1990

Computational Neural Learning Formalisms for Perceptual Manipulation: Singularity Interaction Dynamics Model.

Sandeep Gulati
Louisiana State University and Agricultural & Mechanical College

Order Number 9123194

Computational neural learning formalisms for perceptual manipulation: Singularity interaction dynamics model

Gulati, Sandeep, Ph.D.

The Louisiana State University and Agricultural and Mechanical Col., 1990

U·M·I

300 N. Zeeb Rd.
Ann Arbor, MI 48106

Computational Neural Learning Formalisms For Perceptual Manipulation : Singularity Interaction Dynamics Model

A Dissertation

Submitted to the Graduate Faculty of the
Louisiana State University and
Agricultural and Mechanical College
in partial fulfillment of the
requirements for the degree of
Doctor of Philosophy

in

The Department of Computer Science

by

Sandeep Gulati

B. Tech, Indian Institute of Technology, New Delhi, 1986
M.S., Louisiana State University, Baton Rouge, 1988

December 1990

Acknowledgements

I would like to express my deepest gratitude to my mentors, my dissertation advisor Professor Sitharama Iyengar, and Dr. Jacob Barhen for channelizing my intellectual energies and guiding me during my maturing process as a researcher. As my major professor, Dr. Iyengar introduced me to research and bold thinking, and has significantly contributed all along. The methodological approach adopted in this dissertation has been heavily influenced by his work in high performance computing. All along, he provided an exceptionally productive research environment, insisting on solving deep problems and accomplishing quality work. His unreserved support and enormous patience through some difficult times will be particularly remembered.

It has been a most enriching experience, both scientific and personal, to collaborate with and learn from Dr. Jacob Barhen. His erudition and enthusiasm for quantitative reasoning and rigor are truly inspiring. In fact, the indoctrination to neural networks, nonlinear science and dynamical systems theory occurred under his insightful guidance. I have enjoyed a very intense and productive collaboration with him over the last four years. A significant portion of the work presented here has its basis in his contributions to nonlinear sensitivity theory and its applications to energy modeling and nuclear science.

Dr. Michail Zak's fundamental contributions to neural networks have had a significant influence on this work. Interaction with him has yielded extensive qualitative insight into dynamical systems and chaotic phenomena. I am thankful to Professor Subhash Kak for his constant encouragement and strong interest in this work over the years. The extension of neural networks applications to signal processing evolved from interaction with him. My sincere appreciation is also expressed to my committee members, Professor Donald Kraft for provid-

ing a highly conducive research environment and setting high standards, and Professors Doris Carver and Bush Jones for their care, constant support and suggestions and help. I thank my friend S.T. Venkataraman for helping me hone in my interest in perceptual robotics. His work on task dependent control and dexterous manipulation has helped serve as a source of inspiration. He provided me guidance, advice and strength whenever I needed it. Nikzad Toomarian, with his uncompromising standards on rigor and integrity, has always been a source of objective criticism. In addition, Ken Kreutz, Fernando Pineda and Amir Fijany have helped tremendously. I would like to thank my peers Rao, Viji, Jana, Mohan, M. Stinson, John Fuller and R. Subbiah for the many interesting discussions I have shared with them.

I would have to express my gratitude to my parents Surendra and Swarn, my brother Poonam, and my special friends Shashi and Maneesh, for their love, support and understanding without which I could never have reached this stage.

I also thank Professor Iyengar for continually supporting me as a research assistant, and providing excellent research facilities at the LSU Robotics Research Laboratory. I would like to thank Dr. Barhen for the opportunity to conduct research at NASA's Jet Propulsion Laboratory, during various stages of my tenure as a student at LSU, and Dr. Barry Jacobs of NASA's Goddard Space Flight Center and Dr. Charles Weisbin of DOE's Oak Ridge National Laboratory. A part of the research was carried out at the Center for Space Microelectronics Technology within the Jet Propulsion Laboratory, California Institute of Technology, under contract NAS-918 with NASA. Support for the work also came from agencies of the U.S. Department of Defense, including the Innovative Science and Technology Office of the SDIO, DOE and NASA. The robotics research is supported in part by the JPL Director's Discretionary Fund.

TABLE OF CONTENTS

LIST OF FIGURES

List of Tables

Abstract

This dissertation addresses a fundamental problem in computational AI – developing a class of massively parallel, neural algorithms for learning robustly, and in real-time, complex nonlinear transformations from representative exemplars. Provision of such a capability is at the core of many real-life problems in robotics, signal processing and control. The concepts of terminal attractors in dynamical systems theory and adjoint operators in nonlinear sensitivity theory are exploited to provide a firm mathematical foundation for learning such mappings with dynamical neural networks, while achieving a dramatic reduction in the overall computational costs. Further, we derive an efficient methodology for handling a multiplicity of application-specific constraints during run-time, that precludes additional retraining or disturbing the synaptic structure of the "learned" network.

The scalability of proposed theoretical models to large-scale embodiments in neural hardware is analyzed. Neurodynamical parameters, e.g., decay constants, response gains, etc., are systematically analyzed to understand their implications on network scalability, convergence, throughput and fault tolerance, during both concurrent simulations and implementation in concurrently asynchronous VLSI, optical and opto-electronic hardware. Dynamical diagnostics, e.g., Lyapunov exponents, are used to formally characterize the widely observed dynamical instability in neural networks as "emergent computational chaos". Using contracting operators and nonconstructive theorems from fixed point theory, we rigorously derive necessary and sufficient conditions for eliminating all oscillatory and chaotic behavior in additive-type networks. Extensive benchmarking experiments are conducted with arbitrarily large neural networks (over 100 million interconnects) to verify the methodological robustness of our network "conditioning" formalisms.

Finally, we provide insight for exploiting our proposed repertoire of neural learning formalisms in addressing a fundamental problem in robotics - manipulation controller design for robots operating in unpredictable environments. Using some recent results in task analysis and dynamic modeling we develop the "Perceptual Manipulation Architecture". The architecture, conceptualized within a perceptual framework, is shown to be well beyond the state-of-the-art model-directed robotics. For a stronger physical interpretation of its implications, our discussions are embedded in context of a novel systems' concept for automated space operations.

Chapter One

Introduction

In the past few years the quest for efficient computational approaches to artificial intelligence and cognitive engineering has undergone a significant evolution. Not only is this transformation, from discrete symbolic reasoning to massively parallel connectionist neuroprocessing of compelling scientific interest, but also is of tremendous practical interest. It is changing the very rubric of information processing and problem solving. In general, the scientific and engineering community is contested with two basic categories of problems. First, there are problems that are clearly defined and deterministic. They are targeted for situations that are completely deterministic, precisely controllable, and can best be handled by computers employing rigorous, precise logic, algorithms, or production rules. This class deals with *structured problems* such as sorting, data processing and automated assembly in controlled workspace. On the other hand, there are scenarios such as maintenance of nuclear plants, undersea mining, battle management and assembly/repair of space satellites, that lead to computational problems that are inherently ill-posed and ill-conditioned [17,61,149]. Such *unstructured problems* entail providing for situations that may have received no prior treatment or thought. Decisions need to be made, based on information that is incomplete, often ambiguous, plagued with imperfect or inexact knowledge, and involve the handling of large sets of competing constraints that can tolerate "close enough" solutions. The outcome depends, on very many inputs and their statistical variations, and there is not a clear logical method for arriving at the answer. In summary, this category encapsulates problems that cannot be satisfactorily addressed using traditional computational paradigms such as

Random Access Machines [3,216], Markov Algorithms [218], Universal Turing Machines [4], Cellular Automata [271], Recursive Function Theory [247] or Production Systems [148,208,286]. The focus of artificial intelligence and machine learning has traditionally been to understand, and engineer systems that can address such unstructured computational problems.

Engineered intelligent systems, e.g., expert systems with some embedded reasoning, autonomous robots and rovers for space applications, behave with remarkable rigidity when compared to their biological counterparts, especially in their ability to recognize objects or speech, to manipulate and adapt in an unstructured environment and to learn from past experience. They lack common sense knowledge and reasoning, knowledge structures for recognizing complex patterns - they fail to recognize their own limitations. They are insensitive to context and are likely to give incorrect responses to queries that are outside the domains for which they are programmed. Algorithmic structuring fails to match the biological computational machinery when it comes to taking sensory information and acting on it, specially when the sensors are bombarded by a range of different, and in some cases competing stimuli. On the other hand, the biological machinery is capable of providing satisfactory solutions to such ill-structured problems with remarkable ease and flexibility [14,96,97,175,205]. A key emphasis underlying any paradigmatic development for unstructured computation today, is to understand how the aforementioned unstructured computations are interpreted, organized and carried out by the biological systems. The latter exhibit a spontaneous emergent ability that enables them to self-organize and adapt their structure and function.

A major reason for this limited technical success in emulating some of the more fundamental aspects of human intelligence lies in the differences between the organization, structuring of knowledge, and the dynamics of biological neuronal

circuitry and its emulation using the *symbolic processing paradigm* [41,76,124]. For example, it has been widely hypothesized [133,154,185-187,296] that "analogy and reminding guide all our thought patterns and that being attuned to vague resemblances is the hallmark of intelligence". Thus, it would be naive to expect that logical manipulation of symbolic descriptions as an adequate tool. Furthermore, there is substantial psychophysical evidence [60,121,184,154,296] that while the beginner learns through rules, the expert discards such rules. Instead he discriminates thousands of patterns in his domains of expertise acquired through experience, and how to respond to them.

It is rapidly becoming evident that many of the unstructured problems characterized above, can be best solved not with traditional AI techniques, but by "analogy", "subsymbolic" [133,251,263] or pattern matching techniques [276,285]. While AI attempts to do this, *neural networks*, a biologically inspired, computational and information processing paradigm, provides us with an inherently better, but not a unique tool. Our focus is on examining the capabilities of neural networks for learning, which is central to the "deeper" question of its feasibility to artificial intelligence. In the remainder of this dissertation, we concentrate on developing a repertoire of computational formalisms that can provide us an enabling basis for solving a significantly complex problem, that has been addressed for last several decades, namely, *functional synthesis, i.e., learning nonlinear mappings to abstract functional invariants, statistical invariants logical invariants and spatial invariants from representative examples.* However, before we formally introduce neural networks, and embark into the technical core of this thesis, we present arguments juxtaposing the suitability of neural networks versus Formal AI, to solving problems in computational learning. A detailed discussion on the latter subject may be found in Gulati, Barhen and Iyengar [111].

1.1. Neural Networks for AI Modeling

Over the last three decades, formal AI and neural network researchers have extensively examined the problems in pattern recognition, adaptive machine learning, perception and sensory-motor control, providing an insightful assessment of what is difficult and what is easy [121,144]. Although both disciplines have similar goals, there is not much overlap between their projected capabilities. The basis of both paradigms may be traced back to hypotheses of Weiner [286] and Leibniz [182], wherein they identified humans as goal-seeking, complicated machine composed of an "intelligent" brain and highly redundant motor systems. It is able to detect errors, change course, and adapt its behavior so that achievements of goals is more efficient. However, subsequent development of intelligent systems has pursued two distinct schools of thought; *symbolic* and *neurobiological, subsymbolic or connectionist.* AI researchers concentrated on what the brain did irrespective of how it was accomplished biologically, while the latter focussed on how the brain performed.

Rooted in the *"rationalist, reductionist tradition in philosophy"* [251], AI assumes that there is a fundamental underlying formal representation and logic that mirrors all primitive objects, actions and relations that make up the world, and that has the necessary and sufficient means for general intelligent action. As its most forceful proponents, Newell and Simon [221], hypothesized that once such a representation were to become available, the operations of human cybernetic machinery could be fully automated and described in terms of mathematical theorems and formal logic. All knowledge could be formulated into rules, and behavioral aspects of human reasoning and perception could be emulated by following rules or manipulating symbols, without regard to the varying interpretations of symbols. Further, intelligent behavior arises from amalgamation of symbols in patterns that were not anticipated when the rules were written.

Expert systems are product of such a line of investigation. However, as discussed by Reeke and Edelman in [93] over the years AI researchers have unsuccessfully struggled against fundamental systems engineering issues summarized as :

(a) **Coding Problem :** (Leibniz [182]) finding the suitable universal symbol system , i.e., the ultimate simples in terms of which all complex can be understood;

(b) **Procedure Problem :** specifying in advance all actions that must be taken for all possible combinations of input;

(c) **Category Problem :** (Minsky [214,215]) specifying a sufficient set of rules to define all possible categories and phenomena that the system might have to account for;

(d) **Homunculus Problem :** (Minsky [214]) This pins the fundamental problem of AI to the old puzzle of "infinite regress" in a universal symbol system. For example, when you look at an object, say a computer, how is the image of the computer registered in the brain ? All explanations hitherto proposed by AI pin down this process on some "intelligent device" inside the brain that shall be in charge of doing the registering. But then the same problem has to be faced again in order to explain how the "device" does the registering, and so on ad infinitum.

(e) **Nonmonotonic Reasoning Problem :** designing rules that can function as retractable hypothesis, to mitigate the problems that arise when rules get executed without context-consistency checks

(f) **Developmental Problem :** devising mechanisms that can enable programmed systems exist: self-learn, self-organize their structure and function and self-replicate without explicit external manipulation, akin to adaptive biological systems;

Since formal AI has not been able to surmount the above problems, using logical reasoning alone, Hofstader [133], Smolensky in [251] and others [306] have suggested recourse to alternate scientific paradigms - neural networks. In a radical philosophical and paradigmatic departure from AI, the neural network community argues that logical reasoning is not the foundation on which cognition is based, but instead, an emergent behavior that results from observing a sufficient number of regularities in the world [251]. Its theoretical underpinnings lie in biological detail and rigorous mathematical disciplines such as theory of dynamical systems [47,100,127-128,237], statistical physics [229], etc., in an attempt to discover and validate principles that make intelligence possible, by observing existing intelligent systems, i.e., the brain. They hold the view that cognitive machinery is built from many simple nonlinear interacting elements - neural networks that store knowledge in their internal states and self-organize in response to their environments. Intelligent behavior, then manifests from collective interactions of these units.

While formal AI or the Symbolic community also treated human brain as an hierarchical system of components that obey laws of physics and chemistry, and could be described as solutions to mathematical equations relating computable functions over the inputs and outputs of neurons, it assumed that given a sufficient amount of information, i.e., computing power, neuronal dynamics, one could compute a person's next state. However, it ignored the framework of interpretation, "context-sensitivity", within which the humans process information, make commitments and assume responsibility. Instead, its primary focus became to design rule-systems that processed symbols without regard to their meanings. Thus, it completely ignored the considerable amount of subsymbolic or subconscious processing precedes our conscious decision making, and subsequently leads to the filtration out of infinity of situations so that the appropriate rule may be used. In sharp contrast, rather than creating logical problem-solving procedures,

neural network researchers use only an informal understanding of the desired behavior to construct computational architectures that can address the problem, thereby eliminating the fundamental AI limitation, i.e., context sensitivity in [93].

In summary, unlike AI, there is no recognition, recall and reminding; neural networks focus on association, units and patterns of activation. Thus, rather than focusing on symbols, symbolic manipulation, or formal logic procedures, neurocomputation primarily entails recognizing statistically emergent patterns and processing alternatives obtained by relaxing various features that characterize the situation [208,222]. Therein lies the amenability and performance potential of neural networks to the development and application of human-made systems that can emulate the neuronal information processing operations, e.g., real-time high performance pattern-recognition [12,95-98,126,166-170], knowledge-processing for inexact knowledge domains [227], and precise sensory-motor control of robotic effectors [57,81,199,235], that computers and AI machines are not suited for. They are *ideally suited for tasks where a holistic overview is required [298], i.e., abstract relatively small amounts of significant information from large data streams* such as in speech recognition [93] or language identification [296]. On the other hand, digital computers and AI are ideal for algorithmic, symbolic, logical and high precision numeric operations that neural networks are not suited for. The two fields complement each other in that they approach the same problems from different perspectives.

Having motivated the applicational and paradigmatic potential of neural networks, we now briefly summarize their evolutionary history, followed by formal characterization of their properties.

1.2. Background: Artificial Neural Networks

Though the notion of imitating human neuronal processing system, cognitive capabilities and biological phenomena to model computation for AI research was introduced a couple of decades back, it is only recently that they are gaining recognition as a viable alternative to the traditional Von Neumann architectures. Recent advances in our understanding of anatomical and functional architecture [17,19,49,61,202] chemical composition [179], electrical and organizational processes [120,137-138,190-192,196] occurring in the brain and nervous system along with the advances in hardware technology and capability are leading to physical and electro-optical realizations [121] of randomly organized interconnect, networks with computationally useful collective properties such as time sequence retention, error correction and noise elimination, recognition and generalization.

Development of detailed models of neural networks began with the work of McCulloch and Pitts [206,17], wherein using logical elements they demonstrated that synchronous neural nets could perform all quantifiable processes, e.g., arithmetic, classification application of logical rules. Hebb [120] demonstrates that repeated activation of a group of neurons by another through a particular synapse leads it to synchronously activate groups of neurons to which it is weakly connected, thereby organizing into strongly connected assemblies. Neumann [97] injects the notion of redundancy in neurocomputing by constructing networks which activated many neurons to do the job of one. Winograd and Cowan [297] extended his work to introduce the notion of distributed representation wherein each neuron partially represented many bits. The field was put on a firm mathematical basis by Rosenblatt [248], who conjectured that intelligent behavior based on a physical representation was likely to be hard to formalize. As per his arguments, it was easier to axiomatize a physical system and then investigate the system analytically to determine its behavior, than to axiomatize the behavior

and then design a physical system by techniques of logical synthesis. He engineered his ideas, by attempting to automate procedures by which a feedforward network of McCulloch and Pitts neurons [206], named Perceptrons by him, could learn to automate the procedure by which a network of neurons learned to discriminate patterns and respond appropriately. A detailed study of perceptrons, led Minsky and Papert [215] to strong criticism of the field. Thereafter, neural network receded into a long slump. However, as observed by Ladd [175], they were misleading in interpreting/suggesting that this class was at the heart of connectionism, and their [215] analysis was not valid for systems that were more complex, including multilayered perceptrons and neurons with feedback.

The resurgence of the field is due to the more recent theoretical contributions by Kohonen [166-170], Grossberg [94-99], Amari [11-14], Fukushima [88], Carpentar [62-63], Hopfield [134-135], etc. Hopfield's illuminating contributions have extended the applicability of neuromorphic techniques to the solution of combinatorially complex optimization problems [136]. In the areas of VLSI and opto-electronic implementations, major achievements have resulted from the efforts of Mead [209], Psaltis and Farhat [242], Hecht-Nielson [121], and others.

As shown in Fig. 1.1.1, *Artificial Neural Systems* may be characterized as distributed computational system comprising of a large number of processing units each of which has selected characteristics of biological neurons connected to each other in a directed graph of varying configuration [151]. They have been defined [170] as

> *"massively parallel, adaptive dynamical systems modeled on the general features of biological networks, that can carry out useful information processing by means of their state response to initial or continuous input. Such neural systems interact with the objects of the real world and its statistical characteristics in the same way as biological systems do."*

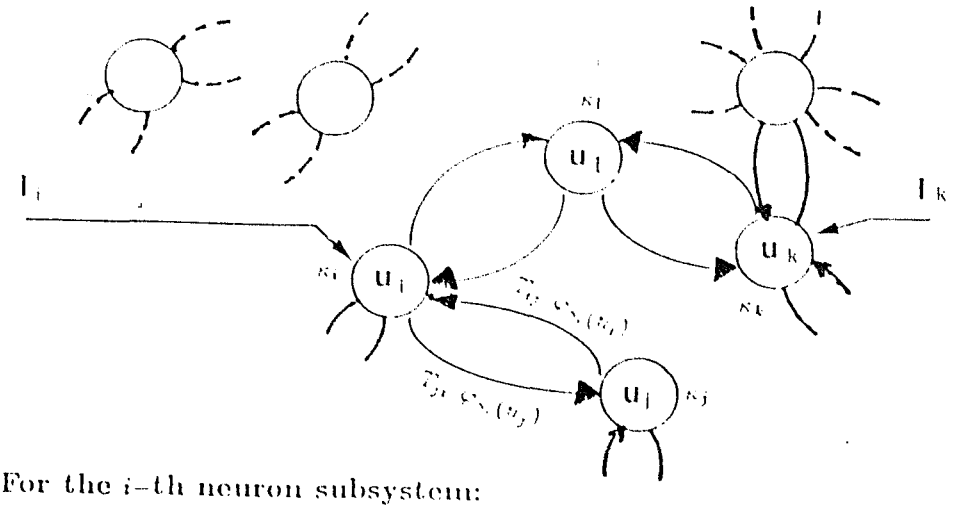


Figure 1.1.1. Artificial Neural System.

Grossberg [98], Hartley and Szu [264] have classified neural networks to be any system that satisfies (a) nonlinearity, (b) nonlocal, i.e., exhibit long-range interactions across a network of locations, (c) nonstationary, i.e., interactions are reverabative or iterative, and (d) has nonconvex "energy-like" function.

In contrast to the existing notions on applicative and symbolic computing, the potential advantages of neuronal processing arise as a result of their ability to perform massively parallel, asynchronous and distributed information processing. Neurons with simple properties and interacting according to relatively simple rules can accomplish collectively complex functions. This is based on their

ability to provide a collectively-computed solution to a problem on the basis of analog input information resulting from a high degree of random interconnectivity, storage and simplicity of individual operations [41,141]. Neural network modeling then is a discipline which attempts to "understand brain- like computational systems" [24,263] and has been variously termed as "computational neuroscience" [14], "parallel distributed processing" [251], "connectionism" [222], etc.

The bulk of neural network models can be classified into two categories; those that are intended as computational models of biological nervous systems or *neurobiological models* [61,137,138,199], and those that are intended as biologically-inspired models of computational devices with technological applications, also referred to as *Artificial Neural Systems (ANS)* [6,9,62,98,225]. Although our primary emphasis is on ANS, we will be highlighting the influence of neurobiology on the formulation of artificial neural models, and the resulting computational implications. To get a sense of the required number and interconnectivity of neuronal circuitry for intelligent behavior, we begin by examining biological neural networks. Most existing neural network models are based on idealizations of the biological neuron and the synaptic conduction mechanisms, shown in figure 1.1.2(a) and 1.1.2(b), respectively. As shown in Fig 1.1.2(a), each neuron is characterized by a cell body or the cyton and thin branching extensions called dendrites and axons that are specialized for inter-neuron transmission. The dendrite is a passive receiving and transmitting agent, the axon electrochemically charged, highly active brain cell entity. The dendrites receive inputs from other neurons and the axon provides outputs to other neurons. The neuron itself is imbedded in an aqueous solution of ions, and its selective permeability to these ions establishes a potential gradient responsible for transmitting information. The electrochemical input signals or the neurotransmitter is funnelled to the neuron from other neurons, to which it is connected is through sites on the their

surface, called synapses (see Fig. 1.1.2(b)). The input signals are combined in various ways, triggering the generation of an output signal by a special region near the cell body. However, the neurobiological phenomenon that is of particular interest, is the changing chemistry of synapse as information flows from one neuron to another. The synapse instantaneously decides when the information is inessential and should not be resupplied. The weight of the individual charge is regarded as the determining factor. On the transmitting or pre-synaptic side of the synapse, triggering of the synaptic pulse releases a neurotransmitter, that diffuses across a gap to the receiving side of the synapse. On the post-synaptic or receiving side, the neurotransmitter binds itself to receptor molecules, thereby affecting the ionic channels and changing the electrochemical potential. The magnitude of this change is determined by many factors local to the synapse, e.g., amount of neurotransmitter released, number of post-synaptic receptors, etc. *Therefore, neurocomputation, biological self-organization, adaptive learning and other mental phenomena are largely manifested in changing the effectiveness or "strength" of the synapse and their topology [138].* Additional details on the biological neuron, membrane polarization chemistry and synaptic modification may be found in [9,100].

The above phenomenological insights at the neurobiological level, have led to the mathematical formulation of *simulated neurons*, i.e., the basic building block of neural network models. A functional model for typical simulated neuron is shown in Fig. 1.1.2(c). Four useful areas may be abstracted. The first is the synapse where signals are passed from one neuron to another, and the amount of signal is regulated, i.e., gated or weighted by the strength of the synaptic interconnection. In the activated neuron region, denoted as *summer*, synaptic signals containing excitatory and inhibitory information are combined to affect the tendency of a cell to fire or stay put. The threshold detector determines if the

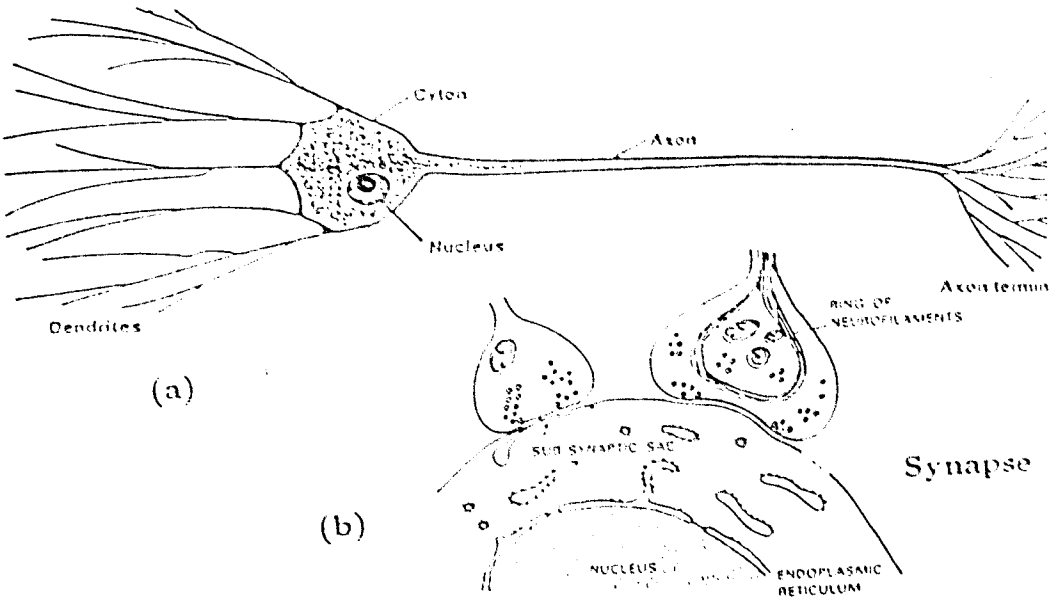


Figure 1.1.2. (a) Biological Neuron; (b) Morphology of neuron-to-neuron connection.

neuron is actually going to fire or not, while axonal paths conduct the output activation energy to other synapses to which the neuron is connected. Useful information properties properties such as generalization [125], classification [62], association [12,91], error correction, time sequence retention [192], etc., emerge as collective properties of systems comprised of aggregations of large number of such simple units. When viewed individually, the dynamics of each neuron bears little semblance to task being performed.

As discussed in the preceding paragraph, of particular computational and

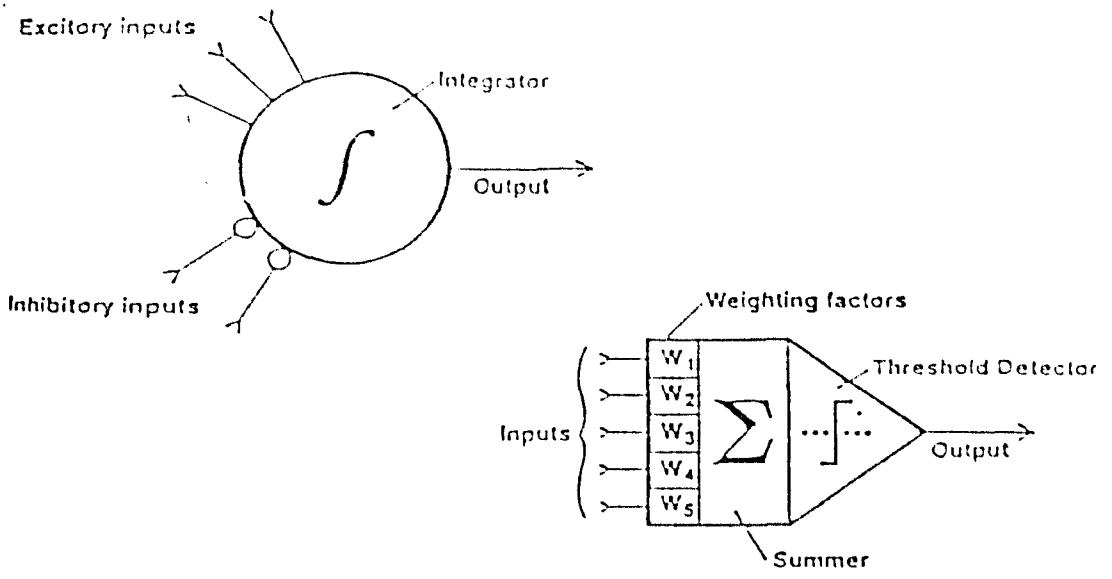


Figure 1.1.3. Mathematical idealization to biological neurons.

modeling interest, are the mathematical notions of synapse and synaptic modification, mechanisms by which such units can be connected together to compute, and the rules whereby such interconnected systems could be made to learn.

1.3. Computational Learning

The paradigmatic strength of neural networks for potential applications, which require solving intractable computational problems or adaptive modeling, arises from their spontaneous emergent ability to achieve *functional synthesis*, and thereby learn nonlinear mappings [36,32,111], and abstract spatial [62,63], functional [180] or temporal [177,178] invariances of these mappings. Thus, relationships between multiple continuous-valued, statistically-related inputs and outputs can be established, based on a presentation of a large number of rep-

resentative examples. Once the underlying invariances have been learned and encoded in the topology and strengths of the synaptic interconnections [36,305], the neural network can generalize to solve arbitrary problem instances. Since the topological mappings for problem-solving are acquired from real-world examples, network functionality is not limited by assumptions regarding parametric or environmental uncertainty, that invariably limit model-based computational strategies [13]. In order to place our subsequent discussions in context with the overall discipline of computational learning (including heuristic, algorithmic and connectionist), in Table 1.4.1. we provide a taxonomy of learning formalisms.

Neural Learning has been defined as the process of adaptively evolving the internal parameters e.g., connection weights, network topology, etc. in response to stimuli being presented at the input and possibly the output buffer. As adaptive dynamical systems (refer Section 1.6.1), neural networks emphasize relaxation, and not heuristic search as the basis of automatic learning. Further, learning can be by programming the net, i.e., setting the weights or by self-organization. These are known as *hard* and *soft* learning respectively. In this thesis we focus on soft learning, which in turn may be either deterministic, stochastic, supervised or unsupervised. Learning in neural networks is said to *supervised* [250-252,269,292], e.g., when the desired response is from a knowledgeable teacher, and the retrieval involves one or more of a set of stimuli patterns that have been repeatedly shown to the system during the training phase. The networks observe the presented inputs, detect the statistical regularities embedded within it and learn to exploit these regularities to draw conclusions when presented with a portion or a distorted version of the original pattern. When a portion of the original pattern is used as a retrieval cue, the learned process is denoted to be *auto-associative* [166]. When the desired input is different from the input then learning is referred to as *hetero-associative* [188].

Paradigm	Categories	Algorithm Description	Nature	Systems
Empirical Learning	Rule-Based;	Real-to-model world mapping	deductive	LAMP, PROMOTY
	Learning via query;	Ordering of questions directs learning	inductive	MARVIN
	Concept Learning	Search in hypothesis space	inductive	OTIS
Case-Based Learning	Adapting case-bases	search through case networks of analogy-explanatory-emulation skills	deductive	TA
Statistical Learning	Bayesian tech., Decision trees	find probable no. of classes, probabilistic descriptions & probability of belongingness	inductive	AutoClass
Genetic Learning	Population genetics	Trial solutions (populations) operated in cycles (generations) by survival of fittest selection followed by genetic recombination (crossover and mutation operators)	inductive	CFS
Explanation-Based Learning	reasoning about operationality (merit of a learning result)	obtain general concept def. on some property that holds for a given training instance	deductive	ROE, MetaLEX
Connectionist (Neural) Learning	Supervised; Unsupervised; Reinforcement;	capture the functional, spatial or temporal concept in the internal synaptic connections or topology of the network	relaxation in an energy landscape	EBP, CL, ART, SID
Distribution Free Learning	Formal Concept Learning	construct a minimal set of maximally general descriptions	inductive, deductive	PAC

Table 1.4.1. Survey of Computational Learning Paradigms

When no desired output is shown the learning is *unsupervised* [306]. It proceeds with a knowledgeable teacher. An intermediate kind of learning is *reinforcement learning* [42,43] where a teacher just indicates whether the response to an input is good or bad, how far and in what direction the current output differs from the desired output, the the network is rewarded or penalized depending on the action it takes in response to each presented stimulus. The network configures itself so as to maximize the reward that it receives. Along with the architecture, learning rules forms the basis of categorizing different neural network models. A detailed taxonomy of different types of learning rules can be found in Lipmann [166].

Further the neural learning rules could take the following forms: *Correlational Learning* wherein parameter changes occur on the basis of local or global information available to a single neuron. A good example is Hebbian learning rule [134], wherein the connection weights are adjusted according to a correlation between the states of the two interconnected neurons. If two neurons were both active during some successful behavior, the connection would be strengthened to express the positive correlation between them. On the other hand, in *Error-corrected Learning*, the rules work by comparing the response to a given input pattern with the desired response and then modifying the weights in the direction of the decreasing error, e.g., perceptron learning rule [248,251], Widrow-Hoff [291], back-propagation in [150,274,275,250-253]. Another model, *Reinforcement Learning* does not require a measure of the desired responses, either at the level of a single neuron or at the level of a network. Only a measure of the adequacy of the emitted response suffices. This reinforcement measure is used to guide random search process to maximize reward. Another category, *Stochastic Learning*, the network neurons influencing each other through stochastic relaxation, e.g., Boltzmann Machine [2].

Two key elements that characterizes the computational power of neural learning formalisms are nature of states of individual neurons and the temporal nature of synaptic updating. The states of individual neurons may be either *Discrete* or *Continuous*. They may be finite, infinite but countable, or uncountable - forming a continuum [264]. It has been shown that networks with a finite number of states is computationally equivalent to a finite state machine (FSM) if the number of neurons is finite or to a Turing machine (TM) if the number is infinite. If the neuron has a continuum of stable states then it is equivalent to a TM. Further, the nature of time variable in neural computation may be either *discrete*, i.e., dynamics is modeled by difference approximations to differential equations or it may be *Continuous*. It has been shown that continuous time networks can resolve temporal behavior [264], which is transparent in networks operating in discrete time. In all the respects, the two classes are computationally equivalent.

1.4. Statement of the Problem

The research problems addressed in this dissertation can be simply stated as follows:

[1] *Derive a new class of fundamental, computational neural learning algorithms that can robustly acquire embedded nonlinear functional, spatial, logical, statistical or behavioral invariants from representative examplars, in hard real-time. Ensure the scalability of proposed learning algorithms to the number of training samples, network dimensions and topography, and the cardinality of input-output attributes. Furthermore, ensure that the paradigmatic complexity of learning formalisms is independent of the complexity of learning problems to which it is applied.*

[2] *Develop efficient mechanisms for explicitly incorporating, a multiplicity, of both application-specific and neural network design constraints*

into the neural learning framework. Furthermore, devise computation structures for avoiding extensive training/retraining costs each time the network is confronted with a novel problem situation, that poses constraints different from the ones used during the training phase.

[2] *Develop a formal framework for simulating / implementing neural network models on massively parallel SIMD / MIMD multiprocessors and asynchronous neural hardware - VLSI, optical and opto-electronic. Devise formal algorithms for determining the neurodynamical constants to ensure scalability and robustness of proposed theoretical models during implementation. Determine conditions that could potentially drive neural network algorithms into dynamical instabilities and chaos, in concurrently asynchronous neurodynamical regimes. Devise a framework for formally characterizing chaos in neural networks.*

[4] *Leverage the above repertoire of tools, to formally conceptualize a computational and architectural framework for a real-life problem that is beyond the state-of-the-art capability in the chosen domain.*

The central theme of this thesis has, in fact been primarily motivated by the latter objective, i.e., the current lack of computationally enabling tools for engineering "intelligent autonomous systems" - autonomous robots, rovers, diagnostic systems etc. Robotics in general, and robust, task-directed, autonomous manipulation for space-based robots in particular, represent our specific application domains of interest. In the initial phases of algorithm development, our robotic implementations are limited to inverse kinematics of redundant manipulators. In a latter chapter, we however elucidate its straightforward applicability to problems in manipulator dynamics, control and sensorimotor coordination. We extrapolate the applicational conceptualization to include perceptual manip-

ulation [282] in dynamically varying, poorly modeled and unpredictable environments.

1.5. Organization of The Dissertation

This work is organized into eight chapters. The first chapter is devoted to presenting the framework and an applications context within which this work is relevant. To this end, we introduced some basic concepts and mathematical tools. Figure 1.5.1 provides an overview of the organization of the thesis. This section presents a brief overview on the contents of the remaining chapters.

Chapter Two motivates a formal framework for deriving supervised learning algorithms for dynamical neural networks. The methodology is generic to networks both with and without feedback. It is based on a recent breakthrough in nonlinear dynamical systems theory - the notion of terminal attractors (introduced by Zak [302,303]). We exploit the concept of terminal attractors to define the Singular Interaction Dynamics (SID) model for learning time-independent data from examples. In a departure from prior neuromorphic algorithms this methodology provides mechanisms for incorporating and in-training "skew" to handle network as well as design constraints. The notion of "virtual attractors" is introduced to guarantee overall network stability. Two versions of the algorithm are derived, *Algorithm* SID_1 and *Algorithm* SID_2 for large dimensional networks. We benchmark the efficiency of *Algorithm* SID_1 system on a multidimensional signal reconstruction problem. *Algorithm* SID_2 is simulated on the 3-DOF and 7-DOF redundant manipulators to validate our theoretical framework and illustrate its computational efficiency.

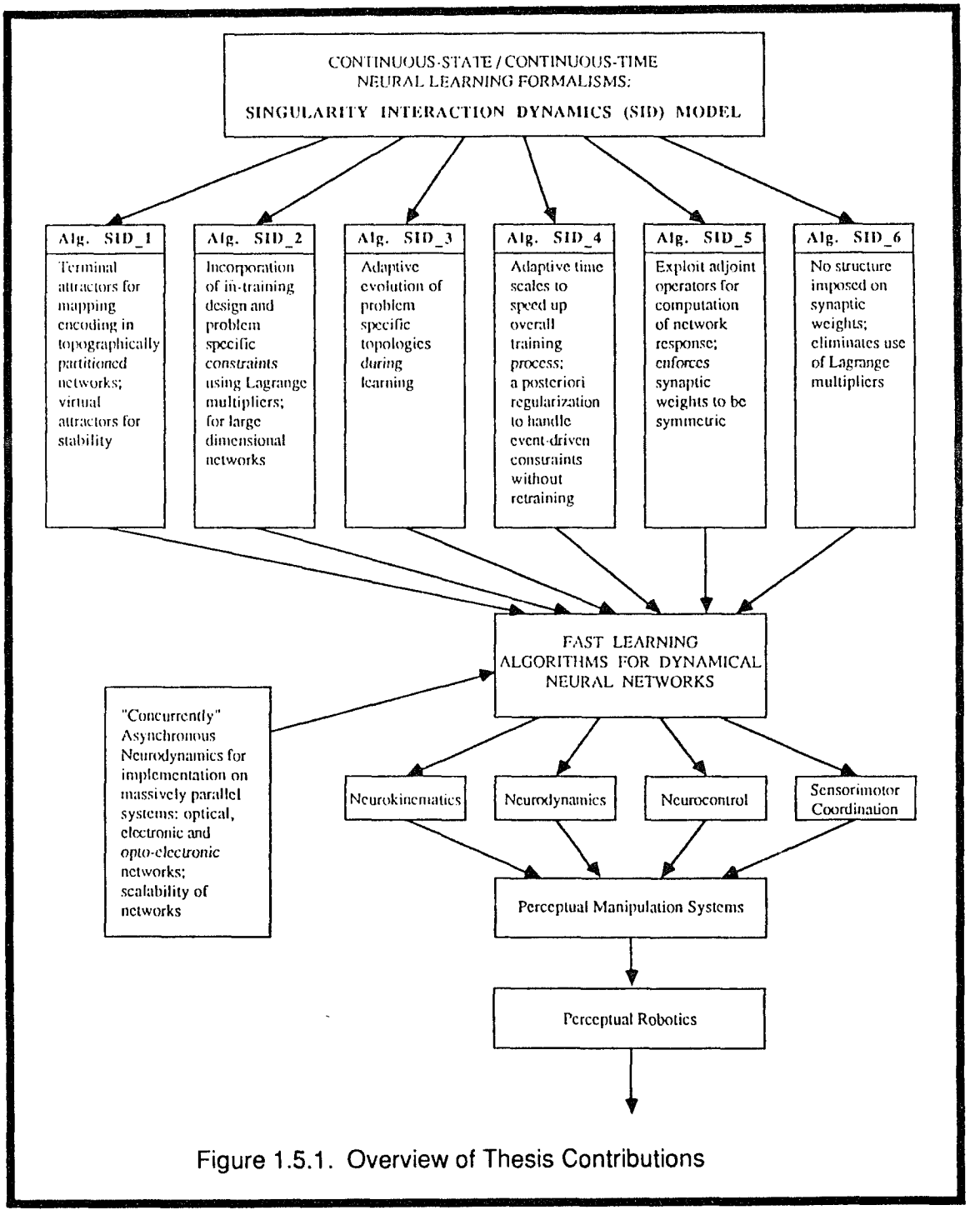


Figure 1.5.1. Overview of Thesis Contributions

Chapter Three extends upon the results in the preceding section to provide a novel manifestation to computational learning based on phenomenology of nonlinear neural networks. We present the *Algorithm* SID_3 neural network model that allows adaptive evolution of network topology in addition to evolution of synaptic strengths. The former objective is achieved by taking recourse to Gauss's Least Constraint Principle [36,107,305] in mechanics. We execute the algorithm on a 3-DOF redundant manipulator. Further, we examine a fundamental limitation in neural learning algorithms - training and retraining costs and the versatility of neural network models. We draw inspiration from mathematical physics to introduce renormalization concept for task-directed aposteriori regularization. In the previous chapters, we had largely exploited the notion of terminal attractors to obtain computational speedup per dynamical iteration. In this chapter we introduce mechanisms for speeding the overall training process. We exploit the notion of adaptive time scales in terminal attractor formalism (introduced by Zak [303]) for supervised learning. These constructs are used to extend the neural learning formalisms - *Algorithm* SID_4. We benchmark the algorithm using problems from the signal processing domain. Furthermore, we explore another important concern in the design and implementation of learning algorithms, namely the implications of selecting particular kinds of numerical tools used in neural network simulations.

Chapter Four couples adjoint sensitivity theory with neural networks,and learning in particular. We provide a brief introduction to the notion of forward and adjoint operators. Heretofore, we had resorted to heuristics in the section of our derivations, dealing with dynamic propagation of backward error. In this chapter, we eliminate all heuristic overtones, to obtain a formal framework for global computation of sensitivities. These concepts are used to formally derive another version of neural learning algorithm - *Algorithm* SID_5.

Chapter Five exploits the concept of *adjoint operators* to enable computation of changes in the network's response due to perturbations in all system parameters, using the solution of a single set of appropriately constructed linear equations. The lower bound on speedup per learning iteration over conventional methods for calculating the neuromorphic energy gradient is $O(N^2)$, where N is the number of neurons in the network. The learning objective is reformulated to derive *Algorithm* SID_6 for hyperfast learning in dynamical neural networks. We demonstrate the computational efficacy of our approach by benchmarking simulations on complex signal processing problems with current state-of-the-art neuromorphic models.

Chapter Six addresses a fundamental issue which directly impacts the scalability of current theoretical neural network models to applicative embodiments, in both software as well as hardware - inherent and unavoidable concurrent asynchronicity of emerging fine-grained computational ensembles and the consequent chaotic manifestations in the absence of proper conditioning. In this chapter we introduce a mathematical framework for systematically reconditioning additive-type models and derive a neuro-operator, based on the chaotic relaxation paradigm, whose resulting dynamics is neither "concurrently" synchronous nor "sequentially" asynchronous. Necessary and sufficient conditions guaranteeing concurrent asynchronous convergence are established in terms of contracting operators. Lyapunov exponents are also computed to characterize the network dynamics and to ensure that throughput-limiting "emergent computational chaos" behavior in models reconditioned with concurrently asynchronous algorithms was eliminated. Implementation results are provided for massively parallel networks ranging from few 100 synapses to over 100 million interconnects.

In Chapter Seven, we shift gears to introduce a robotic manipulation problem that is beyond the state-of-the-art "model-directed" robotics. We exploit the aforementioned repertoire of neural learning algorithms developed in this work to provide an enabling element for such systems. We motivate the need to conceptualize such robotic systems within a "perceptual framework" rather than existing model-based formalisms. We propose a new Task-based Perceptual Manipulation Architecture, wherein the key control element comprises of metric-driven neural networks. A technical critique of the proposed architecture is presented vis-a-vis existing robot architectures. To provide a context to our computationally enabling learning framework, we propose a novel systems concept for applications to space robotics. We execute the architecture on a functional manifestation of this application.

Chapter Eight provides a summary of contributions, concluding remarks and outlines directions along which this work can be extended.

1.6. Mathematical Constructs

In solving the aforementioned problems, we have made a deliberate attempt to base our arguments and formalisms on firm mathematical foundations, rather than relying on qualitative arguments or brute-force techniques. We draw from a variety of rigorous mathematical disciplines, such as theory of dynamical systems [47,100,127-128,237], existence theorems [226], renormalization theory and critical phenomena [295], etc. These mathematical constructs have provided us extremely attractive computational tools for deriving efficient algorithms presented herein. Furthermore, even though the mathematics used in this dissertation may be unfamiliar to some readers, the results we obtain are mostly using constructive algorithmic techniques. Also, the chapters are self-contained and structured to be read in isolation. In the remaining section of this chapter, we provide a

terse introduction to some of the broad constructs that we will be employing in the thesis.

1.6.1. Dynamical Systems Theory

Dynamical systems are those that change their state with time t. A dynamical system of *order* n is defined by two properties [47]:

[1] The state of the system is represented by n real variables, x_1, $x_2, \cdots x_n$, or one real vector $\bar{r}$ of dimension n, which may be considered as coordinates of an abstract $n-$ dimensional space named the *phase space.*

[2] The motion of the system is represented by a vector function $\bar{r}(t)$ of time satisfying a first-order vector differential equation of motion, $d\bar{r}/dt = \dot{\bar{r}} = v(\bar{r}, t)$ where v is a given sufficiently well-behaved velocity function of $\bar{r}$ and t, whose value for a particular $\bar{r}$ and t is the *phase velocity.*

If the system is not subject to any external influences that depend on the time, the system is said to be *autonomous.* The velocity field $v(\bar{r})$ of an autonomous order-n system can be expressed as the negative gradient of a potential $U(\bar{r})$. Thus, $v(\bar{r}) = -dU/d\bar{r}$ where, for some constant U_o, $U(\bar{r}) = U_o - \int_0^{\bar{r}} dx' v(x')$. At each zero x_k of the velocity field $v(x)$, $v(x_k) = 0$, so that a system initially at x_k remains there for all time. The points x_k represent states of the *equilibrium*: they are named *fixed points.* When the velocity field has only simple zeros, the fixed points are *stable* around x_k if $v(x)$ is a decreasing function of x, so that neighboring states approach x_k, and they are *unstable* around x_k if $v(x)$ is an increasing function of x so that neighboring states leave x_k. Neural networks as dynamical systems are governed by an update equation on the state vector that is nonlinear and iterative. If the error norm of the sequence of state vectors become smaller and smaller tending to a fixed point, x_k say at zero, for all initializing points x_o less than a unit distance from the fixed point, then the

domain x_o is said to be the basin of attraction of the attractor at zero. If the numbers in the sequence increase gradually tending to infinity, then infinity is the attractor for all points $x_o > 1$.

Limit points (fixed non-time varying), limit cycle (periodic,time varying), strange attractor (sensitive dependence on initial conditions, time varying) and strings (limit points with filamentary basin of attraction, correlated vectors) could all constitute attractors in dynamical systems [47]. In the nonlinear dynamical systems formulation to neural networks, all information is stored at fixed points, fixed limit cycles in state space that act as attractors with prescribed basins of attraction, such that initial configurations of neurons in some neighborhood or basin of attraction of that memory state will be attracted to it. As we demonstrate in the sequel, in the dynamical systems formulation, computing with neural networks thus implies designing networks with prescribed phase-space behavior.

1.6.2. Nonlinear Sensitivity Theory

The general objective of sensitivity analysis is to quantitatively derive sensitivities of the system's response (i.e., system performance measures of interest) due to perturbations and uncertainties in the input parameters. The simplest sensitivity procedure is what may be termed the "brute-force" method, which evaluates a parametric sensitivity by repeating the computation with a perturbed input parameter while holding the others fixed. The computational costs involved in this approach are in general prohibitive for most real-life applications [273]. Two alternative deterministic (rather than statistical) formalisms, labeled the "Forward Sensitivity Method" (FSM) and "Adjoint Sensitivity Method" (ASM) exist in order to evaluate the sensitivity of the response to variations in the system parameters. Originally proposed by Cacuci [59,60] for engineering applications

wherein the parameters can neither be accurately estimated nor known statistically, the concept has over the years been collectively developed by Barhen, Oblow, Alsmiller, Toomarian, etc [10]. It has previously been extensively applied in the fields of energy modeling [27] and nuclear reactor thermal hydraulics [273] at the Oak Ridge National Laboratory. The forward sensitivity formalism is formulated in normed linear spaces, and the existence of the Gateaux differentials of the operators appearing in the problem is shown to be both necessary and sufficient for its validity. It provides the exact elementary sensitivity coefficients (SC) (i.e., partial derivatives of all model state functions with respect to an input parameter), of all dependent variables with respect to a single input parameter [155,311]. On the other hand, adjoint sensitivity theory provides the exact sensitivity coefficients of a single response function (primary dependent variable) with respect to all input parameters.

ASM is computationally more economical for problems involving many parameter alterations or a large data base and comparatively few functional responses. On the other hand, FSM is efficient for problems involving a few parameters only, as it requires that a different set of equations be solved for each parameter to obtain the complete elementary SC matrix, as the actual form of the differentiated equation depends on the particular parameter under consideration. Application of FSM and ASM to neural learning is discussed in further detail in Chapters Four and Five.

1.6.3. Renormalization Theory

The "renormalization-group" approach in physics was introduced by Wilson [295] for dealing with problems involving many length scales. Its purpose is to eliminate an energy scale, length scale or any other term that could produce an effective interaction with arbitrary many coupling constants. The strategy is to

tackle the problem in steps, one step for every length scale. For example in the case of critical phenomena (e.g., liquid gas transitions, magnetic transitions, alloy transitions, etc.), the problem is to technically carry out the statistical averages over thermal fluctuations on all scale sizes. The renormalization approach is to essentially integrate the fluctuations in sequence, starting with fluctuations on an atomic scale and then moving to successively larger scales until fluctuations on all scales have been averaged out. The concept works as follows: As the fluctuations on each length scale are integrated out, a new free energy functional $F_{L+\delta L}$ is generated from the previous functional. This process is repeated many times. If F_L and $F_{L+\delta L}$ are expressed in dimensionless form, then one finds that the transformation leading from F_L to $F_{L+\delta L}$ is repeated in identical form many times. This transformation group is called the "renormalization group". As L becomes large, the free energy F_L approaches a fixed point of the transformation, and thereby becomes independent of details of the system at the atomic level. This explains the universality of critical behavior for different kinds of system at the atomic level. The same "fixed point" interaction describes all these systems.

In this dissertation, the above methodology is abstracted and applied during *a posteriori* regularization (constraint satisfaction) to prioritize and modulate network response in the presence of a multiplicity of external constraints during run-time. Along the lines discussed here, the constraint satisfaction procedure as discussed in Chapter Three entails, inclusion of the gradient of a constraint in the operational network dynamics normalized in a manner such that when the constraint is satisfied the term vanishes.

1.6.4. Nonconstructive Existence Theorems

In Chapter Six we derive our results using a sampling of various existence and uniqueness theorems for minimization of convex functionals on $\Re^n$, which

are proved by the nonconstructive means, e.g., *gradient operators* (i.e., operators F which are the derivatives of a real-valued functional g. These are used to study questions of existence and uniqueness of solutions of $Fx = 0$ by finding the minimizers of g.), *contraction operators*, (i.e., when F is not a gradient operator). The latter are used extensively in our derivations of asynchronous neuro-operator in Chapter Six. Some of the key results that we use from nonlinear analysis are as follows:

A mapping $G : D \subset \Re^n \rightarrow \Re^n$ is nonexpansive on a set $D_o \subset D$ if $|Gx - Gy| \leq |x - y|$, $\forall x, y, \in D_o$ and strictly nonexpansive on D_o if strict inequality holds in (1) whenever $x \neq y$. Also, any nonexpansive mapping on D_o is Lipschitz-continuous on D_o. A linear operator $A \in \Re^n$ is nonexpansive iff $|A \leq 1$. In our analysis we consider neural systems of the form $x - Gx = 0$. Any solution x^* of this equation, that is any point x^* in the domain of G for which $x^* = Gx^*$, is a fixed point of G. Thus the sequence, $X^{k+1} = Gx^k$, $k = 0, 1, \cdots$ converges to fixed point of G. Further, if $G : D \subset R^n \rightarrow R^n$ is strictly nonexpansive on D_o and $x^*, y^* \in D_o$ are any two fixed points, then

$$|x^* - y^*| = |Gx^* - Gy^*| < |x^* - y^*|$$

is a contradiction, which implies that $x^* = y^*$; that is strictly nonexpansive mappings can have at most one fixed point. Further a mapping $G : D \subset \Re^n \rightarrow \Re^n$ is *contractive* if there is an $\alpha < 1$ such that $|Gx - Gy| \leq \alpha|x - y|$ for all $x, y \in D_o$. Linear operators are contractive on all of $\Re^n$ iff $|A| < 1$. Also note that contractive mappings are norm dependent. In the case that $D_o = D = \Re^n$, G is a global contraction on all of $\Re^n$. As discussed in Chapter Six, the existence of unique fixed points is then given by the Contraction Mapping Theorem [226]. The contraction mapping property plays a major role in several parts of this thesis. In fact, a number of stability and convergence results used here have their basis in the variants and generalizations of contraction theorem, such as

the Inverse and Implicit function theorems, Sard's theorem, etc. For a more broader treatment on the subject see [226].

Chapter Two

Singularity Interaction Dynamics (SID) Model

2.1. Introduction

In this chapter we introduce efficient, adaptive dynamical neural network formalisms for learning nonlinear transformations from randomly sampled examples. A key characteristic of our algorithms is their firm mathematical basis in the nonlinear dynamical systems theory, introduced in Chapter One. Specifically, our methodology is based on a recent breakthrough in nonlinear dynamical systems theory - the concept of "terminal" attractors [302,303], that were shown to correspond to singular solutions of the nonlinear neural dynamics with infinite local stability. Using topographically-mapped interacting terminal attractors, we construct a neural network whose synaptic elements can rapidly acquire the functional invariances embedded within a few training samples and subsequently generalize to predict responses over the operational domain. Appropriately, we name the neural network *Singularity Interaction Dynamics (SID) Model.* In a departure from prior neural learning algorithms this methodology provides mechanisms for incorporating an in-training "skew" to satisfy network as well as design constraints during the learning phase. Two algorithmic versions are derived - *Algorithm SID_1* targeted for problems involving few samples and reduced cardinality of input-output space; and *Algorithm SID_2* for large dimensional networks.

In an attempt to validate our theoretical framework and benchmark its computational efficacy we implement *Algorithm SID_1* on sequential digital hardware

to perform signal reconstruction from noisy, multidimensional phase data. The *Algorithm SID_2*,on the other hand, is simulated to solve a significantly more complex problem from the robotics domain - the *inverse kinematics problem*, commonly encountered during the design of real-time, adaptive systems operating in redundant environments. Specifically, the simulations are performed on a 3-DOF planar redundant manipulator and a 7-DOF cartesian redundant manipulators.

The organization of the remaining portion of the chapter is as follows. In section 2.2 we specify the neural network architecture, and derive corresponding learning equations in terms of new algorithms for constrained differential optimization which strictly enforce the Lyapunov stability criteria. In particular, we elucidate the notion of "terminal attractors" based on non-Lipschitzian dynamics, and describe their implications towards neural modeling. In Section 2.3 we discuss our simulation experiments with the two models. In Section 2.3.1. we define the multidimensional signal sorting problem, present a rudimentary system architecture and discuss our results. Section 2.3.2 provides details on the inverse kinematics problem, including a brief discussion on the existing algebraic, geometric and neuromorphic strategies and their limitations. We then present the results of our experiments with learning the inverse kinematics of 3-DOF and a 7-DOF redundant manipulators. The last section presents the conclusions of this chapter.

2.2. Neurodynamics Model

2.2.1. Network Specification

Consider a fully connected neural network with N graded-response neurons, implementing a functional mapping from the N_X-dimensional input space to the N_Q-dimensional output space.

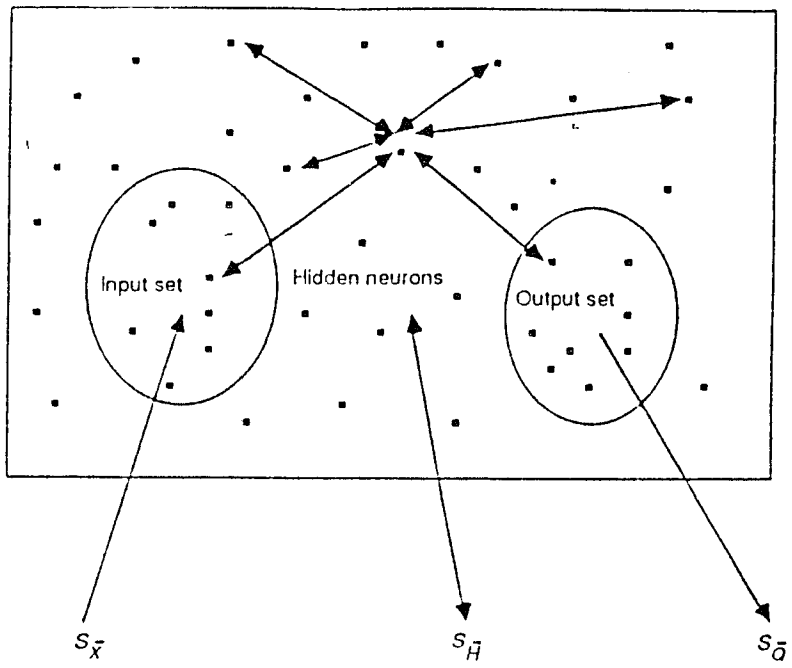


Figure 2.2.1.1. Topographically partitioned neuro-attractor map for the fully connected neural network model.

As shown in Fig. 2.2.1.1, the network is topographically partitioned into three mutually exclusive regions comprising of a set of input neurons, S_X, that receive the end-effector task coordinates, an output set S_Q, which provides the angular coordinates required to achieve the desired end-effector motion and a set of "hidden" neurons, S_H, whose sensitizations partially encode the input/output mapping being learnt. The network is presented with K randomly sampled training pairs of cartesian- and joint-space variables, $\{ \bar{x}^k, \bar{q}^k \mid k = 1, \ldots, K \}$ obtained by solving the well-posed forward kinematics formulation (see Paul [233]).

2.2.2. Learning Objectives

The neuromorphic reformulation to the computational learning problem requires determining synaptic interconnection strengths that can accurately capture the transcendental transformations embedded within the training samples. Our approach is based upon the minimization of a constrained "neuromorphic energy" function [37,113] given by the following expression

$$E = \frac{1}{2K} \sum_{k=1}^{K} \left(\frac{1}{N_X} \sum_{l \in S_X} [\, u_l^k - x_l^k \,]^2 + \frac{1}{N_Q} \sum_{l \in S_Q} [\, u_l^k - q_l^k \,]^2 \right) + \sum_r \lambda_r \, g_r(\cdot) \qquad (2.2.2.1)$$

where u_l^k denotes the l-th neuron's activity when processing the k-th training sample, g_r ($\cdot$) reflect network design considerations related to specific applications e.g., manipulability [58,197], and λ_r denotes the Lagrangian multiplier corresponding to the r-th application or design requirement. The proposed objective function includes contributions from two sources:

[1] It enforces the convergence of every neuronal state in S_X and S_Q to attractors corresponding to the presented end-effector task coordinates and joint coordinates respectively, for every sample pair in the training set.

[2] It enforces the synaptic elements to satisfy network constraints of the type,

$$g_r(\cdot) = \frac{1}{2} \sum_i \sum_j (i - j) \, T_{ij}^2 \qquad (2.2.2.2)$$

which minimize the interconnection strengths in line with the Gauss's Least Constraint Principle [30,107,305]. Alternately, $g_r(\cdot)$ could represent an auxiliary design criteria [92 and references therein], e.g., motion-time of joints,

operational ranges, manipulability, torque optimization, etc. We now proceed with the formal derivation of the learning equations (time evolution of the synaptic weights) by minimizing the energy function given in eqn. (2.2.2.1).

2.2.3. Neuromorphic Constrained Optimization

In the past, several neuromorphic algorithms have been proposed for constrained minimization of non-convex energy functions. For details the reader may refer to Hopfield and Tank [136], Barhen et al. [37], Gulati et al [113] and Platt and Barr [240]. In order to motivate and distinguish our optimization approach from the existing techniques, we first briefly examine some of the features which limited the general applicability of previous approaches. Hopfield and Tank's method for the Traveling Salesman problem [136] involved the minimization of an energy function of the type,

$$E = f(\bar{u}) + \sum_r W_r \, [\, g_r(\bar{u})]^2 \qquad (2.2.3.1)$$

A first difficulty with this model is that the specific constraint strengths, W_r, were determined heuristically, i.e., by "*anecdotal exploration*". Furthermore, the adopted penalty function construction was known to easily lead to constraint violation. Also, as the dimensionality of constraints increases the constraint strengths get harder to set [240]. A recently proposed alternative, i.e., Platt and Barr's Basic Differential Multiplier Method, [240], alleviates some of these limitations by modifying the objective function to

$$E = f(\bar{u}) + \sum_r \lambda_r \, g_r(\bar{u}) \qquad (2.2.3.2)$$

where λ_r denote the *Lagrange multipliers* corresponding to the constraints

$g_r(\bar{u}) = 0$. A straightforward (but naive) application of Lyapunov's stability requirements, (i.e. $\dot{E} < 0$), would result in the following equations of motion:

$$\dot{u}_i = -\frac{\partial E}{\partial u_i} = -\frac{\partial f}{\partial u_i} - \sum_r \lambda_r \frac{\partial g_r(\bar{u})}{\partial u_i} \qquad (2.2.3.3)$$

and

$$\dot{\lambda}_r = -\frac{\partial E}{\partial \lambda_r} = -g_r(\bar{u}) \qquad (2.2.3.4)$$

However, for some pathological cases the above algorithm could result in $\lambda_r \rightarrow 0$, i.e., the constraints might no longer be satisfied. Hence, Platt and Barr suggested the following heuristic change :

$$\dot{\lambda}_r = +g_r(\bar{u}). \qquad (2.2.3.5)$$

However, their proof of correctness upon inclusion of the above heuristic is based on assumptions which are extremely restrictive in nature. Specifically, the necessary condition to achieve stability requires establishing equivalence to a damped mass system, which in itself is a nontrivial mathematical exercise. In contrast (see below) , the methodology we propose, guarantees unconditional stability.

Lyapunov's stability criteria require an energy function to be monotonically decreasing in time. Since in our model the internal dynamical parameters of interest are the synaptic interconnection strengths T_{nm} and the Lagrange multipliers λ_r, this implies that

$$\dot{E} = \sum_n \sum_m \frac{\partial E}{\partial T_{nm}} \dot{T}_{nm} + \sum_r \frac{\partial E}{\partial \lambda_r} \dot{\lambda}_r < 0 \qquad (2.2.3.6)$$

One can choose

$$\tau_T \, \dot{T}_{nm} \;=\; -\frac{\partial E}{\partial T_{nm}} \tag{2.2.3.7}$$

where τ_T is an arbitrary but positive time-scale parameter. Then substituting in Eqs. (2.2.3.6) we have

$$\sum_r \frac{\partial E}{\partial \lambda_r} \, \dot{\lambda}_r \;<\; \tau_T \, \dot{T} \oplus \dot{T}. \tag{2.2.3.8}$$

In the above expression $\oplus$ denotes tensor contraction , i.e.,

$$\dot{T} \oplus \dot{T} \equiv \sum_i \sum_j \dot{T}_{ij} \, \dot{T}_{ij}.$$

Inequality 2.2.3.8 will be true *a fortiori* if for some $\theta > 0$,

$$\sum_r \dot{\lambda}_r \frac{\partial E}{\partial \lambda_r} + \theta \;<\; \tau_T \, \dot{T} \oplus \dot{T}.$$

The equations of motion for the Lagrange multipliers λ_r must now be constructed in such a way that Eq. (2.2.3.8) is strictly satisfied. Noting that the analytic expression for the energy function results in $\frac{\partial E}{\partial \lambda_r} = g_r(\cdot)$, we adopt the following model:

$$\dot{\lambda}_r \;=\; \tau_T \, \frac{\dot{T} \oplus \dot{T} \;-\; \theta}{\bar{g} \oplus \bar{g} \;+\; \theta} \, g_r(\cdot) \tag{2.2.3.9}$$

where $\bar{g} \oplus \bar{g} \equiv \sum_r g_r(\cdot) \, g_r(\cdot)$, and θ is an arbitrary positive constant. It is easy to see that $\dot{E} < 0$ is then strictly satisfied.

On differentiating (2.2.2.1) with respect to T_{nm} we get

$$\begin{aligned} \frac{\partial E}{\partial T_{nm}} \;=\; & \frac{1}{K} \sum_k \left\{ \frac{1}{N_X} \sum_{l \in S_X} \left[u_l^k - x_l^k \right] \frac{\partial u_l^k}{\partial T_{nm}} \right. \\ & \left. + \frac{1}{N_Q} \sum_{l \in S_Q} \left[u_l^k - q_l^k + N_Q \sum_r \frac{\partial g_r}{\partial u_l^k} \right] \frac{\partial u_l^k}{\partial T_{nm}} \right\} \end{aligned} \tag{2.2.3.10}$$

If we define,

$$\hat{I}_l^k = \begin{cases} \frac{1}{KN_Q}\left[u_l^k - q_l^k + N_Q \sum_r \frac{\partial q_r}{\partial u_l^k} \right] & \text{if } l \in S_Q \\ 0 & \text{if } l \in S_H \\ \frac{1}{KN_X}\left[u_l^k - x_l^k \right] & \text{if } l \in S_X \end{cases} \qquad (2.2.3.11)$$

we can rewrite (2.2.3.10) as

$$\frac{\partial E}{\partial T_{nm}} = \sum_l \sum_k \hat{I}_l^k \frac{\partial u_l^k}{\partial T_{nm}} \qquad (2.2.3.12)$$

where the index l is defined over the entire set of neurons. Equations [2.2.3.7, 2.2.3.9 and 2.2.3.11] constitute a dissipative nonlinear dynamical system, the flow of which generally converges to a manifold of lower dimensionality in the phase space. In this chapter we focus on network convergence to point attractors, i.e., state-space vector locations corresponding to the presented , joint- and Cartesian-space coordinates. Of crucial importance is to know how stable those attractors are and how fast can they be reached. In this vein, we first briefly review a novel mathematical concept in nonlinear dynamical systems theory, the *terminal attractor*, and its properties, that subsequently will enable us to formalize neural network algorithms for learning the inverse kinematics mapping.

2.2.4. Terminal Attractor Neurodynamics

Hopfield and others [13,15,16,75,134,135,153] have shown that artificial neural networks store memory states or patterns in terms of the fixed points of the network dynamics, such that initial configurations of neurons in some neighborhood or *basin of attraction* of that memory state will be attracted to it. But the static attractors considered so far in nonlinear dynamical system formulations in general, and in neural network models in particular, have represented regular

solutions of the differential equations of motion as shown in Fig. 2.2.4.1. The theoretical relaxation time of the system to these "regular attractors" can theoretically be infinite, and they suffer from convergence to spurious states and local minima. The concept of *terminal attractors* in dynamical systems, was initially introduced by Zak [302], to obviate some of the above limitations, thereby significantly improving the performance characteristics of associative memory neural network models.

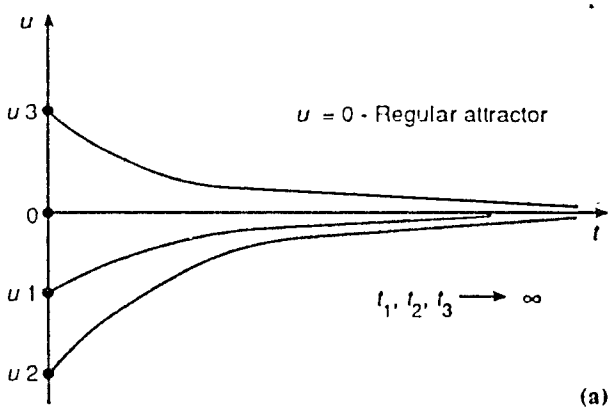


Figure 2.2.4.1. (a) Asymptotic Relaxation of regular attractor to the fixed point of neurodynamics.

The existence of terminal attractors was established by Zak using the following argument. At equilibrium, the fixed points, $\bar{p}$, of an N-dimensional, dissipative dynamical system

$$\dot{u}_i - f_i(u_1, u_2, , \cdots, u_N) = 0 \quad i = 1, 2, \cdots, N \qquad (2.2.4.1)$$

are defined as its constant solutions $\bar{u}^{\infty}(\bar{p})$. If the real parts of the eigenvalues, η_μ of the matrix $M_{ij} = \left[\frac{\partial f_i}{\partial u_j}(\bar{p})\right]$ are all negative, i.e., Re $\{\eta_\mu\} < 0$ then these points are globally asymptotically stable [4]. Such points are called static attractors since each motion along the phase curve that gets close enough to $\bar{p}$, i.e., enters a so called basin of attraction, approaches the corresponding constant value as a limit as $t \to \infty$. An equilibrium point represents a repeller if at least one of the eigenvalues of the matrix M has a positive real part. Usually, nonlinear neural network deal only with systems which satisfy the Lipschitz conditions, i.e., $\mid \partial f_i \; / \; \partial u_j \mid < \infty$ This condition guarantees the existence of a unique solution for each of the initial phase space configurations. That is why a transient solution cannot intersect the corresponding constant solution to which it tends, and therefore, the theoretical time of approaching the attractors is always infinite. Fig. 2.2.4.1(a) shows the temporal evolution to such an attractor.

In contrast, Zak's [303] notion of terminal attractors is based upon the violation of Lipschitz conditions. As a result of this violation the fixed point becomes a singular solution which envelops the family of regular solutions, while each regular solution approaches the terminal attractor in finite time, as displayed in Fig. 2.2.4.1(b). To formally exhibit a terminal attractor which is approached by transients in finite time, consider the simplest one-dimensional example:

$$\dot{u} = -u^{1/3} \qquad (2.2.4.2)$$

This equation has an equilibrium point at $u = 0$ at which the Lipschitz uniqueness condition is violated, since

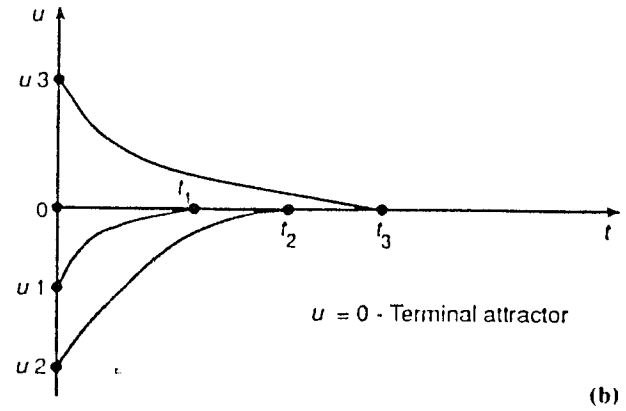


Figure 2.2.4.1. (b) Terminal Attractor as a singular solution to the dynamical system.

$$\frac{d\dot{u}}{du} = -\frac{1}{3}u^{-2/3} \longrightarrow -\infty \ at \ u \longrightarrow 0 \qquad (2.2.4.3)$$

Since here the Re $\{\eta\} \longrightarrow -\infty < 0$ this point is an attractor with "infinite" local stability. As a consequence the dynamical system is locally bestowed with "infinite attraction power", enabling rapid clamping of neuronal potentials to the fixed points; in our case this implies immediate relaxation to the desired attractor coordinates, x_I and q_I. Also, the relaxation time for the solution corresponding to initial conditions $u = u_0$ to this attractor is finite. It is given by

$$t_0 = -\int_{u_0}^{u \to 0} \frac{du}{u^{1/3}} = \frac{3}{2}u_0^{2/3} < \infty \qquad (2.2.4.4)$$

i.e., this attractor becomes terminal. As shown in Fig. 2.2.4.2, it represents a singular solution which is intersected by all the attracted transients. In partic-

ular, static terminal attractors occur for $k = (2n+1)^{-1}$ and $n \geq 1$, while for $k = 2n+1$ and $n \geq 0$ all attractors are regular. It has been shown (Zak [301]) that incorporation of terminal attractors into neural dynamics leads to the elimination of all spurious states in associative memories. This property is critical to providing an accurate generalization ability, since it ensures that interpolations/extrapolations of joint configurations are not based on false attractors. For details on implication of terminal attractor dynamics for neural learning algorithms see [35,302]. In our proposed neuromorphic framework, terminal attractor dynamics then provides a mechanism that can implicitly exploit the time-bounded terminality of phase trajectories and the locally infinite stability.

2.2.5. "Virtual" Attractor Computation

The "energy" function defined in Eqs. (2.2.2.1) specified the functionality of our fully connected neural network, i.e., learn the inverse kinematics mapping. We now need to select the network dynamics for evolving the synaptic elements, such that, the latter's convergence to steady state fulfills the above objective. So to capture the kinematic invariances consider following coupled neurodynamics:

$$\tau_u \dot{u}_l^k + u_l^k = \varphi_\gamma [\sum_{l'} T_{ll'} u_{l'}^k] - I_l^k \qquad (2.2.5.1)$$

Here u_l represents the mean soma potential of the lth neuron (u_l^k is the neuron's activity when processing the kth training sample), $T_{ll'}$ denotes the synaptic coupling from the l'-th to the l-th neuron, and I_l^k captures the input/output contribution in a terminal attractor formalism. Though I_l^k influences the degree of stability of the system and the convergence to fixed points in finite time, it does not further affect the location of existing static attractors. In Eqn. (2.2.5.1),

$\varphi_\gamma(\cdot)$ denotes the sigmoidal neural response function with gain γ; typically,

$$\varphi_\gamma(z) \;=\; \tanh(\gamma \cdot z).$$

In topographic maps, N_T neurons are generally used to compute a single value of interest in terms of spatially-coded response strengths. Here we use the simplest possible model (where $N_T = 1$), but encode the information through terminal attractors. Thus, the topographic map is given by

$$I_l^k \;=\; \begin{cases} (\; u_l^k \;-\; x_l^k \;)^{1/3} & \text{if } l \in S_X \\ 0 & \text{if } l \in S_H \\ (\; u_l^k \;-\; q_l^k \;)^{1/3} & \text{if } l \in S_Q \end{cases} \qquad (2.2.5.2)$$

where x_l^k and q_l^k are the attractor coordinates provided by the training sample, to be denoted for brevity as a_l^k. Our basic operating assumption for the dynamical system defined by (2.2.5.1) is that at equilibrium, for l = 1,...,N :

$$\dot{u}_n \;\longrightarrow\; 0 \qquad \text{and} \qquad u_n \;\longrightarrow\; a_n$$

This yields the fixed point equations :

$$a_n \;=\; \varphi_\gamma \left[\sum_m T_{nm}\, a_m \right] \qquad (2.2.5.3)$$

In associative memory applications, these equations can in principle be used to determine the synaptic coupling matrix T, resulting in each memory pattern being stored as a fixed point. The key issue is that some of these fixed points may actually be repellers. The terminal attractors are thus used to guarantee that each fixed point becomes an attractor, i.e., spurious states are suppressed. Here however, we are in the process of learning a mapping between two spaces and as indicated in Fig. 2.2.1.1, attractor coordinates have been defined for only two of the three topographic regions of the network, i.e., the input set S_X, and the output set S_Q. Consequently, the fixed point equation

$\bar{a} = \varphi(T\bar{a})$ may not necessarily be defined, since for $| S_H | > 0$, $\{ a_n \mid n \in S_H \}$ are not defined, and cannot be used for directly computing T.

This necessitates the development of an alternative strategy, whereby "virtual" attractor coordinates are first determined for the hidden units. These coordinates are virtual since they correspond to a current estimate $\hat{T}$ of the synaptic connectivity matrix. This is achieved by considering the fixed point equations as *adaptive conservation equations* which use the extra degrees of freedom made available by the hidden neurons in S_H. Let $\{ \hat{u}_j = a_j \mid j \in S_H \}$ denote the *virtual attractors* to which the unknowns, $\{ u_j \mid j \in S_H \}$ are expected to converge to. Then at equilibrium, Eqs. (2.2.5.3) yield

$$\begin{aligned}
\varphi^{-1}(x_i) &= \sum_{i' \in S_X} \hat{T}_{ii'} x_{i'} + \sum_{j' \in S_H} \hat{T}_{ij'} \hat{u}_{j'} + \sum_{l' \in S_Q} \hat{T}_{il'} q_{l'} \qquad \forall i \in S_X \\
\varphi^{-1}(\hat{u}_j) &= \sum_{i' \in S_X} \hat{T}_{ji'} x_{i'} + \sum_{j' \in S_H} \hat{T}_{jj'} \hat{u}_{j'} + \sum_{l' \in S_Q} \hat{T}_{jl'} q_{l'} \qquad \forall j \in S_H \\
\varphi^{-1}(q_l) &= \sum_{i' \in S_X} \hat{T}_{li'} x_{i'} + \sum_{j' \in S_H} \hat{T}_{lj'} \hat{u}_{j'} + \sum_{l' \in S_Q} \hat{T}_{ll'} q_{l'} \qquad \forall l \in S_Q
\end{aligned} \tag{2.2.5.4}$$

where $\hat{T}_{jl}$ denotes the current estimate of synaptic coupling from lth neuron to the jth neuron, and $\hat{u}_j$ represents a virtual attractor whose value is isomorphic to the current level of knowledge in the network. Now define,

$$\begin{aligned}
\psi_i &= \varphi^{-1}(x_i) - \sum_{i'} \hat{T}_{ii'} x_{i'} - \sum_{l'} \hat{T}_{il'} q_{l'} \qquad \forall i \in S_X \\
\psi_j &= \sum_{i'} \hat{T}_{ji'} x_{i'} + \sum_{l'} \hat{T}_{jl'} q_{l'} \qquad \forall j \in S_H \\
\psi_l &= \varphi^{-1}(x_l) - \sum_{i'} \hat{T}_{li'} x_{i'} - \sum_{l'} \hat{T}_{ll'} q_{l'} \qquad \forall l \in S_Q.
\end{aligned} \tag{2.2.5.5}$$

Then consistency with the terminal attractor dynamics assumptions requires that $\{ \hat{u}_j \mid j \in S_H \}$ be simultaneous solutions to the following "conservation" equations

$$\begin{aligned} \sum_{j' \in S_H} \hat{T}_{ij'} \hat{u}_{j'} &= v_i \qquad \forall\, i \in S_X \\ \varphi^{-1}(\hat{u}_j) - \sum_{j' \in S_H} \hat{T}_{jj'} \hat{u}_{j'} &= v_j \qquad \forall\, j \in S_H \\ \sum_{j' \in S_H} \hat{T}_{lj'} \hat{u}_{j'} &= v_l \qquad \forall\, l \in S_Q \end{aligned} \tag{2.2.5.6}$$

The above system of equations for $\hat{u}$ is generally overdetermined. A number of standard algorithms exist to obtain a good approximate solution to such a system. In our implementation we use an iterative approach, e.g., gradient descent or conjugate gradient descent (a survey of gradient descent algorithms may be found in [2,312]).

$$\begin{aligned} \hat{E} = {} & \frac{1}{2N_X} \sum_i \left(\psi_i - \sum_{j'} \hat{T}_{ij'} \hat{u}_{j'} \right)^2 + \\ & \frac{1}{2N_H} \sum_j \left(\hat{u}_j - \varphi[\sum_{j'} \hat{T}_{jj'} \hat{u}_{j'} + \psi_j] \right)^2 + \\ & \frac{1}{2N_Q} \sum_l \left(\psi_l - \sum_{j'} \hat{T}_{lj'} \hat{u}_{j'} \right)^2 \end{aligned} \tag{2.2.5.7}$$

to obtain the virtual attractors, $\hat{u}_j \; \forall j \in S_H$ by minimizing the above energy function.

2.2.6. Singularity Interaction Dynamics Formalism

2.2.6.1. *Algorithm SID_1* : A "naive" Formalism

Under an adiabatic framework, at equilibrium, the fixed point equations developed in Eqs. (2.2.5.3), can be rewritten to yield

$$b_n^k \;=\; (u_n^k)^\infty \;=\; \varphi^{-1}(a_n^k) \tag{2.2.6.1.1}$$

where a_n^k denote attractor coordinates; and

$$b_n^k \;=\; \sum_m T_{nm}\, \varphi(u_n^k)^\infty \;=\; \sum_m T_{nm}\, a_m^k \tag{2.2.6.1.2}$$

If we write

$$A \;=\; \left[\begin{array}{cccccc} \bar{a}^1 & \cdots & \bar{a}^k & \cdots & \bar{a}^K \end{array}\right] \;=\; A_{N\cdot K}$$

$$B \;=\; \left[\begin{array}{cccccc} \bar{b}^1 & \cdots & \bar{b}^k & \cdots & \bar{b}^K \end{array}\right] \;=\; B_{N\cdot K},$$

then from Eqs. (2.2.6.1.1) and (2.2.6.1.2)

$$B_{N\cdot K} \;=\; T_{N\cdot N} \cdot A_{N\cdot K} \tag{2.2.6.1.3}$$

or equivalently

$$B_{N\cdot K} \cdot \tilde{A}_{K\cdot N} \;=\; T_{N\cdot N} \cdot \left(A_{N\cdot K} \cdot \tilde{A}_{K\cdot N} \right) \tag{2.2.6.1.4}$$

where $\sim$ denotes the transpose operator. Thus, from elementary linear algebra,

$$T \;=\; B\, \tilde{A}\, (A\, \tilde{A})^{-1}. \tag{2.2.6.1.5}$$

Since one would need to verify that $A \cdot \tilde{A}$ is nonsingular at every learning iteration (an $O(N^3)$ operation, the above system cannot be efficiently employed for learning in neural networks for problems wherein large numbers of training

samples are necessary for abstracting the nonlinear map. Thus, notwithstanding the conceptual simplicity, we feel this algorithm will have limited usage.

A number of alternative methods exist for solving the above nonlinear system of equations. We adopt a dynamical systems approach, wherein Eqn. (2.2.6.1.5) is formulated as the steady state solution to the system and rewritten in the terminal attractor formalism. Thus, the learning rule can then be stated as:

$$\dot{T}_{nm} = \left\{ \left(B_{N \cdot K} \cdot \tilde{A}_{K \cdot N} \right)_{nm} - \left[T_{N \cdot N} \cdot \left(A_{N \cdot K} \cdot \tilde{A}_{K \cdot N} \right) \right]_{nm} \right\}^{\frac{1}{3}} \tag{2.2.6.1.6}$$

At equilibrium the above system yields the learned synaptic strengths, T_{nm}. We now summarize the computational structure of the learning algorithm.

Computation Learning Algorithm: SID_1

/** This algorithm describes a computational structure for encoding nonlinear mappings using topographically partitioned terminal attractors ***/

Input: input/output attractor coordinates x^k and q^k; network dimension; neurodynamical decay constants κ_i; neural response function φ; temporal grid; convergence criteria; initialization domain.

Output: learned interconnection matrix T_{nm}.

Algorithm Singularity Interaction Dynamics_1

[1] Randomly initialize: $\forall\, n,\ m \in \{ 1, \cdots, N \}$ and $\forall\, k \in \{ 1, \cdots, K \}$:

$$T_{nm} = \Re [-\epsilon, +\epsilon]$$

[2] Learn synaptic matrix T :

Iterate over $\nu = 1, \cdots, N_T$

[3] **Loop** over training samples, $k = 1, \cdots, K$

[3.1] Initialize u_n in $\Re\ [\ -\epsilon,\ +\epsilon\ ]$

[3.2] Estimate virtual attractors, $\{\hat{u}_n^k \mid n \in S_H\}$
from conservation equations Eqs. (2.2.5.4) to Eqs. (2.2.5.7)

[3.3] Evolve network dynamics using Eqs. (2.2.5.1)

$$\dot{u}_n + \kappa_n u_n = \sum_m T_{nm}^{\nu} \varphi(u_m) + {}^k I_n$$

where

$${}^k I_n = \begin{cases} [a_n^k - \varphi(u_n)]^{1/3} & \text{if } n \in S_X \cup S_Y \\ [\tilde{u}_n^k - \varphi(u_n)]^{1/3} & \text{if } n \in S_H \end{cases}$$

Output: $\varphi({}^k\tilde{u}_n^{\nu})$ and ${}^k\tilde{I}_n^{\nu}$

[4] **Endloop** over training samples $\{k\}$

[5] Update T using Eqs. (2.2.6.1.6)

$$T_{nm}^{\nu+1} = T_{nm}^{\nu} + \tau\Delta\left\{(B \cdot \tilde{A})_{nm} - [T_{nm}^{\nu} \cdot (A \cdot \tilde{A})]_{nm}\right\}^{1/3}$$

where $b_n^k = \varphi^{-1}(a_n^k)$

[6] Check for convergence:

If yes **then** exit **else** goto [2]

[7] **Endloop** over learning iterations $\{\nu\}$

[8] **Exit :** Display results

2.2.6.2. *Algorithm SID_2*: Rigorous Formalism

In the preceding derivation of *Algorithm SID_1*, the synaptic update equation was derived using heuristics. We now modify our methodology to derive the learning rule using a relaxation procedure adapted from Pineda [238]. Returning to the computation of $\partial u_l^k \ / \ \partial T_{nm}$ in Eq. (2.2.3.11). Let us define

$$z_l^k \ = \ \sum_{l'} T_{ll'} \ u_{l'} \tag{2.2.6.2.1}$$

and denote

$$\varphi'_{lk} \ = \ \frac{\partial \varphi(\cdot)}{\partial z_l^k}. \tag{2.2.6.2.2}$$

Then at equilibrium, as $\dot{u}_l^k \ \longrightarrow \ 0$ and $I_l^k \ \longrightarrow \ 0$, we have

$$\frac{\partial u_l^k}{\partial T_{nm}} \ = \ \varphi'_{lk} \left[\ \sum_{l'} \frac{\partial T_{ll'}}{\partial T_{nm}} \ u_{l'}^k \ + \ \sum_{l'} T_{ll'} \ \frac{\partial u_{l'}^k}{\partial T_{nm}} \right] \tag{2.2.6.2.3}$$

which can be rewritten as

$$\sum_{l'} \left[\ \delta_{ll'} \ - \ \varphi'_{lk} \ T_{ll'} \ \right] \frac{\partial u_{l'}^k}{\partial T_{nm}} \ = \ \varphi'_{lk} \delta_{ln} u_m^k \tag{2.2.6.2.4}$$

In the above expression δ_{ij} denotes the Kronecker symbol. We now define, following [238], a weighted coupling matrix

$$A_{ll'}^k \ = \ \delta_{ll'} \ - \ \varphi'_{lk} \ T_{ll'}. \tag{2.2.6.2.5}$$

Then, substituting (2.2.6.2.5) in (2.2.6.2.4), and premultiplying both sides with $[A^{-1}]_{nl}^k$ and summing over l yields

$$\sum_{l} [A^{-1}]_{nl}^k \sum_{l'} A_{ll'}^k \ \frac{\partial u_{l'}^k}{\partial T_{nm}} \ = \ \sum_{l} [A^{-1}]_{nl}^k \ \varphi'_{lk} \ \delta_{ln} \ u_m^k. \tag{2.2.6.2.6}$$

Carrying out the algebra, and relabeling the dummy indices results in:

$$\frac{\partial u_l^k}{\partial T_{nm}} = [A^{-1}]_{ln}^k \, \varphi'_{nk} \, u_m^k. \qquad (2.2.6.2.7)$$

The above expression can now be substituted in Eq. (2.2.6.2.5); the learning equation thus takes the form

$$\tau_T \dot{T}_{nm} = -\sum_l \sum_k \hat{I}_l^k \, [\, A^{-1} \,]_{ln}^k \, \varphi'_{nk} \, u_m^k \qquad (2.2.6.2.8)$$

where the indices l and k run over the complete sets of neurons and training samples.

2.2.6.2.1. Error Propagation Dynamics

A computation of the synaptic interconnection matrix as suggested by Eq. (2.2.6.2.8) would involve a matrix inversion. Since direct matrix inversion is typically nonlocal, we adopt the relaxation heuristic suggested by Pineda [239] to compute the synaptic updates defined by (2.2.6.2.8). Consider the following change of variable

$$v_n^k = \sum_l [\, A^{-1} \,]_{ln}^k \, \hat{I}_l^k \, \varphi'_{nk} \qquad (2.2.6.2.1.1)$$

Then substituting (2.2.6.2.1.1) in (2.2.6.2.8) we have

$$\begin{aligned} \sum_n A_{np}^k \, \frac{v_n^k}{\varphi'_{nk}} &= \sum_l \hat{I}_l^k \sum_n [A^{-1}]_{ln}^k \, A_{pn}^k \\ &= \sum_l \hat{I}_l^k \, \delta_{lp} \\ &= \hat{I}_p^k \end{aligned} \qquad (2.2.6.2.1.2)$$

One can also use the explicit form of A_{np}^k from (2.2.6.2.5) and by substitution in (2.2.6.2.8), we obtain

$$\sum_n A^k_{np} \frac{v^k_n}{\varphi'_{nk}} = \sum_n \delta_{np} \frac{v^k_n}{\varphi'_{nk}} - \sum_n \varphi'_{nk} T_{np} \frac{v^k_n}{\varphi'_{nk}}$$

$$= \frac{v^k_p}{\varphi'_{pk}} - \sum_n T_{np} v^k_n \qquad (2.2.6.2.1.3)$$

Regrouping the previous equations (2.2.6.2.1.2) and (2.2.6.2.1.3), and relabeling the dummy indices yields

$$v^k_n = \varphi'_{nk} \cdot [\sum_p T_{pn} v^k_p + \hat{I}^k_n]. \qquad (2.2.6.2.1.4)$$

We see that v^k_n represents a fixed point solution of a neural network having the following coupled dynamics

$$\tau_v \dot{v}^k_n + v^k_n = \varphi'_{nk} \cdot [\sum_p T_{pn} v^k_p + \hat{I}^k_n] \qquad (2.2.6.2.1.5)$$

Recall that $\hat{I}^k_l$ was defined in Eq. (2.2.5.2). By comparing Eqs. (2.2.6.2.8, 2.2.6.2.1.1 and 2.2.6.2.1.5) we see that the resulting neural learning equations couple the terminal attractor dynamics for u^k_m with the error propagation dynamics for v^k_n, i.e.,

$$\tau_T \dot{T}_{nm} = -\sum_k v^k_n u^k_m \qquad (2.2.6.2.1.6)$$

The complete algorithm is summarized below.

Computational structure of Algorithm SID_2

** This algorithm abstracts nonlinear transcendental functions from input-output examples, subject to design and network constraints using the method of Lagrange Multipliers. **/

Input : Attractor coordinates, a_n^k, network dimension and topographic partitioning, neurodynamical decay constants, neural response function and gains, temporal grid, convergence and scaling criteria, initialization domain.

Output : learned synaptic matrix, T_{nm}.

Algorithm Singularity interaction Dynamics_2

[0] Initialize $\hat{T}$, $\hat{\lambda}$

[1] Learn $\hat{T}$: **iterate IT** = 1, .., NIT

[1.0] **Loop** over training samples, **k** = 1, .., K

[2] Initialize $\bar{u}^k$, $\bar{v}^k$

[2.1] Estimate virtual attractors, $\{\hat{u}_j^k \mid j \in S_H\}$ from conservation equations (2.2.5.5)-(2.2.5.7)

[2.2] Evolve $\bar{u}^k$ for inverse mapping $\bar{x}^k \rightarrow \bar{q}^k$ using terminal attractor dynamics (2.2.5.1)-(2.2.5.2)

[2.3] Compute $\bar{v}^k$ using the error propagation network

[2.4] Store outer product $\bar{u}^k \wedge \bar{v}^k$ increment

[2.5] **Enddo** {**k**}

[1.2] Update $\hat{T}$ using Eq. (2.2.6.2.1.6)

[1.3] Update $\hat{\lambda}$ using Eq. (2.2.3.9)

[1.4] Check for convergence:

If yes **then** exit **else** go to [1]

[1.5] **Enddo** {**IT**}

[2] **exit**

In the following section, we apply the above learning algorithms to real-life problems in signal processing and robotics.

2.3. Simulation Results

We consider the performance of SID_1 and SID_2 in the context of two applications: (1) multidimensional reconstruction of noisy signals; (2) neurokinematics of redundant manipulators.

2.3.1. Signal Sorting with *Algorithm SID_1*

The sorting of radio signals is central to many applications of signal analysis including the identification and characterization of signals in a dense RF environment. A generic architecture [263] which is currently utilized for rapid identification and characterization of communication signals and interference is depicted in Fig. 2.3.1.1 Briefly, the system works as follows: a noise processor and rapid scanning spectrum analyzer (either digital or analog) are used as the first steps in processing the received RF signal. The output from the spectrum analyzer is initially sorted based on the various measured discriminants available from the spectrum analyzer, e.g., amplitude, frequency, time, and angle-of-arrival data. The results of this initial sorting of signal externals is an emergence of distinct signals which can be used for subsequent signal identification and characterization. In addition, once a characteristic signal frequency is identified on subsequent scans, a set-on receiver can be rapidly tuned to the signal of interest for purpose of extracting signal internals (e.g., modulation type).

In developing an effective system for signal identification, a key consideration is the signal sorter. Existing signal sorters typically incorporate some form of histogram analysis or clustering algorithm for sorting the signals into distinct classes and incorporate either content of window-addressable memories for rapid signal identification. The problem with these existing techniques is that they impose the typical estimate-and-process architecture which cannot easily accommodate a rapidly time varying signal environment. In addition, these techniques

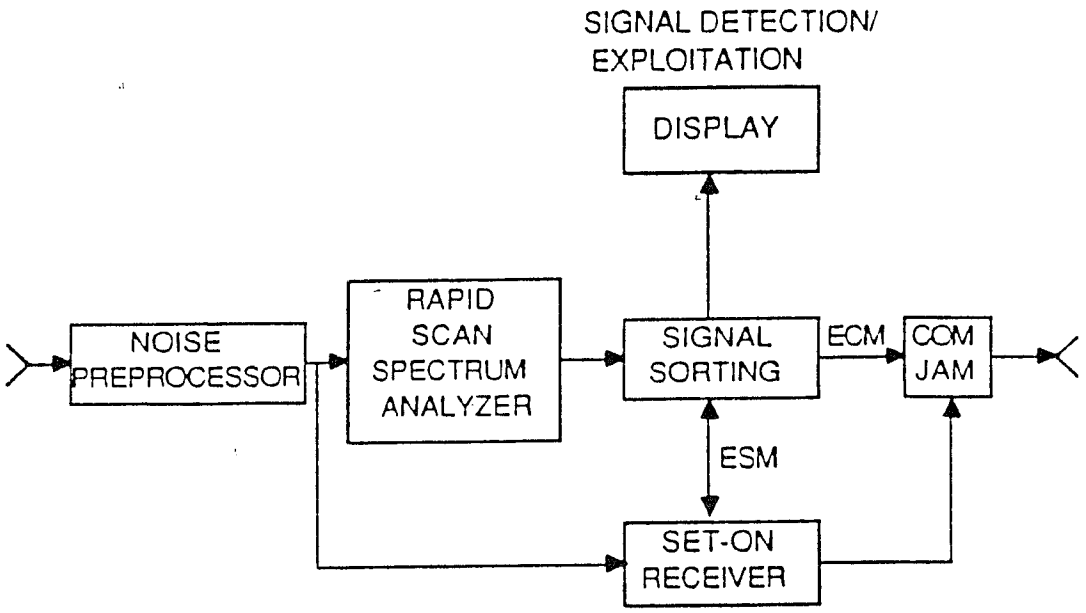


Figure 2.3.1.1. System architecture for communication signal identification and characterization.

do not necessarily incorporate multiple signal discriminant information in an optimal way. Thus, the performance of these existing techniques tends to rapidly degrade as the received signal-to-noise ratio decreases.

Algorithmically, the problem can be formulated as follows: find a mathematical transformation which maps a set of measured data into a set of signal characteristics, e.g., angle-of-arrival, amplitude. The main difficult lies in the fact that it is not a one-to-one mapping: due to inherent uncertainties one set of measured data can correspond to a number of signals with different characteristics. In other words, the formal mathematical approach requires overcoming the well known difficulties (ill-posedness, ill-conditioning) associated with inverse

Fig. 2.3.1.2 shows the feedforward neural network architecture that we implemented for purposes of benchmarking *Algorithm* SID_1. The inputs to the network are the outputs from the spectrum analyzer with a pair of input neurons being assigned to each frequency resolution cell of the spectrum analyzer. One member of each pair corresponds to amplitude data whereas the other member corresponds to angle-of-arrival (AOA) data. This input data is combined via the connection weights and hidden layers of the network and the output is a binary vector indicating whether the $i-$th emitter (or emitter class) is currently active or not. The control setup comprised of a feedforward network with 24 neurons: 16 input neurons, 6 hidden neurons and 2 output neurons. The training data was obtained from the output of simulated wideband spectrum analyzer.We assumed fixed frequency emitters with prematched amplitudes. The training set comprised of a 1000 linearly separable data from 16 frequency channels with varying noise levels. The classical back-propagation algorithm required $O(10^4)$ training iterations to learn the embedded relationship. The multiscale version of terminal-attractor based SID_1 Learning Algorithm, on the other hand was presented only 50 randomly sampled training samples from the set of 1000. It required only 170 training iterations (corresponding to an effective time of 17 milliseconds) to abstract the statistical relationship. More important, during recall, SID algorithm did not perform any false tagging with dropped classes. Detailed results are included in Gulati and Barhen [104].

2.3.2. Neurokinematics with *Algorithm SID_2*

Before presenting our simulation results on SID_2 on inverse kinematics of redundant manipulators, we motivate the relevance of neural networks to solving robotics problems. Space telerobots envisaged for exacting applications in unstructured and hazardous space environments, e.g., satellite servicing, space system construction and maintenance, planetary missions etc., must be able to dex-

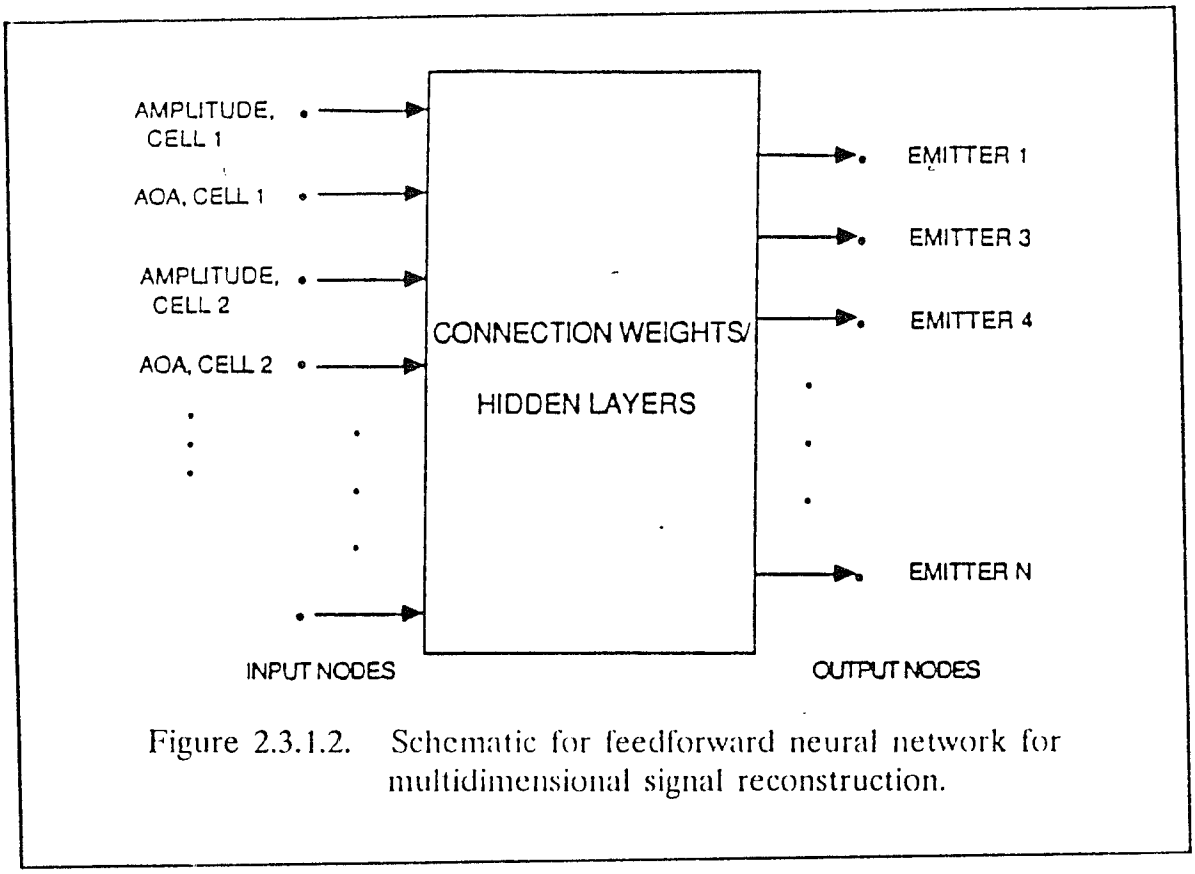


Figure 2.3.1.2. Schematic for feedforward neural network for multidimensional signal reconstruction.

structured and hazardous space environments, e.g., satellite servicing, space system construction and maintenance, planetary missions etc., must be able to dexterously and adaptively manipulate objects in a nonstationary task workspace. Redundancy in the design of robot manipulators has been suggested as one means to enhance their dexterity and adaptability. In contradistinction to other engineering contexts where redundancy implies fault-tolerance or superfluity, redundancy in robotics is determined relative to the task [58]. It refers to a manipulator with more than the minimum number of degrees of freedom necessary to accomplish general tasks. The major objective motivating introduction of redundancy in robot design and control is to use the additional degrees of freedom to improve performance in complex and unstructured environments. It helps overcome kine-

matic, mechanical and other design limitations of non-redundant manipulators. Also, the extra degrees of freedom can be used during real-time manipulator operation to simultaneously achieve end-effector trajectory control while satisfying additional constraints.

There are two primary problems in developing control strategies which take advantage of redundancy. First, given the initial and final end-effector task coordinates, simultaneously generate, in real time a Cartesian-space trajectory that can achieve a goal (*the path planning problem*) and a set of joint space trajectories which cause the end-effector to follow the desired trajectory (*inverse kinematics problem*) while satisfying additional constraints, such as obstacle avoidance, servo-motor torque minimization, singularity avoidance, and joint limit avoidance. Developing algorithms to use the additional degrees of freedom to satisfy constraints is known as the *redundancy resolution* problem [30,33,34,58,107,110]. Secondly, provide adaptive mechanisms for responding to any unforeseen changes in the workspace or the manipulator geometry. Despite a tremendous growth in research activity on "model-based" adaptive control algorithms, the above problems entail a level of computational and paradigmatic complexity far exceeding that what can be provided by the existing strategies. Artificial neural networks, on the other hand, as discussed in Chapter One, due to their ability to perform functional synthesis in real-time, could provide an attractive alternate basis towards designing real-time manipulator control architectures.

2.3.2.1. Manipulator Inverse Kinematics

Manipulator kinematics addresses the problem of computing temporal evolution of joint coordinates from the motion of robot end-effector. A forward kinematic function, Φ, is a nonlinear differentiable function which uniquely relates a set of N_Q joint variables, $\bar{q}$, to a set of N_X task-space coordinates, $\bar{x}$:

$\bar{x} = f(\Phi)$. For serial chain robot manipulators the forward kinematic function is easily derived [233]. The more difficult problem which is of primary practical interest in manipulators kinematics is the inverse problem:

$$\bar{q} = \Phi^{-1}(\bar{x}) \qquad (2.3.2.1.1)$$

i.e., determine one or more sets of joint configurations which take the end-effector into a desired task position and orientation in the workspace. While the inverse kinematic function is highly nonlinear, closed form analytical solutions can be found for a number of non-redundant manipulators with special architecture. Complete positioning capability in Cartesian space can be nominally achieved using only six degrees of freedom. However, most manipulators have degenerate configurations, or kinematic singularities, near which small displacements of the end-effector requires physically unrealizable joint speeds. These singularities effectively lead to a loss of usable workspace and capability, and there is a strong incentive to design robots with additional degrees of freedom. Thus, a robot manipulator is kinematically redundant if the number of its degrees of freedom is greater than the dimension of the end-effector task space. In contradistinction to other engineering contexts, where redundancy per se, implies *fault-tolerance*, i.e component duplication allowing for continued system functionality in the event of an element failure or *superfluity* i.e, an unneeded excess capacity, redundancy in robotics is determined relative to the task [58]. For example, a 6-DOF manipulator could be redundant with respect to tasks with symmetry about one axis, while an arm with 3 or more joints is redundant for achieving any end-effector position in a two-dimensional space. The major objective motivating introduction of redundancy in robot design and control is to use the additional degrees of freedom to improve performance in complex and unstructured environments. It helps overcome kinematic, mechanical and other design limitations of non-redundant manipulators, and simultaneously satisfy additional constraints, such

as obstacle avoidance, minimization of actuator torques, singularity avoidance, providing greater dexterity, minimization of kinetic energy, improvement of some measure of manipulability, etc.

However, incorporation of redundancy injects additional complexity into the problem. For redundant manipulators, the kinematic equations relating the specified end-effector task coordinates to the unknown joint angles may not have a unique solution, and in general the problem is both ill-posed and ill-conditioned [149], i.e., solutions may not necessarily exist, and if they do exist at all, they are likely to be nonunique. Further, such solutions, if they do exist, in general depend discontinuously on the input, i.e., $\bar{x}$ and hence are likely to be unstable to small errors in the input. For elucidation consider Fig. 2.3.2.1.1., which represent the transformation, $K : \bar{Q} \longrightarrow \bar{X}$. Let K denote the kinematics operator Φ. Inevitable measurement errors or signal noise can lead to data which do not lie in the range of K and hence not in the domain of the inverse operator K^{-1}; thus no solution will exist. Similarly, the transformation may not be one-to-one, so that inverse transformation does not yield a unique result. Jeffry and Rosner [149] have analyzed the unstable behavior of such systems, by examing the matrix form of equation $\bar{X} = \Phi\bar{Q}$. If $\bar{X}$ contains errors (say observational) ΔX, then a direct inverse in the least-squares sense, i.e., generalized pseudoinverse, would yield the elementary solution

$$\bar{Q} = (\Phi^T\Phi)^{-1}\,\Phi^T\,[X + \delta X].$$

If the row vectors of $\Phi^T\Phi$ are not fully linearly independent, then $\Phi^T\Phi$ will be nearly singular, and the above inversion will greatly magnify the error, possibly driving it to dominate the solution. Therein lies the basis for ill-conditioned nature of the problem.

Often an infinite number of joint-configurations can be obtained to satisfy a given end-effector configuration. However, it can be shown [58] that the infinity

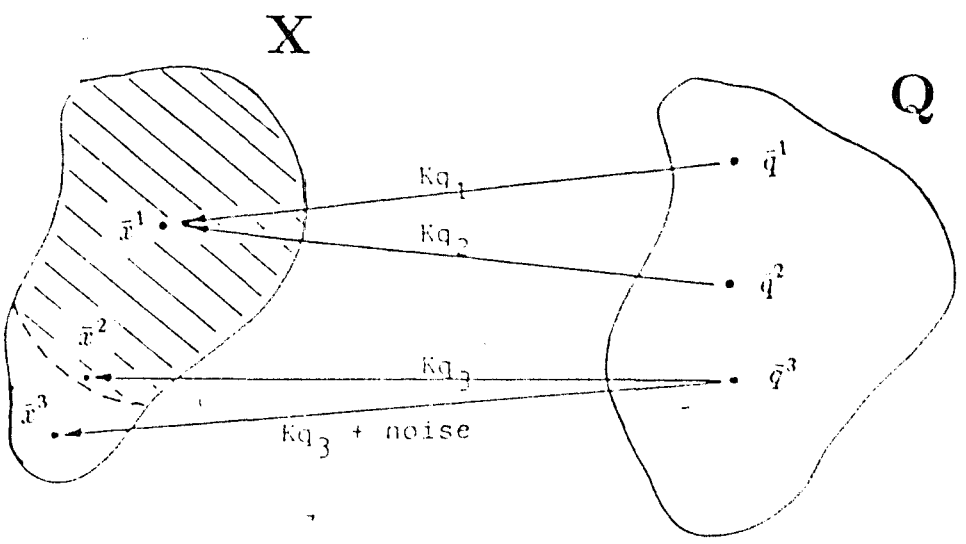


Figure 2.3.2.1.1. Schematic representation of inverse problems and the origin of their Ill-posed and ill-conditioned nature. (adapted from ref: [149]).

of solutions can be mapped into a finite set of manifolds. Because of this infinity of solutions, many redundant manipulator investigators have chosen to focus on the instantaneous or differential kinematics, which uses a jacobian-matrix to relate end-effector velocities to the joint velocities. The jacobian is defined as

$$\dot{\bar{x}} = \mathbf{J}(\bar{q})\,\dot{\bar{q}} \qquad (2.3.2.1.2)$$

For redundant robots the manipulator Jacobian is not uniquely invertible, and pseudo-inverse techniques can be used to select a solution from the infinity of possible solutions in the null space of $J(\bar{q})$. Eq. (2.3.2.1.2) is often referred to as the inverse kinematics solution, although (2.3.2.1.1) is the true inverse kinematics problem.

Given a desired end-effector velocity, $\dot{\bar{x}}$, the joint velocities, $\dot{\bar{q}}$ can simply be determined by:

$$\dot{\bar{q}} = \mathbf{J}^{\dagger}(\bar{q})\dot{\bar{x}} \qquad (2.3.2.1.3)$$

where $\mathbf{J}^{\dagger}(q)$ is the pseudo-inverse, or a weighted pseudo-inverse, of the manipulator Jacobian matrix. This redundancy resolution solution minimizes a weighted quadratic norm of instantaneous joint velocities. The end-effector velocities are typically generated by a path planning algorithm, and the joint velocities computed by Eq. (2.3.2.1.3) are used as the reference input to a joint-space control system.

This solution can be modified by adding a null space component to the joint velocities [66,78,184,288]:

$$\dot{\bar{q}} = \mathbf{J}^{\dagger}(\bar{q})\dot{\bar{x}} + (I - \mathbf{J}^{\dagger}(\bar{q})\mathbf{J}(\bar{q}))\bar{z} \qquad (2.3.2.1.4)$$

where $\bar{z}$ is an arbitrary vector. The term $(I - \mathbf{J}^{\dagger}(\bar{q})\mathbf{J}(\bar{q}))$ projects this arbitrary vector into the null space of the manipulator. Physically, any motion in the null space is an instantaneous internal motion of the manipulator which causes no motion of the end-effector. Many redundancy resolution criteria can be developed as potential functions, and $\bar{z}$ might be the gradient of the resolution potential function, i.e., , $\bar{z} = \alpha\nabla\Psi(\bar{q})$, where α is a weighting factor. Then for a given end-effector configuration, the gradient of this function is used to control joint velocity in the redundant directions, in a manner that forces the manipulator to seek an optimal configuration. However, the pseudo-inverse resolution techniques are generally not cyclic [78,92,256,310], i.e. these techniques do not generate closed joint-space trajectories corresponding to closed end-effector trajectories, thereby posing a serious limitation for practical implementations. Other researchers have used the null space of the jacobian, which corresponds to the self-motion of the robot, to optimize various performance criteria. For

example, Liegeois [184] has developed a gradient projection scheme that utilizes the null space of the Jacobian to optimize a joint-position dependent, scalar performance criterion. However, the existing algebraic methods are, in general, very expensive computationally, and are unable to find global redundancy resolution optima with respect to multiple criteria in real-time. Also the manipulators can have more than one distinct internal motion for a given end-effector location but the instantaneous methods only optimize over one internal motion, and therefore can miss the true optimum which lies on another internal motion [58].

So in the absence of closed form solutions, off-line iterative approximation techniques based on "local-methods" have been used to solve the inverse transformation problem. In this context, Goldenberg et al. [92] have proposed an "augmented task method" that uses a modified Newton-Raphson method to simultaneously obtain all the joint variables. They partition the augmented Jacobian matrix into an invertible non-redundant component and a redundant component to obtain approximate bounds on the magnitude of the joint angles. A nonlinear constrained optimization is then performed to determine the angular displacements for the redundant joints by satisfying some auxiliary criteria. The resulting values are used to compute the Newton-Raphson correction that minimizes an error-residual between desired and current end-effector coordinates. Despite its versatility, this techniques suffers from algorithmic singularities, since it fails to ensure the non-singularity of the Jacobian-matrix partition prior to start of each iteration. Also, for a large number of degrees of freedom, the nonlinear optimization algorithm during each iteration induces a significant computational complexity.

In a significantly different approach Burdick [58] conducts a topological and geometrical analysis of the kinematics of redundant manipulators. Formulating inverse kinematics as a global manifold mapping problem, he uses the singulari-

ties of the forward kinematics to partition the configuration space manifold into disjoint regions. The topological characteristics of these regions and their forward mapping are then used to rigorously analyze kinematic properties, such as bounds nature and number of singularities that must be encountered along an arbitrary cyclic path and bounds on the number of inverse kinematic solutions. Currently formal procedures are being developed for translating this qualitative insight to quantitative algorithms that could aid the design and control of redundant manipulators.

In contrast to the algebraic and iterative strategies mentioned above, neuromorphic approaches to the inverse kinematics problem entail systems composed of many simple processors ("neurons"), fully or sparsely interconnected, whose function is determined by the topology and strength of the interconnections. The synaptic elements of such neural systems must capture the transcendental kinematic transformations using *a priori* generated examples enabling subsequent generalization to other points in the workspace. Thus, the inverse transformation equations do not need to be explicitly programmed or derived. Once they have been learned, the network's inherent self-organizing abilities enable it to adapt to changes in the environment, e.g. planning joint trajectories in the presence of obstacles, or to any unforseen changes in the mechanical structure of the manipulator, with little effort [197]. Within a neuromorphic framework, a solution of the inverse kinematic involves two phases, a training phase and a recall phase. The training phase involves encoding the inverse mapping in the network's synaptic weight space, through repeated presentations of a finite set of *a priori* generated examples, linking cartesian space end-effector coordinates to the corresponding joint angles. Once the network has acquired the nonlinear mapping imbedded within the training set, it can be used to rapidly recall, or generalize the joint configuration corresponding to any given cartesian-space orientation within it's workspace of training, thereby eliminating the computational overheads asso-

ciated with the existing iterative techniques. Also, once the training cycle is completed, the time required to obtain a solution depends in a weak fashion on the number of degrees of freedom.

In the past, Josin [152], Guez [102] and Tawel et al. [267] have applied this generic neuromorphic paradigm to the inverse kinematics problem for a 3-DOF redundant manipulator. In particular, they train a heteroassociative, multi-layered feed-forward neural network using the backpropagation algorithm [250,251]. The following principle is commonly used during the training process. When the system produces a wrong output on presentation of an I/O pair, the learning update rule simply changes each weight in the direction which makes the size of the error decrease as rapidly as possible. The components of this steepest descent direction in weight space are evaluated by using the chain rule to compute the partial derivatives of an error function with respect to each weight. The implementation of this weight change requires recursively propagating an error signal backward through the network, changing weights that had a large effect on the output more than those that did not. This process is repeated until the residual error between the network and target output, over all patterns, falls below a minimum acceptable tolerance.

Despite its conceptual simplicity, there are a number of non-trivial issues, both from the kinematics perspective and from the computational cost perspective that have hitherto limited the efficacy of such neuromorphic solutions to the inverse kinematics problem for redundant motion control. The major limitations, as discerned from the existing implementations, include an unacceptably large number of training iterations ($O(10^6)$ even for generalizing over small manifolds, see Tawel et al [267). Also the interpolated angular coordinates have relatively poor precision as compared to their algebraic or iterative counterparts. Besides, the backpropagation algorithm fails to efficiently scale-up to configura-

seven or more degrees of freedom could not be satisfactorily trained using the standard back-propagation algorithm even after several million iterations. Furthermore the back-propagation algorithm *per se* does not provide any intrinsic mechanism to simultaneously exploit redundancy to increase the task workspace (design constraints) and satisfy additional requirements inherent to operations in an unstructured environment such as obstacle avoidance in real-time.

2.3.2.2. Implementation Results

The neurodynamical learning framework, SID_2, developed in the preceding section 2.2. was applied to a planar 3-degree of freedom redundant manipulator and to a spatial 7-degree of freedom human-arm like manipulator. Though either of the manipulators encompasses configurations which exhibit sufficiently the problematic complexity presented by the inverse kinematics mapping, we experimented with both examples for a variety of reasons. The 3-DOF planar manipulator was primarily used to ascertain the algorithmic correctness of our terminal-attractor-based neural learning algorithm. It provided benchmarks for comparison with the existing backpropagation based neural network solutions [102,251,267], in terms of number of training iterations and training samples needed to stabilize the network, estimates on the number of synaptic elements or neurons required to successfully capture the inverse mapping and the accuracy of recalled or "generalized" joint configurations corresponding to the input end-effector coordinates. Though backpropagation-based neural networks can be applied to planar redundant manipulators with 3-DOF, they failed to scaleup to cases involving seven or more joints. Their practical applicability would thus appear to be severely limited for kinematic control of most industrial robot manipulators. We also empirically analyzed the robustness and computational speed

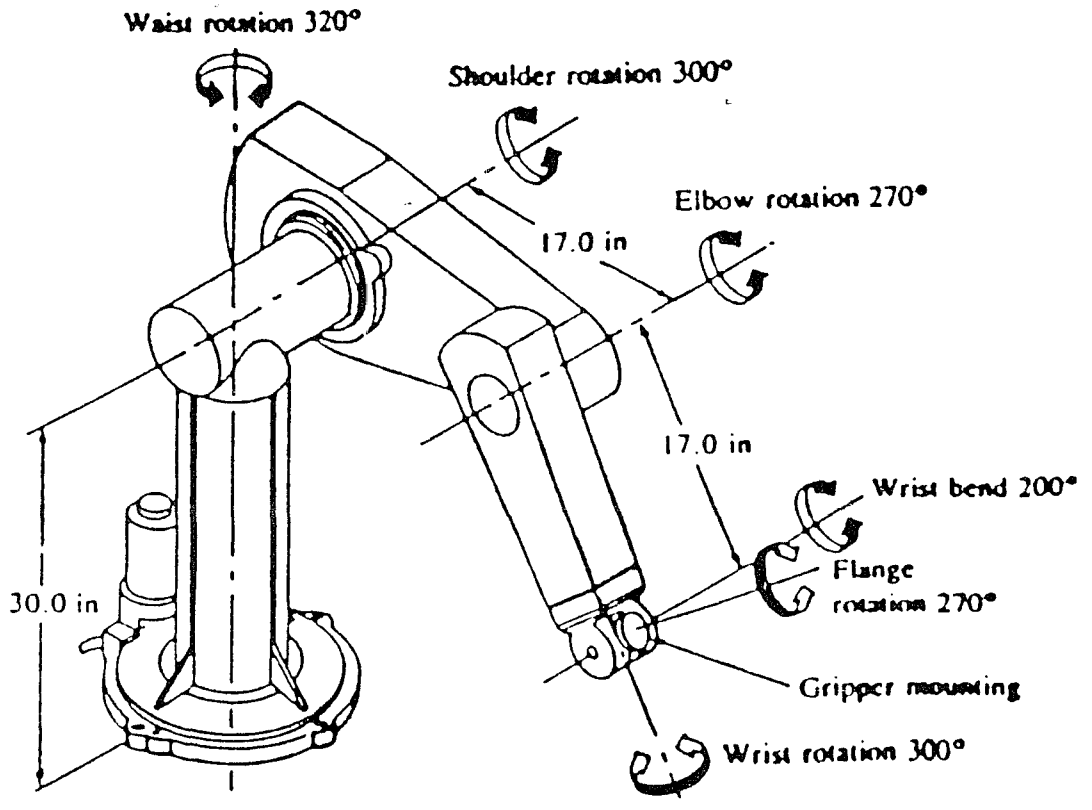
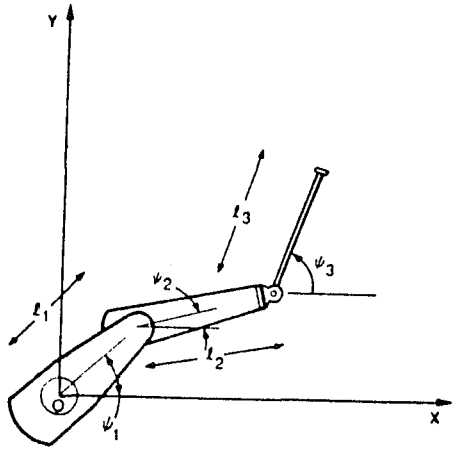


Figure 2.3.2.2.1. (a) Arm configuration for a PUMA 560 Robot;



(b) PUMA 560 Robot Arm constrained to coplanar configurations.

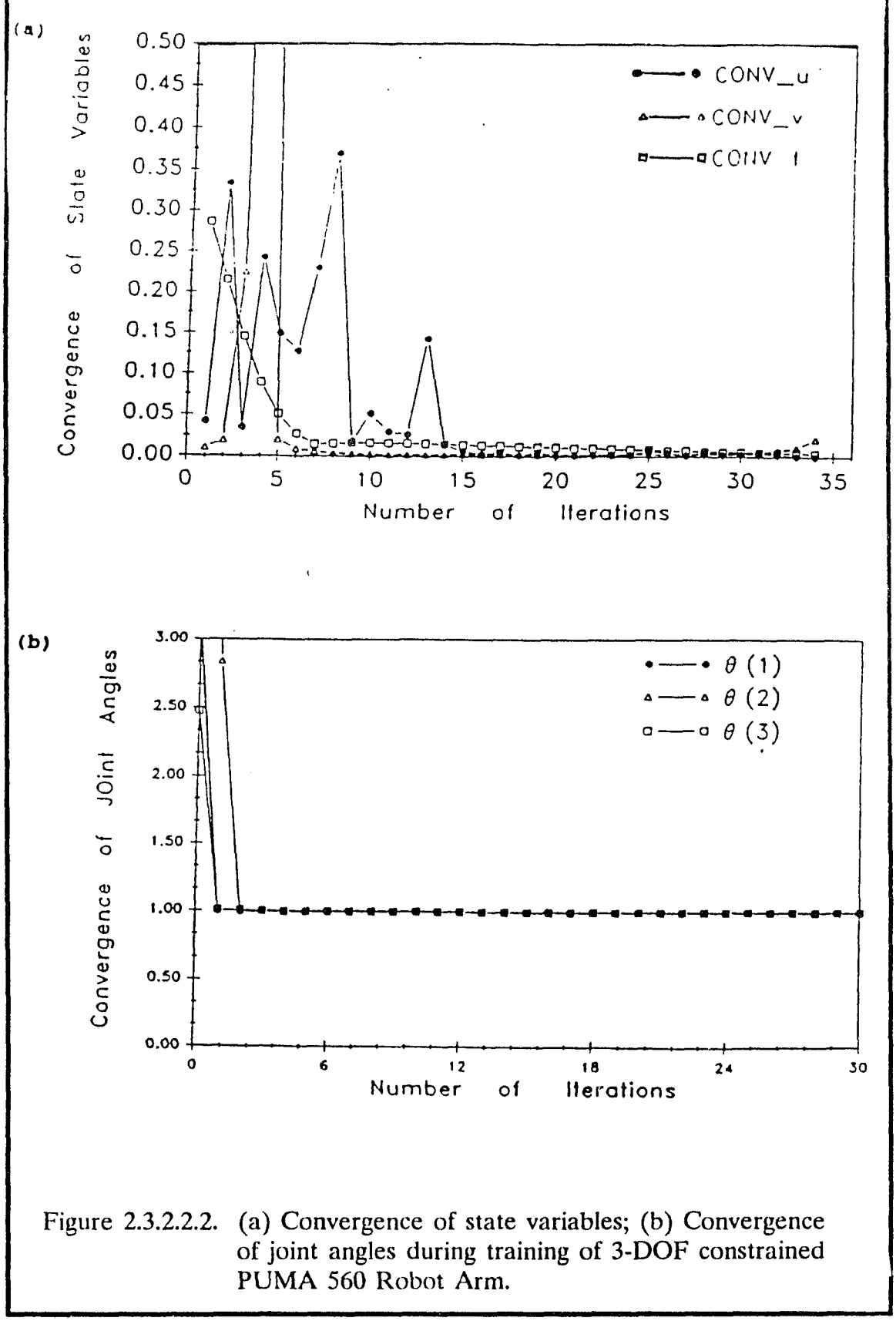


Figure 2.3.2.2.2. (a) Convergence of state variables; (b) Convergence of joint angles during training of 3-DOF constrained PUMA 560 Robot Arm.

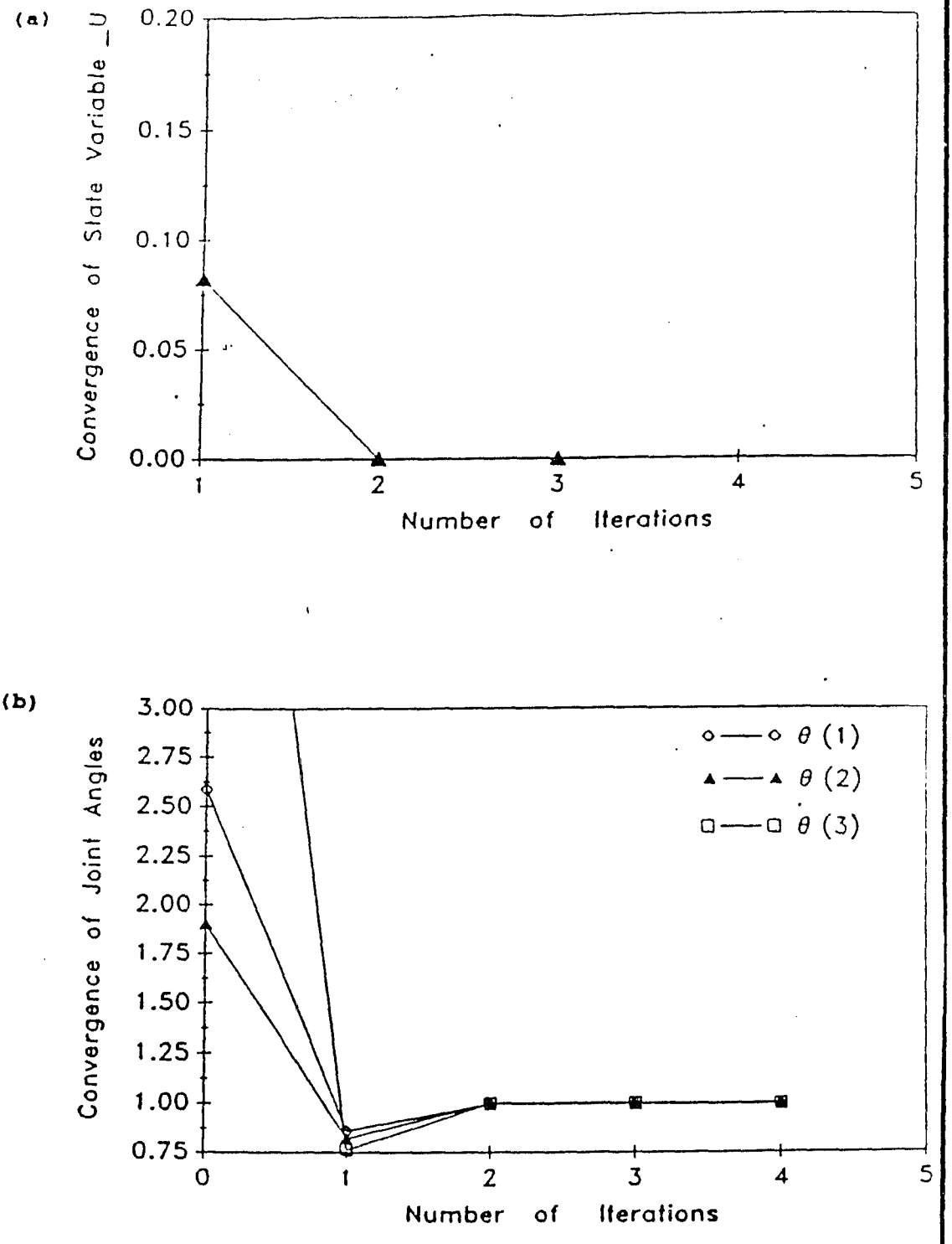


Figure 2.3.2.2.3. (a) Network activity; (b) Normalized behavior of output neurons during recall, in the case of 3-DOF constrained PUMA 560 Robot Arm.

of our artificial neural system on a 7-DOF redundant manipulator to illustrate its efficacy as a viable, real-time alternative to the existing quantitative techniques [30,107]. To provide a context for the subsequent analysis we precede our discussion on the simulation results with a brief description of each candidate manipulator.

The continuous-time, dynamical training procedure was simulated using parameters corresponding to a constrained configuration of the six-jointed PUMA 560 industrial robot [255]. By suppressing the motion of the shoulder, elbow and the wrist joints, the PUMA robot arm, Fig. 2.3.2.2.1(a) was restricted to motion in a vertical plane only as shown in Fig. 2.3.2.2.1(b) Fig. 2.3.2.2.2(a) illustrates the worst-case normalized behavior of the state variables and the synaptic elements during the learning phase, as the neural network acquires the inverse mapping. Figure 2.3.2.2.2(b) shows the convergence of the output neurons to the presented attractors during the learning phase for a particular sample. When learning has stabilized over all the training samples the network switches to its operational mode. Figure 2.3.2.2.3(a) displays the normalized convergence behavior of the neuronal activity, $\bar{u}$ as the network generalizes the joint angles in response to arbitrary cartesian inputs. Figure 2.3.2.2.3(b) illustrates the convergence of interpolated joint angles. Notice the rapid rate of convergence in computing the joint configuration as juxtaposed to conventional techniques.

In addition, the dynamical training algorithm was applied to learn the inverse kinematics transformations for Hemami's simplified 7-DOF manipulator formulation [122] of the human-arm. As shown in Figure 2.3.2.2.4, the following joint motions are available :the joint θ_1 provides back and forth motion about the shoulder, θ_2 provides effector elevation in the vertical plane, θ_3 enables rotation around the upper-arm axis, while θ_4 provides the elbow motion, θ_5 is around the forearm axis and θ_6 and θ_7 lead to the pitch and yaw motions of the wrist,

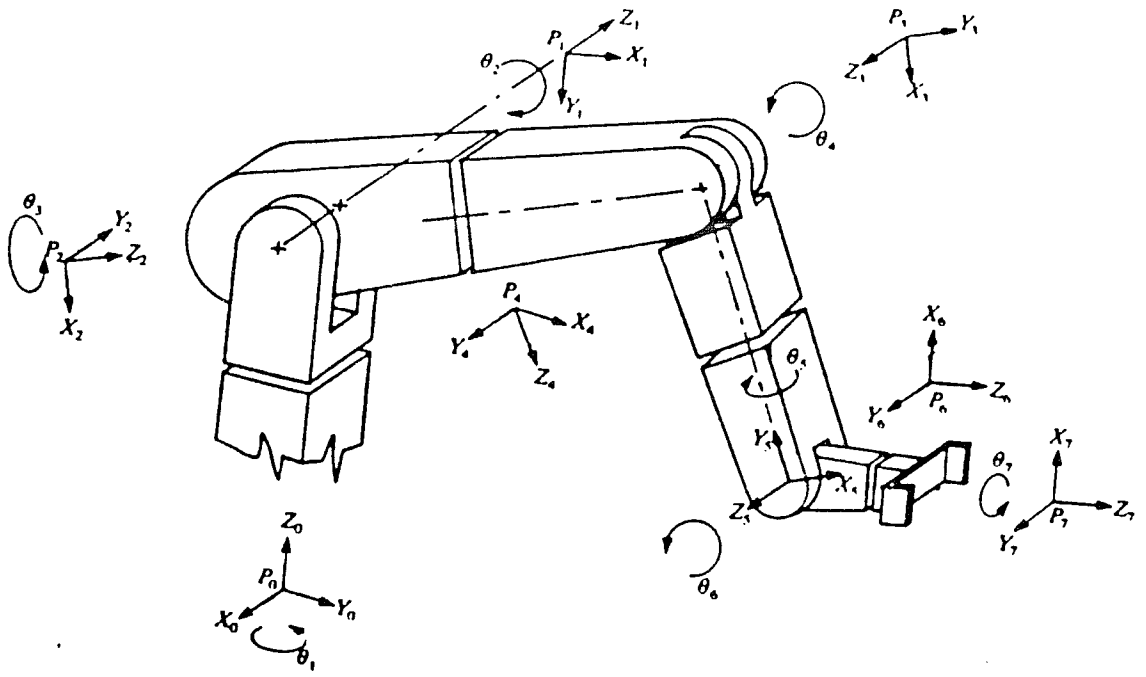
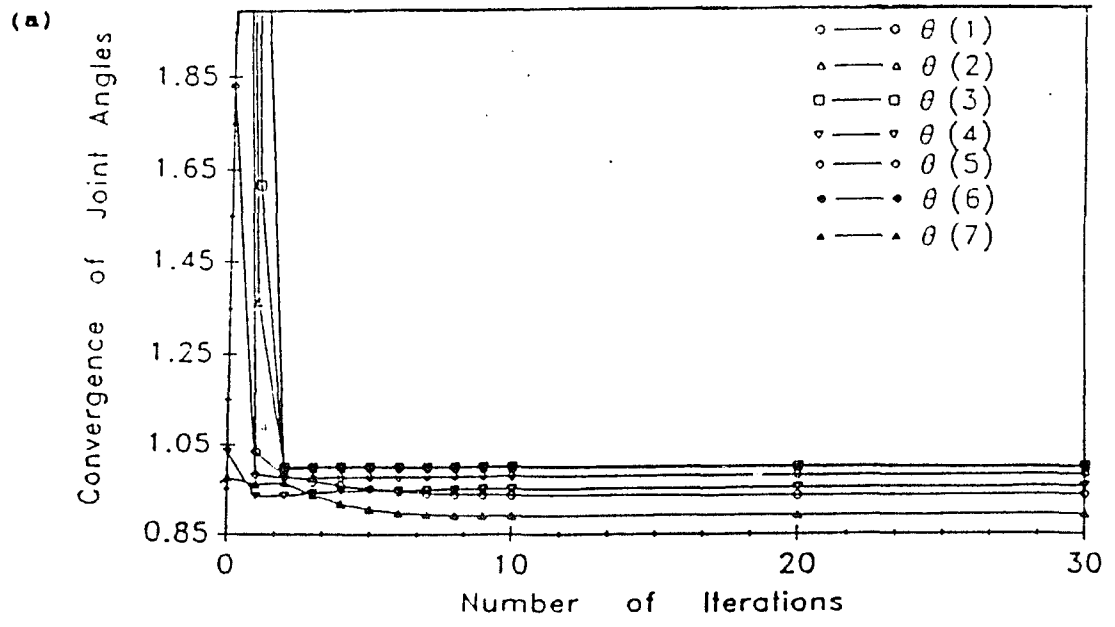


Figure 2.3.2.2.4. Seven DOF Human-arm like manipulator. Adapted from Hemami [122].



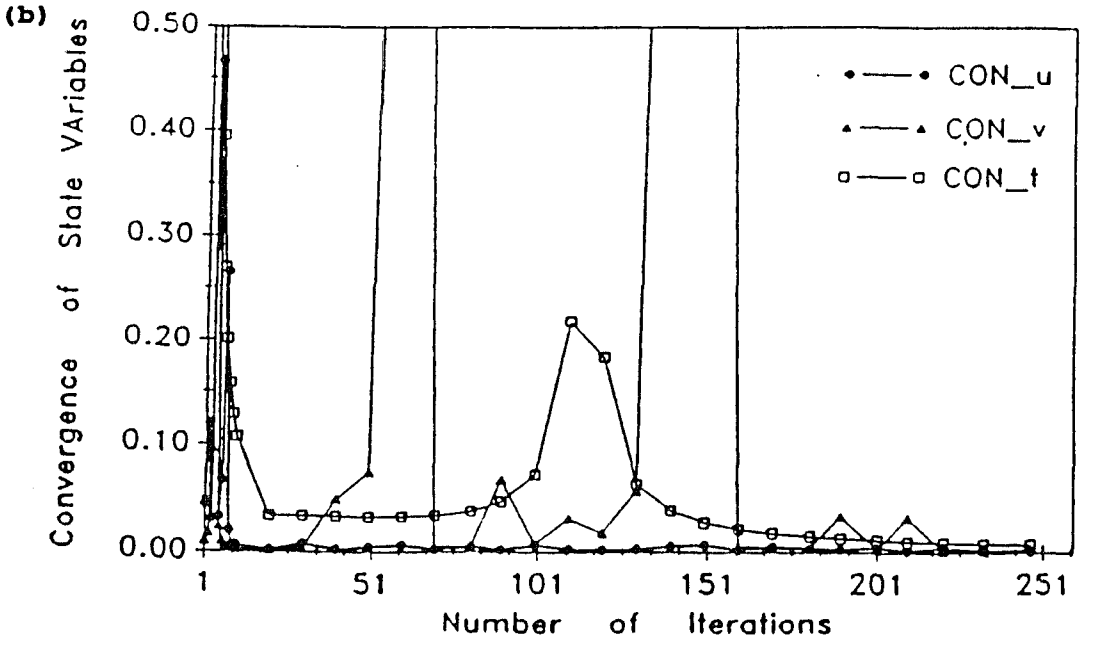


Figure 2.3.2.2.5. (a) Normalized behavior of state variables; (b) Convergence of joint angles, during training phase for a 7-DOF redundant manipulator.

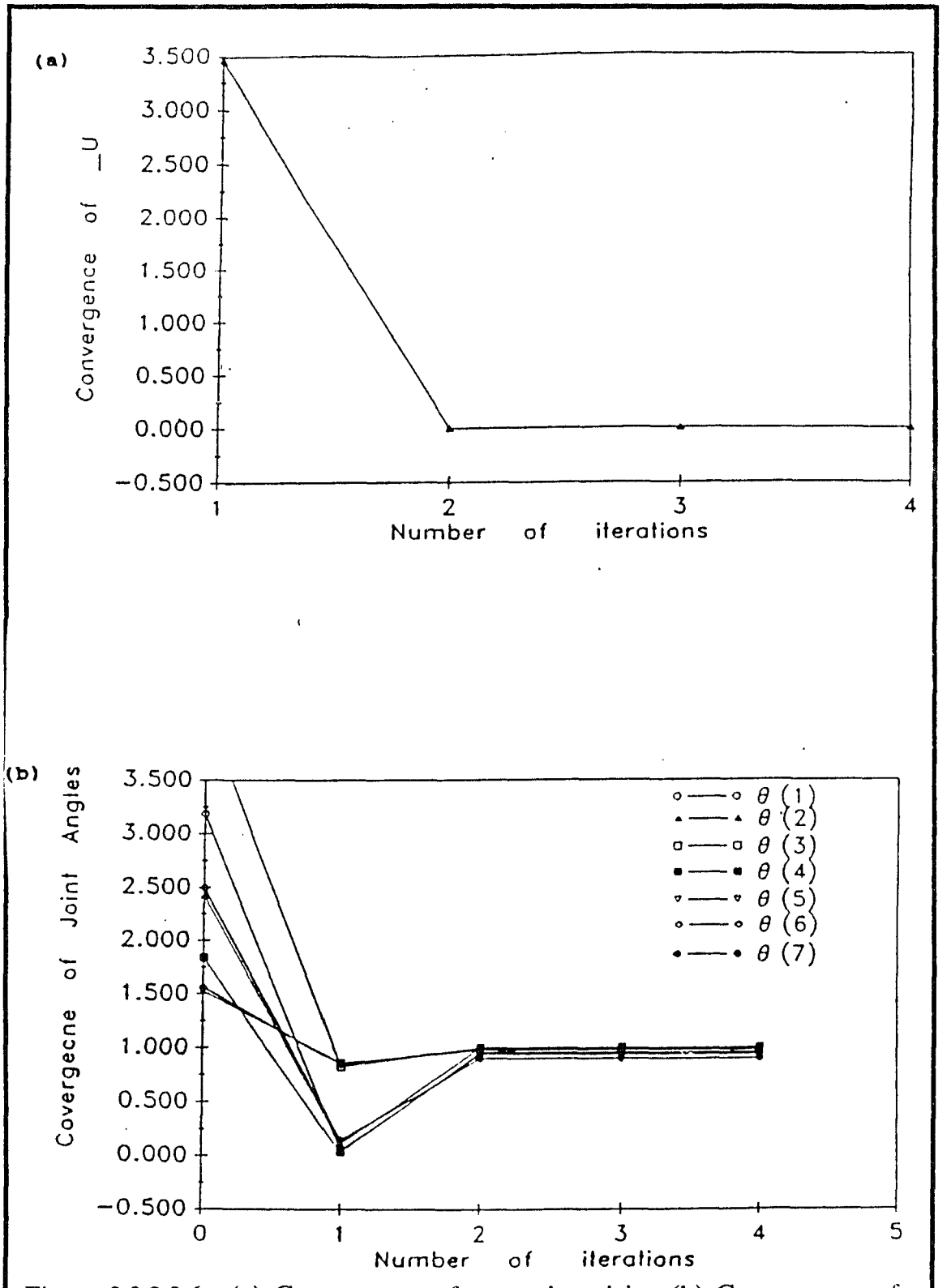


Figure 2.3.2.2.6. (a) Convergence of network activity; (b) Convergence of joint angles, during operational phase for a 7-DOF redundant manipulator.

respectively. Details on the geometric parameters, namely link length, twist angle, joint limits and offsets may be found in [122]. Forward kinematics equations (see Paul [233]), were used to generate training samples of end-effector and joint-space coordinates over the workspace volume of the robot. Figure 2.3.2.2.5(a) displays the normalized, worst-case temporal behavior of state variables $\bar{u}$, i.e., $\max_{\forall\, n} \mid (u_n^{t+1} - u_n^t)/u_n^t \mid$, adjoint variables $\bar{v}$ and the synaptic elements, $\hat{T}$, during the learning phase. Figure 2.3.2.2.5(b) shows the variation in activity at the output neurons during the learning cycle corresponding to one of the training pairs being presented to the network. Note that the system learns the inverse mapping in a few hundred iterations only, as compared to the several million iterations required by the gradient descent-based backpropagation algorithm. The computational efficacy of our neuromorphic learning algorithm may be estimated from the rate of variation in network activity during the operational phase of the network as shown in figures 2.3.2.2.6(a) and 2.3.2.2.6(b) As depicted in Fig. 2.3.2.2.6(a), once the network has acquired the inverse transformations, it may be used to recall or generate the joint angles needed to achieve any arbitrary end-effector coordinate, within the workspace of the manipulator, in very few dynamical iterations. The detailed results of this study will be reported elsewhere.

2.4. Summary

In this chapter we have provided a novel framework for solving a class of complex learning problems in the context of robot manipulation, namely the enhancement of manipulative capability and reliability. Our novel learning paradigm for neural network models, based on the terminal attractor concept, is shown to be computationally competitive with iterative methods currently used

in robotics to solve the inverse kinematics of redundant manipulators. The neuromorphic framework is expected to facilitate the development of robust real-time algorithms for computing joint configurations to achieve arbitrary end-effector trajectories. In addition, this strategy does not appear to suffer from non-cyclicity of motion, as encountered in the pseudo-inverse resolution techniques [58,66,78,92,288] or the algorithmic singularities common to augmented task approaches [92]. Furthermore, unlike the feed forward, backpropagation neural learning approaches, the adaptive dynamical system formulation presented here, provides the flexibility for incorporating arbitrary combinations of kinematic optimization criteria, without imposing high computational overheads. Two options are available for including the redundancy resolution criteria in the algorithm to resolve the nonuniqueness of joint configurations that may satisfy a given end-effector configuration. The constraints may either be included *a priori* , i.e., while generating the training samples themselves, thereby forcing the network to learn only limited aspects of inverse kinematics mapping with a bias towards a particular criterion; or they could be selectively applied in real-time to an operational version of the network (trained to encode the emergent invariants of the inverse kinematic mapping), to regularize the solutions (i.e. provide unique best answers). In addition, it was found that this strategy scales-up to configurations of practical interest, where conventional neural learning techniques, e.g., back propagation appear to fail.

Chapter Three

Constrained Learning in Dynamical Neural Networks

In this chapter we extend our previous results to rederive a theoretical framework for neural learning of nonlinear mappings, wherein both the topology of the network and synaptic interconnection strengths are evolved adaptively. The proposed methodology exploits a new class of mathematical constructs, *terminal attractors* (detailed in Chapter Two), which provide unique information processing capabilities to artificial neural systems. Terminal attractor representations are used not only to ensure infinite local stability of the encoded information, but also to provide a qualitative as well as quantitative change in the nature of the learning process. In particular, the loss of Lipschitz conditions at energy function minima results in a dramatic increase in the speed of learning. Typical performance improvements are in excess of three orders of magnitude over current state-of-the-art backpropagation techniques. To guarantee the unconditional stability of the neural activation dynamics during learning, we introduce the concept of "virtual terminal attractors". Finally, in a significant departure from prior neuromorphic formulations our algorithms also provide a framework for systematically incorporating both *in-training* and *a posteriori* regularization mechanisms to handle design as well as environmental constraints for applications in unstructured environments in real-time.

3.1. Introduction

A considerable effort has recently been devoted to the development of efficient computational methodologies for learning. Artificial neural networks, char-

acterized as massively parallel, coupled, adaptive dynamical systems, provide an ideal framework for interacting with objects of the real world and its statistical characteristics in the same manner as biological systems do. In contrast to existing notions on imperative and symbolic computing, the potential advantages of neuronal processing stem from their ability to perform concurrent, asynchronous and distributed information processing. Neurons with simple properties and interacting within relatively simple architectures can accomplish collectively complex functions such as generalization, error correction, information reconstruction, pattern analysis and learning. Their paradigmatic strength for potential applications arises from their spontaneous emergent ability to achieve *functional synthesis*, and thereby learn *nonlinear mappings* [32,36,111], and abstract spatial [62,63], functional [180] or temporal [177,178] invariances of these mappings. Thus, relationships between multiple continuous-valued inputs and outputs can be established, based on a presentation of a large number of *a priori* generated representative examples. Once the underlying invariances have been learned and encoded in the topology and strengths of the synaptic interconnections, the neural network can generalize to solve arbitrary problem instances. Since the mappings are acquired from real-world examples, network functionality is not limited by assumptions regarding parametric or environmental uncertainty, inherent to model-based approaches. Thus, neural networks provide an attractive self-organizing algorithmic paradigm, that can can automatically learn like biological systems, rather than require explicit programming or symbolic search.

Fundamental to functional synthesis is the ability to accurately and efficiently acquire nonlinear transformations from examples. Although, a number of neural algorithms have been proposed for functional approximation, attention has largely focussed on the back-propagation algorithm because of its simplicity, generality and the promise that it has shown in regard to various applications. However, the increasing perception that back-propagation is too slow to be rel-

evant to most real-world problems, has led to the development of a number of variant algorithms. For discrete systems, Baum [46] has proposed a polynomial time algorithm for learning union of half spaces. Lapedes and Farber [177,178] proposed a master-slave network with sigmoidal nonlinearities to approximate a continuous time series for forecasting. Pineda [238] extended the methodology by deriving a recurrent generalization to back-propagation networks operating in continuous time. In a similar vein, Pearlmutter [234] constructed a procedure for approximating trajectories by minimizing an error functional between output and targeted temporal trajectories. More recently, Williams and Zipser [292] proposed a real-time learning algorithm for training recurrent, continually updated networks to handle temporal tasks.

In a radically different approach, we propose to use a new mathematical construct, i.e., terminal attractor dynamics [303] to acquire the nonlinear mapping. Terminal attractor representations are used not only to ensure infinite local stability of the encoded information, but also to provide a qualitative as well as quantitative change in the nature of the learning process. In particular, the loss of Lipschitz conditions at energy function minima results in a dramatic increase in speed of learning. Typical performance improvements are in excess of three orders of magnitude over current state-of-the-art backpropagation techniques. In a significant departure from prior neuromorphic formulations our algorithms also provide a framework for systematically incorporating event-driven constraints in real-time, avoiding the necessity to retrain the network. Finally, a fundamental problem in neural learning methodologies based on dynamical systems concerns the stability of the activation network as synaptic weights evolve during training. Previous approaches [234,251,305] do not guarantee stability. Here, we introduce the concept of "virtual" terminal attractors which yields an unconditionally stable neurodynamics.

In this context, we present a novel self-organizing neural formalism, which attempts to provide an efficient and accurate solution to the inverse kinematics problem and addresses some of the above concerns. The proposed methodology extends our work in Chapter Two [30,34,108,110], wherein we introduced topographically partitioned, but fully connected networks to acquire the kinematics mapping. Central to our approach is the concept of encoding the training samples as static "terminal-attractors" of the network.

Chapter Two focussed on coupling the mapping encoding and the resolution of kinematic redundancy in an objective function from which the learning equations were derived. Here, we argue a radically different approach, wherein redundancy resolution is carried out at the operational stage. Training now essentially aims at capturing the invariant properties of the nonlinear kinematic mapping, by minimizing the network's "strength" energy, a regulator of the interconnection topology and synaptic strengths. Specifically, this chapter provides a new theoretical framework for learning using artificial neural networks.

3.2. Neurodynamics Model

3.2.1. Network Specification

Consider a densely connected neural network with N graded-response neurons operating in continuously sampled time. To acquire a nonlinear transformation, ζ, from a $\aleph_X$-dimensional input domain to the $\aleph_Y$-dimensional output space, the network is topographically partitioned into three mutually exclusive regions. As shown in Fig. 2.2.1.1, the partition refers to a set of input neurons, S_X, that receive the input components, an output set S_Y, which provides the desired output components and a set of "hidden" neurons, S_H, that encode the representation of the ζ-mapping. The network is presented with K randomly sampled training vector-pairs of input- and output-space coordinates.

We formalize the neural network as an adaptive dynamical system whose temporal evolution is represented by the following coupled differential equations

$$\dot{u}_n \; + \; \kappa \, u_n \;\; = \;\; \sum_m T_{nm} \; \varphi_\gamma(u_m) \;\; + \;\; {}^k I_n \qquad (3.2.1.1)$$

where u_n represents the *mean soma potential* of the nth neuron and T_{nm} denotes the synaptic coupling from the mth to the nth neuron. The constant κ characterizes the decay of neuron activity. The sigmoidal function $\varphi_\gamma(\cdot)$ modulates the neural response, with gain given by γ; typically, $\varphi_\gamma(z) \;=\; \tanh(\gamma \cdot z)$. Without loss of generality, γ will be set to unity in the sequel. The "source" term, ${}^k I_n$ encodes component-contribution by the attractors of the k-th training sample via the expression

$${}^k I_n \;\; = \;\; \begin{cases} [{}^k a_n \; - \; \varphi(u_n)\,]^\beta & \textbf{if } n \; \in \; S_X \\ 0 & \textbf{if } n \; \in \; S_H \cup S_Y \end{cases} \qquad (3.2.1.1)$$

The specific attractor coordinates, ${}^k a_n$, are given by ${}^k x_n$ if $n \;\in\; S_X$ and ${}^k y_n$ if $n \;\in\; S_Y$, for $\{\, {}^k\bar{x},\; {}^k\bar{y} \;\mid\; k \;=\; 1,\, \cdots,\, K\}$ taken from a training set scaled to the range $[-1, +1]$. In Section 2.2.4, it was shown that, for $\beta \;=\; (2i+1)^{-1}$ and i a strictly positive integer, such attractors have infinite local stability and provide opportunity for learning in real-time.

3.2.2. Energy Function and Network Stability

Our basic operating assumption for the dynamical system defined by Equations (3.2.1.1) is that at equilibrium, i.e., as $\dot{u}_n \;\rightarrow\; 0$, for $n \;=\; 1, \cdots,\; N$,

$$u_n \;\; \rightarrow \;\; {}^k\tilde{u}_n(T). \qquad (3.2.2.1)$$

The superscript $\sim$ will be used to denote quantities evaluated at steady state. This yields the fixed point equations :

$$\kappa \; {}^k\tilde{u}_n \;\; = \;\; \sum_m T_{nm} \; \varphi({}^k\tilde{u}_m) \; + \; {}^k\tilde{I}_n \qquad (3.2.2.2)$$

Note that, in contradistinction to Hopfield [135], Pineda [235], and others [75] ${}^{k}I_n$, is a function of the state variable u_n and does not represent a constant external input bias to the network. It influences the system's degree of stability and provides a dynamically varying input modulation to the neuron, thereby enforcing convergence to fixed points in finite time, without affecting the location of existing static attractors. For an arbitrary synaptic matrix T, the asymptotic attractor contribution ${}^{k}\tilde{I}_n$ differs from zero. The key objective of learning is then to

> *adaptively evolve the interconnection topology of the neural network; and determine the synaptic strengths, so that the* $S_X \longrightarrow S_Y$ *mapping be accurately computed over the training set, in terms of the specified attractors;* i.e., $\forall\, k = 1, \cdots, K$
>
> $$ {}^{k}\tilde{I}_n = 0 \qquad \forall\ n \in S_X \cup S_Y $$

To proceed formally with the development of a learning algorithm, we propose an approach based upon the minimization of a constrained "neuromorphic energy-like function" $E(T, \lambda)$ given by the following expression

$$ E(T,\lambda) = \frac{1}{2} \sum_n \sum_m \omega_{nm} \left(T_{nm}^2 - T_{nm}\, T_{mn} \right) + \frac{1}{\alpha} \sum_k \sum_n {}^{k}\lambda_n\, {}^{k}\Gamma_n^{\alpha} \quad (3.2.2.3) $$

where

$$ {}^{k}\Gamma_n = \begin{cases} {}^{k}a_n - \varphi({}^{k}\tilde{u}_n) & \text{if } n \in S_X \\ 0 & \text{if } n \in S_H \cup S_Y \end{cases} \quad (3.2.2.4) $$

Typically, positive values like $\frac{4}{3}$ and 2 are used for α. The weighting factor ω_{nm} is constructed in such a fashion as to favor locality of computation. The indices n, m span over all neurons in the network. Lagrange multipliers corresponding to the $k-n$th constraint are denoted by ${}^{k}\lambda_n$. The proposed objective function includes contributions from two sources.

[a] It enforces convergence of every neuron in S_X and S_Y to attractors corresponding to the components in the input-output training samples, thereby prompting the network to learn the underlying functional invariances.

[b] It regulates the topology of the network by minimizing interconnection strengths between distant synaptic elements in line with Gauss's least constraint principle [107].

As already discussed in Chapter Two, additional problem-specific constraints could also be incorporated in the neuromorphic energy function. But, in contradistinction to the traditional approaches, our methodology incorporates them directly into the trained (operational) network, as discussed in Section 3.5

Lyapunov stability requires an energy-like function to be monotonically decreasing in time. Since in our model the internal dynamical parameters of interest are the synaptic strengths T_{nm} of the interconnection topology and the Lagrange multipliers ${}^{k}\lambda_n$, this implies that

$$\dot{E} \;=\; \sum_i \sum_j \frac{\partial E}{\partial T_{ij}}\, \dot{T}_{ij} \;+\; \sum_l \sum_i \frac{\partial E}{\partial {}^{l}\lambda_i}\, {}^{l}\dot{\lambda}_i \;<\; 0 \qquad (3.2.2.5)$$

One can always choose, with $\tau > 0$

$$\dot{T}_{ij} \;=\; -\tau\, \frac{\partial E}{\partial T_{ij}} \qquad (3.2.2.6)$$

where τ introduces an adaptive parameter for learning to be specified in the sequel. Then, substituting in Eq. (3.2.2.5) and denoting by $\oplus$ tensor contraction, i.e., sum over all relevant indices, one obtains

$$\nabla_\lambda E \;\oplus\; \dot{\lambda} \;<\; \tau\, \nabla_T E \oplus \nabla_T E \qquad (3.2.2.7)$$

The equations of motion for the Lagrange multipliers ${}^{l}\lambda_i$ must now be constructed in such a way that Eq. (3.2.2.7) be strictly satisfied. In addition, when the

constraints are satisfied, i.e., ${}^{l}\tilde{I}_n \rightarrow 0$, we require that ${}^{l}\dot{\lambda}_i \rightarrow 0$. We have adopted the following analytical model for the evolution of λ,

$$ {}^{l}\dot{\lambda}_i = \tau \frac{\nabla_T E \oplus \nabla_T E}{\Lambda + 1/(\Lambda + \theta)} {}^{l}[\nabla_\lambda E]_i \qquad (3.2.2.8) $$

where $\Lambda = \nabla_\lambda E \oplus \nabla_\lambda E$ and $0 < \theta \ll 1$. It is straightforward to prove that this model fulfills the above requirements.

3.2.3. Adaptive Learning

We now focus on the derivation of an algorithm for computing $\nabla_T E$ and $\nabla_\lambda E$. An adiabatic framework is assumed. On differentiating Eqs. (3.2.2.3) with respect to T_{ij} we get

$$ \frac{\partial E}{\partial T_{ij}} = \omega_{ij}\, T_{ij} - \sum_k \sum_n {}^{k}\lambda_n \, {}^{k}\Gamma_n^{\alpha-1} \, {}^{k}\hat{\varphi}_n \frac{d}{dT_{ij}} {}^{k}\tilde{u}_n. \qquad (3.2.3.1) $$

where ${}^{k}\hat{\varphi}_n$ denotes the derivative of the neural response. We must compute $\frac{d}{dT_{ij}} {}^{k}\tilde{u}_n$ from the network fixed point equations (3.2.2.2). This requirement is a major distinction from previous results for associative memory [13]. There, the constraints (${}^{k}\Gamma_n$ in our notation) were simply the fixed point equations for the *complete* $N-$dimensional memory patterns, i.e., $\varphi^{-1}(a_n^k) = \sum_m T_{nm}\, a_m^k$ (again in our notation). Such a representation and the resulting formalism are inadequate for learning nonlinear mappings ($S_X \rightarrow S_Y$) problems, since here $| S_H | \neq 0$.

We proceed as follows :

$$ \begin{aligned} \frac{d}{dT_{ij}} \kappa \, {}^{k}\tilde{u}_n &= \frac{d}{dT_{ij}} \left\{ \sum_m T_{nm}\, \varphi({}^{k}\tilde{u}_m) + {}^{k}\tilde{I}_n \right\} \\ &= \sum_m \delta_{ni}\, \delta_{mj}\, \varphi({}^{k}\tilde{u}_m) + \sum_m T_{nm}\, {}^{k}\hat{\varphi}_m \frac{d}{dT_{ij}} {}^{k}\tilde{u}_m + \frac{d}{dT_{ij}} {}^{k}\tilde{I}_n \end{aligned} \qquad (3.2.3.2) $$

which after some algebraic manipulation yields

$$\frac{d}{dT_{ij}}\,{}^k\tilde{u}_n \;=\; {}^k\left[A^{-1}\right]_{ni}\,\varphi({}^k\tilde{u}_j) \tag{3.2.3.3}$$

In the above expression, the matrix A is defined as

$${}^kA_{ni} \;=\; {}^k\eta_i\,\delta_{ni} \;-\; T_{ni}\,{}^k\hat{\varphi}_i \tag{3.2.3.4}$$

where

$${}^k\eta_n \;=\; \begin{cases} \kappa \;+\; \frac{1}{3}[a_n^k \;-\; \varphi({}^k\tilde{u}_n)]^{-2/3}\,{}^k\hat{\varphi}_n & \textbf{if } n \;\in\; S_X \cup S_Y \\ \kappa & \textbf{if } n \;\in\; S_H \end{cases} \tag{3.2.3.5}$$

To summarize the calculations up to this stage, we can write

$$\frac{\partial E}{\partial T_{ij}} \;=\; \omega_{ij}\,T_{ij} \;-\; \sum_k \sum_n {}^k\lambda_n\,{}^k\Gamma_n^{\alpha-1}\,{}^k\hat{\varphi}_n\,{}^k[A^{-1}]_{ni}\,\varphi({}^k\tilde{u}_j) \tag{3.2.3.6}$$

A computation of the energy gradient using Eq. (3.2.3.6) would involve a matrix inversion. Since direct matrix inversion is typically nonlocal, we use a variant of a relaxation procedure suggested by Pineda [10]. Consider the following change of variable

$${}^k\tilde{v}_i \;=\; \sum_n {}^k\lambda_n\,{}^k\Gamma_n^{\alpha-1}\,{}^k\hat{\varphi}_n\;\;{}^k[A^{-1}]_{ni} \tag{3.2.3.7}$$

Multiplying both sides of Eq. (3.2.3.7) by ${}^kA_{im}$ and summing over i yields

$$\sum_i {}^kA_{im}\,{}^k\tilde{v}_i \;=\; \sum_n {}^k\lambda_n\,{}^k\Gamma_n^{\alpha-1}\,{}^k\hat{\varphi}_n \sum_i {}^k[A^{-1}]_{ni}\,{}^kA_{im} \;=\; {}^k\lambda_m\,{}^k\Gamma_m^{\alpha-1}\,{}^k\hat{\varphi}_m \tag{3.2.3.8}$$

One can also directly use the explicit form of ${}^kA_{im}$ from (3.2.3.4) to obtain

$$\begin{aligned} \sum_i {}^kA_{im}\,{}^k\tilde{v}_i \;&=\; \sum_i {}^k\eta_m\,\delta_{im}\,{}^k\tilde{v}_i \;-\; \sum_i T_{im}\,{}^k\hat{\varphi}_m\,{}^k\tilde{v}_i \\ &=\; {}^k\eta_m\,{}^k\tilde{v}_m \;-\; {}^k\hat{\varphi}_m \sum_i T_{im}\,{}^k\tilde{v}_i \end{aligned} \tag{3.2.3.9}$$

Regrouping equations (3.2.3.8) and (3.2.3.9) results in the fixed point equation

$$ {}^{k}\eta_m \, {}^{k}\tilde{v}_m \;=\; {}^{k}\hat{\varphi}_m \, [\sum_i T_{im} \, {}^{k}\tilde{v}_i \;+\; {}^{k}\lambda_m \, {}^{k}\Gamma_m^{\alpha-1}] \qquad (3.2.3.10) $$

To obtain ${}^{k}\tilde{v}_i$, we permute the dummy indices i and m and form the dynamical system

$$ \dot{v}_i \;+\; {}^{k}\eta_i \, v_i \;=\; {}^{k}\hat{\varphi}_i \, [\sum_m T_{mi} \, v_m \;+\; {}^{k}\lambda_i \, {}^{k}\Gamma_i^{\alpha-1}] \qquad (3.2.3.11) $$

The equilibrium points ${}^{k}\tilde{v}_i$ (obtained when, $\dot{v}_i \;\rightarrow\; 0$) are then used in the computation of $\nabla_T E$, i.e.,

$$ \frac{\partial E}{\partial T_{ij}} \;=\; \omega_{ij} T_{ij} \;-\; \sum_k {}^{k}\tilde{v}_i \, \varphi({}^{k}\tilde{u}_j) \qquad (3.2.3.12) $$

Finally, by combining Eqs. (3.2.2.6) and (3.2.3.12) the *neural learning equations* can be expressed as

$$ \dot{T}_{ij} \;=\; -\tau \, [\omega_{ij} \, T_{ij} \;-\; \sum_k {}^{k}\tilde{v}_i \, \varphi({}^{k}\tilde{u}_j)] \qquad (3.2.3.13) $$

To compute $\nabla_\lambda E$ we return to the definition of E to obtain

$$ \frac{\partial E}{\partial \, {}^{l}\lambda_i} \;=\; \frac{1}{\alpha} \sum_k \sum_n \delta_{ni} \, \delta_{kl} \, {}^{k}\Gamma_n^{\alpha} \;=\; \frac{1}{\alpha} \, {}^{l}\Gamma_i^{\alpha} \qquad (3.2.3.14) $$

Thus, the temporal evolution of the Lagrange multipliers, Eqs. (3.2.2.8) can be described by the equation

$$ {}^{l}\dot{\lambda}_i \;=\; \frac{\tau}{\alpha} \, \frac{\nabla_T E \oplus \nabla_T E}{\Lambda \;+\; 1/(\Lambda \;+\; \theta)} \, {}^{l}\Gamma_i^{\alpha} \qquad (3.2.3.15) $$

based on the above results, we now summarize learning algorithm SID_3.

Algorithm : Singularity Interaction Dynamics_3

Input : input/output attractor coordinates, a_n; network dimension; neural response function; neural gain; neuronal decay constants; state variable initialization domain; topological constraints, time scales

Output : network topology; learned synaptic strengths, T_{nm};

algorithm Singularity Interaction Dynamics_3

[1] Initialize : $\forall\, n,\ m\ \in\ \{\ 1,\ \cdots,\ N\ \}$ and $\forall\, k\ \in\ \{\ 1,\ \cdots,\ K\}$:

$$T^0_{nm} \ = \ \Re\,[\ -\epsilon,\ +\epsilon\ ]$$

$${}^k\lambda^0_n \ = \ \Re\,[\ -\epsilon,\ +\epsilon\ ]$$

[2] Learn synaptic matrix T : **Iterate** $\nu\ =\ 1,\ \cdots,\ N_T$

initialize outer product array : $\Sigma^\nu_{nm}\ =\ 0.$

[3] **Loop** over training samples, $k\ =\ 1,\ \cdots,\ K$

[3.1] Evolve network dynamics using

$$\dot{u}_n\ +\ \kappa\ u_n\ \ =\ \ \sum_m T^\nu_{nm}\ \varphi(u_m)\ +\ {}^kI_n$$

where

$${}^kI_n\ =\ \begin{cases} [a^k_n\ -\ \varphi(u_n)]^{1/3} & \text{if}\ \ n\ \in\ S_X \cup S_Q \\ 0 & \text{if}\ \ n\ \in\ S_H. \end{cases}$$

Output: $\varphi({}^k\tilde{u}^\nu_n)$ and ${}^k\tilde{I}^\nu_n$

[3.2] Evolve "importance" dynamics using

$$\dot{v}_n\ +\ {}^k\eta^\nu_n\ v_n\ \ =\ \ {}^k\hat{\varphi}^\nu_n\ [\ \sum_m T^\nu_{mn}\ v_m\ +\ {}^k\lambda^\nu_n\ {}^k\Gamma^{\alpha-1,\nu}_n\ ]$$

where

$$ {}^{k}\eta_n^{\nu} = \begin{cases} \kappa^u + \frac{1}{3}\left[a_n^k - \varphi({}^{k}\tilde{u}_n^{\nu})\right]^{\frac{-2}{3}} {}^{k}\hat{\varphi} & \text{if } n \in S_X \cup S_Q \\ \kappa^u & \text{if } n \in S_H \end{cases} $$

Output : ${}^{k}\tilde{v}_n^{\nu}$

[3.3] Update outer product contribution

$$ \Sigma_{nm}^{\nu} = \Sigma_{nm}^{\nu} + {}^{k}\tilde{v}_n^{\nu}\, \varphi({}^{k}\tilde{u}_m^{\nu}) $$

[4] **Endloop** over $\{k\}$

[5] Compute $\nabla_T E$

$$ (\nabla_T E)_{nm}^{\nu} = \Sigma_{nm}^{\nu} - \omega_{nm}\, T_{nm}^{\nu} $$

[6] Compute $\nabla_\lambda E$

$$ {}^{k}(\nabla_\lambda E)_n^{\nu} = \frac{1}{\alpha}\, {}^{k}\Gamma_n^{\alpha,\nu} $$

[7] Update T

$$ T_{nm}^{\nu+1} = T_{nm}^{\nu} + \tau\Delta\, (\nabla_T E)_{nm}^{\nu} $$

[8] Update λ using

$$ {}^{k}\lambda_n^{\nu+1} = {}^{k}\lambda_n^{\nu} + \tau\Delta\, \frac{\nabla_T E \oplus \nabla_T E}{\nabla_\lambda E \oplus \nabla_\lambda E + \theta}\, {}^{k}(\nabla_\lambda E)_n^{\nu} $$

[9] Check for convergence:

If yes **then** exit **else** goto [2]

[10] **Endloop** over learning iterations $\{\nu\}$

[11] **Exit :** Display results

3.2.4. Adaptive Time Scales

So far the adaptive learning rate, i.e., τ in Eq. (3.2.2.6), has not been specified. Now we will show that by an appropriate selection of this parameter the convergence of the dynamical systems (3.2.3.13) and (3.2.3.15) can be considerably improved. We seek τ in the form [36,40,105,305]

$$\tau \propto \mid \nabla E \mid^{-\beta} \tag{3.2.4.1}$$

where ∇E denotes the vector with components $\nabla_T E$ and $\nabla_\lambda E$. It is straightforward to show that

$$\frac{d}{dt} \mid \nabla E \mid = -\chi \mid \nabla E \mid^{1-\beta} \tag{3.2.4.2}$$

as ∇E tends to zero, where χ is an arbitrary positive constant. If we evaluate the relaxation time of the energy gradient, we find that

$$t_E = \int_{|\nabla E|_0}^{|\nabla E| \to 0} \frac{d \mid \nabla E \mid}{\mid \nabla E \mid^{1-\beta}} = \begin{cases} \infty & \text{if } \beta \leq 0 \\ \frac{1}{\beta} \mid \nabla E \mid_0^\beta < \infty & \text{if } \beta > 0 \end{cases} \tag{3.2.4.3}$$

Thus, for $\beta \leq 0$ the relaxation time is infinite, while for $\beta > 0$ it is finite. The dynamical systems (3.2.3.13) and (3.2.3.15) suffer a qualitative change for $\beta > 0$: they loose uniqueness of solutions. The equilibrium point $\mid \nabla E \mid = 0$ becomes a singular solution being intersected by all the transients, and the Lipschitz condition is violated, as one can see from

$$\frac{d}{d \mid \nabla E \mid} \left(\frac{d \mid \nabla E \mid}{dt} \right) = -\chi \mid \nabla E \mid^{-\beta} \longrightarrow -\infty \tag{3.2.4.4}$$

where $\mid \nabla E \mid$ tends to zero, while β is strictly positive. Such infinitely stable points are "terminal attractors", as discussed in section 2.1. The qualitative

effect is depicted in Fig. 3.2.5.1. By analogy with these previous results we choose $\beta = 2/3$, which yields

$$\tau = \left(\sum_n \sum_m [\nabla_T E]^2_{nm} + \sum_k \sum_i {}^k[\nabla_\lambda E]^2_n \right)^{-\frac{1}{3}} \qquad (3.2.4.5)$$

Finally, inspection of Eq. (3.2.3.13), i.e., $\dot{T}_{ij} \propto {}^k\tilde{v}_i \, \varphi({}^k\tilde{u}_j)$, suggests a possible physical interpretation for the quantities ${}^k\tilde{v}_i$. They measure the "importance" for neuron i of signals coming from all neurons j to which it is connected.

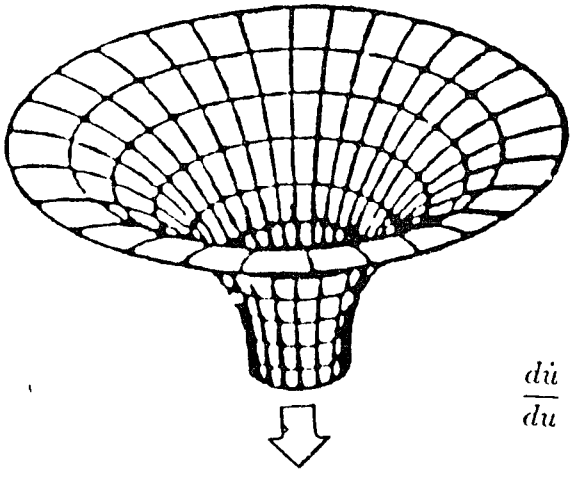


Figure 3.2.4.1. Qualitative illustration of the terminal attractor effect on the convergence of dynamical systems.

3.2.5. Virtual Terminal Attractors

Neural Learning methodologies based on dynamical systems must consider the fundamental problem of network stability as synaptic weights evolve during

training. Previously published approaches [235] ignored the issue and actually do not guarantee stability. To make some progress, we observe that if each node of the activation network had an associated terminal attractor , the resulting neurodynamics would be unconditionally stable. However, since by definition of a mapping, data are provided only for neurons in the input and output topographic partitions, "virtual" attractors must be determined for the hidden units. These attractors are virtual since they correspond to a current estimate of the synaptic connectivity matrix. Specifically, Eq. (3.2.1.2) has to be modified to read

$$ {}^{k}I_n \;=\; \begin{cases} [{}^{k}a_n \;-\; \varphi(u_n)\,]^{\beta} & \textbf{if } n \in S_X \\ [{}^{k}z_n \;-\; \varphi(u_n)\,]^{\beta} & \textbf{if } n \in S_H \cup S_Y \end{cases} \qquad (3.2.5.1) $$

where the virtual attractor coordinates z_n are obtained by considering the fixed point equations (3.2.2.2), as adaptive conservation equations which utilize the extra degrees of freedom made available by the hidden neurons in S_H. Thus, if we define

$$ {}^{k}\tilde{H}_n \;=\; \sum_{l \in S_X \cup S_Y} T_{nl}\; \varphi({}^{k}a_l) \qquad (3.2.5.2) $$

we can compute the virtual attractors from

$$ \dot{z}_n \;=\; -\,[\kappa z_n \;-\; \sum_{m \in S_H} T_{nm}\; \varphi(z_m) \;-\; {}^{k}\tilde{H}_n]^{1/3} \qquad (3.2.5.3) $$

Note that the above expression also involves terminal attractor dynamics.

3.3. Computation Learning Algorithm - SID_4

First we introduce some nomenclature. Let $\Re[-\epsilon, +\epsilon]$ represent a uniform random number distribution over the real interval $[-\epsilon, +\epsilon]$, where $\mid \epsilon \mid \ll 1$. Let N_T denote the maximum number of learning iterations, indexed by ν. The integration time-step is denoted by Δ. The learning rate being proportional to the outer product of the vectors ${}^{k}\tilde{v}$ and $\varphi({}^{k}\tilde{u})$, a two dimensional array Σ is

defined for collecting the contributions of each training sample during a particular iteration.

Algorithm : Singularity Interaction Dynamics_4

Input : input/output attractor coordinates, a_n; network dimension; neural response function; neural gain; neuronal decay constants; state variable initialization domain; topological constraints;

Output : network topology; learned synaptic strengths, T_{nm};

algorithm Singularity_Interaction_Dynamics_4

[1] Initialize : $\forall\, n,\ m\ \in\ \{\ 1,\ \cdots,\ N\ \}$ and $\forall\, k\ \in\ \{\ 1,\ \cdots,\ K\}$:

$$T^0_{nm} \ = \ \Re\ [\ -\epsilon,\ +\epsilon\ ]$$

$$^k\lambda^0_n \ = \ \Re\ [\ -\epsilon,\ +\epsilon\ ]$$

[2] Learn synaptic matrix T : **Iterate** $\nu\ =\ 1,\ \cdots,\ N_T$

initialize outer product array : $\Sigma^\nu_{nm}\ =\ 0.$

[3] **Loop** over training samples, $k\ =\ 1,\ \cdots,\ K$

[3.1] Evolve network dynamics using

$$\dot{u}_n\ +\ \kappa\ u_n\ \ =\ \ \sum_m T^\nu_{nm}\ \varphi(u_m)\ +\ {}^kI_n$$

where

$$^kI_n\ =\ \begin{cases} [a^k_n\ -\ \varphi(u_n)]^{1/3} & \text{if}\ \ n\ \in\ S_X \cup S_Q \\ 0 & \text{if}\ \ n\ \in\ S_H. \end{cases}$$

Output: $\varphi(^k\tilde{u}^\nu_n)$ and $^k\tilde{I}^\nu_n$

[3.2] Evolve "importance" dynamics using

$$\dot{v}_n\ +\ {}^k\eta^\nu_n\ v_n\ \ =\ \ {}^k\hat{\varphi}^\nu_n\ [\ \sum_m T^\nu_{mn}\ v_m\ +\ {}^k\lambda^\nu_n\ {}^k\Gamma^{\alpha-1,\nu}_n\ ]$$

where

$$ {}^{k}\eta_n^{\nu} = \begin{cases} \kappa^u + \frac{1}{3}\left[a_n^k - \varphi({}^{k}\tilde{u}_n^{\nu})\right]^{\frac{-2}{3}} {}^{k}\hat{\varphi} & \text{if } n \in S_X \cup S_Q \\ \kappa^u & \text{if } n \in S_H \end{cases} $$

Output : ${}^{k}\tilde{v}_n^{\nu}$

[3.3] Update outer product contribution

$$ \Sigma_{nm}^{\nu} = \Sigma_{nm}^{\nu} + {}^{k}\tilde{v}_n^{\nu}\, \varphi({}^{k}\tilde{u}_m^{\nu}) $$

[4] **Endloop** over $\{k\}$

[5] Compute $\nabla_T E$

$$ (\nabla_T E)_{nm}^{\nu} = \Sigma_{nm}^{\nu} - \omega_{nm}\, T_{nm}^{\nu} $$

[6] Compute $\nabla_\lambda E$

$$ {}^{k}(\nabla_\lambda E)_n^{\nu} = \frac{1}{\alpha}\, {}^{k}\Gamma_n^{\alpha,\nu} $$

[7] Compute τ using Eq. (3.2.4.5)

$$ \tau = \left(\sum_n \sum_m \left[\, \nabla_T E \,\right]_{nm}^2 + \sum_k \sum_i {}^{k}\left[\nabla_\lambda E\right]_n^2 \right)^{-\frac{1}{3}} $$

[8] Update T

$$ T_{nm}^{\nu+1} = T_{nm}^{\nu} + \tau\Delta\, (\nabla_T E)_{nm}^{\nu} $$

[9] Update λ using

$$ {}^{k}\lambda_n^{\nu+1} = {}^{k}\lambda_n^{\nu} + \tau\Delta\, \frac{\nabla_T E \oplus \nabla_T E}{\nabla_\lambda E \oplus \nabla_\lambda E + \theta}\, {}^{k}(\nabla_\lambda E)_n^{\nu} $$

[10] Check for convergence:

If yes **then** exit **else** goto [2]

[11] **Endloop** over learning iterations $\{\nu\}$

[12] **Exit :** Display results

Note that even though the formulae given for algorithmic steps 8 and 9, above are based on the Euler approximation for simplicity of notation, more sophisticated integration techniques must always be used. In addition, since it is well known that a discrete approximation of a continuous dynamical system subsumes absence of singularities, extreme caution needs to be exercised during implementation of the above algorithm in a digital computational environment. For example, infinitesimal integration steps are typically required to overcome effects that can be induced by interacting terminal attractors, or, spatial singularities induced by the fractal boundaries of attraction.

3.4. Operational Network

During run-time, i.e., after the invariant characteristics of the nonlinear transformation have been encaptured by the network, the above neurodynamics can be used to compute joint-space configurations corresponding to arbitrary task-space coordinates. However, the "operational" version of the neural network differs from the "in-training" model in two respects. Firstly, network dynamics in the operational phase no longer includes the "source" contribution for neurons in the output set S_Q. It is assumed that if the system has truly acquired the topological invariances of the inverse kinematics from the presented training samples, then the network's emergent generalization ability will always force convergence to the appropriate fixed points of the neurodynamics. States of the output neurons in S_Q can then be examined to extract the joint-space configuration computed by the network. For inputs that correspond to one of the training

examples this implies heteroassociative recall. On the other hand, presenting inputs that do not correspond to the stored states of the system, entails burning new fixed points in the phase-space landscape of the network. An analytical argument validating the neural network's capability for computing a continuum of outputs in response to an arbitrary input stimuli is included in [235].

Secondly, we attempt to resolve kinematic redundancy during run-time by directly encoding application-specific as well as environmental constraints into the dynamics of the operational network. This is a significant departure from existing models. There are essentially three modes for handling kinematic, design and workspace constraints using neural networks:

(a) generate the training samples such that they always conform to certain selected criteria. For example, Tawel et al. [267] train their network using only those samples that yield a manipulator configuration with minimal potential energy. They therefore eliminate the need for redundancy resolution as all joint-configurations computed by the network will conform to the above kinematic criteria.

(b) In our earlier derivation in Chapter Two, we provided a systematic mechanism for incorporating constraint information into the neuromorphic objective function.

(c) Here, in a radically different approach from renormalization group theory (as summarized in Section 1.6.4), we propose to add constraint information directly into the dynamics of the operational network, without disturbing the synaptic elements or the interconnection topology.

The first two approaches *skew the network behavior* in that it learns only limited aspects of the inverse kinematic mapping, and are limited in applicability to situations targeting structured environments. If a manipulator trained in such a

specific manner were to be deployed in unstructured environments, where each different task necessitates a different constraint resolution, we would need to retrain the network for each such additional constraint, thereby severely limiting real-time performance.

In the present approach, the network is initially made to learn all inverse kinematics solution manifolds. Subsequently, mechanisms are provided at run-time whereby a specific pose that can satisfy the problem constraints is selected. The network dynamics in the operational mode then takes the following form:

$$\dot{u}_n + \kappa u_n = \sum_m W_{nm} \varphi_\gamma(u_m) + I_n^x + \sum_r {}^r\lambda_n [\nabla_{\bar{u}} g_r(\rho_r, \bar{u})] \quad (3.4.1)$$

where W_{nm} denotes the learned interconnection matrix, and the "working" input source I_n^x is now defined as,

$$I_n^x = [x_n - \varphi(u_n)]^{-1/3} \quad (3.4.2)$$

if $n \in S_X$ and 0 if $n \in S_H \cup S_Q$. The constraints $g_r(\cdot)$ relate to application-specific considerations , e.g., obstacle avoidance, singularity avoidance, manipulability etc. The arguments ρ_r denote constraint-specific normalization parameters, so constructed that the gradient contribution to the dynamics vanishes whenever the actual constraint is satisfied by the neuronal activities. For example, if our only requirement were to maximize the joints configurational entropy (an "academic" but conceptually simple constraint !), then the last term of Eq. (3.4.1) would reduce to $\lambda [1 + \mathrm{Log}_e(\rho u_m)] \delta_{mn}$, where $m \in S_Q$, and the renormalization parameter ρ would be given by $N_Q / [\epsilon \sum_{m' \in S_Q} u_{m'}]$. The Lagrange multipliers ${}^r\lambda$ reflect the importance of the $r-$th constraint and κ is again related to the characteristic decay of neuron activity.

3.5. Simulation Results

The computational framework developed in the preceding section has been applied to a number of problems including signal reconstruction [104] and robotics, e.g., inverse kinematics of redundant manipulators [35,105], that involve learning nonlinear mappings. In the sequel we discuss two of our experiments which involved learning a continuous clipping nonlinearity and the inverse kinematics of redundant 3-DOF planar manipulator. This function has been extensively benchmarked in [189], and provides an adequate basis for illustrating the computation efficacy of our proposed formulation as compared to the existing neural learning algorithms. The test setup included a fully connected network with one neuron in the input and output sets respectively, and two hidden neurons. The training set consisted of 8 randomly chosen training samples. Since our learning methodology involves singular solutions of highly coupled, continuous dynamical systems, extreme caution must be exercised when simulating the algorithms in a digital computing environment. For example, explicit methods such as Euler or Runge-Kutta cannot be used, since the presence of singularities induces extreme stiffness. Practically this would require an integration time-step of infinitesimal size, resulting in numerical round-off errors of unacceptable magnitude. Clearly, fully implicit integration techniques have to be used. For the simulations reported below, Eqs (3.2.1.1. and 2.3.11) are integrated using a fourth-order Kaps-Rentrop scheme discussed in [241].

To illustrate our remarks on the sensitivity of results to the integration schemes we first comment on the results shown in Fig. 3.5.1(a)-(d), which were obtained using the simplistic Euler discretization. Fig. 3.5.1(a) shows the evolution of the normalized least-mean-square (LMS) error, i.e., the difference between the network output and the target output averaged over all training samples. Fig 3.5.1(b) displays the worst-case temporal behavior of the input and output neuron

states normalized to the unit interval. Notice the rapid convergence of the input state due to the terminal attractor effect. However, as the system approaches a singularity, the adjoint equations (3.2.3.5) becomes hyperstiff. Clearly, the coefficient η in Eqn. (3.2.3.9), tends to infinity near the singularity, and induces strong instabilities as shown by the spikes in the graph. This change is subsequently reflected in the oscillations of the energy-gradient contractions, $\nabla_\lambda E \oplus \nabla_\lambda E$ and $\nabla_T E \oplus \nabla_T E$, plotted in Fig. 3.5.1(d) Finally, the behavior of the adaptive time-scale parameter is depicted in Fig. 3.5.1(c). The system as a whole is never stable and the state of the output neuron oscillates in a close neighborhood of the singularity.

The learning algorithm was then implemented using a fourth-order Kaps-Rentrop implicit integration framework. At this stage only the activation dynamics Eq. (3.2.1.1) and the "adjoint" dynamics Eqn. (3.2.3.11) are integrated implicitly. Figure 3.5.2 shows the LMS error during the training phase. The worst-case convergence of the output state neuron to the presented attractor is displayed in Fig. 3.5.3. Note that the system learns the nonlinear map orders of magnitude faster than by the back-propagation algorithm (0.2 secs vs 5000 secs effective time). Fig. 3.5.4 shows the evolution of the energy gradient components. This network yielded an interpolation/recall precision with error under 0.3% for 80% of the test samples. The worst-case error detected was 3.5%. In contrast, a multi-layer perceptron network with 2 hidden layers (detailed network configuration is discussed in [189]) and trained using the backpropagation algorithm, required 5000 seconds effective training time on Neural Ware's Neuralworks-II package. The worst-case error detected was 5% using 20 training samples. In reference [189] 1000 training samples were used to yield an error of 0.01%.

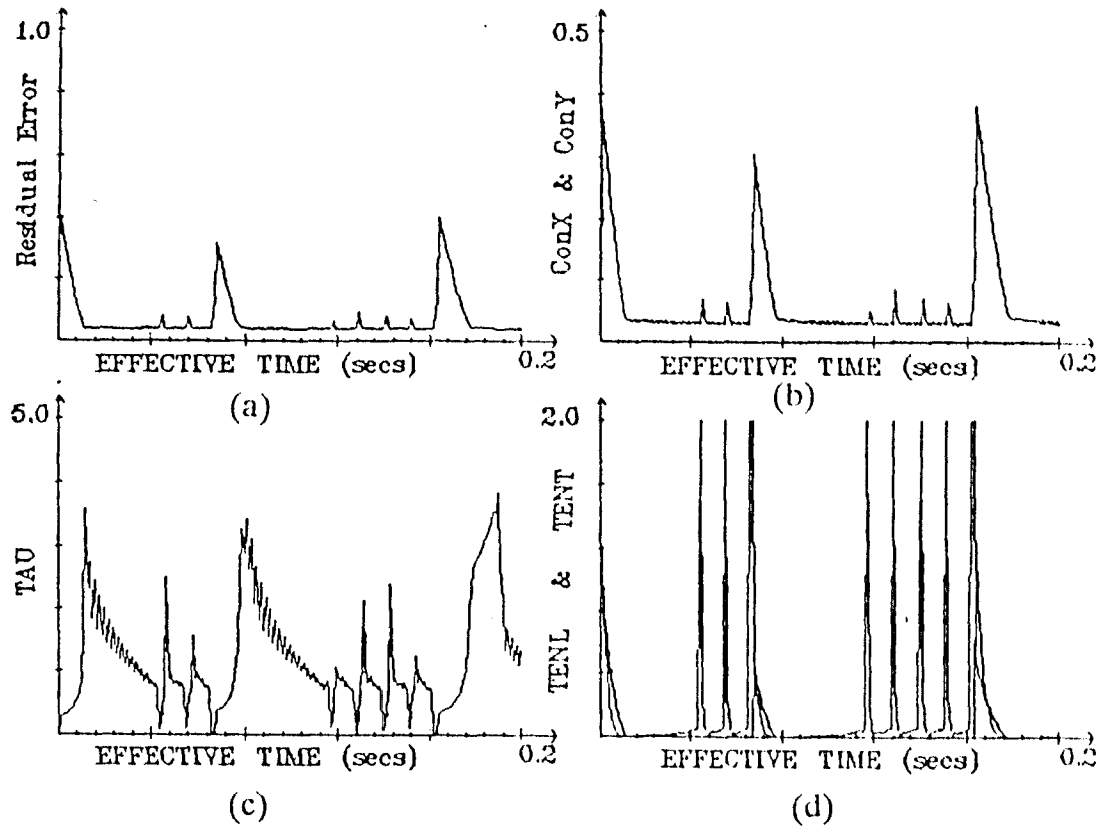


Figure 3.5.1(a-d). Oscillations resulting from numerical instabilities when naive integration techniques, e.g., Euler, Runge-Kutta, are used to implement SID neural learning formalisms.

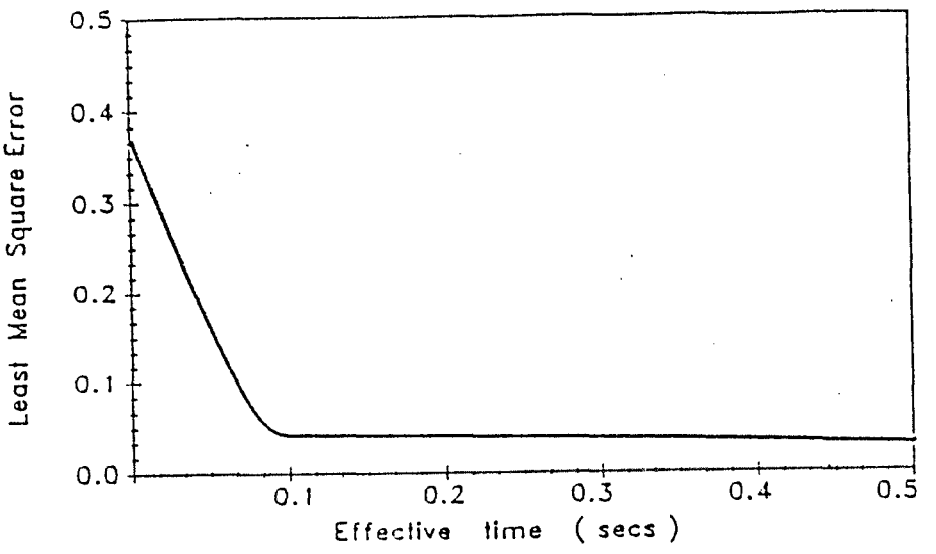


Figure 3.5.2. Profile of LMS error during learning of the clipping nonlinearity

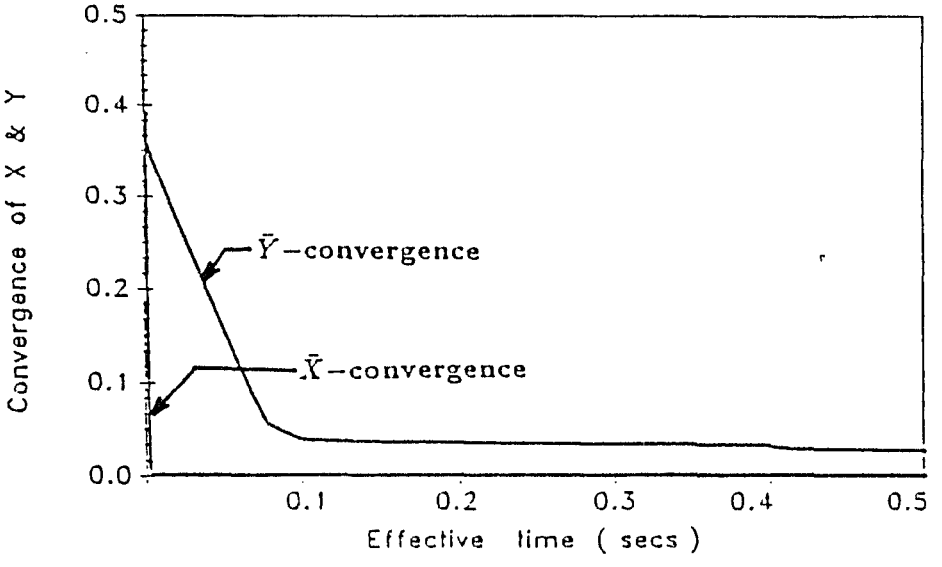


Figure 3.5.3. Worst case convergence of input and output state to the presented attractors.

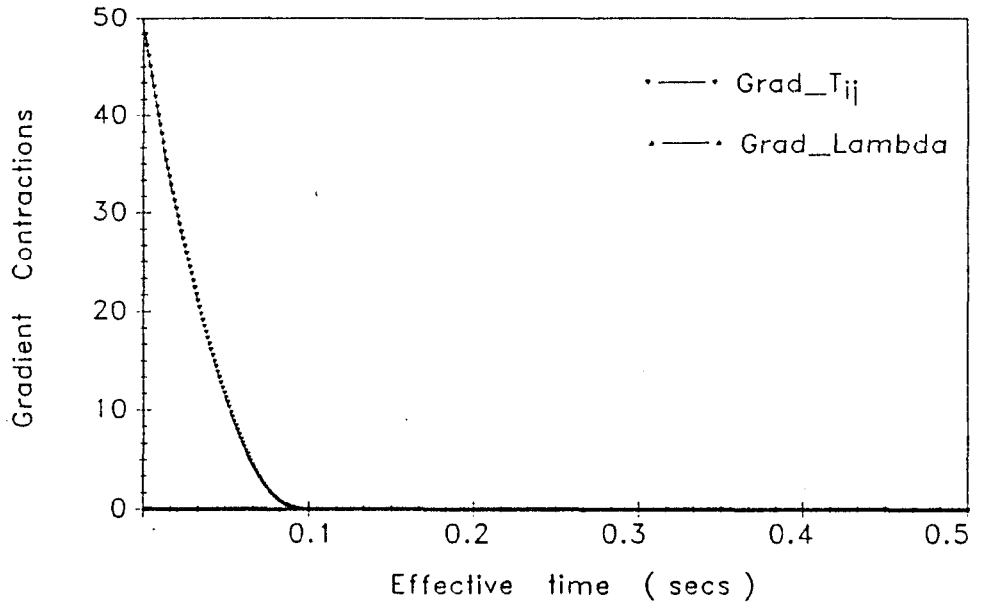


Figure 3.5.4. Temporal evolution of the energy gradient tensor contractions.

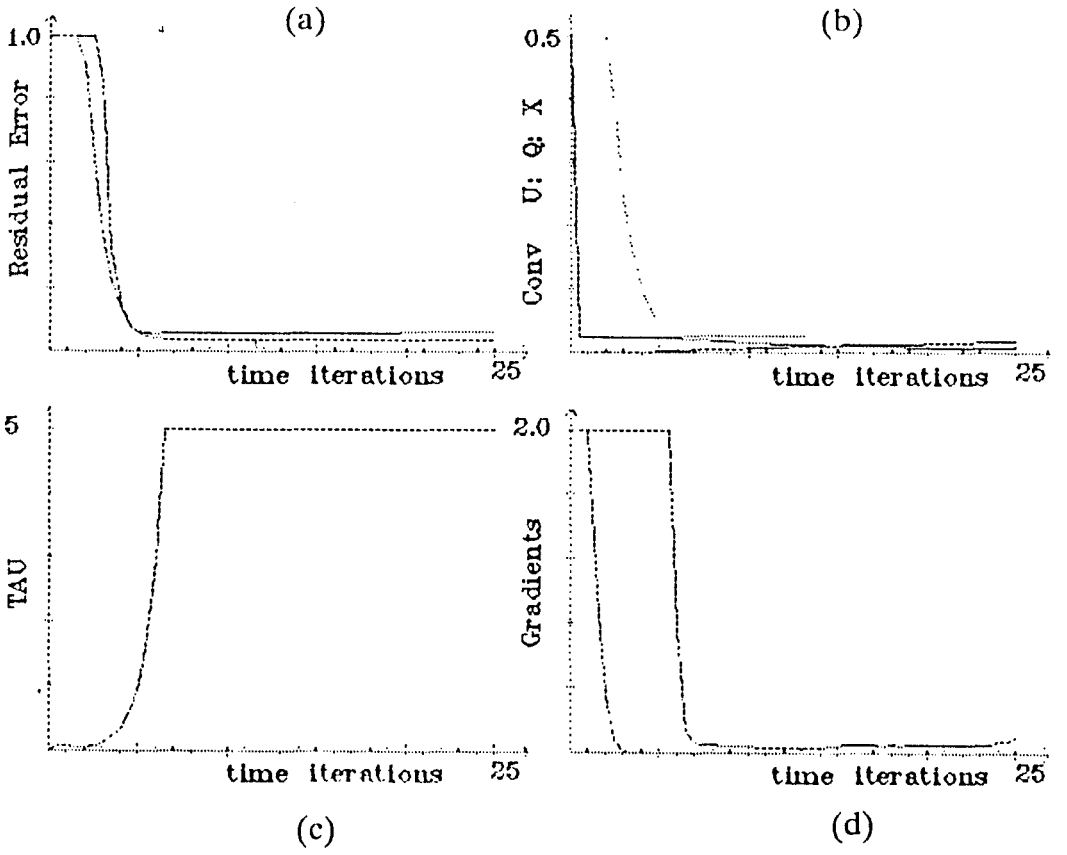


Figure 3.5.5(a-d). Learning the inverse kinematics of a 3-DOF planar arm (ref: [314]) using ***Algorithm SID_4.***

Figure 3.5.5(a-d) illustrates our results during learning the manipulator inverse kinematics of the 3-dof planar robot arm described in [267], using randomly sampled joint and cartesian-coordinates. Figure 3.5.5(a) shows the evolution of the normalized least-mean square (LMS) error, i.e.,

$$\sqrt{\frac{(\sum_{k=1}^{K} Output_{network} - Output_{actual})^2}{N_Y}}.$$

The algorithm *SID_4* enabled learning with an LMS error under 0.05% in an effective time of 2.5ms. Fig. 3.5.5(b) displays the worst-case temporal behavior of the input and output neuron states normalized to the unit interval. From the graph we can immediately get a qualitative feel for the terminal attractor effective during learning. The behavior of adaptive time constant τ is shown in Figure 3.5.5(c). Finally, in Fig. 3.5.5(d) we present the temporal evolution of the energy gradient tensor contractions, $\nabla_{\mathbf{T}} \oplus \nabla_{\mathbf{T}}$ and $\nabla_{\lambda} \oplus \nabla_{\lambda}$. Specifics on network parameters, training sample selection and additional implementation details are included in Gulati and Barhen [105].

3.6. Summary

In this chapter we have presented a new theoretical framework for adaptive learning using artificial neural networks. Continuous nonlinear mappings and topological transformations constitute the main application targets for the proposed methodology. Our methodology introduces the concept of topographically partitioned, but fully connected networks, to facilitate the encoding of training samples as terminal attractors. In a significant departure from prior neuromorphic formulations to the inverse kinematics problem, we completely decouple redundancy resolution issues from learning the inherent properties of the inverse kinematic transformation. Rather than training with samples optimized *a priori* with respect to a particular kinematic objective, or, encoding such considerations as learning goals, we suggest an *a posteriori* regularization approach.

Hence, initially the network is trained using pairs sampled over one or more solution manifolds. Then, during run-time, the desired constraints are included in the dynamics such that network convergence necessarily ensures constraint satisfaction. Notice that this method requires no additional training.

Chapter Four

Application of Adjoint Sensitivity Theory in Neural Networks

In continuation to our incremental derivation of high performance computational neural learning algorithms we draw from mathematical constructs in sensitivity theory (introduced in Chapter One) for nonlinear systems to provide a brief introduction to the notion of forward and adjoint operators. The formalism exploits the concept of *adjoint operators* to enable a fast global computation of the network's response to perturbations in all system parameters. This formalism eliminates the heuristic overtones adopted in previous derivations. The concepts are used to derive another version of neural learning algorithm - SID_4.

4.1. Introduction

A considerable effort has recently been devoted to the development of efficient computational methodologies for learning. Attention has largely focussed on the back-propagation algorithm because of its simplicity, generality and the promise that it has shown in regard to various applications [250,251]. More recently, Pineda [134,135] has derived a generalization to back-propagation for recurrent networks. In a similar vein, Williams and Zipser [292] have presented algorithms for learning tasks with temporal dependencies. Pearlmutter [234] has proposed a similar technique which minimizes an error functional between output and targeted temporal trajectories. In a significantly different approach, Barhen, Gulati, Zak, Toomarian [35,36,105,111] recently introduced neural formalisms to efficiently learn nonlinear mappings using a new mathematical construct, i.e., terminal attractors [303]. Terminal attractor representations were used not only

to ensure infinite local stability of the encoded information, but also to provide a qualitative as well as quantitative change in the nature of the learning process. In particular, they imply loss of Lipschitz conditions at energy function minima, which results in a dramatic increase in the speed of learning.

The development of learning algorithms is generally based upon the minimization of a "neuromorphic" energy-like function. A fundamental requirement of all previously mentioned methods is the computation of the gradient of this objective function with respect to the various parameters of the neural architecture, e.g., synaptic weights, neural gain, etc. In the present paper we introduce a new methodology for their efficient analytical computation, as a single solution of a set of "adjoint" equations. Cacuci, Barhen, Oblow, Toomarian, etc., have already successfully used adjoint operators in the fields of energy economy modeling [10] and nuclear reactor thermal hydraulics [27,273] at the Oak Ridge National Laboratory, where the concept flourished during the past decade [223,311].

4.2. Sensitivity Theory

Consider, for the sake of generality, that a problem of interest is represented by the following system of N coupled nonlinear equations

$$\bar{\varphi}(\bar{u},\bar{p}) \;=\; 0 \qquad (4.2.1)$$

where $\bar{\varphi}$ denotes a nonlinear operator. If differential operators appear in Eq. (4.2.1), then a corresponding set of boundary and/or initial conditions to specify the domain of φ must also be provided. The learning model discussed in this paper focuses on the adiabatic approximation only (steady state networks).

Let $\bar{u}$ and $\bar{p}$ represent the N-vector of dependent variables and the M-vector of system parameters, respectively. We will assume that generally $M >> N$

and that elements of $\bar{p}$ are, in principle, independent. Furthermore, we will also assume that, for a specific choice of parameters, a unique solution of Eq. (4.2.1) exists. Hence, $\bar{u}$ is an implicit function of $\bar{p}$. A system response, R, represents any result of the calculations that is of interest. Specifically

$$R = R(\bar{u}, \bar{p}) \tag{4.2.2}$$

i.e., R is a known nonlinear function of $\bar{p}$ and $\bar{u}$ and may be calculated from (4.2.2) when the solution $\bar{u}$ in Eq. (4.2.1) has been obtained for a given $\bar{p}$. The problem of interest is to compute the "sensitivities" of R, i.e., the derivatives of R with respect to parameters p_μ, $\mu = 1, \cdots, M$. By definition

$$\frac{dR}{dp_\mu} = \frac{\partial R}{\partial p_\mu} + \frac{\partial R}{\partial \bar{u}} \cdot \frac{\partial \bar{u}}{\partial p_\mu} \tag{4.2.3}$$

4.2.1. Forward Sensitivity Theory

Since the response R is known analytically, the computation of $\partial R/\partial p_\mu$ and $\partial R/\partial \bar{u}$ is straightforward. The quantity that needs to be determined is the vector $\partial \bar{u}/\partial p_\mu$. Differentiating the state equations (4.2.1), we obtain a set of equations to be referred to as "forward" sensitivity equations

$$\frac{\partial \bar{\varphi}}{\partial \bar{u}} \cdot \frac{\partial \bar{u}}{\partial p_\mu} = -\frac{\partial \bar{\varphi}}{\partial p_\mu} \tag{4.2.1.1}$$

To simplify the notations, we are omitting the "transposed" sign and denoting the N X N forward sensitivity matrix $\partial \bar{\varphi}/\partial \bar{u}$ by A, the N-vector $\partial \bar{u}/\partial p_\mu$ by ${}^\mu\bar{z}$ and the "source" N-vector $-\partial \bar{\varphi}/\partial p_\mu$ by ${}^\mu\bar{s}$. Thus

$$A\,{}^\mu\bar{z} = {}^\mu\bar{s} \tag{4.2.1.2}$$

4.2.2. Adjoint Sensitivity Theory

Computation of the response gradient using the forward sensitivity equations would require solving a system of N nonlinear algebraic equations for each parameter p_μ, since the source term in Eq. (4.2.1.2) explicitly depends on μ. This difficulty is circumvented by introducing adjoint operators. Let A^* denote the formal adjoint of the operator A [10,27,311]. However, note that Adjoint operators can only be considered for densely defined linear operators on Banach spaces. For the neural application under consideration we will limit ourselves to real Hilbert spaces only. Such spaces are self-dual. Furthermore, the domain of an adjoint operator is determined by selecting appropriate adjoint boundary conditions. The associated bilinear form evaluated on the domain boundary must generally be also included. The *adjoint sensitivity equations* can then be expressed as

$$A^* \, {}^\mu\bar{z}^* \;=\; {}^\mu\bar{s}^*. \tag{4.2.2.1}$$

By definition, for algebraic operators

$$\begin{aligned} {}^\mu\bar{z}^* \cdot (A \, {}^\mu\bar{z}) \;&=\; {}^\mu\bar{z}^* \cdot {}^\mu\bar{s} \;=\; {}^\mu\bar{z} \cdot (A^* \, {}^\mu\bar{z}^*) \\ &=\; {}^\mu\bar{z} \cdot {}^\mu\bar{s}^* \end{aligned} \tag{4.2.2.2}$$

Since Eq. (4.2.3) can be rewritten as

$$\frac{dR}{dp_\mu} \;=\; \frac{\partial R}{\partial p_\mu} + \frac{\partial R}{\partial \bar{u}} \, {}^\mu\bar{z} \, , \tag{4.2.2.3}$$

if we identify

$$\frac{\partial R}{\partial \bar{u}} \;\equiv\; {}^\mu\bar{s}^* \;\equiv\; \bar{s}^* \tag{4.2.2.4}$$

we observe that the source term of the adjoint equations is independent of the specific parameter p_μ. Hence, *the solution of a single set of adjoint equations will*

provide all the information required to compute the gradient of R with respect to all parameters. To underscore that fact we shall denote ${}^{\mu}\bar{z}^{*}$ as $\bar{v}$. Thus

$$\frac{dR}{dp_{\mu}} = \frac{\partial R}{\partial p_{\mu}} + \bar{v} \cdot {}^{\mu}\bar{s}. \tag{4.2.2.5}$$

4.3. Applications to Neural Learning

Along the lines adopted in Chapter Two, consider a densely connected neural network with N graded-state response neurons operating in continuously sampled time. To acquire a nonlinear transformation, ζ, from a $\aleph_X$-dimensional input domain to the $\aleph_Y$-dimensional output space, the network is topographically partitioned into three mutually exclusive regions with no topological restrictions. As illustrated in Fig. 2.3.1.1, the partition refers to a set of input neurons, S_X, that receive the input components, an output set S_Y, which provides the desired output components and a set of "hidden" neurons, S_H, that encode the representation of the ζ-mapping. The network is presented with K randomly sampled training vector-pairs of input- and output-space coordinates.

4.3.1. Neurodynamics Derivation

We formalize the neural network as an adaptive dynamical system whose temporal evolution is governed by the following energy function

$$\begin{aligned} E_N = & \sum_n \kappa_n \int_o^{u_n} u' \, \frac{d\varphi(\gamma u')}{du'} du' - \frac{1}{2} \sum_n \sum_m T_{nm} \, \varphi(\gamma u_n) \, \varphi(\gamma u_m) \\ & + \frac{3}{4} \sum_n \Big[a_n - \varphi(\gamma u_n) \Big]^{4/3} \end{aligned} \tag{4.3.1.1}$$

where u_n represents the mean soma potential of the nth neuron. The constant κ_n characterizes the decay of neuron activity. The sigmoidal function $g(\cdot)$ modulates

the neural response, with gain given by γ_m; typically, $g(\gamma z) = \tanh(\gamma z)$ or $1/(1+exp(-\gamma.z))$. Further, W denotes a symmetric positive matrix, whose W_{nm}-th element corresponds to the coupling between the $m-$th to the $n-$th neuron. In the sequel we explicate the conditions for constructing W. While the first two terms, are derived from an underlying electrical circuit interpretation [134], the third term enforces the convergence of a neuron to its presented attractor, if any, typifying the nonlinear mapping to be learned.

The specific attractor coordinates are given by

$$
\begin{aligned}
{}^{k}a_n &\equiv {}^{k}x_n \qquad \textbf{if } n \in S_X \\
{}^{k}a_n &\equiv {}^{k}y_n \qquad \textbf{if } n \in S_Y
\end{aligned}
$$

for $\{ {}^{k}\bar{x}, {}^{k}\bar{y} \mid k = 1, \cdots, K\}$ taken from a training set scaled to the range $[-1, +1]$.

As shown by Cohen and Grossberg [70,98], Hopfield [135] and others [15,74, 75], such neurodynamical systems, e.g., Eqs. (4.3.1.1), will approach an equilibrium point in response to an arbitrary, but sustained input if the underlying function is global lyapunov, i.e., $\frac{dE_N}{dt} < 0$. For the purpose of deriving the activation neurodynamics, the only "parameter" of E_n are $\overline{u}$; i.e., $\overline{\kappa}$, W and $\overline{a}$ are fixed. The neural response, φ and its corresponding gain are assumed to be available. In order to ensure global stability, we require

$$
\frac{d}{dt}E_N = \sum_i \frac{dE}{du_i}\frac{du_i}{dt} < 0 \qquad (4.3.1.2)
$$

Computing $\frac{dE}{du_i}$ from Eqn. (4.3.1.1), we obtain

$$
\begin{aligned}
\frac{dE}{du_i} = {} & \kappa_i u_i \hat{\varphi}_i - \frac{1}{2}\sum_m W_{im}\,\hat{\varphi}_i\,\varphi(u_m) \\
& - \frac{1}{2}\sum_n W_{ni}\,\varphi(u_n)\,\hat{\varphi}_i - [a_i - \varphi(u_i)]^{1/3}\hat{\varphi}_i \qquad (4.3.1.3)
\end{aligned}
$$

where $\hat{\varphi}_i$ denotes the derivative of the neural response. So for $\gamma \neq 1$, $\hat{\varphi}_i = \gamma \cdot \varphi'(\gamma u_i)$. Since the matrix W is symmetric, and n is a dummy index,

$$\frac{dE}{du_i} = \kappa_i u_i \hat{\varphi}_i - \hat{\varphi}_i \sum_m W_{im} \varphi(u_m) - \hat{\varphi}_i \Big[a_i - \varphi(u_i) \Big]. \qquad (4.3.1.4)$$

Enforcing the Lyapunov stability requirement on Eqs. (4.3.1.4) yields smallskip

$$\sum_i \dot{u}_i \left[\kappa_i \, u_i - \sum_j W_{ij} \, \varphi(u_j) - [a_i - \varphi(u_i)]^{1/3} \right] \hat{\varphi} < 0. \qquad (4.3.1.5)$$

Since $\hat{\varphi}_i$ is always positive, we can choose as our network dynamics

$$\dot{u}_i = -\kappa_i \, u_i + \sum_j W_{ij} \, \varphi(u_i) + \Big[a_i - \varphi(u_i) \Big]^{1/3}. \qquad (4.3.1.6)$$

The symmetric interconnection matrix W can be simply constructed from the learned synaptic matrix T such that

$$W_{ij} = \sum_\ell T_{\ell i} \, T_{\ell j} = T^{\mathrm{T}} T. \qquad (4.3.1.7)$$

. Taking into account the $k-$dependence, the $n-$th neuron's temporal evolution can be symbolically represented as

$$\dot{u}_n + \kappa \, u_n = \sum_m W_{nm} \, \varphi(\gamma \cdot u_m) + {}^k I_n \qquad (4.3.1.8)$$

where the "source" term, ${}^k I_n$ encodes component-contribution by the presented attractors of the k-th training sample via the expression

$${}^k I_n = \begin{cases} [{}^k a_n - \varphi(u_n)]^\beta & \text{if } n \in S_X \cup S_Y \\ 0 & \text{if } n \in S_H \end{cases} \qquad (4.3.1.9)$$

The topographic input, output and hidden network partitions S_X, S_Y and S_H are architectural requirements related to the encoding of mapping-type problems.

In previous chapters we have demonstrated that in general, for $\beta = (2i+1)^{-1}$ and i a positive integer, such attractors have infinite local stability and provide opportunity for learning in real-time.

4.3.2. Computational Learning Objectives

To proceed formally with the development of a learning algorithm, we consider an approach based upon the minimization of a constrained "neuromorphic" energy-like function E given by the following expression:

$$E(\bar{u},\bar{\lambda},\bar{p}) \;=\; \frac{1}{2}\sum_{n}\sum_{m}\omega_{nm}\,(\,T^{2}_{nm}\;-\;T_{nm}\,T_{mn}\,)\;+\;\frac{1}{\alpha}\sum_{k}\sum_{n}{}^{k}\lambda_{n}\;{}^{k}\Gamma^{\alpha}_{n} \tag{4.3.2.1}$$

where the constraints are of the form

$${}^{k}\Gamma_{n} \;=\; \begin{cases} {}^{k}a_{n} \;-\; g(\gamma_{n}\;{}^{k}\tilde{u}_{n}) & \textbf{if } n \;\in S_{X} \\ 0 & \textbf{if } n \;\in\; S_{H}\cup S_{Y} \end{cases} \tag{4.3.2.2}$$

As derived in Chapter Two, a positive value such as 2 is used for α. The weighting factor ω_{nm} is constructed in such a fashion, as to favor locality of computation. The indices n, m span over all neurons in the network. Lagrange multipliers corresponding to the nk–th constraint are denoted by ${}^{k}\lambda_{n}$. The superscript $\sim$ denotes quantities evaluated at steady state.

The proposed objective function, Eqn. (4.3.2.1), includes contributions from two sources.

[1] It enforces convergence of every neuron in S_X and S_Y to attractor coordinates corresponding to the components in the input-output training patterns, thereby prompting the network to learn the underlying invariances.

[2] It regulates the topology of the network by enforcing symmetry, and by minimizing interconnection strengths between distant synaptic elements to favor locality of computation.

Lyapunov stability requires an energy-like function to be monotonically decreasing in time. In our model the internal dynamical parameters of interest are the synaptic strengths T_{ij} of the interconnection topology, the characteristic decay constants κ_i, the gain parameters γ_i and the Lagrange multipliers ${}^k\lambda_i$. This implies that we require

$$\dot{E} = \sum_i \sum_j \frac{dE}{dT_{ij}} \dot{T}_{ij} + \sum_i \frac{dE}{d\kappa_i} \dot{\kappa}_i + \sum_i \frac{dE}{d\gamma_i} \dot{\gamma}_i + \sum_l \sum_i \frac{dE}{d^l\lambda_i} {}^l\dot{\lambda}_i < 0 \qquad (4.3.2.3)$$

One can always choose, with $\tau_T > 0$

$$\dot{T}_{ij} = -\tau_T \frac{dE}{dT_{ij}} \qquad (4.3.2.4)$$

where τ_T introduces an adaptive parameter for learning, as detailed in Chapter Three. Similar expressions can be constructed for $\dot{\kappa}$ and $\dot{\gamma}$, e.g.,

$$\dot{\kappa}_i = -\tau_\kappa \frac{dE}{d\kappa_i} \qquad (4.3.2.5)$$

and

$$\dot{\gamma}_i = -\tau_\gamma \frac{dE}{d\gamma_i} \qquad (4.3.2.6)$$

with τ_κ, $\tau_\gamma > 0$. Then, substituting in Eq. (4.3.2.3) and denoting tensor **contraction** $\oplus$, i.e., sum over all relevant indices (e.g., $\nabla_\kappa E \oplus \nabla_T E = \sum_i \sum_i dE/d\kappa \cdot dE/d\kappa$), one obtains

$$\nabla_\lambda E \oplus \dot{\lambda} < \tau_T (\nabla_T E \oplus \nabla_T E) + \tau_\kappa (\nabla_\kappa E \oplus \nabla_\kappa E) + \tau_\gamma (\nabla_\gamma E \oplus \nabla_\gamma E) \qquad (4.3.2.7)$$

Without loss of generality, one can assume $\tau = \tau_T = \tau_\kappa = \tau_\gamma$.

The equations of motion for the Lagrange multipliers ${}^l\lambda_i$ must now be constructed in such a way that Eqn. (4.3.2.7) is strictly satisfied. In addition, when the constraints are satisfied, i.e., as ${}^l\Gamma_n \rightarrow 0$ in Eqn. (4.3.9), we require that ${}^l\dot{\lambda}_i \rightarrow 0 \ \forall \ l$. We have adopted the following analytical model for the evolution of λ_i.

$$ {}^l\dot{\lambda}_i = \tau \frac{\Pi}{\Lambda + (1/(\Lambda + \theta))} {}^l[\nabla_\lambda E]_i \qquad (4.3.2.8) $$

where

$$ \Pi = \nabla_T E \oplus \nabla_T E + \nabla_\kappa E \oplus \nabla_\kappa E + \nabla_\gamma E \oplus \nabla_\gamma E $$

and ,

$$ \Lambda = \nabla_\lambda E \oplus \nabla_\lambda E. $$

Also, θ denotes an arbitrary positive constant. Using simple backsubstitution, it can be immediately shown that this model fulfills the above requirements.

4.3.3. Adaptive Learning Algorithms

We now focus on the derivation of an efficient algorithm for computing the "sensitivity" or parametric gradient contributions, $\nabla_\mu \mathbf{E}$ for $\mu \in \{T, \kappa, \gamma, \lambda\}$. An adiabatic computational framework is assumed. For instance, on differentiating Eqs. (4.3.2.1) with respect to T_{ij} we get

$$ \frac{\partial E}{\partial T_{ij}} = \omega_{ij} T_{ij} - \sum_k \sum_n {}^k\lambda_n \, {}^k\Gamma_n^{\alpha-1} \, \gamma_n \, {}^k\hat{\varphi}_n \frac{d}{dT_{ij}} {}^k\tilde{u}_n. \qquad (4.3.3.1) $$

where ${}^k\hat{\varphi}_n$ denotes the derivative of the neural response. We must compute $\frac{d}{dT_{ij}} {}^k\tilde{u}_n$ from the network fixed point equations (4.3.1.8).

A fundamental issue that needs to be addressed at this stage is the computational complexity of obtaining $\nabla_\mu \mathbf{E}$. For example, the computation of $\nabla_T E$ in

Eqs. (4.3.2.4) requires that N algebraic equations be solved for each parameter T_{ij}, that is N^3 equations at each iterative relaxation step. Similar requirements exist for evaluating $\nabla_\kappa E$ and $\nabla_\gamma E$ as given by Eqs. (4.3.2.5) and (4.3.2.6), since one must obtain the values of $\frac{d^k \tilde{u}_n}{d\kappa_i}$ and $\frac{d^k \tilde{u}_n}{d\gamma_i}$ respectively.

In relating adjoint theory to the neural learning algorithms, we identify the neuromorphic energy-like function, E in Eq. (4.3.2.1), with the system response. Let $\bar{p}$ denote the following system parameters:

$$\bar{p} \;=\; \{\, T_{11}, \cdots T_{NN} \mid \kappa_1, \cdots \kappa_N \mid \gamma_1, \cdots \gamma_N \mid \cdots \} \tag{4.3.3.2}$$

The adiabatic solution to the nonlinear equations of motion (4.3.1.8), for each training pattern k, $k = 1, \cdots, K$ is given by

$$^k\varphi_n(\, ^k\tilde{u}, \bar{p}) \;=\; -\kappa_n \, ^k\tilde{u}_n \;+\; \sum_m T_{nm}\, g(\gamma_m \, ^k\tilde{u}_m) \;+\; ^kI_n \;=\; 0. \tag{4.3.3.3}$$

So, in principle, $^k\tilde{u}_n = {}^k\tilde{u}_n\,[T,\ \bar{\kappa},\ \bar{\gamma},\ ^ka_n,\ ^k\lambda_n,\ \cdots]$. Using Eqs. (4.2.2.1), the forward sensitivity matrix can be computed and compactly expressed as

$$\begin{aligned} ^kA_{nm} \;=\; \frac{\partial\, ^k\varphi_n}{\partial\, ^k\tilde{u}_m} \;&=\; [\,-\kappa_n \;+\; \frac{\partial\, ^kI_n}{\partial\, ^k\tilde{u}_m}]\;\delta_{nm} \;+\; T_{nm}\, ^k\hat{g}_m\, \gamma_m \\ &=\; ^k\eta_n\, \delta_{nm} \;+\; \gamma_m\, ^k\hat{g}_m\, T_{nm} \end{aligned} \tag{4.3.3.4}$$

where $\hat{g}_m$ represents the derivative of g_m with respect to u_m, and

$$^k\eta_n \;=\; \begin{cases} \kappa + \frac{1}{3}\,[a_n^k - \varphi(\gamma_n \, ^k\tilde{u}_n)]^{-2/3}\;\gamma_n \, ^k\hat{\varphi}_n & \textbf{if } n \in S_N \\ \kappa & \textbf{if } n \in S_H \cup S_Y \end{cases}. \tag{4.3.3.5}$$

By analogy with Eqs. (4.2.3.3), the adjoint sensitivity matrix can be expressed as

$$^kA^*_{nm} \;=\; ^k\eta_m\delta_{mn} \;+\; \gamma_n \, ^k\hat{g}_n\, T_{mn}. \tag{4.3.3.6}$$

Using Eqs. (4.2.3.3) and (4.3.2.1), we can compute the adjoint source, $^ks^*_n = \frac{\partial E_L}{\partial\, ^k\tilde{u}_m}$ as

$$
\begin{aligned}
{}^{k}s_n^* &= \frac{1}{\alpha} \sum_{k'} \sum_{n'} {}^{k'}\lambda_{n'} \; \alpha \; {}^{k'}\Gamma_{n'} \; \frac{d \; {}^{k'}\Gamma_{u'}}{d \; {}^{k'}\tilde{u}_n} \\
&= - \; {}^{k}\lambda_n \; [\; {}^{k}a_n \; - \; \varphi({}^{k}\gamma_n \, {}^{k}\tilde{u}_n) \;] \; \gamma_n \; {}^{k}\hat{\varphi}_n
\end{aligned}
\tag{4.3.3.7}
$$

For any parameter p_μ, we have chosen $\dot{p}_\mu \; \propto \; -\tau_\mu \frac{dE_L}{dp_\mu}$, (e.g. in Eqs. (4.3.2.4), (4.3.2.5) and (4.3.2.6)) to strictly enforce Lyapunov's stability criteria, where $\frac{dE_L}{dp_\mu} \; = \; \frac{\partial E_L}{\partial p_\mu} \; + \; \sum_k \sum_n \frac{\partial E_L}{\partial \; {}^{k}\tilde{u}_n} \cdot \frac{d \; {}^{k}\tilde{u}_n}{dp_\mu}$, the parametric gradient contribution can be rewritten as

$$
\frac{dE_L}{dp_\mu} \; = \; \frac{\partial E_L}{\partial p_\mu} \; - \; \sum_k \sum_n {}^{k}v_n \; \frac{\partial \; {}^{k}\phi_n}{dp_\mu}. \tag{4.3.3.8}
$$

In the above expression ${}^{k}v_n$ denotes an element of the solution to the adjoint system,

$$
{}^{k}A^* \; {}^{k}\tilde{v} \; = \; {}^{k}\tilde{s}, \tag{4.3.3.9}
$$

To proceed with our derivation of learning equation, dE/dT_{ij} in the adjoint operator formalism, we differentiate the activation dynamics, Eqs. (4.3.1.8), with respect to each synaptic element, T_{ij} to obtain

$$
\sum_m \; [\; {}^{k}\eta_m \; \delta_{mn} \; + \; T_{mn} \; {}^{k}\hat{g}_n \; \gamma_n \;] \; {}^{k}\tilde{v}_m \; = \; - \; {}^{k}\lambda_n \; {}^{k}\Gamma_n^{\alpha-1} \; \gamma_n \; {}^{k}\hat{g}_n \tag{4.3.3.10}
$$

Notice that the above system, (4.3.3.10), is linear in ${}^{k}\tilde{v}$. Furthermore, its components can be obtained as the equilibrium points, (i.e., $\dot{v}_i \; \rightarrow \; 0$) of the concomitant dynamical system

$$
\dot{v}_n \; - \; {}^{k}\eta_n \; v_n \; = \; \gamma_n \; {}^{k}\hat{g}_n \; [\; \sum_m T_{mn} \; v_m \; + \; {}^{k}\lambda_n \; {}^{k}\Gamma_n^{\alpha-1} \;] \tag{4.3.3.11}
$$

To proceed with our derivation of learning algorithms, we differentiate the steady-state equations, Eqs. (4.3.3.2), with respect to each parameter, p_μ, to obtain the

forward source term, ${}^{\mu}s_n^k$:

$$ {}^{\mu}s_n^k = -\Big([\, -{}^k\tilde{u}_i \,]\, \delta_{p_\mu,\kappa_i} + [\, \delta_{ni}\, g(\gamma_j\, {}^k\tilde{u}_j) \,]\, \delta_{p_\mu,T_{ij}} + [\, T_{ni}\, {}^k\hat{g}_i\, {}^k\tilde{u}_i + \frac{\partial\, {}^kI_n}{\partial\gamma_i} \,]\delta_{p_\mu,\gamma_i} \Big) \qquad (4.3.3.12) $$

Substituting Eq. (4.3.3.12) in (4.2.3.3), and recalling that our abstract response corresponds here to the energy function E, yields

$$ \frac{dE}{dp_\mu} = \frac{\partial E}{\partial p_\mu} + \sum_k {}^k\tilde{v}\, .^{\mu}\, \bar{s}^k $$

The explicit energy gradient contributions for parameters $p_\mu = T,\ \bar{\kappa},\ \bar{\gamma}$ immediately result :

$$ \frac{dE}{dT_{ij}} = \omega_{ij}\, T_{ij} - \omega_{ji}\, T_{ji} - \sum_k {}^k\tilde{v}_i\, g(\gamma_j\, {}^k\tilde{u}_j) \qquad (4.3.3.13) $$

$$ \dot{\kappa}_i = -\tau_\kappa \frac{dE}{d\kappa_i} = -\tau_\kappa \left[\frac{\partial E}{\partial \kappa_i} + \sum_k \frac{\partial E}{\partial {}^k\bar{u}} \frac{\partial {}^k\bar{u}}{\partial \kappa_i} \right] = -\tau_\kappa \left[\frac{\partial E}{\partial \kappa_i} - \sum_k \sum_n {}^k\tilde{v}_n \frac{\partial\, {}^kf_n}{\partial \kappa_i} \right] \qquad (4.3.3.14) $$

Since $\frac{\partial E}{\partial \kappa_i} = 0$ and $\frac{\partial\, {}^k\phi_n}{\partial \kappa_i} = -{}^k\tilde{u}_n\delta_{ni}$ we have

$$ \frac{dE}{d\kappa_i} = \sum_k {}^k\tilde{u}_i \sum_n {}^k\tilde{v}_n \qquad (4.3.3.15) $$

Also, from Eqs. (4.3.2.7), (4.3.3.3) and (4.3.3.8) we can write

$$ \dot{\gamma}_i = -\tau_\gamma \frac{dE}{d\gamma_i} = -\tau_\gamma \left[\frac{\partial E}{\partial \gamma_i} + \sum_k \frac{\partial E}{\partial {}^k\bar{u}} \frac{\partial {}^k\bar{u}}{\partial \gamma_i} \right] = -\tau_\gamma \left[0 - \sum_k {}^k\tilde{v}_n \frac{\partial\, {}^kf_n}{\partial \gamma_i} \right]. \qquad (4.3.3.16) $$

Differentiating Eqn. (4.3.3.12), $\frac{\partial\, {}^k\phi_n}{\partial \gamma_i} = T_n i {}^k\tilde{\varphi}_i\ {}^k\tilde{u}_i$ and substituting in Eqn. (4.3.1.6), yields

$$\frac{dE}{d\gamma_i} = -\sum_k {}^k\lambda_i\ {}^k\Gamma_i^{\alpha-1}\ {}^k\hat{g}_i\ {}^k\tilde{u}_i + \sum_k\sum_n [\, T_{ni}\ {}^k\hat{g}_i\ {}^k\tilde{u}_i + \frac{\partial\, {}^kI_n}{\partial \gamma_i}\,]\ {}^k\tilde{v}_n \quad (4.3.3.17)$$

Substituting Eqs. (4.3.3.2)-(4.3.3.17) into Eqs. (4.3.1.5) and (4.3.1.6), we then obtain the complete learning dynamics.

The above derivation can be summarized to yield computational learning algorithm SID_5.

Input : attractor coordinates, a_n^k, network dimension and topographic partitioning, neural response function, temporal grid, convergence and scaling criteria, initialization domain.

Output : Learned synaptic matrix, T_{nm} and network topology.

Algorithm : Singularity Interaction Dynamics_5

[1] Initialize : $\forall\, n,\ m\ \in\ \{\ 1,\ \cdots,\ N\ \}$ and $\forall\, k\ \in\ \{\ 1,\ \cdots,\ K\}$:

$$T_{nm}^0 = \Re\,[\ -\epsilon,\ +\epsilon\ ]$$
$${}^k\lambda_n^0 = \Re\,[\ -\epsilon,\ +\epsilon\ ]$$
$$\kappa_n^0 = \Re\,[\ -\epsilon,\ +\epsilon\ ]$$
$$\gamma_n^0 = \Re\,[\ -\epsilon,\ +\epsilon\ ]$$

[2] Learn synaptic matrix T : **Iterate** $\nu\ =\ 1,\ \cdots,\ N_T$

initialize outer product array : $\Sigma_{nm}^{\nu} = 0.$

[3] **Loop** over training samples, $k\ =\ 1,\ \cdots,\ K$

[3.1] Evolve network dynamics using Eqs. (4.3.2.1)-(4.3.2.2)

$$\dot{u}_n \; + \; \kappa \; u_n \;\; = \;\; \sum_m T^{\nu}_{nm} \; \varphi(u_m) \; + \; {}^{k}I_n$$

where

$${}^{k}I_n \;\; = \;\; \begin{cases} [a^k_n \; - \; \varphi(u_n)]^{1/3} & \text{if} \;\; n \; \in \; S_X \\ 0 & \text{if} \;\; n \; \in \; S_H \cup S_Y \end{cases}$$

Output: $\varphi({}^{k}\tilde{u}^{\nu}_n)$ and ${}^{k}\tilde{I}^{\nu}_n$

[3.2] Evolve "importance" dynamics using Eqs. (4.3.3.11)

$$\dot{v}_n \;\; - \;\; {}^{k}\eta_n \; v_n \;\; = \;\; \gamma_n \; {}^{k}\hat{g}_n \; [\; \sum_m T_{mn} \; v_m \; + \; {}^{k}\lambda_n \; {}^{k}\Gamma^{\alpha-1}_n \;]$$

Output : ${}^{k}\tilde{v}^{\nu}_n$

[3.3] Update outer product contribution

$$\Sigma^{\nu}_{nm} \;\; = \;\; \Sigma^{\nu}_{nm} \; + \; {}^{k}\tilde{v}^{\nu}_n \; \varphi(\gamma^k_m \tilde{u}^{\nu}_m)$$

[4] **Endloop** over $\{k\}$

[5] Compute $\nabla_T E$

$$(\nabla_T E)^{\nu}_{nm} \;\; = \;\; \Sigma^{\nu}_{nm} \; - \; \omega_{nm} \; T^{\nu}_{nm} \; + \; \omega_{mn} \; T^{\nu}_{mn}$$

[6] Compute $\nabla_\lambda E$

$${}^{k}(\nabla_\lambda E)^{\nu}_n \;\; = \;\; \frac{1}{\alpha} \; {}^{k}\Gamma^{\alpha,\nu}_n$$

[7] Compute τ using Eq. (3.2.4.5)

[8] Update T

$$T^{\nu+1}_{nm} \;\; = \;\; T^{\nu}_{nm} \; + \; \tau\Delta \; (\nabla_T E)^{\nu}_{nm}$$

[9] Update λ using

$$ {}^{k}\lambda_n^{\nu+1} \;=\; {}^{k}\lambda_n^{\nu} \;+\; \tau\,\Delta \frac{\Pi}{\Lambda \;+\; (1/(\Lambda+\theta))}\; {}^{k}[\nabla_\lambda E]_n^{\nu} $$

where

$$ \Pi \;=\; \nabla_T E \oplus \nabla_T E \;+\; \nabla_\kappa E \oplus \nabla_\kappa E \;+\; \nabla_\gamma E \oplus \nabla_\gamma E $$

and ,

$$ \Lambda \;=\; \nabla_\lambda E \oplus \nabla_\lambda E. $$

[10] Update κ_n using Eqs. (4.3.3.15)

[11] Update γ_n using Eqs. (4.3.3.17)

[12] Check for convergence:

If yes **then** exit **else** goto [2]

[13] **Endloop** over learning iterations $\{\nu\}$

[14] **Exit :** Display results

4.4. Summary

In this chapter we have presented a powerful theoretical framework for learning continuous nonlinear mappings using artificial neural networks. Central to our approach is the concept of *adjoint operators* which enables a fast computation of energy function gradients with respect to all system paramters using a single solution of the adjoint equations. In the next chapter, we exploit the powerful mathematical constructs introduced in this chapter to derive fast learning algorithms for dynamical neural networks.

Chapter Five

Adjoint Operator Algorithms for Fast Learning

In this chapter we extend our results on application of adjoint operators to neural learning, presented in Chapter Four, to derive a new computational framework for faster supervised learning in dynamical nonlinear neural networks is presented. It exploits the concept of *adjoint operators* to enable computation of changes in the network's response due to perturbations in all system parameters, using the solution of a single set of appropriately constructed linear equations. The lower bound on speedup per learning iteration over conventional methods for calculating the neuromorphic energy gradient is $O(N^2)$, where N is the number of neurons in the network.

5.1. Introduction

The biggest promise of artificial neural networks as computational tools lies in the hope that they will enable fast processing and synthesis of complex information patterns. In particular, considerable efforts have recently been devoted to the formulation of efficient methodologies for learning (e.g., Rumelhart et al. [251], Pineda [239], Pearlmutter [234], Williams and Zipser [292], Barhen et al.[35,36], Gulati et al. [105,107] and Zak [305-307]). The development of learning algorithms is generally based upon the minimization of a neuromorphic energy function. The fundamental requirement of such an approach is the computation of the gradient of this objective function with respect to the various parameters of the neural architecture, e.g., synaptic weights, neural gains, etc. The

paramount contribution to the often excessive cost of learning using dynamical neural networks arises from the necessity to solve, at each learning iteration, one set of equations for each parameter of the neural system, since those parameters affect both directly and indirectly the network's energy.

In this chapter we show that the concept of adjoint operators (discussed in Section 4.2.), when applied to dynamical neural networks, not only yields a considerable algorithmic speedup, but also puts on a firm mathematical basis prior results for "recurrent" networks, the derivations of which sometimes involved much heuristic reasoning. The applications of adjoint operators to neural learning has been discussed in detail in [38,39] and references therein.

In the sequel we first motivate and construct, in the most elementary fashion, a computational framework based on adjoint operators. We then apply our results to the Cohen-Grossberg-Hopfield (CGH) additive model, enhanced with terminal attractor capabilities. We conclude by presenting the results of a few typical simulations.

5.2. Adjoint Operator Theory

For completeness sake, we proceed by recapitulating some of the key results in nonlinear sensitivity theory, introduced in Section 1.6.2 and later discussed in Section 4.2. Consider, for the sake of simplicity, that a problem of interest is represented by the following system of N coupled nonlinear equations

$$\bar{\varphi}(\bar{u}, \bar{p}) = 0 \tag{5.2.1}$$

where $\bar{\varphi}$ denotes a nonlinear operator. If differential operators appear in Eq. (5.2.1), then a corresponding set of boundary and/or initial conditions to specify

the domain of φ must also be provided. In general an inhomogeneous "source" term can also be present. The learning model discussed in this chapter focuses on the adiabatic approximation only. Nonadiabatic learning algorithms, wherein the response is defined as a functional, will be discussed in a forthcoming article. Let $\bar{u}$ and $\bar{p}$ represent the N-vector of dependent state variables and the M-vector of system parameters, respectively. We will assume that generally $M >> N$ and that elements of $\bar{p}$ are, in principle, independent. Furthermore, we will also assume that, for a specific choice of parameters, a unique solution of Eq. (2.1) exists. Hence, $\bar{u}$ is an implicit function of $\bar{p}$. A system "response", R, represents any result of the calculations that is of interest. Specifically

$$R = R(\bar{u}, \bar{p}) \tag{5.2.2}$$

i.e., R is a known nonlinear function of $\bar{p}$ and $\bar{u}$ and may be calculated from Eq. (5.2.2) when the solution $\bar{u}$ in Eq. (5.2.1) has been obtained for a given $\bar{p}$. The problem of interest is to compute the "sensitivities" of R, i.e., the derivatives of R with respect to parameters p_μ, $\mu = 1, \cdots, M$. By definition

$$\frac{dR}{dp_\mu} = \frac{\partial R}{\partial p_\mu} + \frac{\partial R}{\partial \bar{u}} \cdot \frac{\partial \bar{u}}{\partial p_\mu} \tag{5.2.3}$$

Since the response R is known analytically, the computation of $\partial R/\partial p_\mu$ and $\partial R/\partial \bar{u}$ is straightforward. The quantity that needs to be determined is the vector $\partial \bar{u}/\partial p_\mu$. Differentiating the state equations (5.2.1), we obtain a set of equations to be referred to as "forward" sensitivity equations

$$\frac{\partial \bar{\varphi}}{\partial \bar{u}} \cdot \frac{\partial \bar{u}}{\partial p_\mu} = -\frac{\partial \bar{\varphi}}{\partial p_\mu} \tag{5.2.4}$$

To simplify the notations, we are omitting the "transposed" sign and denoting the N by N forward sensitivity matrix $\partial \bar{\varphi}/\partial \bar{u}$ by A, the N-vector $\partial \bar{u}/\partial p_\mu$ by ${}^\mu\bar{q}$ and the "source" N-vector $-\partial \bar{\varphi}/\partial p_\mu$ by ${}^\mu\bar{s}$. Thus

$$A\, {}^\mu\bar{q} = {}^\mu\bar{s} \tag{5.2.5}$$

Since the source term in Eq. (5.2.5) explicitly depends on μ, computing dR/dp_μ, requires solving the above system of N algebraic equations for each parameter p_μ. This difficulty is circumvented by introducing adjoint operators. Let A^* denote the formal adjoint of the operator A. Adjoint operators can only be considered for densely defined linear operators on Banach spaces (see e.g., Cacuci [310]). For the neural application under consideration we will limit ourselves to real Hilbert spaces. Such spaces are self-dual. Furthermore, the domain of an adjoint operator is determined by selecting appropriate adjoint boundary conditions[1]. The associated bilinear form evaluated on the domain boundary must thus be also generally included. The adjoint sensitivity equations can then be expressed as

$$A^* \; {}^{\mu}\bar{q}^* \;=\; {}^{\mu}\bar{s}^*. \tag{5.2.6}$$

By definition, for algebraic operators

$${}^{\mu}\bar{q}^* \cdot (A \; {}^{\mu}\bar{q}) \;=\; {}^{\mu}\bar{q}^* \cdot {}^{\mu}\bar{s} \;=\; {}^{\mu}\bar{q} \cdot (A^* \; {}^{\mu}\bar{q}^*) \;=\; {}^{\mu}\bar{q} \cdot {}^{\mu}\bar{s}^* \tag{5.2.7}$$

Since Eq. (5.2.3), can be rewritten as

$$\frac{dR}{dp_\mu} \;=\; \frac{\partial R}{\partial p_\mu} + \frac{\partial R}{\partial \bar{u}} \; {}^{\mu}\bar{q}\,, \tag{5.2.8}$$

if we identify

$$\frac{\partial R}{\partial \bar{u}} \;\equiv\; {}^{\mu}\bar{s}^* \;\equiv\; \bar{s}^* \tag{5.2.9}$$

we observe that the source term for the adjoint equations is independent of the specific parameter p_μ. Hence, the *solution of a single set of adjoint equations will provide all the information required to compute the gradient of R with respect to all parameters.* To underscore that fact we shall denote ${}^{\mu}\bar{q}^*$ as $\bar{v}$. Thus

$$\frac{dR}{dp_\mu} \;=\; \frac{\partial R}{\partial p_\mu} - \bar{v} \cdot \frac{\partial \bar{\varphi}}{\partial p_\mu} \tag{5.2.10}$$

We will now apply this computational framework to a CGH network enhanced with terminal attractor dynamics. The model developed in the sequel differs

from our earlier formulations in Chapter Two, in avoiding the use of constraints in the neuromorphic energy function, thereby eliminating the need for differential equations to evolve the concomitant Lagrange multipliers. Also, the usual activation dynamics is transformed into a set of equivalent equations which exhibit more "congenial" numerical properties, such as "contraction".

5.3. Applications to Neural Learning

We formalize a neural network as an adaptive dynamical system whose temporal evolution is governed by the following set of coupled nonlinear differential equations

$$\dot{z}_n \; + \; \kappa_n \; z_n \;\; = \;\; \sum_m \omega_{nm} \; T_{nm} \; g_\gamma(z_m) \;\; + \;\; {}^k I_n \qquad (5.3.1)$$

where z_n represents the mean soma potential of the nth neuron and T_{nm} denotes the synaptic coupling from the $m-$th to the $n-$th neuron. The weighting factor ω_{nm} enforces topological considerations. The constant κ_n characterizes the decay of neuron activity. The sigmoidal function $g_\gamma(\cdot)$ modulates the neural response, with gain given by γ_m; typically, $g_\gamma(z) \; = \; \tanh(\gamma z)$. The "source" term ${}^k I_n$, which includes dimensional considerations, encodes contribution in terms of attractor coordinates of the k-th training sample via the following expression

$${}^k I_n \;\; = \;\; \begin{cases} [{}^k a_n]^{1-\beta} \; [{}^k a_n \; - \; g_\gamma(z_n) \;]^\beta & \textbf{if } n \in S_X \\ 0 & \textbf{if } n \in S_H \cup S_Y \end{cases} . \qquad (5.3.2)$$

The topographic input, output and hidden network partitions S_X, S_Y and S_H are architectural requirements related to the encoding of mapping-type problems for which a number of possibilities exist [35,36]. In previous chapters we have recapitulated that in general, for $\beta \; = \; (2i+1)^{-1}$ and i a strictly positive integer, such attractors have infinite local stability and provide opportunity for learning in real-time. Typically, β can be set to $1/3$. Assuming an adiabatic framework,

the fixed point equations at equilibrium, i.e., as $\dot{z}_n \rightarrow 0$, yield

$$\frac{\kappa_n}{\gamma_n} g^{-1}({}^k\tilde{u}_n) = \sum_m \omega_{nm} T_{nm} {}^k\tilde{u}_m + {}^k\tilde{I}_n \qquad (5.3.3)$$

where $u_n = g_\gamma(z_n)$ represents the neural response. The superscript $\sim$ denotes quantities evaluated at steady state. Operational network dynamics is then given by

$$\dot{u}_n + u_n = g_\gamma \left[\frac{\gamma_n}{\kappa_n} \sum_m \omega_{nm} T_{nm} u_m + \frac{\gamma_n}{\kappa_n} {}^kI_n \right] \qquad (5.3.4)$$

To proceed formally with the development of a supervised learning algorithm, we consider an approach based upon the minimization of a constrained "neuromorphic" energy function E given by the following expression

$$E(\bar{u}, \bar{p}) = \frac{1}{2} \sum_k \sum_n [\, {}^k\tilde{u}_n - {}^ka_n \,]^2 \qquad \forall\, n \in S_X \cup S_Y \qquad (5.3.5)$$

We relate adjoint theory to neural learning by identifying the neuromorphic energy function, E in Eq. (5.3.5), with the system response R. Also, let $\bar{p}$ denote the following system parameters:

$$\bar{p} = \{\, T_{11}, \cdots T_{NN} \mid \kappa_1, \cdots \kappa_N \mid \gamma_1, \cdots \gamma_N \mid \cdots \}$$

The proposed objective function enforces convergence of every neuron in S_X and S_Y to attractor coordinates corresponding to the components in the input-output training patterns, thereby prompting the network to learn the embedded invariances. Lyapunov stability requires an energy-like function to be monotonically decreasing in time. Since in our model the internal dynamical parameters of interest are the synaptic strengths T_{nm} of the interconnection topology, the characteristic decay constants κ_n and the gain parameters γ_n this implies that

$$\dot{E} = \sum_n \sum_m \frac{dE}{dT_{nm}} \dot{T}_{nm} + \sum_n \frac{dE}{d\kappa_n} \dot{\kappa}_n + \sum_n \frac{dE}{d\gamma_n} \dot{\gamma}_n < 0 \qquad (5.3.6)$$

For each adaptive system parameter, p_μ, Lyapunov stability will be satisfied by the following choice of equations of motion

$$\dot{p}_\mu = -\tau_p \frac{dE}{dp_\mu} \tag{5.3.7}$$

Examples include

$$\dot{T}_{nm} = -\tau_T \frac{dE}{dT_{nm}} \quad ; \quad \dot{\gamma}_n = -\tau_\gamma \frac{dE}{d\gamma_n} \quad ; \quad \dot{\kappa}_n = -\tau_\kappa \frac{dE}{d\kappa_n}$$

where the time-scale parameters τ_T, τ_κ and $\tau_\gamma > 0$. Since E depends on p_μ both directly and indirectly, previous methods required solution of a system of N equations for each parameter p_μ to obtain dE/dp_μ from $d\tilde{u}/dp_\mu$. Our methodology (based on **adjoint operators**), yields all derivatives $dE/dp_\mu, \forall\, \mu$, by solving a single set of N linear equations.

Using the neurodynamics, Eqs. 5.3.1, the nonlinear neural operator for each training pattern k, $k = 1, \cdots K$, at equilibrium is given by

$${}^k\varphi_n\,({}^k\tilde{u}, \bar{p}) = g\left[\frac{1}{\kappa_n}\sum_{m'} \omega_{nm'}\, T_{nm'}\, {}^k\tilde{u}_{m'} + \frac{1}{\kappa_n}\, {}^k\tilde{I}_n\right] - {}^k\tilde{u}_n = 0 \tag{5.3.8}$$

where, without loss of generality we have set γ_n to unity. So, in principle ${}^k\tilde{u}_n = {}^k\tilde{u}_n\,[T, \bar{\kappa}, \bar{\gamma}, {}^k a_n, \cdots]$. Using Eqs. (5.3.8), the forward sensitivity matrix can be computed and compactly expressed as

$$\begin{aligned} {}^k A_{nm} = \frac{\partial\, {}^k\varphi_n}{\partial\, {}^k\tilde{u}_m} &= {}^k\hat{g}_n \frac{1}{\kappa_n}\left[\omega_{nm}\, T_{nm} + \frac{\partial\, {}^k\tilde{I}_n}{\partial\, {}^k\tilde{u}_m}\right] - \delta_{nm} \\ &= \frac{1}{\kappa_n}\, {}^k\hat{g}_n\, w_{nm}\, T_{nm} - {}^k\eta_n\, \delta_{nm}. \end{aligned} \tag{5.3.9}$$

where

$$ {}^k\eta_n \;=\; \begin{cases} 1 \;+\; \frac{[{}^k a_n]^{2/3}}{3\kappa_n}\; {}^k\hat{g}_n\; [{}^k a_n - {}^k\tilde{u}_n\,]^{-2/3} & \textbf{if } n \;\in\; S_X \\ 1 & \textbf{if } n \;\in\; S_H \cup S_Y \end{cases} . \qquad (5.3.10) $$

Above, ${}^k\hat{g}_n$ represents the derivative of g with respect to ${}^k\tilde{u}_n$, i.e., if $g \equiv tanh$, then

$$ {}^k\hat{g}_n \;=\; 1 \;-\; [{}^k g_n]^2 $$

where

$$ {}^k g_n \;=\; g\Big[\; \frac{1}{\kappa_n} \left(\sum_m \omega_{nm}\; T_{nm}\; {}^k\tilde{u}_m \;+\; {}^k\tilde{I}_n \right) \Big] \qquad (5.3.11) $$

Recall that the formal adjoint equation is given as $A^*\bar{v} \;= \bar{s}^*$; here

$$ {}^k A^*_{nm} \;=\; \frac{1}{\kappa_m}\; {}^k\hat{g}_m\; \omega_{mn}\; T_{mn} \;-\; {}^k\eta_m\; \delta_{mn} \qquad (5.3.12) $$

Using Eqs. (5.2.9) and (5.3.5), we can compute the formal **adjoint source**

$$ {}^k s^*_n \;\equiv\; \frac{\partial E}{\partial\, {}^k\tilde{u}_n} \;=\; \begin{cases} {}^k\tilde{u}_n \;-\; {}^k a_n & \textbf{if } n \in S_X \cup S_Y \\ 0 & \textbf{if } n \in S_H \end{cases} \qquad (5.3.13) $$

The system of adjoint fixed-point equations can then be constructed using Eqs. (5.3.12) and (5.3.13), to yield :

$$ \sum_m \frac{1}{\kappa_m}\; {}^k\hat{g}_m\; \omega_{mn}\; T_{mn}\; {}^k\tilde{v}_m - \sum_m {}^k\eta_m\; \delta_{mn}\; {}^k\tilde{v}_m \;=\; {}^k s^*_n \qquad (5.3.14) $$

Notice that the above coupled system, (5.3.14), is linear in ${}^k\tilde{v}$. Furthermore, it has the same mathematical characteristics as the operational dynamics (5.3.4).

Its components can be obtained as the equilibrium points, (i.e., $\dot{v}_i \rightarrow 0$) of the **adjoint neural dynamics**

$$\dot{v}_n + {}^k\eta_n \, v_n = \sum_m \frac{1}{\kappa_m} {}^k\hat{g}_m \, \omega_{mn} \, T_{mn} \, v_m - {}^k s_n^* \tag{5.3.15}$$

As an implementation example, let us conclude by deriving the learning equations for the synaptic strengths, T_μ. Recall that

$$\frac{dE}{dT_\mu} = \frac{\partial E}{\partial T_\mu} + \sum_k {}^k\tilde{v} \cdot {}^{\mu k}\bar{s} \qquad \mu = (i,j) \tag{5.3.16}$$

We differentiate the steady state equations (5.3.8) with respect to T_{ij}, to obtain the forward source term,

$$\begin{aligned} {}^{\mu k}s_n = -\frac{\partial \, {}^k\varphi_n}{\partial T_{ij}} &= -{}^k\hat{g}_n \left[\frac{1}{\kappa_n} \sum_l \omega_{nl} \, \delta_{in} \, \delta_{jl} \, {}^k\tilde{u}_l + 0 \right] \\ &= -\frac{1}{\kappa_n} {}^k\hat{g}_n \, \delta_{in} \, \omega_{nj} \, {}^k\tilde{u}_j \end{aligned} \tag{5.3.17}$$

Since by definition, $\partial E / \partial T_{nm} = 0$, the explicit energy gradient contribution is obtained as

$$\dot{T}_{nm} = -\tau_T \left[-\frac{\omega_{nm}}{\kappa_n} \sum_k {}^k\tilde{v}_n \, {}^k\hat{g}_n \, {}^k\tilde{u}_m \right] \tag{5.3.18}$$

We compute the adaptive learning rates, i.e., τ_p in Eq.(5.3.7), along the lines specified in Section 3.2.4. Without loss of generality, we shall assume $\tau_T = \tau_\kappa = \tau_\gamma = \tau$, and we shall seek τ in the form []

$$\tau \propto |\nabla E|^{-\beta} \tag{5.3.19}$$

where ∇E denotes the vector with components $\nabla_T E$, $\nabla_\gamma E$ and $\nabla_\kappa E$. It is straightforward to show that

$$\frac{d}{dt} \mid \nabla E \mid = -\lambda \mid \nabla E \mid^{1-\beta} \tag{5.3.20}$$

as ∇E tends to zero, where λ is an arbitrary positive constant. If we evaluate the relaxation time of the energy gradient, we find that

$$t_E = \int_{|\nabla E|_0}^{|\nabla E| \to 0} \frac{d \mid \nabla E \mid}{\mid \nabla E \mid^{1-\beta}} = \begin{cases} \infty & \text{if } \beta \leq 0 \\ \frac{1}{\beta} \mid \nabla E \mid_0^\beta < \infty & \text{if } \beta > 0 \end{cases} \tag{5.3.21}$$

Thus, for $\beta \leq 0$ the relaxation time is infinite, while for $\beta > 0$ it is finite. The dynamical system (5.3.19) suffers a qualitative change for $\beta > 0$: it loses uniqueness of solution. The equilibrium point $\mid \nabla E \mid = 0$ becomes a singular solution being intersected by all the transients, and the Lipschitz condition is violated, as one can see from

$$\frac{d}{d \mid \nabla E \mid} \left(\frac{d \mid \nabla E \mid}{dt} \right) = -\lambda \mid \nabla E \mid^{-\beta} \longrightarrow -\infty$$

where $\mid \nabla E \mid$ tends to zero, while β is strictly positive. Such infinitely stable points are "terminal attractors". By analogy with our previous results we choose $\beta = 2/3$, which yields

$$\tau = \left(\sum_n \sum_m [\nabla_T E]_{nm}^2 + \sum_n [\nabla_\gamma E]_n^2 + \sum_n [\nabla_\kappa E]_n^2 \right)^{-1/3} \tag{5.3.22}$$

The introduction of these adaptive time-scales dramatically improves the convergence of the corresponding learning dynamical systems.

We now summarize the complete neural learning formalism.

5.4. Algorithm SID_6

Input : Network dimension and topographic partitions; Attractor coordinates, ${}^k a_n$; Neural response function, g; Topology matrix, w; Temporal grid and convergence criteria; Initialization domain and scaling parameters;

Output : Learned synaptic interconnection matrix, T;

Algorithm : Singularity Interaction Dynamics_6

[1] Initialize : $\forall\, n,\ m\ \in\ \{\ 1,\ \cdots,\ N\ \}$ and $\forall\, k\ \in\ \{\ 1,\ \cdots,\ K\}$:

$$T^0_{nm} \ = \ \Re\,[\ -\epsilon,\ +\epsilon\ ]$$
$$^k\lambda^0_n \ = \ \Re\,[\ -\epsilon,\ +\epsilon\ ]$$

[2] Learn synaptic matrix T : **Iterate** $\nu\ =\ 1,\ \cdots,\ N_T$

initialize outer product array : $\Sigma^\nu_{nm}\ =\ 0.$

[3] **Loop** over training samples, $k\ =\ 1,\ \cdots,\ K$

[3.1] Evolve network dynamics using Eqs. (5.3.1)

$$\dot{z}_n\ +\ \kappa_n\ z_n\ \ =\ \ \sum_m \omega_{nm}\ T_{nm}\ g_\gamma(z_m)\ \ +\ \ {}^k I_n$$

where from Eqs. (5.3.2)

$${}^k I_n\ =\ \begin{cases} [{}^k a_n]^{1-\beta}\ [{}^k a_n\ -\ g_\gamma(z_n)\ ]^\beta & \textbf{if } n \in S_X \\ 0 & \textbf{if } n \in S_H \cup S_Y \end{cases}$$

Output: ${}^k\tilde{u}^\nu_n$ and ${}^k\tilde{I}^\nu_n$

[3.2] Evolve "importance" dynamics, Eqs. (5.3.15), using

$$\dot{v}_n\ +\ {}^k\eta_n\ v_n\ \ =\ \ \sum_m \frac{1}{\kappa_m}\ {}^k\hat{g}_m\ \omega_{mn}\ T_{mn}\ v_m\ -\ {}^k s^*_n$$

where

$$ {}^k s_n^* \;=\; \begin{cases} {}^k\tilde{u}_n \;-\; {}^k a_n & \textbf{if } n \in S_X \cup S_Y \\ 0 & \textbf{if } n \in S_H \end{cases} $$

and from Eqs. (5.3.11)

$$ {}^k\hat{g}_n \;=\; 1 \;-\; [{}^k g_n]^2 $$

$$ {}^k g_n \;=\; g\Big[\; \frac{1}{\kappa_n} \left(\sum_m \omega_{nm}\, T_{nm}\, {}^k\tilde{u}_m + {}^k\tilde{I}_n \right) \Big] $$

Output : ${}^k\tilde{v}_n^\nu$

[3.3] Update outer product contribution

$$ \Sigma_{nm}^\nu \;=\; \Sigma_{nm}^\nu \;-\; \frac{1}{\kappa_n}\, {}^k\tilde{v} \cdot\; {}^k\hat{g}_n\;\; \omega_{nm}\; {}^k\tilde{u}_m $$

[4] **Endloop** over $\{k\}$

[5] Compute $\nabla_T E$

$$ (\nabla_T E)_{nm}^\nu \;=\; \Sigma_{nm}^\nu $$

[6] Compute τ using Eq. (5.3.22)

$$ \tau \;=\; \left(\sum_n \sum_m [\; \nabla_T E \;]_{nm}^2 \;+\; \sum_n [\nabla_\gamma E]_n^2 \;+\; \sum_n [\nabla_\kappa E]_n^2 \right)^{-1/3} $$

[7] Update T

$$ T_{nm}^{\nu+1} \;=\; T_{nm}^\nu \;-\tau_T\, \Delta \left[\; -\frac{\omega_{nm}}{\kappa_n} \sum_k {}^k\tilde{v}_n\; {}^k\hat{g}_n\; {}^k\tilde{u}_m \;\right] $$

[8] Check for convergence:

If yes **then** exit **else** goto [2]

[9] **Endloop** over learning iterations $\{\nu\}$

[10] **Exit** : Display results

5.5. Simulation Results

The computational framework developed in the preceding section has been applied to a number of problems that involve learning nonlinear mappings, including Exclusive-OR, the hyperbolic tangent and trigonometric functions, e.g., sin. Some of these mappings (e.g., XOR) have been extensively benchmarked in the literature, and provide an adequate basis for illustrating the computational efficacy of our proposed formulation. Figures 5.5.1(a)-5.5.1(d) demonstrate the temporal profile of various network elements during learning of the XOR function. A six neuron feedforward network was used, that included self-feedback on the output unit and bias. Fig. 5.5.1(a) shows the LMS error during the training phase. The worst-case convergence of the output state neuron to the presented attractor is displayed in Fig. 5.5.1(b). Notice the rapid convergence of the input state due to the terminal attractor effect. The behavior of the adaptive time-scale parameter τ is depicted in Fig. 5.5.1(c). Finally, Fig. 1(d) shows the evolution of the energy gradient components.

The test setup for signal processing applications, i.e., learning the sin function and the tanh sigmoidal nonlinearlity, included a 8-neuron fully connected network with no bias. In each case the network was trained using as little as 4 randomly sampled training points. Efficacy of recall was determined by presenting 100 random samples. Fig. 5.5.2 and 5.5.3(a)-(b) illustrate that we were able to approximate the sin and the hyperbolic tangent functions using 16 and 4 pairs

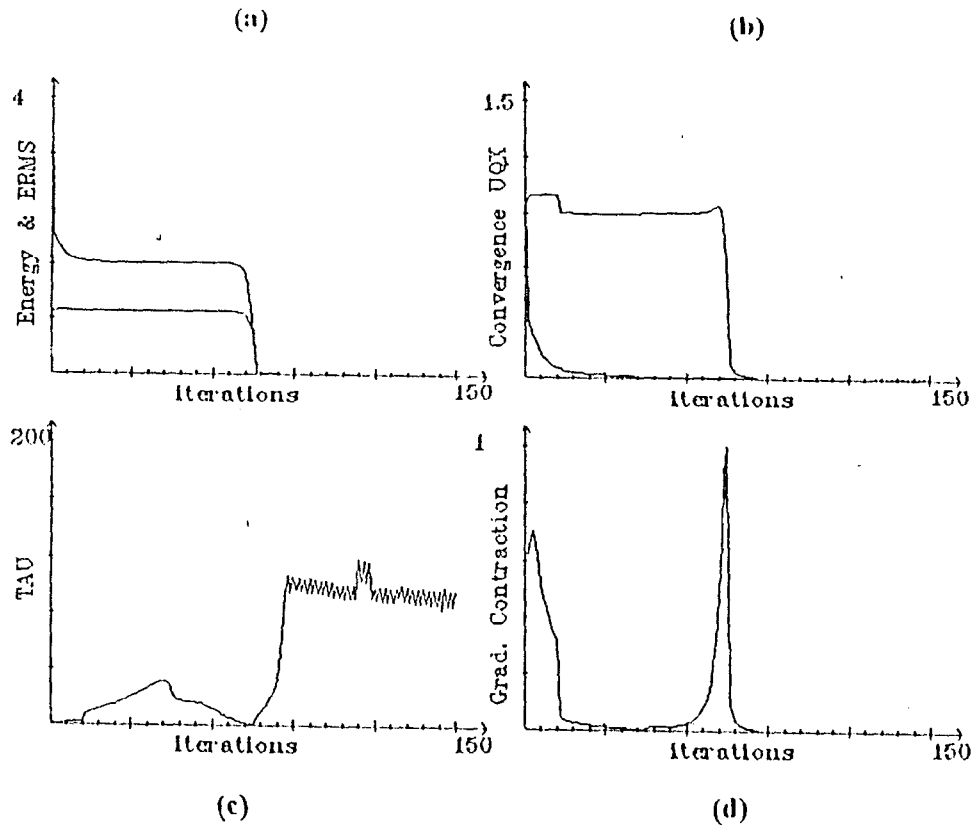


Figure 5.5.1(a-d). Temporal evolution of state variables, RMS error, time scales and energy gradient contractions during learning of Exclusive-OR (XOR) function.

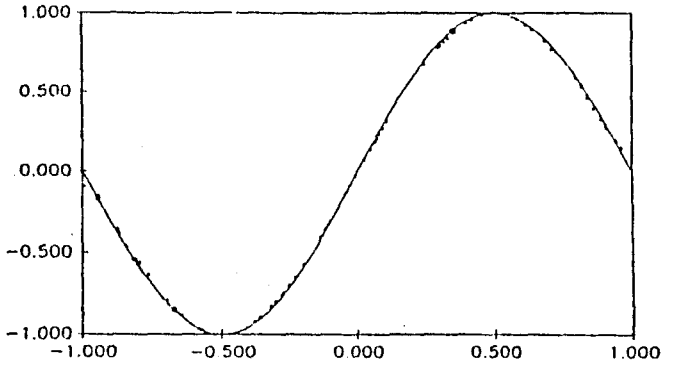
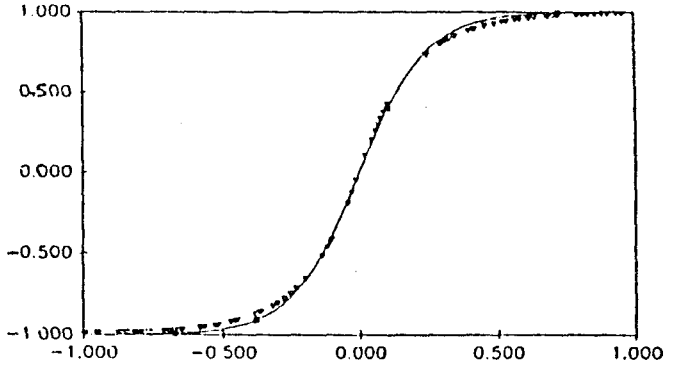


Figure 5.5.2. Learning the Sin function using a fully connected, 8-neuron network with no bias. Training set comprised of 4 randomly chosen points.

(a)



(b)

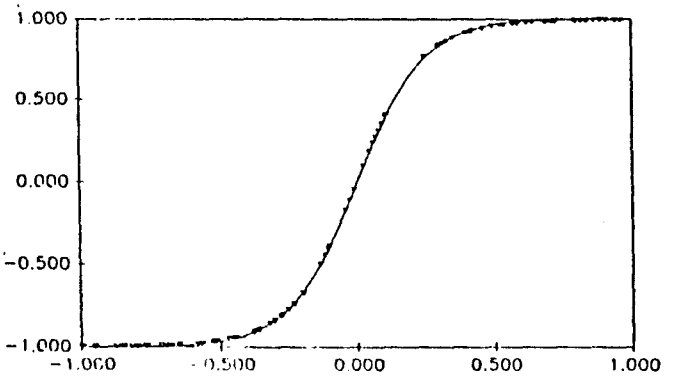
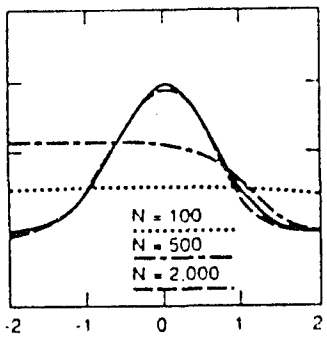


Figure 5.5.3. Learning the Hyberbolic tangent function using a fully connected 8-neuron network with no bias. (a) Using 4 randomly selected samples; (b) Using 16 randomly selected training samples.



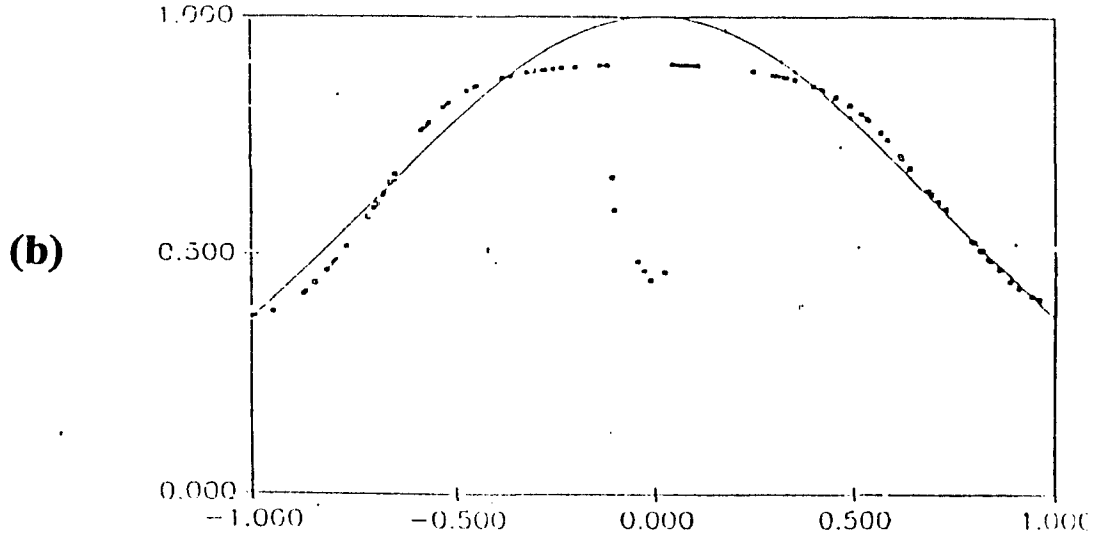


Figure 5.5.4. Approximating the Gaussian pulse function. (a) Results obtained by Lippmann and Beckman [189]. (b) Results using ***Algorithm SID_6.***

respectively. Fig. 5.5.3(a) demonstrates the network performance when 4 pairs were used to learn the hyperbolic tangent. Fig. 5.5.4 and Fig. 5.5.5 show our results while learning the Gaussian pulse. These results reflect a performance improvement of over two orders of magnitude as compared to state-of-the-art supervised neural learning algorithms [189,239,251], both in terms of learning time and the number of training sample. Further, our models require atleast an order of magnitude less number of neurons as compared to the backpropagation algorithm. For example, as shown in Fig. 5.5.6, Lippmann and Beckman in [189] required over 2000 samples to learn the Gaussian pulse function using a 27 neuron, multilayered perceptron with linear input and output nodes. The network comprised of 1 unit in the input layer, 20 units in the hidden layer, 5 units in the second hidden layer and a single unit output layer.

We would like to mention that since our learning methodology involves terminal attractors, extreme caution must be exercised when simulating the algorithms in a digital computing environment. Our discussion on sensitivity of results to the integration schemes in Chapter Three emphasizes that explicit methods such as Euler or Runge-Kutta shall not be used, since the presence of terminal attractors induces extreme stiffness. Practically, this would require an integration time-step of infinitesimal size, resulting in numerical round-off errors of unacceptable magnitude. Implicit integration techniques such as the Kaps-Rentrop scheme should therefore be used.

5.6. Summary

In this chapter we have presented a theoretical framework for faster learning in dynamical neural networks. Central to our approach is the concept of *adjoint operators* which enables computation of network neuromorphic energy gradients with respect to all system parameters using the solution of a single set of linear

equations. If C_F and C_A denote the computational costs associated with solving the forward and adjoint sensitivity equations (Eqs. 2.5 and 2.6), and if M denotes the number of parameters of interest in the network, the speedup achieved is

$$S^{F-A} = \frac{M\ C_F}{C_A}$$

If we assume that $C_F \simeq C_A$ and that $M = N^2 + 2N + \cdots$, we see that the lower bound on speedup per learning iteration is $O(N^2)$. Finally, particular care must be exercised when integrating the dynamical systems of interest, due to the extreme stiffness introduced by the terminal attractor constructs.

Chapter Six

"Chaotic Relaxation" in Concurrently Asynchronous Neural Networks

Upto this stage our focus has been primarily on deriving a computationally enabling framework for learning using dynamical neural networks. In this chapter, we now analyze a fundamental issue which directly impacts the scalability of current theoretical neural network models to applicative embodiments. in both software as well as hardware. This pertains to the inherent and unavoidable concurrent asynchronicity of emerging fine-grained computational ensembles and the consequent chaotic manifestations in the absence of proper conditioning. The latter concern is particularly significant since the computational inertia of neural networks in general and our dynamical learning formalisms manifests itself substantially, only in massively parallel hardware - optical, VLSI or optoelectronic. We introduce a mathematical framework for systematically reconditioning additive-type models and derive a neuro-operator, based on the chaotic relaxation paradigm, whose resulting dynamics is neither "concurrently" synchronous nor "sequentially" asynchronous. Necessary and sufficient conditions guaranteeing concurrent asynchronous convergence are established in terms of contracting operators. Lyapunov exponents are also computed to characterize the network dynamics and to ensure that throughput-limiting "emergent computational chaos" behavior in models reconditioned with concurrently asynchronous algorithms was eliminated.

The organization of this chapter is as follows: In Section 6.2. we present a characterization of asynchronous iterative computation and introduce the chaotic

relaxation paradigm. Section 6.3. describes the reformulation of the Hopfield model in terms of contracting neuro-operators. We further define the necessary and sufficient conditions for convergence. To guide the implementation and validation of our asynchronous neuro-operator, we simulate an associative memory model in a concurrent environment in section 6.4. In section 6.5., we compute the Lyapunov exponents, for both the existing and modified model, to provide a fundamental insight to the network dynamics and to dispel potential misgivings as to the main origin of oscillatory behavior observed hitherto.

6.1. Introduction

The bulk of existing neural network models are defined as an aggregation of adaptive dynamical systems, interacting through a densely interconnected synaptic network. Computation to be performed by the network is then encoded in terms of the connection strengths between pairs of neurons. Based on mathematical idealizations to biological behavior, the updating regimes of existing models, both discrete and continuous, may be classified into two basic algorithmic modes [68,135,111,315]. A *concurrent synchronous* mode, where all neurons are updated simultaneously and an *asynchronous* mode, wherein only one randomly selected neuron is allowed to update its state on the basis of its inputs. The firing decision for a particular neuron is allowed only after state information has been received from all other neurons to which it is connected. However, computational connotation of asynchronicity implies uncoordinated systemwide concurrent activity, while the biological manifestation of global asynchrony results from delays in nerve signal propagation, refractory periods and adaptive thresholding [190,198]. So, under their current framework, neural network models *per se* do not compute using a systemwide on-off uncoordinated switching (with random delays) of individual neurons. In fact, asynchrony as discussed by [68,135], in the context of existing artificial neural networks is essentially a

"sequential randomness", lacking implicit concurrency. Thus, the paradigmatic advantages of neuronal processing, that essentially stem from an ability to perform massively parallel, asynchronous and distributed information processing, cannot be fully realized under the existing neurodynamical relaxation models. Further, the biggest promise of artificial neural networks as computational tools lies in the hope that they will resemble the information processing in biological systems. Notwithstanding many successes along such intent, as indicated in Chapter One, it is rapidly becoming evident that current neurodynamical models are plagued by fundamental limitations. We elucidate the argument using the Cohen-Grossberg-Hopfield (CGH) additive model [98] (that provided the basis of our preceding derivations), as a typical representative of artificial neural networks:

$$\dot{u}_i + \kappa_i u_i = \sum_{j=1}^{n} T_{ij} g_\gamma(u_j) + I_i, \qquad i = 1, \cdots, n \tag{6.1.1}$$

where $u_i(t)$ is the mean soma potential of the $i-$th neuron, T_{ij} are the synaptic interconnections, and g_γ is the sigmoidal function modulated by gain γ. The external input is denoted by I_i. It is claimed that the neuronal performance in this model is collective, but not parallel (Barhen et al. in [81],[29,306]). For example, a small perturbation in the activity of the $i-$th neuron instantaneously affects all other neurons:

$$\frac{\partial \dot{u}_i}{\partial u_j} = \frac{dg_\gamma}{du_j} T_{ij} \neq 0 \qquad i \neq j \tag{6.1.2}$$

In contradistinction, the biological systems exhibit both collective and parallel performances. For instance, the different limbs in the human body mechanically independent. Their performance is parallel. At the same time they exhibit collective performance since their activity is coordinated by the brain.

We analyze the implications of this lack of *true* behavioral asynchrony at two levels, namely, problems encountered during digital VLSI, optical or opto-

electronic computations and during discrete-time simulations on large-scale asynchronous computational ensembles. Concurrent synchronous activation typically requires complex global synchronization circuitry to neutralize the clock skew effects resulting from the variations in the physical [209,224] or optical path lengths [257] of the actual synaptic interconnections. This extra circuitry would limit the overall network performance to operate at the rate of slowest neuron (i.e., one with the largest time constant). An alternate strategy involves clocking the system at a time constant slower than the slowest neuron, but again, throughput suffers. Not only does such circuitry lack a biological basis, it also enforces rigid firing sequences that are often difficult to sustain because of signal leakages and component instability. Macukow et al. [198], showed that in large-scale networks such self-induced pathological activation could destabilize the entire neuromorphic system.

Even though the computational and paradigmatic gains expected from neuromorphic architectures will actually manifest from systems built around massively parallel and analog hardware, discrete-time simulations on asynchronous multiprocessors remain the primary benchmarking testbed for large-scale problems. Therefore, the algorithmic implications of "sequential" or the tightly coupled "synchronous" nature of neuronal interrogation in simulations cannot be ignored. In general, during synchronous computation the processors must communicate their partial results to each other, at every instance of time specified by the precedence-constrained task graph obtained from the problem decomposition [37,109,112,172]. Hence, the distributed concurrent algorithms are mathematically equivalent to the sequential algorithms. These overheads, in the form of load imbalance due to processor inactivity, lower processor utilization and enhance resource contention due to communication and coordination requirements, and lead to a severe performance degradation in real-time neural network applications [113]. For instance, simulating the sequentially asynchronous nature

of backpropagating networks on a concurrent computer introduces latencies, for which specific time bounds can be obtained in terms of the critical path of the corresponding task graph. Additionally, the existing approaches are lacking in fault tolerance, as updating decisions by a particular neuron require global interrogation, i.e., the status of each neuron to which it is connected. Failure to receive firing input from some inoperative neuron in the sequentially asynchronous setup could lead to blocking of the entire network. Thus, a model is necessitated wherein each neuron is associated with a decision algorithm that requires only local information to reach globally optimal decisions, as in the cellular automata approach. This also precludes the necessity for neural signals or activation potentials to remain stable for long intervals as in synchronous implementation.

In this chapter we introduce a mathematical framework for reconditioning artificial neural network algorithms such that their embodiments are truly asynchronous and concurrent. For illustrative purposes, the discussion will focus on CGH-type additive networks. The ideas however, generalize in a straightforward fashion to other classes of neurodynamics, e.g., shunting type [95]. Hereafter, we do not distinguish between the simulation of a neural network on a concurrent computer or its subsequent hardware implementation, i.e., a neuron is considered as a "virtual" computing processor. Besides yielding a closer emulation of biological information processing, this approach is expected to provide guidance for large-scale fabrication of concurrent hardware.

6.2. "Chaotic Relaxation" Paradigm

In order to obviate the throughput-limiting "Feigenbaum bottleneck" arising from an extensive usage of the cooperative problem-solving approach, and the resulting "sequentiality" in the neurodynamical activation profile, we introduce

a *chaotic relaxation* paradigm in the neural network dynamics. It is inspired from the seminal work of Chazan and Miranker [67], who showed that chaotic relaxation schemes could significantly reduce programming effort, communication overheads and turnaround time during concurrent computing on asynchronous multiprocessors. In our effort to design truly asynchronous neural networks we further draw motivation from *fixed point* techniques by Baudet [45], Miellou [212], Kung [172] and Bertsekas [48]. But before introducing the chaotic relaxation schema in conjunction to neurocomputation, we briefly summarize key attributes abstracted from concurrent asynchronous computational algorithms, to reinforce the paradigmatic divergence of neurodynamical relaxation in existing models from the biological phenomena and future hardware embodiments.

6.2.1. Concurrent Asynchronous Computation

In contrast to the synchronized iterative techniques (see Kung [172]), the execution profile of concurrent asynchronous algorithms is not constrained by the underlying task decomposition graph for the problem. Concurrent tasks capable of uncoordinated execution are implemented as a collection of functionally, but not dynamically, cooperating processes, with no explicit dependencies to enforce waiting at synchronization points for the purpose of swapping partially computed results. Thus, instead of waiting for specific inputs from other tasks, they may continue, or terminate according to whatever information is available in the state variables. The computation *per se* is essentially iterative in nature, with the dynamics controlled by state variables and, possibly, previous history.

Thus, asynchronous computation provides an implicitly effective strategy for designing systems capable of delivering high throughput and real-time operational responses, since the synchronization and coordination restrictions are

eliminated, and computations can be carried out without having to wait to receive all the messages implied by the precedence constraints. Also, the chaotic relaxations during asynchronous computation stagger data communication and memory accesses, alleviating the Von Neumann bottleneck [31]. In a neural network model, this implies that in contradistinction to the existing schema, computational functions could be implemented using neurons that are allowed to fire without having to wait to receive excitatory or inhibitory input signals from **all** other neurons to which they are connected, in order to evaluate if a firing threshold is exceeded.

In addition, asynchronous dynamics may lead to *true* fault tolerance, as it will enable neurons to remain idle for finite periods. This is analogous to the existence of "refractory" of recharging period in biological neurons as discussed by Choi [69]. But more important is the implication for hardware embodiments. Elimination of inter-neuron dependence, facilitates immediate replacement (or rerouting) of the failed segment, and resumption of processing without disturbing or reinitializing the entire configuration. Another advantage is the potential for implementation on large-scale heterogeneous computational ensembles, i.e., systems in which the different processing nodes may have different performance capabilities, to achieve hierarchical neuronal processing. The latter ability would lead to a reduction of complexity in interleaving operations, to provide for unpredictable activity fluctuations during neurocomputation. In summary, concurrently asynchronous dynamics defines an operational framework that implicitly confers to that essential for neurocomputation.

6.2.2. Concurrent Asynchronous Neurodynamics

In the subsequent development of our theory on concurrently asynchronous neural networks, we adopt a terminology in line with the generalized definition of

chaotic iterations, originally introduced by Chazan and Miranker [67], and later generalized by Baudet [45]. Consider an additive-type neural network with N neurons, and let $\bar{u}$ denote the continuous-valued configuration vector of neuron activations in $\Re^n$. Let the components of $\bar{u}$ be given by u_i, for $i = 1, \cdots, N$. A temporal state sequence in terms of the neural coordinates u_i will be denoted by $u_i(t)$, for $t = 0, 1, \ldots$. Let $\bar{\varphi}$ be the nonlinear network operator from $\Re^n$ to $\Re^n$, whose components will be expressed as $\varphi_i(u_1, u_2, \cdots, u_N)$.

A concurrently asynchronous neural iteration, denoted by the tuple $(\bar{\varphi}, \bar{u}(0), \xi, \psi)$, corresponding to the neuro-operator $\bar{\varphi}$, and starting with a given vector $\bar{u}(0)$, is then a sequence of state iterates, $\bar{u}(t)$, of vectors on $\Re^n$, defined recursively by:

$$u_i(t) = \begin{cases} u_i(t-1) & \text{if } i \notin S_t \\ \varphi_i(u_1(x_1(t)), \ldots, u_n(x_n(t))) & \text{if } i \in S_t \end{cases} \qquad (6.2.2.1)$$

where S_t: $1 \leq | S_t | \leq N$ denotes the set of neurons that update during the $t-$th iterate, and $x_i(t)$, indexes the availability of the $i-$th neuron's *most recent* updated state. Previous updating regimes implicitly assumed $x_i(t) = t-1$. The set $\xi = \{S_t \mid t = 1, 2, \cdots\}$ is a sequence of *nonempty* subsets of neurons, that fired during each successive iterate. Also, $\psi = \{ (x_1(t), \cdots, x_N(t)) \mid t = 1, 2, \cdots \}$ denotes the latest update configuration for the network with respect to the $t-$th iterate. In addition, the following assumptions are made on sets ξ and ψ :

(a) $x_i(t) \leq t - 1$, $t = 1, 2, \cdots$, [6] i.e., each subsequent neuronal update uses only previously available state information;

(b) $x_i(t)$, considered as a function of t, tends to infinity as t tends to infinity [45], i.e., more and more recent state information must be used in evolving the set of neurons in the network;

(c) non-starvation condition [45], i.e., i occurs infinitely many often in the update-sets S_t, for $t = 1, 2, \ldots$, i.e., $\exists\ s < \infty$ such that each neuron is considered at least once in every s successive updates.

Manifestation of chaotic relaxation in neural networks with concurrent asynchronous updating can be intuited as follows: at some operating instant t, an idle neuron, i, initiates the update of its state to $u_i^t = \varphi_i(\bar{u}_{N-\{i\}}^{t-1})$. If the state $\bar{u}(t)$ differs from $\bar{u}(t-1)$ by a set of components $\{\ u_i \mid i \in S_t\ \}$, then the $i-$th neuron may update itself using state information already available from the previous updates, and not wait to receive the results of ongoing activations. The precise strategy for selecting state information available depends on the degree of synchronization desired in the system. For example, in a fully asynchronous operation, the most recently available states could be selected. Alternately, in the vein of Chazan and Miranker [67], a more restricted selection could be considered, which limits the choice of available components to those that are produced no prior to some fixed number, k, of steps, such that for $t = 1, 2, \cdots$, the inequality $t - x_i(t) \leq k$ be satisfied. We now state a criterion for the asynchronous convergence of the neuro-operator, $\bar{\varphi}$.

6.2.3. Contraction Theorems

Amari [11,13] and others [75,127,128,134-136,207] have shown that, in general, the phenomenology of nonlinear neural networks, modeled as adaptive dynamical systems, is essentially a phase space flow towards static attractors. Associative recall, combinatorial optimization, learning, etc. on the other hand, are merely different functional manifestations of this phenomenology, wherein

the nature of activation neurodynamics exercises an integral regulatory influence on functional efficacy, stability and scalability. We exploit this commonality in dynamical behavior to derive in the sequel a contracting neuro-operator, that significantly enhances the scalability of existing systems to massively parallel asynchronous embodiments.

The concept of contraction plays a fundamental role in the iterative solution of nonlinear equations. It is most useful [226] to express contraction in terms of vector norms, which induces a partial ordering on $\Re^n$.

Definition [226] : An operator $\bar{\varphi} : D \subset \Re^n \rightarrow \Re^n$ is called a Φ-contraction on a set $D_o \subset D$, if there exists a linear operator $\Phi \in L(\Re^n)$ with the following properties :

[i] $\mid \bar{\varphi}(\bar{u}) - \bar{\varphi}(\bar{v}) \mid \leq \Phi \mid \bar{u} - \bar{v} \mid \quad \forall\, \bar{u},\, \bar{v} \in D_o$

[ii] $\Phi \geq 0$

[iii] $\rho(\Phi) < 1$

The first property implies Lipschitz-continuity; indeed Φ is often referred to as the Lipschitz matrix of $\bar{\varphi}$. The latter requirements (non-negativity and spectral radius of Φ) generalize the typical specification of the contractive constant used in conjunction with the usual norm on $\Re^n$. The existence of a fixed point is then given by the following theorem.

Contraction-Mapping [226] : Suppose that $\bar{\varphi} : D \subset \Re^n \longrightarrow \Re^n$ is a Φ-contraction on the closed set $D_o \subset D$, such that $\bar{\varphi}(D_o) \subset D_o$. Then, for any $\bar{u}(0) \in D_o$, the sequence $\bar{u}(t+1) = \bar{\varphi}[\bar{u}(t)]\ , t = 0,\ 1,\ \cdots$ converges to the only fixed point of $\bar{\varphi}$ in D_o, and the error estimate

$$\mid \bar{u}(t) - \bar{u}(\infty) \mid \leq (\mathbf{I} - \Phi)^{-1} \Phi \mid \bar{u}(t) - \bar{u}(t-1) \mid$$

for $t = 0, 1, \cdots$, holds.

Chazan and Mirankar [67] first applied these concepts to establish the convergence of asynchronous iterations. Their results were later generalized by Baudet [45].

Baudet's Theorem : If $\bar{\varphi} : \Re^n \longrightarrow \Re^n$ is a Φ-contraction mapping on a closed subset $D \subset \Re^n$ and if $\bar{\varphi}(D) \subset D$, then any asynchronous iteration corresponding to $\bar{\varphi}$ and starting with a vector $\bar{u}(0) \in D$, converges to a unique fixed point of $\bar{\varphi}$ in D.

We now derive necessary and sufficient conditions for convergence of the concurrent asynchronous neurodynamics characterized by Eqs. (2.2.1).

6.3. Asynchronous Neuro-Operator Derivation

Consider the temporal evolution of a fully connected, additive-type neurodynamical system, e.g., a Hopfield model defined by the following system of coupled differential equations:

$$\dot{u}_i + a_i u_i = \sum_j T_{ij} g_j(\gamma_j u_j) + I_i. \tag{6.3.1}$$

Here u_i represents the internal state (e.g., mean soma potential) of the $i-$th neuron, T_{ij} denotes the synaptic coupling from the $j-$th to the $i-th$ neuron and I_i is the external input bias. The sigmoidal function g_j modulates the neural response, γ_j denotes the transfer function gain for the $j-$th neuron and a_i represents the inverse of a characteristic time constant or the decay scaling term. Let $\varphi_i(\bar{u})$ denote the $i-$th component of the asynchronous operator introduced in (2.2). Using Euler's difference approximation to the above system of continuous-time differential equations, i.e., $\dot{u}_i = (u_i^{t+1} - u_i^t) \ / \ \Delta$, where Δ denotes the

discretization stepsize, the $i-$th component of the above defined Hopfield operator is given by

$$\varphi_i(\bar{u}) \;=\; u_i \;+\; \Delta\Big[\,-\,a_i u_i \;+\; \sum_j T_{ij}\; g_j(\gamma_j u_j) \;+\; I_i\,\Big] \tag{6.3.2}$$

Then for any two phase-space coordinates, $\bar{u}, \bar{v}$ in the domain of attraction

$$\begin{aligned}\varphi_i(\bar{u}) \;-\; \varphi_i(\bar{v}) \;&=\; (u_i \;-\; v_i)\;(\;1 \;-\; \Delta a_i\;) \\ &+\; \Delta\sum_j T_{ij}\;[\;g_j(\gamma_j u_j) \;-\; g_j(\gamma_j v_j)\;]\end{aligned} \tag{6.3.3}$$

On taking the vector norm, the above system yields,

$$\begin{aligned}|\,\varphi_i(\bar{u}) \;-\; \varphi_i(\bar{v})\,| \;\le\; |\;u_i \;-\; v_i\;|\cdot|\;1 \;-\; \Delta a_i\;| \;+ \\ \Delta\sum_j\;|\,T_{ij}\,|\cdot|\,g_j(\gamma_j u_j) - g_j(\gamma_j v_j)\,|\end{aligned} \tag{6.3.4}$$

We assume that for each neuron the response function, g_j: $\Re \;\rightarrow\; [-1,+1]$, is of class C^1, and that $|\;\; g_j' \;\;| \;\le\; 1$. This is obviously the case for the usually considered neural response functions , i.e., $g(\gamma u) \;=\; \tanh(\gamma u)$ or $g(\gamma u) \;=\; [1 \;+\; e^{-\gamma u}]^{-1}$. Then the Mean Value Theorem implies that there exists a $z \;\in\; \Re$ such that,

$$g_j(\gamma_j u_j) \;-\; g_j(\gamma_j v_j) \;=\; g_j'(z)\;\gamma_j\;(u_j \;-\; v_j)$$

Thus

$$|\,g_j(\gamma_j u_j) \;-\; g_j(\gamma_j v_j)\,| \;<\; |\,\gamma_j\,|\,|\,u_j \;-\; v_j\,| \tag{6.3.5}.$$

Regrouping all terms, we obtain

$$\begin{aligned}|\,\varphi_i(\bar{u}) - \varphi_i(\bar{v})\,| \;\le\; |\;1 \;-\; \Delta a_i\;|\cdot|\;u_i \;-\; v_i\;| \;+ \\ \Delta\sum_j\;|\;T_{ij}\;|\cdot|\,\gamma_j\,|\cdot|\,\bar{u}_j - \bar{v}_j\,|\end{aligned} \tag{6.3.6}$$

Let us now define a matrix Φ as follows:

$$\Phi_{ij} \;=\; |\,1 \;-\; \Delta a_i\,|\;\delta_{ij} \;+\; \Delta\cdot|\,\gamma_j\,|\cdot|\,T_{ij}\,| \tag{6.3.7}$$

We see that Φ is nonnegative; furthermore, since

$$|\;\varphi_i(\bar{u}) \;-\; \varphi_i(\bar{v})\;| \;\leq\; \sum_j \Phi_{ij}\;|\,u_j \;-\; v_j\,|$$

or equivalently

$$|\,\bar{\varphi}(\bar{u}) \;-\; \bar{\varphi}(\bar{v})\;| \;\leq\; \Phi\cdot|\;\bar{u} \;-\; \bar{v}\;| \tag{6.3.8}$$

we deduce that the neuro-operator $\bar{\varphi}$ is Lipschitzian with Lipschitz matrix Φ. From Baudet's theorem, for $\bar{\varphi}$ to converge to a fixed point in an appropriate basin of attraction, the spectral radius of Φ must be less than one. Now, using Bechenbach and Bellman's theorem [226] we can write

$$\min_{1\leq i\leq N}\left\{\frac{\sum_{j=1}^{N}\Phi_{ij}y_j}{y_i}\right\} < \rho(\Phi) < \max_{1\leq i\leq N}\left\{\frac{\sum_{j=1}^{N}\Phi_{ij}y_j}{y_i}\right\} \tag{6.3.9}$$

where $\bar{y}$ denotes any positive vector. In particular, we can choose all vector components equal. The contraction then translates into

$$\rho(\Phi) \;<\; 1 \quad\Longleftrightarrow\quad \begin{cases} \max\limits_i \sum\limits_j \Phi_{ij} \;<\; 1 \\ \Phi_{ij} \;>\; 0 \end{cases}. \tag{6.3.10}$$

for all i, j. This induces constrained interrelationships between the values of a_i, Δ, γ_j and T_{ij}, i.e.,

$$\max_i\left\{|\,1-\Delta a_i\,| \;+\Delta\sum_j\;|\,\gamma_j\,||\,T_{ij}\,|\right\} \;<\; 1 \tag{6.3.11}$$

and

$$|\;1-\Delta a_i\;|\,\delta_{ij} \;+\; \Delta\,|\,\gamma_j\,||\,T_{ij}\,| \;>\; 0 \tag{6.3.12}$$

To fix the ideas, without loss of generality, consider simplest the situation where all gain parameters are equal to γ. Then convergence under a concurrently asynchronous regime will be guaranteed if one chooses, e.g.,

$$[1] \quad a_i < \frac{1}{\Delta} \; ; \; \gamma < \frac{a_{i'}}{\sum_j |T_{i'j}|} \; ; \; |\gamma_j| = \gamma$$

$$[2] \quad a_i < \frac{1}{\Delta} \; ; \; \gamma < -\frac{a_{i'}}{\sum_j |T_{i'j}|} \; ; \; |\gamma_j| = -\gamma$$

$$[3] \quad \frac{2}{\Delta} > a_i > \frac{1}{\Delta} \; ; \; \gamma < \frac{2 - a_{i'}}{\Delta \sum_j |T_{i'j}|} \; ; \; |\gamma_j| = \gamma$$

$$[4] \quad \frac{2}{\Delta} > a_i > \frac{1}{\Delta} \; ; \; \gamma < -\frac{2 - a_{i'}}{\Delta \sum_j |T_{i'j}|} \; ; \; |\gamma_j| = -\gamma$$

Notice that the latter inequality invalidates the often made "high gain" approximation, at least for chaotic relaxation regimes. In the following sections, we discuss concurrent simulations on massively parallel neural networks.

6.4. Simulation Results

The concurrently asynchronous conditioning methodology developed in the preceding section, was implemented on a hypercube multiprocessor [31] for Hopfield's content associative memory model [134]. In our initial implementations, a fully-connected network with 128 neurons and six orthogonal patterns were used. Despite its conceptual simplicity, this model encompasses the paradigmatic essence of additive neuronal interactions, and has been extensively benchmarked in terms of correctness, efficacy, capacity and scalability [68,198,200-201]. The study precluded a critique on the model's functionality and focussed instead on the activation mode implications. In particular, our experimentation was aimed at the following objectives: (a) verify algorithmic correctness and asynchronous

convergence in a concurrent processing environment; (b) analyze the implications of ill-conditioned parameters, e.g., violation of conditions necessary for contraction, Eqs. (6.3.13), and; (c) benchmark computational efficacy with respect to the existing updating regimes.

Figure 6.4.1, juxtaposes the discretized temporal evolution of a noise corrupted input probe to the nearest stored memory under synchronous, "concurrently" asynchronous (specified using Eqs. (2.2.1), (6.3.2) and (6.3.13)), and sequentially asynchronous updating. With proper conditioning, all three systems converged to the nearest stored memory. In this example, synchronous update yields the fastest rate of convergence. Recall, however, that it often suffers from, both *horizontal* as well as *vertical* oscillations [68]. In addition, it imposes severe clock synchronization constraints, the implications of which were outlined in Section 6.1. Though the concurrently asynchronous update mode was simulated in a partially concurrent (e.g., 32 neurons / hypercube node) environment, convergence to the stored attractor indeed validates our methodology. Note that convergence in the latter case was achieved despite communication delays on the hypercube and globally inconsistent state information, i.e., neurons on different nodes operated assuming different states for the network. Also, as expected, the "sequentially" asynchronous mode led to the slowest convergence.

When the conditions, given by Eqs. (6.3.13) were violated, undesirable behavior abounded. The system failed to converge and oscillated instead, as depicted in Figure 6.4.2. This behavior raises a hitherto unaddressed fundamental issue regarding network dynamics, i.e., the exact nature of noise observed in concurrent neural networks. Is it due to horizontal oscillations, vertical oscillations [68] or is it a manifestation of chaos, or merely numerical instability induced by discretizing continuous dynamical systems ? These issues are briefly taken up in the next section.

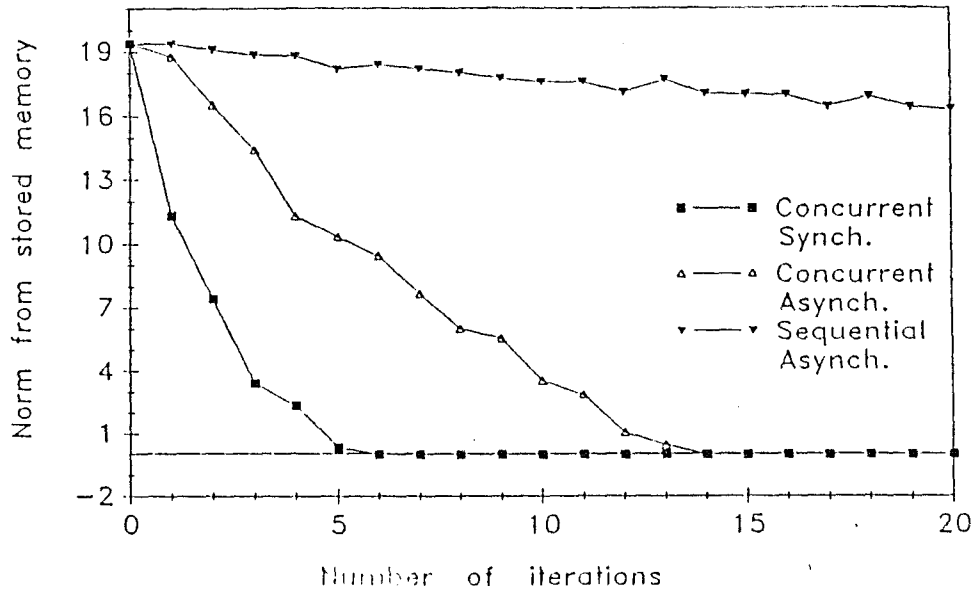


Figure 6.4.1. Concurrent simulation on NCUBE/Four to benchmark relaxation rate with synchronous, "sequentially" asynchronous and concurrently asynchronous updating regimes.

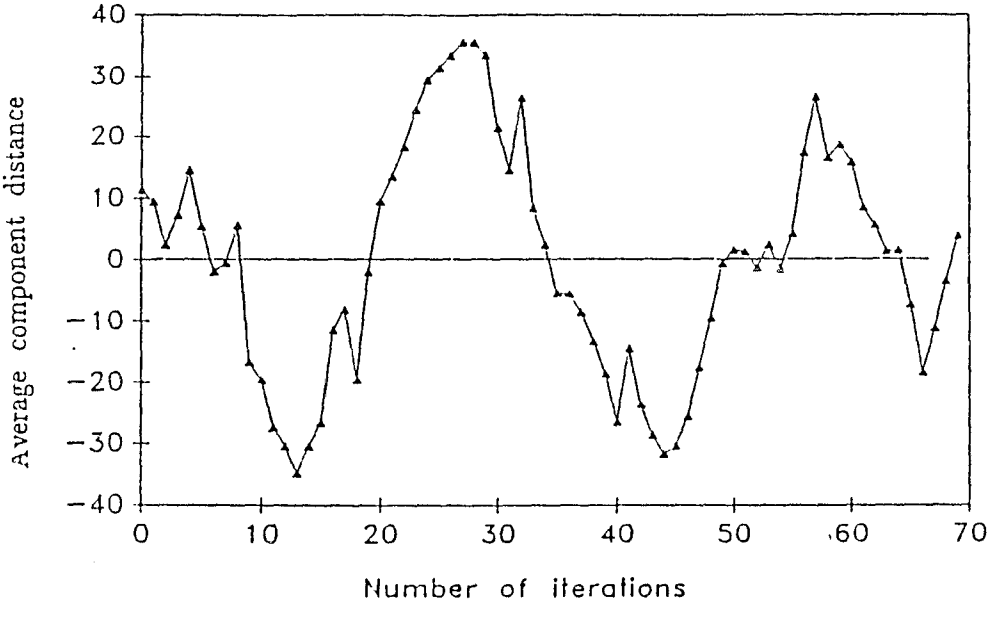


Figure 6.4.2. Chaotic oscillations with improper conditioning in concurrently asynchronous updating.

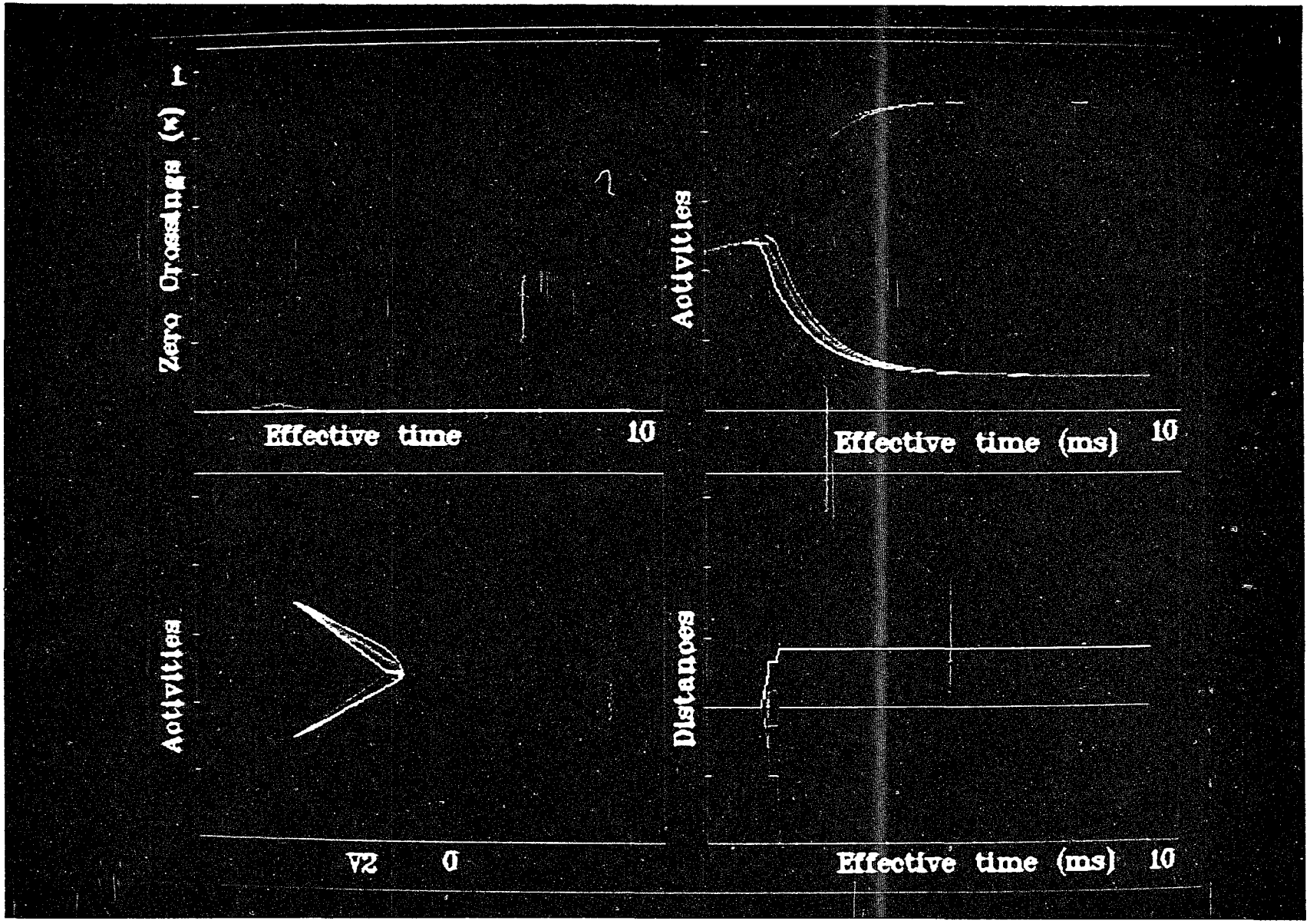


Figure 6.4.3. (a)-(d) Evolution of state variables under concurrently asynchronous updating.

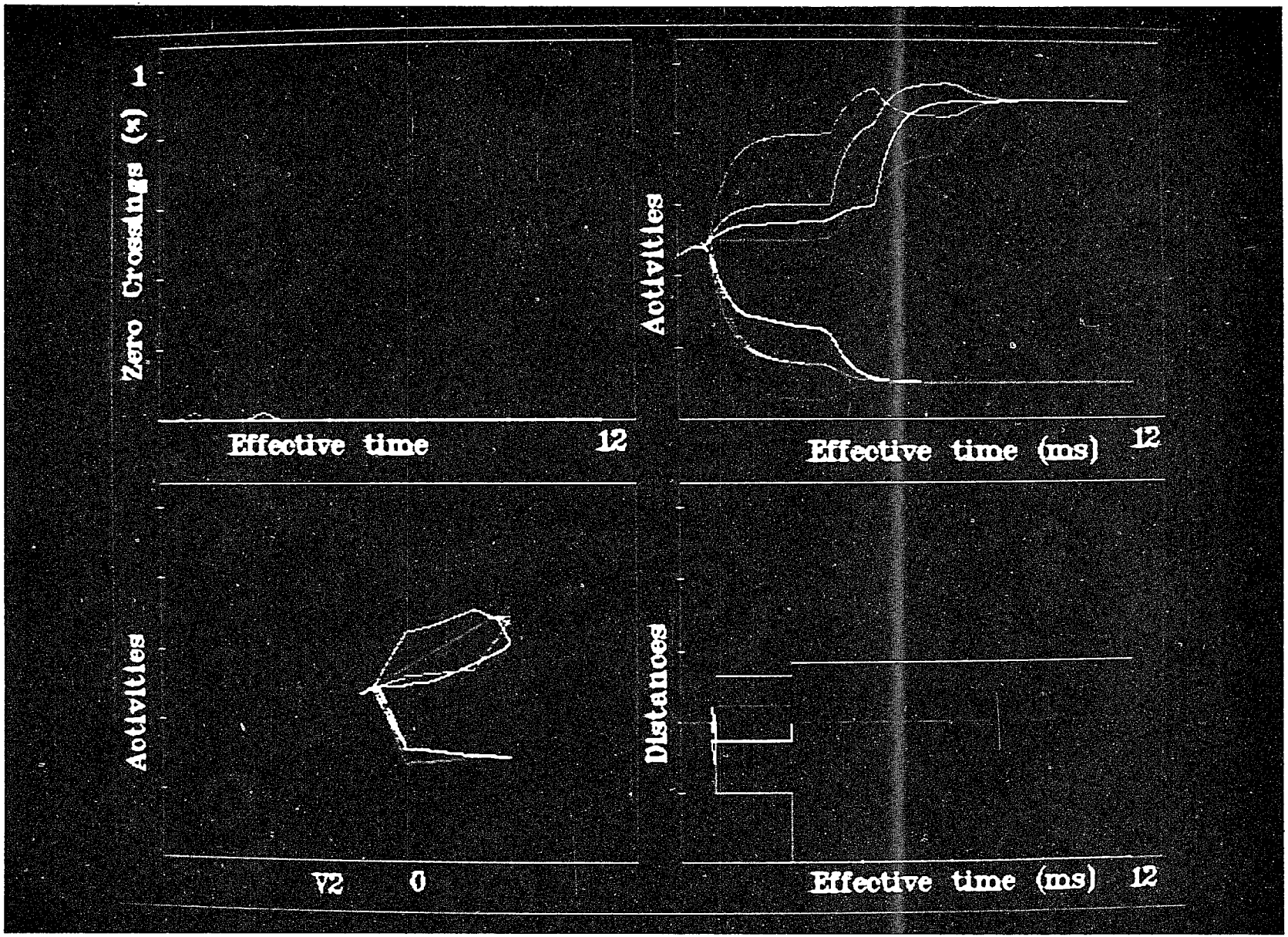


Figure 6.4.4 (a)-(d) Evolution of state variables under concurrently asynchronous updating with small time delays.

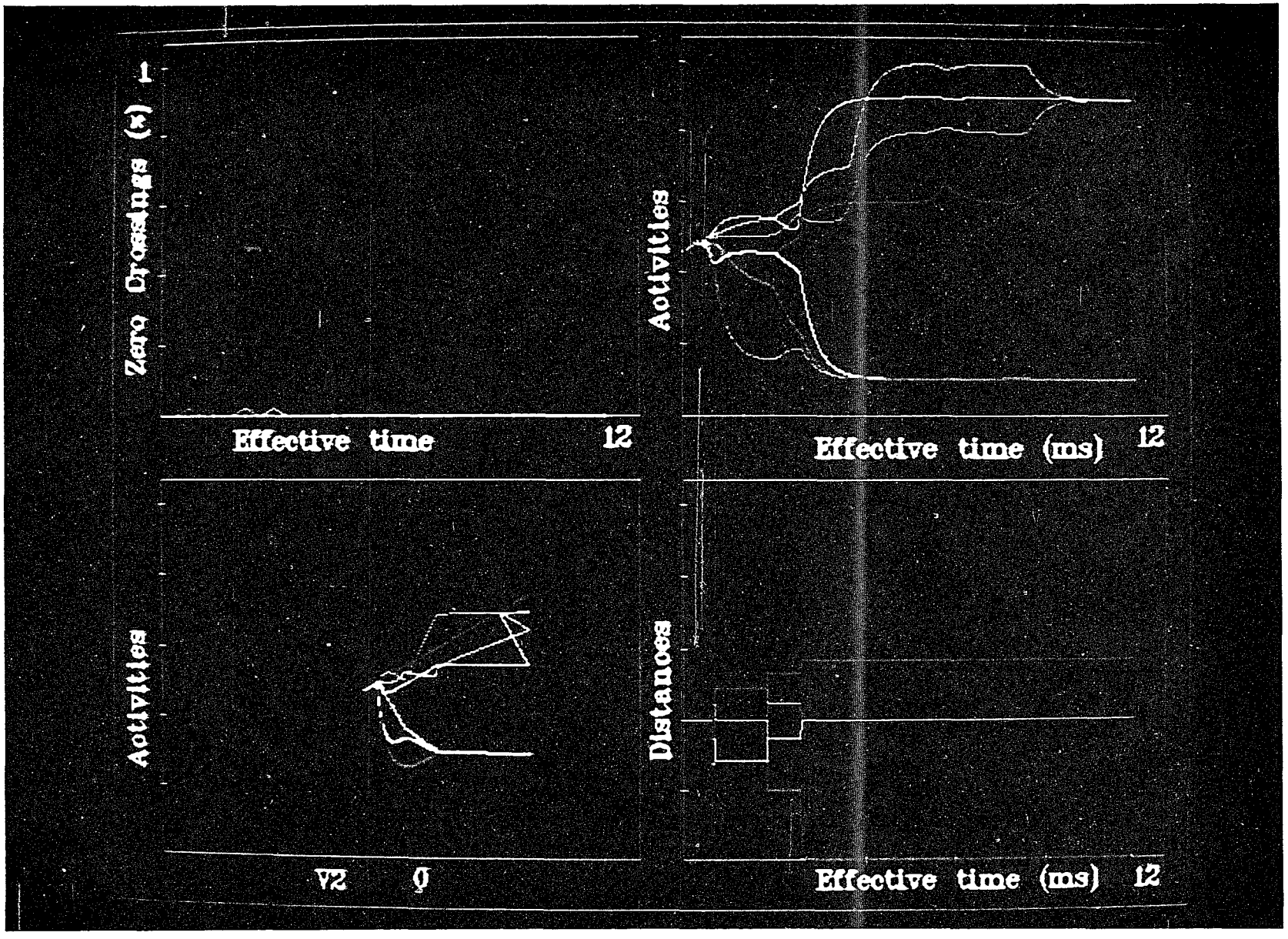
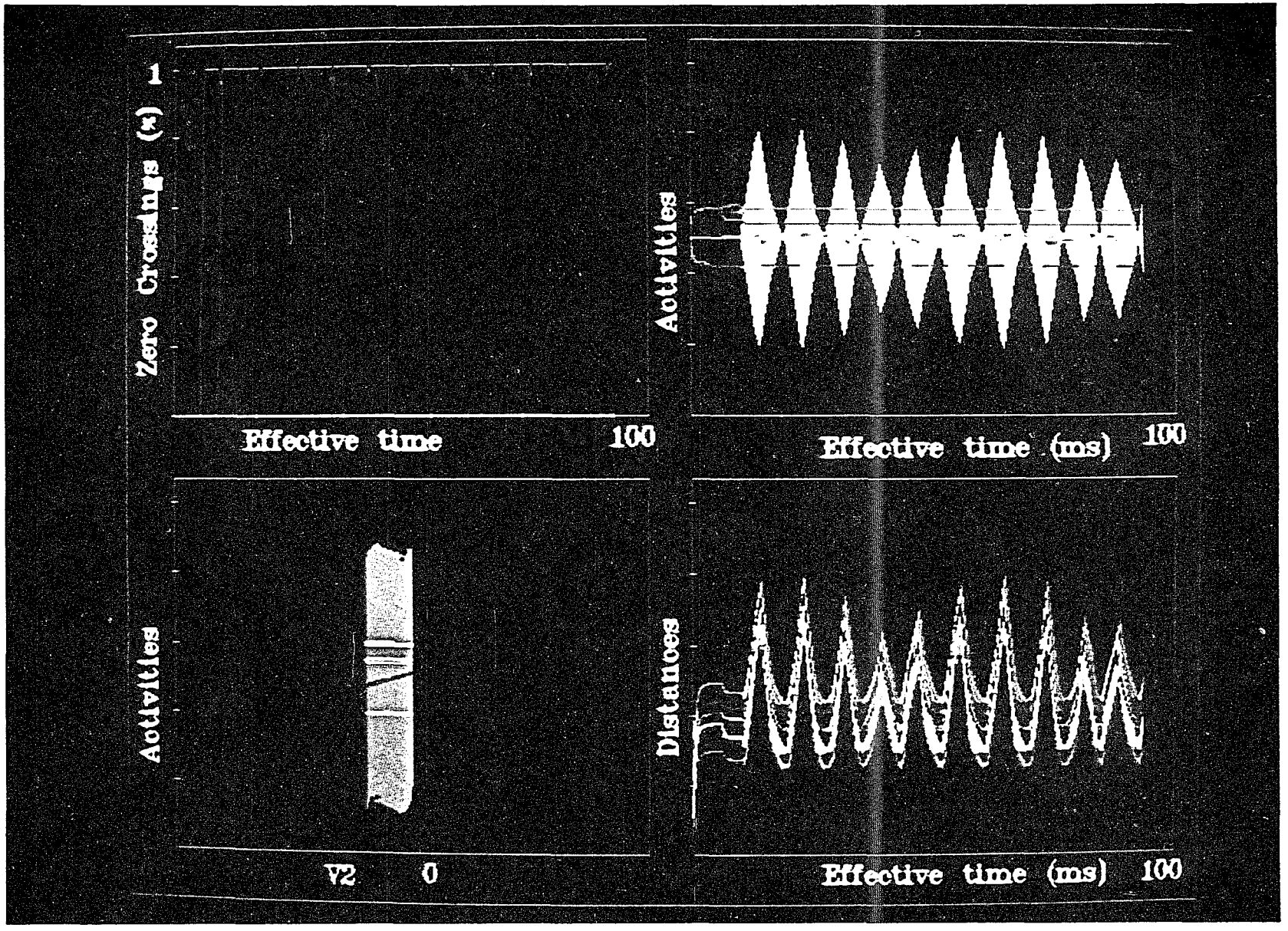


Figure 6.4.5 (a)-(d) Evolution of state variables under concurrently asynchronous updating with large time dalays.

Figure 6.4.6. (a)-(d) Evolution of state variables under concurrently asynchronous updating with one ill-conditioned neuron.



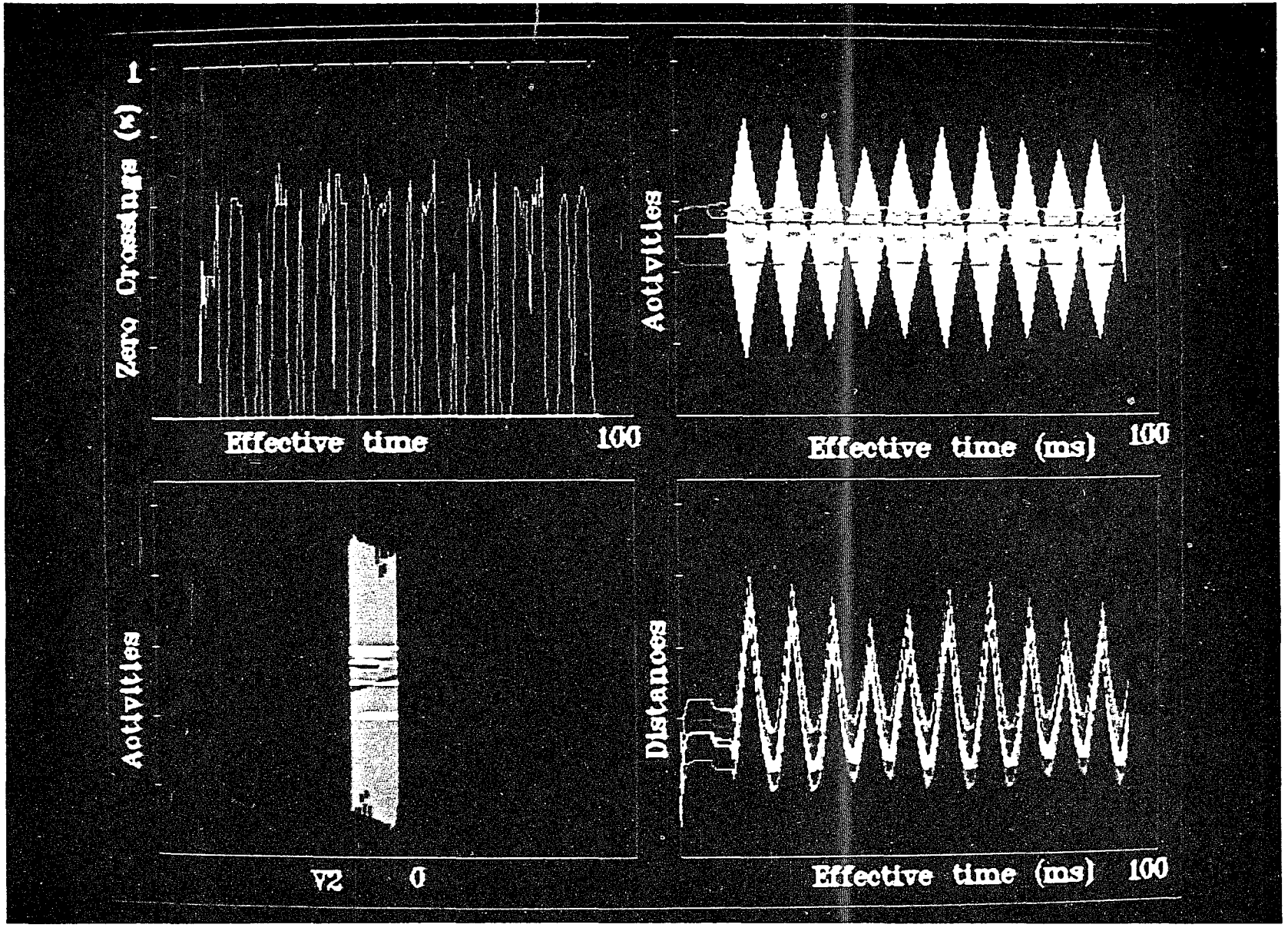
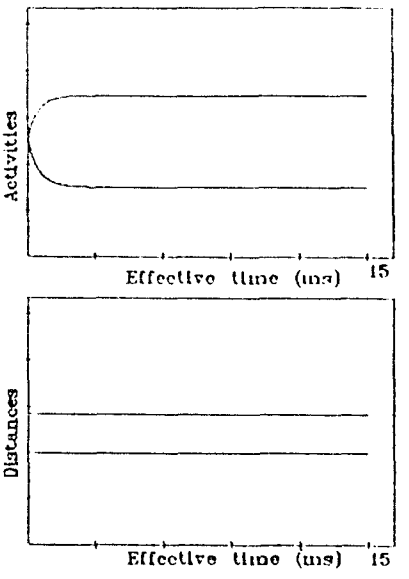
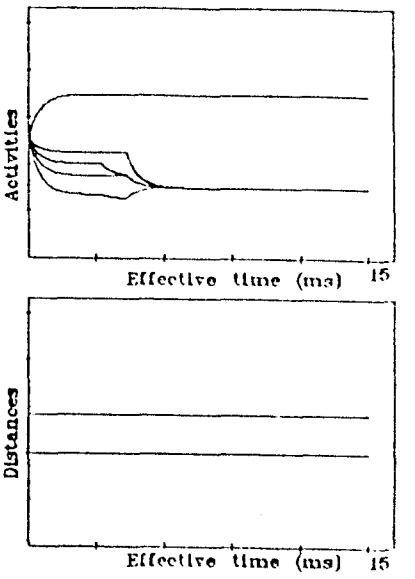


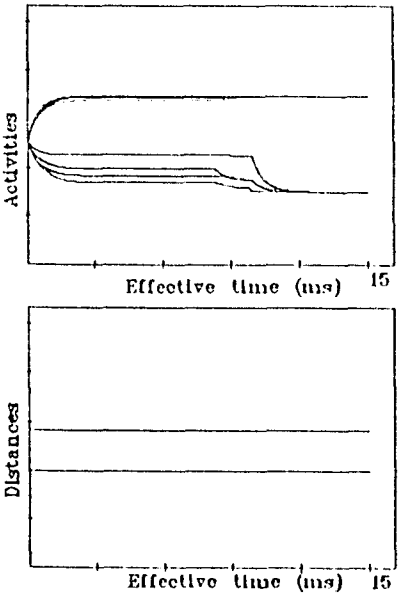
Figure 6.4.7. (a)-(d) Evolution of state variables under concurrently asynchronous updating with two ill-conditioned neurons



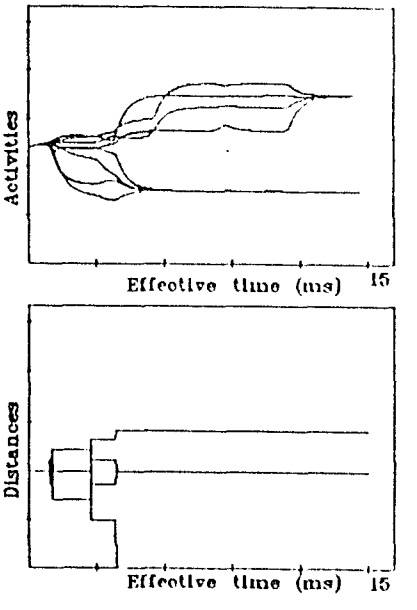
(a) $Delay = 0\Delta$



(b) $Delay \leq 100\Delta$



(c) $Delay \leq 1000\Delta$



(d) $Delay \leq 2000\Delta$

Figure 6.4.8. (a)-(d) Temporal evolution of neuronal activities in the presence of varying time delay.

In order to study the scalability and validity of our results to massively parallel network implementations, we simulated the associative memory network on a number of neural network models, ranging from 8 neurons to 10000 neurons (100 million synaptic interconnect). The latter simulations were performed on the JPL Cray XMP-18. It was found that when the neurodynamical parameters were derived using results from Section 6.3, the network was robust even in the presence of extremely large time delays (over 2000Δ). Also, we found that the network always converged to the nearest stored memory. In the absence of proper conditioning, the network exhibited sustained oscillations and convergence to spurious memories.

The color plates 6.4.3 to 6.4.7 display the neuronal and network behavior under different conditions of time delay and dynamical parameter conditioning, for the associative memory example discussed in the previous section. The network comprises of 8 neurons, $V1$, $V2, \cdots, V8$. Although, the formalism is universally robust (as our tests in the previous paragraph indicate), here we use a low-dimensional network to illustrate the qualitative behavior observed. The following indexing strategy has been adopted in the plates: upper left region, (a), indicates the frequency of sign changes, i.e., zero crossing, in the state of the neuron (e.g., a neuron changing state from +ve to -ve and vice versa) over an interval of 100 discretized time iterations; upper right region, (b), indicates the absolute magnitude of neuronal activity as it evolves in time; lower left region, (c), includes the plot of all neuronal activities against the activity of neuron $V2$; and lower right region, (d), includes the temporal evolution of Euclidean distance of the test probe from the nearest stored memory to which the network is expected to converge to. Plate 6.4.3, displays convergence under concurrently "synchronous" neurodynamical conditions (i.e., with no propagation time delay). Plates 6.4.4 and 6.4.5, illustrate convergence under concurrently asynchronous conditions with propagation delays upto 10Δ and 100Δ, respectively. Plates

6.4.6 and 6.4.7 provide a good example of aperiodic oscillations that occur when the stable dynamical bounds (derived using Eqs. 6.3.11 and Eqs. 6.3.12) are ignored. We violate the condition for one neuron, V1, in Figure 6.4.6 and for two neurons, V1 and V4 in Figure 6.4.7 These effects are resummarized in the composite Figure 6.4.8.

6.5. Elimination of "Emergent" Computational Chaos

In contradistinction to prevalent notions on instability in neural networks [68], that attribute oscillatory behavior mainly to the topology of the interconnection matrix, we hypothesize that it is primarily a *manifestation of "emergent computational chaos" induced by ill-conditioned parameters in the model.* This hypothesis is strengthened by the following observations. Simulations have shown that the same type of model may exhibit radically different dynamical behavior with slightly different parameters. For example, small perturbations in time scales, delay distribution, transfer gain etc., may lead the system to oscillate back-and-forth from one basin of attraction to another. An analysis of the Lyapunov spectrum, computed using the time series, obtained from a mapping which follows the evolution of a mapping which follows the evolution of the average component difference from a stored memory provided a validation to the chaotic dynamics in such ill-conditioned models.

Lyapunov exponents [1,299] essentially provide a dynamical diagnostic for measuring the exponential rates of convergence or divergence of phase trajectories. For a continuous dynamical system in n-dimensional phase space, the $i-$th dimensional Lyapunov exponent is defined as,

$$\lambda_i = \lim_{t\to\infty} \frac{1}{t} \log_2 \frac{p_i(t)}{p_i(0)} \tag{6.5.1}$$

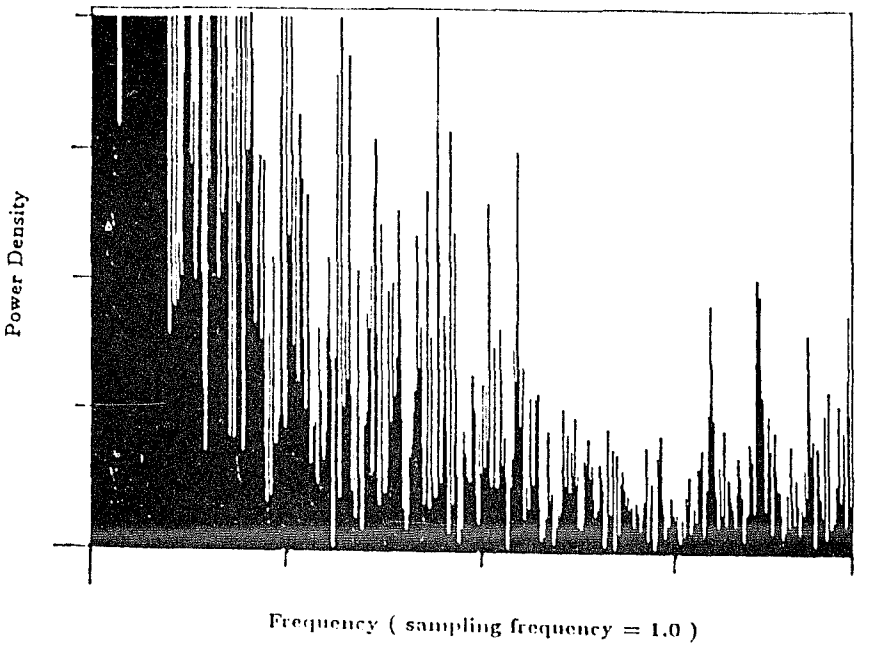


Figure 6.5.1. Power spectrum for time series generated during chaotic oscillations.

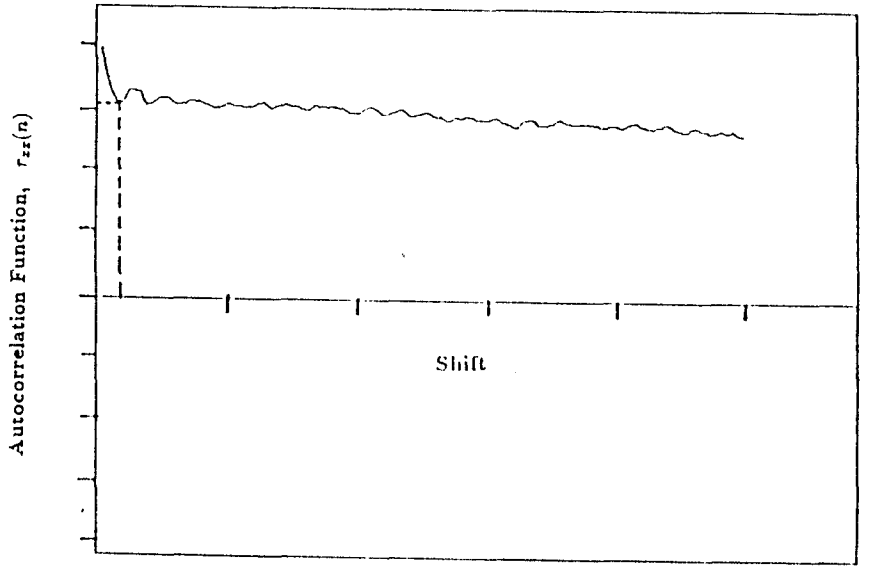


Figure 6.5.2. Autocorrelation function for the time series obtained during chaotic oscillations.

where $p_i(t)$ denotes the length of contracting / expanding principal ellipsoid axis, corresponding to the temporal deformation of the phase space. Any dynamical system characterized by a negative sum of Lyapunov exponents, but containing one or more positive terms is said to be chaotic, with the magnitude of such exponents reflecting the time scale on which the system dynamics becomes unpredictable [1,299]. Our simulations for an ill-conditioned neural model ($\Delta = 0.002$, $a = 1000$, $\gamma = 10000$ and $\sum \mid T_{ij} \mid = 84$) led to a value of $\simeq +1.49$ for the largest exponent. Since λ_1 is positive, we conclude the system to be if not chaotic, at least exponentially stochastic (since we did not compute the sum of all exponents). Also, when the largest Lyapunov exponent was determined for

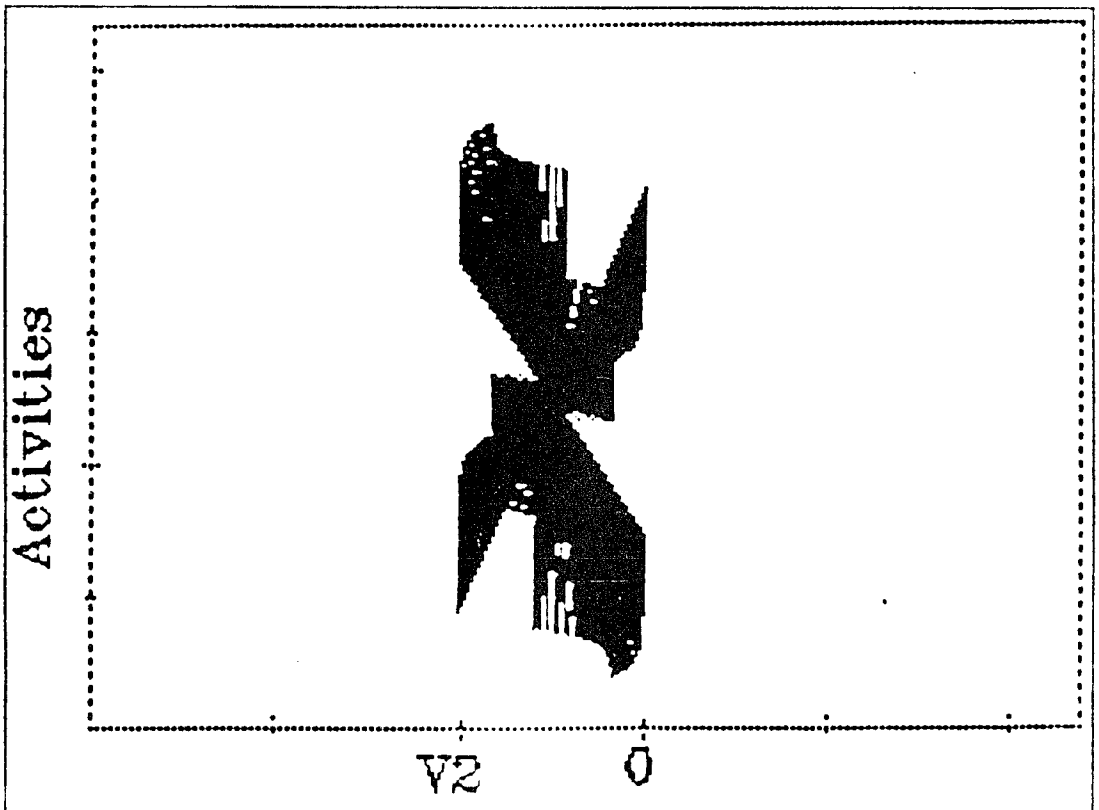


Figure 6.5.3. Strange Attractor for the chaotic time series (e.g., zero crossing), when the neurodynamical constants for neurons were ill-conditioned.

a contracting concurrently asynchronous network ($\Delta = 0.002$, $a = 100$, $\gamma = 1$), the value $\lambda_1 = -1.09$, was found, thereby proving that our conditioning methodology eliminates "emergent" chaos in concurrent neuromorphic models.

Dissipative nonlinear chaotic systems which evolve on a strange attractor can generate a broadband power spectrum, while the strange attractor "lives" in a phase space of finite (and often small) dimension. Figure 6.5.1(a) illustrates the power spectrum for the time series generated during asynchronous convergence to the stored memory, while Figure 6.5.1(b) displays the power spectrum during chaotic oscillations. The autocorrelation function,

$$A(K) = X_N \int_0^T x(k+t)\, x(t)\, dt \qquad , k \geq 1 \qquad (6.5.2)$$

where X_N is the normalization constant, chosen such that $A(0) = 1$. Figure 6.5.2. shows the autocorrelation function during chaotic oscillations. The strange attractor for the chaotic time series is shown in Figure 6.5.3.

6.6. Summary

This chapter presents a radically different insight into the neurodynamical implications of "sequentially" asynchronous and synchronous neuronal algorithms. Despite significant advances in concurrent hardware technology, full realization of the potential advantages of neural processing in solving real-life problems has been severely limited due to previous assumptions for asynchronicity. Electronic embodiments based on current mathematical frameworks lead to biological inconsistencies and require substantially complex circuitry. In a similar vein, such frameworks also limit the network scalability, stability and throughput in discrete-time simulations. It was hypothesized that, contrary to existing notions that attribute dynamical instability in the current models to the topology of the interconnection matrix, we ascribe it to "emergent chaos". Lyapunov

exponents were computed to prove that improperly conditioned neurodynamical equations of motion do indeed exhibit chaotic relaxation behavior.

We exploited this insight to provide a strategy for systematically reconditioning the existing mathematical framework for additive networks, such that their VLSI, optical and opto-electronic embodiments are truly asynchronous, and thereby, eliminate the network instability ascribed to "emergent" chaos. We derived a neuro-operator that enables chaotic relaxations to achieve concurrently asynchronous updating. Necessary and sufficient conditions guaranteeing concurrently asynchronous convergence were defined in terms of Φ-contraction mapping. Lyapunov exponents were calculated for our proposed neuro-operator to ensure that the reconditioned system is devoid of chaotic behavior. Future directions include extension of our theory to shunting-type [8] neural networks. We also intend to theoretically analyze the implications of chaotic relaxation on network parameters, such as synaptic efficacies, transfer characteristics, network architecture and capacity.

Chapter Seven

Neural Learning Formalisms for Perceptual Manipulation Systems

7.1. Introduction

In this chapter we analyze the relevance of our theoretical results as an enabling technology, in the context of their applicability to difficult real-life problems in robotics. Leveraging the powerful repertoire of neural learning formalisms developed in the preceding chapters, we introduce a modular architectural framework for synthesizing a new generation of robot control systems that could potentially provide a previously unattainable level of robustness during task execution. To put our arguments in perspective, we choose a specific manifestation of this generalized architecture, that relates to synergistic man-machine systems [117] (i.e, for space applications: namely, Robot-Assisted Extravehicular Activity). A key requirement of such systems is a high degree of operational safety, robustness, execution speed and stable adaptation to active unstructured environments (humans) in the workspace. We conduct an analysis of the functional and architectural requirements to show such robotic systems to be well beyond the state-of-the-art, "model- directed" robotic methodologies. The latter, best suited for problems where reliable and accurate models can be determined prior to control design, break down in this context. Consequently, we conceptualize the architecture in a "perception-directed" framework. A technical critique of the proposed architecture in comparision to the existing robot architectures is included. Additional details may be found in Venkataraman and Gulati [281].

We begin with a brief review of the state-of-the-art in symbiotic (cooperative) man-machine systems for space operations, along with some insight into the applications context. In the past, man-machine integration has typically been implemented in the form of bilateral teleoperation, shared/traded control or supervised autonomy [258]. The primary emphasis has been on the enhancement of task execution speed and robustness in the presence of uncertainties through an effective integration of human perception, planning and control skills into the robot command structures (human may be considered as a complex time dependent, nonlinear command shaping/filtering function). Synergistic man-machine task execution for space applications [117], however, has not received adequate attention, with the exception of Akin [5] who has proposed teleoperated robotic assistants for astronauts, with limited capability and the Extravehicular Activity Retriever [EVAR] robotic system being developed at NASA/JSC (Erickson et al. [81]). Notwithstanding its methodological and computational simplicity, Akin's approach [5] is limited by its ability to render the teleoperator with sufficient task-observability vis-a-vis the state-of- the-art in robotic telepresence. Also, since the robot simultaneously interacts with two humans (operator and the astronaut), both of whom potentially represent active unstructured dynamical environments (i.e., with time-varying interaction impedance) its range of stable operations [158] is rather limited. The EVAR [81] system, on the other hand, is targeted for autonomous robotic retrieval of "free floating astronauts", tools, debris, etc. in space. However, its architectural design precludes capabilites such as adaptation to dynamically varying kinesthetic interactions with the astronauts, adaptive sensory motor-control, time-boundedness responses and true shared task execution (i.e., robot and astronaut jointly accomplish a task). Since the latter are essential for developing space-based systems, the necessity to revisit the problem groundup cannot be overstated.

Of fundamental concern in *realistic* synergistic man-machine systems for

space applications is human safety (e.g., avert astronaut tumble during Orbital Replacement Unit (ORU) handoff, avoid space suit [or Extravehicular Mobility Unit (EMU)] damage during astronaut rescue). This implies that task executions in space would be subjected to robustness and response requirements far more stringent than ground-based robotic activities, e.g., industrial assembly. Further, overall system stability (e.g., dampen vibrations arising from energy release of preloaded items during ORU deinsertion) assume significantly complex dimensions in the presence of hybrid elements: man and machine and constraints on power. For instance, as the robot stability depends upon the muscular impedance of the human it is interacting with, the robot must dynamically adapt itself to the astronaut's kinesthetic characteristics during contact operations, e.g., tool handoff, shared transportion of objects, etc. In addition, the execution performance of robotic elements is driven by two conflicting motivations: execute tasks successfully (e.g., precise trajectory tracking for a motion task), while accommodating the changes in astronauts kinesthetic properties and posture (dynamical obstacle avoidance). In this context the following requirements naturally emerge

[1] *Operational safety :* the robotic system representation must exhibit methodological completeness in addition to algorithmic performance and robustness, i.e., an internal representation that is "formally" sufficient for capturing possible eventualities during control execution. For example, an X,Y,Z, Roll, Pitch, Yaw representation in [51] is methodologically complete for representing the kinematics of rigid bodies. Also, a hybrid force-position coordinate system (Raibert and Craig in [244] would be complete for representing frictionless point contacts between two rigid bodies. On the other hand, same representation would become insufficient in the presence of friction. In a similar vein, we require a representation that can capture the complete physics of interaction with the environment, that occurs during prototypical

mechanical task execution by robots. The architecture needs to be based on on a vocabulary of formal task primitives, necessary and sufficient for targeted functionality. Many of the existing architectures take an engineering approach to matching system with requirements, and may not be able to guarantee completeness, e.g., subsumption architecture [54] contains a collection of intuitively engineered finite state machines (FSM), and promote the "add an FSM as and when needed philosophy". It is only with the availability of such a representation, can the controllers be designed for autonomous task execution involving arbitrary forces and motions.

[2] *Stable adaptation to dynamically changing environments* is another key requirement. An essential component of latter is overall system stability, especially during shared task executions when the robotic system is in contact with the human/astronaut. Stability issues assume significantly complex proportions in the presence of a human, since the latter represents a dynamic environment which is both active (i.e., has its own actuators), and unstructured (i.e., dynamic models that characterize the human's *kinesthetic behavior*, i.e., time-varying muscular elasticity may not be known). For adaptation, the robot must dynamically avoid a human, whose motions may be unpredictable (i.e., geometric workspace sharing), as well as stabilize contact interactions with a human whose kinesthetic properties may be dynamically changing (i.e., kinesthetic workspace sharing).

[3] *Execution Performance*, in general, would have to include a set of metrics that can monitor tracking behavior (e.g., errors in joint angles during a motion trajectory following task in the robot's configuration space), and another to guarantee system stability in the presence of unmodeled system dynamics, or unforeseen events (e.g., large changes in structural frequencies due to mechanical contacts). We denote the former a *precision metric*

that guarantees expectation-based tracking performance and the latter, a *compliance metric* that accommodates stable interaction behavior with unmodeled environments. In general, these metrics compete with each other ([281,282] provides a example of this for linear compliance and motion trajectory tracking during surface tracing experiments). For example, accurate motion tracking performance can be realized by making the system stiff (high gains), e.g., during placement of gripper around tool. On the other hand, stability in the presence of unprecedented contacts can be ensured only by making the system lax (low gain), e.g., during tool acquisition. Additional details on precise-compliant controllers are available in [115,281].

In order to motivate our targeted application domain, we briefly sketch the practical problems impeding deployment of robots for space operations. Current in-space servicing options (for details see [114] and references therein) include, Astronaut Extravehicular Activity (EVA), teleoperation off the shuttle [Intravehicular Activity (IVA)] or the space station, Ground Controlled Remote Manipulator (GCRM), or Flight Telerobotic Server (FTS) [144]. In a recent study [114], we have analyzed the technological and feasibility limitations of these options in the context of near-term space operations. Pure teleoperation, for instance is severely limited by operator's kinesthetic precision, task perception and bounds on operator fatigue, in additional to design limitation in space hardware, e.g., backlash, stiction, structural instabilities, etc. [51]. On the other hand, the GCRM approach will cater only to a small portion of in-situ serviceable tasks, a majority of which have been designed to be human serviceable [114]. So regardless of any particular strategy for exploiting teleoperation technologies using the FTS, Remote Manipulator System (RMS), Special Purpose Dexterous Manipulator (SPDM), a sizeable fraction of low earth orbit servicing will be performed by astronauts in EVA or IVA. However, the productivity of astronauts will be limited by the rigors of working in space. Performance will be degraded even

further during EVA due to pre- and post- breathing requirements, a lack of dexterity and payload maneuverability imposed by the space suit, and elaborate safety requirements. It can be summarized, therefore, that any improvement to the productivity of astronauts in space would prove extremely valuable. In light of this, we propose an application of telerobotics to develop astronaut assistants and thereby provide the necessary enhancement to EVA- Astronauts.

Motivated by the above concerns, we conceived the Robot-Assisted Extravehicular Activity (RAEVA) paradigm [114-117,280-282], aimed at the development of robotic systems that can assist astronauts during EVA, i.e., in-space servicing, assembly and inspection. Within this framework, an EVA-Astronaut would perform the actual servicing, inspection and assembly operations, while one or more voice-activated robots would fetch and return servicing tools and Orbital Replacement Units (ORU) from and to their respective placeholders, help the astronaut transport large ORUs and truss members, and share task execution, e.g., supporting thermal blankets during ORU changeout, truss assembly, cryogenic fluid replenishment. Such a system could potentially exploit the benefits of an unexplored class of man-machine systems and improve effective use of crew-EVA time through task sharing (e.g., an astronaut would install an ORU fetched by a robot), In addition, it could amplify crew-EVA capability through task-space sharing (e.g. assembly of long truss members in space) and increase astronaut productively, thereby reducing operational costs and fatigue.

The technological elements underlying RAEVA have several ramifications for space deployment, e.g., Robot-goelogist, Physiotherapy Assistant, Planetary Prospector, Sample Analysis and Preservation Systems for planetary exploration, etc [114]. In the remainder of this chapter, we derive a framework for designing and sequencing versatile high performance controllers that could potentially provide above envisaged capabilites for robotic assistants. However, for ensuring

enhanced readability, without any loss of generality, we limit our arguments and illustrative examples to the RAEVA domain.

7.2. Perceptual Manipulation Systems

Typically, conventional model centered robotic systems , such as HANDEY [195] would implement the above desired functionality as follows: At the core of such systems is a high fidelity database comprising of geometric, kinematic and dynamic models for the robot and environment. These explicit descriptions are manipulated by goals to be achieved, effects of elementary operations available to the system and sensory input to predictively derive planning and control actions. Thereafter, goal conditions for the termination of each task primitive would be provided by a supervisory agent. Paths leading to the goal state will be generated, decomposed into sequence of discrete segments, and fed to a controller to compute the required actuation torques. Task-related constraints, e.g., grasp stability, grasp forces etc., would be incorporated to lend structure to this overall problem solving process. The robot would thus alternate between periods of planning a motion and periods of executing it, while sensing is done between the completion of motion and the beginning of planning.

However, model-directed control and planning approaches (50,51,143,144, 195] rely upon the existence of reliable world-models that can capture the kinematic and dynamical properties of the robot and its environment. Therefore, they are of limited applicability in active environments, whose dynamical properties either vary in an unpredictable manner, or are too complex to be utilized on-line (e.g., models that capture the dynamically varying geometric (posture) and muscular (kinesthetic) properties of the astronaut with high fidelity are either extremely difficult to derive and/or computationally intractable to be utilized on-line). But, in the context of performance-critical robotic systems such as

RAEVA, it is hard to model the kinesthetic (elasticity of muscular interactions) properties of the every object (including human) during contact interactions with the robot. This is a nontrivial consideration, since any simplifying assumptions, such as using linear second order impedance to model kinesthetic coupling, could lead to instabilities during shared task executions. So, reliance on models alone, could lead to reduced operational robustness and loss of range of applicability. Also, since in space we expect a large range of variations in the interactions between the robot and its environment, simple PID control laws that provide local stability may not suffice. And it is very difficult to design optimal control laws that attempt to minimize tracking errors [56], since the exact dynamic models for objects such as flexible thermal blankets, do not exist. Contact stability would require matching robot-environment impedances, so that force controllers may be designed to provide global contact stability with passive environments (lacking actuation capability). Note that simple compliance, stiffness methods that guarantee only local contact stability may not suffice [282].

Thus, the robot control architecture must be able to execute perception-directed formal task primitives, indicating recourse to indicates the need for sensor-directed planning and control. Architectures such as NASREM [8], HANDEY [195], HICS [253], and Meystel's Intelligent Control (IC) Architecture [210] have a tendency to be model-driven, and do not allow real-time integration of information from multiple sensors in real-time. In fact, very few semi-autonomous robotic systems built to date, can provide operational speed, performance and robustness matching the requirements outlined in the previous section.

Further, control of contact interactions will require real-time matching of the contact impedance between the robot and its environment, and the accommodation of unmodelled obstacles in the robot's workspace. The above considerations,

when juxtaposed with the achievement of specified goals assume complex proportions since they represent competing performance indices. For example, the control compliance always degrades the precision with which command trajectories can be tracked. Also, adaptation to dynamic environments requires integration of sensory information into adaptive control algorithms. By definition, this would preclude model-driven approaches, e.g., Saridis' Hierarchial Intelligent Control System [253], Lozano Perez's HANDEY [195]. Since passive [158], active compliance [204] and impedance control methodologies [50,129] have only demonstrated interaction stability with fully constrained, passive environments, they may prove inadequate for contact interactions with an unstructured active environment (human). At present, conventional adaptive control formalisms cannot guarantee the operational speeds necessary to ensure human safety, since the paradigmatic requirement that all sensory information be transformed into the robot state space at control frequencies cannot be matched with the current information processing technology [51].

Further, for operations such as rescuing drifting tools, astronauts, the robotic assistant may require accommodation to environments that change at rates comparable to the sensing and actuation rate of a robot system. This, combined with the fact that there is no intuitive frame of reference in space, would make the problem of associating inertial and orbiting frames with robot difficult. This is made even harder by the fact traditional approaches require calibration of visual geometry to arm movement, relation between link lengths and joint angles to end arm position, and the calibration of actuator signals to joint angles. Reactive planning and control [85,249], and behavior-based planning and control (Brooks in [54],[72]), are some of the recent approaches proposed for imparting speed and robustness to the robot's execution performance. However, at present they are predominantly heuristic-based and not amenable to formal stability analysis.

In addition, architectural segmentations based upon symbolic-numeric information processing (e.g., SOAR), local-global information / interaction contexts (e.g., HANDEY [195]) and process rates (e.g., IC [210]) cannot be applied to our problem either, for the following reasons:

(a) In general, every action comprises of both symbolic and numeric information processing components, i.e., symbolic processing units must have all "relevant" numeric context available, and vice versa. Therefore, any architectural organization based on symbolic/numeric segmentation will fail, since, it is not possible to propagate all numeric context to and fro without the availability of infinite communication bandwidth (impossible to satisfy in practice).

(b) Since robot-assistant and astronaut interactions cannot be adequately modeled, a priori decisions about local/global "context" - "action" funnelling [194] may degrade system performance. For example, typically free motions are planned globally while fine motions (e.g., contact, stability) are analyzed and planned locally, under the implicit assumption free (fine) motion has global (local) effects only. This could easily be violated when excessive force during tool exchange (fine motion) cause astronaut tumble (causing global domain instability), the correction for which requires global analysis.

(c) Any segmentation based upon nominal process rates (e.g., Meystel's IC [210]) can guarantee stability if the system operates in a quasistatic framework; e.g., context switches, (i.e., no contact to contact or vice versa) induce small changes in system structural frequencies, or the segmentations have been arrived at to account for the highest expected structural frequencies. However, the operational speeds required for RAEVA, combined with the varying kinesthetic properties of the human may make process rate estima-

tions difficult. Further, since construction of adequate world model (dynamics) may be extremely complex, it would not be possible to estimate a priori, contact structural frequencies. Consequently proces rate segmentations would have limited use for applications akin to RAEVA.

Any workable solution must be founded upon stable, robust and real-time planning and adaptive control algorithms, driven by perceptual mechanisms that can provide sensory information at (planning and control) loop frequencies. So, planning and control actions would need to be structured in the form of behaviors, which implies rapid generation of response actions directly from information in the sensor space, i.e., perception-directed formal task behaviors are required. As a result, conventional architectural approaches based upon a priori decoupling of planning and control activities (e.g., NASREM [8] considers execution to consist of plan $- >$ send control command $- >$ generate action $- >$ report status sequences) may not deliver the operational speed required from RAEVA. Therefore, the architecture must be able to execute perception-directed formal task behavior to satisfy competing execution metrics. An approach that has shown considerable promise in alleviating the above bottlenecks is to structure planning and control functions as sensorimotor behaviors, i.e., nonlinear transfer functions that directly map a stream of sensory inputs to control outputs. The speedup result from parallel computation, the ability to generate control actions directly from primitive perceptual representations (e.g., edges, range data, tactile maps) and a representation optimized for fusing of information from multiple sensors directly into the control loop. In addition, such systems [173,174] exhibit self-calibration, in the sense that the robot is typically controlled using the consistency between the signals used to move it and the signals used to sense the movement. There are no objective coordinates. To date, Grossberg and Kuperstein ([99] and references therein), have demonstrated a system that can learn to map a target that is visually detected by a mobile camera into a target position

map that is invariant with respect to body-centered coordinates, and that can repair itself if its operating parameters undergo modest damage. It has been used for controlling variable-speed trajectories of a robotic arm without programming the whole trajectory, and for rapidly updating trajectory commands in real-time, to reach objects arbitrarily positioned in space. Uno [277] proposed a speedup, by using a learning algorithm that calculates the actuation signals directly from a goal of the movement represented as the minimization of integral of square of torque change, thereby eliminating a need for transforming trajectory coordinates to the body coordinates. Kawato et al. [156,157] have speeded up this process even further by using a Newton-like iterative method in functional spaces.

On the other hand, recent advances in "behavioral robotics" (Brooks in [51,54,55]) provide exciting results on the potential of sensor-based control in implementing complex reflexes and task-behaviors, *sans* world models. In addition, it has been shown that providing planning and control actions with a task context allows compact specification of goal-oriented constraints, structuring of control algorithms and monitoring execution performance.

In our development of a new architecutural framework, we represent the actions within task structures that explicitly consider dynamic interactions between the robot and the human [279], and develop control algorithms that implement task units in the form of stable sensorimotor behaviors [57,99]. We have developed the *perceptual manipulation architecture* [115-117,280] for the purpose of executing formal task behaviors, that can be monitored by a set of performance metrics. Therefore, it naturally lends itself for developments of RAEVA like systems. In contradistinction to conventional model-based approaches that plan and execute robot actions relying heavily upon apriori world models, the RAEVA architecture has been conceptualized within a "Perceptual Robotics" (PR) formalism [115-116]. To ensure stable and robust interactions between the

machine and a human, robot actions are generated through a "perception- directed reverse engineering of formal task behaviors to satisfy competing execution metrics". In this section, we describe the actual RAEVA architecture. In the following sections, we will compare our approach with the state-of-the-art, and provide a justification for the line of thought followed herein.

We begin by briefly describing the system features for RAEVA. Fig. 7.2.1.1 describes the three basic functional units: a voice-activated Command I/O Unit that recognizes high level supervisory control commands from EVA / IVA crew members; a Static Task Analysis Unit that analyzes Crew-EVA primitives to determine the corresponding robot task primitives, configures the perceptual and control organization and synthesizes relevant execution metrics; and, a Task Perception and Execution Unit that implements a recursive hierarchy comprising of control execution units.

Fig. 7.2.1.2 summarizes the complete system architecture. The basic architectural building block of its Execution Unit is a kernel of behavioral primitives, constructed upon a vocabulary of formal Task Primitives (e.g., Free Motion (FM), Guarded Motion (GM), Fine Motion (FIM), Free Force (FFA), Guarded Force (GFA) and Fine Force Application (FIFA) [279]). The latter have been derived to explicitly represent the complete physics of mechanical interactions, given by the power flow across the contact interface (formally defined in [279] and references therein (thermal, magnetic and other affects are neglected). The architectural decomposition has been derived to allow an explicit representation of dynamical interactions between the robot and its environment at locations where maximal subsystem control under varying "system sensitivities" (system stability versus subsystem control errors) is desired. For example, Figure 7.2.1.3 shows the kernel of task primitives for a dual arm RAEVA configuration. As shown, the kernel

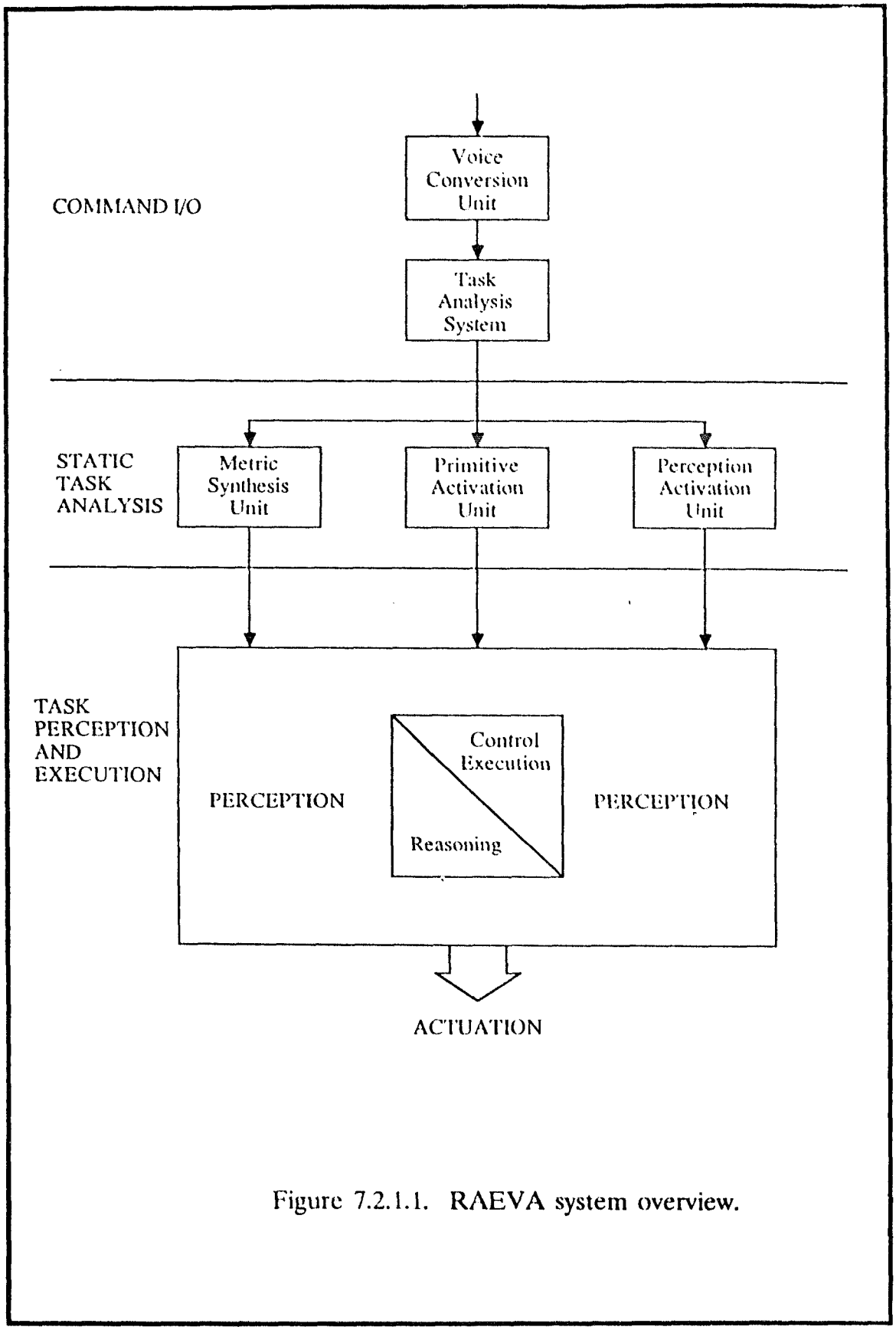


Figure 7.2.1.1. RAEVA system overview.

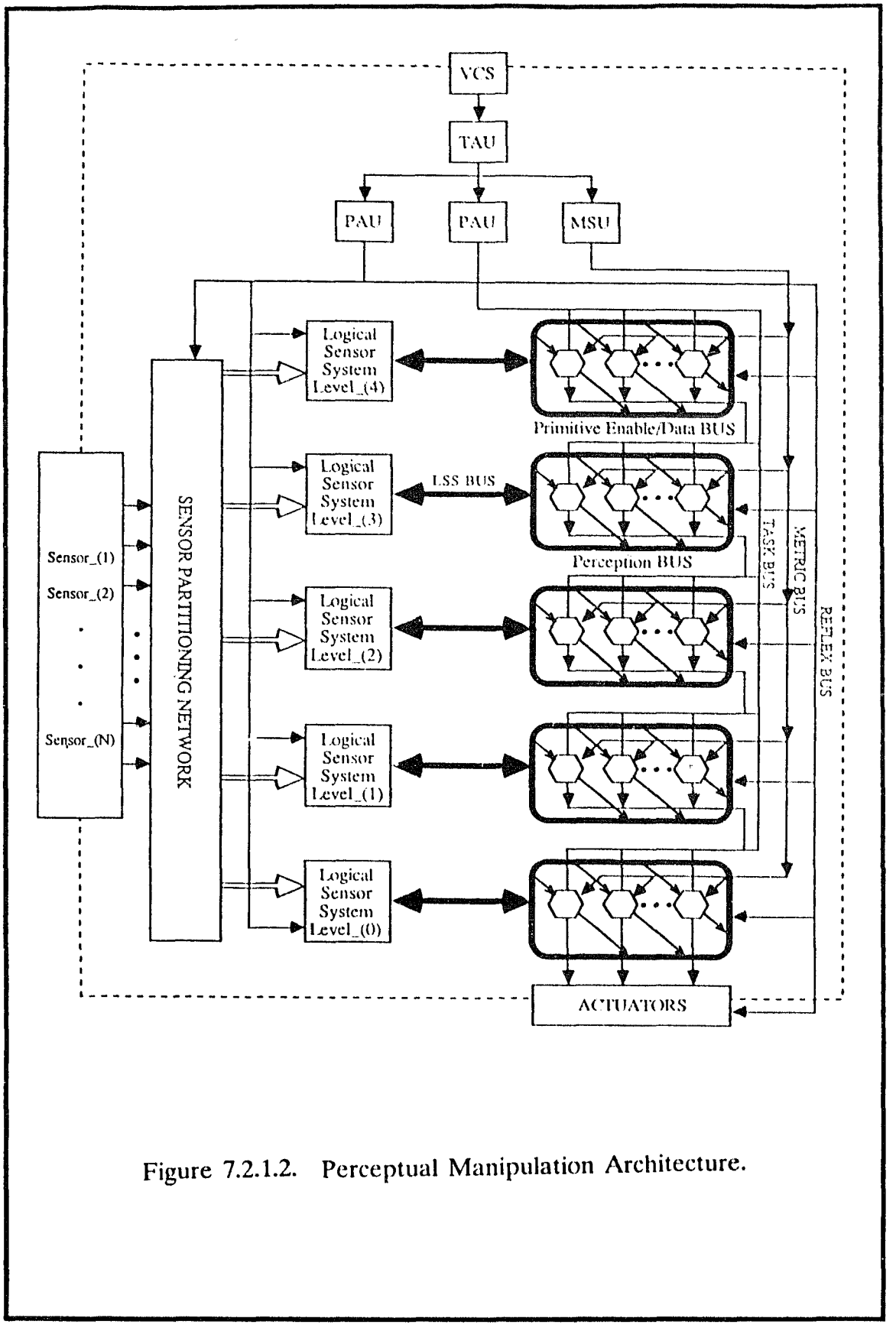


Figure 7.2.1.2. Perceptual Manipulation Architecture.

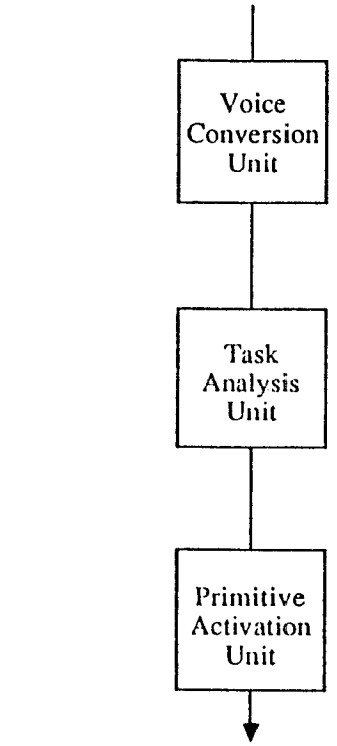
Task Decomposition Primitives

OBJECT LEVEL	Free Motion Guarded Motion Fine Motion	Free Force Application Guarded Force Application Fine Force Application
GRASP LEVEL	Preshape Grasp / Regrasp Re-Acquisition	Rigid Grasp Acquisition Manipulation
MULTIPLE ARM LEVEL	Free Motion Guarded Motion Fine Motion	Free Force Application Guarded Force Application Fine Force Application
ARM LEVEL	Free Motion Guarded Motion Fine Motion	Free Force Application Guarded Force Application Fine Force Application
JOINT LEVEL	Free Motion Guarded Motion Fine Motion	Free Force Application Guarded Force Application Fine Force Application
ACTUATION LEVEL (Tendon Drive, e.g., Asada Hand) (Direct Drive, e.g., Salisbury Hand) (Pneumatic Drive, e.g., Utah Hand) (Geared Drive, e.g., Puma 560, RMS, Robotics Research Arms)	Free Motion Guarded Motion	Free Force Application Guarded Force Application

Figure 7.2.1.3. A Necessary and Sufficient hierarchy of task decomposition primitives for prototypical mechanical interaction tasks (i.e., involving forces and motions only). (ref: [279])

TASK: Shared ORU Acquisition, Removal and Transport During HST Camera Servicing

"ACQUIRE HANDLE FOR SHARED REMOVAL AND MANEUVER"

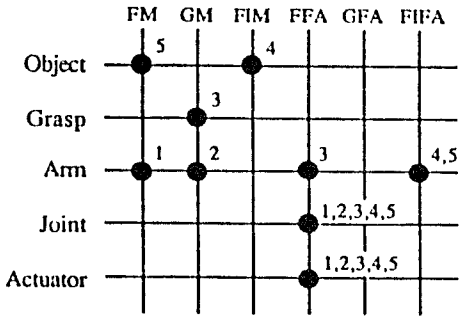


Crew-EVA Primitives Instantiated

ACQUIRE ; MOVE ; GRASP
REMOVE (HOLD) ;
MANEUVER (HOLD)

Task Primitives Instantiated

LOCATE (handle) , REACH ,
APPROACH , GRASP ,
DE-MATE (shared) , TRANSPORT (shared)



Behavioral Primitives Instantiated

1: (Free Motion)*
2: (Guarded Motion)*
3: (Grasp)*
4: (Fine Force Application)*
5: (Fine Force Application)*

Figure 7.2.1.4. Task decomposition hierarchy for a prototypical mechanical interaction task, i.e., shared transportation of loads by a human and robots.

TASK PRIMITIVE USAGE PROFILE DURING RAEVA OPERATIONS

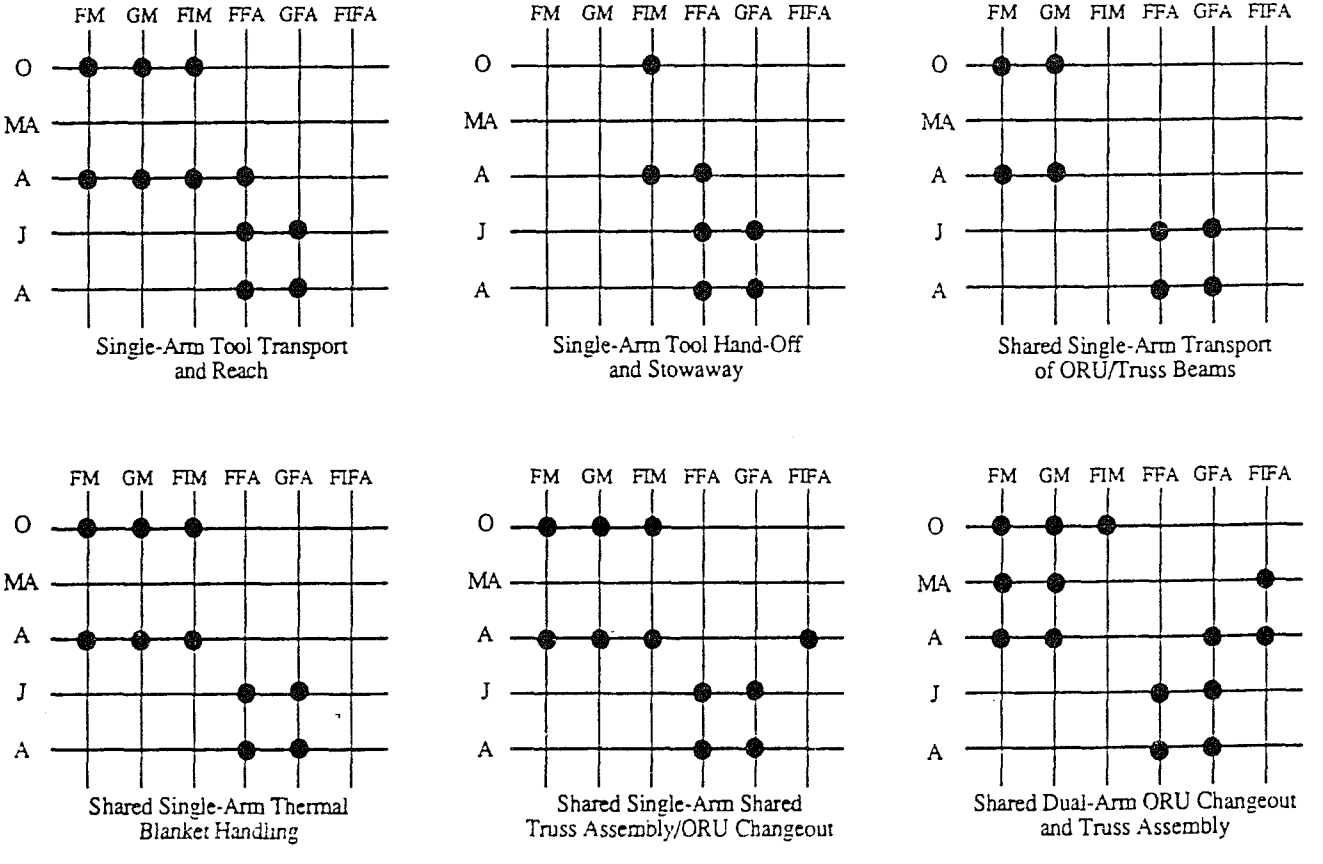


Figure 7.2.1.5. Sequencing profile for task-primitives during prototypical task execution for a class of RAEVA relevant tasks.

of task primitives are arranged to explicitly capture robot-environment interaction modes at the Object, Grasp (Multiple arms), Arm, Joint and the Actuator Levels. Such an architecture would represent a departure from the traditional approaches based upon information processing (symbol/numeric), process rates and interaction context (global/local). Also, unlike iterative robot architectures, e.g., HANDEY, SOAR, HICS, etc., each behavioral primitive recurses within a set of perceptual structures that are engineered to provide complete control observability.

Fig. 7.2.1.4 further elucidates the usage of such primitives, during Shared ORU Acquisition, Removal and Transport. The shared transport sequence begins with a set of recursive instantiations of elemental Arm-level Free Motions until proximity, followed by Guarded Motions until the handle of the ORU is within the gripper workspace. Thereafter, repeated invocations of elemental (Object-level) Grasp followed by acquisition primitive allows stable acquisition of the handle in the gripper. During shared transport, the robotic system would be placed under Arm-level Fine Force Application. The numbers, 1 through 5 represent the temporal ordering of Reach, Approach, Grasp, Acquisition and shared Transport phases of the operation (Venkataraman and Lyons in [283]), while the symbol "*" denotes recursive primitive instantiation until the execution metric are satisfied. Figure 7.2.1.5 presents a sequencing profile for task-primitives during prototypical task execution for a class of RAEVA relevant tasks.

7.2.1. Perception Architecture

Fig. 7.2.1.1(a) and 7.2.1.1(b) expand on the perceptual structures for the above Shared Transportation example. A Task Analysis Unit (TAU) (see Fig. 7.2.1), determines the configuration of a set of sensors, necessary for complete

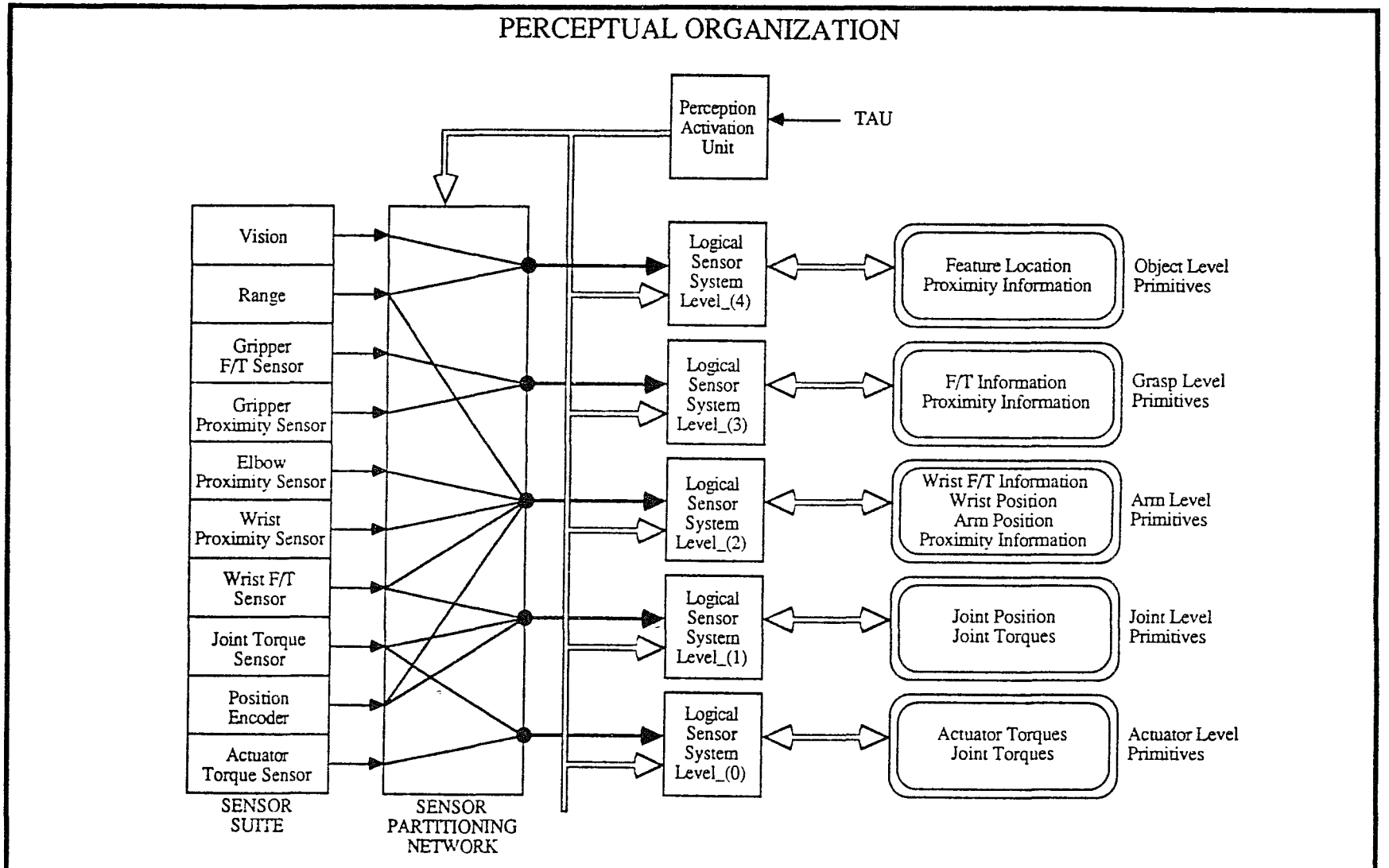


Figure 7.2.1.1(a) Perceptual organization and architecture

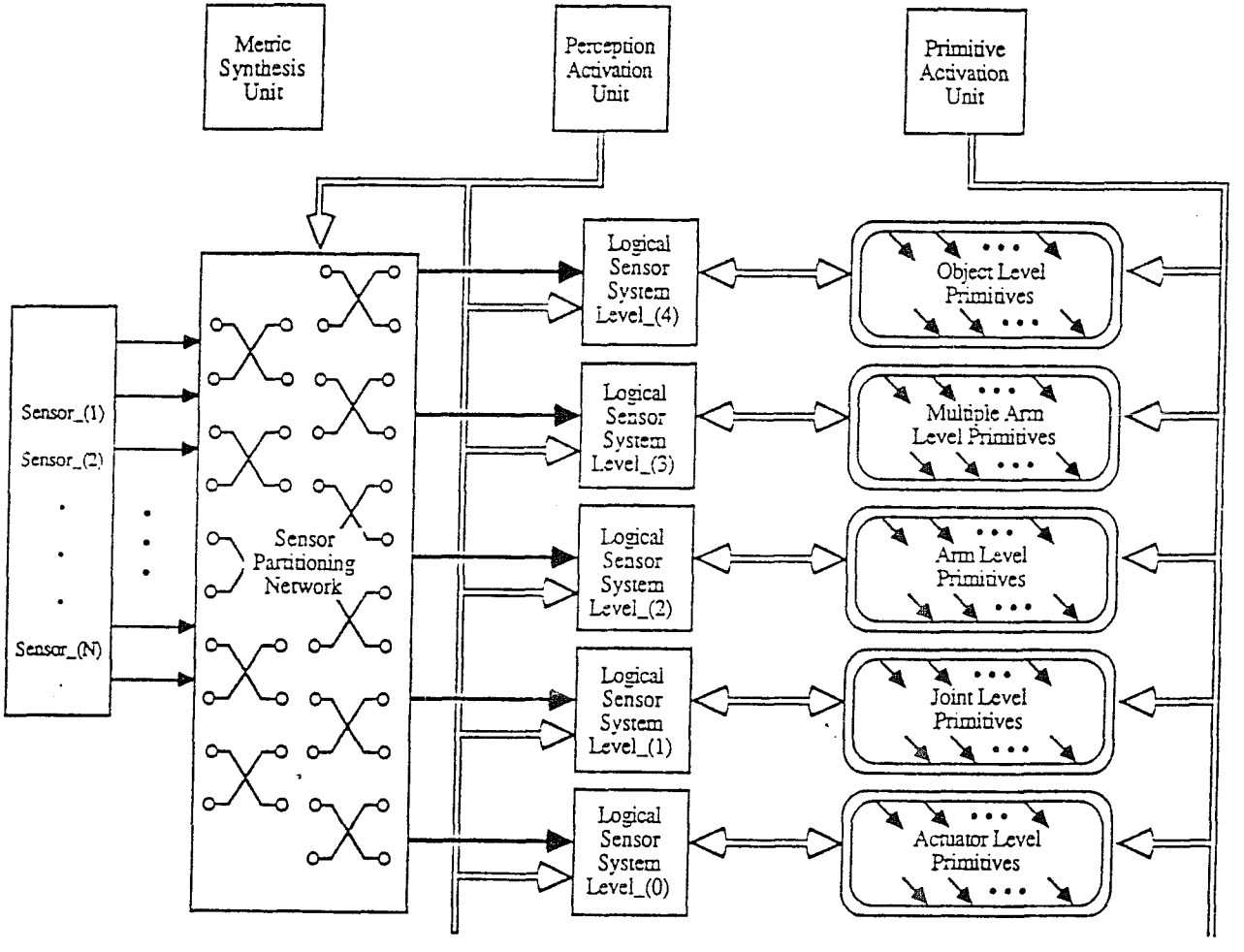


Figure 7.2.1.1(b) Abstract architecture for perception.

control observability during task executions. This information is used by a Perception Activation Unit (PAU) that enables appropriate sensor routing within a Sensor Partitioning Network (SPN). In addition, the necessary sensor integration mechanisms are configured within a Logical Sensor Structure (LSS) for the construction of relevant perceptual features to provide control observability. For example, during guarded motion, PAU would route vision and proximity information through the SPN. Within the LSS, feature level descriptions of the arm will be extracted from the vision and proximity data. The LSS concept is discussed in detail in [123]. The output of LSS elements is routed to the control execution units via the Perception Bus (refer Fig. 7.2.1.4). Each control unit is repeatedly invoked until the output from the latter corresponds with the goal condition.

7.2.2. Control Architecture

Fig. 7.2.2.1 illustrates a typical control execution unit. It comprises of four basic elements: a metric reasoning engine, adaptive execution element, e.g., neural networks, metric enforcing mechanisms and local memory for storing relevant task execution states. The Metric Synthesis Unit (in Fig. 7.2.1.2) only provides global performance considerations, e.g., close up to 5 cm near an ORU. The Metric reasoning engine then computes the corresponding control tracking performance, i.e. the precision metric, to be achieved in the perception-space to match the information provided by the corresponding LSS. For example, if the adaptive execution element is a neural network, then the metric engine needs to set up bounds on network convergence that will guarantee desired performance. Figure 7.2.2.2 provides a schematic comparision of the Perceptual Manipulation Architecture with existing robot manipulation architectures discussed in Section

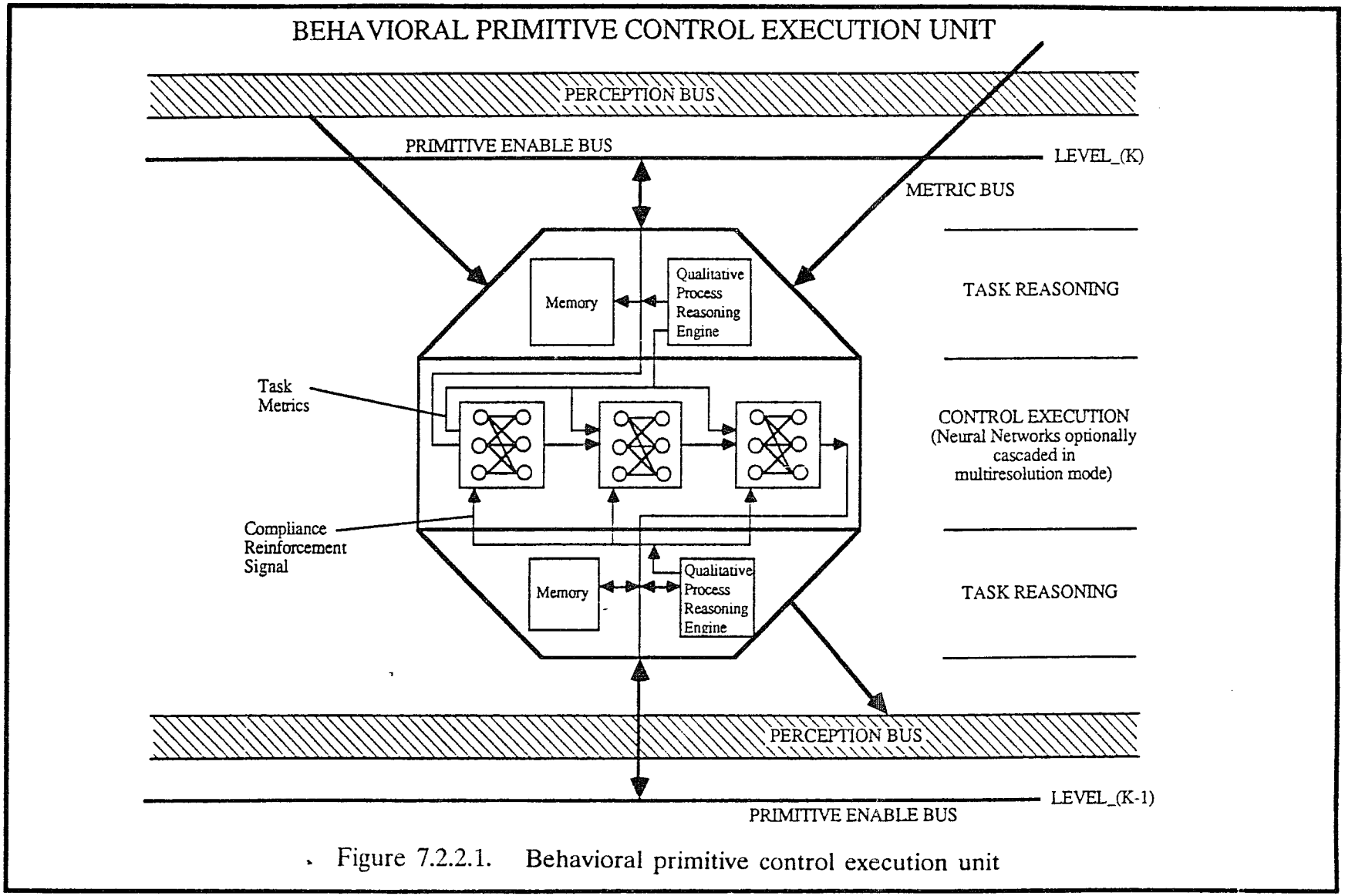


Figure 7.2.2.1. Behavioral primitive control execution unit

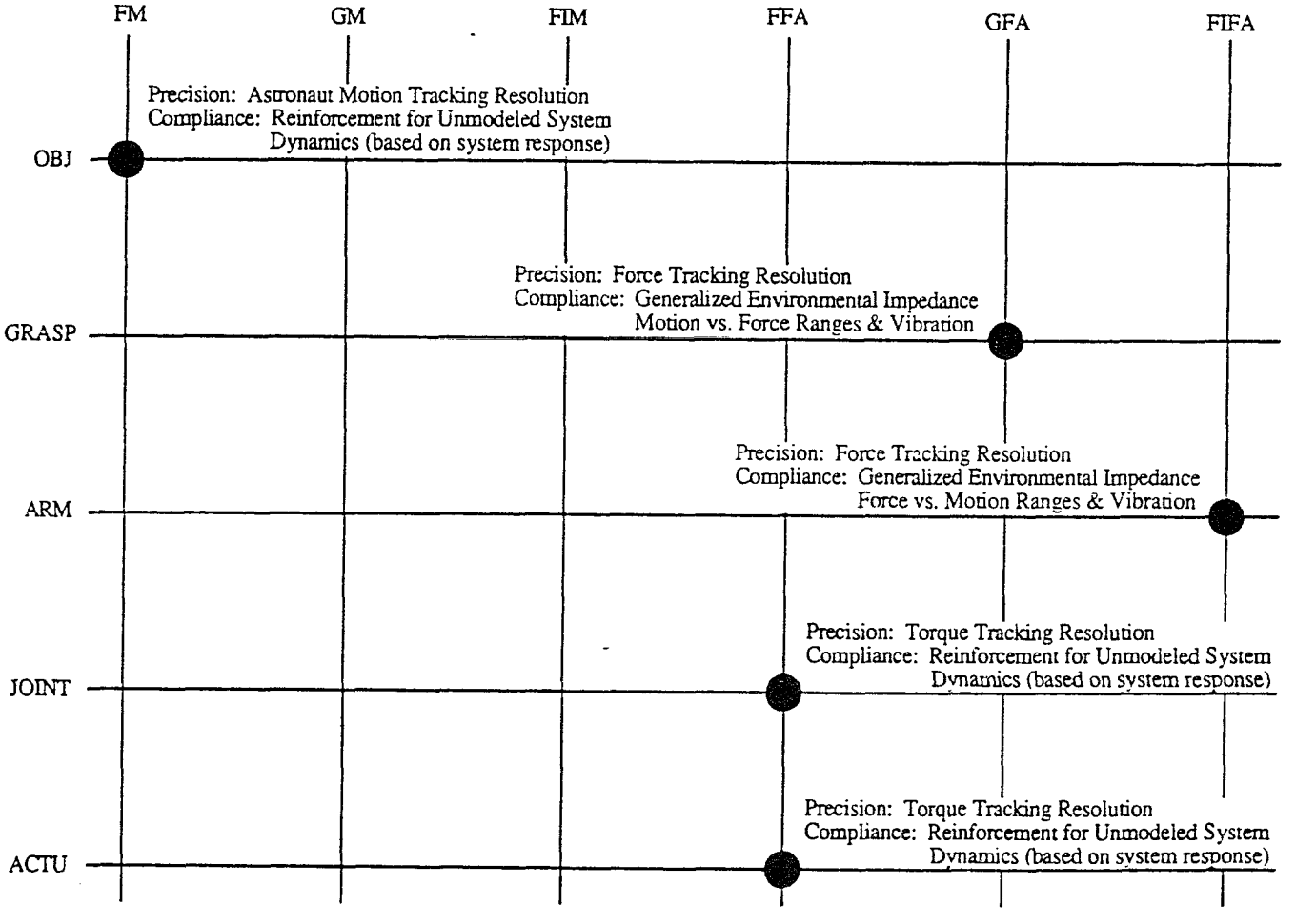


Figure 7.2.2.2. Metric synthesis during prototypical contact interaction task (e.g., shared transportation of loads in space by robot and astronaut).

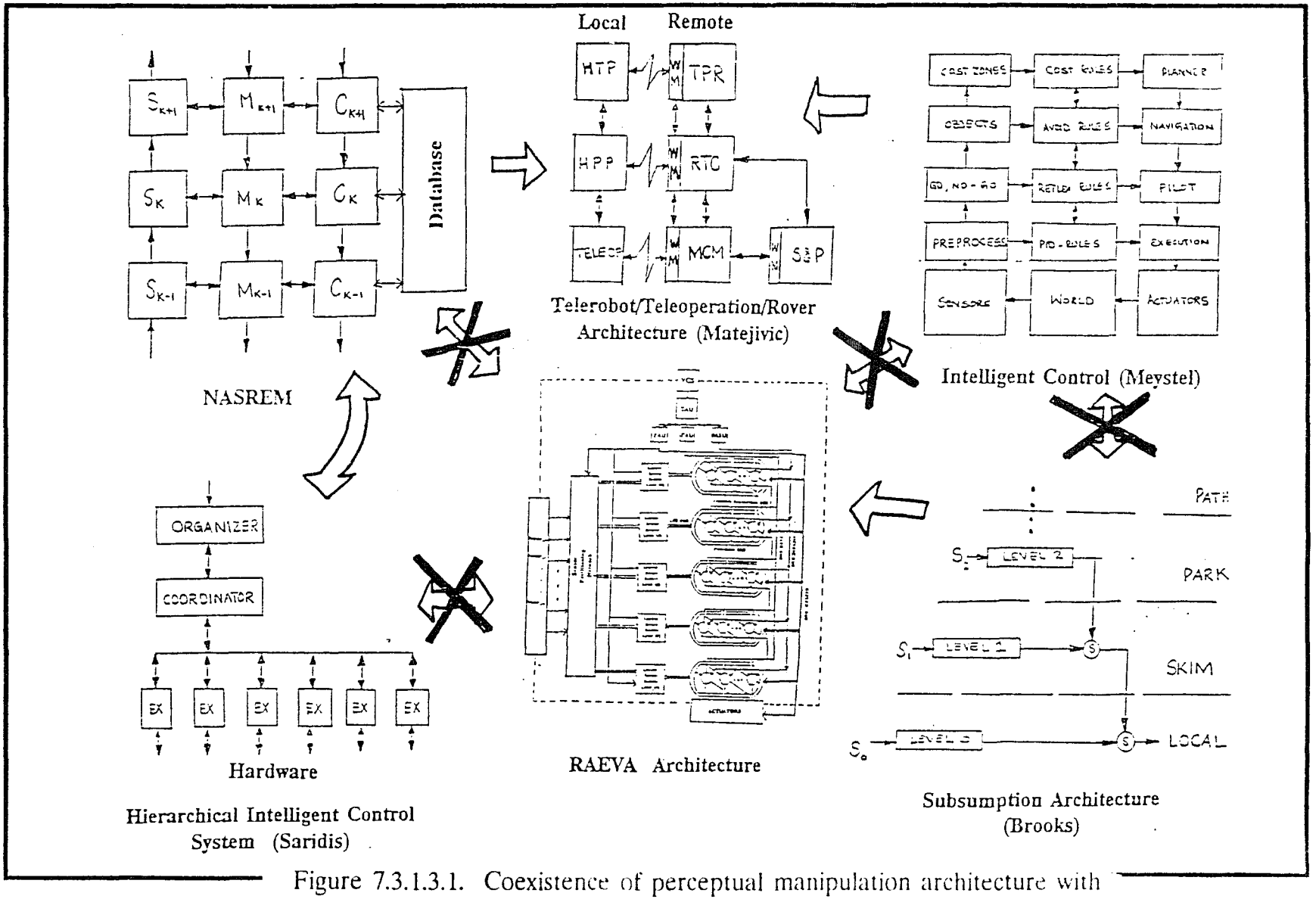


Figure 7.3.1.3.1. Coexistence of perceptual manipulation architecture with existing robotic architectures.

7.2, namely NASREM (Albus et al. [8]), Subsumption Architecture (Brooks [54]), Intelligent Control (IC) Architecture (Meystel [210]), Intelligent Control System (ICS) Architecture (Saridis [253]).

7.3 SID-Based Neurocontrol Algorithms

Proceeding along our derivation of manipulation architecture, we exploit the SID formalisms developed in the preceding chapters to precisely identify and define the internals of the control execution unit discussed in Section 7.2.2.

7.3.1. Task Space Control

Consider the following scenario, wherein we are trying to to incrementally execute vision-guided multijoint positioning to the proximity of the targeted object, say an ORU handle, and attain an orientation that facilitates subsequent acquisition of the handle. This corresponds to executing the Free Motion primitive as defined in the previous section. At the begining of of each control cycle, the distance vector between the end effector of the arm and the ORU handle is computed by the vision system [173] and supplied to a free motion control execution unit. This distance vector is mapped into the robot actuation space in the following manner:

Since *task space* may be informally defined as a space in which system performance sensitivity to control errors is maximal (formal definitions may be found in Venkatarmaman and Lyons in [283], a space constructed out of the relative location and orientation of the robot arm's end effector with respect to the targeted object may be considered as a candidate for free motions. The coordinates may be defined with respect to any arbitrary coordinate system. A particular instance of the former is well known in literature as the robot's *operational*

space introduced by Khatib [160,161]. Free motion sensorimotor control will therefore be responsible for servoing the end effector to some neighborhood of a pre-determined *proximity distance* from the target object. The relative distance would typically form a *Euclidean* space $X \epsilon \mathbf{R}^6$, and would form a set of generalized coordinates in the *Lagrangian* sense [164]. Let X represent the object-end effector distance in the visual space (e.g., reference coordinate system is located at the camera itself). The robot arm dynamics may be expressed in the visual space in the following manner [233]:

$$\begin{aligned} F^a &= I(X)\ \ddot{X} + C\left(X, \dot{X}\right) + G(X) \\ &= f_1\left(X, \dot{X}, \ddot{X}\right) \end{aligned} \tag{7.3.1.1}$$

$$\begin{aligned} \ddot{X} &= I^{-1}\left(-C\left(X, \dot{X}\right) - G(X) + F_a\right) \\ &= f_2\left(X, \dot{X}, F_a\right) \end{aligned} \tag{7.3.1.2}$$

where $F^a \epsilon \mathbf{R}^6$ represents *pseudo* actuation signals in the visual space. $I(.)$, $C(.)$ and $G(.)$ are the inertial, coriolis plus centripetal forces and the gravitational term in the visual space. Note that the relationships derived in [160] between joint inertias, coriolis and gravitational forces, and their counterparts in the operational space, qualitatively hold for the corresponding terms in the above equation as well.

To obtain values of F^a, given X, $\dot{X}$ and $\ddot{X}$, traditional methods would require the calculation of manipulator inertias, and coriolis and gravitational forces in a manipulator's joint space, their transformation into the visual space. This, in general, is computationally intensive, and requires special purpose hardware [51]. In a departure from traditional robotics, we adopt the alternate approach of learning the multivariate, dynamics equation, Eqs. (7.3.1.1) and (7.3.1.2) as a nonlinear input-output mapping between the actuation signal vector F_a, and the

states $X, \dot{X}, \ddot{X}$ using neural learning formalism *Algorithm SID_6*. In addition, to computational speedups resulting from massively, parallel and asynchronous information processing, neural networks can enable adaptation in real-time to changes in the robot's structural characteristics (including the effects of backlash and friction) [152]. Let, $\hat{f}_1$ and $\hat{f}_2$ represent the learnt approximations of f_1 and f_2 in Eqs. (7.3.1.1). The function $\hat{f}_1$ performs the system identification, while $\hat{f}_2$ may be tuned for motion control. Once $\hat{f}_1$ and $\hat{f}_2$ are obtained, the dynamical systems in Eqs. (7.3.1.1) and (7.3.1.2) may be discretized to yield:

$$\begin{aligned} F_k^a &= \hat{f}_1\left(X_k, \dot{X}_k, \ddot{X}_k\right) \\ \dot{X}_{k+1} &= \dot{X}_k + \alpha\ \hat{f}_2\left(X_k, \dot{X}_k, F_k^a\right) \end{aligned} \tag{7.3.1.3}$$

where α denotes the step size. Fig. 7.3.1.1 shows the schematic of the open-loop (during free motion) robotic arm after system identification has been completed. In the system-theoretic sense, Z^{-1} represents the time delay elements.

A multitude of methodologies exist for the design of discrete time controllers for the identified system [22,260]. In our current stages of controller design, we drive the design process using a reference model. This method has been extensively discussed in conventional adaptive control [220]. Briefly, the intent is to design an input (and/or feedback) shaping function that will make the closed-loop controlled system appear exactly as specified by the reference model. As a simplest example, consider the formulation using the Pole Placement Method [82]. The controller for a linear system can be derived as: Assume a system of the form

$$\frac{d^2x}{dt^2} + a\frac{dx}{dt} + bx = u \tag{7.3.1.4}$$

is driven by a reference model of the form

$$\frac{d^2x}{dt^2} + c\frac{dx}{dt} + dx = r \tag{7.3.1.5}$$

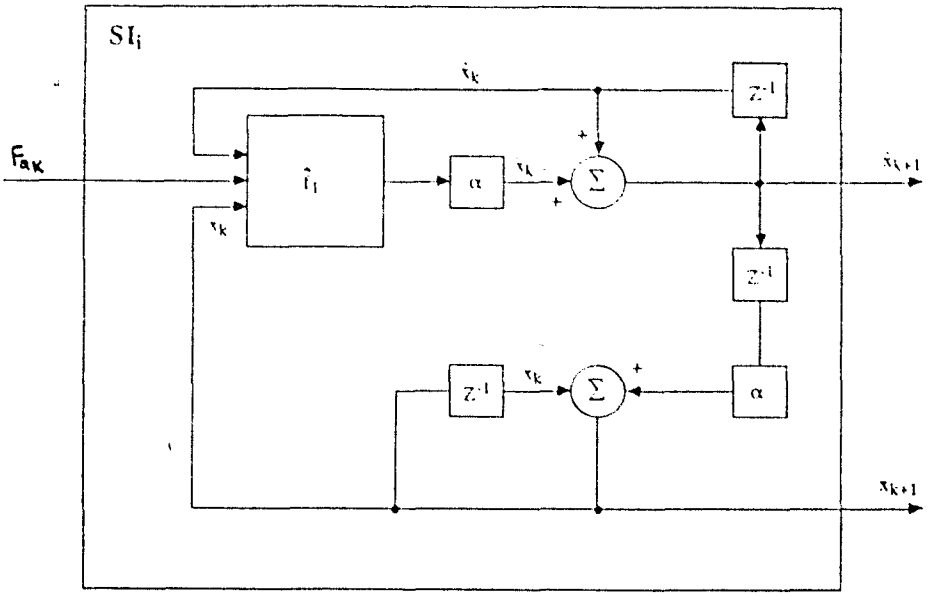


Figure 7.3.1.1. Arm dynamics system identification.

where a, b, c and d denote constants. Assuming full state feedback, the controller can then be simply computed using

$$u = (a - c)\frac{dx}{dt} + (b - d)x + r \qquad (7.3.1.6)$$

In the Laplacian domain, the transfer function is given by

$$Y(s) = \frac{1}{(s+a)(s+b)}U(s)$$

is driven by a reference model

$$Y(s) = \frac{K}{(s+c)(s+d)}r(s) \qquad (7.3.1.7)$$

The corresponding control law can then be written as

$$U(s) = \frac{K(s+a)(s+b)}{(s+c)(s+d)} \qquad (7.3.1.8)$$

We shall briefly discuss our metric-driven derivation of reference models in a later section. For sake of simplicity, consider only linear time invariant reference models [220] with the form

$$\ddot{X} = -\,p_1\,\dot{X} \;-\; p_2\,X \;+\; q \qquad (7.3.1.9)$$

where, $p_i \;\epsilon\; \mathbf{R}^6 \times \mathbf{R}^6$ are the coefficients of the linear system, and $q \epsilon \mathbf{R}^n$ is the reference signal. The actuation signals can then be computed using

$$F_k^a = \hat{f}_1\left(X_k, \dot{X}_k, \left(-p_1\,\dot{X}_k \;+\; -p_2\,X_k \;+\; q_k\right)\right) \qquad (7.3.1.10)$$

Fig. 7.3.1.2 shows the schematic of the controlled system. Note that the output of $\hat{f}_2$ corresponds to the pseudo actuation signals F_a at instant k, and represents an input into the joint-level controller that we discuss shortly.

The transformation of F_a into the robot's joint space is given in the following manner:

$$\tau_a = J^T\,F_a \qquad (7.3.1.11)$$

where, J represents the standard manipulator jacobian [233]. A kew concern during free motion is obstacle avoidance. Many global methods that rely on apriori knowledge about the location (and motions if any) of the obstacles have been proposed [146,161,195,256]. We resort to repulsive potential functions for local obstacle avoidance [160,161]. The other contribution to joint torques, therefore, would be as a result of obstacle avoidance forces. To ensure that obstacles will indeed be avoided, we propose a *log* based potential function of the form [256]:

$$E_o = K\;log\,(Z) \qquad (7.3.1.12)$$

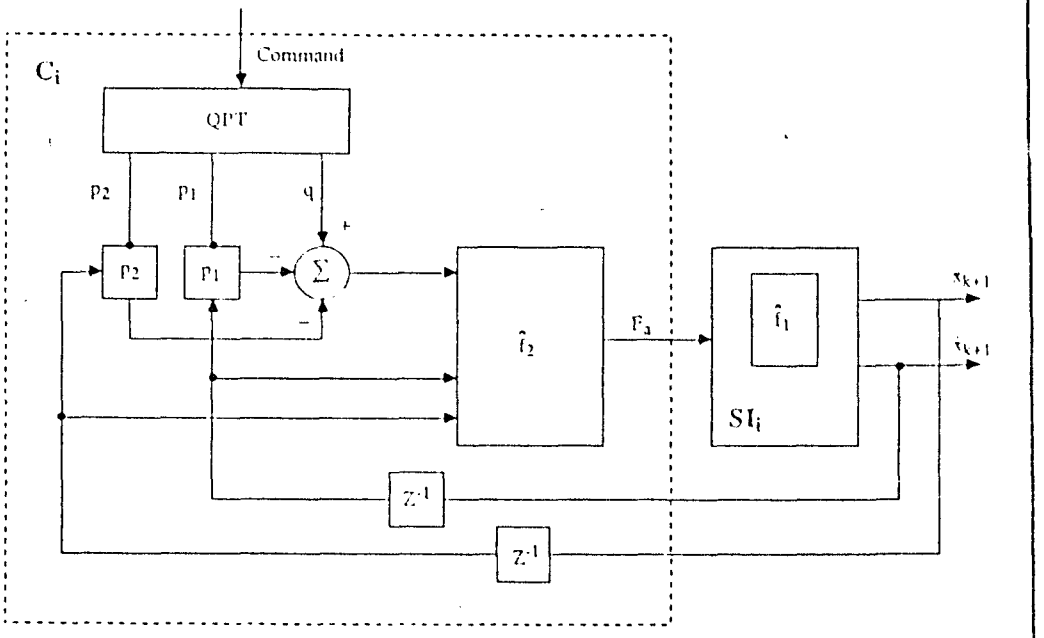


Figure 7.3.1.2. Arm dynamics identification and control.

where, K is a time (in)variant gain, and Z is the perpendicular distance between the links of the arm, and the nearest obstacle. The quantity Z is well known in robotics literature [256]. The repulsive forces are:

$$F_o = \frac{\partial E_o}{\partial Z} = \frac{K}{Z} \tag{7.3.1.13}$$

which implies that as $Z \rightarrow 0$, $F_o \rightarrow \infty$. Assuming K proximity sensors, rigidly

affixed to the body of the arm, the value Z can be computed as:

$$Z = g^{\theta}(\Theta) \tag{7.3.1.14}$$

where, Θ is the corresponding vector of joint angles. At the same instant let the proximity sensor readings be recorded in a vector S. Let the jacobian between the space of Z values, and the joint space be J_o. Since the proximity sensors are fixed at known locations, it is reasonable to assume that J_o will be known. That is, for known values of Θ, the corresponding obstacle avoidance forces in the joint space τ_o may be computed as:

$$\begin{aligned} \tau_o &= J_o^T F_o \\ &= J_o^T \frac{K}{Z} \\ &= J_o^T \frac{K}{g^{\theta}(\Theta)} \\ &= J_o^T \frac{K}{g^{s}(S)} \end{aligned} \tag{7.3.1.15}$$

where $g^s(.)$ is a complex nonlinear mapping function between S and Z to be learnt. Therefore,

$$\begin{aligned} \tau &= \tau_a + \tau_o \\ &= \left[J^T,\ J_o^T\right] \begin{bmatrix} F_a \\ \frac{K}{g^s(S)} \end{bmatrix} \end{aligned} \tag{7.3.1.16}$$

where τ_a and τ_o denote torque required for actuation and obstacle avoidance respectively. A schematic of the decomposition of task space and proximity sensor space information into the robot's joint space is shown in Figure 7.3.1.3. In the figure, the units SI_1 and SI_2 denote system identification units at the joint and arm level respectively. The unit labeled QPT is responsible for parametrized selection of appropriate refernce model [220]. The Singularity Interaction Dynamics formalisms described in Chapter Two through Five may be used for online learning of the nonlinear map, $g^s(.)$, map

$$\tau = g^{\tau}(F_a, S) \tag{7.3.1.16}$$

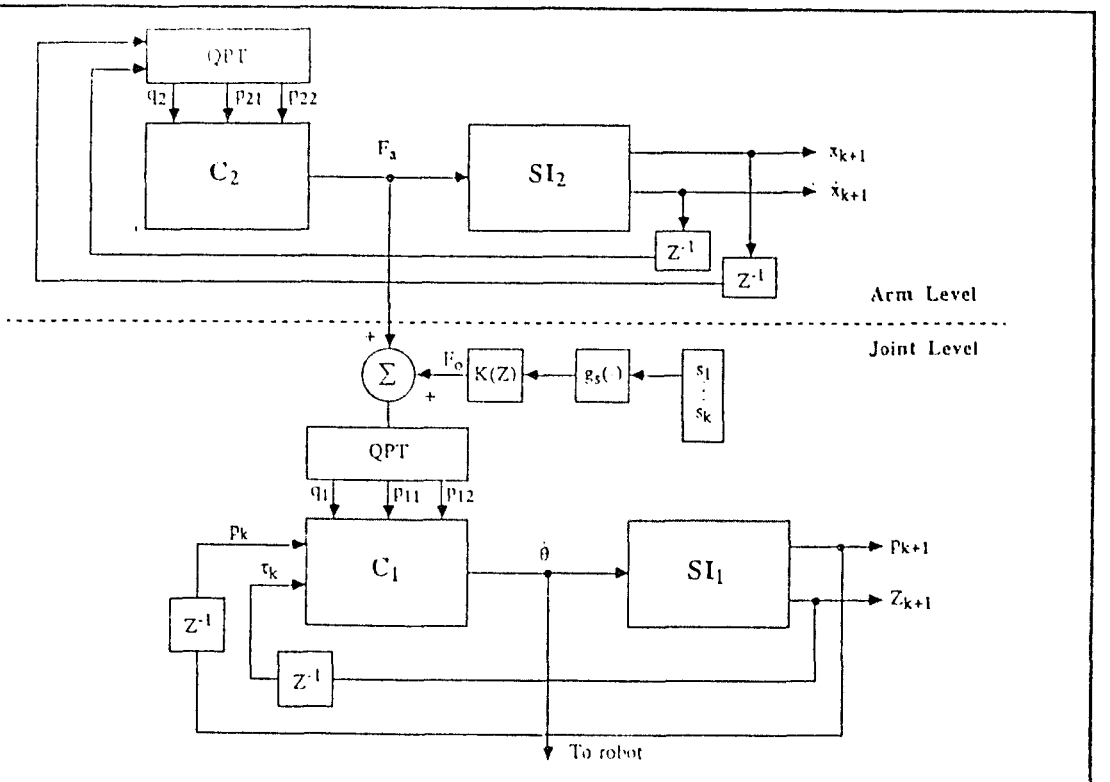


Figure 7.3.1.3. Free motion control.

to provide τ_j (at joint level) in real-time.

7.3.2. Joint Control

As proposed in [129], interconnected dynamical systems may be described using alternating Impedances and Admittances. This allows one to maintain consistency in bookkeeping power flow variables. For example, during free, guarded or fine motions, a robot imparts flow vectors across an interaction port (in its task space) to the (free space) environment around it [279]. Consequently, the robot can be described in the form of a Lagrangian, while the environment is an

admittance. This calls for a Co-Lagrangian (an admittance-like) description for the actuators placed at the joints. As discussed by Venkataraman in [279], the Co-Lagrangians are based on the conservation of *kinetic energy* K and *potential co-energy* P^*. Given that

$$\begin{aligned} dK &= \dot{\Theta}_a^T \, dp \\ dP^* &= \Theta_a^T \, d\tau \end{aligned} \tag{7.3.2.1}$$

And the Co-Lagrangian function L^* is

$$L^* = dK \; + \; dP^* \tag{7.3.2.2}$$

where, p is the angular momentum at the joints, and Θ_a represents the angular motion of the actuators located at the joints. The basic structure of the dynamical equation derived herefrom is similar to that of a Lagrangian, i.e.,

$$\Theta_a = I^*(p)\,\dot{\tau} \; + \; C^*(p,\tau) \; + \; G^*(p) \tag{7.3.2.3}$$

where, I^*, C^* and G^* are the force-momenta duals of inertia, coriolis and gravitation terms for the actuator. Since, is is extremely difficult to obtain the co-Lagrangian dynamics equation for a system, Venkataraman proposed a pseudo-form in [279]. The latter may be derived through a substitution for $\ddot{\Theta}_a$ in the actuator Lagrangian in terms of the task space Lagrangian. In our formulation, we consider the co-Lagrangian as a nonlinear map between actuator velocities Θ_a, and the terms p, τ, $\dot{\tau}$, thereby obviating the need for explicit computation of the pseudo co-Lagrangian. We define nonlinear functions f_3 and f_4 as

$$\begin{aligned} \dot{\Theta}_a &= f_3(p, \tau, \dot{\tau}) \\ \dot{\tau} &= f_4(p, \tau, \Theta_a) \end{aligned} \tag{7.3.2.4}$$

The nonlinear functions, f_3 can be learnt off-line in a manner similar to the process for f_1 and f_2 in Eqs. (7.3.1.1). Let $\hat{f}^3$ be the corresponding learned

nonlinear approximator. Actuation control is performed to mimic a reference model of the form

$$\dot{F} = -p_3 F - p_4 p + r \qquad (7.3.2.5)$$

which implies that f_4 can be learnt analogous to f_2, and parameters of the equations above can be used to compute $\dot{\Theta}_{a_k}$. Fig. 7.3.1.3 summarizes the control architecture for the entire arm free motion control.

7.3.3. Guarded Motion Control

In this subsection we discuss the implementation the behavioral control unit for executing guarded motion. In the case of guarded motion, however, (in addition to the goal specification in the visual perception space), a guarding force needs to be be specified [203,204]. The latter accounts for unprecedented variations in the end effector's environment (Mason in [51]). With this, the contribution to the task space pseudo actuation signal will arise from two sources: one for the actual task space motion control of the arm, and the second, a guarding force F_g. That is,

$$\dot{X}_{k+1} = \dot{X}_k + \alpha \hat{f}_1 \left(X_k, \dot{X}_k, F_a \right) + \frac{\partial C \ (F_{g_d}, F_{g_a})}{\partial F_{g_a} \dot{F}_{g_a}} \qquad (7.3.3.1)$$

where $C(.)$ is a generalized nonlinear compliance function [203], that represents the impedance matching required for a given environment. During free motions, for example, $C = 0$. It is, in general, extremely difficult to determine a compliance function that can capture all expected environmental situations. The compliance functions can be derived by learning the impedances between a robot and its environment [129], based upon notions of coupled robot-environment stability properties.

7.3.4. Fine Force Application Control

Within our proposed scenario, a key primitive of interest is fine force application, as introduced in Section 7.2, to provide for stable contact interactions, e.g., during shared transportation of payloads. In this case, the robot is modelled as an admittance that *admits* the human, whose dynamics takes the form of an impedance. Accordingly, we adopt the dynamic models first proposed in [279], wherein the coupled robot-environment dynamics may be given using:

$$\begin{aligned} \ddot{X} &= f_e\left(X, \dot{X}, F\right) \\ \dot{F} &= f_r\left(p, F, \dot{X}_a, \dot{X}\right) \end{aligned} \tag{7.3.4.1}$$

where the subscripts e and r refer to environment and robot respectively. Combining the above yields

$$\ddot{\Pi} = f_5\left(p, F, X, \dot{X}, \dot{X}_a\right) \tag{7.3.4.2}$$

where, $\Pi = \left[X^T, p^T\right]^T$ and nonlinear function, $f_5 = \left[f_e^T, f_R^T\right]^T$. Eq. (7.3.4.2) possesses the same structure as Eq. (7.3.1.1) and (7.3.1.2). The equivalent of Eqs. (7.3.1.3) can be expressed as:

$$\begin{bmatrix} \underline{0} \\ \dot{X}_a \end{bmatrix} = f_6\left(X, \dot{X}, \ddot{X}, p, F, \dot{F}\right) \tag{7.3.4.3}$$

where, $f_6 = \begin{bmatrix} f_e\left(X, \dot{X}, \ddot{X}, F\right) \\ f_R\left(p, F, \dot{F}, \dot{X}\right) \end{bmatrix}$. Once again $\hat{f}_5$ and $\hat{f}_6$ represent the learnt approximations to f_5 and f_6 respectively. Fig 7.3.4.1 shows the schematic for system identification.

For control, let the reference model be of the form:

$$\begin{aligned} \begin{bmatrix} \ddot{X} \\ \dot{F} \end{bmatrix} &= \begin{bmatrix} -p11 & 0 \\ 0 & -p12 \end{bmatrix} \begin{bmatrix} \dot{X} \\ F \end{bmatrix} + \begin{bmatrix} -p21 & 0 \\ 0 & -p22 \end{bmatrix} \begin{bmatrix} X \\ p \end{bmatrix} + \begin{bmatrix} 0 \\ q \end{bmatrix} \\ \ddot{\Pi} &= -P_1\, \dot{\Pi} - P_2\, \Pi + \begin{bmatrix} 0 \\ q \end{bmatrix} \end{aligned} \tag{7.3.4.4}$$

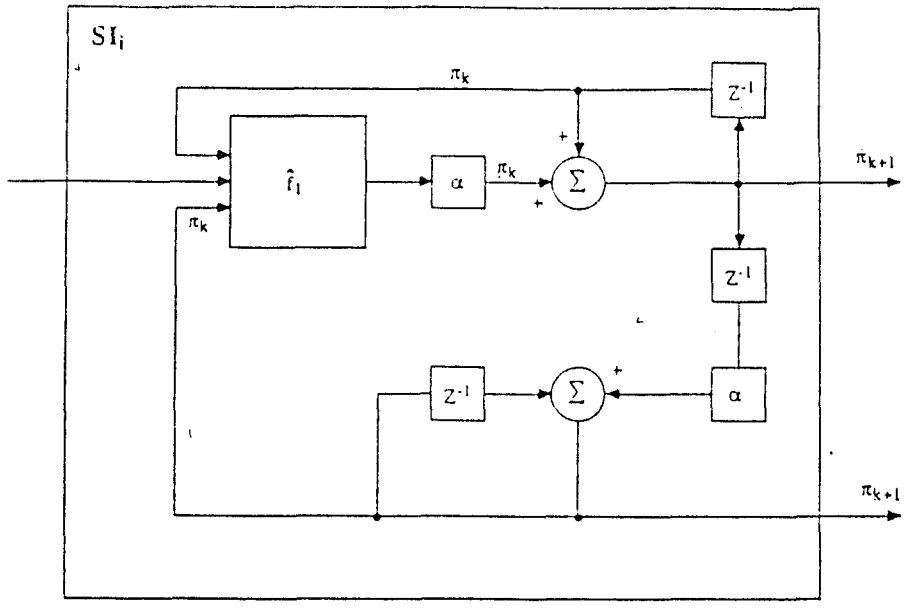


Figure 7.3.4.1. Robot and environment interaction dynamics identification.

The schematics in Fig. 7.3.4.1 and 7.3.4.2 illustrate the control architecture at the task level. Another important issue that needs to be addressed is obstacle avoidance. Although in principle, obstacle avoidance strategies will be identical to that during free motion (as discussed in Section 7.3.1), the underlying energy function requires some modification. We consider a function of the form

$$\begin{aligned} E_o &= K \ log\,(p_o) \\ \dot{Z} &= \frac{\partial E_o}{\partial p_o} \\ &= \frac{K}{p_o} \end{aligned} \tag{7.3.4.5}$$

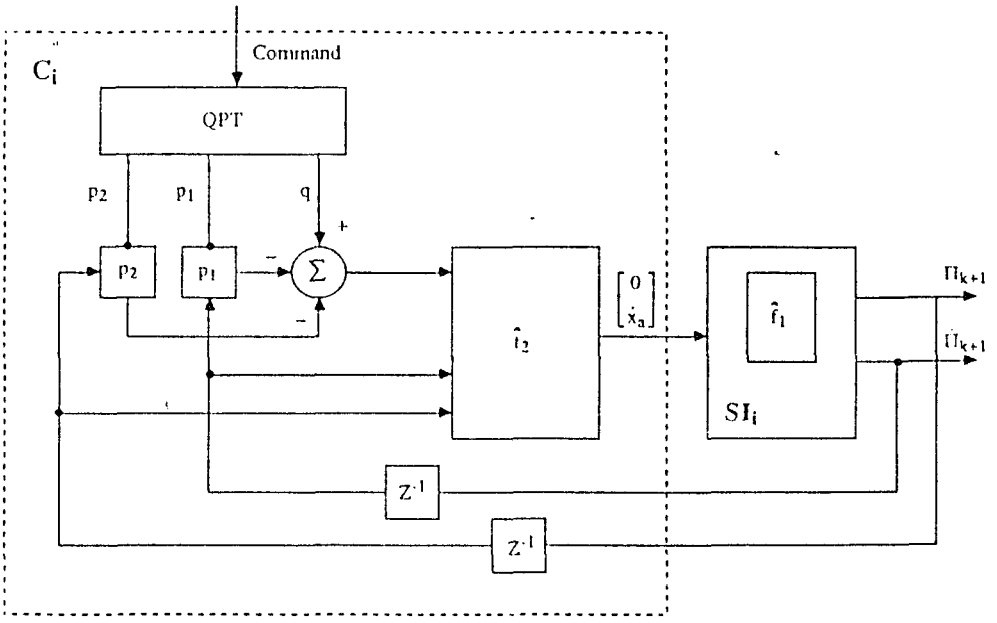


Figure 7.3.4.2. Robot and environment interaction dynamics identification and control.

Once again, $\overline{\dot{X}} = \left[\dot{X}_a^T,\ \dot{Z}^T\right]^T$, and the appropriate joint velocities $\dot{\Theta}$ may be obtained in a manner similar to that in free motion. The function g used in obstacle avoidance during free motions will also vary in the following manner:

$$\begin{aligned} g_\tau^*(p_j) &= p_o \\ g_s^*(p_S) &= p_o \end{aligned} \qquad (7.3.4.6)$$

Since the arm dynamics is described in the admittance form during fine force application, the actuator dynamics will be represented as an impedance that delivers necessary $\dot{\Theta}$ values. Joint lagrangian has a structure identical to Eqs.

(7.3.1.1)-(7.3.1.3). So, the system identification and control methodologies will be identical to those for free arm motions. However, the reference model will be derived here to deliver desired $\dot{\Theta}$ values instead of goal positions. Fig. 7.3.4.2 describes the joint control architecture, while Fig. 7.3.4.3 summarizes the entire fine force application control architecture.

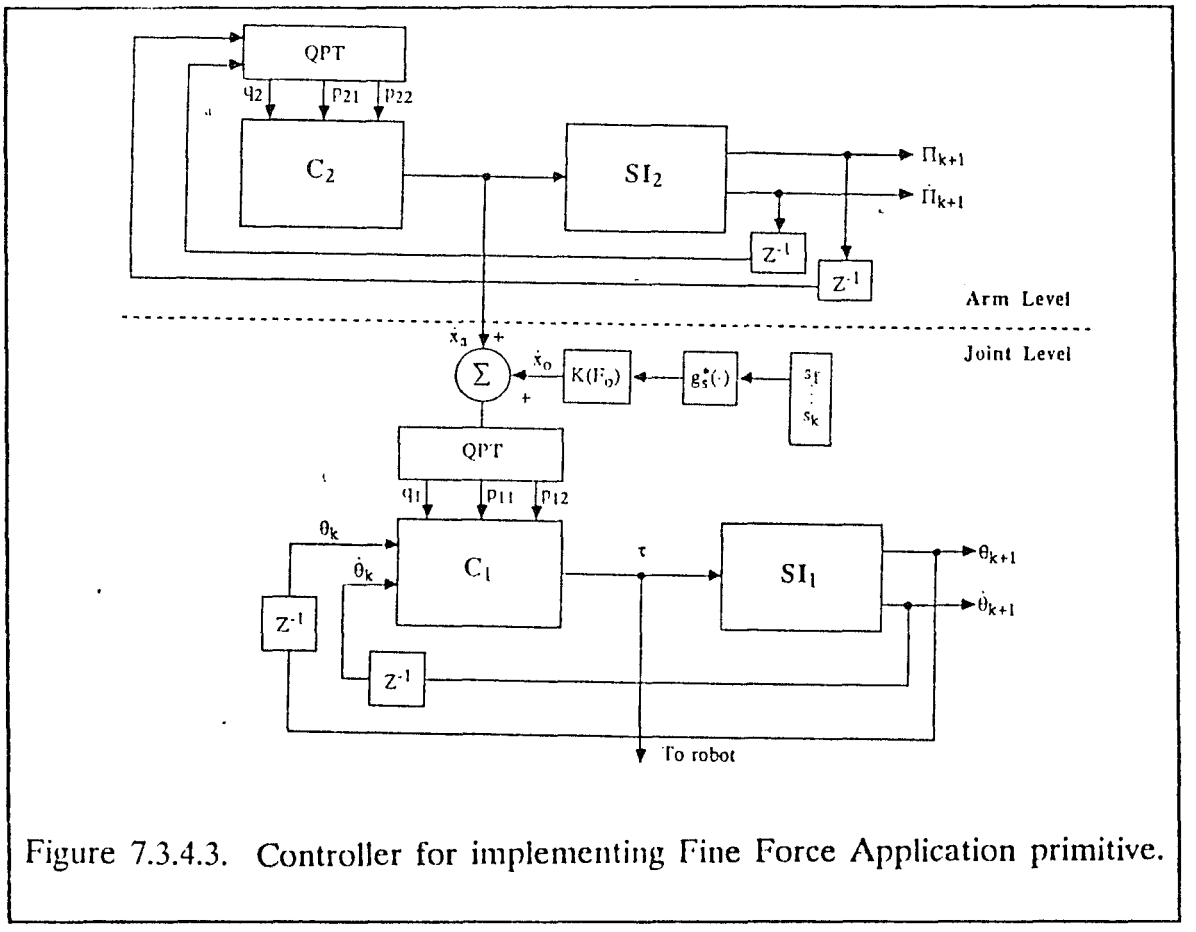


Figure 7.3.4.3. Controller for implementing Fine Force Application primitive.

Each and every control unit (whether they are at the arm or joint level) is driven to *mimic* a reference model. In the preceding discussion we had assumed the form of this model to be known. A key future objective is to (a) study

the control requirements to determine an appropriate form for reference models required for each control primitive (b) derive linear (and subsequently nonlinear [164,165]) reference models using newly emerging research in *Qualitative Physics* ([86] and references therein).

Consider a standard second order system or the form:

$$\ddot{X} = -p_1 \dot{X} - p_2 X + q \qquad (7.3.5.1)$$

The above system is analogous to a mechanical system consisting of a mass m, spring (of stiffness k) and dashpot (of coefficient b), $p_1 = \frac{b}{m}$ and $p_2 = \frac{k}{m}$. If the system is *modulated* by an external input q_e, then $q = \frac{q_e}{m}$. With this connection in mind, it is obvious that to make the system *stiffer*, one can increase the value of k, or reduce oscillations and make the system sluggish by increasing the value of b. Further payloads can be captured with the term m. Therefore, the problem of metric reasoning becomes one of *understanding* the information from sensors, and from other control units in the context of stable task primitive execution, and metric generation refers to the generation of the parameters, p_1, p_2 and q for control.

The controllers for fine motion, free force application etc., at different architectural levels may be derived in a similar manner.

7.4. Summary

In this chapter we have attempted to apply our work in the context of a novel real-world robotic application currently under development. Using recent results in formal task analysis and dynamic modeling [279], we systematically derived learning reformulations to nonparametric system identification and control. The resulting formalisms are well within the addressable complexity of our

SID learning algorithms. In fact, our earlier performance requirments necessitate a real-time learning framework as powerful as SID.

In terms of the application, we argued that operational safety, adaptation to a dynamically active agent (astronaut) in the workspace and overall system stability are fundamental to the design of Robotic-Assistants for synergistic task execution during EVA. The targeted application domain for such systems is EVA-enhancement. Technically, the problem entails building robotic systems with "doing intelligence" as opposed to reasoning intelligence, i.e., systems that can stabilize contact interactions with unstructured active environments, effect damage-proof interactions on fragile passive environments and accommodate dynamic unmodeled obstacles. The functional and technical requirements for the above system were analyzed to show that model-directed architectures and control formalisms have limited applicability to Robotic-Assistants. On the other hand, a natural tie-in to perceptual robotics was established. Further, a different architectural approach to the problem was presented. Unique characteristics of the proposed approach included the engineering of control execution chassis using a vocabulary of behavioral primitives derived from the physics of interactions rather than adhoc macros; perceptual structures engineered for complete control observability; perception-directed execution behavior rather than complete reliance on a priori world-models; and, the usage of competing, precision and compliance metrics to control and interrelate the execution of aforementioned behavioral primitives.

Chapter Eight

Conclusions

This chapter provides a summary of the work detailed in this dissertation, an outline of the specific contributions of the work, suggestions for possible extension of the work, and some concluding remarks.

8.1. Summary of Thesis

This thesis has addressed a fundamental problem in computational AI – developing a new class of massively parallel, computational neural learning algorithms for robustly, abstracting complex nonlinear transformations, e.g., functional, spatial, temporal and statistical invariants, from representative samples, in real-time. Provision of such a capability is at the core of many difficult problems in robotics, signal processing, remote sensing, control, etc. In contradistinction to existing dynamical neural learning formalisms our models, algorithms SID_1 through SID_6, encode information as singular, rather than regular, solutions to neurodynamics using the notion of terminal attractors. The infinite local stability resulting from violation of Lipschitz conditions, enables dramatic speedups during the learning process. Our notion of real-time pertains to operational responses that can be obtained in a few time constants of the individual neurons. In addition, we have addressed the issues of scalability and flexibility in neural networks. In this section we summarize the key results.

In Chapter Two we introduced a formal framework for deriving supervised learning algorithms for dynamical neural networks, both with and without feedback. By exploiting a recent breakthrough in nonlinear dynamical systems theory

- namely, the notion of terminal attractors, we defined neural learning formalisms, that are based on solutions of coupled singular differential equations. The model was appropriately denoted as Singularity Interaction Dynamics Model. In a departure from prior neuromorphic algorithms our methodology provided mechanisms for incorporating an in-training "skew" to handle network as well as design constraints. We showed how constraints could be augmented to the learning objectives using the method of lagrange multipliers. Optimization algorithms were then derived so as to strictly satisfy Lyapunov Stability criteria. The notion of "virtual attractors" was introduced to guarantee overall network stability. Extensive simulation results demonstrate that our model outperforms state-of-the-art back-propagation type neural learning formalisms by two to three orders of magnitude.

In Chapter Three we provided a novel manifestation to computational learning based on phenomenology of nonlinear neural networks. We presented a neural network model that allows adaptive evolution of network topology in addition to evolution of synaptic strengths. The former objective is achieved by taking recourse to Gauss's Least Constraint Principle in mechanics. This is a radical departure from existing connotations of learning. We further examined a fundamental limitation in neural learning algorithms - training and retraining costs and the versatility of neural network models. Motivated by results in "renormalization group theory and critical phenomena" in statistical quantum mechanics, we devised a methodology for "aposteriori regularization" in neural networks. This enables us to satisfy a multiplicity of event-driven constraints in real-time, without training the network each time we are faced with a new constraint. Further, in the previous sections, we had largely exploited the notion of terminal attractors to obtain speedup per learning iteration. In this chapter we showed how to speedup the entire learning process. This was achieved by devising an algorithm for adapting time scales in the terminal attractor formalism. These

constructs were used to rederive the neural learning formalisms. Our benchmarking results for signal processing problems indicated over two orders of magnitude improvement for learning hard nonlinearities. Also the algorithm was found to be over three orders of magnitude training sample stringent as compared to state-of-the-art feedforward neural learning formalisms. In addition, we provided insight on the role of numerical tools used in neural network simulations.

Todate the bulk of neural learning algorithms employed heuristics at some stage or the other. Our own work upto this stage, relied upon an efficient heuristic for inverting matrices, as proposed by Pineda [238,239]. In Chapter Four, we exploited a powerful tool for sensitivity analysis of nonlinear systems to put on a firm mathematical basis our results in computational learning. We provide a formal framework for global computation of sensitivities. In Chapter Five, we exploited the concept of *adjoint operators* to enable computation of changes in the network's response due to perturbations in all system parameters, using the solution of a single set of appropriately constructed linear equations. The lower bound on speedup per learning iteration over conventional methods for calculating the neuromorphic energy gradient is $O(N^2)$, where N is the number of neurons in the network. Our simulation results indicate over three orders of magnitude improvement in training sample stringency.

In Chapter Six we addressed another fundamental issue, which directly impacts the scalability of current theoretical neural network models to applicative embodiments, in both software as well as hardware - namely inherent and unavoidable concurrent asynchronicity of emerging fine-grained computational ensembles and the consequent chaotic manifestations in the absence of proper conditioning. All dynamical systems formulations to neural network models are strewn with parameters - decay constants, response gain, etc. Todate, however, their selection of such has remained largely heuristic, based on "anecdotal explorations".

We conduct a systematic analysis of the implications of these parameters on the neurodynamics, to illustrate that they could have dramatic implications on network scalability, convergence, throughput and fault tolerance, during both concurrent simulations and implementation, in concurrent VLSI, optical and optoelectronic hardware. For the first time we use dynamical diagnostics, Lyapunov exponents to formally characterize the widely observed dynamical instability in neural networks as "emergent computational chaos" and not broadband white noise. Using contracting operators and nonconstructive theorems in Fixed Point Theory, we rigorously derive the necessary and sufficient conditions for eliminating all oscillatory and throughput-limiting "emergent computational chaos". Neural algorithms are derived for conditioning Cohen-Grossberg-Hopfield (CGH) (additive-type) networks to operate under true "concurrent asynchrony". We demonstrated that our results are robust even in the presence of exceptionally large delays (¿ 2000 time constants). The validity of our results was demonstrated by simulated massively parallel networks ranging from few 100 synapses to over 100 million interconnects.

Finally, in Chapter Seven we provided insight for exploiting this powerful repertoire of adaptive neural learning formalisms to provide an enabling core for addressing a fundamental problem in robotics - the design of autonomous robots designed to perform tasks in unstructured and unpredictable environments. Using some recent results in task analysis and dynamic modeling, we propose a Perceptual Manipulation Architecture. The architecture, conceptualized within a "perceptual framework", is shown to be well beyond the state-of-the-art "model-directed" robotics. A technical critique on the proposed architecture is presented to juxtapose it with existing robot architectures. For a stronger physical interpretation of our implications to autonomous robotics, the discussions are embedded in context of a novel systems' concept - Robot-Assisted Extravehicular Activity,

for automated space operations.

8.2. Contributions

In this thesis we have presented an incremental framework for fast learning in dynamical neural networks. The primary contributions of this work are:

[1] In a significant departure from existing dynamical neural network formalisms we introduced an adiabatic learning model, wherein information was encoded as singular and not regular solutions to neurodynamics using terminal attractors. The infinite local stability resulting from this encodation strategy enabled immediate convergence to the fixed points during learning and operational phases, thereby providing dramatic speedup per learning iteration. The concept of terminal attractors was further exploited to define adaptive times scales which could be used to speedup the overall learning process.

[2] In the past, it has been demonstrated that complex information processing in the brain results from an adaptive and problem specific evolution of network topologies. In this thesis, we have demonstrated a framework for evolution of problem specific topologies. As a specific example, Gauss's Least Constraint Principle in Mechanics was exploited to adaptively derive networks with suboptimal topological configurations that favored locality of computation in addition to learning the nonlinear mapping.

[3] Our algorithm, SID_1, provides a computational framework to explicitly encode problem specific constraints during learning. We later showed this to be limited in that it "skewed" network behavior to capture only limited aspects of the mapping of interest. Each different runtime interest would require retraining the network. Thus, in order to avoid excessive training/retraining

computational cost, we exploited advances in "renormalization group theory" in physics to design a methodology for incorporating a multiplicity of constraints at run-time without retraining. Our "a posteriori regularization" formalism provides a unique way for modulating network response without disturbing the synaptic structure of the network.

[4] Most of the computational cost incurred during gradient-descent based, learning, is expended in computing the parametric sensitivities, i.e., dE/dp_μ (as discussed in Chapter Four and Five). All existing learning algorithms hitherto required, a system of N equations to be solved for each parameter p_μ, where N denotes the number of neurons in the network. We draw from results in Adjoint Sensitivity Theory, to obtain all derivatives by solving a single set of N appropriately constructed linear equations. The lower bound on speedup per learning iteration over conventional methods for calculating neuromorphic energy gradient is $O(N^2)$.

[5] All dynamical systems formulations neural network models are strewn with parameters - decay constants, response gain, etc. Todate, however, their selection had remained largely heuristic, based on "anecdotal explorations". For the first time we analyzed the effects of these parameters on the neurodynamics. We reasoned to discover that these parameters could have dramatic implications on network throughput, fault tolerance during both simulations and implementation in concurrent VLSI, optical and opto-electronic hardware. For the first time we use Lyapunov exponent measurements to formally characterize this instability as "emergent computational chaos" and not broadband white noise. We further developed algorithms based on contraction operators for rigorously deriving these parameters to eliminate all oscillatory and chaotic behavior. Our extensive empirical testing with arbitrarily large neural networks demonstrated robustness in the presence of

extremely large activation propagation time delays. Our results reflect a significant improvement on the parametric bounds derived by Marcus and Westervelt for network stability in the presence of activation signal propagation time delays.

On the applications front, we demonstrated the computational and paradigmatic strengths of our theoretical and algorithmic learning formalisms, by embedding them in an architectural framework for solving difficult problems in robotic manipulation systems [30,33-34,103,105,107-111,114-117,280-282], deterministic scheduling and load balancing [113,273], multi-target tracking [140] and multidimensional signal identification and characterization [32,35,104,105]. We defined a "perceptual approach" to designing such systems, demonstrating them to be well beyond the reach of "model-centered" robotics technology. We conjectured as to how the tools developed herein might form an enabling core for high performance task-dependent adaptive control of such systems. Our benchmarking results in signal processing problems and manipulator inverse kinematics problems have already shown a promise in that direction.

8.3. Future Directions

We see many future directions for this work, both in terms of extension on fundamental neural networks theory as well as in the development of advanced real-life applications in robotics, signal processing, remote sensing etc. Our approach in drawing heavily from rigorous results in dynamical systems theory, nonlinear mathematics, system science, statistical physics, etc., reflects our firm conviction that robustness in computational capabilities of neural networks can be derived only from the robustness and rigor of mathematical tools employed to design them. We have and continue to reject adhoc heuristics and brute-force arguments in our model derivation. As we continue to telescope years of biological

evolution into few years of neural net and AI research, we would however from here on like to emulate biological metaphors and principles rather closely. For example, in an attempt to enforce mathematical regularity and tractability in our models, we have resorted to many phenomena that are unbiological. For example, during biological learning, the brain is continually exposed to changing input patterns and has no opportunity to freeze them while waiting for the approach to equilibrium, i.e, learning is nonadiabatic. The recent work by Toomarian and Barhen on nonadiabatic learning of nonlinear mappings in [272] is a step in this direction. Also, the idea that learning can proceed by clamping the output of a system to a desired value while synaptic weights area adjusted according to some rule, violates biological reality. In fact, in our opinion in bulk of current artificial neural network research the only true resemblance to biological neural networks is on an abstract level, i.e., in terms of high processing element connectivity and massively parallelism. We would like to deviate and deals with systems whose fundamental characteristic is variability rather than statistical regularity (Reeke and Edelman in [93]). In specific terms, the following fundamental problems related to the work presented here, are of long-term scientific interest

[1] Systematic and rigorous analysis on how much is learned in the continuous state dynamical systems formulation ? When does unlearning set in ? In fact, for continuous systems an even more fundamental question remains unanswered - What is learnable ? Also, for continuous-state, continuous time neural networks, no insight currently exists for topographic partitioning, i.e., how to choose the hidden neurons.

[2] We would like to extend the constraint satisfaction framework, based on renormalization theory, to include inequality constraints. These are fundamental to problems in robotics, adaptive control of structures and signal processing. Recently [272], the framework has been extended to include

learning of time-dependent state-space trajectories. It would be useful to extend the constraint satisfaction framework into the latter formalism and benchmark capabilities with Jordan's [150] methodology for constrained trajectory learning.

[3] Another interesting problem is to analyze the synaptic weight space to correlate it with the physical system being learned. Currently, there is minimal correlation between the network dynamics and the dynamics of the physical system, thereby defying an understanding of the physics of the system being learned in the context of neural network internals, i.e. topology and weight space.

[4] In continuation of our development of a rigorous framework for autonomous robots along the lines discussed herein, two interrelated theoretical issues are of particular interest; namely understanding task stability and dynamics in the perceptual space, (i.e., from visual, force, and tactile data) and formal derivation of a necessary and sufficient set of task-specific perception primitives akin to control primitives. The latter issue is completely unaddressed to date.

8.4. Concluding Remarks

The work described in this dissertation demonstrates that it is possible to design extremely fast, versatile and robust neural learning algorithms, that can maintain performance even when the models are scaled to realistic size. These learning formalisms could form an enabling core for difficult problems in nonlinear adaptive control, object recognition and behavioral conditioning. While the mainstream of roboticists, continue to address issues relating to the kinematics, dynamics, design and control of complex mechanisms, which is in the domain of

engineering, we as Computer Scientists are working towards development of complex computational systems that will expand the capabilities of these mechanisms and inject rudimentary intelligence and autonomy in these systems. However, unlike AI researchers we emphasize "doing" intelligence rather than "reasoning" intelligence. While this research attempted to look at a problem that is very fundamental in this regard, namely learning complex nonlinear mappings from examples, there is much work left to do, both within the context of the high performance neural learning algorithms, complex concept representation, specially in the context of building intelligent machines that can perceive and manipulate objects with ease. The tools that we have developed as part of this research, will hopefully allow us to make a step in that direction.

Bibliography

[1] Abarbanel, H.D., J.B. Kadtke and R. Brown (1990). "Nonlinear Forecasting", *Phys. Rev.*, B41, 1782-1787.

[2] Ackley, D.H. (1987). *Stochastic Iterated Genetic Hill-Climbing*, Ph.D Thesis, CMU, Pittsburg, PA.

[3] Adrian, E.D. (1914). "The All-or-None Principle in Nerve", *J. Phys. Lond.*, XLVII, 6, 460-474.

[4] Aho, A.V., J.E. Hopcroft and J.D. Ullman (1974). *Design and Analysis of Computer Algorithms*, Reading, MA: Addison-Wesley.

[5] Akin, D.L. (1989). "Space Operations Testing of Telerobots in Neutral Buoyancy", *Proc. NASA Workshop on Space Telerobotics*, Pasadena, CA.

[6] Albus, J.S. (1971). "A Theory of Cerebellar Function", *Math. BioSc.*, 10, 25.

[7] Albus, J.S (1981). *Brains, Behavior and Robotics*, Peterborough, NH: Byte Books.

[8] Albus, J.S., H.G. McCain and R. Lumia (1987). "NASA/NBS Standard Reference Model for Telerobot Control System Architecture (NASREM)", NBS Tech. Note 1235.

[9] Aleksander, I. (Ed.) (1989). *Neural Computing Architectures :The Design of Brain Like Machines*, Cambridge, MA: The MIT Press.

[10] Alsmiller, R.G., J. Barhen and J. Horwedel (1984). "The Application of Adjoint Sensitivity Theory to a Liquid Fuels Supply Model", *Energy*, 9(3), 239-253.

[11] Amari, S.-I. (1972). "Characteristics of Random Nets of Analog Neuron-Like Elements", *IEEE Trans. Sys., Man, Cyber.*, SMC-2(5), 643-657.

[12] Amari, S.-I. (1979). "A Neural Theory of Association and Concept Formation", *Bio. Cyber.*, 26, 175-185.

[13] Amari, S.-I. (1983). "Field Theory of Self-organizing Neural Networks," *IEEE Trans. Sys., Man, Cyber.*, SMC-13(5), 741-748.

[14] Amari, S.-I. and M.A. Arbib (Eds.) (1982). *Competition and Cooperation in Neural Networks*, New York: Springer-Verlag.

[15] Amit, D.J. (1989). *Modeling Brain Function*, Cambridge: Cambridge University Press.

[16] Amit, D.J., H. Gutfreund and H. Sompolinsky (1985). "Storing Infinite Numbers of Patterns in a Spin-Glass Model of Neural Networks", *Phys. Rev. Lett.*, 55, 1530-1533.

[17] Anderson, J.A. (1988). *Neurocomputing*, Cambridge, MA: The MIT Press.

[18] Arbib, M.A. (1987). *Brains, Machines and Mathematics*, 2nd. Ed., New York, NY: Springer-Verlag.

[19] Arbib, M.A. (1989). *The Metaphorical Brain*, New York, NY: John Wiley.

[20] Arbib, M.A. and S.-I. Amari (Eds.) (1988). *Dynamic Interactions in Neural Networks: Models and Data*, New York, NY: Springer-Verlag.

[21] Arbib, M.A., T. Iberall, and D. Lyons (1985). "Coordinated Control Programs for Movements of the Hand," *Exp. Brain Res. Suppl.*, 1, 111-129.

[22] Asada H. and J.E. Slotine (1986). *Robot Analysis and Control*, New York, NY: John Wiley.

[23] Baillieul, J. (1986). "Kinematic Programming Alternatives for Redundant Manipulators ", *Proc. IEEE Int'l Conf. on Rob. and Auto.*, San Francisco, 1698-1704.

[24] Ballard, D.H. (1986). "Cortical Connections and Parallel Processing", *Behav. Brain Sc.*, 9, (1), 67.

[25] Banks, S.P. (1988). *Mathematical Theories of Nonlinear Systems*, Englewood Cliffs, NJ: Prentice Hall.

[26] Barhen, J. (1989). "Prediction of Aperiodic Phenomena", *Personal Communication.*

[27] Barhen, J., D.G. Cacuci and J.J. Wagschal (1982). "Uncertainty Analysis of Time-Dependent Nonlinear Systems", *Nucl. Sci. Eng.*, 81, 23-44.

[28] Barhen, J. and S.Gulati (1989). "Chaotic Relaxation Neurodynamics in Concurrently Asynchronous Networks", *Proc. Int'l Joint Conf. Neural Networks*, II, 619-627.

[29] Barhen, J. and S. Gulati (1990). "Computational Chaos in Massively Parallel Neural Systems", submitted to *Int'l Jour. of Neural Computing.*

[30] Barhen, J. and S. Gulati (1990). "Self-Organizing Neural Architecture for Inverse Kinematics of Redundant Manipulators", *NATO ASI*, F44 (in press).

[31] Barhen, J. and S. Gulati (1990). "Non-Lipschitzian Algorithms for Fast Learning", submitted to *Int'l Jou. of Neurocomputing.*

[32] Barhen, J., and J.F. Palmer (1986). "Hypercube in Robotics and Machine Intelligence", *Comp. Mech. Eng.*, 4(5), 30.

[33] Barhen, J., S. Gulati and N. Toomarian (1990). "Non-Lipschitzian Algorithms for Fast Learning", to appear in *IEEE Transactions on Neural Networks.*

[34] Barhen, J., S. Gulati and M. Zak (1988). "Neurodynamics of Redundant Robots: I. Manipulator Inverse Kinematics", *First Annual Int'l Neural Network Society Meeting*, Boston, MA.

[35] Barhen, J., S. Gulati and M. Zak (1988). "Real-Time Neuromorphic Algorithms for Inverse Kinematics of Redundant Manipulators," in *Proc. SPIE Symposium on Intelligent Robots*, Cambridge, MA, SPIE-1002, 686-696.

[36] Barhen, J., S. Gulati and M. Zak (1989). "Neural Learning of Constrained Nonlinear Transformations", *IEEE Computer*, 22(6), 67-76.

[37] Barhen, J., N. Toomarian and V. Protopopescu (1987). "Optimization of the Computational Load of a Hypercube Supercomputer Onboard a Mobile Robot," *Applied Optics*, 26, 5007-5014.

[38] Barhen, J., N. Toomarian and S. Gulati (1990). "Adjoint-Operator Algorithms for Learning in Neural Networks", *Proc. of Int'l Joint Conf. in Neural Networks*, Washington, D.C.

[39] Barhen, J., N. Toomarian and S. Gulati (1990). "Adjoint-Operator Algorithms for Learning in Neural Networks", *App. Math. Lett.*, 3(3), 13-18.

[40] Barhen, J., M. Zak and S. Gulati (1989). "Fast Neural Learning Algorithms Using Networks with Non-Lipschitzian Dynamics", in *Proc. Neuro-Nimes '89*, 55-68, EC2, Nanterre, France.

[41] Baron, R.J. (1987). *The Cerebral Computer*, Hillsdale, NJ: Lawrence Erlbaum Associates.

[42] Barto, A.G. and R.S. Sutton (1981). "Landmark Learning: An Illustration of Associative Search", *Bio. Cyber.*, 42, 1-8.

[43] Barto, A.G., R.S. Sutton and C.W. Anderson (1983). "Neuronlike Elements That Solve Difficult Control Problems", *IEEE Trans. Sys., Man, Cyber.*, 13, 834-846.

[44] Bastani, F.B., S. Gulati and S.S. Iyengar (1988). "Analysis of Competing Neural Network Knowledge Representation strategies," *First Annual Int'l Neural Network Society Meeting*, Boston, MA.

[45] Baudet G.M. (1983), "Asynchronous Iterative Methods for Multiprocessors," *Jour. of ACM*, 25(2), 226-244.

[46] Baum, E.B., J. Moody and F. Wilczek (1987). "Internal Representations for Associative Memory", *Inst. for Theor. Phys.*, University of California.

[47] Beltrami, E. (1987). *Mathematics for Dynamic Modeling* , New York, NY: Academic Press.

[48] Bertsekas, D.P. (1983). "Distributed Asynchronous Computation of Fixed Points", *Math. Prog.*, 27, 107-120.

[49] Blomfield, S. and D. Marr (1970). "How the Cerebellum May Be Used", *Nature 227,* 1224-1228.

[50] Brady, M., et al. (1983). *Robot Motion: Planning and Control*, Cambridge, MA: MIT Press.

[51] Brady, M. (1989) *Robotics Science*, Cambridge, MA: MIT Press.

[52] Braham, R., and J.O Hamblen (1988). "On the behavior of some Associative Neural Networks", *Biol. Cyber.*, 60, 145-151.

[53] Brindle A.F., W. Kohn, G.M. Lobdell and Albert J.H. (1988). "Robot Path Planning in Space" in *Proc. of 33rd International SAMPE Symp.*, Anaheim, CA, March 1988.

[54] Brooks, R.A. (1986). "A Robust Layered Control System for a Mobile Robot", *IEEE Jou. of Rob. and Aut.*, RA-2, 14-23.

[55] Brooks, R.A. (1989). "A Robot That Walks: Emergent Behaviors from a Carefully Evolved Network", *Neur. Comp.*, 1, 253-262.

[56] Bryson A.E. and Y. Ho (1975), *Applied Optimal Control*, New York, NY: Hemisphere Publishing Corporation.

[57] Bullock, D. and S. Grossberg (1989). *Neural Dynamics of Planned Arm Movements*, New York, NY: North-Holland.

[58] Burdick, J.W. (1988). " Kinematic Analysis and Design of Redundant Robot Manipulators," *Ph. D Thesis*, Dept. of Mech. Engg., Stanford University.

[59 Cacuci, D.G. (1981). "Sensitivity Theory for Nonlinear Systems: 1 Nonlinear Functional Analysis Approach", *Jou. Math. Physics*, 22(12), 2794-2802.

[60] Cacuci, D.G., C.F. Weber, E.M. Oblow and J.H. Marable (1980). "Sensitivity Theory for General Systems of Nonlinear Equations", *Nucl. Sci. Eng.*, 75, 88-110.

[61] Caianiello, E.R. (1961). "Outline of a Theory of Thought Processes and Thinking Machines", *Jou. Theor. Biol. I*, 204-235.

[62] Carpentar, G.A. and S. Grossberg (1987). "A Massively Parallel Architecture for a Self-Organizing Neural Pattern Recognition Machine", *Comp. Vis., Gra., and Image Proc.g*, 37, 54-115.

[63] Carpentar, G.A. and S. Grossberg (1987). "ART 2: Self-Organization of Stable Category Recognition Codes for Analog Input Patterns", *Applied Optics*, 26(23), 4919-4930.

[64] Casdagli, M. (1987). "Nonlinear Prediction of Chaotic Time Series", *Physica D*, 35, 335-356.

[65] Casulli, V. and D. Greenspan (1988). *Numerical Analysis for Applied Mathematics, Science and Engineering*, Redwood City, CA: Addison-Wesley Publishing.

[66] Chang, P.H. (1985). " A Closed Form Solution for the Control of Manipulators with Kinematic Redundancy ", *Proc. IEEE Int'l Conf. on Rob. and Auto.*, San Francisco, CA, 9-14.

[67] Chazan, D. and W. Miranker (1969). "Chaotic Relaxations," *Linear Algebra and its Applications*, 2, 199-222.

[68] Cheung, K.F., L.E. Atlas and R.J. Marks II (1987). "Synchronous vs Asynchronous Behavior of Hopfield's CAM," *Applied Optics*, 26(22), 4808-4014.

[69] Choi, M.Y. (1988). "Dynamic models of Neural Networks," *Phys. Rev. Lett.*, 61(24), 2809-2812.

[70] Cohen, M.A. and S. Grossberg (1983). "Absolute Stability of Global Pattern Formation and Parallel Memory Storage by Competitive Neural Networks", *IEEE Trans. Sys., Man and Cyber.*, SMC-13, 815-826.

[71] Coffman, E.G. (1976). *Computer and Job Shop Scheduling Theory* , New York, NY: John Wiley.

[72] Connell J.H. (1989). "A Behavior-Based Arm Controller", *IEEE Trans. Rob. and Auto.*, 5(6), 784-791.

[73] Cutkosky, M.R. (1985), *Robot Grasping and Fine Manipulation*, Boston, MA: Kluwer Academic Publishers.

[74] *DARPA Study on Neural Networks*, (1988), ASCEA International Press.

[75] Denker, J.S. (Ed.) (1986). *Neural Networks for Computing*, New York: American Institute of Physics, 151.

[76] Denning, P.J. (1988). "Blindness in Designing Intelligent Systems", *American Scientist*, 76, 118-120.

[77] Dreyfus, S.E. (1987). "Neural Nets: An Alternative Approach to AI", *Applied AI Reporter*, 6-15.

[78] Dubey, R.V., J.A. Euler and S.M. Babcock (1988). "An Efficient Gradient Projection Optimization Scheme for a 7-DOF Redundant Robot with a Spherical Wrist ", in *Proc. IEEE Int'l. Conf. on Rob. and Aut.*, Philadelphia, I, 28-36.

[79] Dubowsky, S. and D.T. Desforges (1981). "The Application of Model Reference Adaptive Control of Robot Manipulators", *Trans. ASME Jour. of DSMC*, 101, 193 - 200.

[80] Eckmiller, R. (Eds.) (1990). *Advanced Neural Computers*, Amsterdam: North Holland.

[81] Erickson, J., et al. (1988). "An Intelligent Free Flying Robot", *Proc. 1988 SPIE Symp. in Int. Rob. Sys., Space Station Automation IV*, Cambridge, MA.

[82] Eykhoff, P. (1974). *System Identification*, New York, NY: John Wiley.

[83] Farmer, J.D. and J.J. Sidorowich (1988). "Predicting Chaotic Time Series", *Phys. Rev. Lett.*, 59, 845-848.

[84] Feldman, J.A and D.H. Ballard (1982). "Connectionist Models and their Properties", *Cog. Sci.*, 6, 205-254.

[85] Firby R.J. (1989). "Adaptive Execution in Complex Dynamic Worlds", *YALEU/CSD/RR #672, Yale Tech. Rep.*.

[86] Forbus K.D. (1984). "Qualitative Process Theory" *Art. Int. Jour.*, 24, 85-168.

[87] Fox, G. and W. Furmanski (1989). "Load Balancing by a Neural Network", *Jour. of Supercomputing.*

[88] Fukushima, K. (1987). "Neural Network Model for Selective Attention in Visual Pattern Recognition and Associative Recall", *Applied Optics*, 26(23), 4984-4992.

[89] Garey, M.R. and D.S. Johnson (1979). *Computers and Intractability: a Guide to the Theory of NP-Completeness*, San Francisco, CA: W.H. Freeman.

[90] Gemen, S. and Gemen, D. (1987). "Stochastic Relaxation, Gibbs Distribution, and the Bayesian Restoration of Images", Readings in Computer Vision, Palo Alto, CA: Morgan Kaufmannn Publishers, Inc., 564-584.

[91] Golden, R.M. (1986). "The 'Brain-state-in-a-box' Neural Model is a Gradient Descent Algorithm", *Jour. of Math. Psy.*, 30, 73-80.

[92] Goldenberg, A.A. (1985). "A Complete Generalized Solution to the Inverse Kinematics of Robots," *IEEE Jour. of Rob. and Aut.*, RA-1(1), 14-20.

[93] Graubard, S.R. (1989). *The Artificial Intelligence Debate: False Starts, Real Foundation*, Cambridge, MA: The MIT Press.

[94] Grossberg, S. (1973). "Computer Enhancement, Short Term Memory, and Constancies in Reverberating Neural Networks", *Studies in Appl. Math.* LII(3), 213-257.

[95] Grossberg, S. (1982). *Studies of Mind and Brain: Neural Principles of Learning, Perception, Development, Cognition, and Motor Control*, Amsterdam: Reidel Press.

[96] Grossberg, S. (1987a). *The Adaptive Brain, I: Cognition, Learning , Reinforcement and Rhythm*, Amsterdam: Elsevier/North-Holland.

[97] Grossberg, S. (1987b). *The Adaptive Brain, II: Vision, Speech, Language and Motor-Control*, Amsterdam: Elsevier/North-Holland.

[98] Grossberg, S. (1988). "Nonlinear Neural Networks: Principles, Mechanisms, and Architectures," *Neural Networks*, 1(1), 17-62.

[99] Grossberg S. and Kuperstein M. (1986). *Neural Dynamics of Adaptive Sensory-motor Control Ballistic Eye Movements*, Amsterdam: Elseiver.

[100] Guckenheimer J. and P. Holmes (1983). *Nonlinear Oscillations, Dynamical Systems, and Bifurcations of Vector Fields*, Berlin: Springer-Verlag.

[101] Giles, C.L. and T. Maxwell (1987), "Learning, Invariance and Generalization in High-Order Neural Networks", *Applied Optics*, 26(23).

[102] Guez A. (1988), "Solution to the Inverse Kinematics Problem in Robotics by Neural Networks," in *Proc. of 2nd Int'l Conf. on Neural Networks*, San Diego, CA, 2, 617-624.

[103] Gulati, S. (1990). "Perceptual Manipulation Systems in Manufacturing", *SME Workshop on Neural Networks*, Detroit, MI, Apr. 3-4.

[104] Gulati, S. and J. Barhen (1989). "Predictive Learning Algorithms for Multi-dimensional Signal Reconstruction", *Third Parallel Processing Symposium*, Fullerton, CA.

[105] Gulati, S. and J. Barhen (1989). "Fast Neural Learning Algorithms for Robot Control in Unstructured Environments", *23 Asilomar Conference on Sig., Sys. and Comp.*, Oct 29-31, Pacific Grove, CA.

[106] Gulati, S. and J. Barhen (1990). "Computational Chaos in Massively Parallel Neural Systems", presented at *Fourth Parallel Processing Symposium*, April 4-6, Fullerton, CA.

[107] Gulati, S., J. Barhen and S.S. Iyengar (1990). "Computational Neural Learning Formalisms for Manipulator Inverse Kinematics" submitted to *IEEE Trans. on Sys., Man, and Cyber.*.

[108] Gulati, S., J. Barhen and S.S. Iyengar (1989). "Self-Organizing Neural Formalisms for Manipulator Inverse Kinematics", *ORSA/TIMS Conference*, New York, NY.

[109] Gulati, S., J. Barhen, S.S. Iyengar (1988). "The Pebble Crunching Model for Load Balancing in Concurrent Hypercube Ensembles," *Proc. Third Conference on Hypercube Concurrent Computers and Applications*, 1, Pasadena, CA, 188-198.

[110] Gulati, S., J. Barhen and S.S. Iyengar (1989) "Computational Learning Formalisms for Manipulator Inverse Kinematics", *Proc. of NASA Workshop on Space Telerobotics*, Pasadena, CA, I, 69-76.

[111] Gulati, S., J. Barhen and S.S. Iyengar (1991). "Neurocomputing Formalisms for Learning and Machine Intelligence", *Advances in Computer*, (Ed.) M.C. Yovits, New York, NY: Academic Press (in press).

[112] Gulati, S., S.S. Iyengar and J. Barhen (1990). "The Pebble Crunching Model for Fault Tolerant Load Balancing in Hypercube Ensembles", *Computer Journal*. 33(3), 204-213.

[113] Gulati, S., S.S. Iyengar, N. Toomarian, V. Protopopescu, and J. Barhen (1987). "Nonlinear Neural Networks for Deterministic Scheduling", *IEEE First Int. Conf. on Neural Networks*, IV, 745-752.

[114] Gulati, S. and S.T. Venkataraman (1990a). "Perceptual Manipulation Systems", *JPL Engineering Memo. 3462-90-001*, 1990.

[115] Gulati, S. and S.T. Venkataraman (1990b). "Control Issues Related to Robotic Astronaut Assistance", *Fifth IEEE Int'l Symp. on Int. Control*, Philadelphia, PA, Sep. 5-7.

[116] Gulati, S. and S.T. Venkataraman (1990c). "Robot-Assisted Extravehicular Activity: Part I - Perceptual Manipulation Architecture", in *SPIE Conference on Cooperative Intelligent Robots in Space*, Boston, MA, Nov. 6-11.

[117] Gulati, S. and S.T. Venkataraman (1990d). "Perceptual Robotics: A Vehicle for Synergistic Man-Machine Systems", *IEEE Conf. on Bio. Eng.*, Philadelphia, PA, Nov. 1990. To be submitted to *Machine Intelligence*.

[118] Haken, H. (Ed.) (1988). *Neural and Synergetic Computers*, Heidelberg: Springer-Verlag.

[119] Haussler, D. and L. Pitt (Eds.), (1988). *Proc. of the 1988 Workshop on Computational Learning Theory*, Palo Alto, CA: Morgan Kauffman Publishers.

[120] Hebb, D.O. (1949). "Organization of Behavior", New York, NY: John Wiley.

[121] Hecht-Nielson R., "Neurocomputer Applications in Performance Limits of Optical, Electro-Optical and Electronic Artificial Neural System Processors" in Proc. SPIE, 634, 277.

[122] Hemami A. (1987). "On a Human-Arm Like Mechanical Manipulator", *Robotica*, 5, 23-28.

[123] Henderson, T. (1985). "Logical Sensor Systems", *Jour. of Rob. Sys.*, 1(2), 169-193, 1985.

[124] Hinton, G.E. (1984). "Distributed Representations", *Tech. Report, CMU-CS-84-157*, Dept. of Computer Science, CMU, Pittsburgh.

[125] Hinton, G.E. (1987). "Connectionist Learning Procedures", *Tech. Rep. CMU-CS-87-115*, Comp. Sci. Dept., CMU, Pittsburg, PA.

[126] Hinton, G. and J.A. Anderson (1981). *Parallel Models of Associative Memory*, Hillsdale, NJ: Erlbaum.

[127] Hirsch, M.W. (1987). "Systems of Differential Equations that are Competitive or Cooperative, I: Limit Sets", *SIAM Journal of Math. Ana.*, 13, 167-179.

[128] Hirsch, M.W. (1987). "Systems of Differential Equations that are Competitive or Cooperative, II: Convergence Almost Everywhere", *SIAM Journal of Math. Ana.*, 13, 423-439.

[129] Hogan, N. (1985). "Impedance Control: An Approach to Manipulation: Part I - Theory, Part II - Implementation, Part III - Applications", *Jour. of Dyn., Sys., Meas. and Cont.*, 1-24.

[130] Holland, J.H. (1975). *Adaptation in Natural and Artificial Systems*, Ann Arbor, MI: University of Michigan Press.

[131] Hollerbach, J.M. and K.C. Suh (1985). " Redundancy Resolution of Manipulators through Torque Optimization," in *Proc. of IEEE Int'l Conf. on Rob. and Auto.*, St. Louis, 1016-1021.

[132] Hodgkin, A.L. and Huxley, A.F. (1952). "A Quantitative Description of Membrane Current and its Application to Conduction and Excitation in Nerve", *Jour. Phys. Lond.*, 117, 500-544.

[133] Hofstader, D.R. (1979). *Godel, Escher, and Bach*, New York, NY: Bantom Books.

[134] Hopfield, J.J. (1982). "Neural Networks and Physical Systems with Emergent Collective Computational Abilities", *Proc. Nat'l. Acad. Sci.* 79, 2554-2558.

[135] Hopfield, J.J. (1984). "Neurons with Graded Response have Collective Computational Properties Like Those of Two-State Neurons," *Proc. of Nat'l Acad. Sci.*, 81, 3058-3092.

[136] Hopfield, J.J. and Tank, D.W. (1985). "Neural Computation and Constraint Satisfaction Problems and the traveling Salesman", *Biol. Cyber.*, 52, 141-152.

[137] Ito, M., (1984), "The Cerebellum and Neural Control", New York: Raven.

[138] Ito, M. (Ed.) (1989). *Neural Programming*, Basel: Karger.

[139] Ish-Shalom, J. (1985). "The CS Language Concept: A New Approach to Robot Motion Design", *Int'l Jour. of Rob. Res.*, 4(1), 42-58.

[140] Iyengar, S.S., J. Barhen, S. Gulati, R.L. Kashyap and R.N. Madan (1988). "A Neuromorphic Architecture for Design of Tracking Filters," *1988 Conf. on Adv. Comm. Cont.*, Baton Rouge, LA.

[142] Iyengar, S.S. and A. Moitra (1987). "Parallel Algorithms for a Class of Computational Problems", *Advances in Computers*, (Ed. M.C. Yovits), 26, 93-153.

[143] Iyengar, S.S., C. Jorgenson, N.S.V. Rao and C.R. Weisbin (1986). " Robot Navigation Algorithms Using Learned Spatial Graphs", *Robotica*, 4, 93-100.

[144] Iyengar, S.S. and R.L. Kashyap (1989). "Autonomous Intelligent Machines: An Introduction", *IEEE Computer*, 22(6), 14-15.

[145] Iyengar, S.S. and R.L. Kashyap (1991). "Neural Networks: A Computational Perspective", *Proc. 1991 Genoa Summer School*, Genoa, Italy, Sept.25-29 (in press).

[146] Iyengar, S.S. and J. Oomman (1990). "Path Planning of Robot Manipulators in Noisy Workspaces", in *Proc. Fifth IEEE Int'l Int. Cont. Sym.*, Philadelphia, PA, Sep. 5-7.

[147] Iyengar, S.S., N.S.V. Rao, R.L. Kashyap and V.K. Vaishnavi (1988). "Multidimensional Data Structures: Review and Outlook", *Advances in Computers*, 27, Academic Press Publications.

[148] Iyengar, S.S., A.S. Sabharwal, F.G. Pin and C.R. Weisbin, (1990). "Asynchronous Production Systems for Control of an Autonomous Mobile Robot in Real-time Environment", *App. of Art. Int. Jour.*, (in press).

[149] Jeffrey, W. and R. Rosner (1986). "Optimization Algorithms: Simulated Annealing and Neural Network Processing", *The Astrophysical Journal*, 310, 473-481.

[150] Jordan I.M. (1988). "Supervised Learning and Systems with Excess Degrees of Freedom", *COINS Tech. Rep. 88-27*, MIT.

[151] Jorgensen, C. and C. Matheus (1986). "Catching Knowledge in Neural Nets", *AI Expert*, 86, 31-40.

[152] Josin G. (1988), "Neural-Space Generalization of a Topological Transformation", *Biological Cybernetics*, 59, 283-290.

[153] Stinson, M.C. (1988). "Neural Networks with Asynchronous Control", *Ph.D Dissertation*, Louisiana State Univ., Baton Rouge.

[154] Kaizerman, S., R.G. Fenton, B. Benhabib and G. Zak (1990). "Application of Deterministic Sensitivity Analysis Methods to Industrial Robot Calibration Procedures", *Proc. 1990 Pacific Conference on Manu.*, Australia.

[155] Kanal, L. and Tsao T. (1986). "Artificial Intelligence and Natural Perception", *Proc. of Intelligent Autonomous Systems*, Amsterdam, 60-70.

[156] Kawato, M., K. Fusukawa and R. Suzuki (1987). "A Hierarchical Neural-Network Model for Control and Learning of Voluntary Movement", *Bio. Cyber.*, 57, 169-185.

[157] Kawato M., M. Isobe, Y. Maeda and H. Suzuki (1988). "Coordinates Transformation and Learning Control for Visually-Guided Voluntary Movement with Iteration: A Newton-Like Method in a Function Space", *Biol. Cyber.*, 59(3), 161-178.

[158] Kazerooni, H. (1985). "A Robust Design Method for Impedance Control of Constrained Dynamic Systems", *Ph.D Thesis*, MIT, Cambridge, Massachusetts, 1985.

[159] Keeler, J.D. (1986). " Basins of Attraction of Neural Network Models ", *Proc. of AIP Conference* , 151, Neural Networks for Computing, Snowbird, UT, 259-265.

[160] Khatib, O. (1986). "A Unified Approach for Motion and Force Control of Robot Manipulators: The Operational Space Formulation", *IEEE Jou. of Rob. and Aut.*, RA-3, 43-53.

[161] Khatib, O. (1986). "Real-time Obstacle Avoidance for Robot Manipulators and Mobile Robots", *Int'l Jou. of Rob. Res.*, 5(1), 90 - 98.

[162] Kirkpatrick, S., D.D. Gelatt and M.P. Vecchi (1983). "Optimization by Simulated Annealing", *Science*, 220, 671-680.

[163] Klein, C.A. and C.H. Huang (1983). " Review of Pseudo-Inverse Control for use with Kinematically Redundant Manipulators ", *IEEE Trans. Sys., Man and Cyber.*, SMC-13(2), 245-250.

[164] Koditschek D.E. (1987). "Adaptive techniques for Mechanical Systems", *Proc. Fifth Yale workshop on Appl. of Adaptive Sys. Theory*, New Haven, CT, 259-265.

[165] Koditschek D.E. (1987). "Exact Robot Navigation by Means of Potential Functions: Some Topological Considerations", *Int'l Conf. on Rob. and Aut.*, Raleigh, NC, 1-6.

[166] Kohonen, T. (1977). *Associative Memory: System Theoretic Approach*, Berlin: Springer-Verlag.

[167] Kohonen, T. (1982). "Self-Organized Formation of Topologically Correct Feature Maps", *Biol. Cybern, 43*, 59-70.

[168] Kohonen, T. (1984). *Self-Organization and Associative Memory*, Berlin: Springer-Verlag.

[169] Kohonen, T. (1987). "Adaptive, Associative, and Self-Organizing Functions in Neural Computing", *Applied Optics*, 26(23), 4910-4919.

[170] Kohonen, T. (1988), "An Introduction to Neural Computing", *Neural Networks*, 1(1), 3-16.

[171] Kosko, B. (1987). "Adaptive Bidirectional Associative memories", *Applied Optics*, 26(23), 4947-4960.

[172] Kung, H.T. (1976). "Synchronized and Asynchronized Parallel Algorithms for Multiprocessors," *Algorithms and Complexity: New Directions and Results*, J.F. Traub, (Ed.), New York, NY: Academic Press, 428-464.

[173] Kuperstein, M. (1988). " Neural Network Model for Adaptive Hand-Eye Coordination for Singel Postures", *Science*, 239, 1308-1311.

[174] Kuperstein, M. (1988). "Visual-Motor Coordination of Multijoint Robots Using Parallel Architectures", *Proc. 1988 IEEE International Conference on Robotics and Automation*, Raleigh, NC, 1592-1598.

[175] Ladd, S. (1985). "The Computer and the Brain: Beyond the Fifth Generation", New York: Bantom Books.

[176] Laird, J. (Ed.) (1988). *Proc. Fifth Int'l Conf. on Machine Learning*, Palo Alto, CA: Morgan Kaufmann Publishers.

[177] Lapedes, A. and R. Farber (1986). "A Self-Optimizing, Nonsymmetrical Neural Net for Content Addressable Memory and Pattern Recognition", *Physica D, 22*, 247.

[178] Lapedes, A. and Farber, R. (1987). "Nonlinear Signal Processing using Neural Networks: Predictions and Systems Modeling", *LA-UR87-2662*, Los Alamos National Laboratory.

[179] Lashley, K.S. (1950). "In Search of the Engram", *Symp. Soc. Exp. Biology, 4*, 454.

[180] Le Cun, Y. (1985). "A Learning Scheme for Asymmetric Threshold Networks", *Proc. Congitiva 85*, 599-607.

[181] Le Cun, Y., J.S. Denker and S.A. Solla (1990). "Back-Propagation Applied to Handwritten Zipcode Recognition", *Neu. Comp.*, 4.

[182] Leibniz, J. (1951). "Selections", Philip Wiener (Ed.), New York, NY: Scribner.

[183] Li, Z. and S. Sastry (1986). "Task Oriented Optimal Grasping by Multifingered Robot Hands" *Memo UCB/ERL M86/43*, Electronics Research Laboratory, UC at Berkeley.

[184] Liegeois, A. (1977). "Automatic Supervisory Control of the Configuration and Behavior of Multibody Mechanisms", *IEEE Trans. Sys., Man and Cyber.*, SMC-7Z(12), 868-871.

[185] Linsker, R. (1986). "From Basic Network Principles to Neural Architecture: Emergence of Spatial-Opponent Cells", *Proc. Nat'l. Acad. Sci.*, 3, 7508-7512.

[186] Linsker, R. (1986). "From Basic Network Principles to Neural Architecture: Emergence of Orientation-Selective Cells", *Proc. Nat'l. Acad. Sci.*, 83, 8390-8394.

[187] Linsker, R. (1986). "From Basic Network Principles to Neural Architecture: Emergence of Orientation Columns", *Proc. Natn. Acad. Sci.*, 83, 8779-8783.

[188] Lippmann, R.P. (1987). "An Introduction to Computing with Neural Nets", *IEEE ASSP Magazine*, 4(2), 4.

[189] Lippmann, R.P. and P. Beckman (1988). "Adaptive Neural Net Preprocessing for Signal Detection in Non-Gaussian Noise", D.S. Touretzky, ed., *Neural Information Processing Systems*, New York, NY: Morgan Kaufmann, 124-132.

[190] Little, W.A. (1974). *The Origin of the Alpha Rhythm*, Edinburgh, London: Churchill Livingstone.

[191] Little, W.A. and G.L. Shaw (1975). "A Statistical Theory of Short and Long-Term Memory", *Beh. Biol. 14*, 115-133.

[192] Llinas, R.R., (1988). *The Biology of the Brain: From Neurons to Networks*, New York, NY: W.H. Freeman and Co.

[193] Lotka, A.J. (1956). *Elements of Mathematical Biology*, New York, NY: Dover.

[194] Lozano-Perez, T., M.T. Mason and R.H. Taylor (1984). "Automatic Synthesis of Fine Motion Strategies for Robots", *Int'l Jou. of Rob. Res.*, 3(1), 3-24.

[195] Lozano-Perez, T. et al. (1987). "Handey: A Robot System that Recognizes, Plans, Manipulates" *Proc 1985 IEEE Int'l Conf. on Rob. and Aut.*, Mar. 31 - Apr. 3, Raleigh, NC.

[196] Lynch, G. (1990). *Synapses, Circuits, and the Beginnings of Memory*, Cambridge, MA: The MIT Press.

[197] Maciejewski, A.A. and C.A. Klein, (1987). "Obstacle Avoidance for Kinematically Redundant Manipulators in Dynamically Varying Environments", *Int'l Jour. on Rob. Res.*, 4(3), 109-117.

[198] Macukow, B. and H.S. Arsenault (1987). "Modification of the threshold condition for a content-addressable memory based on the Hopfield Model," *Applied Optics*, 26(1), 34-36.

[199] Malsburg, C. (1985). "Self-Organization of Orientation Sensitive Cells in the Striate Cortex", *Bunsenges. Phys. Chem 89,* 703-710.

[200] Marcus, C.M. and R.M. Westervelt (1988). "Dynamics of Analog Neural Networks with Time Delay", *Advances in Neural Information Processing*, (Ed.) D.S. Touretzky, I, 568-576.

[201] Marcus, C.M. and R.M. Westervelt (1989). " Stability of Analog Neural Networks with Delay," *Phys. Rev. A*, 39, 347-341.

[202] Marr, D. (1969). "A Theory of Cerebellar Cortex", *Jour. Phys. of London, 202*, 437.

[203] Mason, M.T. (1982). "Compliance and Force Control for Computer Controlled Manipulators", *Robot Mation: Planning and Control* , Brady, M. et al. (Eds.), Cambridge, MA: The MIT Press.

[204] Mason, M.T. and K.J. Salisbury (1985). *Robot Hands and the Mechanics of Manipulation*, Cambridge, MA: The MIT Press.

[205] McCorduck, P. (1979). *Machines Who Think*, New York, NY: W.H. Freeman and Co.

[206] McCulloch, W.S. and Pitts, W.H. (1943). "A Logical Calculus of the Ideas Immanent in Nervous Activity," *Bulletin of Mathematical Biophysics, 5*,115.

[207] McEliece, R.J., E.C. Posner, E.R. Rodemich, and S.S. Venkatesh (1987). "The Capacity of the Hopfield Associative Memory", *IEEE Trans. Inf. Theory,* I, 33-45.

[208] McLeisch, M. (Ed.) (1988). "Taking Issue/Forum: An Inquiry into Computer Understanding", *Computer Intelligence*, 4.

[209] Mead, C. (1989). *Analog VLSI and Neural Systems*, Reading, MA: Addison-Wesley.

[210] Meystel, A. (1988). "Intelligent Control in Robotics", *Jour. of Rob. Sys.*, 5(4), 269-308.

[211] Michalski, R., J. Carbonell and T. Mitchell (1982). *Machine learning: An Artificial Intelligence Approach*, Vol I and II, Palo Alto, CA: Morgan Kaufmann.

[212] Miellou, J.C. (1986). "Asynchronous Iterations and Order Intervals," *Parallel Algorithms & Architectures*, (Eds. M. Cosnard el al.), Amsterdam, Holland: Elsevier Science Publishers.

[213] Minsky, M. (1967). *Computation: Finite and Infinte Machines*, New York, NY: Prentice-Hall.

[214] Minsky, M. (1986). *Society of Mind*, New York, NY: Simon and Schuster.

[215] Minsky, M. and S. Papert (1969). *Perceptrons: An Introduction to Computational Geometry*, Cambridge, MA: The MIT Press.

[216] Mjolsness, E. (1987). "Control of Attention of Neural Networks", Yale University, Tech. Report.

[217] Montemerlo, M.D. (1988). "The Space Perspective: Man-Machine Redundancy in Remote Manipulator Systems", in Proc. *NATO Advanced Research Workshop on Robots with Redundancy: Design, Sensing and Control*, Salo, Lago di Garda, Italy.

[218] Nadel, L., L.A. Cooper, P. Culicover and R.M. Harnish (1989). *Neural Connections, Mental Computation*, Cambridge, MA: The MIT Press.

[219] Narendra, K.S. and A.M. Annaswamy (1989).*Stable Adaptive Systems*, Englewood Cliffs, NJ: Prentice-Hall.

[220] Narendra, K.S. and Parthasarthy K., "Identification and Control of Dynamical Systems Using Neural Networks", *IEEE Trans. Neural Networks*, 1(1), 1990, 1-27.

[221] Newell, A. and H. Simon (1976). "Computer Science as Empirical Inquiry: Symbols and Search," *Comm. ACM.*

[222] North, G. (1987). "A Celebration of Connectionism", *Nature, 328*, 107.

[223] Oblow, E.M. (1977). "Sensitivity Theory for General Non-Linear Algebraic Equations with Constraints", *ORNL/TM-5815*, Oak Ridge National Laboratory.

[224] Oh, S., L.E. Atlas, R.J. Marks II and D.C. Park (1988). "Effects of Clock Skew in Iterative Neural Network and Optical Processors," *Proc. of 1988 IEEE Int'l Conf. on Neural Networks*, San Diego, CA, II, 429-435.

[225] Omohundro, S.M. (1987). "Efficient Algorithms with Neural Network Behavior", *Complex Systems*, 273-347.

[226] Ortega, J.M. and W.C. Rheinboldt (1970). *Iterative Solutions of Nonlinear Equations in Several Variables*, New York, NY: Academic Press.

[227] Osherson, D., M. Stob and S. Weinstein (1986). *Systems That Learn*, Cambridge, MA: MIT press.

[228] Parisi, G. (1986). "Asymmetric Neural Networks and the Process of Learning", *Bio. Cybern.*, 50, 51-62.

[229] Parisi, G. (1988). *Statistical Field Theory*, Frontiers in Physics Series, 66, Redwood City, CA: Addison-Wesley Publishing Co.

[230] Parker, D.B. (1985). "Learning-Logic", *TR-47*, MIT Tech. Report.

[231] Parker, D.B. (1986). "A Comparison of Algorithms for Neuron-like Cells", In: *Neural Networks for Computing* (ed. J.S. Denker). *Proc. AIP Conf 151.*

[232] Peretto, P. (1984). "Collective Properties of Neural Networks: A Statistical Physics Approach", *Biol. Cybern.*, 50, 51-62.

[233] Paul, R.P. (1981) *Robot Manipulators : Mathematics, Programming and Control* , Cambridge, MA: The MIT Press.

[234] Pearlmutter, B.A. (1989). "Learning State Space Trajectories in Recurrent Neural Networks", *Neural Computation*, 1(3), 263-269.

[235] Pellionisz, A.J. (1986). "Tensor Network Theory of the Central Nervous System and Sensorimotor Modeling", Eds. G. Palm and A. Aertsen, *Brain Theory*, Berlin: Springer Verlag.

[236] Pellionisz, A.J. and R.R. Llinas (1985). "Tensor Network Theory of the Metaorganization of Functional Geometries in the CNS", *Neurosci.*, 16, 245-274.

[237] Percival, I. and D. Richards (1987). *Introduction to Dynamics*, Cambridge, MA: University Press.

[238] Pineda, F.J. (1987). "Generalization of Back-Propagation to Recurrent Neural Networks," *Phys. Rev. Lett.*, 59(19), 2229-2232.

[239] Pineda, F.J. (1988). "Dynamics and Architecture in Neural Computation", *Journal of Complexity*, 4, 216-245.

[240] Platt, J.C. and Barr, A.H. (1987). "Constrained Differential Optimization", *Proc. of IEEE 1987 NIPS Conf.*, , Denver.

[241] Press, W.H. and S.A. Teukolsky (1989). "Integrating Stiff Ordinary Differential Equations", *Computers in Physics*, 3, 88-91.

[242] Psaltis, D. and N.H. Farhat (1985). "Optical Information Processing Based on an Associative-Memory Model of Neural Networks with Thresholding and Feedback," *Optical Letters*, 10, 98-100.

[243] Orland, H. (1985). "Mean Field Theory for Optimization Problems", *Jour. of Phys., Paris*, 46, L673.

[244] Raibert, M.H. and J.J. Craig (1981). "Hybrid Position/Force Control of Manipulators", *ASME Jour. Dyn., Sys., Meas. and Cont.*, 102.

[245] Ritter, H., T. Martinetz and K. Schulten (1989). "Topology Conserving Maps for Learning Visuo-Motor Coordination", *Neural Networks*, 2, 159-168.

[246] Rivest, R., D. Haussler and M.K. Warmuth, (Eds.) (1989). *Proc. of the Second Annual Workshop on Computational Learning Theory*, Morgan Kauffman Publishers, Palo Alto, CA.

[247] H. Rogers (1987). *Theory of Recursive Functions and Effective Computability*, Cambridge, MA: The MIT Press.

[248] Rosenblatt, F. (1962). *Principles of Neurodynamics, Perceptrons and the Theory of Brain Mechanisms*. Washington, D.C: Spartan Books.

[249] Rosenschein S.J. and S. Kaelbling (1988). "Integrating Planning and Reactive Control", *Teleos Research Tech. Report.* TR-88-43-0213.

[250] Rumelhart, D.E., Hinton, G.E. and Williams R.J., "Learning Internal Representations by Back-Propagating Errors", *Nature, 323*, 533-536.

[251] Rumelhart, D.E., McClelland, J.L. and the PDP Research Group (1986). *Parallel Distributed Processing*, Vol. I and II, Cambridge, MA: The MIT Press.

[252] Rumelhart, D.E. and D.A. Norman (1982). "Simulating a Skilled Typist: A Study of Skilled Cognitive-Motor Performance, *Cognitive Science*, 6, 1-36.

[253] Saridis, G.N., "Knowledge Implementations: Structures of Intelligent Control", *Jour. of Rob. Sys.*, 5(4), 1988, 255-268.

[254] Sejnowski, T.J. and C.R. Rosenberg (1986). "NETtalk: A Parallel Network that Learns to Read Aloud", *Tech. Rep. JHU/EECS-86/01*, Johns Hopkins Univ., Baltimore, MD.

[255] Seraji H. (1987). "Direct Adaptive Control of Manipulators in Cartesian Space", *Jour. of Rob. Sys.*, 4(1), 157-178.

[256] Seraji H. (1989). "Configuration Control of Redundant Manipulators: Theory and Implementation", *IEEE Trans. Rob. and Auto.*, 5(4), 1989, 472-490.

[257] Shamir, J. (1987). "Fundamental Speed Limitations on Parallel Processing," *Applied Optics*, (26), 1567-1568.

[258] Sheridan, T.B. (1987). "Telerobotics", in *Proc. 10th IFAC World Congress on Automatic Control*, Munich, Germany.

[259] Shimada, I. and T. Nagashima (1979). "A Numerical Approach To Ergodic Problem of Dissipative Dynamical Systems, " *Progress of Theoretical Physics*, 61(6), 1605-1616.

[260] Slotine, J.J. (1988). "Adaptive Control A Case Study", *Proc. 1988 IEEE Int'l Conf. on Rob. and Auto.*, Raleigh, NC, 1392 - 1398.

[261] Smieja, F.J. and G.D. Richards (1988). "Hard Learning the Easy Way: Backpropagation with Deformation", *Complex Systems*, 2, 671-704.

[262] Spong M.W. and Vidyasagar M. (1989). *Robot Dynamics and Control*, New York, NY: John Wiley and Sons.

[263] Somoano, R.B (1988). "Applications Analysis Study for the Rapid Prototyping of a Neural Processor", *JPL Report D-8553.*

[264] Soucek, B. and M. Soucek (1988). *Neural and Massively Parallel Computers*, New York, NY: John-Wiley and Sons.

[265] Szu, H. (1986). "Fast Simulated Annealing", *AIP Conference Proceedings of Physics* , Ed. J.S. Denker, New York, NY.

[266] Sutton, R.S. (1984). "Temporal Credit Assignment in Reinforcement Learning", *Ph.D Thesis*, COINS Tech. Rep. 84-02, Univ. Mass., Amherst, MA.

[267] Tawel, R., S. Everhardt and A.P. Thakoor (1988). "Neural Networks For Robotic Control," in *Proc. Conf. on Neural Networks for Computing*, Snowbird, Utah.

[268] Taylor, W.K. (1956). "Electrical Simulation of Some Nervous System Functional Activities", *Information Theory*, (Eds. E.C. Cherry), Butterworths, London.

[269] Tesauro, G. (1987). "Scaling Relationships in Back-propagation Learning: Dependence on Training Set Size", *Complex Systems*, 1, 241.

[270] Thompson, J.M.T. and H.B. Stewart (1986). *Nonlinear Dynamics and Chaos*, New York, NY: John Wiley and Sons.

[271] Toffoli, T. and N. Margolus (1987). *Cellular Automata Machines*, Cambridge, MA: The MIT Press.

[272] Toomarian, N and J. Barhen (1990). "Adjoint Operators And Non-adiabatic Algorithms in Neural Networks", *Appl. Math. Lett.*, (to appear).

[273] Toomarian, N., S. Gulati and J. Barhen (1989). "Deterministic Scheduling in Homogeneous Ensembles Under Hard Deadlines", *Third Parallel Processing Symposium*, Fullerton, CA.

[273] Toomarian, N., E. Wacholder and S. Kaizerman (1987). "Sensitivity Analysis of Two-Phase Flow Problems", *Nucl. Sci. Eng.*, 99(1), 53-81.

[274] Touretzky, D. (Ed.) (1988). *Proc. of 1988 Connectionist Models Summer School*, Palo Alto, CA: Morgan-Kaufmann.

[275] Touretzky, D. (Ed.) (1988,1989). *Advances in Neural Information Processing Systems*, Vols. I and II, San Mateo, CA: Morgan-Kaufman.

[276] Tsypkin, Y.Z (1971). *Adaptation and Learning in Automatic Systems*, New York, NY: Academic Press.

[277] Uno Y., M. Kwato and R. Suzuki (1986). "Formation of Optimum Trajectory in control of Arm Movement", *Japan IEICE Tech. Rep MBE86*, 86-79.

[278] Valiant, L.G. (1984). "A Theory of the Learnable", *Comm. of ACM*, 27(11), 1134-1142.

[279] Venkataraman, S.T., *Task-Dependent Dexterous Hand Control*, Ph.D Thesis, ECE Dept., Univ. of Mass., Amherst, MA, May 1988.

[280] Venkataraman, S.T. and S. Gulati (1990). "Adaptive Sensorimotor Control for Robot-Assisted Extravehicular Activity" *Fourth Parallel Processing Symposium*, Fullerton, CA.

[281] Venkataraman, S.T. and S. Gulati (1990). "Precise-Compliant Sensorimotor Control", *Fifth IEEE International Symposium on Intelligent Control*, Philadelphia, PA, Sep. 5-7.

[282] Venkataraman, S.T. and S. Gulati (1990). "Robot-Assisted Extravehicular Activity: Part II - Sensorimotor Control" *SPIE Conference on Cooperative Intelligent Robots in Space*, Boston, MA.

[283] Venkataraman, S.T., and T. Iberall (Eds.) (1989). *Dexterous Robot Hands*, New York: Springer-Verlag.

[284] Wasserman, P.D. (1989). *Neural Computing*, New York, NY: Van Nostrand Reinhold.

[285] Waltz, D.L. (1988). "The Prospects for Building Truly Intelligent Machines", *Daedalus.*

[286] Weiner, N. (1948). *Cybernetics, Control and Communications in the Animal and the Machine*, New York, NY: John Wiley and Sons.

[287] Werbos, P. (1975). "Beyond Regression: New Tools for Prediction and Behavioral Sciences", *Ph.D Thesis*, Harvard Univ., Cambridge, MA.

[288] White, I. (1988). "The Limits and Capabilities of Machines - A Review", *IEEE Trans. Sys., Man, and Cyber.*, 18(6), 917-938.

[289] Whitney, D.E. (1969). " Resolved Motion Rate Control of Manipulators and Human Prosthesis," *IEEE Trans. on Man-Machine Systems* , MMS-10, 47-53.

[290] Widrow, B. and M.E. Hoff (1960). "Adaptive Switching Circuits", *WESCON Convention Record, 4*, 96-104.

[291] Williams, R.J., and D. Zipser (1989). "A Learning Algorithm for Continually Running Fully Recurrent Neural Networks", *Neural Computation*, 1(3), 270-280.

[292] Willshaw, D.J., Buneman, O.P. and Longuet-Higgens, H.C. (1969). " Non-Holographic Associative Memory", *Nature, 222*, 960.

[293] Willshaw, D.J. and Malsburg, Ch. V.D. (1976). "How Patterned Neural Connections Can be Set Up by Self-Organization", *Proc Royal Soc. Lond. B194*, 431-445.

[294] Wilson, K.G. (1983)., "The Renormalization Group and Critical Phenomena", *Reviews of Modern Physics*, 55(3), 583-600.

[295] Winograd, T. (1976). *Artificial Intelligence and Language Comprehension*, National Institute of Education, Washington, D.C.

[296] Winograd, S. and Cowan, J.D. (1963). *Reliable Computation in the Presence of Noise*, Cambridge, MA: The MIT Press.

[297] Wittgenstein, L. (1975). *Philosophical Remarks*, Chicago, IL: University of Chicago Press.

[298] Wolf, A. et al. (1985). "Determining Lyapunov Exponents from a Time Series", *Physica D*, 16, 285-317.

[300] Yoshikawa, T. (1984). "Analysis and Control of Robot Manipulators with Redundancy", *Robotics Research First Int'l Symp.* , (Ed. M. Brady) and R. Paul, Cambridge, MA: MIT Press, 735-748.

[301] Yovits, M.C., G.T. Jacobi and G.D. Goldstein (1962). *Self-Organizing Systems 1962*, Washington, D.C.: Spartan.

[302] Zak, M. (1988). "Terminal Attractors for Addressable Memory in Neural Networks," *Physics Letters A*, 133, No. 1,2, 18-22.

[303] Zak, M. (1989). "Terminal Attractors in Neural Networks", *Int'l Jour. on Neural Networks*, 2(3).

[304] Zak, M. (1990). "Weakly Connected Neural Networks", *Applied Math. Lett.*, 3(3).

[305] Zak, M. (1989). "The Least Constraint Principle for Learning in Neurodynamics", *Phys. Lett. A*, 135, 25-28.

[306] Zak, M. (1990). "Creative Dynamics Approach to Neural Intelligence", *Bio. Cyber.*, (in press).

[307] Zak, M. (1990). "Unsupervised Learning in Neurodynamics Using Example Interaction Approach", *Appl. Math. Lett*, 2(3), 381-286.

Vita

Sandeep Gulati is a Member of Technical Staff with the Neural Computation and Nonlinear Science Group at the Center for Space Microelectronics Technology within the Jet Propulsion Laboratory, California Institute of Technology. Currently, Gulati is the Principal Investigator for DOD/JTFPO's program on Generalized Nodal Analysis and Template Warping for Tactical Situation Assessment. He is also an investigator in the initiation of a JPL Laboratory Thrust in High Performance Robot Control for Cooperative Tasking Among Heterogeneous Agents. At JPL, he has worked on computational chaos in massively parallel neural systems and fast neural learning algorithms with applications to robotics and signal processing. Alongwith Dr. Jacob Barhen, he was responsible for developing NASA's initiative on "Neural Information Processing Systems" which involved extensive coordination with NASA centers, industry and the academia. He has authored/coauthored over 40 technical papers and invited presentations in archival journals, conferences and workshops. Gulati is a referee for NSF and NASA. In the past, he has been involved with a number of projects supported by the NASA, Department of Energy, Department of Defense, including the Innovative Science and Technology Office of the Strategic Defense Initiative Organization, Oak Ridge National Laboratory (ORNL) and Goddard Space Flight Data Center (GSFDC). Gulati received his B.Tech in Computer Science from the Indian Institute of Technology in May 1986 and an MS from the Louisiana State University in May 1988. Currently he is working on his Doctoral Degree in Computer Science from LSU. His present research interests include Neural Networks theory and applications, Perceptual Manipulation Systems, Robot-Assisted Extravehicular Activity, and High Performance Computing. He has co-chaired an IEEE Workshop on "Strategic Directions in

Computational Robotics: Symbolic, Algorithmic and Neuromorphic" at the 1990 IEEE International Robotics and Automation Conference at Cincinnati, OH in May 1990. Also, he was the Conference Co-Chair for the Fourth Annual Parallel Processing Symposium held at Fullerton, CA in April 1990. He is a member of International Neural Network Society, SPIE and IEEE Computer Society.

DOCTORAL EXAMINATION AND DISSERTATION REPORT

Candidate: SANDEEP GULATI

Major Field: COMPUTER SCIENCE

Title of Dissertation: COMPUTATIONAL NEURAL LEARNING FORMALISMS FOR PERCEPTUAL MANIPULATION : singularity Interaction Dynamics Model

Approved:

Major Professor and Chairman

Dean of the Graduate School

EXAMINING COMMITTEE:

Date of Examination:

August 23, 1990