DISTRIBUTED ADAPTATION IN NEUROMIME NETWORKS

V. V. GRIFFITH
G. H. BOLEN

GOODYEAR AEROSPACE CORPORATION

FOREWORD

This study was initiated by the Biomedical Laboratory of the Aerospace Medical Research Laboratories, Aerospace Medical Division, Wright-Patterson Air Force Base, Ohio. The research was conducted by the Avionics and Electronics Research and Development Division of the Goodyear Aerospace Corporation (GAC), Akron, Ohio, under contract AF33(615)-1891. Mr. V.V. Griffith directed the research for the Goodyear Aerospace Corporation. Lieutenant Colonel Jack E. Steele of the Mathematics and Analysis Branch, Biodynamics and Bionics Division, was the contract monitor for the Aerospace Medical Research Laboratories. The work was performed in support of project 7232, Research on the Logical Structures and Function of the Nervous System," and task 723203, "Principles of Neural Organization." The research covered herein was performed between June, 1964 and June, 1966.

The authors thank Mr. N.D. Diamantides of the Goodyear Aerospace Corporation for his contribution to the Kolmogorov prediction theory section of the report.

This report is filed by the Goodyear Aerospace Corporation as report GER-12738.

This technical report has been reviewed and is approved.

J. W. HEIM, PhD
Technical Director
Biomedical Laboratory
Aerospace Medical Research Laboratories

ABSTRACT

This report describes investigations of networks with adaptive ability distributed through them. It is thought that large-scale adaptive systems can be constructed of adaptive building blocks. These adaptive systems would be flexible in function, reliable and would resist severe damage characteristics of living creatures. Neuron models were tested by interconnecting them into various networks to perform simple control tasks. The test results were evaluated and the evaluation used to improve the theory and the neuron model. Two basic analysis methods were used to study neuromime networks: a sequential machine analysis and an optimal process method applying Pontryagin's maximum principle. The sequential analysis method proved unsatisfactory when applied to an attempted description of an adjustment rule for the neuron model. This difficulty led to an application of optimal processes. The application of Pontryagin's maximum principle to the analysis of the neuron model network described both optimum conditions for a system and criteria useful for developing the adjustment rule.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

SECTION I

INTRODUCTION

This program is an effort to discover whether adaptive systems can be built that are modeled on the known characteristics of neurons, with adaptive ability distributed among the elements. Reference 1 presents evidence that each neuron in living creatures is a self-contained adaptive unit. Specifically, each neuron receives input signals from other neurons. On the basis of these inputs and outputs, each neuron uses some internal criteria to decide whether or not it should produce an output. The adaptation in a neural net is not centralized, but distributed throughout the net with each neuron contributing to the whole. A neural net, then, is less susceptible to damage than systems using centralized control.

The basic premise of the work presented here is that large-scale adaptive systems with great flexibility can be constructed of adaptive building blocks. Each building block adjusts its own behavior according to some relatively simple rule. Reference 2 points out that systems based on this concept should tend to display the wide range of plasticity, reliability, and ability to operate effectively after severe damage that is displayed by living creatures.

To put the results of the research to date in proper perspective, it is necessary to review some of the ideas that were instrumented at the beginning of the research. Present views can then be effectively contrasted with the early views. At the beginning of the program it was believed that each neuromime could adjust itself by considering solely the time histories of its inputs and outputs. The adjustment rule used required the neuron to attempt to minimize its inputs and output while providing the necessary control of its environment. More formally, a "system power" function, $\mathcal{P}$, was defined by

$$\mathcal{P} = R_o^2 + \sum_1^n R_i^2 \tag{1}$$

where R_o and R_i, $i = 1, 2, \ldots, n$, are respectively the neuron's output rate and various input rates. It is readily shown that if the appropriate partial derivatives exist and are well-defined, a minimum of $\mathcal{P}$ is obtained when

$$R_o = -\sum_1^n \frac{\partial R_i}{\partial R_o} R_i \tag{2}$$

For adaptation, the system had to "test" its environment, discover with sufficient accuracy the values of the partial derivatives of Equation 2, and adjust its response accordingly.

Two concepts were considered that incorporated the adjustment rule. In Concept 1, the neuron model adjusted the synaptic weights of its inputs. If a neuron model had 1000 inputs, it would compute 1000 synaptic weights. In Concept 2, the neuron model adjusted the synaptic weight associated with its own output. If the neuron model synapsed on 1000 other neuron models or had 1000 inputs, it would compute only one synaptic weight. Previous research sponsored by GAC used Concept 1. It was believed, however, that Concept 2 was more plausible physiologically. It seemed reasonable that a neuron could more readily adjust the effects of its own endbulbs than the endbulbs of other neurons. Also, some research by Dale (Reference 3) has indicated that all of the endbulbs of any particular neuron are either excitatory or inhibitory. The most recent work uses a combination of both concepts.

In this report the word "neuromime" refers to neuron models that use the transistor circuit developed by L.D. Harmon (Reference 4), which he called the neuromime. The words "neuron model" refer to general simulations of neuron-like elements that do not use Harmon's neuromime as a major functional component.

SECTION II

EXPERIMENTAL WORK

A. GENERAL

Various neuron model networks were tested to further the development of a synaptic weight adjustment rule. The tests and their results are described in detail in Appendix I. The development of the neuron model is described in Appendix II. The material discussed in this section relates the test results and evaluations of the specific areas of study to the development of synaptic weight adjustment rules.

B. FEEDBACK INFORMATION

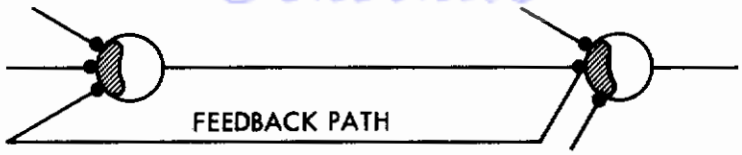
The broad objective of this area of study was to determine the amount and type of information that the neuron model needed to adjust its synaptic weights. As a portion of the broad objective, the effect that feedback information had on the neuron model and its adjustments was studied. Feedback, as used in this context, refers to inputs to a neuron model that are functions of this neuron model's output. As part of the experimental program, various feedback paths were tested and evaluated. For example, the neuron model's own output was fed back both directly and through another neuron model.

Figure 1 shows the two basic feedback concepts that were studied. In physiological terms, the first feedback concept considered feedback paths coming from a neuron model's own synapse. In this concept the neuron model received information about what its own synaptic-weighted output was doing, ignoring any effect that other neuron models might have on the post-synaptic membrane. The second concept considered feedback paths coming from beyond the synaptic junction. The neuron model using the second concept received, in some form, a measure of the post-synaptic potential. In other terms, the neuron model received information about how its own synapse and those of others affected the post-synaptic membrane.

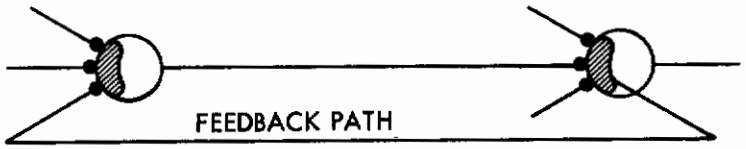
Certain general results were obtained from the tests on the two types of networks. Neuron models using Concept 1 had difficulty computing synaptic weights. Further, the synaptic weights, in certain cases, had the wrong value or even the wrong sign. The tests of neuron models using Concept 2 showed the need for post-synaptic potential information, as suggested by the poor response of neuron models using Concept 1.

The testing of neuron models based on the application of optimal control theory, which will be discussed in Section III, also supports the use of Concept 2. The optimal control theory, in addition, supplied a mathematical tool for describing the feedback relationship.

In general, the feedback path from the post-synaptic potential to the neuron model's input must be weighted by a function of the synaptic weight. Presenting this idea figuratively, the neuron model must "look" through a tinted "lens" and measure the intensity of a light. The "lens" represents the feedback path, the degree of tinting represents a function of the synaptic weight, and the light represents the post-synaptic potential. The neuron model, then, uses the "filtered" post-synaptic potential, along with other information, to adjust its synaptic weights. This general view of the feedback relationship was supported by all of the tests as well as by the optimal control theory. At present, the writers know of no physiological mechanism that will perform the needed communication function. The d-c currents observed by Becker (Reference 5) and the field effects and currents seen by Kohler, et al, (Reference 6) are likely candidates.



CONCEPT 1 - FEEDBACK OF A NEURON MODEL'S OWN OUTPUT



CONCEPT 2 - FEEDBACK OF POST-SYNAPTIC POTENTIAL (PSP) OF DISTAL NEURON MODEL

Figure 1. Two Basic Feedback Concepts

C. RECOVERY AFTER DAMAGE

The objective of this study area was to support one of the premises concerning neuromime networks set forth at the beginning of the contract. The premise was that the neuron models have the ability to readjust their behavior in such a way that system control can be maintained following damage to the system. Damage to the system had two forms: damage to the environment and damage to the neuron model network. Both forms were used to evaluate the network. It should be stated that a network's ability to recover after damage actually consists of two mechanisms, its ability to readjust its own synaptic weights, and the interconnection structure of the neuron models within a network. The first mechanism is discussed here. The second mechanism is discussed in subsection D(Redundancy). The ability of the neuron models to recover must somehow appear in the synaptic weight adjustment rule. Evaluation of the tests on the synaptic weight adjustment rules proved that the neuron models have this ability (refer to Appendix I for details of the testing).

A further general observation supported the view that networks with distributed adaptation were less vulnerable to damage than networks without it. That is, when damage occurred, each neuron model in the network affected by the damage readjusted its synaptic weights to absorb the extra burden created by the damage. The new equilibrium state for each of the neuron models after the readjustment was such that each model was at a minimum energy state. Without this form of distributed adaptation, some neuron models absorbed more of the extra burden than they should and other neuron models absorbed less. As a result, the subsequent adjustment range was limited, and thus the networks were more susceptible to damage.

The experimental work done in this study area aided other study areas. For example, as discussed in subsection B, feedback Concept 1 was discarded as a result of tests designed to evaluate the synaptic weight adjustment rule in terms of the neuron's ability to recover after damage. From this example a general result was implied: the neuron model must receive some measure of information on the activity going on about it before it can adjust to damage. Another example is found in the discussion of redundancy in subsection D. Some redundant network interconnections were inadequate to ensure satisfactory recovery after damage. The redundant network in question did adjust to damage, but the final equilibrium state of the neuron models restricted any further adjustment.

D. REDUNDANCY

The objective of this study area was to develop network structures that would increase the network's ability to adapt to damage. The work in this area is closely related to the work of the previous study area. The difference is that attention here is focused on the interconnections of neuron models, rather than on individual neuron models.

The networks tested were simple, consisting of from two to four neuron models acting in parallel. Figure 2 shows more fully what "neuron models acting in parallel" means in this context. There are three basic relationships. The first is the common sensory input. The second is the combination of all neuron model outputs by a single operation. (If the operation is the addition operation, then the network is in parallel by the narrow, more common sense

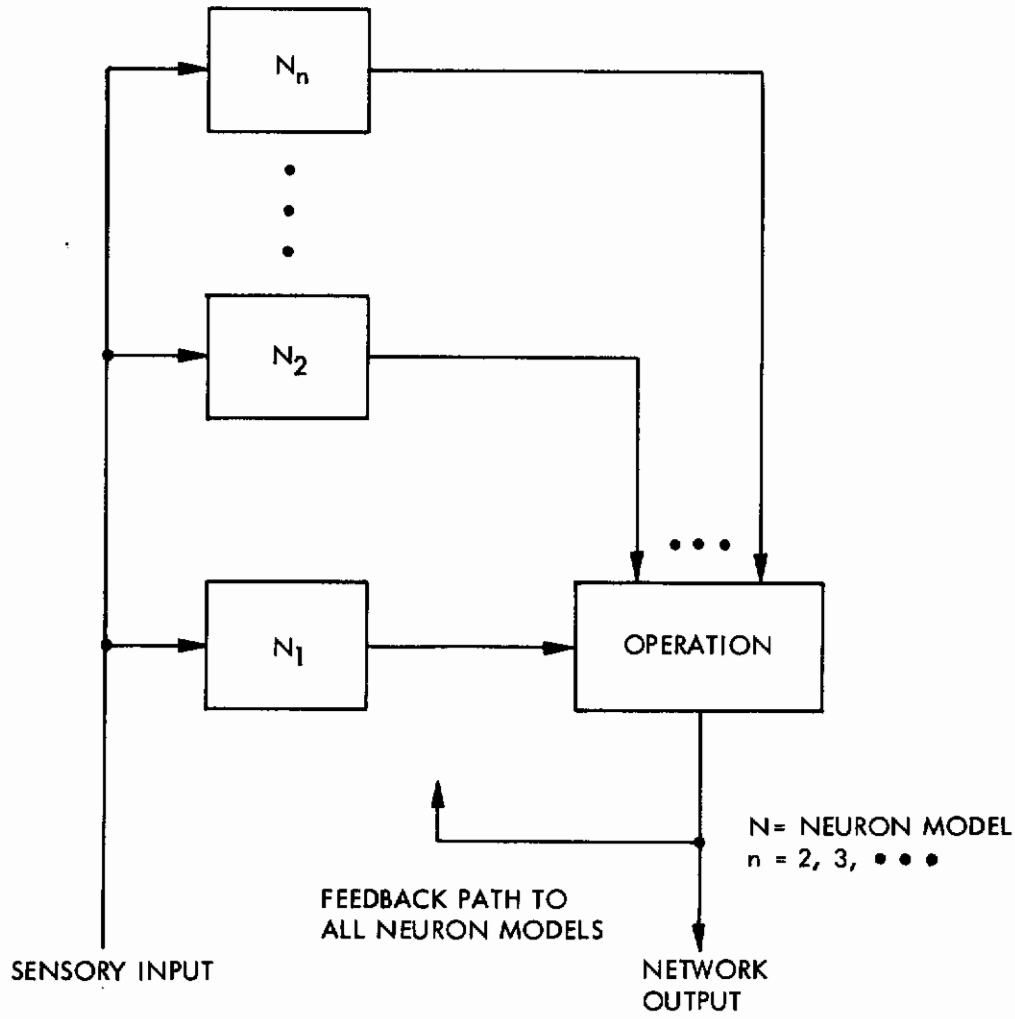


Figure 2. Redundant Network - n Neurons in Parallel

of the word; for example, two resistors put in parallel.) The third relationship is the common feedback of the network output to all neuron models in the network. It should be mentioned that although Figure 2 shows a single, common sensory input, a particular neuron may be a member of several such parallel redundant networks. Using certain correlation techniques, a neuron model can decide which of its responses are affecting which redundant networks. The experimental work done on this study area ignored the other possible redundant networks and concentrated on the properties of a single redundant network.

The operation, shown in Figure 2, was the variable in the experimental work. Various operations were tested and evaluated. In general, the operations tested gave good results. That is, the redundant network improved the damage adjustment range beyond that of the component neuron models. One particular operation had significant physiological value. A concensus operation was tested that resembled the way that actual neurons react to inputs. The output from an actual neuron seems to be closely related to the "majority vote" of its inputs. The test network using the concensus operation contained four interconnected neuron models. The network adapted to failures of one, two, or three neuron models to maintain a constant input-output relation.

Some general results were obtained. A fewer number of elements is required to perform a specific redundant task using the neuron model structure than was originally thought necessary. Further, redundant networks using a neuron model structure maintain a constant relationship between the input and output of the network. In other words, the input-output relation after damage is not just about the same as before the damage, it is the same. The neuron model structure for redundant networks has not been thoroughly studied. However, both experimental and analytical work performed to date show promise that larger networks using many interconnected redundant networks will display the reliability and damage recovery characteristics observed in living creatures.

E. CONVERGENCE OF SYNAPTIC WEIGHT ADJUSTMENTS

The objective of this study area was to demonstrate how the synaptic weights converged to optimum values. The synaptic weight adjustment rule derived from optimal control theory did not define the way the synaptic weight adjustments interacted. In a network with many neuron models, many synaptic weights had to be adjusted simultaneously so that the equilibrium state for all synaptic weights would be optimum. Appendix I shows results of tests designed to demonstrate how various synaptic weights converged to optimum values.

The general results obtained from the tests can be related to observed physiological data. The tests showed that the synaptic weights adjusted rapidly to optimum values at the outset, gradually getting slower as the optimum values were approached. This relates to the learning period observed in creatures.

The test results indicated that the synaptic weights adjust to a non-optimum point that was a satisfactory answer to the control task. As time progressed and "learning" progressed, the synaptic weights finally arrived at the optimum values for that specific control task. This result filled in the details of the learning curve for creatures. More intuitively, this result says that creatures do not try to solve their specific problems optimally the first time. Instead, creatures just try to solve their problems satisfactorily. Then, after many refinements, creatures arrive at the best solution to their problems.

One other subtle result was observed. When the synaptic weights were within a region containing their optimum values, any further changes in the weights had a negligible effect on the total system control. In other words, a region of suboptimal synaptic weight values satisfied the control requirements. The neuron models did not have to compute the optimum values for their synaptic weights. Adjusting the synaptic weights close to the optimum values was satisfactory. If the suboptimal region is large enough,in general, then the accuracy of the synaptic weight adjustment rule is not so critical.

SECTION III

ANALYSIS

A. GENERAL

In the attempt to discover an analytic method of describing adaptive systems and their relationships, two basic approaches were investigated - a sequential machine approach and an optimal process approach applying Pontryagin's maximum principle. The optimal process approach showed great promise and was expanded.

B. SEQUENTIAL MACHINE APPROACH

1. General Analysis

A sequential machine is a collection of states together with a set of allowed transitions between states. The theory of such machines has been extensively applied in the study of digital computers and in the study of the context-free finite state grammars of coding theory. Figure 3 shows a typical example - a sequential machine model of Morse code transmission. The vocabulary is considered to be four symbols - dot, dash, letter space, and word space. The constraint on the system is that two spaces can never occur in succession.

To analyze such a machine, one first defines a state vector. Consider a machine with n states. Let the states be numbered by any convenient method. Form a column vector S with n components. All of the components of the vector will be zeroes except one. One component will be a 1. The number of the component that is a 1 is the number of the state the machine occupies at that instant.

If a particular machine is completely deterministic, the time sequence of the transitions made is a function solely of the initial state and the string of symbols. Matrix A can be written, containing input symbols, which satisfies the function

$$S' = \alpha \cdot AS$$

where

α is the input (or output) symbol at time t

S is the machine state at time t

S' is the machine state at time t + 1

$\cdot$ is a multiplication defined by

$$\alpha \cdot \beta = 1, \ \alpha = \beta; \ \alpha \cdot \beta = 0, \ \alpha \neq \beta$$

We call this matrix the transition matrix.

As an example, the machine of Figure 4 is described by the matrix

$$A = \begin{bmatrix} c & a & b \\ b+e & c & a \\ a+e & b & c \end{bmatrix}$$

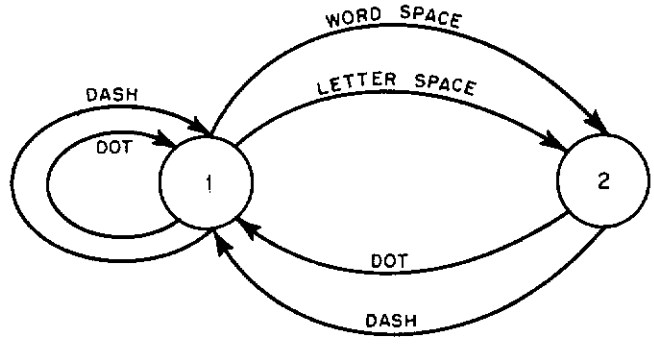


Figure 3. Morse Code Sequential Machine

Assume the machine is in state 1 at t = 0, and an input b occurs. The computation

$$S = b \cdot \begin{bmatrix} c & a & b \\ b & c & a \\ a & b & c \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$$

$$= \begin{bmatrix} 0 & 0 & 1 \\ 1 & 0 & 0 \\ 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix}$$

shows that the machine will be in state 2 at t = 1.

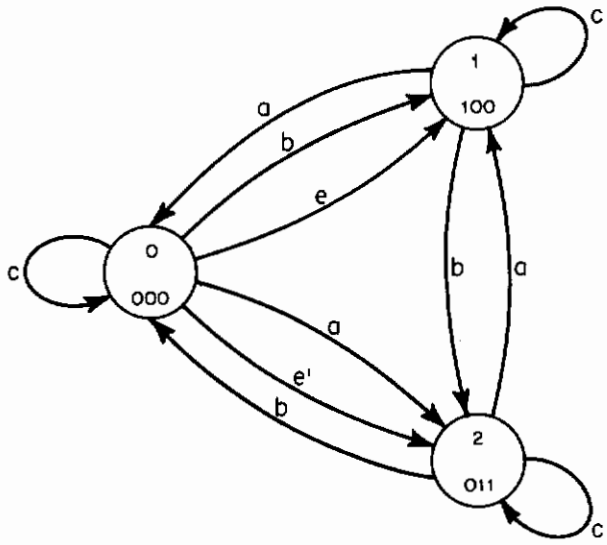


Figure 4. Sequential Machine

2. Application of Finite-State Markov Processes

If machine transitions are assumed to occur randomly instead of resulting from a particular sequence of symbols, one obtains a matrix B for the machine in which b_{ij} is the probability that state i will follow state j. Such a formulation, of course, makes the machine into a finite-state Markov source - another field of study that uses the theory of sequential machines.

The probability that the machine will occupy a particular state after a long time is obtained by finding the eigenvector of the matrix B corresponding to its unity eigenvalue and normalizing the vector so that the sum of its components is 1. Questions of the existence and uniqueness of the eigenvector have been elsewhere analyzed, and will not be discussed here.

As an example, assume that the probabilities of the various transitions in Figure 4 are:

a = b = e = 0.1

c = selected as necessary. (The sum of the probabilities of the transitions leaving a state must be unity.)

$$B = \begin{bmatrix} 0.6 & 0.1 & 0.1 \\ 0.2 & 0.8 & 0.1 \\ 0.2 & 0.1 & 0.8 \end{bmatrix}$$

Routine manipulation reveals that

$$P_S = \begin{bmatrix} 0.2 \\ 0.4 \\ 0.4 \end{bmatrix},$$

indicating that the 0 state will be occupied two-tenths of the time and the 1 and 2 states will each be occupied four-tenths of the time.

The above example assumed two conditions: that all entries in the transition matrix were non-negative, and that the various transition probabilities were normalized. The following example shows that the transition probabilities do not need to be normalized and that a transition matrix that contains negative terms on the diagonal can be constructed having the same eigenvector as the probability matrix.

Assume that a matrix, P, with a unity eigenvalue and eigenvector E, is the probability matrix of a finite-state Markov process. We construct formally a matrix P', such that

$$P' = kP + (1-k)I$$

$$k \neq 0$$

where the factor k converts off-diagonal terms from probabilities of occurrence to frequencies of occurrence, and the term (1-k)I revises the diagonal terms so that each column adds to unity. Straightforward manipulation yields

$$\begin{aligned}(P' - \lambda' I) &= [kP + (1-k)I - \lambda' I] \\ &= [kP + (1-k-\lambda')I] \\ &= k\left[P - \frac{(\lambda' + k - 1')}{k} I\right].\end{aligned}$$

Thus,

$$\lambda = \frac{\lambda' + k - 1}{k}$$

for all of the discrete values of λ. The eigenvalues of P' are related to the eigenvalues of P by

$$k(\lambda - 1) = \lambda' - 1,$$

showing that both matrices have an eigenvalue of 1. It also follows readily that both matrices have the same eigenvectors.

The above result shows that the relative frequencies of occurrences of the various transitions can be used to construct a matrix without first having been normalized to be transition probabilities.

3. Sequential Machine Analysis as Related to Analysis of Neuron Model Networks

To illustrate how sequential machine analysis relates to the analysis of neuron model networks, consider the case of a neuron model network connected to an "environment" consisting solely of an integrator and necessary inversion circuitry as shown in Figure 5. Each pulse from the neuron model net to the environment causes a step increase or decrease in one or the other of the outputs from the environment.

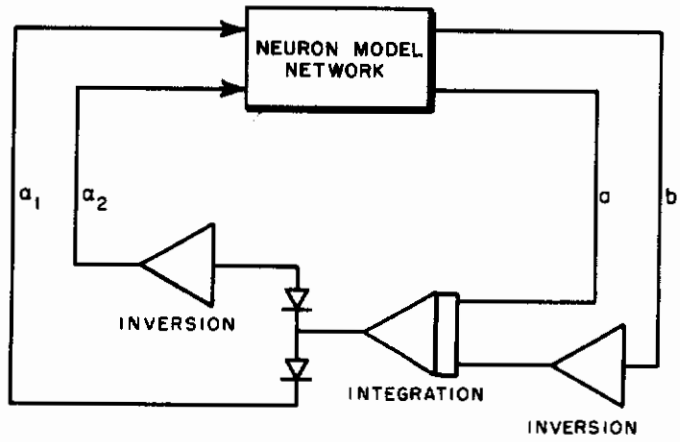


Figure 5. "Environment" with Inversion Circuitry

A sequential machine model of this environment is shown in Figure 6. The pairs of numbers within the state circles represent the amplitudes of the signals on the two output leads from the environment. Transitions a and b are the transitions the neuron model network can produce in the

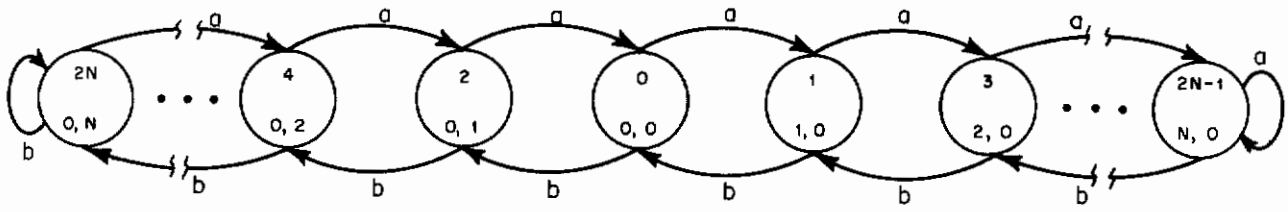


Figure 6. Sequential Machine Model of Figure 5

environment, and transitions e and e' are the transitions introduced by the switch in Figure 5. It is intuitive that such a model would in the limit go over to the integrator of Figure 6.

Initially, with zero synaptic weights, the neuron model tends to select its outputs a or b almost randomly, producing one or the other after the time lag required for neuron model thresholds to change. One can, therefore, assume appropriate probabilities for the occurrences of the transitions a, b, e, e', and use the methods above to compute the fractions of the time the environment will occupy each of the possible states, assuming that the probabilities do not change with time. The basic idea of the neuron model scheme, however, is that the various probabilities will change as a result of experience. The analysis of such a time varying process is somewhat difficult.

Let it be assumed that the matrix B varies with time. One can nevertheless write that at any instant BP = P. Differentiating formally yields

$$\frac{dB}{dt} P + B \frac{dP}{dt} = \frac{dP}{dt}$$

or

$$(I-B) \frac{dP}{dt} - \frac{dB}{dt} P = 0 \qquad (3)$$

We note that the matrix B is constrained to have at least one eigenvalue of unity. (I-B) is singular, and has no inverse. For simplicity, the sum of the components of P is always unity, so that the sum of the components of dP/dt must be zero. Let any matrix be written of the form

$$C = \begin{bmatrix} c_1 & c_1 & c_1 \cdots c_1 \\ c_2 & c_2 & c_2 \cdots c_2 \\ \cdot & & \\ \cdot & & \\ \cdot & & \\ c_n & c_n & c_n \cdots c_n \end{bmatrix} \quad c_i \text{ are real constants.}$$

Then

$$C \frac{dP}{dt} = 0.$$

It can be shown readily that for any nontrivial matrix B, a matrix of the form C can be found such that

$$\det (I - B + C) \neq 0$$

Equation 3 can then be rewritten as

$$\frac{dP}{dt} = (I - B + C)^{-1} \frac{dP}{dt} P,$$

which has a solution of the form

$$P(t) = \left[\exp \int_0^t (I-B+C)^{-1} \frac{dB}{dt} \, dt \right] P(0) \qquad (4)$$

Despite the fact that Equation 4 is in closed form, it is nevertheless quite difficult to evaluate, particularly when one assumes that dB/dt is a function of P. So far, only crude approximations have been used to estimate the time course of P(t) by hand computation. These approximations are discussed below.

It has been assumed for the analysis that each state of the sequential machine produces an output vector which contains only 1's and 0's as entries. The output vectors from each of the states can be inserted as the columns of a matrix Q. Q is nxm, where n is the number of states and m is the number of output lines from the machine. The output of the machine, Y, in any particular state, S, is given by

$$Y = QS.$$

Assume that input α occurs while the machine is in state S. The next state, S', is, from Equation 3,

$$S = \alpha \cdot AS.$$

Therefore,

$$Y' = Q\,\alpha \cdot AS.$$

The change in the input vector as a result of the transition is

$$\delta_\alpha{}^{S_i} = Y' - Y = Q\,(\alpha A - I)S.$$

The probability that this transition will indeed be seen is the probability of the transition taking place once the state has been reached, multiplied by the probability of reaching the state in the first place; i. e. ,

$$P\ \delta_\alpha{}^{S_i} = P_S \cdot P\,(\alpha | S).$$

Summing the possible changes, weighted according to their probabilities, over all possible states yields a vector

$$\phi = \sum_{j=0}^{n-1} P_{S_j}\ P(\alpha | S_j).$$

Similar results are, of course, obtained for b, c, etc. Two schemes for deciding which outputs from the machine should produce which inputs have been examined. The first was that outputs corresponding to components of ϕ that were negative should tend to produce input α, and outputs corresponding to positive components of ϕ should tend to inhibit α. When tested

numerically, this rule was found not to satisfy reasonable criteria on machine behavior. In state 1 of Figure 4, both a and b cause a reduction in one of the outputs. As a result, that particular output will tend to cause both a and b to occur equally often, despite the fact that transition a leads to the zero state and transition b to the most "undesirable" state in the machine.

In an attempt to improve performance, a second scheme was developed:

(1) Compute $\delta_\alpha{}^{s_i}$ as above

(2) Sum the components - call it $\sigma_\alpha{}^{s_i}$

(3) Form the product $\sigma_\alpha{}^{s_i}\ Y_j P_{S_j}$ and sum over j.

This computation scheme is readily seen to resemble the actual computation used in the neuromime networks. Although it gives satisfactory results in the problem case described above, it still seems to fail on more complicated machines. An example is given in Appendix IV.

How does the neuromime network generate the desired Boolean function of the sequential machine analyzed above? Methods using multilevel neuromime networks which generate logic functions are being studied. Most threshold logic assumes cascaded layers of elements. In contrast, the distributed adaptation concept seems to require a "vertical" stacking of threshold elements. Figure 7 shows the basic concept. The lowest level generates a linearly separable logic function that approximates the desired function. Its action is corrected by

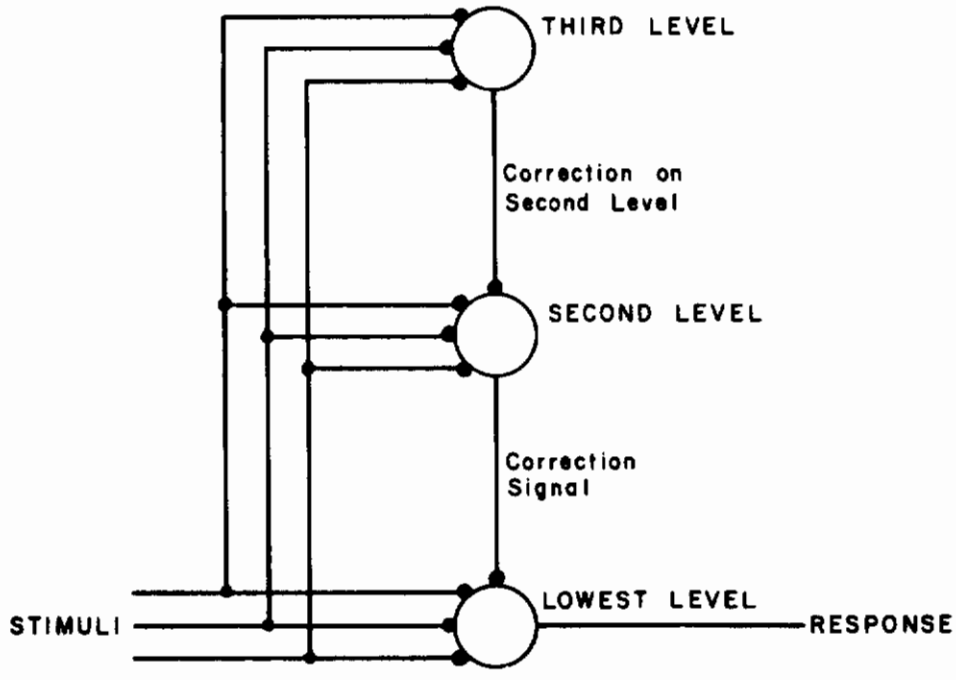


Figure 7. A Multilevel Logic Net

inputs from the next succeeding higher level. It can be shown that multilevel networks of this sort can generate any desired Boolean function.

It is believed that the sort of net organization shown in Figure 7 is more easily defensible physiologically than the usual cascaded connection. Further, the connection scheme allows each neuron model to compute the correlations between its inputs and outputs directly. Correlations would be seen by the lowest level first. The next level would see correlations only after the first level had been adjusted to an approximately correct condition, and so on up the chain. The similarity of the sequential machine to the neuromime network is easily seen.

Out of the sequential machine analysis has come insight into other possible methods of analyzing neuron model networks. The attempt to discover why certain reward-punish adaptation rules did not yield acceptable behavior on certain sequential machines indicated that the neuron model network must consider additional information. When evaluating a particular state, the neuron model network must include a measure of the particular state as an intermediate state in reaching the goal state. This indication led to a possible application of dynamic programming. To a certain extent, the dynamic programming approach requires that the network predict the results of its activities. The prediction problem here is basically one of predicting a stationary time series. Although the usual approach to solving this prediction problem is the Wiener approach, an alternate approach formulated by Kolmogorov seems more suitable for application to neuron model networks. A discussion of this approach is in Appendix III. Beyond suggesting an application of the Kolmogorov prediction theory, the dynamic programming approach has not yielded any significant analytical results. Study of the possible applications of the dynamic programming approach to the analysis of neuron model networks is continuing.

C. DEVELOPMENT AND EXTENSION OF THE OPTIMAL PROCESS APPROACH

1. General

The sequential machine analysis pointed out a problem that is a major barrier to the analysis of adaptive networks - assurance that the network will eventually arrive at an optimum condition. No adjustment method was found that would assure that after a certain "learning" period the neuron model network would follow an optimal path to the zero state. This problem suggested a study of optimal processes.

2. Application of Pontryagin's Maximum Principle

An application of Pontryagin's maximum principle seems to be more suitable to analysis of neuron model networks than other optimization techniques (References 7 and 8). Before the development of this maximum principle, recall that it was postulated in previous work that the neuron tended to minimize "total system power," defined in the following way. Let R_o be the output from a neuron, and let R_i, $i = 1, 2, \ldots$, be the respective inputs. Then the neuron attempts to produce an output as a function of the inputs in such a way as to minimize $\mathcal{P}$, defined by

$$\mathcal{P} = R_o^2 + \sum_{i=1}^{n} R_i^2.$$

This result is a "static" one, i.e., it does not consider possible time dependences of R_o and R_i. The result has now been extended to the dynamic case, with what appears to be the most significant result of the program to date.

Consider Figure 8. The "environment" is assumed to be a linear differential system, described by

$$\dot{Y} = AY + BX$$

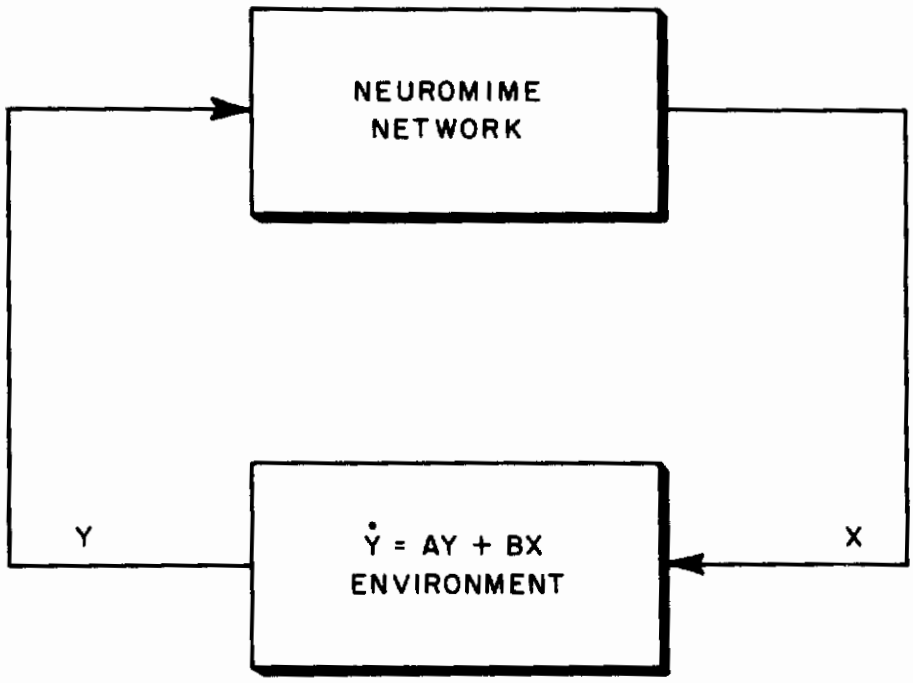


Figure 8. Generalized System

where Y is an n-vector, X is an m-vector, A is an n x n matrix of constants, and B is an n x m matrix of constants. Let the "system power" $\mathcal{P}$ be defined by

$$\mathcal{P} = \frac{1}{2}\int_0^\infty (Y^tY + X^tX)\, dt,$$

and define the optimum control $\overset{v}{X}$ to be that control which produces

$$\overset{v}{v} = \min_{X \epsilon \Omega_X} \int_0^\infty (Y^tY + X^tX)\, dt,$$

where Ω_X is the space of all possible control vectors X.

Expressed concisely, we wish to find a $\overset{v}{v}$ such that

$$\overset{v}{v} = v(\overset{v}{X})$$

$$= \min_{X \epsilon \Omega_X} \frac{1}{2}\int_0^\infty (Y^tY + X^tX)\, dt$$

subject to the constraint that

$$\dot{Y} = AY + BX.$$

Using a slightly modified form of Pontryagin's maximum principle, we can write that the Hamiltonian, H, of the above is

$$H = Y^tY + X^tX + P^t\,(AY + BX)$$

where $P = \text{col}\,(p_1, p_2, \ldots, p_n)$ is a vector function to be determined. Application of the Pontryagin maximum principle yields the results that

$$-\dot{P} = \frac{\partial H}{\partial Y}$$

$$\overset{v}{X} = X: \frac{\partial H}{\partial X} \text{ is minimum}$$

and we obtain for the specific problem at hand the following three simultaneous equations

$$\dot{P} = -A^tP - Y \tag{5}$$

$$X = -B^tP \tag{6}$$

$$\dot{Y} = AY + BX \tag{7}$$

with boundary conditions

$$Y(o) = Y_o$$

$$\lim_{T \to \infty} P(T) = \text{col}(o, o, \ldots, o)$$

$$= \phi.$$

The systems that result from the instrumentation of these equations show striking similarities to several well-known characteristics of neurons.

3. Similarity to Neuron Model Networks

As a first example of the similarities, one can draw an analog computer diagram (Figure 9) that instruments Equations 5, 6, and 7.

It is readily seen that the system would be unstable under most conditions. One can write directly that

$$\ddot{Y} = BB^t\,(sI - A)^{-1}\,(sI + A)^{-1}\,Y.$$

The appearance of the two factors $(sI - A)$ and $(sI + A)$ shows that for every pole in the left half of the s-plane there is a corresponding - and diverging - pole in the right half. From another point of view, $\ddot{Y} = \ddot{Y}\,(s^2)$, which indicates that for every function $Y_s(t)$ satisfying the equations there is another function, $Y_s(-t)$ which also satisfies them. The form of the desired solutions, which converge to $Y = P = \phi$, can thus be seen by assuming $Y(o) = P(o) = \phi$ and introducing small perturbations at $t = o^+$. Inspection of Figure 9 shows that for $P(o) = \phi$, $Y(o^+) = \delta Y$, one obtains

$$\text{sgn}\,[\dot{P}(o)] = -\text{sgn}\,[\delta Y].$$

Also, for $P(o+) = \delta P$, $Y(o) = \phi$, one obtains that

$$\text{sgn}\,[\dot{Y}(o)] = -\text{sgn}\,B^tB\,[\delta P].$$

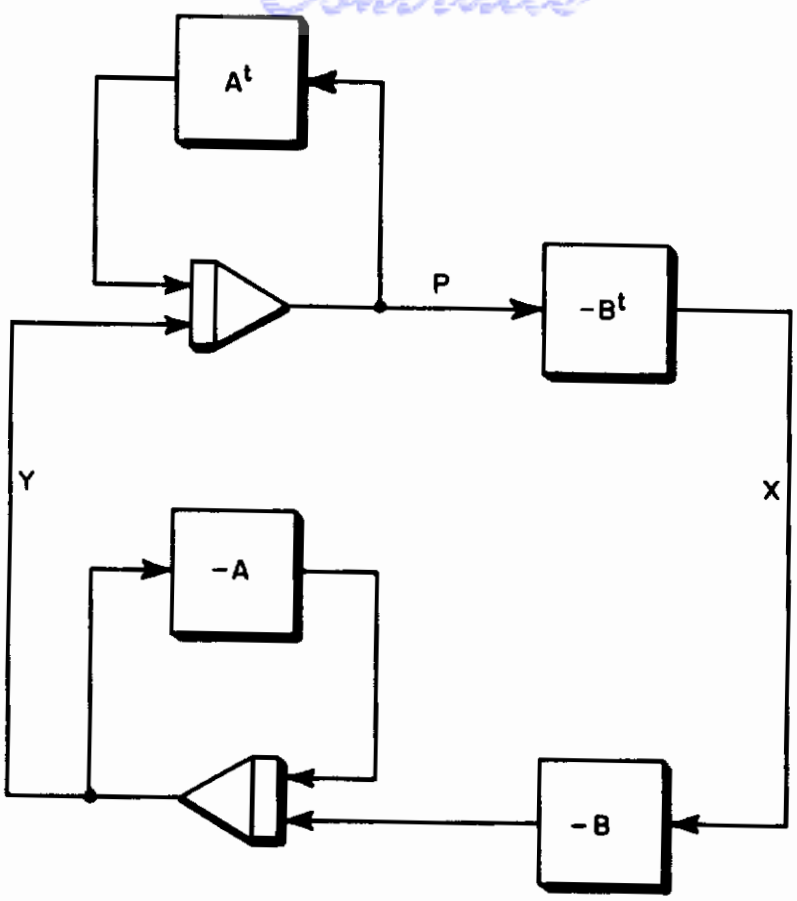


Figure 9. Analog Instrumentation of Equations 5, 6, and 7

Since B^tB is positive and semidefinite, it follows that the diverging conditions are those in which the signs of the non-zero Y_i and P_j are opposites. The system can be made stable by requiring that all the P_j's and Y_i's have the same sign.

The similarity to neurons is obvious, since inputs to and outputs from a neural net are pulse rates that are either positive or zero.

A more powerful result can be obtained. It is shown below that the P_j's, and thus the X_k's, are linear combinations of the Y_i's under certain circumstances. This result shows that a linear combination of the inputs can produce outputs that optimize system behavior. The output rate of a neuron is, to a good approximation, the weighted sum of its inputs. The analogy between the two systems is immediately clear. Consider the system described by

$$\dot{Y} = AY + BX,$$

where B is non-singular and X and Y are of order n. Let $FY_2 = Y$. F can be selected so that $F^{-1}AF = A_2$ is diagonal. We obtain

$$\dot{Y}_2 = A_2Y_2 + F^{-1}BX$$

and setting $F^{-1}B = B_2$ yields

$$\dot{Y}_2 = A_2Y_2 + B_2X.$$

Let $B_2{}^tX_2 = X$. Substitution yields

$$\dot{Y}_2 = A_2Y_2 + B_2B_2{}^tX_2.$$

Let $GX_1 = X_2$ and $GY_1 = Y_2$. G is selected so that $G^{-1}B_2B_2{}^tG = D$ is diagonal. Substitution yields

$$\dot{Y}_1 = G^{-1}AGY_1 + DX_1.$$

Let $D^{-1}X_0 = X_1$, yielding

$$\dot{Y}_1 = G^{-1}AGY_1 + X_0.$$

We can now let $G^{-1}Y = Y_1$, $G^{-1}X = X_0$, yielding

$$\dot{Y} = AY + X$$

where A is diagonal. Applying Pontryagin's maximum principle yields

$$\dot{Y} = AY + X \tag{8}$$

$$\dot{P} = -AP - Y \tag{9}$$

$$X = -P \tag{10}$$

Substituting Equation 10 into Equation 8 yields

$$\dot{Y} = AY - P \tag{11}$$

$$\dot{P} = -AP - Y \tag{12}$$

Assume $P = KY$ where K is diagonal. Substitution in Equations 11 and 12 yields

$$Y = AY - KY \tag{13}$$

$$KY = -AKY - Y \tag{14}$$

We premultiply Equation 13 by K, recalling that $KA = AK$ if both A and K are diagonal, and subtract, obtaining

$$2KA - K^2 + I = 0. \tag{15}$$

Since all matrices of Equation 15 are diagonal, component-by-component solution is possible.

$$k_{ii} = \frac{2a_{ii} \pm 2\sqrt{a_{ii}{}^2 + 1}}{2}$$

$$= a_{ii} \pm \sqrt{a_{ii}{}^2 + 1}$$

It was established earlier that the P_j had to have the same signs as Y_i for stability. One solution for k_{ii} can be discarded as unstable, yielding

$$k_{ii} = a_{ii} + \sqrt{a_{ii}{}^2 + 1}$$

Although the above analysis is valid only for $|B| = 0$, preliminary investigation implies that it can be extended to the case where $|B| = 0$ by partitioning the system into two or more subsystems which take on a multilevel configuration. This result makes the analogy between neuron nets and the theoretical systems even more striking.

4. Steepest-Descent Method

The two conditions obtained from the above analysis were:

(1) Inputs and outputs are non-negative (more strictly, of the same sign).

(2) Outputs are linear combinations of inputs.

These conditions, when applied to the optimal control equations, pointed to the use of the method of steepest descent for the synaptic weight adjustments.

The optimal control equations that describe the neuron model also form a set of error equations. The neuron model uses the set of error equations to adjust its synaptic weights. For example, take the system equation

$$\dot{Y} = AY + BM.$$

The corresponding equation describing the neuron model is,

$$\dot{Y} = \alpha Y + \beta M \tag{16}$$

Equation 16 can be put in the form of an error equation,

$$\dot{Y} - \alpha Y - \beta M = \epsilon \tag{17}$$

Since the neuron model does not start out in its optimal state, Equation 17 is a more realistic physical statement of the neuron model's state.

Taking the partial derivative of Equation 17 with respect to α yields

$$\frac{\partial \epsilon}{\partial \alpha} = -Y$$

We can use this partial derivative to adjust the α synaptic weight according to the following relation:

$$\dot{\alpha} = -\frac{\partial \epsilon}{\partial \alpha} \operatorname{sgn} \epsilon = Y \operatorname{sgn} \epsilon$$

where

$$\begin{aligned} \operatorname{sgn} \epsilon &= 1 \text{ iff } \epsilon > 0 \\ \operatorname{sgn} \epsilon &= -1 \text{ iff } \epsilon < 0 \\ \operatorname{sgn} \epsilon &= 0 \text{ iff } \epsilon = 0 \end{aligned}$$

A similar adjustment relation is available for β and the other neuron model parameters. The above relation is a form of the method of steepest descent, which is discussed in detail in Reference 9.

The steepest descent method described above can be placed on a geometric basis. The error equation, Equation 17, describes a set of error curves. The starting point for a synaptic weight is somewhere on this set of error curves, but not at the optimum value. The steepest descent equations describe the direction that is perpendicular to the specific error

curve and headed for the optimum value. Each synaptic weight adjustment is headed for the optimum value in a direction perpendicular to the error curves as each is crossed. The synaptic weight, then, converges through error space to the optimum value. This method was used exclusively for the tests on networks using the optimal control concept.

5. Instrumentation

The next objective was to find some relatively easily instrumented relation between the variables that can be used for adjusting synaptic weights. Based on the results of the preceding analysis, two criteria were instrumented. The first instrumented criterion used the idea that any p_i was a linear combination of the y_i, other intermediate variables, p_j, and outputs x_k. Formally, let us postulate that there exists a matrix, U, such that

$$UP = Y + A^tP + BX \tag{18}$$

After substituting the postulate in the system equations, routine manipulation yields that

$$UP = (A - A^t + BB^t + U)\,Y + BX + (-A^tA^t + BB^tA + UA^t + U)P \tag{19}$$

If we are justified (we may not be!) in setting the right-hand member of Equation 19 equal term by term to the right-hand member of Equation 18, we obtain ultimately that

$$U = AA^t \tag{20}$$

and

$$(I - A)\,Y = P \tag{21}$$

with the further result, after some manipulation, that

$$A^tP + Y = BB^tP + AA^tP \tag{22}$$

(note that $A^tP + Y = -\dot{P}$)

The result is analogous to the criterion instrumented in the neuron model used for the preliminary tests on networks using the optimal control concept (refer to Appendix I).

If, however, one tests to discover if Equation 19 is really equal term by term to Equation 18, one can determine that a necessary condition is that the matrix A be symmetric, a condition that one does not obtain in the most general case. One can, however, force the matrix A to be symmetric, or even diagonal, by a suitable transformation on the variables Y.

Substituting formally

$$KY' = Y, \quad |K| \neq 0$$

and

$$(K^{-1})\,P' = P$$

in the original system equations, the result is

$$\dot{Y}' = CY' - DD\,P'$$

$$\dot{P}' = -CP' - EY'$$

where C, D, and E are symmetric.

Physically, this result corresponds to saying that if the inputs of the neuron model network are mixed in the proper proportions for each neuron model, the neuron model can use Equation 22 to adjust its synaptic weights. To effect the synaptic weight adjustment, the neuron model uses Equation 22 to test whether its inputs are related correctly to the postsynaptic potentials of the neuron models distal to it.

The second criterion was based directly on the conclusions that the variables, P, are linear combinations of the state variables, Y. Assuming P = KY and substituting this assumption into the original system equations, a new set of equations is obtained

$$\dot{Y} = AY - BB^tKY \tag{23}$$

$$K\dot{Y} = -A^tKY - Y \tag{24}$$

Premultiplying Equation 23 by K and subtracting Equation 24 from it yields

$$(I + A^tK + KA - KBB^tK)\,Y = \phi \tag{25}$$

In general, since Y is not such that Equation 25 would be satisfied,

$$I + A^tK + KA + KAA^tK = KAA^tK + KBB^tK \tag{26}$$

It is easily shown that the matrix K can be symmetric. One assumes that $K = K' + K''$, in which K' is symmetric and K'' is skew.

Substitution in Equation 26 yields that

$$I + A^tK' + A^tK'' + K'A + K''A - K'BB^tK' - K''BB^tK' - K'BB^tK'' - K''BB^tK'' = \phi$$

We form a partial sum S_1 as follows

$$S_1 = I + A^tK' + K'A - K'BB^tK'$$

We note that this is our original equation where K is replaced with K'. We now form a second partial sum of the remaining terms.

$$S_2 = A^tK'' + K''A^t - K''BB^tK' - K'BB^tK'' - K''BB^tK''$$

since K'' is skew, $(K'')^t = -K''$, and we obtain that $A^tK'' = (-K''A)^t$.

Considering the other terms also

$$S_2 = K''A - (K''A)^t + K''BB^tK' - (K''BB^tK')^t - (K'')^tBB^tK''$$

The difference in the first two terms of S_2 is skew. The difference in the next two terms is also skew. But the last term is symmetric. Two skew matrices add, either to form a skew matrix or a null matrix. Since $S_1 + S_2 = [0]$, the sum of the first four terms must vanish, which puts certain restrictions on A and B that would, in general, not be satisfied. In most cases, therefore, K'' must vanish.

Having shown that K can be symmetric, Equation 26 can be factored into

$$(I + KA)\,(I + KA)^t = K(AA^t + BB^t)\,K^t$$

We note that $AA^t + BB^t$ is positive semidefinite. A theorem of matrix algebra states that any positive semidefinite matrix D can be written as CC^t, where C is a real matrix. We obtain

$$(I + KA)\,(I + KA)^t = KCC^tK^t$$

or $(I + KA) = KCQ$

where Q is orthogonal, i.e., $Q^t = Q^{-1}$

Further manipulation yields

$$K(CQ - A) = I$$

or

$$K = (CQ - A)^{-1}, \text{ assuming } |CQ - A| \neq 0$$

Since K is symmetric, CQ - A must be symmetric, from the theorem that the inverse of a symmetric matrix is also symmetric.

From the above we obtain that P = KY if there exists an orthogonal matrix Q_1 and a matrix C such that CQ - A is symmetric and non-singular, and $CC^t = AA^t + BB^t$. Reference 10 gives a proof that a matrix K of constants always exists such that the intermediate variables P can be expressed in terms of the process outputs Y by the function P = KY. The proof follows from assuming that K = K(t), i.e., a time-varying matrix, and substituting in the original equations.

One obtains a Riccati equation in the matrix K. Reference 10 discusses the equation and its solutions, with the principal results for our case that

(1) K is symmetric and positive definite.

(2) Diagonal entries are all greater than zero.

In cases where it is desired to drive Y to zero in some finite time, rather than at infinity, the matrix K becomes time varying. Reference 10 gives a procedure for finding its time history.

The results discussed above complete the necessary proof of existence of an optimal solution. It remains to show that the solutions obtained are stable. This criterion was instrumented for the majority of tests on networks using the optimal control concept. So far, the simulations have been stable, generally, but there is at present no analytical reason to believe this will always be the case.

6. Stability of the System

Using the second instrumented criterion, Y and P are linearly related by Y = (CQ - A)P; from symmetry, also $Y = (Q^tC - A^t)P$. One questions whether there exists a matrix Q which will yield a stable system. To test, one can insert the relation in the original system equation, $\dot{Y} = AY - BB^tP$, to obtain

$$(CQ - A)\dot{P} = A(Q^{-1}C^t - A^t)P - BB^tP$$

or

$$(CQ - A)\dot{P} = (AQ^{-1}C^t - CC^t)P$$

$$= -(CQ - A)Q^{-1}C^tP$$

and finally

$$\dot{P} = Q^{-1}C^tP$$

Stability of the system requires that all the eigenvalues of $Q^{-1}C^t$ have negative real parts. It has not yet been proved that Q can be appropriately selected in general, but several numerical tests of specific systems have always yielded a matrix Q with the necessary properties.

In sum, for the presently instrumented criterion to be effective for any system requires proof that there exists a matrix Q such that

(1) The eigenvalues of $Q^{-1}C^t$ all have negative real parts

(2) $(CQ - A)$ is symmetric

where $C = AA^t + BB^t$. This question is, of course, being pursued intensively.

7. Convergence of Synaptic Weight Adjustment Rule

Since the majority of neuron models tested used the second criterion to adjust their synaptic weights, the conditions under which the second criterion converged were of interest. From the system equations

$$\dot{Y} - AY - BM = \phi$$

$$\dot{Y} - \alpha Y - \beta M = \epsilon_2$$

where α and β are the neuron model's parameters corresponding to the A and B environment parameters. And, making use of the assumption $P = KY$, another set of equations is

$$K\dot{Y} + A^tKY + Y = 0$$

$$\mathcal{K}\dot{Y} + \alpha^t\mathcal{K}Y + Y = \epsilon_1$$

where $\mathcal{K}$ is the neuron model parameter whose set of values must be adjusted to correspond to the K set of values.

One can write that

$$(\alpha - A)Y + (\beta - B)M = \epsilon_2 \tag{27}$$

and

$$(\mathcal{K} - K)\dot{Y} + (\alpha^t - A^tK)Y = \epsilon_1 \tag{28}$$

We consider first ϵ_2. It is clear that if Equation 27 is to hold for all Y and all M simultaneously, we obtain $\epsilon_2 = \phi$ iff $(\alpha - A)$ and $(\beta - B)$ are both zero. However, M is related to Y by the equation

$$M = -\beta^t \mathcal{K} Y$$

thus the non-zero roots of

$$(\alpha - A) - (\beta - B)\,\beta^t\mathcal{K} = \phi \tag{29}$$

are spurious stable points of Equation 27. For any given $\mathcal{K}$, there is a trajectory of such solutions, the trajectory passing through the points $\alpha = A$, $\beta = B$ and $\alpha = A$, $\beta = \phi$. Further, Equation 28 can also be satisfied simultaneously, so that all error signals are zero and the parameters are non-optimum.

Fortunately, factors that result from the particular instrumentation being used intervene. The derivatives of Equation 28 are approximated by lead networks. As a result, the computed values of the $\mathcal{K}_{ij}$ vary slightly during a disturbance. The resulting deviations of $\mathcal{K}$ from its central value cause Equation 29 to fail to be satisfied, and the values of the α_{ij} and β_{ij} are revised. A similar argument can be advanced for Equation 28. Although non-optimum values

of a and $\mathcal{K}$ exist that satisfy the equation, perturbations in $\mathcal{K}$ lead to perturbations in a. Experiments outlined in detail in Appendix I support these results.

The above analysis can be extended to the interaction of neuron models in a redundant network. We are to determine how the synaptic weight adjustments of the various neuron models interact to cause convergence of the network's parameters to optimum values. The analysis, which is similar to that performed above, is presented in detail in Appendix V. The general result is that the synaptic weights of the various neuron models converge simultaneously to optimum values in much the same way as those of a single neuron model. The basic difference is that perturbations in the $\mathcal{K}$ parameter are due to the interaction of other neuron models, as well as to the imperfect method of taking a derivative.

SECTION IV

FUTURE PLANS

Using the neuron model based on the optimal control concept, a network will be instrumented to control a two-axis aircraft simulation. Two immediate goals have been set for the experimental work. The first goal is to determine the network's susceptibility to damage. The second is to determine its ability to adapt after damage. It is anticipated that enough pertinent information will come from this and other simulations to supply hardware specifications.

The analytical work will continue the extension of the present theory to nonlinear systems.

APPENDIX I

NETWORK EXPERIMENTS

A. GENERAL

The tests described in this appendix represent the form and the results of the experimental work on the contract to date. Following is an outline of the experimental work:

(1) Tests using the transistor neuromime hardware

  (a) Networks with contralateral connections

  (b) Networks with internal feedback

  (c) Network with paralleled neuromime output

  (d) Approximation to an aircraft pitch channel

(2) Tests using the new optimal control concept

  (a) Preliminary test circuits

  (b) Tests on more complex systems

  (c) Tests on the convergence of the neuron model's parameters to optimum values

  (d) Tests on redundant networks

Although most of the tests were designed to parallel the analytical work, some tests were performed to aid the instrumentation development. The former tests are emphasized, because their results hold more significance for the contract work. Diagrams of the test circuits, accompanied by brief explanations of the tests, results, and evaluations, are presented in this appendix.

B. TESTS USING THE TRANSISTOR NEUROMIME HARDWARE

1. Networks with Contralateral Connections

Two environment transfer functions were tested with the net - a gain and an integrator (see Figure 10). The purpose of the test was to determine the net's capability to reduce an error signal to zero. The weight computation rates observed were judged to be slower than expected. The time taken for a weight computation generally exceeded five minutes, compared to an expected value of one or possibly two minutes. However, the weights computed for the various synapses usually had the correct algebraic signs. In several cases the wrong sign was computed for a synaptic weight. The reason for the wrong synaptic weight signs was discovered to be a faulty computing method. The synaptic weight was computed continuously. The synaptic weight integrator integrated the input changes occurring during the firing cycle and subtracted the integral of the input changes occurring after firing stopped. The computed synaptic weight was correct only at the end of a firing cycle. If the synaptic weight computation was started or stopped at some time within the cycle, the computed synaptic weight had a wrong sign.

A second order environment was used to test the network (see Figure 11). The net was tested to determine its ability to reduce the error signal to zero and to note its response to a

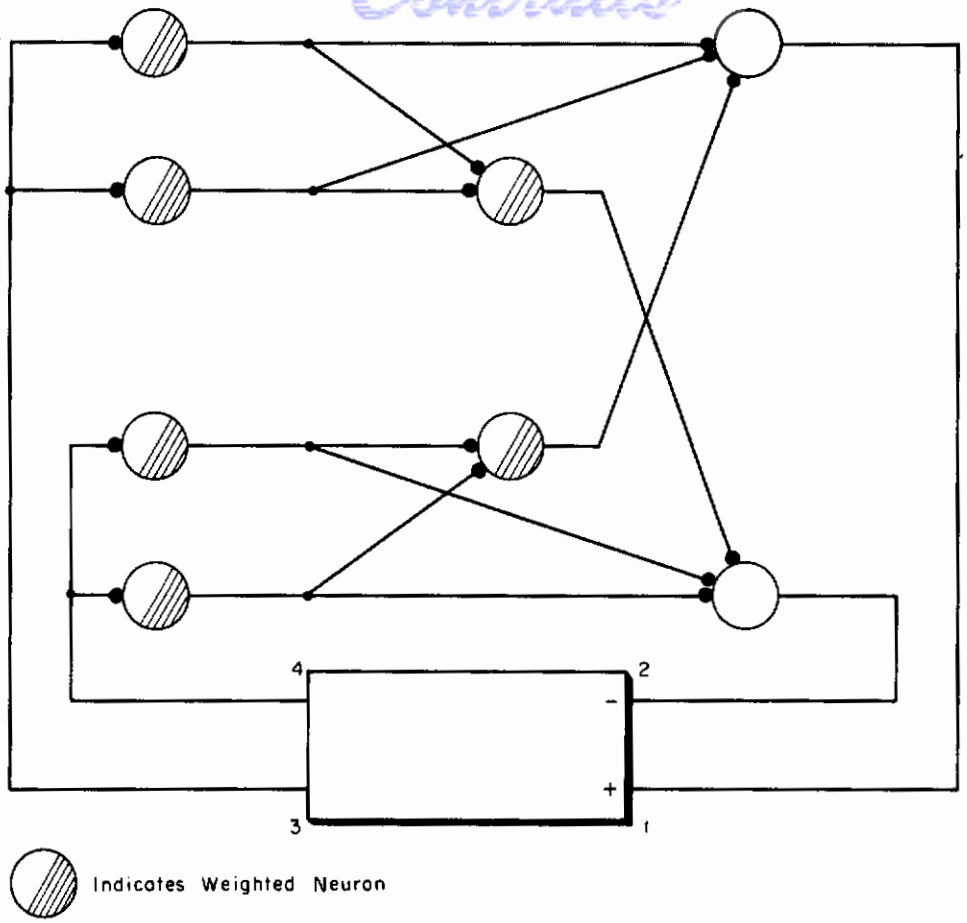


Figure 10. Network with Two Simple Environment Transfer Functions

second order environment. The environment used had the following transfer function:

$$G(S) = \frac{\omega^2}{S^2 + 2\omega\xi S + \omega^2}$$

where

$G(S)$ = environment transfer function

ω^2 = environment gain

ξ = damping variable

S = Laplace transform variable

The value for ω was varied from 0.3 to 10 and the value for ξ varied from 0.03 to 0.8. The results of the test are plotted as a stability curve in Figure 12. The network would not compute synaptic weights with ω's over 5, which limited the test results somewhat. The network made small synaptic weight computations when $1/\omega$ was below the network response time of one second. Some difficulty was experienced when testing the system for the stability points plotted in Figure 12.

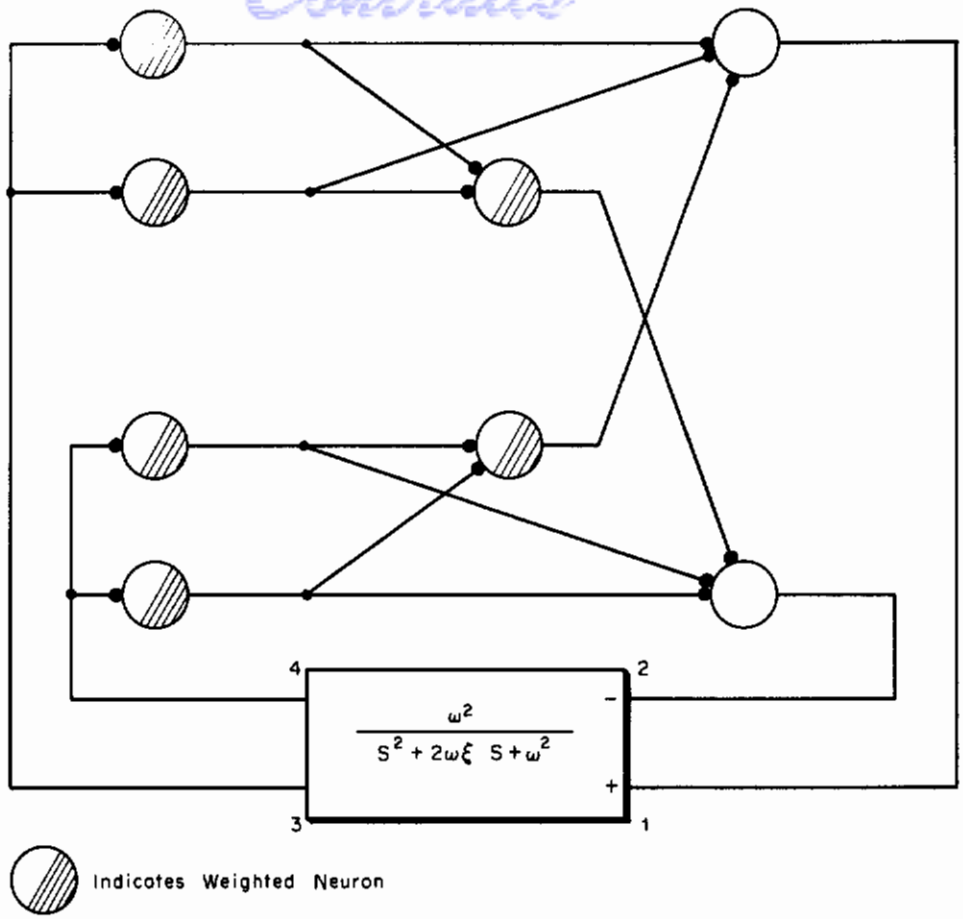


Figure 11. Test Network with Second Order Environment

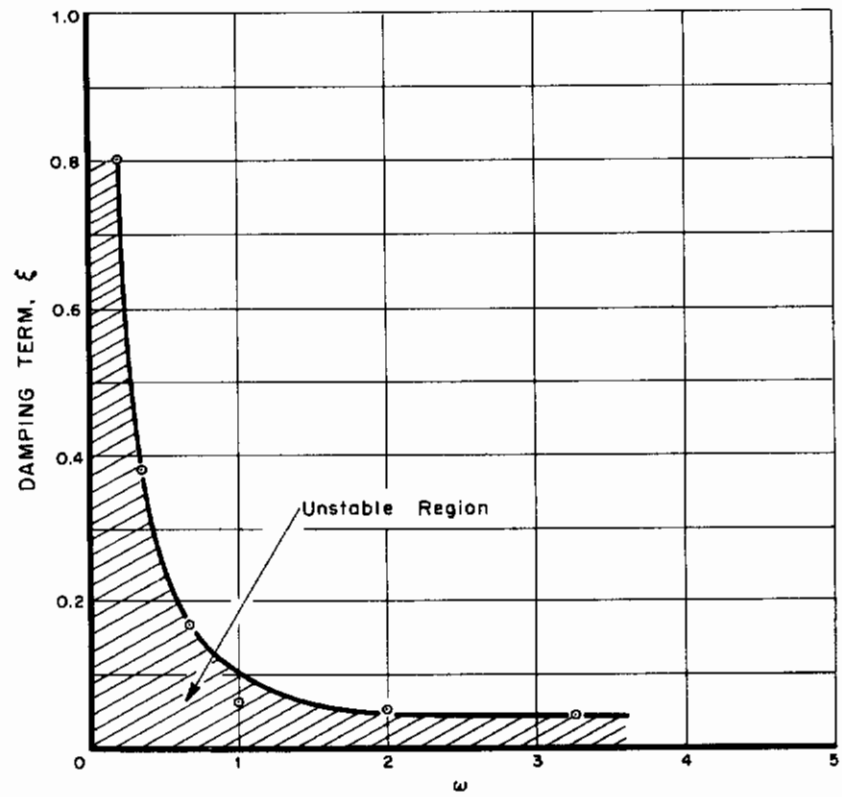


Figure 12. Stability Curve of Second Order Environment Test

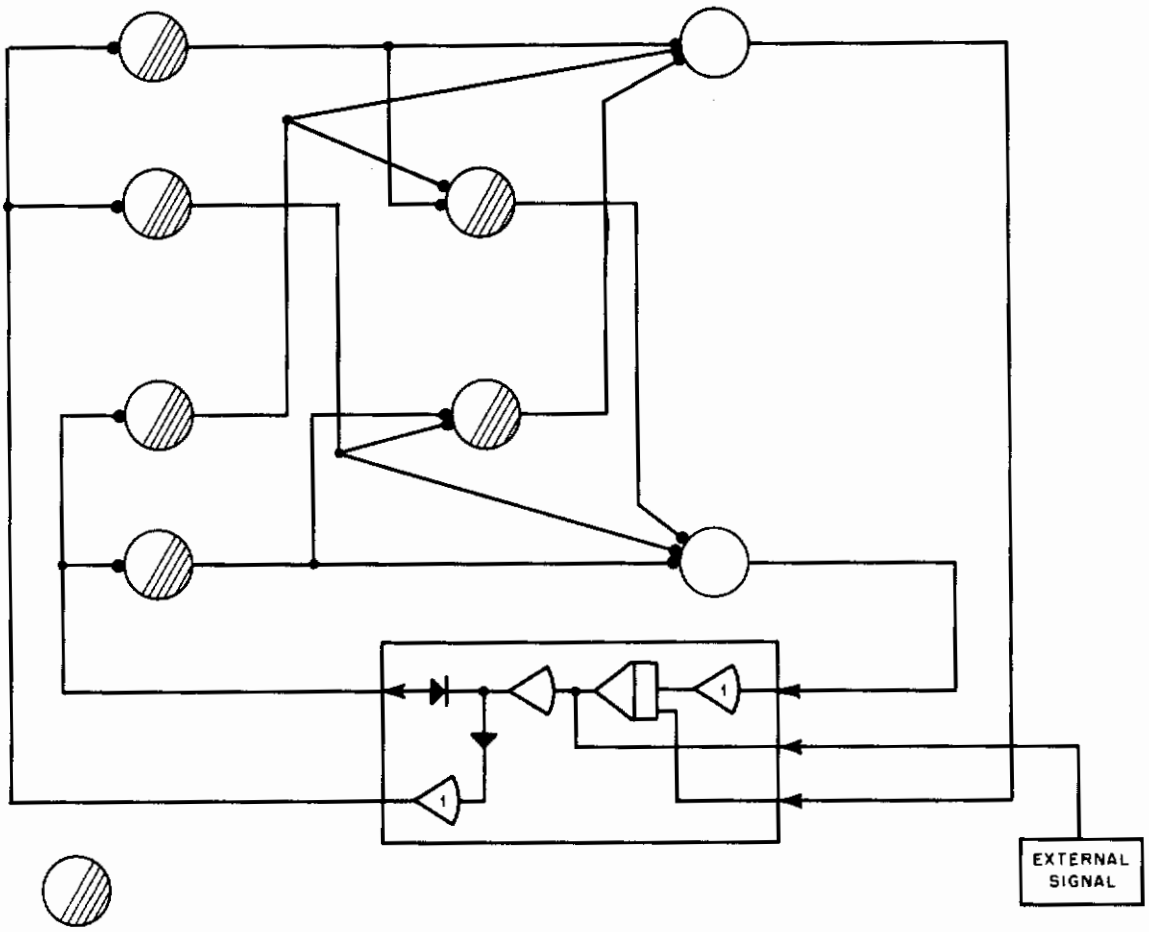


Figure 13. Test Network Connected to a Gain with a Control Reversal

For the next test, the network was connected to a gain, or "environment" transfer function (see Figure 13) and tested to determine its ability to stabilize the system after an "environment" input control reversal. The computation of weights following the input control reversal was sluggish. Preliminary tests showed that several changes should be made on the neuromime model. These changes were made and the new network was tested as outlined later.

2. Networks with Internal Feedback

Environments of a gain and an integrator were used to test four networks (see Figures 14, 15, 16, and 17). These four networks were connected to study the effects of internal feedback. The network was tested with and without internal feedback connections. Further, the network's control connections to the environment were severed to test the ability of internal feedback to assist the network to revise the synaptic weight computations when the network's output no longer affected the environment.

In the network shown in Figure 14, neuromime G2 has an environment change input and a feedback change input. The feedback change input has an opposite effect on neuromime G2 from the environment change input; i. e., neuromime G2 computes a negative synaptic weight with only feedback changes, and a positive synaptic weight with only environment changes. The ratio of the magnitudes of the environment changes to the feedback changes was defined to be A. Three values of A were used: $A<1$; $A = 1$; and $A>1$.

The network without internal feedback computed positive synaptic weights and stabilized the system, i. e., reduced the injected error signal to zero. Then, when the network outputs

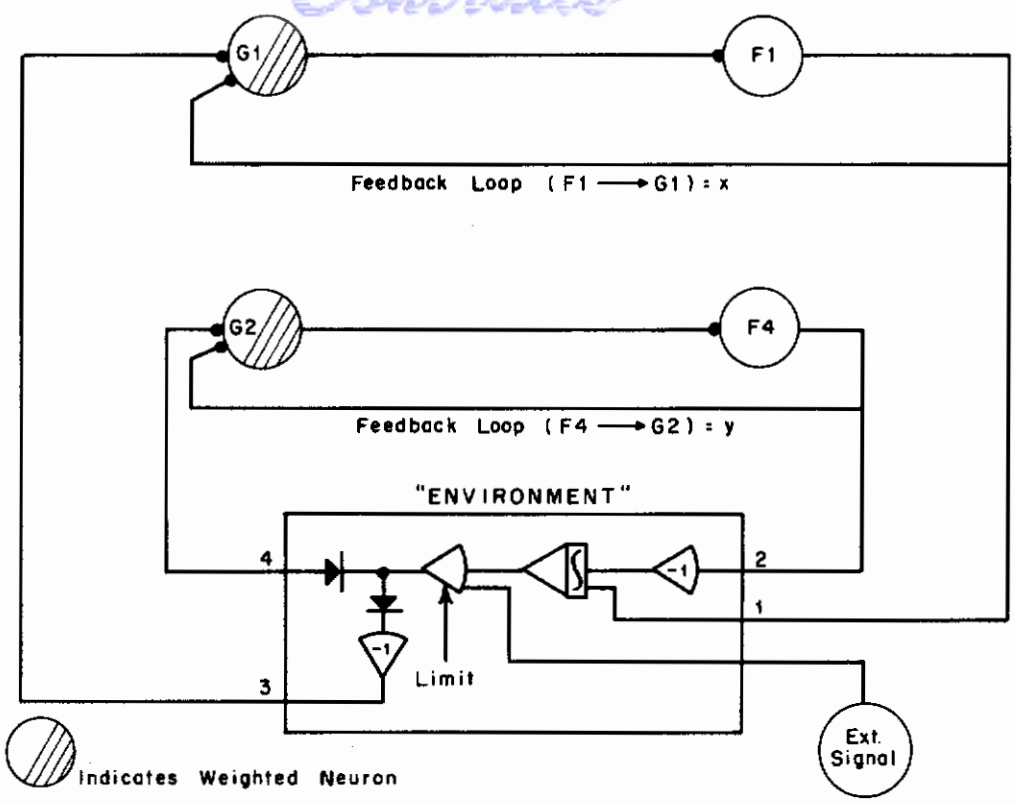


Figure 14. Test Network with Environment Input and Direct Feedback Input

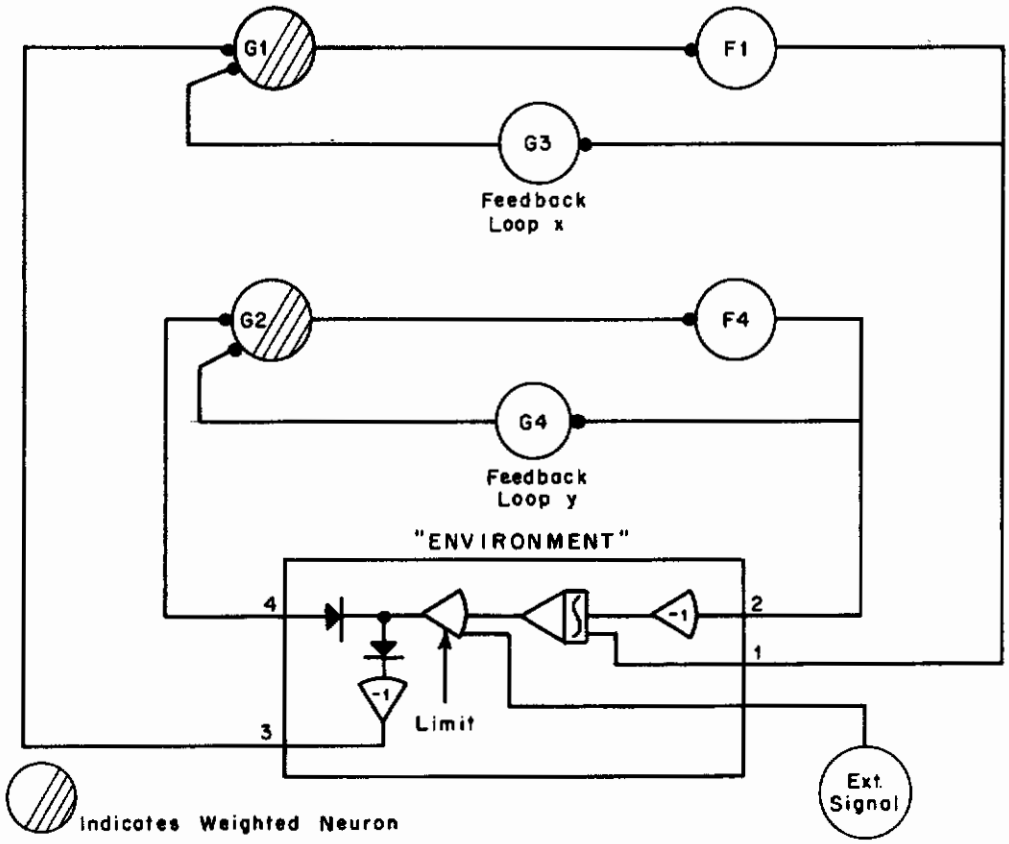


Figure 15. Test Network with Environment Input and Indirect Feedback Input

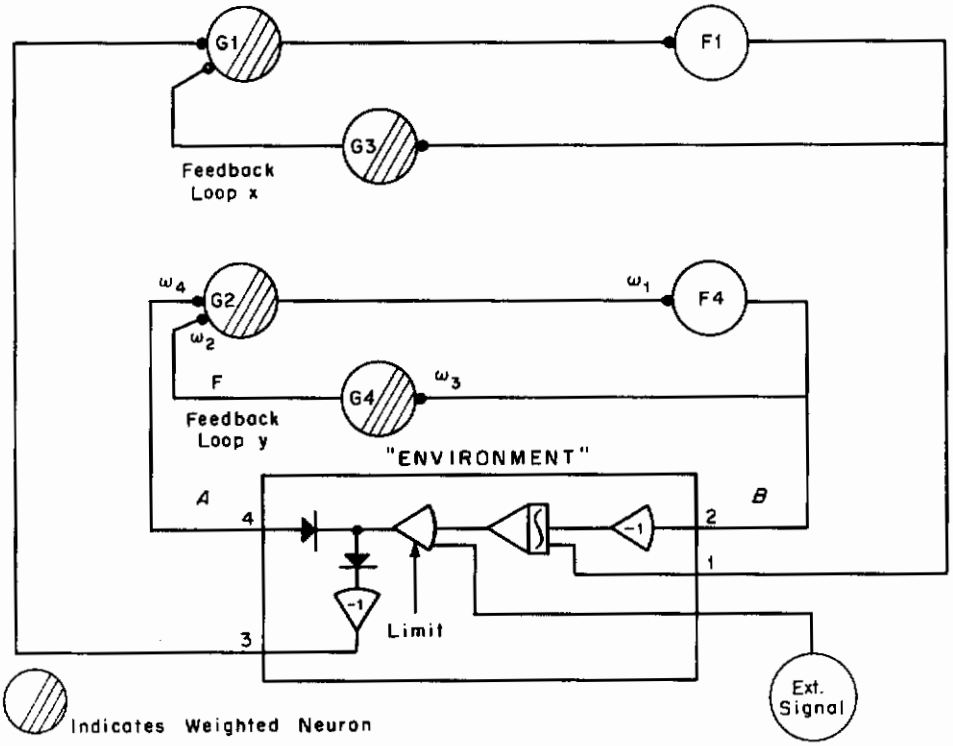


Figure 16. Test Network with Feedback Neuromime Having Computed Synaptic Weight

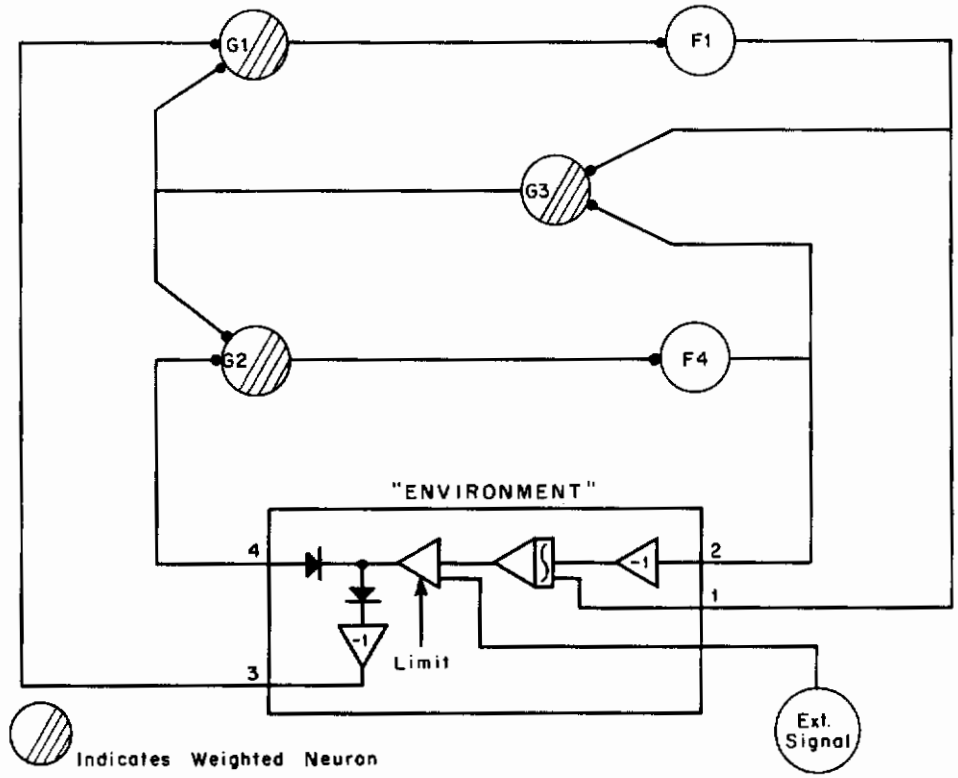


Figure 17. Test Network with Common Feedback Loop

were disconnected from the environment input, the synaptic weight values stayed constant. The network could not and did not revise its computation. The network with internal feedback, when connected to the environment, also computed positive synaptic weights and stabilized the system. When the network outputs were disconnected from the environment input, the synaptic weight computations slowly computed negative values. If neuromime G2's weighted rate and unweighted rate output were equal but opposite in sign, G2's synaptic weight computation stopped. Neuromime F4's input is the sum of the weighted and unweighted rate outputs of neuromime G2. If the weighted and unweighted rate outputs are equal but opposite in sign, their sum is zero. Neuromime F4 can never fire with a zero input. If F4 can never fire, the feedback loop to G2 is broken and all feedback changes seen by G2 stop.

Since neuromime G2 needs input changes to compute a synaptic weight, the broken feedback loop stops G2's input changes and synaptic weight computation.

With $A < 1$, neuromime G2 computed a negative synaptic weight. With $A = 1$, neuromime G2 computed a zero synaptic weight. With $A > 1$, neuromime G2 computed a positive synaptic weight.

In the network shown in Figure 15, neuromime G4 replaced the direct feedback connection from the output of F4 to the input of G2 (as shown in Figure 14). The extra delay time of neuromime G4 was the only difference between that network and the previous one. The test results were the same as the previous one.

In the next network (Figure 16), the feedback neuromime G4 had a computed synaptic weight, in contrast to the previous network. Neuromime G4's synaptic weight became negative in all tests. The other test results were much the same as those reported for the previous network.

The networks shown in Figures 14, 15, and 16 had two separate feedback loops. In the next network, shown in Figure 17, neuromime G3 replaced the two feedback loops. The test results were the same as those reported for the previous networks. Although some interplay between the two separate network channels due to the common feedback loop was expected, none was seen.

In the next network to be tested, a gain and an integrator were used as environments (see Figure 18). The environment control connections could be reversed. The two aims for

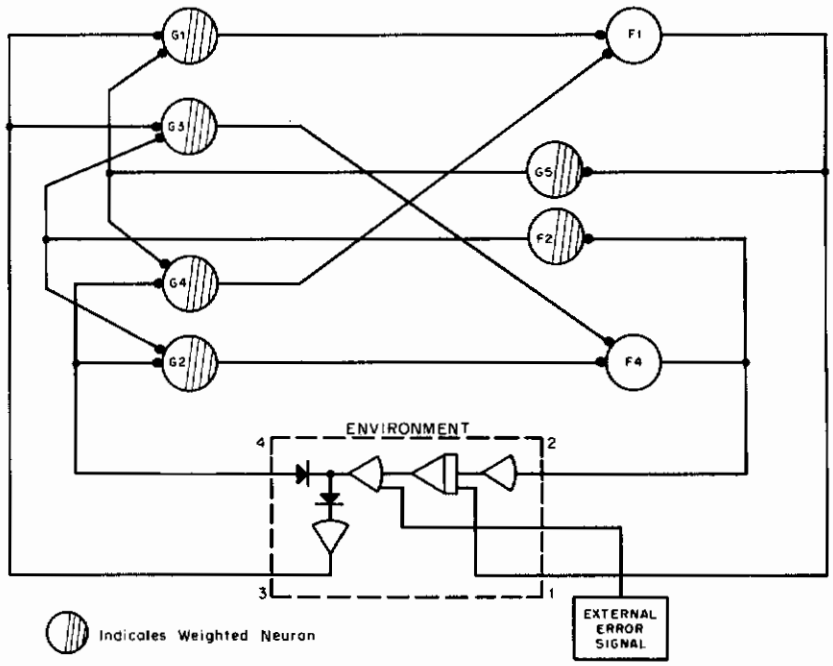


Figure 18. Test Network with Two Environment Transfer Functions with Control Reversal

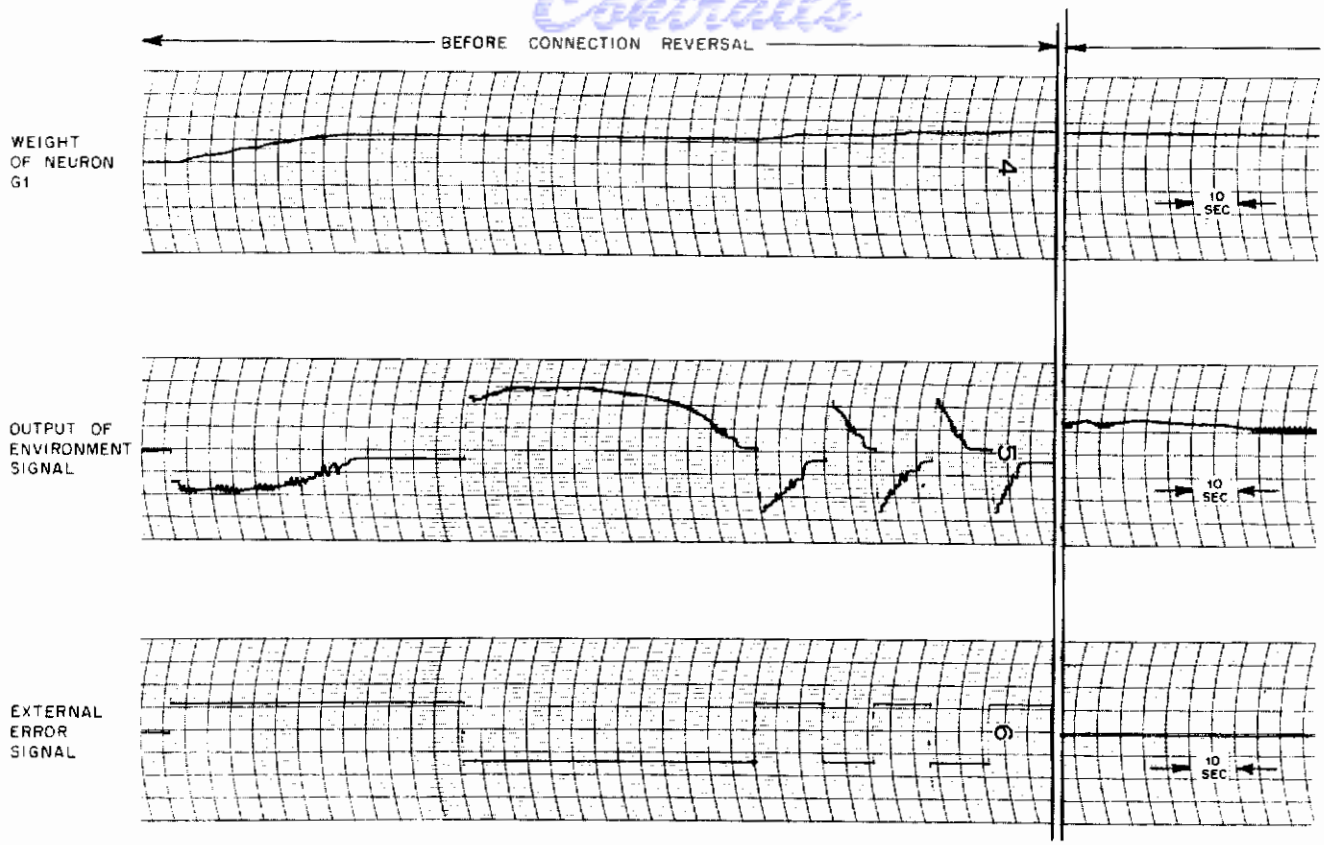


Figure 19. Analog Computer Test Run of Network without Internal Feedback (Sheet 1 of 2)

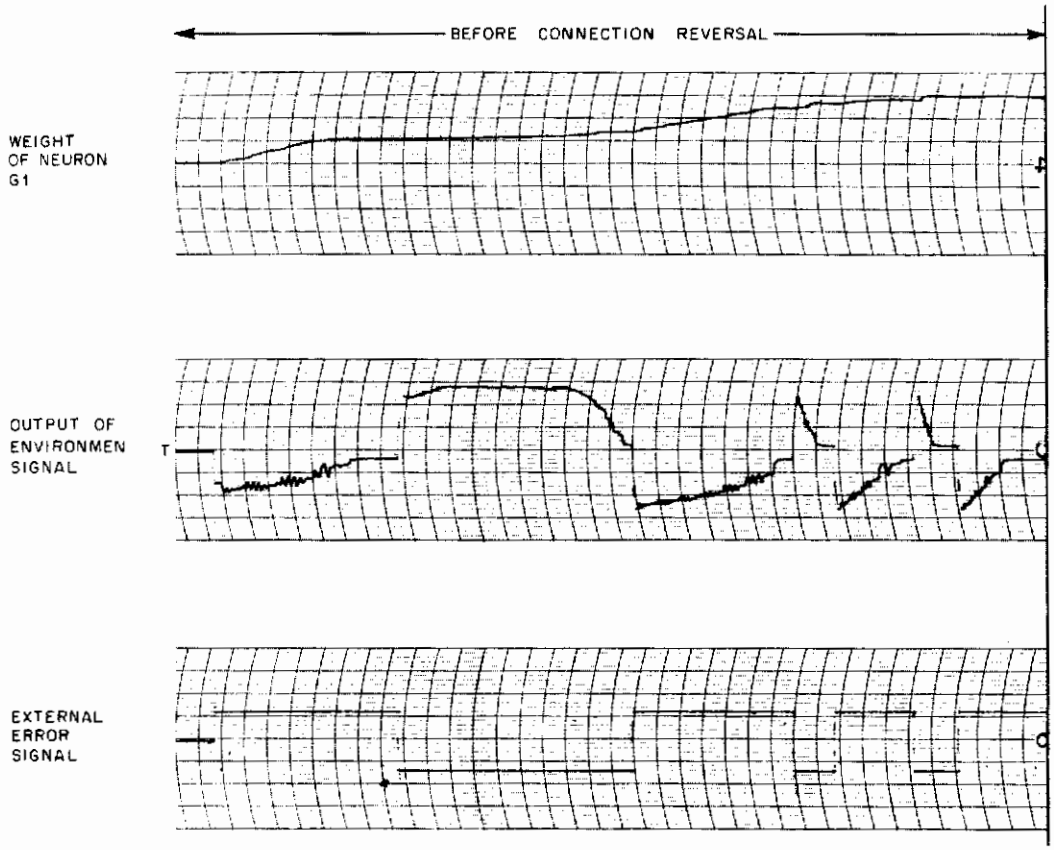


Figure 20. Analog Computer Test Run of Network with Internal Feedback (Sheet 1 of 2)

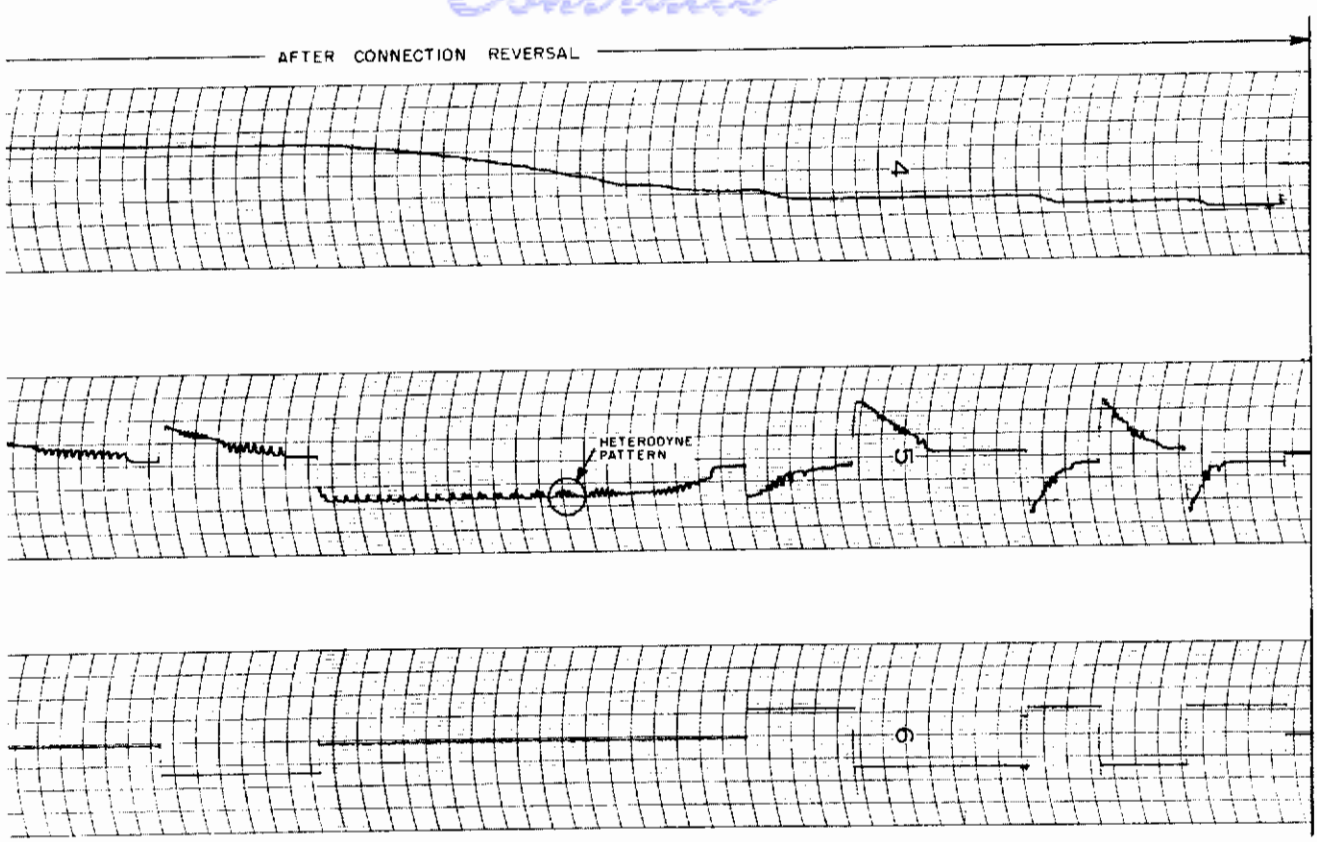


Figure 19. Analog Computer Test Run of Network without Internal Feedback (Sheet 2 of 2)

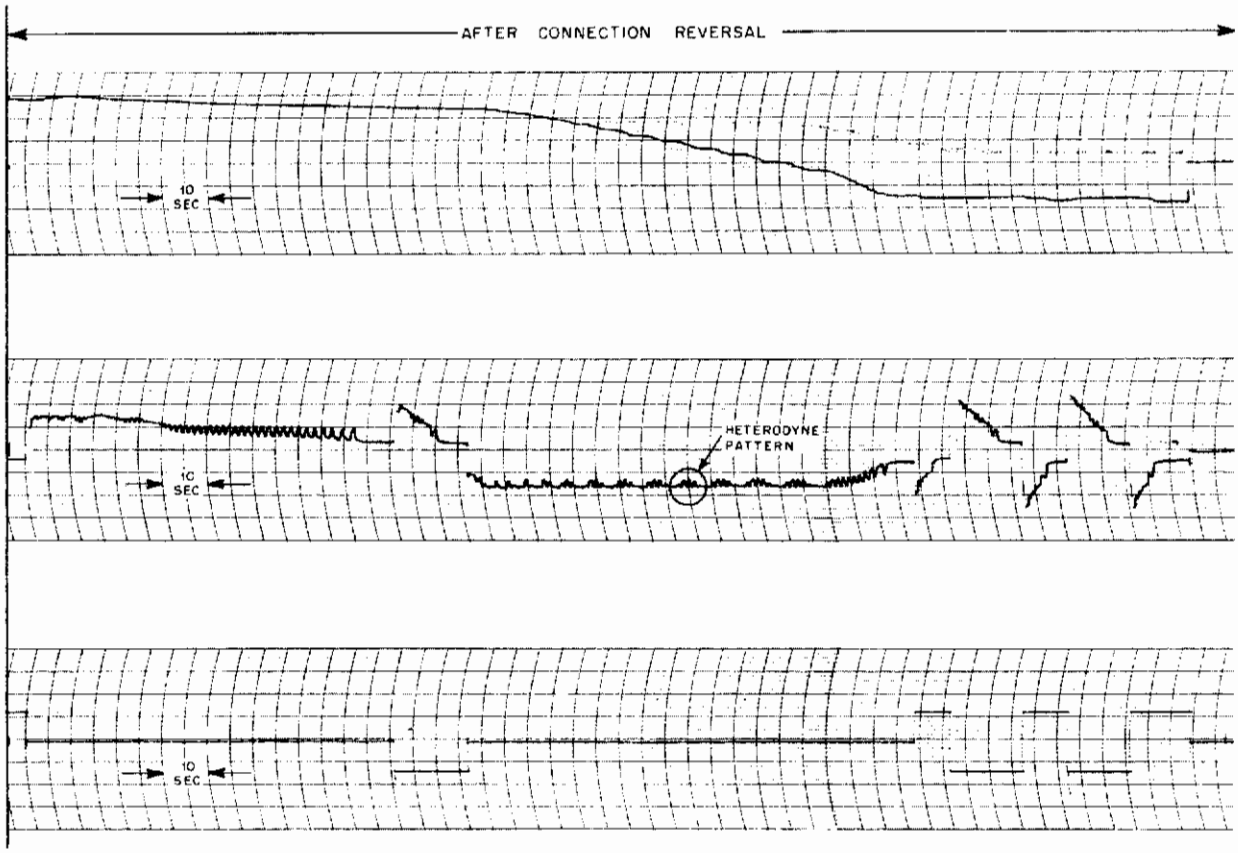


Figure 20. Analog Computer Test Run of Network with Internal Feedback (Sheet 2 of 2)

the series of tests were to determine the ability of a network to adjust to an "environment" input control reversal and to determine the difference in network action made by internal feedback. The test results shown on Figures 19 and 20 are typical analog computer test runs. Since this network used internal feedback, the testing procedure was similar to that of the previous four networks using internal feedback. The ability of the network to adjust to an environment input control reversal is shown on these analog computer test runs. The network did adjust to the control reversal. A careful study of the two test runs, one for the network with feedback and the other for the network without feedback, showed no significant difference. Notice that the neuromime revised its synaptic weight computation to adjust to the new situation.

The analog computer runs show a heterodyne pattern in the environment output error signal caused by the network input to the environment. The interference of one network output with the other produces this heterodyne pattern.

3. Networks with Paralleled Neuromime Output

A gain and an integrator were the environments used in the next network tests (see Figure 21). The purpose of the tests was to study the effects that many neuromimes firing at once had on the environment. Two to four neuromimes were connected to the same stimuli

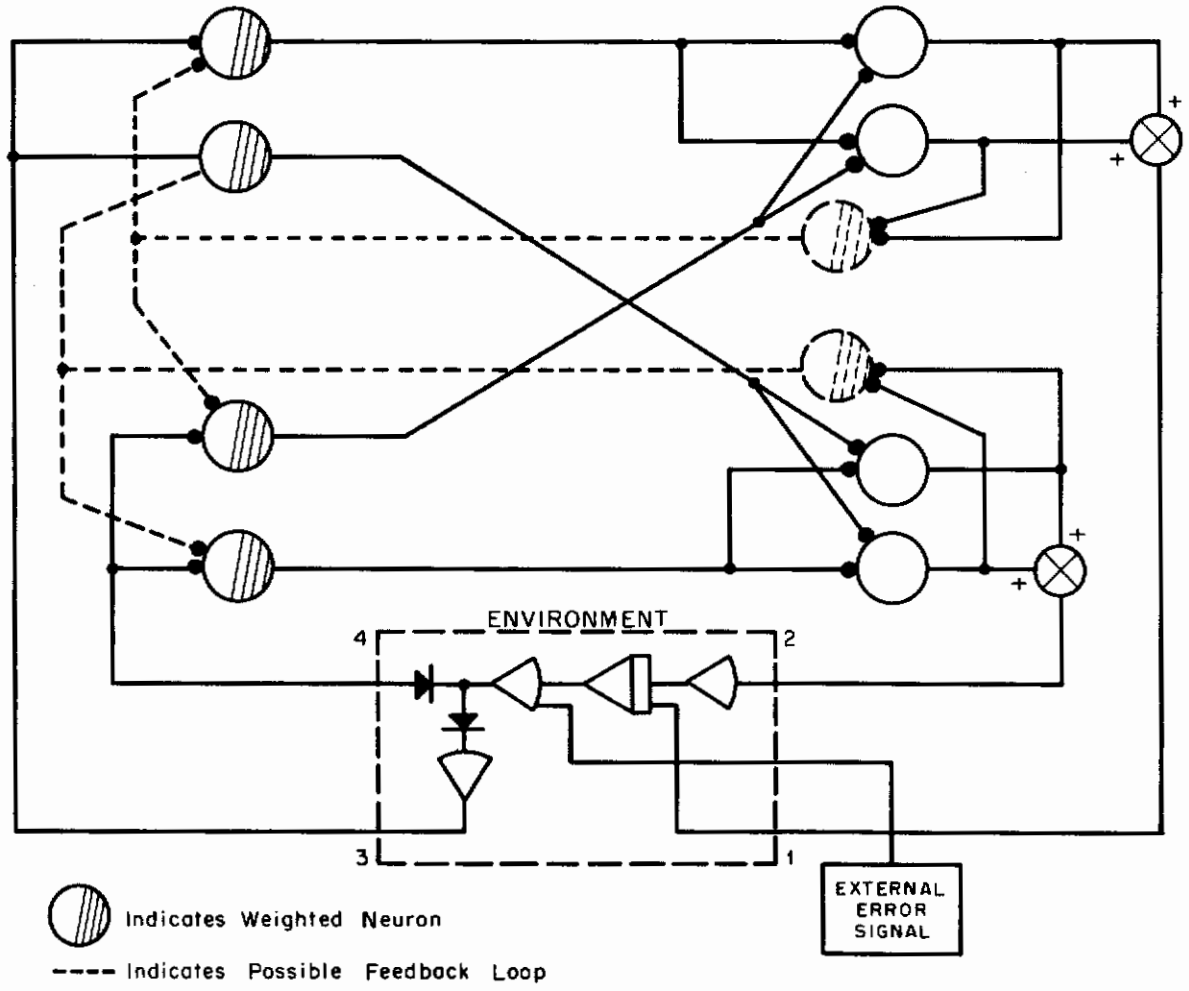


Figure 21. Test Network with an Output of Two Paralleled Neuromimes

and their outputs added together. This sum was the network output and produced a heterodyne pattern like that observed in the previous network.

The network reduced an error signal to zero, but took a longer time to do this task than a similar network with one output neuromime. The synaptic weights were computed correctly, but slowly. When three and four neuromimes were paralleled, the network took longer to perform the control task than the network with two neuromimes paralleled. Of course, the synaptic weight computations for the networks with three and four neuromimes paralleled were also slower than those of the network with only two neuromimes paralleled. The tests showed that the approximation to many neuromimes firing by paralleling two to four neuromimes was too crude.

An attempt was made to test the network with a second-order environment. The environment was designed to allow a control reversal. The purpose of the test was to study the network interconnections necessary to control a second-order environment after a control reversal. However, preliminary tests on the system showed a lack of knowledge of the network interconnections that would be required to control a second-order environment without a control reversal. The network was revised to study the control of a second- and third-order environment.

4. Approximation to an Aircraft Pitch Channel

A third-order environment was used to simulate, rather crudely, an aircraft pitch channel (see Figure 22). The parameters used in the simulation were not designed for any

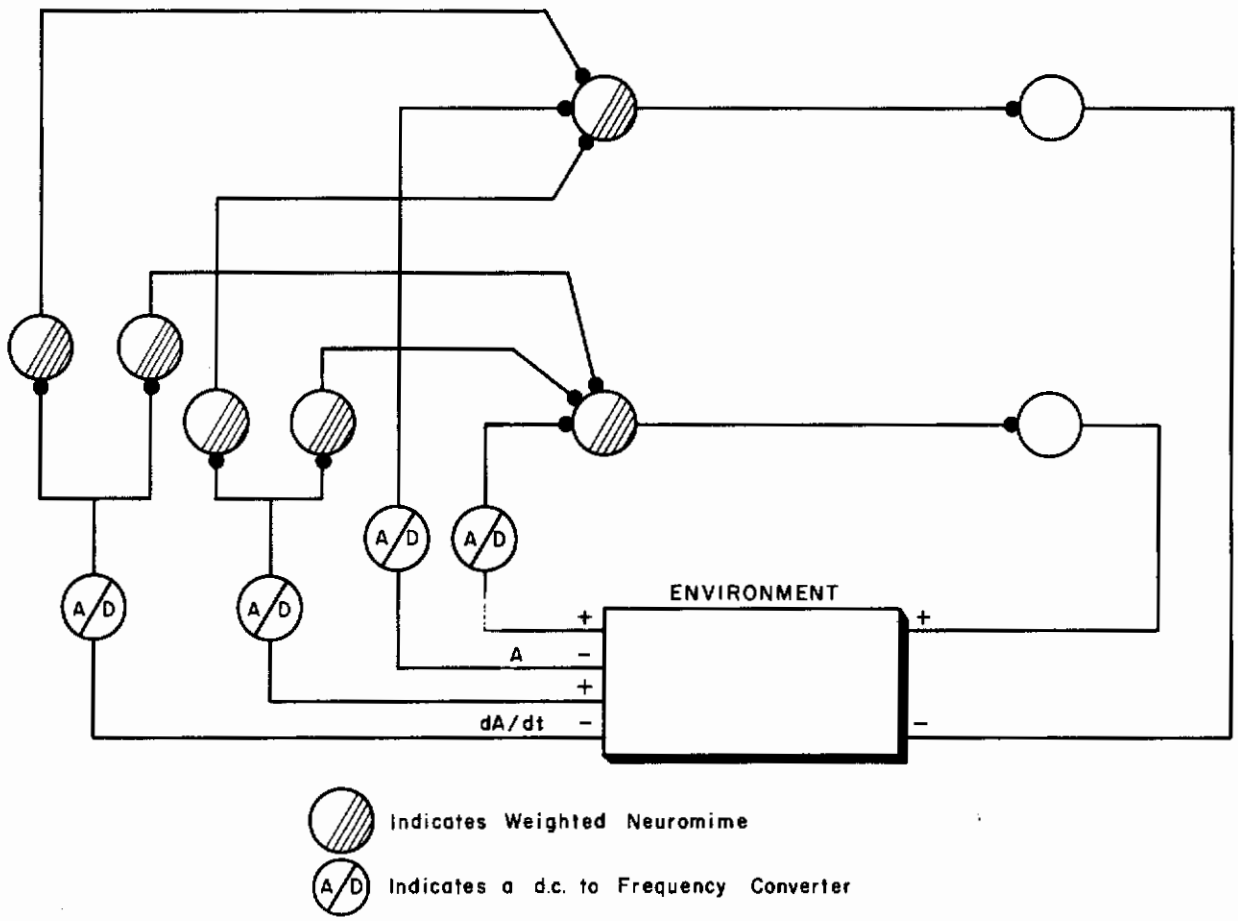


Figure 22. Test Network with Rate Feedback

specific aircraft. The basic study was that of controlling a third-order system. The premise for the tests was that a neuromime network that controls a third-order system could control a third-order pitch channel with specific aircraft pitch parameters. The network was tested both with and without pitch-rate feedback. The results of the pitch-rate feedback tests were evaluated to aid the instrumentation development.

The analog computer recordings (Figures 23 and 24) show that the network reduced pitch error to zero regardless of the presence of rate feedback. Further, all synaptic weights had the proper algebraic sign. However, the synaptic weight computations did not perform as expected. Some synaptic weights were small and had little effect on the control. The overall system response was improved when pitch-rate information was supplied to the neuromime network.

The network without ptich-rate feedback did not reduce the transient "wiggling" response of the pitch-error signal to a step input. Further, this network did not reduce the overshoot which occurs as the pitch-error signal swings through zero. The network with pitch-rate feedback improved the system response in both these conditions.

C. TESTS USING THE NEW OPTIMAL CONTROL CONCEPT

Networks were designed to study the application of optimal control theory to neuron model networks. The networks tested for this phase of the neuron model development used no external hardware. All simulations were done on the analog computer. The equations instrumented for the tests are discussed more fully in the text of this report. In this appendix, the networks tested are described and significant results are summarized to illustrate the development of the neuron model through the application of optimal control theory. In general, the equations used to describe the tests in this section are as follows:

$$\dot{Y} = AY + BM + D \tag{30}$$

$$M = -B^tP \tag{31}$$

$$-\dot{P} = A^tP + Y \tag{32}$$

where Y is the environment output matrix, M is the neuromime network output matrix, D is the environment disturbance matrix, and A and B are environment coefficient matrices. The matrix P entered into the equations from Pontryagin's maximum principle.

The optimal criterion was to minimize a system "cost" function, which closely resembles a minimum energy criterion. The equation is

$$C = \frac{1}{2} \int_0^\infty (Y^tY + M^tM)\,dt.$$

1. Preliminary Test Circuits

The preliminary tests presented here were designed with two immediate goals in mind: first, to develop instrumentation methods for use with the optimal control theory; and second, to illustrate certain parts of the theory.

Figure 25 shows the first circuit using the new optimal control concept. To control the system optimally, initial conditions on the integrators within the neuromime network were set at optimum values. These initial conditions for the parameters of the test network were computed by hand to yield an optimum path. These parameter values were then set on the analog computer and tests were run on the network. The network converged to zero following the computed optimum path. If one of the network parameters was varied away from the computed optimum value, the system diverged.

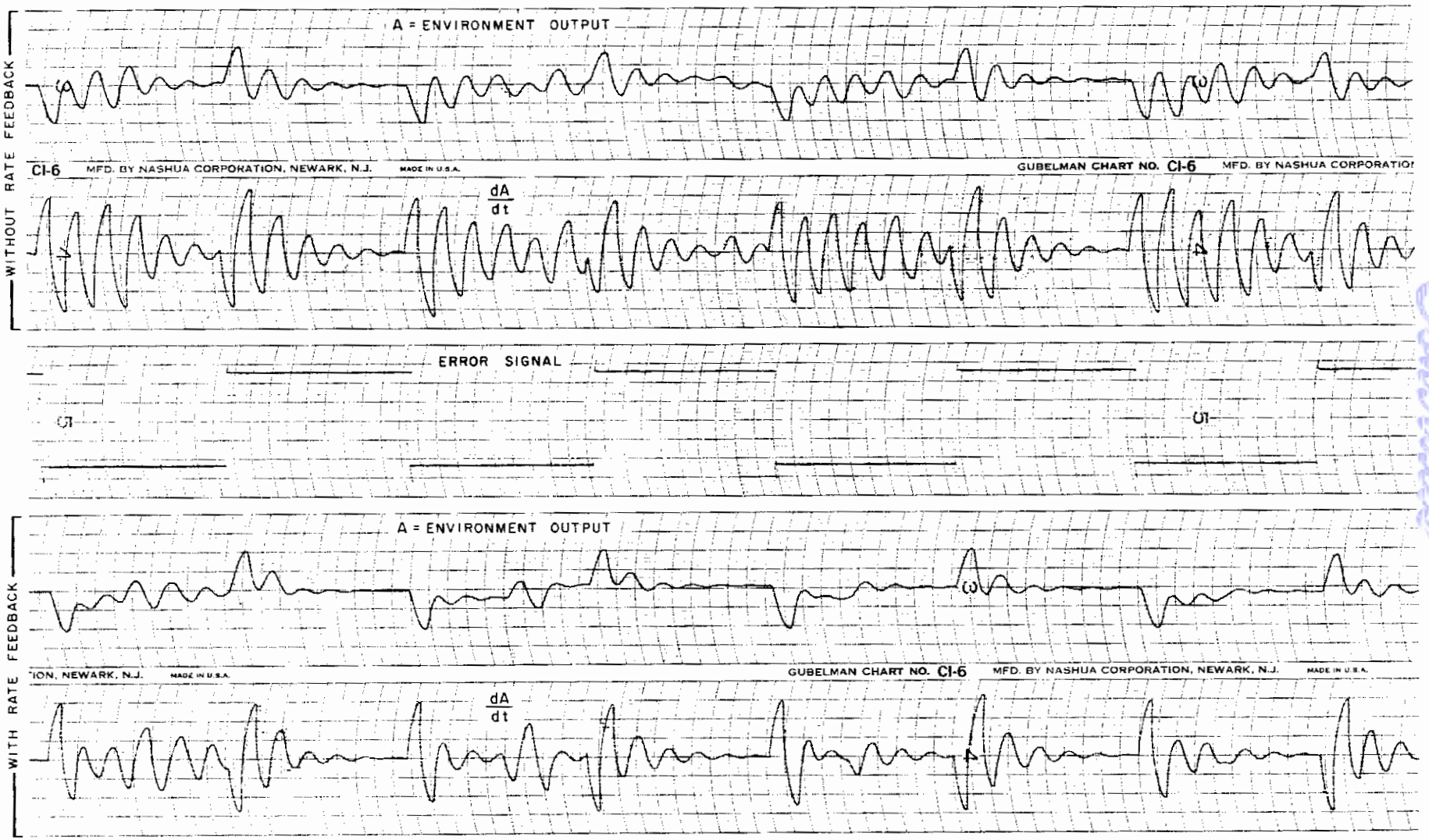


Figure 23. Analog Computer Test Run of Network Using Rate Feedback

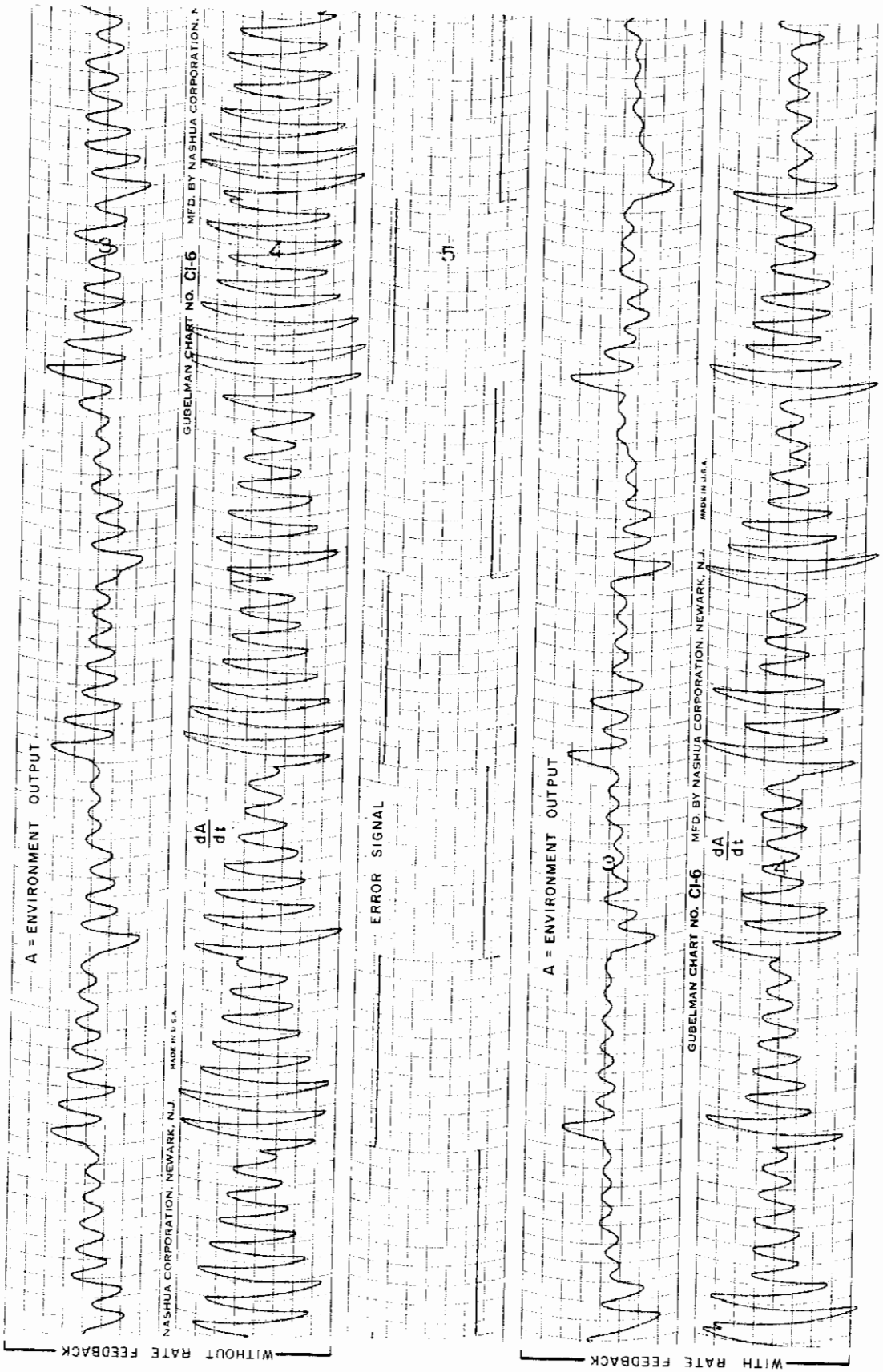


Figure 24. Analog Computer Run of Network Using Rate Feedback - Rerun at Smaller Damping Constant

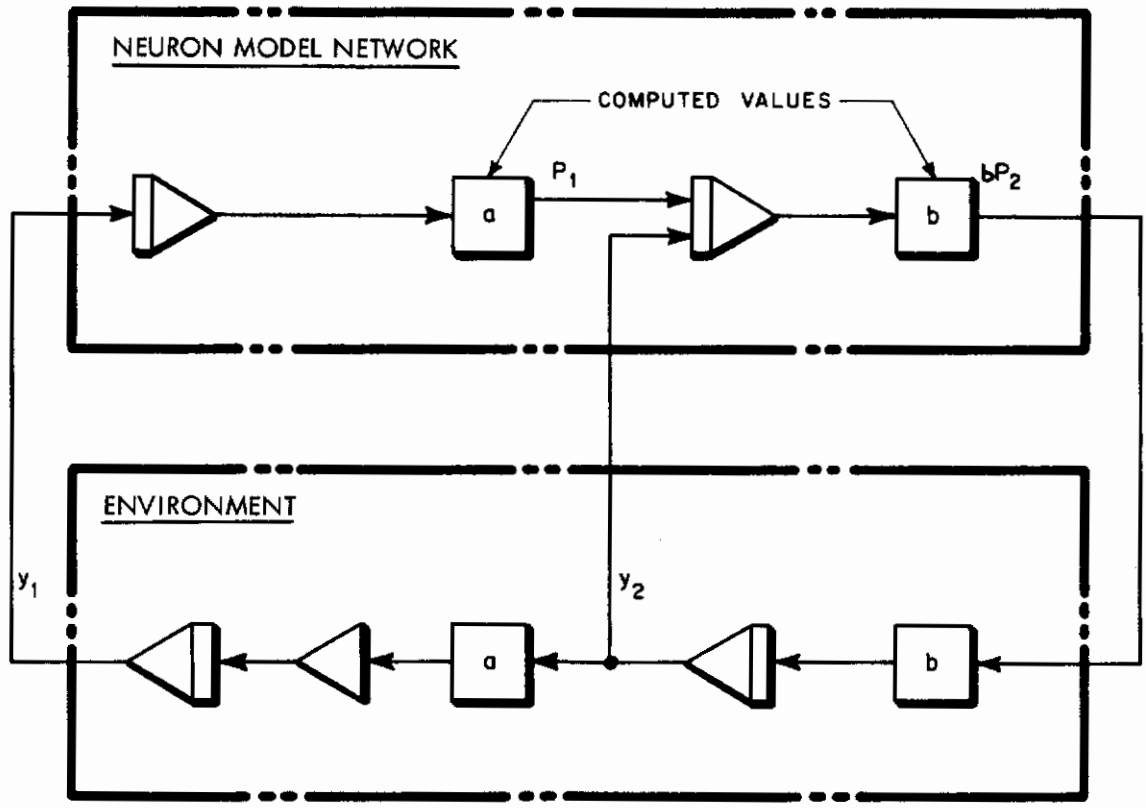
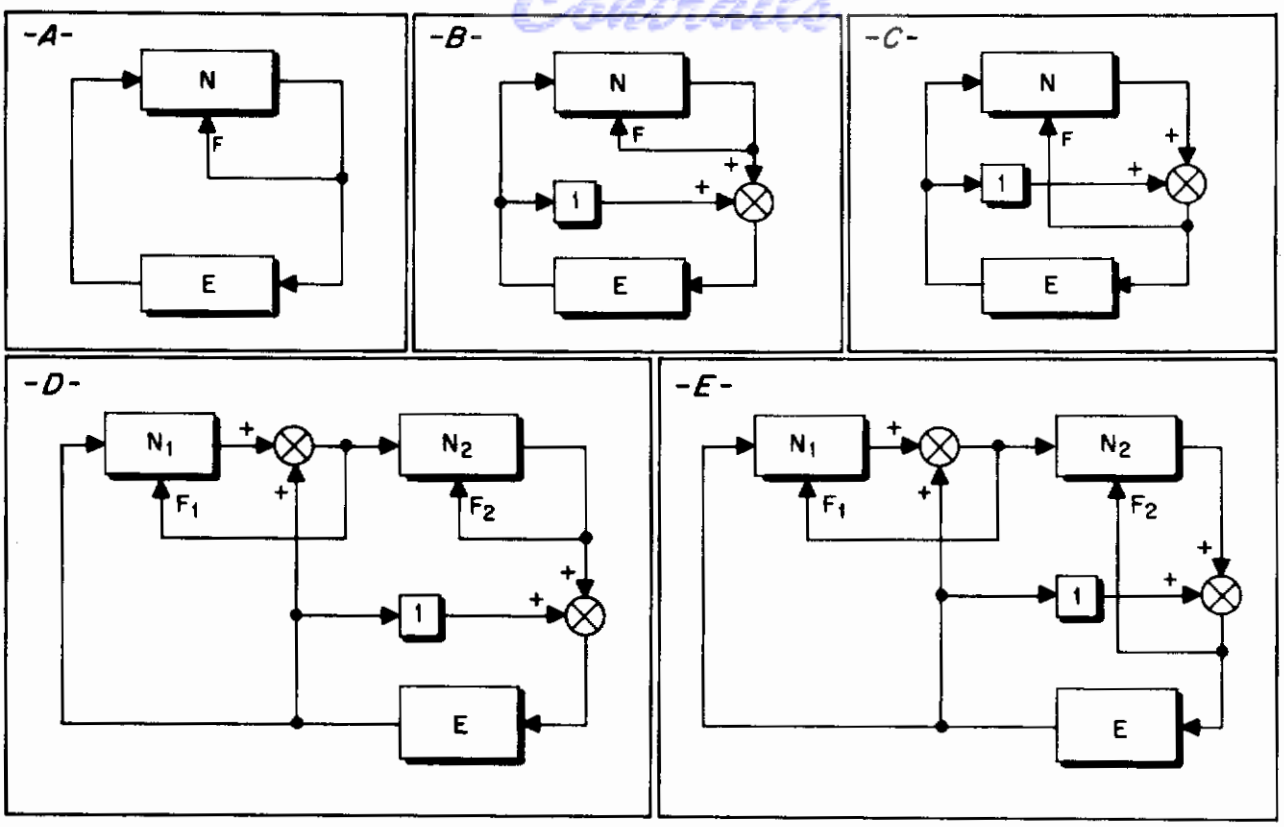


Figure 25. First Test Network for New Concept

The neuron model network used for this test was then extended, and an adjustment rule postulated so that the neuron model computed the optimum parameters automatically. In this adjustment rule a neuron model considered the reaction of its inputs to its output and compared the actual input reaction to the predicted input reaction. Any deviation of the predicted reaction from the actual reaction was integrated, and the deviation integral was used to adjust the predicted reaction. When adjusting the predicted reaction, the neuron model subsequently adjusted its synaptic weight. The result of the adjustments was a predicted input reaction that matched the actual input reaction. At the same time, the optimum synaptic weight was computed. The major problem with the simulation was the difficulty encountered when the size of the system was increased to three or more neurons. The instrumentation problems arising from this attempt were too restrictive.

To further the instrumentation development, six networks were set up and tested to determine the effect of feedback on the computation of synaptic weights. Figure 26 shows block diagrams of five of the networks tested (the sixth will be shown in Figure 28). The networks used either one or two neuron models and two environments, an integrator and an exponential decay. To test the networks, the optimum system response and the corresponding synaptic weights were computed. The networks were then tested and the synaptic weights and system responses recorded. These recorded values were compared with the ideal computed values.

Figure 27 shows a set of typical responses. The responses shown in Figure 27A and 27B were obtained from the test networks shown in Figure 26A and 26B, respectively. Note that in Figure 27A, the predicted value of the input is updated until it matches the actual value. The updating is a result of the synaptic weight adjustment. The neuron models at first computed



N = NEUROMIME MODEL

F = FEEDBACK INFORMATION USED FOR SYNAPTIC WEIGHT COMPUTATION

E = ENVIRONMENT

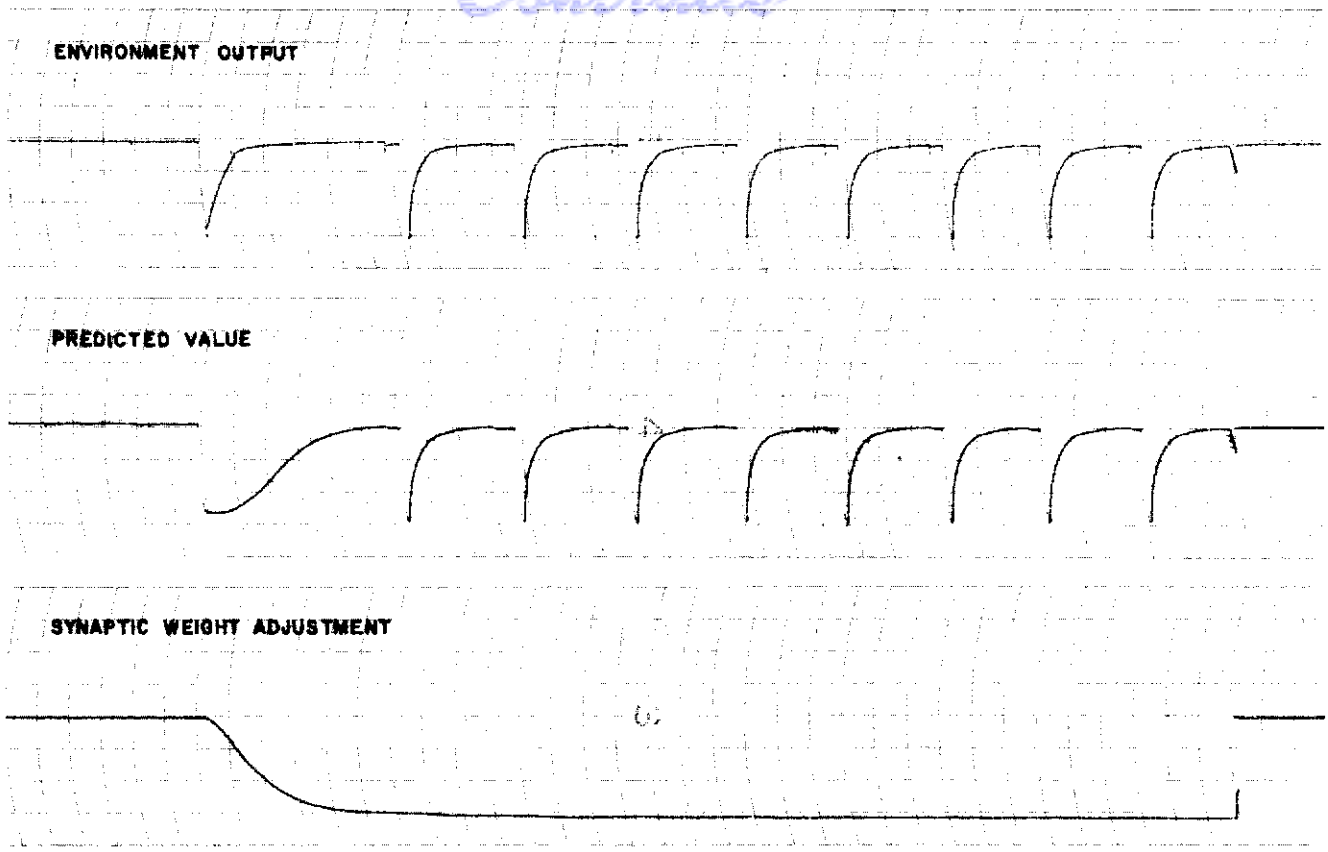
Figure 26. Five Preliminary Test Networks

synaptic weights that overshot the optimum synaptic weight, and then approached the optimum value asymptotically. The environment used for this test was an integrator. One control input, that of the neuron model, was used. In this case, the information fed back to the neuron model was the neuron model's own output.

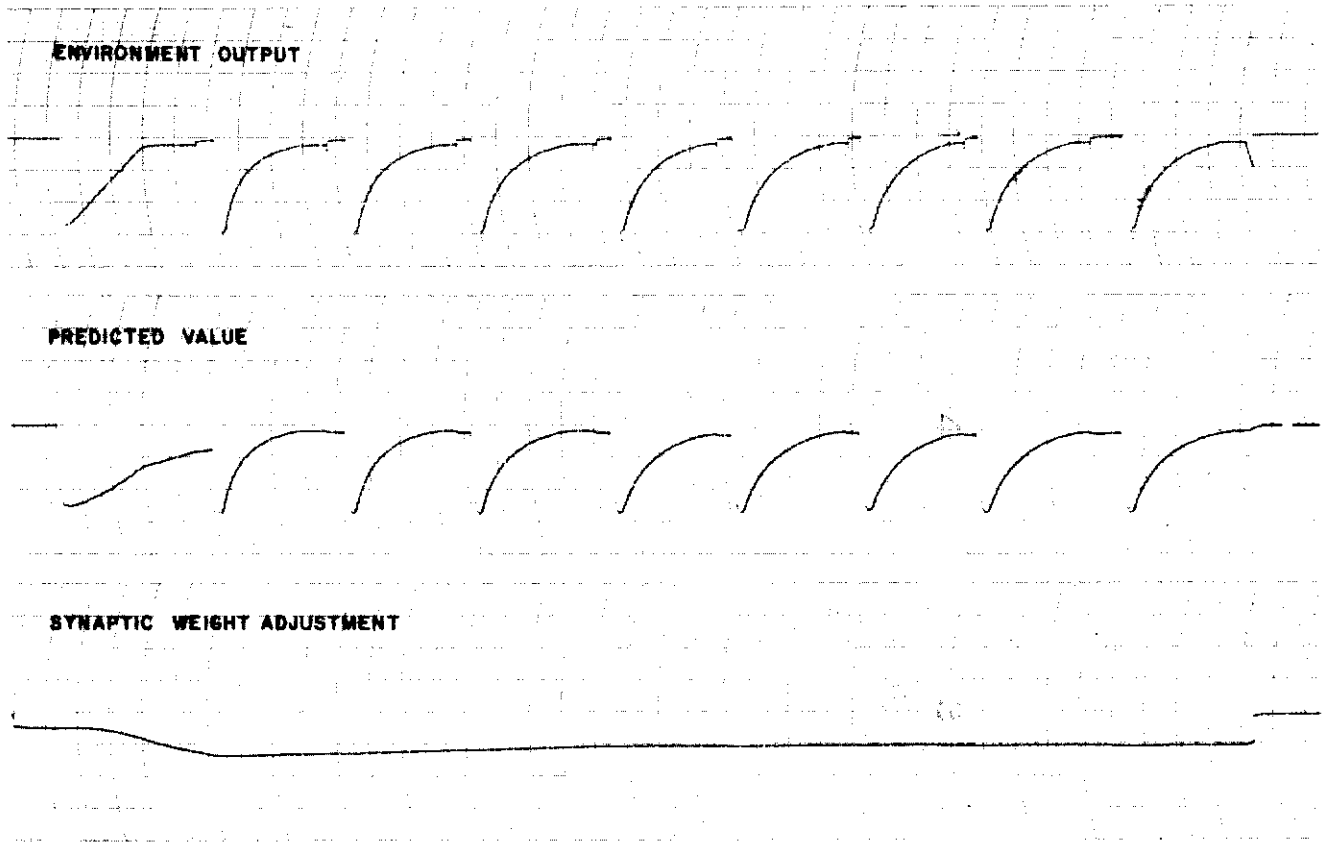
When another control input was added, as in the system shown in Figure 26B, the neuron model computed a non-optimum synaptic weight. The extra control input was viewed as another neuron model or, possibly, many other neuron models. In this case also, the feedback information was the neuron model's own output. Figure 27B shows a typical response to disturbances injected periodically into the system. Notice that the predicted path and the actual path of the input trajectory are different. This difference is caused by the non-optimum synaptic weight computation. The synaptic weight computed was such that the integral of the error between the predicted path and the actual path of the input trajectory was zero.

One general conclusion resulting from analysis of this test series is that the neuron models seem to require feedback information about the activity beyond the synaptic junction, rather than activity at the junction.

A block diagram of the last test in the preliminary test series is shown in Figure 28. This test examined one particular adjustment rule for computing synaptic weights. The environment used for the tests was an integrator. The equations derived for the system in Figure



A. First Test Network (Figure 26A)



B. Second Test Network (Figure 26B)

Figure 27. Typical Responses of Preliminary Test Networks

28, using Pontryagin's maximum principle, were

$$\dot{y} = bx \tag{33}$$

$$-\dot{P} = y \tag{34}$$

$$m = -\beta P \tag{35}$$

where y was the environment output, m the network output, β and P parameters controlled by the neuron model, and b the fixed environment parameter. The adjustment rule used for this system was

$$y = K\dot{y} \tag{36}$$

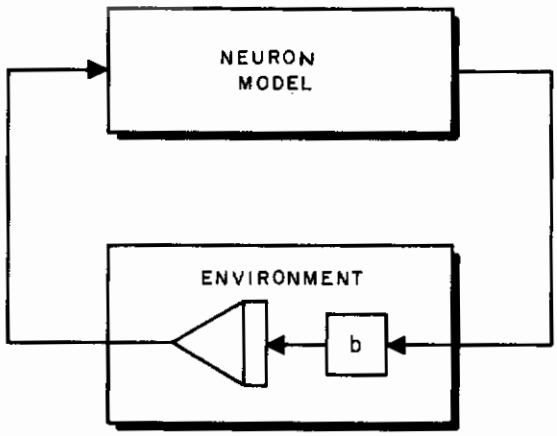


Figure 28. Last Preliminary Network Tested

where K is a parameter controlled by the neuron model. By comparing y with its time derivative $\dot{y}$, the neuron model computed the value of K. Proper manipulation of Equations 33, 34, 35, and 36 yielded the equation

$$P = -Ky, \tag{37}$$

which gave the value for P when the value for K is given. Combining Equations 33 and 35 yielded the equation

$$\dot{y} = -\beta^2 P, \tag{38}$$

which the neuron model used to compute β given the value for P. Notice that Equations 36, 37, and 38 are interdependent. The three equations reacted in a "bootstrapping" fashion to compute simultaneously all three parameters, K, P, and β. The analog computer circuit of the neuron model used for this test is shown in Appendix II, Section B. There were two parts to the test. The first part showed that the hand-calculated optimum parameters were optimum. The second part showed that the neuron models actually computed these optimum parameters.

For the first part of the test, the system equations and the criterion used by the neuron model were used to calculate optimum values of K, P, and β for various values of the environment's parameter, b. The calculated optimum values for K, P, and β were tested to be certain they were indeed optimum. The criterion used to determine optimality was that used in the development of the optimum system (Equations 33, 34, and 35). This criterion is

$$C = \int_0^T (y^2 + m^2)\, dt$$

where C (cost) is to be minimized, and y and m are the neuron model's input and output, respectively. To test the calculated optimum values, the values were set up on the analog computer and the values of C were measured. The values of C for corresponding values of K, P, and β around the optimum values were also measured. The tests were conclusive, proving that the calculated optimum values were indeed optimum. A typical result is plotted on the graph in Figure 29.

For the second part of the test, the system was tested with five values of the environment parameter b. The environment parameter b had a range of 0.5 to 2.0. A step voltage was used as an output from the environment. The neuron model computed the optimum values for K, P, and β as verified by the previous test on optimum values. A set of ideal values for the parameters K and b was plotted on the graph in Figure 30 and compared to the set of values computed by the neuron model.

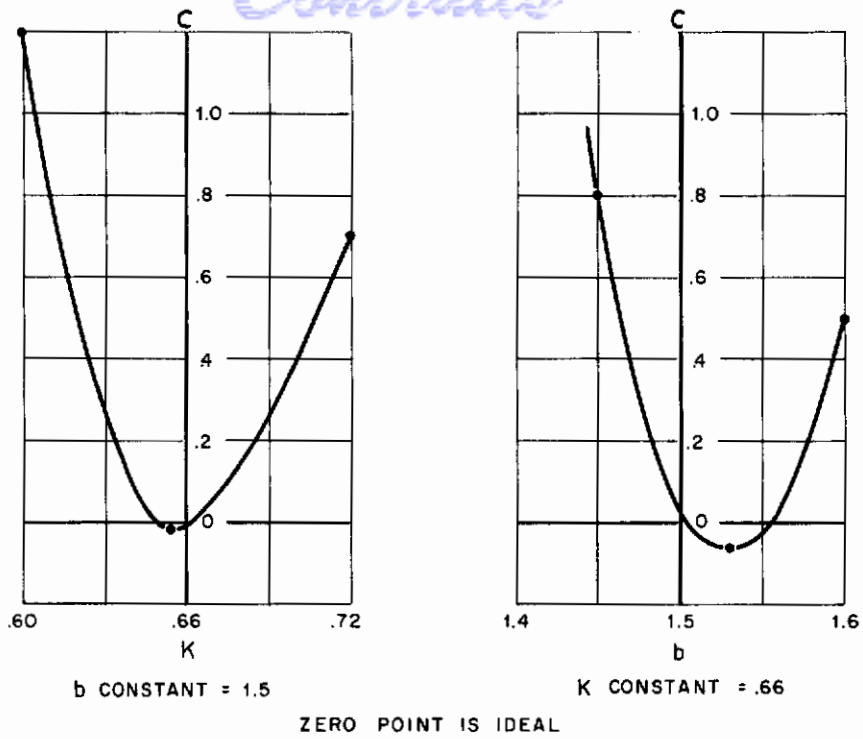


Figure 29. Test Results Showing Cost

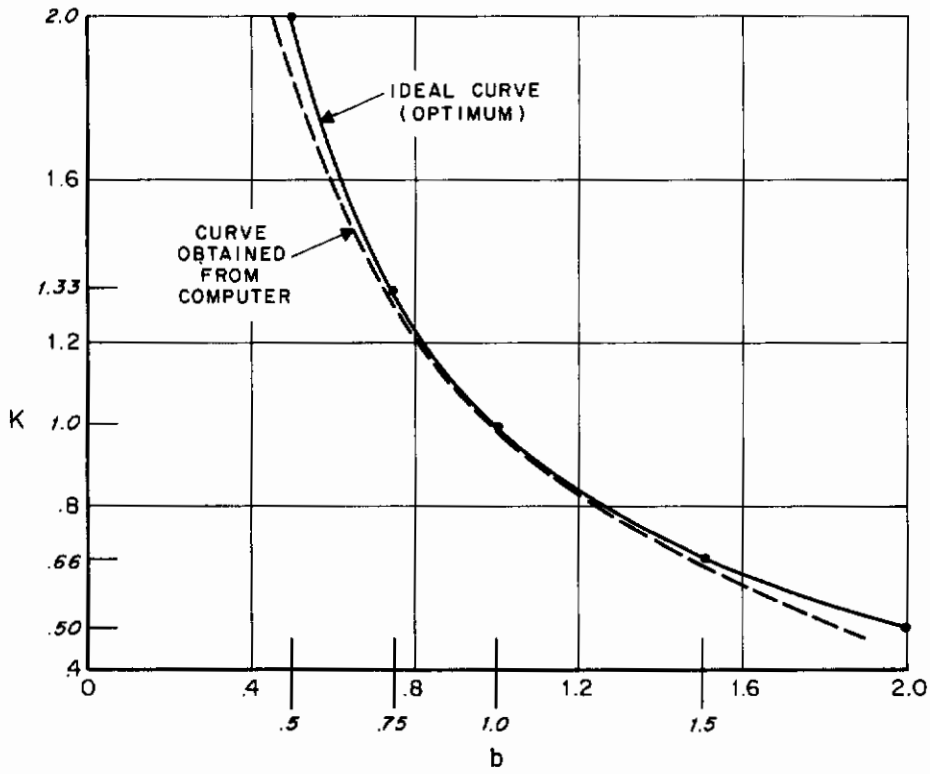


Figure 30. Comparison of Ideal K versus b Curve with Computer K versus β Curve

2. Tests on More Complex Systems

Two tests were performed in this test series. In both tests the environment was second order.

a. First System Tests. The environment for the first test system, shown in Figure 31, was quite simple, having only two parameters. An additional equation was used for the simulation. The equation was

$$P = KY \tag{39}$$

The specific parameter and state variable matrices for this test were as follows:

$$Y = \begin{bmatrix} y_1 \\ y_2 \end{bmatrix} , \quad A = \begin{bmatrix} 0 & a \\ 0 & 0 \end{bmatrix} , \tag{40}$$

$$B = \begin{bmatrix} 0 & 0 \\ 0 & b \end{bmatrix} , \quad P = \begin{bmatrix} P_1 \\ P_2 \end{bmatrix} , \tag{41}$$

$$M = \begin{bmatrix} 0 \\ m \end{bmatrix} , \quad K = \begin{bmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{bmatrix} . \tag{42}$$

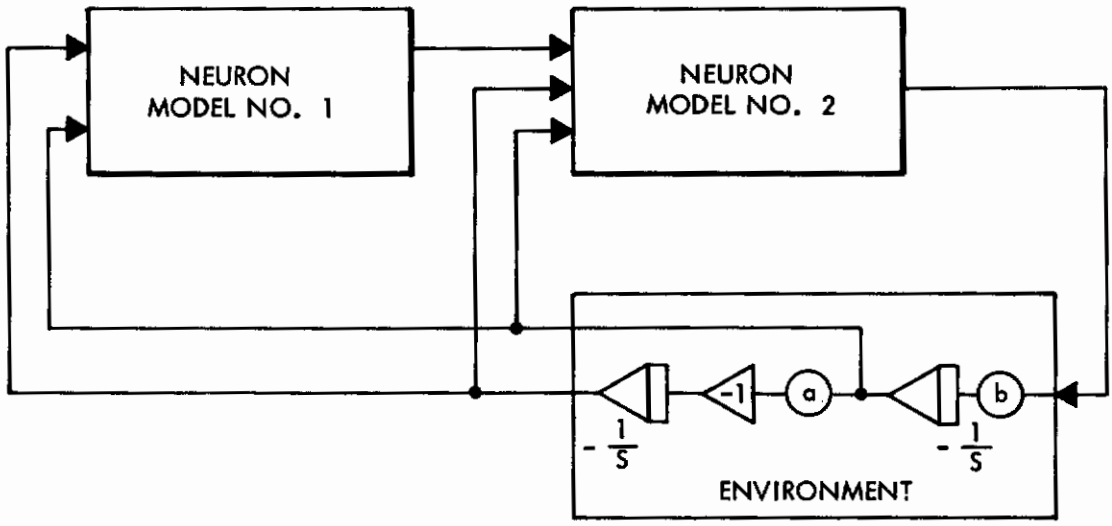


Figure 31. First Complex Test System

Further, the matrix K had the following form in terms of the parameters a and b:

$$K = \begin{bmatrix} \dfrac{\pm\sqrt{1+\dfrac{2a}{b}}}{a} & \dfrac{1}{b} \\ \dfrac{1}{b} & \dfrac{\pm\sqrt{1+\dfrac{2a}{b}}}{b} \end{bmatrix}, \tag{43}$$

where Y is the environment output matrix, M is the neuromime network output matrix, and A and B are environment coefficient matrices. The matrix P entered into the equations from Pontryagin's maximum principle. The matrix K enters into the equations from an assumption described in detail in the main text of this report. The criterion used to evaluate the tests was the same as that described at the beginning of this appendix.

There were three parts to the test. The first part was performed to show that the optimal controller did yield a minimum value for the system cost function. Various parameter value sets were tested by holding all parameters in a set constant except one. This one parameter was varied and the cost function measured. Each parameter value in a set was tested in like manner. During these tests, neuron models made no parameter computations.

The second part of the test was performed to show that the neuron models did compute the optimum parameters. The neuron models tested used a hill-climbing technique to compute their parameters. Various parameter value sets were programmed into the environment and initial conditions set for the neuron model parameters. Then environment disturbances were injected and the neuron models were allowed to compute.

The objective of the third part of the test was related to the second. Tests were performed to observe the reaction of the neuron models to two simultaneous disturbances of the same (and opposite) polarity. The two simultaneous disturbances, which formed the complete disturbance vector, took the form

$$D = \begin{bmatrix} d_1 \\ d_2 \end{bmatrix}.$$

Note that in the second part of the test, either d_1 or d_2 of the disturbance vector was zero.

In general, all of the test objectives were achieved. Figure 32 is a graph of normalized cost versus α. The parameter α corresponds to the entry "a" in the environment coefficient matrix A in Equation 40, and was computed by the neuron model. The zero point on the cost axis is the theoretical point at which the cost is a minimum. The graph shows a 2 percent deviation from the theoretical point. Figure 33 is a comparison of various theoretical values of α and the actual values computed by the neuron models. The straight line is the locus of all points where the theoretical and actual values of α were equal. The graph shows minor deviations of the points tested. No point has more than a 5 percent deviation.

The curves in Figures 34 and 35 are similar to Figures 32 and 33, respectively, but represent typical results of the neuron model computations of the K matrix parameters in Equations 39, 42, and 43. The graphs show accuracies similar to those obtained with the α parameter computations.

Figure 36 is an analog computer recording. It shows the system reaction to a disturbance in the environment. Starting at the top of the chart, the first two channels make up the

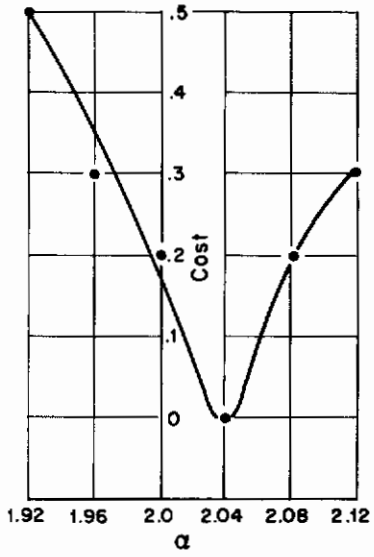


Figure 32. Normalized Cost versus α

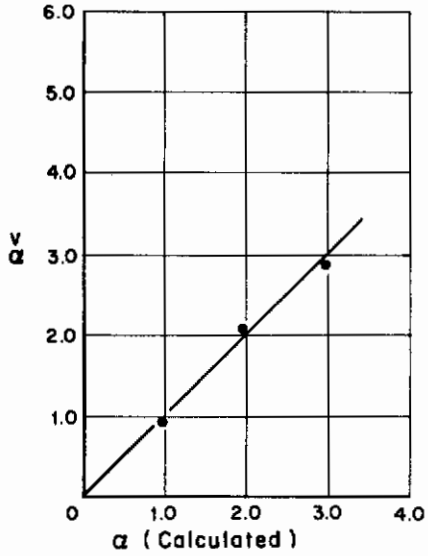


Figure 33. Theoretical versus Actual Values of α

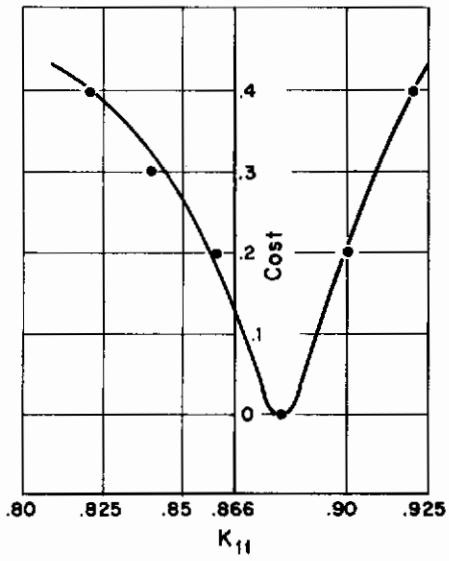


Figure 34. Normalized Cost versus K

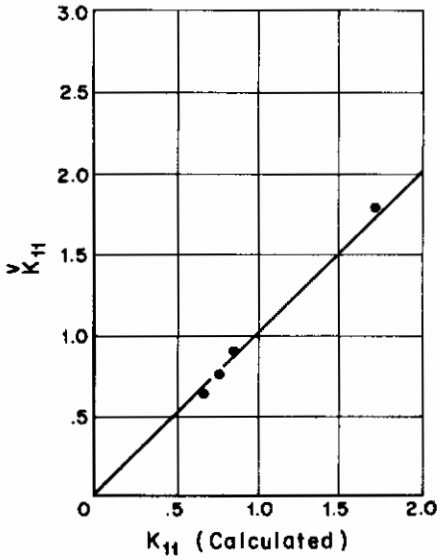


Figure 35. Theoretical versus Actual Values of K

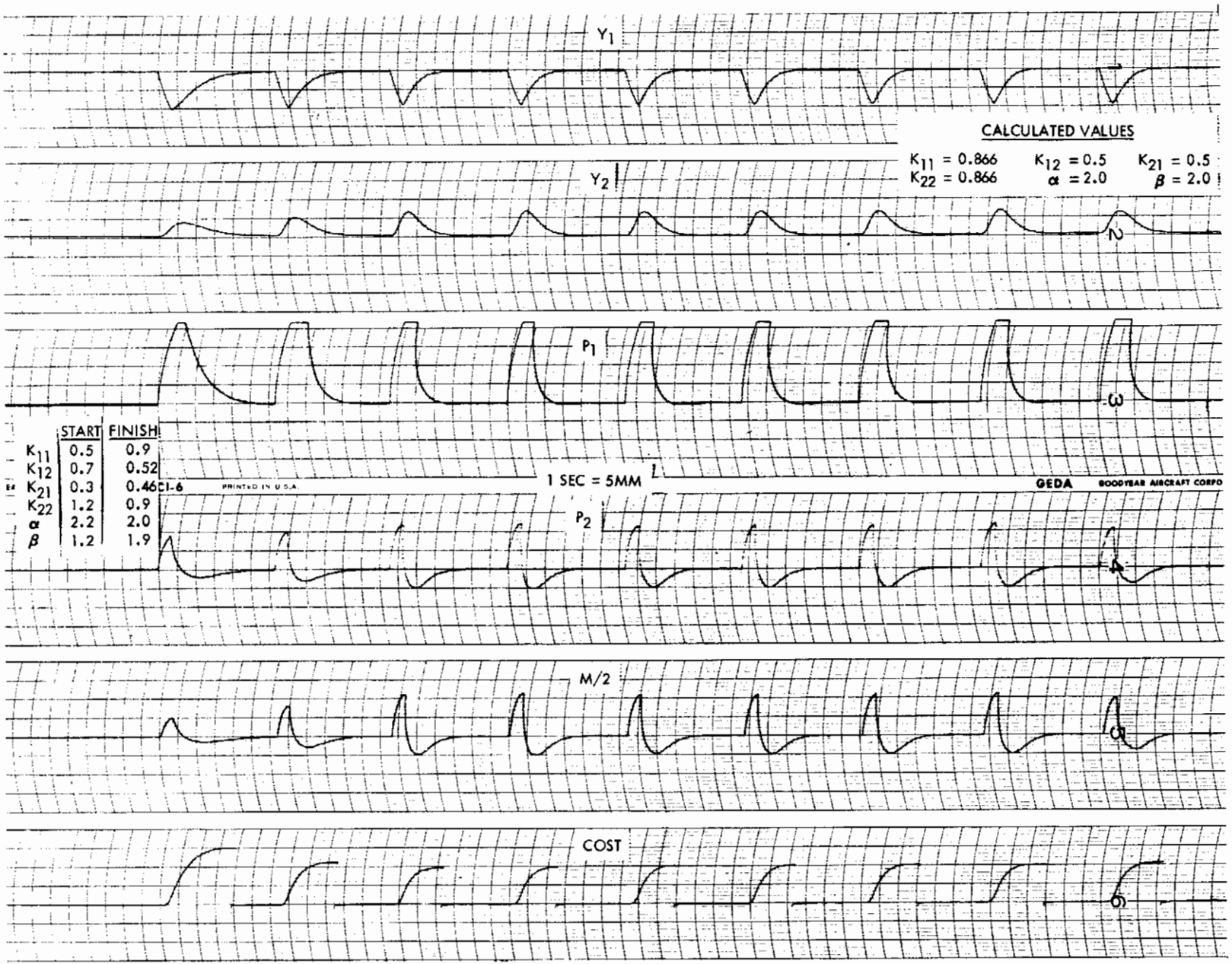


Figure 36. Analog Computer Recording of System Response (First Test Series)

environment output matrix, Y, as in Equation 40. The next two channels, labeled P_1 and P_2, make up the matrix P, as in Equation 41. The last two channels are the neuron model output, M, and the cost function. The system was given ample time to settle down after a disturbance before another disturbance appeared. Note how the value of the cost function decreased as the system adjusted to the optimum parameters. After the third disturbance, the cost function remained fairly constant, indicating completion of the optimum parameter computation.

When one of the neuron model's parameters was held fixed, the neuron model computed parameter values that were not optimum for the total system. However, the parameter values computed still caused the cost function to be reduced. This indicated that the neuron models did the best they could under other than optimum conditions. When part or whole of a neuron model was "damaged," the remaining neuron models readjusted their parameters to allow for the "damage." The result of the readjustment did not yield the optimum system response, but the response was near optimum. Of course, any damage that might disconnect the neuromime network from the environment would stop system control.

In the third part of the test, the correlation between the two disturbances seemed to confuse the neuron model computations. Improvements were made on some of the analog circuits to lessen the confusion. Although the parameters computed by the neuron models were not optimum, the value of the cost function was decreased.

In all tests the system was stable. An exception to this statement occurred when the neuromime network's control output had a reversed sign. In this case the system diverged.

b. Second System Test. The environment used for the second test was more complex than that used for the first test. The second-order environment was changed to admit all entries in the environment coefficient matrix, A, as follows:

$$A = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}. \tag{44}$$

A comparison of Equation 44 with Equation 40 illustrates the increased complexity.

The test on this system was composed of three parts. The first part showed whether or not the system parameters, computed by hand, minimized total system cost. Each parameter affecting the system cost was varied separately, while other system parameters were held constant at the optimum values computed by hand. The result of the test series was a set of cost versus parameter variation curves. The second part showed whether or not the neuron model computed the optimum parameter values. The test was made by simply allowing the neuron models to react to environment disturbances. The result of the tests was a set of analog computer recordings showing a time history of the total system cost as affected by the neuron model parameter adjustments. The third part demonstrated the ability of the neuron model to adapt to external damage. The external damage was an environment parameter change.

Figure 37 shows the results of the first part of the test. A glance at the shape and location of the curves is enough to see that the objective was accomplished. The method used to compute the optimum parameters by hand was shown to be correct. Further, the method can easily be set up for digital computers.

Figure 38 shows the results of the second part. Note the change in the system response as the parameter values approach optimum. The response went from a damped oscillation to an exponential decay, which was the expected result. Note also the reduction in total system cost as the neuron models adjusted their parameters.

Once the optimum parameters were reached, the total system cost tended to oscillate slightly about a minimum value. This slight oscillation was caused by small variations in the

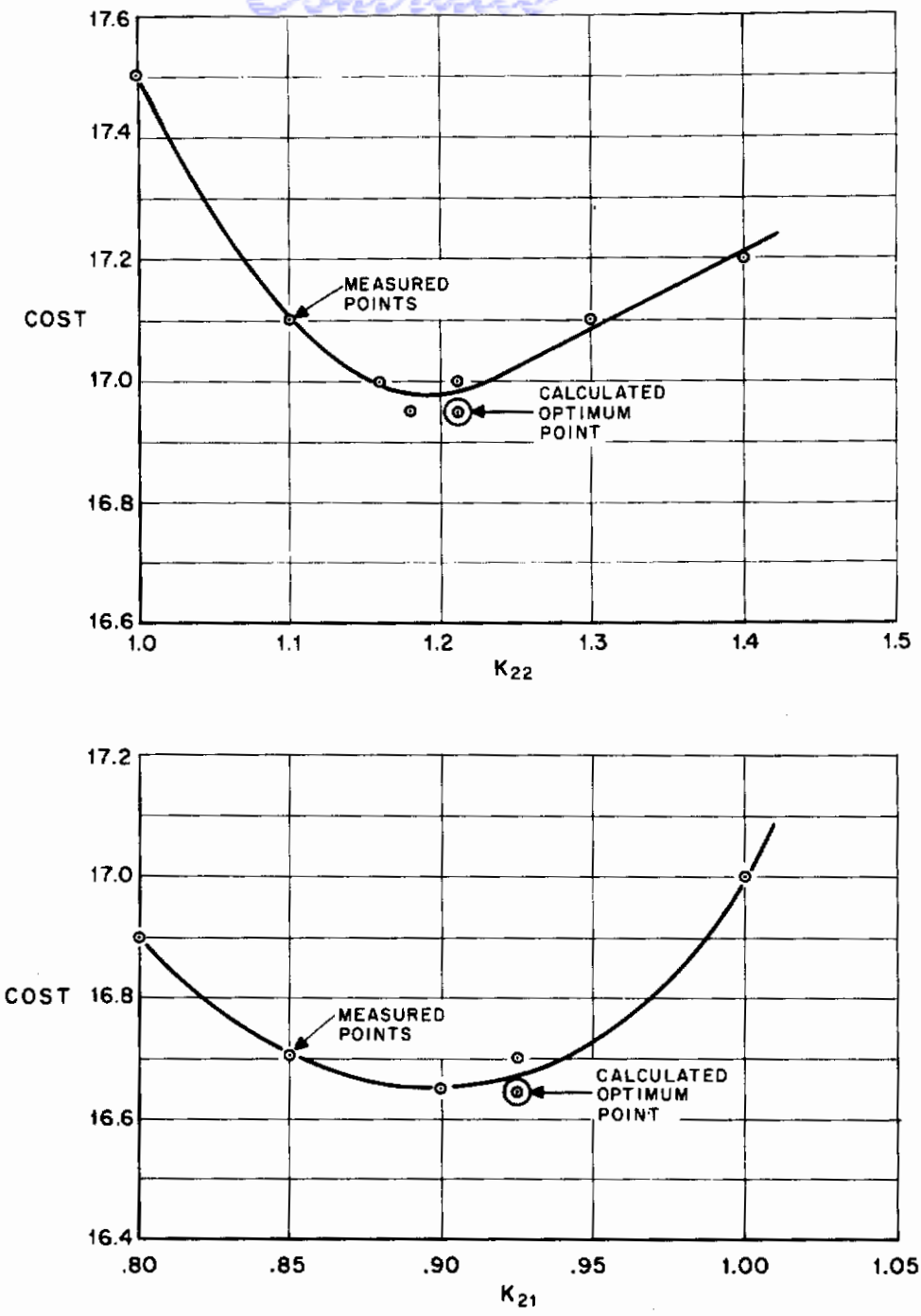


Figure 37. Cost versus Parameter Variation (Second Test Series)

parameters about their optimum values. The total system cost curve was flat around the minimum cost; thus cost variations about the minimum cost were expected to be small.

The results of the third part of the test were as expected. The neuron models adjusted their parameters within their adjustment range to accompany changes in the environment's parameters. Actually, adjusting to a new set of parameters after a parameter change was no different to the neuron models than initally adjusting to the original parameters. The worst case for an environment parameter change was a change to some value outside the neuron model's adjustment range. The neuron models in this case did not compute the optimum

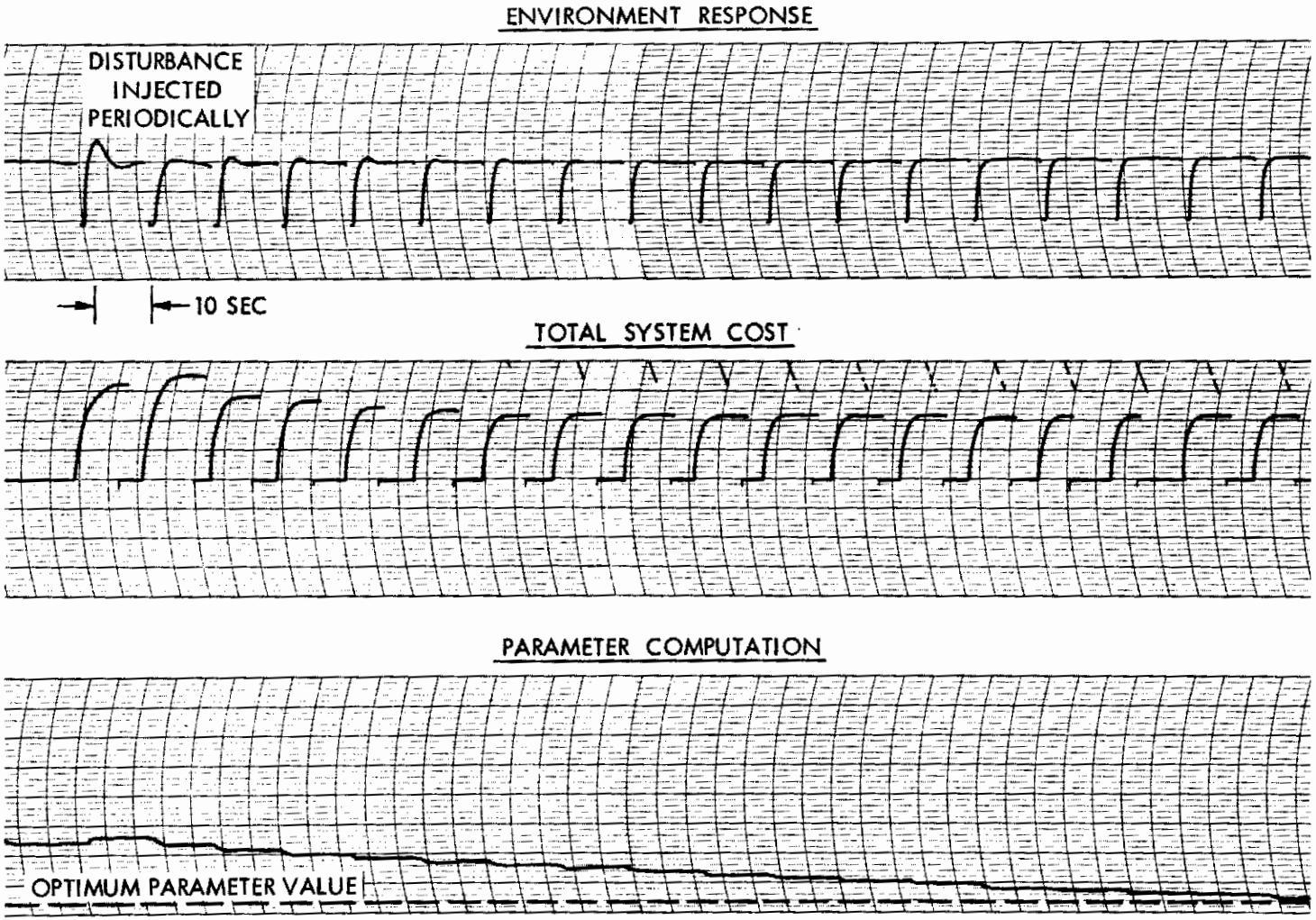


Figure 38. Optimum Parameter Computation (Second Test Series)

parameters. Rather, the maximum (or minimum) values for the neuron model's parameters were computed. Even though in this case total system cost was not minimum, it was reduced.

3. Tests on the Convergence of the Neuron Model Parameters to Optimum Values

Tests were designed and set up to examine the convergence of the optimum control parameters of a simple, first-order system. Basically, the tests performed were quite simple. A first order system was set up on the analog computer. At the beginning of the tests the controller was not the optimal controller. The task of the controller (a neuron model network) was to adjust its parameters and become the optimal controller of the plant.

The analytical basis for the convergence tests comes from the system equations listed at the beginning of this appendix. Without a formal derivation, the equation derived was in the form of an error equation, as follows:

$$(\alpha - A) - (\beta - B)\,\beta^t K = \phi$$

If A and B are given, a complete zero-error trajectory in the α, β plane is defined for each value of K. It was postulated that perturbations in the value of K allowed the α and β parameters to converge to the optimal parameters. In the tests two values of K were switched back and forth to demonstrate the theory. The two values were chosen far enough apart to exaggerate the convergence trajectory. Figure 39 shows the zero-error trajectories in the α, β plane for two values of K.

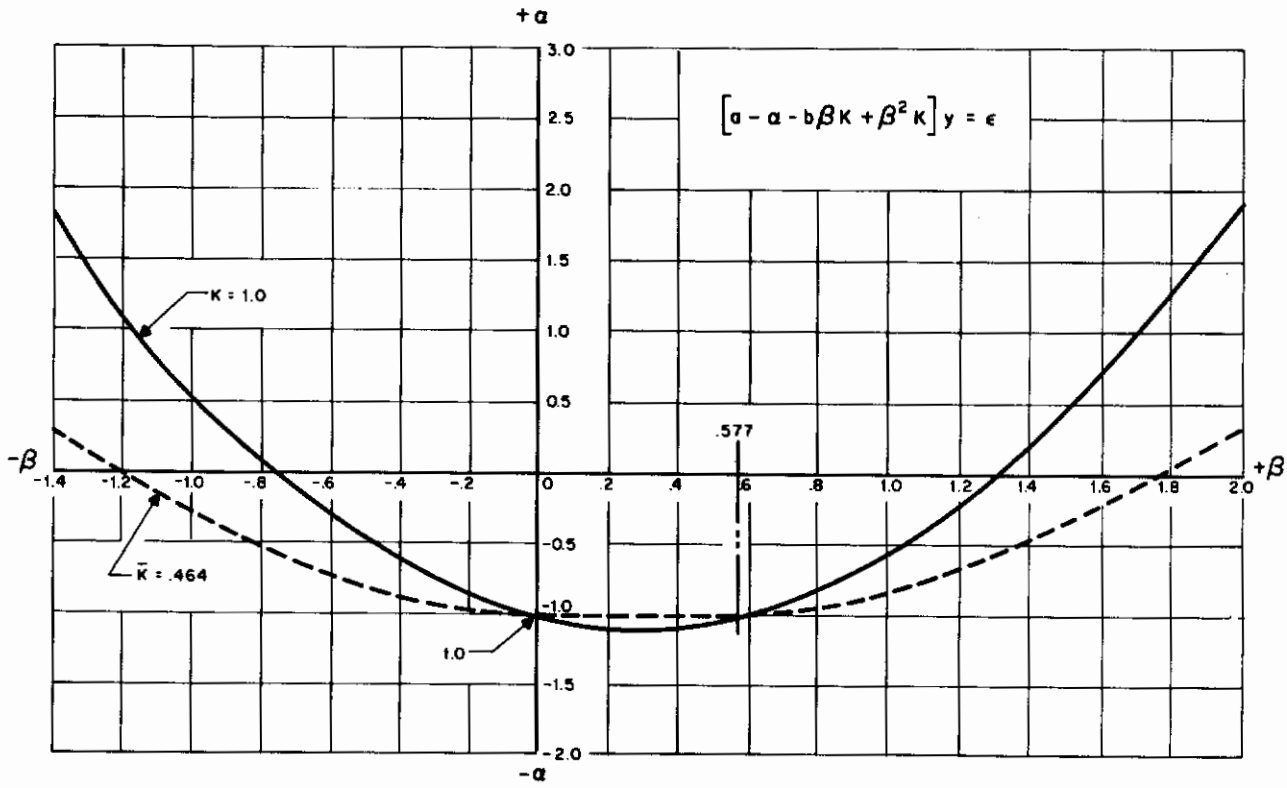


Figure 39. Zero-Error Trajectories for Two K Values

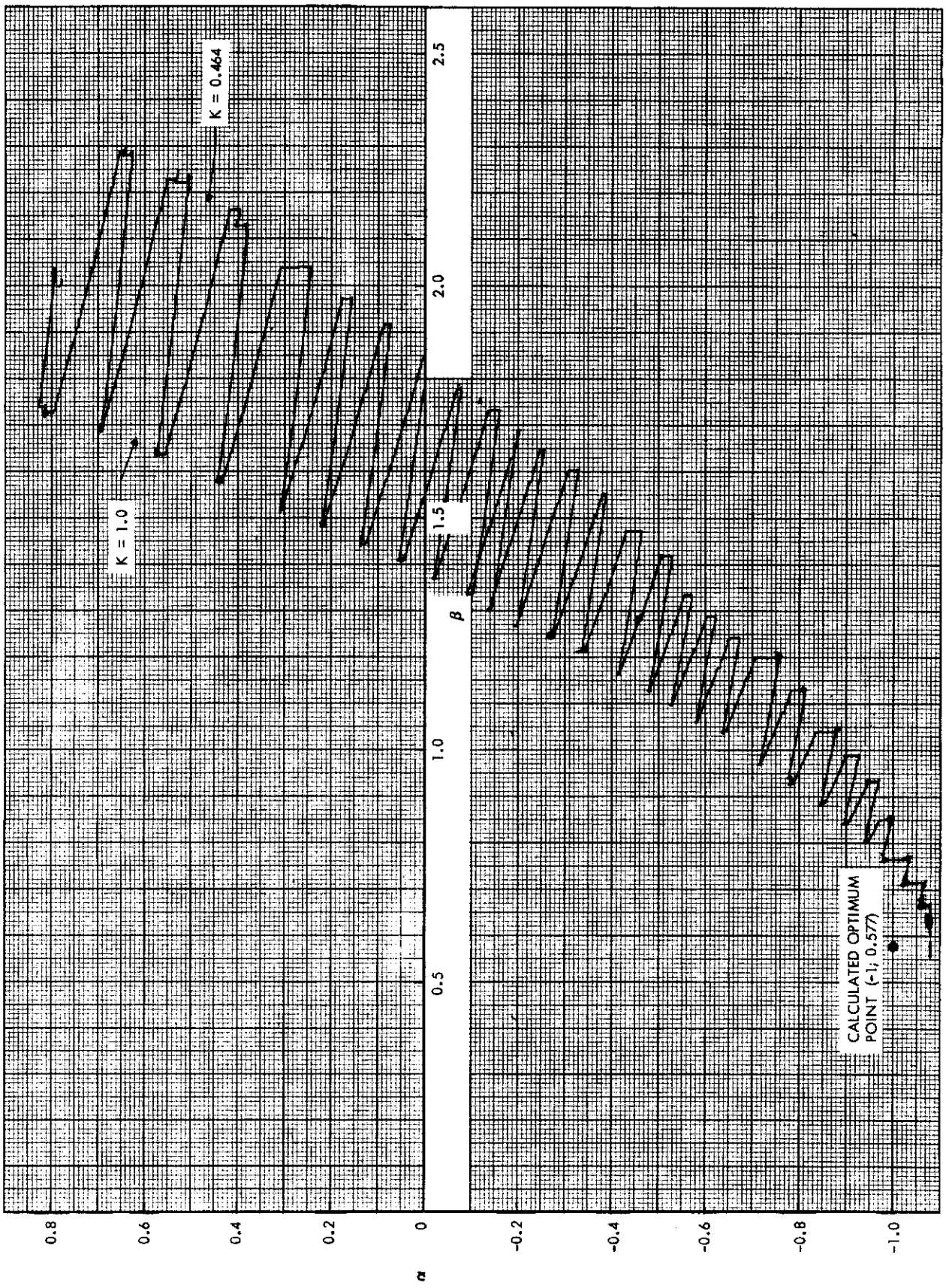


Figure 40. Typical Convergence Path for Initial Starting Point in Far Right Half of Plane

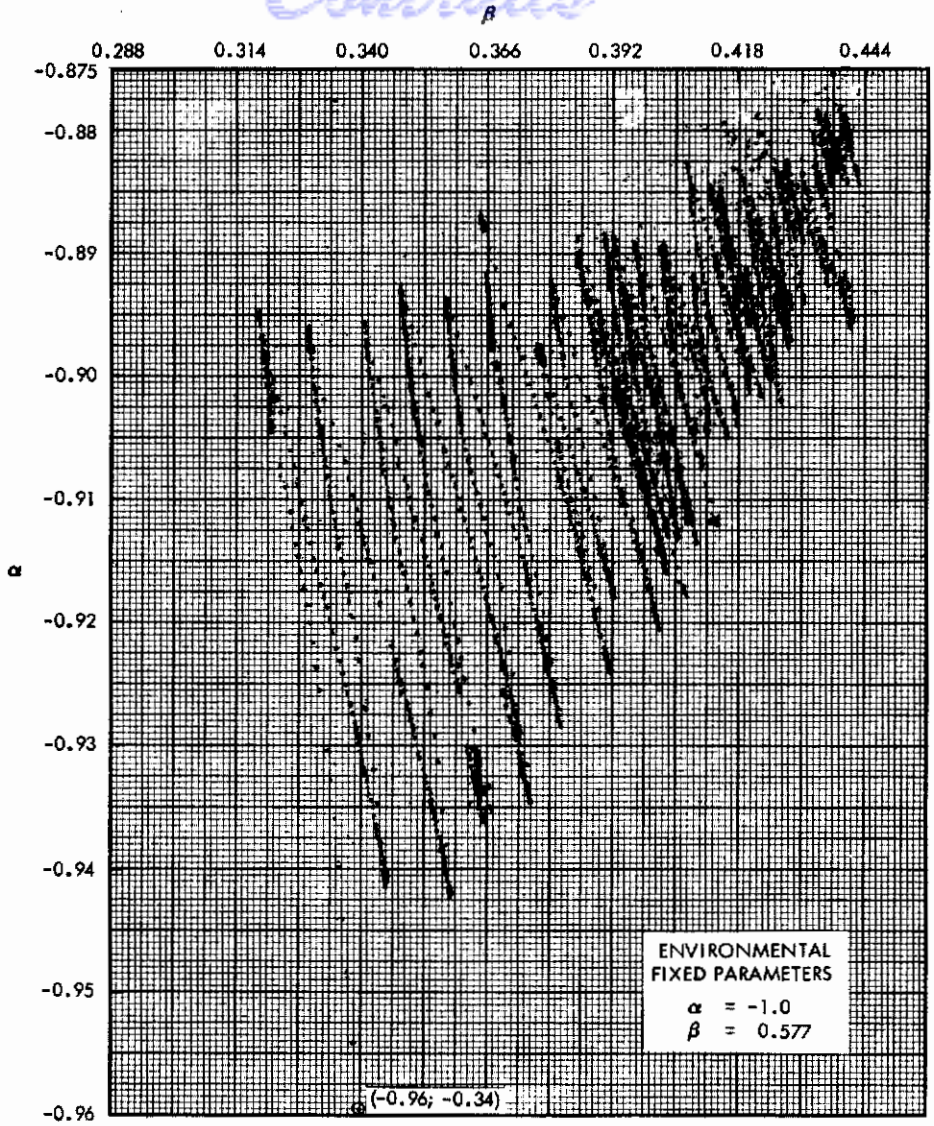


Figure 41. Typical Convergence Path for Initial Starting Point in Region between $\beta = 0$ and $\beta = B$

Figures 40, 41, and 42 show the results of the tests. Each figure is a section of the curve shown in Figure 39. Note the zigzag convergence path in all the figures. This path explains, at least in part, why the parameters in other experiments have taken so long to converge to the optimum values. The error equation and also the curve of Figure 39 show that with $\alpha = A$, β can have two values $\beta = B$ and $\beta = 0$, regardless of the value of K. Whenever the starting point (α, β) was on an error trajectory between those two points, the parameters moved to the right as shown in Figure 41 until they arrived at the $\beta = B$, $\alpha = A$ point. Figure 42 also shows the change in direction of movement for the initial starting point (α, β) as this initial starting point was moved from the left half of the plane to the right half of the plane.

Notice in Figure 42 that when the initial starting point was in the left half of the plane, the parameters did not converge to the optimum values. There were two cases:

(1) As the parameters, which were zigzagging between the two zero-error curves, approached the $\beta = 0$ axis, progress toward the axis stopped.

(2) When the initial starting point for the neuron model's parameters were on the $\beta = 0$ axis, the parameters went straight to the ($\alpha = A$, $\beta = 0$) point and stopped.

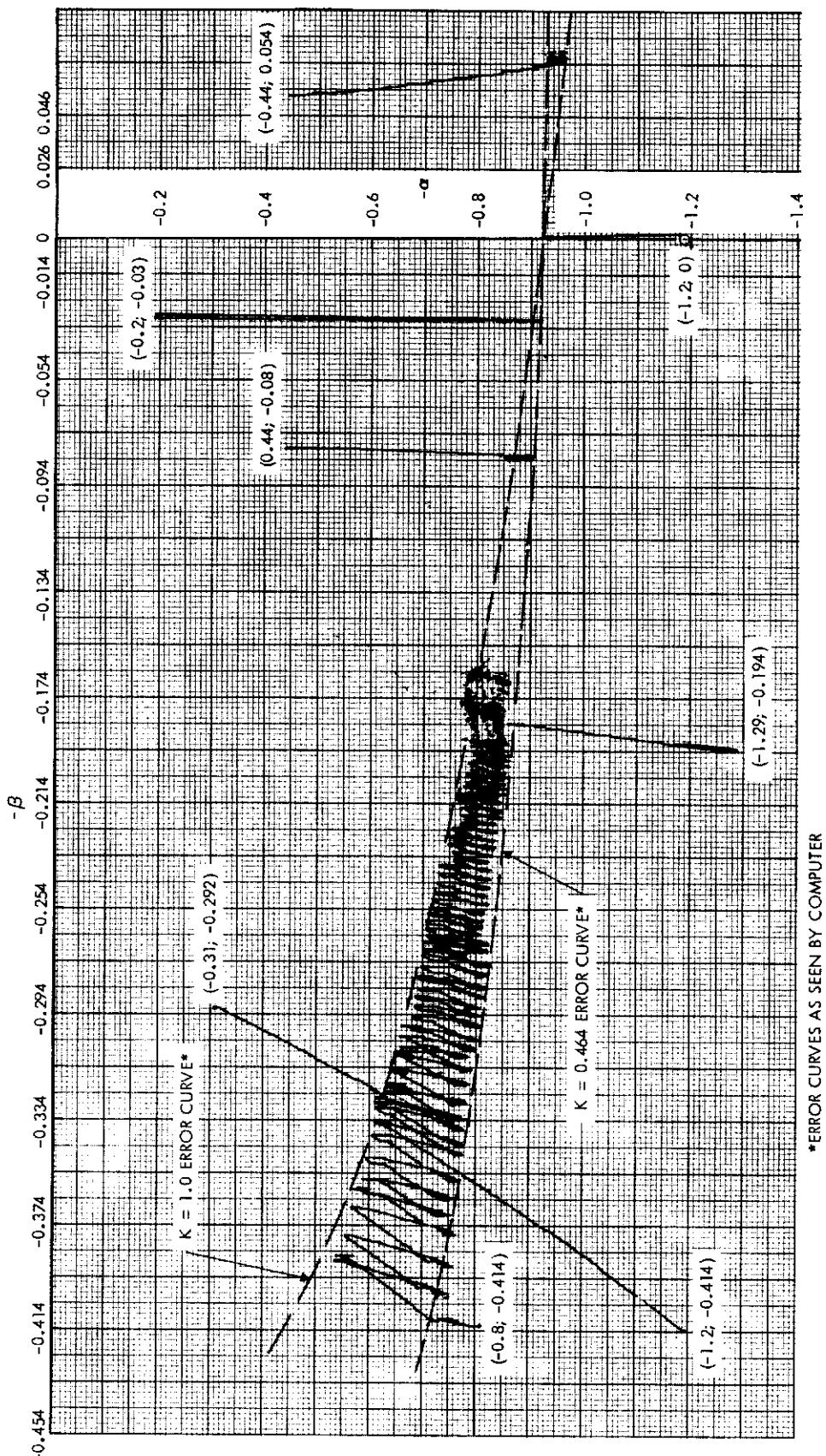


Figure 42. Typical Convergence Path for Initial Starting Point in Left Half of Plane

In the first case, the noise level interfered with the parameter computation and stopped the progress toward the $\beta = 0$ axis. In the second case, the β computation was dormant because $\beta = 0$ initially. The adjustment drive for the neuron model's β parameter was proportional to the initial β, and in this case was zero. With no signal to adjust its β parameter, the neuron model made no β adjustments.

Generalizing the above result, the $\beta = 0$ point must be excluded as a possible value of β to ensure that the neuron model's β parameter will converge to the optimum value.

4. Tests on Redundant Networks

Three tests were performed on redundant networks using the optimal control concept. One of the tests used the transistor hardware previously developed for the neuron model. The other two tests used the analog computer exclusively for the simulation of redundant networks. So far, small-scale networks have been tested. The lessons learned from these small-scale networks will be used as a guideline for the larger networks.

a. Redundant Network Test using Transistor Hardware. The first test used the Harmon neuromime (Reference 4) and other transistor hardware developed for the neuron models. This test was broken into two parts. The first part tested the reaction of the Harmon neuromime to weighted inputs. The second part tested the adjustment properties of a redundant network of neuromime elements.

The test for the first part was straightforward. A Harmon neuromime was set up having a weighted input. The weighted input had a constant pulse rate. The object of the test was to discover the weight required to force the Harmon neuromime to exactly follow the input pulse rate. Three different input pulse rates were used. Figure 43 shows the result of the test. Note that as the input pulse rate increases, the weight required to force the Harmon neuromime to follow also increases. Also, notice the plateau effect of the Harmon neuromime's output. The plateau occurs at a submultiple of the input pulse rate.

The test for the second part was set up as shown in Figure 44. The Harmon neuromimes were not directly in parallel. The inputs to all of the neuromimes were the same. All outputs were weighted and fed to another Harmon neuromime. This neuromime produced the network's output. Synaptic weight computers were set up on the analog computer. The task set for the redundant network was to maintain a constant input-output network relation, regardless of internal failure.

Figure 45 shows the synaptic weight values computed by the neuromimes as failures occurred. The failures were in the form of total neuromime function loss. Note that the synaptic weights computed by each of the surviving neuromimes increase to absorb the failures.

b. Redundant Network Tests using Analog Computer. The second test, shown in Figure 46, used the analog computer exclusively for the simulation. The environment used for this test was second order. For this test two neuron models were simulated in parallel, each contributing its share to the total system control. One of the neuron models had its parameters fixed at optimum values; the remaining neuron model had to compute its own optimum values. The first two columns of Figure 47 show the results of the computations.

After the two neuron models were controlling the system optimally, one of the neuron models was "destroyed." The last two columns of Figure 47 show the results of the one neuron model adapting to the severe internal damage. Even with only two neuron models in parallel, the loss of one did not affect the total system cost too drastically. Neither was the amount of adjustment required of the surviving neuron model drastic.

This result can be extended to networks with 10 or more neuron models. The effect of losing one, two, or three neuron models simultaneously in a network would not appreciably

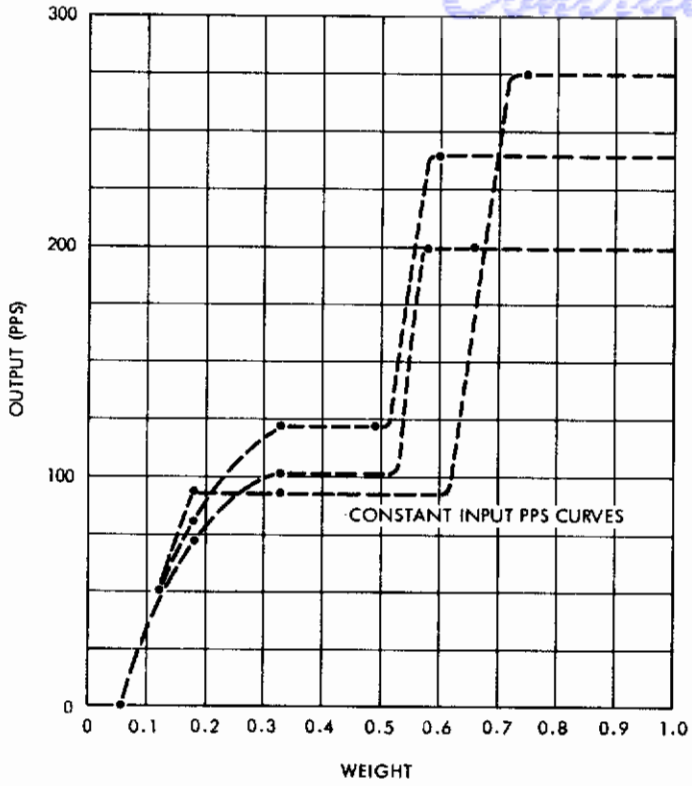


Figure 43. Test Results on Redundant Networks Using Transistor Hardware

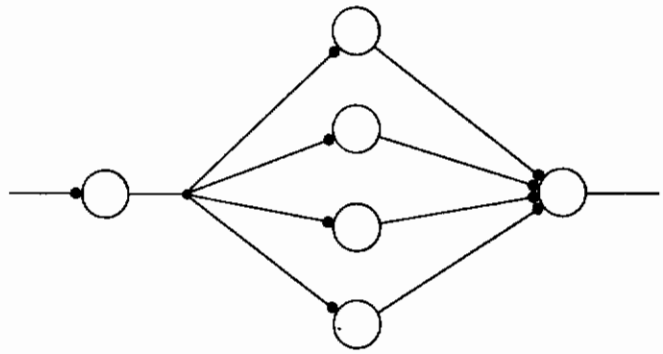


Figure 44. Redundant Network of Neuromime Elements

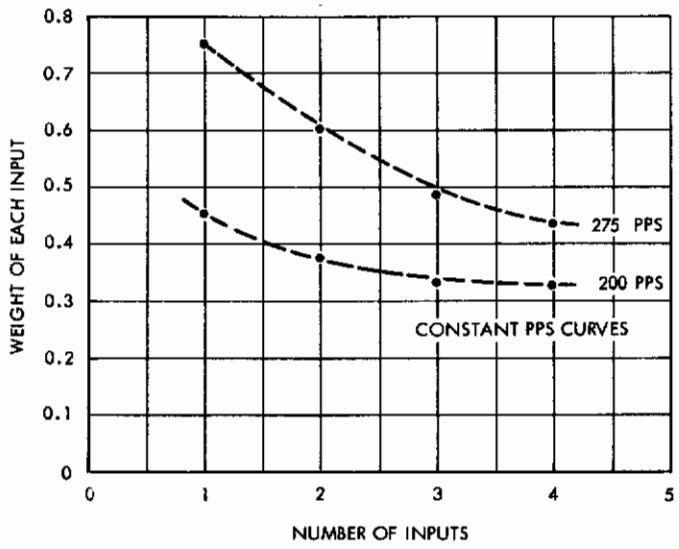


Figure 45. Synaptic Weight Computations of Redundant Network

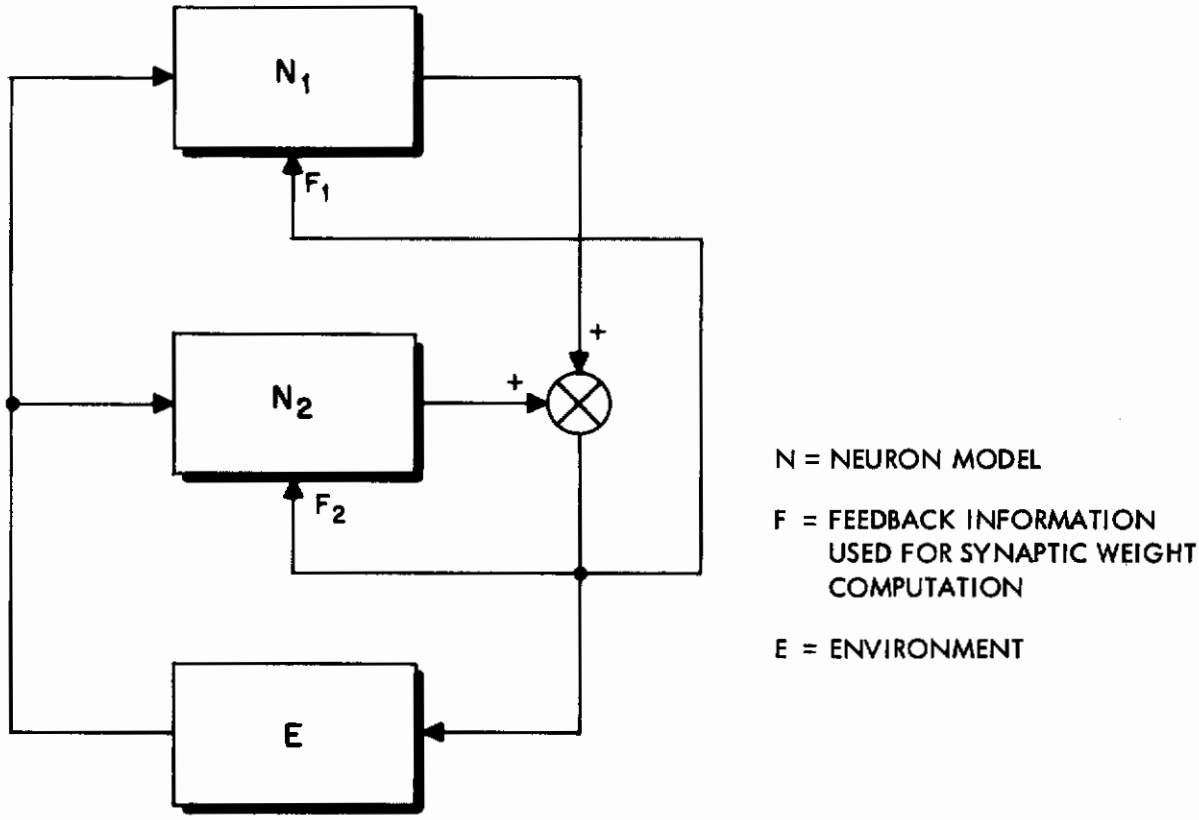


Figure 46. Redundant Network with Two Neuron Models in Parallel

affect the total network response. Further, the larger the number of neuron models in a network, the smaller the dynamic range of an individual neuron model need be.

The third test, similar to the one in Figure 46, also used the analog computer exclusively for the neuron model simulation. Two neuron models were simulated in parallel. Each model contributed to the total system control. This test differed from the second test in that both neuron models were free to compute their optimum parameters. To simplify the instrumentation, a first-order environment was used for this test. Interest was focused on the ability of the two neuron models to begin at some initial state and adjust to the optimum state, with both sharing the control load equally. The optimum state for the two parallel neuron models occurred when the parameters of both were equal, and when these parameters were the optimum values.

Figure 48 shows a plot of parameter differences. The optimum point on the graph is at the origin. The graph shows that the parameters of the two parallel neuron models converged to the optimum values simultaneously. The graph shows only that the parameters of both neuron models adjusted until the neuron models shared equally the control load; it does not show that the final adjusted parameters were optimum. However, the parameters checked well with the optimum parameters when they were checked at the end of each run.

Note the zigzag convergence pattern in Figure 48. This pattern resembles the convergence patterns obtained for the convergence test of the previous section of this appendix. The convergence pattern observed in this test was also due to perturbations in the K parameter. The pattern was obtained by holding the K parameter of one neuron model fixed at a value different from that of the other neuron model.

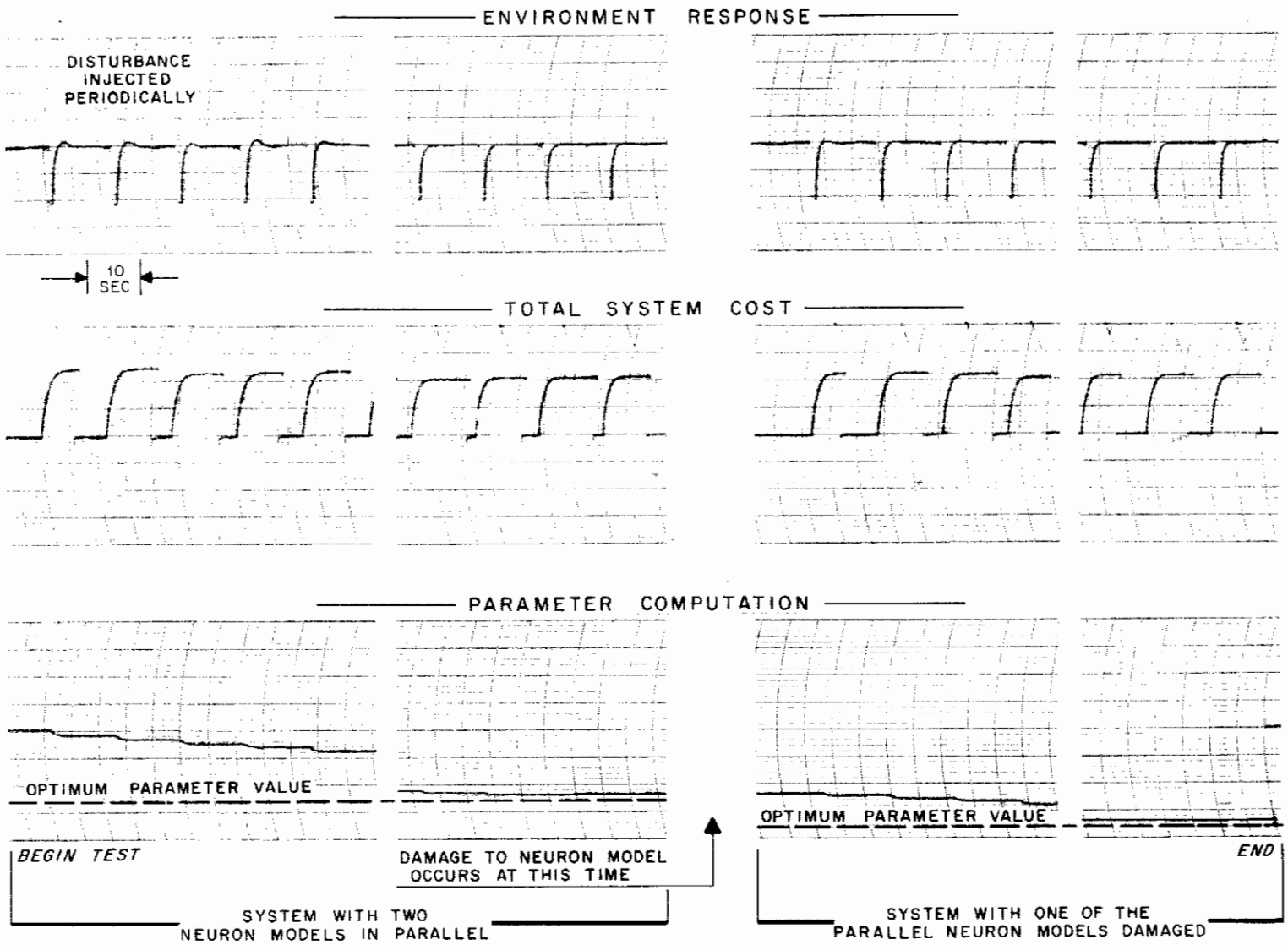


Figure 47. Adaptation after Internal Damage

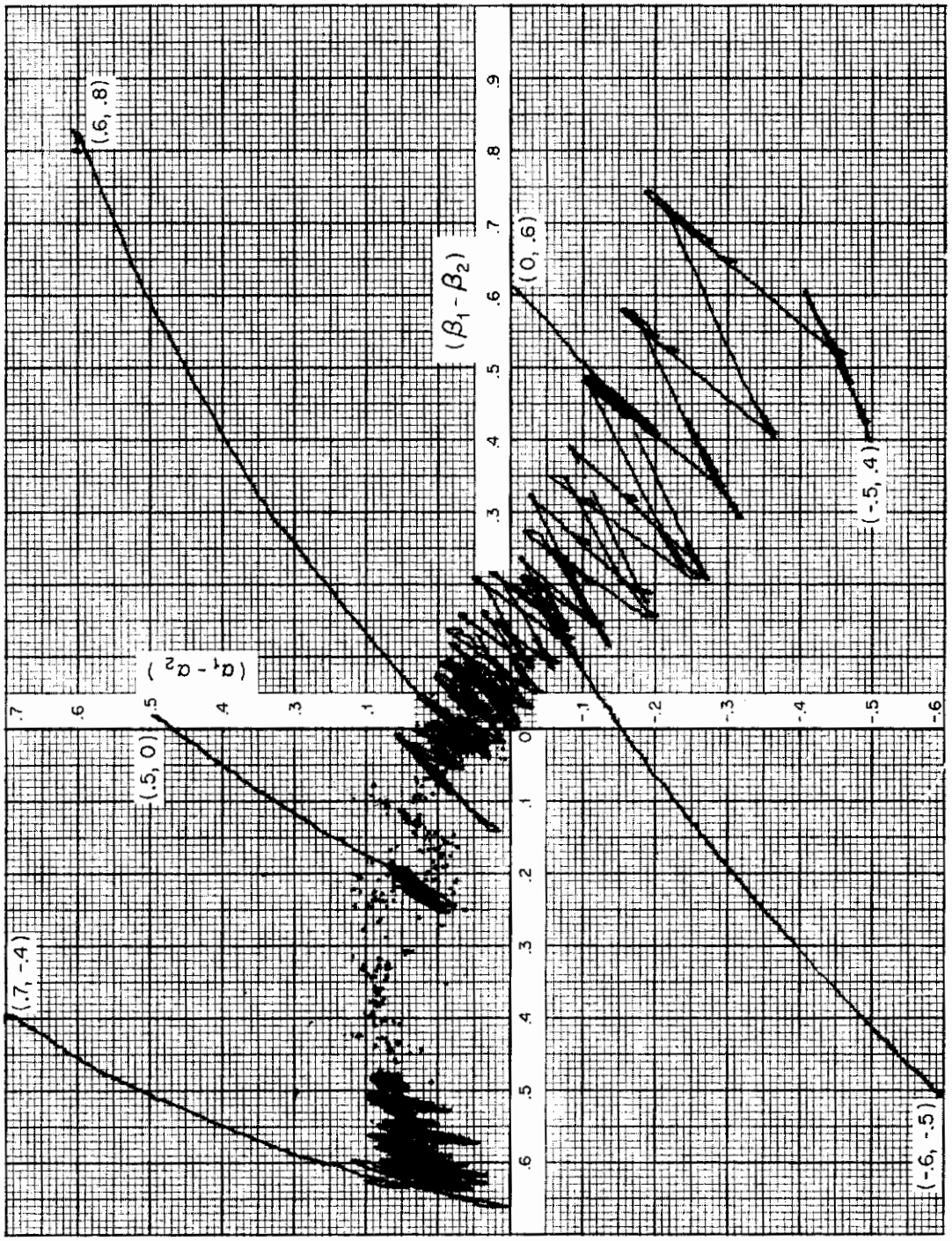


Figure 48. Plot of Parameter Differences - Two Neuron Models in Parallel

APPENDIX II

EQUIPMENT DEVELOPMENT

A. TRANSISTOR CIRCUIT DEVELOPMENT.

Figure 49 shows the transistor threshold circuit of the modified neuromime at the beginning of the program. Tests run on neuromime networks during the contract period indicated certain circuit changes and additions. The threshold circuit shown in Figure 50 evolved from these changes and additions.

Figure 51 is a block diagram showing the threshold circuit operation. The threshold circuit has two inputs and one output. The two inputs are the sums of weighted and unweighted neuromime pulses. The neuromime's output is a burst of pulses whose repetition rate is roughly a concensus of the neuromime's input activity. A separate circuit accepts the first negative-weighted input in a series of inputs and injects a short pulse into the summation point of the threshold integrator. The neuromime output is fed back with a reversed sign and integrated with the two input summations and the short injected pulse. This integral is then summed with the rest threshold potential and the neuromime's output. The threshold equation is then

$$T = T_o - KR_o - \int_0^t \left[\Sigma R_i + g(t) \quad \Sigma W_i R_i - h(t)\ W_i e^{-nt} - AR_o \right] dt.$$

where

T = threshold

T_o = rest threshold

R_i = i<u>th</u> input to neuromime

W_i = i<u>th</u> weight associated with i<u>th</u> input

R_o = neuromime output

$g(t)$ = 0 when $W_i R_i < 0$
= 1 when $W_i R_i > 0$

$h(t)$ = 0 when all $W_i < 0$
= 1 when at least one W_i, say W_k, < 0

K, A, n = constants

t = time variable.

When the neuromime firing threshold was exceeded, the trigger circuit shown in the diagram triggered the gate, and the neuromime fired a burst of pulses. The threshold circuit determined the length of the burst of pulses, and also how long the neuromime waited before producing another burst of pulses. Without inputs, the threshold circuit returned to its quiescent state.

Tests were run on the threshold circuit to determine its response to inputs, weighted and unweighted. The response curves are shown in Figures 52, 53, 54, and 55.

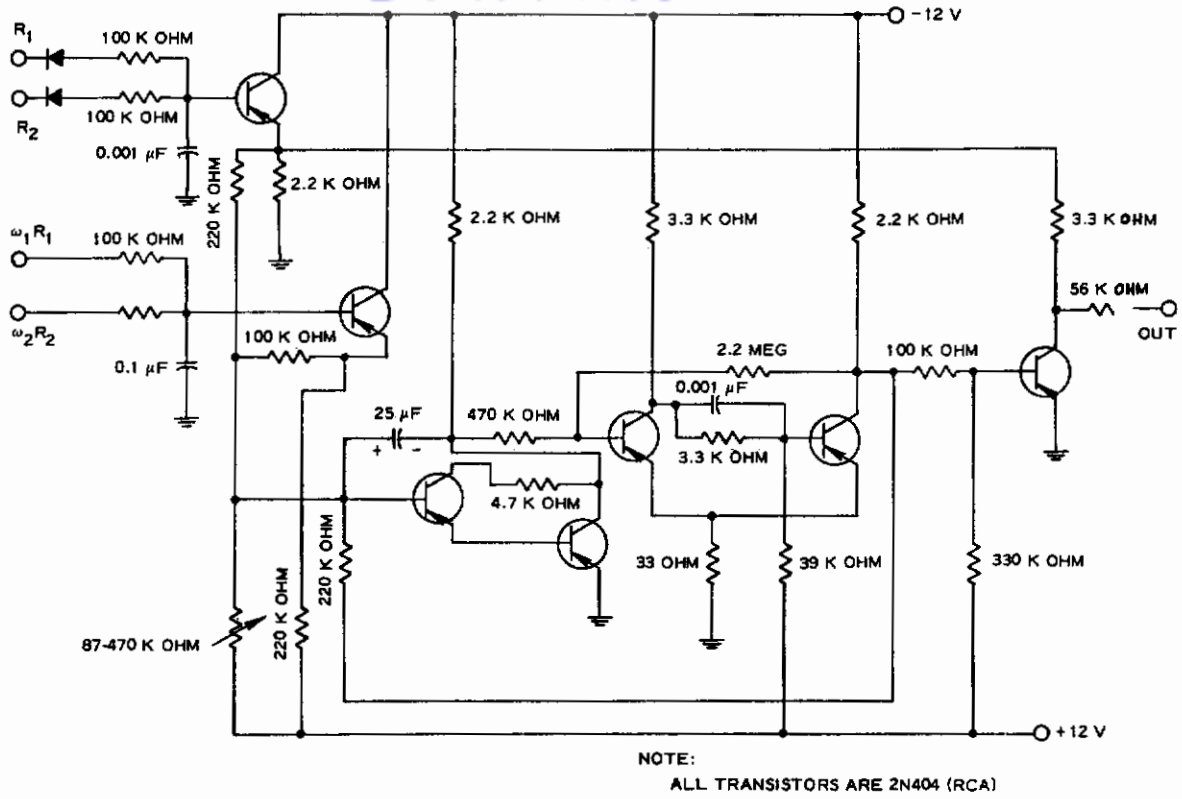


Figure 49. Original Neuromime Transistor Circuit

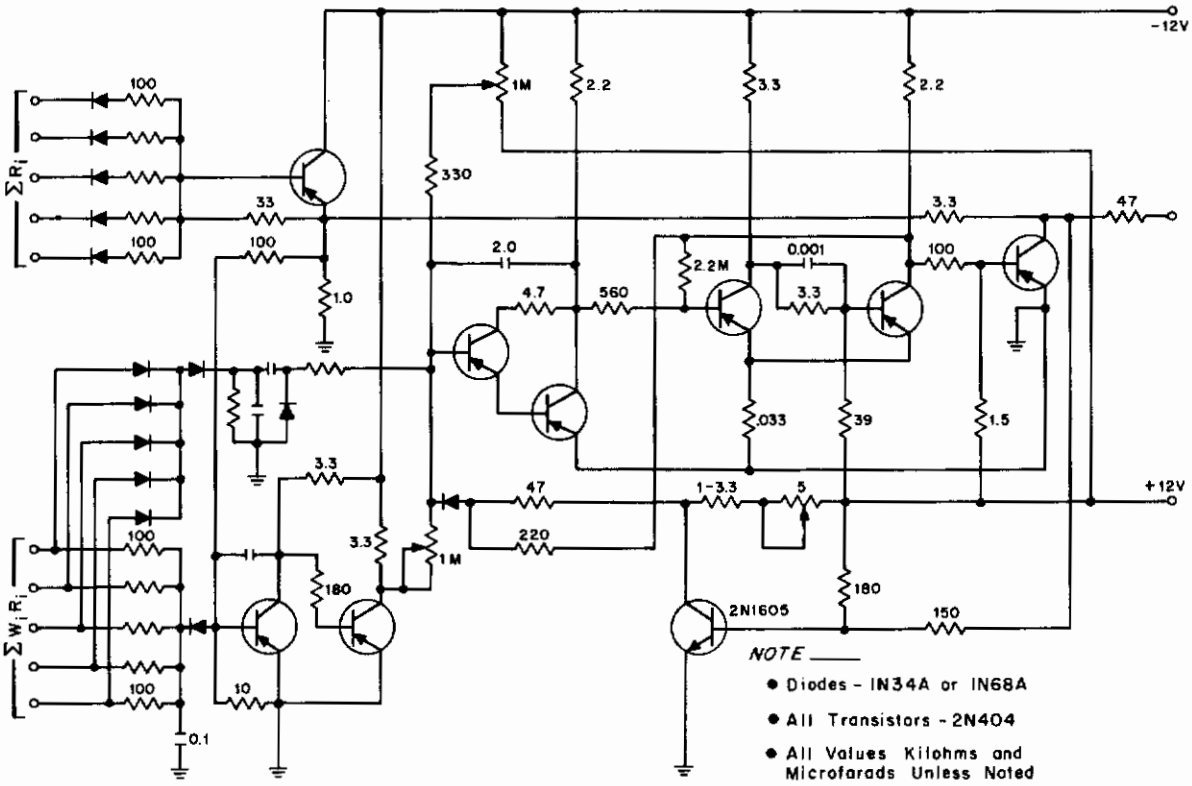


Figure 50. Threshold Circuit

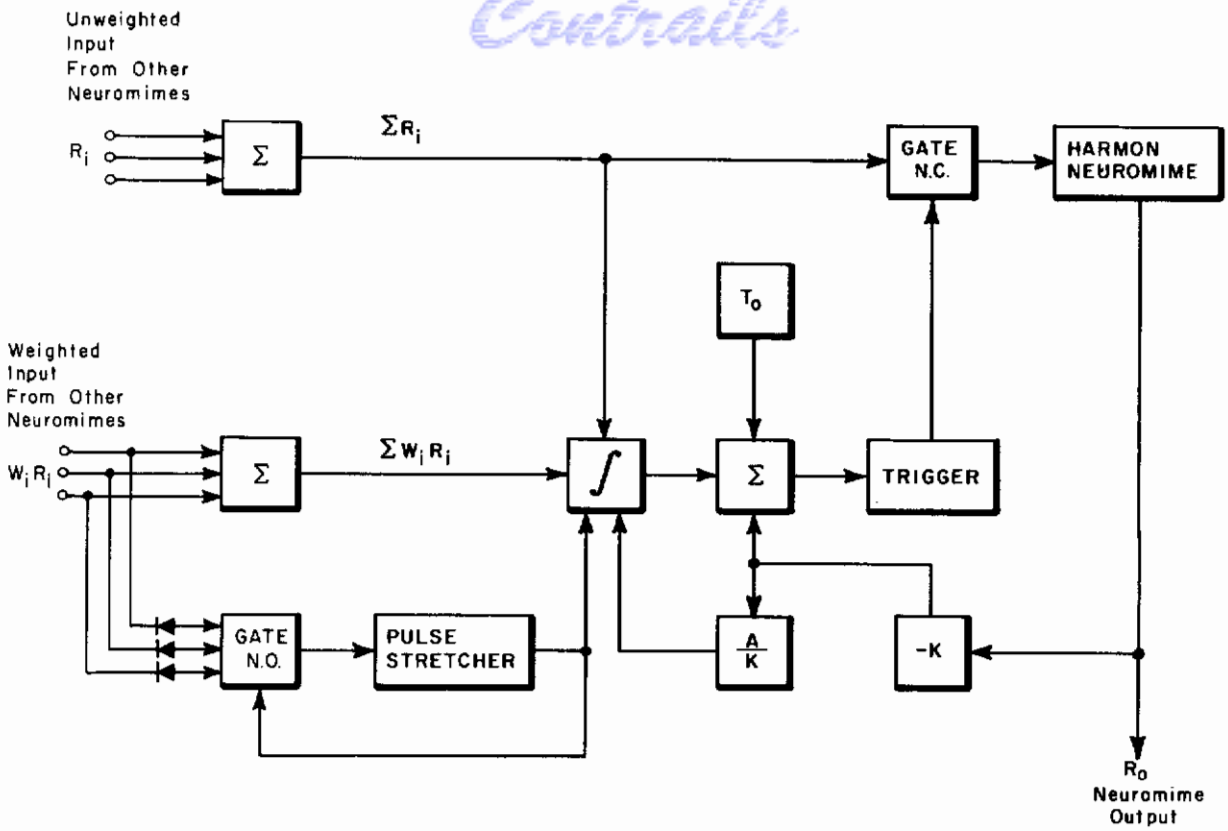


Figure 51. Block Diagram of Threshold Circuit

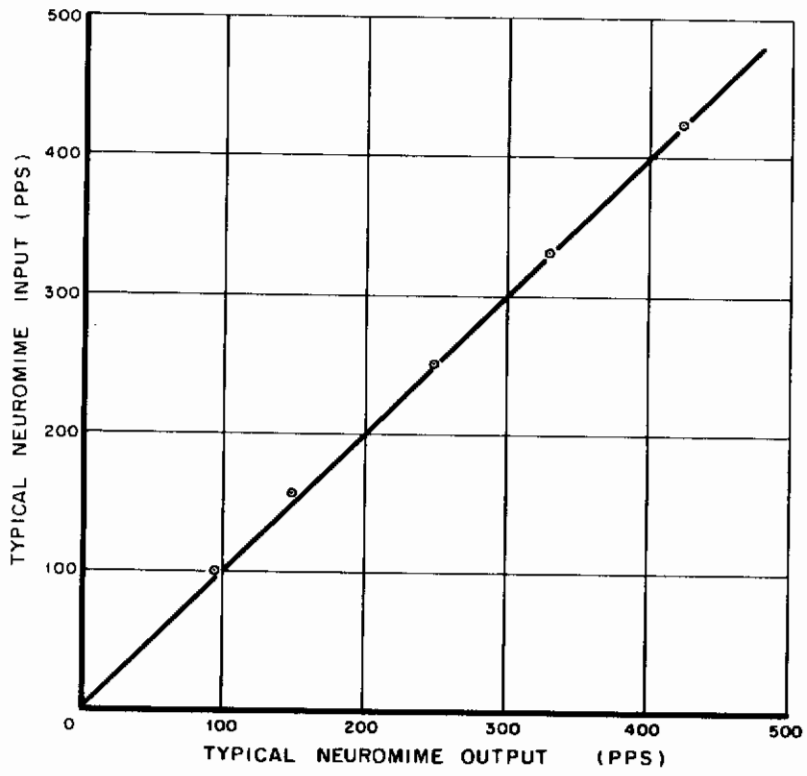


Figure 52. Neuromime Response to Pulses

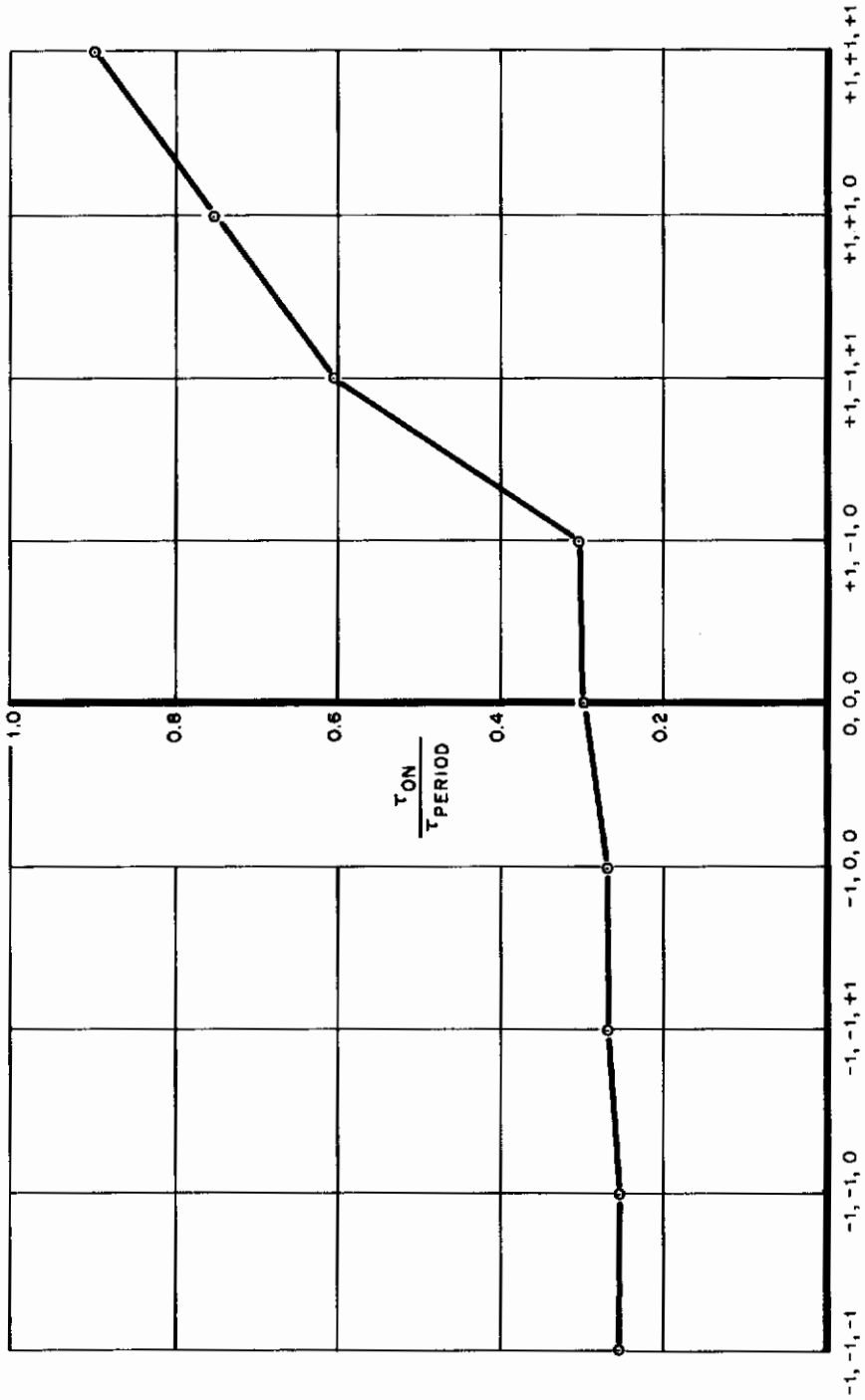


Figure 53. Weighted Inputs

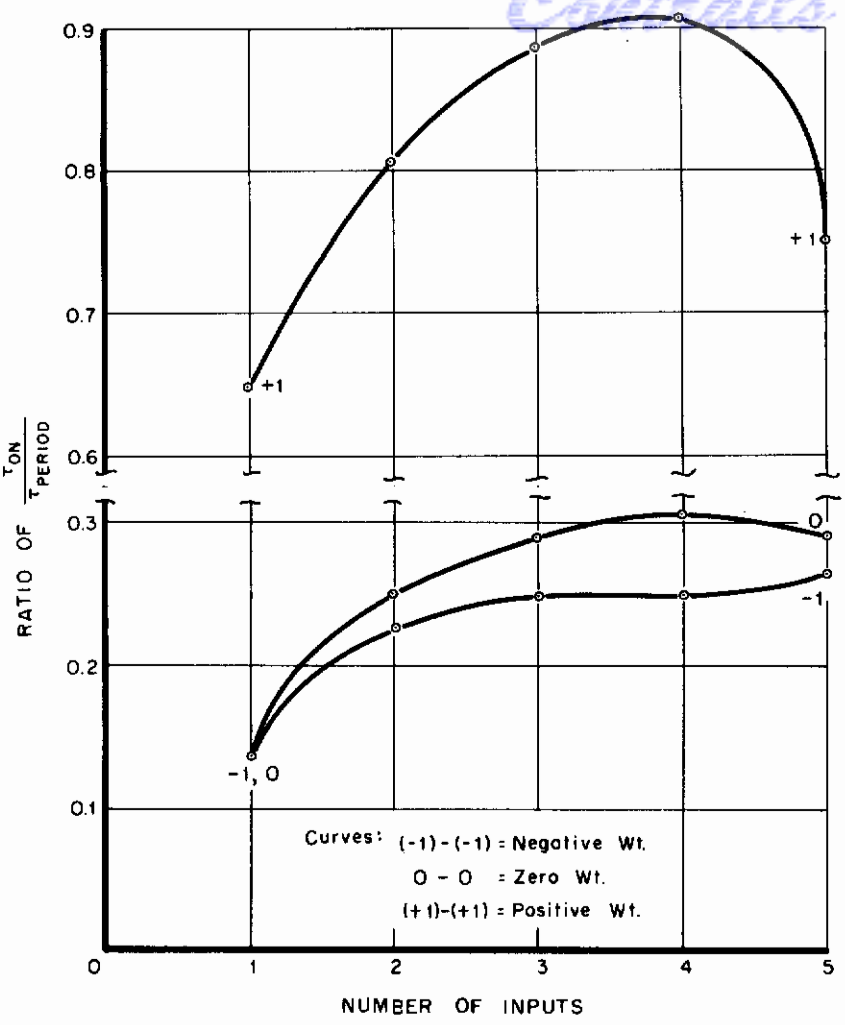


Figure 54. Neuromime Response to Several Inputs

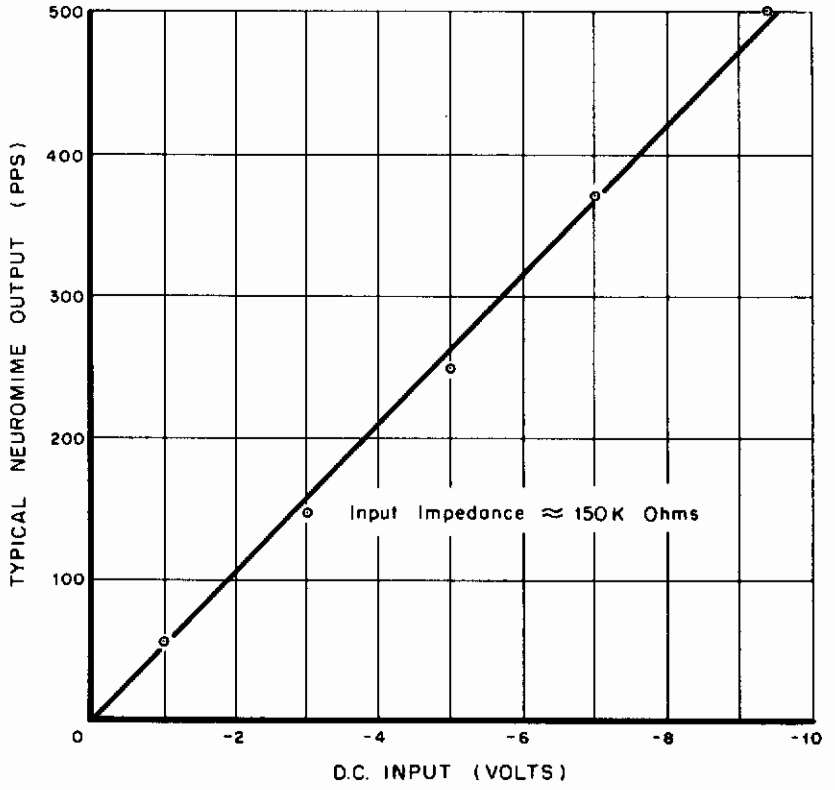


Figure 55. Neuromime Response to Direct Current

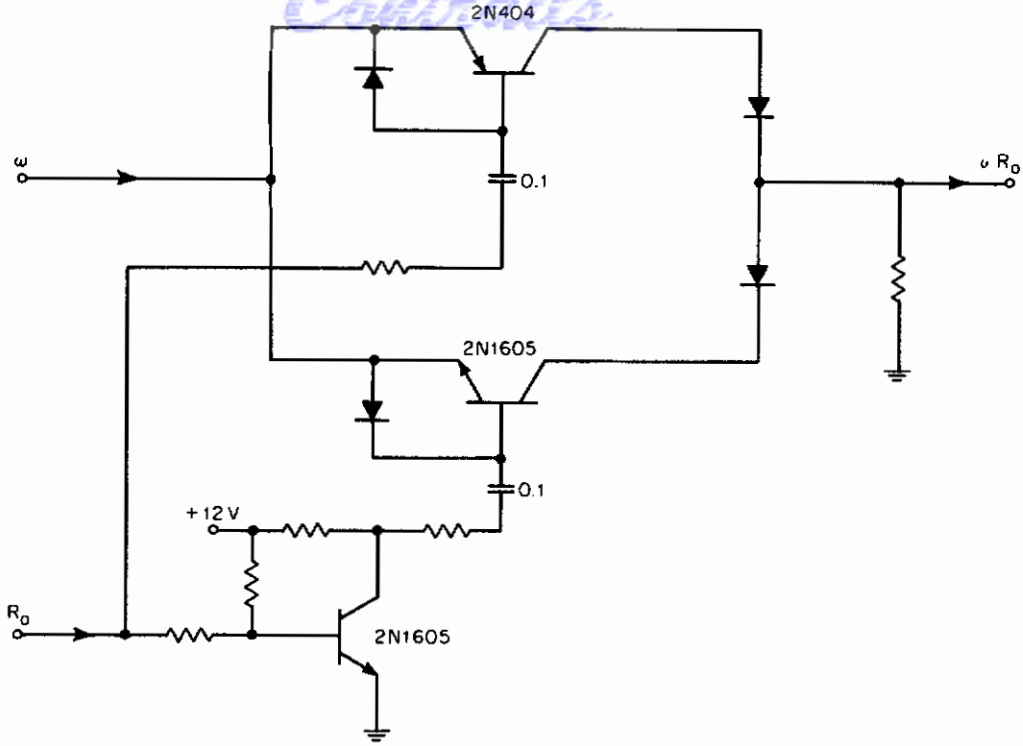


Figure 56. Transistor Gating Circuit

In addition to the circuitry developed for the neuromime model, a transistor circuit was developed as support for the neuromime hardware. This circuit was designed and built to combine a neuromime's output pulse rate and synaptic weight. The circuit turned the synaptic weight (d-c voltage) on and off at the same rate as that of the neuromime. The result was a voltage proportional to the synaptic weight, with a pulse rate equal to that of the neuromime. The circuit is shown in Figure 56.

B. ANALOG CIRCUIT DEVELOPMENT

The analog computer has played two roles in the development of the neuron model. The first role was that of the synaptic weight computer used as support for the transistor hardware. The second role was that of the total neuron model simulation.

Figure 57 shows the analog computer simulation of the synaptic weight computer at the beginning of the program. Tests on the neuromime networks led to several revisions of the analog weight computer circuit. Figure 58 shows the resultant analog computer circuit.

The block diagram of the synaptic weight computer simulation is shown in Figure 59. The weight computer has three inputs and one output. The summation of the neuromime's inputs is the first of the three inputs to the weight computer. It goes through a $\delta(S)$ circuit, which yields, roughly, the time changes of the summation. The changes enter a gate circuit, which is controlled by a gating function, g(S). The gate circuit passes the summation changes unimpeded if it is "open." When closed, no changes pass through the gate circuit. The two remaining inputs to the weight computer, the neuromime's own firing rate and a function, h_i, are used to trigger the gating function.

The gating function, g(S) in the block diagram, controlled the inputs to the weight integrator by operating a gate. As long as the gating function received an input, the gate was

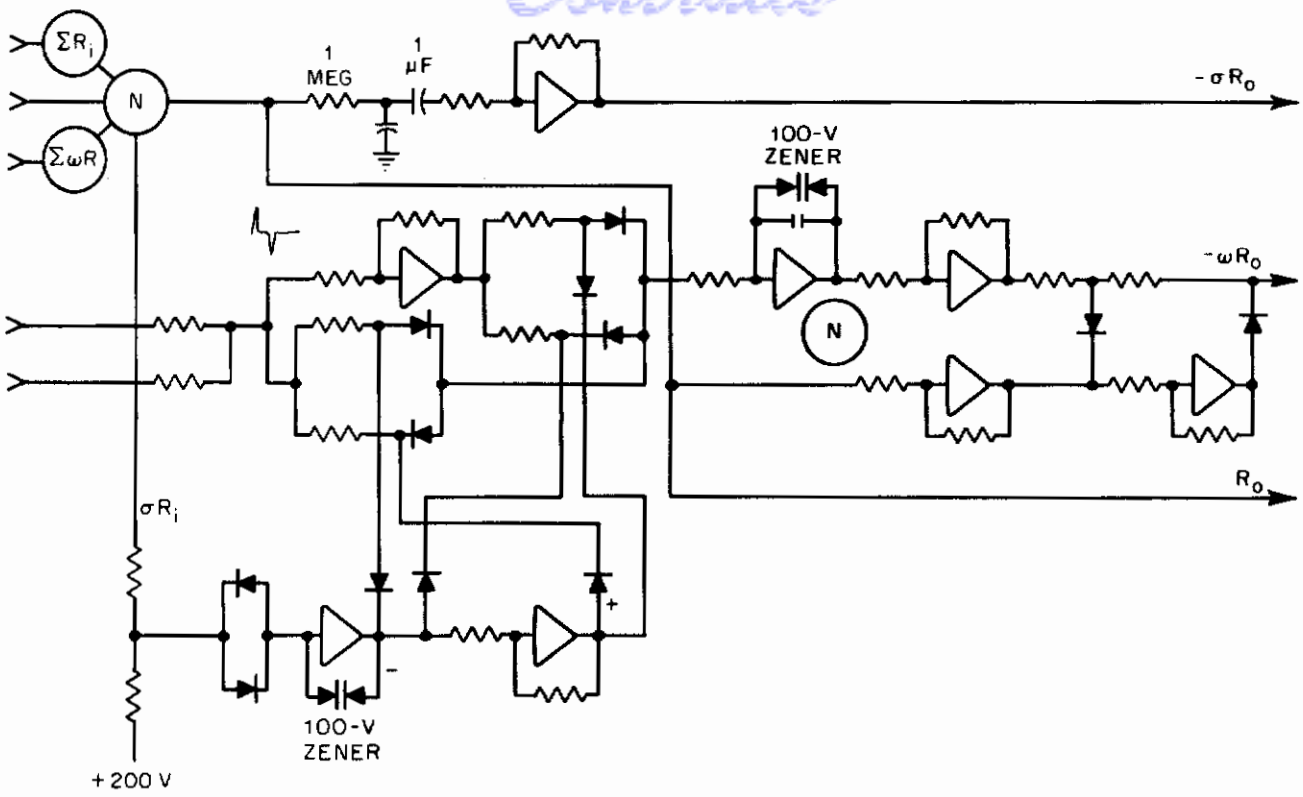


Figure 57. Original Analog Computer Circuit for Weight Computer

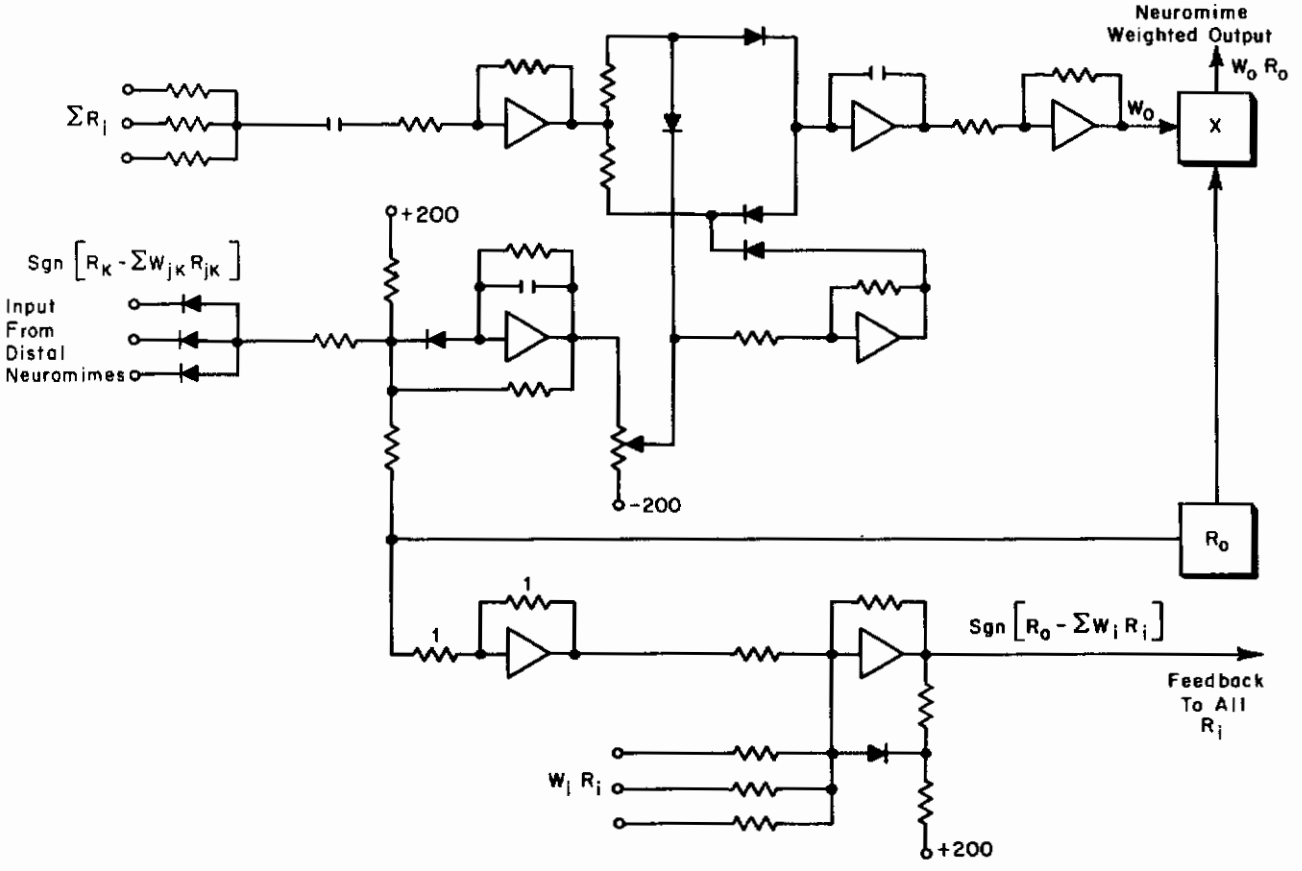


Figure 58. Resultant Synaptic Weight Computer Circuit

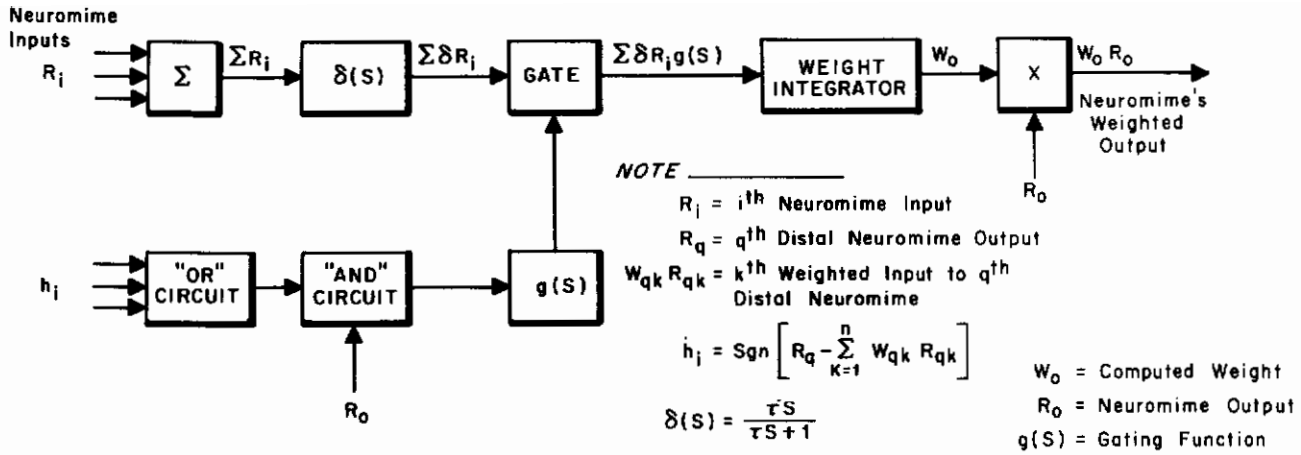


Figure 59. Block Diagram of Synaptic Weight Computer

"open" and the weight integrator received inputs. If the input to the gating function disappeared, the gating function gradually "closed" the gate so that the weight integrator received smaller and smaller inputs until the gate "closed."

The gating function was the most critical circuit of the weight computer simulation, and subject to more changes than any other weight computer circuit. It controlled the times that the weight integrator could compute. Figure 60A shows the weight integrator gate circuit at the beginning of the contract. The neuromime's gating circuit let the weight integrator integrate the inputs during the neuromime's firing period.

When the distal neuromimes stopped firing, the gating circuit changed the algebraic signs of the weight integrator inputs and let the synaptic weight integration continue. The gating circuit was changed as shown on Figure 60B to let the weight integrator work only during the distal neuromimes' firing cycle.

Further neural net tests showed that the gating circuit should allow some weight computation after a neuromime stops firing. The gating circuit was changed to let the weight integrator integrate during a neuromime's firing time and shortly after (see Figure 60C).

Further tests pointed out another fault with the synaptic weight computer. The gating function circuit allowed some neuromimes to compute synaptic weights without ever firing. Figure 60D shows a gating function circuit that corrected this fault. In this circuit, no synaptic weight could be computed unless the neuromime was firing.

Other tests showed that the gating function was adequate when a neuromime had a single input, but was inadequate when the neuromime had more than one input. A neuromime was not getting enough information from its distal neuromimes to discover whether or not it was causing the distal neuromimes to fire. The gating circuit shown in Figure 61 allows a neuromime to compute a synaptic weight only if it is firing, or has recently fired, and at least one of the distal neuromimes is firing against the orders of its weighted inputs.

The second role played by the analog computer was that of the complete simulation of the neuron model. This neuron model development was inspired by the application of optimal control theory to neuromime networks. Two basic models were tested, each using a different method to compute its synaptic weights.

Figure 62 shows the first neuron model using the optimal control concept. In this model, the synaptic weight computer computed the weights by comparing a "predicted" trajectory with the actual trajectory of the input signal. Any difference between the "predicted" trajectory and the actual trajectory was used to correct the synaptic weights.

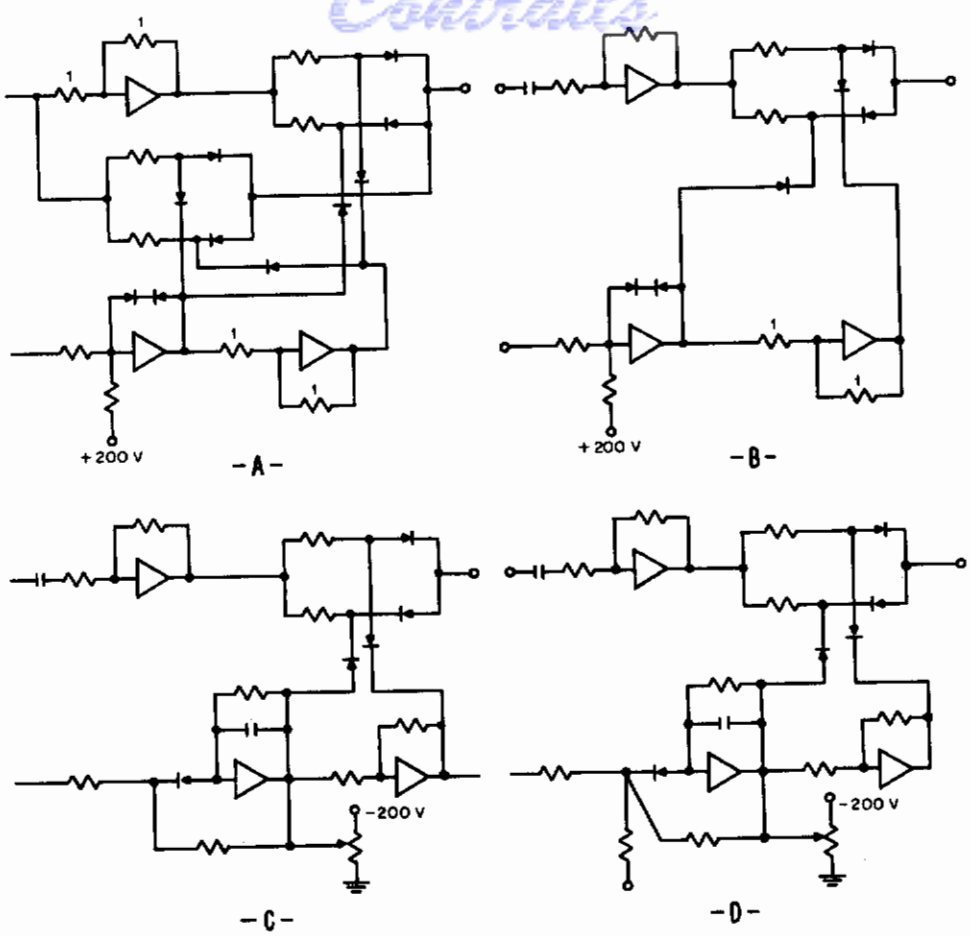


Figure 60. Gating Circuit Development

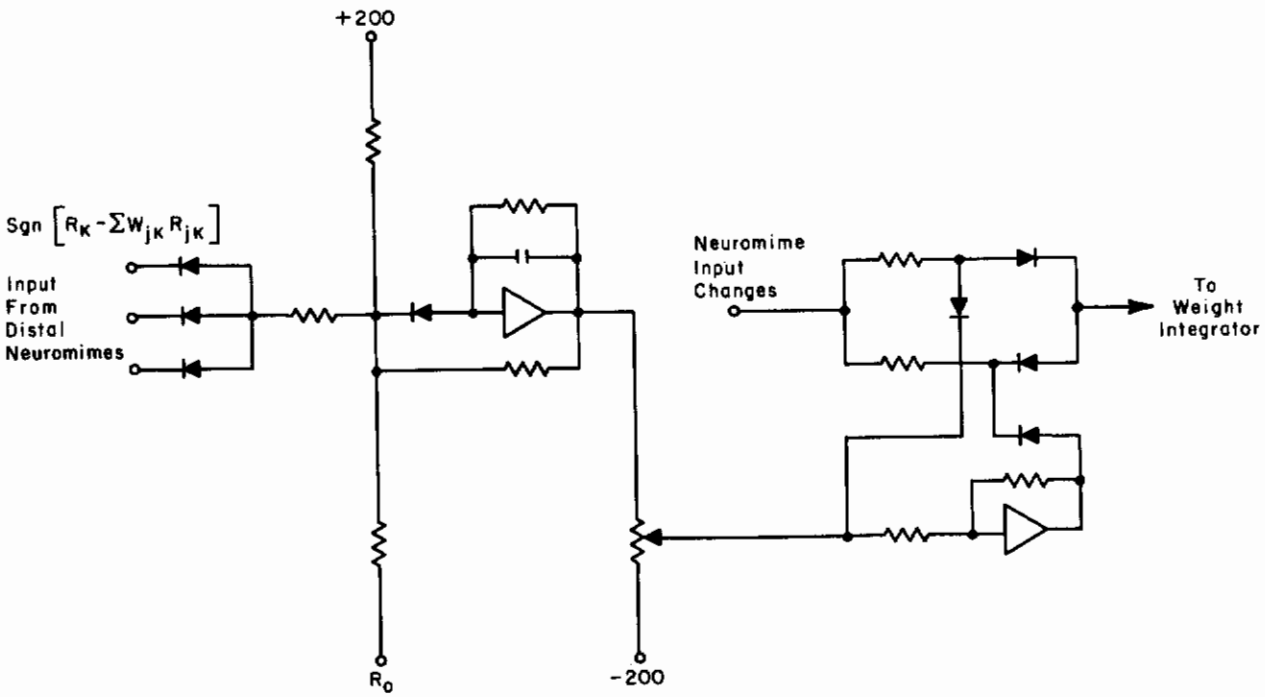


Figure 61. Analog Gating Function Circuit

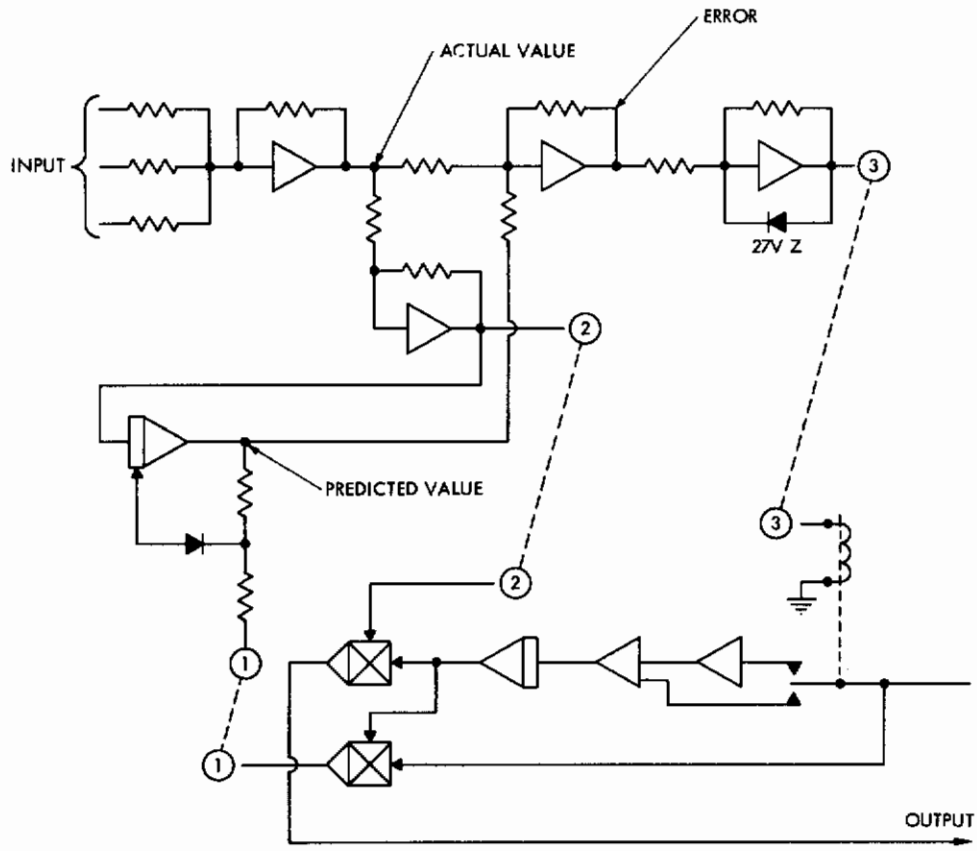


Figure 62. First Neuron Model Using Optimal Control Concept

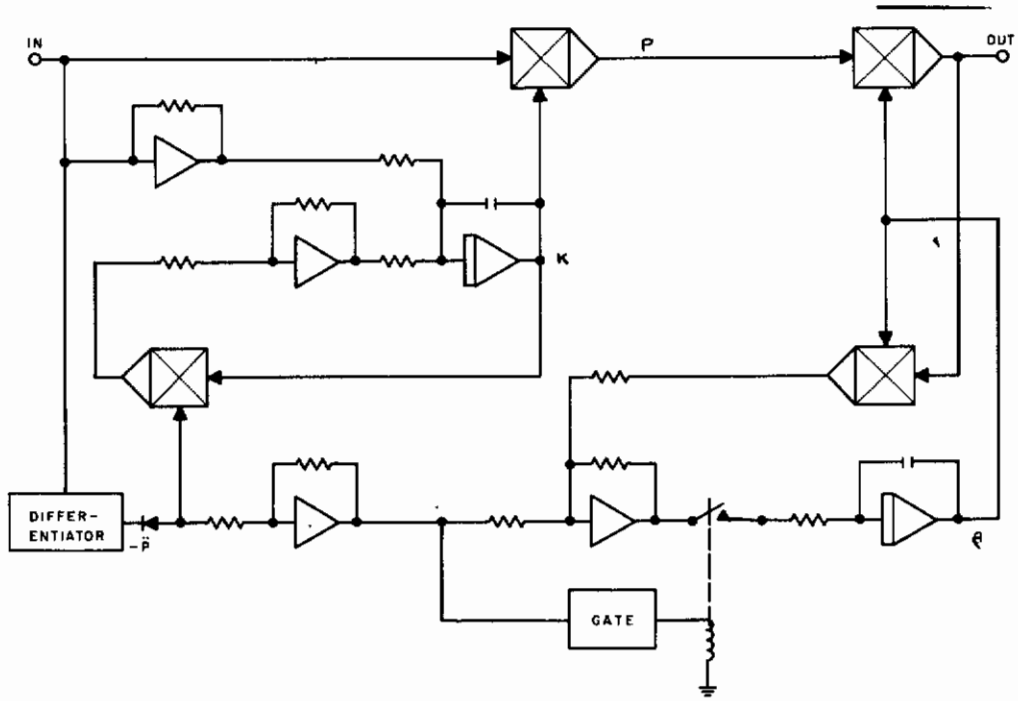


Figure 63. Second Neuron Model Using Optimal Control Concept

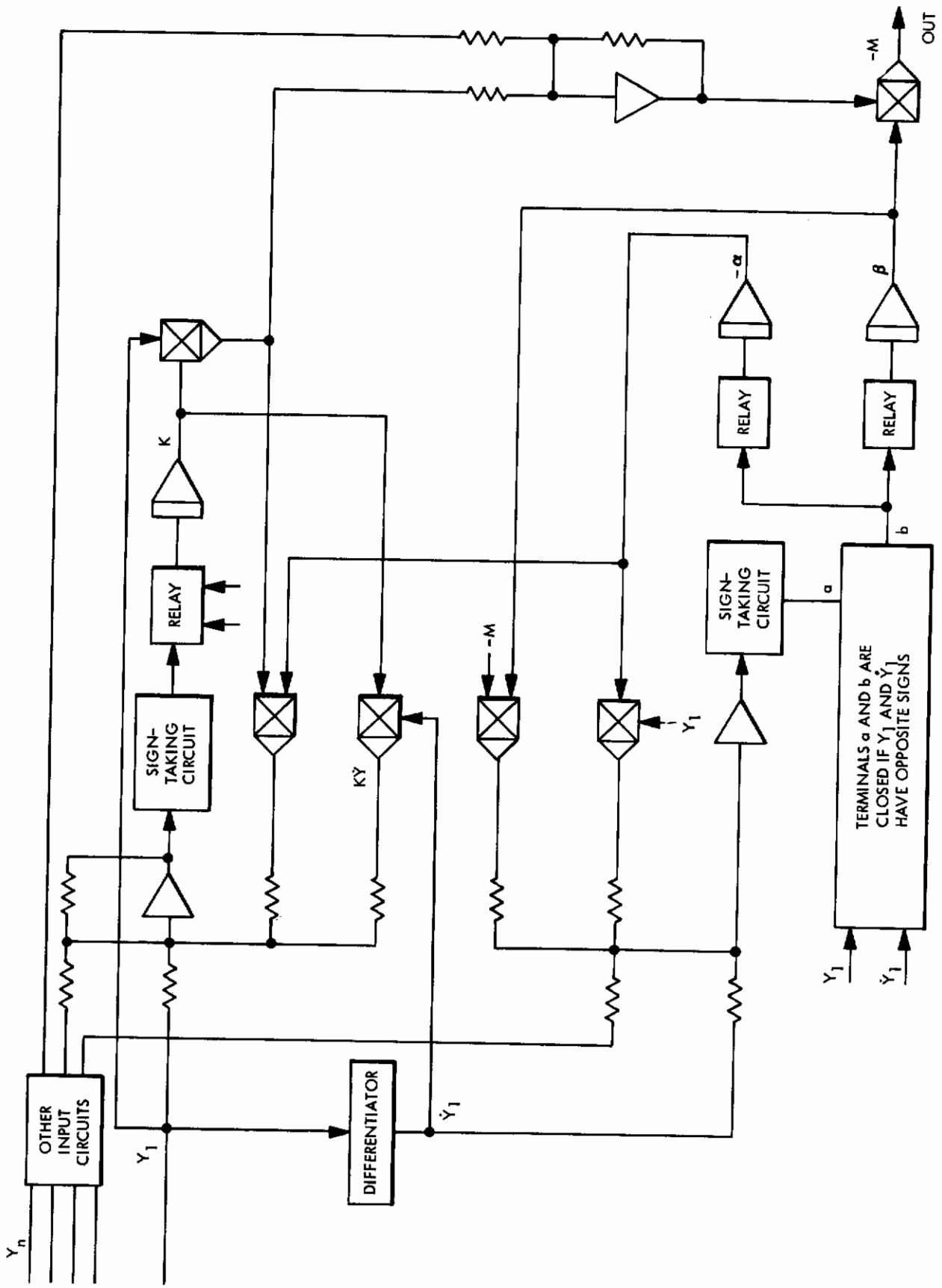


Figure 64. Present Neuron Model

The method used to compute the synaptic weights was based on the optimal control equations given in Appendix I. However, preliminary tests showed that this computing method was too difficult to instrument for neuron models having several inputs.

Figure 63 shows the second basic neuron model using the optimal control concept. The synaptic weight computing method for this model was based on a modification of the optimal control equations. The modification is an assumption. To compute the synaptic weights, the neuron model assumes that the optimal control output is a linear combination of the environment outputs. The neuron model computes the entries for a transformation matrix. This matrix maps the environment output vector into the optimal control vector.

The development of the second basic neuron model was pursued in lieu of the first. The second neuron model showed the ability to compute optimum synaptic weights as well as the ability to accept large numbers of inputs.

Figure 64 shows the results of the neuron model development. Comparison with the neuron model in Figure 63 shows the various changes. Briefly, the significant changes are the addition of a sign-taking circuit and a circuit that indicates when an input variable and its time derivative have opposite signs. The sign-taking circuit enables the neuron model to use a single error signal to compute many parameters. The technique for computing the synaptic weights using the sign-taking circuit is a form of hill-climbing, and is also known as the gradient method of steepest descent. The second additional circuit evolved from certain test results which revealed that the optimal control equations developed were not valid when an environmental disturbance was present. To obviate this invalidity, it was postulated that a disturbance was present when an input variable and its time derivative had the same sign. (This postulate is not true for all cases, but the approximation is good for transient conditions.) This circuit, then, was designed to stop synaptic weight computations during disturbances. Tests on the neuron model using these additional circuits have proved their usefulness.

APPENDIX III

PREDICTION THEORY

The usual approach to the problem of prediction of stationary time series is the Wiener approach. One assumes a weakly stationary random process, x(t) such that the set of possible inputs to the predictor are elements of the process. One then tries to find a weighting function w(t) such that

$$x^*(t) = \int_{-\infty}^{t} w(\tau, \sigma)x(t-\tau)d\tau$$

is the best least squares prediction of $x(t+\sigma)$; i. e.,

$E[(x^*(t) - x(t+\sigma)]^2$ is minimized.

An alternate approach to the prediction problem was formulated by Kolmogorov almost simultaneously with Wiener's work. The Kolmogorov approach, in contrast to Wiener's, does not assume that the entire past history of the signal is available. When expressed in sampled form, the Kolmogorov formulation seems more suitable for application to the neuromime networks under study. Following is a discussion of this application of the Kolmogorov theory.

Suppose a function f(x), where both x and f(x) are real variables, is given in terms of equally spaced samples,

$$f(-n), \ldots f(-2), f(-1), f(0), \tag{45}$$

and its next value f(1) is to be predicted. Further, suppose f(x) is weakly stationary, and therefore

(1) It has a finite second moment,

$$E\left\{[f(x)]^2\right\} < \infty, \tag{46}$$

where E stands for "expected value."

(2) The joint probability of any two of its values is a function of their separation only,

$$E[f(x)\, f(x-\tau)] = \phi_{ff}(\tau). \tag{47}$$

(3) It has a continuous covariance function, $\phi_{ff}(\tau)$.

Assuming the prediction should be based only on the known samples of Equation 45, the simplest process of finding f(1) will be to consider it a linear combination of these samples:

$$f(1) = a_0 f(0) + a_1 f(-1) + \ldots + a_n f(-n). \tag{48}$$

The problem is to find the appropriate values of the weights a_i.

Multiplying Equation 48 successively by the right-hand members of Equation 45, we get the following set of equations:

$$\left.\begin{aligned} f(0)f(1) &= a_0 f(0)f(0) + a_1 f(0)f(-1) + \ldots + a_n f(0)f(-n) \\ f(-1)f(1) &= a_0 f(-1)f(0) + a_1 f(-1)f(-1) + \ldots + a_n f(-1)f(-n) \\ &\vdots \\ f(-1)f(1) &= a_0 f(-n)f(0) + a_1 f(-n)f(-1) + \ldots + a_n f(-n)f(-n) \end{aligned}\right\} \quad (49)$$

The expected values of the left-hand members of each equation in Equation Set 49 will be the following set:

$$\left.\begin{aligned} E[f(0)f(1)] &= a_0 E[f(0)f(0)] + a_1 E[f(0)f(-1)] + \ldots + a_n E[f(0)f(-n)] \\ E[f(-1)f(1)] &= a_0 E[f(-1)f(0)] + a_1 E[f(-1)f(-1)] + \ldots + a_n E[f(-1)f(-n)] \\ &\vdots \\ E[f(-n)f(1)] &= a_0 E[f(-n)f(0)] + a_1 E[f(-n)f(-1)] + \ldots + a_n E[f(-n)f(-n)] \end{aligned}\right\} \quad (50)$$

However, due to the weak stationarity conditions given by Equations 46 and 47, the expected value $E[f(i)f(i \pm \xi)]$, exists and has the value

$$E[f(i)f(i \pm \xi)] = E[f(x)f(x \pm \xi)] \quad (51)$$

Then, by making i successively equal to 0, -1, . . ., -n, and ξ equal to 1, 0, -1, . . ., -n, we get from Equation Set 50

$$\left.\begin{aligned} E[f(x)f(x+1)] &= a_0 E[f(x)f(x)] + a_1 E[f(x)f(x-1)] + \ldots \\ &\qquad + a_n E[f(x)f(x-n)] \\ E[f(x)f(x+2)] &= a_0 E[f(x)f(x+1)] + a_1 E[f(x)f(x)] + \ldots \\ &\qquad + a_n E\left\{f[x]\, f[x-(n-1)]\right\} \\ &\vdots \\ E\left\{f[x]\, f[x+(n+1)]\right\} &= a_0 E[f(x)f(x+n)] + a_1 E\left\{f[x]\, f[x+(n-1)]\right\} \\ &\qquad + \ldots + a_n E[f(x)f(x)] \end{aligned}\right\} \quad (52)$$

By the condition of Equation 47,

$$E[f(x)f(x \pm \xi)] = \phi_{ff}(\pm \xi) \quad (53)$$

where ϕ_{ff} is the autocorrelation function of f(x). Consequently,

$$\left.\begin{aligned}
\phi_{ff}(-1) &= a_0\phi_{ff}(0) + a_1\phi_{ff}(1) + \ldots + a_n\phi_{ff}(n) \\
\phi_{ff}(-2) &= a_0\phi_{ff}(-1) + a_1\phi_{ff}(0) + \ldots + a_n\phi_{ff}(n-1) \\
&\vdots \\
\phi_{ff}\,[-(n+1)] &= a_0\phi_{ff}(-n) + a_1\phi_{ff}\,[-(n-1)] + \ldots + a_n\phi_{ff}(0)
\end{aligned}\right\} \tag{54}$$

Since f(x) was assumed to be a real variable, its autocorrelation is an even function, i. e.

$$\phi_{ff}(-\xi) = \phi_{ff}(\xi). \tag{55}$$

Therefore,

$$\left.\begin{aligned}
\phi_{ff}(1) &= a_0\phi_{ff}(0) + a_1\phi_{ff}(1) + \ldots + a_n\phi_{ff}(n) \\
\phi_{ff}(2) &= a_0\phi_{ff}(1) + a_1\phi_{ff}(0) + \ldots + a_n\phi_{ff}(n-1) \\
&\vdots \\
\phi_{ff}(n+1) &= a_0\phi_{ff}(n) + a_1\phi_{ff}(n-1) + \ldots + a_n\phi_{ff}(0).
\end{aligned}\right\} \tag{56}$$

This is a system of (n + 1) equations in the (n + 1) unknowns $a_0, a_1, \ldots a_n$, and can be put in a matrix form suitable for a computer solution:

$$\begin{bmatrix}
\phi_{ff}(0) & \phi_{ff}(1) & \phi_{ff}(2) & \ldots & \phi_{ff}(n) \\
\phi_{ff}(1) & \phi_{ff}(0) & \phi_{ff}(1) & \ldots & \phi_{ff}(n-1) \\
\vdots & & & & \\
\phi_{ff}(n) & \phi_{ff}(n-1) & \phi_{ff}(n-2) & \ldots & \phi_{ff}(0)
\end{bmatrix}
\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}
=
\begin{bmatrix} \phi_{ff}(1) \\ \phi_{ff}(2) \\ \vdots \\ \phi_{ff}(n+1) \end{bmatrix} \tag{57}$$

It can be shown that the values of the weights a_i found from Equation Set 57 minimize the square error between the left- and right-hand members of each equation in Equation Set 50.

If instead of the sample f(1) a more distant sample f(k) is desired, then replace f(1) by f(k) in Equation 48. It is easy to see that in this case Equation Set 57 becomes

$$\begin{bmatrix}
\phi_{ff}(0) & \phi_{ff}(1) & \phi_{ff}(2) & \ldots & \phi_{ff}(n) \\
\phi_{ff}(1) & \phi_{ff}(0) & \phi_{ff}(1) & \ldots & \phi_{ff}(n-1) \\
\vdots & & & & \\
\phi_{ff}(n) & \phi_{ff}(n-1) & \phi_{ff}(n-2) & \ldots & \phi_{ff}(0)
\end{bmatrix}
\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}
=
\begin{bmatrix} \phi_{ff}(k) \\ \phi_{ff}(k+1) \\ \vdots \\ \phi_{ff}(k+n) \end{bmatrix}$$

$$k \geqq 1. \tag{58}$$

In a similar fashion, a missing sample f(0) from a set of equally spaced samples of the function f(x),

$$f(-n),\ f[-(n-1)],\ \ldots\ f(-1),\ f(0),\ f(1),\ \ldots,\ f(m-1),\ f(m), \tag{59}$$

can be reconstituted.

Similar to the previous analysis, we express f(0) as

$$f(0) = a_{-n}f(-n) + \ldots + a_{-1}f(-1) + a_1f(1) + \ldots + a_mf(m). \tag{60}$$

After multiplying Equation 60 successively by the members of Equation 59, the expected values will be

$$\left.\begin{aligned}
E[f(-n)f(0)] &= a_{-n}E[f(-n)f(-n)] + \ldots + a_{-1}E[f(-n)f(-1)] \\
&\quad + a_1E[f(-n)f(1)] + \ldots + a_mE[f(-n)f(m)] \\
&\vdots \\
E[f(-1)f(0)] &= a_{-n}E[f(-1)f(-n)] + \ldots + a_{-1}E[f(-1)f(-1)] \\
&\quad + a_1E[f(-1)f(1)] + \ldots + a_mE[f(-1)f(m)] \\
E[f(1)f(0)] &= a_{-n}E[f(1)f(-n)] + \ldots + a_{-1}E[f(1)f(-1)] \\
&\quad + a_1E[f(1)f(1)] + \ldots + a_mE[f(1)f(m)] \\
&\vdots \\
E[f(m)f(0)] &= a_{-n}E[f(m)f(-n)] + \ldots + a_{-1}E[f(m)f(-1)] \\
&\quad + a_1E[f(m)f(1)] + \ldots + a_mE[f(m)f(m)]
\end{aligned}\right\} \tag{61}$$

Using the conditions of Equations 47, 51, and 55 as before, Equation Set 61 becomes the set

$$\left.\begin{aligned}
\phi_{ff}(n) &= a_{-n}\phi_{ff}(0) + a_{-(n-1)}\phi_{ff}(1) + \ldots + a_{-1}\phi_{ff}(n-1) \\
&\quad + a_1\phi_{ff}(n+1) + \ldots + a_{m-1}\phi_{ff}(n+m-1) + a_m\phi_{ff}(n+m) \\
&\vdots \\
\phi_{ff}(1) &= a_{-n}\phi_{ff}(n-1) + a_{-(n-1)}\phi_{ff}(n-2) + \ldots + a_{-1}\phi_{ff}(0) \\
&\quad + a_1\phi_{ff}(2) + \ldots + a_{m-1}\phi_{ff}(m) + a_m\phi_{ff}(m+1) \\
\phi_{ff}(1) &= a_{-n}\phi_{ff}(n+1) + a_{-(n-1)}\phi_{ff}(n) + \ldots + a_{-1}\phi_{ff}(2) \\
&\quad + a_1\phi_{ff}(0) + \ldots + a_{m-1}\phi_{ff}(m-2) + a_m\phi_{ff}(m-1) \\
&\vdots \\
\phi_{ff}(m) &= a_{-n}\phi_{ff}(n+m) + a_{-(n-1)}\phi_{ff}(n+m-1) + \ldots + a_{-1}\phi_{ff}(m+1) \\
&\quad + a_1\phi_{ff}(m-1) + \ldots + a_{m-1}\phi_{ff}(1) + a_m\phi_{ff}(0)
\end{aligned}\right\} \tag{62}$$

In matrix form, Equation Set 62 is

$$[A] = \begin{bmatrix} \phi_{ff}(0) & \phi_{ff}(1) \ldots \phi_{ff}(n-1) & \phi_{ff}(n+1) \ldots \phi_{ff}(m+n-1) & \phi_{ff}(m+n) \\ \vdots & & & \\ \phi_{ff}(n-1) & \phi_{ff}(n-2) \ldots \phi_{ff}(0) & \phi_{ff}(2) \ldots \phi_{ff}(m) & \phi_{ff}(m+1) \\ \phi_{ff}(n+1) & \phi_{ff}(n) \ldots \phi_{ff}(2) & \phi_{ff}(0) \ldots \phi_{ff}(m-2) & \phi_{ff}(m-1) \\ \vdots & & & \\ \phi_{ff}(n+m) & \phi_{ff}(n+m-1) \ldots \phi_{ff}(m+1) & \phi_{ff}(m-1) \ldots \phi_{ff}(1) & \phi_{ff}(0) \end{bmatrix}$$

therefore

$$[A] \begin{bmatrix} a_{-n} \\ \vdots \\ a_{-1} \\ a_1 \\ \vdots \\ a_m \end{bmatrix} = \begin{bmatrix} \phi_{ff}(n) \\ \vdots \\ \phi_{ff}(1) \\ \phi_{ff}(1) \\ \\ \phi_{ff}(m) \end{bmatrix} \tag{63}$$

Finally, suppose a second function g(x) is known to be the result of the function f(x) being modified by a linear operator whose response is to be determined. This is the case of plant identification. In the general case, the present state of the response depends on all past values of the input function f(x). We may write

$$g(0) = a_0 f(0) + a_1 f(-1) + \ldots + a_n f(-n) \tag{64}$$

Successive multiplication of the terms of Equation 64 by the terms of Equation 45 yields

$$\left.\begin{aligned} E[f(0)g(0)] &= a_0 E[f(0)f(0)] + a_1 E[f(0)f(-1)] + \ldots \\ &\qquad + a_n E[f(0)f(-n)] \\ E[f(-1)g(0)] &= a_0 E[f(-1)f(0)] + a_1 E[f(-1)f(-1)] + \ldots \\ &\qquad + a_n E[f(-1)f(-n)] \\ &\vdots \\ E[f(-n)g(0)] &= a_0 E[f(-n)f(0)] + a_1 E[f(-n)f(-1)] + \ldots \\ &\qquad + a_n E[f(-n)f(-n)] \end{aligned}\right\} \tag{65}$$

Therefore,

$$\left.\begin{aligned}
\phi_{fg}(0) &= a_0\phi_{ff}(0) + a_1\phi_{ff}(1) + \ldots + a_n\phi_{ff}(n) \\
\phi_{fg}(-1) &= a_0\phi_{ff}(-1) + a_1\phi_{ff}(0) + \ldots + a_n\phi_{ff}(n-1) \\
&\vdots \\
\phi_{fg}(-n) &= a_0\phi_{ff}(-n) + a_1\phi_{ff}[-(n-1)] + \ldots + a_n\phi_{ff}(0),
\end{aligned}\right\} \quad (66)$$

where

$$\left.\begin{aligned}
\phi_{fg}(\xi) &= E\,[f(x)g(x-\xi)] \\
\phi_{ff}(\xi) &= E\,[f(x)f(x-\xi)]
\end{aligned}\right\} \quad (67)$$

Here again, because of Equation 55,

$$\left.\begin{aligned}
\phi_{fg}(0) &= a_0\phi_{ff}(0) + a_1\phi_{ff}(1) + \ldots + a_n\phi_{ff}(n) \\
\phi_{fg}(-1) &= a_0\phi_{ff}(1) + a_1\phi_{ff}(0) + \ldots + a_n\phi_{ff}(n-1) \\
&\vdots \\
\phi_{fg}(-n) &= a_0\phi_{ff}(n) + a_1\phi_{ff}(n-1) + \ldots + a_n\phi_{ff}(0);
\end{aligned}\right\} \quad (68)$$

and in matrix form,

$$\begin{bmatrix}
\phi_{ff}(0) & \phi_{ff}(1) & \phi_{ff}(2) & \ldots & \phi_{ff}(n) \\
\phi_{ff}(1) & \phi_{ff}(0) & \phi_{ff}(1) & \ldots & \phi_{ff}(n-1) \\
\vdots & & & & \\
\phi_{ff}(n) & \phi_{ff}(n-1) & \phi_{ff}(n-2) & \ldots & \phi_{ff}(0)
\end{bmatrix}
\begin{bmatrix} a_0 \\ a_1 \\ \vdots \\ a_n \end{bmatrix}
=
\begin{bmatrix} \phi_{fg}(0) \\ \phi_{fg}(-1) \\ \\ \\ \phi_{fg}(-n) \end{bmatrix} \quad (69)$$

The negative sign in the argument of the cross-correlation function ϕ_{fg} means that the physical significance of the weights a_i may be deduced from the output $g(x)$ of a linear device being caused by an input $f(x)$;

$$g(x) = \int_0^x f(x-\tau)w(\tau)d\tau \quad (70)$$

where $w(\tau)$ is the unit input response or weighting function of the device.

For the case of a discrete, instead of continuous, system where f(x), and therefore g(x), is given in terms of equally spaced samples, Equation 70 has the counterpart

$$g(x) = \sum w(\tau) f(x - \tau)$$

$$\tau = 0, 1, 2, \ldots . \qquad (71)$$

hence

$$g(x) = w(0)f(x) + w(1)f(x - 1) + \ldots + w(n)f(x - n) . \qquad (72)$$

Since this equation is true for any value of $x \geq 0$, it will hold for $x = 0$, i.e.,

$$g(0) = w(0)f(0) + w(1)f(-1) + \ldots + w(n)f(-n). \qquad (73)$$

It can be seen from Equations 64 and 73 that

$$a(i) = w(i)$$

$$i = 0, 1, 2, \ldots . \qquad (74)$$

Therefore, a_i is the value of the system input response at the point $x = i$.

APPENDIX IV

ANALYSIS OF A PARTICULAR SEQUENTIAL MACHINE

None of the rules for "learning" so far investigated will serve to drive the sequential machine shown in Figure 65 to the zero state. Yet a suitable set of logic functions is

$$a = \alpha_1 + \alpha_3,\ b = \alpha_2 + \alpha_3.$$

Applying this rule for "learning" yields the following result. The matrix A for the particular example given is

$$A = \begin{bmatrix} x_1 & a & 0 & b & 0 \\ b & x_2+e & b & 0 & a \\ e & 0 & x_3 & 0 & a \\ a & 0 & a & e+x_4 & b \\ 0 & b & e & a & e+x_5 \end{bmatrix}$$

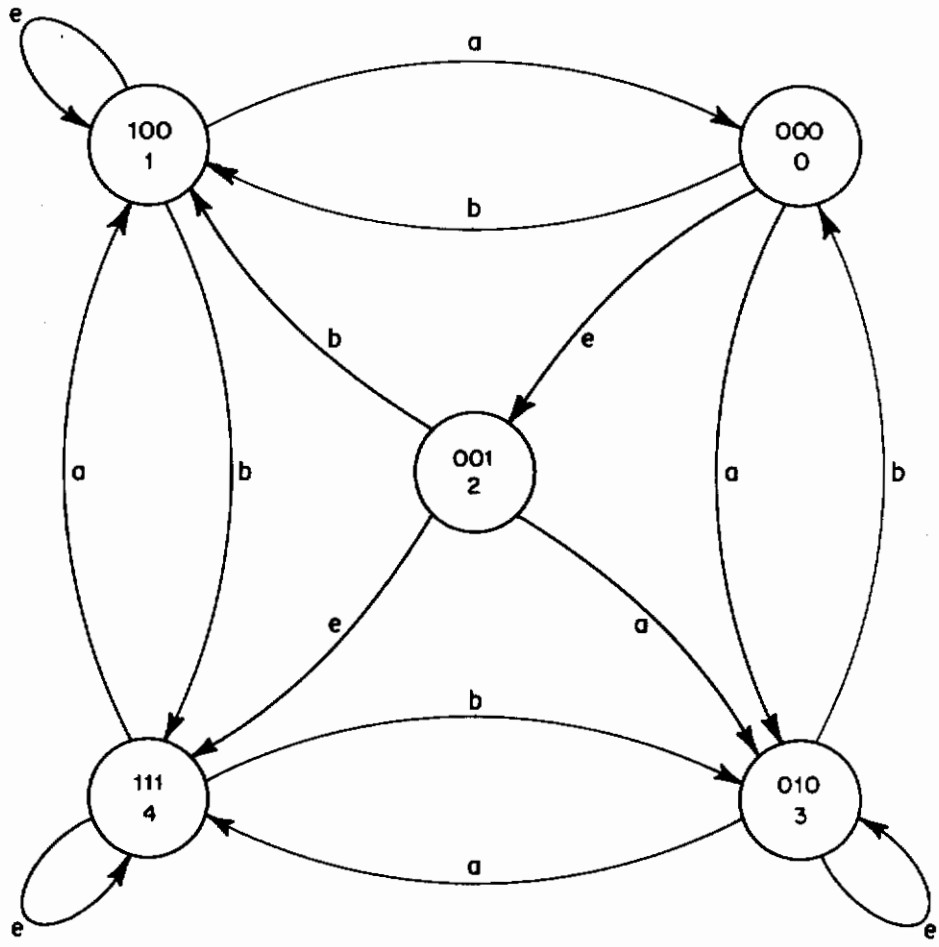


Figure 65. Sequential Machine

Assuming that the probabilities of each of the transitions between states on the figure are 0.1, we obtain that

$$C_m = \begin{bmatrix} 0.7 & 0.1 & 0 & 0.0 & 0 \\ 0.1 & 0.8 & 0.1 & 0 & 0.1 \\ 0.1 & 0 & 0.7 & 0.1 & 0 \\ 0.1 & 0 & 0.1 & 0.8 & 0.1 \\ 0 & 0.1 & 0.1 & 0.1 & 0.8 \end{bmatrix}$$

(The values of the x's are chosen to make each column add to 1, since the sum of the probabilities of all transitions out of a particular state must be one minus the probability of remaining in that state.) The eigenvector is computed to be

$$S_o = \begin{bmatrix} 1/6 \\ 1/4 \\ 1/18 \\ 1/4 \\ 5/18 \end{bmatrix} \cong \begin{bmatrix} 0.19 \\ 0.25 \\ 0.05 \\ 0.25 \\ 0.28 \end{bmatrix}$$

If the signals from the machine are labeled α, β, γ respectively, one obtains that the correlations seen between transitions and changes in the input are proportional to

$\phi_3 =$

	α	β	γ
a	1	0	1
b	0	-1	0

The correlations in each of the states are as follows:

$\phi_0 =$

	α	β	γ
a	0	1	0
b	1	0	0

$\phi_1 =$

a	-1	0	0
b	0	1	1

$\phi_2 =$

a	0	1	-1
b	1	0	-1

$\phi_3 =$

a	1	0	1
b	0	-1	0

$\phi_4 =$

	0	-1	-1
	-1	0	-1

Routine arithmetic yields

$$\phi(\alpha, \beta) = \sum_{i=0}^{4} P_i \phi_i = \begin{vmatrix} 0 & -1/18 & -1/12 \\ -1/18 & 0 & 1/12 \end{vmatrix}$$

It can now be assumed that the neural net, having computed these correlations, alters the probabilities of occurrences of the various transitions so as to increase the probabilities of those transitions that usually lead to a reduction in the input signals (and decrease the probabilities of those transitions that cause increases in the input signals). We assume as a crude approximation that those transitions that show correlations of -1/18 have their probabilities increased by 0.2 from 0.1 to 0.3, and that those showing correlations of -1/12 have their probabilities increased by 0.3 to 0.4. If multiple correlations are present, the transition probability is increased by the sum of all the indicated transition probability changes. The new matrix of the Markov process becomes

$$A_m^1 = \begin{bmatrix} 0.7 & 0.1 & 0 & 0.1 & 0 \\ 0.1 & 0.6 & 0.4 & 0 & 0.5 \\ 0.1 & 0 & 0.1 & 0 & 0 \\ 0.1 & 0 & 0.4 & 0.6 & 0.5 \\ 0 & 0.3 & 0.1 & 0.3 & 0 \end{bmatrix}$$

with eigenvector

$$S_o^1 = \frac{1}{904} \begin{bmatrix} 180 \\ 270 \\ 20 \\ 270 \\ 164 \end{bmatrix}$$

and

$$\phi' = \begin{vmatrix} 1 & 0 & 36 & 86 \\ 904 & 36 & 0 & 86 \end{vmatrix}$$

We note that the probability of occupying the 3-state is considerably reduced, but that the probability of occupying the 0-state is not significantly increased. Further, the changes in transition probabilities that might result from the new correlation matrix will not significantly alter this qualitative result.

It is clear from the above that the particular sequential machine of the example will not be driven to the zero state by a neuromime correlation computation of the sort just used. Further, this example illustrates certain qualitative results obtained from the sequential machine analysis. Recall that the neuron models tested have all had a common similarity: changes in the inputs to the neuron models due to some particular neuron model's output have always had the same algebraic sign. Simplified to the Boolean case, this common similarity is necessary and sufficient for assuring that the logic function that the neuron model must generate is linearly separable. The qualitative results from the sequential machine analysis indicate that linear separability is not sufficient for convergence of the present neuron model.

APPENDIX V

PARAMETER CONVERGENCE FOR REDUNDANT NETWORKS

A result similar to that obtained in Section III of this report can be derived for redundant networks. The redundant network analyzed in this appendix had two parallel neuron models. A typical set of equations used by the neuron models to compute their parameters was

$$\dot{y} - \alpha_1 y - \beta_1 M = \epsilon_1 \tag{75}$$

$$\dot{y} - \alpha_2 y - \beta_2 M = \epsilon_2 \tag{76}$$

The subscripts refer to the particular neuron model. Since the neuron models must compute their parameters simultaneously, Equations 75 and 76 must be solved simultaneously. Doing so yields

$$(\alpha_1 - \alpha_2) y + (\beta_1 - \beta_2) M = \epsilon \tag{77}$$

But each neuron model contributes to the output control, M, by the following equation:

$$M = m_1 + m_2 \tag{78}$$

Continuing, using the assumption that $P = KY$,

$$m_1 = -\beta_1 P_1$$

$$P_1 = K_1 y$$

$$m_1 = -\beta_1 K_1 y \tag{79}$$

and similarly for m_2,

$$m_2 = -\beta_2 K_2 y \tag{80}$$

Substituting Equations 79 and 80 back into Equation 78 yields

$$M = -(\beta_1 K_1 + \beta_2 K_2) y \tag{81}$$

Then, combining Equations 77 and 81 yields

$$\epsilon = \left[(\alpha_1 - \alpha_2) - (\beta_1 - \beta_2)(\beta_1 K_1 + \beta_2 K_2) \right] y \tag{82}$$

But Equation 82 must hold for all y. Thus the error equation is obtained:

$$\alpha_1 - \alpha_2 - (\beta_1 - \beta_2)(\beta_1 K_1 + \beta_2 K_2) = \phi \tag{83}$$

The similarity of Equation 83 to the error equation derived in Section III is immediately clear. A plot of parameter variations, which is given in Appendix I, further supports the similarity.

The error equation just derived can be extended to N neuron models by solving N equations simultaneously. Further, to make the analysis valid for different network structures

(other than "parallel"), the relation between the network output and the internal neuron model components must be generalized. That is

$$M = f(m_1, m_2, \ldots m_i)$$
$$i \leq n$$
$$n = 1, 2, \ldots$$

the m_i in turn can be functions of other neuron models within the network.

Analysis of this type of problem for a large redundant network with a multilevel neuron model structure becomes mathematically intractable. The use of a digital computer is suggested.

REFERENCES

1. Griffith, V. V., "A Model of the Plastic Neuron", IEEE Transactions on Military Electronics, Vol MIL-7, No. 2 and 3; April - July, 1963.

2. Study of Distributed Adaptation, GAP-1932, Goodyear Aerospace Corporation, Akron, Ohio; February, 1963.

3. Dale, H., Walter Ernest Nixon Memorial Lecture - "Pharmacology and Nerve Endings," Procedures Roy. Soc. Med., Vol 28, pp 319-332; January, 1935.

4. Harmon, L. D., "Studies with Artificial Neurons, I: Properties and Functions of an Artificial Neuron," Kybernetik, 1, 3, 89-101; 1961.

5. Becker, R. O., Some Observations Indicating the Possibility of Longitudinal Charge Flow in the Peripheral Nerves, Second Annual Bionics Symposium, Ithaca, N. Y., Plenum Press, New York, N. Y.; 1961.

6. Kohler, W.; Held, R.; and O'Connell, D. N.; "An Investigation of Cortical Currents," Procedures Am. Phil. Soc., Vol 96, p 3; 1952.

7. Leitmann, George, Optimization Techniques, Academic Press, 1962.

8. Merriam, C. W., Optimization Theory and the Design of Feedback Control Systems, McGraw-Hill Book Co; New York, 1964.

9. Tompkins, C. B., "Methods of Steepest Descent," Modern Mathematics for the Engineer, edited by E. F. Beckenbach, McGraw-Hill Book Co; New York, 1956.

10. Kalman, R. E., "Contributions to the Theory of Optimal Control," Boletin De La Sociedad Matematica Mexicana, pp 102-119; 1960.

Security Classification

DOCUMENT CONTROL DATA - R&D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Goodyear Aerospace Corporation Akron, Ohio	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP N/A

3. REPORT TITLE

DISTRIBUTED ADAPTATION IN NEUROMIME NETWORKS

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Interim Report - June 1964 to June 1966

5. AUTHOR(S) (Last name, first name, initial)

Griffith, V.V., and Bolen, G.H.

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
January 1967	85	10

8a. CONTRACT OR GRANT NO.	9a. ORIGINATOR'S REPORT NUMBER(S)
AF33(615)-1891 b. PROJECT NO. 7232 c. Task 723203 d.	GER-12738 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) AMRL-TR-66-225

10. AVAILABILITY/LIMITATION NOTICES

Distribution of this document is unlimited

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Aerospace Medical Research Laboratories Aerospace Medical Div, AFSC Wright-Patterson AFB, Ohio

13. ABSTRACT

This report describes investigations of networks with adaptive ability distributed through them. It is thought that large-scale systems can be constructed of adaptive building blocks. These adaptive systems would be flexible in function, reliable and would resist severe damage characteristics of living creatures. Neuron models were tested by interconnecting them into various networks to perform simple control tasks. The test results were evaluated and the evaluation used to improve the theory and the neuron model. Two basic analysis methods were used to study neuromime networks; a sequential machine analysis and an optimal process method applying Pontryagin's maximum principle. The sequential analysis method proved unsatisfactory when applied to an attempted description of an adjustment rule for the neuron model. This difficulty led to an application of optimal processes. The application of Pontryagin's maximum principle to the analysis of the neuron model network described both optimum conditions for a system and criteria useful for developing the adjustment rule.

DD FORM 1 JAN 64 1473

Security Classification

Security Classification

14. KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
Neuron Model Networks						
Distributed Adaptation						
Adaptive Control						

INSTRUCTIONS

1. **ORIGINATING ACTIVITY:** Enter the name and address of the contractor, subcontractor, grantee, Department of Defense activity or other organization *(corporate author)* issuing the report.

2a. **REPORT SECURITY CLASSIFICATION:** Enter the overall security classification of the report. Indicate whether "Restricted Data" is included. Marking is to be in accordance with appropriate security regulations.

2b. **GROUP:** Automatic downgrading is specified in DoD Directive 5200.10 and Armed Forces Industrial Manual. Enter the group number. Also, when applicable, show that optional markings have been used for Group 3 and Group 4 as authorized.

3. **REPORT TITLE:** Enter the complete report title in all capital letters. Titles in all cases should be unclassified. If a meaningful title cannot be selected without classification, show title classification in all capitals in parenthesis immediately following the title.

4. **DESCRIPTIVE NOTES:** If appropriate, enter the type of report, e.g., interim, progress, summary, annual, or final. Give the inclusive dates when a specific reporting period is covered.

5. **AUTHOR(S):** Enter the name(s) of author(s) as shown on or in the report. Enter last name, first name, middle initial. If military, show rank and branch of service. The name of the principal author is an absolute minimum requirement.

6. **REPORT DATE:** Enter the date of the report as day, month, year; or month, year. If more than one date appears on the report, use date of publication.

7a. **TOTAL NUMBER OF PAGES:** The total page count should follow normal pagination procedures, i.e., enter the number of pages containing information.

7b. **NUMBER OF REFERENCES:** Enter the total number of references cited in the report.

8a. **CONTRACT OR GRANT NUMBER:** If appropriate, enter the applicable number of the contract or grant under which the report was written.

8b, 8c, & 8d. **PROJECT NUMBER:** Enter the appropriate military department identification, such as project number, subproject number, system numbers, task number, etc.

9a. **ORIGINATOR'S REPORT NUMBER(S):** Enter the official report number by which the document will be identified and controlled by the originating activity. This number must be unique to this report.

9b. **OTHER REPORT NUMBER(S):** If the report has been assigned any other report numbers *(either by the originator or by the sponsor)*, also enter this number(s).

10. **AVAILABILITY/LIMITATION NOTICES:** Enter any limitations on further dissemination of the report, other than those imposed by security classification, using standard statements such as:

(1) "Qualified requesters may obtain copies of this report from DDC."

(2) "Foreign announcement and dissemination of this report by DDC is not authorized."

(3) "U. S. Government agencies may obtain copies of this report directly from DDC. Other qualified DDC users shall request through _____."

(4) "U. S. military agencies may obtain copies of this report directly from DDC. Other qualified users shall request through _____."

(5) "All distribution of this report is controlled. Qualified DDC users shall request through _____."

If the report has been furnished to the Office of Technical Services, Department of Commerce, for sale to the public, indicate this fact and enter the price, if known.

11. **SUPPLEMENTARY NOTES:** Use for additional explanatory notes.

12. **SPONSORING MILITARY ACTIVITY:** Enter the name of the departmental project office or laboratory sponsoring *(paying for)* the research and development. Include address.

13. **ABSTRACT:** Enter an abstract giving a brief and factual summary of the document indicative of the report, even though it may also appear elsewhere in the body of the technical report. If additional space is required, a continuation sheet shall be attached.

It is highly desirable that the abstract of classified reports be unclassified. Each paragraph of the abstract shall end with an indication of the military security classification of the information in the paragraph, represented as *(TS), (S), (C), or (U)*.

There is no limitation on the length of the abstract. However, the suggested length is from 150 to 225 words.

14. **KEY WORDS:** Key words are technically meaningful terms or short phrases that characterize a report and may be used as index entries for cataloging the report. Key words must be selected so that no security classification is required. Identifiers, such as equipment model designation, trade name, military project code name, geographic location, may be used as key words but will be followed by an indication of technical context. The assignment of links, rules, and weights is optional.

GPO 886-551

Security Classification