

GRAPHICAL REPRESENTATIONS OF
ATTRACTOR AND VECTOR SPACE
FOR NEURAL NETS

By

BENNETT S. CARTER

Bachelor of Science

Whitman College

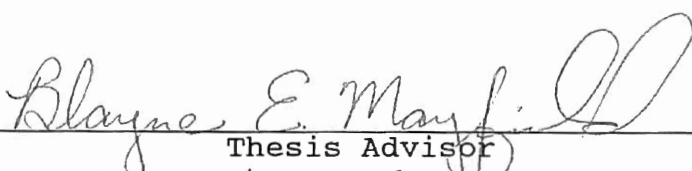
Walla Walla Washington

1984

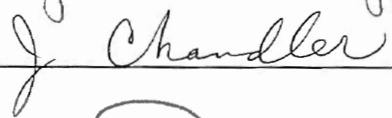
Submitted to the Faculty of the
Graduate College of the
Oklahoma State University
in partial fulfillment of
the requirements for
the degree of
MASTER OF SCIENCE
May, 1993

GRAPHICAL REPRESENTATIONS OF
ATTRACTOR AND VECTOR SPACE
FOR NEURAL NETS

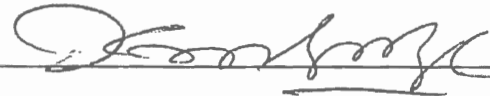
Thesis Approved:




Thesis Advisor



J. Chandler



[unclear]



Dean of the Graduate College

ACKNOWLEDGMENTS

I sincerely appreciate the help of those who helped during this study. I would like to especially express my appreciation for the instruction, advice, encouragement and patience of Mr David W. Miller without whom this would not have been possible. Appreciation is also extended to Dr. Blayne Mayfield, Dr. K. M. George and Dr. John Chandler for serving on my committee.

I would also like to offer special thanks to my wife Leslie Carter for putting up with my long hours and for not requesting id more than a few times when I would disappear for weeks at a time.

TABLE OF CONTENTS

Chapter	Page
I. INTRODUCTION.	1
Visualization of Large Quantities of Data. . .	1
Presenting Large Quantities of Data.	2
The Three Parts of Computer Visualization. . .	3
Analysis.	3
Display	3
Interaction	4
Overview of Thesis	5
II. LITERATURE REVIEW	8
III. MATHEMATICAL BACKGROUND	10
Derivation of Equation for Weight Calculation.	10
Linear Algebra	11
IV. PROGRAM VALIDATION.	17
V. BASIC FINDINGS.	22
VI. CONCLUSIONS	49
Further Study.	49
REFERENCES	51
APPENDIXES	52
APPENDIX A - STARTING SCREEN.	53
APPENDIX B - DISPLAY AND DISPLAY OPTIONS.	55
APPENDIX C - WORKING WITH IMAGES.	61
APPENDIX D - DEALING WITH CHAOS	62
APPENDIX E - SUMMARY OF OPTIONS	64
APPENDIX F - INPUT FILE DESCRIPTIONS.	67

LIST OF TABLES

Table	Page
I. Training Values for Recurrent XOR Networks.	17
II. Outputs for the Three Node XOR Network.	19
III. Outputs for the Four Node XOR Network	20
IV. Outputs for the Five Node XOR Network	21
V. Basic Structure of Five Node XOR Network.	23
VI. Weight Matrix for XOR with Two-Cycle Delay.	28
VII. Weight Matrix for XOR with Three-Cycle Delay.	28
VIII. Weight Matrix for XOR with Four-Cycle Delay	28
IX. Outputs of Node Zero in the Five Node XOR After the Second Iteration.	30
X. Outputs of Nodes Zero and One in the Four Node XOR After the First Iteration.	33
XI. Final Coordinates for the Five Cyclic Points in the Five Node XOR with Nodes One and Two Graphed	34

LIST OF FIGURES

Figure	Page
1. Vector Field, Five-node Network, Iteration = 3, X Axis is Node 1, Y Axis is Node 2.	5
2. Flow Chart for the Program.	7
3. Sample Network Configuration.	12
4. Idealized Critical Point Examples	16
5. Vector Field, Five-Node Network, Iteration = 1, X Axis is Node 4, Y Axis is Node 2.	24
6. Vector Field, Five-Node Network, Iteration = 2, X Axis is Node 4, Y Axis is Node 2.	24
7. Vector Field, Five-Node Network, Iteration = 3, X Axis is Node 4, Y Axis is Node 2.	25
8. Vector Field, Five-Node Network, Iteration = 4, X Axis is Node 4, Y Axis is Node 2.	25
9. Vector Field, Four-node Network, Iteration = 1, X Axis is Node 4, Y Axis is Node 3.	26
10. Vector Field, Four-node Network, Iteration = 2, X Axis is Node 4, Y Axis is Node 3.	26
11. Vector Field, Four-node Network, Iteration = 3, X Axis is Node 4, Y Axis is Node 3.	27
12. Vector Field, Four-node Network, Iteration = 4, X Axis is Node 4, Y Axis is Node 3.	27
13. Vector Field, Three-node Network, Iteration = 1, X Axis is Node 2, Y Axis is Node 1.	31
14. Vector Field, Five-Node Network, Iteration = 1, X Axis is Node 4, Y Axis is Node 2.	31
15. Vector Field, Four-Node Network, Iteration = 1, X Axis is Node 2, Y Axis is Node 1.	33

Figure	Page
16. Five Point Cyclic Attractor, Five-Node Network, Iteration = 4, X Axis is Node 2, Y Axis is Node 1.	37
17. Time Chart for Five Point Cyclic Attractor, Five-Node Network, Iteration = 4, X Axis is Node 2, Y Axis is Node 1.	37
18. Short Pass of Chaotic Attractor, Five-Node Network, Iteration = 4, X Axis is Node 2, Y Axis is Node 1.	38
19. Longer Pass of Chaotic Attractor, Five-Node Network, Iteration = 4, X Axis is Node 3, Y Axis is Node 1.	38
20. Time Chart of Chaotic Behavior, Five Node Network, Iteration is 4, X Axis is Node 3, Y Axis is Node 1.	39
21. 750 Paths of Chaotic Attractor, Five-Node Network, Iteration is 4, X Axis is Node 3, Y Axis is Node 1.	39
22. 3000 Paths of Chaotic Attractor, Five-Node Network, Iteration = 4, X Axis is Node 3, Y Axis is Node 1.	40
23. 6000 Points of Chaotic Attractor, Five-Node Network, Iteration = 4, X Axis is Node 3, Y Axis is Node 1.	40
24. Path of Chaotic Attractor, Five-Node Network, Iteration is 4, X Axis is Node 3, Y Axis is Node 1, Driving Force = 0.57147	44
25. Points of Chaotic Attractor, Five-Node Network, Iteration is 4, X Axis is Node 3, Y Axis is Node 1, Driving Force = 0.57147	44
26. Path of Chaotic Attractor, Five-Node Network, Iteration is 4, X Axis is Node 3, Y Axis is Node 1, Driving Force = 0.58025	45
27. Points of Chaotic Attractor, Five-Node Network, Iteration is 4, X Axis is Node 3, Y Axis is Node 1, Driving Force = 0.58025	45
28. Path of Cyclic Attractor, Five-Node Network, Iteration is 4, X Axis is Node 3, Y Axis is Node 1, Driving Force = 0.58025	46

Figure	Page
29. Points of Cyclic Attractor, Five-Node Network, Iteration is 4, X Axis is Node 3, Y Axis is Node 1, Driving Force = 0.58299	46
30. Path of 4 point Cyclic Attractor, Five-Node Network, Iteration is 4, X is Node 3, Y is Node 1, Driving Force = 0.64000.	47
31. Points of 4 Point Cyclic Attractor, Five-Node Network, Iteration is 4, X is Node 3, Y is Node 1, Driving Force = 0.64.	47
32. Cyclic Attractor Pushed to Chaos, Five-Node Network, Iteration is 4, X is Node 2, Y is Node 1, Driving Force = 1.1777777777	48
33. Cyclic Attractor Pushed to Chaos, Five-Node Network, Iteration is 4, X is Node 2, Y is Node 1, Driving Force = 1.1777777777	48

CHAPTER I

INTRODUCTION

Visualization of Large Quantities of Data

As scientific data grows in both complexity and sheer volume, visualization plays an increasingly important role in understanding a system as a whole. Vast amounts of data can be displayed as surfaces, solids or vector fields, often revealing underlying structures which offer insights to their behavior.

For example, studies of fluid flow traditionally have been performed by photographing smoke or oil patterns to produce high-resolution two-dimensional images. Not only does the amount of data captured in this fashion increase rapidly for time-dependent systems, but recent numerical simulations have involved higher dimensions, resulting in an exponential growth of data.

This explosion of data is the driving force behind the increasing reliance upon visualization. A simple three dimensional data set of $100 \times 100 \times 100$ would produce 1.0^6 points. If it varied over 100 time steps then data for a total of 1.0^8 points results.

Presenting Large Quantities of Data

The only way to make use of this amount of data is to organize it into a form that clearly presents the order and structure inherent in the object being studied or simulated. Since the purpose of this project is to view the behavior of recurrent neural nets (networks where one layer's outputs are the inputs for a previous layer), a way of coherently presenting the information acquired must be found. It is important to point out that even if it were possible to display all data computed for a vector field describing a neural net's behavior, the resulting image would be completely unintelligible. Instead, the computer must analyze the raw data and draw from it the elements that best reveal the behavior of the system as a whole.

To begin such an analysis, a basic approach to the problem must first be decided upon. Since a network can be thought of as performing a mapping from a multidimensional vector space into itself, its behavior can be described by a vector field which is analogous to smooth fluid flow in many dimensions. For this reason a logical approach is to study recurrent neural network behavior in terms of topological dynamics. In this framework, the sources of this smooth flow are called repelling points, while the sinks are called attractors.

Even a well thought out display cannot give all of the information desired. To this end the display of the network, as well as computer visualization in general, has three parts: analysis, display, and interaction. Each will be discussed briefly.

The Three Parts of Computer Visualization

Analysis

In order to extract useful information, it is necessary for the computer to perform some sort of analysis of the raw data. This project displays a vector field, which has properties analogous to the slope of the network response functions as projected into two dimensions.

The vector field itself is produced by using arrows to indicate the direction of variations of the vector components for several positions, These variations are described to first order by the jacobian matrix of the first partial derivative of the vector components.

Display

To increase the amount of understandable information on the display, colors are used to denote magnitude while an arrow shows the direction of the change in the vector field. Eigenvalues and eigenvectors are also displayed to give a better understanding of the neighborhood around that point. It should be pointed out that what is being graphed is just

one slice of the vector field projected to a two dimensional display. The slice displayed depends on the two nodes chosen to be graphed on the x and y axis of the display. For the sake of illustration, figure 1 on page 5 shows a typical image.

Interaction

No display format can show all of the available information at one time. For this reason if some specific piece of information is required, it must sometimes be requested through user interaction. In this project, interactive information is usually displayed at the bottom of the image in two boxes that show eigenvector and eigenvalue information in numerical and graphical form for a given location. In the image that follows, the lower left box gives this information for the current location of the cursor, while the right box gives the information for the location reached when the state of the network is either pushed or allowed to relax to some location.



Figure 1 Vector Field, Five-node Network, Iteration = 3, X Axis is Node 1, Y Axis is Node 2

Overview of Thesis

The purpose of this project is to graphically demonstrate the dynamics of some simple recurrent networks, and to view the network's behavior when it is either driven or allowed to relax from any given state to a final attractor or attractors.

The basic operation of the program involves providing one or more input files that either specify the fixed points that the network is to have, or give the weights that describe an already existing recurrent network.

The reason for these two different types of input files has to do with the evolution of the program. Initially, the input files specified the desired fixed points to give a better idea of what kind of behavior was to be found and from this to decide how to best display and interact with the network. Several features were added at this point to try to get a better view of the dynamics of the network. After the program had been tested with some basic networks in this fashion, attention was turned to the recurrent networks described in the paper "Experimental Analysis of the Real-time Recurrent Learning Algorithm" (Williams & Zipser, 1989). This paper gave the weights and configuration for three recurrent networks with three, four, and five nodes. Each implemented the exclusive or function (XOR) but with different amounts of delay from the time of input to the corresponding output. The numbers of delay cycles for the three, four and five node networks were two, three and four cycles respectively. Each of these networks was studied to verify that the project did indeed display real behavior. This is discussed in detail in the section on Program Verification.

A flow chart is provided to give a better overall view.

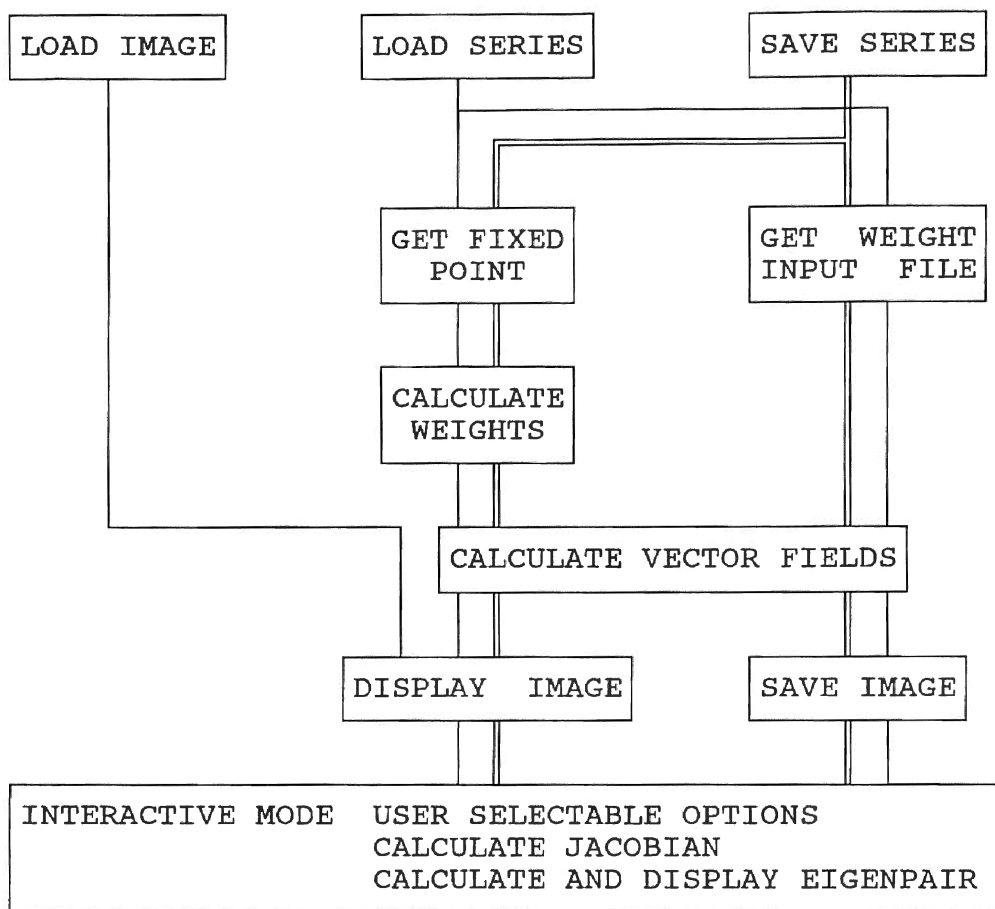


Figure 2 Flow Chart for the Program

CHAPTER II

LITERATURE REVIEW

Comparatively little work is being done with recurrent neural nets because they are more difficult to understand than feed forward networks.

Most papers on recurrent networks are designed to try to see what can be done with them and to try to understand what occurs. When a topic is new, such as neural nets are, visualization is an effective way to try to get an overview of the data. Unfortunately, while neural nets are being widely investigated only rarely does the graphical display receive any attention. A few exceptions are the Hinton diagrams in 'The Geometry of Backpropagation Training: Visualization, Heuristics, and Theory' (Arnaldo & Miller, 1990) and 'Visualizing Processes in Neural Networks' (Tesauro & Wejchert 1991) which uses several visualization methods to investigate learning in neural networks which use the back-propagation algorithm.

Hinton's diagrams represent synaptic strength (weight) data by drawing squares where their size is proportional to the magnitude of the weight, while the color indicates the sign.

Because a Hinton diagram does not necessarily reveal a network's topology, Wejchert and Tesauro (1991) took this a step further by displaying the strength of weights in terms of arrows pointing to a node, the longer the arrow, the stronger the weight magnitude, and once again the color indicates sign.

To provide additional information, they used other visual aids such as displaying the network's error and output as it is trained as well as showing trajectories as projected down onto a plane.

Visualization tools such as these are useful to show the dynamics of a network as it is trained, and from this to make new heuristics, as found by Arnaldo and Miller (1990).

For recurrent networks, a logical visualization scheme would be to picture the vector field and the attractor points, as they change during the learning process and in the final configuration. This will permit a better understanding of how the possible solutions (attractors) evolve as a network is trained. Since little or no attention has been paid to this topic, it is fitting that it be addressed.

CHAPTER III

MATHEMATICAL BACKGROUND

Derivation of Equation for Weight Calculation

The following shows how the weights are calculated for networks where the fixed points are defined in the input file. The networks to be modeled are ones where there are n levels with at most m nodes per level and the output from every node in level i goes to each node in the next level, and all nodes in level n go to level 1. For such a network, the output of node i is expressed by

$$O_i = 1 / (1 + \exp(-(W_{1i} * O_1 + W_{2i} * O_2 + \dots + W_{mi} * O_m)))$$

Where O_i is the output of node i . This can be rewritten as $\ln(O_i) - \ln(1 - O_i) = W_{1i} * O_1 + W_{2i} * O_2 + \dots + W_{mi} * O_m$ as a result of the following derivation.

$$O_i = \frac{1}{1 + \exp(-(W_{1i} * O_1 + W_{2i} * O_2 + \dots + W_{mi} * O_m))}$$

$$1 / O_i = 1 + \exp(-(W_{1i} * O_1 + \dots + W_{mi} * O_m))$$

$$1 / O_i - 1 = \exp(-(W_{1i} * O_1 + \dots + W_{mi} * O_m))$$

$$\ln((1 - O_i) / O_i) = -(W_{1i} * O_1 + \dots + W_{mi} * O_m)$$

$$\ln(1 - O_i) - \ln(O_i) = -(W_{1i} * O_1 + \dots + W_{mi} * O_m)$$

$$\ln(O_i) - \ln(1 - O_i) = W_{1i} * O_1 + \dots + W_{mi} * O_m$$

It is important to notice that $W_{1i} * O_1 + W_{2i} * O_2 + \dots$ is just the dot product of inputs to node i and the appropriate weights. Thus the problem can be represented as $Ax = b$ where the solution vector b is $\ln(O_i) - \ln(1 - O_i)$, vector x is the weights to be calculated from each node to node i , and A is the output from each node for each fixed point.

Since the value of the fixed point for a node is its output, matrix A is fairly easy to obtain. By the time A is obtained, b is also known, and the equation can be solved for x , which represents the weights to the current node.

Linear Algebra

Once the entire weight matrix has been either read in from a file or calculated, the next step is to make the jacobian matrix from which the vector field will be graphed. The purpose for using the jacobian is somewhat analogous to trying to find the shape of a simple two-dimensional curve by taking the first derivative, looking for the critical points (areas where the derivative is zero), and recreating the curve from that. For a dynamic system in an n -dimensional space, one needs to look for the fixed points to understand the basic topology. If one graphs the resulting vector field and finds the areas with little motion, these areas will surround fixed points. Attracting points are the easiest to find in this way, but sometimes others can be visually recognized. To classify the fixed

points precisely requires taking the derivatives of the components of the vector and looking at the fixed points. Since the jacobian takes the derivatives of the components of the vector, it represents the behavior of the neighborhood around any point where it is evaluated. As an example, the following network will be used.

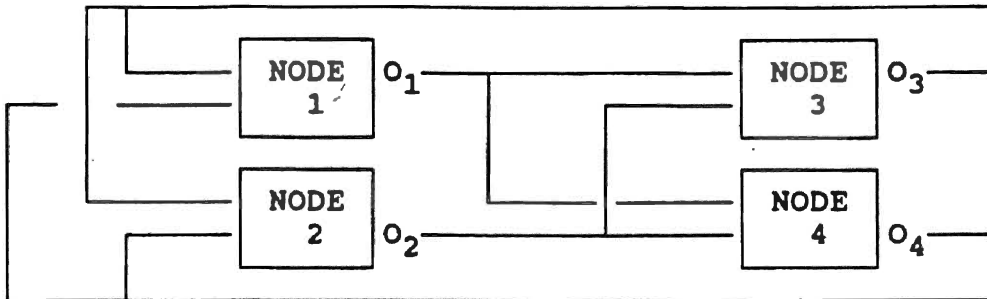


Figure 3 Sample Network Configuration

Since the program can only display two-dimensional images only two nodes can be used, and the resulting jacobian will be a 2X2 matrix in the form:

$$\begin{bmatrix} \frac{\partial O_1^{t+i}}{\partial O_1^t} & \frac{\partial O_1^{t+i}}{\partial O_2^t} \\ \frac{\partial O_2^{t+i}}{\partial O_1^t} & \frac{\partial O_2^{t+i}}{\partial O_2^t} \end{bmatrix}$$

where i is the number of iterations and O_j^{t+1} is the output of node j at time $t+i$. The partials may be broken down by

the chain rule so that for the above network, where i equals 2, the Jacobian matrix element

$$\frac{\partial O_j^{t+2}}{\partial O_k^t}$$

can be interpreted as the rate of change of the output of node j at time $t+2$, with respect to the output of node k at time t .

The partial $\frac{\partial O_1^{t+2}}{\partial O_1^t}$ can be broken down by the chain rule to

$$\text{become } \frac{\partial O_1^{t+2}}{\partial O_3^{t+1}} * \frac{\partial O_3^{t+1}}{\partial O_1^t} + \frac{\partial O_1^{t+2}}{\partial O_4^{t+1}} * \frac{\partial O_4^{t+1}}{\partial O_1^t}$$

where each partial of the form: $\frac{\partial O_i}{\partial O_j}$ is $O_j * (1 - O_j) * W_{ji}$

In addition to displaying the vector field, the eigenvalues and eigenvectors are also shown since they represent the degree of change of the neighborhood around a point as it is mapped from one place to another. This mapping from point to point can be thought of as what happens as the network relaxes from one point to another. The interpretation of these eigenvalues and eigenvectors is easiest at a fixed point since the neighborhood around such a point moves very little. The reason for this is that, by definition, a fixed point maps from itself, to itself. Since the fixed point does not move, and the vector field is continuous, the neighborhood around it will move very little.

There are three types of fixed points possible in these displays. One is an attractor which is typically rather easy to recognize since the network relaxes to it from all sides. A second type of fixed point is the repulsor, which is just the opposite of the attractor since it repels on all sides. The third type of fixed point is the saddle point, which attracts on one axis and repels on another.

Topologically, this third type of fixed point is analogous to a saddle, hence its name. Several attractors and saddle points were found, but no repulsors were discovered in any of the networks.

The eigenvalues help give an understanding of the fixed point by indicating the amount of change in the neighborhood of a network as it relaxes from one point to another.

A quick interpretation of the eigenvalue and its corresponding eigenvector (also called an eigenpair), is that the eigenvector is simply mapped into a scaled version of itself, and the scale factor is the eigenvalue.

Thus real eigenvalues with an absolute value less than one indicate a squashing of the neighborhood in the axis of the eigenvalue's corresponding eigenvector. Similarly, an absolute eigenvalue greater than one indicates a stretching effect. For this reason the eigenvalues are much less than one at attractors (very close to zero), since the neighborhood tends to change less and less. Saddle points are quite another matter however, since a saddle point

attracts in only one direction, its eigenpair values can be much larger.

In addition to the squeezing and stretching effect mentioned, there can also be a rotation effect which happens when the eigenpair values are complex. Under such conditions the network may spiral into an attractor or out of a repulsor. Since these rotations are two-dimensional, and the pairs of eigenvalues are complex conjugates of each other, there is no analog of a saddle point in this case.

The interpretation of the eigenpairs is easiest at a fixed point since the behavior of the neighborhood is known from the start. However, when considering the eigenpair at any point that is not a fixed point, it can be more difficult to understand. In general, the eigenpairs represent the change of the whole neighborhood near a given point as it jumps to another neighborhood. The eigenpairs measure the net change that occurs when the bulk movement is cancelled out. This is why the behavior of the eigenpairs is different from the arrows. It cancels out the bulk movement of the arrows and just looks at the change of the neighborhoods as they ride along together.

In the case of complex eigenvectors the vectors in the neighborhood of a point are mapped onto scaled and rotated versions of themselves.

To show just what a fixed point looks like in a vector field, the images below show idealized examples of critical points.

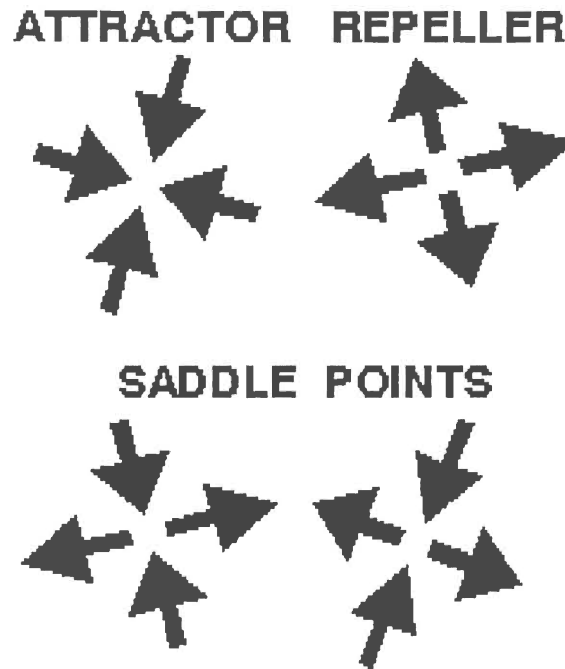


Figure 4 Idealized Critical Point Examples

CHAPTER IV

PROGRAM VALIDATION

To verify the program's operation it was tested on the three recurrent networks described in the paper 'Experimental Analysis of the Real-time Recurrent Learning Algorithm' (Williams and Zipser 1989). In each network a single unit was taught on every cycle to output the XOR of the inputs occurring two or more cycles previously.

As a first step in verification, a separate program was written to calculate the outputs of each node for these networks for the training input values given in table I.

TABLE I
TRAINING VALUES FOR RECURRENT
XOR NETWORKS

X = 0.0, Y = 0.0	X = 1.0, Y = 0.0
X = 0.0, Y = 1.0	X = 1.0, Y = 1.0

To verify the test programs' accuracy its output was compared to hand calculations. Finally, these results were compared

to the thesis results by using the features 'M', 'm', and 'o'. Use of each of these features is mentioned individually in the appendix.

The basic outline is to do the following:

- 1) The 'M' feature can be used to 'move' the x and y locations to any coordinate.
- 2) The 'm' option works in conjunction with the 'M' feature by allowing the system to relax one time step from the location that was set by 'M'.
- 3) Use the 'o' option to view each node's output. Compare them to the outputs calculated by the other program.

The verification performed here is two-fold.

The output of the nodes of the network are compared to the values calculated by the test program to check for errors.

The outputs of the network must also solve the exclusive or problem at the appropriate time step in order to verify that the network was properly trained in the first place.

The outputs for the three, four and five node XOR networks are displayed in table2 II, III and IV on pages 19, 20 and 21 respectively.

TABLE II
 OUTPUTS FOR THE THREE NODE XOR NETWORK

INPUT	Time	Node 0	Node 1	Node 2
X = 0	1	0.062973	0.057342	0.026597
Y = 0	2	0.063553	0.057483	0.064788
X = 0	1	0.000303	0.952574	0.024127
Y = 1	2	0.000364	0.951920	0.976459
X = 1	1	0.9168	0.000136	0.024127
Y = 0	2	0.923046	0.000177	0.969084
X = 1	1	0.047426	0.043107	0.021881
Y = 1	2	0.047734	0.043153	0.043091

Node number 2 was trained to give the desired signal after a delay of two cycles.

TABLE III
 OUTPUTS FOR THE FOUR NODE XOR NETWORK

X = 0, Y = 0				
Time	Node 0	Node 1	Node 2	Node 3
1	0.924	0.0009	0.9644	0.01799
2	0.997	0.0003	0.1646	0.9972
3	0.853	0.0003	0.0357	0.0444

X = 0, Y = 0				
Time	Node 0	Node 1	Node 2	Node 3
1	0.021881	0.10910	0.9526	0.013387
2	0.012235	0.114632	0.984029	0.99112
3	0.010956	0.126574	0.965877	0.974335

X = 1, Y = 0				
Time	Node 0	Node 1	Node 2	Node 3
1	0.02188	0.1192	0.94268	0.013387
2	0.012185	0.122880	0.978655	0.990405
3	0.010906	0.136103	0.955725	0.973381

X = 1, Y = 1				
Time	Node 0	Node 1	Node 2	Node 3
1	0.000041	0.947846	0.924142	0.009952
2	0.000011	0.813369	0.083521	0.995071
3	0.000019	0.826423	0.015528	0.016665

Node number three was trained to give the signal after a delay of three cycles.

TABLE IV
 OUTPUTS FOR THE FIVE NODE XOR NETWORK

X = 0, Y = 0					
Time	Node 0	Node 1	Node 2	Node 3	Node 4
1	0.858149	0.091123	0.990048	0.099750	0.008852
2	0.884214	0.078558	0.024179	0.099853	0.999801
3	0.934080	0.073634	0.015861	0.054201	0.002225
4	0.940860	0.87861	0.015815	0.095171	0.004232

X = 1, Y = 0					
Time	Node 0	Node 1	Node 2	Node 3	Node 4
1	0.880797	0.956893	0.991837	0.000500	0.007392
2	0.072144	0.811000	0.11406	0.000129	0.999602
3	0.064438	0.813568	0.948972	0.000072	0.002420
4	0.039447	0.827320	0.960938	0.000141	0.999723

X = 0, Y = 1					
Time	Node 0	Node 1	Node 2	Node 3	Node 4
1	0.869892	0.000500	0.991837	0.960834	0.007392
2	0.62987	0.000128	0.011411	0.838361	0.999642
3	0.049023	0.000120	0.943651	0.738485	0.002622
4	0.054743	0.000155	0.965972	0.872232	0.999752

X = 1, Y = 1					
Time	Node 0	Node 1	Node 2	Node 3	Node 4
1	0.890903	0.099750	0.993307	0.099750	0.005486
2	0.912043	0.085372	0.026426	0.099307	0.999640
3	0.950922	0.080272	0.018154	0.053941	0.001220
4	0.955124	0.095456	0.019992	0.094562	0.002354

Node number 4 is trained to give the desired signal after a delay of four cycles.

CHAPTER V

BASIC FINDINGS

To get an idea of how the networks described in the paper by Williams and Zipser (Reference) solve the XOR problem, images were not only created for each view of the network, but also for each time step of those views to see how the state of the network changes as time goes on.

As was mentioned in their paper, the networks essentially organize themselves into feedforward networks by reducing all recurrent weights, and allowing only those weights that go forward in the network to have large values.

The effect of this format can be seen by viewing the images of any two nodes as they vary with time. In general, once the solution has propagated to both of these nodes, the overall topology remains relatively constant for the remaining iterations. As an example, the basic structure of the five node network is shown in table V on page 23.

TABLE V
 BASIC STRUCTURE OF FIVE
 NODE XOR NETWORK

From Layer	1	2	3	4
To Node(s)	1,3	0	2	4

To illustrate how the images change very little once the solution reaches the nodes being examined, the images for four iterations of the five node network are shown in figures 5 through 8 on pages 24 and 25. Nodes one and three are graphed. Notice how they change relatively little, which is as expected since their primary contribution is made in the first iteration.

Contrast this to the images formed by graphing nodes two and three in figures 9 through 12 on pages 26 and 27. Notice how changes continue until the third iteration when node two makes its primary contribution.

In addition to the above findings, the results also indicate that two different ways of solving the XOR problem were used. The three and five node networks' solutions were similar, but varied greatly from the four node network's solution. The weights for these networks are given in tables VI, VII and VIII on page 28.



Figure 5 Vector Field, Five-Node Network, Iteration 1, X Axis is Node 4, Y Axis is Node 2

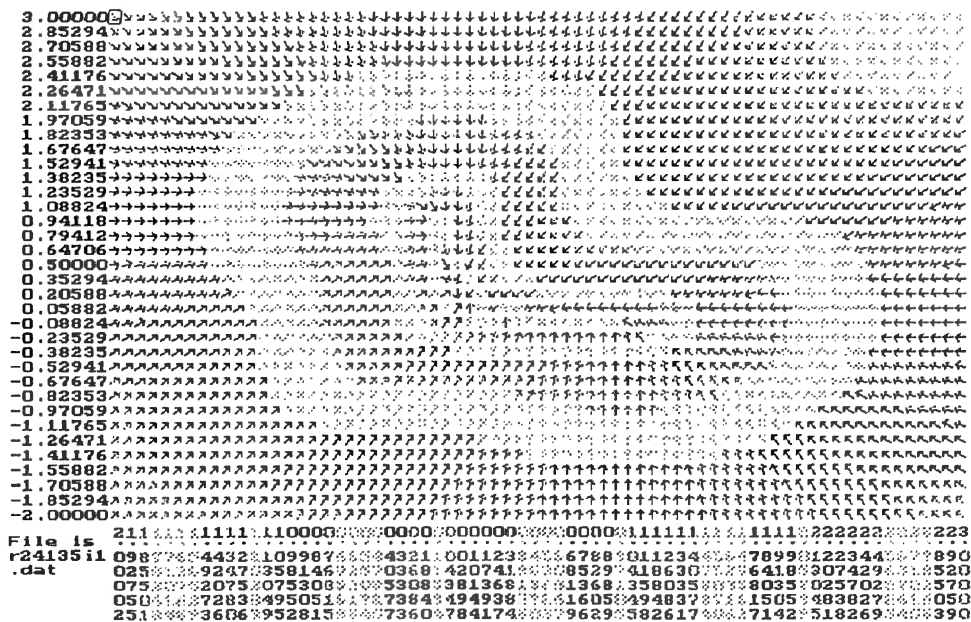


Figure 6 Vector Field, Five-Node Network, Iteration 2, X Axis is Node 4, Y Axis is Node 2



Figure 7 Vector Field, Five-Node Network, Iteration 3, X Axis is Node 4, Y Axis is Node 2

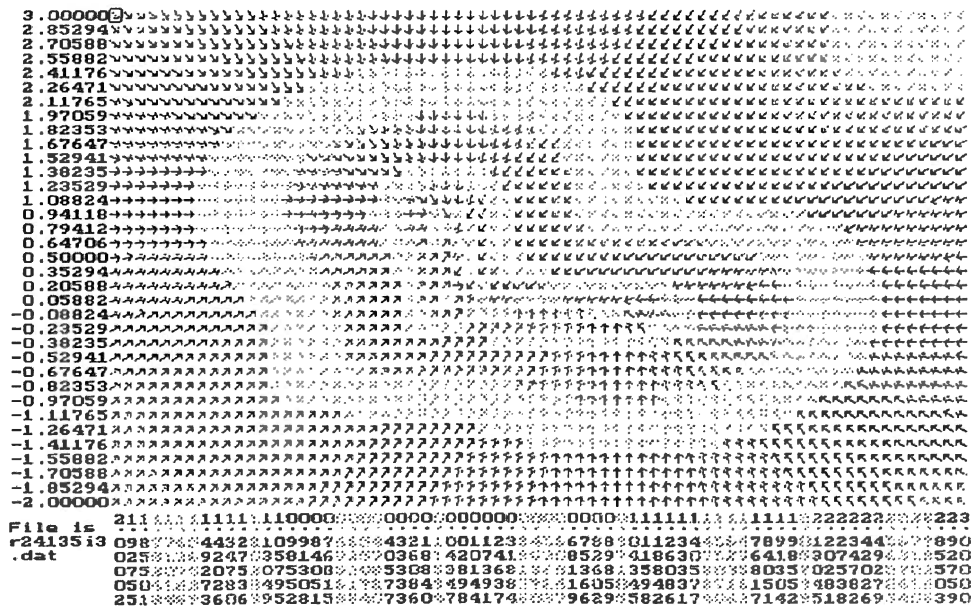


Figure 8 Vector Field, Five-Node Network, Iteration 4, X Axis is Node 4, Y Axis is Node 2

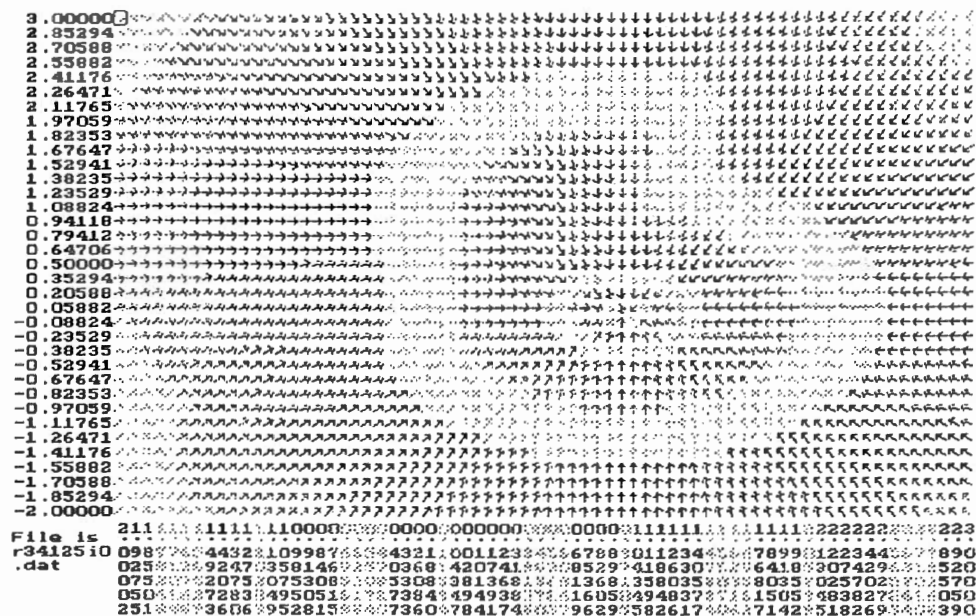


Figure 9 Vector Field, Four-Node Network,
Iteration 1, X Axis is Node 4,
Y Axis is Node 2

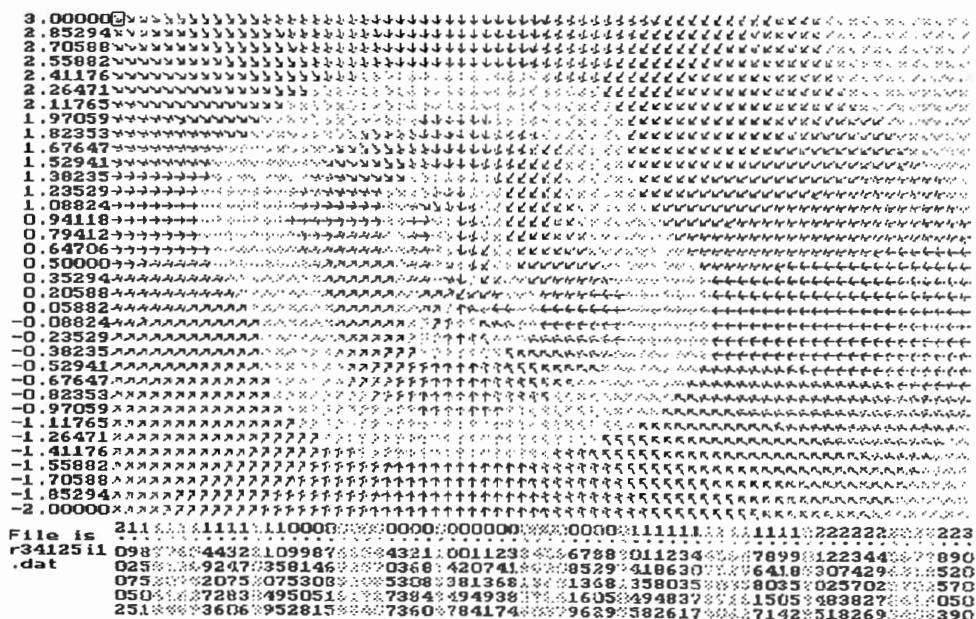


Figure 10 Vector Field, Four-Node Network,
Iteration 2, X Axis is Node 3,
Y Axis is Node 2



Figure 11 Vector Field, Four-Node Network, Iteration 3, X Axis is Node 4, Y Axis is Node 3



Figure 12 Vector Field, Four-Node Network, Iteration 4, X Axis is Node 4, Y Axis is Node 3

TABLE VI
WEIGHT MATRIX FOR XOR WITH
TWO-CYCLE DELAY

Node	BIAS	X	Y	INTERNODE CONNECTIONS		
0	-2.7	5.1	-5.4	0.1	0.2	-0.3
1	-2.8	-6.1	5.8	0.3	0.0	-0.6
2	-3.6	-0.1	-0.1	7.8	7.8	-0.3

TABLE VII
WEIGHT MATRIX FOR XOR WITH
THREE-CYCLE DELAY

Node	BIAS	X	Y	INTERNODE CONNECTIONS			
0	2.5	-6.3	-6.3	-0.7	-0.9	-0.5	-0.1
1	-7.0	5.0	4.9	-1.6	-1.8	0.3	0.1
2	3.3	-0.5	-0.3	-7.6	-7.3	2.2	-0.9
3	-4.0	-0.3	-0.3	1.0	1.4	9.3	-1.4

TABLE VIII
WEIGHT MATRIX FOR XOR WITH
FOUR-CYCLE DELAY

Node	BIAS	X	Y	INTERNODE CONNECTIONS				
0	1.8	0.2	0.1	1.9	-6.3	-0.2	-6.3	0.3
1	-2.3	5.4	-5.3	0.2	-1.9	0.0	-1.6	-0.1
2	4.6	0.2	0.2	-9.2	-1.0	-0.2	-1.1	-0.4
3	-2.2	-5.4	5.4	0.2	-1.8	0.2	-2.0	-0.5
4	-4.6	-0.3	-0.3	-1.1	-0.5	14.3	-0.4	-0.8

In the three node network nodes 0 and 1 perform the same function as nodes 1 and 3 in the five node network. The similarities of these nodes in the three and five node networks can be seen as early as the first time step. This early similarity is expected since these nodes have large input weights in both networks, while all other input weights are small. Thus they are effectively the first layer of the network. See figures 13 and 14 on page 31. As it turns out, these two nodes perform the bulk of the XOR function by themselves. Their work can be understood by examining the images produced as shown in figures 13 and 14 on page 30.

When the network is placed in any region near $(0,0)$ or $(1,1)$ the state of the network relaxes very close to $(0,0)$ in just one time step. If however, one input is near one and the other is near zero, then the state of the network will either move quickly to a corner attractor, or it will move slowly toward one of the saddle points. In either case one of the nodes will output a number near one, while the other outputs a number near zero. The result is that the next layer needs only to sum the outputs from the previous layer.



Figure 13 Three-node Network, Iteration = 1,
X Axis = Node 2, Y Axis = Node 1



Figure 14 Five-node Network, Iteration = 1,
X Axis = Node 4, Y Axis = Node 2

This is indeed what happens in the three node network, but in the five node network, it is negated first, with the result that the second layer, which consists of node number zero, has the following outputs for the inputs given below in table IX.

As can be seen, node zero is the negation of the XOR solution. However, since the weight from node zero to node two (node number two is the third layer) is also negative, the final result is to negate the outputs of node zero, leaving node two with the outputs listed in table IX.

This shows that the XOR problem is actually solved in the third iteration, and is just sent along to the next and final layer at which time the output is expected.

TABLE IX
 OUTPUTS OF NODES ZERO AND TWO
 IN THE FIVE NODE XOR NETWORK
 AFTER THE SECOND ITERATION.

INPUTS		OUTPUTS	OUTPUTS
X	Y	for Node 0	for Node 2
0	0	0.88421406	0.01586072
0	1	0.06298731	0.94765098
1	0	0.07214407	0.94897223
1	1	0.91204341	0.01815388

The four-node network approaches the XOR problem from another direction. Rather than having both nodes in the first layer output low values when both inputs are alike, it does just the opposite, creating values near one when inputs are both near one or near zero, and creates low values when only one input is near one. A summary of the outputs of nodes 0 and 1 is presented in table X on page 33 while a corresponding image is displayed in figure 15 on page 33.

The image of the first layer of the four node network is not quite as easy to understand since it sets up the negation of the XOR problem. Here, whenever one input is near one and the other is near zero, the network will relax to a state near the origin, but if both nodes are either near zero or one, then the network will relax in a counterclockwise direction to make one of the outputs near one. Since this is essentially the opposite of what was produced in the three and five node networks, simply summing the outputs will not produce the desired results. Instead, the outputs must be first negated and then summed. To accomplish this, the weights from nodes 0 and 1 to node 2 are both of the same magnitude as in the other networks, but this time both are negative. The end result is that, again, the XOR problem is solved before the final iteration and the result is just passed along to the final node in the network.



Figure 15 Vector Field, Four-Node Network,
Iteration = 1, X Axis is Node 2,
Y Axis is Node 1

TABLE X

OUTPUTS OF NODES ZERO AND ONE IN
THE FOUR NODE XOR AFTER
THE FIRST ITERATION

INPUTS		OUTPUTS for	
X	Y	Node 0	Node 1
0	0	0.92414182	0.00091105
0	1	0.02188127	0.10909682
1	0	0.02188127	0.11920292
1	1	0.00004108	0.94784644

While a great many attractors were located, only a few saddle points were either found or implied by the images (whenever two attractors are on one image there must be a saddle point in between them whether the program located it or not). Also, a few images did not settle down to any fixed point, but instead would jump from one point to a second point and then jump back to the first. In one image, this cyclic behavior was shown to be much more complex, having a cycle duration of five. After the network relaxes for about 80 cycles it settles into a cycle with the coordinates displayed in table XI.

Though the network was allowed to relax for over fifty thousand cycles, it never deviated from these locations. The image with the path drawn on it and its corresponding time chart are shown in figures 16 and 17 on page 35.

TABLE XI

FINAL COORDINATES FOR THE FIVE
CYCLIC POINTS IN THE FIVE
NODE XOR WITH NODES ONE
AND TWO GRAPHED

Cycle	X	Y
1	0.92902539	0.02672312
2	0.06074804	0.76261312
3	0.27659350	0.00116059
4	0.88288307	0.25166257
5	0.31202917	0.54508098

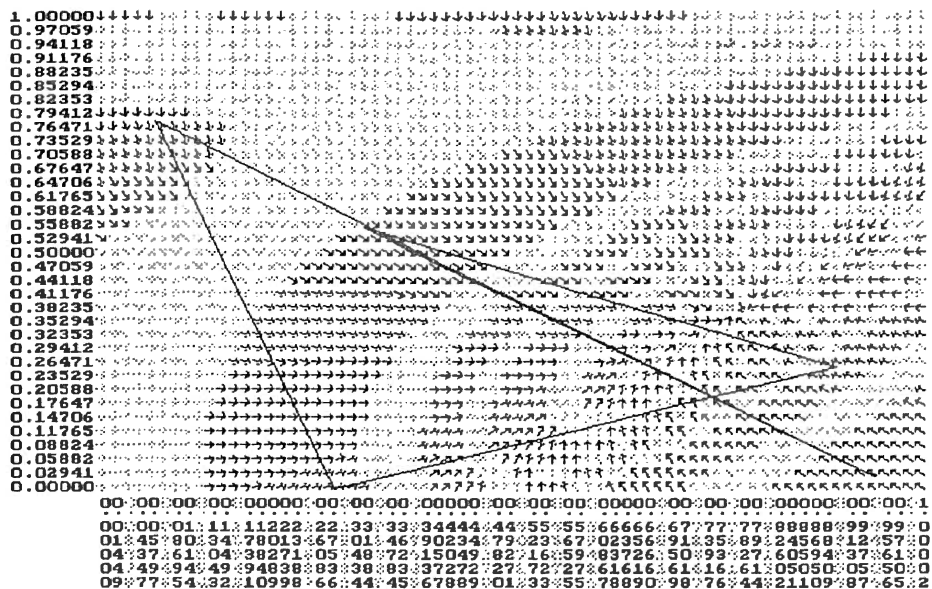


Figure 16 Five Point Cyclic Attractor,
 Five-Node Network, Iteration = 4,
 X Axis = Node 2, Y Axis = Node 1

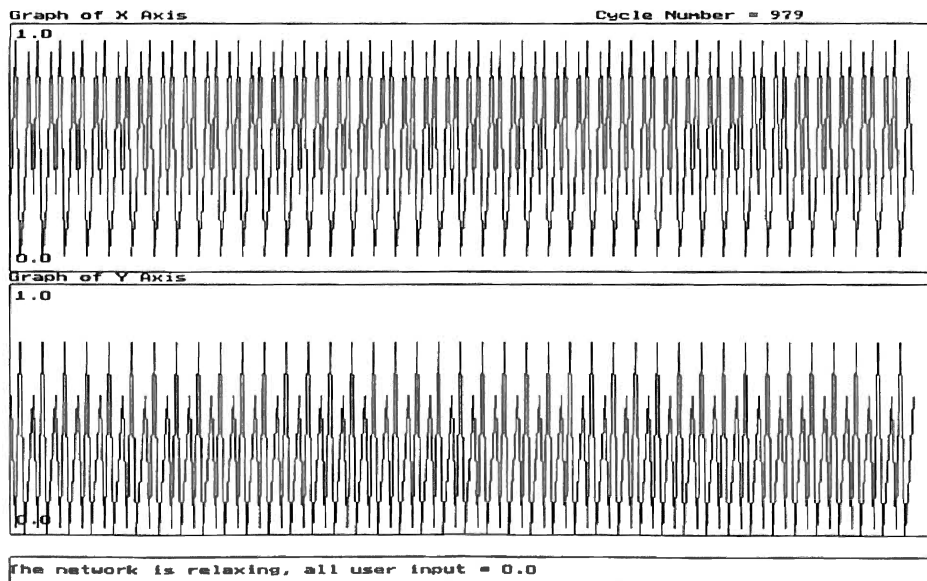


Figure 17 Time Chart for
 Five Point Cyclic Attractor,
 Five-Node Network, Iteration = 4,
 X Axis = Node 2, Y Axis = Node 1

In addition to this image which produced a cyclic attractor, another image was created that was tested for over two hundred thousand cycles and never repeated. Upon examination, this behavior was shown to be chaotic in nature. It should also be mentioned that it is not the just a result of numerical roundoff error since the cyclic attractor mentioned above was analyzed to ten digits accuracy for fifty thousand cycles, and once it settled down to a set path it never varied. One loop of the chaotic attractor is shown in figure 18 on page 37.

While loops similar to the one previously displayed are endlessly repeated, the number of cycles for each path is usually 11 but can reach 13 cycles. This longer loop occurs when a path like the one in figure 19 on page 37 is created.

To further illustrate the similarity of each path, the time series ('k' feature) graphs the X axis and the Y axis with respect to time as shown in Figure 20 on page 39. The graph clearly shows the similarity of each path giving further strength to the idea that it is caused by a chaotic attractor since they will often produce images with self-similar patterns. Each path in the time chart consists of a slow curve which corresponds to the slow movement near $(1,0)$, and then a sharp peak which corresponds to the rapid movement toward $(0,1)$. The time charts also pinpoint the paths containing longer lengths. These are shown as areas where there are two sharp peaks very close to each other. These points are the 8th, 11th and 14th major peaks in time

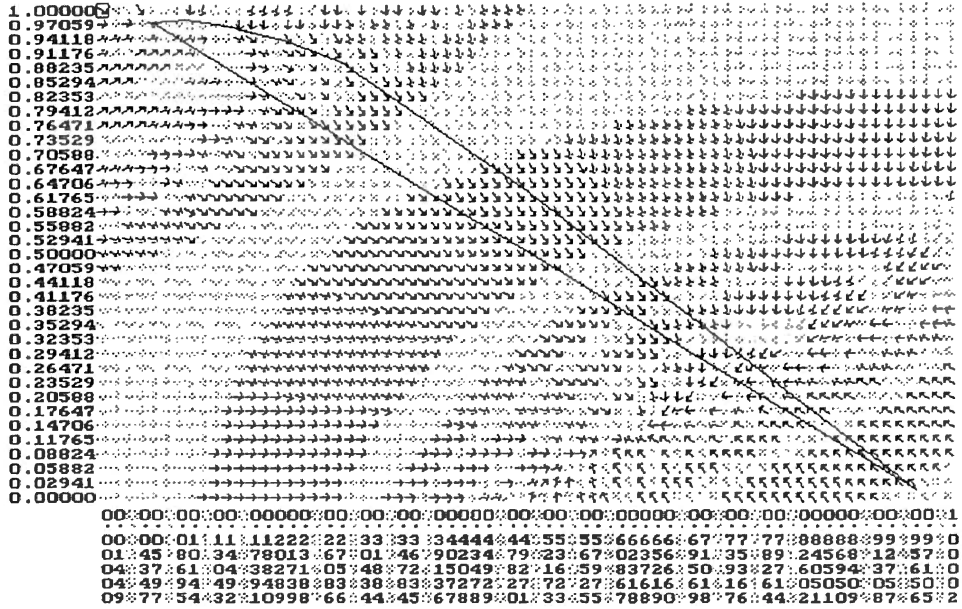


Figure 18 Short Pass of Chaotic Attractor,
Five-Node Network, Iteration = 4,
X Axis = Node 2, Y Axis = Node 1

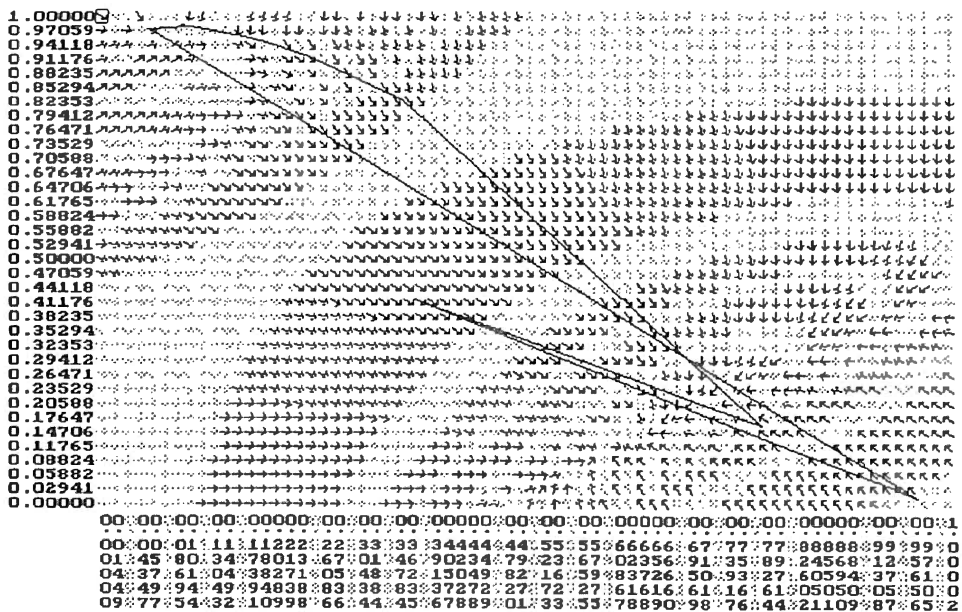


Figure 19 Longer Pass of Chaotic Attractor,
Five-Node Network, Iteration = 4,
X Axis = Node 3, Y Axis = Node 1

chart in figure 20 on page 39. Figure 21 on page 39 shows the image after 750 cycles, while figure 22 on page 40 shows the same image after 3000 cycles.

Interestingly enough, the only points the system relaxes to are the ones on the outside border of this curved shape on the previous images. To see this use the 'K' feature to provide the same view of 'Chaotic' imaging, except that a dot is placed only where the system relaxes to. No lines are drawn at all. Figure 23 on page 40 is the image after about 6000 cycles.

Notice that the top of the curve is more densely traced than the bottom. Each point on the lower part of the curve was drawn when the network followed a longer than normal path similar to the one shown in Figure 19. While the outline took about 6000 cycles before it looked fairly complete, the top part was virtually finished after just a few hundred cycles. This is easily explained by looking at the time chart (Figure 20 on page 39) which shows that the system was following this upper curve almost all the time. These ventures, which usually came from the (1,0) region, always resulted in a rapid return to this area, hence the double peak. These two curves represent two distinctly different types of behavior between which the network seems to bounce back and forth. The upper is followed for several steps, while the lower curve is not, so that the network is constantly returning to the upper curve whenever it finds itself on the lower one.

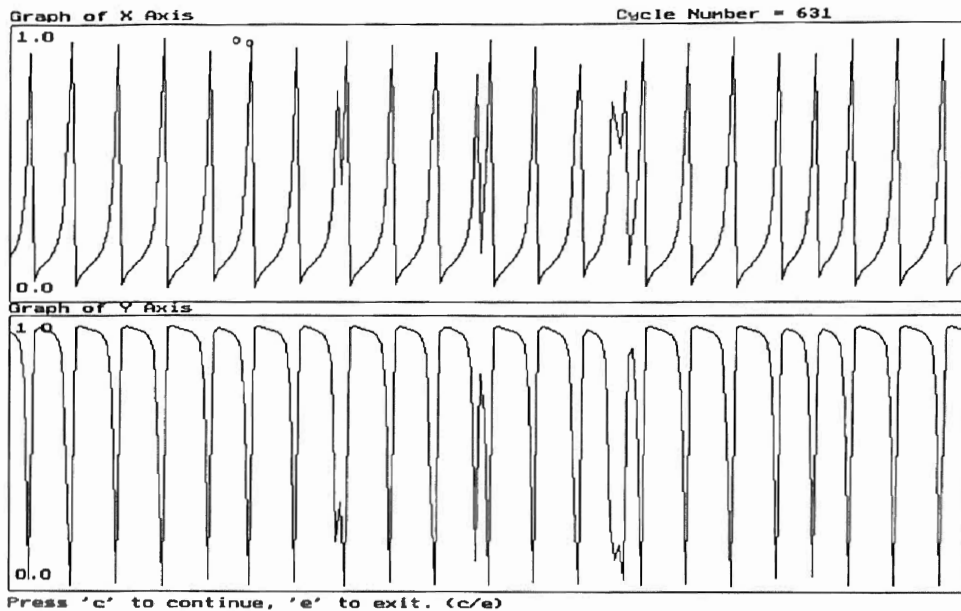


Figure 20 Time Chart of Chaotic Behavior,
 Five Node Network, Iteration is 4,
 X Axis = Node 3, Y Axis = Node 1

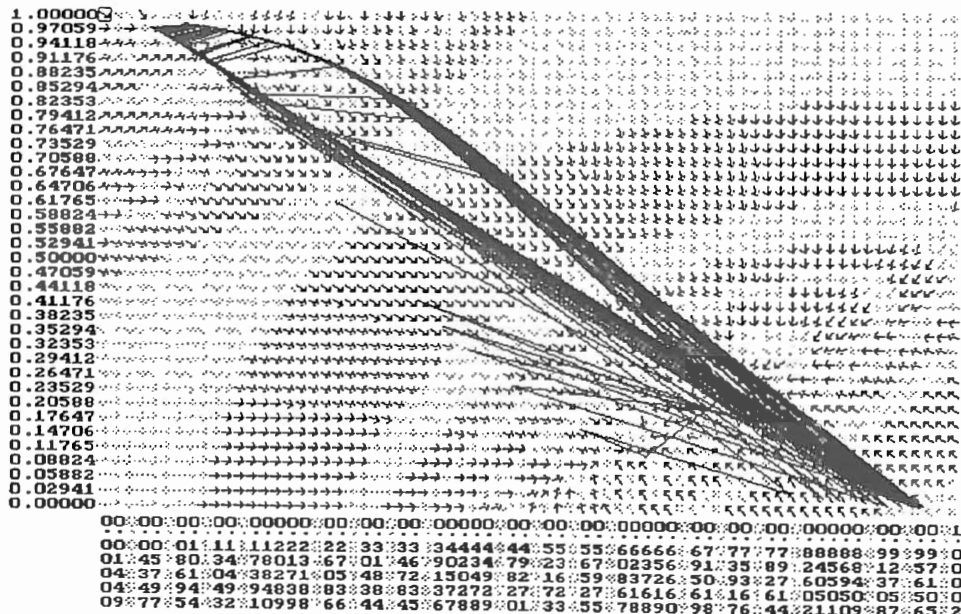


Figure 21 750 Paths of Chaotic Attractor,
 Five-Node Network, Iteration is 4,
 X Axis = Node 3, Y Axis = Node 1

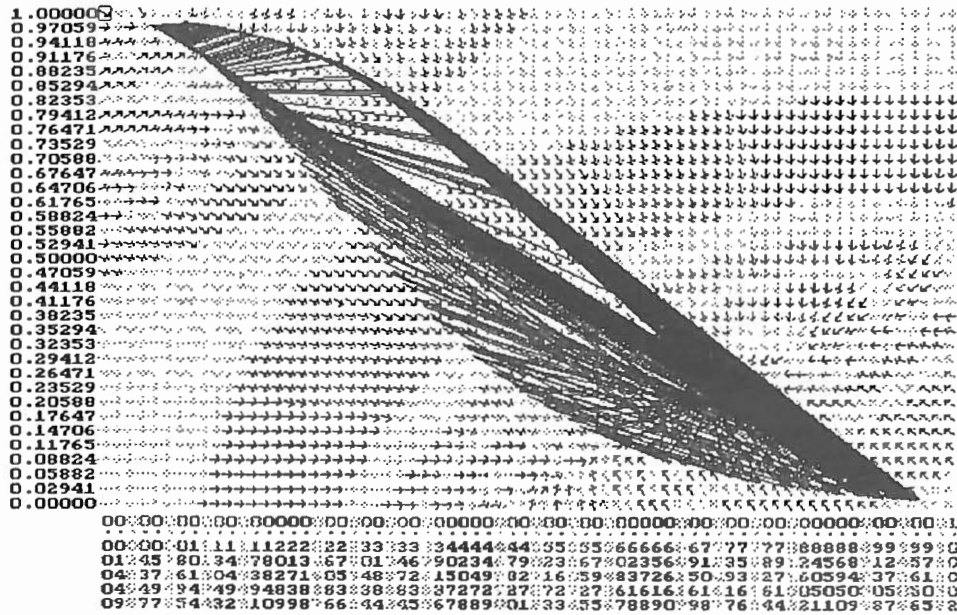


Figure 22 3000 Paths of Chaotic Attractor,
Five-Node Network, Iteration = 4,
X Axis = Node 3, Y Axis = Node 1

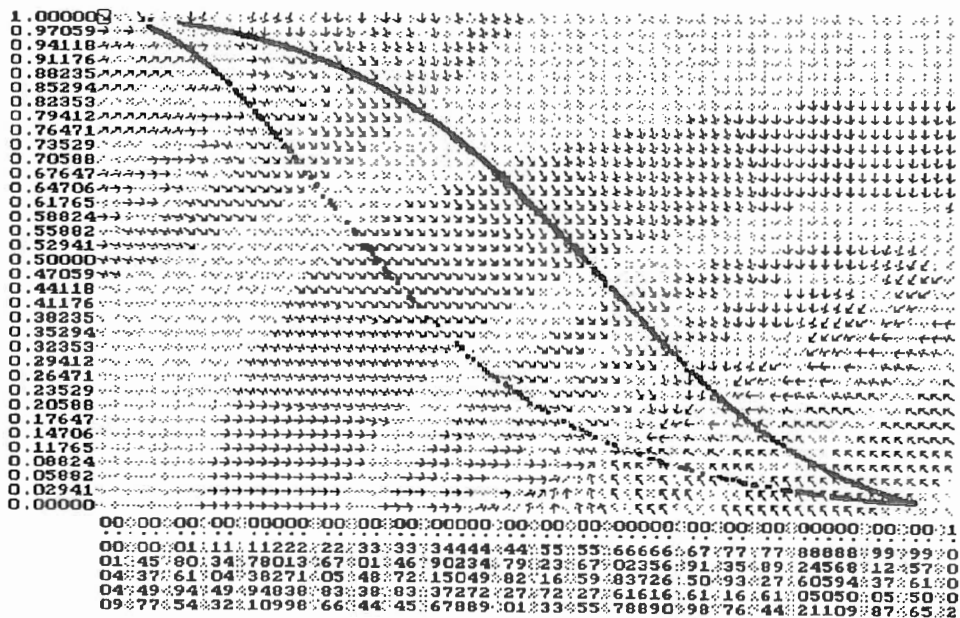


Figure 23 6000 Points of Chaotic Attractor,
Five-Node, Network, Iteration = 4,
X Axis = Node 3, Y Axis = Node 1

Interesting effects can be seen when driving forces are applied to the network. A driving force is one that is added on to the other inputs before the squashing function is applied. Thus, the sequence of events for a network is:

- A) Initially set all node outputs to zero.
- B) Sum the weighted inputs from the X axis, the Y axis and the Bias (which is always just 1.0).
- C) Add to this sum the driving input for each node.
- D) Apply the squishing function $1 / (1 + e^{-(\text{total_input})})$

The following shows the effects of applying a driving force on an otherwise chaotic system. As the input increases, sudden changes occur to the network.

Figures 24 and 25 on page 42 show the paths and points of a driving force of 0.571468 to both nodes 1 and 3.

In Figures 26 and 27 on page 43 the driving force is increased to 0.580247. The system is still in chaos, but the points show a sharply decreased range of attractors.

In Figures 28 and 29 on page 44 the driving force becomes 0.582990 and the system moves from chaos to a 64 point cyclic attractor.

As the driving force is increased, the number of attracting points is repeatedly reduced by a factor of 2, from 64 to 32, then to 16, then to 8 and finally to 4. Since these images are quite similar, only the final state is shown in Figures 30 and 31 on page 45 as the driving force to both of the graphed nodes is increased to 0.64 and the number of attractors is reduced to 4.

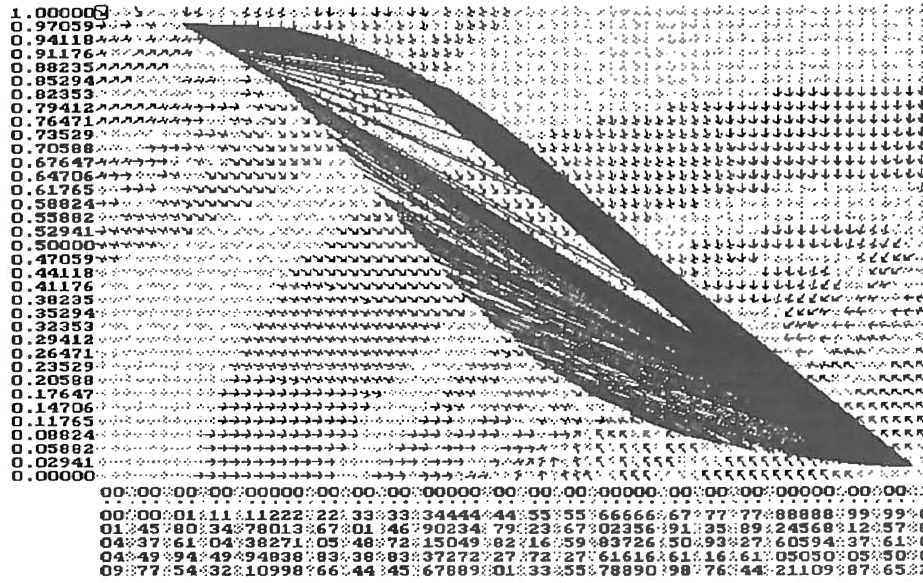


Figure 24 Path of Chaotic Attractor,
 Five-Node Network, Iteration = 4,
 X Axis = Node 3, Y Axis = Node 1,
 Driving Force = 0.57147

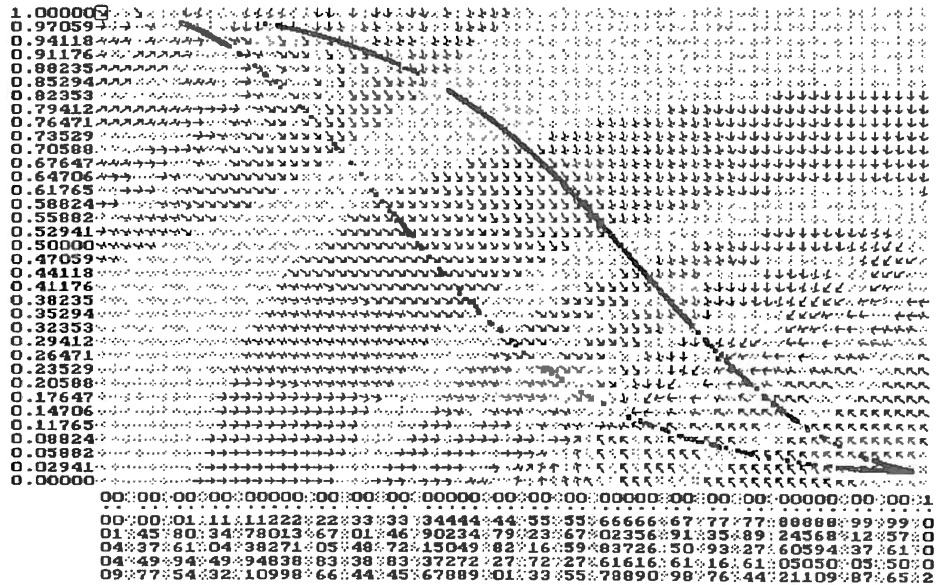


Figure 25 Points of Chaotic Attractor,
 Five-Node Network, Iteration = 4,
 X Axis = Node 3, Y Axis = Node 1,
 Driving Force = 0.57147



Figure 26 Path of Chaotic Attractor,
 Five-Node Network, Iteration = 4,
 X Axis = Node 3, Y Axis = Node 1,
 Driving Force = 0.58025



Figure 27 Points of Chaotic Attractor,
 Five-Node Network, Iteration = 4,
 X Axis = Node 3, Y Axis = Node 1,
 Driving Force = 0.58025

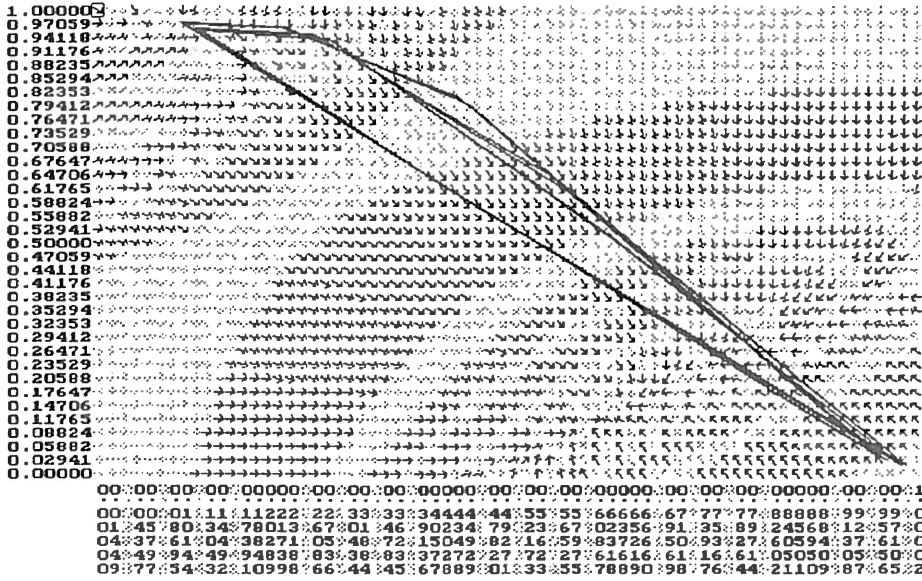


Figure 28 Path of Chaotic Attractor,
 Five-Node Network, Iteration = 4,
 X Axis = Node 3, Y Axis = Node 1,
 Driving Force = 0.58025



Figure 29 Points of Chaotic Attractor,
 Five-Node Network, Iteration = 4,
 X Axis = Node 3, Y Axis = Node 1,
 Driving Force = 0.58025

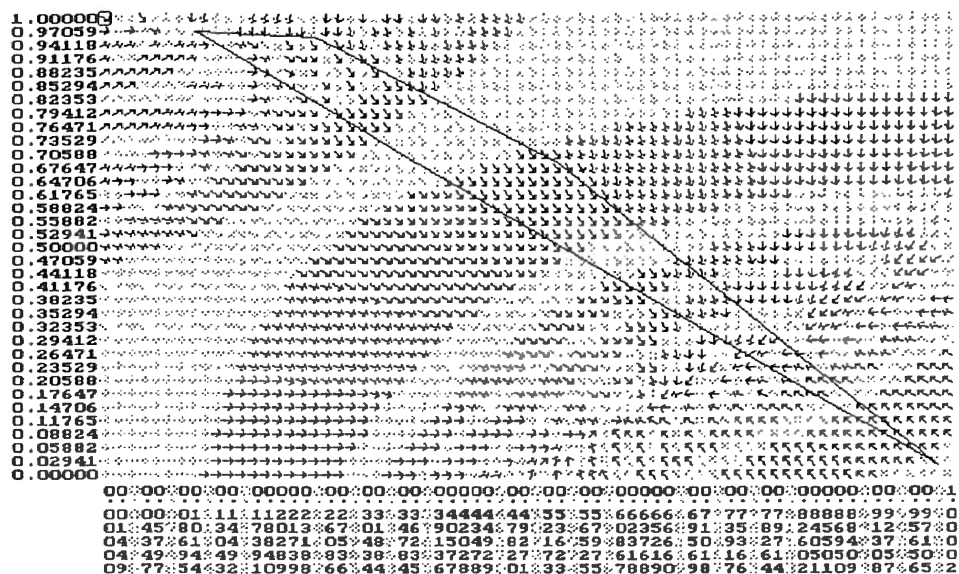


Figure 30 Path of 4 Point Cyclic Attractor,
Five-Node Network, Iteration = 4,
X = Node 3, Y = Node 1,
Driving Force = 0.64000



Figure 31 Points of 4 Point Chaotic Attractor,
Five-Node Network, Iteration = 4,
X = Node 3, Y = Node 1,
Driving Force = 0.64000

It should be noted that this behavior of the number of attractors being reduced by a factor of two is consistent with chaotic bifurcation where the number of attractors repeatedly doubles until some point at which the system becomes chaotic. Such chaotic systems have regions where first an attractor of period three is found and then doubled until chaos results, then after a time a cyclic attractor of period four is found and it also doubles until chaos sets in. This is then repeated for several other numbers. The system being examined shows these behaviors. If the driving force is increased from 0.64 (four cycle attractor) to 0.91, chaos sets in again. Increasing it further to 0.96 and 1.00 gives cyclic attractors of 6 and 3 cycles respectively. In addition to this, systems with cyclic and even some with point attractors can be pushed to chaotic behavior. As an example, a time chart of the five cycle attractor mentioned above (Figure 16) is displayed here with a driving input of 1.177777777777 to both graphed nodes. The result is chaos as can be seen in figures 32 and 33 on page 47.

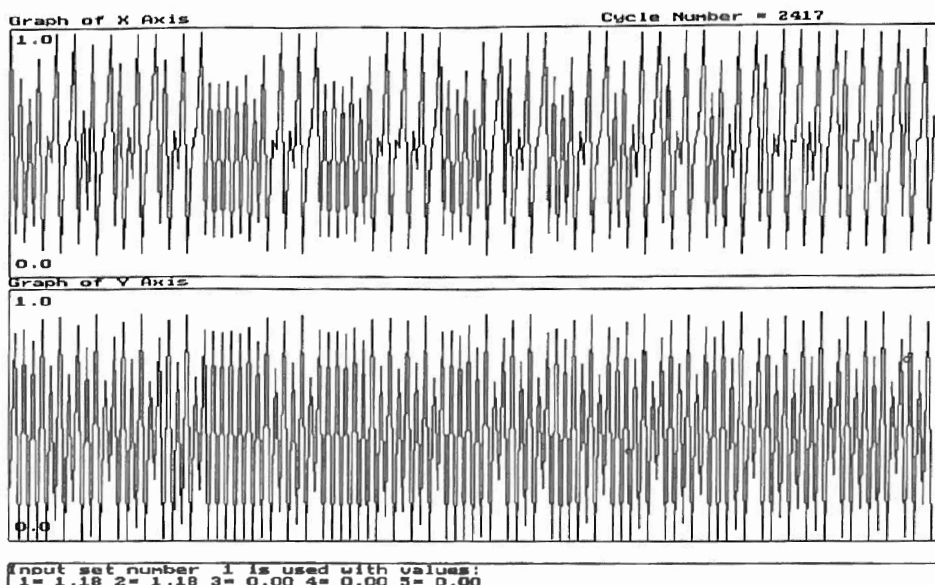


Figure 32 Cyclic Attractor Pushed to Chaos,
 Five-Node Network, Iteration = 4,
 X = Node 2, Y = Node 1,
 Driving Force = 1.1777777777

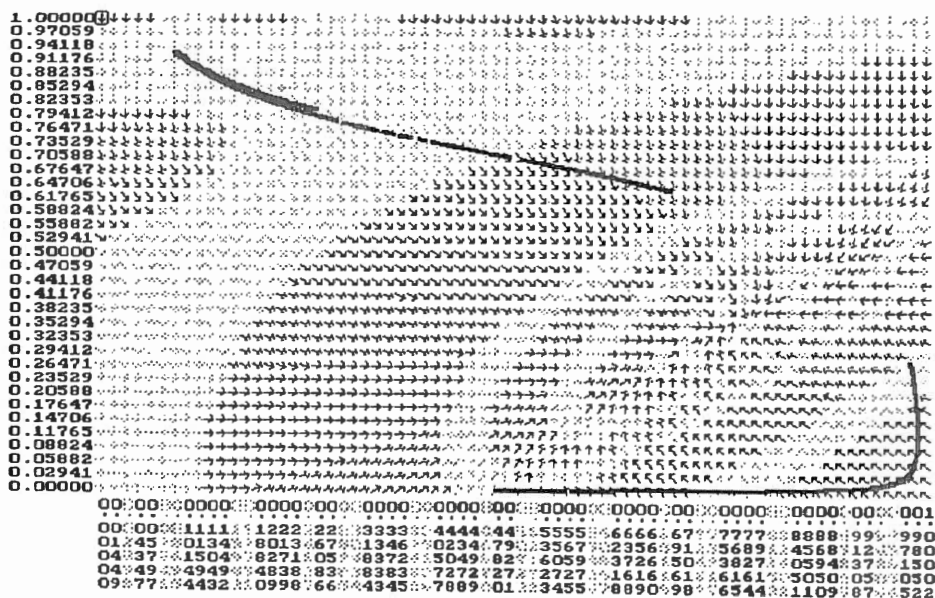


Figure 33 Cyclic Attractor Pushed to Chaos,
 Five-Node Network, Iteration = 4,
 X = Node 2, Y = Node 1,
 Driving Force = 1.1777777777

Not all of the images can be affected in this way. Many have only a point attractor, and no amount of input appears change that. At most, the point attractor simply moves around in response to the new input. In general, interesting changes in behavior occur only when one of the following changes occurs the to eigenvalues. They go from real to complex, from less than one to greater than one, or from positive to negative (or just the opposite). Since the eigenvalues determine how the neighborhood will be mapped as it jumps along with the point in question, the way they behave is important in understanding when change in behavior occurs.

CHAPTER VI

CONCLUSIONS

Classically, the study of dynamical systems has only tried to look at the critical points of a system, and not deal with the entire space. This came about as a result of the impossibility to explore in detail a high dimensional space, but does not mean that such a study could not prove interesting. Only since computers have been available could one explore a system's behavior in more detail. This project shows the use of visualizing the behavior of a neural net by clearly showing various behaviors that otherwise would have likely gone unnoticed and allowing another way to view the behavior of the network.

Further Study

A few further topics in are suggested by this project. For example, a more detailed study of the effects of driving inputs on the vector fields would be a logical next step in the study of these systems. Such a study could show how the vector fields as changed as different driving forces are applied.

Ideally, a training program could be devised that specifically trains for chaotic behavior when certain inputs are applied, and cyclic behavior when others are presented.

REFERENCES

- Arnaldo, Carlos M., Miller, William D. (1990). "The Geometry of Backpropagation Training: Visualization, Heuristics, and Theory". Oklahoma State University.
- Gleick, James, (1987). CHAOS Making a New Science Viking Penguin.
- Golub, Gene H. and Van Loan, Charles F. (1989). Matrix Computations. John Hopkins University Press.
- Hagar, William W. (1988). Applied Numerical Linear Algebra Prentice Hall.
- Helman, James and Hesselink, Lambertus. (1989). "Representation and Display of Vector Field Topology in Fluid Flow Data Sets". Computer, August. pages 27-36.
- Miller, William David, (1991). "Homotopy Analysis of Recurrent Neural Nets". Oklahoma State University.
- Peitgen, H. O., Richter, P. H. (1986). The Beauty of Fractals Images of Complex Dynamical Systems. Springer-Verlag Berlin Heidelberg.
- Pineda, Fernando J. (1991). "Dynamics and Architecture for Neural Computation", Journal of Complexity, 4, 216-245.
- Tesauro, G., Wejchert, J. (1991) "Visualizing processes in neural networks", IBM Journal of Research and Development, vol. 35, No 1/2, January/March. 244-253.
- Wasserman, Philip D. (1989). Neural Computing Theory and Practice. Van Nostrand Reinhold, 1989.
- Williams, Ronald J., Zipser, David. (1989). "Experimental Analysis of the Real-time Recurrent Learning Algorithm", Connection Science, vol. 1, No. 1. 87-111.

APPENDIXES

APPENDIX A

STARTING SCREEN

When the program starts up, the first screen presents four options which are as follows:

- 1) Load a fixed point or weight file: These files are created by the user to describe a network either by specifying the desired fixed points or by supplying the weights of a previously created network. Once the program displays the image of the network, it can be saved to a file for use latter. The format for both types of files will be discussed in detail in Appendix E.
- 2) Load an image file: This is the type of file discussed above that holds the data needed to display the network's image and allow interaction. It primary advantage is that once the image is calculated (which may take 30 minutes or more), it can be saved and restored in a matter of seconds.
- 3) Save a series of images: This option is basically just a batch version of the first option. A file is given that contains the names of all fixed point or weight files to be displayed and a corresponding file name to save the image to. Each the image for each input file is then created and saved as specified. Since several files would take a

considerable amount of time to calculate this approach allows it to proceed without any user interaction. The files can then be quickly viewed later. The format for a file that saves a series of images is described in detail in appendix E.

4) Load an image group: This allows a user to create a list of already calculated images and have the ability to go quickly forward and backward through the list. The format for this file is described in detail in appendix E.

APPENDIX B

DISPLAY AND DISPLAY OPTIONS

The display is divided into two parts. The main section is a grid of up to 70 by 35 points with an arrow indicating the direction of the network vector field at that location. The colors indicate the magnitude of the vector. For reference sake, the coordinates of the displayed image area are listed in rows on the left hand side as well as in columns just below the image. The column numbers are displayed in several colors for better readability.

The second part is used to display most of the user interaction information. This part is located below the grid section and usually is broken up into two parts. The left side shows the eigenvector and eigenvalues in both numerical and graphical form for the current location of the cursor. The right side shows the same information for the location that the network relaxes to when the 'fall' or 'move' feature is used. This area is also used when the network is pushed with the 'push' option. In addition to the above, certain interactive options either display additional information or require input from the user. In

this case the needed information and prompts are also displayed in the lower display area, temporarily erasing the eigenvector and eigenvalue information. Each of the features mentioned here are described below in the Appendix on Viewing/Interaction Options.

Clearly an important part of the project is to be able to move around the display screen, viewing the eigenvalues and vectors as you go and using the 'fall' feature to see where the network will relax to at that point. To accommodate this, there are several ways to move. First, the arrow keys move one step in their respective directions, and with the shift key down, they move ten spaces for faster movement. Some other keys on the key pad also have uses. In particular, the HOME key moves to the upper left corner, while the Page Up key moves to the upper right corner. Similarly, the END key moves to the lower left and the Page Down moves to the lower right corner. These keys are summarized below.

Since knowing where the attractors are is important in determining the makeup of the network, pressing the 'a' key will toggle the display of the attractors that were found in this screen. Each attractor will have a different identifying color. It should be noted that the program might not find all attractors for a given image. Sometimes only a small area would 'relax' close enough to an attractor for it to be noticed. Such attractors will only be found if one of the displayed points happens to be in an area that

relaxes to the attractor. This happens often with saddle points that have attractions on only one axis. Resulting in a topology where the chance of any plotted point ever relaxing close enough to the attractor to register it is small.

To help in getting more exact locations of the attractors, another their coordinates can be displayed by just pressing 'c'. Each location will be given with an identifying color corresponding to the attractor's own color.

As another way of understanding the topology of the image, the line option draws a series of points which have the least movement in the x axis and another for the least movement on the y axis. The x and y lines formed by these points can only intersect at fixed points, since their crossing would imply that there is no motion in either the x or y direction (and therefore no motion at all), which is by definition a fixed point. To toggle the x axis line, press the 'x' key, to toggle the y axis line, toggle the 'y' key. Both lines can be toggled simultaneously by pressing the 'l' key.

The combination of actions shown demonstrates how the validity of the network was established by allowing the network's image to relax from various locations, and comparing the resulting outputs to the expected outputs. On a lower lever, it is this ability to let the network relax to the fixed points that allow the program to locate the

attractors in the first place.

There are two ways to relax the network. The first is to simply move the cursor with the arrow keys to the desired location and then press the 'f' key to allow the network to 'fall' toward the attractor of the current location's neighborhood.

At any time, the user can see what the output of all of the nodes by pressing the 'o' key.

It should be noted that this 'falling' process is not a continuous one, but rather a single jump that is determined solely by the topology of the starting point and has nothing to do with the terrain in between the starting and ending points of the jump. Since this is basically how neural nets on digital computers operate, this style does provide a useful and valid approach.

In addition to using the arrow keys for movement to any point on the display grid, the program allows the option of giving the exact coordinates for any input location. This is done using the 'M' key to enter the Set Move Mode. In this mode the current x and y input values are displayed, and the user is allowed to change them by pressing either 'x' or 'y' and then entering the new location. To exit this mode, press 'm'. Once out of this mode, the user can essentially 'Move' to that location and 'fall' one time step by pressing the 'm' key.

In addition to seeing where any given individual point relaxes to, the program also allows one to see where the

entire image as a whole behaves by entering the Time Mode. This mode will show when each point relaxes sufficiently close to an attractor by drawing a box around the point of the same color as the identifying color of the attractor. In this way, one can see the attractors for area of the image.

To enter the Time Mode, press the 't' key. Instructions will now appear below indicating that you can press 's' to step one item unit in the given direction. The initial direction is +1, but pressing '-' will change the direction to be -1. Similarly, pressing '+' (or '=') will change the direction back to +1. Each time the 's' key is depressed, a time indicator shows the current time step, and all points that settled to a fixed point at that time will have a box placed around them of the identifying color of the attractor that they settle to. If a point settles to one attractor at one time and then moves on to another, the color of the surrounding box will change to that of the new attractor's color to indicate that it has now relaxed to another attractor.

To exit the Time Step Mode, press the 't' key. The screen will automatically return to what it was before.

Another feature is the ability to 'push' the state of the network around by specifying user inputs to each node. This is done by pressing the 'i' key to enter the Set Input Mode. When in this mode, the inputs to a node can be adjusted by first typing the node number and pressing ENTER,

and the giving the new input value. To exit this mode, press the 'i' key again.

Once the inputs have been set, press the 'p' key to 'push' the network around. the eigenvalues and eigenvectors for the location pushed to will be displayed in the lower right side of the screen.

To quit the program, simply press 'q'. A prompt will ask if you wish to quit, press 'n' if you do not wish to quit, press 'y' if you do.

APPENDIX C

WORKING WITH IMAGES

Since the images produced by this program can take 30 minutes or more to be created (on a 386SX 16Mhz system), a set of image handling options are build into the program. These options include the ability to load an already saved image by pressing the 'L' key and then giving the file name (complete with path if it is not in the current directory). A feature to save an individual file is also available by pressing the 'S' key and then giving a file name.

The program also allows the user to load a series of images while the program is running. By pressing the 'G' key, the user can load a file containing the names of a series of images. Once a series is loaded, the user can move forward to the next image and backward to the prior image by pressing 'N' and 'P' respectively.

The ability to work with series of images has turned out to be useful since it allows the user to quickly and easily see several images and move between them. This coupled with the option to let the program create several images and save each to its respective file has been very useful in terms of essentially permitting a form of batch image creation which needs no user input once it is started.

APPENDIX D

DEALING WITH CHAOS

During the course of project development, signs of chaos were detected. Four tools have been added specifically to study this behavior. The features are:

- 1) 'C' Begin chaos line mode, this is also useful to display the path of cyclic attractor. It works by toggling between normal operation of the 'fall' feature, (which draws a small square to indicate the current state of the network), and a line mode which draws a line from the previous state to the current one. In this way a path is created showing how the state of the network moves.
- 2) 'K' This toggles a mode that works like normal 'fall' mode, except that the small rectangles are never erased, thus leaving a record of which points the network has relaxed to. This has the ability to produce a more simplified view than the 'C' feature. For example, use of this feature on the Lorenz attractor would eventually produce an image of the overall structure, while the 'C' option would show the path taken which would produce far too many lines to view the fine circular string-like structure.
- 3) 'F' This is basically a fast fall feature. Pressing 'F' at any time causes the network to relax repeatedly until

a key is pressed at which time it stops and displays how many relaxation cycles had passed. This is especially useful when the 'C' or 'K' modes are on since it will rapidly draw the lines or dots, thus providing a fast way to view the path or outline of a system. It can also check the stability of a cyclic attractor. Simply start the line mode, 'C', after the system has settled down to a stable set of points. If the attractor is truly cyclic, then once a full cycle has been drawn, nothing new should be displayed.

4) 'p' This is identical to the 'F' feature except that it performs the last push until a key is pressed.

5) 'k' This places the user in the time chart mode where two charts show the relationship of changes in the x and y axis as time increases. This part will show any periodic behavior as well as chaotic behavior where the system never repeats a prior path, but repeatedly comes close.

This mode works by displaying two charts, one with x graphed with respect to time, and the other with y graphed with respect to time. As they are displayed, the small rectangle used by the 'fall' feature is displayed to show the current state of the network. The program waits for a key to be pressed before moving on to the next time cycle, in this way one can move quickly through a great many cycles, or one can go single step to see how the graphs relate to some specific behavior in the network.

When the screen is filled, an option to continue or exit to the main program is given.

APPENDIX E

SUMMARY OF OPTIONS

ARROW KEYS

(10 steps) right	(shift) right arrow
(10 steps) left	(shift) left arrow
(10 steps) up	(shift) up arrow
(10 steps) down	(shift) down arrow
upper left corner	Home
upper right corner	Page Up
lower left corner	End
lower right corner	Page Down

VIEWING ATTRACTOR INFORMATION

attractors	a	colors	c	
lines	x-lines	x	y-lines	y
	both x and y	l		

SEEING WHERE A POINT RELAXES TO

fall	f
Show node outputs	o
Show time steps	t
INSIDE TIME SET	
Set step to +1	+ (or =)
Set step to -1	-
Set one time unit	s
Exit time mode	t
Set Move	M
INSIDE MOVE MODE	
Set x location	x (or X)
Then input location and press ENTER	
Set y location	y (or Y)
Then input location and press ENTER	
Exit Set Move mode	m (or M)
Move	m

PUSHING A NETWORK

Set inputs for push i

 INSIDE PUSH MODE

 Select node for user input n (or N)

 Then give the node number and press ENTER

 Then give the user input values and press ENTER

Set range r

 Give input range (helps find bifurcation points)

Push (with inputs set by i) p

WORKING WITH IMAGES OR SERIES OF IMAGES

Load image	L	Save image	S
Next image in series	N	Prior image in series	P
Load image group	G		

The format for each file type is listed below:

DEALING WITH CHAOS

Time Chart mode	k		
Toggle draw line mode	C	Toggle plot dot mode	K
Fast fall	F	Fast push	p

QUITTING THE PROGRAM

Quit q

APPENDIX F

INPUT FILE DESCRIPTIONS

WEIGHT INPUT FILE

NOTE: Data appears only after the first SPACE after '-->'

-->Fixed_or_Weight W

-->SIZE 3

-->ITERATIONS 1

-->External_Input_Number(Bias_counts_as_1) 3

NOTE: THERE MUST ALWAYS BE 3 INPUTS (X,Y,BIAS)

NODE_TO_NODE_WEIGHTS COL1 COL2 COL3

-->(ROW1) 0.1 0.2 -0.3

-->(ROW2) 0.3 0.0 -0.6

-->(ROW3) 7.8 7.8 -0.3

INPUT WEIGHTS FIRST XOR INPUTS (X then Y) THEN BIAS

X Y BIAS

-->(ROW1) -2.7 5.1 -5.4

-->(ROW2) -2.8 -6.1 5.8

-->(ROW3) -3.6 -0.1 -0.1

XAXIS

--> 0.0 1.0 70

YAXIS

--> 0.0 1.0 35

DATA INPUT FILE

This is created by the Save command, and its format is not important since it not meant to be viewed by a user.

FIXED POINT INPUT FILE

NOTE: Data appears only after the first SPACE after '-->'

--> F (Only the first letter matters, 'F' means Fixed Point)
(4 nodes in the network)

--> 4 (NUMBER_OF_NODES_(SIZE)) Note: Only the
(A path of two iterations in length is needed to go from a
node in the first layer back to a node in the first layer
again)

--> 2 (NUMBER_OF_ITERATIONS) first number after
There will be 2 fixed points (one for each layer)

--> 2 (NUMBER_OF_FIXED_POINTS) '-->' matters.
(The points have a dim of 2, one for each node in the layer)

--> 2 (DIMENSION OF POINTS)

(THE_FOLLOWING_ARE_THE_FIXED_VALUES)

(Fixed values are CAPITOL LETTERS)

EX-->	POINT	DIM	SIZE	VALUE1	VALUE2
	A	2		.33333	.33333

(A is the first fixed value)

--> A .333333 .333333

(B is the second fixed value)

--> B .666666 .75

(THE_FOLLOWING_IS_THE_LIST_OF_FIXED_VALUES_FOR_EACH_NODE)

(Nodes are assigned numbers)

Node 1 has as its fixed values the first element of A and B

--> 1 A 1, B 1;

Node 1 has as its fixed values the second element of A and B

--> 2 A 2, B 2;

--> 3 A 1, B 1;

--> 4 A 2, B 2;

(THE_CONNECTION_MATRIX)

example --> 0001 Here the value 1.0 is the weight FROM

example --> 0000 node #4 TO node #1

example --> 0000 (View this as a 4 X 4 matrix)

example --> 0000

--> 0011 (This is a 4 X 4 connection matrix)

--> 0011

--> 1100

--> 1100

(X range and intervals)

(0.00 is min, 1.00 is max MUST be in this order!)

(70 is max intervals)

--> 0.00 1.00 70

(Y range and intervals)

(0.00 is min, 1.00 is max MUST be in this order!)

(35 is max intervals)

--> 0.00 1.00 35

LOAD SERIES FILE

First Letter Must be 'L'

The next line Must start with the number of images to in the series. Then each line starts with the name of the image files. They will be loaded in the order given.

L

2

r312i3.dat
r231i1.dat

SAVE SERIES FILE

The first letter must be 'S', the next line must start with the number of images in the series. After this, each line should start with the name of the weight or fixed point file to be created, followed by the name of the image file it is to be saved as. The end of the line has two numbers, the maximum number of fixed points to look for at each individual point (5 is given, though no network has found more than 2), finally, the maximum number of 'relaxations' per points is given. If this number is too large (> 100) the image may take hours to compute, if it is too small (< 5) the image may be computed quickly, but it will find many false attractors. The recurrent networks tested seemed to work fine with this value set to 20 or 25.

The following is an example of a file designed to calculate and save a series of images.

```
S
2
r312i3.m    r312i3.dat  5  25
r231i3.m    r231i1.dat  5  25
```

The following are the keys for the file options listed above. The user is also given the chance to enter one of these modes at start up. The initial screen, gives choice of:

- 3) Load image (same as 'L')
- 4) Load image series (same as 'G')
- 5) Load a weight or fixed point file (no key stroke equivalent is offered since this often takes so long to calculate that it should be done in the batch mode style. See option 6))
- 6) Save an image group. This is the batch style of computing and saving several images, allowing the computer to grind away unattended for many hours.

VITA 2

Bennett S. Carter

Candidate for the Degree of
Master of Science

Thesis: GRAPHICAL REPRESENTATIONS OF ATTRACTOR
AND VECTOR SPACE FOR NEURAL NETS

Major Field: Computer Science

Biographical:

Personal Data: Born in Spokane, Washington, September
24, 1961, the son of Robert T. and Sue Ann Carter.

Education: Graduated from Lewis and Clark High School,
Spokane, Washington, in June 1980; received
Bachelor of Arts Degree in Physics from Whitman
College at Walla Walla Washington in May 1984;
completed requirements for the Master of Science
degree at Oklahoma State University in May, 1993.