LOGIC OF COMPUTERS GROUP

Department of Computer and Communication Sciences
2080 Frieze Building
The University of Michigan
Ann Arbor, Michigan 48104

THE UNIVERSITY OF MICHIGAN
COLLEGE OF LITERATURE, SCIENCE, AND THE ARTS
Computer and Communication Sciences Department

ANALYSIS OF A BASE MODEL AND A LUMPED MODEL OF A NETWORK OF NEURONS

by

Aggarwal, S.adhir, Bethke, A.D., and Zeigler, B.P.

January 1975

Technical Report No. 165

with assistance from

National Science Foundation
Grant No. DCR71-01997

Computer simulation models are increasingly being used to study complex systems that have a large number of components and many interrelationships. In general, such models are largely "lumped versions" of a more complicated hypothesized "base model". Zeigler [12] provided the conceptual foundation for this viewpoint of computer simulation by considering the triple of a base model, which captures all structural beliefs about the real system, the lumped model, which is the "simplified version" of the base model and is actually to be simulated, and the computer on which the simulation is to be performed.

In this paper, we investigate the relationship between a base model and a lumped model. The base model is a probabilistic model of a network of neurons. The lumping procedure yields a lumped model that is deterministic. In the limiting case of a base model that has an infinite number of neurons and neighbors per block, this lumping procedure yields an exact structure morphism between the base model and the lumped model. When we consider only a finite number of neurons and neighbors per block, the structure morphism becomes only approximate. In order to investigate to what extent base model behavior could still be predicted by the lumped model, a computer simulation of the two models was implemented. The aim was to show that the lumped model retained a great deal of information about base model behavior. Since the base model was probabilistic, it is clear that this information would be of a statistical nature. Furthermore, behavior on an identical time scale would be highly unlikely. We would hope, however, to be able to predict patterns of long term behavior and perhaps make accurate short term predictions given the base model state.

Investigations similar to ours are being carried out by a group at Syracuse University. Harth et al. [6], Wong and Harth [10], Anninos et al. [1], and Csermely et al. [4] have explored cooperative

phenomena in neural networks and have proposed an approach they call netlet theory. Netlets are simply collections of neurons that have statistically identical properties and are connected in a uniform (though probabilistic) manner. Although the netlet model is a deterministic one, the assumptions involved in the way neurons are interconnected allow one to probabilistically predict the firing level at time t+1 given the firing activity of the net at time t. The approach adopted by this group is a detailed analysis of the netlet dynamics using the strong regularity conditions. It has many similarities in our work in that we also attempt a detailed analysis of a block of neurons. However, ours is a probabilistic base model, there are differences in the regularity conditions and we study a lumped model with a view to predicting the base model. A detailed description of our base model-lumped model pair follows.

Base Model

Components

NEURONS--elements modeling basic behavior assumed to be characteristic of real brain cells.

INPUT·WIRES--sources of external net excitation.

Descriptive Variables

For each NEURON:

RECOVERY·STATE--with range the non-negative integers

RECOVERY·STATE = i indicates that i time units have elapsed since the NEURON last fired; thus i=0 means the NEURON is now firing.

NOISE--with range the real numbers;(when NOISE = r the actual threshold for firing of the NEURON is

THRESHOLD (i) + r where THRESHOLD (i) is the threshold value characteristic of RECOVERY·STATE i. NOISE is a random variable generated for each NEURON.

STRENGTH--with range the real numbers; STRENGTH = x indicates that the sum total of all NEURON inputs is x.

FIRING·STATE--with range {0,1}; 1 means the NEURON is firing (emitting a pulse), 0 that it is not firing.

For each INPUT·WIRE

INPUT·LEVEL--with range the real numbers;
INPUT·LEVEL = x indicates that the excitation level on the INPUT·WIRE is x (measured in the same units as STRENGTH).

For each BLOCK (designated set of NEURONS)

and for each RECOVERY·STATE·i:

%·IN·STATE·i--with range the rationals in [0,100];
the percentage of NEURONS in the BLOCK in RECOVERY·STATE i.

PARAMETERS:

For each BLOCK:

NOISE·DISTRIBUTION--the probability distribution for random variable NOISE.

THRESHOLD--a real valued function with RECOVERY·STATE as argument, giving the minimum strength needed to fire a NEURON in the absence of NOISE.

For each NEURON:

NEURON·NEIGHBORS--with range the subsets of NEURONS indicates the NEURONS whose outputs impinge on the NEURON.

EXTERNAL·NEIGHBORS--with range the subsets of INPUT·WIRES indicates the external inputs impinging on the NEURON.

and for each NEIGHBOR (NEURON or EXTERNAL) of the NEURON:

SYNAPS--with range the reals; determines the influence that a NEIGHBOR has on the NEURON: positive indicates facillatory, negative indicates inhibitory.

Component Interaction

1. Each NEURON receives inputs from other NEURONS called NEURON·NEIGHBORS as well as from external sources called EXTERNAL·NEIGHBORS. We just refer to NEIGHBORS when the context is clear.
2. The NEURONS are grouped into disjoint sets called BLOCKS. This partitioning of the NEURONS places the following restrictions on the possible NEURON·NEIGHBORS of a NEURON:

   Suppose that a NEURON α belongs to BLOCK i.

   a) If α has some NEIGHBORS in BLOCK j (which may equal i) then every NEURON in BLOCK i must have at least one NEIGHBOR in BLOCK j.

   b) The SYNAPS weight is the same for the outputs of all NEIGHBORS of α from the same BLOCK. The NEIGHBORS of α coming from BLOCK j all have the same "influence" on it, but this may be different from that of the neighbors of α coming from BLOCK k ($k \neq j$).

   c) BlOCK j exerts the same "influence" on each NEURON in BLOCK i. We measure the "influence" of BLOCK j on α by the product of the number of α's NEIGHBORS from BLOCK j and their common SYNAPS value. This product is to take the same value for all NEURONS in BLOCK i, so that the "influence" of BLOCK j on each NEURON in BLOCK i is the same.

d) The characteristics of all NEURONS in a BLOCK, namely the THRESHOLD function and NOISE probability distribution, are the same.

e) The EXTERNAL·NEIGHBORS and their SYNAPS weights are the same for all NEURONS in a BLOCK.

3. Each NEURON operates at each time step as follows:

a) the STRENGTH of the input volley is computed by adding together the outputs (which may be 0 or 1) of all the EXTERNAL·NEIGHBORS and NEURON·NEIGHBORS of α weighted by their respective SYNAPS values.

b) If this STRENGTH is great enough, the NEURON will fire, i.e. will put out a 1 for the next time step; otherwise it will put out a 0. The 1/0 output can be thought of as a pulse/no pulse sent out by the NEURON, with only the pulse being able to influence the activities of another NEURON.

c) The minimum STRENGTH required to fire is dependent on the RECOVERY·STATE of the NEURON and NOISE. Generally, the longer it has been since the NEURON last fired, the easier it is to fire, indeed if it has just fired it may be impossible to fire for some time (this period is called the absolute refractory period). This fact is formalized in the shape of the THRESHOLD function. This function determines for each RECOVERY·STATE value i (time since last fired) a value, THRESHOLD (i) which can be thought of as the minimum strength required if there were no noise.

It is also assumed that the NEURON is perturbed by additive noise independently generated at each step for each NEURON from the NOISE probability distribution.

Thus if the NEURON is now in RECOVERY·STATE i, the minimum STRENGTH required to fire it is THRESHOLD (i) plus the sampled

value of NOISE.

d) If the NEURON fires, its RECOVERY·STATE is set to 0, otherwise it is incremented by 1. Thus, as implied, it records the time elapsed since the last firing of this NEURON.

Lumped Model

Components

BLOCK'S--each BLOCK' represents in lumped form the behavior of a corresponding BLOCK of base model NEURONS. (' indicates the correspondence).

INPUT·WIRE'S--each INPUT·WIRE' of the lumped model represents a corresponding INPUT·WIRE of the base model.

Descriptive Variables

For each BLOCK':

For each i = 0,1,2,...:

%·IN·STATE·i' -- with range [0,100]; %·IN·STATE·i' = q predicts that q/100 of the base model NEURONS in BLOCK are in RECOVERY·STATE·i. Thus %·IN·STATE·i' of BLOCK' corresponds to %·IN·STATE·i of BLOCK. We use the term PROBABILITY·IN·STATE·i interchangeably with %·IN·STATE·i'.

STRENGTH' -- with range the real numbers; STRENGTH' = x' indicates that the sum total of inputs to BLOCK' is x'.

For each INPUT·WIRE':

INPUT·LEVEL'--with range the real numbers.

PARAMETERS:

For each BLOCK':

BLOCK'·NEIGHBORS -- with range the subsets of BLOCK'S; specifies the BLOCK'S which send input to the BLOCK'.

EXTERNAL'·NEIGHBORS -- with range the subsets of INPUT·WIRES'.

and for each NEIGHBOR (BLOCK' or EXTERNAL'):

WEIGHT' -- with range the real numbers; determines the influence of the NEIGHBOR on the BLOCK'.

NOISE·DISTRIBUTION' -- a cummulative density function (c.d.f.). (will be the c.d.f. of NOISE)

THRESHOLD' -- a function from the non-negative integers to the reals (to be identical with THRESHOLD).

Component Interaction

1. Each BLOCK' receives input from other BLOCK'S (called BLOCK'·NEIGHBORS) and external sources (called EXTERNAL'·NEIGHBORS).
2. Each BLOCK' at each time step operates as follows:

   a) The STRENGTH' of the input to the BLOCK' is computed as the weighted sum of all NEIGHBOR (BLOCK' and EXTERNAL') outputs.

   b) Using this STRENGTH' and the current values of the %·IN·STATE·i' (one for each RECOVERY·STATE i) the next values of the %·IN·STATE·i' variables are computed. For each i, the current value of %·IN·STATE·i' is multiplied by 1 - F(x - THRESHOLD'(i)) to obtain the new value of %·IN·STATE·i+1'. Here, F is the NOISE·DISTRIBUTION (i.e., the cumulative distribution function), and x is the value of STRENGTH'. This accounts for the fraction of base model NEURONS in RECOVERY·STATE·i which have not fired (hence go to i+1); the remainder then, are those that do fire (hence go to RECOVERY·STATE·0) and the lumped model reflects this as a sum of all such contributions to obtain the new %·IN·STATE·0'. We will discuss the derivation of this lumped transition rule after we define the correspondence between the base and lumped structures.

c) The output of BLOCK' is just %·IN·STATE·0', which, if the lumped model is valid, is the percentage of NEURONS in the BLOCK which are firing i.e. outputting a pulse (1). (The rest are in the various RECOVERY·STATES and since they output a 0, will not influence any activity at the next time step.)

Correspondences between lumped and base model

Components

Each BLOCK' (lumped model component) corresponds to a BLOCK (partition class) of NEURONS (base model components).

Descriptive Variables

For each BLOCK'

For each i = 0,1,2,...

%·IN·STATE·i' = %·IN·STATE·i

STRENGTH' = STRENGTH of any NEURON in the corresponding BLOCK (this STRENGTH is the same for each NEURON in a BLOCK by base model postulate 2.)

PARAMETERS:

INPUT'·WIRES = INPUT·WIRES

For each BLOCK':

BLOCK'·NEIGHBORS = the set of BLOCK's whose corresponding BLOCKs influence NEURONS in the BLOCK, (corresponding to BLOCK').

EXTERNAL'·NEIGHBORS = EXTERNAL·NEIGHBORS of any NEURON in the BLOCK (a BLOCK constant according to Postulate 2.)

THRESHOLD' = THRESHOLD

NOISE·DISTRIBUTION' = NOISE·DISTRIBUTION for any NEURON in the BLOCK (postulate 2, again).

and for each NEIGHBOR (BLOCK' or EXTERNAL')

WEIGHT' = the product of the number of corresponding NEIGHBORS and their common SYNAPS values (postulate 2) of any NEURON in BLOCK .

for each INPUT·WIRE':

INPUT·LEVEL' = INPUT·LEVEL

A structure morphism is a mapping from the state set of the base model onto the state set of the lumped model that preserves the transition structure of the models when they are viewed as automatons. That is, if δ is the transition function of the base model, and δ' is the transition function of the lumped model, then $h \circ \delta = \delta' \circ h$ where h is the structure morphism. It can be shown [13] that if the recovery states of the neurons in the base model are independent and identically distributed initially, then in the limiting case of an infinite number of neurons and neighbors per block, there exists a natural structure morphism from the base model to the lumped model. (If the state set of the base model is the cross product of the recovery states of each neuron, then %·IN·STATE·i is a function of this state set. Mapping %·IN·STATE·i to %·IN·STATE·i' defines h). Intuitively, this is because the expected value of input to each neuron is the same (per block, because of an infinite number of neighbors) and the noise distribution will be exactly as NOISE·DISTRIBUTION (because of an infinite number of neurons).

We shall now derive the form of lumped component interaction.

For each NEURON, the following conditional distribution can be computed: $P_{i,i+1}(x)$ = the probability that the NEURON goes next to RECOVERY·STATE·i+1 (does not fire) given its present RECOVERY·STATE=i and its input STRENGTH=x. Now, if F(Z) is the NOISE·DISTRIBUTION then

$$P_{i,i+1}(x) = \text{Prob}\{x < \text{THRESHOLD}(i)+Z\}$$

$$= 1 - F(x - \text{THRESHOLD}(i))$$

Also, since a NEURON can only go to RECOVERY·STATE·0 if it does not go to i+1, we have

$$P_{i,0}(x) = 1 - P_{i,i+1}(x) = F(x - \text{THRESHOLD}(i))$$

This conditional distribution can be presented in the form of a Markov matrix (ignoring the input x = x(t) which depends on the time step).

$$P = \begin{bmatrix} P_{00} & P_{01} & 0 & 0 & \cdot & \cdot \\ P_{10} & 0 & P_{12} & \cdot & \cdot & \cdot \\ P_{20} & 0 & 0 & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & & & \\ \cdot & \cdot & \cdot & & & \\ \cdot & \cdot & \cdot & & & \end{bmatrix}$$

Let $P(t) = (P_0(t), P_1(t), \ldots)$ be the vector o
NEURON is in state i at time t. Then we have

$$P(t+1) = P(t)\,P(x(t))$$

If we define the lumped model transition behavior as

$$\%\cdot\text{IN}\cdot\text{STATE}'(t+1) = \%\cdot\text{IN}\cdot\text{STATE}'(t)P(x(t))$$

then we have the structure morphism in the limiting infinite case.

We shall use this same transition function for the lumped model when we investigate the behavior of the base model with a finite number of neurons and neighbors per block. Of course, our mapping from the base model to the lumped model is no longer a structure morphism but is an "approximate" structure morphism.

To investigate to what extent the lumped model still predicts base model behavior under this "approximate" structure morphism, we carried out a computer simulation of the two models. The simulation consisted of a package of Fortran programs that allowed great flexibility in specifying connection patterns, initial states and parameter settings. Certain statistics were

automatically compiled. More information on the simulation programs can be found in Appendix 1.

In the following, we will consistently use "run" to mean a single simulation of the base or lumped model with a fixed set of parameters and random number seed. An "experiment" will refer to several runs with identical parameters but different random number seeds. A set of related experiments will be called a "series of experiments" or simply a "series".

Two restrictions on the base model were imposed in order to simplify the computer implementation.

(1) There is a maximum of 20 recovery states possible. A neuron in the highest recovery state that does not fire remains in the same recovery state. In our test runs we used only recovery states 0-6.

(2) For all i,j, all neurons in block i have the same number of neighbors in block j with equal synapse weights and these neighbors are randomly chosen from the neurons in block j.

Since the EXTERNAL·NEIGHBORS and their SYNAPS weights are the same for all neurons in a BLOCK we will use the term "external input" for the external influence.

First Series of Experiments

Connection Pattern

The base model consisted of a single block with no connections between neurons in the block. We varied the size of the block using 25, 50, 75, 100, 500 and 1000 neurons.

Parameter Settings

1. Background: 0
2. Noise: Gaussian, with mean 0 and standard deviation of 10, 20 and 40.
3. Threshold: Exponential decay $27e^{-s}$ where s is the recovery state.

Initial State

All neurons were initially in recovery state 6.

Input segment

Input levels of -100, -80,...,80, 100, were tried. The length of the input segment was sufficient to allow a "steady state" to be reached.

Not all possible combinations of experiments were conducted. The following table shows the experiments actually done (x's).

St. Dev. \ Size	25	50	75	100	500	1000
10	x			x		
20	x	x	x	x	x	x
40	x					

In this series of experiments, it is clear that if there were an infinite number of neurons in a block, we would have a structure morphism from the base model to the lumped model. Since there are no neigh-

bors, each neuron receives the same total input. The only difference is that each neuron samples independently from a noise distribution. Let us assume that there is an exact agreement between the probability vector of recovery states in the lumped model and the percent-in-state vector of the base model at time t. Then, at time t+1, the actual percent-in-state vector of the base model can be considered a sample from the distribution predicted by the lumped model.

Whether or not the observed differences between the actual percent-in-state vectors and the predicted percent-in-size vector are reasonable can be tested using the chi-square statistic. In effect, we would be testing the random number generator. Such a chi-square test would surely given good results if the lumped model was reset at each time step. However, the aim of our investigations is to show that the lumped model can be used to estimate behavior of the base model without of course knowing the base model behavior. Resetting the lumped model correctly would clearly be possible only if base model behavior is known. Without resetting the lumped model, error propagation becomes an added consideration and is discussed later.

Before giving the results of the chi-square tests, it will be instructive to examine the behavior of the lumped model. Because there are no connections between blocks, for a fixed level of external input the lumped model is actually a homogeneous Markov chain with the following transition matrix.

$$P = \begin{bmatrix} p_{00} & 1-p_{00} & 0 & 0 & 0 & 0 & 0 \\ p_{10} & 0 & 1-p_{10} & 0 & 0 & 0 & 0 \\ p_{20} & 0 & 0 & 1-p_{20} & 0 & 0 & 0 \\ p_{30} & 0 & 0 & 0 & 1-p_{30} & 0 & 0 \\ p_{40} & 0 & 0 & 0 & 0 & 1-p_{40} & 0 \\ p_{50} & 0 & 0 & 0 & 0 & & 1-p_{50} \\ p_{60} & 0 & 0 & 0 & 0 & 0 & 1-p_{60} \end{bmatrix}$$

Let x be the external input and σ be the standard deviation of the noise, then

$$p_{i0} = F(x-27e^{-i})$$

where F is a cumulative distribution function (c.d.f.) that is normal with mean 0 and standard deviation σ $(N(0,\sigma))$.

We note that the finite transition matrix above is irreducible, aperiodic and positive recurrent[1]. Thus, the markov chain possesses a long run distribution. That is, regardless of the initial probability vector of recovery states, the lumped model will approach the same steady state distribution in the limit. The limiting distributions for various input levels are shown in figures 1 and 2.

The behavior of the base model was qualitatively similar to that of the lumped model although the base model percentages tended to fluctuate about a "steady state". A quantitative analysis was obtained using the standard chi-square test. For each time step we calculated a chi-square value assuming that the base model percentages were a sample from the theoretical distribution represented by the lumped model probability vector. If the expected number of neurons in a particular recovery state was less than five, then this recovery state was aggregated with another recovery state. Thus, the number of degrees of freedom ranged from 0 to 6. For a given number of degrees of freedom R, the chi-square statistics obtained should be X^2_R distributed. Figures 3 and 4 show the distributions that were actually obtained for each experimental run. The values do in fact appear to be X^2 distributed. A second level chi-square analysis was conducted assuming the hypothesis that we were sampling from a chi-square distribution. Figure 5 shows the results of this analysis.

---

[1] See Feller, An Introduction to Probability Theory and Its Applications, Chapter XV.

One would not expect the chi-square tests between the lumped and base models to be extremely good because of the problem of error propagation. The chi-square test assumes that sampling is independent from time step to time step. Thus, if at each time step we were to reset the lumped model, we would expect good chi-square results. One way to eliminate most of the correlation from one time step to the next due to not resetting the lumped model is not to sample at every time step. Rather, one might sample say at every fifth time step. We did an analysis sampling at every fifth time step for the case of 100 neurons with a standard deviation of 20. The results obtained were extremely good and are also shown in Figures 3 and 5.

We shall now consider the question of error propagation. Zeigler [13] introduced the important idea of considering the error at a time step as composed of two parts - (1) due to sampling error and (2) due to not resetting the lumped model. The second component of the error can be analyzed solely in terms of lumped model behavior. Figure 6 is a schematic representation of error propagation. Because of the structure of the approximate homomorphism h, ε_1 and ε_2 are due to sampling errors. The additional error $k\varepsilon_1$ is due to not resetting the lumped model. To investigate this additional error, we will describe what happens to two states of the lumped model that differ by a distance ε.

Assume that r and s are two different probability vectors (states) of the lumped model. That is, $r = (r_0,\ldots,r_6)$ and $s = (s_0,\ldots,s_6)$. Let distance be measured by the metric d defined as:

$$d<r,s> = \max_{0\leq i\leq 6} |r_i - s_i|$$

Let $a = r-s$. Then, a will be called the error vector. Notice that $\sum_{i=0}^{6} a_i = 0$. Now let

$$d\langle rP, sP\rangle = \max_i |(aP)_i|,$$

where

$$aP = \left(\sum_{i=0}^{6} a_i p_{i0},\ a_0(1-p_{00}),\ a1(1-p_{10}),\ldots,a_4(1-p_{40}),\ a_5(1-p_{50})\right.$$

$$\left.+a_6(1-p_{60})\right).$$

Now, if there exists $k<1$, such that for all pairs of lumped model states r,s,

$$d\langle rP, sP\rangle \leq k\, d\langle r,s\rangle$$

then the error propagation is ultimately bounded. (Zeigler [13]). Unfortunately, a simple example shows that such a condition cannot be guaranteed. Consider the two probability vectors

$r = (.8,0,0,0,0,.1,.1)$ and

$s = (.9,.1,0,0,0,0,0)$.

Let the external input be -20 and the standard deviation of the noise be 20. Then $p_{00} = .01$, $p_{10} = .07$, $p_{20} = .12$, $p_{30} = .14$, $p_{40} = .15$, $p_{50} = .16$, $p_{60} = .16$.

$a = (-.1,-.1,0,0,0,.1,.1)$

$aP = (.024,-.099,-.093,0,0,0,.168)$

Thus, $d\langle r,s\rangle = .1$ but $d\langle rP,sP\rangle = .168$.

In one transition then, we cannot be assured that two lumped model states will converge.

An interesting situation is the case of only three different recovery states. Such a situation arises quite naturally if one considers, for example, firing, absolutely refractory, and quiescent as the three recovery states. It can easily be shown that if

$$P = \begin{bmatrix} p_{00} & p_{01} & p_{02} \\ p_{10} & p_{11} & p_{12} \\ p_{20} & p_{21} & p_{22} \end{bmatrix}$$

for $0<p_{ij}<1$, $i,j = 0,1,2$, there exists a $k<1$ such that $d<rP,sP> \leq kd<r,s>$ for all vectors r,s. That is, in one transition the error will be reduced. For a matrix of the type that we have been considering,

$$P = \begin{bmatrix} p_{00} & 1-p_{00} & 0 \\ p_{10} & 0 & 1-p_{10} \\ p_{20} & 0 & 1-p_{20} \end{bmatrix}$$

this still holds.

This first series of tests has shown that analyzing the behavior of the lumped model seems to be a reasonable way to study expected base model behavior. We now turn to another series of experiments that has positive feedback between neurons.

Second Series of Experiments

Connection Pattern

The base model had 1 block. The number of neurons ranged from 50 to 1000 and the number of neighbors varied from 20 to 100. [See below]. The total input weight [synapse wt x #nbrs] was held at a constant value of +100.

Parameter Settings

1. Background: -20
2. Noise: Gaussian with mean 0 and standard deviation 10.
3. Threshold: $200e^{-s}$

Initial State

All neurons start in state 0.

Input Segment

The external input was 0.

Experiments

Size \ NBRS	100	80	60	50	30	20
50				x		x
100		x		x		x
200			x	x		
400	x	x	x	x		
800			x	x		
1000				x	x	x

The purpose of this series was to analyze the effect of feedback within a block of neurons. We wish to show that the lumped model is still capable of explaining the behavior of the base model.

The first step is to explain the observed behavior of the lumped model. In our experiments, we used a starting state of all neurons in recovery state 0 (i.e. all firing). The subsequent behavior of the lumped model is shown in Figure 7. As can be seen, the lumped model firing level very quickly reaches a constant value and in fact from the output we observed that the lumped model reaches a steady state.

Experimentally, we tried other initial starting states but in all cases we reached the same longterm steady state.

In an attempt to prove that there would only be one steady state we looked at the lumped model as a nonhomogeneous markov chain.

The following equations define a non-homogeneous markov chain, and follow immediately from the transition rule defining the lumped model.

Let $S(t)$ be the row vector representing the state at time t, thus, $S(0)$ is the initial state.

Let $P(t) = [p_{ij}(t)]$ be the single step transition matrix.

Then,

(1) $S(t) = S(t-1)P(t-1)$

i.e. $S_k(t) = \sum_{i=0}^{6} S_i(t-1)p_{ik}(t-1)$

(2) $x(t) = KS_0(t)$ where K is the total feedback

(call x(t) the input at time t)

(3) $p_{i0}(t) = F(x(t)+\beta-\theta_i) = F_i(x(t))$

where β is background level

θ_i is threshold in recovery state i

and F is the c.d.f. associated with the noise.

(Note: we use $N(0,\sigma)$ for F)

From equations (1), (2), and (3) we get

$$(4) \qquad p_{i0}(t) = F_i\left[K_i \sum_{j=0}^{6} S_j(t-1)p_{j0}(t-1)\right]$$

Recall that

$$(5) \qquad P(t) = \begin{bmatrix} p_{00}(t) & 1-p_{00}(t) & 0 & \cdots & 0 \\ p_{10}(t) & 0 & 1-p_{20}(t) & \cdots & 0 \\ p_{20}(t) & 0 & 0 & \cdots & \vdots \\ \vdots & \vdots & \vdots & & \vdots \\ \vdots & \vdots & \vdots & & 1-p_{50}(t) \\ p_{60}(t) & 0 & 0 & \cdots & 1-p_{60}(t) \end{bmatrix}$$

Using the theory of non-homogeneous Markov chains, Paz[9], we were able to show that the lumped model is a weakly ergodic chain. We would like to show that the model is strongly ergodic but we haven't been able to prove this so far.

We shall require the following definitions and propositions from non-homogeneous Markov chain theory, which can be found in Paz [9].

Let $P = [p_{ij}]$ be a stochastic matrix. Then define

$$\delta(P) = \sup_{u,v} \sum_j (p_{uj}-p_{vj})^+ \quad \text{where } k^+ = \max[0,k]$$

$$||P|| = \sup_i \sum_j |p_{ij}|$$

Proposition $0 \le \delta(P) \le 1$

Definition

A non-homogeneous Markov Chain is an infinite sequence of Markov matrices $\{P_i\}_{i=0}^{\infty}$ such that P_i is the matrix of transition probabilities at time t=i.

Let $H_{mn} = \prod_{i=m}^{n} P_i$

Definition

A non-homogeneous Markov chain is weakly ergodic if

$$\lim_{n\to\infty} \delta(H_{mn}) = 0, \quad m = 0,1,2,\ldots$$

A non-homogeneous markov chain is strongly ergodic if $\exists$ a constant matrix Q [i.e. all rows equal] such that

$$\lim_{n\to\infty} ||H_{mn} - Q|| = 0$$

Theorem (Given without proof)

A markov chain is weakly ergodic iff $\exists$ a subdivision of the chain into blocks of matrices $\{H_{i_j,i_{j+1}}\}$ such that $\sum_{j=0}^{\infty} (1- (H_{i_j,i_{j+1}}))$ diverges.

Proposition The lumped model is weakly ergodic.

Proof:

Given the transition matrix, from (5), we have that $\delta[P(t)]$

$$= \max_{0\le k\le 6} [1-p_{k0}(t)]$$

From (3)

$$p_{k0}(t) = F(x(t)+\beta-\theta_i)$$

$$= F(KS_0(t)+\beta-\theta_i)$$

Now assuming that F is strictly monotone increasing, we have

$$0 < F(\beta-\theta_i) \le p_{k0}(t) \le F(K+\beta-\theta_i) < 1$$

Thus $p_{k0}(t)$ is uniformly bounded away from and if we let subdivisions in the previous theorem be just $H_{i_j,i_{j+1}} = P(j)$ then we have that $1-\delta(P(t))$ is bounded away from 0 uniformly.

Unfortunately, weak ergodicity does not tell us much. It tells us

only that there is a sequence of constant matrices $\{E_{mn}\}$ such that

$$\lim_{n\to\infty} ||H_{mn} - E_{mn}|| = 0$$

We would like to say that there exists $\{E_m\}$ such that

$$\lim_{n\to\infty} ||E_{mn} - E_m|| = 0 \quad \text{i.e. strongly ergodic}$$

[It can be shown that E_m is independent of m]

Strong ergodicity would say that the non-homogeneous process reaches a long run distribution (i.e. steady state).

Note however, that even strong ergodicity would not allow us to say that the same long run distribution was reached independent of the starting state. This is because our non-homogeneous chain is itself a function of the starting state.

One attempt to prove strong ergodicity assuming weak ergodicity was the following.

Assuming weak ergodicity, we know that H_{1n} is approximately a constant matrix for n sufficiently large. Strong ergodicity would say that the transition matrix $P(n)$ at time step n, would be one that would map H_{1n} into H_{1n} almost exactly. Can we determine the form of such a matrix and must our non-homogeneous process have such a form after some fixed number of time steps? That is, let the constant matrix that H_{1n} approximates be C.

$$\therefore \quad CP = C$$

$$\text{or} \quad C(P-I) = 0$$

Let $[C_0,\ldots,C_6]$ be a row of C. Then the above equation is equivalent to saying that C is in the null space of P-I.

Now

$$P\text{-}I = \begin{bmatrix} p_{00}-1 & 1-p_{00} & 0 & 0 & 0 & \cdots & 0 \\ p_{10} & -1 & 1-p_{10} & \cdots & & & \vdots \\ p_{20} & 0 & -1 & 1-p_{20} & \cdots & & \vdots \\ \vdots & & & & & & 0 \\ \vdots & & & & & & 1-p_{50} \\ p_{60} & \cdots & & & & & -p_{60} \end{bmatrix}$$

It is immediately clear that since there are 6 linearly independent columns the null space of P-I is 1 dimensional. Consequently, since we are interested in only stochastic vectors, there is at most 1 stochastic vector in the null space of P-I.

This brings us back to the obvious problem of showing that S(t) is in the null space of (P-I)(t) for some t.

It appears that we must examine the recurrence relations (eq 1,4) to get any further.

(1) $$S_k(t) = \sum_{i=0}^{6} S_i(t-1)P_{ik}(t-1)$$

(2) $$x(t) = KS_0(t)$$

(3) $$p_{i0}(t) = F_i(x(t))$$

(4) $$p_{i0}(t) = F_i[K \sum_{i=0}^{6} S_j(t-1)p_{j0}(t-1)]$$

We will attempt to get bounds on $p_{ik}(t)$ as $t\to\infty$.

Assuming the threshold function is a monotone decreasing function (of the recovery state), we have that

$$0 \le F_0(x) \le F_1(x) \le \ldots \le F_6(x) \le 1 \quad \text{for all } x.$$

Consequently,

$$p_{i0}(0) = F_i(KS_0(0))$$

$$p_{i0}(1) = F_i[K \sum_{i=0}^{6} S_j(0)F_j(KS_0(0)]$$

$$\geq F_i[KF_0(KS_0(0))]$$

$$p_{i0}(2) = F_i[K \sum_{i=0}^{6} S_j(1)p_{j0}(1)]$$

$$\geq F_i[K \sum_{i=0}^{6} S_j(1)F_j[KF_0(KS_0(0))]]$$

$$\geq F_i[KF_0[KF_0(KS_0(0))]]$$

Inductively,

$$p_{i0}(t) \geq F_i[(KF_0)^t(KS_0(0))]$$

Similarly, we can get an upper bound

$$p_{i0}(t) \leq F_i[(KF_6)^t(KS_0(0))]$$

Since F_i is monotone, we have either

$$KF_i(x) \geq x$$

$$\therefore (KF_i)^t(x) \geq (KF_i)^{t-1}(x) \geq (KF_i)^{t-2}(x) \ldots \geq KF_i(x) \geq x$$

or

$$(KF_i)^t(x) \leq (KF_i)^{t-1}(x) \ldots \leq KF_i(x) \leq x$$

In either case since we have a bounded monotone sequence, the limit exists as $t\to\infty$ [may depend on $KS_0(0)$]. Thus

$$\lim_{t\to\infty} F_i[(KF_0)^t(KS_0(0))] \leq \underline{\lim_{t\to\infty}}\, p_{i0}(t) \leq \overline{\lim_{t\to\infty}}\, p_{i0}(t)$$

$$\leq \lim_{t\to\infty} F_i[(KF_6)^t(KS_0(0))].$$

We thus have (possibly crude) bounds for the limiting transition probabilities if they exist. (Note that the limit points are fixed points of KF_0 or KF_6.)

An example of calculating such bounds is now presented. Let K = 100, $\sigma = 20$, $\theta_i = 200e^{-i}$, $\mu = 0$, $\beta = -20$.

$$\lim_{t\to\infty} (KF_0)^t(KS_0(0)) = 0$$

since this is the only fixed point of KF_0. Also $\lim_{t\to\infty} (KF_6)^t(KS_0(0)) = 100$ [see Figure 8].

$$\therefore F_1[0] = F[-20 - \frac{200}{e}] = 0$$

$$F_1[100] = .74$$

So, $0 \le p_{10} \le .74$ in the limit.

Other bounds that we calculated for this situation were

$$0 \le p_{00} \le 0$$

$$.02 \le p_{60} \le 1 \quad .$$

We have seen that analytic results for long run distributions and steady states appear extremely difficult to obtain.

We now present a heuristic "proof" that the lumped model with the parameters of this series can in fact reach one and only one steady state independent of the initial state, if it reaches a steady state at all.

We assumed that in the steady state, the first six recovery states 0,...,5 had approximately the same probabilities. This is plausible because the threshold is fairly high in states 0,...,5. Therefore, it is highly probable that a neuron that fires will not fire again until it reaches recovery state 6, and consequently, for a steady state, the first six recovery states must have the same probabilities.

Moreover, nearly all the neurons that do fire will in fact have been in recovery state six.

Therefore,

$$S = [S_0, S_0, S_0, S_0, S_0, S_0, S_6] \quad \therefore S_6 = 1 - 6S_0$$

and assuming $p_{60}S_6 = S_0$

we get

$$p_{60} = \frac{S_0}{S_6} = \frac{S_0}{1-6S_0}$$

Now, $$p_{60} = F_6[KS_0] = \Phi\left(\frac{KS_0-\theta_6-\beta}{\sigma}\right) = \Phi\left(\frac{100S_0-20.5}{10}\right)$$

So $$\Phi\left(\frac{100S_0-20.5}{10}\right) = \frac{S_0}{1-6S_0}$$ [Note: Φ is the standard normal c.d.f.]

is satisfied in the steady state. Graphing each side of the above equation as a function of S_0 (See figure 9), we find that the only possible solution S_0 is very close to the observed value.

Since we would like to use the deterministic lumped model to predict the behavior of the probabilistic base model, we need to analyze the possible effects of error propagation. We analyze error propagation by observing what happens to two states in the lumped model that are near each other. The metric that we use could be for example the following.

Let r and s be two states of the lumped model then $d\langle r,s\rangle = \max_{0\le i\le 6} |r_i - s_1|$.

Instead of using the seven-dimensional vector that we actually have, we attempted to make the problem more tractable by just using two recovery states, 0 and 6. Thus, we mapped the next state transition of the lumped model starting with different probabilities in states 0 and 6. If the probabilities did not add up to 1, we uniformly spread the remaining probability in states 1 through 5.

This gave us the "force field" in figure 10. Although not quite accurate since there is not a homomorphism from the 7-dimensional state space to the two dimensional, we still obtained a good understanding of the behavior

of the lumped model. The steady state point is clearly locatable. Also, we observe that there are some states which diverge quite significantly. Consequently, if a base model received enough noise to push it far enough away from the steady state, then the trajectory of the base model should follow the force field trajectory defined by the lumped model.

To see whether this was actually true, we graphed several base model runs. We first simply plotted the firing level of runs with 400 neurons and 50 neighbors. (Figure 11) On the surface, this does not appear to match in any way the lumped model run (Figure 7) which is also a plot of the firing level vs. time. However, we decided to plot the same base runs using the same two dimensional "state space" as the "force field" of the lumped model. This plot (Figure 12) reveals quite clearly that once the base model is displaced from equilibrium (by noise) it tends to follow the "Force field" of the lumped model. We plotted other base model runs using the same two dimensional "state space". (Figures 13, 14, 15, 16, 17). They show the qualitative similarity between base model runs and lumped model runs. We see that when the neighborhood size is reasonably large (Figures 13, 15, 16) the base model spends much of its time near equilibrium. When it is displaced by noise it follows a lumped model trajectory that is not too large. Also, it seems that there is almost a critical neighborhood size in the sense that even with 1000 neurons, if the neighborhood size is small, for example 20, 30 and 50 neighbors (Figure 14) then there is always sufficient noise to kick the base model away from equilibrium.

To analyze it more quantitatively, one might want to now make the lumped model probabilistic and see if the probability distribution of the trajectories of the lumped model can be made to approximate those of the base model.

The analyses of these two series of experiments has given us encouraging results. The lumped model can be used for general predictions of long term behavior. The lumped model vector field provides a conditional statistical prediction of future behavior and is reasonably accurate for short term predictions. That is to say, if we know the base model state at a particular time, we have a qualitative idea of future trajectories of the base model from that point on. Hopefully, in future work, this qualitative feeling can be made more precise by assigning probabilities to the expected trajectories. We notice that the lumped model is useful in predicting base model behavior even when we have reasonably small numbers of neurons in a block, say 100, and when the neighborhood size is above approximately 50 neurons. The use of positive feedback did not lead to unusual behavior of the base model, not predictable by the lumped model. Thus, it is reasonable to expect that blocks of finite numbers of neurons with complex interconnections can be lumped and still retain significant aspects of behavior.

We have seen that with a very simple lumped model, we have been able to retain important aspects of the behavior of the extremely complex base model. The simplicity of the lumped model relative to the base model was apparent in the computer implementation. The simulation program for the lumped model was far simpler than for the base model. Also, the lumped model simulation used less computer memory and required less computer time per simulated time step. Appendix 2 gives a comparison between the running times of the base model and the lumped model.

Bibliography

1. Anninos, P.A., Beek, B., Csermely, T.J., Harth, E.M. and Pertile, G., "Dynamics of Neural Structures", *J. Theor. Biol. 26*, 1970, 121-148.

2. Arbib, M.A., "Automata Theory and Control Theory: A Rapprochement", *Automatica, 3*, pp. 161-189, 1966.

3. Bacon, G.C., "The Decomposition of Stochastic Automata", *Inform. & Control, 7*, 1964, 320-329.

4. Csermely, T.J., Harth, E., and Lewis, N.S., "The Netlet Theory and Cooperative Phenomena in Neural Networks", *J. Dynamic Systems, Measurement and Control*, Sept. 1973.

5. Gray, J.N., and Harrison, M.A., "The Theory of Sequential Relations", *Information and Control, 9*, 1966, 435-468.

6. Harth, E.M., Csermely, T.J., and Lindsay, R.D., "Brain Functions and Neural Dynamics", *J. Theor. Biol., 26*, 1970, 93-120.

7. Hartmanis, J. and Stearns, R.E., *Algebraic Structure Theory of Sequential Machines*, Prentice Hall, 1969.

8. Mihram, G.A., *Simulation: Statistical Foundations and Methodology*, Academic Press, 1972.

9. Paz, A., *Introduction to Probabilistic Automata*, Academic Press, 1971.

10. Wong, R., and Harth, E., "Stationary States and Transients in Neural Populations", *J. Theor. Biol., 40*, 1973, 70-106.

11. Zeigler, B.P., "Modelling and Simulation: Structure Preserving Morphisms for Discrete and Continuous Systems", In: *Computers and Automata*, Proc. 21st International Conf., Polytechnic Institute of Brooklyn, Brooklyn, N.Y., 1972.

12. ____, "On the Formulation of Problems in Modelling and Simulation within the Framework of Mathematical Systems Theory", *Proc. 6th Int. Congress on Cybernetics*, Namur, Belgium, 1972.

13. ____, *Theory of Modelling and Simulation.* (to appear).

```
// JOB   111102222030333
// XEQ TD      FX
// FOR BASE
*LIST ALL
*NONPROCESS PROGRAM
*ONE WORD INTEGERS
*IOCS(TYPEWRITER,KEYBOARD,1443 PRINTER, MAGNETIC TAPE)
C  MAIN PROGRAM FOR THE BASE PROGRAM
C
      EXTERNAL KBDE, A7E
      INTEGER OLDST(5,1000),SSWCH,DSWCH,FILOK
      REAL INEXT, NOISE, NOIS
C DECLARATION FOR EXTRA COMMON BLOCK VARIABLES
      INTEGER STATE(5,1000)
C BASIC COMMON BLOCK
      INTEGER T,OUTF,U,V,P
      REAL NOIST(5,3),INPU(5)
      COMMON NBLKS,NUMBR(5),NSTAT,T,U,THPR(5,5),NOIST,INPR1(5),
     1  INPR2(5),EXINP(5,30),INPU,BKGND(5),WEIGT(5,5),OUTF,V,P
C EXTRA COMMON BLOCK FOR BASE MODEL
      COMMON STATE,NFILE,KOUNT(5)
C
C INITIALIZATION ROUTINE LINKS TO THIS ROUTINE, SO INITIALIZATION IS COMPLE
C IF SENSE SWITCH 4 IS UP USE PDP7 TELETYPE AS BACKUP CONSOLE
      IF (SSWCH(4)) 110,110,100
100   CALL SEFIN(A7E)
      GO TO 115
110   CONTINUE
      CALL SEFIN(KBDE)
115   CONTINUE
C
C GET OUTPUT FILE NUMBER AND INITIALIZE IT
      WRITE(V,950)
950   FORMAT('ENTER OUTPUT FILE NO. (0 FOR NO OUTPUT FILE)')
      CALL RDINT(OUTF)
      IF(OUTF) 975,975,960
960   CALL FOPO(OUTF)
      IF (FILOK(0)) 965,965,962
962   WRITE(V,963)
963   FORMAT('OUTPUT FILE WILL NOT OPEN')
      GO TO 110
965   CONTINUE
      CALL FPUTI(NBLKS)
      CALL FPUTI(NSTAT)
      DO 970 I= 1,NBLKS
970      CALL FPUTI(NUMBR(I))
975   CONTINUE
      CALL PRNTI
    5 CONTINUE
C TEST FOR RESETTING PARAMETERS
      IF (DSWCH(15)) 15,15,13
13    CALL RSETB
15    CONTINUE
C  UPDATE STATE VECTOR -- OLDSTATE = STATE
      DO 10 I=1,NBLKS
         JFINL = NUMBR(I)
         DO 10 J=1,JFINL
   10    OLDST(I,J)= STATE(I,J)
C  TEST FOR STOPPING CONDITION
      IF(DSWCH(14)) 30,30,20
```

```
20      IF(OUTF) 25,25,22
22      IF(FILOK(0)) 23,24,23
23      WRITE(V,27)
27      FORMAT('ERROR-OUTPUT FILE OVERFLOW')
24      CALL FCLO
25      CALL EXIT
   30 CONTINUE
C   COMPLTE NEXT STATE (LOOP OVER ALL BLOCKS)
        CALL FOPI(NFILE)
       CALL FGETR(DUMY)
       DO 1000 I=1,NBLKS
          INPU(I) = INEXT(I)
          JFINL = NUMBR(I)
          LIMIT = KOUNT(I)
C   LOOP OVER NEURONS WITHIN A BLOCK
          DO 1000 J=1,JFINL
             NOIS = NOISE(I)
             VOLLY = 0.
C   SUM INPUTS TO NEURON (I,J) FROM ITS NEIGHBORS
             DO 50 K = 1, LIMIT
C NEURON (X,Y) IS ENCODED IN NBRS AS 1001 * X + Y
                CALL FGETI(NBR)
                KK1 = NBR/1001
                KK2 = NBR - KK1*1001
                IF(OLDST(KK1,KK2)) 50,40,50
   40           VOLLY = VOLLY + WEIGT(I,KK1)
   50           CONTINUE
C   ADD THE EXTERNAL INPUT AND COMPUTE THRESHOLD
             VOLLY = VOLLY + INPU(I) + BKGND(I)
             THOL = NOIS + THOLD(I,OLDST(I,J))
C   DOES THE NEURON FIRE?
             IF(VOLLY - THOL) 60,70,70
   60        STATE(I,J) = OLDST(I,J) + 1
             IF (STATE(I,J)-NSTAT) 1000,65,65
65           STATE(I,J) = NSTAT - 1
             GO TO 1000
   70        STATE(I,J) = 0
 1000        CONTINUE
       CALL PRNT(OLDST)
       T=T + 1
       GO TO 5
       END
*STOREMO              BASE
                      *
```

```
// JOB   111102222030333
// XEQ TD       FX
// FOR
*LIST ALL
*NONPROCESS PROGRAM
*ONE WORD INTEGERS
      FUNCTION DIST(I,X,CFACT)
C COMPUTES DISTRIBUTION FUNCTION FOR NOISE FUNCTIONS
C
      REAL GAUSS(33)
C BASIC COMMON BLOCK
      INTEGER T,OUTF,U,V,P
      REAL NOIST(5,3),INPU(5)
      COMMON NBLKS,NUMBR(5),NSTAT,T,U,THPR(5,5),NOIST,INPR1(5),
     1  INPR2(5),EXINP(5,30),INPU,BKGND(5),WEIGT(5,5),OUTF,V,P
      DATA GAUSS/ .5,.5398,.5793,.6179,.6554,.6915,.7257,.758,.7881,
     1 .8159,.8413,.8643,.8849,.9032,.9192,.9332,.9452,.9554,.9641,
     2 .9713,.9773,.9821,.9861, .9893,.9918,.9938,.9953,.9965,.9974,
     3 .9981,.9986,.999,.9993 /
C NCIST(I,1) INDICATES THE TYPE OF DISTRIBUTION USED FOR BLOCK I
C NOIST(I,1) = 1.0 FOR GAUSSIAN       NCIST(I,1) = 2.0 FOR UNIFORM
C NCIST(I,2), NOIST(I,3) ARE THE MEAN AND DEVIATION FOR GAUSSIAN DISTRIBUTI
C NCIST(I,2), NOIST(I,3) ARE THE LOWER AND UPPER LIMITS FOR UNIFORM DISTRIB
C
      K = NOIST(I,1)
      GO TO (100,200),K
C GAUSSIAN DISTRIBUTION
100   CONTINUE
C  CFACT IS THE SECOND ORDER CORRECTION FACTOR ADDED TO THE VARIANCE
      STDEV = SQRT(NOIST(I,3)**2 + CFACT)
      Z = (X-NOIST(I,2))/STDEV
      IF (STDEV) 110,110,105
105   CONTINUE
      Y = ABS(Z)
      INDEX = 10.0 * Y + 1.0
      IF(INDEX-32) 120,120,110
110   DIST = 1.0
      GO TO 130
120   DIST = GAUSS(INDEX) + (GAUSS(INDEX+1) - GAUSS(INDEX)) * (10.0*Y
     1  -INDEX + 1)
130   IF(Z) 135,140,140
135   DIST = 1.0 - DIST
140   RETURN

C UNIFORM DISTRIBUTION
200   CONTINUE
      A = NOIST(I,2)
      B = NCIST(I,3)
      IF(X-A) 210,210,220
210   DIST = 0.0
      RETURN
220   IF(X-B) 240,230,230
230   DIST = 1.0
      RETURN
240   DIST = (X-A)/(B-A)
      RETURN
      END
*STOREMC              DIST
                      *
```

```
// JOB   111102222030333
// XEQ TD       FX
// FOR
*LIST ALL
*ONE WORD INTEGERS
*NONPROCESS PROGRAM
      REAL FUNCTION INEXT(I)
C THIS FUNCTION PROVIDES THE EXTERNAL INPUT
C
C THE EXTERNAL INPUT MAY BE:  PERIODIC
C                             EXPONENTIAL DECAY
C                             CUTOFF AFTER SOME INITIAL INPUTS
C                             PULSE -- VALUES TAKEN FROM KEYBOARD
C
      INTEGER PRD(5),CTOFF(5)
C BASIC COMMON BLOCK
      INTEGER T,OUTF,U,V,P
      REAL NOIST(5,3),INPU(5)
      COMMON NBLKS,NUMBR(5),NSTAT,T,U,THPR(5,5),NOIST,INPR1(5),
     1  INPR2(5),EXINP(5,30),INPU,BKGND(5),WEIGT(5,5),OUTF,V,P
      EQUIVALENCE(INPR2(1),PRD(1),CTOFF(1))
      DATA KOUNT/0/
      MOD(M,N) = M-M/N*N
C
C  INPR1(I) DETERMINES TYPE   1=PERIODIC  2=EXP DECAY  3=CUTOFF  4=PULSE
C  EXINP(I,J) IS THE INPUT FOR BLOCK I FOR THE JTH STEP IN CYCLE
C
      K= INPR1(I)
      GO TO (1000,2000,3000,4000),K
C PERIODIC INPUTS (PERIOD IS 'PRD')
 1000 CONTINUE
      INDEX = MOD(T,PRD(I)) + 1
      INEXT = EXINP(I,INDEX)
      RETURN
C  EXPONENTIAL DECAY
 2000 CONTINUE
      INEXT = EXINP(I,1) * EXP(-EXINP(I,2) * T)
      RETURN
C  CUTOFF (AT T='CTOFF', INPUT GOES TO ZERO)
 3000 CONTINUE
      IF(T - CTOFF(I)) 3300,3500,3500
 3300 INEXT = EXINP(I,T+1)
      RETURN
 3500 INEXT = 0.0
      RETURN
C PULSE INPUT
4000  CONTINUE
      IF (T-EXINP(I,3)) 4500,4100,4900
C QUERY FOR NEW INFORMATION
4100  KOUNT = KOUNT + 1
      IF(MOD(KOUNT,6)-1) 4150,4110,4150
4110  WRITE(V,4200) T,I
      GO TO 4175
4150  WRITE(V,4220) T,I
4220  FORMAT(/,'TIME IS:',I4,'  BLOCK',I4)
4175  CONTINUE
4200  FORMAT(/,'TIME IS:',I4,'   BLOCK',I4,5X,'ENTER PULSE HEIGHT, PULSE
     1 END TIME, NEXT PULSE TIME')
      CALL RDFLT(EXINP(I,1))
```

```
      CALL RDFLT(EXINP(I,2))
      CALL RDFLT(EXINP(I,3))
      GO TO 4000
C TO PULSE OR NOT TO PULSE
4500  IF (T-EXINP(I,2)) 4600,4600,4900
4600  INEXT = EXINP(I,1)
      RETURN
4900  INEXT = 0.0
      RETURN
      END
*STOREMD              INEXT
                      *
```

```
// JOB   111102222030333
// XEQ TD       FX
// FOR LUMPD
*LIST ALL
*NONPROCESS PROGRAM
*ONE WORD INTEGERS
*IOCS(TYPEWRITER,KEYBOARD,1443 PRINTER, MAGNETIC TAPE)
C
      INTEGER FIRS, DSWCH, FILOK
      REAL CLDST( 5,20), INEXT
C DECLARATION FOR EXTRA COMMON BLOCK VARIABLES
      INTEGER FIRST(5),SOFLG
C BASIC COMMON BLOCK
      INTEGER T,OUTF,U,V,P
      REAL NOIST(5,3),INPU(5)
      COMMON NBLKS,NUMBR(5),NSTAT,T,U,THPR(5,5),NOIST,INPR1(5),
     1  INPR2(5),EXINP(5,30),INPU,BKGND(5),WEIGT(5,5),OUTF,V,P
C EXTRA COMMON BLOCK FOR LUMPED MODEL
      COMMON FIRST, LAST(5), NBRS(25), STATE(5,20),SUMSQ(5,5),SOFLG
C
C INITIALIZE
      CALL INITL
5     CONTINUE
C TEST FOR RESETTING PARAMETERS
      IF (DSWCH(15)) 15,15,13
13    CALL RSETL
15    CONTINUE
C UPDATE THE STATE VECTOR
      DO 10 I=1,NBLKS
         DO 10 J=1,NSTAT
10          OLDST(I,J)= STATE(I,J)
C TEST FOR STOPPING CONDITION
      IF(DSWCH(14)) 30,30,20
20    IF(OUTF) 25,25,22
22    IF(FILOK(0)) 23,24,23
23    WRITE(V,27)
27    FORMAT('ERROR-OUTPUT FILE OVERFLOW')
24    CALL RCLO
25    CALL EXIT
30    CONTINUE
C COMPUTE NEXT STATE
      DO 1000 I=1,NBLKS
         CFACT = 0.0
         INPU(I) = INEXT(I)
         VOLLY = INPU(I) + BKGND(I)
         FIRS = FIRST(I)
         LAS = LAST(I)
C SUM INPUTS FROM NEIGHBORS BLOCKS
         DO 50 J= FIRS,LAS
            K = NBRS(J)
            IF(SOFLG) 50,50,40
40          CFACT = CFACT + OLDST(K,1)*(1.0-OLDST(K,1))*SUMSQ(I,K)
50          VOLLY = VOLLY + WEIGT(I,K)*CLDST(K,1)
C COMPUTE NEW PERCENT IN STATES FOR THIS BLOCK
         SUM=0.0
         DO 100 K=2,NSTAT
            PROB = DIST(I,THOLD(I,K-2)-VOLLY,CFACT)
            STATE(I,K) = PROB*OLDST(I,K-1)
100         SUM = SUM + STATE(I,K)
```

```
            PROB = DIST(I,THOLD(I,NSTAT-1)-VOLLY)
            STATE(I,NSTAT) = STATE(I,NSTAT) + PROB*OLDST(I,NSTAT)
            SUM = SUM + PROB*OLDST(I,NSTAT)
            STATE(I,1) = 1.0 - SUM
1000        CONTINUE
         CALL PRNT2(OLDST)
         T = T + 1
         GO TO 5
         END
*STOREMO                LUMPO
                        *
```

```
// JOB    111102222030333
// XEQ TD       FX
// FOR
*LIST ALL
*ONE WORD INTEGERS
*NONPROCESS PROGRAM
      REAL FUNCTION NOISE(I)
C
      INTEGER RAND
C BASIC COMMON BLOCK
      INTEGER T,OUTF,U,V,P
      REAL NOIST(5,3),INPU(5)
      COMMON NBLKS,NUMBR(5),NSTAT,T,U,THPR(5,5),NOIST,INPR1(5),
     1  INPR2(5),EXINP(5,30),INPU,BKGND(5),WEIGT(5,5),OUTF,V,P
C   I IS BLOCK NUMBER  NOIST(I,1)=1 IS GAUSSIAN,  =2 IS UNIFORM
C   FOR GAUSSIAN DISTRIBUTION: NOIST(I,2)=MEAN  NOIST(I,3)=DEVIATION
C   FOR UNIFORM DISTRIBUTION:  NOIST(I,2)=LOWER LIM. NOIST(I,3)=UPPER LIM
C
      K = NOIST(I,1)
      GO TO (100,200),K
C   GAUSSIAN DISTRIBUTION
  100 CONTINUE
      IRAND = 0
      DO 105 J=1,6
  105    IRAND = IRAND + RAND(2001)
      ZRAND = IRAND/1000.0 - 6.0
      NOISE = NOIST(I,3) * .70711 * ZRAND + NOIST(I,2)
      RETURN
C   UNIFORM DISTRIBUTION
  200 CONTINUE
      NOISE = RAND(2001) * (NOIST(I,3) - NOIST(I,2))/2000.0 + NOIST(I,2)
      RETURN
      END
*STOREMD              NOISE
                      *
```

```
// JOB   111102222030333
// XEQ TD       FX
// FOR
*LIST ALL
*ONE WORD INTEGERS
*NONPROCESS PROGRAM
      SUBROUTINE PRNT(OLDST)
      INTEGER NIS(20),OLDST(5,1000)
C BASIC COMMON BLOCK
      INTEGER T,OUTF,U,V,P
      REAL NOIST(5,3),INPU(5)
      COMMON NBLKS,NUMBR(5),NSTAT,T,U,THPR(5,5),NOIST,INPR1(5),
     1  INPR2(5),EXINP(5,30),INPU,BKGND(5),WEIGT(5,5),OUTF,V,P
C  PRINTS NUMBER OF NEURONS IN EACH STATE FOR EACH BLOCK
      WRITE(U,100) T
  100 FORMAT(/,/,/,' TIME STEP',I7)
C  FOR EACH BLOCK CALCULATE NUMBER AND PERCENT NEURONS IN EACH STATE
      DO 500 I=1,NBLKS
         WRITE(U,101) I, INPU(I)
  101    FORMAT(/  ,I6,'TH BLOCK',10X,'EXTERNAL INPUT IS',F8.2)
         DO 200 J=1,NSTAT
  200       NIS(J) = 0
         LIMIT = NUMBR(I)
         DO 300 J=1,LIMIT
            K = OLDST(I,J) + 1
  250       NIS(K) = NIS(K) + 1
300         CONTINUE
         DO 450 J=1,NSTAT
            K = NIS(J)
            IF(K) 400,400,310
  310       PCENT = K * 100.0/LIMIT
            J1= J - 1
            WRITE(U,102) J1,K,PCENT
  102       FORMAT(10X,'NEURONS IN STATE',I4,'   **',I5,', OR',F7.1,' PE
     1RCENT')
400    IF(OUTF) 450,450,410
410    CALL FPUTI(K)
450    CONTINUE
500      CONTINUE
       RETURN
       END
*STOREMD              PRNT
                      *
```

```
// JOB   111102222030333
// XEQ TD       FX
// FOR
*LIST ALL
*NONPROCESS PROGRAM
*ONE WORD INTEGERS
      SUBROUTINE PRNTI
C PRINTS OUT INITIAL INFORMATION FOR EITHER THE BASE OR THE LUMPED MODEL
      EXTERNAL PRINT
C BASIC COMMON BLOCK
      INTEGER T,OUTF,U,V,P
      REAL NOIST(5,3),INPU(5)
      COMMON NBLKS,NUMBR(5),NSTAT,T,U,THPR(5,5),NOIST,INPR1(5),
     1  INPR2(5),EXINP(5,30),INPU,BKGND(5),WEIGT(5,5),OUTF,V,P
      CALL SFOUT(PRINT)
      CALL CRET
      DO 2010 I= 1,NBLKS
 2010    WRITE(U,2002) I,NUMBR(I)
 2002    FORMAT(/' BLOCK',I2,' HAS',I5,' NEURONS')
      DO 2070 I=1,NBLKS
         WRITE(U,2061) I
2061     FORMAT(/,/,' THRESHOLD FUNCTION FOR BLOCK',I3)
         IF (THPR(I,1)-1.0) 2063,2063,2065
2063     WRITE(U,2064) (THPR(I,J),J=2,5)
2064     FORMAT('+',35X,'IS LINEAR WITH',4F8.3,'  AS THE RESTH, MAXTH, A
     1RP, RRP, RESPECTIVELY')
         GO TO 2070
2065     WRITE(U,2066) (THPR(I,J),J=2,5)
2066     FORMAT('+',35X,'IS EXPONENTIAL WITH',4F8.3,'  AS THE RESTH, M,
     1K, CUTOFF, RESPECTIVELY')
2070     CONTINUE
      DO 2090 I=1,NBLKS
         WRITE(U,2071) I
2071     FORMAT(/,/,' NOISE FUNCTION FOR BLOCK',I3)
         IF (NOIST(I,1)-1.0) 2072,2072,2080
2072     WRITE(U,2073) (NOIST(I,J), J=2,3)
2073     FORMAT('+',30X,'IS GAUSSIAN WITH MEAN',F8.3,'  AND DEVIATION',
     1   F8.3)
         GO TO 2090
2080     WRITE(U,2081) (NOIST(I,J), J=2,3)
2081     FORMAT('+',30X,'IS UNIFORM WITH LOWER BOUND',F8.3,'  AND UPPER
     1  BOUND', F8.3)
2090     CONTINUE
      DO 2110 I=1,NBLKS
         WRITE(U,2095) I
2095     FORMAT(/,/,' EXTERNAL INPUT TO BLOCK',I3)
         KK = INPR2(I)
         K = INPR1(I)
         GO TO (2096,2100,2105,2115),K
2096     WRITE(U,2097) KK, (EXINP(I,J), J=1,KK)
2097     FORMAT('+',30X,'IS PERIODIC WITH PERIOD',I4,'  AND VALUES',
     1   6F8.3,/,(30X,12F8.3))
         GO TO 2110
2100     WRITE(U,2101) (EXINP(I,J), J=1,2)
2101     FORMAT('+',30X,'IS EXPONENTIAL WITH M=',F8.3,'  AND K=',F8.3)
         GO TO 2110
2105     WRITE(U,2106) KK, (EXINP(I,J), J=1,KK)
2106     FORMAT('+',30X,'IS CUTOFF WITH LENGTH',I4,'  AND VALUES',6F8.3,
     1   /,(30X,12F8.3))
```

```
          GO TO 2110
2115      WRITE(U,2116) EXINP(I,3)
2116      FORMAT('+',30X,'IS PULSE WITH FIRST PULSE AT T=',F5.0)
2110      CONTINUE
        DO 2130 I=1,NBLKS
          WRITE(U,2121) I, BKGND(I)
2121      FORMAT(/' BACKGROUND FOR BLOCK',I3,'  IS',F8.3)
2130      CONTINUE
        IF(OUTF) 2200,2200,2150
2150    WRITE(U,2170) OUTF
2170    FORMAT(/' OUTPUT FROM THIS RUN IS IN FILE',I5)
2200    CONTINUE
        CALL EOPAG
        CALL CRET
        RETURN
        END
*STOREMD              PRNTI
                      *
```

```
// JOB    111102222030333
// XEQ TD       FX
// FOR
*LIST ALL
*NONPROCESS PROGRAM
*ONE WORD INTEGERS
      SUBROUTINE PRNT2(OLDST)
C PRINTS NUMBER AND PERCENTAGE OF NEURONS IN EACH STATE FOR EACH BLOCK
C
      REAL OLDST( 5,20)
C BASIC COMMON BLOCK
      INTEGER T,OUTF,U,V,P
      REAL NOIST(5,3),INPU(5)
      COMMON NBLKS,NUMBR(5),NSTAT,T,U,THPR(5,5),NOIST,INPR1(5),
     1  INPR2(5),EXINP(5,30),INPU,BKGND(5),WEIGT(5,5),OUTF,V,P
C
      WRITE(U,100) T
  100 FORMAT(/,/,/,'0TIME STEP',I7)
      DO 500 I=1,NBLKS
         WRITE(U,101) I, INPU(I)
  101    FORMAT(/  ,I6,'TH BLOCK',10X,'EXTERNAL INPUT IS',F8.2)
         DO 250 J=1,NSTAT
           ENMBR = OLDST(I,J) * NUMBR(I)
            IF (OLDST(I,J)) 200,200,150
150          PCENT = OLDST(I,J) * 100.
             J1 = J-1
             WRITE(U,102) J1,ENMBR, PCENT
  102        FORMAT(10X,'NEURONS IN STATE',I4,'    **',F8.2,', OR',F7.1,
     1 ' PERCENT')
200          IF(OUTF) 250,250,210
210          CALL FPUTR(OLDST(I,J))
250          CONTINUE
500      CONTINUE
      RETURN
      END
*STOREMD             PRNT2
                     *
```

```
// JOB   111102222030333
// XEQ TD       FX
// FOR
*LIST ALL
*NONPROCESS PROGRAM
*ONE WORD INTEGERS
      FUNCTION THOLD(I,RCVRY)
      INTEGER RCVRY
C BASIC COMMON BLOCK
      INTEGER T,OUTF,U,V,P
      REAL NOIST(5,3),INPU(5)
      COMMON NBLKS,NUMBR(5),NSTAT,T,U,THPR(5,5),NOIST,INPR1(5),
     1  INPR2(5),EXINP(5,30),INPU,BKGND(5),WEIGT(5,5),OUTF,V,P
C
C THPR(I,1) = 1.0 INDICATES A LINEAR DECAY
C THPR(I,1) = 2.0 INDICATES AN EXPONENTIAL DECAY    M * EXP(-K*RCVRY)
C
      IF (THPR(I,1)-1.0) 5,5,100
C THPR(I,2) IS RESTING THRESHOLD, THPR(I,3) IS MAXIMUM THRESHOLD
C THPR(I,4) IS LENGTH OF ARP, THPR(I,5) IS LENGTH OF RRP
5     CONTINUE
C TEST WHETHER IN ARP OR NOT
      IF (RCVRY-THPR(I,4)) 10,20,20
C IN ARP
10    THOLD = 1.0E35
      RETURN
C IN RRP OR PAST RRP?
20    IF (RCVRY-THPR(I,4)-THPR(I,5)) 30,40,40
C IN RRP, SO COMPUTE THRESHOLD AS LINEAR FUNCTION OF RCVRY
30    THOLD = THPR(I,3) - (THPR(I,3)-THPR(I,2))/THPR(I,5)*
     1   (RCVRY-THPR(I,4))
      RETURN
C PAST RRP   RESTING THRESHOLD
40    THOLD = THPR(I,2)
      RETURN
C
C THPR(I,2) IS RESTING THRESHOLD, THPR(I,3) IS M,
C THPR(I,4) IS K, THPR(I,5) IS CUTOFF
100   IF (RCVRY-THPR(I,5)) 105,150,150
105   THOLD = THPR(I,3)*EXP(-THPR(I,4)*RCVRY)
      RETURN
150   THOLD = THPR(I,2)
      RETURN
      END
*STOREMC             THOLD
```

APPENDIX 1

This appendix contains listings of some of the programs used in the computer simulation of the base and lumped models. We do not include the initialization routines which set up the parameter values and neuron connection pattern. Some of the routines called from our programs are part of the Logic of Computers Group[1] program library. These supporting routines are primarily input/output routines. For further information, contact the authors.

[1]The Logic of Computers Group is a research facility within the Department of Computer and Communication Sciences of the University of Michigan. Its computer facilities include an IBM 1800 with disk bulk storage interfaced to a DEC PDP7 with graphic CRT display.

APPENDIX 2

Running times for base model in second series of experiments.

Values given are average length of time required per simulated time step (in seconds).

NBRS \ Neurons	25	50	100	200	400	800
1	3.7	4.2	5.1	6.4	9.6	16.15
10	3.75	4.25	5.55	8.2	13.4	23.75
20	3.85	5.05	6.75	10.4	17.8	32.55
40	---	5.55	8.8	14.9	26.5	49.55
80	---	---	12.95	23.6	42.95	---

For Comparison, the lumped model requires only 2.75 seconds per timestep -- regardless of the number of neurons or neighbors.

We would expect the running time of the base model to be of the form

$T = K+N(an+b)$ since the main loop of the program is:

```
Loop over all neurons (N)
    Computer threshold and external input
    Loop over neighbors of this neuron (n)
        Sum up inputs
    Compare input and threshold.
```

In fact, $K = 3.2$, $a = .001$, $b = .015$ gives a rather good fit. See Figures 18 and 19.

LUMPED MODEL STANDARD DEVIATION OF 10

FIGURE 1

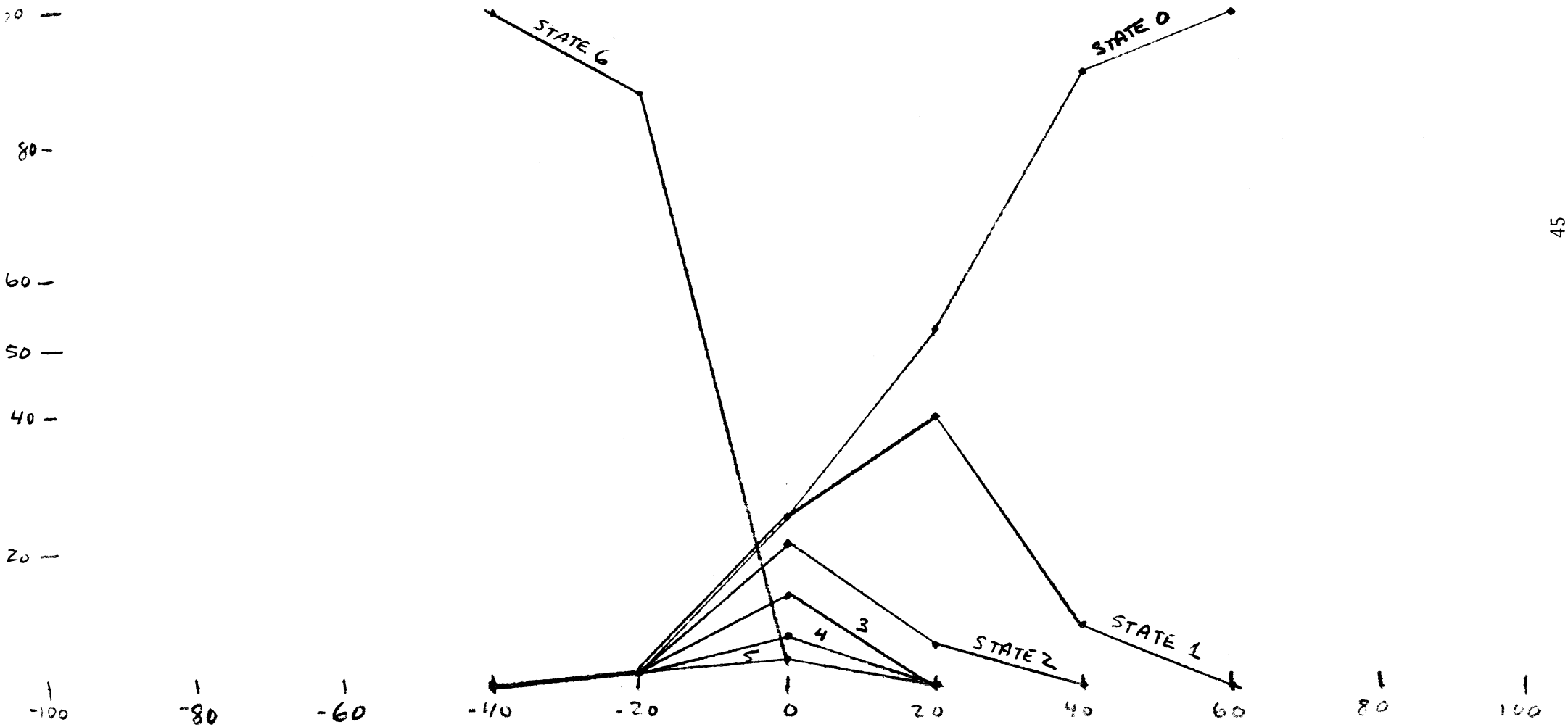
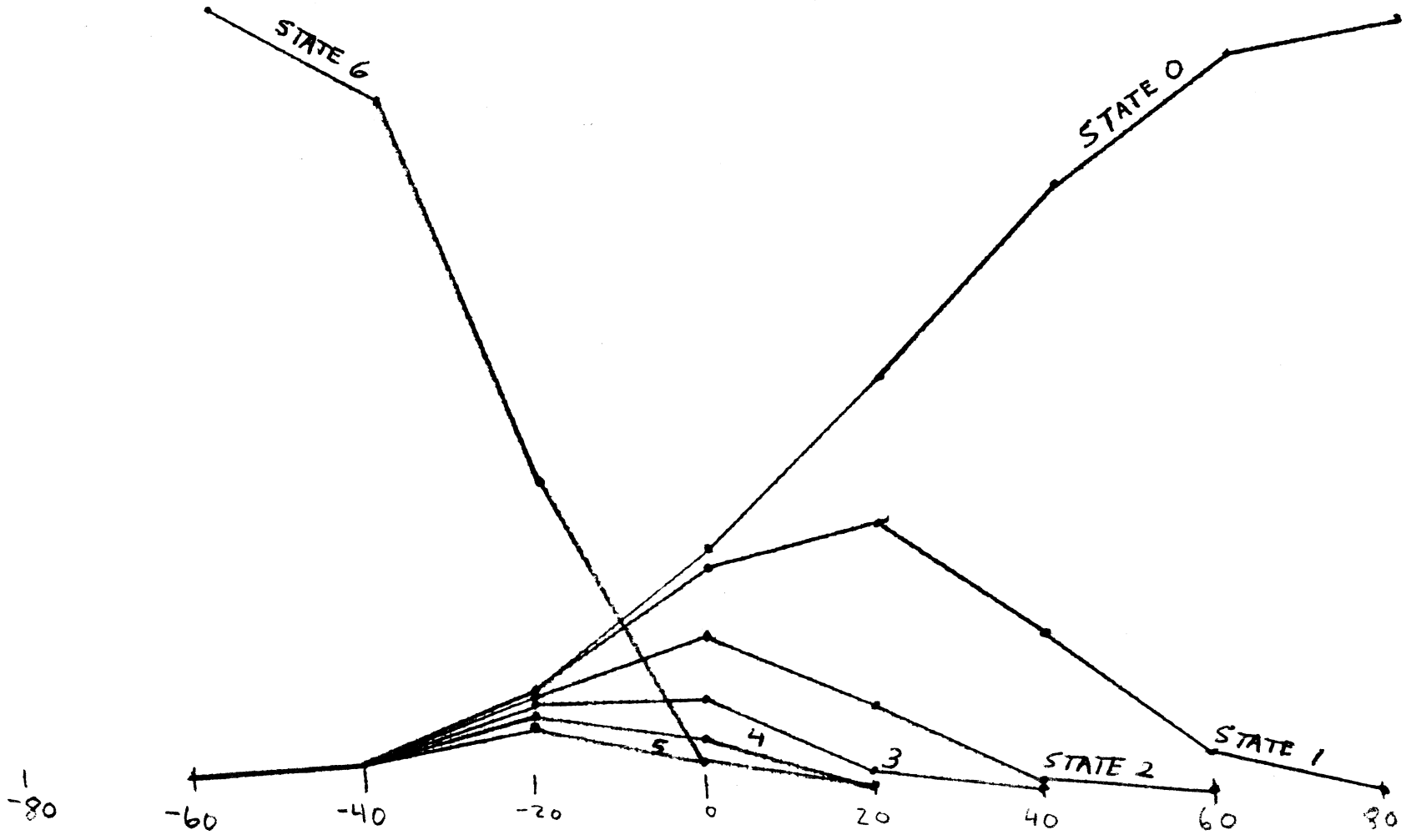


FIGURE 2

LUMPED MODEL STANDARD DEVIATION OF 20



				Theoretical Distribution of Chi-Square Values					
				99	95	90	75	50	5
				Actual Distribution of Chi-Square Values					
Size	St. Dev.	Sample Size	Deg. Freedom						
25	20	93	2	99	96	88	74	60	8
50	20	30	2	100	97	90	73	37	3
75	20	30	2	100	97	97	80	47	3
100	20	44	2	98	95	91	73	36	7
500	20	42	2	95	93	88	74	62	2
1000	20	42	2	100	98	90	83	48	5
25	10	12	2	100	100	92	83	58	8
100	10	36	2	100	97	94	80	50	3
25	40	96	2	100	95	90	79	49	4
100	20	174	2	99	95	91	74	48	6

(Last row uses samples from every 5th time step)

FIGURE 3

				Theoretical Distribution of Chi-Square Values					
				99	95	90	75	50	5
Size	St. Dev.	Sample Size	Deg. Freedom	*Actual Distribution of Chi-Square Values*					
25	20	66	1	100	97	88	73	53	0
50	20	66	4	100	97	92	79	48	0
75	20	51	4	100	100	100	84	61	10
100	20	54	5	96	94	90	68	39	7
500	20	105	6	100	97	93	76	40	1
1000	20	105	6	98	94	88	73	57	7
.25	10	57	3	100	100	95	82	60	16
100	10	48	5	98	96	90	77	54	4
25	40	32	3	100	94	78	78	59	3

FIGURE 4

Second Level Chi-Square Analysis

Using 1st Level Chi-Square From				2nd Level Chi-Square	Deg. Freedom	% Confidence of Acceptance
Size	St. Dev.	Deg. Freedom	Sample Size			
25	20	2	93	8.27	5	10
50	20	2	30	2.65	3	30
75	20	2	30	2.30	3	50
100	20	2	44	4.93	3	10
500	20	2	42	4.88	3	10
1000	20	2	42	3.72	3	25
25	10	2	12	0.17	2	90
100	10	2	36	1.58	3	50
25	40	2	96	2.54	5	75
25	20	1	66	2.05	3	50
50	20	4	66	3.43	3	30
75	20	4	51	1.38	3	70
100	20	5	54	4.63	3	20
500	20	6	105	7.67	5	10
1000	20	6	105	5.27	5	30
25	10	3	57	1.80	3	50
100	10	5	48	0.56	3	90
25	40	3	32	8.33	3	2.5
25	20	2	174		5	90

(Last row uses samples from every fifth time step)

FIGURE 5

ERROR PROPAGATION

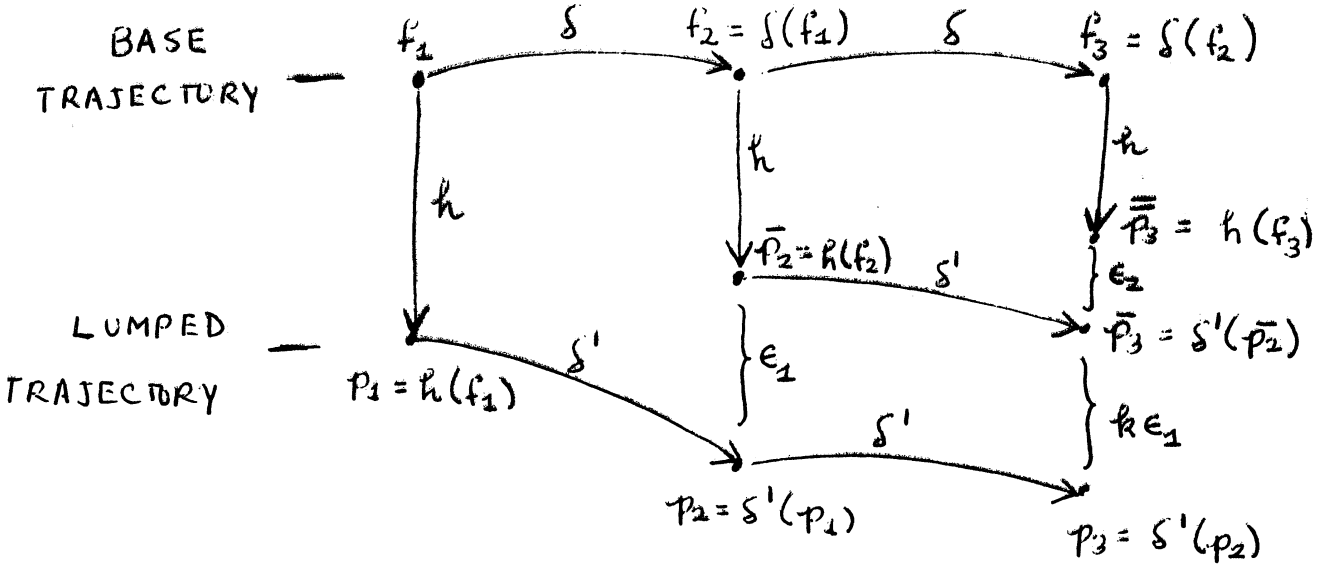


FIGURE 6

FIGURE 7

FIRING LEVEL (RECOVERY STATE 0)

LUMPED

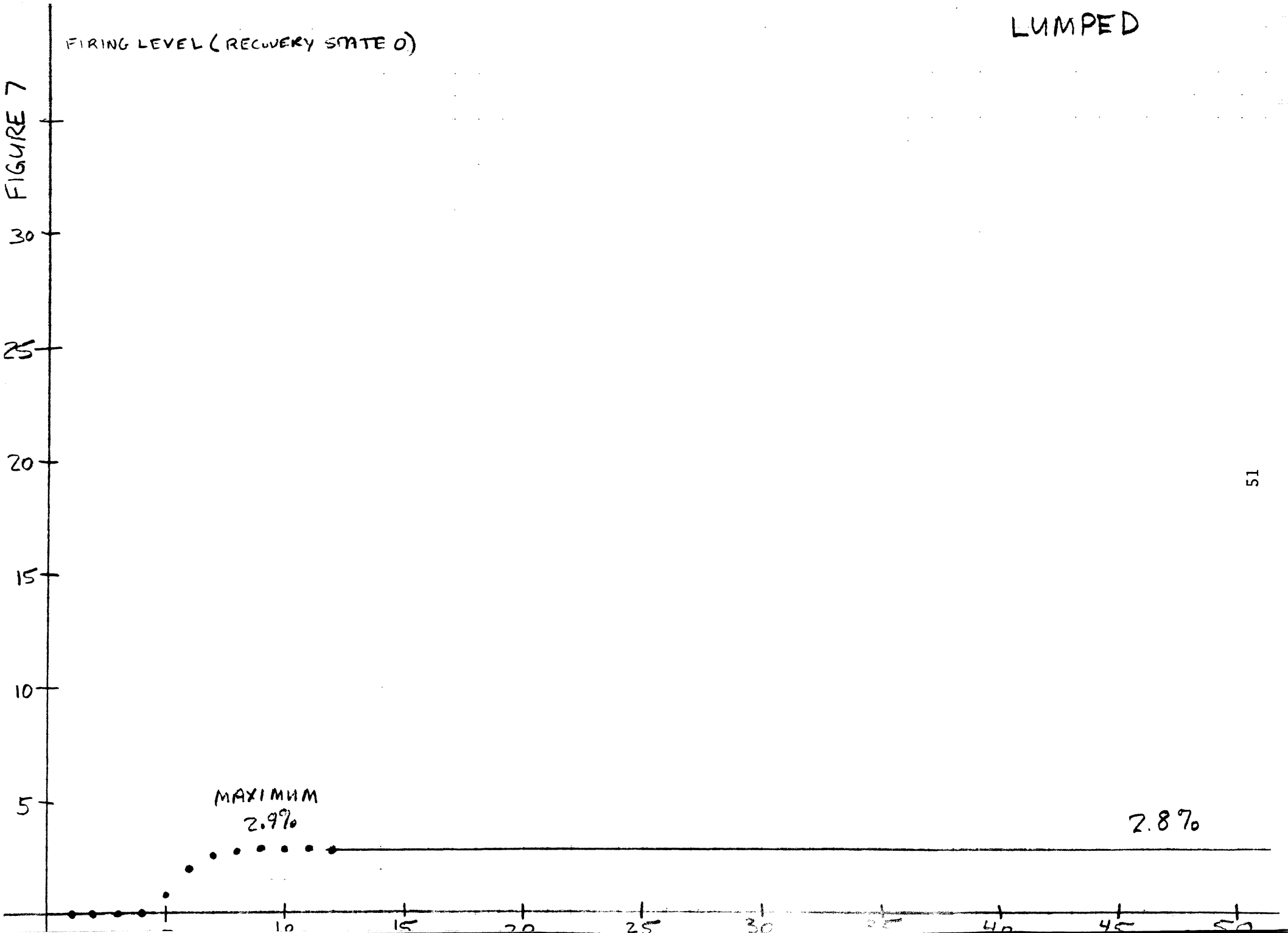


FIGURE 8

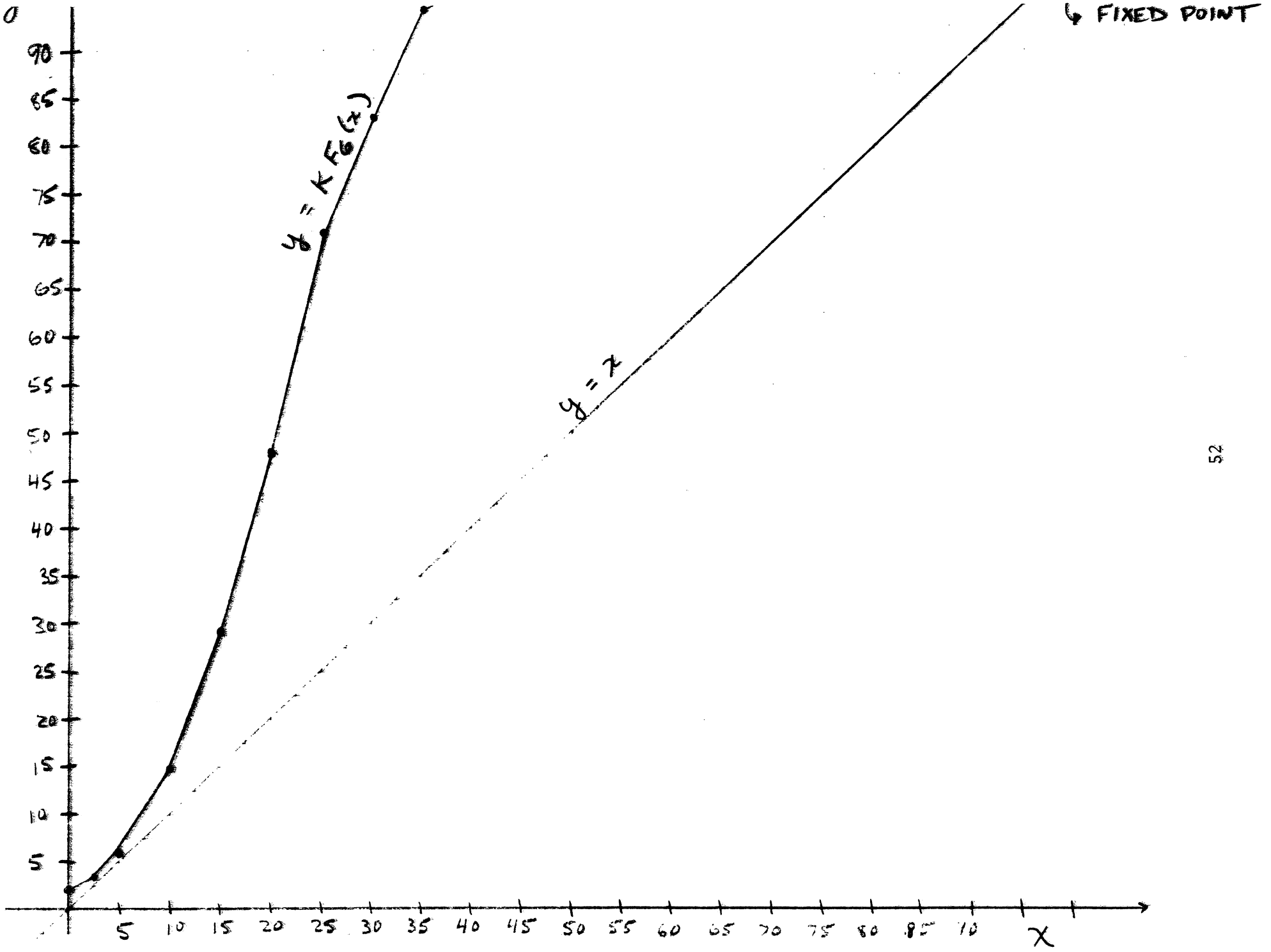

0
90
85
80
75
70
65
60
55
50
45
40
35
30
25
20
15
10
5
y = KF6(x)
y = x
↳ FIXED POINT
5 10 15 20 25 30 35 40 45 50 55 60 65 70 75 80 85 90
X

FIGURE 1

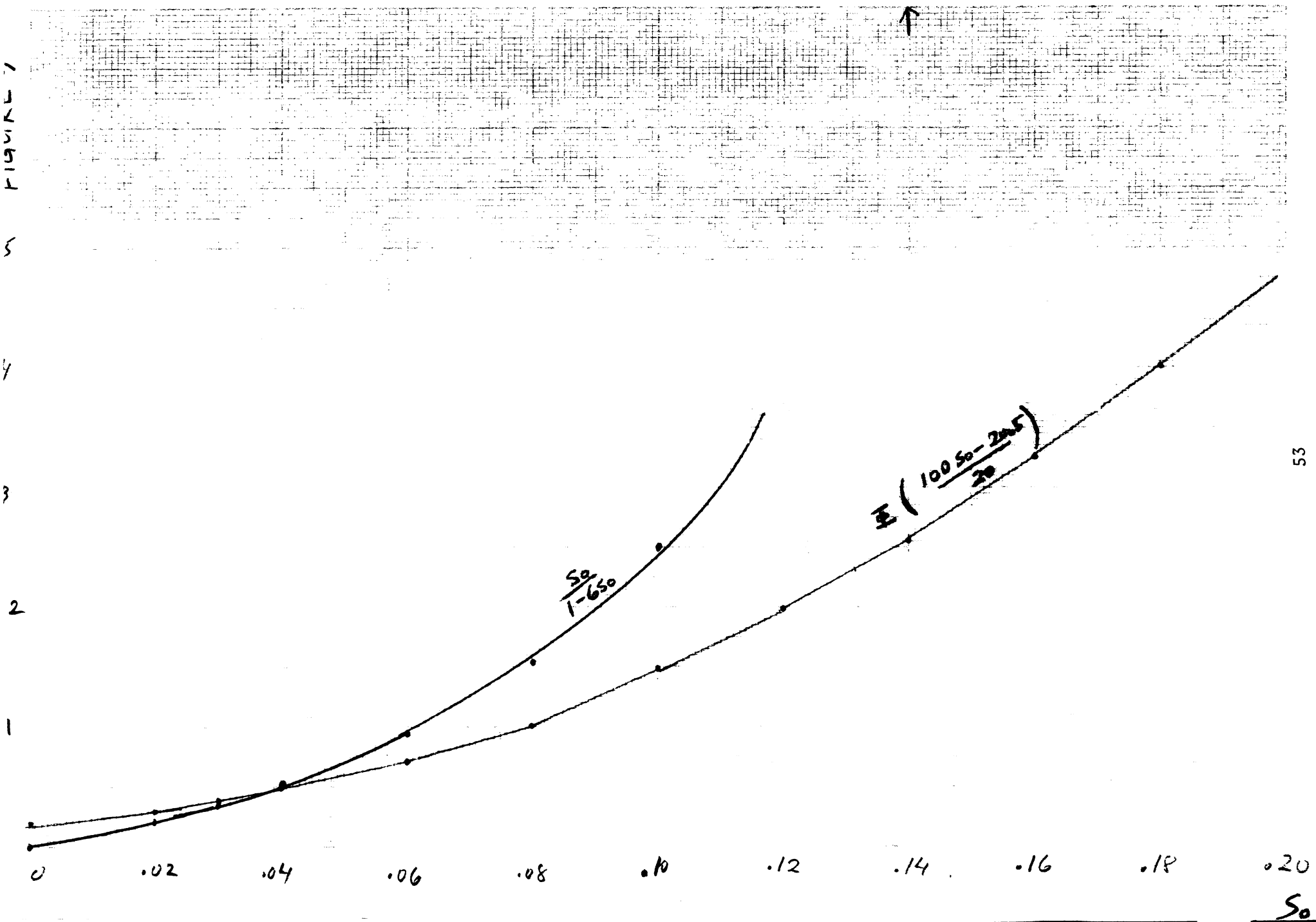

5
4
3
2
1
$\frac{S_o}{1-6S_o}$
$\Sigma\left(\frac{100\,S_o - 2}{20}\right)$
0
.02
.04
.06
.08
.10
.12
.14
.16
.18
.20
S_o

FIGURE 10

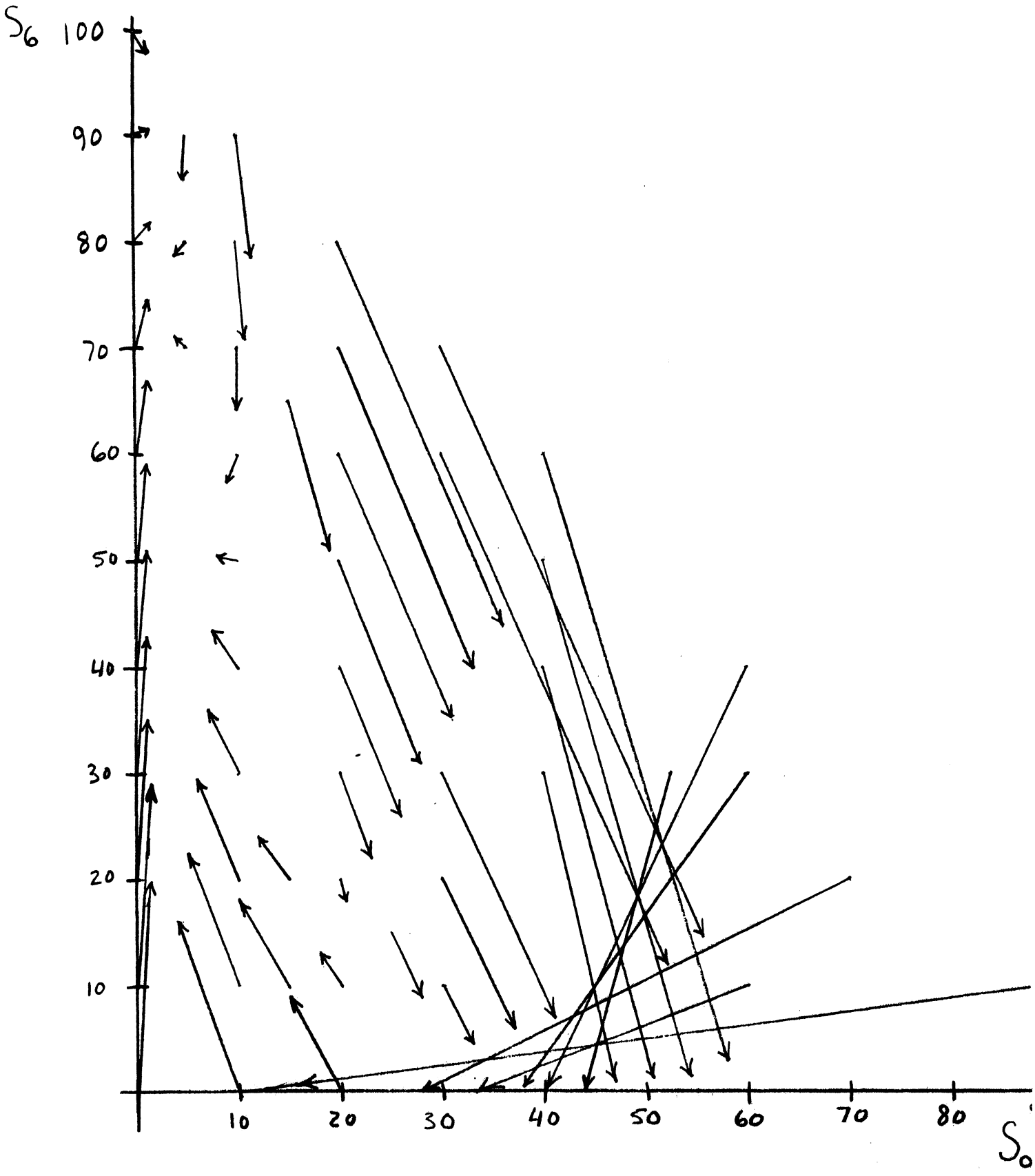
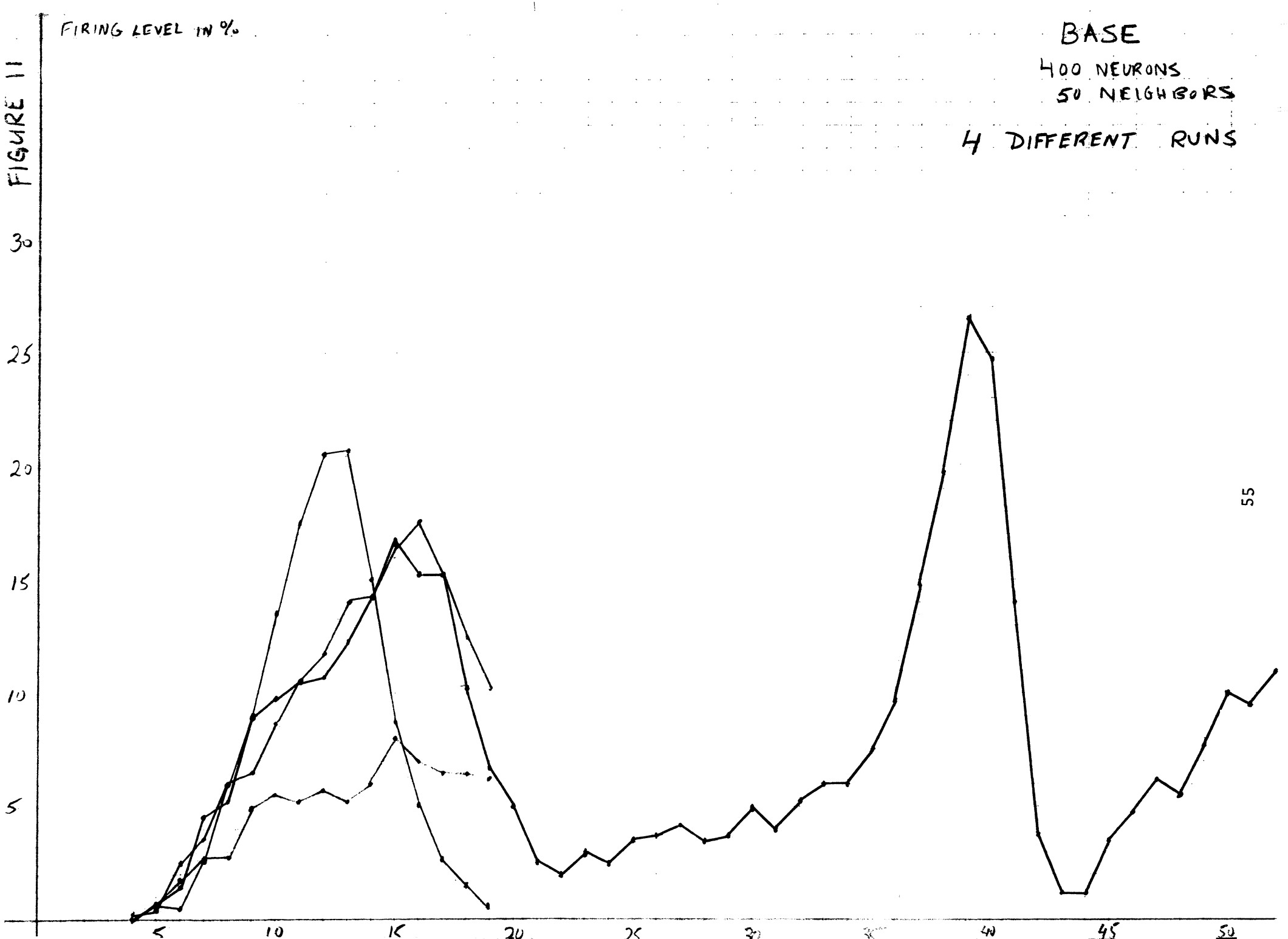


FIGURE 11


FIRING LEVEL IN %
BASE
400 NEURONS
50 NEIGHBORS
4 DIFFERENT RUNS
30
25
20
15
10
5
5
10
15
20
25
30
40
45
50
55

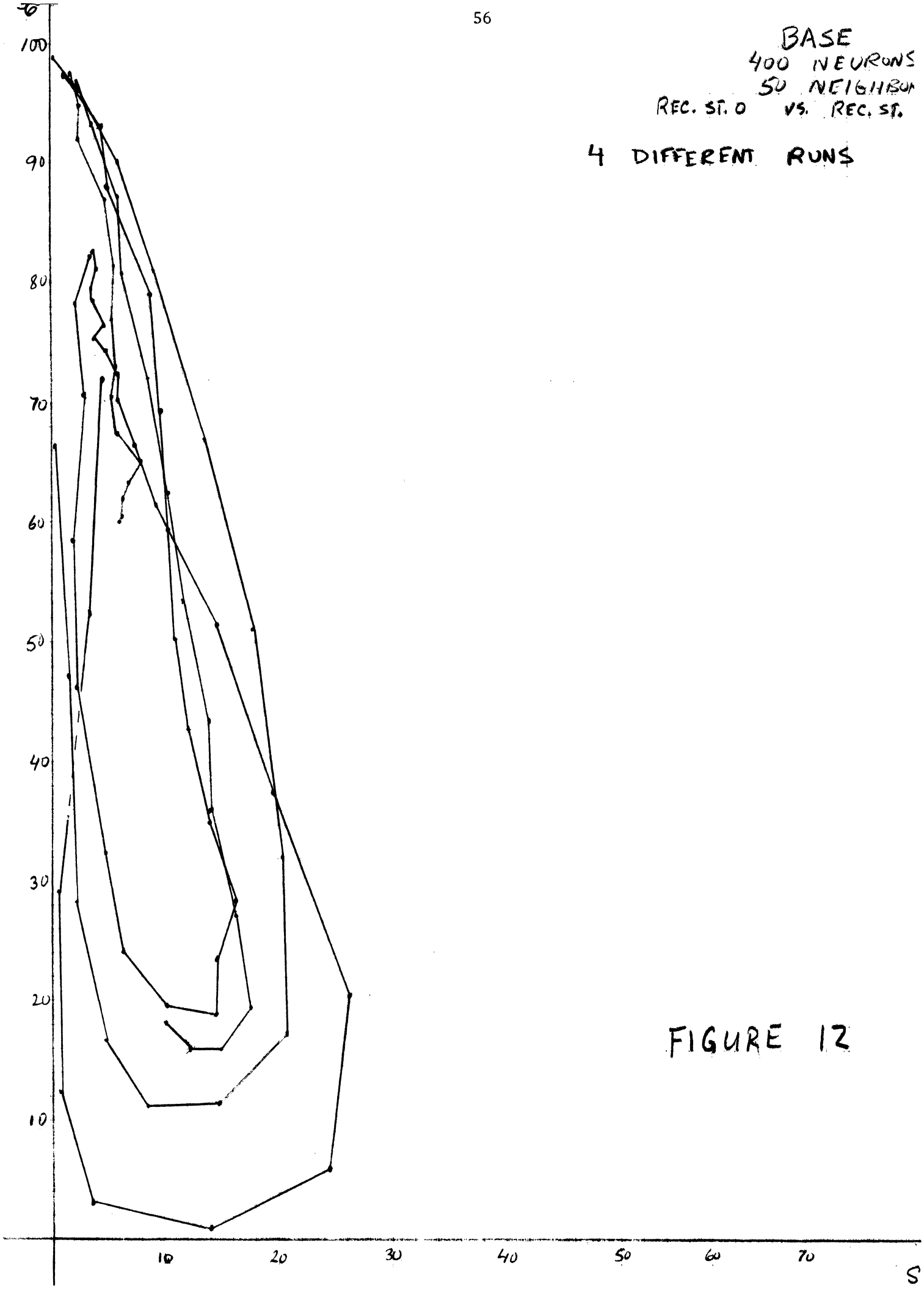
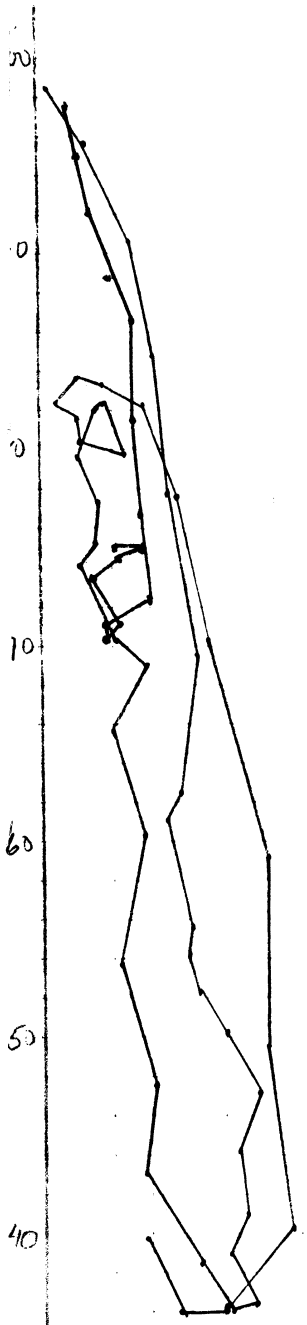

BASE
400 NEURONS
50 NEIGHBOR
REC. ST. 0 VS. REC. ST.
4 DIFFERENT RUNS
100
90
80
70
60
50
40
30
20
10
10
20
30
40
50
60
70
S

FIGURE 12



BASE
400 NEURONS
100 NEIGHBORS
REC. ST 0 VS. REC ST 6

2 DIFFERENT RUNS

FIGURE 13

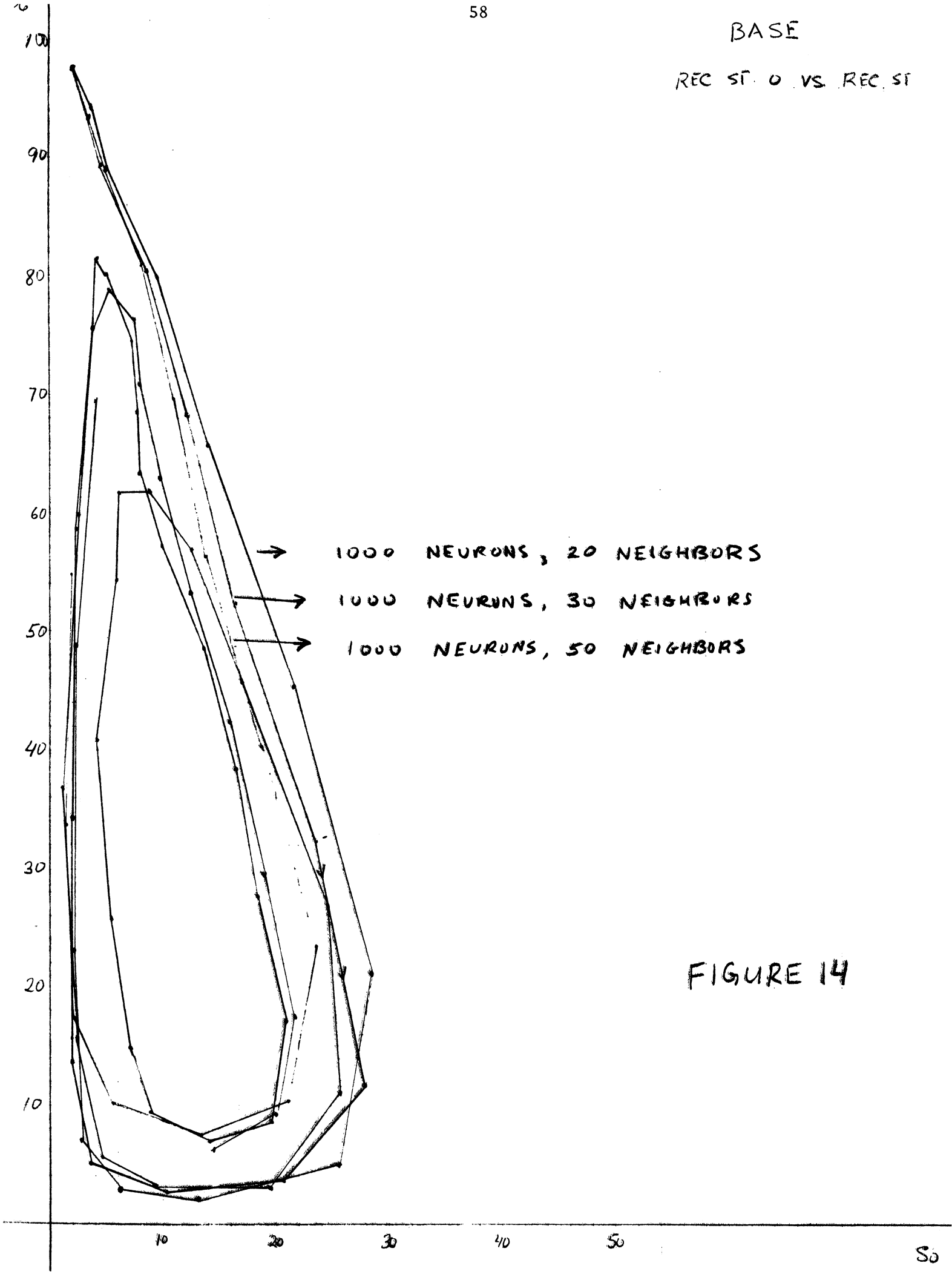

BASE
REC ST. 0 VS. REC ST
100
90
80
70
60
50
40
30
20
10
10
20
30
40
50
50
1000 NEURONS, 20 NEIGHBORS
1000 NEURONS, 30 NEIGHBORS
1000 NEURONS, 50 NEIGHBORS

FIGURE 14

BASE
400 NEURONS
60 NEIGHBORS
REC. ST. 0 VS. REC. ST. 6

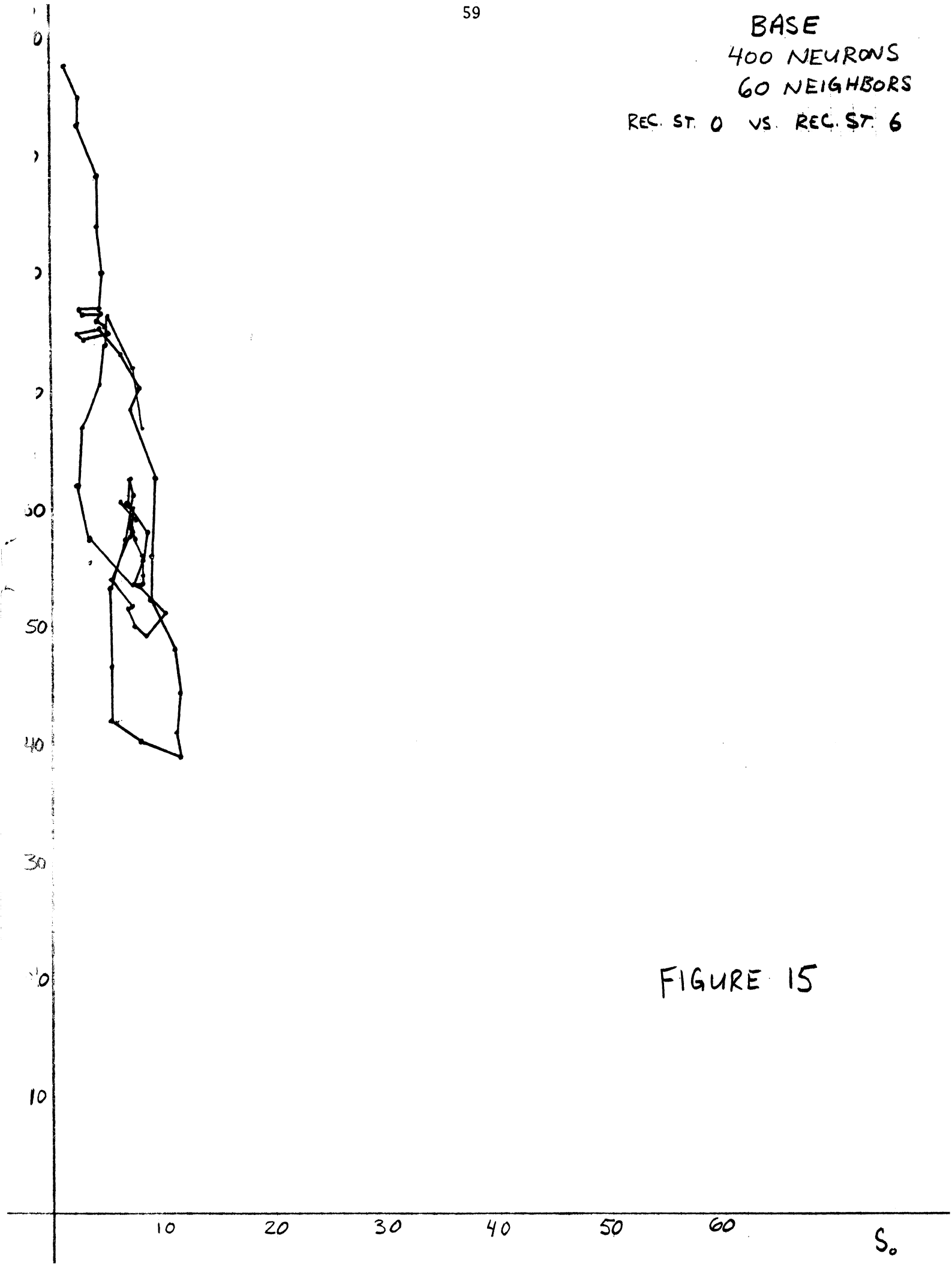


FIGURE 15

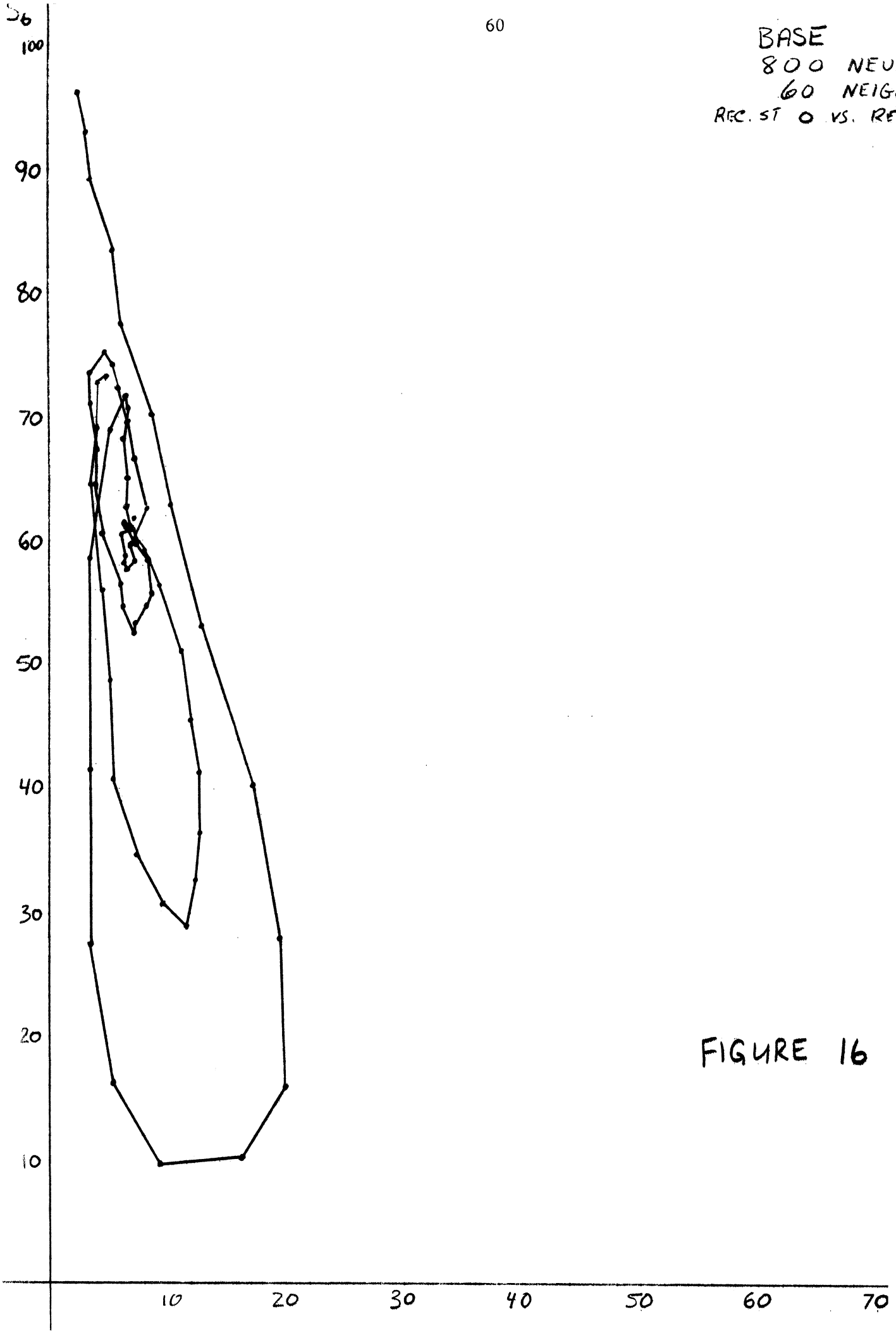

BASE
800 NEURONS
60 NEIGHBOR
REC. ST 0 VS. REC. ST. 6
100
90
80
70
60
50
40
30
20
10
10
20
30
40
50
60
70

FIGURE 16

BASE

REC ST. 0 VS REC.ST 6

50 NEURONS
50 NEIGHBORS

2 DIFFERENT RUNS

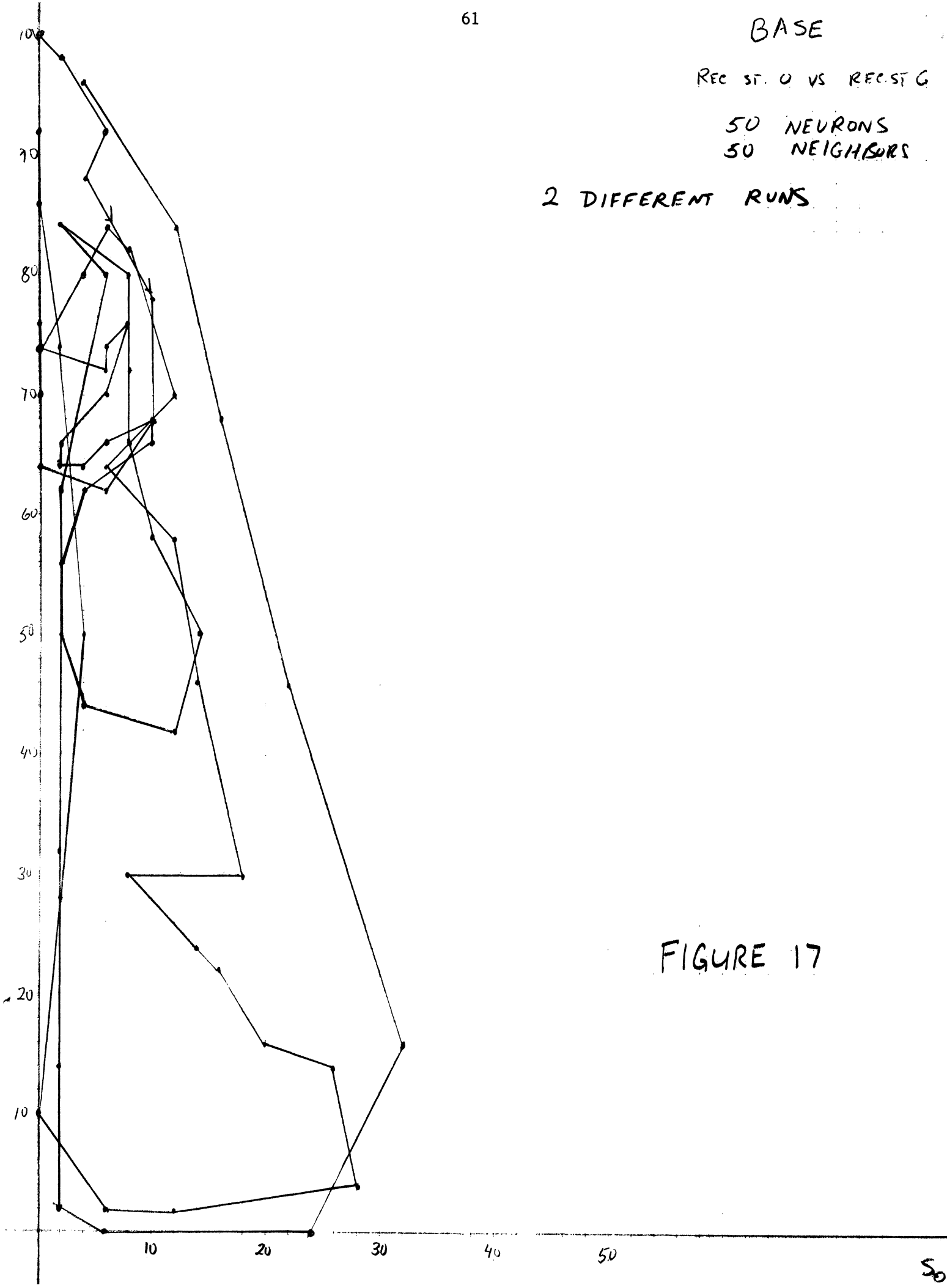
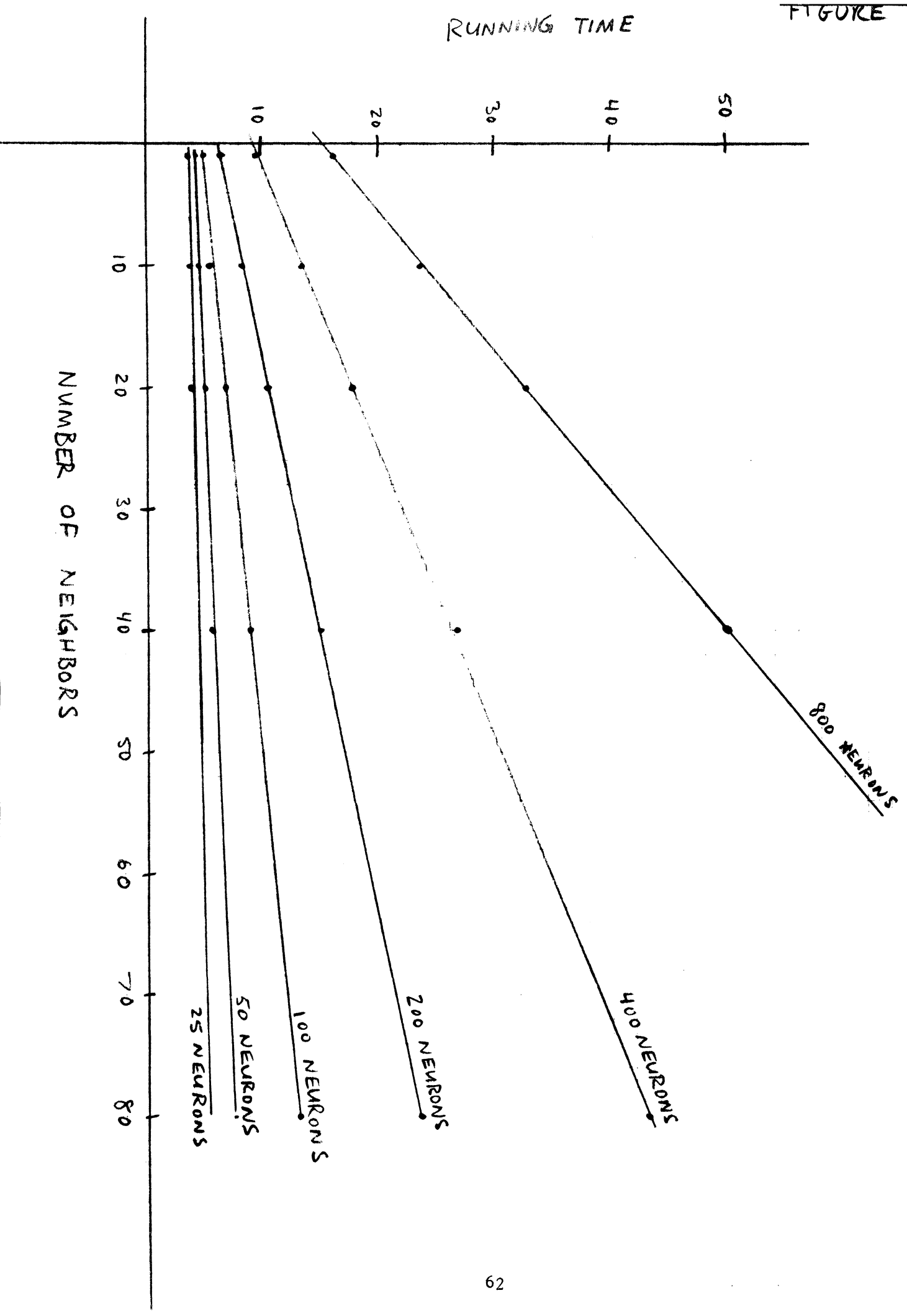
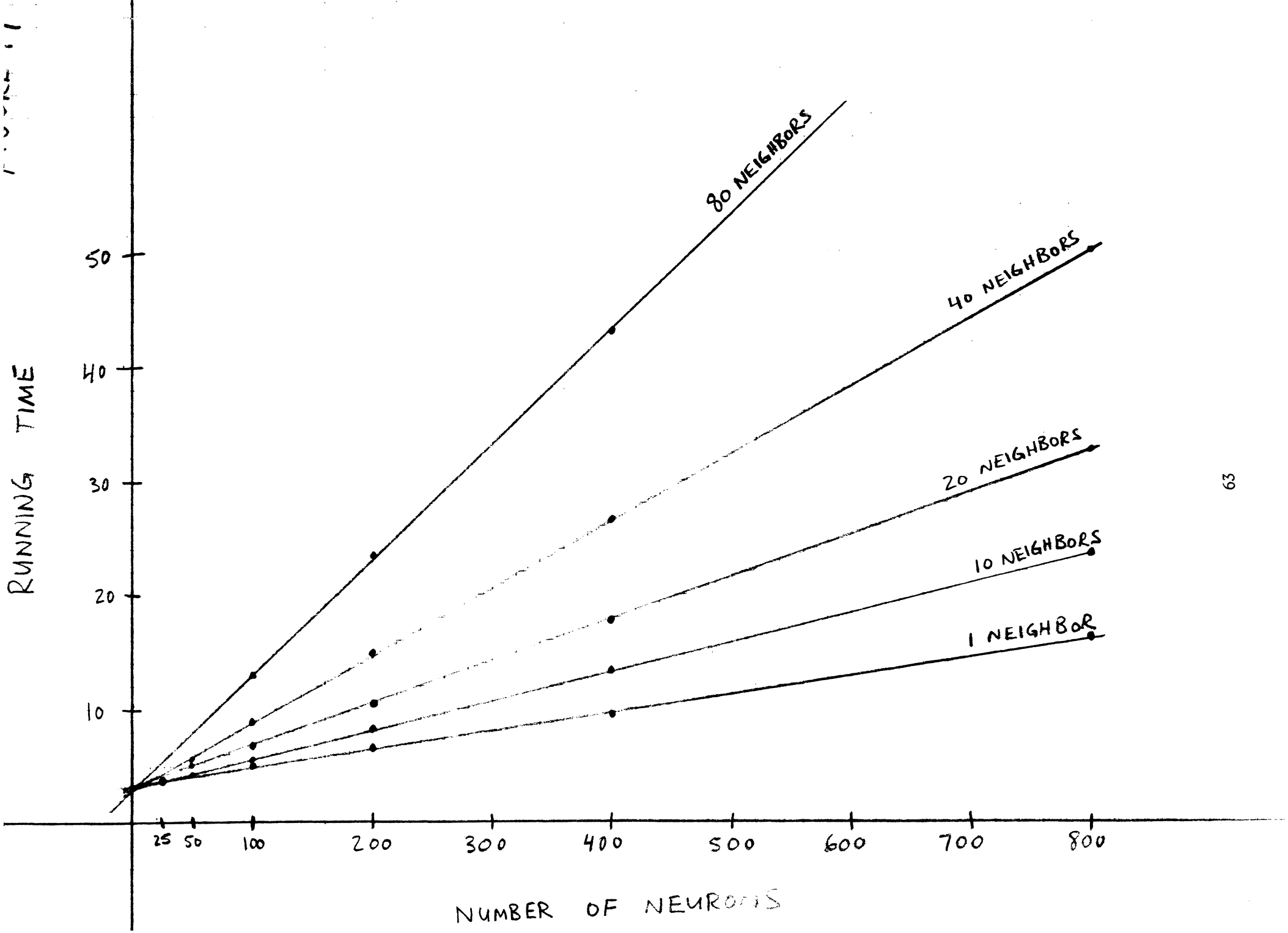


FIGURE 17

FIGURE 18


RUNNING TIME
10
20
30
40
50
NUMBER OF NEIGHBORS
10
20
30
40
50
60
70
80
800 NEURONS
400 NEURONS
200 NEURONS
100 NEURONS
50 NEURONS
25 NEURONS


80 NEIGHBORS
40 NEIGHBORS
20 NEIGHBORS
10 NEIGHBORS
1 NEIGHBOR
RUNNING TIME
10
20
30
40
50
25
50
100
200
300
400
500
600
700
800
NUMBER OF NEURONS