activity instantly and flexibly change in response to external and internal influences on the system. Chaotic dynamics may provide that balance, so important to biological organisms, between rigidity and disorder. Intentional behavior may live near a kind of phase transition. Just as water exists in three phases: solid, liquid and gaseous, so might behavior patterns have a similar phase transition. Chaotic dynamics allows the patterns of behavior to live in a liquid state that can rapidly converge to an ordered regime under the influence of environmental and internal control parameters (Kauffman, 1993, 1995, 2000).

The other main thrust of this research proposal is along the lines of what Freeman calls the basic intentional system (Freeman, 1999b, 1999a, 2000). Intentional behavior is something that all higher biological organisms exhibit but which most models of behavior have so far been unable to capture. Theories of cognitive embodiment (Hendriks-Jansen, 1996; Clark, 1997) have begun to tackle this issue and explore models that may capture intentionality. Intentionality is a very basic property of biological behavior, so simple that its importance is easily overlooked. Traditional AI has concentrated on the impressive reasoning and linguistic abilities of humans. Early on it was felt that these higher level cognitive abilities were the important problems. If they could be solved then it was thought that the easy problems such as perception, navigation, orientation and spatial representation would quickly be overcome (Hendriks-Jansen, 1996; Clark, 2001). However the difficulty of getting classical AI models to display truly flexible behavior has lead some to completely reverse this traditional viewpoint. Now it is thought by many that solving the seemingly basic problems of embodiment in the world may provide the foundation upon which higher level reasoning skills need to be built. Only when the foundation is correctly built will the characteristic flexibility and complexity of behavior in logic, reasoning and human intuition for problem solving be possible.

Freeman argues that this embodiment of behavior in biological organisms arises from a certain neurodynamical configuration of vertebrate brains, which he calls the basic intentional system (Freeman, 2000). He identifies the major parts of the limbic system as the basis for intentional behavior in animal brains. Intentional behavior is that ability of organisms to orient themselves in time and space. To form spatio-temporal maps of themselves and their environments. To learn and develop from experiences within the environment. Further it is the ability to direct behavior into the environment in the service of the needs and desires of the organism, e.g. to be goal-oriented and goal-directed. Freeman argues that the basic intentional system forms the basis upon which all other complex behaviors develop.

Models of intentionality basically have to do with how the whole or complete organism dynamically organizes and constructs goal states and generates behavior to approach, evaluate and satisfy those goals. In a more traditional autonomous agent view, this boils down to solving the action selection problem, but in a way that does not depend on hard-coding the goals and desires into the organism. Thelen and Smith's concept of the ontogenetic landscape (Thelen & Smith, 1994) provides a metaphorical analysis of the way in which a landscape of attractors is hierarchically organized and constructed in the service of the needs and desires of the organism. Freeman's principles of chaotic neurodynamics provide us with the guideposts for building systems capable of the formation of such attractor landscapes. These concepts provide us with the key to building hierarchical, self-organizing, goal context mechanisms for autonomous agents.

In Skarda and Freeman (Skarda & Freeman, 1987), the authors hypothesize that

> ... convergence to an attractor in one system (e.g. the olfactory bulb) in turn destabilizes other systems (e.g. the motor system), leading to further state changes and ultimately to manipulation of and action within the environment.

In other words, the dynamics in some systems may act as control parameters that lead to bifurcations in other systems to produce behavior. Further, a basic property of the architecture of the limbic system is that activity does not simply flow in one direction (from input to output) but recurrent connections allow for the mutual coupling of such systems. In effect the perceptual system does not simply influence the motor system in producing goal-directed behavior, but the motor system also influences the perceptual system in setting up and guiding expectations and attention. The perceptual categories formed by the organism are not simple, static structures, but are dynamic patterns of activity. Through such influences, both internal and external, behavior is generated. So static perceptual categories give way to dynamic patterns of activity, both from internal needs and external perceptions, that guide action. Thus, such a system may be capable of generating Gibson's "affordances" (Gibson, 1979), in which perceptual information and internal drives are seamlessly combined to identify and afford opportunities for adaptive behavior to the organism.

An objective of this research is to begin to build a model of a basic intentional system. The physical architecture of such a model will be based upon the only know example of a true intentional system: the biological limbic system. The principles of chaotic neurodynamics will provide the guideposts for the construction of such a system. The goal of the research is to discover how exactly chaotic dynamics may be used to not only form perceptual categories, but to also form hierarchical, intentional goal states. Also, how through a process of ontogenetic development, such goal states can be created and utilized to generate behavior, attention and expectation in the organism. In essence, can we begin to model some of the properties of intentional behavior and affordances through chaotic dynamics and self-organization of patterns of activity.

The objective of this research, therefore, is to try and fill in some of the missing gaps in our knowledge of how neuronal group dynamics might support higher-level cognitive mechanisms. In particular, we need a better understanding of how neuronal groups produce aperiodic dynamics and how they are used in not only perceptual processes but other types of cognitive mechanisms as well. Towards this end, we will show our development of a simplification of the K-set neuronal population model. This simplification has many advantages for use in autonomous agent modeling of neuronal group dynamics. Because of the efficiency of the simplified model, we are now able to construct and examine much more complicated neuronal group architectures, and explore how they work using autonomous robotic agents. We will develop the simplified model and then demonstrate the uses that aperiodic dynamics may be put to in order to develop cognitive mechanisms in autonomous agents.

3.1 Research Questions and Goals

The following research has these central research questions in mind:

- Can chaotic dynamics be used to form perceptual categories in autonomous agents?
- Can the formation of perceptual attractors be extended to form attractors for memory systems and cognitive maps?
- Do chaotic dynamics improve the performance and flexibility of perceptual categorization and behavior generation for autonomous adaptive agents?
- How, exactly, can we go from perceptual models using the principles of chaotic neurodynamics, to a more complete model of a basic intentional system?

These questions lead to some of the following goals of the research:

- The simplification of the K-set equations into discrete difference equations. This simplification allows for more complex autonomous agent models of neuronal group dynamics to be constructed and explored.
- The demonstration of chaotic dynamics in forming perceptual categories in an autonomous agent using the KA simplification.
- The demonstration of the self-organization of behavior in the action selection of an autonomous agent.
- The demonstration of the development of spatio-temporal cognitive maps using chaotic dynamics in autonomous agents.

Chapter 4

Research Results

4.1 KA: Engineering Simplification of K-Set Neural Population Model

Freeman's K-sets (Freeman, 1975; Skarda & Freeman, 1987; Freeman, 1991) are a neural population model capable of replicating the nonlinear dynamics observed in biological perceptual systems. The K-set is a model of the dynamics of a neuronal population and as such describes how the average population current density changes in response to external stimulation, internal arousal and other factors. The K-set neural population model is described by a second order differential equation. A series of such equations can be used to model a given neural architecture (e.g. the olfactory system), and the dynamics resulting from the interaction of such populations. The form and parameters of these equations were determined by observing the effects of stimulation and other experiments on real biological neural populations prepared through brain slicing techniques.

In addition to being a model of neural population dynamics, the K-set hierarchy is also a description of the neural architectures needed for the production of complex, aperiodic dynamics in biological neural tissue. By modeling increasingly higher levels of neural structure, the K-sets help us understand the production of intelligent behavior from the simple components of nonlinear feedback mechanisms, to the self-organization of perceptual and behavioral mechanisms via aperiodic, nonlinear dynamics. The K-set hierarchy of models are therefore multi-layer, recurrent neural models that both capture the architecture and dynamics of brain structures as well as attempt to bridge our understanding of how such dynamics support higher-level cognitive mechanisms.

The K-sets model the aperiodic dynamics observed in cortical sensory systems and begin to explain how such dynamics contribute to the recognition and learning of sensory patterns in biological brains. Recent work in aperiodic dynamics in cortical systems (Shimoide et al., 1993; Freeman, 1997, 1999b, 2000; Harter, 2001; Tsuda, 2001; Kozma et al., 2003) have begun to move beyond sensory systems to look at how such dynamics may also help us better understand the production of intelligent behavior in biological agents.

The motivation behind the KA model is to develop a simplification of the original K-sets that is still capable of performing the essential dynamics, but is simpler and faster and therefore more suitable for use in large-scale simulations of more complete autonomous agent architectures. The KA is to be used in developing autonomous agents that take advantage of aperiodic dynamics for perception, memory and action.

The K-set provides a starting point for the basic units to be used in this research. However, in order to use the K-sets to simulate an even moderately sized model architecture requires the solution of a large number of lumped simultaneous differential equations. This process can be very time consuming even for a relatively small population or a few seconds of simulated activity. This cost is prohibitively expensive for use in a real-time autonomous agent.

A simple solution is to use a discrete difference equation, of the type used in standard ANNs. However, standard ANN methods do not inherently display more complex dynamics and are specifically designed to converge to a point attractor. In fact oscillatory behavior is often taken as a sign of divergence, and is usually considered an indication of failure in standard approaches.

The simplification of the continuous, ordinary differential equations of the original model to a discrete difference equation has many additional advantages. The discrete-time difference equations are more mathematically tractable, and therefore more easily analyzed in terms of their properties and constraints. The discrete-time difference equations developed for the KA model are of the same class used in artificial neural network (ANN) research, and therefore the results of the KA are more directly comparable to that body of research. The efficiency gains of the discrete KA model should also not be underestimated. We will show that the KA is 3 to 5 times faster in general when simulating models in comparison to the original K-set equations.

In this section we describe the discrete, deterministic engineering simplification of the K-sets for adaptive agents named KA-sets. Although the KA-sets are not an exact discretization of the original K-set equations, they have been designed to emulate the important dynamical behavior of neuronal populations. In this section I will describe the internal mechanisms of the KA model, along with many experiments using the KA models to demonstrate the principles of chaotic neurodynamics, their behavior under various conditions and their capability in replicating the important findings of the original K-set models (Harter & Kozma, 2001a, 2002, 2004d).

4.1.1 KA Model Description

The purpose of the model presented here is to provide elementary units capable of the complex mesoscopic dynamics observed in the brains of biological organisms. These units model the dynamics of populations of neurons, rather than a single neuron. The KA units presented here are also designed to be computationally efficient, so that they may be used to build real-time control architectures for autonomous agents, as well as being mathematically more tractable.

At its heart the KA model uses a discrete time difference equation to replicate the dynamics of the original second order ordinary differential equations of the K-sets. A unit in the KA model simulates the dynamics of a neuronal population. Each KA unit simulates an activity level, which represents an average population activity density. The basic form of the difference equation can be given simply as shown in Equation 4.1, which states that the current at time step t is a function of the current in the two previous time steps, as well as the external influence from the net input of units connected to the simulated unit.

$$a_i(t) = F(a_i(t-1), a_i(t-2), net_i(t-1)) \tag{4.1}$$

The evolution equation of the KA unit can be described by three components that are combined to compute the simulated current at time t from the current and the rate of change of the current at time $t-1$ and $t-2$. These three influences on the simulated current are 1) a tendency to decay back to the baseline steady state $dec_i(t)$; 2) a tendency to maintain the momentum of the current in a particular direction $mom_i(t)$; and 3) the influences of external excitation or inhibition as input to the unit $net_i(t)$.

When isolated neural populations are externally stimulated away from their baseline steady state, once the external stimulation is removed the population experiences an exponential decay back to the baseline. In the KA model the tendency to return to the baseline steady state is modeled by a decay term. The resting, or baseline state of the current in these model is defined as an activity level of 0. The effect of decay is described as:

$$dec_i(t) = -a_i(t) \times \alpha \tag{4.2}$$

Here α is a parameter that indicates the rate of decay. Since the difference is proportional to the current, the effect is to cause the decay to be rapid when the activity of the unit is far from the baseline, while the rate of decay slows down as the activity approaches the steady state.

Neural populations exhibit a certain amount of momentum in the dynamics of their activity over time. In essence, once a population's current begins to move in a certain direction (positive or negative) it tends to keep moving in that direction even for some time after any influence pushing it has been removed. In the original K-set models, this was observed when stimulating a isolated brain-slice population. After stimulation ceased the population rapidly returned to its resting level. However in the process of decaying back to the baseline it will undershoot and actually go below the baseline steady state for some time before returning to equilibrium. This slight oscillation in the neural populations is what necessitates the used of the second order term of the differential equations, as only second order equations are capable of such oscillatory behavior. The momentum term is needed in the KA difference equation in order to capture this dynamic behavior of the population. Similarly, the momentum term is also second order, as it relies on two previous time steps in order to calculate its influence.

To simulate the momentum of a units activity, we need to use a function of the previous two time steps. This is necessary so that we can simulate a momentum based on the rate of change of the activity of the unit, as well as the other reasons mentioned above. We first define the rate of change of the activity at time t, $r_i(t)$. This is the difference of the activity of the unit at time t from the activity at a previous time step $t-1$. The rate of change at time t is thus:

$$r_i(t) = a_i(t) - a_i(t-1) \tag{4.3}$$

With the rate at time t defined, we can describe the momentum as shown below:

$$mom_i(t) = r_i(t) \times \beta \tag{4.4}$$

Where β is a parameter that controls how much of an influence the momentum has on the dynamics of the model. β can be thought of as a percentage which indicates what

Table 4.1: KA Model Variables

Variable	Description
$a_i(t)$	Simulated activity of i^{th} population at time t
$dec_i(t)$	Difference at time t due to decay to baseline
$mom_i(t)$	Difference at time t due to momentum
$r_i(t)$	Rate of change of the activity at time t
$o_i(t)$	Transfer function of the activity of the i^{th} unit at time t
$net_i(t)$	Difference at time t due to external net input

portion of the momentum at the present time step should continue into the next time step.

The effect of the net input at time t is the same as in the K-model and is shown in Equation 4.5. This is the standard summation of the activity of the input units through a transfer function multiplied by the connection strength. The output or transfer function $o_j(t)$ of a KA unit is a function of the activity of the unit. The KA model uses the same asymmetric sigmoid transfer function and summation mechanism of the original K-sets. The transfer function is shown in Equation 4.6. The ϵ parameter is a scaling factor that indicates the level of arousal of the KA unit. Arousal in biological organisms is a function of history and experience, and can vary with things like surprise and familiarity with the current situation.

$$net_i(t) = \sum_j w_{ij} o_j(t) \tag{4.5}$$

$$o_j(t) = \epsilon\{1 - exp[\frac{-(e^{a_j(t)} - 1)}{\epsilon}]\} \tag{4.6}$$

Table 4.2: Comparison of Features and Equations of KA and K-Set Neural Population Models

KA	K
Discrete	Continuous
2^{nd} order difference equation	2^{nd} order ordinary differential equation
$a_i(t) = dec_i(t-1) + mom_i(t-1) + net_i(t-1)$	$\alpha\beta\frac{d^2 a_i(t)}{dt^2} + (\alpha+\beta)\frac{da_i(t)}{dt} + a_i(t) = net_i(t)$
$dec_i(t) = -a_i(t) \times \alpha$	
$mom_i(t) = r_i(t) \times \beta$	
$net_i(t) = \sum_j w_{ij} o_j(t)$	$net_i(t) = \sum_j w_{ij} o_j(t)$
$o_j(t) = \epsilon\{1 - exp[\frac{-(e^{a_j(t)}-1)}{\epsilon}]\}$	$o_j(t) = \epsilon\{1 - exp[\frac{-(e^{a_j(t)}-1)}{\epsilon}]\}$
9 sec.	32 sec.

The sum of the influences in Equations 4.2, 4.4 and 4.5 represent the total influence that will be applied to the activity of the unit in the next time step.

$$a_i(t) = dec_i(t-1) + mom_i(t-1) + net_i(t-1) \tag{4.7}$$

In other words, activity of a KA unit is a function of the decay and momentum terms along with the influence from the net input of external units. We sum the values from these three influences to determine the new activity of a KA unit. In Table 4.1 we provide a summary of all of the variables used in the KA model.

In Table 4.2 we show a comparison of the features and equations of the K and KA models. The KA model is a discrete, 2^{nd} order difference equation, as opposed to the original continuous 2^{nd} order ordinary differential equation of the K-sets. Both the K and KA models use time constants as parameters (α and β) in order to tune the dynamics of the models to those observed from real neural populations. It should be noted, however, that these time constant parameters are different between the two models and will take on different values in order to achieve the same dynamics. We will discuss how we determined the α and β parameters for the KA model in the next section. Both of the models use the same net input and asymmetric sigmoidal transfer function to describe the influence of activation passed between the population units. The final item of this table shows the total time needed to run a simulation of a K/KA III that contained a total of 513 units and over 10,000 connections. The simulation was of 10 seconds of activity in the neural model and both were coded and executed using Matlab 6.5 on a 1.0 GHz Pentium class computer. The KA model executes the simulation in just under 10 seconds, while the K model takes over three times as long to run the same simulation. We will discuss more results of this type comparing the efficiency of the two models in coming sections.

4.2 Determination of Momentum and Decay Time Constants in the KA Model

In Figure 4.1 we show an example of how the α and β time constants were derived for the original K-model. In essence the brains of animals were subjected to deep anesthesia, which inhibits the interaction between neural populations. In effect this allowed the dynamics of populations to be studied in isolation. An isolated population was then subjected to varying intensities and time variations of stimulation and inhibition. The activity level dynamics of the isolated population were recorded in response to these external impulses. α and β parameters were then chosen to best fit the ODE dynamics to the observed dynamics of the biological populations.

We use a similar method to determine the parameters of the KA model that allow it to closely approximate the original K model dynamics. Keep in mind, however, that the α and β parameters of the two models represent different time constants, and as such will be set at different values in the two models. We take as our target the dynamics of a K

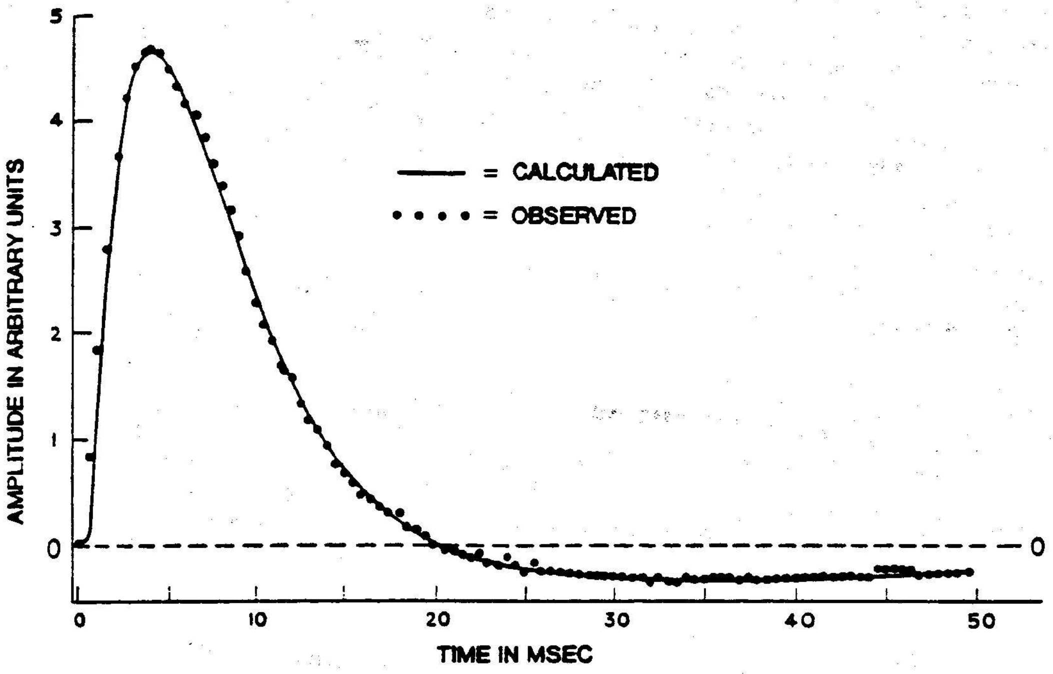


Figure 4.1: Open loop impulse response of a real neural population in response to external stimulation (an impulse). The dots show the impulse response to the external excitation observed in the pyriform cortex under deep anesthesia. The fitted curve is generated by a sum of exponential terms. Many such examples like this were used to determine the α and β time constants of the original K-set model (from Freeman & Shimoide, 1994, p. 122).

model unit, and subject it to varying intensities of external stimulation and inhibition, for varying lengths of time. We then find the decay, momentum and other parameters that allow the KA responses to best approximate the original K model, using least squared fit to measure the difference.

Therefore we subjected a K unit to levels of stimulation ranging from -0.49 to 0.5 in 0.01 increments (intensity = [-0.49:0.01:0.5]). We also varied the time each stimulation level was applied to the K unit from 1 to 50 ms in 1 ms increments (time duration = [1:1:50]). We ran the simulation of the K unit for 500ms in Matlab 6.5, and captured its response to the 5000 different combinations of intensity and time durations. These 5000 time series represented the target dynamics we tuned the KA model to replicate.

With the 5000 samples of the K unit dynamics, we then systematically explored the parameter space of the α decay and β momentum parameters of the KA model, to determine which combination of α and β provided a best fit in the least squared sense to the 5000 simulated samples. In Figure 4.2 and 4.3 we show example portions of this parameter search. We exhaustively searched the α and β parameter space to find a combination that replicate the dynamics of these 5000 samples of the K0 unit by a KA0. We applied the same 5000 combinations of the intensity and time duration of stimulation to a KA unit for the various α and β values. Through a systematic search we could reduce the difference in the dynamics to an arbitrarily small amount. We used a hill-climbing algorithm in order to zero-in on the exact values for the parameters that provided very good approximations of the original dynamics. We found that a decay rate of $\alpha = 0.1505$ and a momentum of $\beta = 0.0985$ produced a good fit of the KA to the K model.

The parameter space defined by the momentum and decay parameters ends up forming a smooth function in the KA model, with only 1 global minima. This makes it easy to find the appropriate parameters to fit the KA single unit dynamics to the original K0 unit. For example, in Figure 4.4 we show a part of the decay and momentum parameter space of the KA model. Here we plot decay along the X axis from values ranging from 0.1

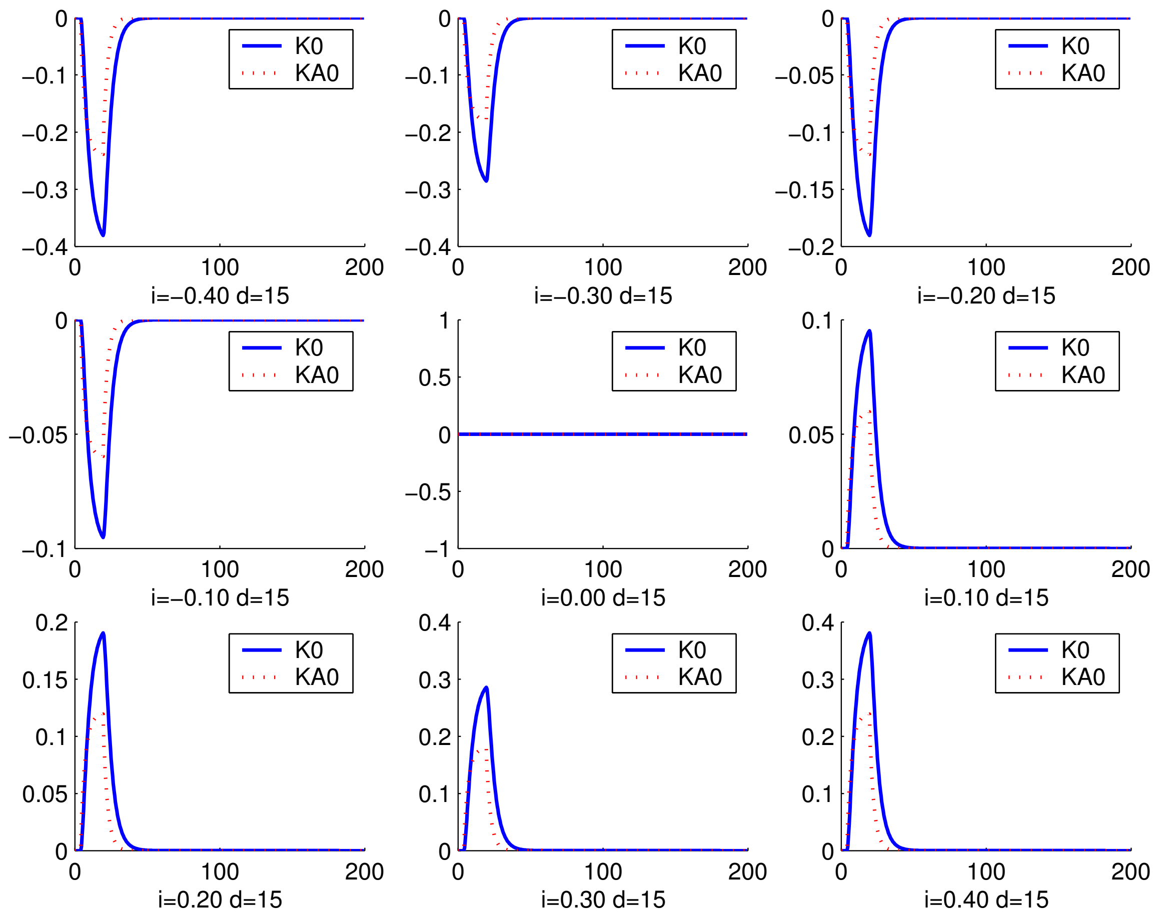


Figure 4.2: First example of fitting the decay and momentum time constants of the KA model to the original K-Set dynamics. This figure shows the fit for decay = 0.25 and momentum = 0.6. We show 9 example time series where the intensity of stimulation is varying from -0.4 to 0.4. In all cases the duration of the stimulation is the same at 15ms. These particular decay and momentum parameters are not very good fits for the original K0.

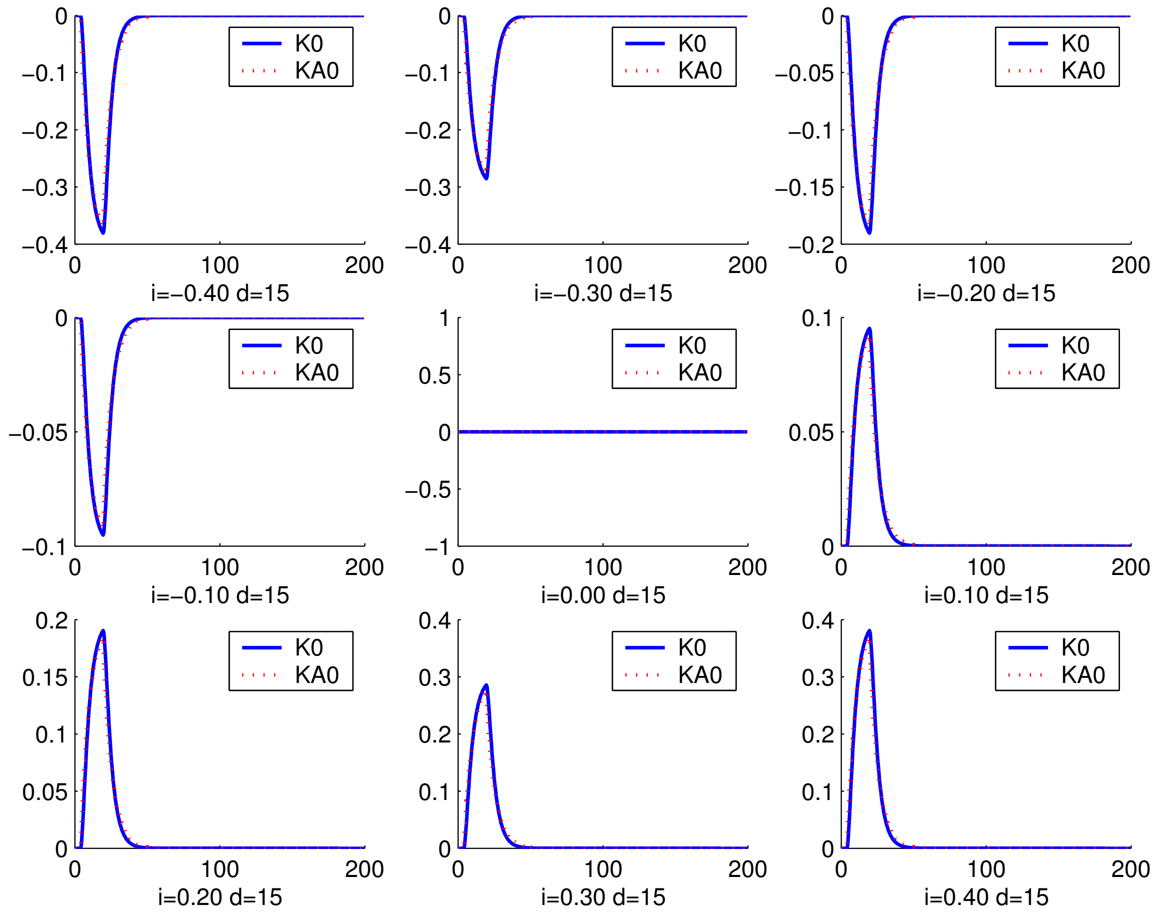


Figure 4.3: Second example of fitting the decay and momentum time constants of the KA model to the original K-Set dynamics. In this figure decay is 0.25 and momentum is 0.1, close to the global minimum. Again we show 9 example time series where the intensity of stimulation is varied from -0.4 to 0.4 and duration of stimulation is held at 15ms. These parameter show a good fit to the original K0.

Table 4.3: KA Model Parameters

Parameter	Description	Default
α	Rate of decay to baseline	0.1505
β	Rate of momentum	0.0985
ϵ	Transfer function arousal level	5.0

to 0.2, and momentum is plotted on the Y axis from 0 to 0.5. Color is used to indicate the error in the fit at each point in the α/β parameter space. We can see visually that the space is smooth, and there is a global minima in the error somewhere in the area of $\alpha = 0.15$ $\beta = 0.1$.

Table 4.3 summarizes the parameters of the KA model discovered by the parameter fitting process and used for the simulations and experiments described in the rest of this work. The arousal level parameter ϵ is only significant when we have networks of units connected together, it does not affect the dynamics of a single unit in isolation. Since we are using the same asymmetric sigmoidal transfer function in both the K and KA models, we have used a standard arousal level of 5.0 in the experiments described next. Future work is needed to explore the uses of the arousal level in models of cognition, and its possible relation to more global and slow-changing dynamics such as neuro-chemical processes that affect the dynamics of populations in brains.

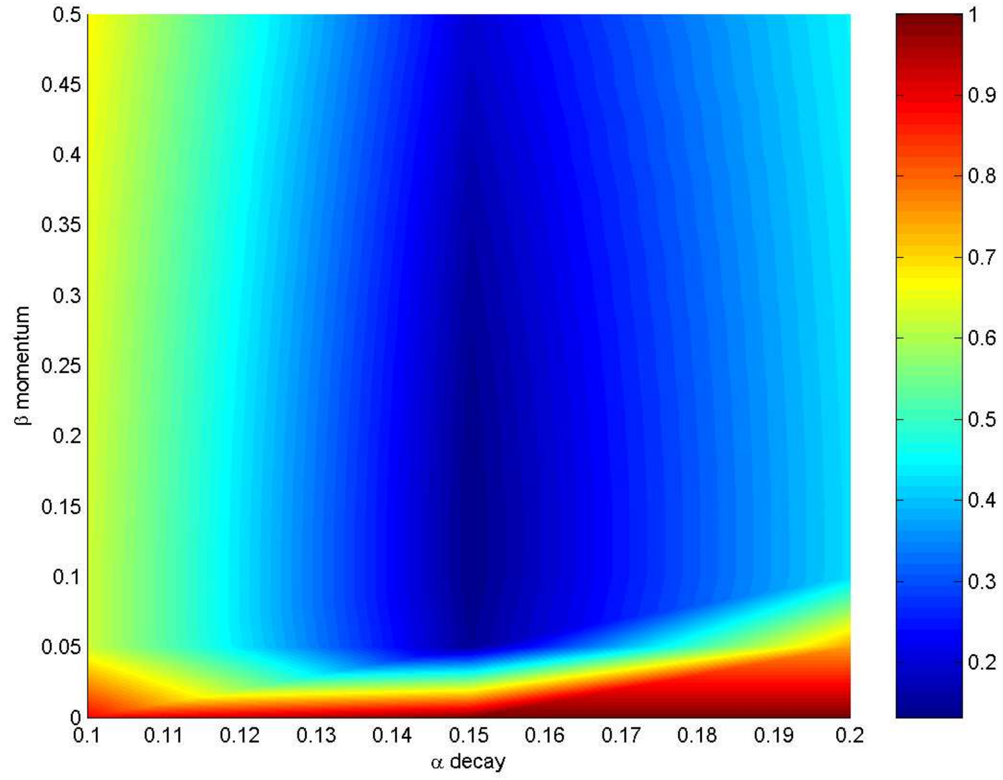


Figure 4.4: A portion of the α / β parameter space of the KA model. Decay (α), plotted along the X axis, varies here from 0.1 to 0.2. Momentum (β), plotted along the Y axis, varies from 0.0 to 0.5. Color indicates the calculated error (using sum squared difference) between the dynamics of the KA0 and K0 unit over the 5000 sample time series. A global minimum is present in the area of $\alpha = 0.15$ $\beta = 0.1$

4.3 Comparison of KA model with Standard Artificial Neural Networks (ANNs)

The generic form of the difference equation used in standard ANN models can be stated as:

$$a_i(t) = F(a_i(t-1), net_i(t-1)) \tag{4.8}$$

Here the activation of a unit i in a ANN model is a function of the activation of the unit in the previous time step along with the net influence of input from other units in the previous time step. In the vast majority of ANN models, however, the influence of the activity of the unit by the units activity in a previous time step is ignored, and thus the normal usage simplifies to:

$$a_i(t) = net_i(t-1) \tag{4.9}$$

In other words the activity of a unit in the next time step depends solely on the net input to the unit from externally connected units. This is a reasonable simplification in strictly feed-forward networks, since there is only a single time step being simulated. On the introduction of the input to the first layer, the activity simply flows forward in one direction in the network. However, this simplification becomes less useful in the realm of recurrently connected networks, where the dynamics of a unit over time can be simulated, and such dynamics may effect the performance of the network. Most research in recurrent ANNs still only use the simplified equation. This means that even in recurrent ANN research, the dynamics of the units depends solely on the activity of connected units in the previous time step. The units do not have nor use any intrinsic dynamics of their own.

The KA model is a simplification of the K-sets. One of the purposes of both models is to captures the dynamics of an isolated neural population in response to external stimulation. As such, both the K and KA models have intrinsic dynamics associated with a neuronal unit, such that in the absence of external stimulation they will continue to modify their activity levels as a function of the passage of time. This can of course be seen most clearly in the KA difference equations, which include terms that depend on the previous activity of a unit in determining the activity of the next time step, which differentiates these models from the vast majority of ANN modeling. Further, both the K and KA models depend on a second order term in order to correctly replicate the dynamics of neural populations. The second order term is necessary in the K-model ODE in order to capture the damped oscillations of neural populations. Similarly, in the KA model, the momentum parameter depends on two previous time steps of the activation of the unit in order to capture this type of behavior.

Another difference between the K and KA models on one side, and ANN models on the other is, of course, the form of the transfer function. In all cases, the nonlinearity of the transfer function is an important feature in capturing the nonlinear nature of neural functioning. ANN research uses many different transfer functions, though the most popular is the standard sigmoidal transfer function used in models using real activation values:

$$o_i(t) = \frac{1}{1 + e^{-a_i(t)}} \tag{4.10}$$

However, the K and KA models use a particular asymmetric sigmoidal transfer function (Equation 4.6), that has a firmer basis in biological networks. The asymmetric transfer function used was derived by Freeman and associates by studying the nonlinear passing of activation between biological neural populations (Freeman & Shimoide, 1994). The asymmetry is an important property in the transfer function as it means that excitatory input causes a destabilization of the dynamics of networks. This destabilization is essential in the collapse of aperiodic attractors observed in biological perceptual systems.

In Table 4.4 we summarize some this comparison of the KA and ANN models. Both are models that abstract some aspect of neural functioning to describe the dynamics of neural populations. Both use discrete difference equations to describe the activity of units and how it changes over time in response to connections with other units. The vast

Table 4.4: Comparison of Features and Equations of KA and ANN Models

KA	ANN
Discrete difference equation	Discrete difference equation
$a_i(t) = F(a_i(t-1), a_i(t-2), net_i(t-1)$	$a_i(t) = F(net_i(t-1))$
$net_i(t) = \sum_j w_{ij} o_j(t)$	$net_i(t) = \sum_j w_{ij} o_j(t)$
$o_j(t) = \epsilon\{1 - exp[\frac{-(e^{a_j(t)}-1)}{\epsilon}]\}$	$o_j(t) = \frac{1}{1+e^{-a_j(t)}}$
multi-layer	multi-layer
Highly recurrent	Strictly feed-forward
Hebbian, habituation	Backpropagation

majority of research in standard ANNs use a simplified discrete equation, that simply depends on the activity of connected units at a previous time step to determine the activity of the unit in the current time step. The KA model (and original K-sets) model the intrinsic dynamics of isolated neural populations. Both need a second order term in order to capture the description of these dynamics. In the discrete KA model case, two previous time steps are needed in order to describe the momentum of a neural population. Both ANN and KA models use nonlinear transfer function in order to describe the influence of activation being passed between units in a simulation. The form of the transfer function in the K and KA models is an asymmetric sigmoidal transfer function that has a firmer basis in biological observations. The asymmetry is important in the K family of models as it allows for the destabilization of populations of units in response to inputs. Standard ANN research use multi-layer models, mostly of the strictly feed-forward variety, though recurrent models are also very widely studied. The biological models of the K family of equations are always multi-layered highly recurrent models that capture the architecture of brain regions. A final difference between the KA and

ANN models is the form that learning takes. Backpropagation is the main type of learning mechanism used in standard ANN research, and even in recurrent models a form of backpropagation through time is used to update the strengths of the connections between units. Backpropagation is a powerful learning tool but it suffers from the fact that there is little evidence from biological brains for the presence of such a mechanism. The KA and K models use Hebbian learning, habituation and other mechanisms to adjust the weight space in simulations. These learning mechanisms have a firmer basis in biology and have been directly observed as processes in brains. The learning mechanisms used by the KA model will be discussed more thoroughly in later sections when we describe simulations using the KA model to control autonomous agents.

4.4 K-II, KA-II and Oscillatory Dynamics

The K-II set is a connection of four K units, two of which are strictly excitatory and two of which are strictly inhibitory, with 10 recurrent connections. A typical configuration of a K-II is shown in Figure 4.5, Left. The K-II forms a basic component of the K-set hierarchy, upon which larger functional units are built. The K-II is capable of producing two main types of dynamics: damped and sustained oscillations (Figure 4.5, Right). Which type is exhibited will depend on the exact values of the 10 weights between the 4 units. Also, varying the weights will cause variations in the intrinsic frequencies with which the K-II will oscillate. KA units can be used to form a KA-II in exactly the same manner as the original K-sets shown in the figure. Also varying the weights in a KA-II cause variations of the damped and sustained oscillations, and the frequency of oscillation, just as in K-II. Similar weights give similar results in the dynamics of both the K and KA models.

The α and β parameter settings determined in the previous section are good enough to obtain concordance between the K and KA models when we connect together units

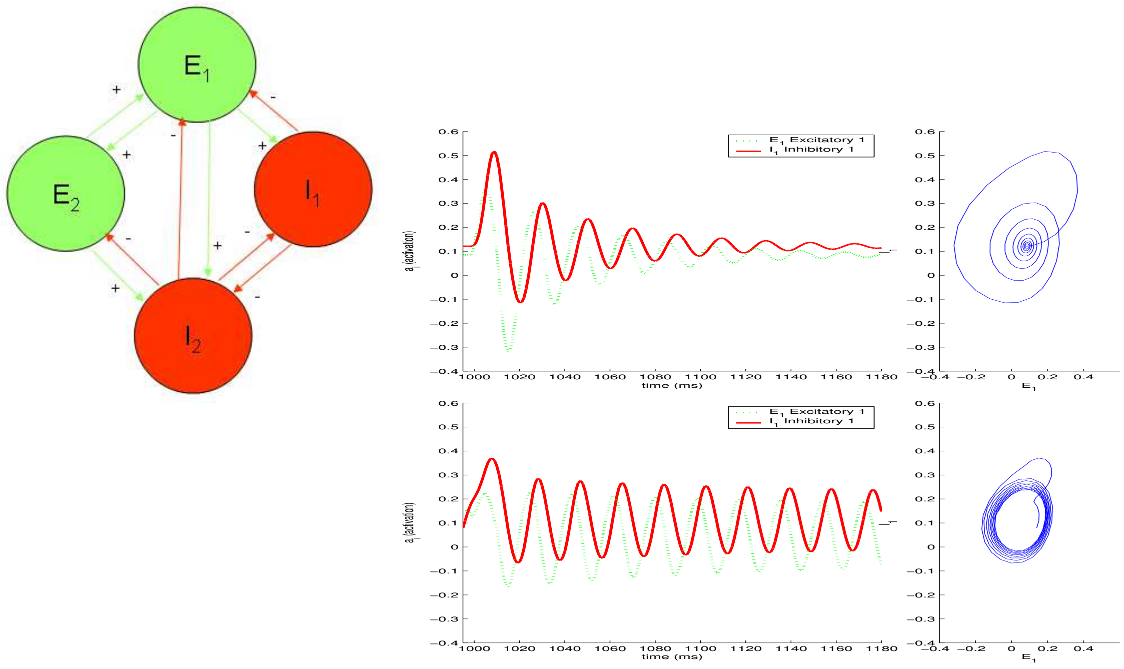


Figure 4.5: (Left) Diagram of the basic K-II architecture. Two excitatory units E_1 and E_2 are connected via recurrent connections to two inhibitory units I_1 and I_2. There are 10 connections typically used out of a possible 12. Depending on the values of these 10 connections the K-II will display two types of basic behavior, damped (Right, Top) and sustained oscillations (Right, Bottom). Varying the weights also varies the intrinsic frequency with which the K-II oscillates. Notice that the sustained oscillation is oscillating at a frequency of around 50Hz.

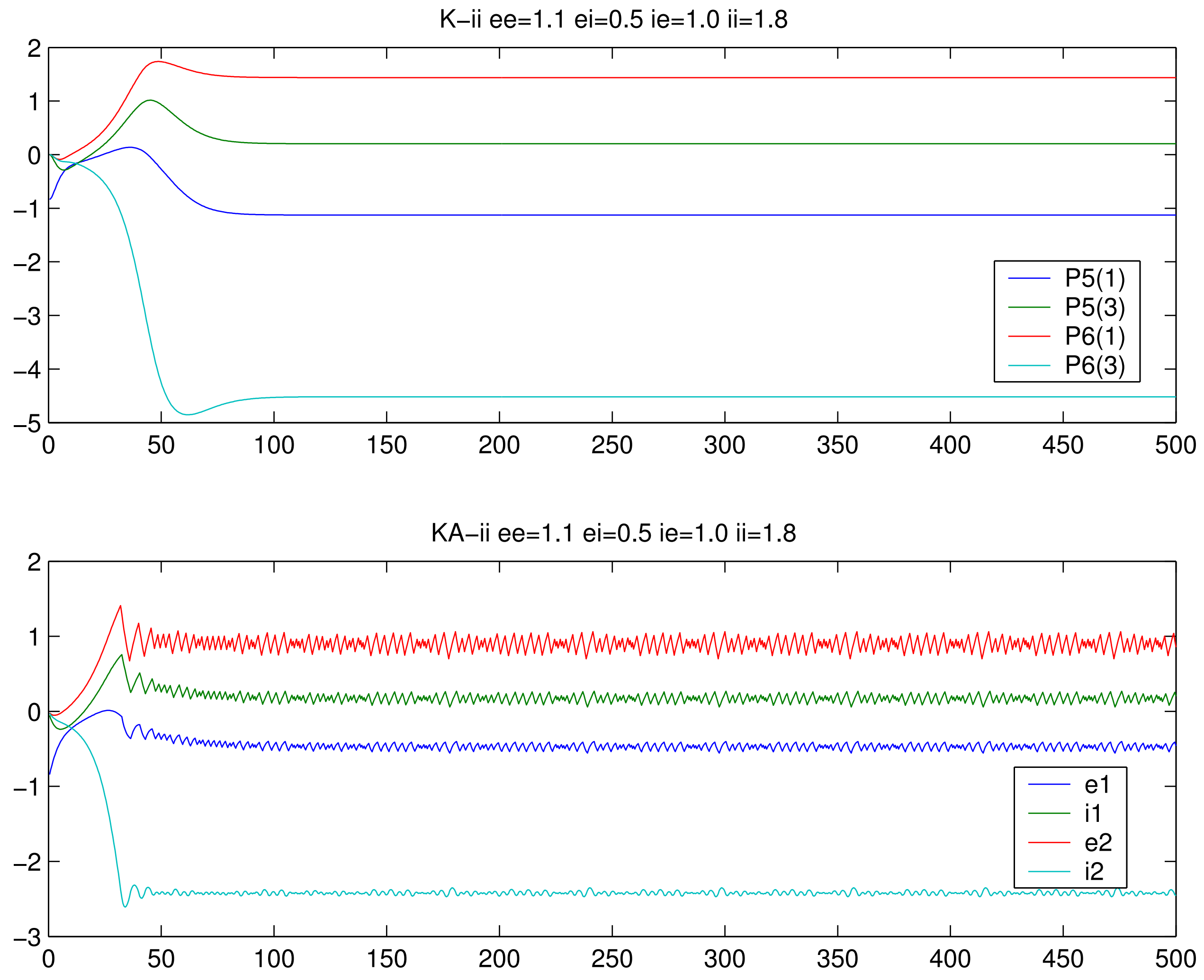


Figure 4.6: Comparison of K-II and KA-II (all internal parameters equal).

to form larger functional components, such as a K-II. Here we show one example of the similarity in dynamics between a K-II and a KA-II. In Figure 4.6 we show the time series of a K-II and KA-II, with all ten of the internal weight parameters ($w_{ee} = 1.1$, $w_{ei} = 0.5$, $w_{ie} = 1.0$ and $w_{ii} = 1.8$) set to equal values. In both models, arousal $\epsilon = 5.0$. The four units in both groups respond roughly the same way and settle down to about equivalent levels, though the KA exhibits a tendency to sustain an oscillation that the original K-II does not have.

4.5 K-III, KA-III and Chaotic Dynamics

The real power of the K and KA models derives from their abilities to replicate the oscillatory and aperiodic dynamics observed in neural populations. As mentioned previously, the K models are typically built up into hierarchies of excitatory/ inhibitory populations in order to model functional brain areas. Connecting three or more KII (or KA-II) sets together using excitatory and inhibitory connections, forms what is called a KIII (or KA-III) set. If the KII components have incommensurate oscillatory behavior (e.g. they are heterogeneous components that oscillate at different characteristic frequencies), then the KIII as a whole is unable to settle into one frequency, and the group displays aperiodic behavior.

The KA units are also capable of replicating the chaotic dynamics of the original K-sets when connected in a KA-III configuration. Figure 4.7 shows a generic architecture of a K-III set. When three or more nonhomogeneous K-II units are connected with various positive and negative projecting connections, the resulting system is able to generate aperiodic dynamics with many of the characteristics of biological nervous systems. In this section we show that the KA units when connected to form a similar KA-III are also capable of replicating aperiodic dynamics with the desired properties.

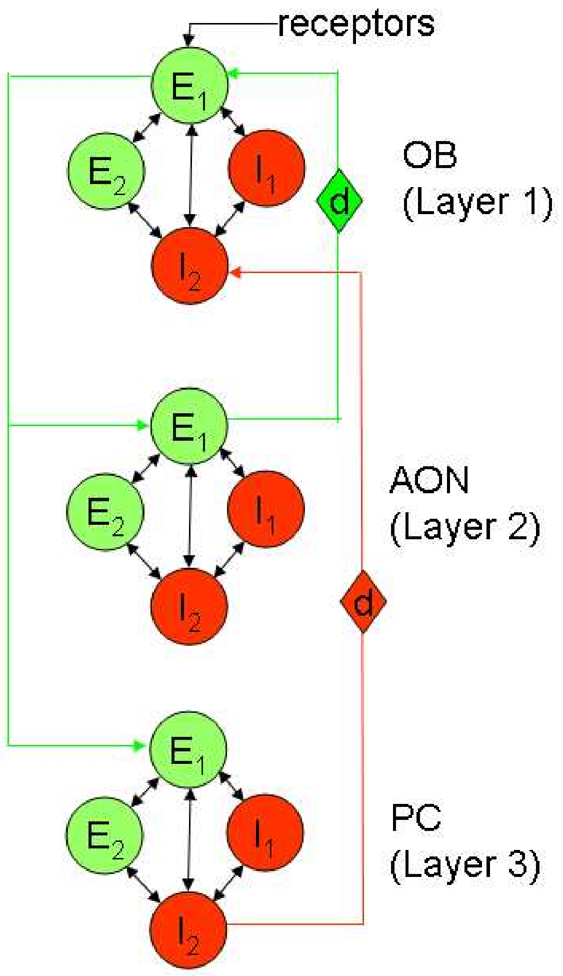


Figure 4.7: A generic model of the K-III configuration. The K-III consists of three K-II sets connected with positive and negative feedback. Some connections, especially those projecting back towards the input areas, may contain time delays. The K-II sets are not homogeneous, they have different internal parameters which give them different intrinsic characteristic frequencies. Originally the K-III modeled the layers of the olfactory system, the olfactory bulb (OB), the anterior olfactory nucleus (AON) and the prepyriform cortex (PC). In generic K-III for various tasks we usually refer to these as the 3 layers of the K-III, where layer 1 is usually where receptor input is received, and layer 3 is where measures are taken to determine output.

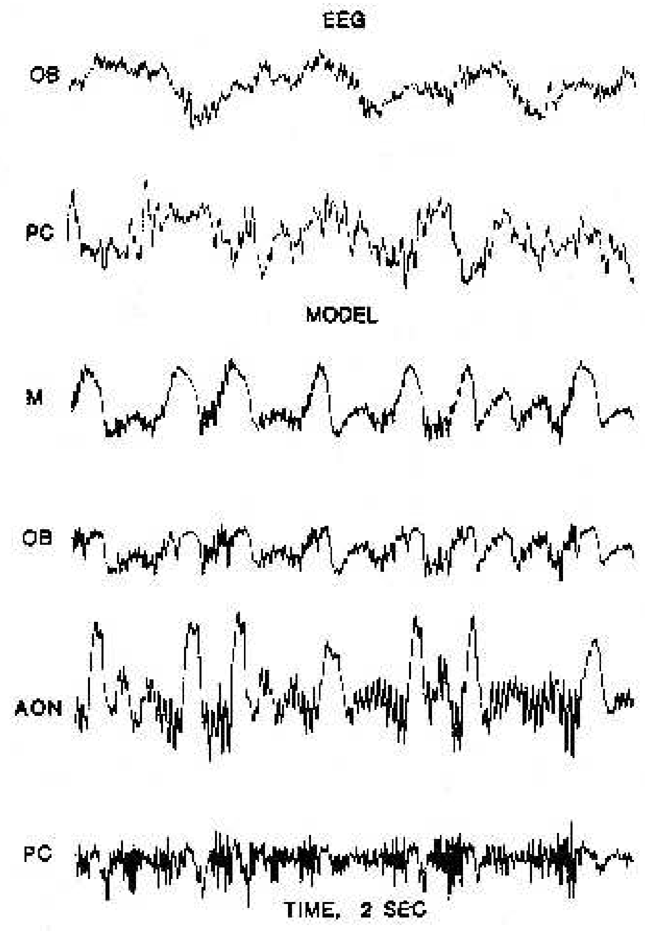


Figure 4.8: Comparison of basal chaotic EEGs of the olfactory bulb and the cortex (upper two traces) and simulated basal time series from the K-III model (lower four traces) (Figure from Freeman, 1987).

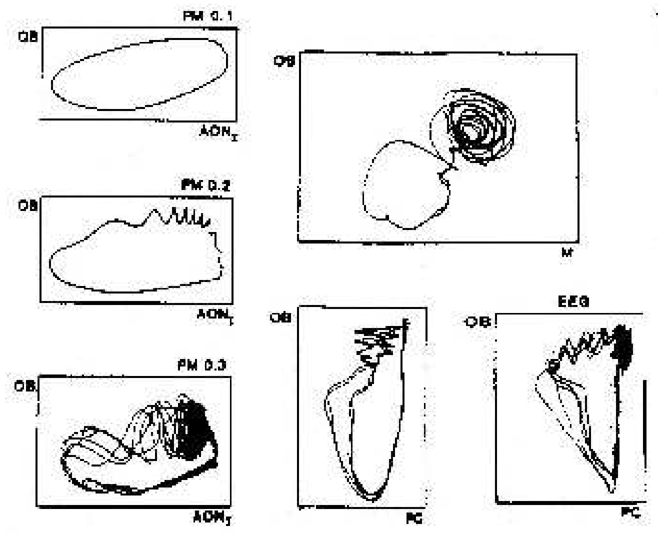


Figure 4.9: Phase space traces (left) show K-III model showing a low-dimension limit cycle (above), a high-dimension limit cycle (middle), and a chaotic attractor (bottom). The lower right figures compare the K-III model phase space with a rat EEG data during a seizure (Figure from Freeman, 1987).

In Figure 4.8 we show time series captured by Freeman (Freeman, 1987) of the basal chaotic EEGs of the olfactory bulb of rabbits. This figure also contains time series of the K-III model replicating these dynamics. Similarly, in Figure 4.9, we show phase space plots of the K-III model displaying low-dimension, high-dimension and chaotic background activity. All of these dynamics were observed in the olfactory perceptual system and replicated by the K-III model, by varying the connection patterns between layers.

When KA units are connected to form a generic KA-III, they are capable of reproducing these types of quasi-periodic and chaotic dynamics. We show one example in Figure 4.10 of a chaotic background state generated by the KA. Many more examples of KA-III dynamics may be found in Appendix A.

In this figure we show time series (top three plots) power spectrum distributions (middle three plots) and phase space (bottom three plots) representations. The plots are

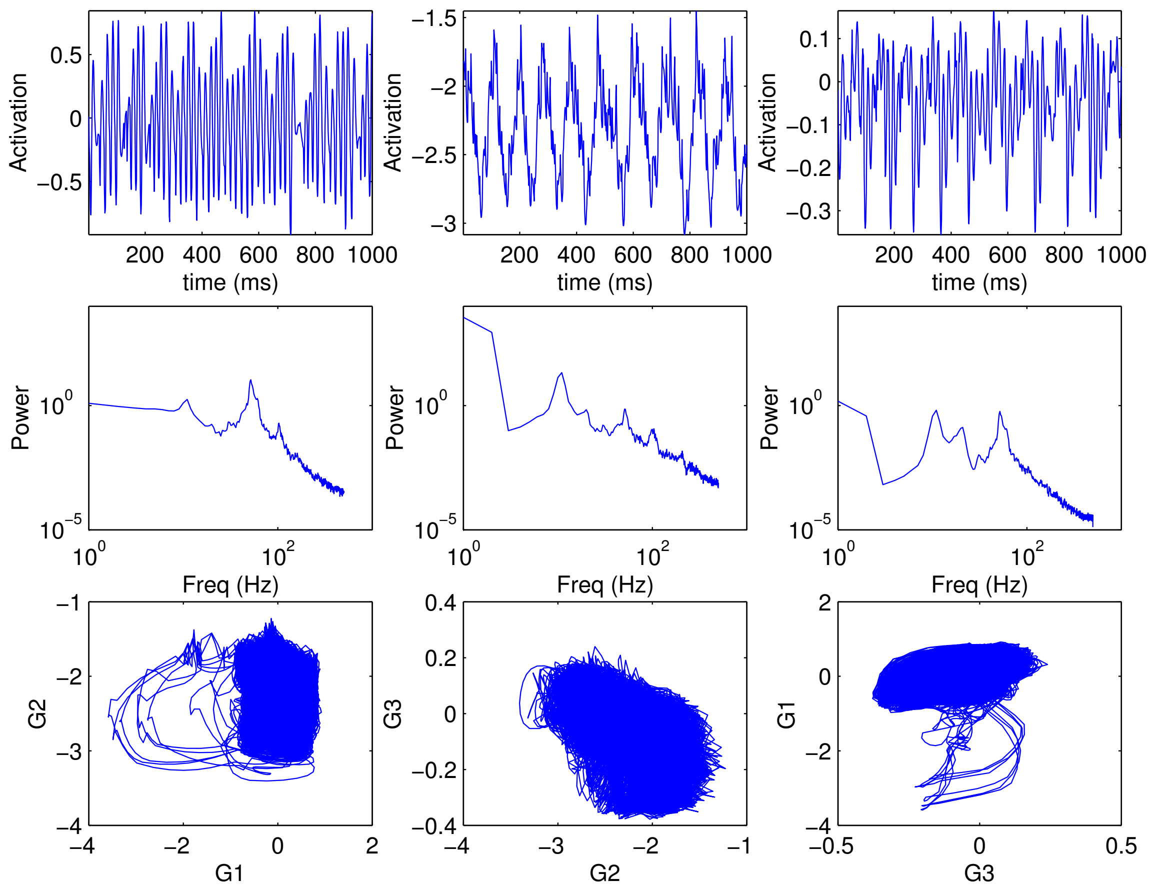


Figure 4.10: Chaotic dynamics generated in an example KA-III. Top figures are time series of Layer 1 (left) Layer 2 (center) and Layer 3 (right). Middle are power spectrum distributions (PSD) of the three layers respectively. And bottom are phase space plots of Layer 1 vs. Layer 2, Layer 2 vs. Layer 3 and Layer 3 vs. Layer 1.

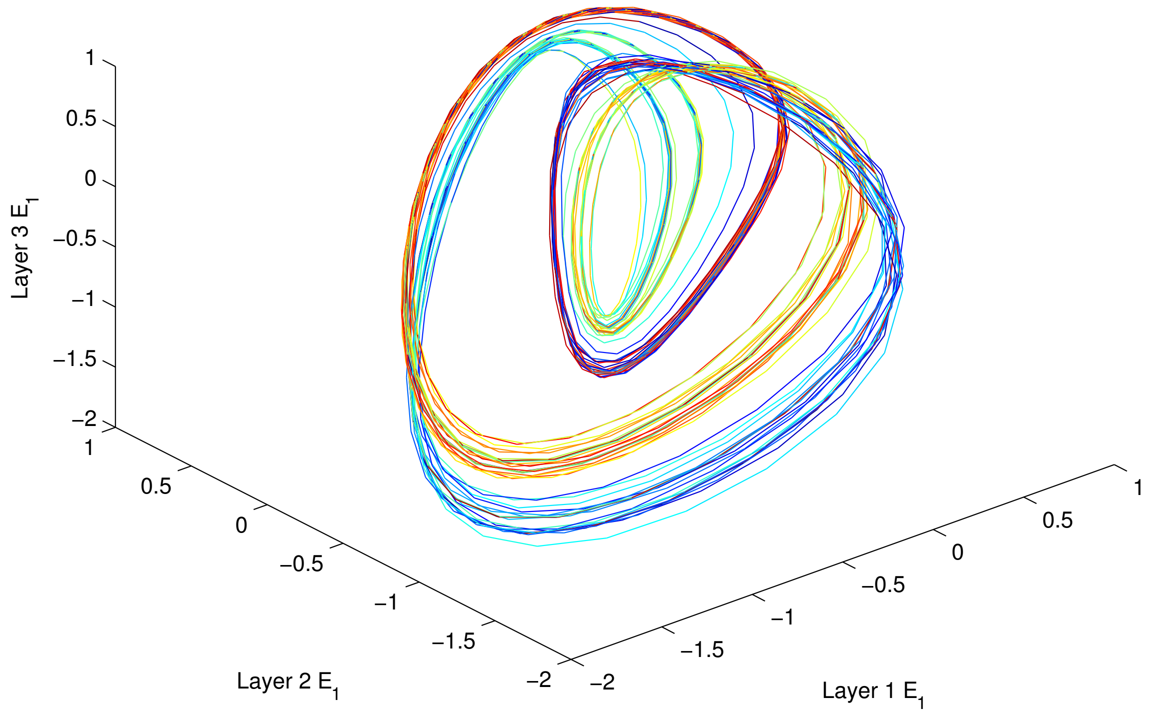


Figure 4.11: Another example of a phase space plot generated using a KA-III. In this 3-dimensional plot we show the activity of the E_1 unit in Layer 1 on the x axis vs. the activity of Layers 2 and 3 on the y and z axis respectively. Color is used to give some indication of points that are close together in the phase space.

data from the E_1 unit of Layers 1, 2 and 3 respectively, and correspond to the similar unit in the original K-III OB, AON and PC areas. We will discuss the power spectrum produced by the KA model in the next section in more detail. The state space plots are further indications of the aperiodic nature of the KA-III dynamics. A measure of the Lyapunov exponent of the time series indicates a strictly positive first exponent that can range from 0.0 to 0.3 depending on the connectivity pattern and weights between the KA-II groups (and the parameters of the KA-II groups themselves). A 3-dimensional view of a phase space generated by a KA-III is shown in Figure 4.11.

The original K-III model was also capable of replicating the abnormal dynamics of brain seizures observed in the olfactory system when the neural tissue was over-stimulated. Figure 4.12 shows an example of the time series of the rat EEG during such a seizure and the K-III models ability to reproduce these dynamics. During seizure, the neuronal group activation takes on a much more quasi-periodic nature.

The KA-III are also capable of reproducing these types of quasi-periodic dynamics. In Figure 4.13 we show one such example. Again we show time series, power spectrum and phase space plots of the dynamics in the three layers of the KA model.

4.6 Power Spectra of K and KA Sets

In the original K model, the purpose of the K-III set was to model the chaotic dynamics observed in rat and rabbit olfactory systems. The K-III set was not only capable of producing time series similar to those observed in the olfactory systems under varying conditions of stimulation and arousal, but also of replicating major power spectrum characteristics of these time series. The power spectrum is a measure of the power of a particular signal (or time series as for example that obtained from an EEG recording of a biological brain) at varying frequencies. The typical power spectrum of a rat EEG (see Figure 4.14, Top) shows a central peak in the 20-60 Hz range, and a $1/f^{\alpha}$ form of

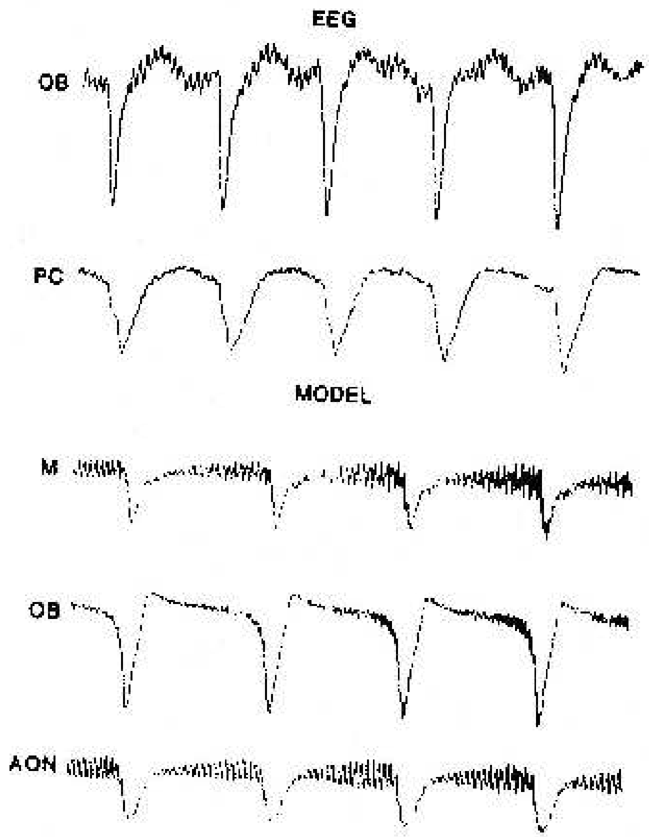


Figure 4.12: Examples of 2-second time segments of EEGs recorded from rat (above) during seizure, comparing these with the outputs of the K-III model. (Figure from Freeman, 1987)

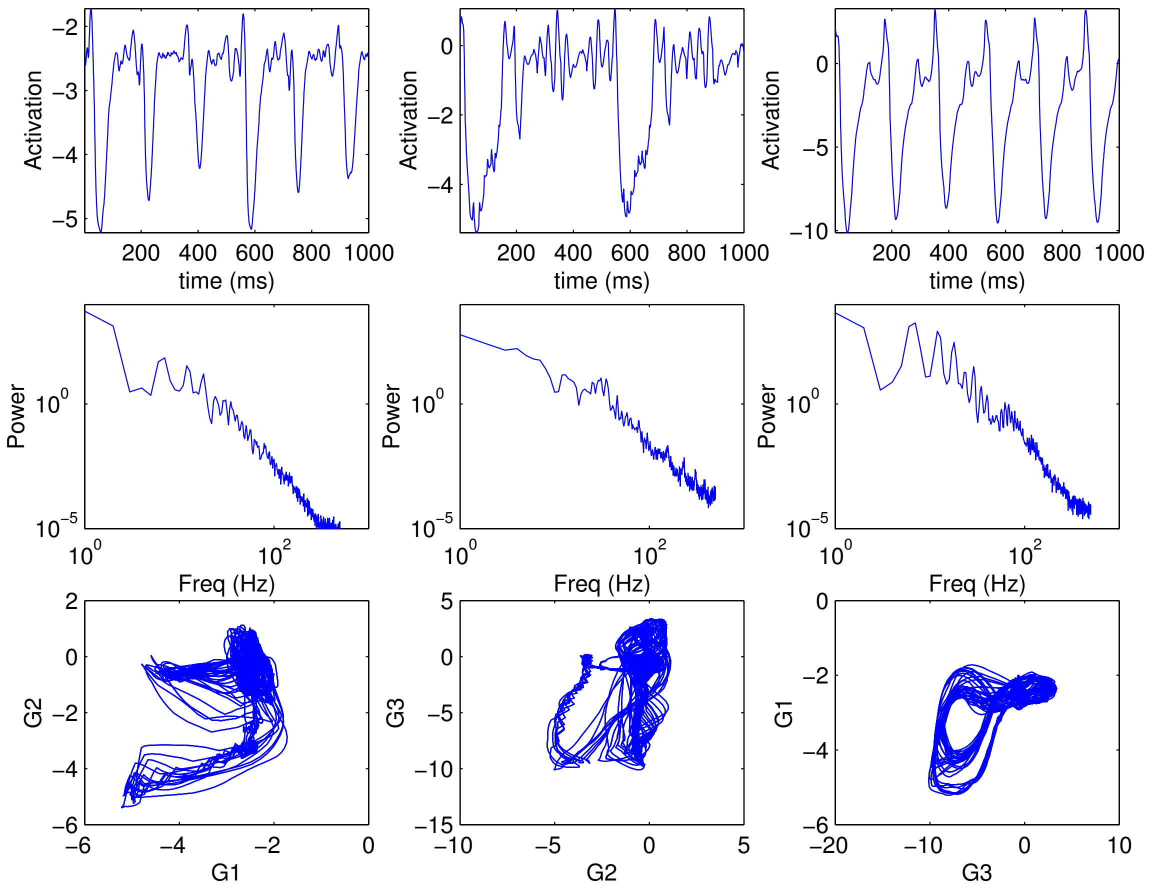
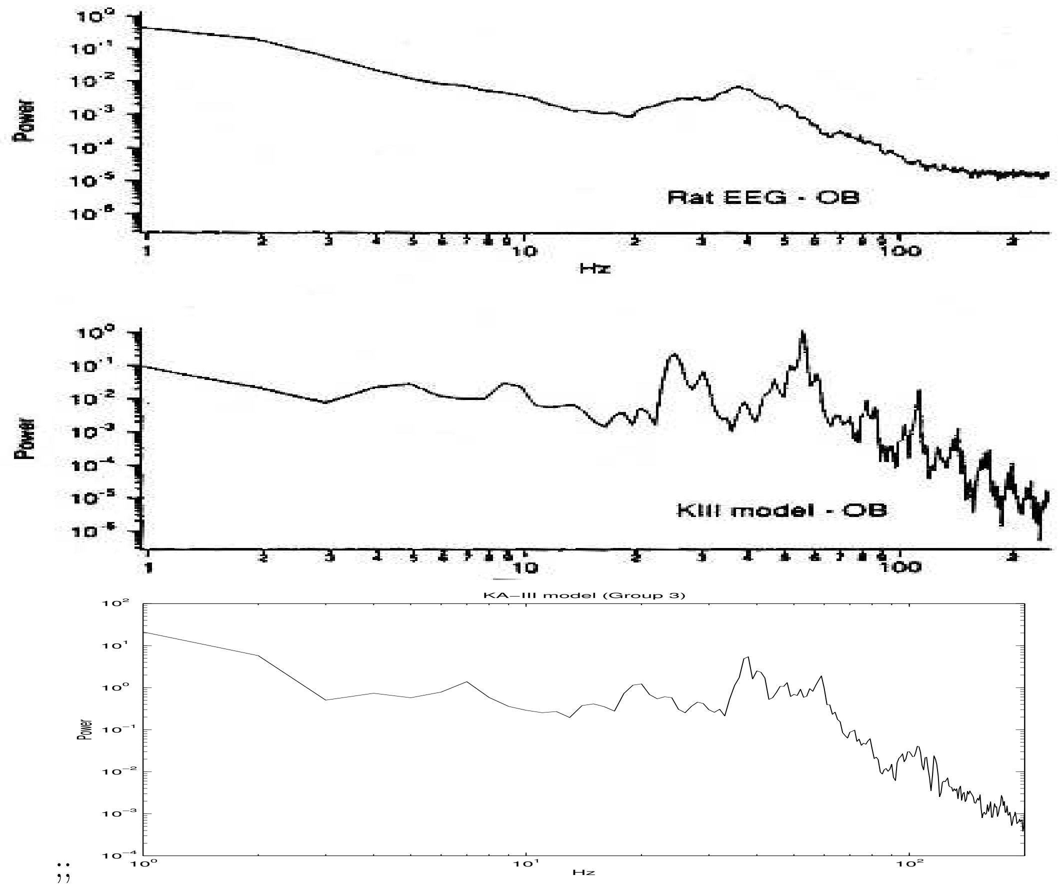


Figure 4.13: Example of seizure dynamics in a KA-III model. Top figures are time series of Layer 1 (left) Layer 2 (center) and Layer 3 (right). Middle are power spectrum distributions (PSD)and bottom are phase space plots. All show the dynamics of a KA-III that is displaying quasi-periodic dynamics similar to seizure states in biological brains.



;;

;;

Figure 4.14: Comparison of power spectrum distributions of rat EEG (top) with KIII (middle) and KA-III (bottom) models. Both K and KA show 1/f spectra with a slope of around -2 and a central frequency peak in the γ band. (Rat OB data and K-III model from (Kay, 1990), KA power spectrum from (Harter & Kozma, 2004)).

the slope. The measured slope of the power spectrum varies around α = -2.0. $1/f^{\alpha}$ type power spectra are abundant in nature and are characteristic of critical states, between order and randomness, at which chaotic processes operate (Solé & Goodwin, 2000). The atypical part of the experimental EEG spectra is the central peak, indicating stronger oscillatory behavior in the γ frequencies. This central peak in the 20-60 Hz range is known as the γ frequency band, and is associated with cognitive processes in biological brains. In Figure 4.14, Middle, we show an example of the original K-IIIs power spectra and on the bottom we show an example of the KA-III models ability to replicate these spectra. In particular the power spectrum analysis shows the typical $1/f^{\alpha}$ power spectrum with a slope of around -2 and a central frequency peak in the γ band for both the KIII and KA-III models.

4.7 Measurement of Lyapunov in KA-III while Varying from Low-Dimensional Periodic to Chaotic Dynamics

In order to better understand the generation of chaos in the KA-III model we will now look at how the measured Lyapunov exponent of a KA-III time series changes in response to varying connection weights between the layers of the KA-III. The Lyapunov exponent is a measure of how fast near by points generated by a system diverge from one another. Therefore the Lyapunov exponent gives an estimate of how chaotic a time series is. Negative Lyapunov exponents indicate that the signal is not chaotic, but is instead a point attractor. The closer the exponent approaches to zero, the longer the signal takes before settling down to its point attractor. A zero exponent is an indication of a periodic time series. The more positive the exponent, the more the signal is moving from a quasi-periodic dynamic to a true chaotic dynamic. We use the TSTool Matlab toolkit to calculate the Lyapunov exponent. The calculation used in TSTool is based on Wolf's method for calculating the Lyapunov exponent (Wolf, Swift, Swinny, & Vastano, 1985).

We now demonstrate the effects of changing the weights between the layers of a KA-III on the calculated Lyapunov exponent. We first found a particular configuration of the KA-III that was displaying highly chaotic behavior. The measured Lyapunov exponent of the time series was around 0.4, indicating a fairly strong chaotic signal. We then systematically varied the strength of the connections between the layers from 100% down to 0% of their original values, in 1% increments. The 100% level is simply the original KA-III. When the strength of the weights is reduced down to 0%, in effect the KA-II layers become isolated and are no longer connected to one another. At this level, the isolated KA-II should display their intrinsic damped or sustained oscillatory behavior, which will have a calculated Lyapunov exponent of around 0.

Ten simulated time series were generated for each weight setting from 0% to 100% , and the Lyapunov exponent calculated on the resulting time series. Figure 4.15 plots the effects of scaling the weights on the Lyapunov exponent for this KA-III. It is seen that for small scaling factors the Lyapunov exponents approach zero. This is in accordance with the expectations that the dynamics of the individual KA-II units have a limit cycle oscillatory nature with zero Lyapunov exponent. The Lyapunov exponents incrementally increase with increasing weight scaling factor, until a certain level at around 0.25, where a gap in the chaotic behavior is observed. Strong chaotic behavior resumes above weight values of 0.35.

4.8 Learning Mechanisms for KA

In this section we discuss in more detail the learning mechanisms that will be used in the KA multi-layer recurrent neurodynamical models. In the simulations with autonomous agent architectures we will be using two of the four types of learning mechanisms de-

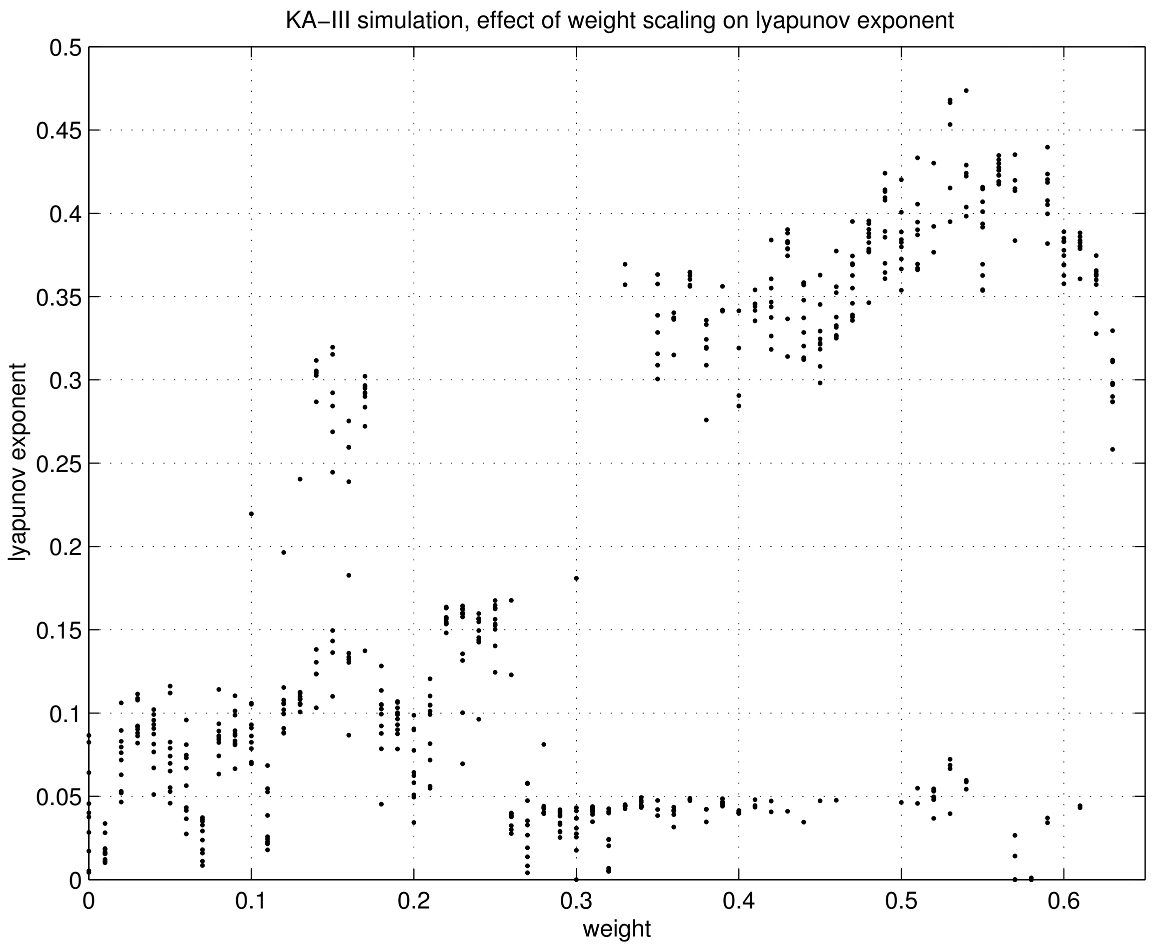


Figure 4.15: Effects of weight scaling between the KA-II layers of a KA-III on the calculated Lyapunov exponent of the time series. The intergroup excitatory weights are scaled from 0.0 to 0.65 in 0.01 increments. Ten points are shown at each weight value, representing runs with ten different initial conditions.

scribed previously, Hebbian synaptic weight updating and habituation of unreinforced stimuli.

4.8.1 Hebbian Mechanisms in the KA Population Model

The basic idea behind Hebbian mechanisms is that when the activity of two connected neural units co-occur, they have some statistical relationship to one another. We can exploit this relationship by increasing the likelihood that in the future if one of the units is active the other becomes active. This can be done by increasing the strength of the weight between the units. In other words, units that tend to fire together should have the weights between them strengthened so that they are more likely to fire together in the future. The converse of this rule is also true, if the units do not tend to fire together, then the strength of any connection between them should weaken over time. This simple mechanism defines a type of competitive process among the links between neural units.

Hebbian learning is a simple concept, but it is very powerful in shaping the weight space of a neural model to process stimuli. Hebbian mechanisms allow the models to capture statistical regularities in the stimulation patterns that occur in the environment of the organism.

In Figure 4.16 we show an example of the simplest formal definition of the Hebbian learning mechanism. In this figure we consider a pre-synaptic node A and post-synaptic node B connected by a link with weight w_{BA}. The activity or firing rates are represented by the values a_A and a_B. For simple models where the activity of the units a_i is a slow changing process, we can correlate the activity between the units to determine the difference we wish to apply to the weight as:

$$\Delta w_{BA} = \varepsilon a_A a_B \tag{4.11}$$

Here the proposed change to the weight Δw_{BA} is simply a function of the product of the activity of the pre and post-synaptic nodes times a learning rate parameter ε.

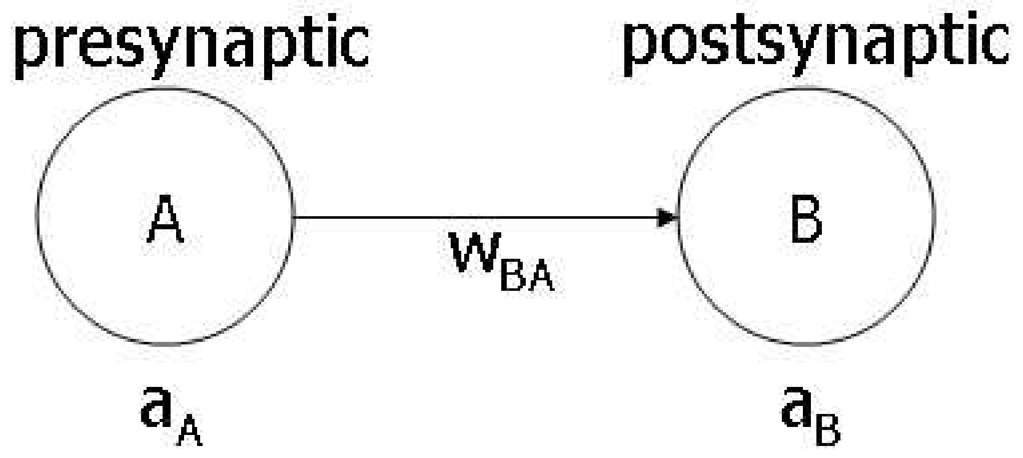


Figure 4.16: In Hebbian learning, pre and post-synaptic nodes are connected by a link with weight w_{BA}. The change in the weight between nodes is a function of the correlated activity of the nodes a_A, a_B over time.

In spiking models where the units are modeling the spiking activity of a single neuron, we can't use the activity level of the unit as this has no real meaning. Instead we must look at the firing rate of the unit over some time period. We can determine if a unit is more or less active by comparing its current firing rate to what its normal or average firing rate usually is. The slightly more complex Hebbian rule thus becomes:

$$\Delta w_{BA} = \varepsilon(a_A - \overline{a_A})(a_B - \overline{a_B}) \tag{4.12}$$

Here $\overline{a_A}$ and $\overline{a_B}$ represent the average firing rates of the pre and post-synaptic nodes respectively. In these firing rate models, notice that the current firing rate can be lower than the average, which can lead to negative, or decreasing weight changes. This may or may not be what is wanted depending on the type of model being experimented with. For example, it may or may not make sense to strengthen the weight between two units when they both have less than average activity at the same time.

The K family of models, including the KA model, are neural population models, not models of single neurons. Therefore the concept of the firing rate of a node is not relevant. The activity level in KA units represents an average current density for the population. However, unlike the simple case, this average population current can change rapidly which makes a simple Hebbian equation inadequate for our use. We instead need to develop a concept of the activity of a unit over some time window. In the KA models we use the root mean square to calculate the activity over a time window:

$$rms(i, a, b) = \sqrt{\frac{1}{b-a}\sum_{t=a}^{b} a_i(t)^2} \tag{4.13}$$

This states that the root mean square intensity of unit i over the time interval a to b is given by taking the sum of the squares of the activity over the time interval, dividing it by the time interval, and taking the square root. The root mean square is a better measure of the activity of a unit over a time interval than simply taking the average of the units activity. The root mean square is invariant with respect to the average activity level, which makes comparing the *rms* of two units more plausible.

Given the definition of the *rms* to calculate the activity of a unit over an interval, we can define the Hebbian equation for the KA model. Normal Hebbian rules compare the activity of a unit to its average activity. We instead compare the average activity of a unit over an interval to the average activity of some subset population of units. This is necessary as determining a base or average activity is not a straightforward proposition in the K family neural population models. We therefore determine how the activity of a unit is varying by comparing it to the current average activity of a population.

The Hebbian equation used by the KA experiments is given by:

$$\Delta w_{BA} = \varepsilon(rms(A, a, b) - \overline{rms(sea, a, b)})(rms(B, a, b) - \overline{rms(sea, a, b)}) \tag{4.14}$$

where $rms(sea, a, b)$ is a spatial ensemble average of some population of units that the units A and B belong to. The learning rate parameter, ε, is determined experimentally for each simulation by tuning it to have optimum performance as defined by the simulation. In a similar manner, the time window used is also determined experimentally for each problem. The normal time window is take from the current time to some time in the past which can vary from 50 to 250 time steps.

4.8.2 Habituation in the KA Model Experiments

The second learning mechanism used in the following experiments is habituation. Habituation is defined as a diminished response to sensory stimuli that is not reinforced. Sensory signals that are repeatedly encountered but never co-occur with appetitive or aversive signals become diminished in the organism. This phenomena is very familiar to people as, for example, we quickly "tune out" background noise such as an air-conditioner in our environment.

In the KA models, Hebbian learning only occurs when reinforcement signals are generated in the organism. In the following experiments, reinforcement signals are usually hard-coded in the agent such that when it finds food sources positive Hebbian reinforcement occurs, or when it bumps into obstacles negative Hebbian reinforcement may occur. When a reinforcement signal is not currently being produced by the organism, habituation of stimuli may be performed.

Habituation of stimuli is performed in KA by lessening the strength of connections to neural units that are more active than an average population activity during times of non-reinforcement. The basic weight modification for habituation is defined as:

$$\Delta w_{BA} = -\eta |(rms(B, a, b) - \overline{rms(sea, a, b)})| \tag{4.15}$$

Here the habituation weight of a link from A to B Δw_{BA} is a function of how far the unit B's activity is above or below a spatial ensemble average of some subpopulation of units, times a habituation decay constant η. Again η is determined experimentally for each simulation by tuning it for optimum performance. For some cases where we only want to habituate nodes whose activity is higher than the average (not those that are lower), we can use $\Delta w_{BA} = 0$ if the *rms* of B is lower than the spatial ensemble average.

Hebbian modification and habituation are usually performed on all plastic connections in the simulation. That is to say, some connections in a simulation are not variable, and therefore do not learn and change in response to environmental experiences. The internal links within the KA-II are examples of non-plastic connections in the simulations. Plastic connections are usually those links between units within a layer of a KA-III.

4.9 Development of Basic Object Avoidance Behavior in an Autonomous Agent Using KA

The original K sets have been shown to be good models of the olfactory cortical dynamics. They can replicate the complex dynamics and power spectra of biological cortical recordings. Also the K sets are capable of learning through habituation and Hebbian modification to form attractors that come to be associated with perceptual stimuli. The K sets have also been extended to more abstract tasks to demonstrate their use in standard pattern recognition tasks (Shimoide et al., 1993; Kozma & Freeman, 2000, 2001a). In this first experiment, we use the KA model to learn to produce simple actions by a robotic agent in a virtual environment. The agent learns to associate unconditioned responses, such as turning away when bumping into a wall, with conditioned stimuli, such as the recognition of an obstacle at a distance. This association of unconditioned responses with conditioned stimuli allows the agent to navigate in its environment without bumping into obstacles.

4.9.1 Agent and Environment Setup

In this section we show an example of using KA units to produce simple behavior in an autonomous agent. The Khepera virtual simulation environment is used for these experiments (Michel, 1996). We use the same experimental setup and robot architecture that has been explored in the original Distributed Adaptive Control models of Verschure and Pfeifer (Verschure et al., 1992) to test the development of object avoidance behavior. Figure 4.17 shows the morphology of the Khepera robot and the internal architecture used to perform the experiment. The Khepera robot is a simple robot that contains 8 infra-red (labeled IR_{1-8} in the figure) and 8 light sensors (not used in this experiment), as well as 8 touch sensors. The sensors are spread around the perimeter of the robots body with a heavier concentration of sensors towards the front of the robot.

In this task, the simulated Khepera robot is originally endowed with a set of unconditioned reflexive behaviors that allow it to wander around in its environment, bumping into obstacles and turning away from them. These unconditioned responses are triggered from the touch sensors, which constitute a set of unconditioned stimuli that define the basic behavior of the agent. For example, if the robot bumps into an object on the left side of its body, it backs up a little and turns away to the right and then attempts to continue forward. The Khepera robot is equipped with two independent motors attached to wheels, that allow the robot to move forward, backward and turn in place.

4.9.2 Experiment Description

In Figure 4.17B. we show the architecture used to perform the experiment. We hardwire a set of four reflexive behaviors to perform appropriate action sequences to allow the robot to wander in the environment. Three of the behaviors (bump right, bump left, bump front) respond to the robot running into an obstacle (touch) and trigger appropriate behavior sequences to escape from the obstacle (e.g. backup, turn away, then continue forward). The fourth behavior causes the robot to instinctively move forward (wander) whenever there are no obstacles obstructing it (e.g. when none of the other behaviors are currently active).

In the experiment, we also have a set of units that are connected to the long range infra-red sensors (labeled 'Sensory' in Figure 4.17B.). The infra-red sensors can sense obstacles at a distance from the robot. Initially these senses are connected with random

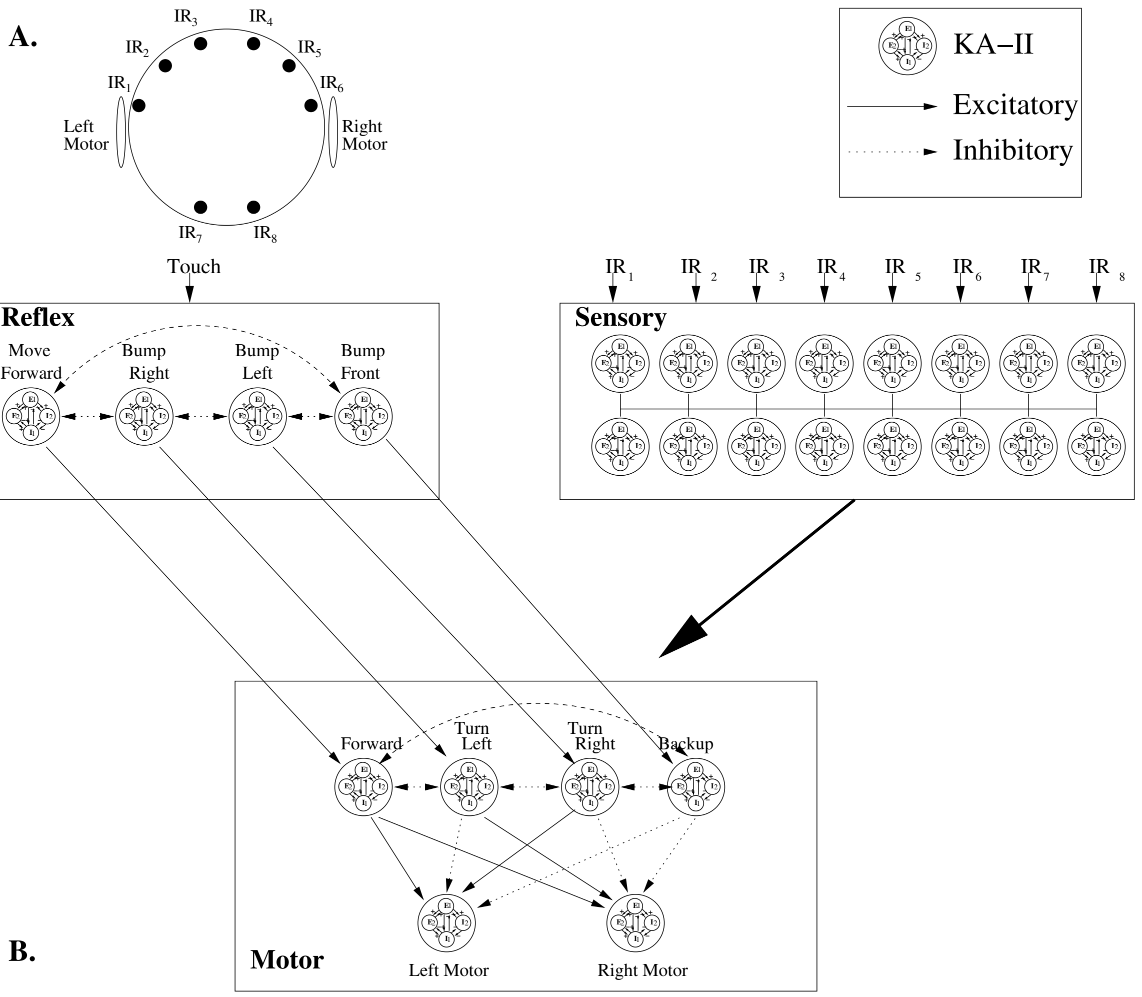


Figure 4.17: A. The morphology of the Khepera agent with 8 infra red sensors positioned around the body and 2 motors for movement. B. The internal architecture of the Khepera agent. Reflexes are hardcoded to produce exploration of the environment. Sensors gradually learn to produce behaviors to avoid objects at a distance.

weights to the four basic motor behaviors. All sensory units are fully connected to the four behaviors (not shown in figure). There are 16 KA-II units used in the sensory input, 8 of these units receive direct stimulation from the corresponding infra-red sense, while the other 8 receive the inverse of the infra-red sense. When a unit receives direct stimulation from an infra-red sensor, it becomes most increasingly activated as the sensor approaches an obstacle in the environment. Units that receive inverse stimulation are most highly activated when the obstacle is far away or no obstacle is detected by the sensor. This allows units to be sensitive to both the presence and absence of an obstacle in a particular direction from the robot.

We use Hebbian learning (Kozma & Freeman, 2001a) on the connections between the 'Sensory' and the 'Motor' units. Since these connections are initially random, typically they do not affect the behavior of the robot in the beginning. The reflexive behaviors cause the robot to move around in the environment. Later on the robot may bump into something on its left. This will cause some of the Motor behaviors to be performed, such as turning right. Since the Sensory units that are connected to sensors on the left side of the body have become stimulated while approaching the obstacle, they remain highly active when the right turn behavior is activated. This allows the strength of the connection between the Sensory unit for detection of obstacles on the left and the right turn behavior to become strengthened due to learning. Similar strengthening is happening between units that sense the absence of obstacles on the right and the right turn behavior as well. Eventually the link becomes strong enough to activate the right turn behavior when an object is sensed at a distance, before the robot actually bumps into it. Therefore we can say that the robot has learned a type of object avoidance behavior through coupling of the activity of its sensors with its motor behaviors.

4.9.3 Results

In Figure 4.18 we show the results of learning object avoidance by the KA units. In this figure we show the average performance of the robot over 50 independently conducted simulations. We plot both the results with only reflexive behavior (No Learning) and with the Sensory units connections being manipulated through Hebbian modification (Learning). Along the X axis we show the time (in seconds) that the simulation has been running. We plot the total number of times that the robot has bumped into an object in the environment. In the case of the 'No Learning' condition, the robot continues to move and bump into obstacles in the environment. In the Learning condition, the robot quickly begins to avoid objects, and will eventually learn to move through the environment without bumping into anything at all.

4.9.4 Discussion

The previous experiment is a demonstration that the KA units can be used to build control architectures for autonomous agents. Hebbian modification of the weights between the unconditioned sensory stimuli and the conditioned reflexive behavior is sufficient to learn environmental regularities of obstacles presented to the infra-red sensors and begin to associate them with appropriate actions. This simple demonstration, however, does not yet use the aperiodic properties of the KA model to form the perceptual categories. In the next experiment we show how a KA-III may be used to generate aperiodic dynamics for use in controlling an autonomous agent.

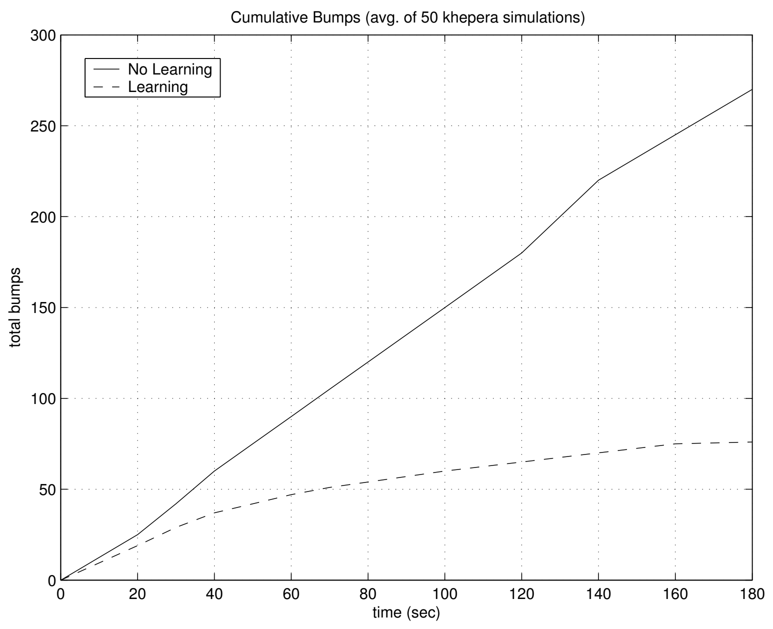


Figure 4.18: Results of object avoidance simulation. As time goes by, the robot learns to bump into things less and less. This figure represents the average results of 50 independent simulations. Time (in seconds) is along the X axis, and the cumulative bumps is plotted along the Y axis. We show the results without learning (only reflexive behavior) and with learning among the KA units turned on.

4.10 Development of Appetitive/Aversive Behavior in an Autonomous Agent Using KA

We now demonstrate the use of aperiodic dynamics in forming perceptual categories in an environment and then associating those categories with appropriate actions. In this experiment we will use a KA-III to discriminate between two categories of food sources in the environment, edible and poisonous. The attractor associated with poisonous food sources will become associated with aversive behavior which will allow the agent to learn to avoid poisonous food sources without having to taste them.

4.10.1 Agent and Environment Setup

In this experiment we are again using the Khepera robot in a simulator. This experiment was performed, however, using the Webots V4.0 3-d physics based mobile robot simulator simulating a Khepera mark II robot. Figure 4.19 (bottom left) shows the morphology of the Khepera agent which is still the same as in the previous experiment. The Khepera robot is a simple agent that contains 8 infra-red and 8 light sensors. The sensors are distributed around the perimeter of the robots body, but concentrated towards the front of the robot. The robot has two independently controlled wheels that allow it to move forward, backward, and turn left and right in place. The environment for this experiment is shown in Figure 4.19. In the environment we place 6 simulated food sources, 3 of which are good tasting and edible by the agent, and 3 of which are poisonous and have a correspondingly bad taste. We use light sources, and the Khepera agents light sensors, to simulate the production of an environmental gradient which may be followed by the agent to locate a food source (see Figure 4.20). This gradient following is similar to simple tropic behaviors, such as following a chemical gradient (chemotropism, of which smell is an example), or following a light gradient (phototropism).

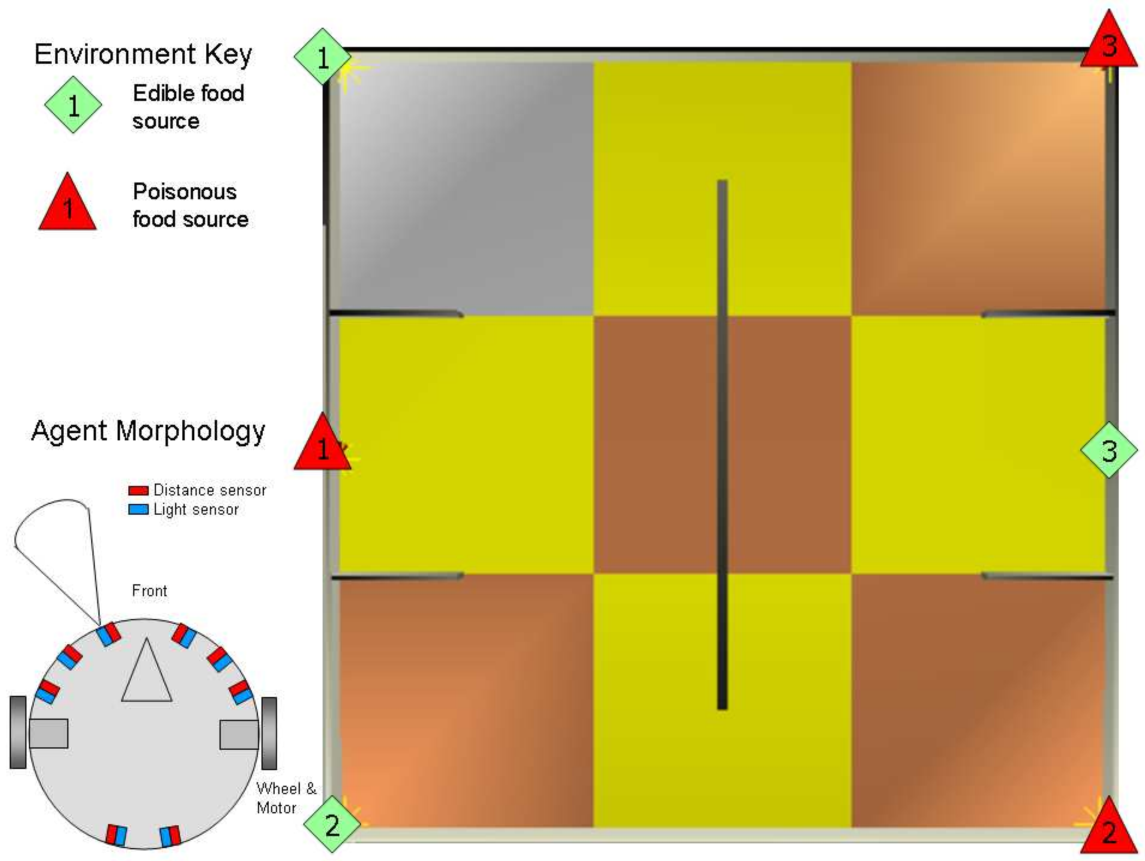


Figure 4.19: Environment and agent used in the appetitive/aversive experiments. The agent (bottom left) is equipped with sensors spaced around its body and two independently controlled wheels for movement. The environment contains 6 food sources, 3 poisonous and 3 edible. Each food sources emits a property (perhaps like smell) which produces a gradient in the environment that is perceptible and followable by the agent.

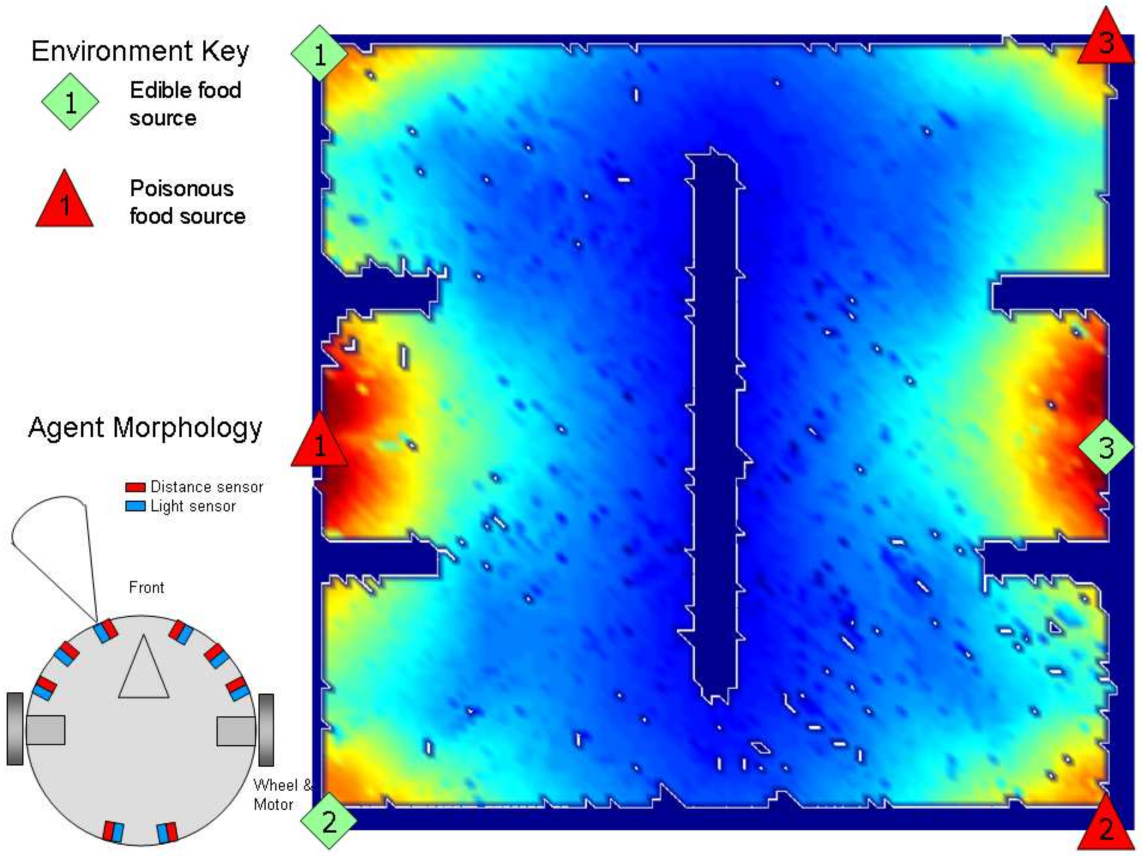


Figure 4.20: This figure illustrates the environmental gradients produced by the food sources in the environment. Red color indicates a very strong signal, while blue is very weak or nonexistent as far as the agents senses are concerned. The agent is given reflexes that allow it to detect and follow these gradients to their sources.

The architecture used for learning appetitive/aversive behavior is shown in Figure 4.21, and is inspired by the Darwin series of robotic agents (Edelman et al., 1992; Almássy et al., 1998; Sporns et al., 1999). The agent receives sensory information from four types of senses. A touch sense, a short-range distance sensor that can detect the presence of obstacles (using the infra-red sensors), a sense of smell for following environmental gradients (using the light sensors), and a simulated sense of taste for detecting good/bad food sources. The touch sense (not shown in Figure 4.19 bottom left), and distance sense are hardwired to reflexive behaviors that cause the agent to wander in the environment and instinctively approach all food sources it encounters by following the gradient to them. There are 5 touch sensors, that allow the agent to detect when it is touching an object in the front of its body, behind it, or to the left and right, or when it is not touching anything at all. There are also 8 short-range obstacle senses that allow the agent to detect the presence of obstacles at a short distance.

The smell sense (using light sources to detect environmental gradients), is hardwired, along with the touch and obstacle sense, to produce approach behavior to detected food sources. The agent simply follows the gradient (avoiding obstacles) to its source. There are 8 smell senses (light sensors) positioned around the body of the agent. Finally, a sense of taste is simulated for the agent using 2 sensors. When the agent touches an edible food source, this produces appetitive behavior (consumption) and pleasure signals in the value system of the agent. Poisonous sources produce avoidance behavior, which causes pain signals and behaviors that make the agent move away from the food source.

We use a KA-III to model the olfactory system and form perceptual categories of the smells in the environment. The olfactory KA-III is composed of three areas, the olfactory bulb (OB), anterior olfactory nucleus (AON) and prepyriform cortex (PC) (see Figure 4.21). These three areas are connected through positive and negative feedback to one another. The OB has projections from KA-I units that receive stimulation from the environmental smells.

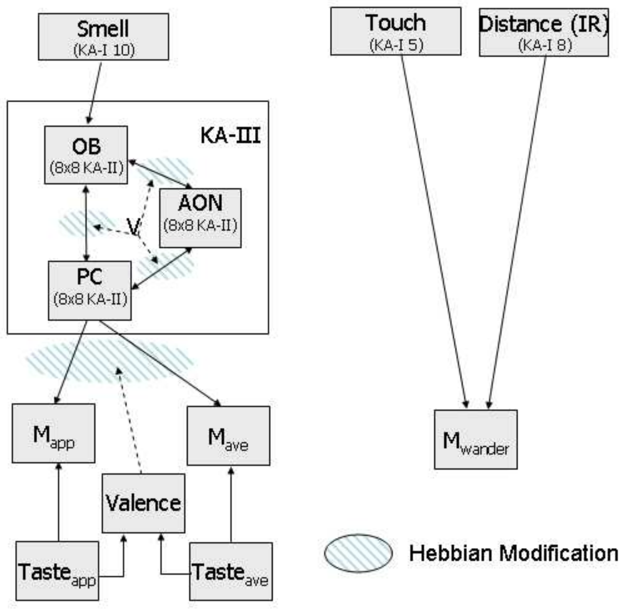


Figure 4.21: Architecture of the neural model used for the appetitive/aversive task. There are four areas which receive direct stimulation from the agents sensors: Smell (using light-source gradients), Touch, Distance (short-distance obstacle sense using infra-red), and a simulated taste ($Taste_{app}$ and $Taste_{ave}$). Touch and distance senses are initially hardwired to produce search behavior if no food source is in range. When a food source is detected, the agent approaches and consumes it (approach and appetitive behavior M_{app}). Some food sources are edible, and some are poisonous. The agent is hardwired to trigger avoidance behavior when a poisonous food source is tasted (M_{ave}). The agent learns to identify poisonous food sources at a distance from smell and trigger avoidance behavior without having to first taste it. We use a simulated olfactory system to learn the good and bad smells. The olfactory simulation consists of an olfactory bulb (OB), anterior olfactory nucleus (AON) and prepyriform cortex (PC). Each of these areas is an 8x8 matrix of KA-II units. The three areas together form a KA-III capable of aperiodic dynamics and the formation of perceptual categories in the manner of biological brains. Weights between the 3 olfactory areas and from the PC to the M_{app} and M_{ave} are modified in response to pain and pleasure signals by Hebbian modification.

4.10.2 Experiment Description

There are four areas which receive direct stimulation from the agents sensors: Smell (using light-source gradients), Touch, Distance (short-distance obstacle sense using infra-red), and a simulated taste ($Taste_{app}$ and $Taste_{ave}$). Touch and distance senses are initially hardwired to produce search behavior if no food source is in range. When a food source is detected, the agent approaches and consumes it (approach and appetitive behavior M_{app}). Some food sources are edible, and some are poisonous. The agent is hardwired to trigger avoidance behavior when a poisonous food source is tasted (M_{ave}). The agent learns to identify poisonous food sources at a distance from smell and trigger avoidance behavior without having to first taste it. We use a simulated olfactory system to learn the good and bad smells. The olfactory simulation is composed of the OB, AON and PC areas. Each of these areas is an 8x8 matrix of KA-II units. The three areas together form a KA-III capable of aperiodic dynamics and the formation of perceptual categories in the manner of biological brains.

The input to the olfactory system are the 10 KA-I units of the sense of smell. These units are stimulated from the 8 light sensors and from two additional signals which are invariant with regards to the type of smell being encountered, edible or poisonous. Simply put, each food source has a distinct and characteristic odor which is detectable by the agent. The light sensors give information on the direction of the smell, and the invariant stimuli provides information that lets the agent form appropriate categories.

The task of the simulated olfactory system is to learn to differentiate between the two basic types of smells in the environment and to trigger appropriate behaviors at a distance, before it actually has to taste the food source. Therefore, we want the olfactory system to become connected in an appropriate way to the appetitive and aversive behaviors, and to learn to override the instinctive approach behavior when it detects that the food source is poisonous.

We use two types of learning in the simulation, Hebbian modification and habituation. Hebbian modification is directed by the valence system (V), attached to the simulated sense of taste. Tasting good food sources causes pleasure signals to be generated, which strengthens the connections (e.g. via Hebbian modification) between active units. Poisonous foods sources cause pain, and a reversal of this effect by weakening connections between active units. The second learning mechanism is simple habituation. During times when the agent can not detect the presence of a salient smell, habituation of the stimulus occurs. This has the effect of lessening the response of the simulated olfactory system to unimportant stimuli (Kozma & Freeman, 2001a).

The expected effect of this simulation is to form two distinct perceptual categories of the environmental smells. These should become strong enough to eventually activate appetitive or aversive motor actions at a distance, before the agent tastes the food source.

4.10.3 Results

In Figures 4.22 and 4.23 we give an example of the change in behavior that results from the agent being exposed to and forming perceptual categories of the smell sense. Figure 4.22 shows activity in the M_{app} and M_{ave} motor units as well as activity in the Taste units before learning has occurred. The agent first approaches an edible food source when it is detected in the environment. The $Taste_{app}$ becomes active when the agent reaches the food source since it is edible, which provides a pleasure signal for learning purposes. Next the agent approaches a poisonous food source. When the agent reaches this source, $Taste_{ave}$ becomes active and a pain signal is generated. This in turn causes aversive M_{ave} motor units to become active.

In Figure 4.23, we show activity in the motor units after the agent has spent some time in the environment learning. Again, the agent first approaches an edible food source, and eventually reaches and consumes it, and therefore $Taste_{app}$ becomes active. Next the agent detects a poisonous food source. Now we see that instead of M_{app} appetitive

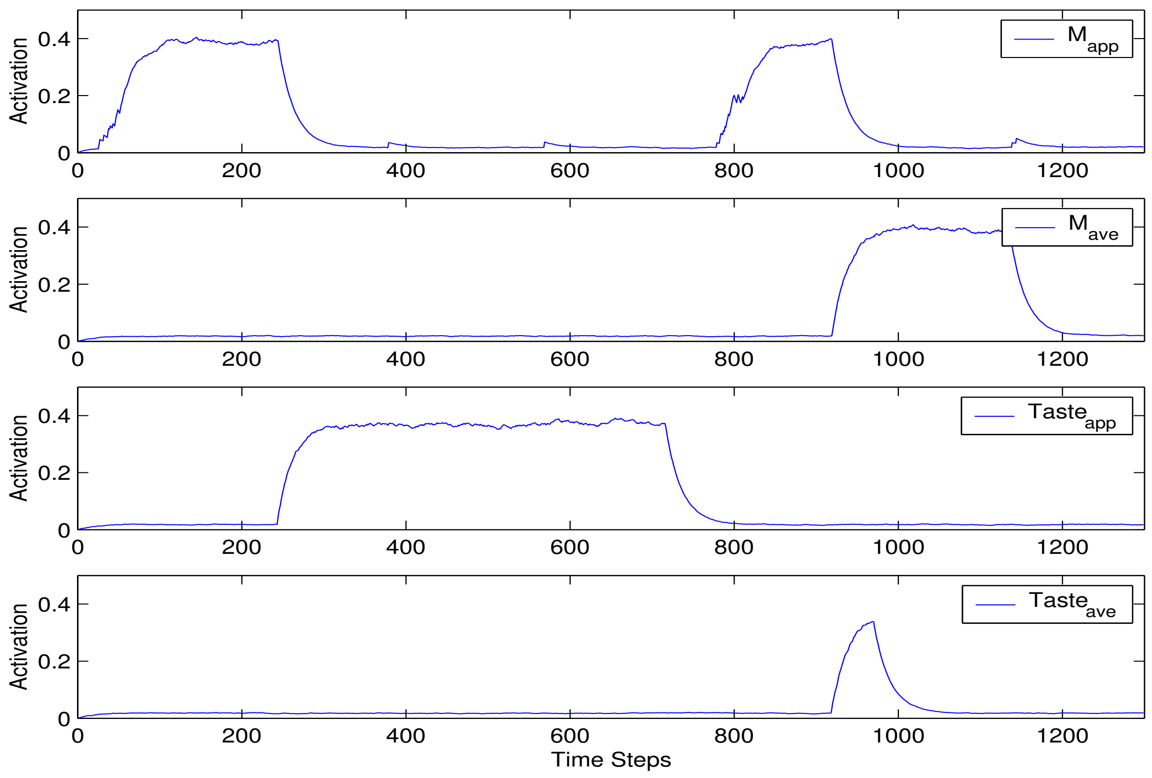


Figure 4.22: Change in motor and taste unit activity before learning. We show time series plots of the activity of a unit in the M_{app}, M_{ave}, $Taste_{app}$ and $Taste_{ave}$ areas. In this figure we show activity before learning has taken place. First the agent approaches and consumes an edible food source. Then the agent approaches and tastes a poisonous food source, causing $Taste_{ave}$ to become active.

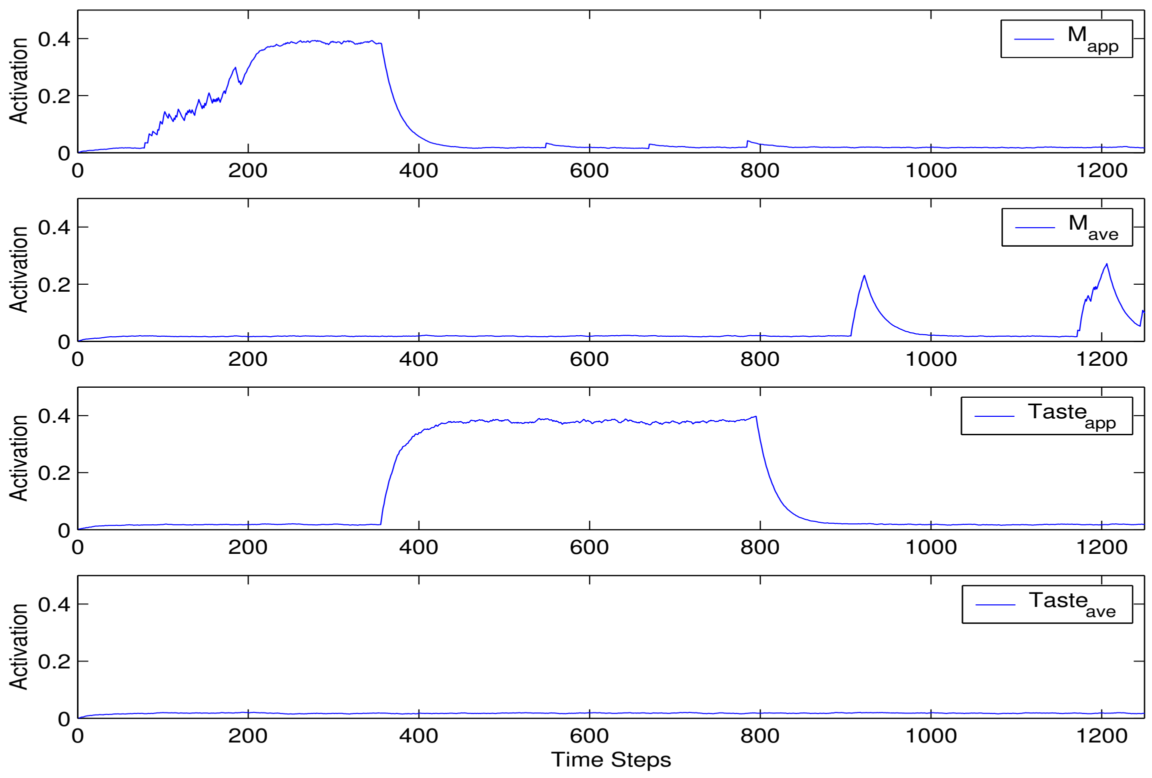


Figure 4.23: Change in motor and taste unit activity in response to learning. We show time series plots of the activity of a unit in the M_{app}, M_{ave}, $Taste_{app}$ and $Taste_{ave}$ areas. In this figure, the same behavior still occurs for edible food sources, which happens first in these time series. But upon detecting a poisonous food source, M_{ave} becomes immediately active and the agent avoids the food source without tasting it.

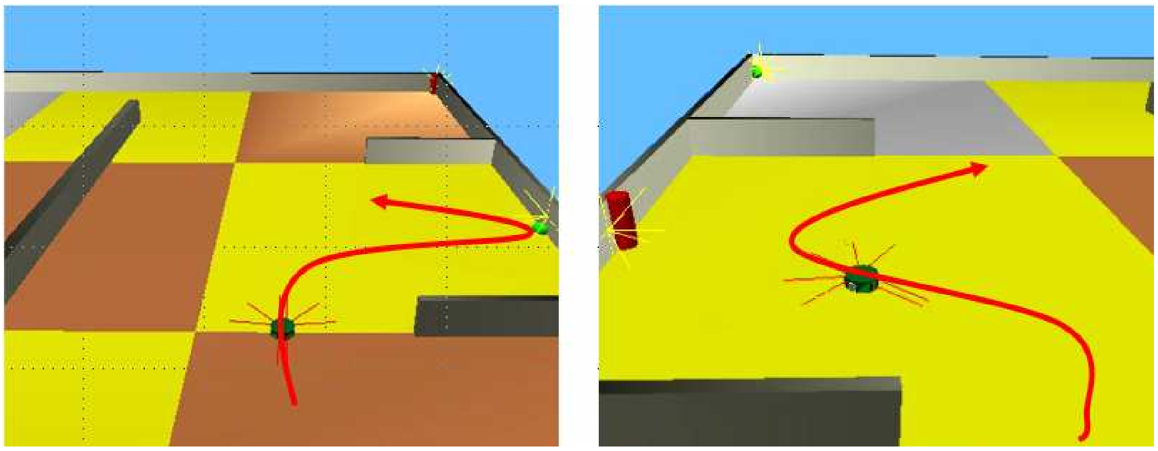


Figure 4.24: Example behavior of the agent in appetitive/aversive task after learning. Left, robot detects edible food source, approaches and consumes it. Right, robot triggers aversive behavior and avoids the poisonous food source before reaching it.

approach behavior becoming active, instead the M_{ave} aversive behavior becomes activated. The agent avoids the poisonous food source, and never tastes it, thus we see no activity on $Taste_{ave}$. In figure 4.24 we show examples of the path of the agent in the environment, after learning, when detecting an edible food source (left) and a poisonous food source (right).

After learning has occurred, the agent is able to discriminate between poisonous and edible food sources at a distance. In Figure 4.25, left, we show an example path of the agent without learning. In this figure the agent approaches all food sources it detects in the environment. The right diagram shows the resulting change in behavior due to learning. In this case, the agent learns to avoid poisonous food sources at a distance, though is behavior is not perfect as can be seen in the fact that the agent still approaches and gathers poisonous food source 3. Both figures show five minutes of simulated activity of the agent in the environment.

We may also measure the efficiency gains that the agent achieves due to learning by determining the number of edible and poisonous food sources it is able to gather during an experiment. In Figures 4.26 and 4.27 we show quantitative results of the formation of

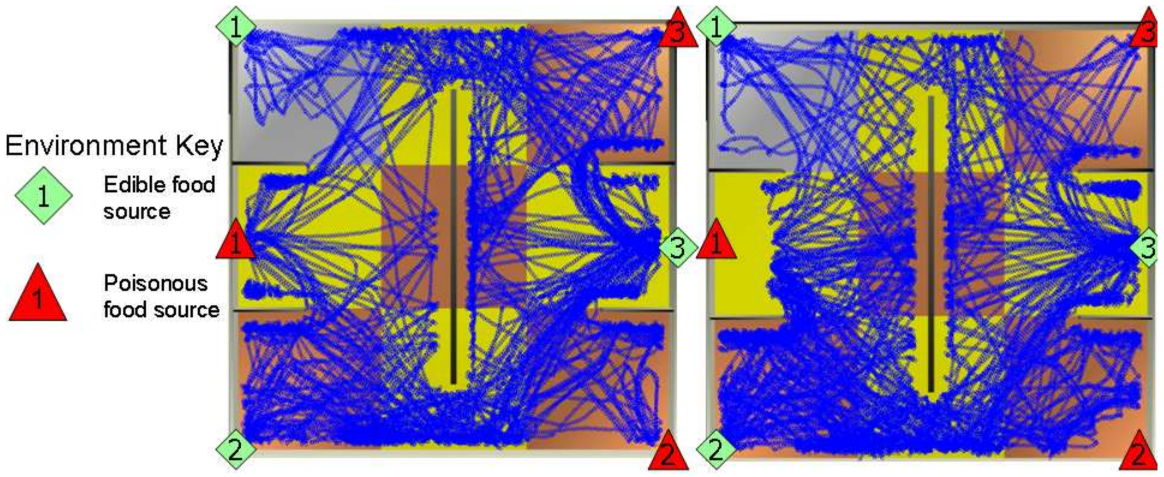


Figure 4.25: Path of robot before and after learning in appetitive/aversive task. Left shows behavior of agent with no learning. The agent does not discriminate between types of food sources and approaches and consumes them all. Right is behavior of agent with learning. The agent learns to discriminate between poisonous and edible food sources and it avoids the former at a distance. Both figures show an example experiment of duration 5 minutes.

Table 4.5: Summary of the Resulting Change in Behavior due to KA-III in Appetitive/Aversive Experiment

No Learning		Learning	
Edible	Poisonous	Edible	Poisonous
59	82	98	40

the edible/poisonous category by the agent. The first figure shows the typical behavior of the agent without learning. In this case it gathers both edible and poisonous food sources, about 70 of each in a simulated 6000 seconds. In Figure 4.27 we see that with learning, the agent is able to gather significantly more edible food sources in the same period while avoiding many of the poisonous ones. The behavioral gains resulting from the KA-III are summarized in Table 4.5 where we show the total edible and poisonous food units gathered over the experiment in both learning and no learning cases.

These results show the quantitative change in behavior that occur from the learning mechanisms using the KA units in the agent. We will next discuss the changes in the KA-III that occur in this experiment in the formation of the edible/poisonous categories.

4.10.4 Discussion

What develops in the simulated olfactory regions are patterns of Amplitude Modulation (AM) which are indicative of meanings formed by the agent. In Figure 4.28 we show examples of the AM patterns formed in the PC region. The left contour map shows the AM pattern formed for edible smells, and the right map for poisonous smells. These contour maps were generated by recording the activity in the 8x8 array of the PC for 50ms, then calculating the amplitude of each of the 64 units for the 50ms (e.g. by using the standard deviation). We then plotted the results as a standard contour map.

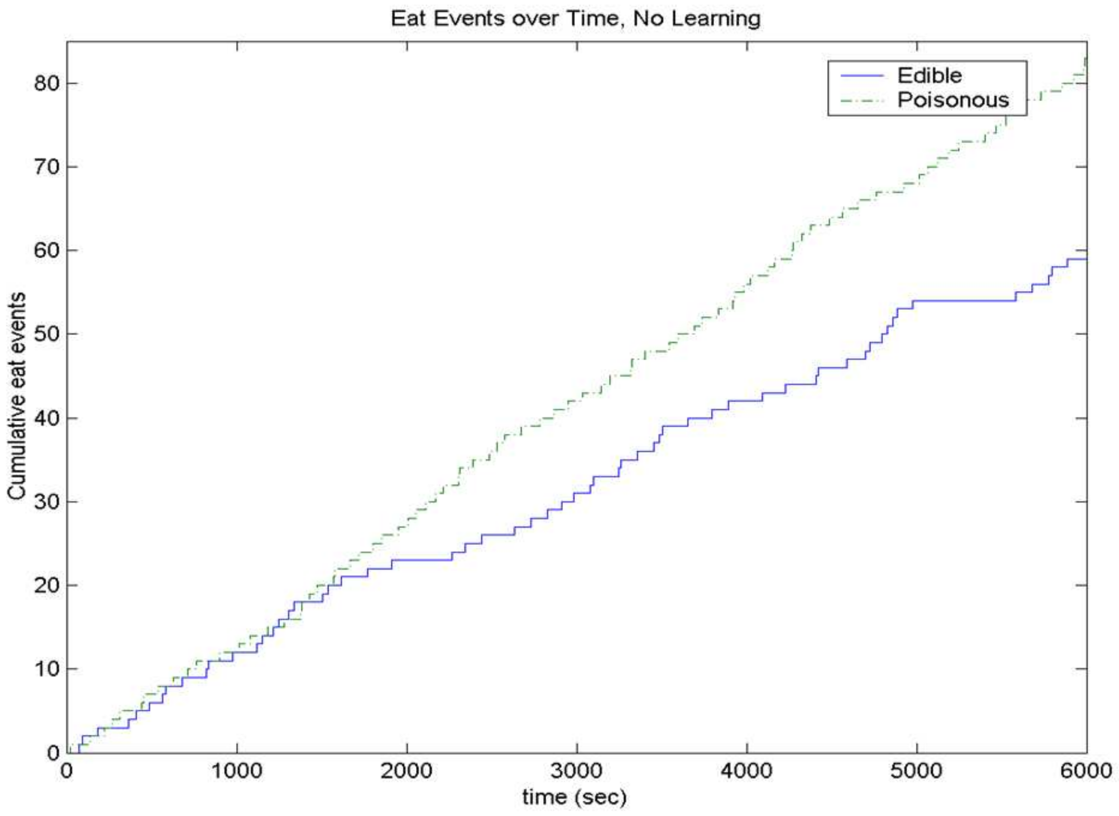


Figure 4.26: This figure shows an example of the performance of the agent in the appetitive/aversive task without learning. We plot the cumulative number of poisonous and edible food sources gathered as a function of time. The results show that without learning the agent gathers around 80 poisonous and 60 edible food sources in 6000 seconds of simulated time.

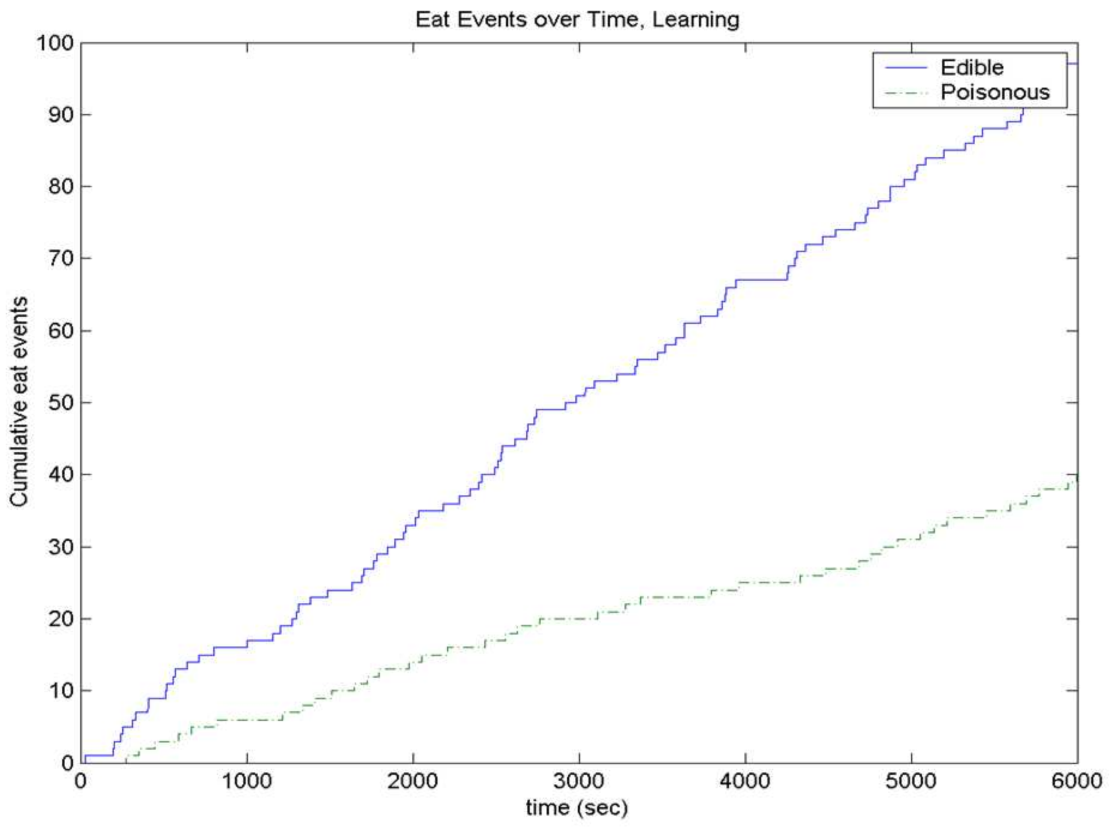


Figure 4.27: This figure shows an example of the resulting performance of the agent in the appetitive/aversive task with learning. Again we plot the cumulative number of poisonous and edible food sources gathered as a function of time. With learning the agent increases significantly the number of edible food sources it gathers, and decreases the number of poisonous ones.

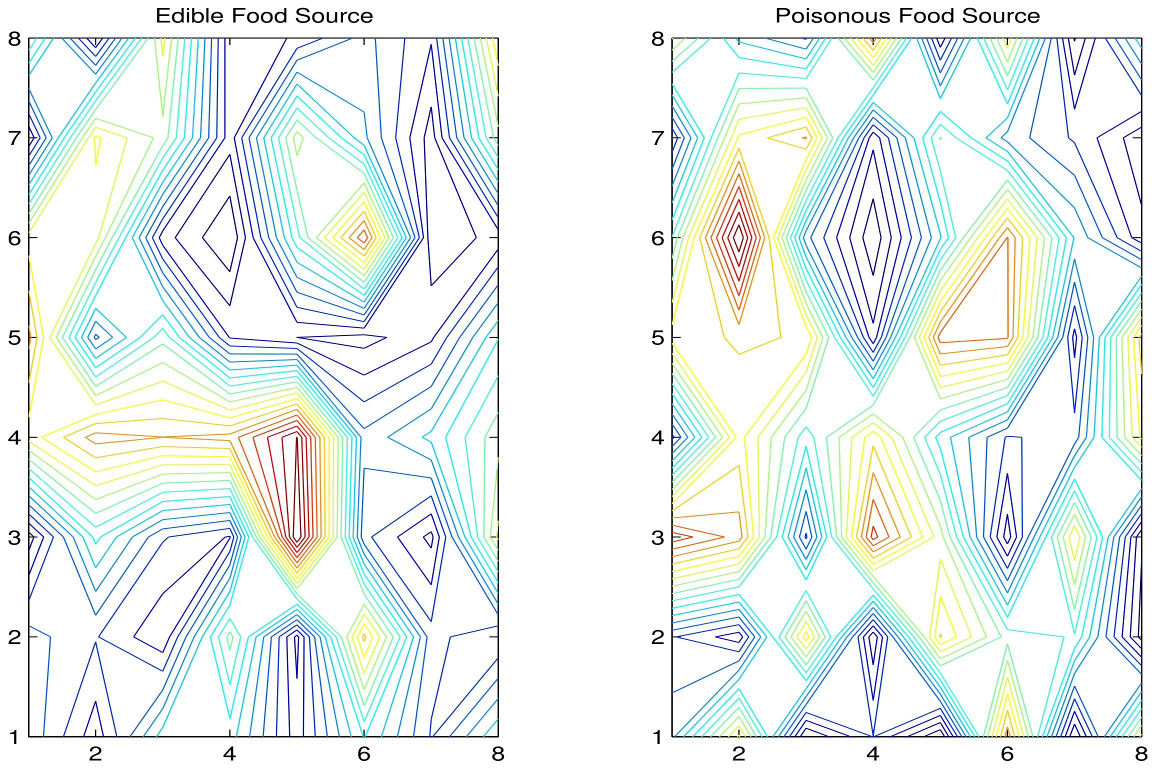


Figure 4.28: Examples of AM patterns formed in response to edible (left) and poisonous (right) smell stimuli. AM patterns in response to like stimuli (e.g. to edible food sources) will be more similar to each other than to other stimuli as measured by Euclidean distance.

The AM patterns thus formed are not static entities. Therefore, they will not be exactly the same between two presentations of the same type of stimuli. However, they do form two distinct categories, and the AM patterns for one type of stimuli (edible) exhibited will be closer to each other, than those formed for the other stimuli (poisonous), as measured by the Euclidean distance between the AM patterns.

In Figure 4.29 we show an example of this type of variance. Here we generate a phase space plot of the activity of one of the units (unit in row 7, col 3) in the PC area, with a 7ms time delay. This phase space plot shows the activity over a 5 second time period. The left side is the activity in the unit when a edible smell is being presented, the right side is for a poisonous smell. In this figure, it is difficult to see any difference, and it appears that both stimuli produce similar dynamics.

However, if you look at where in the phase space the activity occurs for each of the stimuli, you can detect the difference. In Figure 4.30 we show the same plots, but now we only plot the points in the space, and don't connect them with lines. Now you can see that, though the unit visits the same areas of the phase space in both cases, it is more likely to be in one area of the phase space for edible stimuli, and in another for poisonous stimuli. In essence what you see is that unit continues to visit all areas of its chaotic attractor no matter what type of stimuli is occurring. However, it will tend to more actively reside in one part of the attractor in response to a particular stimuli. Different stimuli cause the unit to reside in different wings of the chaotic attractor. This can be seen as differences in the density of the activity of the dynamics in the phase space plots.

In this experiment we showed how AM patterns shaped in a KA-III can be connected to behavior in order to learn to associate a unconditioned stimulus, such as the aversive behavior of the agent, with an conditioned stimuli. In this case the conditioned stimuli is the learned AM patterns formed by the agent that differentiate between edible and poisonous food source stimuli from the smell sensors. We next look at a more complex example of the formation of AM pattern categories in a KA-III.

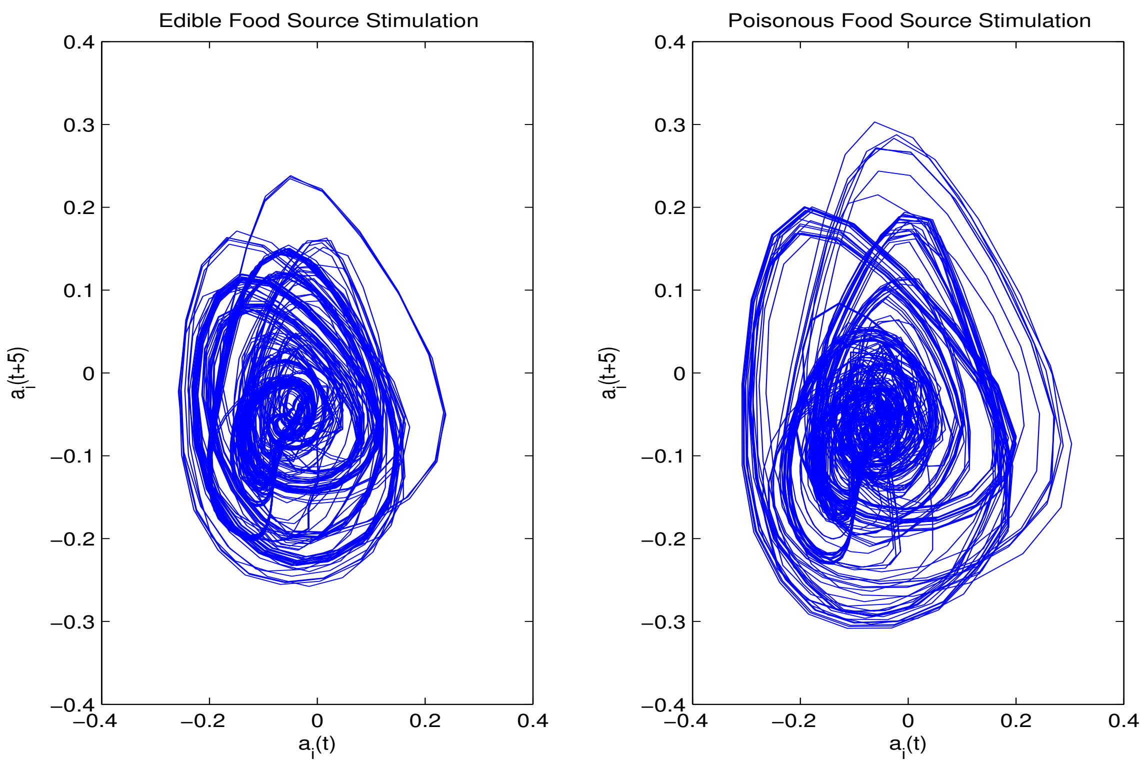


Figure 4.29: Phase space plot of the activity of unit (row 7, col 3) in the PC area. We show the dynamics of the unit plotted against itself with a time delay of 7ms. for a 5 sec. period. Left plot is in response to stimulation by an edible food source, while right plot is in response to a poisonous food source.

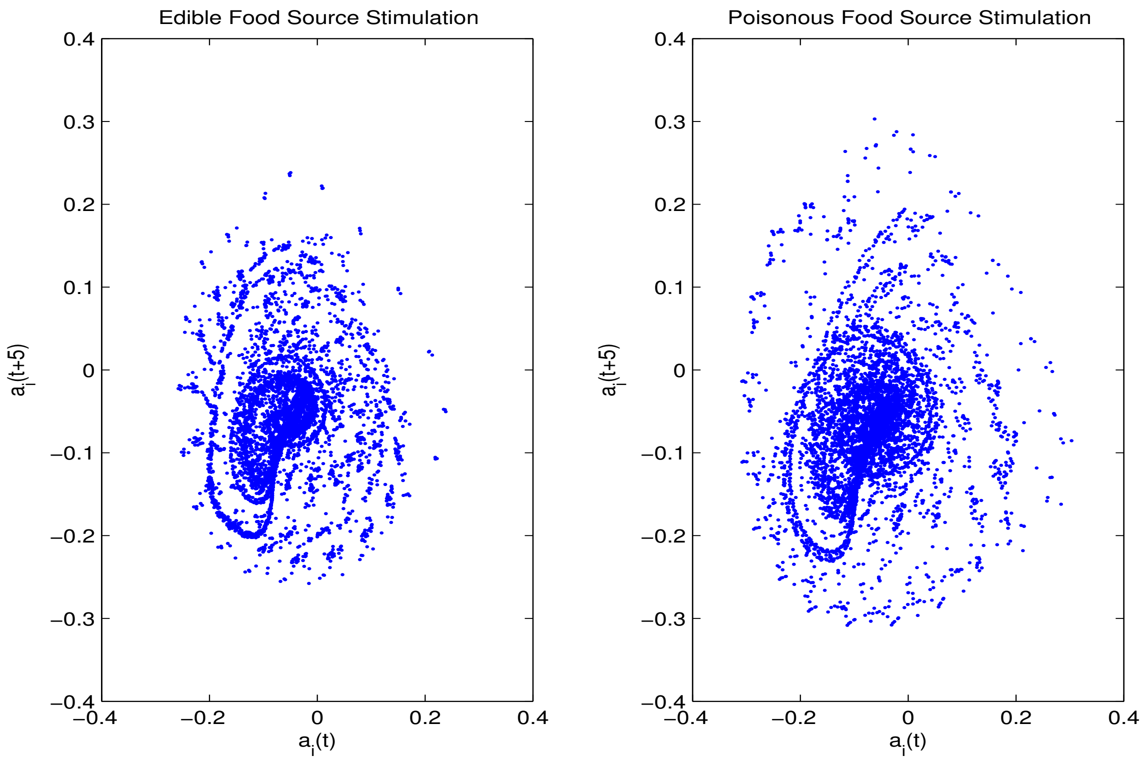


Figure 4.30: The same phase space plot as previous figure, except we only plot the points where the dynamics of the unit occurs. You can see the difference in density of the spaces visited by the unit in response to the different types of stimuli. Left is in response to edible stimuli, and right is in response to poisonous stimuli.

4.11 Amplitude Modulation for Place Cell Formation in a Simulated Hippocampus Using KA

The previous experiment showed the possibility of using the learned AM pattern categories in order to produce behaviors in an autonomous agent. In this experiment we show a more complex example of the formation of AM patterns in response to environmental stimuli. The task in this experiment is to learn environmental locations. Each location will come to be represented by a chaotic attractor in the KA-III dynamics. We will use a simulated hippocampus in this experiment to form place-cell like patterns that represent the recognition by the agent of important locations.

4.11.1 Agent and Environment Setup

According to the current theories of aperiodic dynamics in perceptual mechanisms, perceptual meanings are formed through aperiodic attractors in the spatio-temporal activation of neuronal groups in the perceptual cortex. The same basic mechanisms of aperiodic dynamic in perception may also used by the biological brain in other areas to form memory and behavior producing structures (Kozma et al., 2003). Here we use the basic KA-III architecture to simulate the formation of place cells in the hippocampus of an autonomous agent.

In this experiment, we used the Khepera virtual environment simulator (Michel, 1996). Figure 4.31 (bottom left) shows the morphology of the Khepera agent. The Khepera robot is a simple agent that contains 8 infra-red and 8 light sensors. It has two independently controlled wheels that allow it to move forward, backward, and turn left and right in place. The environment for this experiment is shown in figure 4.31. In the environment we place 8 light sources, which will be used as salient environmental locations (i.e. they can be thought of as good food sources for the agent in the environment). The light sources are detectable to the agent at a distance, and the range where the food source is detectable is indicated in Figure 4.31. In addition to the 8 salient environmental locations, there are 4 landmarks. The landmarks are always detectable to the agent, and it knows the distance and direction to each of the 4 landmarks as part of its orientation sensory information.

The architecture of the simulated hippocampus is shown in Figure 4.32. The portions of the architecture that form the cognitive map of the environment are simulated by a KA-III. These are the CA1, CA2 and CA3 areas, and are based on biological evidence of the structure of the hippocampus. Each of the CA areas contains an 8x8 array of KA-II units (for a total of 64 units in each CA region). Each CA area is connected to the other 2. The interconnection of these 3 CA regions via inhibitory and excitatory feedback forms a KA-III unit. The connections between CA regions will be changed via Hebbian modification.

Orientation beacons are fed into the hippocampal simulation through the DG region (Figure 4.32, left). The DG again contains an 8x8 matrix of KA-0 units. Orientation signals from the 4 landmarks are fed into the DG units. Each of the 4 landmarks has 8 units associated with the direction to the landmark, and 8 units associated with the distance. Directions are broken into 8 cardinal units, North, NorthEast, East, SouthEast, South, SouthWest, West and NorthWest. Units are sensitive to the direction of a particular landmark using a graded response with a normal distribution, instead of a simple 1 unit is active and the others being inactive. Similarly there are 8 cardinal distance values VeryClose, Close, MediumClose, Medium, MediumFar, Far, VeryFar, Distant. Again a graded response with normal distribution is applied to the units. The DG area connects with the CA3 area, and the connections between these areas are also subject to Hebbian modification.

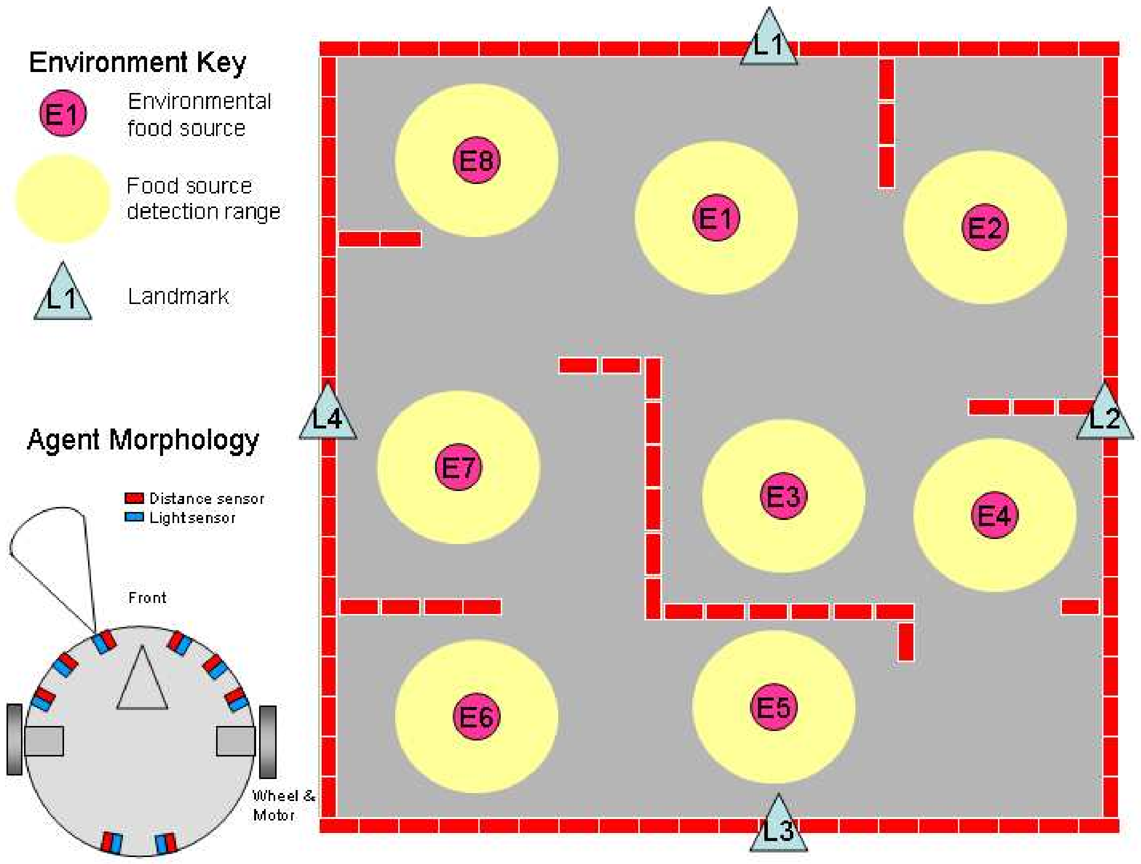


Figure 4.31: Agent morphology (bottom left) and environmental setup for hippocampal simulations using KA-III. The environment contains landmarks, used as allocentric reference points by the agent, and salient environmental locations, such as food sources. The agent is only able to detect the presence of a food source when it is within a certain range.

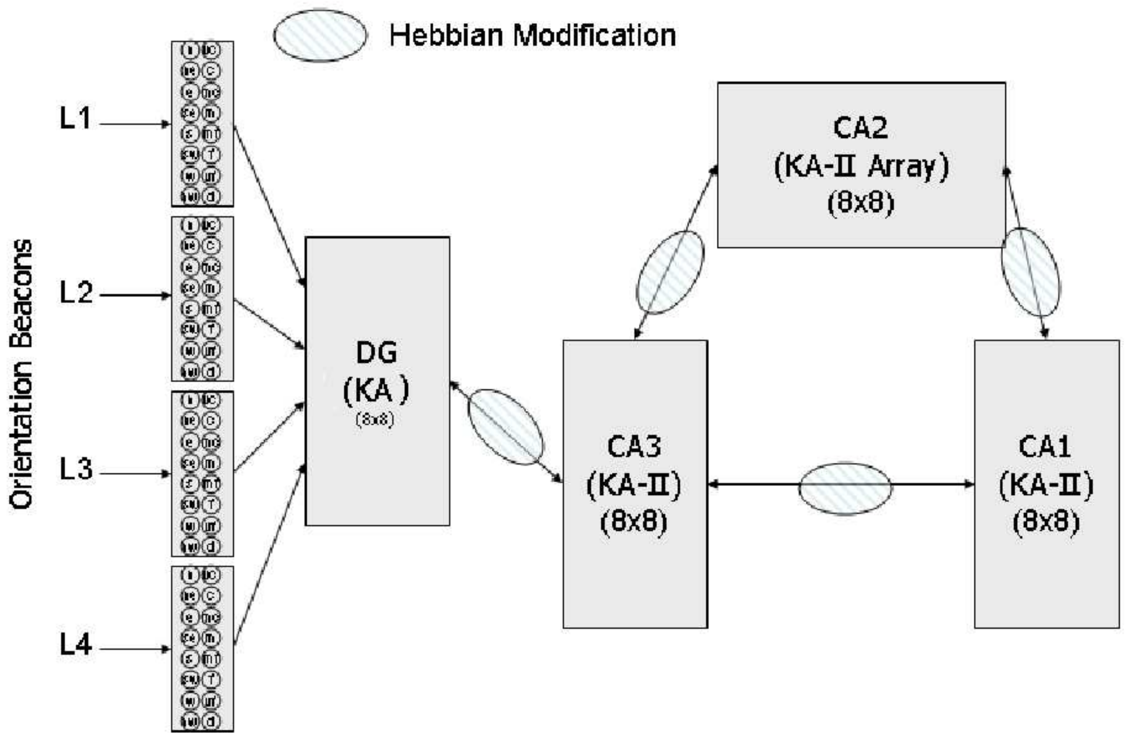


Figure 4.32: Architecture of KA-III hippocampal simulations. CA layers contain 8x8 matrices of KA-II units. Orientation signals enters the KA-III hippocampal simulation through the DG, which contains an 8x8 matrix of single KA-0 units. Hebbian modification is preformed on weights between layers.

4.11.2 Experiment Description

We use two types of learning in the simulation, Hebbian modification and habituation. Hebbian modification only occurs when the robot is within a certain range of a light source. Therefore the light sources provide a certain valence signal that acts as a stimulus to learn environmentally salient locations. When the robot is not within proximity to a light source, no reinforcement signal is produced. During these times habituation of the stimulus occurs. This has the effect of lessening the response of the simulated hippocampus to unimportant regions in the environment (Kozma & Freeman, 2001a).

The expected effect of this stimulation is to form 2 distinct types of dynamical patterns in the CA regions. When the agent is out of range of an environmentally salient location, the dynamics should be in the high-dimensional chaotic state, receptive to input but not indicative of recognizing a salient event. When in range of a light source, the system should transition to a low dimensional attractor, indicative of recognition of the important location. Further, the spatial amplitude modulation patterns in the CA regions upon such recognition should form 8 unique patterns, one for each of the recognized regions.

The agent is allowed to roam in the environment, using a low level mechanisms to produce efficient, but random wandering. The agent roams for some time, 10,000 time steps in our simulations. In our simulation 10 time steps approximates 1 second of real world running time, therefore the totaled simulated time of an experiment is 1000 seconds.

4.11.3 Results

We first look at the amplitude modulation (AM) patterns produced by the hippocampal simulation characteristic of the 8 salient regions in the agents environment. In Figure 4.33 we show examples of the amplitude of the activity of the 8x8 units in the CA3, displayed as contour maps. We next explain these contour maps and how they were produced.

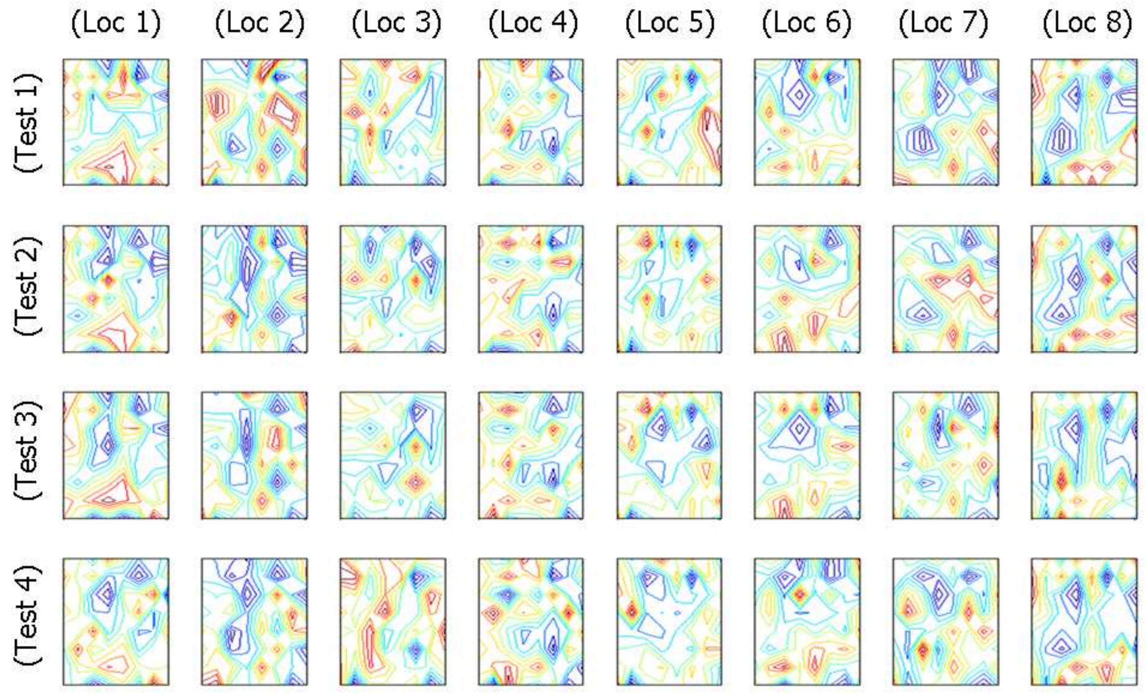


Figure 4.33: Example of AM patterns formed in the simulated CA3 hippocampal region. In this figure we show contour maps of amplitudes of the activity in the 8x8 units in the CA3 region. We show the AM patterns for 4 test cases around each of the 8 salient locations in the environment.

After the agent was allowed to wander in its environment, and Hebbian modification was applied to the simulated hippocampus, we tested the response of the simulated hippocampus in the following manner. We randomly chose 4 points close to each of the 8 environmentally salient regions. We placed the agent at these points in the environment for 1 second where it received the appropriate perceptual inputs (e.g. distance and direction information to the 4 orientation landmarks). We captured the responses of the 8x8 units in the CA3 region during this 1 second of activity. We then calculated the standard deviation of the 8x8 CA3 units during the 1 second of activity to come up with a single matrix of 8x8 values characterizing the amplitude of the activity of each of the units. We plotted these amplitude patterns as contour maps and displayed the results in Figure 4.33.

The results in Figure 4.33 show visually the response of the simulated hippocampus after learning. The CA3 responds with a similar AM patterns for points close to a particular location; for example test points a, b, c and d at location 8 show 2 valleys to the left of the 8x8 CA3 and 3 or so peaks near the bottom of the CA3 region. However, the AM patterns formed at the different locations seem to be different and distinct from one another.

Although the contour maps give evidence of the formation of unique AM patterns in response to environmental locations, we can more clearly see the results by measuring the similarity of the AM patterns to one another. In Table 4.6 we show the results of measuring the Euclidean distance between the 4 points tested at the 8 environmental locations to each other. We show each of the target location/test patterns and which of the other AM patterns (excluding itself) the target was closest to. We treat the AM pattern as a 64 dimensional vector and use simple Euclidean distance to determine closeness of one AM pattern to another. As can be seen, the AM patterns formed in the CA3 region of the KA-III hippocampal simulation form fairly coherent patterns. Patterns around an environmentally salient location tend to be most similar to one another. The

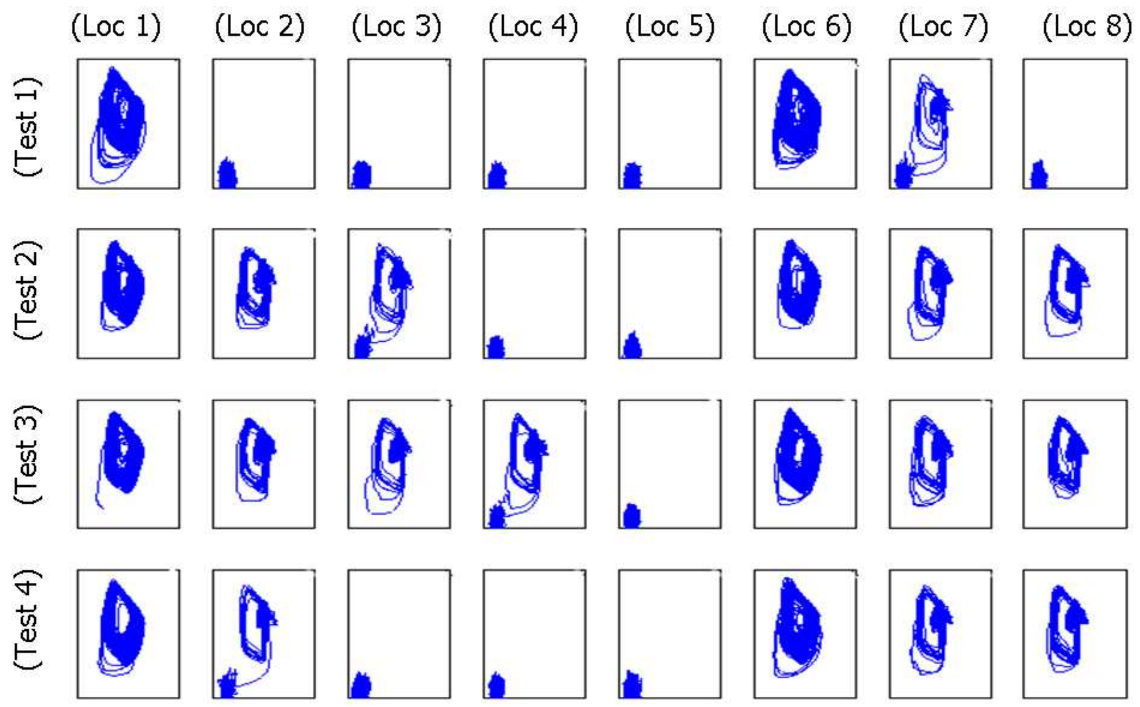


Figure 4.34: State space plot of activity of unit (column 3, row 7) in CA3 region during testing of KA-III hippocampal simulation.

performance is not perfect, however. For example, location 2 test points c and d ended up being closest to AM patterns formed at location 7. The fairly unique formation of AM patterns around the environmental locations can be interpreted as the formation of "place cell" like patterns in the simulated hippocampus.

As another example of the aperiodic attractors formed by the hippocampal simulation, we show a phase space plot of the activity of a single unit in the CA3 region. Figure 4.34 shows a state space representation of the activity of a unit (in column 3, row7) in the CA3 region. This figure plots the activity of the unit against itself with a 12ms time delay. Again we show the activity of the unit in response to 4 tests (rows in the figure) at the 8 environmental locations (columns in the figure).

This figure shows an example of how the KA-III may produce subsymbolic type representations of the input patterns. In this case, the unit seems to have formed at least 3 distinct aperiodic attractors. The unit activity at locations 1 and 4 seem to be similar,

Table 4.6: Results of similarity (distance) measure of CA3 hippocampal AM patterns

Target		Closest		Target		Closest	
Loc	Test	Loc	Test	Loc	Test	Loc	Test
1	a	1	c	5	a	5	b
1	b	1	a	5	b	5	a
1	c	1	a	5	c	5	d
1	d	1	b	5	d	5	c
2	a	2	c	6	a	6	b
2	b	2	c	6	b	6	c
2	c	7	b	6	c	6	b
2	d	7	a	6	d	1	a
3	a	3	d	7	a	7	b
3	b	3	c	7	b	7	a
3	c	3	b	7	c	6	b
3	d	3	a	7	d	7	c
4	a	4	b	8	a	8	d
4	b	4	a	8	b	8	d
4	c	4	a	8	c	8	d
4	d	4	a	8	d	8	b

and similarly attractors for locations 2,3,7,8 and also attractors for locations 4 and 5. Although the attractors produced are not perfectly consistent, they may be capturing some kind of similarity in a portion of the perceptual input between these 3 groups of locations. They therefore are dividing the environment into 3 subsymbolic patterns. Each of the 64 units in the CA3 region, when you observe the state space plots, performs a similar but different partitioning of the input stimuli. The resulting attractors in the 8x8 CA3 region produce the unique AM patterns observed at each of the locations in the environment.

4.11.4 Discussion

In this experiment we have shown that the KA-III is able to successfully form distinct attractors that are correlated with environmental stimulation. The KA-III simulation of the hippocampus resembles the formation of place-cells in biological brains. Peaks of activity in the AM patterns are similar and consistent to one another when the agent is revisiting a previously encountered location. Though we have successfully formed some representation of the agents environment using KA-III, we have not demonstrated how this representation might be used to do a more high-level task, such as path planning and navigation. The next experiment takes a look at the navigation problem in the context of KA-III. However, as will be seen, the full understanding of how AM patterns may come to not only represent the environment, but also be used in goal-directed activity remains unsolved.

4.12 Navigation in a Martian Environment using KA Object Avoidance

In this experiment, we revisit the problem of having the agent navigate through an environment. In the current simulation, however, the agent is given a goal location and it must combine both its obstacle avoidance behaviors to avoid running into things and getting stuck, as well as behavior to try and move closer to the goal location. We use a KA-III to demonstrate a simple goal-navigation in a realistic Martian debris-field landscape.

4.12.1 Agent and Environment Setup

We will now demonstrate the use of the KA-III model to perform goal-oriented navigation in a simulated Martian-like debris field that might be encountered by a Martian planetary rover (Huntsberger, 2001; Tunstel, 2001). We again use the Webots simulation environment (Michel, 2003), which is a physics based 3-d graphical simulation environment for real robotic platforms (see Figure 4.35). The Martian debris-field environment was generated using an algorithm from NASA Ames research center, that characterizes the typical field in terms of the size and distributions of boulders and rocks. We used these statistics to generate a simulated debris field of 20 meters x 20 meters. Though the size and distributions of rocks is typical, the environment is completely flat (2-dimensional). It does not contain any slopes or cliffs that would be found in a more realistic simulation and pose other obstacles for a robotic vehicle.

We are using a different robotic model in this simulation. The robot is a simulation of a Scout IV model robot. It is a 4 wheeled robot but with a similar sensor layout to the Khepera robot. It has 20 infrared sensors distributed around the perimeter of the body that can detect the distance to obstacles in a given direction.

The architecture of the agent for this simulation is the same as used in the appetitive/aversive experiments shown in Figure 4.21. In this experiment the agent is hardcoded to instinctually move and avoid objects, using the agents touch and distance sensors (Figure 4.21, right). A goal location is selected in the environment, which emits a simulated gradient field that can be detected by the agents sensors. We think of the gradient field as a type of chemical gradient, such as might be followed by an animals olfactory system

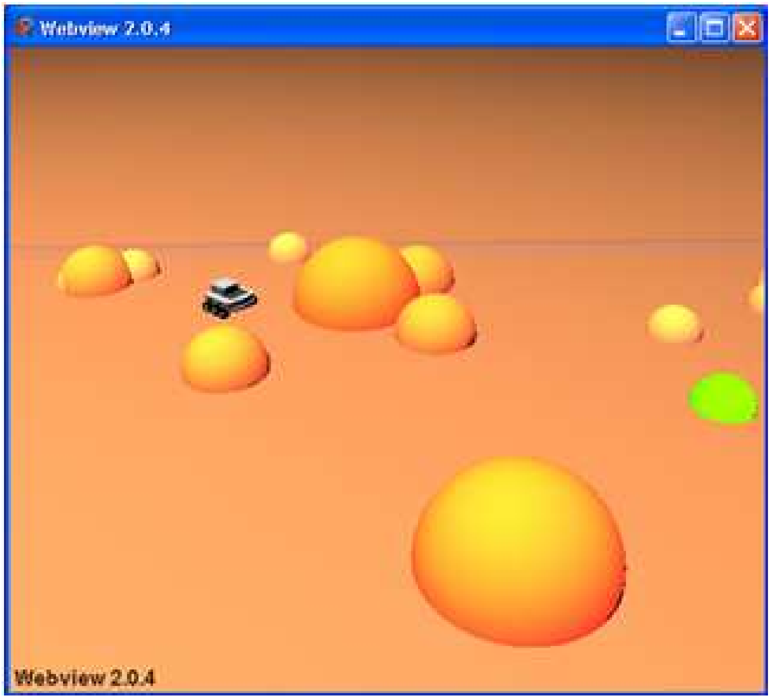


Figure 4.35: The Webots environment used for the goal-directed navigation demonstration. The goal-location is marked in green in this figure.

to locate a target. The gradient is invariant with respect to the obstacles in the environment, therefore the agent must combine local information about obstacles with the global information provided by the gradient to successfully navigate to the target.

4.12.2 Experiment Description

The simulated olfactory system receives sensory input on the strength of the smell signal from sensors positioned around the robots body. The task of the simulated olfactory system is to turn the agent towards the strongest direction of the gradient in order to follow the gradient to its source. The agent is trained by receiving positive reinforcement signals when it moves in a direction that reduces the distance from the agent to the goal. These signals are used to guide the Hebbian modification of weights among units between the simulated perceptual system and from the perceptual to the motor systems. Reinforcement, however, is not received if the olfactory system is interfering with the

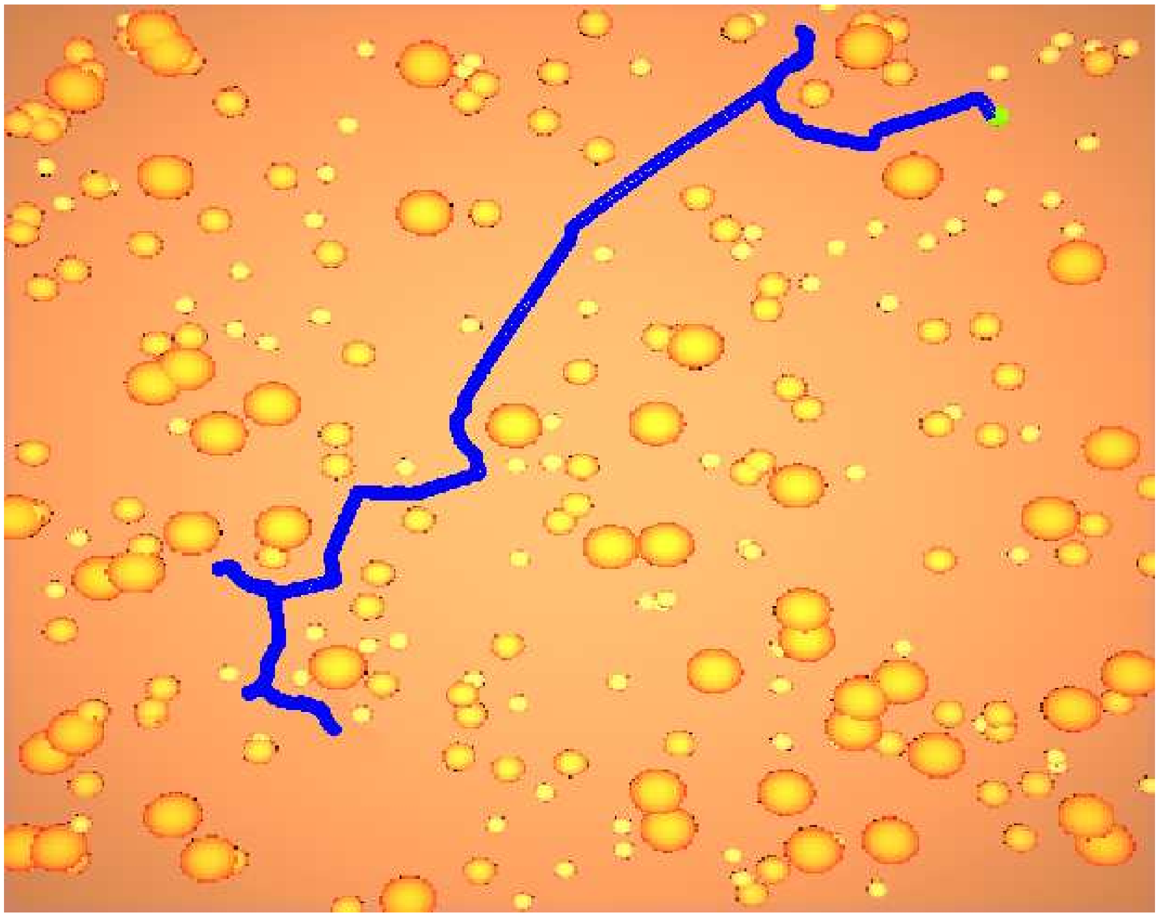


Figure 4.36: Results of a simulated run of the agent navigating to its goal location (upper right, green rock) in the simulated Martian environment. The agent avoids local obstacles while following the environmental gradient to the goal. The environment is Mars-like simulation of a debris field encountered in typical Martian rover landing sites.

object avoidance signals and causing the agent to run into an obstacle.

4.12.3 Results

In Figure 4.36 we show one result of the agents behavior after training. The agent starts in the lower left corner of the debris field and successfully finds a path to the goal location. In this example, the agent combines local information to perform object-avoidance, with global information in order to follow the environmental gradient.

The figure showed one successful navigation of the agent to the goal location. It is necessary, however, to note that with this simple perception/action system the agent can easily get lost or fall into unproductive, repetitive behavior, depending on the starting/goal positions and obstacle placement. For example, sometimes the agent follows the gradient towards the goal only to find that it is blocked by obstacles. The object-avoidance behaviors can cause the agent to turn away and go back the way it came for a time. However the gradient-following may then kick back in and cause the agent to turn back in the direction it has already explored. This repetitive behavior represents a failure of the agent to remember its environment or become bored.

Chapter 5

Conclusion

5.1 Significance

Aperiodic dynamics are a necessary component of intelligent behavior though, they in and of themselves are not alone sufficient. Self-organizing mechanisms of mental structures in biological brains require many types of properties. Hierarchical and mesh synaptic organization, competitive and cooperative processes and self-sustaining autocatalytic loops. In this research we have explored some of these properties using autonomous agent simulations. We have developed a discrete time difference equation version of the K-set neural population model. We have used this model to build multi-layer highly recurrent agent architectures that demonstrate the usefulness and ability of these types of self-organizing aperiodic dynamics in developing mental representations and flexible action selection mechanisms for autonomous agents.

In this work we have developed a discrete time model of neural dynamics in neural networks with excitatory and inhibitory connections. We have built a hierarchy of KA models, starting from the KA-I and KA-II units with fixed point and limit cycle dynamics, to the KA-III model with complex aperiodic dynamics. We have demonstrated the feasibility of generating chaotic oscillations in KA-III and compared the dynamics of the KA model to the original K sets. The developed KA units can be used to build an adaptive autonomous system that explores an environment and generates behavioral strategies in order to solve a given task. The chaotic dynamics described in this research plays a crucial role in the fast and robust operation of the system. The K and KA series of models represent steps to a better understanding of how aperiodic dynamics observed in the cortical systems of biological brains play a part in the production of intelligent behavior.

In this research we have described some of the work on understanding why nonconvergent dynamics appear in biological brains, and how such dynamics might be necessary to the production of intelligent behavior and cognition. Aperiodic dynamics have been shown to be essential in critical brain states involved in the self-organization of meaning and memory. They have been detected in the organization of perceptual meanings in biological brains, and proved useful for pattern classification tasks. Some of the principles of nonconvergent dynamics that appear useful in the organization of neuronal dynamics have been known for awhile, such as homeostatic control through positive and negative feedback. Evidence has shown, however, that brains make use of self-organizing dynamics that go well beyond this simple mechanism.

Aperiodic dynamics, if necessary for perceptual mechanisms, are probably also necessary for the self-organization of memory, goal structure and resource allocation, behavior, and motor systems. The former have been studied for some time in neuro and cognitive science, but we are only beginning to form theories on how they might function in complete systems. Embodiment of a cognitive system, being structurally coupled with the environment, is also necessary for such dynamics to self-organize.

Biological organisms evolved 4 basic functions in becoming intentional agents: “What”, “Where”, “Why” and “How”. These systems are involved in short and long-term memory, and the formation of structures for tracking and realizing the goals and needs of the organism. KIV is a model that incorporates aperiodic dynamics to self-organize patterns of meaning and memory, along with a description of the architecture of the 4 necessary functions, and how phase transitions and attractors in one area influence and affect such attractors in others to produce behavior. We have begun to explore the formation of aperiodic dynamics to self-organize such behavior producing systems.

This research offers a new simplified neural model that will significantly enhance our abilities to explore such highly-recurrent self-organizing neural population models using autonomous agents. We have demonstrated the efficiency and other advantages this simplification has over the original K-set ordinary differential equations. We have also demonstrated the feasibility of using aperiodic dynamics for the action selection mechanisms in autonomous agents using the KA model.

5.2 Future

The simulations presented in this research are still quite limited in demonstrating the principles of intelligence that have been discussed in this work. Future work along these lines needs to expand the models ability to demonstrate higher-level cognitive processes using aperiodic dynamics. For example, we have demonstrated the formation of aperiodic attractors in an autonomous agent to categorize environmental locations. This self-organization of "place cells" within a cognitive map is the first step towards goal-directed navigation abilities in biological organisms.

However, it is still unclear how the next step occurs in such an aperiodic mechanism of cognitive maps. An agent needs to not only form representations of locations in the environment, but use those representations to navigate in the environment to goal locations. We have speculated that in the hippocampus, the aperiodic attractors are hierarchically organized. The results of these areas, that categorize the environmental locations, become the input to other areas that form higher-level associations between connected locations. Conceptually, this hierarchically higher level would form attractors that represent the ability to move between two encountered environmental locations. Because of the recurrent nature of the mechanisms, this attractor both represents the ability to move between locations, and it can also stimulate a sequence of locations in the lower-level area. This means that the organism forms an intention of moving from the current location to a location that it knows is reachable from its current place. It also forms an expectation that, since it has a goal of moving to the new place, it will expect to arrive there and will be surprised if something goes wrong and it can not traverse to the new location along the remembered route.

The formation of higher-level attractors to represent movement between locations is the first step in a goal-directed planning mechanism using aperiodic dynamics. Intentions, expectations and goals play important parts in these mechanisms. The next step is the formation of a chain, or sequential temporal patterns, representing the movements that the organism intends to perform to go from the current location to some goal location in its environment. This temporal sequence has some of the properties of working memory in cognitive systems. Environments that are too complicated to represent and plan paths in more than 7 ± 2 steps are probably hierarchically organized into areas. Therefore the agent can plan how to navigate within an area to get to another area. Once it reaches another area, that attractor space must be switched to become the current context of navigation.

Another future direction indicated by this research is the development of just such a hierarchically organizing mechanism. We speculate that the attractors in the third layer of the KA-III in various areas become input to higher-level KA-IIIs. These higher levels, in the cortex, then form attractors that are organizations of the relationships between the lower-level formed attractors. We could imagine that there could be many such levels of KA-IIIs that form more and more abstract meanings of environmental stimuli for the biological organism. This hierarchical self-organizing mechanism of KA-IIIs would require competitive and cooperative processes. The KA-IIIs would need to compete in some manner. Those that form better abstractions of a lower-level would become more useful and used in the future by the organism. Those that loose out would fade away, and maybe be co-opted by other areas for new uses.

In any case it is clear that there is a large potential for using mechanisms such as the KA model for forming a better understanding of the neural correlates of higher-level cognitive functions. This research has only presented pieces of a complete K-IV limbic system. All of the simulations shown here have used hard-coded valence systems. Another future direction is the development of a more realistic midline forebrain area in an autonomous agent, that is responsible for the goal-formation of such a valence system. The development, in the future, of this and the other mechanisms outlined above using K or KA-set neural population model would represent a great leap in our understanding of the mechanisms of cognition. The brain is a nonlinear complex dynamical system. These highly-recurrent, self-organizing neural population models hold great promise in unlocking the secrets of the brain.

References

Abraham, F. D., Abraham, R. H., Shaw, C. D., & Garfinkel, A. (1990). *A visual introduction to dynamical systems theory for psychology.* Santa Cruz, CA: Aerial Press.

Almássy, N., Edelman, G. M., & Sporns, O. (1998). Behavioral constraints in the development of neuronal properties: A cortical model embedded in a real world device. *Cerebral Cortex*, *8*, 346–361.

Anderson, J. A., Silverstein, J. W., Ritz, S. A., & Jones, R. S. (1977). Distinctive features, categorical perception, and probability learning: Some applications of a neural model. *Psychological Review*, *84*, 413–451.

Baars, B. J. (1988). *A cognitive theory of consciousness.* Cambridge, MA: Cambridge University Press.

Baerends, G. P. (1970). A model of the functional organization of incubation behavior in herring gull. *Behaviour, An International Journal of Comparative Ethology*, *Suppl. 17*, 261–310.

Bak, P., Tang, C., & Wiesenfeld, K. (1987). Self-organized criticality: An explanation of 1/f noise. *Physical Review Letters*, *59*(4), 381–384.

Blumberg, B. M. (1994). Action-selection in hamsterdam: Lessons from ethology. In D. Cliff, P. Husbands, J.-A. Meyer, & S. W. Wilson (Eds.), (pp. 108–117). Cambridge, MA: The MIT Press.

Brooks, R. A. (1990). Elephants don't play chess. *Robotics and Autonomous Systems*, *6*, 3–15.

Brooks, R. A. (1993). A robot that walks: Emergent behaviors from a carefully evolved network. In R. Beer, R. Ritzmann, & T. McKenna (Eds.), *Biological neural networks in invertebrate neuroethology and robotics.* Academic Press.

Clark, A. (1997). *Being there: Putting brain, body, and world together again.* Cambridge, MA: The MIT Press.

Clark, A. (2001). *Mindware: An introduction to the philosophy of cognitive science.* Oxford, NY: Oxford University Press.

Cliff, D., Husbands, P., Meyer, J.-A., & Wilson, S. W. (Eds.). (1994). *From animals to animats 3: Proceedings of the third international conference on simulation of adaptive behavior.* Cambridge, MA: The MIT Press.

Coppinger, R., & Coppinger, L. (2001). *Dogs: A startling new understanding of canine origin, behavior and evolution.* Scribner.

De Landa, M. (1997). *A thousand years of nonlinear history.* New York, NY: Zone Books.

Digney, B. L. (1996). Emergent hierarchical control structures: Learning reactive/hierarchical relationships in reinforcement environments. In P. Maes, M. J. Matarić, J.-A. Meyer, J. Pollack, & S. W. Wilson (Eds.), (pp. 363–372). Cambridge, MA: The MIT Press.

Edelman, G. M., Reeke, G. N., Gall, W. E., Tononi, G., Williams, D., & Sporns, O. (1992). Synthetic neural modeling applied to a real-world artifact. *Proceedings of the National Academy of Science*, *89*, 7267–7271.

Elman, J. L. (1990). Finding structure in time. *Cognitive Science*, *14*, 179–211.

Elman, J. L. (1991). Distributed representations, simple recurrent networks, and grammatical structure. *Machine Learning*, *7*, 195–225.

Franklin, S. P. (1995). *Artificial minds.* Cambridge, MA: The MIT Press.

Franklin, S. P. (1997). Autonomous agents as embodied AI. *Cybernetics and Systems*, *28*(6), 499–520.

Franklin, S. P., & Graesser, A. C. (1997). Is it an agent, or just a program?: A taxonomy for autonomous agents. In *Proceedings of the agent theories, architectures, and languages workshop* (pp. 193–206). Berlin: Springer-Verlag.

Freeman, W. J. (1975). *Mass action in the nervous system.* New York, NY: Academic Press.

Freeman, W. J. (1987). Simulation of chaotic EEG patterns with a dynamic model of the olfactory system. *Biological Cybernetics*, *56*, 139–150.

Freeman, W. J. (1991). The physiology of perception. *Scientific American*, *264*(2), 78–85.

Freeman, W. J. (1995). *Societies of brains: A study in the neuroscience of love and hate.* Hillsdale, NJ: Lawrence Erlbaum Associates.

Freeman, W. J. (1997). Olfactory system: Odorant detection and classification. In D. Amit & G. Parisi (Eds.), *Building blocks for intelligent systems: Brain components as elements of intelligent function* (Vol. 3, pp. 1–1). Academic Press.

Freeman, W. J. (1999a). Consciousness, intentionality and causality. In R. Núñez & W. J. Freeman (Eds.), (pp. 143–172). Bowling Green, OH: Imprint Academic.

Freeman, W. J. (1999b). *How brains make up their minds.* London: Weidenfeld & Nicolson.

Freeman, W. J. (2000). The neurodynamics of intentionality in animal brains may provide a basis for constructing devices that are capable of intelligent behavior. In *NIST workshop on metrics for intelligence: Development of criteria for machine intelligence.* National Institute of Standards and Technology (NIST), Gaithersburg, MD.

Freeman, W. J. (2003). The wave packet: An action potential for the 21st century. *Journal of Integrative Neuroscience.* (in press)

Freeman, W. J., Burke, B. C., & Holmes, M. D. (2003). Aperiodic phase re-setting in scalp EEG of beta-gamma oscillations by state transitions at alpha-theta rates. *Human Brain Mapping*, *19*, 248–272.

Freeman, W. J., & Kozma, R. (2000). Local-global interactions and the role of mesoscopic (intermediate-range) elements in brain dynamics. *Behavioral and Brain Sciences*, *23*(3), 401.

Freeman, W. J., Kozma, R., & Werbos, P. J. (2000). Biocomplexity: Adaptive behavior in complex stochastic dynamical systems. *BioSystems*, *59*, 109–123.

Freeman, W. J., & Shimoide, K. (1994). New approaches to nonlinear concepts in neural information processing: Parameter optimization in a large-scale, biologically plausible corticle network. In Zornetzer (Ed.), *An introduction to neural and electronic networks* (pp. 119–137). Academic Press.

Gibson, J. J. (1979). *The ecological approach to visual perception.* Houghton Mifflin.

Gleick, J. (1987). *Chaos: Making a new science.* New York, NY: Viking.

Graesser, A. C., Person, N., Harter, D., & The Tutoring Research Group. (2001). Teaching tactics and dialog in AutoTutor. *International Journal of Artificial Intelligence in Education*, *12*(3), 257–279.

Graesser, A. C., VanLehn, K., Rosé, C., Jordan, P., & Harter, D. (2001). Intelligent tutoring systems with conversational dialogue. *AI Magazine*, *22*(4), 39–51.

Graesser, A. C., Wiemer-Hastings, P., Wiemer-Hastings, K., Harter, D., Person, N., & The Tutoring Research Group. (2000). Using latent semantic analysis to evaluate the contributions of students in AutoTutor. *Interactive Learning Environments*, *8*(2), 129–148.

Grossberg, S. (1980). How does a brain build a cognitive code? *Psychological Review*, *87*, 1–51.

Harter, D. (2001). Ontogenetic development of skills, strategies and goals for autonomously behaving systems. In *Proceedings of the 5th world multi-conference on systemics, cybernetics and informatics (SCI 2001)* (pp. 178–181). Orlando, FL.

Harter, D., Graesser, A. C., & Franklin, S. P. (2001). Bridging the gap: Dynamics as a unified view of cognition. *Behavioral and Brain Sciences*, *24*(1), 45–46.

Harter, D., & Kozma, R. (2001a). Models of ontogenetic development for autonomous adaptive systems. In *Proceedings of the 23rd annual conference of the cognitive science society* (pp. 405–410). Edinburgh, Scotland.

Harter, D., & Kozma, R. (2001b). Ontogenetic development of behavior for simple tasks. In *Proceedings of the artificial intelligence and soft computing conference (ASC 2001)* (pp. 401–407). Cancun, Mexico.

Harter, D., & Kozma, R. (2001c). Task environments for the dynamic development of behavior. *Lecture Notes in Computer Science*, *2074*, 300–306.

Harter, D., & Kozma, R. (2001d). Task environments for the dynamic development of behavior. In *Proceedings of the intelligent systems design and applications 2001 workshop (ISDA 2001)* (pp. 300–309). San Francisco, CA.

Harter, D., & Kozma, R. (2002). Simulating the principles of chaotic neurodynamics. In *Proceedings of the 6th world multi-conference on systemics, cybernetics and informatics (SCI 2002)* (Vol. XIII, pp. 598–603). Orlando, FL.

Harter, D., & Kozma, R. (2003). Nonconvergent dynamics and cognitive systems. *Cognitive Science*. (submitted)

Harter, D., & Kozma, R. (2004a). Aperiodic dynamics and the self-organization of cognitive maps in autonomous agents. In *Proceedings of 17th international florida artificial intelligence research society conference (FLAIRS)* (pp. 424–429). Miami Beach, FL.

Harter, D., & Kozma, R. (2004b). Aperiodic dynamics for appetitive/aversive behavior in autonomous agents. In *Proceedings of the 2004 ieee international conference on robotics and automation (ICRA)* (pp. 2147–2152). New Orleans, LA.

Harter, D., & Kozma, R. (2004c). Aperiodic neurodynamics using a simplified k-set neural population model. In *Proceedings of the 2004 international joint conference on neural networks (IJCNN'04).* Budapest, Hungry. (accepted)

Harter, D., & Kozma, R. (2004d). Chaotic neurodynamics for autonomous agents. *IEEE Transactions on Neural Networks.* (submitted)

Harter, D., & Kozma, R. (2004e). Complex systems approaches to emergent goal formation in cognitive agents. In *Proceedings of the 26th annual meeting of the cognitive science society (CogSci 2004).* Chicago, IL. (submitted)

Harter, D., & Kozma, R. (2004f). Complex systems approaches to the ontogenetic development of behavior. In *1st intelligent systems technical conference of the american institute of aeronautics and astronautics AIAA.* Chicago, IL. (submitted)

Harter, D., & Kozma, R. (2004g). The dynamics of intentional behavior. In *Symposium proposal proceedings of the 26th annual meeting of the cognitive science society (CogSci 2004).* Chicago, IL. (submitted)

Harter, D., & Kozma, R. (2004h). Navigation and cognitive map formation using aperiodic neurodynamics. In *From animals to animats 8: The eighth international conference on the simulation of adaptive behavior (SAB'04).* Los Angeles, CA. (accepted)

Harter, D., & Kozma, R. (2004i). Self-organization of cognitive maps in autonomous agents using an aperiodic neural population model. *International Journal of Artificial Intelligence Technology.* (in progress)

Harter, D., Kozma, R., & Achunala, S. (2003). Constraints and the dynamic mechanisms of behavior generation. *Dynamical Psychology.* (submitted)

Harter, D., Kozma, R., & Franklin, S. P. (2001a). Models of ontogenetic development: The dynamics of learning. In *Proceedings of the 2001 learning workshop* (p. 37). Snowbird, UT.

Harter, D., Kozma, R., & Franklin, S. P. (2001b). Ontogenetic development of skills, strategies and goals for autonomously behaving systems. In *Proceedings of the fifth international conference on cognitive and neural systems (CNS 2001)* (p. 18). Boston, MA.

Haugeland, J. (1997). What is mind design? In J. Haugeland (Ed.), *Mind design II* (pp. 1–28). Cambridge, MA: The MIT Press.

Hendriks-Jansen, H. (1996). *Catching ourselves in the act: Situated activity, interactive emergence, evolution and human thought.* Cambridge, MA: The MIT Press.

Hilborn, R. C. (1994). *Chaos and nonlinear dynamics: An introduction for scientists and engineers.* Oxford, NY: Oxford University Press.

Hopfield, J. J. (1982). Neuronal networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Science*, *81*, 3058–3092.

Huntsberger, T. (2001). Biologically inspired autonomous rover control. *Autonomous Robots*, *11*, 341–346.

Kauffman, S. A. (1993). *The origins of order: Self-organization and selection in evolution.* Oxford, NY: Oxford University Press.

Kauffman, S. A. (1995). *At home in the universe: The search for the laws of self-organization and complexity.* Oxford, NY: Oxford University Press.

Kauffman, S. A. (2000). *Investigations.* Oxford, NY: Oxford University Press.

Kelso, J. A. S. (1995). *Dynamic patterns: The self-organization of brain and behavior.* Cambridge, MA: The MIT Press.

Kohonen, T. (1972). Correlation matix memories. *IEEE Transaction on Computers*, *C-21*, 353–359.

Kozma, R. (2001). Fragmented attractor boundaries in the KIII model of sensory information processing - evidence of cantor encoding in cognitive processes. *Behavioral and Brain Sciences*, *24*(5).

Kozma, R., Alvarado, M., Rogers, L., Lau, B., & Freeman, W. J. (2001). Emergence of un-correlated common-mode oscillations in the sensory cortex. *Neurocomputing*, *38-40*, 747–755.

Kozma, R., & Freeman, W. J. (1999). A possible mechanism for intermittent oscillations in the KIII model of dynamic memories - the case study of olfaction. In *Proceedings IJCNN 1999* (pp. 52–57).

Kozma, R., & Freeman, W. J. (2000). Encoding and recall of noisy data as chaotic spatio-temporal memory patterns in the style of the brains. In *Proceedings of the IEEE/INNS/ENNS international joint conference on neural networks (IJCNN'00)* (pp. 5033–5038). Como, Italy.

Kozma, R., & Freeman, W. J. (2001a). Chaotic resonance - methods and applications for robust classification of noisy and variable patterns. *International Journal of Bifurcation and Chaos*, *11*(6), 1607–1629.

Kozma, R., & Freeman, W. J. (2001b). Classification of EEG patterns using nonlinear dynamics and identifying chaotic phase transitions. *CNS*2001 Special Issue of Neurocomputing.*

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), 125–145.

Kozma, R., Freeman, W. J., & Erdi, P. (2002). The KIV model - nonlinear spatio-temporal dynamics of the cortical-hippocampal system. In *Proceedings of the 2002 computational neuroscience conference CNS*2002.* Chicago, IL.

Kozma, R., Freeman, W. J., & Erdi, P. (2003). The KIV model - nonlinear spatio-temporal dynamics of the primordial vertebrate forebrain. *Neurocomputing*, *52-54*, 819–826.

Kozma, R., Harter, D., & Achunala, S. (2002). Action selection under constraints:

Dynamic optimization of behavior in machines and humans. In *Proceedings of the IEEE/INNS/ENNS international joint conference on neural networks (IJCNN'02)* (pp. 2574–2579). Washington, DC.

Kozma, R., Harter, D., & Franklin, S. P. (2001). Self-organizing ontogenetic development for autonomous adaptive systems (SODAS). In *Proceedings of the international joint conference on neural networks (IJCNN 2001)* (pp. 633–637). Washington, DC.

Laird, J. E., Newell, A., & Rosenbloom, P. S. (1987). SOAR: An architecture for general intelligence. *Artificial Intelligence*, *33*, 1–64.

Langton, C. G. (Ed.). (1995). *Artificial life: An overview.* Cambridge, MA: The MIT Press.

Lorenz, K. (1957). The nature of instinct: The conception of instinctive behaviour. In C. H. Shiller & K. S. Lashley (Eds.), *Instinctive behaviour: The development of a modern concept.* International University Press Press.

Maes, P., Matarić, M. J., Meyer, J.-A., Pollack, J., & Wilson, S. W. (Eds.). (1996). *From animals to animats 4: Proceedings of the fourth international conference on simulation of adaptive behavior.* Cambridge, MA: The MIT Press.

Matarić, M. J. (1991). Navigating with a rat brain: A neurobiologically-inspired model for robot spatial representation. In J.-A. Meyer & S. W. Wilson (Eds.), (pp. 169–175). Cambridge, MA: The MIT Press.

Matarić, M. J. (1995). Integration of representation into goal-driven behavior-based robots. In L. Steels & R. Brooks (Eds.), (pp. 165–186). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

McCulloch, W. S., & Pitts, W. (1943). A logical calculus of the ideas immanent in nervous activity. *Bulletin of Mathematical Biophysics*, *5*, 115–133.

McGaugh, J. L., Weinberger, N., & Lynch, G. (Eds.). (1992). *Brain organization and memory: Cells, systems, and circuits.* Oxford, NY: Oxford University Press.

Meyer, J.-A., Roitblat, H. L., & Wilson, S. W. (Eds.). (1993). *From animals to animats 2: Proceedings of the second international conference on simulation of adaptive behavior.* Cambridge, MA: The MIT Press.

Meyer, J.-A., & Wilson, S. W. (Eds.). (1991). *From animals to animats: Proceedings of the first international conference on simulation of adaptive behavior.* Cambridge, MA: The MIT Press.

Michel, O. (1996). *Khepera simulator package version 2.0.* Downloaded from WWW at http://wwwi3s.unice.fr/ om/khep-sim.html. (Freeware mobile robot simulator written at the University of Nice Sophia-Antipolis)

Michel, O. (2003). *Webots v4.0 3-d physics based mobile robot simulator.* www.cyberbotics.com.

Miller, C. S., & Laird, J. E. (1996). Accounting for graded performance within a discrete search framework. *Cognitive Science*, *20*(4), 499–537.

Newell, A. (1980). Physical symbol systems. *Cognitive Science*, *4*, 135-183.

Newell, A. (1990). *Unified theories of cognition.* Cambridge, MA: Harvard University Press.

Newell, A., & Simon, H. A. (1972). *Human problem solving.* Englewood Cliffs, NJ: Prentice-Hall.

Newell, A., & Simon, H. A. (1976). Computer science as empirical inquiry: Symbols and search. *Communications of the Association for Computing Machinery*, *19*, 113–126.

Nunez, P. L. (2000). Toward a quantitative description of large scale neocortical dynamic function and EEG. *Behavioral and Brain Sciences*, *32*(3), 371-437.

Núñez, R., & Freeman, W. J. (Eds.). (1999). *Reclaiming cognition: The primacy of action, intention and emotion.* Bowling Green, OH: Imprint Academic.

Oyama, S. (1985). *The ontogeny of information: Developmental systems and evolution.* Cambridge, MA: Cambridge University Press.

Pfeifer, R., Blumberg, B. M., Meyer, J.-A., & Wilson, S. W. (Eds.). (1998). *From animals to animats 5: Proceedings of the fifth international conference on simulation of adaptive behavior.* Cambridge, MA: The MIT Press.

Pfeifer, R., & Scheier, C. (1998). *Understanding intelligence.* Cambridge, MA: The MIT Press.

Port, R. F., & van Gelder, T. (Eds.). (1995). *Mind as motion: Explorations in the dynamics of cognition.* Cambridge, MA: The MIT Press.

Rosenblatt, F. (1958). The perceptron: A probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*, 386–408.

Rumelhart, D. E., McClelland, J. L., & The PDP Research Group. (1986). *Parallel distributed processing: Explorations in the microstructure of cognition.* Cambridge, MA: MIT Press.

Schiff, S. J., Jerger, K., Duong, D. H., Chang, T., Spano, M. L., & Ditto, W. L. (1994). Controlling chaos in the brain. *Nature*, *370*, 615–620.

Shimoide, K., Greenspon, M. C., & Freeman, W. J. (1993). Modeling of chaotic dynamics in the olfactory system and application to pattern recognition. In F. H. Eeckman (Ed.), *Neural systems analysis and modeling* (p. 365-372). Boston: Kluwer.

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.

Solé, R., & Goodwin, B. (2000). *Signs of life: How complexity pervades biology.* New York, NY: Basic Books.

Sporns, O., Almássy, N., & Edelman, G. M. (1999). Plasticity in value systems and its role in adaptive behavior. *Adaptive Behavior*, *7*(3-4).

Steels, L., & Brooks, R. (Eds.). (1995). *The artificial life route to artificial intelligence: Building embodied, situated agents.* Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Thelen, E. (1995). Time-scale dynamics and the development of an embodied cognition. In R. F. Port & T. van Gelder (Eds.), (pp. 69–100). Cambridge, MA: The MIT Press.

Thelen, E., & Smith, L. B. (1994). *A dynamic systems approach to the development of cognition and action.* Cambridge, MA: The MIT Press.

Tinbergen, N. (1951). *The study of instinct.* Oxford: Oxford University Press.

Trappenberg, T. P. (2002). *Fundamentals of computational neuroscience.* Oxford, NY: Oxford University Press.

Tsuda, I. (2001). Towards an interpretation of dynamic neural activity in terms of chaotic dynamical systems. *Behavioral and Brain Sciences*, *24*(4), 793–847.

Tsuda, I., & Yamaguchi, A. (1998). Singular-continuous nowhere differentiable attractors in neural systems. *Neural Networks*, *11*, 927–937.

Tunstel, E. (2001). Ethology as an inspiration for adaptive behavior synthesis in autonomous planetary rovers. *Autonomous Robots*, *11*, 333–339.

Turing, A. M. (1950). Computing machinery and intelligence. *Mind*, *59*, 433–460.

Verschure, P. F. M. J., Kröse, B., & Pfeifer, R. (1992). Distributed adaptive control: The self-organization of behavior. *Robotics and Autonomous Systems*, *9*, 181–196.

Verschure, P. F. M. J., & Voegtlin, T. (1999). A bottom-up approach towards the acquisition, retention, and expression of sequential representations: Distributed adaptive control III. *Neural Networks*, *11*, 1531–1549.

Verschure, P. F. M. J., Wray, J., Sporns, O., Tononi, G., & Edelman, G. M. (1995). Multilevel analysis of classical conditioning in a behaving real world artifact. *Robotics and Autonomous Systems*, *16*, 247–265.

Von Neumann, J. (1958). *The computer and the brain.* New Haven, CT: Yale University Press.

Werbos, P. J. (1974). *Beyond regression: New tools for prediction and analysis in the behavioral sciences.* Ph.D. thesis, Harvard, Cambridge, MA.

Wiemer-Hastings, P., Graesser, A. C., & Harter, D. (1998). The foundations and architecture of autotutor. *Lecture Notes in Computer Science*, *1452*, 334–340.

Wiener, N. (1965). *Cybernetics, second edition: or the control and communication in the animal and the machine.* Cambridge, MA: The MIT Press.

Wolf, A., Swift, J. B., Swinny, H. L., & Vastano, J. A. (1985). Determining Lyapunov exponents from a time series. *Physica D*, *16*, 285–317.

Appendixes

Appendix A

Examples of Dynamics Generated by KA-III

Table A.1: KA-III Example Dynamics, KA-II Group Parameters Used

	Group 1 (G1)				Group 2 (G2)				Group 3 (G3)			
Ex	w_{ee}	w_{ei}	w_{ie}	w_{ii}	w_{ee}	w_{ei}	w_{ie}	w_{ii}	w_{ee}	w_{ei}	w_{ie}	w_{ii}
A1	1.8539	0.9285	0.2457	0.1184	0.6418	1.4336	0.7366	1.1069	1.6998	0.8333	1.5197	0.1930
A2	0.46480	0.8973	1.8498	1.2019	0.7049	1.0898	1.1358	0.5874	0.1754	0.8623	1.5842	1.0818
A3	0.6684	1.1985	1.0421	0.6167	1.2659	1.0681	1.8641	1.7487	1.4308	0.5382	0.7110	0.2065
A4	1.2485	0.3904	1.1079	0.3422	0.2656	0.2863	0.4529	0.1850	1.0479	1.5745	1.3749	1.6877
A5	1.8458	1.0427	1.6773	1.6976	1.2502	0.3607	1.6931	0.3359	1.5196	0.4884	0.5886	0.8103
A6	0.2468	0.8283	0.1295	0.5662	1.2685	1.1039	0.6688	0.4936	0.9206	1.3986	1.2810	0.6570
A7	0.6249	0.2756	1.5805	1.4084	0.4860	0.2096	1.4960	1.2402	0.1028	1.7408	1.3190	1.8534
A8	0.8309	0.9594	0.8897	1.8855	1.4115	1.7030	1.8726	0.1905	1.8049	1.3210	0.5288	0.4377
A9	1.8554	1.5698	1.5603	1.8839	0.9677	1.2772	0.1586	0.4308	0.9152	1.8163	1.2721	0.9492
A10	0.2142	1.7337	0.2034	0.8594	0.2786	1.7221	0.8588	1.2360	0.8179	1.0855	1.0425	0.7699
A11	1.3517	1.6774	1.4712	1.0467	0.9639	0.6893	1.3281	1.2880	1.7763	0.2403	1.6246	0.1696
A12	1.0601	1.5978	0.2336	1.3941	0.2995	0.8579	1.3703	1.2026	1.4996	0.6674	0.8886	0.3025
A13	0.1171	1.1761	1.5069	1.8597	1.3882	1.1851	0.3202	0.3331	0.9386	0.1480	0.1196	1.8940
A14	1.7663	0.4057	1.8795	1.1570	0.3627	1.8031	1.7456	1.5699	1.5251	0.4369	0.1737	1.8731
A15	0.1875	1.5809	1.3109	1.8543	0.8252	0.2943	1.7526	1.6432	0.4868	1.5426	1.4421	0.8788

Table A.2: KA-III Example Dynamics, Excitatory Intergroup Weights and Delays

	$G1 \rightarrow G2$		$G1 \rightarrow G3$		$G2 \rightarrow G1$		$G2 \rightarrow G3$		$G3 \rightarrow G1$		$G3 \rightarrow G2$	
Ex	w_e	d_e	w_e	d_e	w_e	d_e	w_e	d_e	w_e	d_e	w_e	d_e
A1	0.232841	5	0.221927	6	0.185048	21	0.117421	5	0.467524	22	0.276123	16
A2	0.377983	4	0.345081	7	0.160817	19	0.464409	7	0.392669	18	0.282983	23
A3	0.220960	3	0.296607	4	0.342107	17	0.339608	5	0.179103	19	0.199442	23
A4	0.344364	8	0.397060	4	0.440834	19	0.483027	6	0.101405	20	0.471407	18
A5	0.319645	3	0.126312	5	0.147359	18	0.469319	7	0.241940	20	0.108746	16
A6	0.351429	7	0.291779	4	0.270962	20	0.104983	3	0.411906	17	0.123929	21
A7	0.296672	7	0.214717	6	0.447515	22	0.251944	8	0.336918	23	0.331891	23
A8	0.327073	8	0.177856	6	0.168265	19	0.285654	5	0.103372	16	0.143255	23
A9	0.335447	5	0.237209	3	0.192079	20	0.195522	4	0.256560	23	0.389094	19
A10	0.397823	8	0.165711	6	0.235706	24	0.265255	8	0.392624	18	0.214770	18
A11	0.140322	7	0.286568	5	0.208915	21	0.408498	4	0.314647	24	0.417782	20
A12	0.315625	8	0.173113	7	0.105686	20	0.388880	5	0.406686	22	0.236224	17
A13	0.237456	7	0.470708	4	0.201903	17	0.329915	3	0.324881	23	0.304854	22
A14	0.451200	5	0.460596	7	0.263962	19	0.173098	7	0.146993	17	0.212072	18
A15	0.455897	7	0.109468	5	0.479702	19	0.385898	5	0.404199	18	0.439346	24

Table A.3: KA-III Example Dynamics, Inhibitory Intergroup Weights and Delays

	$G1 \rightarrow G2$		$G1 \rightarrow G3$		$G2 \rightarrow G1$		$G2 \rightarrow G3$		$G3 \rightarrow G1$		$G3 \rightarrow G2$	
Ex	w_i	d_i	w_i	d_i	w_i	d_i	w_i	d_i	w_i	d_i	w_i	d_i
A1	0.266881	3	0.439417	5	0.311380	15	0.265730	6	0.394670	19	0.224777	20
A2	0.231119	3	0.240205	8	0.388418	18	0.348371	4	0.229439	17	0.312212	24
A3	0.205215	4	0.334103	4	0.239148	15	0.141743	4	0.124281	17	0.225676	24
A4	0.454866	7	0.471227	5	0.219314	15	0.410404	5	0.475576	24	0.367052	24
A5	0.217528	4	0.228897	8	0.412685	24	0.256272	3	0.186032	23	0.452025	20
A6	0.102593	8	0.423770	8	0.235887	22	0.293249	3	0.280043	22	0.302687	16
A7	0.266595	8	0.224641	7	0.114388	15	0.375349	4	0.315325	15	0.391950	20
A8	0.148815	3	0.214300	7	0.132522	23	0.208076	7	0.201720	24	0.235908	16
A9	0.373966	5	0.116075	5	0.440423	24	0.286329	5	0.128263	24	0.158096	15
A10	0.141813	4	0.499837	6	0.280794	15	0.156808	5	0.107266	18	0.251766	22
A11	0.100804	4	0.296324	3	0.152075	18	0.232171	3	0.336022	16	0.330642	16
A12	0.346894	8	0.287474	5	0.447354	19	0.347179	7	0.258233	18	0.261922	21
A13	0.392900	8	0.218787	7	0.423052	19	0.210258	7	0.401422	21	0.197223	16
A14	0.487237	5	0.326039	3	0.258633	23	0.174478	4	0.214108	16	0.270948	20
A15	0.413873	7	0.104797	7	0.168519	16	0.120010	5	0.219468	17	0.441363	16

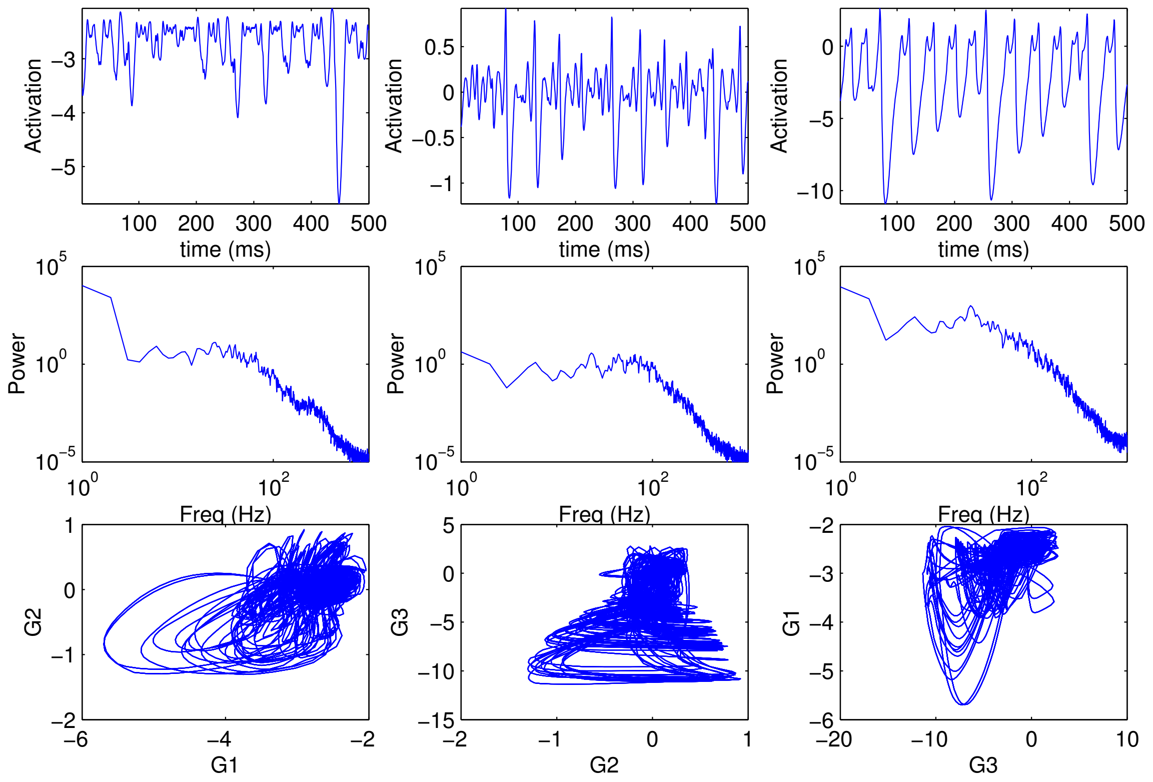


Figure A.1: Top figures are time series of G1 (left) G2 (center) and G3 (right). Middle are power spectrum distributions (PSD) of G1, G2 and G3 respectively. And bottom are state space plots of G1/G2, G2/G3 and G3/G1.

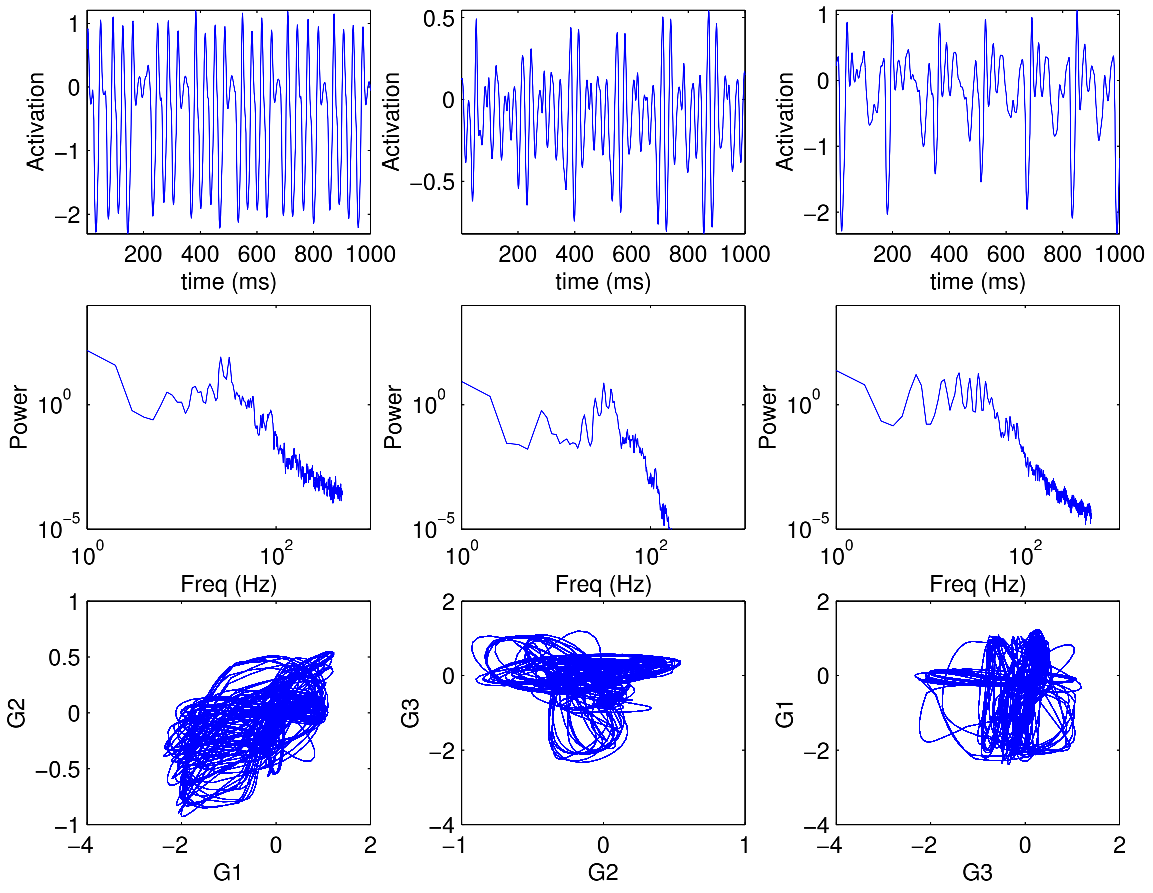


Figure A.2: Top figures are time series of G1 (left) G2 (center) and G3 (right). Middle are power spectrum distributions (PSD) of G1, G2 and G3 respectively. And bottom are state space plots of G1/G2, G2/G3 and G3/G1.

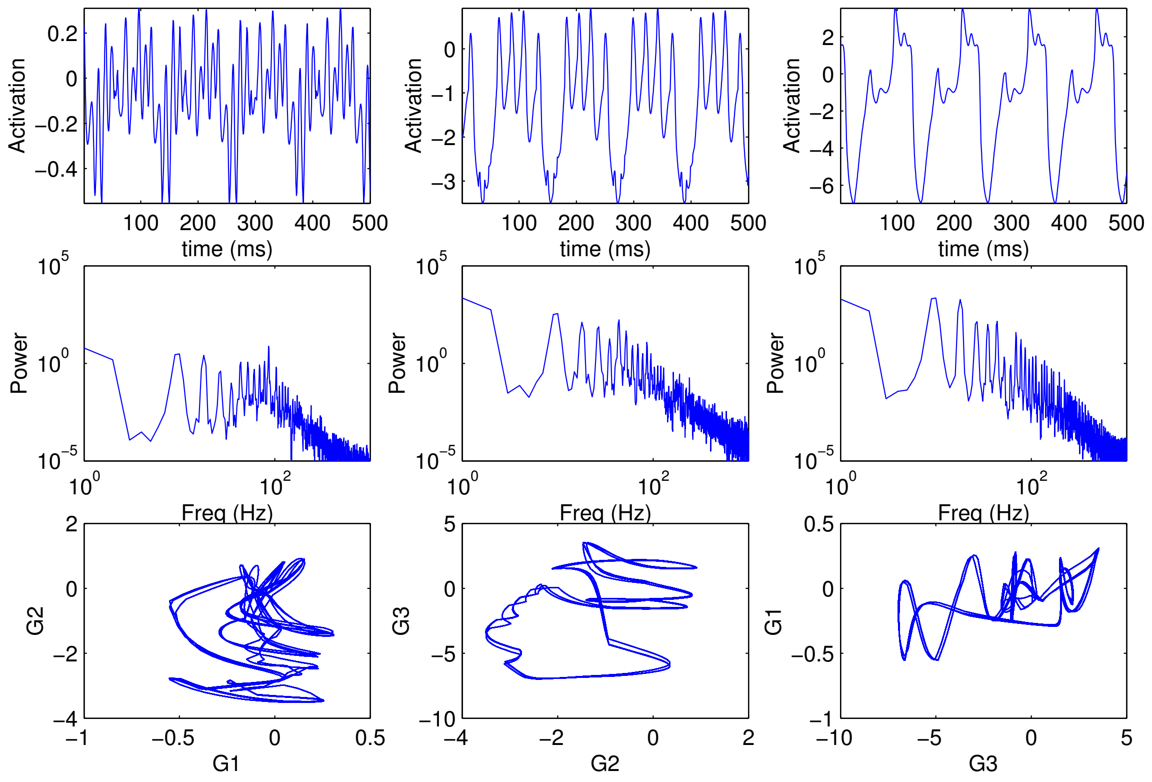


Figure A.3: Top figures are time series of G1 (left) G2 (center) and G3 (right). Middle are power spectrum distributions (PSD) of G1, G2 and G3 respectively. And bottom are state space plots of G1/G2, G2/G3 and G3/G1.

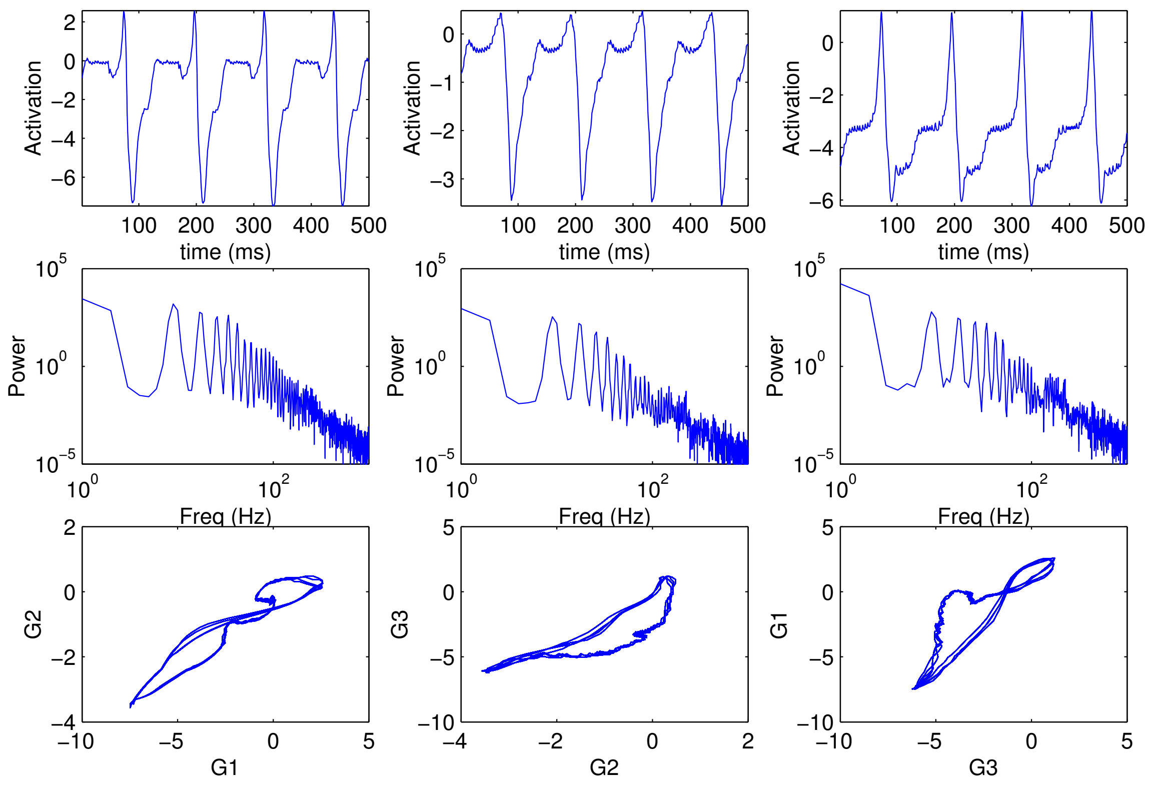


Figure A.4: Top figures are time series of G1 (left) G2 (center) and G3 (right). Middle are power spectrum distributions (PSD) of G1, G2 and G3 respectively. And bottom are state space plots of G1/G2, G2/G3 and G3/G1.

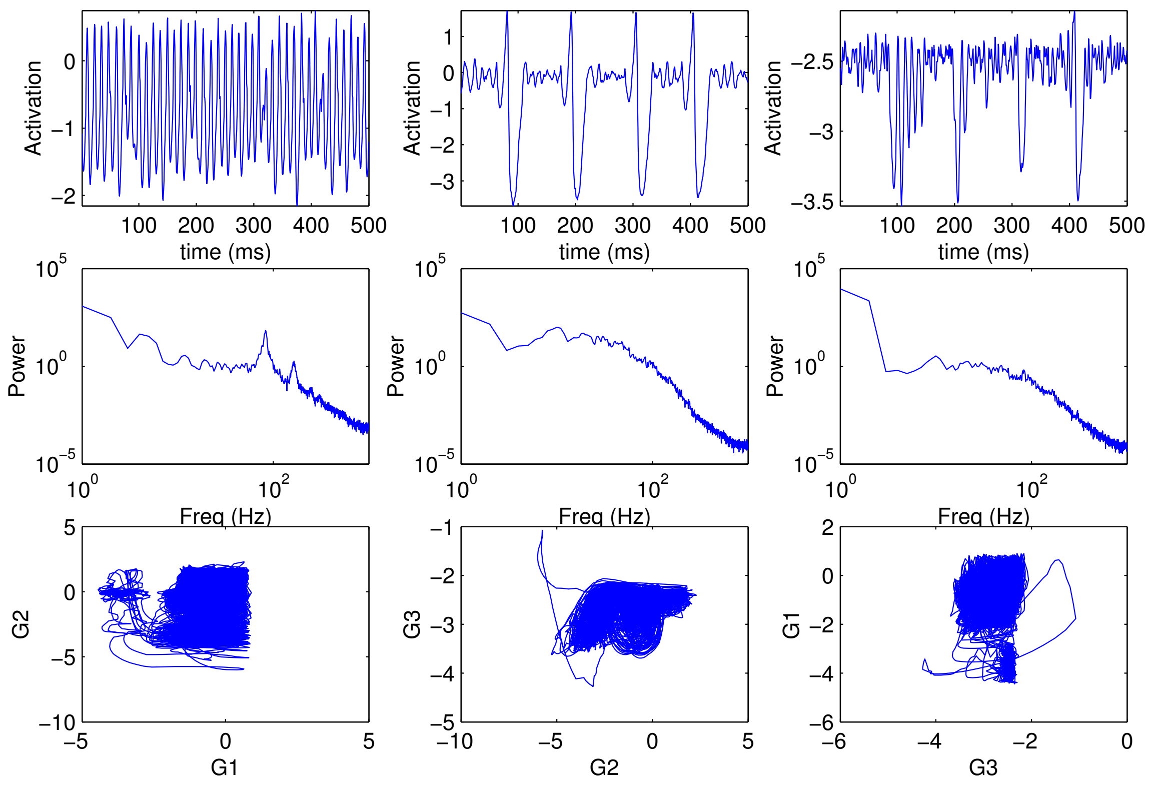


Figure A.5: Top figures are time series of G1 (left) G2 (center) and G3 (right). Middle are power spectrum distributions (PSD) of G1, G2 and G3 respectively. And bottom are state space plots of G1/G2, G2/G3 and G3/G1.

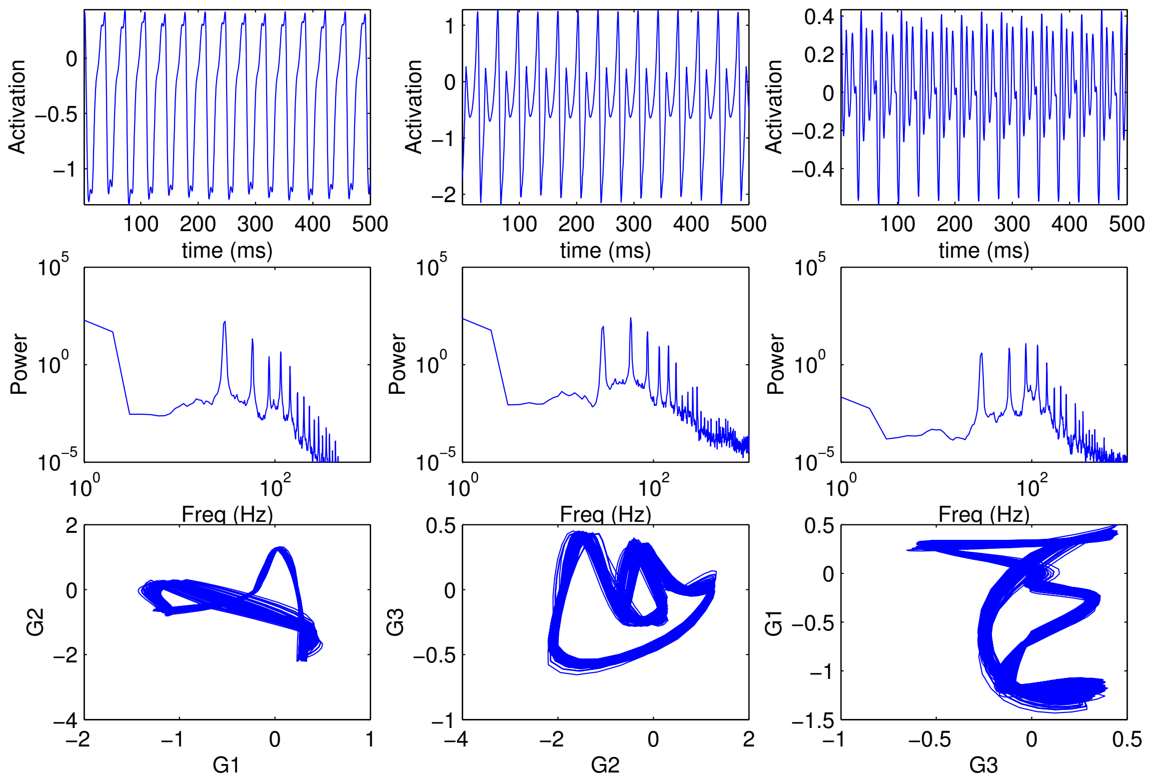


Figure A.6: Top figures are time series of G1 (left) G2 (center) and G3 (right). Middle are power spectrum distributions (PSD) of G1, G2 and G3 respectively. And bottom are state space plots of G1/G2, G2/G3 and G3/G1.

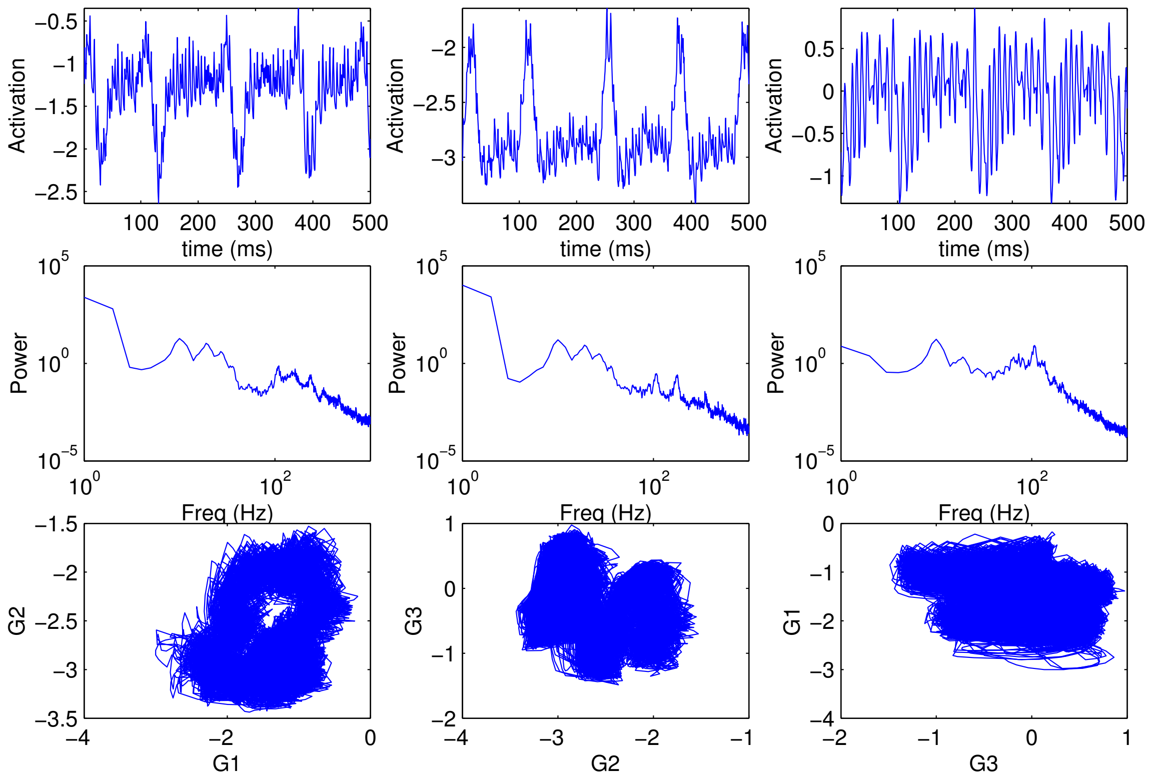


Figure A.7: Top figures are time series of G1 (left) G2 (center) and G3 (right). Middle are power spectrum distributions (PSD) of G1, G2 and G3 respectively. And bottom are state space plots of G1/G2, G2/G3 and G3/G1.

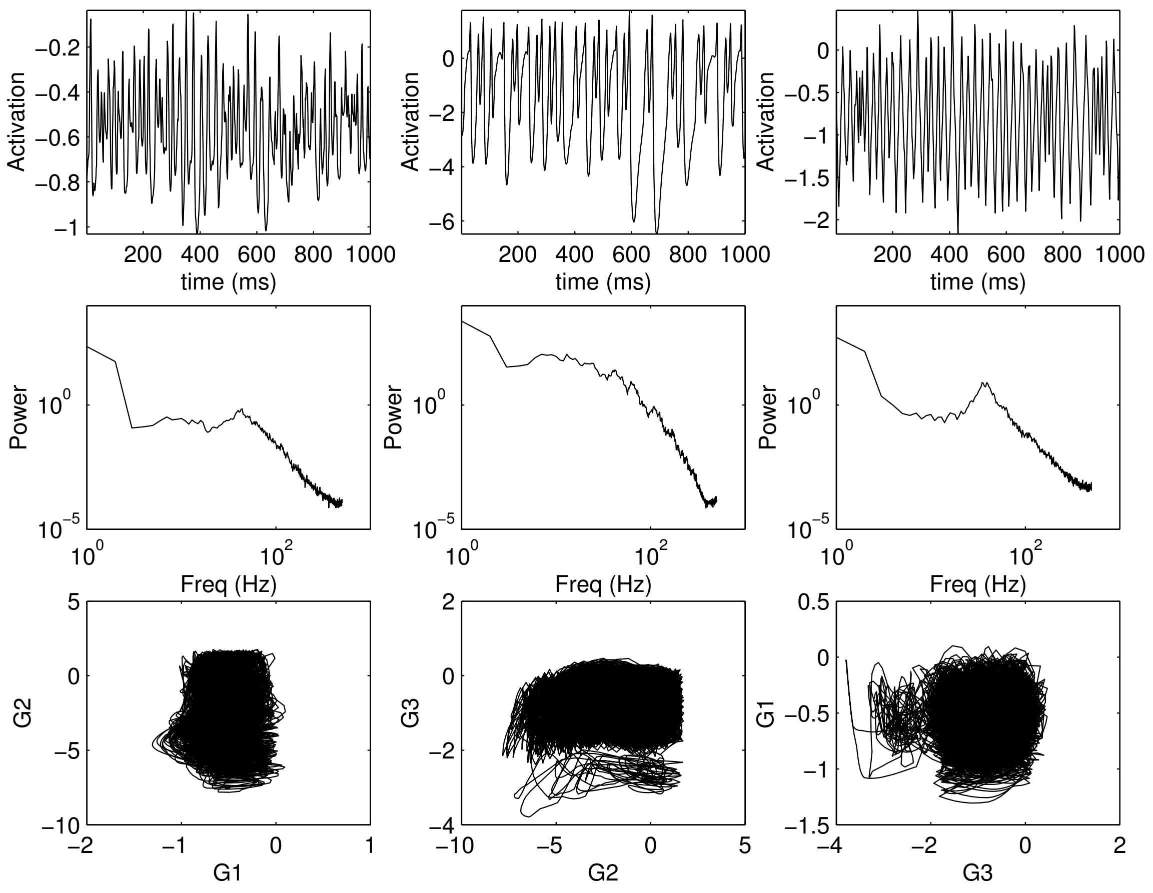


Figure A.8: Top figures are time series of G1 (left) G2 (center) and G3 (right). Middle are power spectrum distributions (PSD) of G1, G2 and G3 respectively. And bottom are state space plots of G1/G2, G2/G3 and G3/G1.

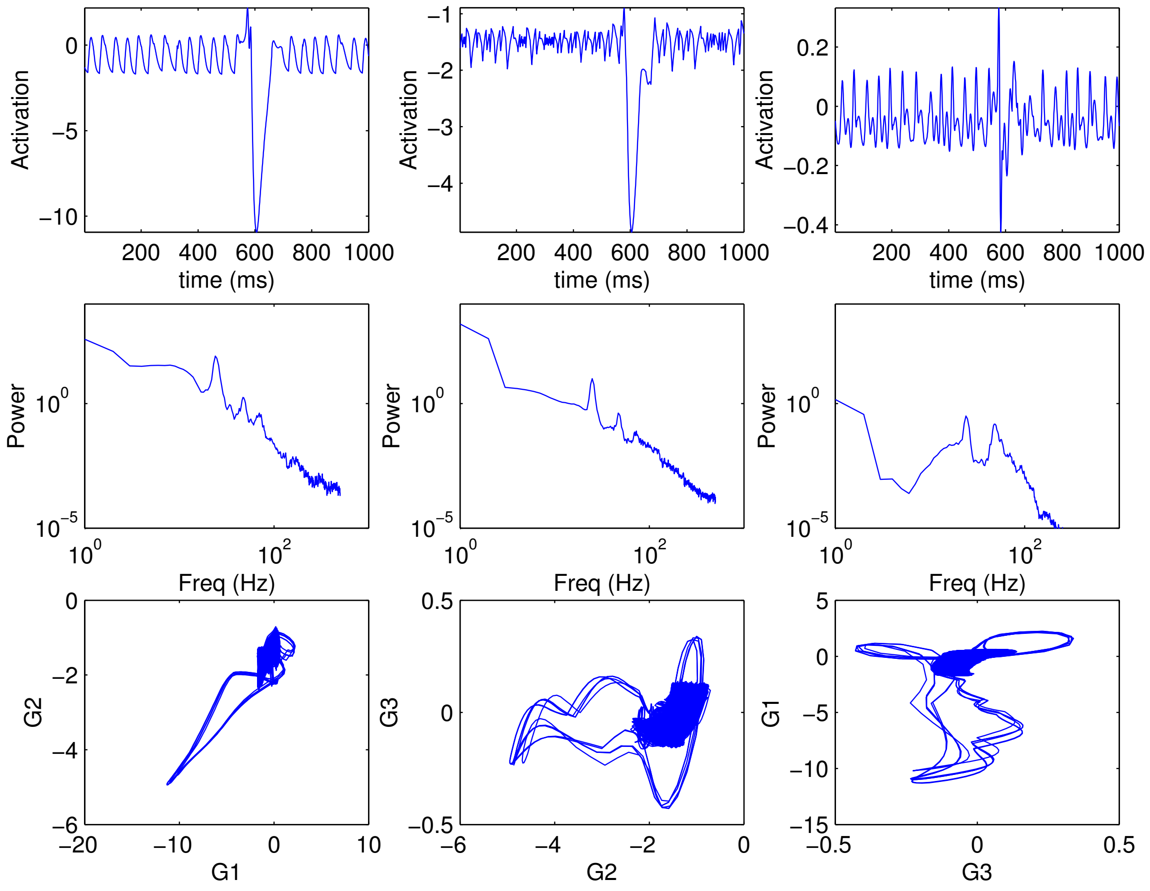


Figure A.9: Top figures are time series of G1 (left) G2 (center) and G3 (right). Middle are power spectrum distributions (PSD) of G1, G2 and G3 respectively. And bottom are state space plots of G1/G2, G2/G3 and G3/G1.

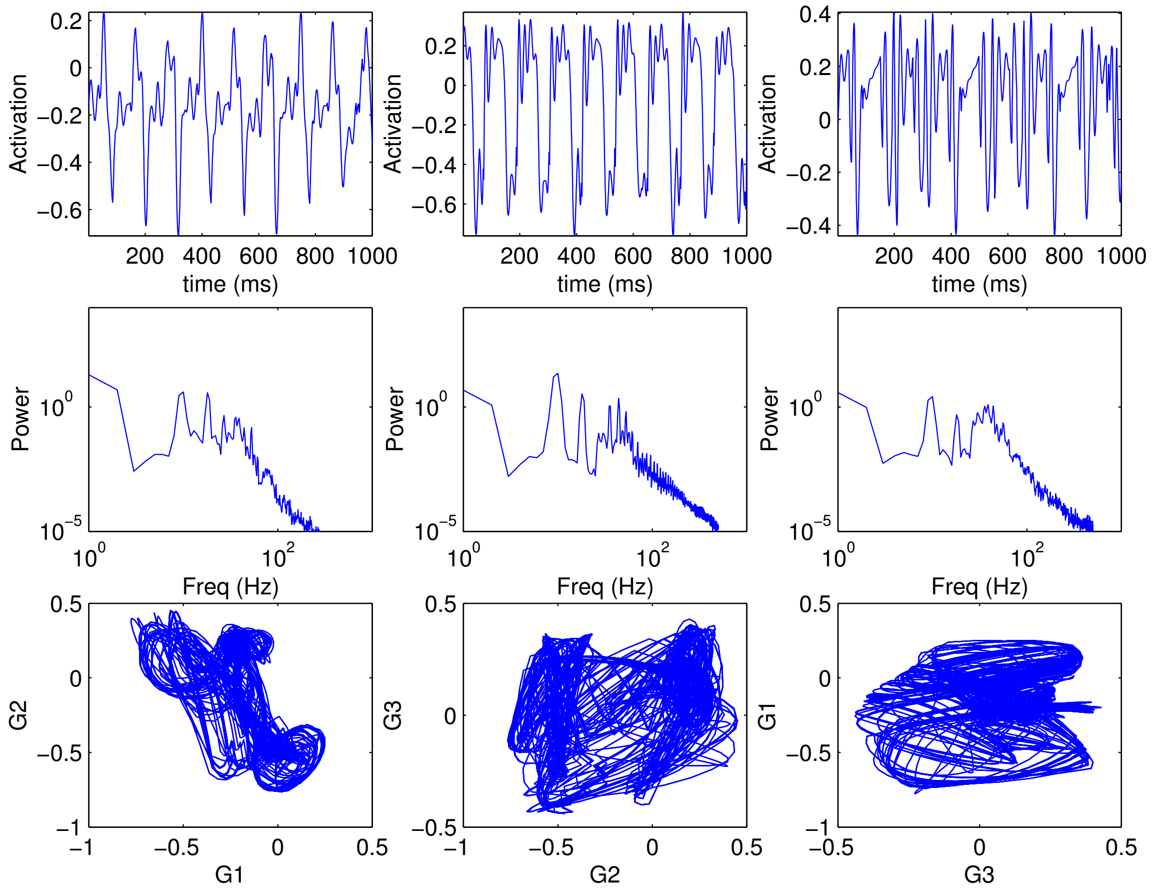


Figure A.10: Top figures are time series of G1 (left) G2 (center) and G3 (right). Middle are power spectrum distributions (PSD) of G1, G2 and G3 respectively. And bottom are state space plots of G1/G2, G2/G3 and G3/G1.

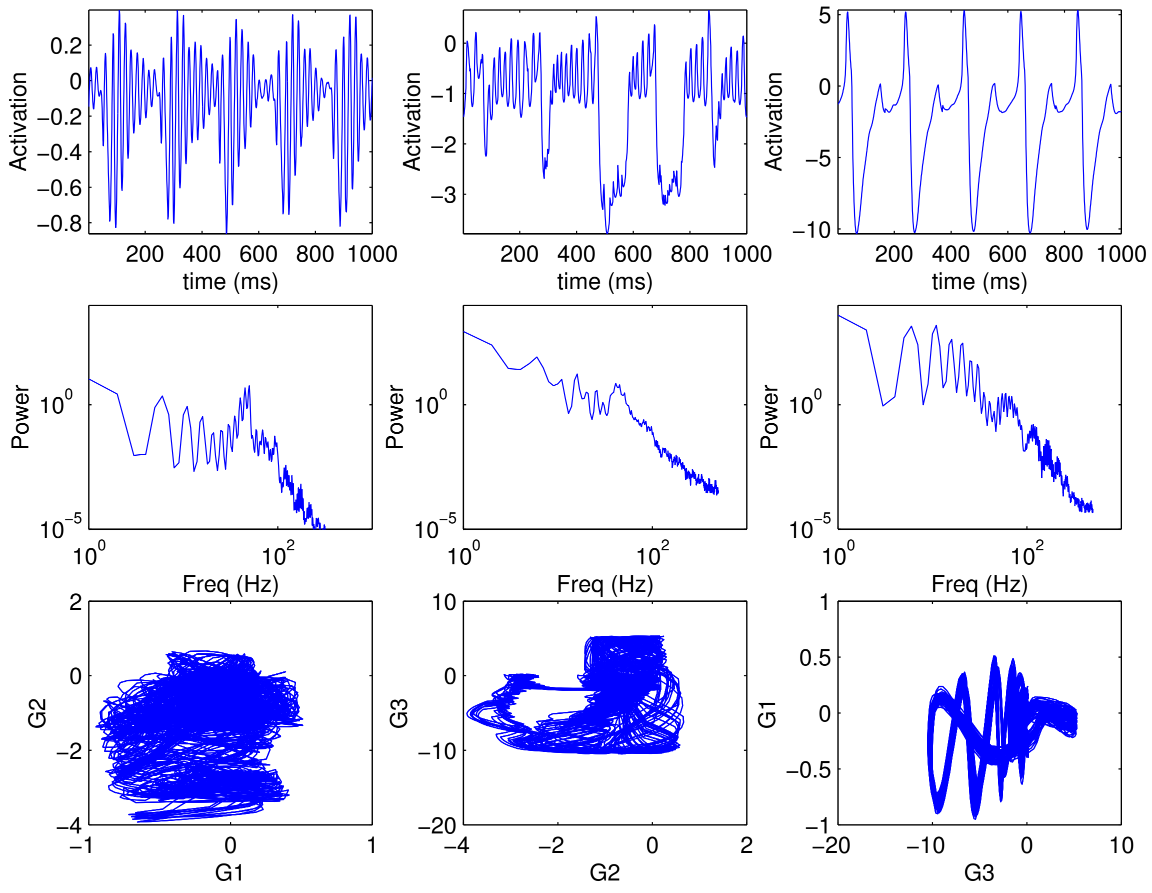


Figure A.11: Top figures are time series of G1 (left) G2 (center) and G3 (right). Middle are power spectrum distributions (PSD) of G1, G2 and G3 respectively. And bottom are state space plots of G1/G2, G2/G3 and G3/G1.

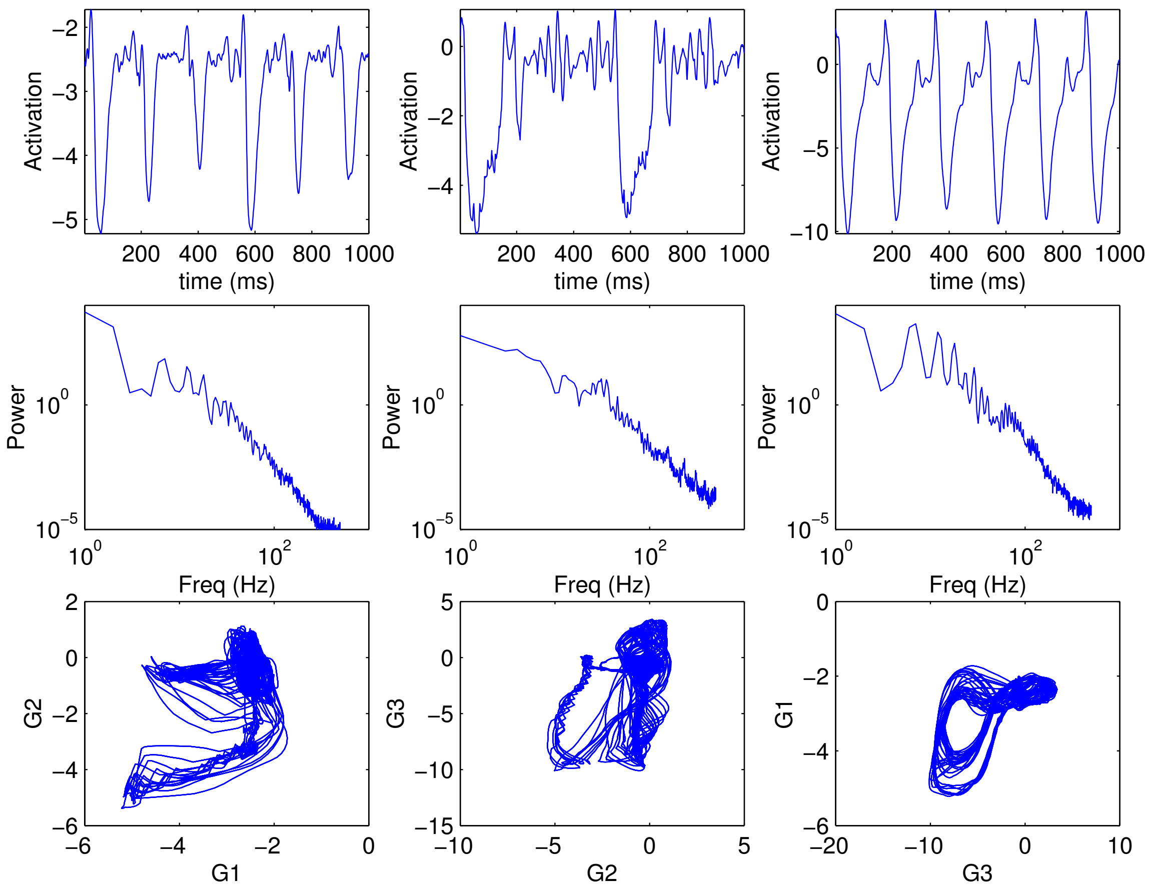


Figure A.12: Top figures are time series of G1 (left) G2 (center) and G3 (right). Middle are power spectrum distributions (PSD) of G1, G2 and G3 respectively. And bottom are state space plots of G1/G2, G2/G3 and G3/G1.

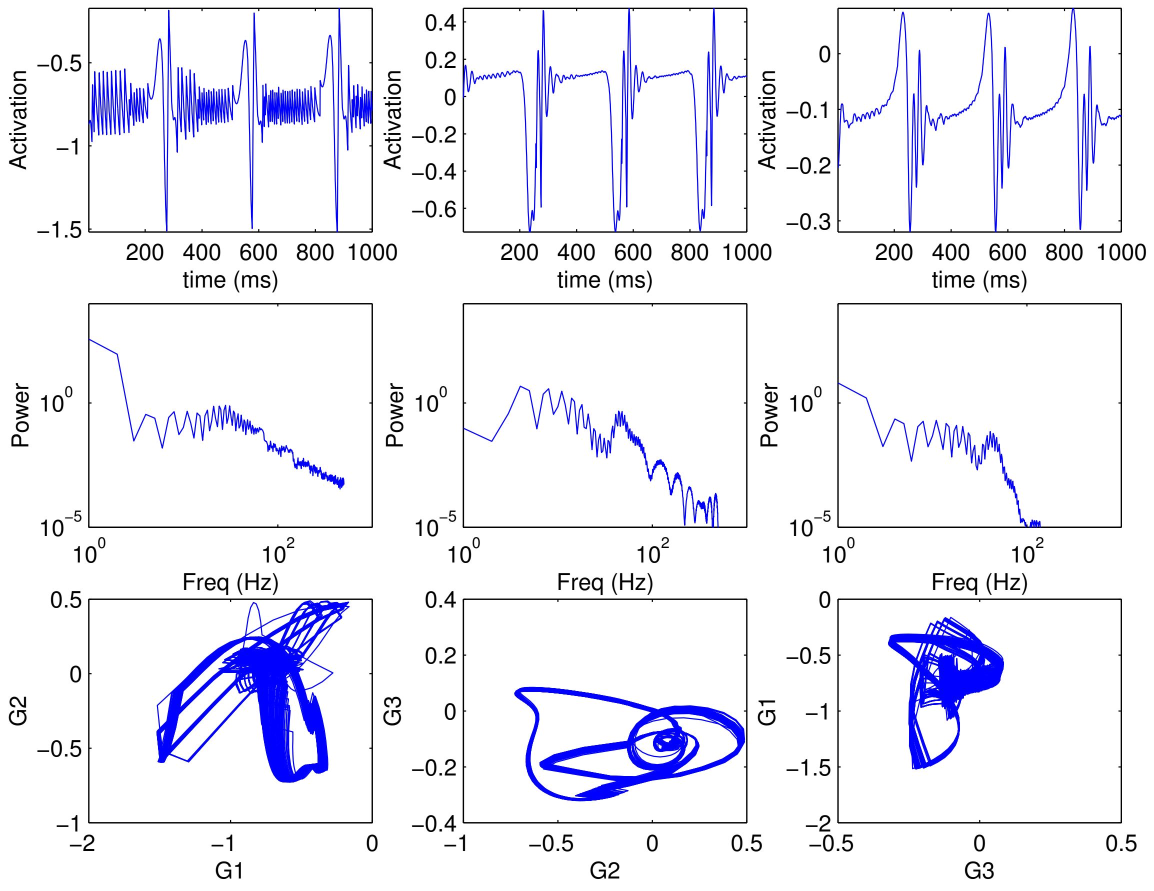


Figure A.13: Top figures are time series of G1 (left) G2 (center) and G3 (right). Middle are power spectrum distributions (PSD) of G1, G2 and G3 respectively. And bottom are state space plots of G1/G2, G2/G3 and G3/G1.

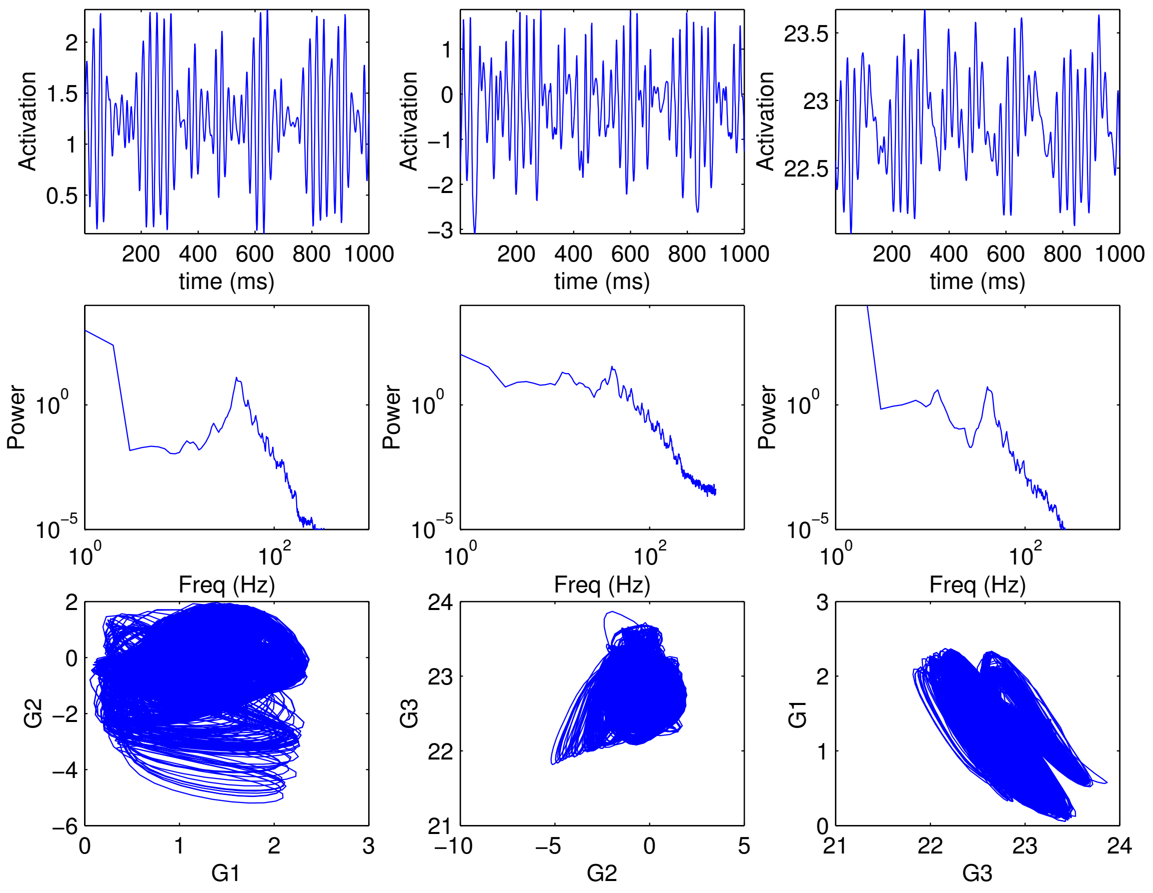


Figure A.14: Top figures are time series of G1 (left) G2 (center) and G3 (right). Middle are power spectrum distributions (PSD) of G1, G2 and G3 respectively. And bottom are state space plots of G1/G2, G2/G3 and G3/G1.

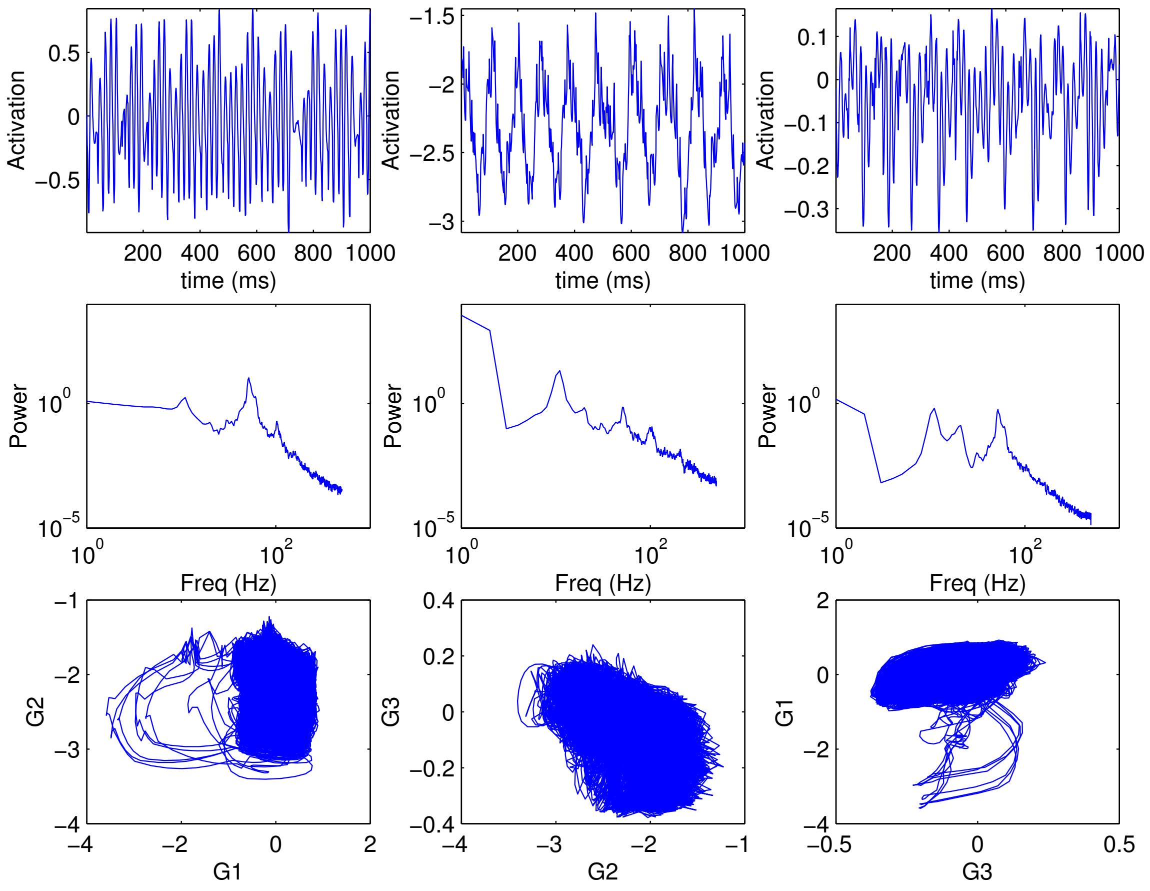


Figure A.15: Top figures are time series of G1 (left) G2 (center) and G3 (right). Middle are power spectrum distributions (PSD) of G1, G2 and G3 respectively. And bottom are state space plots of G1/G2, G2/G3 and G3/G1.

Appendix B

KA Source Code Listing

B.1 Axon

B.1.1 Axon.prop

```
# empty for now
LRDRPOW: 2.0
```

B.1.2 Axon.h

```
class Neuron;

class Axon
{
  public:
   static double LRDRPOW;

   Neuron* source;
   Neuron* dest;
   int time;
   double weight;
   int learn;
   int ipq;    // index into pulse queue
   double* pq; // pulse queue (circular)

   Axon(Neuron* source, Neuron* dest, int time, double weight);
   Axon(Neuron* source, Neuron* dest, int time, double weight, int learn);
   ~Axon();
   void initAxon(Neuron* source, Neuron* dest, int time, double weight, int learn);
   void pulse(double pulse);
   void reset();
   void hebb(double n, double lr, double dr, double avg, double maxWeight);
   void habituation(double n, double lr, double dr, double avg, double maxWeight);
   void waste(double ratio);
   void showPulses();
   static void goodEventGroup(double lr, double dr, double targetAvg, int group, double maxWeight);
   static void noEventGroup(double lr, double dr, double targetAvg, int group, double maxWeight);
   static void goodEvent(double lr, double dr, double targetAvg, double avgStd);

   static Axon* axons[][20000];
   static int axonCount[];
};
```

B.1.3 Axon.cpp

```
//
// $Id: Axon.cpp,v 1.6 2002/05/12 15:37:00 dharter Exp $
//
// This module implements a Axon to connect up Neuron objects.
// W.J. Freeman "How Brains Make Up Their Minds" pg. 47-58

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "Axon.h"
#include "Neuron.h"
#include "Properties.h"

// static member
const int MAXAXONS=20000;
const int MAXGROUPS=2;

double Axon::LRDRPOW = Properties::doubleValueForProperty("LRDRPOW");

Axon* Axon::axons[MAXGROUPS][MAXAXONS];
int Axon::axonCount[MAXGROUPS];

// Constructor
Axon::Axon(Neuron* source, Neuron* dest, int time, double weight)
{
   initAxon(source, dest, time, weight, 0);
}

Axon::Axon(Neuron* source, Neuron* dest, int time, double weight, int learn)
{
   initAxon(source, dest, time, weight, learn);
}


void Axon::initAxon(Neuron* source, Neuron* dest, int time, double weight, int learn)
{
   this->source = source;
   if (this->source == 0)
      fprintf(stderr, "<Axon::initAxon> invalid null source\n");

   this->dest = dest;
   if (this->dest == 0)
      fprintf(stderr, "<Axon::initAxon> invalid null dest\n");

   this->time = time;
   this->weight = weight;
   this->learn = learn;
   //this->pq = (double *)calloc(this->time, sizeof(double));
   this->pq = new double[this->time];
   for(int i=0; i<this->time; i++)
   {
      this->pq[i] = 0.0;
   }
   this->ipq = 0;

   if (axonCount[learn-1] >= MAXAXONS)
   {
      fprintf(stderr, "Axons::constructor exceeding MAXAXONS count\n");
   }
   else
   {
      if (learn)
      {
 axons[learn-1][axonCount[learn-1]] = this;
 axonCount[learn-1]++;
 //fprintf(stderr, "Axons::constructor added axon %d, learn=%d\n", axonCount, learn);
      }

   }
}

// Destructor
Axon::~Axon()
{
}

// Receive pulse from source and send pulse to destination after
// appropriate delay.  Delay is implemented by use of a circular queue.
void Axon::pulse(double pulse)
{
   pq[ipq] = pulse * weight;
   ipq += 1;
   if (ipq >= time)
   {
      ipq = 0;
   }
   if (pq[ipq] != 0)
   {
      dest->pulse(pq[ipq]);
   }
}

// Reset ourself back to 0 state
void Axon::reset()
{
   for (int i=0; i<time; i++)
   {
      pq[i] = 0;
   }
   ipq = 0;
}

// Perform hebbian learning
void Axon::hebb(double n, double lr, double dr, double targetAvg, double maxWeight)
{
   double dw, sc, dc, nlr, ndr;
   sc = source->current;
   dc = dest->current;

   sc = sc - source->avgCurrent;
   dc = dc - dest->avgCurrent;
   //nlr = (pow(targetAvg-fabs(targetAvg - weight),LRDRPOW))*(lr/pow(targetAvg,LRDRPOW));
   nlr = (pow(maxWeight-weight,LRDRPOW))*(lr/pow(maxWeight,LRDRPOW));
   //nlr = lr;
   dw = nlr * sc * dc;

   weight += dw;

   //ndr = (pow(fabs(weight - targetAvg),LRDRPOW))*(dr/pow(targetAvg,LRDRPOW));
   ndr = (pow(weight,LRDRPOW))*(dr/pow(maxWeight,LRDRPOW));
   //ndr = dr;
   weight -= ndr;

   if (weight > maxWeight) weight = maxWeight;
   if (weight < 0.0) weight = 0.0;
   //fprintf(stderr, "Axon::hebb> sc=%f dc=%f dw=%f nlr=%f ndr=%f weight=%f\n", sc, dc, dw, nlr, ndr, weight);
}

// Perform habituation (rev hebb?)
void Axon::habituation(double n, double lr, double dr, double targetAvg, double maxWeight)
{
   double dw, sc, dc, nlr, ndr;
   sc = source->current;
   dc = dest->current;

   sc = sc - source->avgCurrent;
   dc = dc - dest->avgCurrent;
   //nlr = (pow(targetAvg-fabs(targetAvg - weight),LRDRPOW))*(lr/pow(targetAvg,LRDRPOW));
   nlr = (pow(maxWeight-weight,LRDRPOW))*(lr/pow(maxWeight,LRDRPOW));
   //nlr = lr;
   dw = nlr * sc * dc;

   weight -= dw;

   //ndr = (pow(fabs(weight - targetAvg),LRDRPOW))*(dr/pow(targetAvg,LRDRPOW));
   ndr = (pow(weight,LRDRPOW))*(dr/pow(maxWeight,LRDRPOW));
   //ndr = dr;
   weight -= ndr;
   if (weight > maxWeight) weight = maxWeight;
   if (weight < 0.0) weight = 0.0;
   //fprintf(stderr, "Axon::hebb> sc=%f dc=%f dw=%f nlr=%f ndr=%f weight=%f\n", sc, dc, dw, nlr, ndr, weight);
}



// Reduce our weight by some percentage
void Axon::waste(double ratio)
{
   weight *= ratio;
}



void Axon::showPulses()
{

   for (int i=0; i<time; i++)
   {
      printf("%0.2f ", pq[i]);
   }
}

// static member functions for causing learning on axons
void Axon::goodEventGroup(double lr, double dr, double targetAvg, int group, double maxWeight)
{
   //double totWeight;
   double avg;
   Axon* axon;
   int i,j;

   // do hebbian learning on all all axons, at the same time calculate the
   // total (new) sum of all axon weights.
   //totWeight = 0.0;
   for (j=0; j<axonCount[group]; j++)
   {
      axon = axons[group][j];
      axon->hebb(axonCount[group], lr, dr, targetAvg, maxWeight);
      //totWeight += axon->weight;
   }



   // adjust total weight space
   //double targetWeight = targetAvg * (double)axonCount[group];
   //double ratio = targetWeight / totWeight;
   //for (j=0; j<axonCount[group]; j++)
   //{
   //   axon = axons[group][j];
   //   axon->waste(ratio);
   //}

}

// static member functions for causing habituation on axons
void Axon::noEventGroup(double lr, double dr, double targetAvg, int group, double maxWeight)
```

```
{
   //double totWeight;
   double avg;
   Axon* axon;
   int i,j;

   // do hebbian learning on all all axons, at the same time calculate the
   // total (new) sum of all axon weights.
   //totWeight = 0.0;
   for (j=0; j<axonCount[group]; j++)
   {
      axon = axons[group][j];
      axon->habituation(axonCount[group], lr, dr, targetAvg, maxWeight);
      //totWeight += axon->weight;
   }


   // adjust total weight space
   //double targetWeight = targetAvg * (double)axonCount[group];
   //double ratio = targetWeight / totWeight;
   //for (j=0; j<axonCount[group]; j++)
   //{
   //   axon = axons[group][j];
   //   axon->waste(ratio);
   //}

}


// static member functions for causing learning on axons
void Axon::goodEvent(double lr, double dr, double targetAvg, double avgStd)
{
   double totWeight[MAXGROUPS];
   double avg;
   Axon* axon;
   int i,j;

   //fprintf(stderr, "<Axon::goodEvent> entered lr=%f dr=%f targetAvg=%f\n", lr, dr, targetAvg);

   // do hebbian learning on all all axons, at the same time calculate the
   // total (new) sum of all axon weights.
   for (i=0; i<MAXGROUPS; i++)
   {
      totWeight[i] = 0.0;
      for (j=0; j<axonCount[i]; j++)
      {
 //printf("Axon::goodEvent learning group %d axon %d\n", i, j);
 axon = axons[i][j];
 axon->hebb(axonCount[i], lr, dr, targetAvg, 0.0);
 totWeight[i] += axon->weight;
      }
   }

   // adjust total weight space
   for (i=0; i<MAXGROUPS; i++)
   {

      double targetWeight = targetAvg * (double)axonCount[i];

      //fprintf(stderr, "<Axon::goodEvent> totWeight[%d](after learn)=%f, targetWeight=%f targetAvg=%f count=%d\n", i, totWeight[i], targetWeight, targetAvg, axon

      double ratio = targetWeight / totWeight[i];
      //double fudgeamt = (targetWeight - totWeight[i]) / axonCount[i];

      //fprintf(stderr, "<Axon::goodEvent> doing adjustment, ratio=%f\n", ratio);
      for (j=0; j<axonCount[i]; j++)
      {
 axon = axons[i][j];
 axon->waste(ratio);
      }
   }

}
```

B.2 KAii

B.2.1 KAii.prop

```
DEFAULTSIZE: 1
```

B.2.2 KAii.h

class NeuronGroup;

class KAii public: static int DEFAULTSIZE;

NeuronGroup* e1; NeuronGroup* e2; NeuronGroup* i1; NeuronGroup* i2;

double ee; double ei; double ie; double ii; int size;

KAii(double ee, double ei, double ie, double ii); KAii(double ee, double ei, double ie, double ii, int size); KAii(); void tick(); void setArousal(double arousal); void reset(); double pulseCount();

private:

;

B.2.3 KAii.cpp

```
//
// $Id: KAii.cpp,v 1.1 2002/01/29 16:44:54 dharter Exp $
//

#include <stdio.h>
#include "KAii.h"
#include "NeuronGroup.h"
#include "Neuron.h"
#include "Properties.h"

// read in properties from property file
```

```
int KAii::DEFAULTSIZE = Properties::intValueForProperty("DEFAULTSIZE");

// Constructor
KAii::KAii(double ee, double ei, double ie, double ii)
{
   KAii(ee, ei, ie, ii, DEFAULTSIZE);
}

KAii::KAii(double ee, double ei, double ie, double ii, int size)
{
   this->ee = ee;
   this->ei = ei;
   this->ie = ie;
   this->ii = ii;
   this->size = size;

   // create kii neural units
   this->e1 = new NeuronGroup(Neuron::EXCITATORY, size);

   this->e2 = new NeuronGroup(Neuron::EXCITATORY, size);
   this->i1 = new NeuronGroup(Neuron::INHIBITORY, size);
   this->i2 = new NeuronGroup(Neuron::INHIBITORY, size);

   // set up connections w/ proper weights
   this->e1->addConnection(this->e2, this->ee);
   this->e2->addConnection(this->e1, this->ee);
   this->i1->addConnection(this->i2, this->ii);
   this->i2->addConnection(this->i1, this->ii);

   this->e2->addConnection(this->i1, this->ei);
   this->i1->addConnection(this->e2, this->ie);

   this->e1->addConnection(this->i2, this->ei);
   this->i2->addConnection(this->e1, this->ie);

   this->e1->addConnection(this->i1, this->ei);
   this->i1->addConnection(this->e1, this->ie);

}

// Destructor
KAii::~KAii()
{
}

void KAii::tick()
{
   e1->tick();
   e2->tick();
   i1->tick();
   i2->tick();
}

void KAii::setArousal(double arousal)
```

```
{
   e1->setArousal(arousal);
   e2->setArousal(arousal);
   i1->setArousal(arousal);
   i2->setArousal(arousal);
}

void KAii::reset()
{
   e1->reset();
   e2->reset();
   i1->reset();
   i2->reset();
}
```

B.3 NeuroUtilities

B.3.1 NeuroUtilities.h

```
class NeuroUtilities
{
  public:
   // static global utility functions
   static double range(double min, double max);

   NeuroUtilities();
   ~NeuroUtilities();
};
```

B.3.2 NeuroUtilities.cpp

```
//
// $Id: NeuroUtilities.cpp,v 1.1 2002/01/29 16:44:54 dharter Exp $
//
// Implement utility functions that are useful globally.

#include <stdlib.h>
#include "NeuroUtilities.h"

double NeuroUtilities::range(double min, double max)
{
   double spin = (double)((double)random() / (double)RAND_MAX);
   double range = max - min;
   double ret = ((range * spin) + min);
   return ret;
}
```

```
// constructor
NeuroUtilities::NeuroUtilities()
{
}

// destructor
NeuroUtilities::~NeuroUtilities()
{
}
```

B.4 Neuron

B.4.1 Neuron.prop

```
# Neuron firing constants
PULSE: 1.0
TIME: 1

#AROUSAL: 3.0
AROUSAL: 5.0

# Max number of connections
AXONCONNECTIONS: 255

# constants for pulse-wave conversions
# DECAY (alpha) is the decay to baseline slope constant
#DECAY: 0.0785
DECAY: 0.07

# MOMENTUM (beta) is the momentum constant
#MOMENTUM: 0.01
MOMENTUM: 0.6

# INPUTSCALING (gamma) is the input scaling constant
#INPUTSCALING: 0.08
INPUTSCALING: 0.15

# saturation constants
#SATURATIONTHRESHOLD: 0.95
#SATURATIONPOWER: 0.5
SATURATIONTHRESHOLD: 0.75
SATURATIONPOWER: 0.5
```

B.4.2 Neuron.h

```
class Axon;
class NeuronGroup;
class Neuron
{
  public:
   enum NeuronType
   {
      EXCITATORY = 1,
      INHIBITORY = 2
   };

   // class constants (obtained from properties)
   static double PULSE;
   static int TIME;
   static double AROUSAL;

   static int AXONCONNECTIONS;

   static double DECAY;
   static double MOMENTUM;
   static double INPUTSCALING;

   static double SATURATIONTHRESHOLD;
   static double SATURATIONPOWER;

   // class variables
   int type;
   int numConnections;
   double arousal;

   double current;
   double prevCurrent;
   double input;
   double prevInput;

   double adjmax;
   double adjmin;
   double adjrange;

   double pulseAmount;

   double cq[50];
   int cqi;
   double avgCurrent;

   Axon** connections;
   NeuronGroup* group;

   Neuron(int type);
   Neuron(int type, double arousal);
   ~Neuron();
   void addConnection(Neuron* neuron, double weight);
   void addConnection(Neuron* neuron, double weight, int time);
   void addConnection(Neuron* neuron, double weight, int time, int learn);
   void addConnection(NeuronGroup* ng, double weight);
   void addConnection(NeuronGroup* ng, double weight, int time);
   void addConnection(NeuronGroup* ng, double weight, int time, int learn);
   Axon* getConnection(Neuron* neuron);
   void tick();
   void reset();
   void pulse(double ratio);
   void setArousal(double arousal);
   void rest();
   void setGroup(NeuronGroup* group);
   double getAverageWeight();
   double getStd();
   static void tickClock();
   static void newSim();
   static double getAvgStd();

  private:
   void initNeuron(int type, double arousal);
   void ensembleWavePulse();
   void fire(double ratio);
   static Neuron* neurons[];
   static int neuronCount;

};
```

B.4.3 Neuron.cpp

```
//
// $Id: Neuron.cpp,v 1.7 2002/05/12 15:37:00 dharter Exp $
//
// This module implements a Neuron (unit) to give dynamics as those shown in
// W.J. Freeman "How Brains Make Up Their Minds" pg. 47-58

#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include "Neuron.h"
#include "NeuronGroup.h"
#include "Axon.h"
#include "Properties.h"

// static member
const int MAXNEURONS = 5000;
Neuron* Neuron::neurons[MAXNEURONS];
int Neuron::neuronCount = 0;

// read in properties from property file
double Neuron::PULSE = Properties::doubleValueForProperty("PULSE");
int Neuron::TIME = Properties::intValueForProperty("TIME");
double Neuron::AROUSAL = Properties::doubleValueForProperty("AROUSAL");

int Neuron::AXONCONNECTIONS = Properties::intValueForProperty("AXONCONNECTIONS");

double Neuron::DECAY = Properties::doubleValueForProperty("DECAY");
double Neuron::MOMENTUM = Properties::doubleValueForProperty("MOMENTUM");
double Neuron::INPUTSCALING = Properties::doubleValueForProperty("INPUTSCALING");

double Neuron::SATURATIONTHRESHOLD = Properties::doubleValueForProperty("SATURATIONTHRESHOLD");
double Neuron::SATURATIONPOWER = Properties::doubleValueForProperty("SATURATIONPOWER");

#define CQSPAN 50

// Constructors
Neuron::Neuron(int type)
{
   initNeuron(type, AROUSAL);
}

Neuron::Neuron(int type, double arousal)
{
   initNeuron(type, arousal);
}

void Neuron::initNeuron(int type, double arousal)
{
   // initialize class variables
   this->type = type;
   this->numConnections = 0;
   this->connections = new Axon*[AXONCONNECTIONS];
   this->setArousal(arousal);

   this->current = 0.0;
   this->prevCurrent = 0.0;
   this->input = 0.0;
   this->prevInput = 0.0;
   this->avgCurrent = 0.0;
   for (int i=0; i<CQSPAN; i++)
   {
      cq[i] = 0.0;
   }
   cqi = 0;

   if (type == EXCITATORY)
   {
      pulseAmount = PULSE;
   }
   else
   {
      pulseAmount = -PULSE;
   }

   if (neuronCount >= MAXNEURONS)
   {
      fprintf(stderr, "Neuron::constructor exceeding MAXNEURONS count\n");
   }
   else
   {
      neurons[neuronCount] = this;
      neuronCount++;
   }
 }


// Destructor
Neuron::~Neuron()
{
}


// Add an axon connection from ourself to some other Neuron
void Neuron::addConnection(Neuron* neuron, double weight)
{
   addConnection(neuron, weight, TIME, 0);
}

void Neuron::addConnection(Neuron* neuron, double weight, int time)
{
   addConnection(neuron, weight, time, 0);
}

void Neuron::addConnection(Neuron* neuron, double weight, int time, int learn)
{
   Axon* axon = new Axon(this, neuron, time, weight, learn);
   connections[numConnections] = axon;
   numConnections++;

   if (numConnections > AXONCONNECTIONS)
   {
      // future: dynamically increase size of array, or use a list?
      fprintf(stderr, "<Neuron::addConnection> numConnections exceeding connection array length\n");
   }
}

// Add an axon connection from ourself to a NeuronGroup
void Neuron::addConnection(NeuronGroup* ng, double weight)
{
   addConnection(ng, weight, TIME, 0);
}

void Neuron::addConnection(NeuronGroup* ng, double weight, int time)
{
   addConnection(ng, weight, time, 0);
}

void Neuron::addConnection(NeuronGroup* ng, double weight, int time, int learn)
{
   for (int i=0; i<ng->size; i++)
   {
      Axon* axon = new Axon(this, ng->neurons[i], time, weight, learn);
      connections[numConnections] = axon;
      numConnections++;

      if (numConnections > AXONCONNECTIONS)
      {
 // future: dynamically increase size of array, or use a list?
 fprintf(stderr, "<Neuron::addConnection> numConnections exceeding connection array length\n");
      }
   }

}


Axon* Neuron::getConnection(Neuron* neuron)
{
   Axon* res;
   int done = 0;

   for (int i=0; i<numConnections; i++)
   {
      res = connections[i];
      if (res->dest == neuron)
      {
 done = 1;
 break;
      }
   }
   if (!done)
      res = NULL;

   return res;
}

// Cause ourself to simulate a single time click by doing our pulse-wave
// and wave-pulse conversions.
void Neuron::tick()
{
   double delta, delta1, delta2, delta3;

   //printf("<Neuron::tick> upon entering current=%f prevCurrent=%f\n", current, prevCurrent);

   // calculate differences for this time step
   delta1 = -current * DECAY;  // difference due to decay to baseline
   delta2 = (current - prevCurrent) * MOMENTUM; // maintain momementum
   delta3 = input * INPUTSCALING; // add in influence from pulses/input
   //printf("<Neuron::tick> current =%f prevcurrent=%f input=%f scaleing=%f delta1=%f delta2=%f delta3=%f DECAY=%f MOMENTUM=%f\n", current, prevCurrent, input, IN
   delta = delta1 + delta2 + delta3;

   // saturate difference if approaching upper or lower boundary
//     if (((current + delta) > SATURATIONTHRESHOLD) && (delta > 0.0))
//     {
//        double ratio = (1.0 - current) / (1.0 - SATURATIONTHRESHOLD);
//        ratio = pow(ratio, SATURATIONPOWER);
//        delta = ratio * delta;
//     }

//     if (((current + delta) < -SATURATIONTHRESHOLD) && (delta < 0.0))
//     {
//        double ratio = (1.0 + current) / (1.0 - SATURATIONTHRESHOLD);
//        ratio = pow(ratio, SATURATIONPOWER);
//        delta = ratio * delta;
//     }

   //printf("    <Neuron::tick> delta=%f delta1=%f delta2=%f delta3=%f\n", delta, delta1, delta2, delta3);

   // update variables to reflect differences for this time step
   prevCurrent = current;
   prevInput = input;
   input = 0.0;
   current = current + delta;
//   if (current > 1.0) current = 1.0;
//   if (current < -1.0) current = -1.0;

   cq[cqi] = current;
   cqi += 1;
   if (cqi >= CQSPAN)
   {
      cqi = 0;
   }
   double sum = 0.0;
   for (int i=0; i<CQSPAN; i++)
   {
      sum += cq[i];
   }
   avgCurrent = sum / CQSPAN;

   //printf("    <Neuron::tick> before pulse gen current=%f prevCurrent=%f\n", current, prevCurrent);
   // send out pulse if needed
   ensembleWavePulse();

}

// Simulate neuron ensembles wave-pulse dynamics (asymetric sigmoid)
void Neuron::ensembleWavePulse()
{
   double c, ratio;
   //int newrefperiod;

   // calculate new refactory period based on current current (so to speak)
   //printf("<Neuron::ensembleWavePulse> entered\n");

   //c = (current * 4.0);
   ratio = arousal * (1.0 - exp(-(exp(current)-1.0)/arousal));
   if (ratio < -1.0)
      ratio = -1.0;

   //ratio = (ratio - adjmin) / adjrange;
   //printf("<Neuron::ensembleWavePulse> c=%f arousal=%f ratio=%f adjmin=%f adjrange=%f\n", c, arousal, ratio, adjmin, adjrange);

/*
   if (ratio < 0.005)
   {
      newrefperiod = -1;
   }
   else
   {
      newrefperiod = (int)(((1.0 - ratio) * PULSEPERIOD) + 1);
   }

   if ((refperiod < 0) && (newrefperiod > 0))
   {
      refperiod = newrefperiod;
   }



   if ((newrefperiod < refperiod) && (newrefperiod > 0))
   {
      refperiod = newrefperiod;
   }
   else
   {
      refperiod--;
   }
*/
   if (ratio > -1)
   {
      //printf("<Neuron::ensembleWavePulse> firing\n");
      fire(ratio);
   }
   else
   {
      //printf("<Neuron::ensembleWavePulse> resting\n");
      rest();
   }
   //fprintf(stdout, "     <Neuron::ensembleWavePulse> current=%0.5f ratio=%0.5f refperiod=%d newrefperiod=%d\n", current, ratio, refperiod, newrefperiod);
}

// fir a pulse to all of our connections
void Neuron::fire(double ratio)
{
   //printf("<Neuron::fire> ratio=%f\n", ratio);

   for (int i=0; i<numConnections; i++)
   {
      //printf("<Neuron::fire> pulsing connection %d of %d\n", i, numConnections);
      //printf("<Neuron::filre> pullsing connection=%d\n", connections[i]);

      connections[i]->pulse(pulseAmount * ratio);
      //if (current > 0)
      //printf("Neuron::fire sending out=%f pulse=%f ratio=%f current=%f\n", p*ratio, p, ratio, current);
   }

}

// Do not fire, give it a rest man
void Neuron::rest()
{
   double delta;

   // send pulse to our connections
   for (int i=0; i<numConnections; i++)
   {
      connections[i]->pulse(0.0);
   }
}

// Receive a pulse fired from someone connected to us
void Neuron::pulse(double pulse)
{
   //printf("<Neuron::pulse> entered with pulse=%f input=%f\n", pulse, input);
   input += pulse;
}

// Set our arousal level
void Neuron::setArousal(double arousal)
{
   double top, bot, dif;

   this->arousal = arousal;
   adjmax = arousal * (1.0 - exp(-(exp(4.0)-1.0)/arousal));
   adjmin = arousal * (1.0 - exp(-(exp(-4.0)-1.0)/arousal));
   adjrange = adjmax - adjmin;

}

// Reset ourself back to 0
void Neuron::reset()
{
   current = 0.0;
   prevCurrent = 0.0;
   input = 0.0;
   //refperiod = -1;
   for (int i=0; i<numConnections; i++)
   {
      connections[i]->reset();
   }

}

void Neuron::setGroup(NeuronGroup* group)
{
   this->group = group;
}

double Neuron::getAverageWeight()
{
   double totweight;

   totweight=0.0;
   for (int i=3; i<numConnections; i++)
   {
      //fprintf(stderr, "<Neuron::getAverageWeight> con[%d]=%f\n", i, connections[i]->weight);
      totweight += connections[i]->weight;
   }
   //fprintf(stderr, "    totweight=%f count=%d res=%f\n", totweight, (numConnections-3), totweight/(numConnections-3));
   return totweight / (numConnections-3);

}

// Calculate standard deviation over the saved interval of activity
double Neuron::getStd()
{
   double sumsq = 0.0;
   for (int i=0; i<CQSPAN; i++)
   {
      sumsq = sumsq + pow(cq[i]-avgCurrent, 2.0);
   }
   sumsq = sumsq / CQSPAN;
   return pow(sumsq, 0.5);
}



// static member functions for causing all neurons to tick their clock
void Neuron::tickClock()
{
   //printf("Neuron::tickClock entered\n");
   for (int i=0; i<neuronCount; i++)
   {
      //printf("Neuron::tickClock ticking neuron %d of %d\n", i, neuronCount);
      //printf("Neuron::tickClock neuron=%d\n", neurons[i]);

      neurons[i]->tick();
   }
}

void Neuron::newSim()
{
   for (int i=0; i<neuronCount; i++)
   {
      delete(neurons[i]);
   }
   neuronCount = 0;
}

double Neuron::getAvgStd()
{
   double sum = 0.0;
   //fprintf(stderr, "<Neuron::getAvgStd> entered neuronCount = %d\n", neuronCount);

   for (int i=0; i<neuronCount; i++)
   {
      sum += neurons[i]->getStd();
      //fprintf(stderr, "<Neuron::getAvgStd> neuron[%d] std = %f\n", i, neurons[i]->getStd());
   }
   double avg = sum / (double)neuronCount;
   //fprintf(stderr, "<Neuron::getAvgStd> sum=%f avg=%f\n", sum, avg);

   return avg;
}
```

B.5 NeuronGroup

B.5.1 NeuronGroup.h

```
class Neuron;

class NeuronGroup
{
  public:
   Neuron** neurons;
   int type;
   int size;

   NeuronGroup(int type, int size);
   ~NeuronGroup();
   void addConnection(NeuronGroup* ng, double weight);
   void addConnection(NeuronGroup* ng, double weight, int time);
   void addConnection(NeuronGroup* ng, double weight, int time, int learn);
   void addConnection(Neuron* neuron, double weight);
   void addConnection(Neuron* neuron, double weight, int time);
   void addConnection(Neuron* neuron, double weight, int time, int learn);
   void tick();
   void pulse(double pulse);
   void setArousal(double arousal);
   void reset();
   double current();
   double avgCurrent();
   double getWeight(NeuronGroup* ng);
   double getWeight(Neuron* neuron);

  private:

};
```

B.5.2 NeuronGroup.cpp

```
//
// $Id: NeuronGroup.cpp,v 1.2 2002/01/29 16:44:54 dharter Exp $
//
// A group of neurons that act together as a single population.  In many ways
// the NeuronGroup is the basic unit.  At the limit (when the group size = 1)
// The NeuronGroup becomes a wrapper for, and behaves as, a single unit.

#include <stdio.h>
#include "NeuronGroup.h"
#include "Neuron.h"
```

```
#include "Axon.h"

// Constructor
NeuronGroup::NeuronGroup(int type, int size)
{
   this->type = type;
   this->size = size;
   this->neurons = new Neuron*[size];
   for (int i=0; i<size; i++)
   {
      neurons[i] = new Neuron(type);
      neurons[i]->setGroup(this);
   }
}

// Destructor
NeuronGroup::~NeuronGroup()
{
}

// Add an axon connection from ourself to some other NeuronGroup
void NeuronGroup::addConnection(NeuronGroup* ng, double weight)
{
   addConnection(ng, weight, Neuron::TIME, 0);
}

void NeuronGroup::addConnection(NeuronGroup* ng, double weight, int time)
{
   addConnection(ng, weight, time, 0);
}

void NeuronGroup::addConnection(NeuronGroup* ng, double weight, int time, int learn)
{
   Neuron* source;
   Neuron* dest;

   for (int i=0; i<size; i++)
   {
      source = neurons[i];
      for (int j=0; j<ng->size; j++)
      {
 dest = ng->neurons[j];
 source->addConnection(dest, (weight/(double)size), time, learn);
      }
   }
}

// Add an axon connection from ourself to some other single Neuron
void NeuronGroup::addConnection(Neuron* neuron, double weight)
{
   addConnection(neuron, weight, Neuron::TIME, 0);
}

void NeuronGroup::addConnection(Neuron* neuron, double weight, int time)
```

```
{
   addConnection(neuron, weight, time, 0);
}

void NeuronGroup::addConnection(Neuron* neuron, double weight, int time, int learn)
{
   for (int i=0; i<size; i++)
   {
      neurons[i]->addConnection(neuron, (weight/(double)size), time, learn);
   }
}

// do a time click for all units in group
void NeuronGroup::tick()
{
   for (int i=0; i<size; i++)
   {
      neurons[i]->tick();
   }
}

// send a pulse to all units in group
void NeuronGroup::pulse(double pulse)
{
   for (int i=0; i<size; i++)
   {
      neurons[i]->pulse(pulse);
   }
}

// set arousal to all units in group
void NeuronGroup::setArousal(double arousal)
{
   for (int i=0; i<size; i++)
   {
      neurons[i]->setArousal(arousal);
   }
}

// reset the neurons in group back to initial (zero) state
void NeuronGroup::reset()
{
   for (int i=0; i<size; i++)
   {
      neurons[i]->reset();
   }
}

// accessor method to return average current of group
double NeuronGroup::current()
{
   double sum = 0.0;

   for (int i=0; i<size; i++)
   {
      sum += neurons[i]->current;
   }
   return (sum / (double)size);
}

// accessor method to return average current of group
double NeuronGroup::avgCurrent()
{
   double sum = 0.0;

   for (int i=0; i<size; i++)
   {
      sum += neurons[i]->avgCurrent;
   }
   return (sum / (double)size);
}

// find the current weight between ourself and another neuron group
double NeuronGroup::getWeight(NeuronGroup* ng)
{
   double weightSum = 0.0;

   // look at each of the Neurons in our group
   for (int i=0; i<size; i++)
   {
      Neuron* n = neurons[i];
      // look at each axon of the Neuron.
      for (int j=0; j<n->numConnections; j++)
      {
 Axon* a = n->connections[j];
 if (a->dest->group == ng)
 {
    weightSum += a->weight;
 }
      }
   }
   return (weightSum / (double)size);
}

double NeuronGroup::getWeight(Neuron* neuron)
{
   double weightSum = 0.0;

   // look at each of the Neurons in our group
   for (int i=0; i<size; i++)
   {
      Neuron* n = neurons[i];
      // look at each axon of the Neuron.
      for (int j=0; j<n->numConnections; j++)
      {
 Axon* a = n->connections[j];
 if (a->dest == neuron)
 {
    weightSum += a->weight;
 }
      }
   }
   return (weightSum / (double)size);
}
```

B.6 Properties

B.6.1 Properties.h

```
struct PropertyTable
{
      char *key;
      char *value;
};



class Properties
{
  public:
   // static member functions
   static void readPropertyFile(char *filename);
   static int intValueForProperty(char *property);
   static double doubleValueForProperty(char *property);
   static char* stringValueForProperty(char *property);
   static void displayProperties();
   Properties(char *filename);
   ~Properties();

  private:
   static PropertyTable* table[];
   static int tablei;
   static char *basePropertyFile;
   static int base;
   static void insert(char *key, char *value);
   static void init();
};
```

B.6.2 Properties.cpp

```
//
// $Id: Properties.cpp,v 1.2 2002/01/25 03:31:31 dharter Exp $
//
// Get properties from files and make them available globally.  Used so that
// we can avoid hard coding constants into programs.
```

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include "Properties.h"

// static member vars & functions
const int MAXENTRIES = 100;
PropertyTable* Properties::table[MAXENTRIES];
int Properties::tablei = 0;
char *Properties::basePropertyFile = "/home/dharter/work/proj/sodas/neurodyn/paramexp/prop/Axon.prop:/home/dharter/work/proj/sodas/neurodyn/paramexp/prop/Neuron.p
int Properties::base = 0;

void Properties::readPropertyFile(char *filename)
{
   const int BUFFER=1024;
   char* key;
   char* value;
   char line[BUFFER];
   char *l;
   FILE* pfile;

   if (!base) init();

   pfile = fopen(filename, "r");
   if (!pfile)
   {
      fprintf(stderr, "<Properties::readPropertyFile> couldn't find file '%s'\n", filename);
      return;
   }

   while(fgets(line, BUFFER, pfile) != NULL)
   {
      if (line[0] == '#')
      {
 continue;
      }

      if ((line[0] == ' ') || (line[0] == '\t') || (line[0] == '\n'))
      {
 continue;
      }

      l = line;
      key = strsep(&l, ": \t\n");
      value = strsep(&l, " \t\n");  // get rid of initial white space
      while (strlen(l) && !strlen(value))
 value = strsep(&l, "\n");
      insert(key, value);
   }
}

void Properties::insert(char *key, char *value)
{
   // search for existing
   PropertyTable* entry;

   for (int i=0; i<tablei; i++)
   {
      entry = table[i];
      if (strcmp(entry->key, key) == 0)
      {
 free(entry->value);
 entry->value = strdup(value);
 return;
      }

   }

   // if doesn't exist, insert it
   entry = new PropertyTable;
   entry->key = strdup(key);
   entry->value = strdup(value);
   table[tablei] = entry;
   tablei++;
   if (tablei > MAXENTRIES)
   {
      fprintf(stderr, "<Properties::readPropertyFile> exceding MAXENTRIES\n");
   }
}

int Properties::intValueForProperty(char *property)
{
   if (!base) init();

   for (int i=0; i<tablei; i++)
   {
      if (strcmp(table[i]->key, property) == 0)
      {
 int val = atoi(table[i]->value);
 return val;
      }
   }

   fprintf(stderr, "<Properties::intValueForProperty> couldn't find property: <%s>\n", property);

   return -1;
}

double Properties::doubleValueForProperty(char *property)
{
   if (!base) init();

   for (int i=0; i<tablei; i++)
   {
      if (strcmp(table[i]->key, property) == 0)
      {
 double val = atof(table[i]->value);
 return val;
```

```
      }
   }

   fprintf(stderr, "<Properties::doubleValueForProperty> couldn't find property: <%s>\n", property);

   return -1.0;
}

char* Properties::stringValueForProperty(char *property)
{
   if (!base) init();

   for (int i=0; i<tablei; i++)
   {
      if (strcmp(table[i]->key, property) == 0)
      {
 return table[i]->value;
      }
   }

   fprintf(stderr, "<Properties::stringValueForProperty> couldn't find property: <%s>\n", property);

   return NULL;
}


void Properties::displayProperties()
{
   for (int i=0; i<tablei; i++)
   {
      PropertyTable* entry = table[i];
      printf("<%s>: <%s>\n", entry->key, entry->value);
   }
}

// constructor
Properties::Properties(char *filename)
{
}

// destructor
Properties::~Properties()
{
}

void Properties::init()
{
   char *files;
   char *l;
   char *propertyFile;

   base = 1;

   files = getenv("NEURO_PROPERTIES");
   if (!files) files = basePropertyFile;

   l = strdup(files);
   while (l && strlen(l))
   {
      propertyFile = strsep(&l, ":\n");
      readPropertyFile(propertyFile);
   }
}
```