SELF-ORGANIZING CONTROL SYSTEMS
The Application of Pattern Recognition in Learning Systems

by
Ralph P. Iwens

ERL Technical Memorandum M-125
27 July 1965

ELECTRONICS RESEARCH LABORATORY
University of California, Berkeley

The research herein was supported by the Air Force Office of Scientific Research under Grant AF-AFOSR-292-64.

ACKNOWLEDGEMENT

The author is grateful to Professor E. I. Jury for his useful comments and discussions during the course of this work.

ABSTRACT

A method for implementing a class of self-organizing control systems is described. A self-organizing control system is able to learn which control to apply to a plant so that the plant is controlled in a near optimum manner. The criterion of optimality considered here is minimum time control.

Learning is considered as a problem in pattern recognition, and a pattern space is defined as an augmented state space. The members of this pattern space are the plant patterns which are to be classified by the system into different categories, each of which corresponds to a certain control choice. A heuristic training program is developed with the aid of which the system learns how to establish the different categories in pattern space so that a near optimum control results. After the learning period the system is ready for actual operation: it measures the plant patterns every T seconds, classifies the pattern into the appropriate category and then applies the corresponding control. Two examples of the proposed method are given. Fortran codes of the programs are presented in the appendix.

A similar approach to learning systems was proposed by K. S. Fu.[1,2] He essentially considers a minimum square error optimality criterion but does not use pattern recognition in the same way as it is used in this paper.

TABLE OF CONTENTS

I. INTRODUCTION

A learning system consists of a plant, a controller, and an information processor. The information processor may be considered as a source of artificial intelligence. Without a priori knowledge of the plant, the system must learn how to control the plant according to a prescribed control objective. Typically, a system error is defined and the control objective is to reduce the system error to zero. Moreover, it is desired that the control objective be accomplished in a near optimum manner, in which the optimality criterion must be specified and furnished to the information processor. One may specify, for instance, that the system error be reduced to zero in minimum time or with a minimum amount of energy consumption.

Learning is accomplished through experience. Before the system can be used to perform its task, it must go through a training phase in which it will learn how to control the plant. During this training period the system subjects itself to a large number of tests which are steered and evaluated by the information processor. The source of intelligence in the information processor stems from a heuristic program.

The main objective here is not to find an optimal control for a system but rather to demonstrate learning and self-organizing features of physical systems which are without life organisms in the biological sense. A block diagram of a learning system is shown in Figure 1.

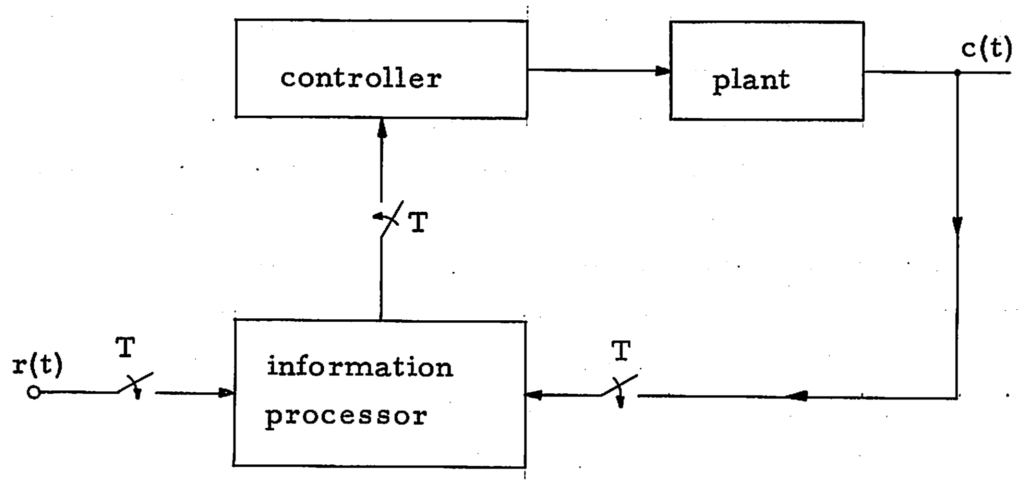


Fig. 1. Simplified representation of a learning system.

II. LEARNING AND PATTERN RECOGNITION

The number of different controls, u_i, that the controller may apply to the plant are limited to members of a set of admissible controls, Ω_u. These restrictions are imposed so as to simplify the problem, but frequently they may also be necessary because of the physical limitations of the controller. In determining Ω_u we should make use of any information available on the characteristics of controls for the desired control objective. For instance, if time optimal control is desired, one may restrict Ω_u to two, constant-control choices, one for maximum-positive forcing and one for maximum-negative forcing, since it is known that these are the time-optimal controls for linear plants. The unknown plant may not be linear, but the proposed controls constitute at least a good guess for being suitable controls.

The system must now learn which of the finite number of control choices should be applied to the plant in which control situations, so as to perform the control objective in a near optimum manner. Control situations are characterized by the state of the plant, the present environmental conditions under which the plant operates, and other information which may be essential to control the plant, such as the instantaneous value of time-varying plant parameters. Each control situation is then characterized by an (n + k) - tuple $\underline{P} = (x_1, \ldots, x_n, m_1, \ldots, m_k) = (\underline{x}, \underline{m})$, where $\underline{x}$ is the state vector of the plant, and $\underline{m} = (m_1, \ldots, m_k)$ is a vector whose components describe the state of the plant environment and instantanious values of time-varying parameters. The vector space to which $\underline{P}$ belongs is called the pattern space $\mathcal{P}$, and $\underline{P}$ is a pattern characterizing a particular control situation. A typical pattern for a second order plant with varying time constant τ and output x would be $\underline{P} = (x, \dot{x}, \tau, T)$ where T is the temperature in which the plant operates.

These patterns must now be classified into categories and each category corresponds to a particular control $u_i \epsilon \Omega_u$. Thus the system must learn how to partition the pattern space and to classify each occurring control situation into the appropriate category and then apply the control associated with this category.

III. ESTABLISHING CATEGORIES

Let E^d be a d-dimensional pattern space in which, according to their properties, patterns are to be classified into R categories. Each category may be thought of as representing a region in the pattern space, separated from other category-regions by boundary hyper-surfaces. These boundaries of the pattern categories are called decision surfaces. The problem is to determine these decision surfaces since this solves the pattern classification problem. There exist cases where an infinite number of decision surfaces are required to classify patterns into a finite number of categories, but we are not concerned with these cases here and the representation of a category by a region in pattern space (where the region must not necessarily be simply connected) is justified.

The decision surfaces of a pattern classifier with R categories can be implicitly defined by a set of R functions $g_1(\underline{P})$, $g_2(\underline{P}), \ldots,$ $g_R(\underline{P})$, which are called discriminant functions. These functions are chosen such that for all $\underline{P}$ in category R_i, $g_i(\underline{P}) > g_j(\underline{P})$ for $i, j = 1, \ldots,$ R, $j \neq i$. In other words, for all patterns $\underline{P}$ belonging to category R_i, the discriminant function $g_i(\underline{P})$ has the largest value. Suppose now that the regions R_i and R_j are adjacent. Then the decision surface separating the two regions is given by

$$g_i(\underline{P}) - g_j(\underline{P}) = 0. \tag{1}$$

In the case where only two pattern categories exist, one has two discriminant functions, $g_1(\underline{P})$ and $g_2(\underline{P})$. For all $\underline{P}$ in R_1, $g_1(\underline{P}) > g_2(\underline{P})$, and for all $\underline{P}$ in R_2, $g_2(\underline{P}) > g_1(\underline{P})$. Then instead of working with these two discriminant functions, only <u>one</u> function

$$g_1(\underline{P}) - g_2(\underline{P}) = g(\underline{P}) \tag{2}$$

need be considered. Then for any pattern $\underline{P}$, if $g(\underline{P}) > 0$, $\underline{P}$ belongs to R_1, if $g(\underline{P}) < 0$, $\underline{P}$ belongs to R_2, and $g(\underline{P}) = 0$ implies that $\underline{P}$ is on the decision surface.

To determine the discriminant functions, it is assumed that they belong to a set of functions which can be represented in the form

$$g(\underline{P}) = \phi(\underline{P}, \underline{W}) = W_1 f_1(\underline{P}) + W_2 f_2(\underline{P}) + \ldots + W_M f_M(\underline{P}) + W_{M+1}. \quad (3)$$

Each member of this family of discriminant functions is called a ϕ-function. Then we define a ϕ-function with weights $W_1, \ldots, W_M$, W_{M+1}, as any discriminant function which depends linearly on the weights W_i.

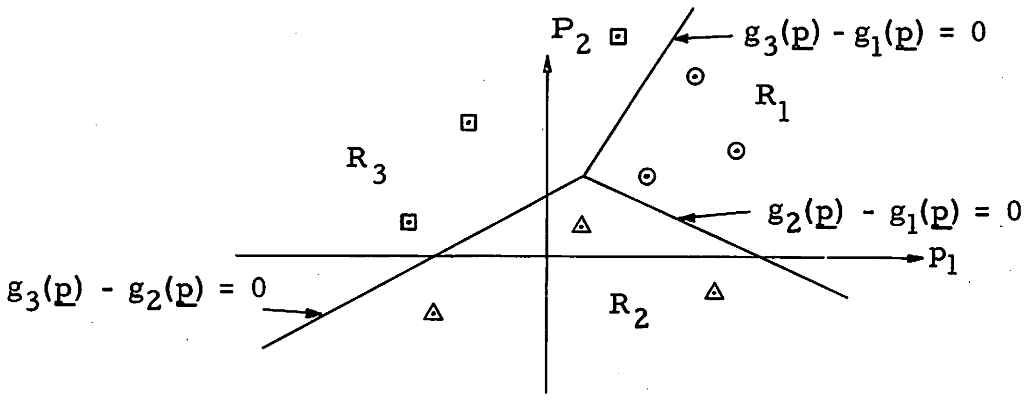
After the functions f_i, $i = 1, \ldots, M$ are specified, the actual value of $g(\underline{P})$ for a given pattern $\underline{P}$ depends only on the choice of the weight vector $\underline{W} = (W_1, \ldots, W_M, W_{M+1})$. The functions f_i must, of course, be linearly independent. More will be said about the choice of the functions f_i later on. For the moment, assume that the f_i's are given, so that only the weights need to be specified in order to determine the discriminant function $g(\underline{P})$.

The weights are determined by a training procedure. For this, a set of training patterns must be available about which it is known to which of the R categories each member of the training set belongs. The weights of the R discriminant functions are then so adjusted that the training patterns are all classified correctly, i.e., for all $\underline{P}$ belonging to R_i, $g_i(\underline{P})$ takes on the largest value of all the discriminant functions, $i = 1, \ldots, R$. This training procedure is known as non-parametric training of a pattern classifier. It consists of an iterative procedure which examines pattern after pattern and adjusts the weights of the discriminant functions until all patterns in the training set are classified correctly. Rosenblatt, Widrow et al. proposed this method and showed that the training procedure will converge, i.e., that numerical values for the components of $\underline{W}$ can be found provided it is true that the discriminant functions chosen can separate the pattern categories.

For example, consider the patterns shown in Fig. 2a. Suppose that it is assumed that the patterns are linearly separable, that is, that the discriminant functions are of the form

$$g_i(\underline{P}) = W_{i1}p_1 + W_{i2}p_2 + W_{i3} \quad i = 1, 2, 3. \tag{4}$$

It is obvious that decision planes (here straight lines) can be found which will separate the pattern categories, and thus the training procedure will converge.



P_2
$g_3(\underline{p}) - g_1(\underline{p}) = 0$
R_1
R_3
$g_2(\underline{p}) - g_1(\underline{p}) = 0$
p_1
$g_3(\underline{p}) - g_2(\underline{p}) = 0$
R_2

Fig. 2a

However, in the case shown in Fig. 2b, the assumption of linear separability is not valid and a training procedure using discriminant functions of the form as given in Eq. 4 would fail.

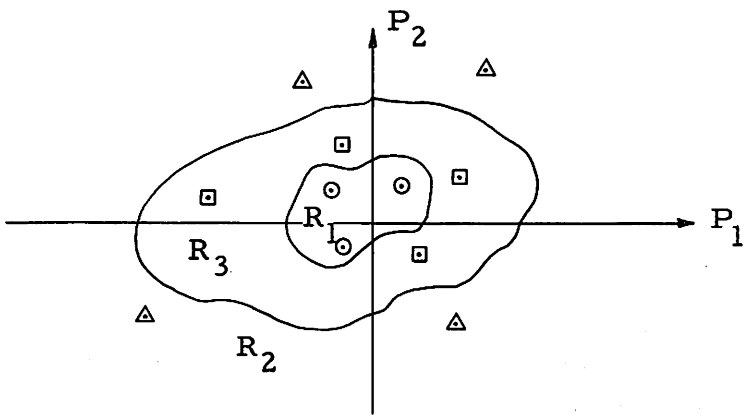


Fig. 2b

It should be noted that the decision surfaces separating the different pattern categories as shown in Fig. 2a and 2b are not unique. It is important therefore, that the set of training patterns be truly representative of all possible patterns which may be encountered when the pattern classifying machine is in actual operation. If the set of training patterns were poorly chosen, many misclassifications would occur later since all decision surfaces have been determined according to the training patterns.

If, in general, the choice of discriminant functions is restricted to ϕ-functions, the next questions are how to choose the functions f_i, and how many of them to choose, such that the probability is large that the associated discriminant functions can classify all the training patterns correctly into their appropriate categories. This probability measures then the effectiveness of a particular family of discriminant functions, where by a family we mean a collection of discriminant functions which differ only in the numerical values of their weights, but are otherwise of identical mathematical form. This probability has been calculated by Cover and others for the case of two-pattern categories. The classification of a set of patterns into two categories is called a pattern dichotomy, and if this classification is obtained by a ϕ-function, it is called a ϕ-dichotomy. For a ϕ-function with $M + 1$ adjustable weights and a set $\mathcal{X}$ of N patterns we like to know what is the probability $P_{N,M}$ that this ϕ-function can implement the pattern dichotomy. Since there exist 2^N possible dichotomies of N patterns, we obtain this probability by dividing the number of ϕ-dichotomies by 2^N. Cover et al. have obtained an expression for the number of ϕ-dichotomies for a set of N d-dimensional patterns. It is given by

$$\Phi(N, d) = 2 \sum_{i=0}^{M} \binom{N-1}{i} . \qquad (5)$$

Note that (5) is independent of d and the f_i's, but a function of M and N, where $M + 1$ is the number of adjustable weights of the ϕ-function. This result is true under one very mild condition: the N patterns must be in ϕ-general position, which means that there exists no surface

$\phi(\underline{P}, \underline{W}) = 0$ in the pattern space containing $M + 1$ members of the training set. The probability that a pattern dichotomy of N patterns in ϕ-general position can be obtained by a ϕ-function with $M + 1$ degrees of freedom ($M + 1$ weights) is then given by

$$P_{N,M} = 2^{1-N} \sum_{i=0}^{M} \binom{N-1}{i}. \tag{6}$$

Let $N = \lambda\,(M + 1)$. Figure 3 shows a plot of $P_{\lambda(M+1),\,M}$ vs λ for various M.

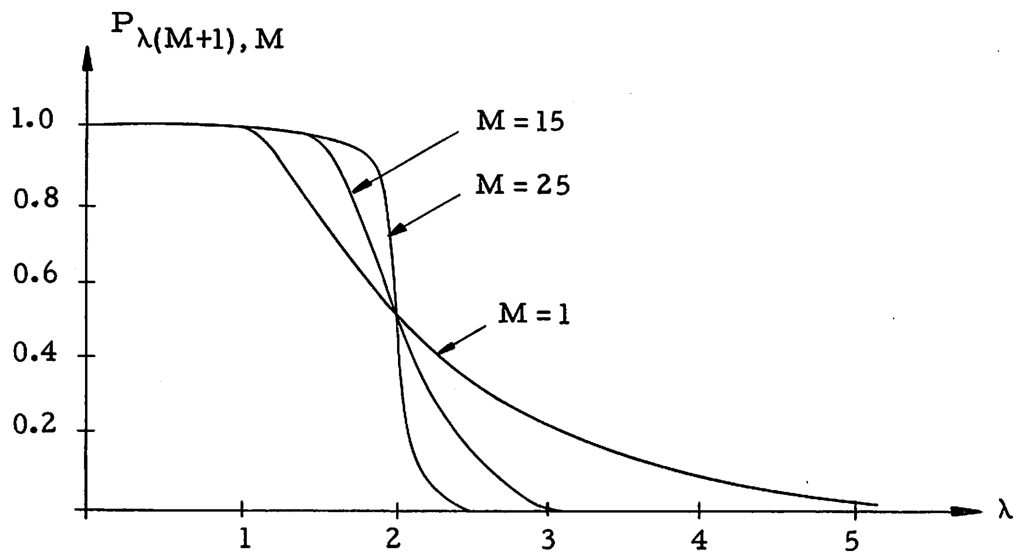


Fig. 3

From Figure 3 one observes a sharp threshold effect around $\lambda = 2$, i. e., $N = 2(M + 1)$. This means that for large M one can almost be certain to obtain any specific dichotomy of fewer than $2(M + 1)$ training patterns with a given ϕ-function, while one is almost certain to fail to achieve any specific dichotomy of more than $2(M + 1)$ training patterns.

The conclusion to this may be summed up as follows: with no a priori knowledge about the decision surfaces of the pattern categories available, the choice of the M linearly independent functions f_i appearing in the ϕ-function is immaterial; however, in order to succeed in determining the weights W_j, $j = 1, \ldots, M + 1$, of $\phi(\underline{P}, \underline{W})$, it is important that for a training set of N patterns $\phi(\underline{P}, \underline{W})$ has more than $\frac{N}{2}$ adjustable weights.

The following two sections describe the operation of the heuristic training program which has been used in the examples.

IV. COLLECTING THE TRAINING PATTERNS

The problem of learning how to control the plant was considered to be equivalent to classifying the plant patterns into categories corresponding to the different controls. From the previous section it is clear that a set of training patterns is required so that these categories can be established.

Problem:

We shall consider the problem of driving the state vector of the plant to the origin using admissible controls $u_i \epsilon \Omega_u = \{u_1 = 1; u_2 = -1\}$. This is equivalent to driving the plant pattern vector from

$$\underline{P}_t = (x_1(t_0), \ldots, x_n(t_0), m_1(t_0), \ldots, m_k(t_0)) \text{ to}$$

$$\underline{P}_{t_f} = (0, \ldots, 0, m_1(t_f), \ldots, m_k(t_f)),$$

where as a rule the environmental patterns cannot be controlled by u_i. Furthermore, the state of the plant shall be driven to the origin in minimum time. Note that only two categories have to be established, therefore only one discriminant function must be found.

In order to collect training patterns, the information processor will cause the controller to apply a control u_i in Ω_u for a time interval of length T to the plant. The information processor observes the behavior of the plant under the influence of this control, then decides whether the control chosen was a good control for this particular control situation, or whether uncertainty exists about the desirability of this control, or whether the control is undesirable. The criterion for the desirability of a control is the second and third time derivative of the norm of the state vector $\underline{X}$ (i.e., the time derivatives of the distance between the plant pattern $\underline{P}$ and the manifold $\mathcal{M}$ in pattern space, $\mathcal{M} = \{ \underline{P} \epsilon \mathcal{P} / \underline{P} = (0, \ldots, 0, m_1, \ldots, m_k) \}$). If the second or third time derivative of the norm of $\underline{X}$ remains negative during the time T in which the control u_i is applied, then u_i is called a good control for this plant pattern.

It may seem that the first time derivative of the norm of $\underline{X}$ should have been used. The first derivative was not chosen since even for very simple systems there exist regions in the state space where neither of the controls in Ω_u will move the phase point toward the origin, and other regions where each of the controls in Ω_u will move the phase point toward the origin. Thus the first derivative will give no information. In these situations the question to ask is: Which control increases the norm of the state vector by a smaller amount, or, which control decreases the norm of the state vector by a larger amount? This leads to the examination of higher order derivatives.

Instead of using time derivatives of the norm of the state vector it is useful at times to use a weighted norm $||\underline{X}||_{\underline{V}} = (< \underline{X}, \underline{VX} >)^{1/2}$ where $\underline{V}$ is a positive definite n x n matrix. This is advantageous if one wants to measure distance differently in different regions of the state space. Loci of constant $||\underline{X}||_{\underline{V}}$ are then in general hyper-elliptic shells. Usually one is uncertain in which region of the state space that distance should be measured differently. But with the relative scaling decided upon, the region may be found by trial and error ("learned by experience") by rotation of $< \underline{X}, \underline{VX} >$. This is equivalent to an orthogonal transformation on the matrix $\underline{V}$, and is therefore easily done.

From here on we shall refer in general to the norm and/or weighted norm of the state vector as the sub-index of performance (SIP) of the system.

Since the equations of the plant dynamics are not known, the time derivatives of SIP must be approximated by measuring differences and taking ratios. The process of collecting training patterns proceeds then as follows:
First the entire pattern space is bounded so that only patterns which lie in a well defined region of the space around the origin are considered. For instance, one may consider only patterns whose Euclidean norm is less than some number $K < \infty$ or, as in the present work, one may put different bounds on individual components of the pattern vector. These bounds must be large enough so that it is unlikely that in practice a plant pattern will occur outside of this region.

Suppose that for some time the controller has applied controls $u_i \epsilon \Omega_u$ to the plant, measuring the plant pattern vector every T seconds. At $t = NT$ the second time derivative of SIP, normalized to the increment T, can be approximated by

$$\Delta^2 SIP(N) = \Delta SIP(N) - \Delta SIP(N-1), \tag{7}$$

where

$$\Delta SIP(N) = SIP(N) - SIP(N-1). \tag{8}$$

T must be chosen small so that the pattern vector does not change its position much over an interval of 2 - 3 T. Depending on the sign of Δ^2 SIP(N) and other previous events, decisions are made about the desirability of the previous control and also what the next control should be. Since a number of calculations and logical operations are involved, which are performed by the information processor, the control to be applied at $t = NT$ is not based on observations made at $t = NT$ since it would take too long to evaluate these observations and determine which new control to use. The control to be applied at $t = NT$ is therefore calculated between $(N-1)T$ and NT. This results effectively in a time lag of T seconds on many decisions and introduces some difficulties. With a fast information processor available, this predetermining of controls is unnecessary. However, for the sake of feasibility considerations the first approach is taken here. This then reinforces the previously mentioned requirement that pattern vectors measured within a few successive intervals T be "close together" in pattern space. For this reason T is adaptive. The pattern space is divided into regions in each of which a different T may be used. In general T decreases with the distance from the origin in order to preserve this notion of "closeness of patterns." The flow chart of Program I shows how decisions are made and orders are executed by the information processor. The flow charts are presented in Section VI.

Whenever ten pattern vectors have been collected, the iterative training procedure mentioned earlier is applied to all patterns collected so far and weights for the discriminant function are calculated. This method allows one to decide when to stop the pattern collecting and training procedure, that is, when the weights of the discriminant function stop changing and no new information can be gained by collecting more training patterns.

The discriminant function used is of the ϕ-type. In agreement with the theory of pattern recognition, the ϕ-function should have more than $N/2$ weights and functions f_i if N training patterns were collected. However, the problem encountered here is of not so general a nature as is often the case in pattern recognition problems. Most likely the two categories to be established in pattern space will be simply connected regions, and it may even be known that the separation surface has to pass through the origin of the state space. Therefore ϕ-functions with by far less than $N/2$ weights can be used. Since the pattern space of interest has been bounded, one can represent the decision surface by an orthogonal expansion over a finite interval and choose the first M terms of the series as the functions f_i in $\phi(\underline{P}, \underline{W})$. The number M must ultimately be determined by trial and error, i. e., the information processor may learn the proper value of M by increasing M if the system fails to obtain a pattern dichotomy with the value of M given initially.

At the end of the pattern collecting and training procedure the information processor will run a test on the system. Just as in the training procedure, the pattern vector will be measured every T seconds, but the discriminant function which has just been determined will be used to classify each measured plant pattern and thus determine the appropriate control. If the control objective is met successfully, the pattern collecting and training procedure is complete. If the plant cannot be controlled, the discriminant function is poor. This may be due to

1) M was too small,
2) T was too large,
3) The training patterns are not well distributed and thus not representative of actually occurring plant patterns.

In this case the information processor will iterate on the pattern collecting and training procedure by using a larger M and a smaller T. The training patterns are usually well distributed over the bounded pattern space since the plant is reset at random to initial states whenever the state vector leaves the bounded pattern space or very closely approaches the origin.

If the test on this system is successful, the discriminant function <u>still will not</u> represent the switching surface for time optimal control. At most, it approximates the switching surface for minimum-square-error control, since the training patterns were essentially collected by a minimum-square-error criterion.

V. GRADIENT TECHNIQUES

The weights of the discriminant function determined by the training procedure will now be adjusted so that the discriminant function represents a good approximation of the true time-optimal switching surface. During this adjustment procedure, time varying parameters as well as environmental conditions of the system must remain constant over short time intervals during which the weights are adjusted. Therefore, the following discussion is restricted to the state space of the system and does not consider the entire pattern space.

The cost function of the system now becomes a function of $\underline{W}$, that is,

$$J = \psi(\underline{W}) = NT, \tag{9}$$

where ψ is an unknown scalar-valued function and NT is the time required to drive the system from some initial state to within a predetermined distance Δ of the origin of the state space. Thus, J is a different function for each initial state. But if the switching surface could be exactly described by the discriminant function with the appropriate numerical values for the weights, then each function $J_{\underline{x}_i(0)} = \psi_i(\underline{W})$ would have an absolute minimum for the same value of $\underline{W}$. The absolute minimum for an individual $J_{\underline{x}_i(0)}$ does not occur at some fixed point $\underline{W}$, but $J_{\underline{x}_i(0)}$ attains its absolute minimum for all $\underline{W}$ which belong to a certain manifold in the space $\mathcal{W}$. To see this, consider the following example for a particular $J_{\underline{x}(0)}$ of a double integrator with time varying gain K(t). Let

$$g(\underline{P}) = W_1x_1 + W_2x_2 + W_3K(t)x_1^3, \tag{10}$$

and let

$$\underline{X}(0) = (-4, 0) \text{ and } K(t) = 1, \ 0 \leq t \leq t_1, \ t_1 > NT.$$

The switching surface $g(\underline{P}) = 0$ is given by

$$W_1x_1 + W_2x_2 + W_3x_1^3 = 0. \tag{11}$$

Now from Fig. 4 it can be seen that $J_{\underline{x}(0)} = \psi(\underline{W})$ will have a minimum for all $\underline{W}$ such that

$$-2W_1 + 2W_2 - 8W_3 = 0 . \qquad (12)$$

Therefore, any $\underline{W}$ in the manifold described by (12) will minimize $J_{\underline{x}(o)}$.

Each other $J_{\underline{x}_i(o)}$ will have its manifold $\mathcal{M}_i$ such that for all $\underline{W} \epsilon \mathcal{M}_i$ $J_{\underline{x}_i(o)}$ is at its absolute minimum. If the true switching curve could be exactly described by $g(\underline{P})$, then the desired value of $\underline{W}$ would be the intersection of all these manifolds $\mathcal{M}_i \epsilon \mathcal{W}$.

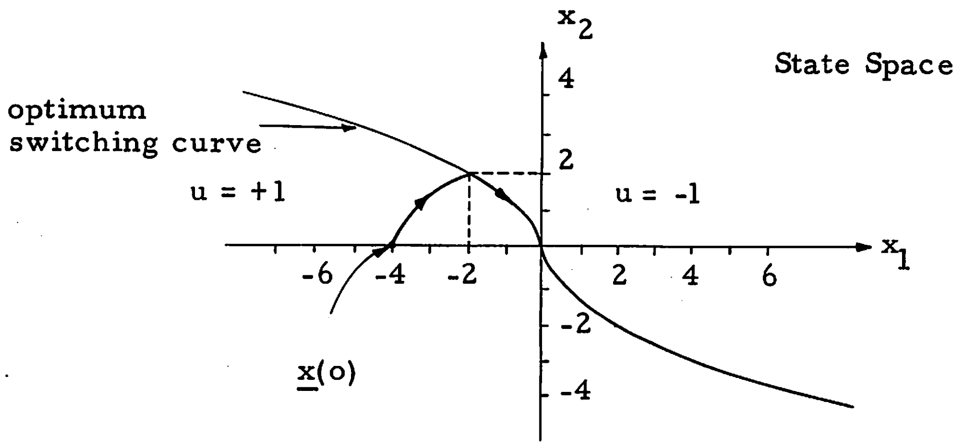


Fig. 4

In practice one should not hope for the case that $g(\underline{P})$ can describe the switching surface exactly. It almost never occurs since switching surfaces are usually described piecewise by different functions. However, for a well selected discriminant function with sufficiently many weights, there exists $\underline{W}_o$ such that $g(\underline{P}, \underline{W}_o) = 0$ is a good approximation for the switching surface over the bounded region of the pattern space.

It is assumed now that the $\underline{W}$ determined by the training procedure represents a point within the convex region, containing the manifold $\mathcal{M}_i$, of each $J_{\underline{x}_i(o)}$. Now gradient techniques will be applied to determine $\underline{W}_0$. The initial condition $\underline{x}_i(0)$ is selected at random and the gradient $\nabla\psi_i(\underline{W})$ is approximated by difference quotients $\frac{\Delta J_{\underline{x}_i(0)}}{\Delta W_i}$, $j = 2, \ldots , M + 1$.

The difference quotients are obtained through measurements of $\Delta J_{\underline{x}_i(0)}$ when the individual components of $\underline{W}$ are perturbed by ΔW_j, $j = 2, \ldots, M + 1$ (one of the weights is redundant). This requires of course resetting the plant M times to the same initial conditions (with no change in the environment). Therefore the $\underline{P}_i(0)$ can only be selected from a restricted region of the pattern space where this is possible. After M measurements have been obtained $\underline{W}$ is changed by

$$\Delta\underline{W} = - \Delta\sigma \nabla\psi_i(\underline{W}) \tag{13}$$

where $\Delta\sigma$ is a scalar increment which determines the magnitude of $\Delta\underline{W}$. Then another $\underline{x}_i(0)$ is selected at random and the process is repeated until some sort of convergence can be detected. No matter how long the process is continued and no matter how small the increments $\Delta\underline{W}$ are chosen, exact convergence to some point $\underline{W}$ will not occur. This is because for no $J_{\underline{x}_i(0)}$ will the gradient ever be zero, since $\phi(\underline{P}, \underline{W})$ cannot represent the switching surface exactly. Therefore, if $\underline{W}$ starts to vary only within a small percentage range of its value, the gradient technique should be viewed as successfully terminated and $\underline{W}^*$ chosen as the mean of the range over which $\underline{W}$ varies. However, this kind of convergence can not be guaranteed. There is no certainty that the $\underline{W}$ with which the gradient technique was started lies within the convex region of the "deepest valley" of each function $J_{\underline{x}_i(0)}$. For nothing is known about the functions $\psi_i(\underline{W})$, and in general they seem to be very unwieldy functions and not convex at all. In addition to this, $J_{\underline{x}_i(0)}$ is quantized by the sampling period T which affects the measurements of the gradients. The following examples will demonstrate however, that this method can be used successfully, although it may not be successful in all cases. Since no proof for the convergence of the gradient technique, as applied here, can be given, the method should be viewed as a heuristic. It essentially represents an improvement scheme which man would use while learning some particular task. A flow chart of the program follows. Fortran codes ~~are~~ presented in the Appendix ~~II which~~ implements the gradient techniques in the following section.

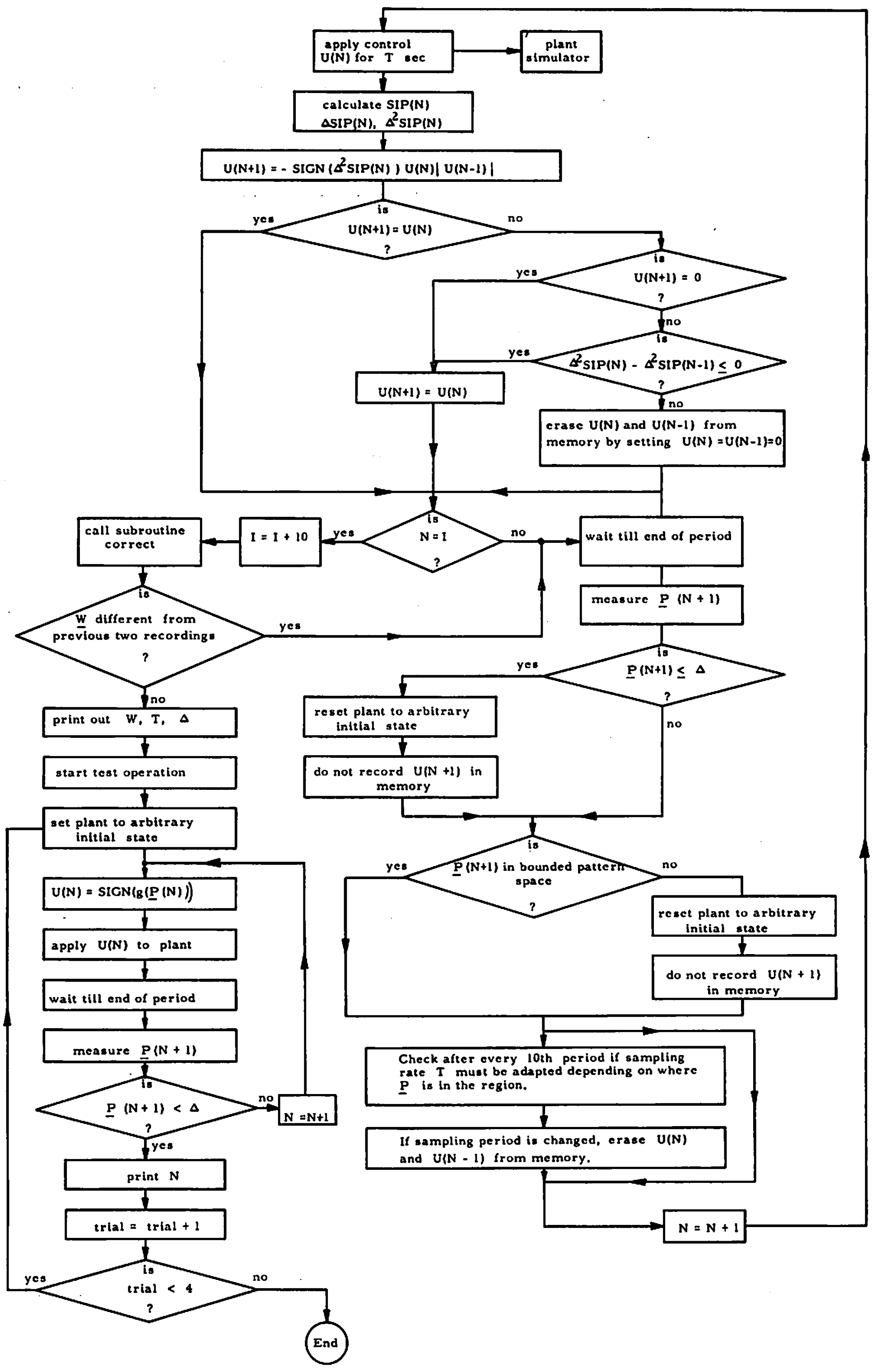
VI. COMPUTER PROGRAMS

A. DATA ON COMPUTER PROGRAMS:

1. Storage
   a. During Learning Phase:
      1) Memory lists: $(M+4) \times$ (number of training patterns collected) $+ M + 20$ words.
      2) Storage for execution of program: Approximately 120 words.
2. During Actual Operation of the System: (Special purpose computer is sufficient).
   a. Memory lists: $M + 2$ words.
   b. Storage for execution of program: Approximately 50 words.
   M = number of functions f_i contained in $\phi(\underline{P}, \underline{W})$.
   The number of training patterns collected typically varies between 100 and 400.

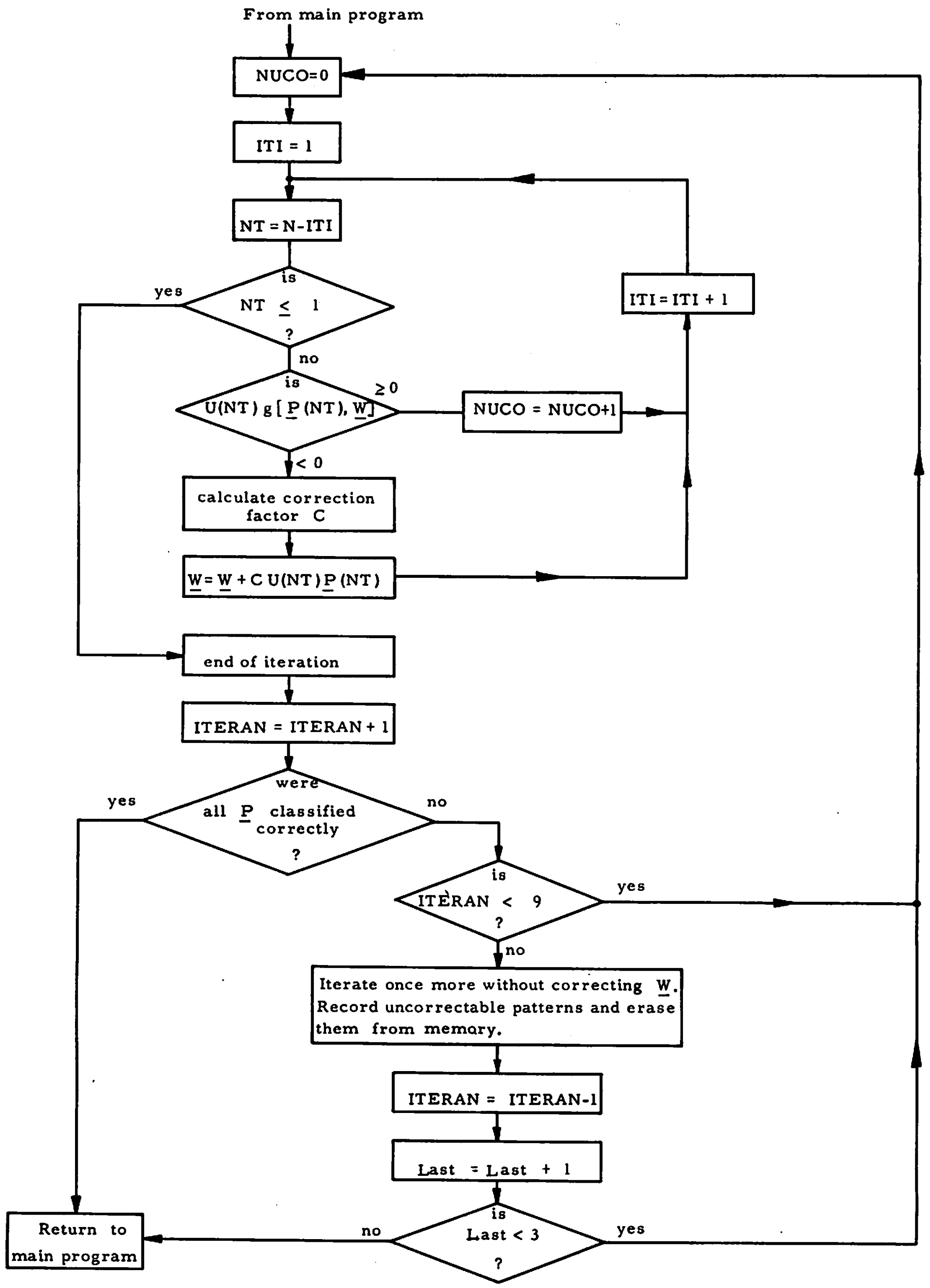
B. COMPUTING TIME:

Computing time for these programs includes the simulation of the plant. Compilation time of the Fortran codes is excluded. The programs were run on an IBM 7094 machine.
Program I and Subroutines: 0.8 minutes for 300 training patterns; $M = 4$. Program II and Subroutines: 3.35 minutes for 200 iterations; $M = 4$. The flow charts of Programs I and II follow.

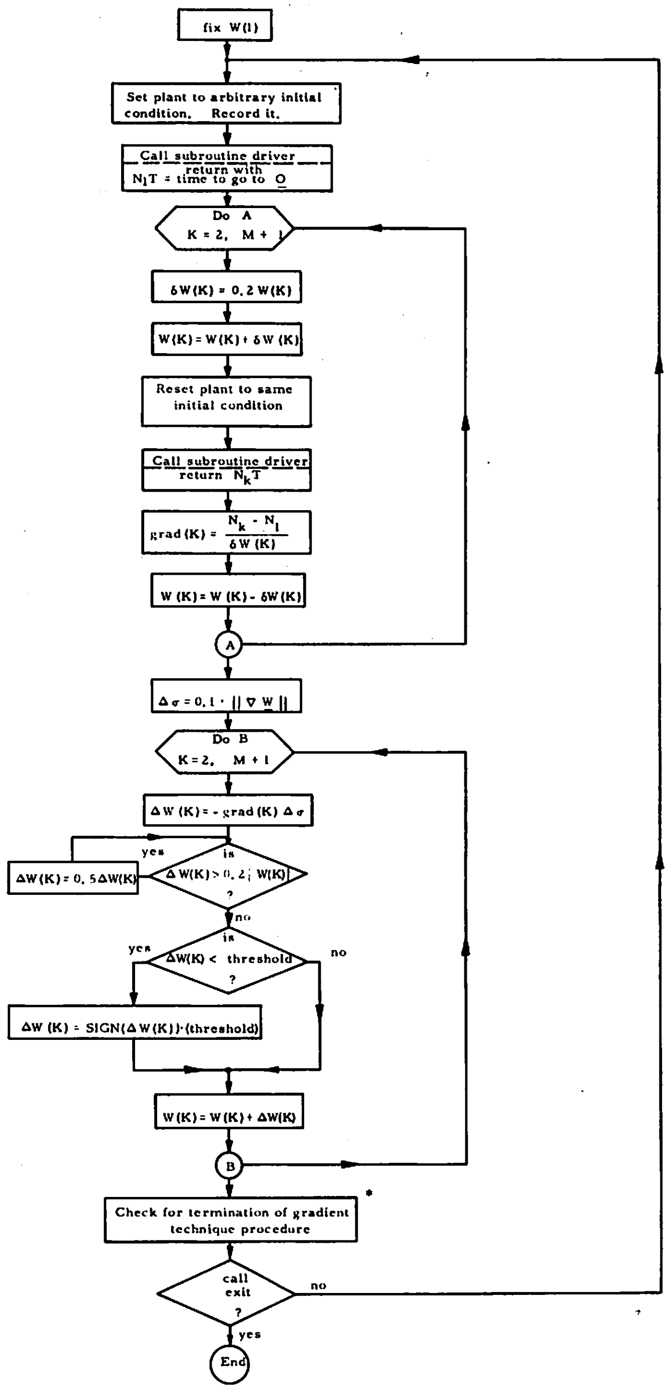


apply control U(N) for T sec
plant simulator
calculate SIP(N) ΔSIP(N), Δ²SIP(N)
U(N+1) = - SIGN (Δ²SIP(N)) U(N)| U(N-1) |
is U(N+1) = U(N) ?
yes
no
is U(N+1) = 0 ?
is Δ²SIP(N) - Δ²SIP(N-1) ≤ 0 ?
U(N+1) = U(N)
erase U(N) and U(N-1) from memory by setting U(N) =U(N-1)=0
is N = I ?
I = I + 10
call subroutine correct
wait till end of period
measure P (N + 1)
is P(N+1) ≤ Δ ?
reset plant to arbitrary initial state
do not record U(N +1) in memory
is W different from previous two recordings ?
print out W, T, Δ
start test operation
set plant to arbitrary initial state
U(N) = SIGN(g(P (N)))
apply U(N) to plant
wait till end of period
measure P (N + 1)
is P (N + 1) < Δ ?
N =N+1
print N
trial = trial + 1
is trial < 4 ?
End
is P (N+1) in bounded pattern space ?
reset plant to arbitrary initial state
do not record U(N + 1) in memory
Check after every 10th period if sampling rate T must be adapted depending on where P is in the region.
If sampling period is changed, erase U(N) and U(N - 1) from memory.
N = N + 1

Flow Chart of Program I: Pattern Collecting and Training Procedure.

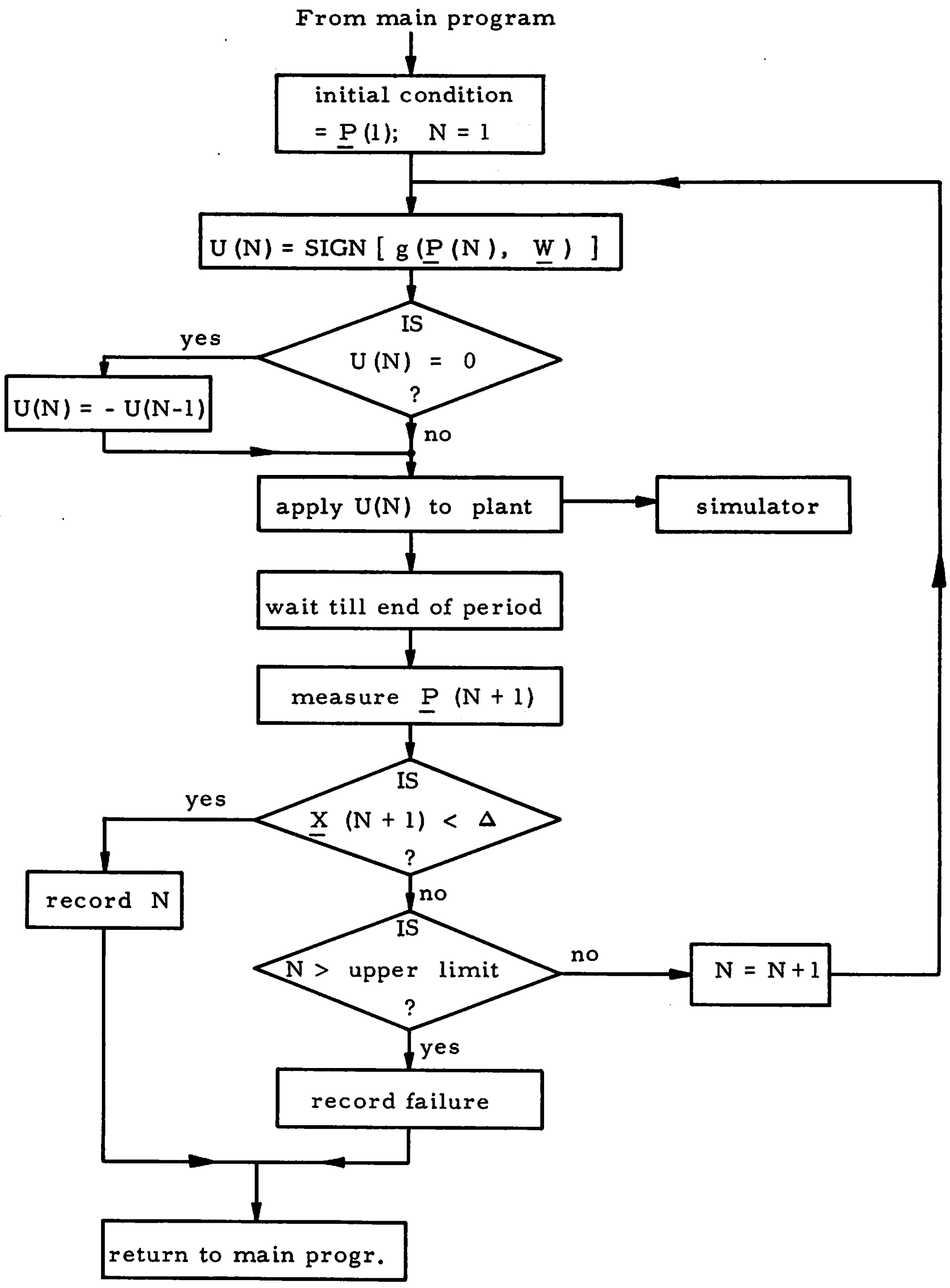


Flow Chart of Program I Subroutine Correct.


fix W(1)
Set plant to arbitrary initial condition. Record it.
Call subroutine driver return with N_1T = time to go to O
Do A K = 2, M + 1
δW(K) = 0.2 W(K)
W(K) = W(K) + δW (K)
Reset plant to same initial condition
Call subroutine driver return N_kT
grad (K) = $\frac{N_k - N_1}{\delta W(K)}$
W (K) = W (K) - δW(K)
A
Δσ = 0.1 · ‖∇ W‖
Do B K = 2, M + 1
ΔW (K) = - grad (K) Δσ
yes
is ΔW(K) > 0.2 | W(K)| ?
ΔW(K) = 0.5ΔW(K)
no
yes
is ΔW(K) < threshold ?
no
ΔW (K) = SIGN(ΔW(K))·(threshold)
W(K) = W(K) + ΔW(K)
B
Check for termination of gradient technique procedure *
call exit ?
no
yes
End

*
The termination procedure may consist of checking whether the last 10 values of W vary only within a prescribed percentage of the value of the latest W, or it may just consist of checking if a predetermined number of iterations has been performed.

Flow Chart of Program II Gradient Techniques.



Flow Chart of Program II Subroutine Driver.

VII. EXAMPLES

In the following two examples the knowledge of the plant is only used to simulate the plant and to compare results with those obtained from mathematical analysis when the plant dynamics are known.

Example 1.

In this example no environmental patterns are considered so that the pattern space is identical to the state space of the system. The system shown in Fig. 5 is to learn to drive the plant from any initial condition in as short a time as possible to within one unit of the origin of the state space. The state space is bounded by $|x_1| \leq 20$, $|x_2| \leq 20$.

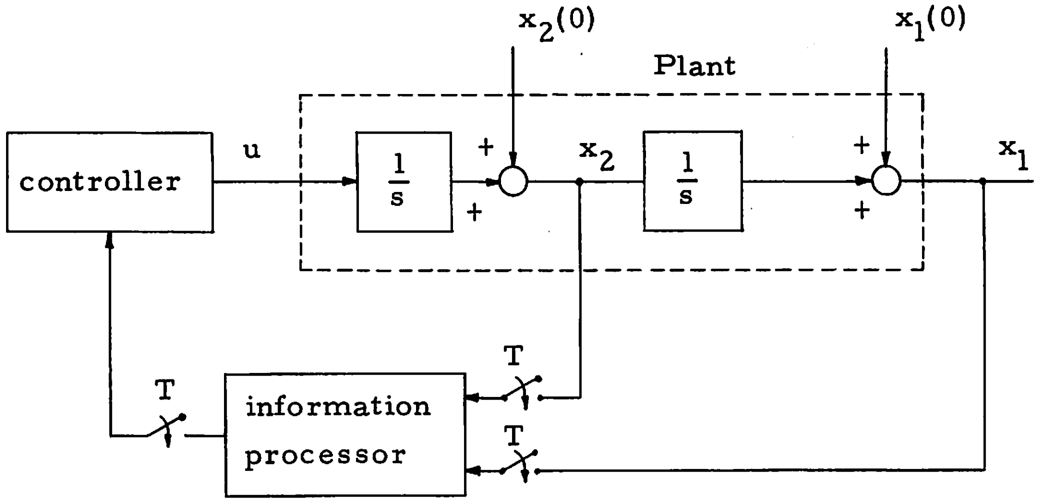


Fig. 5

Step 1:

Pattern collecting and training procedure: Program No. 1 is used to supervise this step. Fig. 6 shows three different decision surfaces that were obtained from the pattern collecting and training procedure. The control objective could be fulfilled with any one of the three curves.

Curve 1.

An attempt was made to express the decision surface by the first three terms of an orthogonal expansion (Fourier Series) over the interval [-20, 20]. Thus, $g(\underline{x})$ was assumed of the form

$$g(\underline{x}) = W_1 x_2 + W_2 \sin\frac{\pi x_1}{40} + W_3 \sin\frac{3\pi x_1}{40} + W_4 \sin\frac{5\pi x_1}{40}. \tag{14}$$

Sub-index of performance:

$$SIP = [<\underline{x}, \underline{V}\ \underline{x}>]^{1/2},$$

where $\underline{V} = \underline{P\Omega P}^*$

with

$$\underline{P} = \begin{bmatrix} \cos 60^\circ & -\sin 60^\circ \\ \sin 60^\circ & \cos 60^\circ \end{bmatrix} \qquad \underline{\Omega} = \begin{bmatrix} 9 & 0 \\ 0 & 1 \end{bmatrix}$$

The training procedure yielded:

$$\underline{W} = (-0.05, -0.25, -0.13, 0.09).$$

Not enough training patterns were collected near the boundaries of the bounded state space, thus explaining the drooping of the learned decision surface near the boundaries.

<u>Curve 2.</u>

Assumed form of the discriminant:

$$g(\underline{x}) = W_1x_1 + W_2x_2 + W_3x_2^2 + W_4x_2^3 . \tag{15}$$

Sub-index of performance:

SIP = $[<\underline{x}, \underline{V}\,\underline{x}>]^{1/2}$ with $\underline{V}$ as for curve (1).

The training procedure yielded:

$\underline{W} = (-0.09, -0.19, 0.00, -0.03)$.

<u>Curve 3.</u>

Assumed form of discriminant:

$$g(\underline{x}) = W_1x_1 + W_2x_2 + W_3x_2^2 + W_4x_2^3 \tag{16}$$

which is the same as (14).

Sub-index of performance:

SIP = $[<\underline{x}, \underline{x}>]^{1/2} = || x ||$.

<u>Curve 4.</u>

Ideal switching curve of the system computed for comparison purposes.

During pattern collecting and training procedure the sampling period T was 0.5 and 0.25 sec, respectively, (in different regions of the state space) for all three learned switching curves.

Step 2: Gradient Techniques:

Program 2 is used to supervise this step. In this example curve (2) of Fig.6 is used as the starting point of the gradient technique. Fig. 7 summarizes the entire learning procedure. It shows:

a. the training patterns which were collected and used

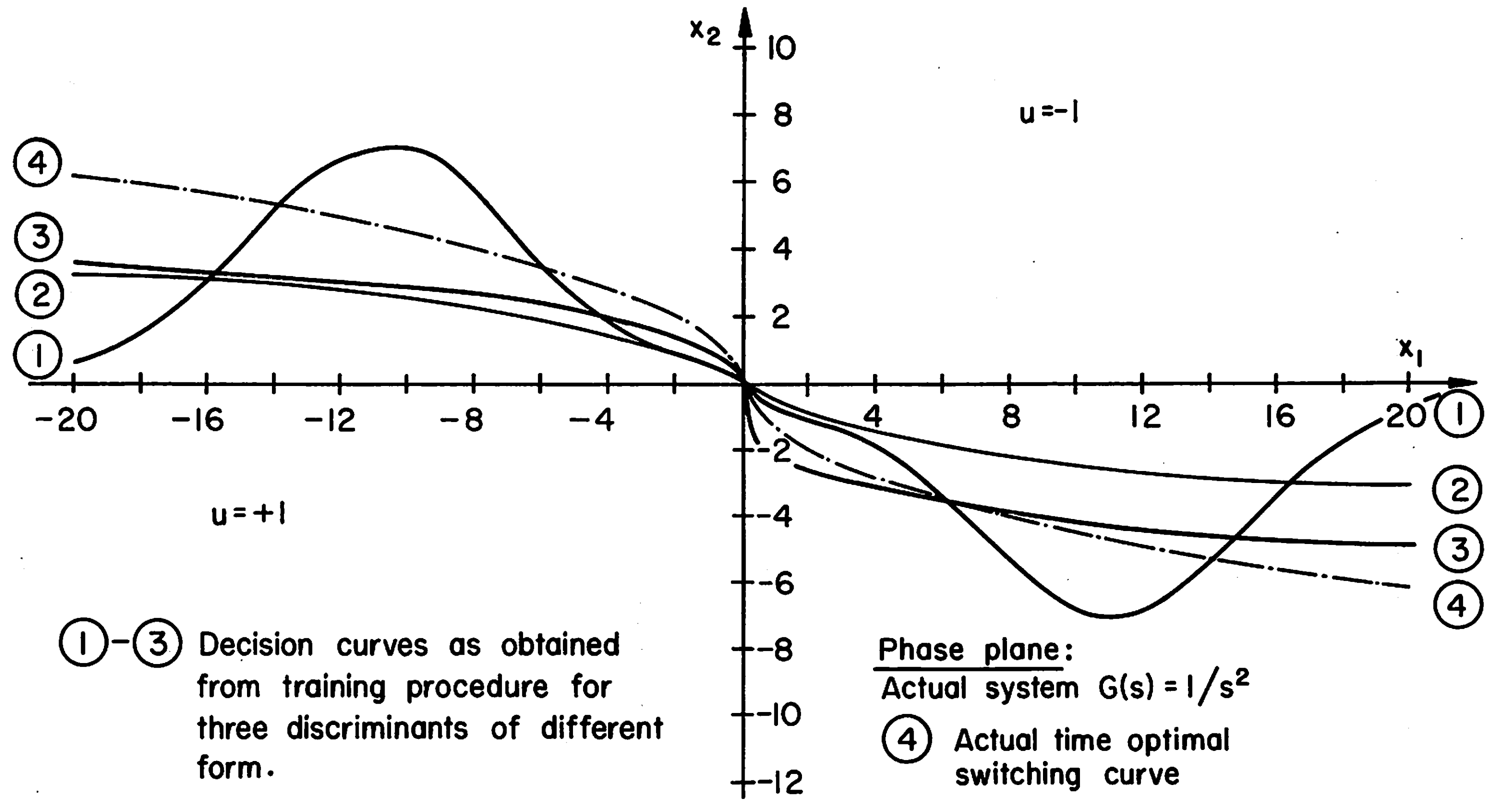


Fig. 6. Three decision curves obtained from the pattern collecting and training procedure.

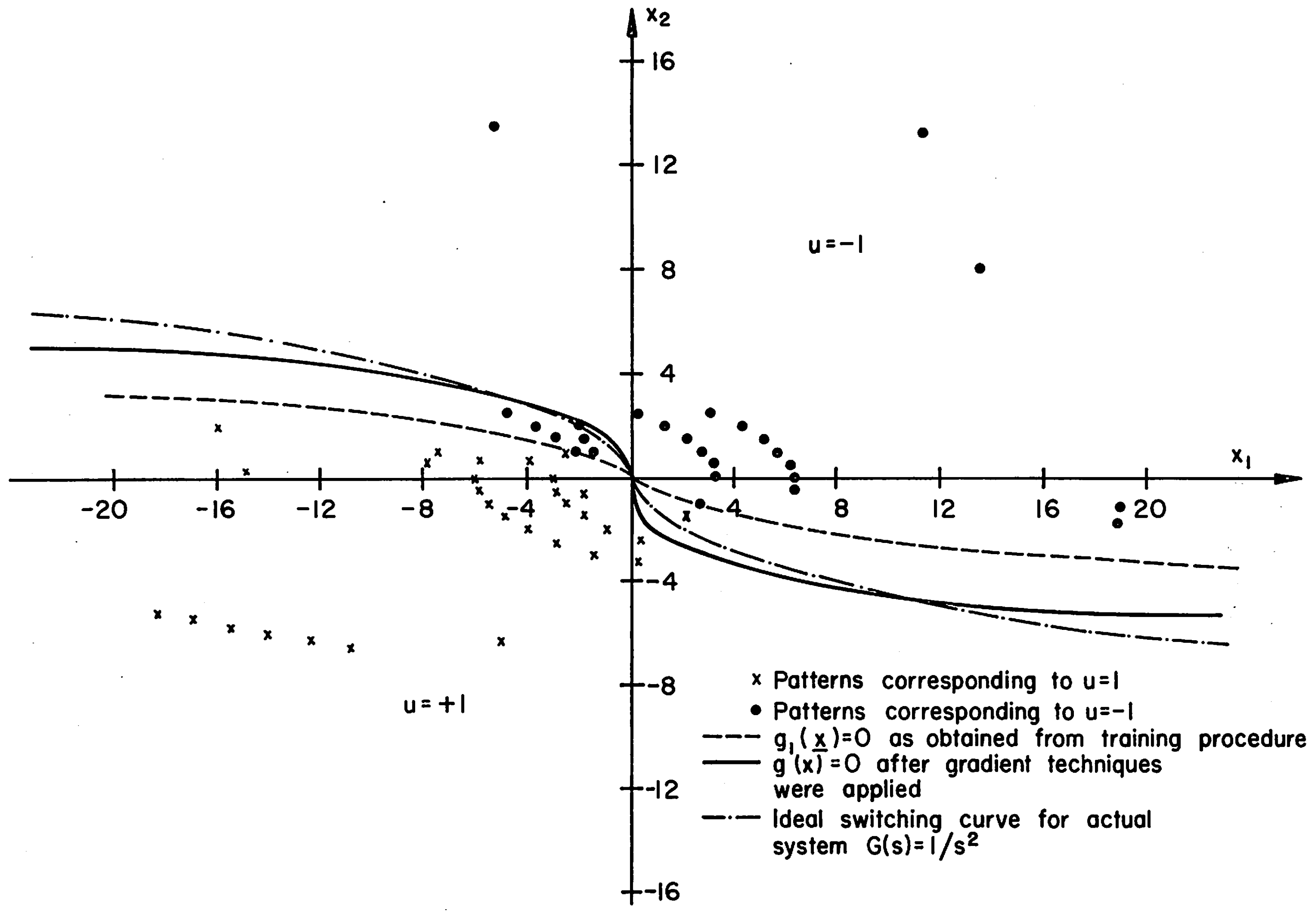


Fig. 7. Illustrating learning procedure.

in the training procedure,

b. the resulting decision surface (dashed curve, same as Curve 2 of Fig. 6):

$$g_1\,(\underline{x}) \overbrace{= -0.1}^{-0.09x_1} 9x_2 - 0.03x_2^{\,3} = 0, \tag{17}$$

c. the ultimate decision surface (switching curve) obtained after gradient techniques were applied (solid line):

$$g(\underline{x}) = -2\ \ 25x_1 - 0.08x_2 - 0\ \ 25x_2^{\,2} - 0\ \ 3x_2^{\,3} = 0. \tag{18}$$

Then if

$$g(\underline{x}(N)\,) < 0 \Longrightarrow u\,(N) = -1,$$
$$g(\underline{x}(N)\,) = 0 \Longrightarrow u\,(N) = -u\,(N-1),$$
$$g(\underline{x}(N)\,) > 0 \Longrightarrow u\,(N) = +1.$$

For comparison purposes the ideal switching curve of the system is also shown in Fig. 7 (dash-dot-dash).

Fortran codes of programs (1) and (2) as used in this example are presented in the appendix.

Example 2: Regulator Problem: Dual Mode Operation.

In this example a plant with time varying gain $K(t)$ is considered. The system shown in Fig. 8 is to learn to drive the system error to zero in minimum time. The state variables chosen are $x_1 = e$ and $x_2 = \dot{e}$. When the system error is within Δ units of the origin the usual negative feedback mode of operation is used to drive the error exactly to zero This avoids a limit cycle around the origin. The system is considered to be a regulator and as reference inputs only steps $r(t) = k1(t)$, $|k| \leq 100$, are considered. The pattern space is a three-dimensional space $\mathcal{P} = \{\underline{P} \mid \underline{P} = (x_1, x_2, K)\}$. It has been bounded by $|x_1| \leq 100$, $|x_2| \leq 100$, $0 < K \leq 100$.

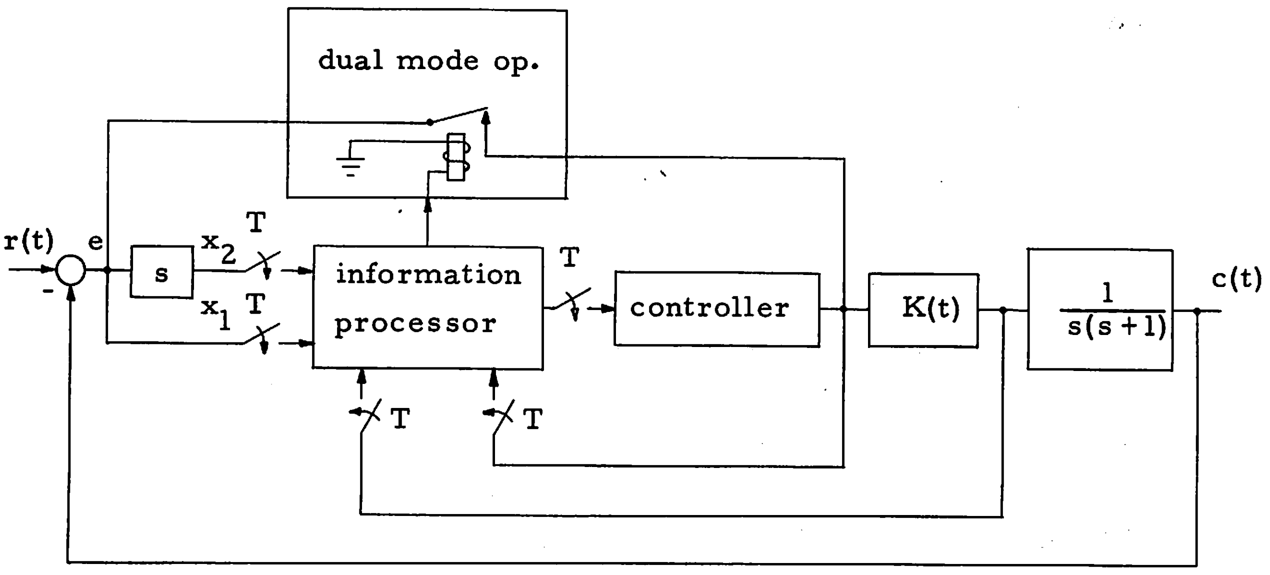


Fig. 8

Step 1: Pattern Collecting and Training Procedure:

Program (1) is used to supervise this step. A discriminant function of the form

$$g(\underline{P}) = W_1x_1 + W_2x_2 + W_3x_2^2 + W_4x_2^3 + W_5\,\frac{x_2}{K} + W_6\left(\frac{x_2}{K}\right)^2 + W_7\left(\frac{x_2}{K}\right)^3 \tag{19}$$

was assumed. As the results of the training procedure showed, this was not a very good choice. A discriminant function with a larger degree of freedom, i.e., more adjustable weights, should have been chosen. Curve 1 in Fig. 9 shows the decision surface that was obtained from the training procedure. It is essentially the same curve for all K, $10 \leqslant K \leqslant 100$.

Sub-index of performance:

$$SIP = [\ <\underline{x},\ \underline{V}\ \underline{x}>\ ]^{1/2}$$

where $V = \underline{P}\,\underline{\Omega}\,\underline{P}^*$

with

$$\underline{P} = \begin{bmatrix} \cos 30^{\circ} & -\sin 30^{\circ} \\ \sin 30^{\circ} & -\cos 30^{\circ} \end{bmatrix} \qquad \underline{\Omega} = \begin{bmatrix} 9 & 0 \\ 0 & 1 \end{bmatrix}$$

The training procedure yielded:

$\underline{W}$ = (10.0, 0.113, -0.0781, 0.0156, 0.00088, -0.00082, 0.00039).

Training patterns cannot be shown in Fig. 9 since the actual pattern space is three-dimensional.

Although the training procedure did not distinguish between different K's, $10 \leq K \leq 100$, the system was able to fulfill the control objective (i.e., drive the error to zero) with the determined discriminant function for all K's, $10 \leq K \leq 100$.

Step 2: Gradient Techniques:

Program (2) is used to supervise this step. Since the pattern collecting and training procedure did not account for any difference in the decision surface for different gains K, the gradient technique method will be applied for different constant K's. That is, the range of K will be quantized and a number of decision surfaces for different K's will be determined. Fig. 9 shows the resulting decision surfaces (switching curves) for three different K's: K = 10, K = 50, K = 100. The corresponding discriminant functions are all of the form

$$g(\underline{x}) = W_1x_1 + W_2x_2 + W_3x_2^{\,2} + W_4x_2^{\,3}. \qquad (20)$$

The different weight vectors $\underline{W}$ are given respectively as:

a. for K = 10; $\underline{W}_{10}$ = (10.0, -0.000148, -0.00105, 0.00065)

b. for K = 50; $\underline{W}_{50}$ = (10.0, -0.000103, -0.000860, 0.000417)

c. for K = 100; $\underline{W}_{100}$ = (10.0, -0.00012, -0.00147, 0.000141).

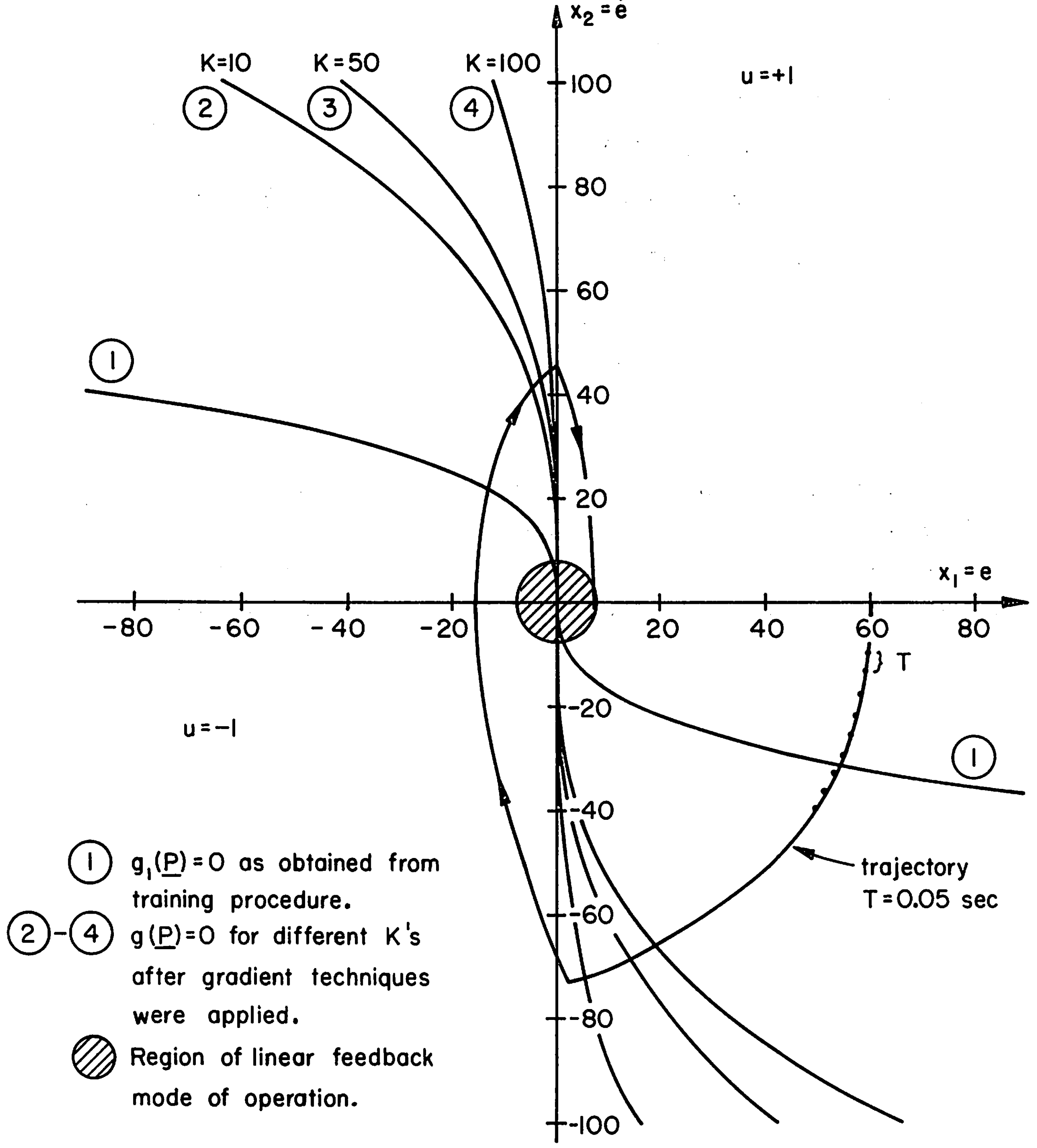


Fig. 9. Results of learning process.

Then we have again:

$$g(\underline{x}(N),\ K(N)) < 0 \implies u(N) = -1$$

$$g(\underline{x}(N),\ K(N)) = 0 \implies u(N) = -u(N-1)$$

$$g(\underline{x}(N),\ K(N)) > 0 \implies u(N) = +1$$

A typical trajectory for $r(t) = 60 \cdot 1(t)$ is shown in Fig. 9. The crosshatched area around the origin is the region where conventional linear feedback is used as shown in Fig. 8. The controller is bypassed in this mode of operation.

During the entire learning process the sampling period was $T = 0.05$ sec throughout the entire bounded pattern space with the exception of the region where $K \leq 15$; there $T = 0.1$ sec. For comparison, the ideal switching curves of the system are shown in Fig. 10.

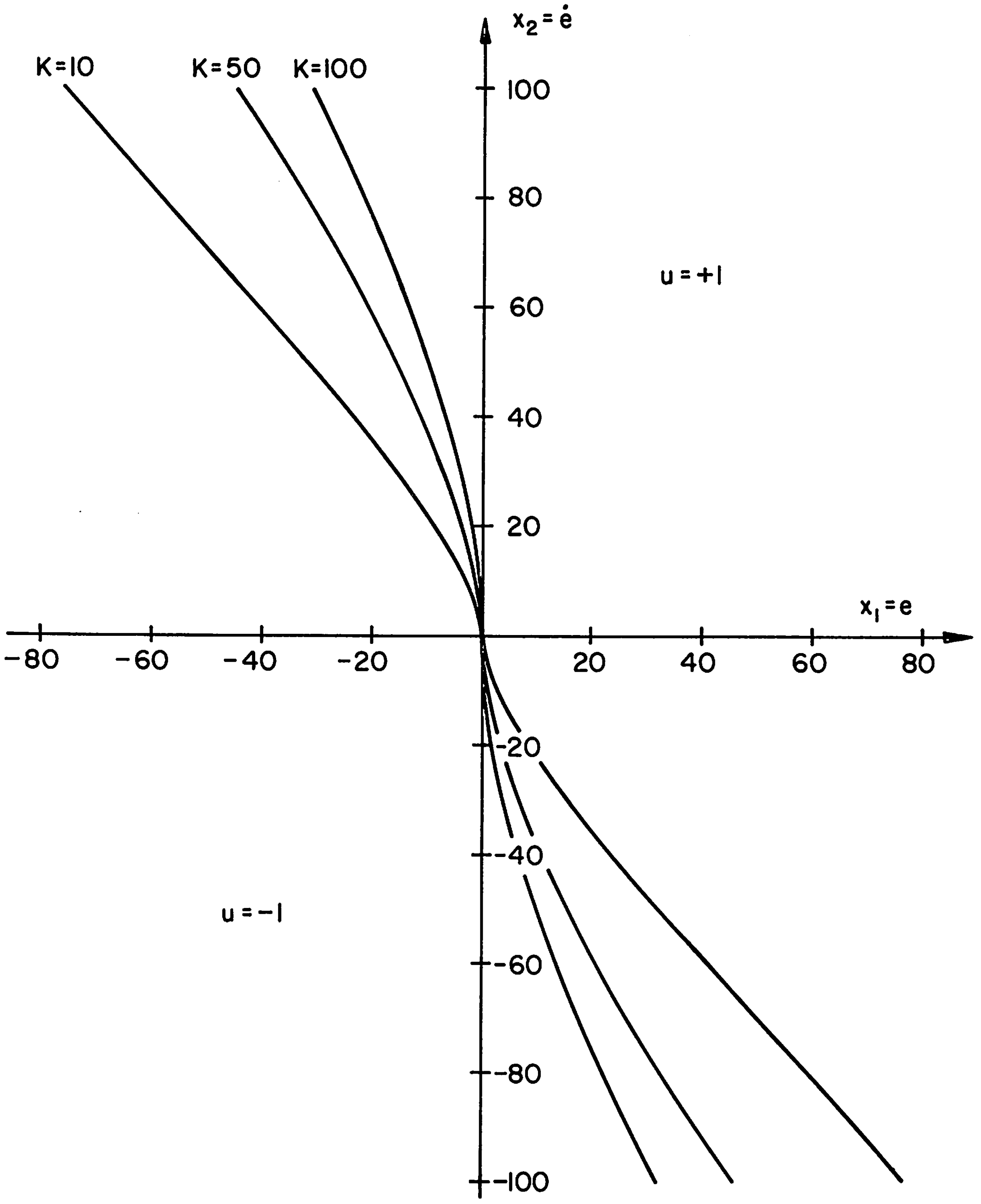


Fig. 10. Ideal switching curves for system shown in Fig. 8.

CONCLUDING COMMENTS

A possible method for implementing self-organizing control systems by means of pattern recognition has been presented. Two examples of the method were given. The class of systems that has been considered is characterized by allowing only two control choices. In particular, the case of time-optimal control has been considered with the set Ω_u of admissible controls restricted to $u = \pm 1$.

Training patterns were collected and then pattern recognition techniques applied to partition the pattern space, which contains the plant patterns characterizing different control situations. The training patterns were collected according to a sub-index of performance, SIP. Here, SIP was such that the control choice was termed good for a certain plant pattern, if the second and/or third time derivative of the weighted norm of the state vector of the system remained negative during the application of the control.

When more than two control choices are allowed, this kind of SIP is not very well suited. With a different SIP the pattern recognition and training techniques can be used to extend the method such that it can then also be applied to systems where more than two controls must be allowed; for instance, for systems which are to operate with a minimum amount of fuel consumption. Therefore, it seems most advantageous to direct further research towards finding better SIP's which can be used to collect truly representative training plant-patterns and their associated controls for the cases where more than two admissible controls exist.

The main advantage of this method over others is that the learning process is short and that the memory requirements during the learning process, and afterwards during actual operation of the system, are quite small. When the learning process is terminated, all the information necessary for controlling the plant is contained in the M-component weight vector $\underline{W}$ of the discriminant function. Large memory requirements for self-organizing systems were some of the main difficulties of other methods proposed.

```
C     FORTRAN CODES FOR PROGRAMS I AND II AS USED IN EXAMPLE 1 .

C     PROGRAM 1 - PATTERN COLLECTING AND TRAINING PROCEDURE

$FMS
*     FORTRAN
      COMMON Y,W, U, DIPS, DDIP, MVALUE, AL, M, N
      DIMENSION Y(4,151)
      DIMENSION X(2,151)
      DIMENSION W(4)
      DIMENSION U(151)
      DIMENSION DIPS(151)
      DIMENSION DDIP(151)
      DIMENSION MVALUE(5)
      DIMENSION AL(5)
      DIMENSION VALUE(5)
      DIMENSION FRAC(5)
      DIMENSION SIPS(151)
      EQUIVALENCE ( W(1), W1 )
      EQUIVALENCE ( W(2), W2 )
      EQUIVALENCE ( W(3), W3 )
      EQUIVALENCE ( W(4), W4 )
      DISCRIF(A,B,C,D)=W1*A+W2*B+W3*C+w4*D
      READ 1, SP,DELTA,X(1,2),X(2,2),(W(K),K=1,4)
    1 FORMAT (F5.3/F5.2/2F10.2/4F10.2)
      READ 2, (MVALUE(I),I=1,5), (AL(I),I=1,5), (VALUE(L),L=1,5), (FRAC(
     1L),L=1,5)
    2 FORMAT(5I4,5F8.4/5F8.3,5F5.3)
      MOR=0
      M=10
      KILL=1
      U(1)=1.0
      U(2)=1.0
      SIPS(1)=0.0
      DIPS(1)=0.0
      DDIP(1)=0.0
      P=SP
      DO 3 J=1,4
    3 Y(J,1)=0.0
      DO 100 N=2,150
      IF(X(1,N)**2+X(2,N)**2) 101,101,4
    4 Y(1,N)=X(1,N)
      Y(2,N)=X(2,N)
      Y(3,N)=X(2,N)**2
      Y(4,N)=X(2,N)**3
      IF(U(N)) 5,6,6
    5 X(1,N+1)=-0.5*P**2+X(2,N)*P+X(1,N)
      X(2,N+1)=-P+X(2,N)
      GO TO 7
    6 X(1,N+1)=0.5*P**2+X(2,N)*P+X(1,N)
      X(2,N+1)=P+X(2,N)
    7 SIPS(N)=SQRTF(3.0*X(1,N)**2+4.0*1.732*X(1,N)*X(2,N)+7.0*X(2,N)**2)
      DIPS(N)=SIPS(N)-SIPS(N-1)
      DDIP(N)=DIPS(N)-DIPS(N-1)
      U(N+1)=-SIGNF(1.0,DDIP(N))*U(N)*ABSF(U(N-1))
      IF(U(N+1)) 9,8,9
    8 U(N+1)=U(N)
    9 IF(U(N+1)-U(N)) 10,12,10
   10 IF(DDIP(N)-DDIP(N-1)) 8,8,11
   11 U(N-1)=0.0
      U(N)=0.0
   12 GO TO (15,14,13,13),KILL
   13 U(N-1)=0.0
   14 U(N)=0.0
      KILL=1
   15 IF(X(1,N+1)**2+X(2,N+1)**2-DELTA) 20,20,16
   16 IF(ABSF(X(1,N+1))-20.0) 17,18,18
   17 IF(ABSF(X(2,N+1))-20.0) 24,18,18
   18 PRINT 19, N
   19 FORMAT(1H ,12HNEW DATA AT I3)
      GO TO 22
   20 MOR=MOR+1
      PRINT 21, N,(W(K),K=1,4)
   21 FORMAT(1H ,13HAT ORIGIN AT I3,22H PLUS ONE WITH WEIGHTS 4F10.2)
   22 KILL=KILL+2
      READ 23,X(1,N+1),X(2,N+1)
   23 FORMAT(2F10.3)
      GO TO 28
   24 IF(N-M) 100,26,26
   26 M=M+10
   28 CALL CORECT
      DO 30 L=1,5
      IF(X(1,N+1)**2+X(2,N+1)**2-VALUE(L)) 31,30,30
   30 CONTINUE
      L=5
   31 P1=P
      P=FRAC(L)*SP
      IF(P-P1) 32,100,32
   32 KILL=KILL+1
  100 CONTINUE
  101 PRINT 102
  102 FORMAT(1H1,53X,13HFINAL RESULTS///)
      PRINT 103,P,DELTA,(W(K),K=1,4),MOR
  103 FORMAT(22H SAMPLING PERIOD T IS F6.3,7H SECOND/42H SQUARE DISTANCE
     1 FROM ORIGIN DELTA EQUALS F4.2/9H WEIGHTS 4F10.2/29H NUMBER OF TIM
     2ES NEAR ORIGIN I3/)
      PRINT 104, ((X(K,N),K=1,2),U(N),SIPS(N),DIPS(N),DDIP(N),N,N=2,
     1150)
  104 FORMAT(1H1,9X,4HX(N),9X,4HU(N),5X,7HSIPS(N),3X,7HDIPS(N),4X,7HDDIP
     1(N),8X,1HN//(1H ,6F10.2,I10))
      PRINT 105
  105 FORMAT(1H1,25HTRIAL OPERATION OF SYSTEM//)
      KTR=0
      P=SP
  110 READ 111, X(1,1),X(2,1)
  111 FORMAT (2F10.3)
      IF(X(1,1)**2+X(2,1)**2) 112,250,112
  112 DO 200 N=1,200
      Y(1,N)=X(1,N)
      Y(2,N)=X(2,N)
      Y(3,N)=X(2,N)**2
      Y(4,N)=X(2,N)**3
      U(N)=SIGNF(1.0,DISCRIF(Y(1,N),Y(2,N),Y(3,N),Y(4,N)))
  113 IF(U(N))114,116,118
  114 X(1,N+1)=-0.5*P**2+X(2,N)*P+X(1,N)
      X(2,N+1)=-P+X(2,N)
      GO TO 120
```

```
116 U(N)=-U(N-1)
    GO TO 113
118 X(1,N+1)=0.5*P**2+X(2,N)*P+X(1,N)
    X(2,N+1)=P+X(2,N)
120 IF(X(1,N+1)**2+X(2,N+1)**2-DELTA) 122,122,200
122 PRINT 124, N,X(1,1),X(2,1)
124 FORMAT(1H ,13HAT ORIGIN AT I3,29H FROM INITIAL CONDITION X(1)=F10.
   13,6H X(2)=F10.3)
    KTR=KTR+1
    GO TO (110,110,110,250),KTR
200 CONTINUE
250 CALL EXIT
    END
```

```
C      SUBROUTINE TO PROGRAM 1 - ITERATIVE CORRECTION PROCEDURE
*      FORTRAN
       SUBROUTINE CORECT
       COMMON Y,W, U, DIPS, DDIP, MVALUE, AL, M, N
       DIMENSION Y(4,151)
       DIMENSION W(4)
       DIMENSION U(151)
       DIMENSION DIPS(151)
       DIMENSION DDIP(151)
       DIMENSION MVALUE(5)
       DIMENSION AL(5)
       DIMENSION MIS(150)
       EQUIVALENCE ( W(1), W1 )
       EQUIVALENCE ( W(2), W2 )
       EQUIVALENCE ( W(3), W3 )
       EQUIVALENCE ( W(4), W4 )
       DISCRIF(A,B,C,D)=W1*A+W2*B+W3*C+W4*D
       SQNORMF(A1,A2,A3,A4)=A1**2+A2**2+A3**2+A4**2
       ITERAN=0
       LAST=0
     2 DO 3 I=1,5
       IF(M-MVALUE(I)) 4,3,3
     3 CONTINUE
       I=5
     4 JNUCO=0
     6 DO 15 ITI=1,148
       NT=N-ITI
       IF(NT-1) 16,16,7
     7 IF(U(NT)*DISCRIF(Y(1,NT),Y(2,NT),Y(3,NT),Y(4,NT))) 8,10,10
     8 C=AL(I)*ABSF(DISCRIF(Y(1,NT),Y(2,NT),Y(3,NT),Y(4,NT)))/SQNORMF(Y(1
      1,NT),Y(2,NT),Y(3,NT),Y(4,NT))
       DO 9 K=1,4
     9 W(K)=W(K)+U(NT)*C*Y(K,NT)
       GO TO 15
    10 JNUCO=JNUCO+1
    15 CONTINUE
       ITI=149
    16 ITERAN=ITERAN+1
       IF(JNUCO-ITI+1) 17,30,17
    17 GO TO (4,4,4,4,4,4,4,4,4,18),ITERAN
    18 DO 25 JET=1,148
       NT=N-JET
       IF(NT-1) 30,30,19
    19 IF(U(NT)*DISCRIF(Y(1,NT),Y(2,NT),Y(3,NT),Y(4,NT))) 20,25,25
    20 MIS(NT)=NT
       PRINT 21,MIS(NT)
    21 FORMAT(1H ,26H UNCORRECTABLE PATTERN AT I3)
       U(NT)=0.0
    25 CONTINUE
       ITERAN=ITERAN-1
       LAST=LAST+1
       GO TO (4,4,30),LAST
    30 RETURN
       END
*      DATA
```

```
C       PROGRAM 2 - GRADIENT TECHNIQUES
$EMS
*       FORTRAN
        COMMON X,W,J,M,P
        DIMENSION  X(4,200)
        DIMENSION W(4)
        DIMENSION DELTW(4)
        DIMENSION DW(4)
        DIMENSION J(6)
        DIMENSION GRAD(4)
        EQUIVALENCE ( W(1), W1 )
        EQUIVALENCE ( W(2), W2 )
        EQUIVALENCE ( W(3), W3 )
        EQUIVALENCE ( W(4), W4 )
        NORMF(A1,A2,A3,A4)=SQRTF(A1**2+A2**2+A3**2+A4**2)
        READ 2, (W(K),K=1,4),SP,DELTA
      2 FORMAT(4F10.5/F5.3/F6.2)
        PRINT 3
      3 FORMAT(1H1,26X,7HGRAD(K),46X,4HW(K),28X,5HINDEX//)
        DO 100 INDEX=1,200
        READ 4,X(1,1),X(2,1)
      4 FORMAT(2F10.3)
        IF(X(1,1)**2+X(2,1)**2) 110,110,5
      5 M=1
        P=SP
        GRAD(1)=0.0
        CALL DRIVER
     30 DO 40 K=2,4
        DELTW(K)=0.2*W(K)
        W(K)=W(K)+DELTW(K)
        CALL DRIVER
     35 FL=J(M-1)-J(1)
        GRAD(K)=P*FL/DELTW(K)
        W(K)=W(K)-DELTW(K)
     40 CONTINUE
        DSIG=0.1*NORMF(GRAD(1),GRAD(2),GRAD(3),GRAD(4))
        DO 60 K=2,4
        DW(K)=-GRAD(K)*DSIG
     42 IF(ABSF(DW(K))-0.2*ABSF(W(K))) 45,45,43
     43 DW(K)=0.5*DW(K)
        GO TO 42
     45 IF(ABSF(DW(K))-0.10E-06) 46,47,47
     46 DW(K)=0.10E-06*SIGNF(1.0,DW(K))
     47 W(K)=W(K)+DW(K)
     60 CONTINUE
        PRINT 62, (GRAD(K),K=1,4),(W(K),K=1,4),INDEX
     62 FORMAT(1H ,4E15.6,4E15.6,I5)
    100 CONTINUE
    110 CALL PLOT
        CALL EXIT
        END
```

```
C       SUBROUTINE TO PROGRAM 2 - SIMULATING BEHAVIOR OF PLANT
*       FORTRAN
        SUBROUTINE DRIVER
        COMMON X,W,J,M,P
        DIMENSION X(4,200)
        DIMENSION W(4)
        DIMENSION J(6)
        DIMENSION U(200)
        EQUIVALENCE ( W(1), W1 )
        EQUIVALENCE ( W(2), W2 )
        EQUIVALENCE ( W(3), W3 )
        EQUIVALENCE ( W(4), W4 )
        DISCRIF(A,B,C,D)=W1*A+W2*B+W3*C+W4*D
     12 DO 25 N=1,200
        X(3,N)=X(2,N)**2
        X(4,N)=X(2,N)**3
        U(N)=SIGNF(1.0,DISCRIF(X(1,N),X(2,N),X(3,N),X(4,N)))
     13 IF(U(N)) 14,16,18
     14 X(1,N+1)=-0.5*P**2+X(2,N)*P+X(1,N)
        X(2,N+1)=-P+X(2,N)
        GO TO 20
     16 U(N)=-U(N-1)
        GO TO 13
     18 X(1,N+1)=0.5*P**2+X(2,N)*P+X(1,N)
        X(2,N+1)=P+X(2,N)
     20 IF(X(1,N+1)**2+X(2,N+1)**2-DELTA) 26,26,25
     25 CONTINUE
        N=200
     26 J(M)=N
        M=M+1
     30 RETURN
        END
*       DATA
```

REFERENCES

1. K. S. Fu, "Learning Control Systems," Proc. COINS Symposium, June 17, 18, 1963, Evanston, Illinois.

2. M. D. Waltz and K. S. Fu, "A Computer Simulated Learning Control System," 1964 IEEE Convention, New York, New York.

BIBLIOGRAPHY

Aizerman, M. A., "Automatic Control Learning Systems," Proc. of the International Federation of Automatic Control Congress, Fall, 1963.

Andrew, A.M., "Learning in Control Systems," Control, Sept. 1960.

Freeman, H., "On the Digital Computer Classification of Geometric Line Patterns," Proc. of the National Electronics Conference, vol. 18, October 1962.

Fu, K. S., "A Statistical Approach to the Design of Intelligent Machines - Pattern Recognition and Learning," Cybernetic, No.2, 1962.

Fu, K. S., "Introduction to Mathematical Learning Theory", Class Notes EE 681, Spring 1963, Purdue University, Lafayette, Indiana.

Gibson, J. E., "Adaptive Learning Systems", Proc. 1962 National Electronics Conference, Chicago, Illinois.

Gibson, J. E., "Nonlinear Automatic Control", Chapter 11, McGraw-Hill, New York, 1963.

Group Report No.1, "A Philosophy and A Review of the State of the Art of Learning Control Systems," Control and Information Systems Laboratory, School of Electrical Engineering, Purdue University, Lafayette, Indiana.

Mesarovic, M. D., "Self Organizing Control Systems" Symposium on Discrete Adaptive Processes, 1962, JACC, New York.

Mishkin, E. and Braun, L., "Adaptive Control," McGraw-Hill, New York, 1963.

Nilsson, N. J., "Introduction to the Theory of Trainable Pattern Classifying Machines," Stanford Research Institute, Menlo Park, California, March 1964.

Sebestyen, G. S., "Pattern Recognition by an Adaptive Process of Sample Set Construction," Trans. IRE on Information Theory, vol. IT - 8, September 1962.

Slansky, J., "Adaptation and Feedback," Symposium on Discrete Adaptive Processes, 1962, JACC, New York.

Widrow, B., "Adaptive Sampled Data Systems," Tech. Rept. No. 2104-1, Stanford Electronics Laboratories, Stanford University, Stanford, California, July 15, 1960.

Widrow, B., "Pattern Recognition and Adaptive Control," Symposium on Discrete Adaptive Processes, 1962, JACC, New York.