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

A Dissertation Proposal
by
DEREK SHAWN HARTER

Submitted to the Department of Mathematical and Computer Sciences
University of Memphis

in partial fulfillment of the requirements for the degree of
DOCTOR OF PHILOSOPHY

December 2, 2002

Contents

List of Figures

List of Tables

Abstract

How does the brain compute? This, in its simplest form, is the question that the discipline of computational neuroscience is trying to answer. The view of the brain as, in some sense, performing a computation has been a unifying idea for all areas of cognition from neurobiology to artificial intelligence (AI). This idea has spawned many hypotheses on how neural activity gives rise to cognition. Much progress has been made in our understanding, from low level models of neural activity (e.g. Hodgkins/Huxley equations of neurobiology), to more abstract models of how groups of neurons might compute (e.g. parallel distributed processing (PDP) and artificial neural network (ANN) models of AI). However, despite much progress, a complete story of how brains compute, from the lowest levels of molecular biology to the mysteries of cognition and consciousness, still eludes us.

The fundamental idea of this proposal is this: Do more complex dynamics, such as oscillatory and chaotic behavior, play a role in understanding how brains compute? The hypothesis of the proposed research is that, yes, such dynamics do play an important role in the processes of cognition. The goal of the proposed research is to demonstrate how such mechanisms might operate in the perceptual, memory and behavioral systems of an autonomous agent. A secondary goal is to demonstrate what advantages, if any, chaotic neurodynamics might convey to the operation of an autonomous agent. The tool of choice are autonomous agents in virtual environments, with simple yet complete perceptual, motor, memory and motivational systems.

Chaotic dynamics in perception and behavior generation represent an additional level of understanding of how brains compute. Another insight to be explored in this research is the simplest possible architecture, making use of such dynamics, that

produces intentional, motivated behavior. The basic limbic system of vertebrates represents one such possible architecture. In this proposal I will present a plan for moving towards the demonstration of complex dynamics in a simulated limbic system to produce intentional behavior in an autonomous agent.

Chapter 1

Introduction

There have always been two main goals for people working in the field of Artificial Intelligence (AI). One goal has to do with building machines that can perform tasks that have heretofore required the intelligent activity of a human being. Examples include the diagnosis of disease, the playing of games like chess, or the exploration of environments for the purposes of discovery or more specific goals (performing scientific experiments, bomb disposal, etc.). The development of intelligent artifacts have often taken inspiration from how biological brains might perform these tasks. However, the main focus of such research has been towards the development of intelligent behavior, and has not been specifically concerned with the biological plausibility of the solutions thus produced.

The second main goal of AI has been as a tool of Cognitive Science in forming and testing hypotheses of how biological brains might be organized to perform intelligent behavior; and in particular the intelligent behavior of human brains. In such research, the goal is not only producing intelligent artifacts, but creating models that shed light on some of the processes of intelligent behavior in actual biological brains.

Up until the mid '80s, a hidden assumption of the Cognitivist, or Symbolic paradigm, was that the hard problems of intelligent behavior had to do with things like logical reasoning and linguistic abilities (Clark, 2001; Johnson-Laird, 1988). But

slow progress in scaling up models of such behavior have lead many to rethink this position. If the hard problems were once believed to be language and reasoning; then it was thought once those were solved, things like motor skills, spatial orientation and navigation would be easily conquered. But such a position has lead to systems that seem to be very inferior when compared to the performance of biological organisms. They can perform some very constrained tasks intelligently, but the systems are incredibly brittle and do not scale up well. Further, by pushing off the problem of interacting with the real world to mere perception, seemingly intractable problems such as the frame problem in robotics have continually plagued such approaches (Pylyshyn, 1987).

In reaction to these difficulties, many people have begun to turn the problem on its head. No longer are mere perception, motor skills, orientation and navigation seen as the simple problems whereas logic, reasoning and language are the hard problems. Perceptual and motor skills may seem like child's play to us, but this very appearance of simplicity hides a deep and fundamental ability of biological organisms to perform and act in the world. Perceptual and motor skills are now seen by many as the difficult problems that, once solved, can provide a firm foundation on which abilities such as reasoning and language can be built. It has been argued that a key component in producing such behavior in animals is the basic limbic system (Freeman, 2000), which provides the basic architecture needed for intentional behavior.

These types of problems in cognitivist models have lead to a renewed interest in taking inspiration from biological systems in order to better capture their flexibility and complexity of behavior. Such motivations have been behind the renewed interest in connectionism in the mid '80s, as well as continuing alternatives to the cognitivist paradigm of the computer metaphors of the brain. Biologically inspired approaches have lead to important advances in our conceptions of how brains function. Parallel

and distributed representations capture many important strengths and weaknesses of biological systems including: subsymbolic representations, pattern completion, learning, generalization and graceful degradation.

Later generations of connectionist thought have continued to extend the models to cover other important properties of biological behavior. By adding recurrent connections, connectionist models can deal with and produce dynamically generated temporal structure, or in other words, patterns that extend not only in space but in time. This is an important shift from perceptual patterns as relatively static entities, to more dynamic, real-time, environmentally complex patterns. Another important extension in connectionist networks is towards more complex, and biologically motivated architectures. These networks are moving away from the traditional three layer networks, to models with many processing areas or maps, and complex hard coded wiring and recurrent connections. Such work, in parallel with ideas of dynamics, embodiment and autonomous agents, are beginning to model complete organisms that perceive, act and learn within an environment.

This work, however, has continued to be carried out using neural models, abstracted from results in computational neuroscience, of the simplest variety. Oscillatory and chaotic dynamics are usually strictly avoided in such models. This leaves many questions open, however, on what use biological brains might have with all of the noisy and varying oscillatory patterns that they produce. In short, while the highly abstract ANN models have enlightened us in many fundamental ways on how neural tissue computes, it seems clear that many more fundamental principles remain to be articulated on how collective groups of neurons develop into perceptual, memory and behavior producing systems.

The continuing progression of connectionist models has produced impressive insights into how simple organisms may organize and develop perceptual and behavioral

patterns. We are now at a point of asking how such models can be further expanded to capture even more of the abilities of biological organisms. Current research into the neurodynamics of biological brains (Skarda & Freeman, 1987; Freeman, 1999b) suggests that one missing ingredient of connectionist models may be chaotic dynamics. Chaotic dynamics have been shown to occur in the mesoscopic [1] organization of neural patterns during perception (Skarda & Freeman, 1987). It has been hypothesized that such chaotic dynamics, far from being noise or an epiphenomenon, may actually be essential to the flexible formation of perceptual categories and behavior. The exploration of the uses of chaos as a fundamental property of the development of behavior in autonomous agents has not yet been adequately explored. If the hypothesized functioning of chaos as useful and necessary for biological brains is true, the use of chaos in models of perception and action selection may provide useful extensions to their capabilities. Further, it has been suggested (Freeman, 1991, 2000; Harter & Kozma, 2001a) that chaotic dynamics may be useful for more than just perceptual tasks. As chaotic dynamics allows for the flexible formation and instantaneous access of perceptual categories, it may also be essential in the formation of hierarchical goal states for intentional behavior. Chaotic dynamics has been shown to be useful in the self-organization of attractors for perceptual processing. The basics of intentional behavior have to do with the self-organization and formation of hierarchical goal-states of an organism that guide intentional behavior out into the environment. Just as chaotic dynamics may be necessary for the flexible self-organization of perceptual patterns, it may also be necessary for the self-organization of hierarchical goal-states of intentional behavior.

This proposal will first present an overview of some of the ideas and areas that are

[1] Mesoscopic dynamics refer to a level of organization between the microscopic activity of single neurons and the macroscopic activity of whole brain areas. Mescoscopic dynamics are generated by groups of interconnected and cooperating neurons.

key influences on the proposed research. This section includes a discussion of the dynamical cognitive hypothesis, a review of influential research combining autonomous agents and computational neuroscience relevant to this proposal, and a review of the literature of chaotic neurodynamics in neuroscience and cognition. The next section will present some preliminary results that have been obtained in developing a discrete and deterministic version of the K-Sets (Freeman, 1975, 1987) that will be used as fundamental units for the proposed research. These K-sets for autonomous agents (KA-sets) are faster, discretized versions, of the original continuous K-sets, suitable for computation in real-time autonomous agents. I will present work done with the KA-sets in implementing some of the fundamental principles of neurodynamics (Freeman, 1999b), including how they generate oscillatory and chaotic dynamics. I will also present some results of using hebbian mechanisms for learning among the oscillatory units of the KA-sets. In the final section I describe the proposed research project. I will describe the problems and agents to be used plus some preliminary ideas on what might be done to demonstrate the use of chaotic neurodynamics for the control of an autonomous agent and the simulation of basic intentional systems.

Chapter 2

Review

2.1 Symbolic Cognition

To date there have been two major approaches in Cognitive Science to the question of how minds work. These can be broadly described as the cognitivist, or symbolic approach, and the connectionist, or parallel distributed approach to cognition. While both viewpoints had their origins in the early 1900's and were articulated at the birth of AI and modern Cognitive Science, the symbolic approach has held sway for much of that time as the dominant viewpoint.

The symbolic approach to cognition is typified by 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 perception 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 and ACT-R.

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 toy 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 Connectionism

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 representa-

tions 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. In short, connectionist models are an attractive alternative model of cognition.

2.2.1 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 (Rosenblatt, 1958; McCulloch & Pitts, 1943), was revived in the mid '80s with the PDP research groups work (among others) on parallel distributed processing (Rumelhart, McClelland, & The PDP Research Group, 1986). First-generation connectionist systems were typified by a multi-layer architecture (usually composed of two or three layers) with strictly feed-forward connections. Such architectures are very familiar to practitioners of AI and Neural Network research. As stated previously, 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.

2.2.2 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.

2.2.3 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. I will discuss some examples of third generation connectionist models in section 2.5.

2.3 Dynamical Cognitive Hypothesis

Connectionist models have begun to be expanded to capture the more dynamic and time-varying properties of real biological networks. Another view of cognition, compatible with this dynamical connectionism, emphasizes the dynamics of change (from neurochemical, to electrical to behavioral). This view of cognition uses the formal language of dynamical systems theory and has been emerging recently in cognitive science as a result of the perceived defects of symbolic and standard connectionist models (van Gelder, 1998; Bechtel, 1998; Hendriks-Jansen, 1996; Clark, 1997, 2001; Port & van Gelder, 1995).

The paradigm of dynamics views what the brain does in terms of attractors, bifurcations, order parameters, instabilities and phase transitions in complex systems. There is a subtle, yet significant, shift in perspective when viewing cognition from a dynamical viewpoint. Neurobiological and connectionist models are increasingly interpreting their results in terms of dynamics. The behavior of simple elements in networks lends itself naturally to the language of dynamics, and symbolic analogs are hard to articulate at these levels. However, the dynamical paradigm is increasingly being extended to higher, more abstract levels of cognitive modeling. Dynamics increasingly forms the explanatory language in psychopyhsics, perception, developmental psychology, cognitive psychology, situated and autonomous agents, artificial intelligence and social psychology (van Gelder, 1998). It is central to a number of general approaches, such as ecological psychology (Gibson, 1979; Kugler, Kelso, & Turvey, 1982), synergetics (Haken & Stadler, 1990; Tschacher & Dauwalder, 1998), and morphodynamics (Thom, 1983). In this section I review some important research that views cognition through the use of dynamical explanations. We highlight some of the implications of viewing cognition through a dynamical perspective, and I relate this work to the proposed research.

2.3.1 Kelso: The Self-Organization of Behavior

J. A. Scott Kelso believes that cognition can best be understood as the unfolding of a dynamic system through time (Kelso, 1995; Kugler et al., 1982). His approach to understanding cognition emphasizes the concept of self-organization: the way groups of things cooperate to form emergent, coherent patterns. In Kelso's words:

> "The thesis here is that the human brain is *fundamentally* a pattern-forming self-organized system governed by nonlinear dynamical laws. Rather than compute, our brain 'dwells' (at least for short times) in metastable states: it is poised on the brink of instability where it can switch flexibly and quickly. By living near criticality, the brain is able to anticipate the future, not simply react to the present. (Kelso, 1995, p. 26)"

Elementary Concepts of and Conditions for Self-Organization

Kelso postulates seven concepts and conditions for self-organization to occur. In abbreviated form they are (Kelso, 1995, p. 16):

1. Patterns arise spontaneously as the result of large numbers of interacting components. The nature of the interactions must be nonlinear.
2. The system must be dissipative and far from equilibrium.
3. Relevant degrees of freedom (order parameters) are created by coordination between parts, but in turn influence the behavior of the parts (circular causality).
4. Order parameters are found near nonequilibrium phase transitions, where loss of stability gives rise to new or different patterns and/or switching between patterns.
5. Fluctuations continuously probe the system, allowing it to feel its stability and providing opportunities to discover new patterns. Fluctuations are positive sources of noise, not just something to be rid of.
6. Control parameters lead the system through different patterns, they do *not* act as a code or prescription for the emerging patterns.
7. The dynamics of the system may have simple (fixed point, limit cycle) or complicated solutions including deterministic chaos and stochastic aspects, giving rise to enormous behavioral complexity.

Kelso's seven conditions of self-organization are important for several reasons. First of all, by articulating such conditions we make concrete what is meant by self-organization, a term that is often abused. These conditions can act as essential

guide-posts towards developing systems that display emergent behavior in brain like ways. We can be more confident that our models exhibit self-organization if they meet the conditions proposed here.

The first condition is one common in connectionist models. An important point is that the interactions of the components must be nonlinear. Through nonlinear interactions develop patterns of the whole that are greater than and different from the behavior of the parts. Principle 5, dealing with fluctuations, is interesting in many ways. In a self-organized system, positive sources of noise are not problems, but are actually necessary to the correct functioning of the system. The purpose of fluctuations, as stated, is to probe the system continuously such that new stable patterns may be found. Deterministic chaos provides a controlled source of noise that fulfills the requirement of fluctuations for self-organization.

Illustration of a Dynamic Model: Phase Transitions in Rhythmic Behavior

As a simple illustration of a dynamical model of behavior in this section I will briefly present the Haken-Kelso-Bunz (HKB) model of phase transition's in rhythmic finger movements. Many patterns of locomotive behavior are based upon oscillatory rhythms (though, interestingly, not completely periodic as the patterns must maintain a certain amount of flexibility in order to deal with environmental irregularities). For example, walking is basically rhythmic motion of alternating limbs. Horses have 3 basic gait patterns: walking, trotting and galloping. It is a well-known fact that horses (as well as other quadrupeds and humans) employ restricted ranges of speed for any given mode of locomotion (Kelso, 1995). Going to fast or to slow out of this range causes a spontaneous transition into a different mode of locomotion. Interestingly, the preferred speed for any mode of locomotion is such that it is the most efficient speed in terms of energy expenditure vs. oxygen consumption (Hoyt & Taylor, 1981). This is a common example of an order parameter of developmental dynamics, in which

embodied proprioception of bodily functions helps to develop patterns that are close to optimal for certain purposes.

Haken, Kelso, and Bunz (1985) developed a dynamical model of a simple phase transition in rhythmic behavior. In this experiment, they asked people to wave the index fingers of their hands back and forth (like windshield wipers on a car). Their fingers were hooked up to devices that could measure their position (or angular displacement) over time. The were asked to rhythmically wave their fingers to the beat of a metronome. At low speeds there are two stable states (attractors) of behavior. We can call these in phase (both fingers are at the same angle, relative to the center of the body), or out of phase (exactly 180°) apart). At high speeds, one of these patterns (out of phase) becomes unstable and is unable to be naturally produced. When the speed of the metronome is gradually increased, people naturally shift from an out of phase pattern to being in phase once a critical speed is reached.

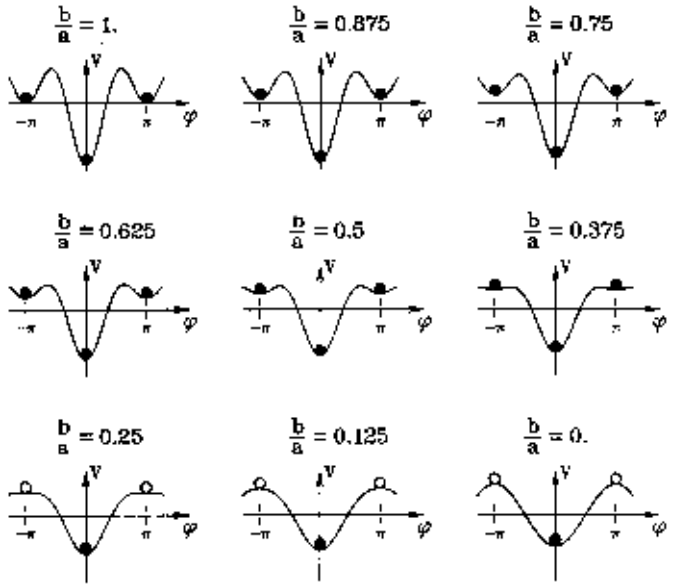

Figure 2.1: The HKB model of coordination. Upper left is low speed of the metronome, which transitions to the lower right and a high speed (from Kelso, 1995, p. 55)

Haken, Kelso and Bunz were able to develop a first order differential equation that describes the dynamics of this rhythmic behavior. Figure 2.1 shows a visual representation of the state space of this system. In these figure, the speed of the metronome acts as an order parameter of the system. In the upper left panel, the speed is very low, and there are two stable attractors of behavior of the system (at 0° and $\pm\pi^\circ$ relative phase). As the speed of the metronome increases (moving from left to right and then top to bottom), the out of phase stable attractor basin eventually disappears. This point engenders a behavioral phase transition from out of phase to in phase behavior.

2.3.2 Thelen and Smith: Developmental Systems, Dynamics and Cognition

The ontogenetic development of behavior provides a powerful mechanisms by which organisms learn to organize effective patterns of behavior for performing the necessary tasks of survival (Thelen & Smith, 1994; Iverson & Thelen, 1999; Oyama, 1985; Thelen, 1995). There are many properties of this type of development. It is fundamentally a self-organizing process, in which the constraints of body and environment guide the system towards discovering certain patterns of behavior. Development of behavior in organisms is not so much a process of finding complex chains of effective behaviors, but in finding salient perceptual cues and effective manipulations that simplify and transform the task environment into problems that are directly recognizable and solvable. Problem solving in natural cognitive systems is more often the application of many transformations until the problem is sufficiently simplified to be directly solved. Clark (1997) calls such phenomena action loops. Kirsh and Maglio (1994) call actions that are primarily performed to transform and simplify the task environment epistemic actions.

Thelen and Smith (Thelen & Smith, 1994; Thelen, 1995) envision the development

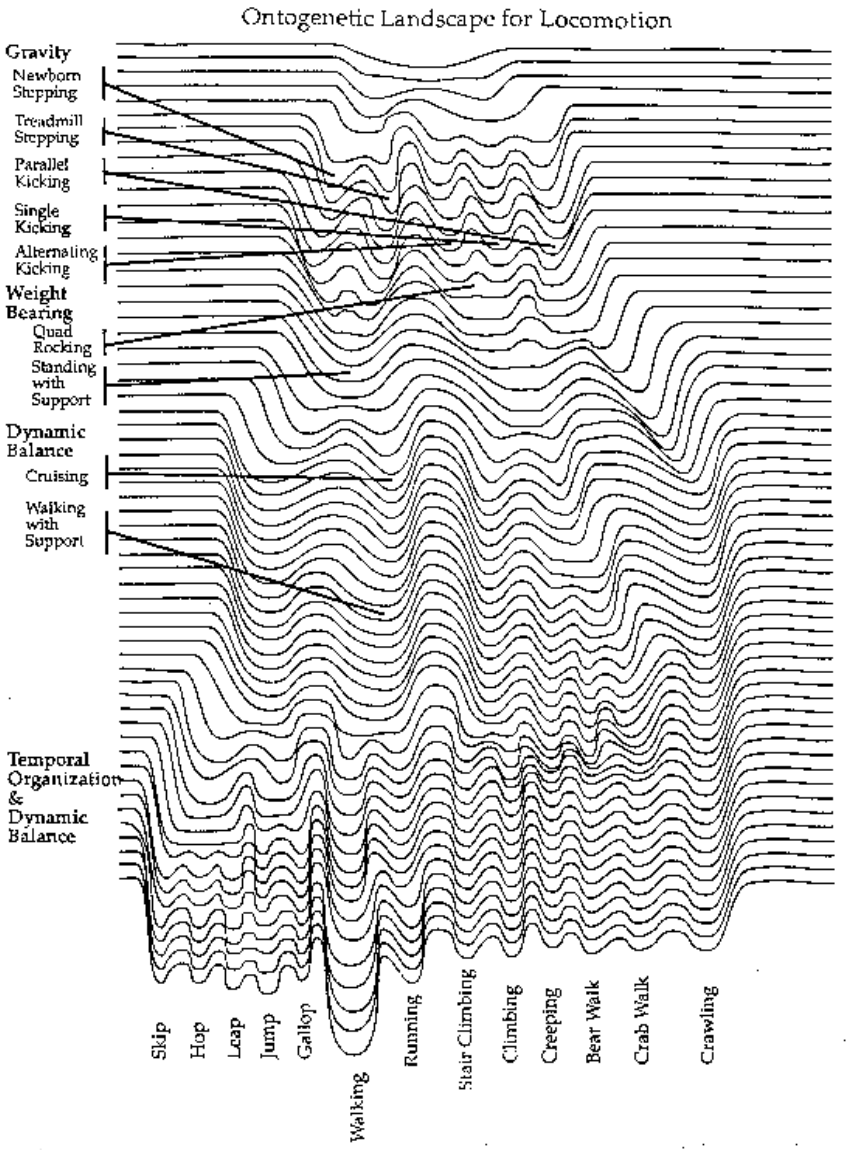

Figure 2.2: Metaphorical ontogenetic landscape for locomotion (from Thelen & Smith, 1994, p. 124)

of behavior in cognitive systems as an ontogenetic landscape of stable and unstable attractors and repellors (figure 2.2). 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.

Natural cognitive systems display both physical and behavioral development. Physical changes in a maturing organism are continually reshaping the ontogenetic landscape, destabilizing previously stable solutions, and forcing the system into finding new patterns of behavior. Natural cognitive systems also display this flexibility in the development of behavior for problem solving. Sequences of behaviors are not learned so much as behaviors that change the state of the environment and thus cue the next behavior. Therefore chains of complex behaviors are seen as loops of simple perception/action feedback pairs that reliably transform a problem into a desired goal or result.

2.4 Embodiment

Embodied cognition is an emerging viewpoint in cognitive science that emphasizes many differing aspects from the standard cognitive hypothesis (Clark, 1997; Hendriks-Jansen, 1996; Pfeifer & Scheier, 1998; Varela, Thompson, & Rosch, 1993; Franklin, 1995, 1997). Many of the ideas of embodied cognition are compatible with and complement a dynamical perspective.

In the standard view of cognition, the mind is the product of the manipulation of symbolic representations of the problem in order to produce solutions and generate intelligent behavior (Johnson-Laird, 1988; Newell & Simon, 1972, 1976; Newell, 1990). The environment is perceived and transduced into symbolic representations. These symbols encode the current state of the environment and the problem to be solved. They can be manipulated, independent of the environment, to discover solutions to the problem and produce intelligent behavior for the organism. In an embodied view of cognition, intelligence in biological organisms does not arise through the

static manipulation of amodal symbols and representations. Instead, organisms are seen to be embedded in their environments in fundamental ways. Through their real time experiences with their bodies and environments, they begin to embody the salient aspects of situations in ways that guide future perception and behavior towards improved performance. Experience with their ecological niche develops expectations of the environmental regularities that are of benefit to the intrinsic needs and desires of the organism. The organism actively learns to seek out expected stimuli that are relevant to the desires and needs of the organism at a particular moment.

There are many concepts associated with an embodied perspective of cognition. I will briefly present some of the more important ones in the next sections.

2.4.1 Embodied Organisms are Complete Organisms

Biological organisms are currently the only examples capable of producing a full range of intelligent, adaptive behavior. Standard views of cognition place no special emphasis on the fact that these natural examples of cognition are **complete** organisms. In the standard view of cognition, it seems plausible that by connecting together many specialized subsystems that solve problems in limited, specialized domains, eventually a complete intelligence will be produced.

According to an embodied perspective, we are not likely to understand natural cognition from such a piecemeal approach to studying and building systems. Instead, we must examine and build complete cognitive systems. In this context, complete refers to systems that are autonomous and adaptive. Autonomous systems are those that have certain intrinsic needs, and that are able to produce behavior that is capable of satisfying those needs consistently over time. Pfeifer (1996), Pfeifer and Scheier (1998) characterizes autonomy as the ability of the organism to maintain its critical, intrinsic values within a zone of viability. This is often referred to as "homeostasis". Adaptivity refers to organisms that are capable of modifying their behavior so that

they can more efficiently maintain their critical parameters in their zones of viability (Franklin & Graesser, 1997).

Studying complete cognitive systems is important for several reasons. Classical approaches to modeling cognition often tackle restricted problems in limited domains. The hope is that the techniques developed can then be scaled up to the full problems of cognition. Embodied cognition, with its emphasis on complete systems, maintains that the answer is not to start with restricted portions of the cognitive system. Instead we should begin by studying simple, but complete, organisms, in more realistic environments (Brooks, 1990; Pfeifer & Scheier, 1998). Only complete organisms are capable of developing embodied representations and displaying intentional behavior.

2.4.2 Active, Action-Oriented Representations

Another important difference of embodied and classical perspectives concerns the nature of the representations developed and used by the organism. In a classical perspective, symbols are seen as passive structures that are syntactically manipulated to produce solutions. In an embodied perspective, representations are much more intimately tied to the intrinsic needs of the organism. Clark (1997) calls such structures *action-oriented representations*. Action-oriented representations are not passive representations of the state of the environment as it exists at some time. They are continuously updated from sensory information, and they continuously prescribe possibilities for action. In this sense they are not *representations* at all, in the traditional meaning of the word. Gibson (1979) has called this the concept of affordances, where the representations *afford* opportunities for action for the organism.

2.4.3 The World Represents Itself

Classical models of cognition often experience an exponential explosion of computational power as the environment increases in complexity. An embodied approach to

cognition avoids this problem because it advocates the use of simple, cheap, action-oriented representations. From an embodied perspective, it is better to use cheap and active sensing to inform oneself of the state of the environment, rather than building complex representations of the environment. Brooks (Brooks, 1995) states this principle as "the world is its own best model". Embodied cognition avoids the use of costly and detailed representations. Cheap, quick, active, specialized sensing of the environment is preferred. Instead of maintaining a complex representation of the state of the environment, we simply direct specialized sensory apparatus to directly perceive the information required for behavior. This approach helps keep the need for computation from exploding in complex environments.

2.4.4 Emergence of Solutions through Collective Activity

A key concept of embodied cognition is the emergence of solutions from many parallel, distributed activities. In an embodied perspective, intelligence is seen as emerging from the parallel activity of many cooperating and competing processes. As in connectionist models, parallel emergence of solutions provides many benefits to the behavior of the system. Such emergent solutions are robust and resistant to damage; tolerant of noisy, incomplete data; satisfy general goals and yet are variable and context dependent. They are also fast, able to produce solutions easily in real time demanding environments. Unlike most classical connectionist modeling, embodied cognition views recurrent, non-linear interactions as a crucial property in the emergence of solutions.

2.4.5 Developing Within the Environment

The emergence of solutions through many parallel processes is not simply a product of the non-linear interactions of components in the organism's brain. Intelligent behavior also emerges as the product of the interaction of simple behaviors with

a complex environment. Simple, instinctive behaviors are seen as intelligent when they are coupled with local environmental cues (Braitenberg, 1984). Development of action-oriented representations aids in this process. Organisms learn simple actions that, when coupled with appropriate learned stimuli, yield intelligent, purposeful behavior.

Clark (1997) states that embodied minds use extensive external scaffolding, where the environment itself is used as a type of external memory. The ecological niche of the organism provides many consistent cues for intelligent behavior. Most intelligent behavior in natural organisms involves the fast recognition and exploitation of such opportunities, not in complex planning and reasoning. Also, most organisms tend to offload complex planning and reasoning tasks onto the environment. They do this by allowing the state of the environment to represent the progression of the problem solving task. One example, given by (Rumelhart et al., 1986), is in the behavior of people when multiplying large numbers. Most people can instantly recognize and produce the answer to simple, single digit multiplication problems, of the type 7 x 7 = 49. However, when given the task of multiplying large numbers together, say 4356 X 1897, they invariably resort to pencil and paper (when denied the used of a calculator). People do not compute large chains of complicated reasoning and logic. Instead they offload the representation of the progress of the task onto the environment by maintaining the state of the problem solving task with environmental cues. In this case, people make marks on paper (the environment) to keep track of their problem solving progress, while reducing the problems to those simple ones that they can directly recognize and solve. Embodied cognition sees this type of external scaffolding not as simply useful, but as a prevalent and pervasive method used by cognitive systems to reduce computational complexity and perform problem solving tasks in real time.

2.4.6 Better Imperfect than Late

Biological cognition is exemplified by fast pattern completion. It has evolved to produce behavior in real time. The behavior does not necessarily have to be perfect, so long as it is good enough for the continued survival of the organism (at least until the next crisis occurs). Organisms are continually presented with threats and dangers that must be handled immediately in order to ensure their survival. Such requirements do not favor solutions that take large amounts of time. Natural cognition seems to be built upon a foundation of fast pattern recognition and behavior generation keyed to threats and opportunities for action. The embodied cognitive viewpoint recognizes this fundamental feature of natural cognitive systems. According to Port and van Gelder:

> "The cognitive system is not a discrete sequential manipulator of static representational structures; rather, it is a structure of mutually and simultaneously influencing *change*. Its processes do not take place in the arbitrary, discrete time of computer steps; rather, they unfold in the *real* time of ongoing change in the environment, the body, and the nervous system. (Port & van Gelder, 1995, p. 3)"

2.5 Autonomous Agents in Computational Neuroscience

One reaction to some of the typical problems encountered by the cognitivist paradigms is to begin to study and emphasize complete autonomous agents (Pfeifer & Scheier, 1998; Maes, 1990; Steels & Brooks, 1995; Brooks, 1995). By focusing on complete, if simple, agents (the thinking goes) we force ourselves to face the hard problems of perception and action right from the beginning. Rather than focusing on narrow, isolated faculties (memory, language, vision) and tasks (memorizing lists, reaction times), the study of complete agents allows us to study how such disparate systems might be integrated to perform the functions of cognition. The hope is that by studying integrated systems we will gain more (or new) insights into cognition than

can be gleaned by studying the components in isolation. In addition, the development of complex behavior may in fact be as much a result of the coupled interaction between the biological organism and a complex, changing environment. Models of complete agents in complex environments are more likely to offer insights into these types of developmental systems.

Complete agents are usually perceived as having a number of properties. They are self-sufficient and autonomous. This means that they have their own goals and needs, and are able to act within their world to satisfy these need. Autonomous agents are situated in their environments. They are embodied. They display adaptivity but, at the same time, are often surprisingly specialized for operations within their ecological niche (Franklin & Graesser, 1997; Pfeifer & Scheier, 1998).

A dynamical perspective often plays a role at many levels of explanation in complete autonomous agent research. For example, many people conceive of the agent and environment as a tightly coupled pair of interacting dynamical systems. In this view, behavior does not completely originate from the internals of the agent, nor is it completely a property of the environment. Instead, behavior is an emergent property of the agent/environment interaction. A conclusion of this insight is that it is not enough to study the cognitive agent if you completely wish to understand its behavior. The agent/environment system must be studied together to fully understand the behavior.

The research proposed here is very much in the tradition of complete agent approaches to cognition. I am proposing to explore two major questions in the light of complete autonomous agents. First of all, what roles might complex dynamics, like chaos, play in the formation of memory, perception and behavior. And secondly, what insights might we glean from the limbic system that will improve the performance of agents in producing intentional behavior. In the rest of this section I present a

few examples of work that is combining models from neurobiology and computational neuroscience, with an autonomous agent perspective.

2.5.1 Verschure and Pfeifer: Distributed Adaptive Control (DAC)

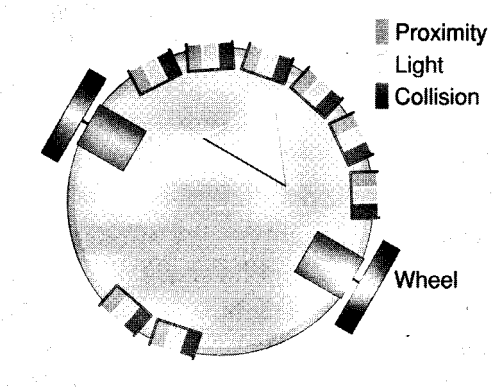

Figure 2.3: Generic agent scheme used for DAC experiments (Khepera like) (from Pfeifer & Scheier, 1998, p. 155)

The first model I will discuss, is a simple, yet instructive, model of classical conditioning. Distributive Adaptive Control (DAC), developed by Paul Verschure and Rolf Pfeifer (Verschure, Kröse, & Pfeifer, 1992; Verschure & Pfeifer, 1993; Verschure, Wray, Sporns, Tononi, & Edelman, 1995; Verschure, 1998; Pfeifer & Scheier, 1998), illustrates some of the strengths of the complete agent approach to studying cognition. The agent used was the khepera robot, shown in figure 2.3, which has three types of sensors: collision detectors, proximity sensors (infra-red) and target sensors (light). Eight sensors of each type are arranged around the body of the agent, mostly concentrated in the front part of the body.

The architecture for the DAC simulations is shown in figure 2.4. In this figure, solid arrows represent hard coded connections, while dotted arrows are weights that will be modified by learning. As can be seen, there are areas of units, labeled C, P and T, which receive stimulation from the collision, proximity and target sensors

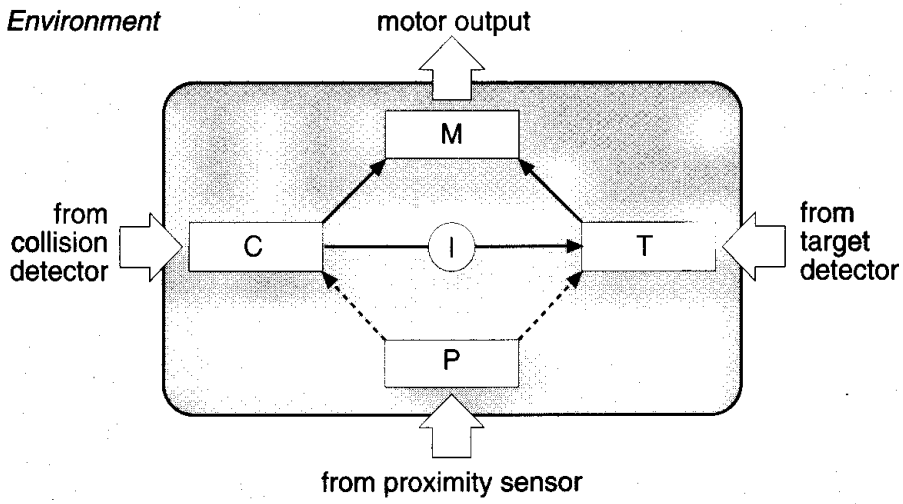

Figure 2.4: Architecture for DAC experiments (from Pfeifer & Scheier, 1998, p. 164)

respectively. These areas each contain 8 simple neural units, that receive stimulation from a single one of the sensors. The motor area M contains units that produce three simple behaviors: turn left, turn right and move forward. The robot is initially hard coded with a set of instinctive, reflexive behaviors. In this case, the weights between the C and M areas are hard coded such that, if there is a collision with an object on the right side of the agent, the agent turns to the left. Similarly the agent reacts to a collision on the left by turning to the right. When no collisions are detected, the agent is hard coded to move forward.

In the first simulation, in which we ignore the use of the target sensors for now and concentrate solely on the collision and proximity sensors, the robot initially wanders forward in its environment, colliding with objects, turning away from them and moving on to collide with the next object. The units in the proximity area (P) are fully connected with the units in the collision area (C), and these weights are plastic and can be modified through hebbian learning. Whenever the agent collides with an object, both the collision and proximity sensors will be highly active. Because of the coactivation of activity between these units, the weights between corresponding

collision and proximity sensors will be strengthened. Eventually, the activation of the proximity sensors alone will be enough to stimulate the appropriate collision sensor and cause avoidance behavior to be produced. Since the proximity sensor becomes active at a distance, the resulting behavior is that the robot learns to avoid objects at a distance without colliding into them (figure 2.5).

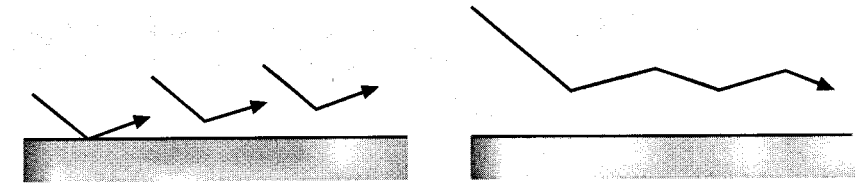

Figure 2.5: Development of DAC behavior over time. (left) Obstacle avoidance behavior. Initially the robot hits obstacles. Over time, it starts turning away before hitting. (right) Wall-following behavior. When light sources are found along walls, wall-following behavior emerges over time (taken from Pfeifer & Scheier, 1998, p. 161)

In the second simulation, the target sensors are now used to develop a type of wall-following behavior. In this simulation, light sources are placed around the outside edges of the wall. The target area units (T) are hard coded to cause the agent to turn towards and approach the strongest light source (indicated by strongest stimulation of one of the light source sensors). The I unit in figure 2.4 provides inhibition between the collision and target areas. The behavior of the agent is to approach and collide with the wall. At this point the target area units are inhibited by the collision area, and reflexive turning away is performed. When the agent has turned sufficiently far away, inhibition decreases which allows approach to again occur. The resulting behavior is to follow the wall, as shown in figure 2.5. If learning is allowed to occur between the proximity (P) and target (T) layers, eventually the robot learns to associate light sources with proximity to walls. After this has happened, the light sources can be removed and the agent will continue to approach walls. If we think of the light as representing food sources, we can say that the robot has learned that food is to

normally be found along walls. It follows walls now, even in the absence of light, in the hopes of finding food.

In this simulation we can describe the behavior of the agent as learning to correctly generalize from experience. It starts avoiding obstacles and finally ends up anticipating them as a result of the conditioning between collisions and proximity sensors. In a similar manner, light sources are appropriately generalized to approach behavior of any wall, through the coactivation of proximity and target sensors.

2.5.2 Edelman: DARWIN

The next agent I will present is work done by Gerald M. Edelman and research associates on the DARWIN agents (Edelman et al., 1992; Almássy, Edelman, & Sporns, 1998; Friston, Tononi, Reeke, Sporns, & Edelman, 1994; Sporns, Almássy, & Edelman, 1999; Edelman, 1987; Edelman & Tononi, 2000). The DARWIN series of models have been developed by Edelman et al. since the early 1990's. The purpose of the various models have been to demonstrate some of the principles of Edelman's theory of neuronal group selection and recurrent maps. I will concentrate solely on the latest incarnation, DARWIN V, that has been implemented on a real world robot called NOMAD.

The purpose of DARWIN V was to explore questions of the specific roles of self-generated movements and behavioral interactions with an environment in the development of complex neuronal properties (Almássy et al., 1998). In other words, how important are active (as opposed to passive) interactions with the environment in developing representations and behaviors. The DARWIN V architecture is shown in figure 2.6. The DARWIN robot comes equipped with two types of sensors, a color camera (with 64x64 pixel resolution) and a sensor that registers electrical conductance of objects it touches (which they equate with a type of taste sense).

In the agents environment are placed two types of objects that have regular re-

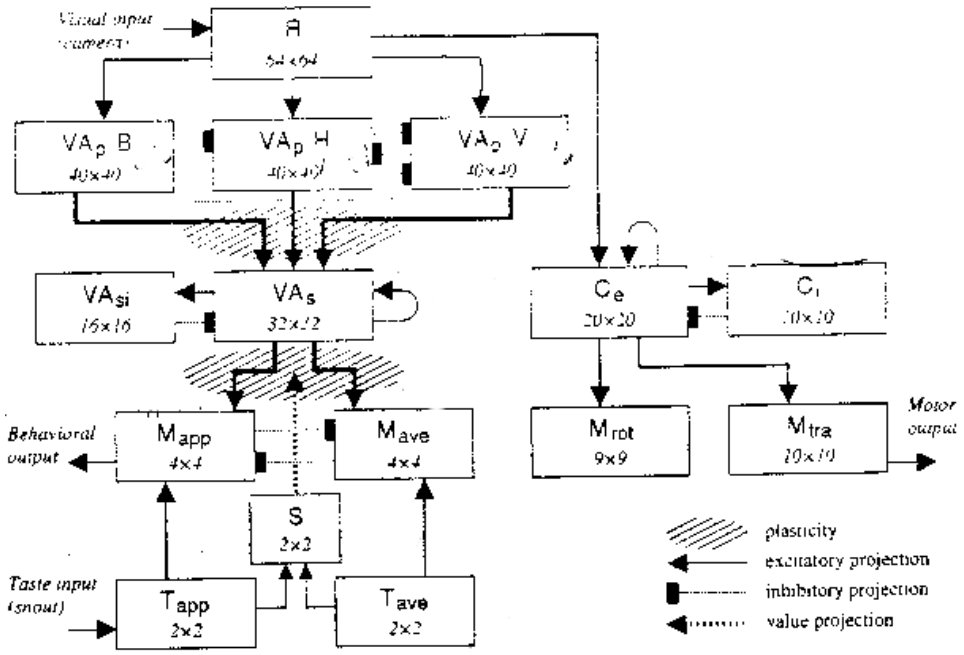

Figure 2.6: Architecture for DARWIN V experiments. Each box represents a modeled area containing neuronal units, numbers within boxes indicate dimensions of the neuronal array (from Almássy & Sporns, 1998, p. 316).

lationships between the properties that the agent is sensitive to. Some objects are painted with blobs (visual) and are made of metal so that they conduct electricity (taste). Other objects are painted with stripes and are coated with a material that will not conduct electricity. In the experiment, nonconductive objects (the stripey ones) are considered to taste good to the agent (e.g. a source of food) and are hard coded into the agent's value system as such. Blobby objects, that are conductive, are poisonous to the agent. The agent is hard coded in such a way that initially it approaches and tastes all objects it comes across in its environment. The goal of the experiment, similar to the DAC setup, is to learn which visual properties (blobs or stripes) are reliably associated with good tasting or bad tasting food, and to learn to avoid the blobby objects without having to actually taste them, simply by seeing them at a distance and avoiding them.

In the DARWIN V architecture (figure 2.6), the taste sensors are hard coded

to the appetitive (T_{app}) and aversive (T_{ave}) areas, which in turn cause appetitive (M_{app}) and aversive (M_{ave}) motor behaviors. These connections are hard coded, as described previously, for the good and bad tasting objects. The colliculus (C_e and C_i) areas receive input from the reticular (R) formation (camera sensor). These objects are hard coded to sense the brightest area in the camera's field of view, which will correspond to the closest object, and to then rotate (M_{rot}) and move (M_{tra}) the robot to approach that object. The effect of these hard coded connections is to cause the agent to approach the closest object and attempt to taste it. The reticular (R) area projects into three areas (VA_pB, VA_pH and VA_pV) which are coded to respond selectively to blobs, and horizontal and vertical stripes in the camera view. The results of these three visual processing areas are integrated into the VA_s area. Connections between the three VA_p and VA_s areas are plastic, as well as those between VA_s and the appetitive and aversive motor areas (M_{app} and M_{ave}). Plastic connections are those that are subject to modification through hebbian learning. In the DARWIN experiments, learning only takes place when salient events occur (such as tasting good, or bad food). The value area (S) turns on hebbian plasticity during these important events.

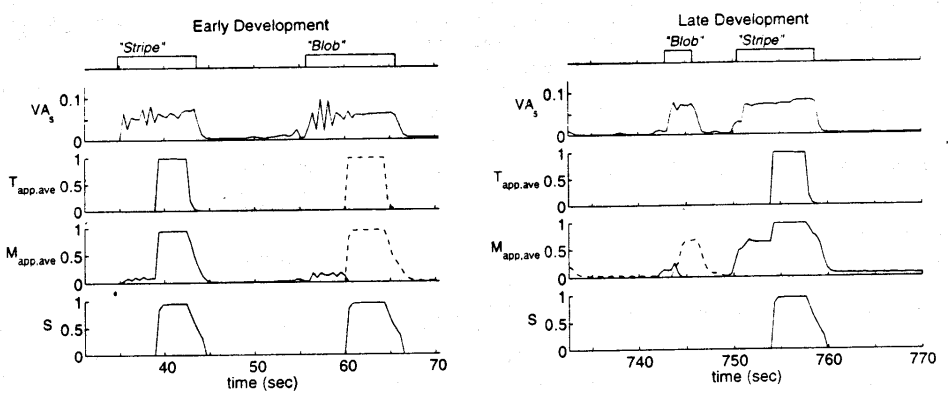

Figure 2.7: Comparison of average group activity in DARWIN V during early (left) and late (right) visual development. Solid lines are appetitive activity, while dashed lines are aversive (from Almássy & Sporns, 1998, p. 316)

Initially the weights from the visual (VA) to the motor (M) areas are not strong enough to affect behavior. However, whenever the robot finds and tastes a good object, learning is turned on. In this case, certain patterns (from striped, good tasting objects) will be active in VA_s. Since these patterns are active when the appetitive behavior is active (M_{app}), these weights will be strengthened. Likewise, when a bad tasting blobby object is encountered, weights will be strengthened between blobby patterns in VA_s and M_{ave}. Eventually, the weights are strong enough between VA_s and the motor areas that they can activate appetitive or aversive behavior on their own, simply from visually identifying the object. Some sample results of the activity of units before and after learned behavior has developed are shown in figure 2.7. In the left panel showing an early phase of the simulation, the robot must taste the bad tasting blob before aversive motor behaviors become active. In the right panel, however, when it sees the blob in the visual area this causes the appetitive motor behavior to become active, before the robot actually gets to taste the object.

2.6 Chaotic Dynamics in Cognition

That the brain is a complex nonlinear dynamical system seems beyond dispute (Ward, 2002, p. 237). When the well known philosopher and neuroscientist, Patricia Churchland was asked this question she replied that it was obviously so, but she then went on to question what, if anything, saying this does to enhance our understanding of how brains work (Kelso, 1995). Some believe that understanding the brain and its relationship to behavior and mind does require an understanding of its dynamics. If the brain is a nonlinear system, do the dynamics create chaotic or edge-of-chaos functioning? Is the chaos high dimensional, or low dimensional? What functions, if any, does chaotic dynamics play in perception, memory and behavior?

ANN models have greatly increased our understanding of how neural like units

might collectively achieve certain results, such as pattern completion, generalization and graceful degradation of performance. They have proved useful in modeling some types of perception and behavior generation in complete autonomous agents. In the past most classic ANN models have relied solely on point attractor dynamics to achieve their results. Oscillatory dynamics, while not completely ignored, have played much less of a role, and chaotic dynamics even less so, especially in the area of autonomous agent research. Brains are saturated with oscillating and chaotic patterns. There remains much yet to be explained of how this type of activity in the brain contributes to mental life. Many principles have yet to be fully articulated in formal models and demonstrated in actual systems.

Many phenomenon that appear in nature (outside of the realm of cognition) are beginning to be explained by invoking concepts of chaotic dynamics. For example in predator-prey models (Solé & Valls, 1992; Hassell, Comins, & May, 1991), extinction of species (Solé, Manribia, Benton, Kauffman, & Bak, 1997), growth of cities (Makse, Havlin, & Stanley, 1995), traffic patterns, heart rhythms (Poon & Merrill, 1997) and many more. Models that use chaotic dynamics to explain cognitive behavior are also beginning to appear in many places (Ward, 2002; Beer, 2000). Psychological models, that invoke chaos, have been done for mental illness (Scheier & Tschacher, 1996) and memory (Clayton & Frey, 1997) to name a few.

Chaotic dynamics have begun to be explored as possible principles for the organization of behavior in biological brains (Freeman, 1991, 1999b; Tsuda, 2001). In this section I will present the most well known and detailed analysis of chaos in the operation of perception, Freeman's work with the olfactory system of the rabbit.

2.6.1 Chaotic Neurodynamics in Olfactory 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 dy-

namics. 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.

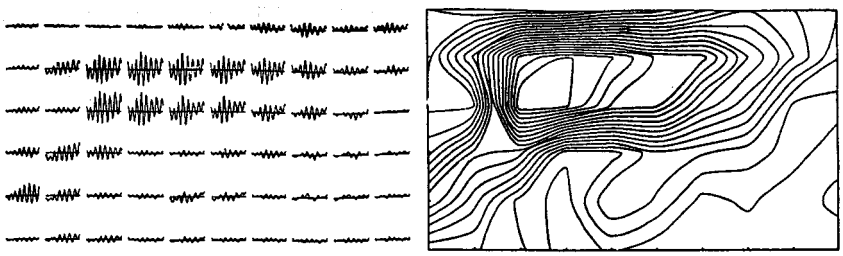

Figure 2.8: EEG carrier wave patterns (left) and contour map (right) of olfactory cortex activity in response to a recognized smell stimulus (from Freeman, 1991, p. 80)

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 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 2.8 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 spatiotemporal 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 (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.

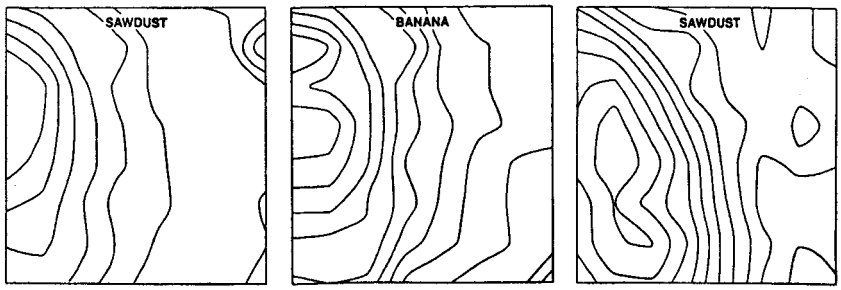

Figure 2.9: Change in contour maps of olfactory bulb activity with the introduction of a new smell stimulus (from Freeman, 1991, p. 81)

In figure 2.9 is shown 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 is shown the contour pattern of activity for sawdust (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 patterns continually produce 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. 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, as discussed in section 3.6, 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.

2.6.2 Freeman's K-Sets: Modeling the Olfactory System

From these experiments and data collected from the olfactory system of rabbits, Freeman and associates developed a mathematical model of neural populations called the K-model (so named in honor of Katchalsky, a famous and influential neuroscientist) (Freeman, 1975, 1987). The K-models dynamics are designed to model the dynamics of the mean field (e.g. average) amplitude of a neural population. A nonlinear, second order, ordinary differential equation was developed to model the dynamics of such a population. The parameters for this equation were derived by experimentation and

observation of isolated neural populations of animals prepared through brain slicing techniques.

The basic ODE equation of a neural population of the K-model is:

$$AQ(x(t)) + RI(t) = \frac{1}{ab}\frac{d^2}{dt^2}[x(t)] + \frac{a+b}{ab}\frac{d}{dt}[x(t)] + x(t) \tag{2.1}$$

In this equation x(t) is the activity level (mean field amplitude) of the neural population. a and b are timing constants (derived from observing biological population dynamics, as in figure 3.7). The right side of the equation expresses the intrinsic dynamics of the population unit.

On the left side of the equation are factors that allow for external input to the population (RI(t)) and stimulation from other populations of units (A is the weight matrix). Q is the transfer function used in the model, which was derived again from observation of biological populations that relates the mean field amplitude of a population to the stimulation it gives to connected populations. Q takes the form of an asymmetric sigmoid function, and is given by the equation:

$$Q(v) = Q_m\{1 - exp[\frac{-(e^v - 1)}{Q_m}]\} \tag{2.2}$$

where Q_m is a parameter that indicates the level of arousal in the population (high values indicated a more aroused, motivated state), and v is the mean field amplitude (x(t) in the previous equation).

As stated, these equations model the dynamic behavior of the activity of isolated neural populations. In Freeman's K-model, these are the basic units that are connected together to form larger cooperating components. Two excitatory or inhibitory units together form a KI set. A KI excitatory with a KI inhibitory pair form a KII set of four units (see figure 3.17 and section 3.2.4).

Freeman and associates used these models of neural populations to construct a model of the olfactory system and replicate the dynamics observed from the EEG

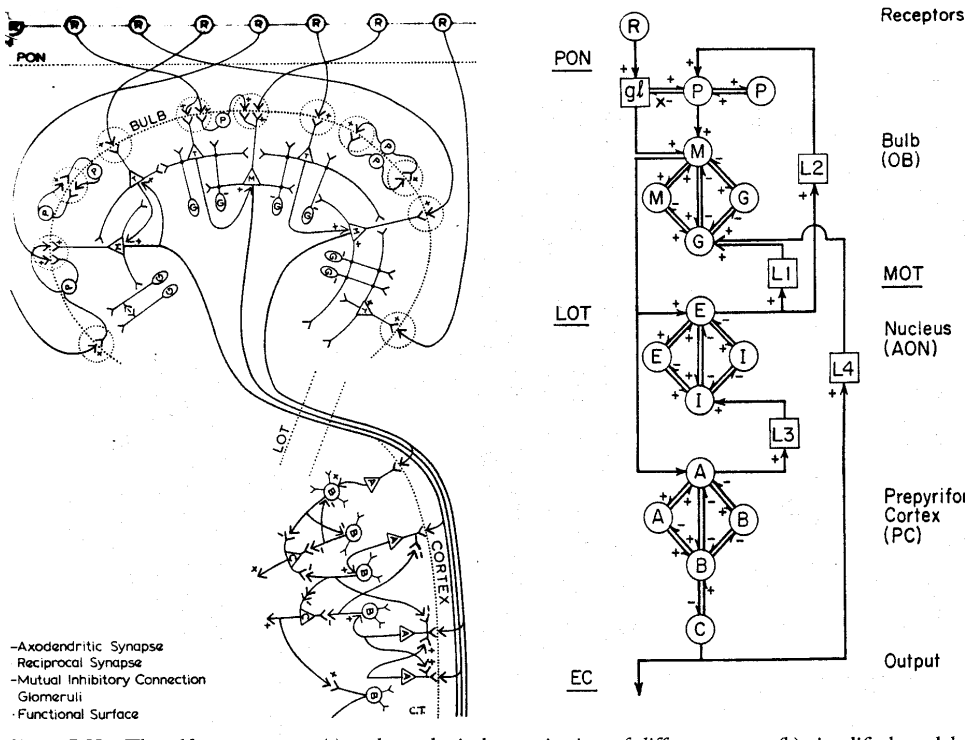

Figure 2.10: The olfactory cortex: (left) real topological organization of different areas; (right) simplified KIII model of the olfactory bulb (from Skarda & Freeman, 1987, p. 165)

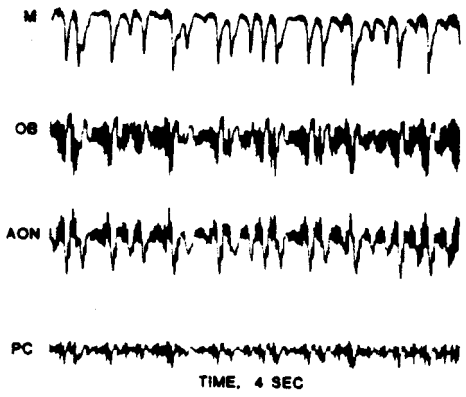

Figure 2.11: Simulated chaotic background activity generated by the KIII model (from Skarda & Freeman, 1987, p. 166)

experiments. In figure 2.10 is shown, on the left, a schematic diagram of the olfactory system, and on the right, the simplified KIII model developed to replicate the dynamics observed in olfaction. In this figure, you can see three groups of KII

units (M/G, E/I and A/B pairs), connected together. These three groups represent mitral/granule (M/G) dynamics in the olfactory bulb (OB); excitatory/inhibitory (E/I) dynamics in the anterior olfactory nucleus (AON); and excitatory/inhibitory A/B pair dynamics in the prepyriform cortex respectively. Three or more groups of KII units connected together form a KIII unit, which is capable of producing chaotic dynamics (see sections 2.6.3 and 3.2.4).

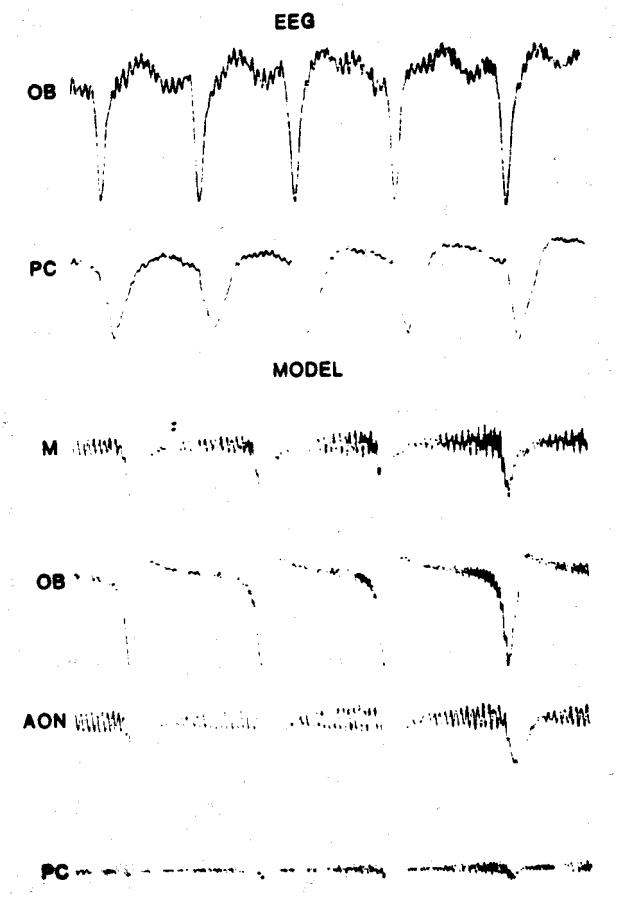

Figure 2.12: Comparison of biological EEG recordings of OB and PC to those generated by KIII model (from Skarda & Freeman, 1987, p. 167)

The KIII model is used to replicate the dynamics observed in the olfactory bulb. For example, in figure 2.11 we see some examples of chaotic background activity (e.g. the attentive state, before stimulus) generated by the model. In figure 2.12 are comparisons of biological EEG recordings and those produced by the KIII model.

2.6.3 Principles of Chaotic Neurodynamics

As a result of his work in olfaction and the K-models, Freeman (Freeman, 1999b) has developed a set of principles of how chaotic dynamics is formed in brains, and how these contribute to producing cognition. These principles of chaotic neurodynamics are reproduced in their entirety in appendix A. The KI, KII and KIII models are reflected in the first four principles, which state how neural populations produce oscillatory and chaotic dynamics through particular types of feedback mechanisms. Principles 5-7 deal with the creation of the carrier wave, and the formation of meanings through synaptic modifications by spatial patterns of amplitude modulation. These principles codify the results of the formation of spatial activity patterns in the olfactory bulb. The remaining principles (8-10) propose principles that lead from the recognition of meanings, to intentional behavioral patterns. The creation of intentional behavior through brain dynamics will be discussed in the next section.

These ten principles represent a new set of ideas on how brain dynamics may come to embody meaning and intentional behavior in biological brains. They go beyond the principles that have been articulated by standard ANN modeling in AI, cognitive science and autonomous agents. To date, these principles have not been implemented in any full scale autonomous agent, to test their usefulness as models of cognition.

2.7 Basic Intentional Systems

I now turn from a discussion of chaotic neurodynamics embodied in the first 7 principles (appendix A), to the formation of intentional behavior patterns. Intentional behavior, in the words of Freeman (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 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 endogeneous (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 in the service of the needs

and desires of the organism. This formation is the same as Freeman's tenth principle of the formation of a sequence of patterns that integrates and directs intentional behavior. Freeman's principles, along with the mechanisms of self-organization in dynamical systems, begin to show us exactly how such hierarchies of goal states may come to develop in actual biological brains.

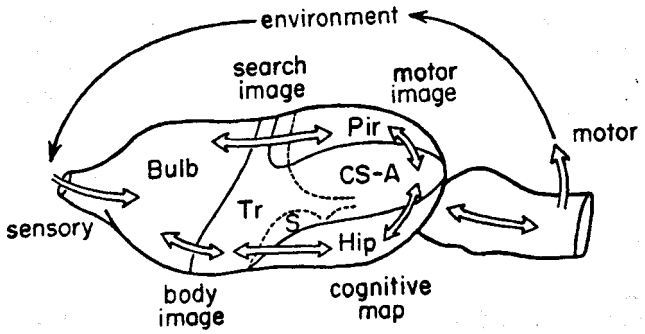

Figure 2.13: A schematic representation of the basic vertebrate limbic system. There are three major divisions, sensory areas, cortex, and motor areas, along with a hippocampus for cognitive maps and other types of long term memory (from Freeman, 2001)

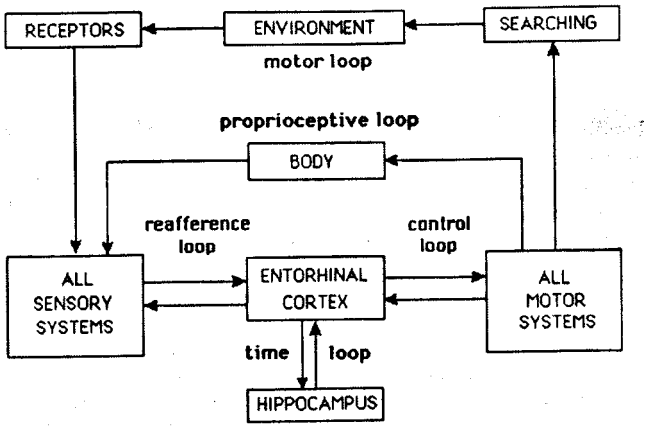

Figure 2.14: A diagram of the major interactions of the limbic system. This diagram shows the forward and backward feedback flows between the three major areas of the limbic system, along with their relationship to the body and external environment (from Freeman, 2001)

Brain scientists have known that the minimal nervous system that is capable of supporting the basics of intentional behavior is what is called the limbic system. Phylogenetically, the development of the basic limbic system first appears in amphibians, such as the salamandar. This system is comprised of the phylogenetically oldest parts of the forebrain, along with the paleocortex and the deeper lying motor nuceli, as well as some form of a primitive hippocampus. Figure 2.13 shows a schematic illustration of a prototypical vertebrate limbic system. Figure 2.14 is a more diagrammatic representation of the feedback relationships between the major areas of the 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.

Chapter 3

Preliminary Experiments

As stated previously, the ultimate goal of the proposed research is to demonstrate the use of oscillatory and chaotic dynamics for perceptual and behavioral control of an autonomous agent. In this section I will present some preliminary work and results leading up to this goal.

A fundamental component needed for this proposal are suitable basic computational units capable of the required oscillatory and chaotic dynamics. Freeman's K-sets (Freeman, 1975; Skarda & Freeman, 1987) provide such basic units. The K-set model 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 original K-sets were described by a second order differential equation (see section 2.6.2). 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.

The K-set provides a starting point for the basic units to be used in the proposed 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.

I have developed a discrete, deterministic version 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 basic 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 hebbian learning mechanisms for oscillatory units (Harter & Kozma, 2002b, 2002a, 2001a; Kozma, Harter, & Franklin, 2001).

3.1 KA Model Description

At its heart the KA model uses a 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 current density. The basic form of the difference equation can be stated simply as shown in equation 3.1, which states that the current at time step $t+1$ depends on the current at the present time t plus some change that is applied to the current.

$$c_{t+1} = c_t + \mu_t \tag{3.1}$$

The evolution equation of a KA unit can be described as three components that are combined to compute the simulated current at time $t+1$ from the current and the rate of change of the current at time t. These three influences on the simulated current are 1) a tendency to decay back to the baseline steady state; 2) a tendency to maintain the momentum of the current in a particular direction under excitation or depression; and 3) the influences of external excitation or inhibition as input to the unit. In addition a saturation effect is simulated such that as the unit moves towards its theoretical maximum or minimum, it becomes increasingly difficult to continue to push the current towards those extremes. I describe these influences in the next sections.

3.1.1 Baseline Decay

In the KA model the simulated current has a tendency to return back to its baseline steady state. The activity level is constrained to range from -1.0 to 1.0, with a current of 0 being the resting current. The effect of decay is described by equation 3.2.

$$\mu_t^d = -c_t \times \alpha \tag{3.2}$$

Here α is a parameter that indicates the rate of decay. Since the decay is proportional to the negative of the current, the effect is to cause the decay to be rapid for values of the current far from the baseline, while it slows down as the current approaches 0.

We can illustrate the effect of the decay term on the activity level by showing the result of decay for various values of α. In Figures 3.1 and 3.2 we use the equation

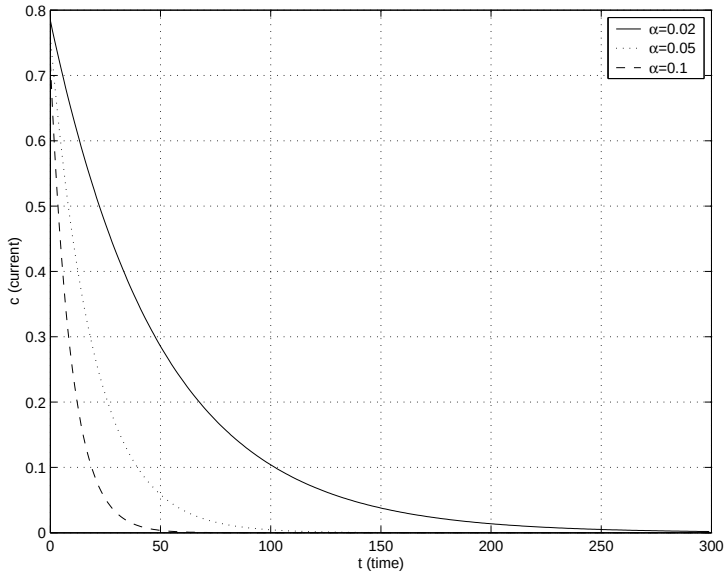

Figure 3.1: Decay to baseline steady state for various α values from a positive initial current.

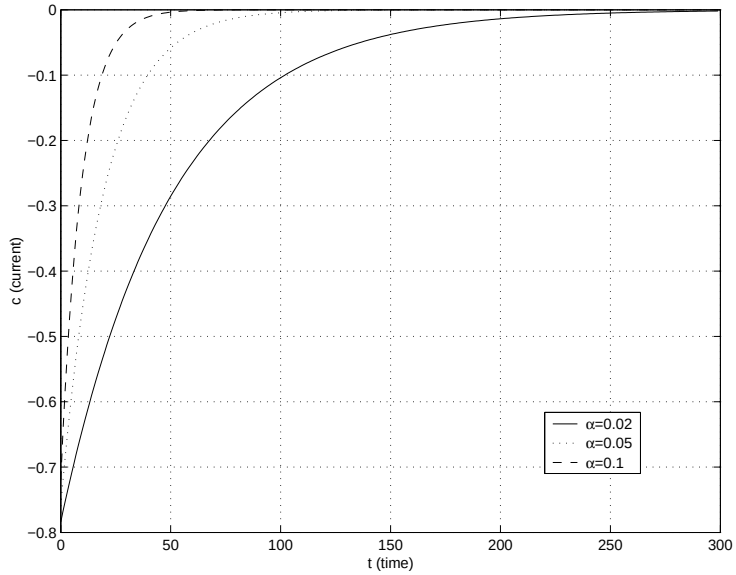

Figure 3.2: Decay to baseline steady state for various α values from a negative initial current.

$c_{t+1} = c_t + \mu_t^d$ to demonstrate only the effects of decay on the activity. Figure 3.1 displays how the current decays from a starting current of 0.8 for α values of 0.02, 0.05 and 0.1 respectively. Figure 3.2 is the same except we now use an initial value of -0.8 for the current.

The α parameter effectively controls the rate of decay back to the baseline state. In the difference equations, α can be thought of as the percentage that the current should decay at each time step.

3.1.2 Momentum

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 ODE equations of the K-set, this was reflected in the use of a 2nd order term. The KA model adds momentum by looking two steps back in time to determine the rate of change of activity in the unit.

We first define the rate of change of the current at time t (r_t). This is defined as the difference of the current at time t from the current at the previous time step $t-1$. The rate at time t is given in equation 3.3.

$$r_t = c_t - c_{t-1} \tag{3.3}$$

With the rate at time t defined, we can describe the momentum as shown in equation 3.4.

$$\mu_t^m = r_t \times \beta \tag{3.4}$$

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 again as a percentage which indicates what portion of the momentum at the current time step should continue into the next time step. In figure 3.3 we show the effects of 3 different β parameters on the momentum. In this figure, $c_{-1} = 0.5$ and $c_0 = 0.6$ (therefore $r_0 = 0.1$).

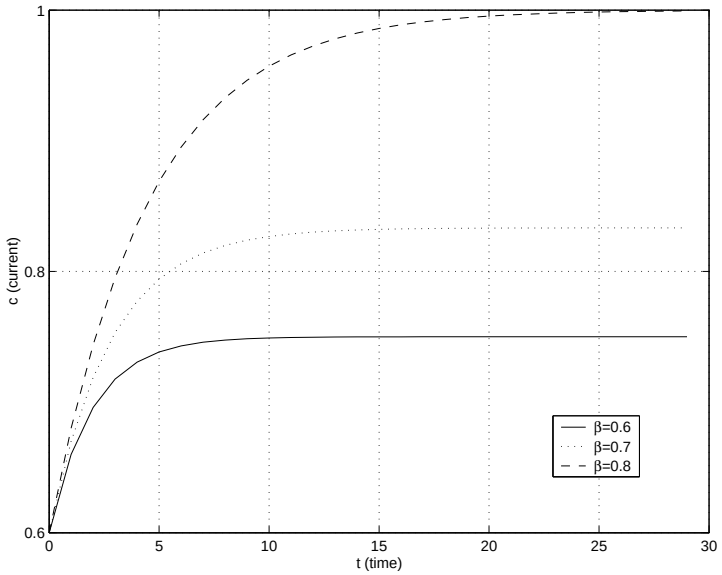

Figure 3.3: Momentum for various β values. $c_{-1} = 0.5$ and $c_0 = 0.6$.

3.1.3 External Stimulation

In the KA model, units may be connected together with other units to form networks. A KA unit is strictly either excitatory or inhibitory. Excitatory units cause the activity level to be increased in units they are connected to while inhibitory units have the opposite effect. The effect of excitation or inhibition is scaled by a weight on the link between the units.

Figure 3.4 represents a portion of a generic KA network. The simulated current for the unit in layer j is affected by the currents of the connected units in layer i. The effect of the external stimulation at time t is calculated by equation 3.5. In this equation f is a transfer function of the current (discussed in the next section). w_{ji} is the weight of the connection between the i^{th} and j^{th} units, and γ is a scaling factor. γ is used to tune the input for various network configurations. For example in a simulation with approximately 10 connections to each unit, γ might be set at 0.1. However, if the simulation has approximately 100 connections per unit, γ might only be 0.01. The γ parameter scales the input in order to keep the model within

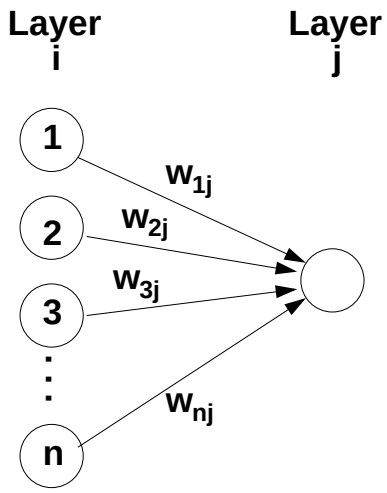

Figure 3.4: A generic KA Network. Units have connections to other units with weights associated with each connection

reasonable bounds and will vary depending on the connectivity of the simulation being tested.

$$\mu_t^e = \sum_{i=1}^{N} f(c_t^i) w_{ji} \gamma \tag{3.5}$$

Asymmetric Sigmoid Transfer Function

The output of a KA unit is calculated as a function of its current. We use the asymmetric sigmoid function, shown in equation 3.6, to calculate the output.

$$o_t = \epsilon\{1 - exp[\frac{-(e^{c_t} - 1)}{\epsilon}]\} \tag{3.6}$$

o_t is used as the output if the unit is excitatory, otherwise $-o_t$ is used.

$$o_t = \begin{cases} o_t & \text{if unit is excitatory} \\ -o_t & \text{if unit is inhibitory} \end{cases} \tag{3.7}$$

The ϵ parameter is a scaling factor that indicates the level of arousal of the KA unit. In the KA model, we take the result of equation 3.6 and scale them so that

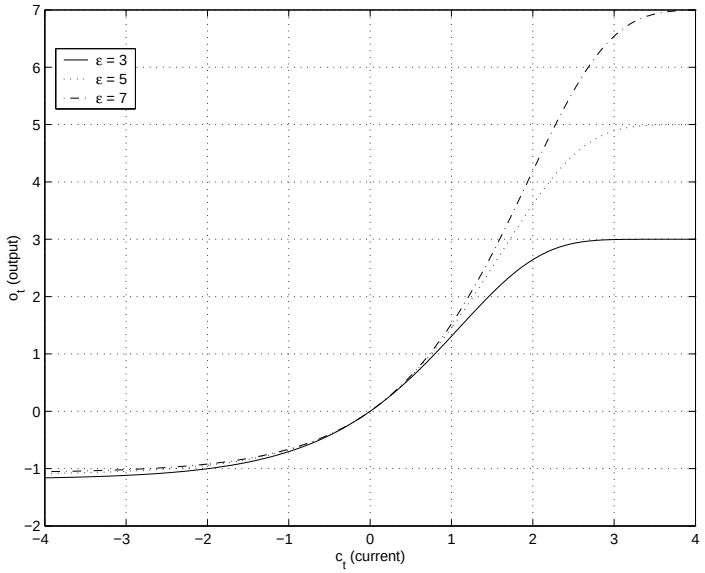

Figure 3.5: The asymmetric sigmoid transfer function of the KA model. We show the function for 3 values of the ϵ arousal parameter (3, 5 and 7).

the resulting output values range between 0 and 1. Figure 3.5 shows the transfer function, before this normalization is performed, for ϵ values of 3, 5 and 7.

3.1.4 Saturation

The sum of the components in equations 3.2, 3.4 and 3.5 represent the total difference that we are proposing to apply to the current as a result of the influence of decay, momentum and external stimulation respectively:

$$\mu'_t = \mu^d_t + \mu^m_t + \mu^e_t \tag{3.8}$$

Consider μ'_t as a provisional difference that is being proposed to be applied to the current. Before this difference is applied, we first check for the saturation of the unit. Saturation begins to occur when a unit goes above (or below) a saturation threshold. η is the saturation threshold parameter and λ is a parameter that determines the rate of saturation. Equation 3.9 displays how the saturation is applied.

$$\mu_t = \begin{cases} \mu'_t & \text{if } |c_t + \mu'_t| \leq \eta \\ \mu'_t(\frac{1-|c_t|}{1-\eta})^\lambda & \text{if } |c_t + \mu'_t| > \eta \end{cases} \tag{3.9}$$

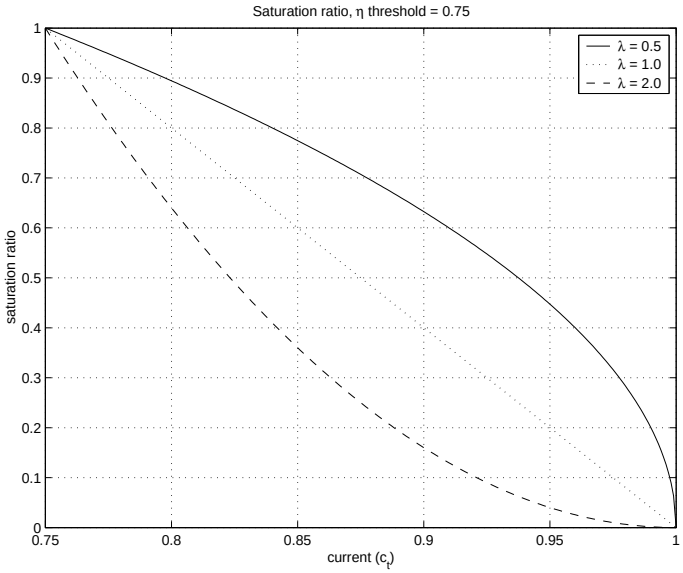

Figure 3.6: The saturation ratio displayed as a function of the current c_t for saturation threshold $\eta = 0.75$. We show the saturation for various values of λ.

In other words, if we are still below the threshold then we simply use μ'_t as the difference. If we are above the threshold, the μ'_t value gets scaled by the value $(\frac{1-|c_t|}{1-\eta})^\lambda$. Figure 3.6 displays how μ'_t is scaled for a η threshold of 0.75 and for various saturation parameters ($\lambda = 0.5$ $\lambda = 1.0$ and $\lambda = 2.0$). Notice that at the threshold (0.75) the ratio is 1.0, which indicates that we use the full value of Δ'_t. However as c_t increases towards the maximum, the ratio value steadily decreases. For $\lambda > 1$, saturation is quicker for values just above threshold, and slower as it approaches the maximum. The opposite holds for $\lambda < 1$ where initially the saturation ratio is slowly increasing, until it quickens as it approaches the maximum.

3.1.5 KA Single Unit Dynamics

I will now show the results of applying equations 3.1-3.9 to simulate a single population's dynamics. Table 3.1 is a recap of all of the variables used in the KA model.

Table 3.1: KA Model Variables

Variable	Description
c_t	Simulated current at time t
r_t	Rate of change of current at time t.
μ_t	Difference to be applied to current at time t
μ_t^d	Difference at time t due to decay to baseline
μ_t^m	Difference at time t due to momentum
μ_t^e	Difference at time t due to external input

Table 3.2: KA Model Parameters

Parameter	Description	Default
α	Rate of decay to baseline	0.035
β	Rate of momentum	0.9
γ	Input scaling parameter	0.025
ϵ	Transfer function arousal level	3.0
η	Saturation threshold	0.75
λ	Saturation scaling ratio	0.5

Table 3.2 lists the parameters of the KA model along with typical values that will be used for the parameters in all subsequent simulations.

Figure 3.7 is a figure taken from (Freeman & Shimoide, 1994) which shows the open loop impulse of an isolated neuronal population under deep anesthesia. This type of response is typical of an isolated population with a steady decay and slight overshoot before returning to the baseline state. In figure 3.8 we show a simulation of a single unit of the KA model. In this simulation we provided external excitatory stimulation for the first 5 time steps and then let the unit return to baseline. Comparing figures 3.7 and 3.8 it can be seen that the KA model provides a good fit to the simulation of an isolated populations dynamics. This figure also illustrates that the parameters shown in table 3.2 are appropriately chosen to use 1 to 1 time scales between the model and biological data. All simulations from here on use 1 simulated time step to

represent a millisecond of time.

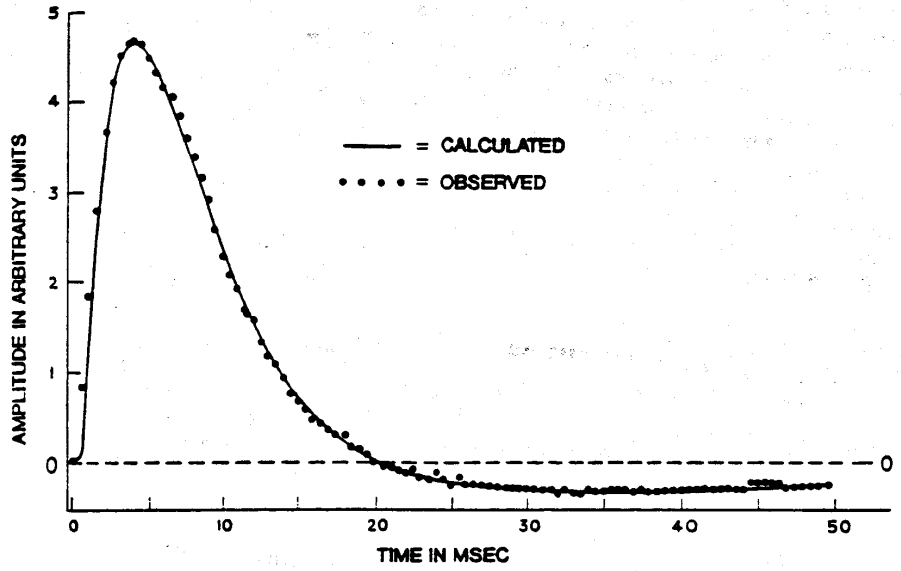

Figure 3.7: The dots show the open loop impulse response to external excitation of the pyriform cortex under deep anesthesia. The fitted curve is generated by a sum of exponential terms (from Freeman & Shimoide, 1994, p. 122)

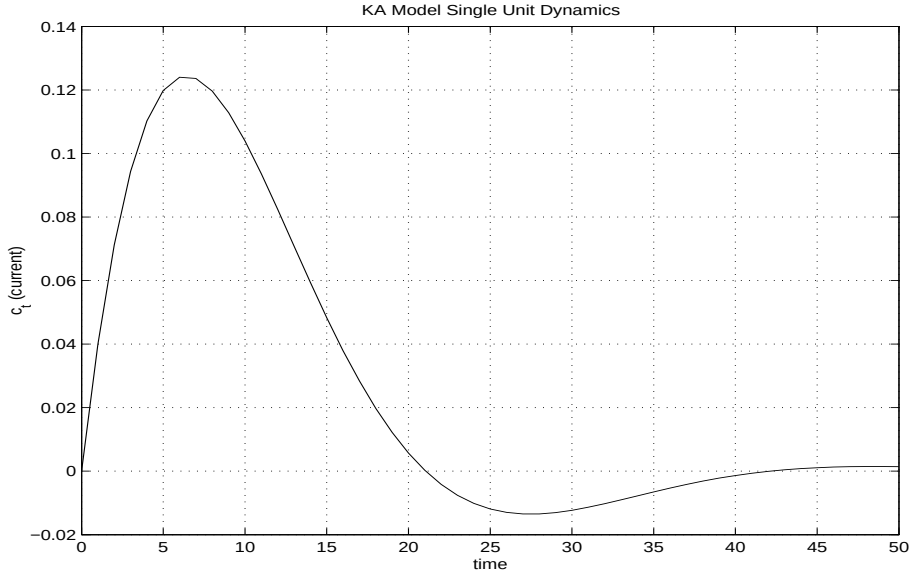

Figure 3.8: KA simulation of single isolated unit dynamics under external stimulation.

3.2 KA Demonstration of Principles

Freeman in (Freeman, 1999b, p. 37) postulates ten building blocks of neurodynamics that help to explain how neural populations create the chaotic dynamics of intentionality (shown in appendix A). The first three principles deal with the formation of non-zero steady-state and oscillatory dynamics through various types of feedback in neural populations through excitatory and inhibitory connections. The forth principle deals with the formation of chaotic dynamics. In this section I will use the KA model to demonstrate these first four principles of neurodynamics. These results have also been reported in (Harter & Kozma, 2002b).

3.2.1 Principle 1: Non-Zero Point Attractor

The first principle of chaotic neurodynamics (Appendix A) deals with the formation of a non-zero point attractor through excitatory feedback. A single isolated unit in the KA model has a tendency to return back to its baseline steady state. However, two or more excitatory units, when connected together, can maintain their activity indefinitely through mutual positive feedback. Figure 3.9a. is a diagram of the configuration for this experiment. In this simulation, two excitatory units are connected together. In figure 3.10 we show the results of the simulation for various values of the weight parameters between the two units. Notice that in all cases, the two units maintain their currents at a non-zero, positive point. Also, as the weight between the excitatory units is increased, the steady state point attractor also increases. Note that in this simulation, and in all simulations presented here, the behavior is produced through a completely deterministic processes with no noise or random components. Often the systems are started from random initial conditions, but the dynamics from there on are completely determined by the models evolution equations.

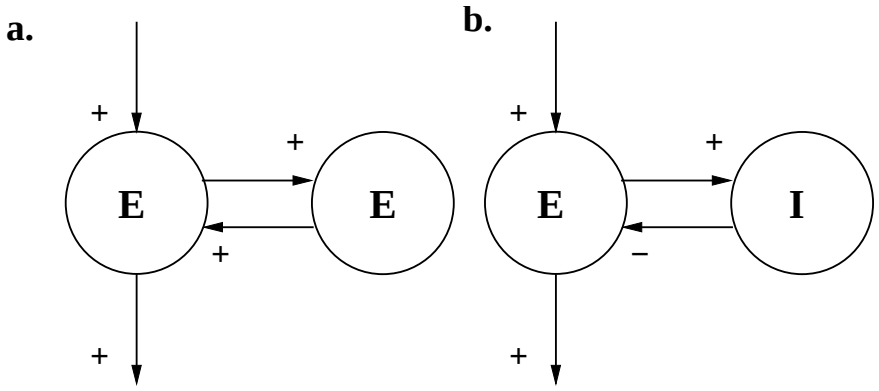

Figure 3.9: Configuration of the a) Excitatory-Excitatory; and b) Excitatory-Inhibitory simulations.

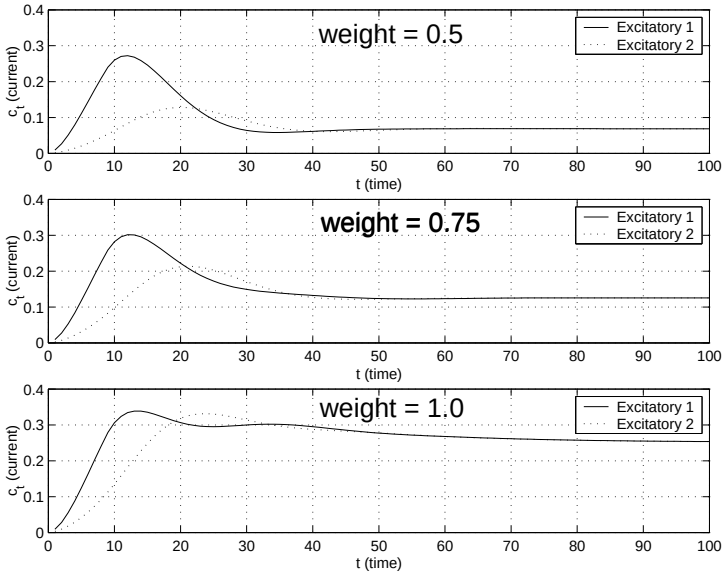

Figure 3.10: KA simulation of positive feedback for 3 different weight values.

3.2.2 Principle 2: Oscillation

The second principle of neurodynamics (Appendix A) states that oscillations emerge as a result of feedback among mixed excitatory-inhibitory populations. In the demonstration of principle 2 we use a similar setup as before. As shown in figure 3.9b. we now use feedback between an excitatory and inhibitory population.

Figure 3.11 shows a time series of an excitatory-inhibitory pair. In this time series we stimulate the excitatory unit for 10ms. As a result of this stimulation, the

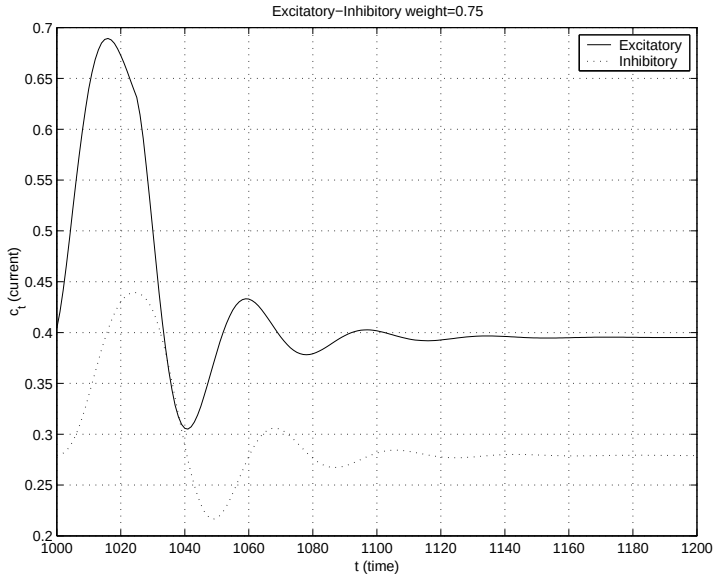

Figure 3.11: KA simulation excitatory-inhibitory, weight=0.75.

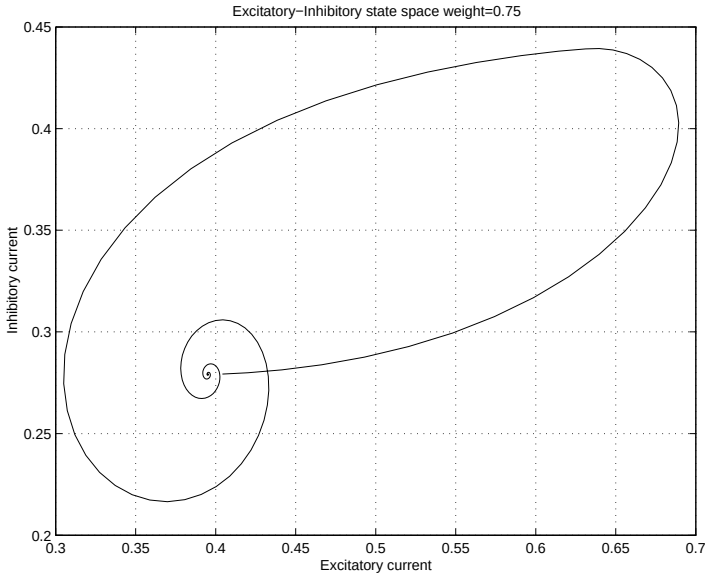

Figure 3.12: KA simulation excitatory-inhibitory state space, weight=0.75.

activity of the excitatory unit increases. The inhibitory unit also begins to increase but with a slight time lag. As the activity level of the inhibitory unit begins to increase, it in turn begins to inhibit the excitatory unit. This causes the activity of the excitatory unit to be pushed down eventually below its non-zero steady state. The loss of excitatory stimulation, in turn, causes the inhibitory unit to loose activation.

At the loss of inhibition, the excitatory unit is then able to rebound and increase its activity level. This feedback pattern continues for a couple of cycles. However, since the weight between the pair of units is below a critical threshold, the resulting oscillations eventually die down back to the original non-zero steady states.

Figure 3.12 is a state space plot of the activity of the excitatory-inhibitory units shown in 3.11. In a state space plot, we plot the activity level of the excitatory unit along the x axis vs. the activity level of the inhibitory unit along the y axis. This is done for all time periods from t=1000ms to t=1200ms. The state space representation of a dynamical system is simply a way of viewing the attractors of a system. In this case we can see that there is a steady state attractor where the excitatory current = 0.39 and the inhibitory current = 0.28. As a result of the initial 10ms stimulation, the system is pushed away from the point attractor, but quickly spirals back to this steady state.

3.2.3 Principle 3: Limit Cycles

The third principle of neurodynamics (Appendix A) states that limit cycles emerge as a result of sustained oscillations when the strength of the feedback between excitatory-inhibitory populations exceeds some threshold. If we increase the weight between the excitatory-inhibitory pair we can observe this effect in the KA units. Figures 3.13 and 3.14 show the time series and state spaces of such a pair still below this critical threshold. Compared to the previous simulation, the oscillations are more sustained and the system takes much longer to return back to the non-zero steady state point attractor.

In figures 3.15 and 3.16 we show an excitatory-inhibitory pair whose connection is above this critical threshold. We again stimulate the pair initially for 10ms, which perturbs the system away from its initial state. However, as can be shown clearly in the state space diagram, the system no longer returns to a point attractor, but is

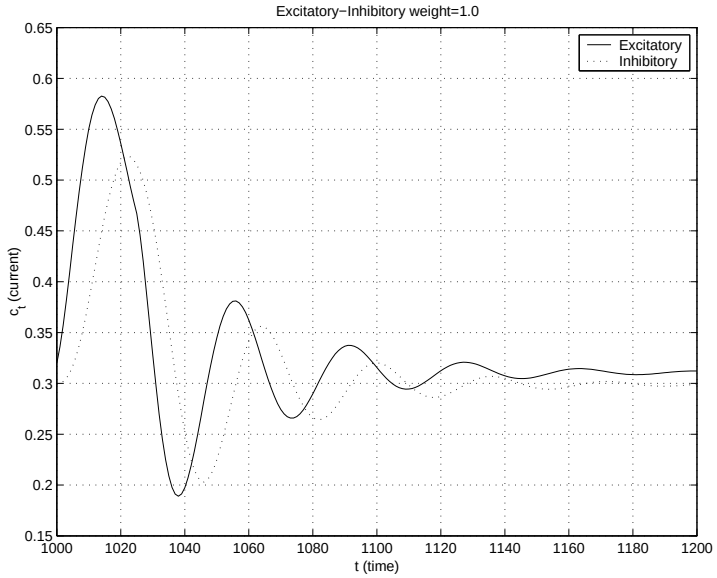

Figure 3.13: KA simulation excitatory-inhibitory, weight=1.0.

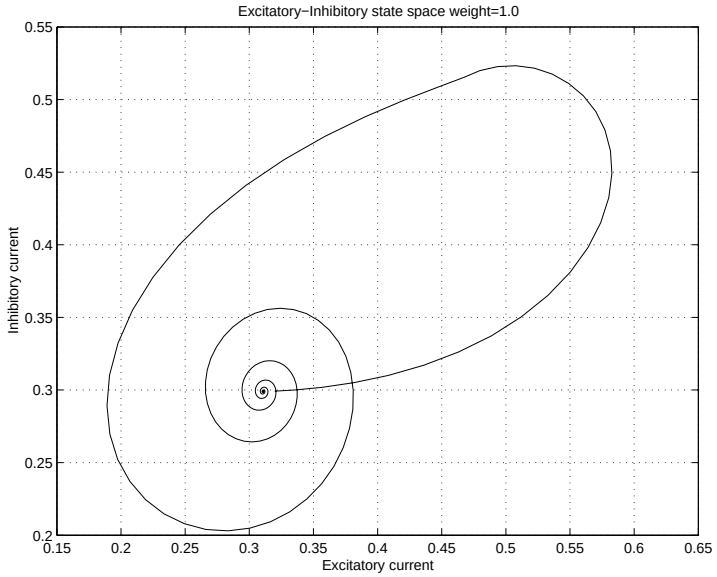

Figure 3.14: KA simulation excitatory-inhibitory state space, weight=1.0.

continually oscillating in a regular limit cycle.

3.2.4 Principle 4: Chaos

Freeman's fourth principle building block of neurodynamics (Appendix A) states that chaotic activity is generated as a result of feedback among three or more mixed

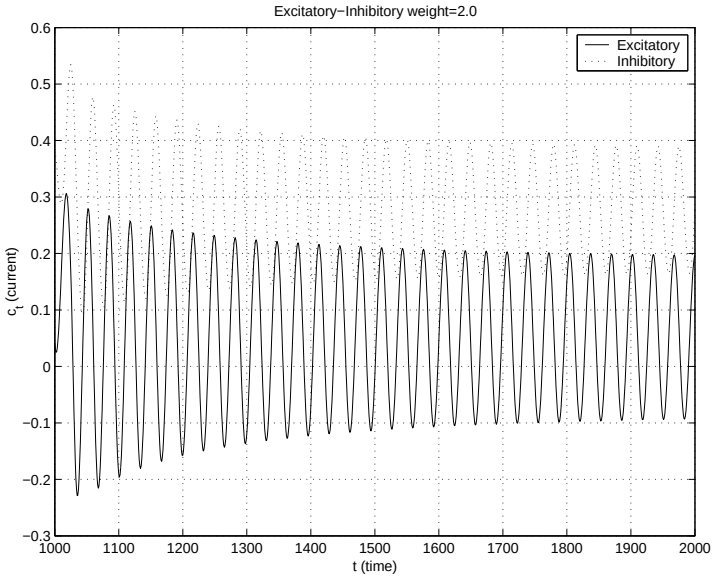

Figure 3.15: KA simulation excitatory-inhibitory, weight=2.0.

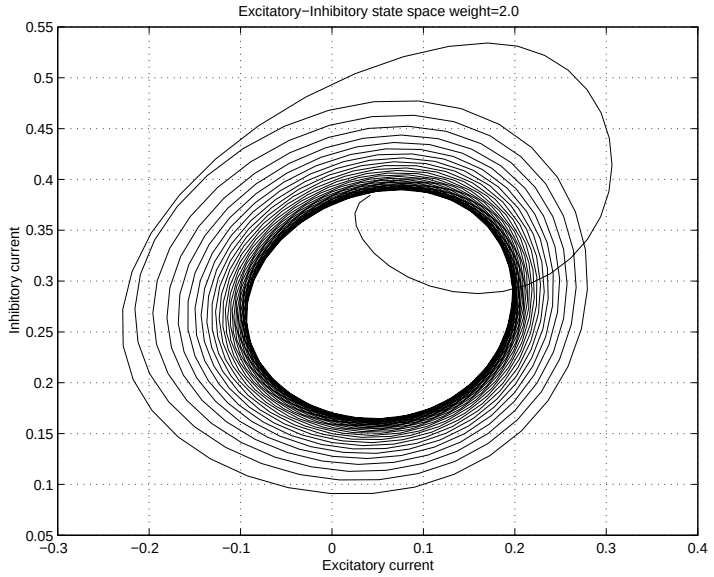

Figure 3.16: KA simulation excitatory-inhibitory state space, weight=2.0.

excitatory-inhibitory populations. As demonstrated in principles 1-4, non-zero steady states, oscillations and limit cycles can be generated through feedback among excitatory-excitatory and excitatory-inhibitory populations. Combining positive and negative feedback is necessary to produce oscillatory behavior. Before we describe the generation of chaotic dynamics, we will first describe a basic component that we form using

KA units that represents a standard mixed excitatory-inhibitory population.

KA-II: A Mixed Excitatory-Inhibitory Population

At a minimum, a mixed population must have two excitatory units in order to generate positive feedback, and thus sustain a non-zero steady state (principle 1). It must also have at least one inhibitory unit to produce oscillations and limit cycles (principles 2 and 3). For reasons of aesthetics, symmetry and history, we typically combine two excitatory units with two inhibitory units to form a basic mixed excitatory-inhibitory population. The configuration of such a population is shown in figure 3.17.

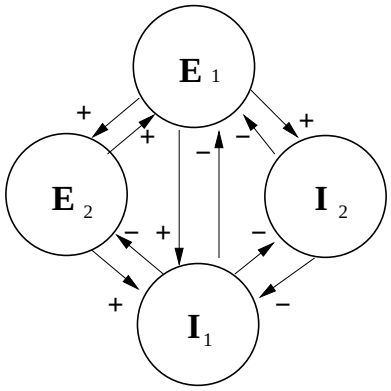

Figure 3.17: KA-II, a standard component used to emulate a mixed excitatory-inhibitory population.

The name given to such a population of two excitatory and two inhibitory units with ten connections between them is a KA-II set. The behavior of the four units in a KA-II set will be determined by the values of the ten weights between the units. As a matter of custom, we usually reduce the ten dimensions of the weights to four dimensions when describing a KA-II set. A KA-II set is usually described by the weights between the excitatory units w_{ee}, between the inhibitory units w_{ii}, from excitatory to inhibitory w_{ei}, and from inhibitory to excitatory w_{ie}. For example, if w_{ei} is set to 1.5, then the weights from E_2 to I_1, E_1 to I_1 and E_1 to I_2 are all set to 1.5.

Therefore a KA-II set is usually described by four parameters w_{ee}, w_{ei}, w_{ie} and w_{ii}. Different values for these four parameters result in different dynamics of the units. For example figure 3.18 shows a simulation of a KA-II when all four of the parameters are set to 1.0. In this simulation all four units are started at different random initial conditions. As a result of the mixed positive and negative feedback, all of the units oscillate (though none appear to form a limit cycle). Also all of the units eventually reach a non-zero steady state, where one of the units is slightly above the baseline and the other three end up in the negative region. This is a typical picture of the behavior of the units of a KA-II set. Depending on the values of the four weight parameters, the units may end up at different steady state attractors, or they may produce limit cycle behavior. I explore further the issue of the behavior of the KA-II set while varying the weight parameters in section 3.3.

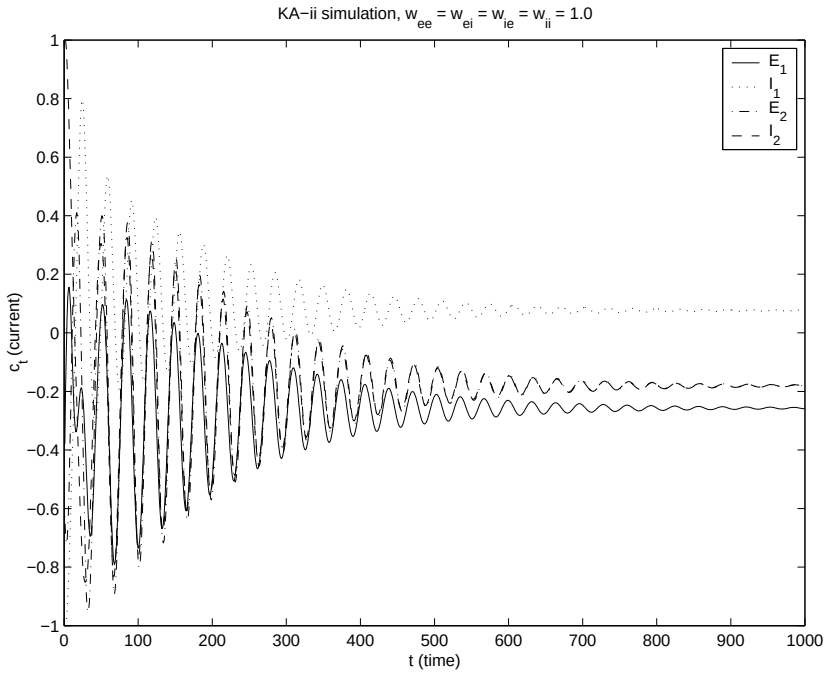

Figure 3.18: KA-II simulation, random initial conditions $w_{ee} = w_{ei} = w_{ie} = w_{ii} = 1.0$.

Chaotic Dynamics

The previous section was a digression to introduce the concept of the KA-II set, a basic mixed excitatory-inhibitory population. I now return to the fourth principle building block, the generation of chaotic dynamics. Chaotic dynamics, according to Freeman, are the result of feedback among three or more mixed excitatory-inhibitory populations. One easy way to produce chaotic looking behavior is to connect three KA-II sets together. Each of the 3 KA-II sets should behave differently, for example by preferring to oscillate at different characteristic frequencies (see section 3.3). The combination of three or more KA-II sets is named a KA-III set.

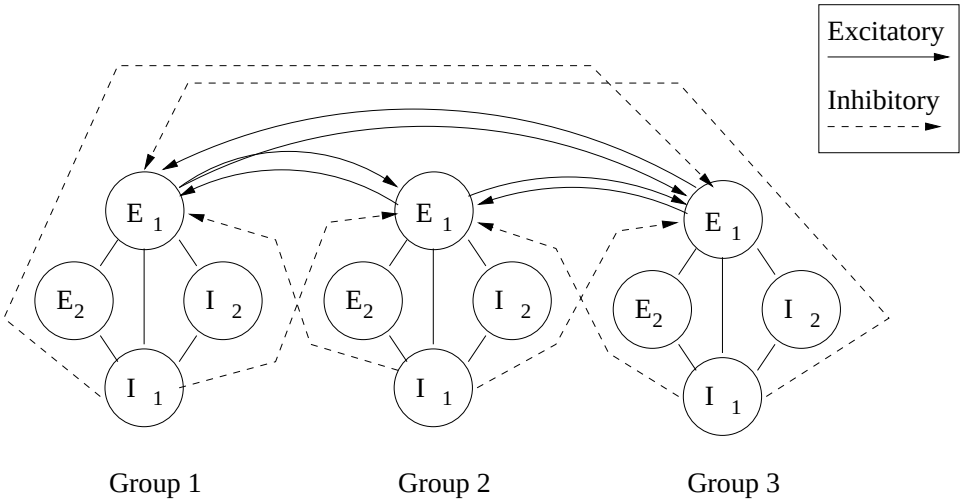

Figure 3.19: Diagram of simple KA-III set. A KA-III set is composed of 3 (or more) KA-II sets, connected with positive and negative feedback.

In figure 3.19 we show a simple example of a KA-III set. In this diagram, three KA-II are connected together by adding weights from the E_1 and I_1 units of each group to the E_1 unit of the other two groups. If we choose three KA-II groups with different characteristics (see section 3.3) and we set the 6 excitatory weights between groups to 0.6 and the 6 inhibitory weights to 0.1, then the deterministic KA model is capable of generating a chaotic time series. Figure 3.20 shows 5 seconds of the time series for the E_1 unit of each of the three groups. Figure 3.21 is a time delay plot of

the time series of the E_1 unit from group 1. Both the time series and the delay plots show activity indicative of chaotic behavior. A calculation of the lyapunov exponent[1] (Wolf, Swift, Swinny, & Vastano, 1985) gives a value of 0.35 for the time series of group 1's E_1 unit. This indicates strong chaotic behavior.

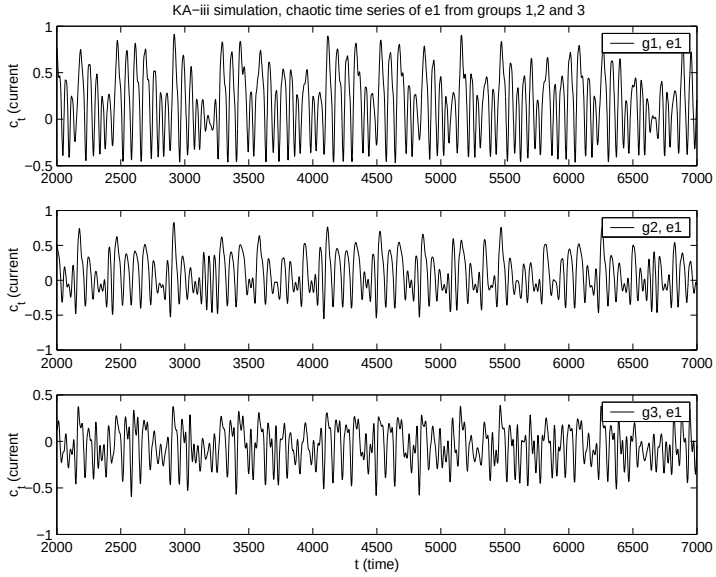

Figure 3.20: KA-III simulation showing the emergence of chaos among 3 mixed excitatory-inhibitory populations. We display the time series of the first excitatory unit of each of the KA-II.

The deterministic, discrete KA model is capable of generating chaotic behavior, in the manner described by Freeman's fourth principle building block of neurodynamics. Using different KA-II groups, and different connectivity patterns between the groups can result in other types of behavior (for example simple limit cycles). However, it is easy to find chaotic regions of the KA-III sets when both positive and negative feedback are used between the KA-II groups.

[1] The lyapunov exponent is measure that can indicate the level of periodicity or chaos in a time series. Negative exponents indicate point attractors; zero exponents are limit cycles, and positive exponents are indications of chaos.

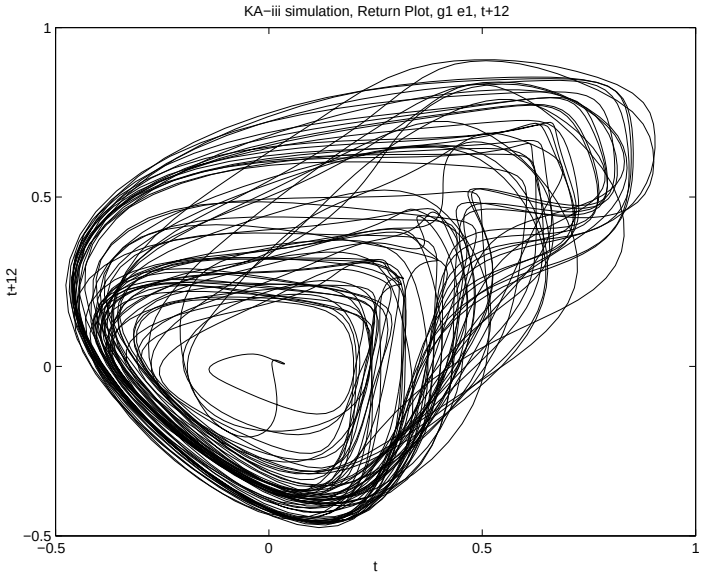

Figure 3.21: Delay plot (t vs. t+12) of group 1 E_1 unit in the KA-III simulation.

3.2.5 Principles 5-10

In total, Freeman postulated 10 basic building blocks of neurodynamics (Appendix A). Principles 5-10 are not as easy to demonstrate using simple examples. These principles have to do with how neural populations embody meaning on sensory input and select actions through chaotic dynamics, hebbian and other learning mechanisms, and important structural properties. I will visit issues of hebbian learning mechanisms in oscillatory and chaotic populations in section 3.6. A major goal, however, of the proposed research is the demonstration of some of the remaining principles in autonomous agents using biologically motivated architectures of KA units.

3.3 KA-II Set Properties

In this section I will be revisiting the KA-II set, and exploring some of the properties and ways of characterizing the behavior of them. As you may recall from section 3.2.4, a KA-II set is a particular configuration of four KA units, connected together with a total of ten connections (figure 3.17). Ten connections would imply a ten dimensional

space that needed to be explored to fully characterize the behavior of the KA-II set. However, as stated previously, we usually simplify this to a four dimensional space by keeping all weights between like typed units to be the same. This means that we ignore a large part of the behavior space of the KA-II, but what is left is still very large and probably serves to illustrate all possible interesting dynamics. This simplification also has biological justifications as mixed excitatory/inhibitory pairs in brains are often of the same connectivity pattern with equivalent weight settings between like types as we use in KA-II groups.

A KA-II set is characterized by the four parameters w_{ee}, w_{ei}, w_{ie} and w_{ii}. Changing these parameters causes changes in the behavior of the individual units within the KA-II. We can characterize the behavior of a KA-II set in various ways. The first and most simple distinction is: does a unit reach a steady state (point) attractor, or does it continue to oscillate (limit cycle). If a unit reaches a steady state, we might like to know what that point is, for example is it positive (above the baseline), negative (below the baseline) or close to the resting level. If a unit oscillates, an important piece of information to know is at what frequency it oscillates. A similar study for the original ODE K-II model was performed by Kozma and Freeman (1999).

3.3.1 Effects of Varying $w_{ee}, w_{ei}, w_{ie}, w_{ii}$ on KA-II

Steady State vs. Limit Cycle Regions

We will first explore the regions of the KA-II set that are oscillatory versus those that reach a fixed point. In this experiment we show the result of varying the four parameters on the E_1 unit of the KA-II group. To determine if the unit oscillates or becomes fixed, we run a simulation for 60,000 time steps (60 simulated seconds). We ignore the first 50,000 time steps and measure the standard deviation of the E_1 units time series from time 50,001 to 60,000. If the standard deviation is 0, this indicates that the unit has reached a fixed point. A positive deviation indicates that the unit

is probably oscillatory.

Figure 3.22 displays the effects of varying the parameters on the oscillatory behavior of the KA-II set. In this figure white areas are where the standard deviation of the time series was positive, which indicates oscillatory behavior; and black areas have 0 standard deviation and are therefore point attractors. In each small subgraph we vary the w_{ee} parameter from 0.0 to 2.0 in 0.1 increments along the x axis and perform a similar variation along the y axis for the w_{ii} parameter. The w_{ei} parameter is varied from 0.5, 1.0, 1.5 and 2.0 from left to right for each different plot, while the w_{ie} parameter varies in a similar way from bottom to top. In the space explored, the figure indicates that the KA-II set does seem to have a good mix of oscillatory and stationary regions.

Mean Activity

Another important characteristic of a KA-II set is the mean activity level that the units settle into. The natural level of activity of a group is an important property to consider when combining groups into KA-III and higher level components. Using the same simulation as described previously, we now show the results of measuring the mean of the current of the E_1 unit.

Figure 3.23 is similar to the previous figure. In this figure, however, we use color to represent the mean of the current observed from time 50,001 to 60,000. This figure does not differentiate between point attractor and limit cycle behavior. Therefore the mean current recorded for oscillatory units is the average, or midpoint, level of activity of the time series. The mean activity level, as depicted in the figure, does show regular variation in response to varying the weights. That is to say, there are no jumbled up regions. The mean activation varies smoothly as you move in any straight line within the parameter space. In general, it does appear that the w_{ee} parameter has a strong effect on the activity level, where higher w_{ee} usually produce

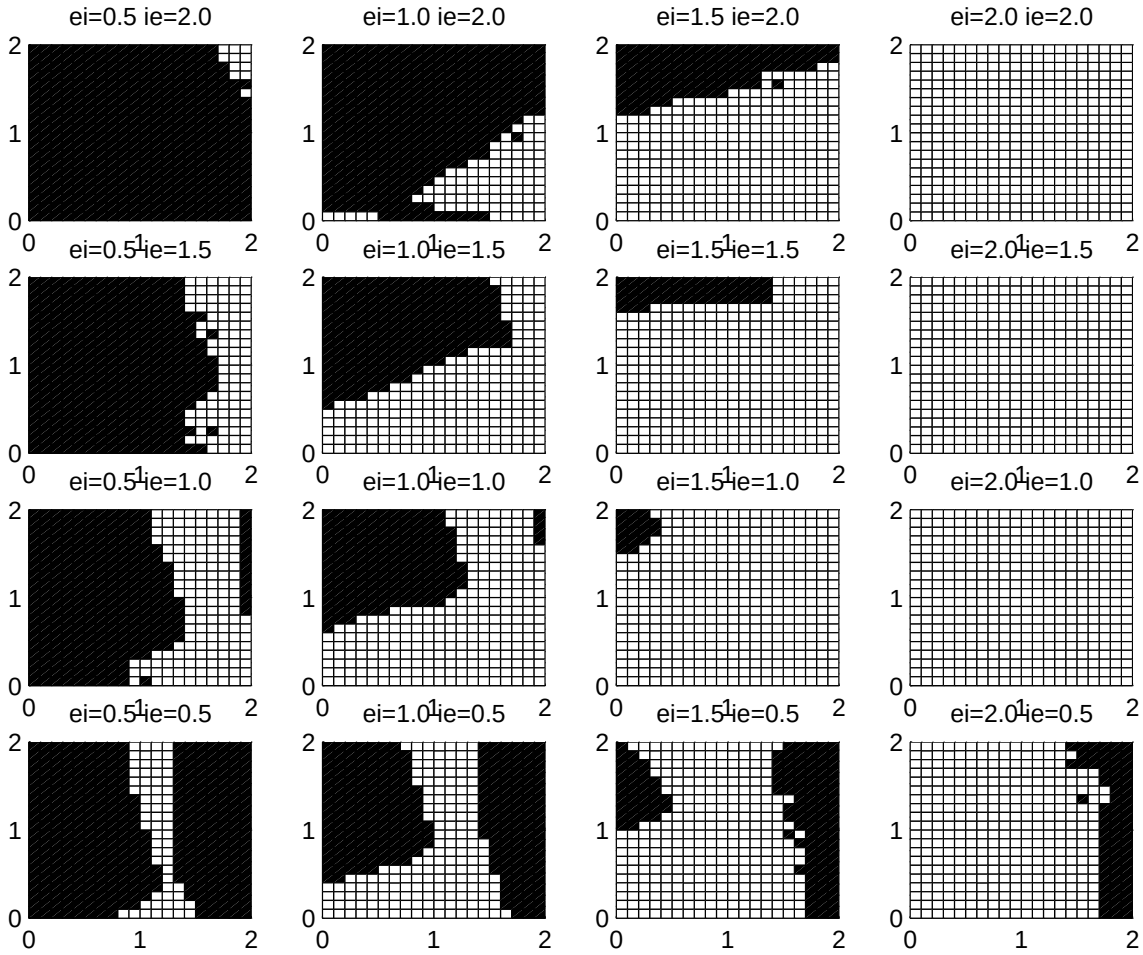

Figure 3.22: Effects of parameter variation on KA-II oscillatory regions. White areas are where the E_1 unit of the KA-II set exhibits a limit cycle attractor. Black regions are fixed points. See text for full description.

higher mean activations. The w_{ii} parameter seems much less of a factor in affecting behavior. Highest mean levels of activity seem to be produced when w_{ei} and w_{ie} are at lower values, the higher these weights are, the more depressed the activity of the E_1 unit.

A simplified view of the parameter space may be obtained by simply considering in what regions the mean is positive (above 0), where it is negative (below 0) and where it is close to the baseline (at 0). Figure 3.24 displays this view of the parameter space of the KA-II. In this figure we have categorized all mean activations above 0.05 as

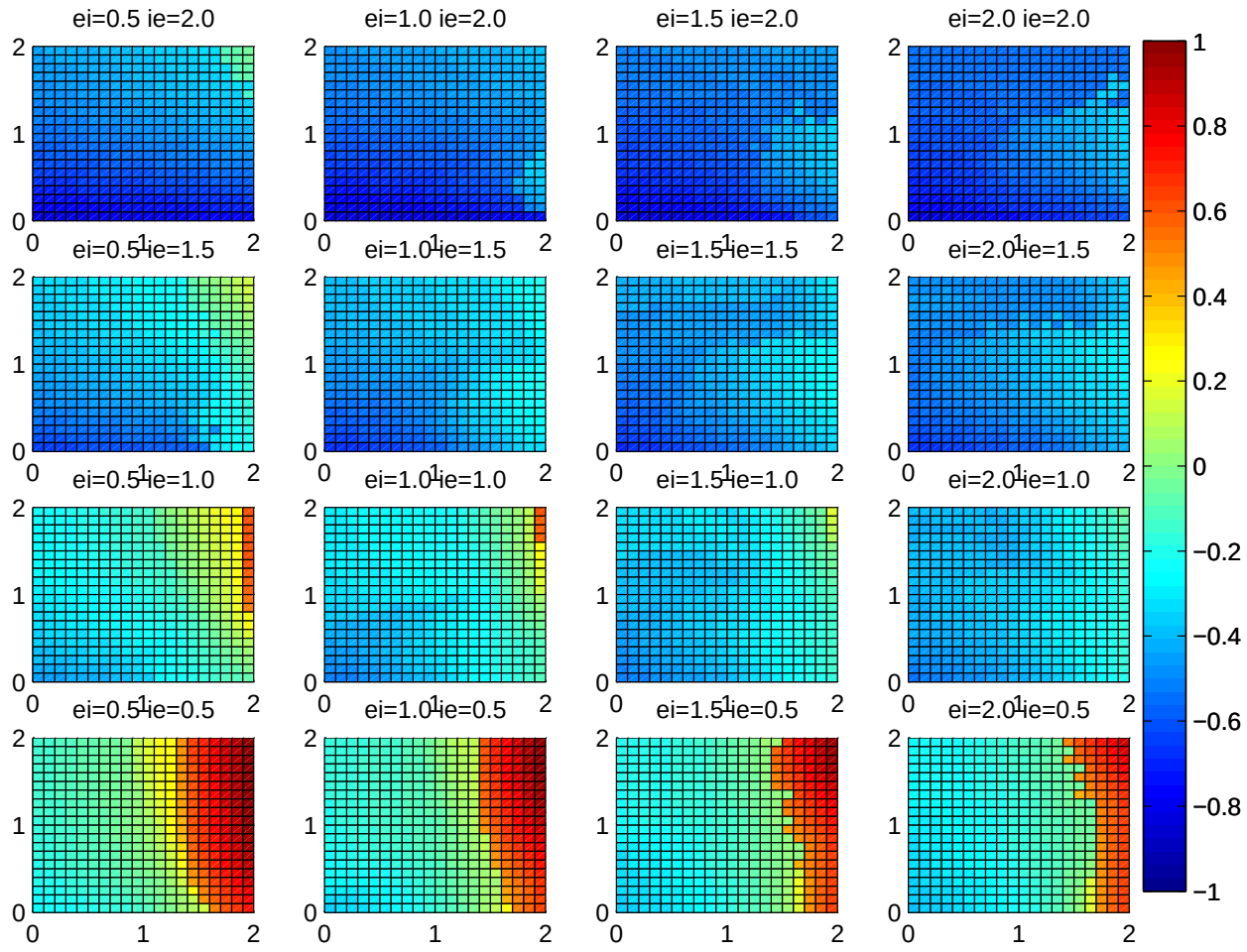

Figure 3.23: Effects of parameter variation on mean activity of KA-II set. Color indicates the mean current of the E_1 unit. See text for full description.

positive, and indicated those points with a red color. Areas where the mean activation were below -0.05 are considered negative and colored blue. And areas between these are colored green. This figure does make it a little clearer, on the subgraphs, that it is movement along the w_{ee} axis that most affects transition from the negative to the positive regions. Also it appears that low values of w_{ie} are most likely to have positive mean activations.

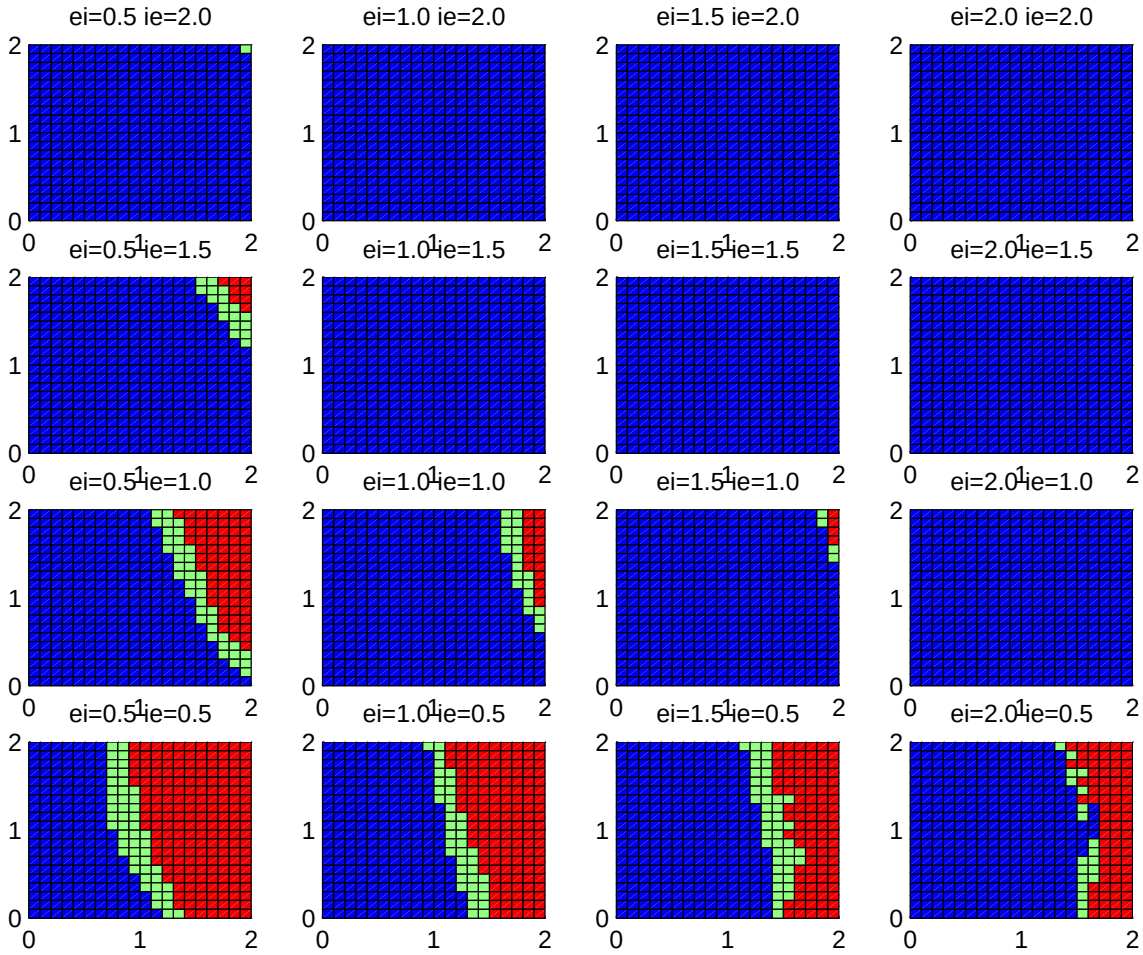

Figure 3.24: Effects of parameter variation on positive/negative regions for KA-II sets. Red areas are positive, green areas are around 0 and blue areas are negative. See text for full description.

Oscillation Frequencies

The final characteristic we will consider of the KA-II behavior is the frequency of the oscillations produced when the E_1 unit settles into a limit cycle. This characteristic frequency of the KA-II group in isolation is an important property. When combining KA-II to form a KA-III we often want to use groups with incompatible frequencies to insure the production of chaotic behavior.

In figure 3.25 we show the results of calculating the frequency of the oscillations for the time series that oscillate from time 50,001 to 60,000. We use color to represent

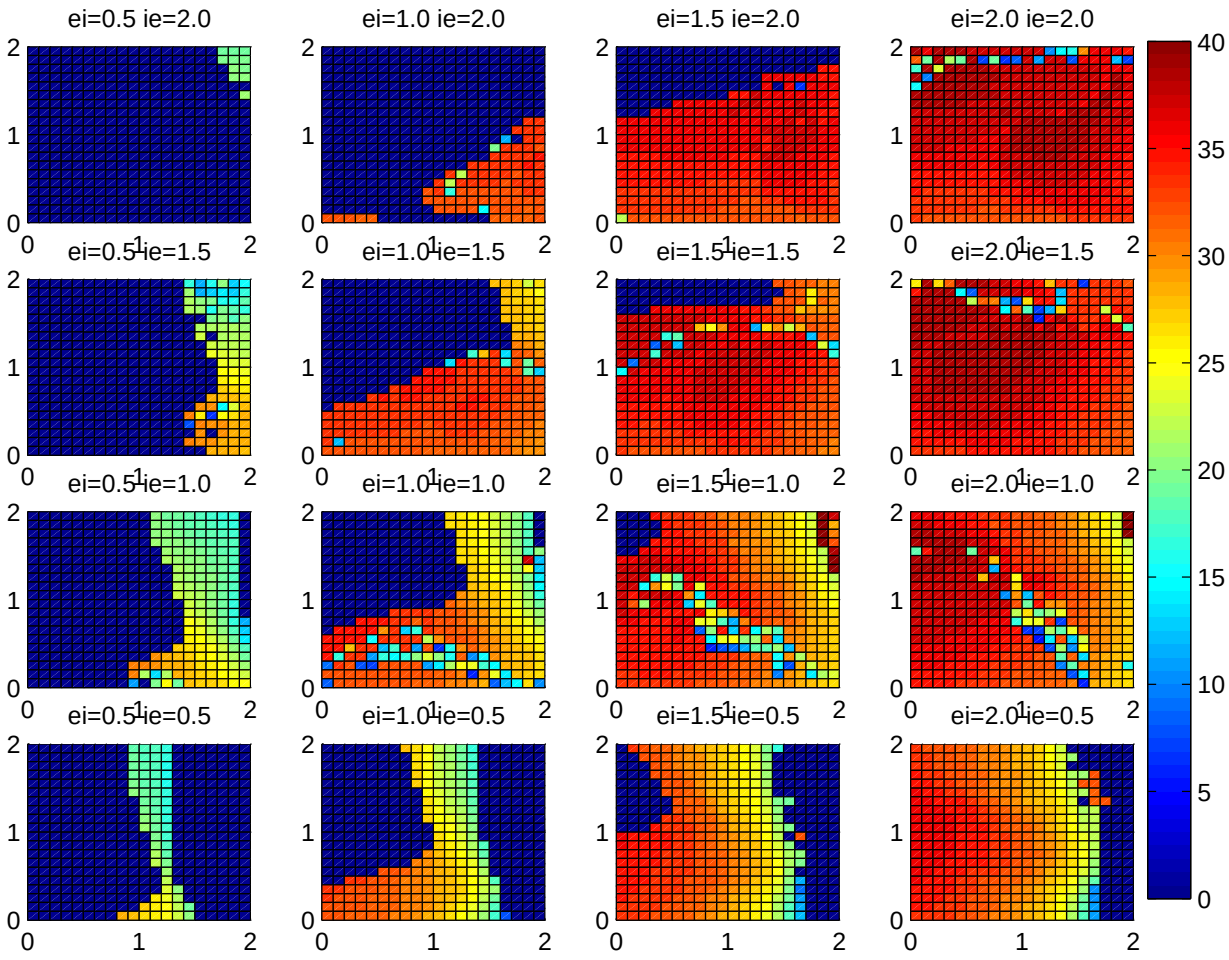

Figure 3.25: Effects of parameter variation on KA-II characteristic frequency. Color indicates frequency in cycles per second of the E_1 unit. See text for full description.

the frequency of the time series, in Hertz. We have used a similar setup in this figure to all previous figures in this section. Comparing this figure to figure 3.22 of oscillator vs. point attractor regions, you will notice that all point attractor regions are shown as having 0 frequency. The oscillations of the E_1 unit range from a low of 20 Hz to a maximum of around 40 Hz.

3.4 Stimulation of KA-II

In the previous section, I explored the behavior of the KA-II groups in isolation. That is to say that the KA-II was started off with a set of random initial conditions and then

Table 3.3: Typical KA-II used for Stimulation Experiments

w_{ee}	w_{ei}	w_{ie}	w_{ii}	mean	std	freq
1.05	1.40	0.44	0.05	-0.1174	0.2883	28 Hz

the simulation was allowed to run with no external stimulation or perturbations of any kind. An understanding of how the KA-II behave in isolation is necessary to making good choices for using them in combined groups of larger architectures. However, the behavior of the KA-II group in isolation will only give us a partial picture of how they will behave when they receive external perturbations. In this section I will explore what happens to a typical oscillating KA-II group when receiving external perturbations (both excitatory and inhibitory). The parameters and characteristics of the KA-II I will be using in this section are given in table 3.3. The mean, standard deviation and frequency shown are for the KA-II in isolation.

3.4.1 Constant Stimulation

The units of a KA-II group respond in various ways to external perturbations. These external inputs may be excitatory or inhibitory, and they may be constant, oscillating or noisy or chaotic.

In figure 3.26 (top) we show the results on the mean activity of the E_1 unit under constant stimulation. The bottom part of the figure shows the effect of the stimulation on the standard deviation of the E_1 unit. In these figure we ran the simulation of the KA-II for 11,000 time steps. We measured the mean and standard deviations after throwing away the first 1000 time steps to allow the system to pass through initial transients. At each time step we applied a constant external input to the E_1 unit. We tried all perturbations from -1.0 (inhibition) to 2.0 (excitation) in 0.01 increment.

Figure 3.26 of the change in mean activation shows a fairly smooth response to

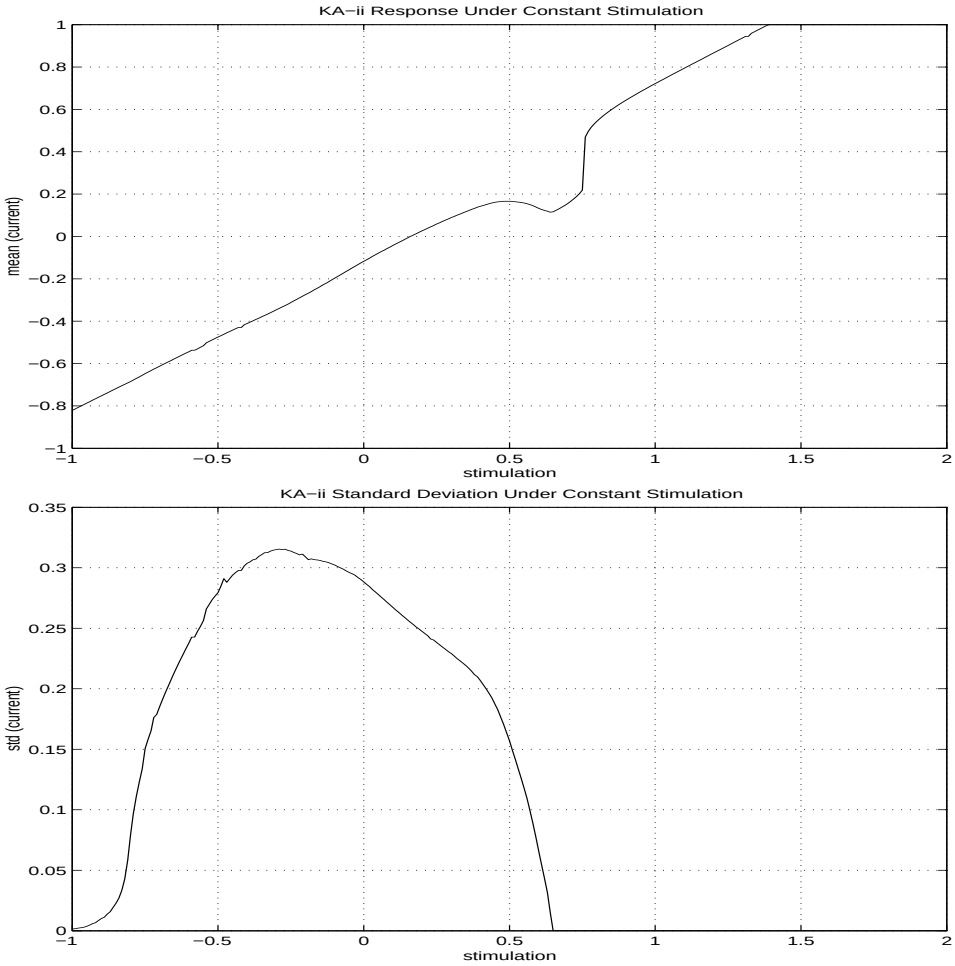

Figure 3.26: Effects of constant stimulation on the mean and standard deviation of a KA-II group. We apply a constant stimulation at every time step of the stimulation. We show the effects on the mean and standard deviation for inhibition of -1.0 up to excitation of 2.0 in 0.01 increments. The top figure shows the results on the mean, while the bottom figure show results on the standard deviation.

external perturbation. As excitation rises the mean activity also increases, until we get above a 0.5 excitatory input. At this point more excitation actually causes a decrease in the mean current, for a limited range of stimulation (0.5 to 0.75). In figure 3.26 (bottom) we show the effects of stimulation on the standard deviation of the current. Measuring the standard deviation, as you recall, is a good indication of when the signal is oscillating. In this case, the E_1 unit oscillates over a wide range, however constant stimulation above 0.6 or so causes the unit to saturate. Figures like

these can be used to find regions where the KA-II group behaves optimally for given applications.

3.4.2 Sinusoidal Stimulation

External stimulation to a KA-II group need not always be constant. In fact, since common architectures are made by interconnecting collections of KA-II groups, constant input may be more of the exception, and we would expect to see oscillating and chaotic inputs. In this section we apply a sinusoidal stimulation to the KA-II E_1 unit. As before, we vary the mean of this input signal from -1.0 to 2.0 in 0.01 increments. Figure 3.27 shows the results of applying this oscillating input, for various standard deviations of the input, on the mean and standard deviation. We used a sine wave, oscillating at 30 Hz, for the input for this simulation.

When the standard deviation of the input signal is 0, we have a constant input signal as shown in the previous section. The effect of sinusoidal input on the KA-II mean seems to be to smooth out nonlinearities in the response of the group. As larger oscillations are input, the mean response continues to flatten out. The effect of oscillating input on the standard deviation of the group is even more pronounced. Sinusoidal input greatly increases the range in which the unit is able to oscillate and avoid saturation.

In the previous two figures, we observed the results of inserting a sinusoidal input of 30 Hz of various standard deviations. Another interesting possibility is to observe the behavior of the KA-II group when oscillating inputs of different frequencies are inserted. In figure 3.28 we show the results of just such a simulation. As before, we vary the mean of the sinusoidal stimulation from -1.0 to 2.0 in 0.01 increments. However, in these figures, we hold the standard deviation at 0.15 while trying different frequencies of input, from 25 to 40 Hz.

The most interesting effects of varying the frequency of the input can be seen

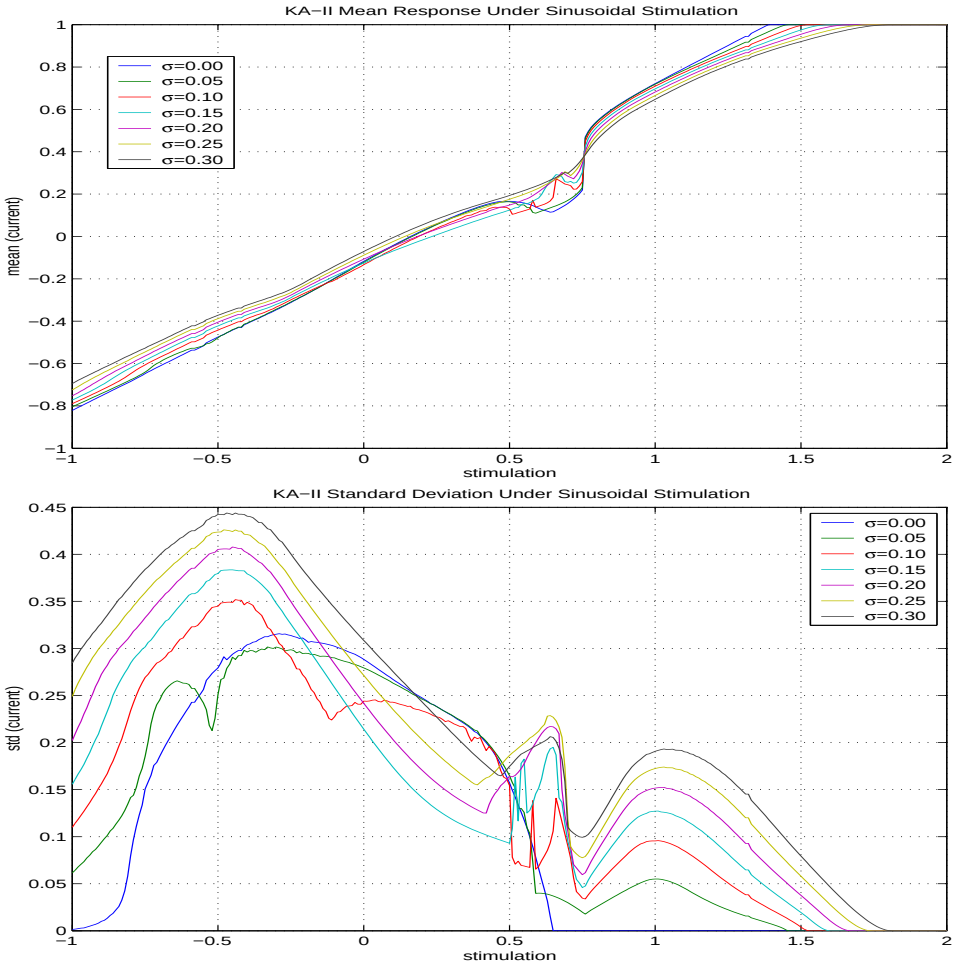

Figure 3.27: Effects of sinusoidal stimulation on the mean and standard deviation of a KA-II group. We plot the effect for different standard deviations (σ) of the input signal. Top figure shows results on the mean activity, while bottom figure plots effects on the standard deviation. See text for full details.

in the response of the standard deviation of the group. Especially noticeable are large peaks in response to 25 and 30 Hz signals. We conjecture that, these being the closest to the KA-II groups natural oscillation frequency, the units are able to become synchronized to the input signal, which reinforces and amplifies the oscillations.

3.4.3 Noisy Stimulation

As a final experiment we will explore the effects of applying noisy input to the KA-II group. In large, non-homogeneous populations, this type of input may be found when

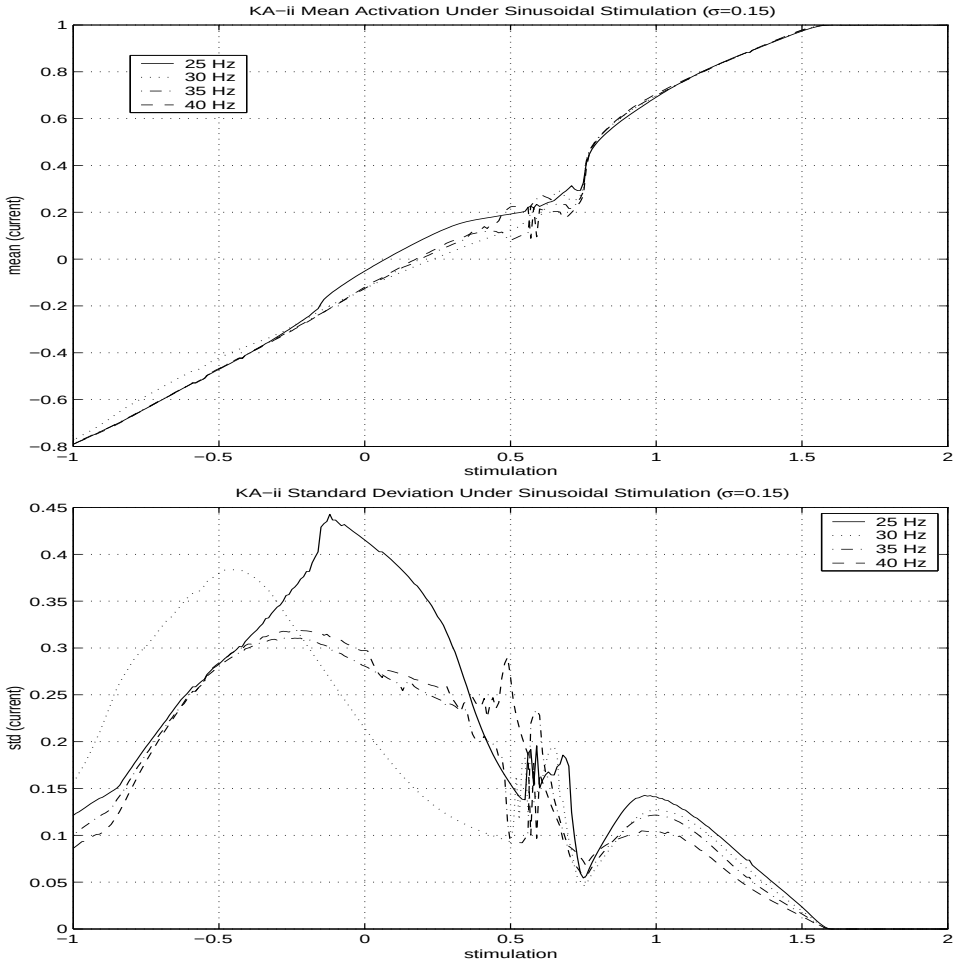

Figure 3.28: Effects of sinusoidal stimulation of varying frequencies on the mean and standard deviation of a KA-II group. We plot the effect for different frequencies of the input signal ranging from 25 to 40 Hz. Top figure shows results on the mean activity, and bottom shows effects on the standard deviation. See text for full details.

the KA-II groups form a KA-III set, and begin to exhibit chaotic behavior. We use a noisy signal here, as a simplistic approximation to chaotic input, to observe the effects of such stimulation on the KA-II group. In particular we are interested to see if the behavior appears more similar to that of constant or oscillatory stimulation.

In this simulation, we inject a nosy signal with a mean ranging from -1.0 to 2.0, as before. We explore the effects of the noise having a deviation from $\sigma = 0$ (constant stimulation), to $\sigma = 0.3$. The difference from the previous section is that the input is

not a regular sine wave, but is instead noise that deviates from the mean in a random but standard manner.

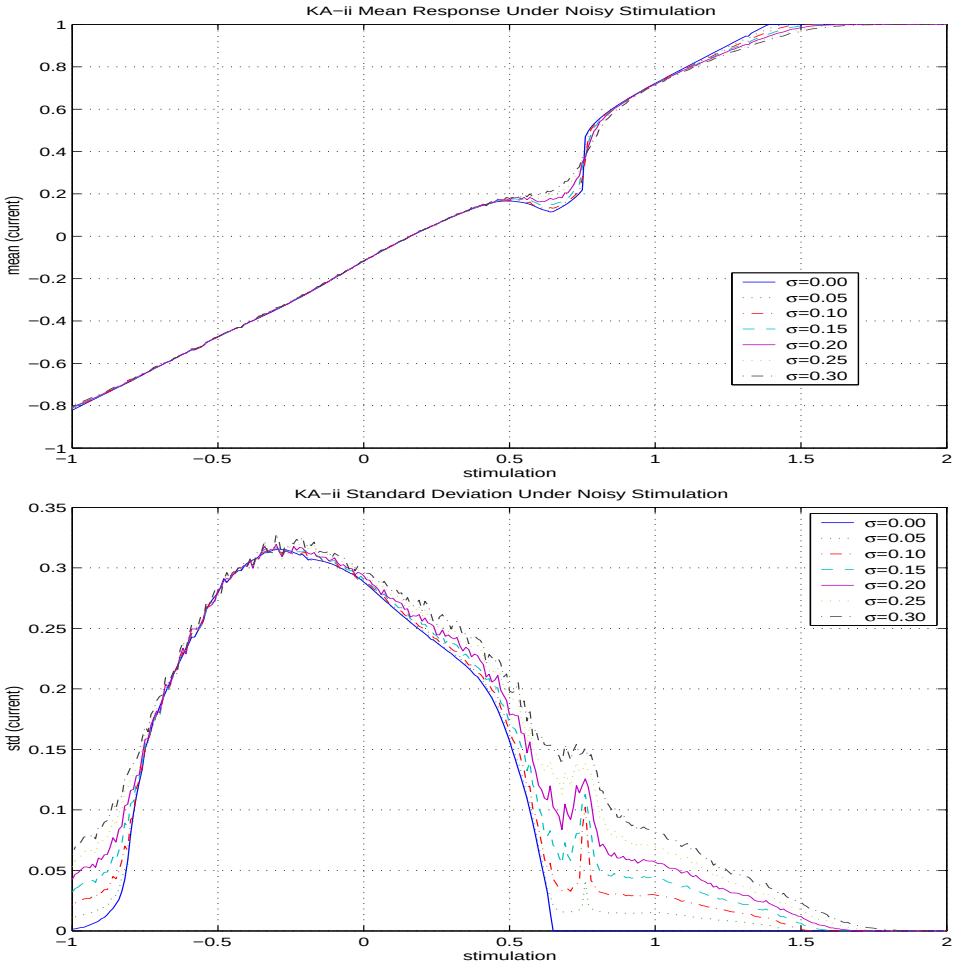

Figure 3.29: Effects of noisy stimulation on the mean and standard deviation of a KA-II group. We plot the effect for different σ values of the noise, ranging from 0.0 to 0.3 in 0.05 increments. Top figure shows effects on the mean, while effects on the standard deviation are shown in the bottom. See text for full details.

Figure 3.29 shows the results of injecting noisy input into the KA-II group. As can be seen, both the mean and standard deviation response show little difference from that of constant stimulation. One slight effect that can be seen in figure 3.29 (bottom) of the effect on the standard deviation is that the noisy input does allow the unit to keep from completely saturating under higher levels of stimulation.

Table 3.4: KA-II used for simulations in generating chaotic behavior

Group #	w_{ee}	w_{ei}	w_{ie}	w_{ii}	mean	std	freq
1)	1.29	1.27	0.65	1.19	-0.0714	0.2446	25 Hz
2)	1.05	1.40	0.44	0.05	-0.1174	0.2883	28 Hz
3)	0.95	1.41	0.80	1.33	-0.2443	0.1359	32 Hz

3.5 KA-III and Chaotic Dynamics

In this section I will explore further the generation of chaotic behavior by the KA model. Recall from section 3.2.4 that chaotic dynamics can result from the combination of three or more mixed excitatory-inhibitory populations. A KA-II set is our name for such a population, and combining three of these, as shown for example in figure 3.19, results in a KA-III set. Such a configuration is capable of generating chaotic behavior, but not necessarily so. The resulting behavior depends on many factors, such as the inherent dynamics of the KA-II groups selected and the connectivity patterns and weights between the groups.

Figures 3.20 and 3.21 were generated using the three KA-II groups shown in table 3.4. In this table, we show some of the characteristics of the KA-II groups in isolation. We used the connectivity pattern shown in figure 3.19, where all E_1 units were interconnected among the groups and all I_1 units had an inhibitory connection to the E_1 units of the other two groups. We used a weight of 0.6 for each of the excitatory connections between groups, and of 0.1 for each of the inhibitory connections. As stated previously, the measured lyapunov exponent of the time series shown in the figure was around 0.35 indicating strong chaotic behavior.

3.5.1 Effect of Weight Variation on Measured Lyapunov Exponent

Although, as stated previously, the generation of chaotic behavior depends on many factors, we will concentrate in this section on the effects of modifying the excitatory weights between groups on the dynamics. In this simulation we set all six of the inter-group excitatory weights to be the same value. We vary these weights from 0.0 to 0.65 in 0.01 increments. At each weight setting we simulate a time series for 30 seconds (throwing away a number of initial transients). I ran ten simulations at each weight setting, with uniformly distributed random initial conditions. I then calculated the lyapunov exponent of the time series for the given weights and plot the results.

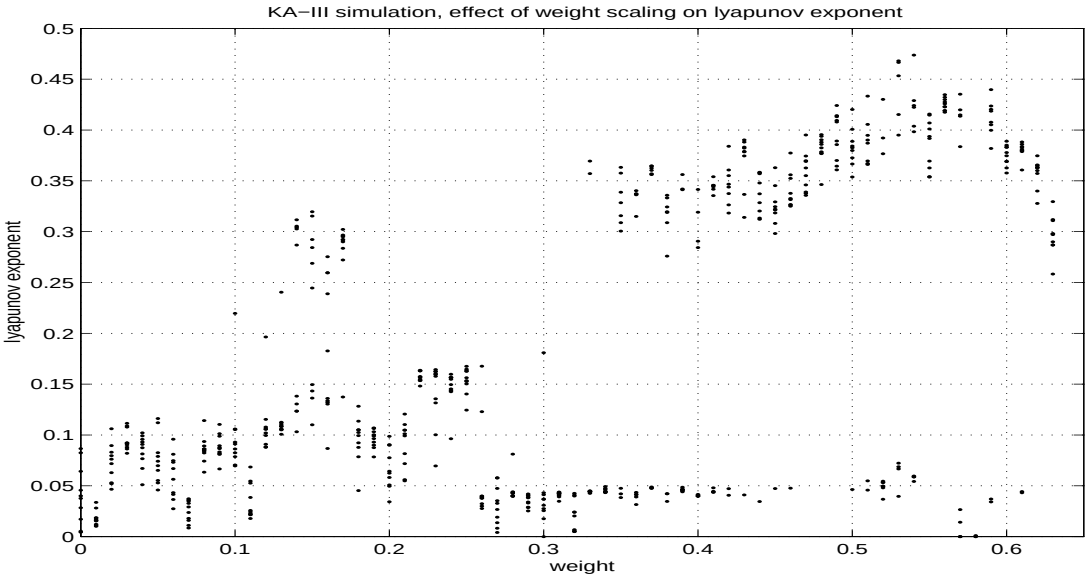

Figure 3.30: Effects of excitatory weight scaling on lyapunov exponent of KA-III simulation. The excitatory weights are scaled from 0.0 to 0.65 in 0.01 increments. Ten plots are shown at each weight, representing runs with ten different initial conditions.

Figure 3.30 shows the results of this simulation. As can be seen in this figure, as the excitatory weights increase we see a general increase in the estimated lyapunov exponent. With weights around 0.3 we observe very periodic appearing behavior. Larger weights begin to appear more chaotic. At almost all weight values, at least

some of the ten simulations ended up being more periodic in nature (the bottom trend in the figure from 0.28 to 0.6). This shows that the final behavior of the group does depend on the initial conditions.

3.6 Hebbian Mechanisms for Learning with Oscillatory (KA) Units

In this section we discuss hebbian mechanisms for use specifically with the KA model units, but more generally for use among oscillating and chaotic units. Freeman's seventh principle building block (appendix A) states that meaning is embodied in neural tissue through synaptic changes due to learning. One important learning mechanism (though not the only one) is hebbian synaptic modification, which performs the major role in learning in KA architectures.

3.6.1 Hebbian Learning

Donald Hebb, in (Hebb, 1949), first provided the insight behind the mechanism that has come to be known as hebbian learning. The basic idea is that coactivation of connected cells should cause a change (in the synapse) that will, in the future, make it more likely that the postsynaptic cell will fire when the presynaptic cell fires. This makes good sense as it allows for associated world events to become encoded in cell firings and synaptic patterns. Hebb did not identify the exact mechanism of his learning when he had this insight, but subsequent research has revealed many candidate mechanisms that fulfill his general framework (Churchland & Sejnowski, 1992).

The simplest formal statement of the general hebbian rule is:

$$\Delta w_{BA} = \varepsilon V_B V_A \tag{3.10}$$

In this equation, the presynaptic unit is designated by A, while the postsynaptic

unit is B. Units A and B have average firing rates of V_A and V_B respectively. ε is a parameter that is used to adjust the rate of modification, or learning rate, applied to the change. The weight between the presynaptic unit A and postsynaptic unit B, w_{BA}, is therefore adjusted by some amount, depending on the average activities of A and B. When both activities are above average, then the change in weight is large. If one or both are low, then the change is small.

3.6.2 Associative Hebbian Equation for KA Model

Many variations of the basic hebbian equation are possible. In the KA model, a meaningful measure of arousal of a unit is how far the unit currently is away from its average state. Recall that typically in KA-II groups, because of excitatory-excitatory feedback, the units achieve some non-zero steady state. Using distance from the baseline (0 current) is therefore not a good measure of arousal for units in KA-II groups. Instead, we need to measure the amount of displacement away from the average steady state.

We will therefore modify the basic hebbian equation for use in the KA units. First we determine the mean activity over some time period of the unit. The mean activity over a time interval of 100ms will be designated by $\overline{V_A}$ and $\overline{V_B}$ for our pre and postsynaptic units. The hebbian equation used for our KA units then becomes:

$$\Delta w_{BA} = \varepsilon(V_B - \overline{V_B})(V_A - \overline{V_A}) \tag{3.11}$$

The quantities $(V_B - \overline{V_B})$ and $(V_A - \overline{V_A})$ determine how far, above or below, the present activity is from the non-zero steady state of the unit. The simultaneous occurrence of activity above (or below) the average in both units will cause relatively larger changes in the weight between the units to occur.

Equation 3.11 is the basic mechanism that will be used in the next examples of hebbian learning in the KA model. One thing to note about this equation is

that it is of a class of hebbian equations called associational hebbian equations. In the basic equation, the firing rate could only be some number at or above 0. As a result, all weight changes were necessarily positive, and the weight could only increase. Associational hebbian equations, such as 3.11, allow values, and therefore weight changes, that can be both positive and negative. I will explore this issue a bit further in the next sections.

In nonassociational mechanisms, strictly increasing weights pose a problem in that the weights usually cannot be allowed to increase indefinitely, and must therefore be limited somehow. This is usually done either through local means, such as a constant decay of each weight, or through global means, such as scaling all weights to keep the total within some upper limit. Since weights can move in both directions in associational mechanisms, these problems are less of a concern here. However, situations can occur, as will be seen, so that some means of regulating the weights is still necessary. In the simulations in the next sections I will be using a global scheme to define an upper bound on the total weight allowed.

3.6.3 Δw Under Phase Shift

The basic mechanism behind hebbian learning is the strengthening of weights between units that fire together. In units that display oscillating or rhythmic behavior this implies a simple fact: units must become synchronized in order for learning to reliably occur between them. While greater arousal above or below the average can cause a faster change; the important factor, over time, will be the units synchronization of activity.

For example, in figures 3.31 and 3.32 we show the activity of two oscillating units in the top part, and the resulting hebbian modification in the bottom part of the figure. In figure 3.31 the pre and postsynaptic units are completely out of phase. When one unit is at its maximum activity, the other is at its minimum. At this

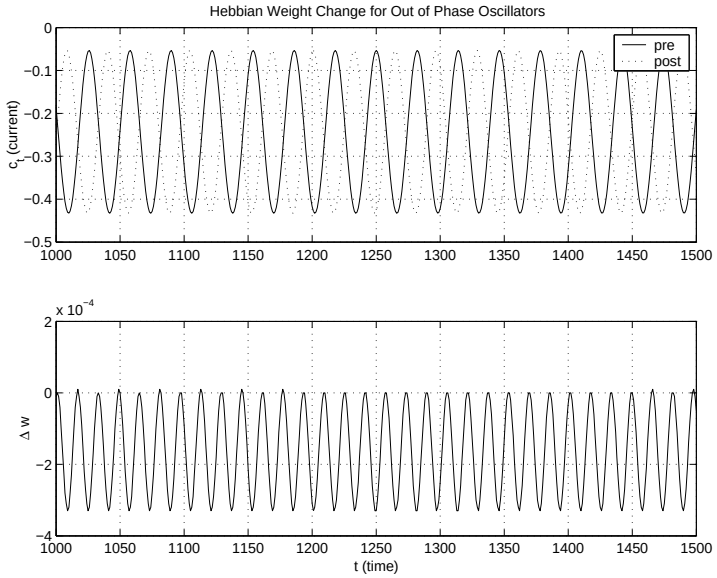

Figure 3.31: Associative hebbian weight change for out of phase oscillating pre and post synaptic units.

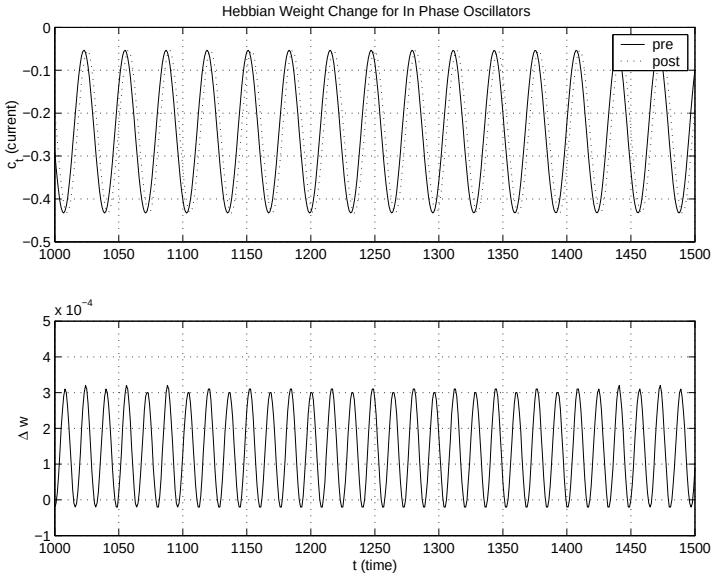

Figure 3.32: Associative hebbian weight change for in phase oscillating pre and post synaptic units.

point, a large negative change in weight is produced by the associative hebbian rule. As can be‘ seen from the bottom part of figure 3.31, the weight modification ranges from 0 to -.0003. The cumulative trend for out of phase units will be a ever decreasing weight.

In figure 3.32 we show the opposite situation, with pre and postsynaptic units that are nearly synchronized. In this situation the weight change is mostly positive and therefore increases over time. In populations started from different initial conditions, some individual weights will have a natural positive trend, while others have a negative trend. From such initial asymmetries can be formed meaningful patterns through competition. In a large and unchanging collection of units, the positive and negative trends might balance out (though not necessarily). However, in oscillating units, one effect of the strengthening of weights between them is to synchronize their oscillations. As more and more clusters of synchronized units are formed through learning, more positive natural trends in weight changes begin to occur as in figure 3.32. Therefore a means of regulating weights is still necessary, even with associational hebbian mechanisms between oscillating units.

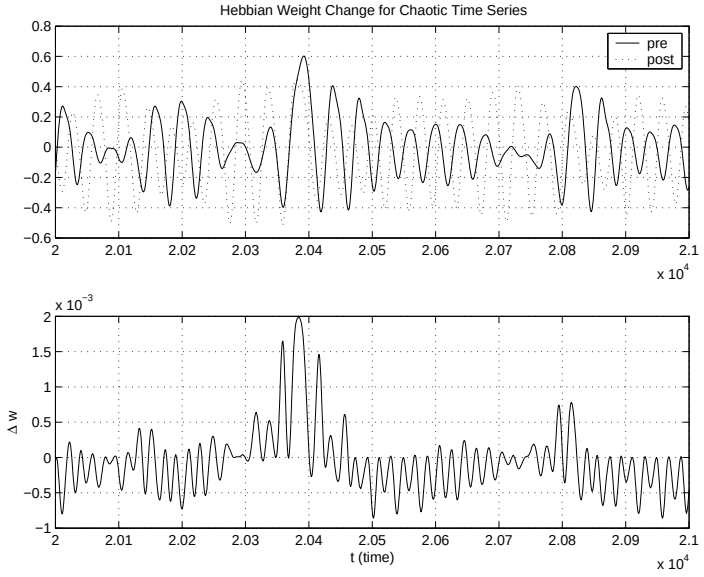

Figure 3.33: Associative hebbian weight change for chaotic time series.

3.6.4 Δw Between Chaotic Units

When three or more KA-II groups are connected together chaotic behavior may result. Figure 3.33 shows an example of hebbian modification between chaotic units.

As can be seen from the bottom part of the figure, the modification of the weight between the units is at times mostly below 0, and at other times above. Uncorrelated chaotic activity is much more likely to show zero drift than oscillatory behavior. In oscillatory behavior, only when the units are exactly 90° out of phase will the average weight change be at 0, a fairly unlikely and unstable situation. In contrast the correlation between chaotic activity produces a random walk that will, on average if the units are truly uncorrelated, have a zero bias trend. This may be one of the roles of chaotic behavior in neural populations, the balancing of hebbian synaptic modifications through the intrinsic dynamics of the populations.

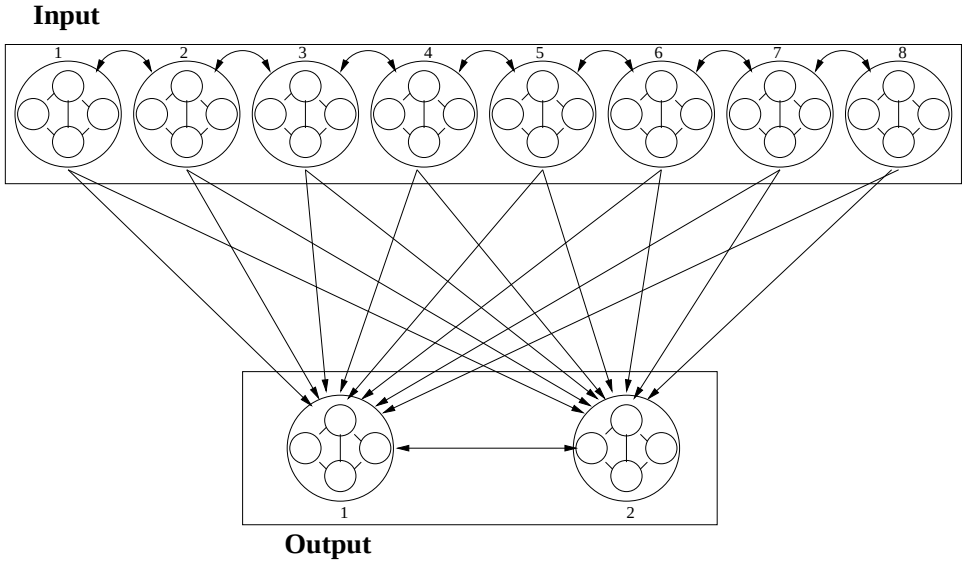

Figure 3.34: Experimental setup for hebbian learning example with oscillating dynamics. The input layer consists of 8 and the output layer of 2 homogeneous KA-II Groups. See text for full description.

3.6.5 Simple Hebbian Learning with KA-II Units

In this section I show a simple example of using hebbian associative learning to modify weights in a network of KA-II units. We will be using the simple architecture shown in figure 3.34. In this setup, we have a layer of 8 input KA-II groups, and a layer of 2 output KA-II groups. We fully connect the excitatory E_1 units of the KA-II

groups in the input layer to one another, to form a 8X8 gird of weights. We have not connected any of the inhibitory units in this layer. We also fully connect the 8 E_1 units of the input layer to the 2 E_1 units of the output layer. We will allow the 8x8 and 8x2 weight matrices to be modified by associative hebbian learning. All internal group weights will not be modified. In this experiment, we have used KA-II groups with identical internal parameters, in order to form homogeneous populations. The absence of intergroup inhibitory weights, along with the homogeneous populations, will bias the system to display synchronized oscillatory behavior.

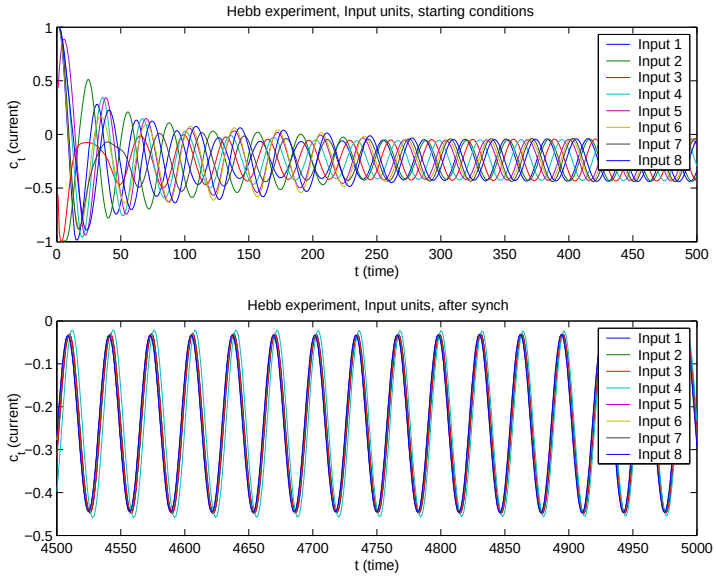

Figure 3.35: Time series of Input units for first and last 500ms before training in hebbian learning example. Units become synchronized because of interconnected excitatory weights.

In the beginning of the experiment, the 8x8 input-input and 8x2 input-output weights are set to random small values. The interconnectivity of all of the units with small influences allows the population to synchronize their oscillatory behavior. In figure 3.35 we show the result of this synchronization. The top part of the figure shows the 8 input units for the first 500ms. The 8 units are started with different random initial conditions. However, by 5000ms, as can be seen in the bottom portion

of the figure, the units have synchronized their behavior as a result of the mutual excitatory influences.

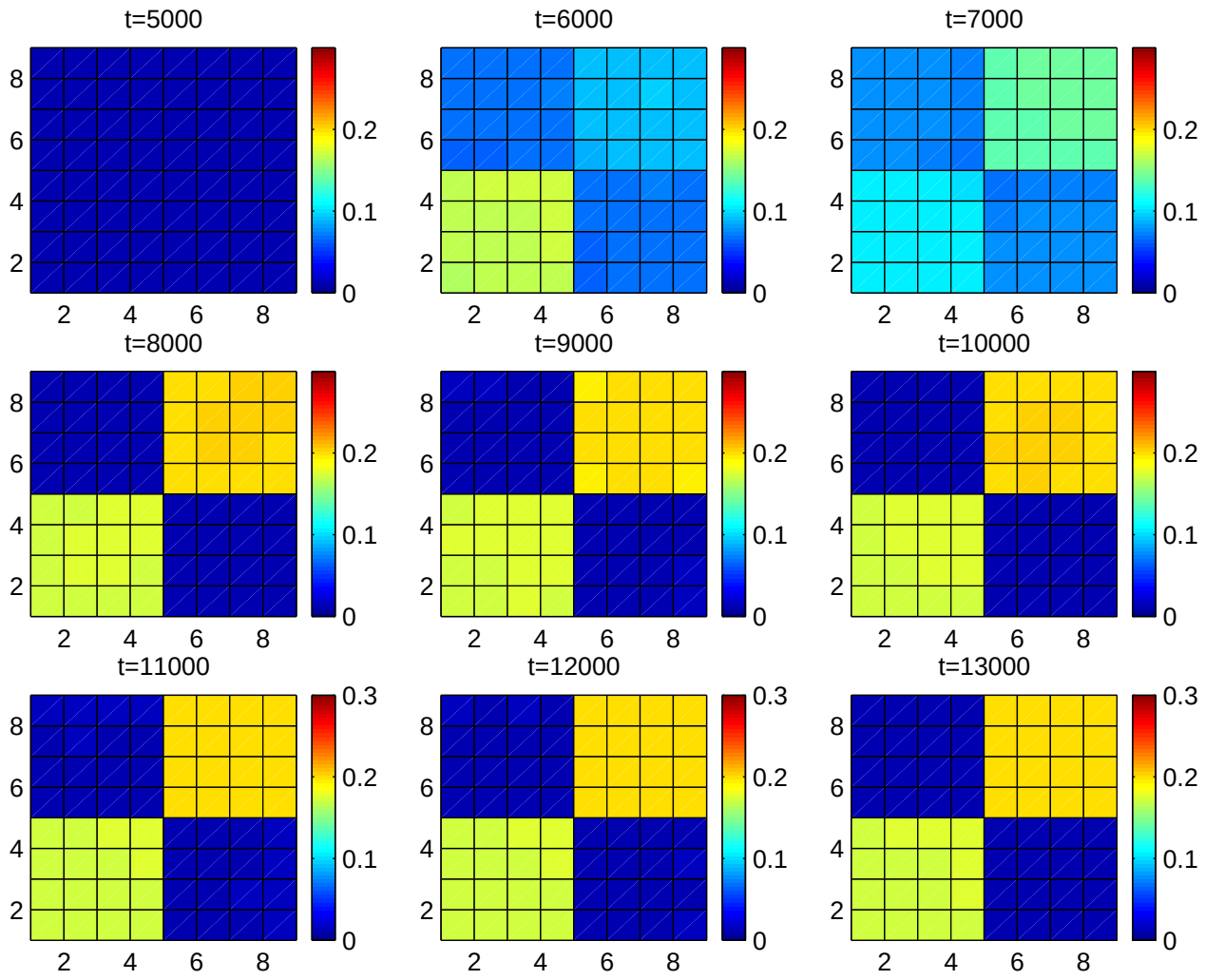

Figure 3.36: Evolution of interconnected input weights in hebbian learning example. All 8 E_1 units in the input layer are fully interconnected. We show the values of the 8x8 weight matrix from time 5000 to time 13000. See text for details.

As a result of the synchronization of behavior, we need to use the global weight regulation scheme mentioned previously to keep the weights from becoming saturated. We now begin the training and learning phase of the simulation. We externally stimulate input units 1-4 and output unit 1 for 50ms (0.25 stimulation), during which time we allow the weights to change through associational hebbian modification. We

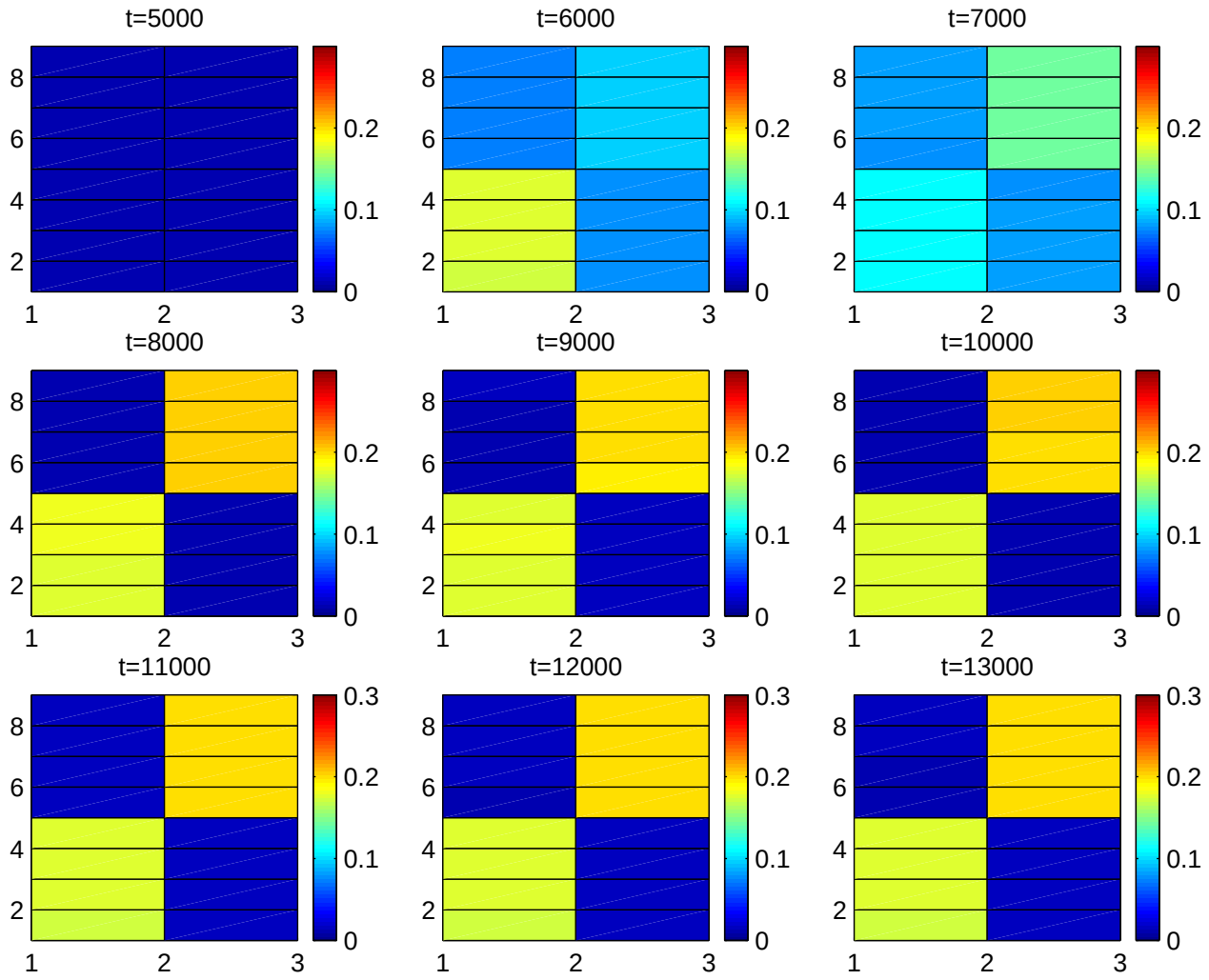

Figure 3.37: Evolution of input layer to output layer weights in hebbian learning example. All 8 E_1 units in the input layer are fully connected to the 2 E_1 units of the output layer. We show the values of the 2x8 weight matrix from time 5000 to time 13000. See text for details.

then allow the system to rest for 450ms, during which time no learning occurs. We then externally stimulate input units 5-8 and output unit 2 for 50ms (0.25 stimulation as well) and again turn on learning. After which another 450ms rest occurs to allow the system to relax. We perform this pattern of excitation and relaxation from t=5000ms to t=20000ms.

Figures 3.36 and 3.37 show the evolution of the input-input and input-output

weights respectively. In these figures color represents the magnitude of the weight between units, blue for a weight of 0 up to dark red for a weight of 0.3. We show snapshots of the weight space every 1000ms, which is the duration of a left/right presentation cycle. In figure 3.36 we can see that the 4 inputs that receive simultaneous stimulation quickly become associated with one another, and form two populations of highly connected groups (1-4 and 5-8). The pattern for the input-output weights is similar in figure 3.37. Here we see that the input units that are stimulated simultaneously with the output unit become highly connected.

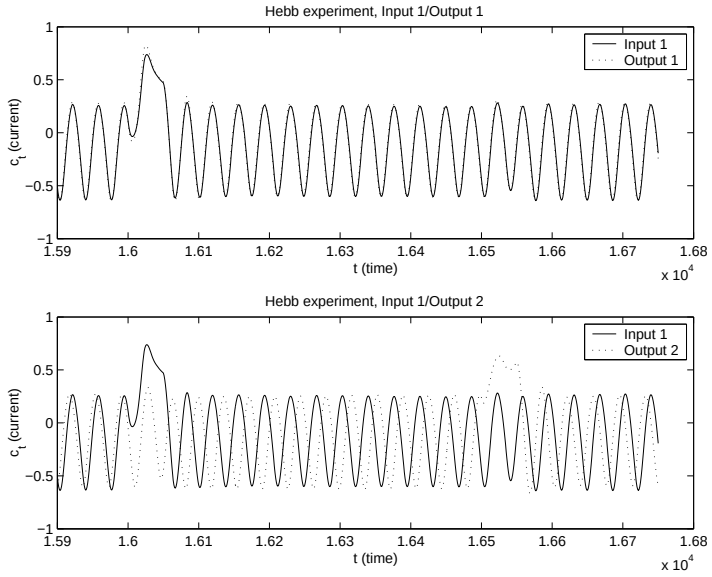

Figure 3.38: Time series of input 1 unit plotted against the output 1 and 2 units in the hebbian learning experiment. Input 1 remains synchronized with output 1, but is out of phase with output 2.

In figure 3.38 we compare the activity of input unit 1 to the output units. The top figure shows how input 1 and output 1 remain synchronized during one learning period from t=16,000ms to t=16,050ms. In contrast, input 1 is pushed out of phase with the output 2 unit, as shown in the bottom potion of the figure. Figure 3.39 shows a similar pattern of synchronicity between the input 5 unit and the output 2 unit. It is this synchronization of activity, more than anything, that is responsible

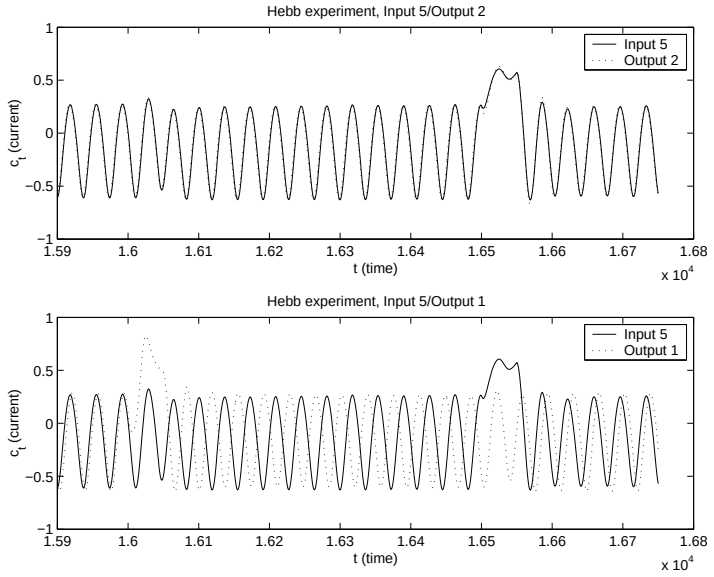

Figure 3.39: Time series of input 5 unit plotted against the output 1 and 2 units in the hebbian learning experiment. Input 5 remains synchronized with output 2, but is out of phase with output 1.

for the correlated change in weights that occur.

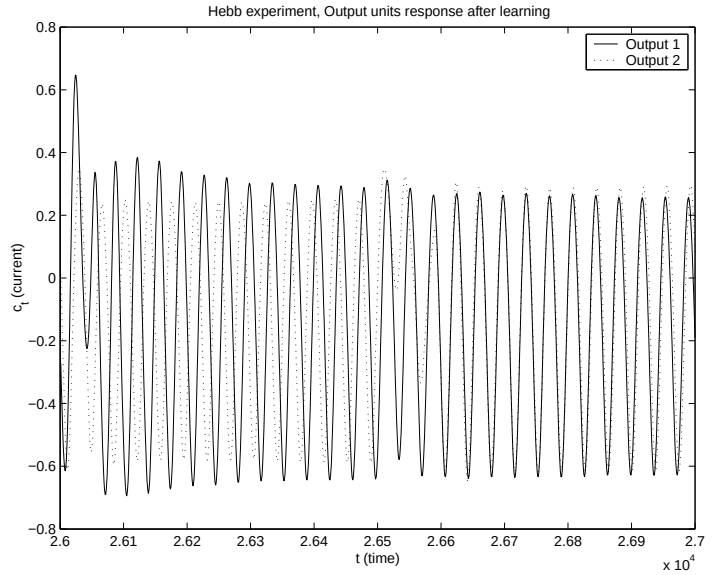

Figure 3.40: Response of the output units after learning in the hebbian example.

In figure 3.40 we show the responses of the output units after learning. In this figure we only stimulated the input units. We stimulated inputs 1-4 from t=26,000ms

to t=26,050ms; and similarly inputs 5-6 from t=26,500ms to t=26,550ms. The figure shows that the two output units now show slight responses to stimulation of the inputs that they became associated with during learning.

Chapter 4

Proposed Research

The purpose of the proposed research is to explore the following ideas. Current work in third generation dynamical connectionist and autonomous agent research (Almássy et al., 1998; Edelman et al., 1992; Matarić, 1991, 1995; Sporns et al., 1999; Verschure et al., 1992, 1995; Verschure & Voegtlin, 1999) has begun to emphasize the importance of exploring whole (complete) systems that are embodied and embedded in an environment (Franklin, 1995; Pfeifer & Scheier, 1998; Steels & Brooks, 1995). 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 (Clark, 1997; Hendriks-Jansen, 1996; Pfeifer & Scheier, 1998). This research proposes to extend such results by exploring the roles of chaotic 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 (Kelso, 1995; Port & van Gelder, 1995; Thelen & Smith, 1994). 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.

The work of Freeman and others (Skarda & Freeman, 1987; Freeman, 1999b; Freeman & Kozma, 2000; Freeman, Kozma, & Werbos, 2000; Tsuda, 2001; Tsuda & Yamaguchi, 1998; Kozma & Freeman, 1999, 2000, 2001; Kozma et al., 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 may provide 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 (Clark, 1997; Hendriks-Jansen, 1996) 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 basic 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.

Therefore the purpose of the proposed 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.

4.1 Statement of Research Objectives

The proposed research has the following central research questions in mind:

- Can chaotic dynamics be used to form perceptual categories in autonomous agents?
- Can the formation of perceptual attractors for perception be extended to form goal-state attractors, and attractors for memory systems and cognitive maps?

- Can chaotic dynamics be used for the hierarchical self-organization of goal-state contexts for the action selection mechanism of an autonomous agent?
- 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 proposed research:

- The demonstration of chaotic dynamics in forming perceptual categories in an autonomous agent.
- The demonstration of hierarchical 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.

4.2 Methods and Materials

In this section I present a brief overview of the agents, environments and tasks that are proposed for use in this research. The neural population KA model, described in the previous chapter, will provide the basic units for all simulations and agents to be developed by this research.

4.2.1 Agent

I am proposing to mainly use the khepera agent for much of this research. The khepera agent is both a virtual agent in a simulated environment, and a real robotic agent. One advantage of the khepera product, is that simulator and real robot have remained tightly coupled in development. Simulations developed on the virtual environment simulator often work with little modification in the real robotic agent.

Figure 2.3 from chapter 2 shows the basic body plan of the khepera agent. The robots morphology is simple, and consists of a circular body with sensors arrayed around the periphery and two independently controlled motorized wheels for performing movement. The typical configuration of the agent consists of 8 sensors in

total, with 6 arrayed in the front of the agent (towards the direction of forward motion) and 2 directed towards the rear. The standard configuration of khepera consists of two types of sensors, one type detects light intensity, and the other is an infra-red range finder that can detect proximity to an obstruction. The simulation of simple binary collision detectors (collision/no collision) can easily be done using the existing proximity sensors. Both types of sensors have a limited range, and are sensitive to a cone with a limited detection arc. The detection fields of the sensors do overlap in area. The wheels allow for forward and backward motion, plus the robot can turn in place, or in slower arcs.

Another robotic agent that may be used to conduct simulations is the Sony AIBO dog, which we have available for our use. This is a real world robot with a more anthropomorphic shape (e.g. a very small dog). It has 4 legs for movement and turning, each with three degrees of freedom. It has a head capable of three degrees of freedom as well, so that it can be turned to scan the environment without moving the position of the robot. The robot is equipped with two main sensors, a single range finder, capable of detecting obstacles at intermediate range, and a simple color camera.

4.2.2 Environment

In figure 4.1 I show a snapshot of the khepera simulator. The environments are easily configurable to allow for any placement of walls (obstructions) and light sources as desired. Light sources are often used to represent food sources or generic desiriable/undesiriable location/features for the agent simulations.

4.2.3 Tasks

Due to the lack of manipulators for the standard agent, tasks are mostly restricted to motion, navigation and exploration of the environment. I am planning on mainly

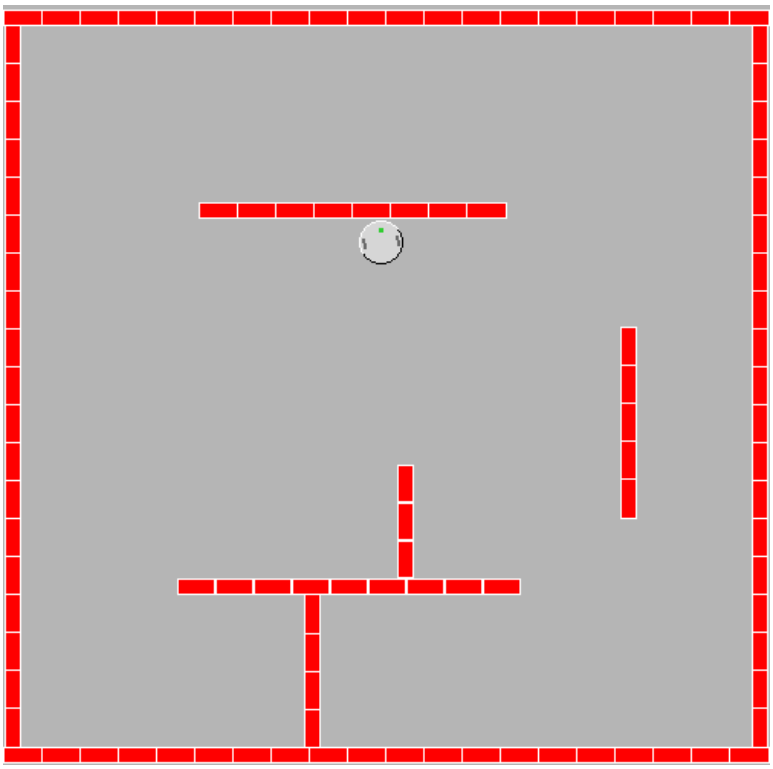

Figure 4.1: A snapshot of the environment of the khepera simulator.

focusing on two types of tasks. In one type of unmotivated exploration, the agent builds a spatio-temporal cognitive map representation of its environment. A second type of task involves motivation, in which the agent uses it cognitive map to reliably navigate to desired locations in order to perform behaviors and satisfy basic needs.

4.3 Proposed K-IV Architecture

In (Kozma, Freeman, & Erdi, 2003, 2002), Kozma et. al. have presented the next logical step up from the K-III model, the K-IV architecture. The purpose of the K-IV architecture is as a model of a basic limbic system. In this model, K-III units are used to capture the dynamics of each of the major portions of the limbic system: sensory, motor, associational and memory (figures 2.13 and 2.14). The combination of these K-III units to form a complete limbic system has been called a K-IV and is shown in

figure 4.2.

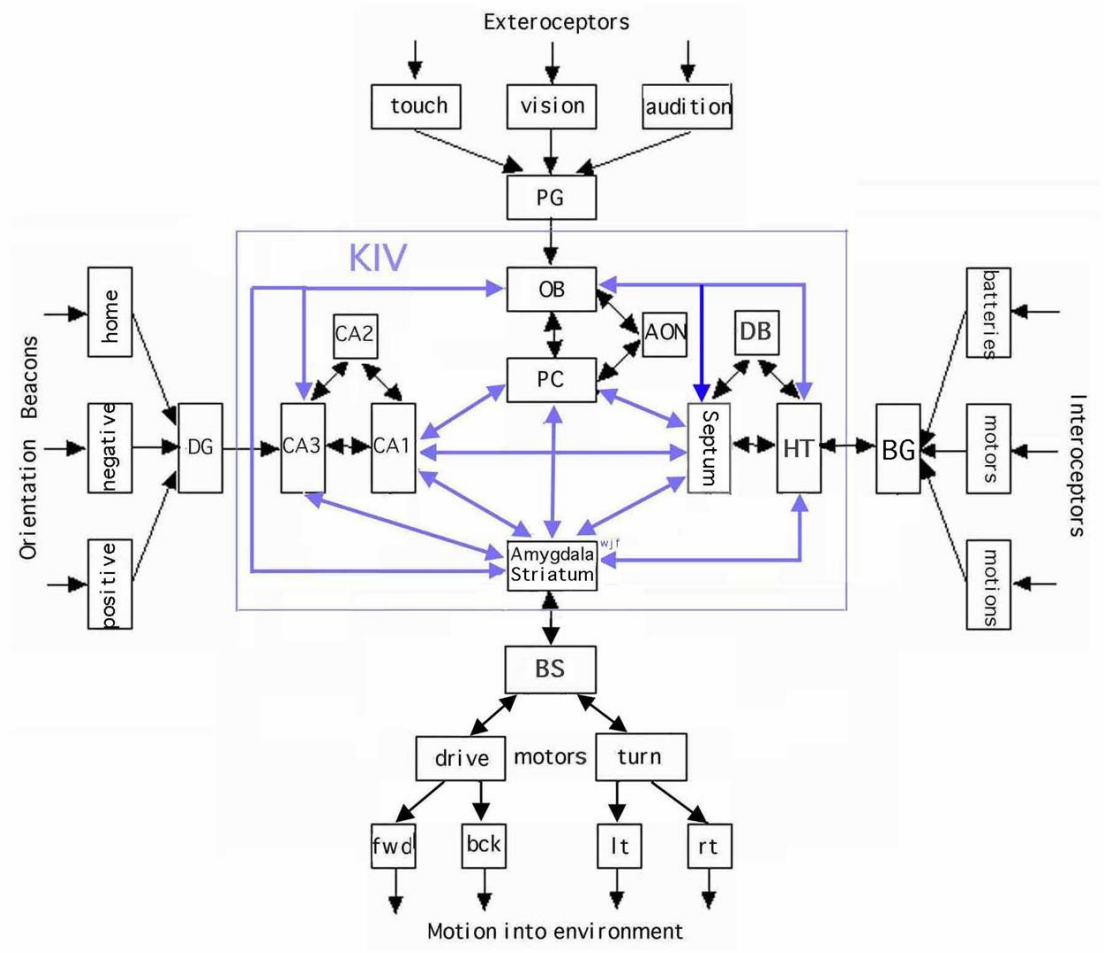

Figure 4.2: Proposed K-IV architecture, a model of a basic intentional system.

Each of the components in the center box of this figure (labeled KIV) forms a K-III system. The system at the top is for sensory dynamics and perceptual categorization (OB, AON and PC). The system to the right is for associational activity (Septum, DB and HT). While the system to the left is for memory (CA1, CA2, CA3). The dynamics in the motor system (Amygdala Striatum) do not require a full K-III and are captured in this architecture with a group of K-II units.

Comparing figure 4.2 to figure 2.14, the top K-III (OB, AON and PC), which receives input from the sensors, corresponds to the Sensory Systems area in figure 2.14.

The right K-III system (Septum, DB and HT) are the dynamics for the associational entorhinal cortex. The left K-III (CA1, CA2 and CA3) provide the dynamics of the hippocampal memory system.

Figure 4.2 represents a very high level representation of a possible model of the basic intentional system. It hypothesizes that dynamics of the major systems; perceptual, associational and memory; can be modeled using chaotic dynamics of the same kind discovered in the olfactory perceptual system, and captured by a K-III unit. The ultimate goal of the proposed research is to begin to work out the details of how to implement a K-IV basic intentional system for control of an autonomous agent.

4.4 Towards a K-IV for Autonomous Agents

Towards this goal of implementing a K-IV I am proposing a series of partial tasks in order to begin to work out some of the necessary details. One of the major subtasks, already completed and described in this proposal, was the development of basic units capable of the K-set dynamics but in discrete versions suitable for real-time autonomous agent simulations. I now present two ideas for the use of the KA units in developing autonomous agents.

4.4.1 KA-III for simple Action Selection

The first subtask would be the demonstration of simple behaviors (exploration, object avoidance, wall following), using a part of the K-IV architecture. In this task, we would remove the hippocampus in the K-IV architecture, and use the K-III of the perceptual, and associational areas, along with the motor control areas. The completion of this task would require working out some important details. For example, how exactly do the dynamics in the Amygdala striatum come to reliably control the simple

behaviors and motions of the agent. Some preliminary work leads us to believe that this is not a simple question. From an engineering perspective it will mean classifying different chaotic attractors and assigning the system to drive motors in certain ways when that attractor is entered into.

4.4.2 KA-III, Hippocampal Cognitive Map Formation and Phase Precession

Another obvious subtask is to model some functions of hippocampal cognitive map formation. In this task we would model only the K-III of the hippocampus, and we would simulate appropriate sensory (and associational) dynamics externally. The hippocampus is known to form so called place cells when learning a cognitive map of its environment. The spatial pattern of these place cells are also known to completely change when new landmarks of a familiar environment appear. This rearrangement of patterns is of course reminiscent of the reshaping of the attractor landscape upon the formation of new perceptual categories. Place cells, I believe, are equivalent to the formation of AM spatial patterns in sensory systems. Development of cognitive maps using a KA-III would allow us to explore the properties of place cells formed using the principles of chaotic dynamics.

Another phenomenon observed in the hippocampus has been called phase precession, or phase advance. This is a well known observed phenomenon in which the firing of place cells in the hippocampus change phase relative to the normal Θ cycle of the brain. This phase advance is correlated with movement towards a known location.

One possible interpretation of this phenomenon is as evidence of what I will call the formation of an intentional loop. What I am suggesting has happened is that the organism has formed an intention to move from its current location to a goal location. The formation of this plan of action would be of a similar kind to the chaining of associational pairs to form a logical inference. In this instance it would be a chaining

of intermediate locations to imagine a path from the current location to where the organism wishes to be. However the intended path is created, I imagine that a type of short term memory, or intentional loop, is then formed. This is a temporal activation of the sequence of place cell (or AM) patterns, that represent the locations to traverse to get from here to the goal.

Therefore in this task the objectives would be 1) to demonstrate the use of AM spatial patterns as place cells in cognitive maps; and 2) to explore the formation of an intentional loop for moving from the current location to a goal location in the phenomenon of phase advance.

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.

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

Bechtel, W. (1998). Representations and cognitive explanations: Assessing the dynamicist's challenge in cognitive science. *Cognitive Science*, *22*, 295–318.

Beer, R. D. (2000). Dynamical approaches to cognitive science. *Trends in Cognitive Sciences*, *4*, 91–99.

Braitenberg, V. (1984). *Vehicles: Experiments in synthetic psychology.* Cambridge, MA: The MIT Press.

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

Brooks, R. A. (1995). Intelligence without reason. In L. Steels & R. Brooks (Eds.), (pp. 25–81). Hillsdale, NJ: Lawrence Erlbaum Associates, Inc.

Churchland, P. S., & Sejnowski, T. J. (1992). *The computational brain.* Cambridge, MA: The MIT 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.

Clayton, K., & Frey, B. (1997). Studies of mental noise. *Nonlinear Dynamics, Psychology and Life Sciences*, *1*, 173–180.

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.

Edelman, G. M. (1987). *Neural darwinism: The theory of neuronal group selection.* New York, NY: Basic Books.

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.

Edelman, G. M., & Tononi, G. (2000). *A universe of consciousness: How matter becomes imagination.* New York, NY: Basic Books.

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. (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., & 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.

Friston, K. J., Tononi, G., Reeke, G. N. J., Sporns, O., & Edelman, G. M. (1994). Value-dependent selection in the brain: Simulation in a synthetic neural model. *Neuroscience*, *59*(2), 229–243.

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

Haken, H., Kelso, J. A. S., & Bunz, H. (1985). A theoretical model of phase transitions in human hand movements. *Biological Cybernetics*, *51*, 347–356.

Haken, H., & Stadler, M. (Eds.). (1990). *Synergetics of cognition.* Springer-Verlag.

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. In *Proceedings of the intelligent systems design and applications 2001 workshop (ISDA 2001)* (pp. 300–309). San Francisco, CA.

Harter, D., & Kozma, R. (2002a). Chaotic neurodynamics for modeling intentional systems. *IEEE Transactions on Neural Networks.* (in progress)

Harter, D., & Kozma, R. (2002b). 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., & 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.

Hassell, M. P., Comins, H., & May, R. M. (1991). Spatial structure and chaos in insect population dynamics. *Nature*, *353*, 255–258.

Hebb, D. O. (1949). *Organization of behavior.* New York: Wiley.

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

Hoyt, D. F., & Taylor, R. C. (1981). Gait and energetics of locomotion in horses. *Nature*, *292*, 239–240.

Iverson, J. M., & Thelen, E. (1999). Hand, mouth and brain: The dynamic emergence of speech and gesture. In R. Núñez & W. J. Freeman (Eds.), (pp. 19–40). Bowling Green, OH: Imprint Academic.

Johnson-Laird, P. N. (1988). *The computer and the mind: An introduction to cognitive science.* Cambridge, MA: Harvard University Press.

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.

Kirsh, D., & Maglio, P. (1994). On distinguishing epistemic from pragmatic action. *Cognitive Science, 18*, 513–549.

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. (2001). 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., & 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, 50.* (in press)

Kozma, R., Harter, D., & Achunala, S. (2002a). 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., & Achunala, S. (2002b). Constraints and the dynamic mechanisms of behavior generation. *Cognitive Systems Research, 4*(1). (submitted)

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.

Kugler, P. N., Kelso, J. A. S., & Turvey, M. T. (1982). On the control and coordination of naturally developing systems. In J. A. S. Kelso & J. E. Clark (Eds.), *The development of movement control and coordination.* Wiley.

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

Maes, P. (Ed.). (1990). *Designing autonomous agents: Theory and practice from biology to engineering and back.* Cambridge, MA: The MIT 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.

Makse, H. A., Havlin, S., & Stanley, H. E. (1995). Modeling urban growth patterns. *Nature*, *377*, 608–612.

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.

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.

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. (1996). Building "fungus eaters": Design principles of autonomous agents. In P. Maes, M. J. Matarić, J.-A. Meyer, J. Pollack, & S. W. Wilson (Eds.), (pp. 3–12). Cambridge, MA: The MIT 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.

Poon, C. S., & Merrill, C. K. (1997). Decrease of cardiac chaos in congestive heart failure. *Nature*, *389*, 492-495.

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

Pylyshyn, Z. W. (Ed.). (1987). *The robot's dilemma: The frame problem in artificial intelligence.* Norwood, NJ: Ablex.

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.

Scheier, C., & Tschacher, W. (1996). Appropriate algorithms for nonlinear time series analysis in psychology. In W. Sulis & A. Combs (Eds.), *Nonlinear dynamics in human behavior* (pp. 27–43). World Scientific.

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. V., Manribia, S. C., Benton, M., Kauffman, S. A., & Bak, P. (1997). Self-similarity of extinction statistics in the fossil record. *Nature*, *388*, 764–767.

Solé, R. V., & Valls, J. (1992). Spiral waves, chaos and multiple attractors in lattice models of interactive populations. *Phys. Lett. A*, *166*, 123–128.

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.

Thom, R. (1983). *Mathematical models of morphogenesis.* Ellis Horwood.

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

Tschacher, W., & Dauwalder, J.-P. (Eds.). (1998). *Dynamics, synergetics and autonomous agents: Nonlinear systems approaches to cognitive psychology and cognitive science.* Singapore: World Scientific.

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

Tsuda, I., & Yamaguchi, A. (1998). Singular-continuous nowhere differentiable attractors in neural systems. *Neural Networks*, *5*, 927–937.

van Gelder, T. (1998). The dynamical hypothesis in cognitive science. *Behavioral and Brain Sciences*, *21*, 615–665.

Varela, F. J., Thompson, E., & Rosch, E. (1993). *The embodied mind: Cognitive science and human experience.* Cambridge, MA: The MIT Press.

Verschure, P. F. M. J. (1998). Distributed adaptive control: Explorations in robotics and the biology of learning. *Informatik/Informatique*, *1*, 25–29.

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., & Pfeifer, R. (1993). Categorization, representations, and the dynamics of system-environment interaction: A case study in autonomous systems. In J.-A. Meyer, H. L. Roitblat, & S. W. Wilson (Eds.), (pp. 210–217). Cambridge, MA: The MIT Press.

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.

Ward, L. M. (2002). *Dynamical cognitive science.* 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.

Appendix A

Principles of Chaotic Neurodynamics

Freeman's ten principle building blocks of chaotic neurodynamics (Freeman, 1999b, p. 37):

1. The **state transition** of an excitatory population from a point attractor with zero activity to a **non-zero point attractor** with steady-state activity by **positive feedback**.
2. The emergence of **oscillation** through **negative feedback** between excitatory and inhibitory neural populations.
3. The state transition from a point attractor to a **limit cycle attractor** that regulates steady-state oscillation of a mixed excitatory-inhibitory cortical population.
4. The genesis of **chaos** as **background activity** by combined negative and positive feedback among three or more mixed excitatory-inhibitory populations.
5. The distributed wave of chaotic dendritic activity that carries a spatial pattern of **amplitude modulation** made by the local heights of the wave.
6. The increase in **nonlinear feedback gain** that is driven by input to a mixed population, which results in construction of an amplitude-modulation pattern as the first step in perception.
7. The **embodiment of meaning** in amplitude-modulation patterns of neural activity, which are shaped by synaptic interactions that have been modified through learning.
8. **Attenuation** of microscopic sensory-driven activity and **enhancement** of macroscopic amplitude-modulation patterns by **divergent-convergent** cortical projections underlying solipsism.
9. The divergence of corollary discharges in **preafference** followed by **multisensory convergence** into the entorhinal cortex as the basis for Gestalt formation.
10. The formation of a sequence of **global amplitude-modulation patterns** of chaotic activity that integrates and directs the intentional state of an entire hemisphere.

Appendix B

KA Variables, Parameters and Equations

The variables, parameters and equations used to implement the KA model are reproduced here for reference. The parameters, and their stated default values, were used for all simulations presented in this proposal. The default values of the parameters were chosen to approximate the open loop behavior and timing of an isolated neural population when stimulated.

B.1 Variables

Table B.1: KA Model Variables

Variable	Description
c_t	Simulated current at time t
r_t	Rate of change of current at time t.
μ_t	Difference to be applied to current at time t
μ_t^d	Difference at time t due to decay to baseline
μ_t^m	Difference at time t due to momentum
μ_t^e	Difference at time t due to external input

B.2 Parameters

Table B.2: KA Model Parameters

Parameter	Description	Default
α	Rate of decay to baseline	0.035
β	Rate of momentum	0.9
γ	Input scaling parameter	0.025
ϵ	Transfer function arousal level	3.0
η	Saturation threshold	0.75
λ	Saturation scaling ratio	0.5

B.3 Equations

$$c_{t+1} = c_t + \mu_t \tag{B.1}$$

$$\mu_t^d = -c_t \times \alpha \tag{B.2}$$

$$r_t = c_t - c_{t-1} \tag{B.3}$$

$$\mu_t^m = r_t \times \beta \tag{B.4}$$

$$\mu_t^e = \sum_{i=1}^{N} f(c_t^i) w_{ji} \gamma \tag{B.5}$$

$$o_t = \epsilon\{1 - exp[\frac{-(e^{c_t} - 1)}{\epsilon}]\} \tag{B.6}$$

$$o_t = \begin{cases} o_t & \text{if unit is excitatory} \\ -o_t & \text{if unit is inhibitory} \end{cases} \tag{B.7}$$

$$\mu_t' = \mu_t^d + \mu_t^m + \mu_t^e \tag{B.8}$$

$$\mu_t = \begin{cases} \mu_t' & \text{if } |c_t + \mu_t'| \leq \eta \\ \mu_t'(\frac{1-|c_t|}{1-\eta})^\lambda & \text{if } |c_t + \mu_t'| > \eta \end{cases} \tag{B.9}$$

$$\Delta w_{BA} = \varepsilon(V_B - \overline{V_B})(V_A - \overline{V_A}) \tag{B.10}$$

Appendix C

KA Source Code Listing

C.1 Axon

C.1.1 Axon.h

```
//
// $Id: Axon.h,v 1.6 2002/05/12 15:37:00 dharter Exp $
//
// This module implements a Axon to connect up Neuron objects.  The Axon
// objects are the links between objects, and as such have a weight value
// and the behavior to change weight values using hebbian learning.

class Neuron;

class Axon
{
  public:
   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);
   void revhebb(double n, double lr, double dr, double avg);
   void waste(double ratio);
   void showPulses();
   static void goodEvent(double lr, double dr, double targetAvg);
   static void goodEvent2(double lr, double dr, double targetAvg);
   static void badEvent(double lr, double dr, double targetAvg);

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

C.1.2 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.  The Axon
// objects are the links between objects, and as such have a weight value
// and the behavior to change weight values using hebbian learning.

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

```
// static member
const int MAXAXONS=20000;
const int MAXGROUPS=1;

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

// Constructors
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);
}

// Common initialization function
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 = 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]++;
      }

   }

}

// 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 avg)
{
```

```
   double dw, sc, dc;
   sc = source->current - source->avgCurrent;
   dc = dest->current - dest->avgCurrent;

   dw = lr * sc * dc;

   weight += dw;
   //weight -= 0.0000020602; // constant habituation decay

   if (weight > 2.0) weight = 2.0;
   if (weight < 0.01) weight = 0.01;
}

// Reverse the sense of hebbian learning
void Axon::revhebb(double n, double lr, double dr, double avg)
{
   double dw, sc, dc;
   sc = source->current - source->avgCurrent;
   dc = dest->current - dest->avgCurrent;

   dw = lr * sc * dc;

   weight -= dw;
   //weight -= 0.0001; // constant habituation decay
   if (weight > 2.0) weight = 2.0;
   if (weight < 0.0) weight = 0.0;
}

// 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 hebbian learning on axons
void Axon::goodEvent(double lr, double dr, double targetAvg)
{
   double totWeight[MAXGROUPS];
   double avg[MAXGROUPS];
   Axon* axon;
   int i,j;


   // do learning
   for (i=0; i<MAXGROUPS; i++)
   {
      totWeight[i] = 0.0;
      for (j=0; j<axonCount[i]; j++)
      {
 axon = axons[i][j];
 axon->hebb(axonCount[i], lr, dr, 0);
 totWeight[i] += axon->weight;
      }
   }



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

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

      if (totWeight[i] > targetWeight)
      {
 double ratio = targetWeight / totWeight[i];
 for (j=0; j<axonCount[i]; j++)
 {
    axon = axons[i][j];
    axon->waste(ratio);
 }
      }
   }

}


// static member functions for causing reverse hebbian learning on axons
void Axon::badEvent(double lr, double dr, double targetAvg)
{
```

```
   double totWeight[MAXGROUPS];
   double avg[MAXGROUPS];
   Axon* axon;
   int i,j;

   // do learning
   for (i=0; i<MAXGROUPS; i++)
   {
      totWeight[i] = 0.0;
      for (j=0; j<axonCount[i]; j++)
      {
 axon = axons[i][j];
 axon->revhebb(axonCount[i], lr, dr, 0);
 totWeight[i] += axon->weight;
      }
   }



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

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

      if (totWeight[i] > targetWeight)
      {
 double ratio = targetWeight / totWeight[i];
 for (j=0; j<axonCount[i]; j++)
 {
    axon = axons[i][j];
    axon->waste(ratio);
 }
      }
   }
}
```

C.2 KAii

C.2.1 KAii.prop

DEFAULTSIZE: 1

C.2.2 KAii.h

```
//
// $Id: KAii.h,v 1.1 2002/01/29 16:44:54 dharter Exp $
//
// The KAii class is a collection class.  It groups 2 excitatory and 2
// inhibitory Neuron objects into the typical KAii configuration with
// 10 internal weights.
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:

};
```

C.2.3 KAii.cpp

```
//
```

```
// $Id: KAii.cpp,v 1.1 2002/01/29 16:44:54 dharter Exp $
//
// The KAii class is a collection class.  It groups 2 excitatory and 2
// inhibitory Neuron objects into the typical KAii configuration with
// 10 internal weights.

#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();
}
```

C.3 NeuroUtilities

C.3.1 NeuroUtilities.h

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

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

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

C.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 in KA simulations.

#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()
{
}
```

C.4 Neuron

C.4.1 Neuron.prop

```
# Neuron firing constants
PULSE: 1.0
TIME: 1
AROUSAL: 3.0

# Max number of connections
AXONCONNECTIONS: 255

# constants for pulse-wave conversions
# DECAY (alpha) is the decay to baseline slope constant
DECAY: 0.035

# MOMENTUM (beta) is the momentum constant
MOMENTUM: 0.9

# INPUTSCALING (gamma) is the input scaling constant
INPUTSCALING: 0.025

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

C.4.2 Neuron.h

```
//
// $Id: Neuron.h,v 1.7 2002/05/12 15:37:00 dharter Exp $
//
// This module implements a basic Neuron (unit) of the KA model.
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 adjmax;
   double adjmin;
   double adjrange;

   double pulseAmount;

   double cq[100];
   int cqi;
   double avgCurrent;

   Axon** connections;
   NeuronGroup* group;

   // class behaviors
   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);
   void tick();
   void reset();
   void pulse(double ratio);
   void setArousal(double arousal);
   void rest();
   void setGroup(NeuronGroup* group);
   double getAverageWeight();
   static void tickClock();
   static void newSim();

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

};
```

C.4.3 Neuron.cpp

```
//
// $Id: Neuron.cpp,v 1.7 2002/05/12 15:37:00 dharter Exp $
//
// This module implements a basic Neuron (unit) of the KA model.

#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 100

// 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->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");
      }
   }

}


// 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;

   // 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
   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;
   }

   // update variables to reflect differences for this time step
   prevCurrent = current;
   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;
```

```
   // send out pulse if needed
   ensembleWavePulse();

}

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

   c = (current * 4.0);
   ratio = arousal * (1.0 - exp(-(exp(c)-1.0)/arousal));
   ratio = (ratio - adjmin) / adjrange;
   if (ratio > 0.005)
   {
      fire(ratio);
   }
   else
   {
      rest();
   }
}

// fire a pulse to all of our connections
void Neuron::fire(double ratio)
{

   for (int i=0; i<numConnections; i++)
   {
      connections[i]->pulse(pulseAmount * ratio);
   }
}

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

   // send (non)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)
{
   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++)
```

```
   {
      totweight += connections[i]->weight;
   }
   return totweight / (numConnections-3);

}

// static member functions for causing all neurons to tick their clock
void Neuron::tickClock()
{
   for (int i=0; i<neuronCount; i++)
   {
      neurons[i]->tick();
   }
}

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

C.5 NeuronGroup

C.5.1 NeuronGroup.h

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

  private:

};
```

C.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.  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);
}
```

C.6 Properties

C.6.1 Properties.h

```
//
// $Id: Properties.h,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.
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();
};
```

C.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/derek/work/sodas/neurodyn/base/Axon.prop:/home/derek/work/sodas/neurodyn/base/Neuron.prop:/home/derek/work/sodas/neuro
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);
   }
}
```