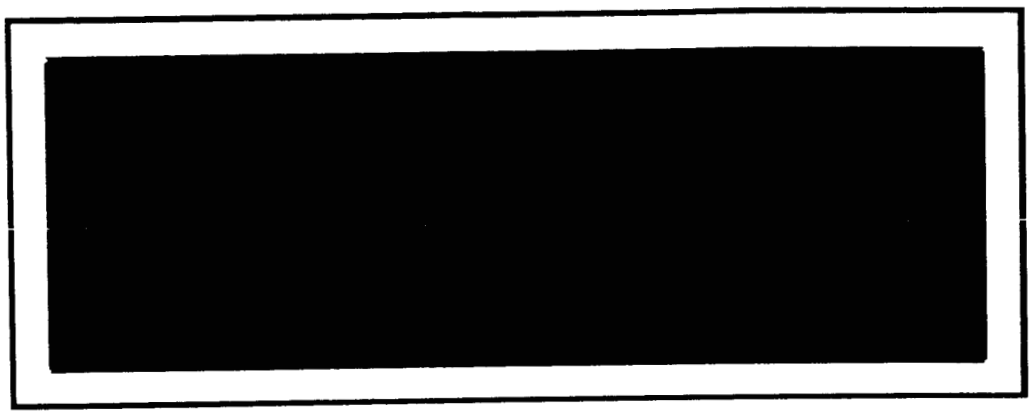


MELPAR INC

A SUBSIDIARY OF WESTINGHOUSE AIR BRAKE COMPANY

7700 ARLINGTON BOULEVARD, FALLS CHURCH, VIRGINIA 22046

HUMAN PERFORMANCE CONTROL

MONITORING SYSTEM

Final Report

Under Contract No. NASW-1085

Prepared by

Melpar, Inc.
7700 Arlington Boulevard
Falls Church, Virginia

March 1966

FOREWORD

This report was prepared by Melpar, Inc., Falls Church, Virginia, on NASA Contract NASW-1085. The work was under the direction of the Life Support and Protective System of NASA Headquarters. Mr. L. Anderson was Project Monitor for the National Aeronautics and Space Administration.

The work represented by this final report began in January 1965 and covered a 14-month period ending in February 1966.

The project was conducted in the Controls Section of the Computer Laboratory under the supervision of Mr. E. M. Connelly. Mr. R. E. Mirabelli was the Principal Investigator and was assisted by Mr. J. E. Whelchel, Miss J. M. Gervinski, and Mr. S. Kissin (consultant).

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

1. INTRODUCTION

This report describes the work done under Contract No. NASW-1085. The contract was divided into three phases. During the first phase, theoretical studies were performed and a mathematical model of a performance control and monitoring system was developed. Applications for trainable logic were developed in the areas of computation and controls. The first phase terminated with the presentation of three application problems which were designed to illustrate facets of the theoretical work.

The second phase of the program started with the selection of one problem by NASA and subsequent development of a computer program. The program simulated a second-order servo controlled by an adaptive logic element trained by monitoring of human performance.

During the final phase of the program the system behavior was observed through experimentation. The success of this approach indicated that the method might indeed be applicable to more complicated systems.

2. RESULTS OF THEORETICAL STUDIES

2.1 Performance Control Systems

2.1.1 General

A performance variable associated with a given system is taken to be simply an attribute of that system. It is distinguished from other attributes of the system in that there is always a known "optimum" range of values for this variable. Most frequently, interest is centered upon the observation and control of this performance. A performance vector is an ordered set of such performance variables. In this light, performance of an aircraft system could correspond to its vector rms deviations from a given flight trajectory. Likewise, the performance of an environmental control system could correspond to the partial pressure deviations from a given temperature-dependent norm.

In general, performance is a function of various random variables. It is itself, therefore, a random variable. Hence, statistical techniques can be applied to establish the properties of this performance. Frequently, control must be indirect, involving prediction or estimation of performance, experimentation, and control decisions.

A block diagram of one such performance control system is provided for illustration in figure 1. This system involves man and machine. The system behavior is subject to control inputs which determine its performance at time t_n. Other attributes of the system can be measured at time t_n, designated as the measurement vector

$$\vec{x}(t_n) = \left[x_1(t_n),\ x_2(t_n),\ \ldots,\ x_r(t_n)\right]$$

The basic assumption is that future performance at time t_{n+1} is some function of these attributes at time t_n, $\vec{x}(t_n)$; i.e., the assumption is that the performance variable can be predicted. The prediction technique employed can be made adaptive, by an updating procedure shown as feedback to the performance prediction.

When the predicted performance falls out of tolerance, control-determinating experiments can be implemented. This would involve a change in the control parameters that would actually change the system parameters, $x(t_n)$, whereby the predicted performance would be changed. Modification could continue until the predicted performance falls back within tolerance. A more probable approach would be to perform experiments which establish distributional properties of $x(t_n)$, thus restricting the class of modifications that are actually implemented. These can be selected on the basis of statistical theory.

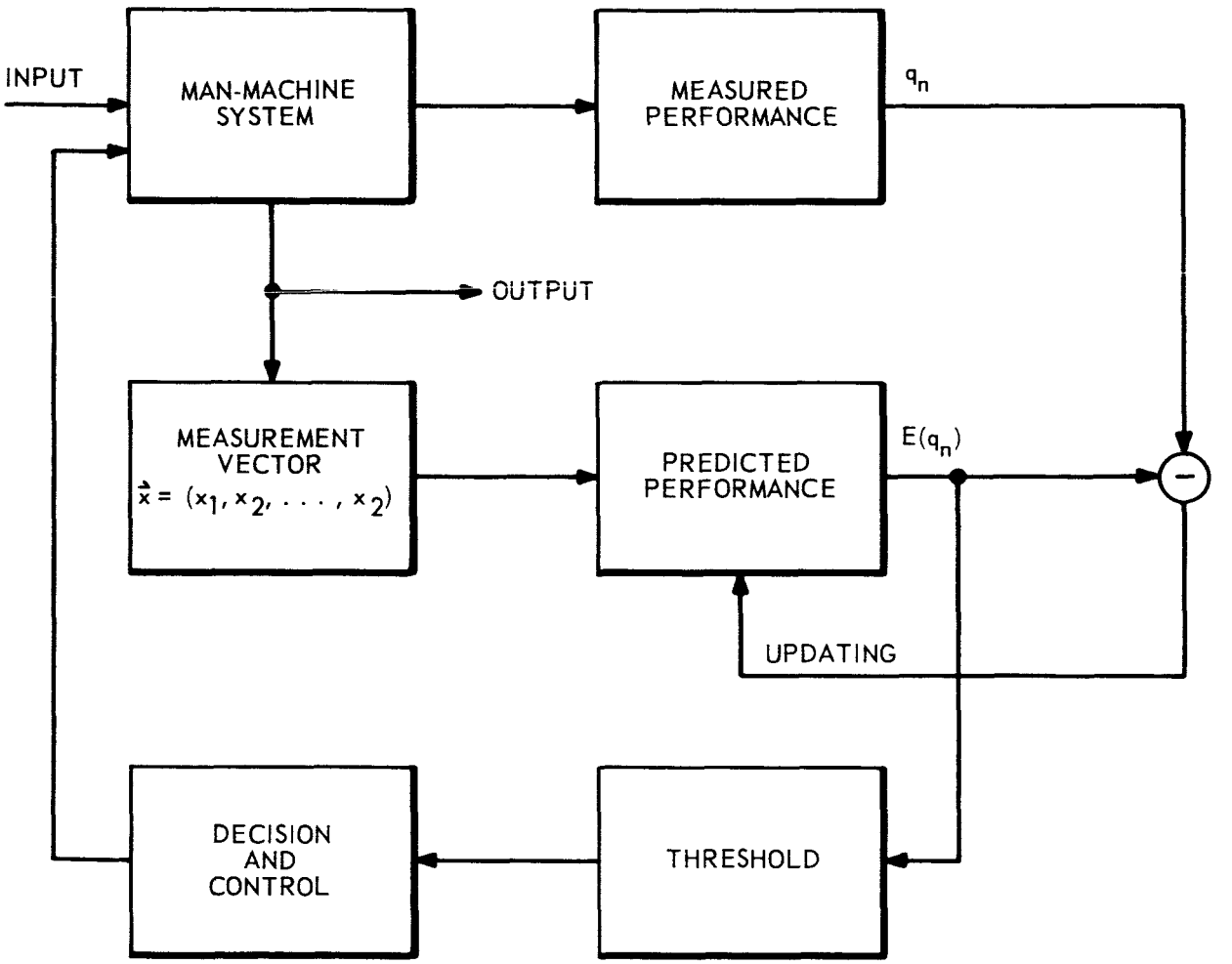


Figure 1. A Performance Control and Monitoring System

The major elements of the above-described performance control system involve adaptive prediction and decision theory. These are considered in greater detail below.

2.1.2 Prediction and Estimation

Various techniques for establishing the performance prediction relationship

$$q_{n+1} = q\left[\vec{x}(t_n)\right]$$

have been developed and are described in the literature. One of these would consider performance, q, as being a discrete valued function; i.e.,

$$q \in q^i, \text{ for } i = 1,2, \ldots,n$$

The r-components of the system measurement vector, $\vec{x}(t)$, would be viewed as the coordinates of a point in r-dimensional space. If one knows the distributions, $p(\vec{x}|q)$, the a priori probabilities, p(q), and the loss matrix (corresponding to an estimate of the relative penalty associated with assigning performance q^i when q^j should be assigned), then the selection of q for a given measurement $\vec{x}$ can be made on the basis of minimum expected loss.[1-6] Other selection criteria can be used, however.

The technique is termed adaptive when either the required distributions, $p(\vec{x}|q)$, or the loss matrix must be obtained from the incoming data. (The a priori probabilities, p(q), may or may not be known.) If measured performance were noise free (i.e., the same as performance), then the problem becomes one of "supervised" learning. More generally, however, measured performance is a random variable about whose distribution little may be known. This makes the problem here much more difficult.[7] Most of the work available avoids some of this difficulty by assuming measured performance to be normally distributed. Futhermore, the form of the distributions, $p(\vec{x}|q)$, is generally assumed to be known. This last restriction could be removed in many cases, however.

Various alternative approaches can be applied to adaptive performance prediction. A familiar curve-fitting technique is described in appendix A, as applied to prediction. Here, the performance, E(q), is represented by some given functional form

$$E\left[q(t_{n+1})\right] = f\left[\vec{x}(t_n), \vec{\theta}(t_n)\right]$$

which is linear in $\vec{\theta}$, where $\vec{x}$ is the measurement vector (an r-tuple) and $\vec{\theta}$ is a vector of unknown parameters (an S-tuple). These parameters are selected so as to minimize the sum of the squares of the deviations of measured versus predicted performance over the discrete time variable. If a fixed set of θ parameters were desired, the weights would be set to unity. Values less than unity permit time variation in the $\vec{\theta}$ parameters. The updating procedure is established by a very simple iterative process, and a well-known theorem is applied to discuss the distributional properties of the parameter. This distribution over the predicted performance is important in forming control decisions, as will be seen in the next section.

2.2 Decisions and Decision Criteria

The simplest form of decision consists of selecting an action from a set of alternative actions with perfect information about the various consequences. Formally, one assumes a set of states-of-nature $\Omega=(\omega_1, \omega_2, \ldots, \omega_r)$ and a set of possible actions $A=(a_1, a_2, \ldots, a_s)$. A loss matrix is assumed, $\left[L(i,j)\right]$, whose elements $L(i,j)$ are the loss associated with selecting action a_i while nature is in state ω_j. With perfect information, one knows both the state-of-nature and the loss matrix. A rational decision would be to select that action which yields the least loss.

With less perfect information, one might be restricted to knowing the cost matrix and only the a priori probabilities of nature being in state j, p(j), for j = 1,2, ..., r (rather than the actual state of nature). In such a case one might make a selection on the basis of its yielding, on the average, the minimum loss; i.e., the expected loss associated with selecting action i is given by,

$$\rho(i) = \sum_{j=1}^{r} L(i,j)\ p(j)$$

wherein one would select that action which minimizes $\rho(i)$.

Consider now the case where one is given the loss matrix, $\left[\ L(i,j)\ \right]$, but has no information on the probabilities over the states-of-nature. One method of establishing a decision is sometimes employed in game theory. Here, one selects an action from a probability distribution over the available actions. This distribution over the available actions is established so that, on the average, the maximum loss (sustained for any possible distribution over the states-of-nature) will be minimized.

A more useful class of decision problems extends the above considerations to include information obtained from experiments or observations.

These are used to modify the established probability distribution over the states-of-nature. For example, let the observation be some parameter (or vector) $\vec{x}$. Let the conditional probabilities $p(\vec{x}|j)$ be known for each state-of-nature, j. Let the a priori probability that nature is indeed in state j, p(j), be also known. As before, let the loss matrix be given by $[L(i,j)]$, where the index i ranges over the set of possible actions and j ranges over the set of possible states-of-nature. For such problems, decisions are now based upon the observation, $\vec{x}$. In fact, a decision rule is defined as any function that maps the observation $\vec{x}$ into action i,

$$i = d(\vec{x})$$

The Bayes decision criterion (applied against the a priori distribution over the states-of-nature) is one that yields the minimum loss on the average; i.e., it is a criterion for selecting a decision rule which minimizes the average loss. Its name is derived from the use of the Bayes theorem in probability, which is used to derive these decisions. It can be readily shown[8] that this minimum average loss criterion implies that one should select action, i, when

$$\sum_{j=1}^{r} = L(i,j)\ p(\vec{x}|j)\ p(j) \leq \sum_{j=1}^{r} L(k,j)\ p(\vec{x}|j)\ p(j)$$

for all possible actions, k. (This is essentially the criterion mentioned in discussing prediction and estimation in n-dimensional space.)

There are various other criteria that can be employed. One of these is the Neyman-Pearson criterion, as generalized to cover cases of more than two possible actions. With only two possible actions, say 1 and 2, the decision-maker can hypothesize that action 1 is called for. He could then test this hypothesis and make two different kinds of errors. An error of the first kind would be made if his observation, $\vec{x}$, led him to select action 2 when action 1 was called for (i.e., when he falsely rejects his hypothesis). An error of the second kind would be made if his observation, $\vec{x}$, led him to select action 1 when action 2 was called for (i.e., when he falsely accepts his hypothesis). Whereas it is desired to minimize the probability of both of these errors, this is not, in general, possible. Normally, the decision rule which decreases the probability of one of these errors will increase the probability of the other type of errors.

The Neyman-Pearson criterion calls for selecting the decision rule (function, d(x), which maps our observations into a selected action) which

minimizes the probability of an error of the second kind, subject to the restriction that the probability of an error of the first kind remains below some preassigned value.[9] The generalization for the case of more than two actions can be accomplished in several ways. In one of these,[10,11] the probability of correct decisions is maximized, while the probability of certain incorrect decisions is constrained to being less than, or equal to, some preassigned constants.

When more than one observation or experiment can be made, it becomes important to establish a criterion for stopping the process of experimentation (or observation) as well as the decision to be made once this has stopped. This is referred to as sequential decision theory and analysis.

Wald[12] developed the sequential probability ratio test (SPRT) for binary-type decisions (accept or reject a hypothesis), terminating experimentation at some point beyond which a Neyman-Pearson type of criterion is satisfied; i.e., numbers corresponding to the acceptable maximum probability of errors of the first and second kinds are first selected. Experimentation ceases and a decision is made only when these conditions are satisfied by one of the two possible actions. His generalization of this test to multivalued decision functions was made on the basis of minimizing the risk of making a wrong decision.

The Bayes sequential decision model postulates a given set of experiments (or observations), which can only be performed in the given order (i.e., experiment i must precede experiment i + 1). These experiments can be similiar to one another or completely different. If the set of experiments is finite, then it is called a truncated sequential theory. As with nonsequential Bayes decisioning, a loss matrix, $[L(i,j)]$, a set of a priori probabilities, $p(j)$, and a set of conditional probabilities, $p(\underset{\sim}{x}|j)$, is presumed. The nature of the observation vector, $\underset{\sim}{x}$, is that of including measurements made by all the experiments. Hence, decisions made on the basis of, say, n-experiments can only use the conditional probability of observation vectors whose first n-coordinates <u>only</u> are known; i.e., one must average the expected loss over all coordinates corresponding to experiments which have not yet been performed. The last requirement is to place a cost on each experiment which, in general, depends upon the outcome of the experiments.

The solution can be shown[8] to be obtainable by taking a dynamic programming type of approach and working backwards. Essentially, experimentation is to be continued only when the current Bayes risk (established, as with nonsequential decisions, to be the average loss anticipated on the basis of

current estimations of the state-of-nature) is greater than the expectation of loss if experimentation continues. The detailed solution is given in appendix C.

2.3 Computational Techniques

2.3.1 Introductory Discussion

There are several areas in which trainable logical networks (TLN) apply to performance control and monitoring systems similar to that described in the preceding section. Its utilization rests upon certain key properties of such networks. One property is that they are finite state devices whose only memory consists of what state it is currently in. Another property is that they can be configured to behave as stochastic devices which can be analyzed as a Markov process. It appears natural, then, to consider their application to such computational techniques as Monte Carlo and Simulation.

Other uses of TLN, however, have been studied previously. One of these was used to obtain high reliability in systems. Another use involved them as control elements. This latter work, although not described here, will be considered relative to its application to the selected study problem. This section will concentrate on the use of TLN's for computation.

The basic element of the TLN, referenced as SOBLN, is a k-level statistical switch. This is simply a switch which can attain any of k-states. Each of these states corresponds to a probability of the switch being closed. It is this element which is fundamental to the computation process described below. The fact that these devices are so flexible, so amenable to high-reliability considerations, and consist of this common element (wherein it is amenable to concepts of microminiaturization) provides reason for the investigation of their utility as basic computing elements as well as their use in problems amenable to solution by other than Monte Carlo techniques.

2.3.2 Basic Arithmetic Operations Using Statistical Switches

There are several methods for implementing the statistical switch to perform the multiplication and division of two numbers. The first method consists of converting both numbers into proper fractions by a scaling operation. Each number is then associated with a probability setting of a k-level statistical switch. The outputs of the switches are sent through an AND gate (alternatively, the switches may be merely placed in series). Since an n-bit counter is the basic element of a statistical switch, the k-level switch is one that is capable of taking on $k = 2^n$ different probability settings.

A Monte Carlo process is initiated with some number of samples, N, taken for convenience to be a power of 2 (say 2^m). The 1 outputs from the above-referenced AND gate will increment an m-bit counter. For a large

enough m, one would expect that approximately

$$P_1 P_2 = \frac{\text{(number of 1's present in the counter)}}{2^{-(m-2n)}}$$

where P_i is the bias setting. To obtain the original, one merely shifts the counter m-2n bit positions to the right. This is a scaling operation which, in effect, corresponds to multiplying by the square of the scale factor in the denominator. Since the switches are independent, the AND function is represented by

$$P(AB) = P(A)\,P(B)$$

The accuracy in this Monte Carlo computation can be analyzed on the basis of the variance of a binomial distribution, given as

$$\sigma^2 = NPq$$

where $q = 1 - P$, and where N is the number of independent samples.

To illustrate this process, consider the following example: Let $N = 1024$. The product of $3 \cdot 5 = 15$ could be performed with k-level switches, where

$$k = 2^n = 2^5 = 32$$

Then,

$$P(A) = \frac{3}{32} \text{ and } P(B) = \frac{5}{32}$$

$$P(AB) = P(A)\,P(B) = \frac{15}{1024}$$

Since $m = 2n$, there would be no shift of the counter after the end of the N-samples. In general, however, one would anticipate much larger values for m, which would require the above-described scaling shift.

The division of a scaler by another scaler can be readily performed by modifying the k-level switch to include a decoder which resets the n-bit counter at any integer, r, where $r<2^n$. Thus, for the operation a/r, where r>a, we merely

use the foreshortened counter k-level switch, with reset occurring on the rth pulse and with the bias level set at a.

2.3.3 Partial Differential Equations

One of the capabilities of a SOBLN as a Monte Carlo simulation device is in the solution of linear partial differential equations of the parabolic or elliptic type. The parabolic type will be considered only briefly because of its relevance to random walk and diffusion processes. The Fokker-Planck equation, mentioned later, is a P.D.E. of the parabolic type.

Consider the general second-order linear partial differential equation with two independent variables:

$$a(x,y)\frac{\partial^2 \varphi}{\partial x^2} + b(x,y)\frac{\partial^2 \varphi}{\partial x \partial y} + c(x,y)\frac{\partial^2 \varphi}{\partial y^2} + f(x,y,\varphi_x,\varphi_y) = 0$$

If (the discriminant) $b^2 - 4ac<0$, the equation is of the elliptic type. Such equations commonly represent equilibrium situations, an example of which is the celebrated Laplace equation,

$$\frac{\partial^2 \varphi}{\partial x^2} + \frac{\partial^2 \varphi}{\partial y^2} = 0$$

Diffusion and heat flow equations are of the parabolic type with discriminant $b^2 - 4ac = 0$. They commonly represent situations with unbalanced equilibrium. Two examples of this are the one- and two-dimensional heat flow equations, described by

$$\frac{\partial u}{\partial t} = K\frac{\partial^2 u}{\partial x^2}; \quad \frac{\partial u}{\partial t} = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2}$$

respectively.

The standard approach to the one-dimensional expression is the separation of variables, yielding the solution

$$u = e^{c_1 t}\left[a \cosh t + b \sinh t\right]$$

A very useful means which exists for obtaining the solution of the two-dimensional equations is the Monte Carlo process, where a particle is considered to be undergoing a series of random walks over a two-dimensional lattice with a tallied score corresponding to the state after t-transitions.[14] For an explanation of this process, consider the following situation. Let a particle undergo a series of random walks starting at (x,y) = (0,0) and continuing over the lattice for t-steps with a transition probability at each point (x,y) associated with having the particle move to (x+1,y), (x-1,y) (x,y+1), (x,y-1). Let each of these transition probabilities be equal. Assume, initially, that the transition probabilities are independent of x and y as well as the past history of the particle. This describes a Markov process with xy states and a symmetric transition matrix. The transition matrix has nonzero terms along the two diagonals on either side of the main diagonal, where all nonzero terms are 1/4. The rest of the entries are zero.

At each point on the lattice there is a probability function P(x,y,t) of finding the particle at (x,y) after t-transitions, starting from (0,0). To show the relationship of this process to the heat equation, note that P(x,y,t) must satisfy the difference equation

$$P(x,y,t+1) = 1/4\ P(x+1,y,t) + 1/4\ P(x-1,y,t) + 1/4\ P(x,y+1,t) + 1/4\ P(x,y-1,t)$$

This follows because the particle must have been at one of the above four positions at time, t, to arrive at (x,y) at time t+1. Subtracting P(x,y,t) from both sides and using the expression for second differences,

$$\Delta f(x) = f(x+1)-f(x)$$

$$\Delta^2 f(x) = \Delta\left[f(x+1)-f(x)\right]$$

$$= f(x+2)-2f(x+1)+f(x)$$

Hence,

$$P(x,y,t+1) - P(x,y,t) = P(x+1,y,t) - 1/2\ P(x,y,t) + P(x-1,y,t)$$

$$+P(x,y+1,t) - 1/2\ P(x,y,t) + P(x,y-1,t)$$

$$= 1/4\left\{\left[P(x+1,y,t) -2\ P(x,y,t) + P(x-1,y,t)\right]\right.$$

$$\left. + \left[P(k,y+1,t) -2\ P(x,y,t) + P(x,y-1,t)\right]\right\}$$

This relates the first difference of P, with respect to t, to the second difference, with respect to x,y. For the limiting case of a finer lattice, the above difference equation is similar to

$$\frac{\partial P}{\partial t} = K\frac{\partial^2 P}{\partial x^2} + \frac{\partial^2 P}{\partial y^2}, \quad K = 1/4$$

This is the two-dimensional heat flow equation.

If we keep a tabulation of the number of times the particle appears in each state (x,y) for a range of t and is divided by the sample size, we have an estimate of P(x,y,t). If, instead of starting at (0,0), one starts at a point on the lattice (x,y) where the starting point is determined by a distribution, f(x,y), we have the initial function P(x,y,0) generating a particular solution of the difference equation.

2.3.4 Matrix Inversion

A brief description is presented here of the inversion of a special type of matrix encountered when concerned with the control of Markovian processes. Another procedure with a different implementation is possible, but less desirable, using the resolvent expansion of a matrix. In the section on application problems dealing with Markovian procedures, the need for matrix inversion is avoided by using the iterative solution of a set of equations. Thus, only the storage of a vector is needed rather than a matrix.

We can invert a matrix of the form I-Q, where Q is a stochastic matrix with all elements $q_{ij} \geq 0$, and where $\sum_j q_{ij} < 1$ for all i. To do this, we extend the dimension of Q by attaching a first row and a first column to it. These are selected so that the new matrix, A, will be stochastic. As an example, let

$$Q = \begin{bmatrix} \frac{3}{4} & 0 \\ 0 & \frac{3}{4} \end{bmatrix}$$

One then forms

$$A = \begin{bmatrix} 1 & 0 & 0 \\ \frac{1}{4} & \frac{3}{4} & 0 \\ \frac{1}{4} & 0 & \frac{3}{4} \end{bmatrix}$$

The matrix A will always be stochastic ($\sum_j a_{ij} = 1$ for all i and $a_{ij} \geq 0$). It describes a Markov process which is absorbing, due to the selection of the elements of the first row. Because the eigenvalues of Q are less than unity, $(I-Q)^{-1}$ will exist and can be shown to equal $I + Q + Q^2 + \cdots$. Using analytic techniques such as the Z-transform, one can show (in closed form) the state probabilities for each transition, t. It is shown[15] that these probabilities can be directly related to $(I-Q)^{-1}$. Thus, using the above-defined Q, one obtains

$$(I-Q)^{-1} = I+Q+Q^2+\cdots = \sum_{t=0}^{\infty} \left(\frac{3}{4}\right)^t \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 4 & 0 \\ 0 & 4 \end{bmatrix}$$

This illustrates that we can allow a series of random walks to occur, where, each time the absorbing state is reached, the process is reinitialized. A TLN can utilize 2^N statistical switches as memory devices, and thus have a capability of handling a matrix of dimension N.

The random walk can proceed in either of two ways. For the first method, let one k-level statistical switch represent the transition matrix with a single training rule generating the matrix. This configuration is feasible for lower order matrices that possess a large amount of symmetry and a small number of nonzero elements, such as the class discussed in section 2.3.3. An alternate method is to include N extra statistical switches, j = 1,2, ..., N, to this TLN. This arrangement allows each of these switches to represent a state of the Markov process and to have a separate training rule applied to each switch. Such a training rule is merely a probability distribution to represent the respective row of the transition matrix and is easier to synthesize.

2.3.5 Linear Difference and Differential Equations

We now consider the stability of an autonomous system of linear differential or difference equations. The system could be a vector matrix state space representation of an nth order linear difference equation. In the Markov decision process application, described in section 3.2, the Jacobi point iterative method of computing an optimal policy results in

$$\vec{d}_k = F^k \vec{d}_o$$

where $\vec{d}_o = \vec{x}_1 - \vec{x}_o$, $\vec{d}_k = \vec{x}_{k+1} - \vec{x}_k$, and where $\vec{d}_k$ represents a vector difference which converges to zero as the iterative process converges to a solution. This convergence can only be guaranteed when the latent roots of the matrix F are <1. To show this, consider the difference equation related to the above expression

$$\vec{d}_{k+1} = F\vec{d}_k$$

with given initial state $\vec{d}_o$. If the latent vectors are independent, we can apply a similarity transformation to diagonalize F, as

$$F = P\ D\ P^{-1}$$

where

$$D = P^{-1} F P = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & 0 \\ & & \ddots & \\ & 0 & & \lambda_n \end{bmatrix}$$

where λ_i is the ith latent root of F. Since

$$F^2 = (P D P^{-1})(P D P^{-1}) = P D^2 P^{-1}$$

one notes that

$$F^k = P D^k P^{-1} = P \begin{bmatrix} \lambda_1^k & & & \\ & \lambda_2^k & & 0 \\ & & \ddots & \\ & 0 & & \lambda_n^k \end{bmatrix} P^{-1}$$

Thus, for large k, $\vec{d}_k = F^k \vec{d}_0 \to 0$, and the process converges to a solution. If the characteristic vectors are not independent, then the matrix can always be transformed into a triangular matrix. There exists a linear transformation, R, such that

$$\vec{d}_k = R \vec{y}_k$$

The original equation,

$$\vec{d}_{k+1} = F \vec{d}_k$$

can now be written as

$$R \vec{y}_{k+1} = F R \vec{y}_k$$

or

$$\vec{y}_{k+1} = R^{-1} F R \vec{y}_k$$

so that

$$D' = R^{-1} F R$$

and is the triangular matrix.

Now, note that we can always write D′ in the form

$$D' = D_1 + D_2$$

where D_1 is a diagonal matrix and D_2 is a nil potent matrix (having nonzero elements only to the right of the main diagonal). Expanding D^p by the binomial theorem gives

$$D^p = D_1{}^p + P D_1{}^{p-1} D_2 + \cdots + D_2{}^p$$

Since D_2 is nil potent (all characteristic roots are 0), there exists a "p_2" such that $D_2{}^{p_2} = 0$. It is also evident that for the diagonal terms in D_1, there exists some p_1 whereby $D_1{}^{p_1}$ will also be zero.

Other considerations apply to the continuous time case. Here, the set of differential equations

$$\dot{\vec{x}} = A \vec{x}$$

has solution

$$\vec{x}(t) = e^{At} \vec{x}(o)$$

$$= (I + A + \frac{A^2}{2!} + \frac{A^3}{3!} + \cdots) \vec{x}(o)$$

which can be evaluated in a manner described in the literature.[17]

2.3.6 Computation of the Inverse of the Least Square Recursion Formula

In the recursion relation for updating a least squares estimate, described in the preceding section, an expression of the form

$$P_{k+1}{}^{-1} = P_k{}^{-1} + \vec{\alpha}\vec{\alpha}^T$$

where P_k is known, is encountered, where $P_k{}^{-1}$ is an n-by-n symmetric matrix and $\vec{\alpha}$ is an n-by-1 column vector. Using a matrix inversion lemma,[13] we can represent the above as

$$P_{k+1} = P_k - P_k \vec{\alpha} (\vec{\alpha}^T P_k \vec{\alpha} + 1)^{-1} \vec{\alpha}^T P_k$$

The above expression can be handled on a Monte Carlo basis, since the underlined vector matrix product $\vec{\alpha}^T P_k$ appears twice, and since $P_k\vec{\alpha}$ is the vector transpose of $\vec{\alpha}^T P_k$. Thus, TLN's could perform a single vector matrix product in the fashion described earlier in this report. Then, a dot product operation is performed to yield the expression $(\vec{\alpha}^T P_k)\vec{\alpha}$, using 2n statistical switches in the fashion identical to the first portion of a vector matrix computation.

After incrementing this result by 1, the resulting scalar $(\vec{\alpha}^T P_k\vec{\alpha} + 1)^{-1}$ is set into a single statistical switch consisting of a mod P-counter (described earlier) with numerator 1. The bias probability is $\frac{1}{\vec{\alpha}^T P_k\vec{\alpha} + 1}$. This switch is placed in series with all switches in the TLN to obtain the desired quantity

$$A = \frac{1}{\vec{\alpha}^T P_k\vec{\alpha} + 1} P_k\vec{\alpha}\vec{\alpha}^T P_k$$

The quantity $(P_k\vec{\alpha})(\vec{\alpha}^T P_k)$ is obtained by a vector-vector product that results in a matrix. The elements of this matrix

$$(a_{ij}) = (P_k\vec{\alpha})(\vec{\alpha}^T P_k)$$

can be stored in the switches themselves to minimize the amount of the input-output logic. Since

$$P_{k+1} = P_k - A$$

we can subtract the elements P_{ij} from a_{ij} and change its sign, this operation being performed by a combinational network at each switch.

This technique, combined with the matrix inversion method described earlier, is sometimes a useful supplement to the standard schemes for solving a system of equations.

2.4 Application Problems

At the conclusion of the theoretical work period, three problems were formulated. Each problem illustrated some facet of the theoretical work done during Phase I. The selection of one of the problems by NASA served as a basis for the computer simulation to be described in section 3.

The three problem statements were:

a. Trainable Controller (Problem No. 1): Given that failures and/or changes in plant characteristics have occurred in an automatic control system, can trainable logic be designed to take over the control function by monitoring human performance on manual control of the system?

b. Markovian Process Control (Problem No. 2): Given a man-machine system that is characterized by its being in a finite set of states, let the transition from one state to another state be responsible as a stationary Markov process. Let the transition matrix, describing this operation, depend upon which of a finite set of policies (modes) the system is selected to operate under. We investigate the optimization of system performance through mode control.

c. Bayes Decision Making (Problem No. 3): This problem is the application of trainable logical networks (TLN) to the on-line solution of Bayes decisions. The specific decision problem is the routing of signals along one or more paths.

The decision computer determines what output channels are to be activated according to the lowest cost Bayes criterion. One output path includes an external evaluation device that can modify the cost matrix contained in the decision computer.

The problem selected by NASA for simulation was Problem No. 1, as shown in figure 2.

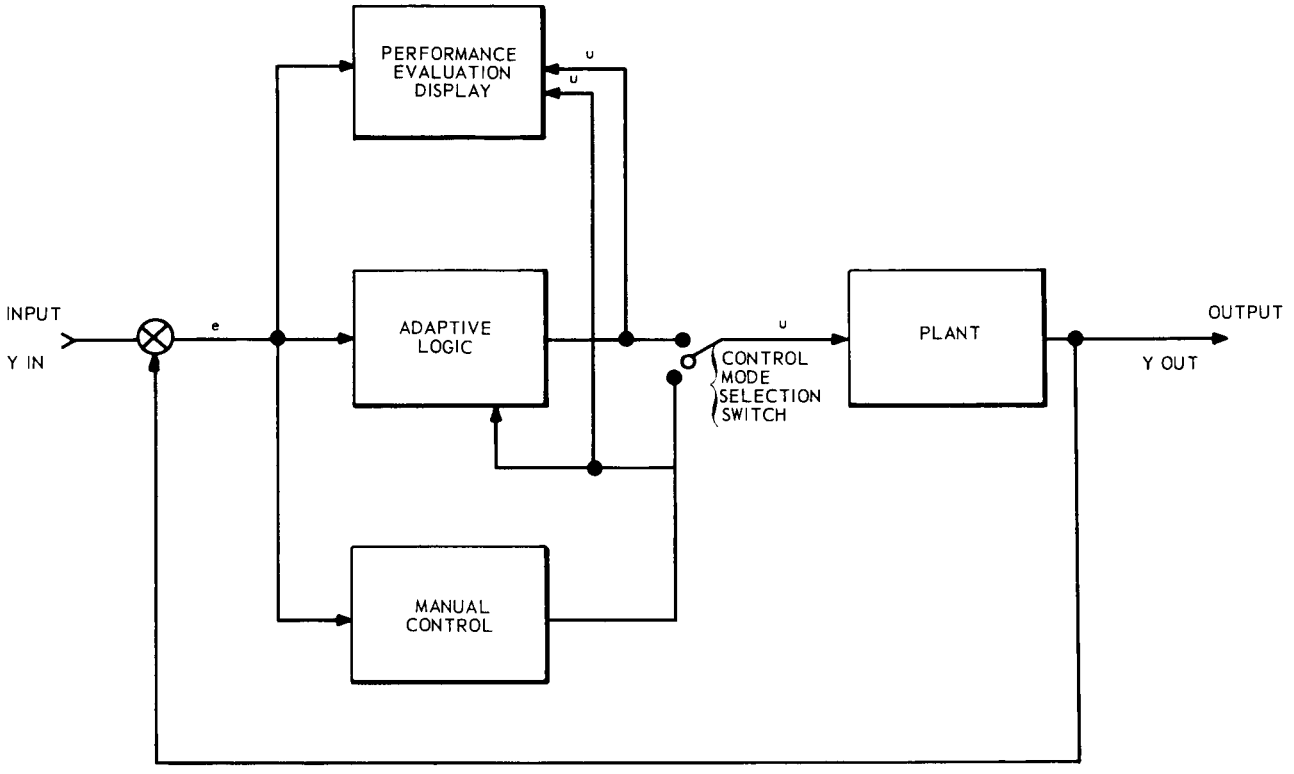


Figure 2. Control Policy Change with Adaptive Logic

3. COMPUTER SIMULATION OF PROBLEM NUMBER 1

3.1 Mathematical Formulation

We may set forth the following framework of the problem in a general state notation. Let

$\bar{x} = x_1\ x_2, \ldots, x_n$ be metered system state variables

$\bar{u} = u_1\ u_2, \ldots, u_m$ be controller policy vector

$\varphi(\bar{u}) \leq 0$ a controller constraint

$G(\bar{x},\bar{u},t) = 0$ a relation between control policy and system variables

$P(\bar{x},\bar{u})$ a performance index which is minimized by proper selection of $\bar{u}(\bar{x})$

Performance generally is marked:

$P(\bar{x},\bar{u})<0$ satisfactory

or

$P(\bar{x},\bar{u}) \geq 0$ unsatisfactory.

For the case we wish to study, we may assume that a policy $\bar{u}(\bar{x})$ has been predetermined such that $P(\bar{x},\bar{u}) < 0$ until a controller failure or plant characteristic change occurs. In the latter case it is necessary to determine a new control policy $\bar{u}^*(\bar{x})$ such that $P(\bar{x},\bar{u}^*)<0$. The new policy is simultaneously determined by the human and transferred to the trainable computer (controller). Additionally, it might be expected that the trainable logic gives some indication to the human when it is ready to take control.

To give more meaning to the general framework, let us specify parameters, constraints, plant equations, costs, etc. Let the plant be a servomotor that is adjusting to command inputs which are step functions. By letting the time intervals between step changes be much greater than the system time constant, the steps can be considered independent in time.

Nature selects any one from a number of plant equations by selecting i and j in the governing differential equation

$$\dddot{y} + a_i \ddot{y} = k_j u_k (\dot{y}, y)$$

After a selection of (i,j), the control problem is to choose k such that a performance index, P, is minimized. As a performance index let us arbitrarily select an index which conserves both fuel and time.

$$P = \int_{t=t_o}^{t=t_f} (C_i |u| + 1)\, dt$$

where i = 1,2, ..., n, and where $t_f - t_o$ is the time required to bring the system output to the input command and C_i is a weighting factor for fuel use.

As a constraint on the controlling policy, let us assume

$$\varphi (u) = |u| - 1 \leq 0$$

and actual permissible values

$$u = (1, 0, -1)$$

Let

starting time	$t = t_o$
input	$z = z_o$
output	$y(t)$
error	$e(t) = z(t_o) - y(t)$
error rate	$\dot{e}(t) = \dot{z}_o - \dot{y}(t) = -\dot{y}(t)$ for step input
control policy	$u_1 = u(\dot{e}, e)$

The differential equation governing error is

$$\ddot{e} + a_1 \dot{e} = k_1 u_1$$

Starting at $t = t_o$ the above variables are:

$$z = z_o$$
$$y = y_o$$
$$e = e_o$$
$$\dot{e} = \dot{e}_o$$

At $t = t_f$, the error, the error rate, and the performance are expected to satisfy the conditions

$$e^2 + \dot{e}^2 \leq C_e$$
$$P \leq P_1$$

A change in the desired control policy occurs when the values of coefficients (a,k,C) are not (a_1,k_1,C_1), and when the corresponding performance threshold is exceeded.

3.1.1 Performance

Thinking of u as a torque-producing parameter and $|u|$ as a rate of fuel consumption, we consider a system which attempts to null its error while minimizing a combination of fuel and time. For a single step input, the functional

$$P(u,t) = \int_{t' = t_o}^{t' = t_f} (C|u| + 1)\, dt'$$

is minimized (where $t_f - t_o$ is the time required to bring the system to the desired output value). By letting the time intervals between step changes be much greater than the system time constant, the steps can be considered independent in time. This being the case, performance may be judged on nulling the error for individual steps. To accomplish this, the function, P, is treated as a cost function and its value is compared with an expected value, E(P). The expected value is:

$$E(P) = \text{minimum cost} + \text{tolerance}$$
$$= \min \int_{t_o}^{t_f} (C|u| + 1)\, d\tau + \gamma$$

where the minimization is over control policy u $(e,\dot{e})$. A warning of performance deterioration is given to the human when

$$E(P) - P \geq 0$$

3.1.2 Control Policy

A control policy, $u(e,\dot{e})$, is a specification of control values (-1, 0, 1) for all points in the error-error rate plane. A convenient method is to divide the phase plane into regions and to specify control values for each region. The proper choice of regions is derived by a laborious computation of switching boundaries for the control variable, u, which minimizes the performance criterion. These boundaries are dependent on both plant parameters and the choice of performance criterion.

A change in the control policy may be brought about either by a change of switching boundaries or by a change of the control values used within the regions defined by the boundaries. It was decided to take the latter approach. The phase plane was divided into more regions than an optimal control policy demands. In addition, the boundaries can be adjusted by input data. The extra regions permit a selection from a larger class of control policies, while the adjustable boundaries permit experiments to be conducted with various values of plant parameters.

When the performance of the automatic control system is judged to be inadequate, the control may be transferred to manual mode.

In the manual mode, adaptive logic monitors the manual control and adapts to an available control policy which most closely resembles that of the human. A block diagram that indicates the flow of information is shown in figure 2. The following section explains the computer implementation of the problem.

3.2 Digital Implementation

The digital program for the Human Performance Control and Monitoring System was written for the SDS 910 computer. The main program is in FORTRAN, and the random number subroutine is in Meta Symbol. A flow diagram of the program is presented in figure 3. A complete list of symbols and the program listings are presented in appendix B.

The program begins by reading in the data for the experiment and setting the system parameters equal to their initial values. The expected performance is computed for the value of y_{in} (desired output) corresponding to TIME = 0, as explained in section 3.1.

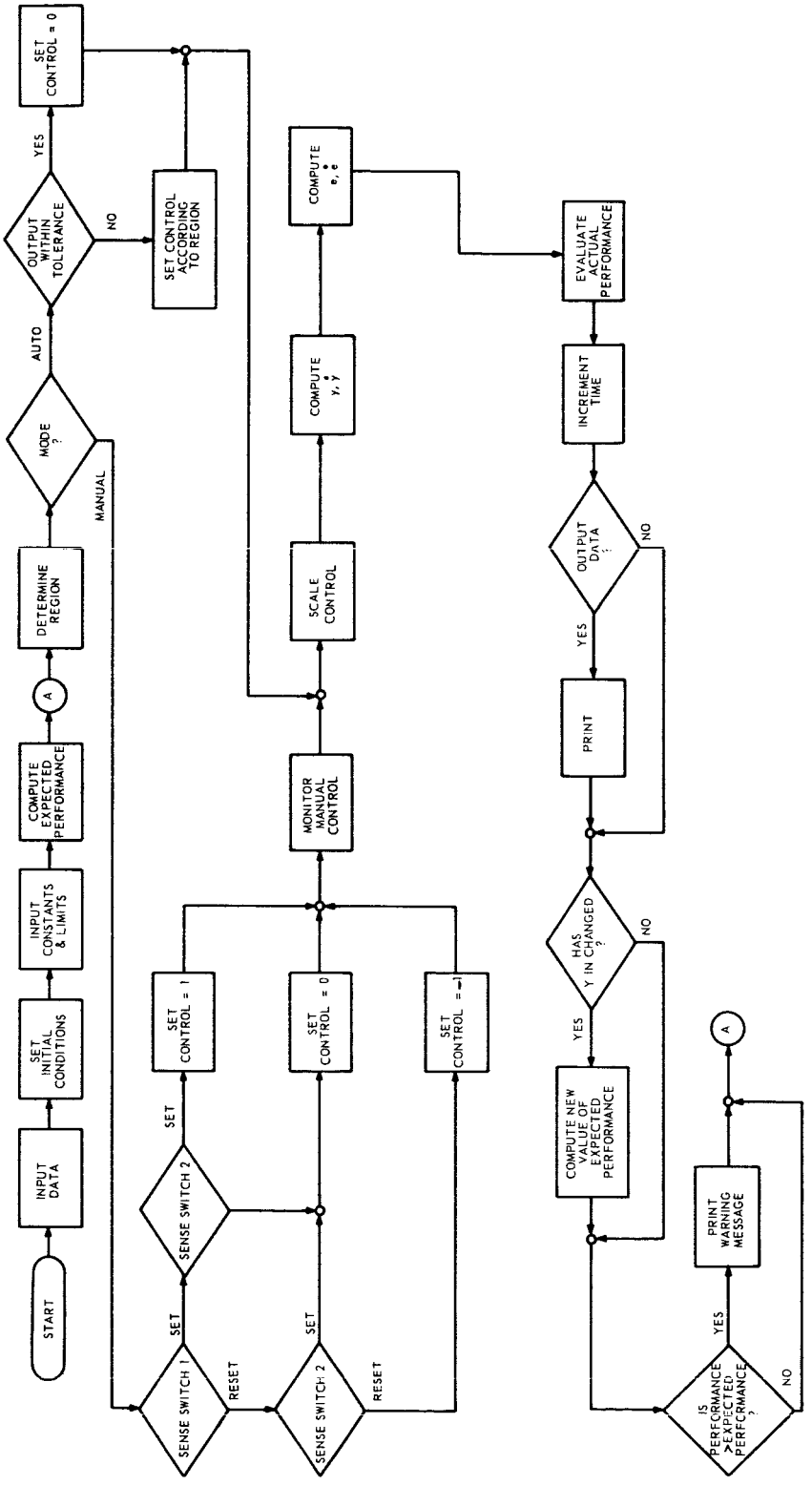


Figure 3. Flow Diagram of Digital Program

The main loop of the program (figure 4) is then completed for each increment of time. The state variables are evaluated as to their position in the phase space, which is presently divided by four straight lines with variable slopes and intercepts and one curve through the origin. This quantizes the space into 32 possible regions. Associated with each region is a control value and a counter that is used when monitoring manual operation. The training takes place by rewarding the counter when the manual control and the control value associated with the region agree, and punishing the counter otherwise. The maximum number of steps in the counter is a variable and is input at the beginning of the experiment. If the counter is decreased to zero, a new random control is generated and is now associated with that region. A specific example of the above procedure follows, where the number of steps needed for training is set at 3.

Time	Manual Control	Trained Control	Counter
t_k	1	1	2
t_{k+1}	1	1	3
t_{k+2}	1	1	3
t_{k+3}	-1	1	2
t_{k+4}	-1	1	1
t_{k+5}	-1	1	0
t_{k+6}	-1	(Random) 0	0
t_{k+7}	-1	(Random) -1	1
t_{k+8}	-1	-1	2
t_{k+9}	-1	-1	3
t_{k+10}	-1	-1	3

A fairly simple method of generating pseudo-random numbers in a binary digital machine was found.[21] For our purpose, the series appears to be generated by random processes. While adequate random numbers were available on punched cards or magnetic tape, they were impractical for our use because of insufficient quantity and slow access. The deterministic method employed is given by the equation

$$R_{n+1} = KR_n \bmod 2^N$$

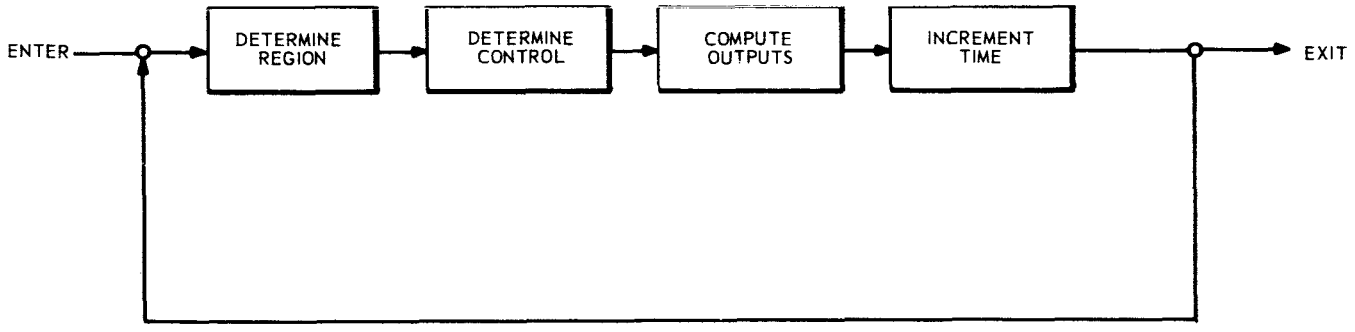


Figure 4. Main Loop of Digital Program

where

R_n = nth random number

R_{n+1} = (n+1)st random number

K = a constant multiplier (the largest odd power of 5 that a 24-bit word will hold)

N = the number of binary digits per word, or 24 in our case

The mod 2^N operation is done by taking K times R_n, and then by setting R_{n+1} equal to the least significant half of the result. It can be shown that, starting with an odd R_o, one will run through 2^{N-2} numbers before repeating a number. Since our random decisions could only take on three values, -1, 1, and 0, only 2 bits of the generated 24 random bits were used per decision, according to the following tabulation.

Random Bits	Decision
0 0	0
0 1	1
1 1	-1
1 0	Not used

This then increases our repeatability factor by 6.

Since four sense switches are available on the SDS 910 computer, it was decided to have SS 4 determine the mode of operation and a combination of SS 1 and SS 2 the control value when in the manual mode. When in the automatic mode, the trained control is used.

SS 4 < Set - Manual mode
SS 4 < Reset - Automatic mode

SS 1	SS 2	Manual Control
Set	Set	1
Set	Reset	0
Reset	Set	0
Reset	Reset	-1

This control value is then altered by the system gain constant, which is input with the initial data.

Straightforward computations, which evaluate the plant equations and the error equations, include:

$$y(t_{k+1}) = \frac{u}{a}\left(\tau - \frac{1}{a}\right) + \frac{\dot{y}(t_k) + a \cdot y(t_k)}{a} + \left(\frac{u}{a^2} - \frac{\dot{y}(t_k)}{a}\right) e^{-a\tau}$$

$$\dot{y}(t_{k+1}) = \frac{u}{a} - \left[\frac{u}{a} - \dot{y}(t_k)\right] e^{-a\tau}$$

$$e(t_{k+1}) = y_{in} - y(t_{k+1})$$

$$\dot{e}(t_{k+1}) = \dot{y}_{in} - \dot{y}(t_{k+1})$$

where

τ = time increment

a = input constant

u = control value

The actual performance is then evaluated where

$$P = \int_{t_o}^{t_k} (C_p|u| + 1)\, dt$$

and checked against the expected performance. Time is incremented, and the data for this loop are output if sense switch 3 is reset. Before repeating the main loop, a check is made to see if the value of y_{in} has changed. If it has, a new value for the expected performance is computed. This process continues until the upper limit of the performance integral is found, which occurs when

$$\dot{e}^2 + e^2 \leq C_e$$

where C_e is a specified constant.

3.3 Experimental Work

The experimental setup was as follows. The computer input data were read in on cards. The data specified the values of constants and the time at which inputs to the control system and the cost weighting of fuel in the performance index would change. The computer output was a typewritten printout. The output consisted of a listing of the following information in eight columns:

a. Time.

b. Control being used.

c. Position.

d. Position rate.

e. Error.

f. Error rate.

g. Performance.

h. Desired position.

A nonzero step input gives rise to an error which the controller must null. With each change in input value, an estimate of expected cost and time of convergence is given.

Convergence occurs when the error and error rate are sufficiently small ($e^2 + \dot{e}^2 \leq 0.01$). If the actual cost exceeded the expected cost before convergence occurred, a message was typed out to indicate that performance was poor.

3.3.1 Choice of Parameter Values

Parameter values were chosen in most instances to help illustrate and emphasize those aspects of the system which were of interest. Where parameters were of limited interest, normalized values were used.

Typical values,

$\Delta T = 0.05$ sec (computation and printout interval)

$a = 1$ (plant damping constant)

$k = 1$ (system gain)

$u = \pm 1, 0$ (torque)

$C_p = 0.1, 2.5, 5.0$ values used in the performance index

$e^2 + \dot{e}^2 \leq 0.01$ is the terminal zone for convergence

The minimum number of time increments necessary to train completely within a phase space control region was set equal to three.

3.3.2 Cost Projection and Choice of Control Regions

The projected cost for nulling an error was obtained by computing the minimum cost and adding a tolerance. The minimum cost calculation, however, did not take into account that the automatic control policy finally adopted had to be selected from a set of admissible controls, possibly none of which minimized the performance function. Figure 5 shows one quadrant of the phase plane with regions defined by switching boundaries. The circle about the origin indicates the terminal region in which no cost is accumulated. The system constants and control torque constrain the plant output rates to less than ±1 rad/second. The weighting given to fuel in the performance index and the permissible switching curves in the phase space were selected such that near optimal policies existed for the performance indices used. An alternative approach would be to complete the expected costs, based upon available control policies. However, if control flexibility is desired, the number of policies to which the controller can be trained must be large. This would result in a prohibitively long computation for the expected cost since each control policy should be examined.

Figures 6 and 7 show the cost incurred when different control policies are used with a given performance index. Performance is shown for four control policies and a minimum fuel cost. Table 1 gives a summary of costs with three different performance indices. Each of the control policies is near optimal for one of the performance parameters (C_p). The four control policies range from a minimum fuel policy to a minimum time policy.

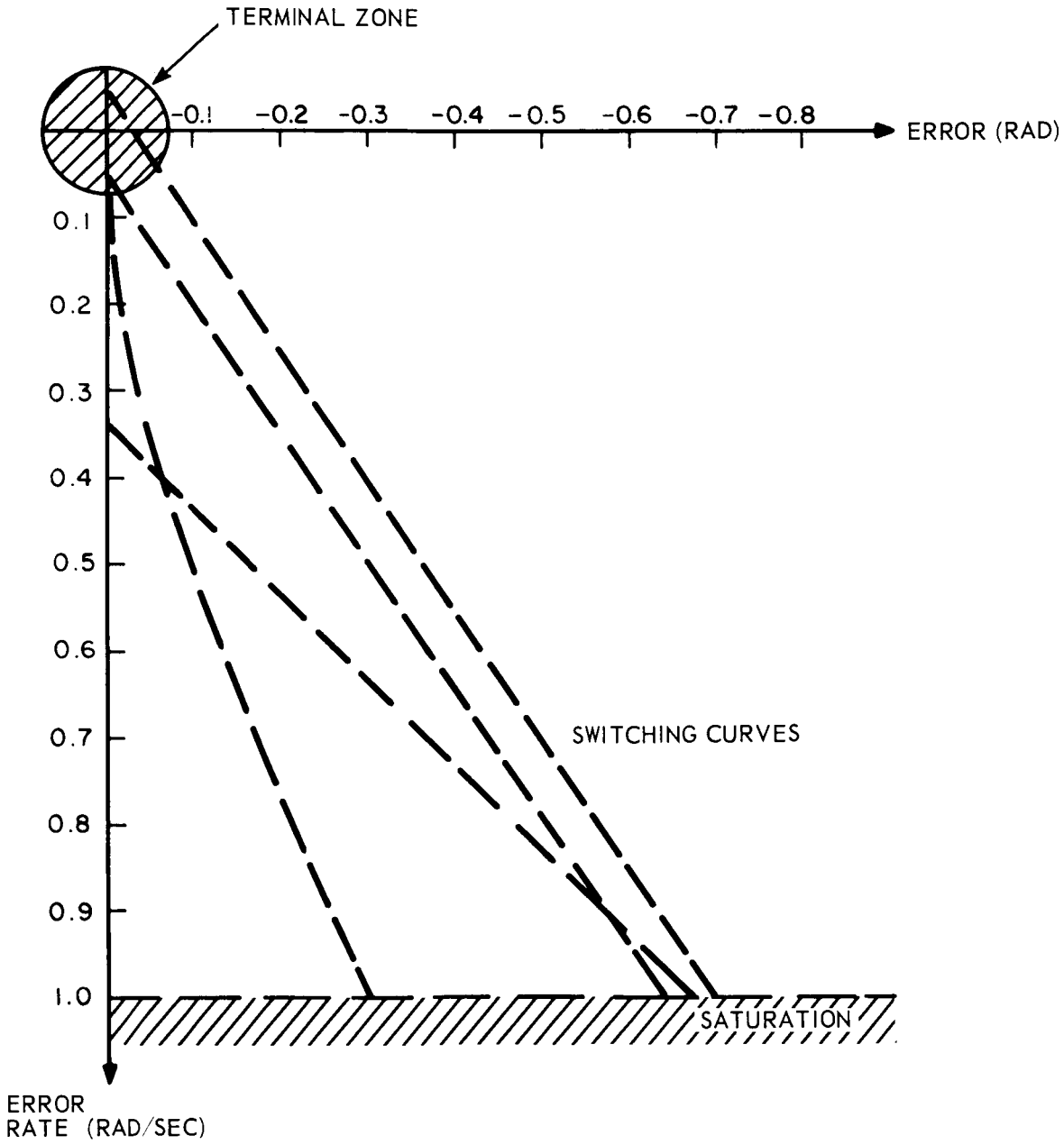
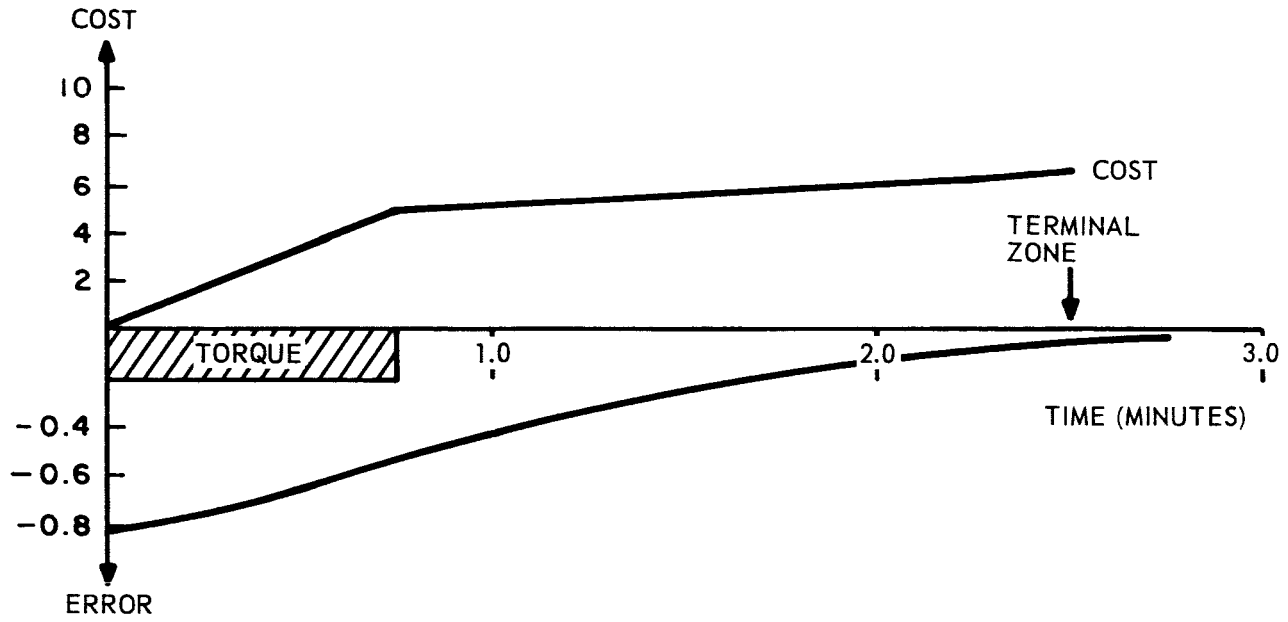
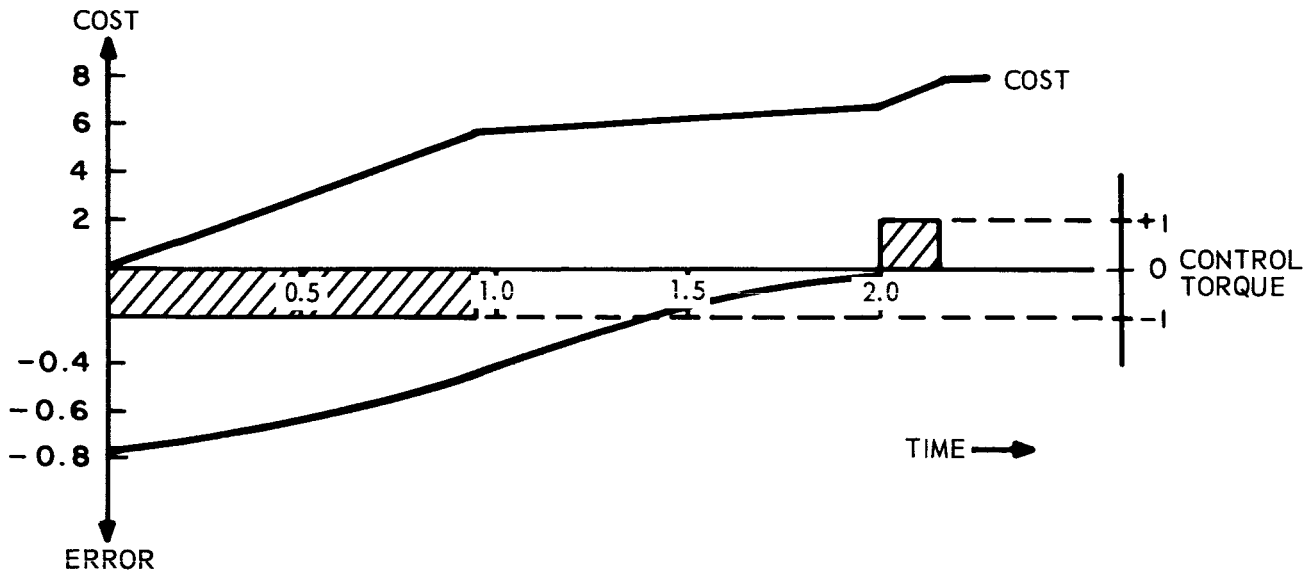


Figure 5. Switching Boundaries in Phase Space

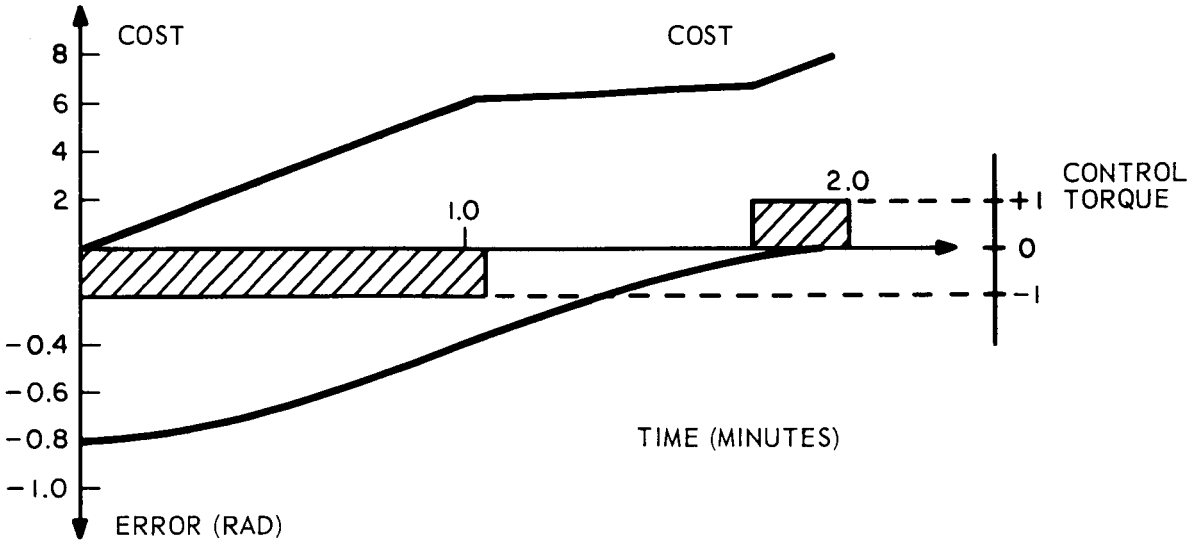


a. MIN FUEL POLICY, MIN FUEL COST

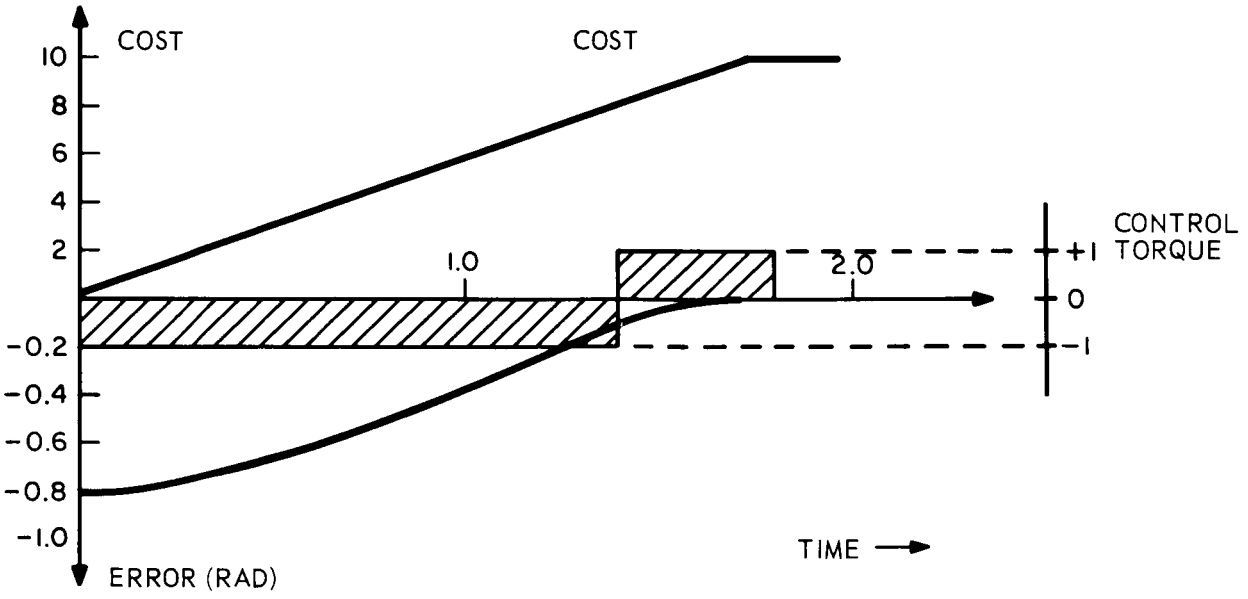


b. CONSERVATIVE FUEL POLICY, MIN FUEL COST

Figure 6. Reference Data for $C_p = 5.0$



a. CONSERVATIVE FUEL POLICY, MIN FUEL COST



b. MIN TIME POLICY, MIN FUEL COST

Figure 7. Reference Data for $C_p = 5.0$

TABLE 1. SUMMARY OF REFERENCE COST DATA

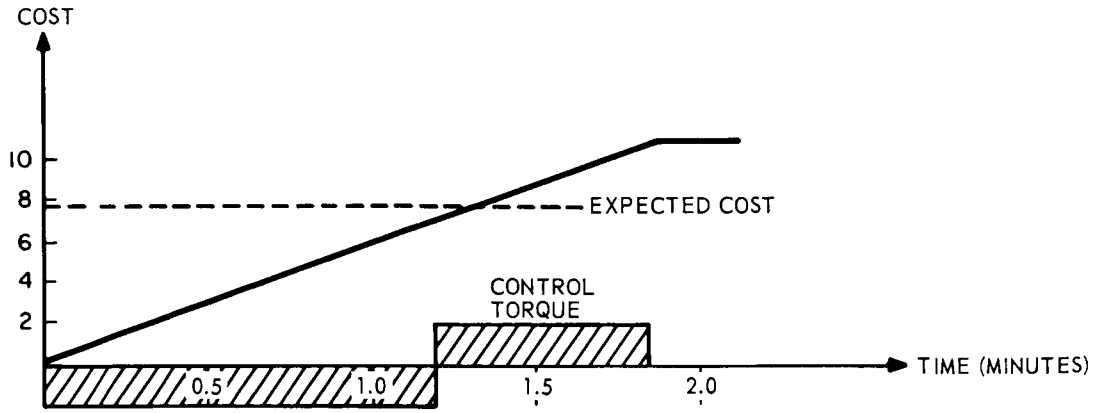
Type of Policy \ Fuel Weight (C_p)	5.0	2.5	0.1
Minimum fuel	7.6	5.2	3.0
Conservative fuel	7.4	4.8	2.25
Conservative fuel	7.5	4.75	2.11
Minimum time	9.9	6.48	1.86
Expected cost E(P)	7.9	5.7	2.04

Expected cost is based on cost to origin while actual cost is based on cost to terminal zone near origin.

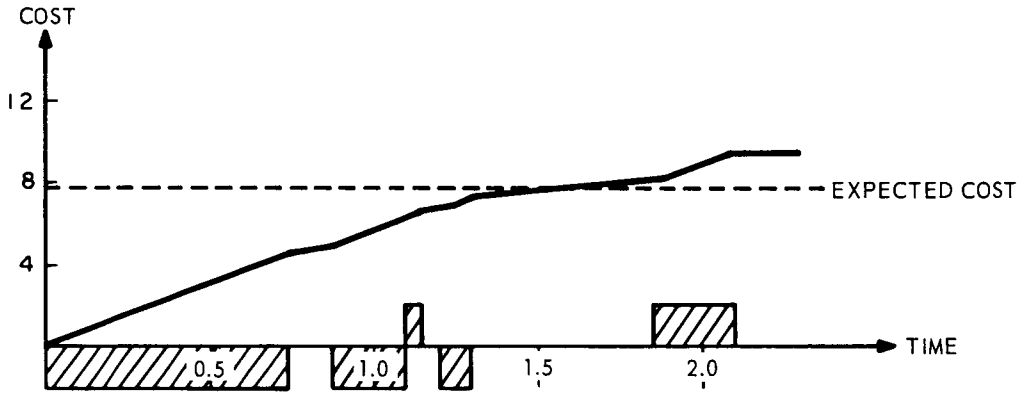
3.3.3 Training to a Control Policy

The control values and resulting costs are graphed in figure 8, with a time optimal policy being used where fuel conservative policy is desired. The result is a high cost to converge to the terminal zone. Retraining starts with approximately the same initial conditions as the previous step (error ≈ 0.8, error rate ≈ 0). The cost for the control used during the retraining interval is lower than the previous policy but still far in excess of the possible minimum. The trained controller has a performance cost below that of the human controller. This occurs because the automatic policy which closely approximates the human controller is closer to optimal than the human control policy. Figure 9 shows the phase plane trajectory during the retraining period, while figure 10 shows the phase plane trajectory resulting from the trained controller.

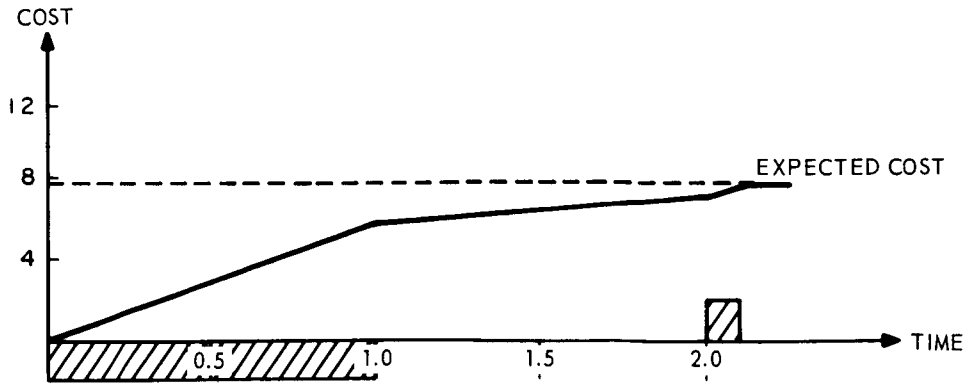
Figures 11, 12, and 13 again show training to a near optimal policy where the performance index weights the cost of fuel somewhat less.



a. INITIAL AUTOMATIC CONTROL



b. MANUAL CONTROL



c. TRAINED AUTOMATIC CONTROL

Figure 8. Experiment Number 1 (C_p = 5.0)

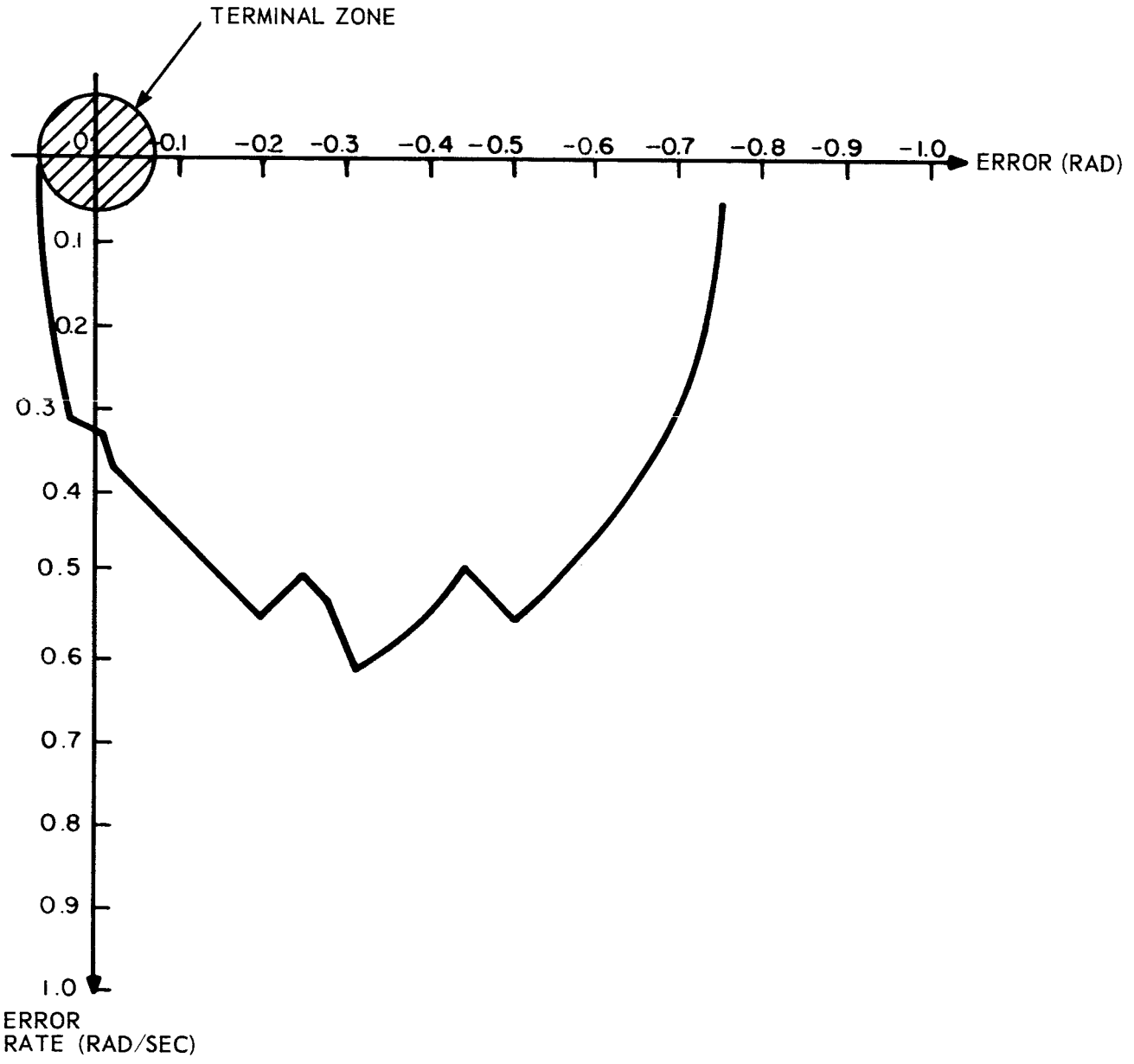


Figure 9. Experiment Number 1 (Phase Plane Trajectory During Training)

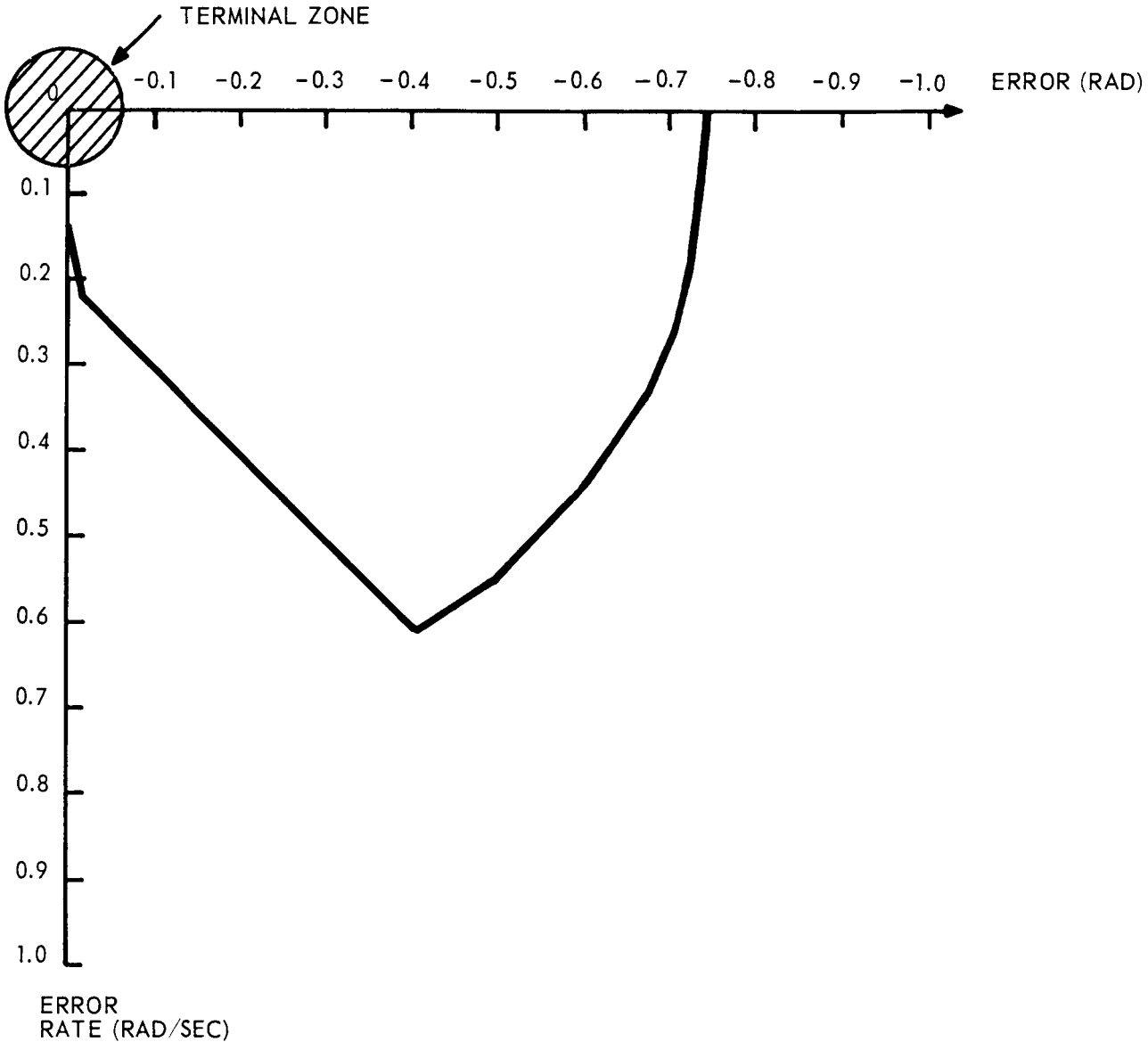
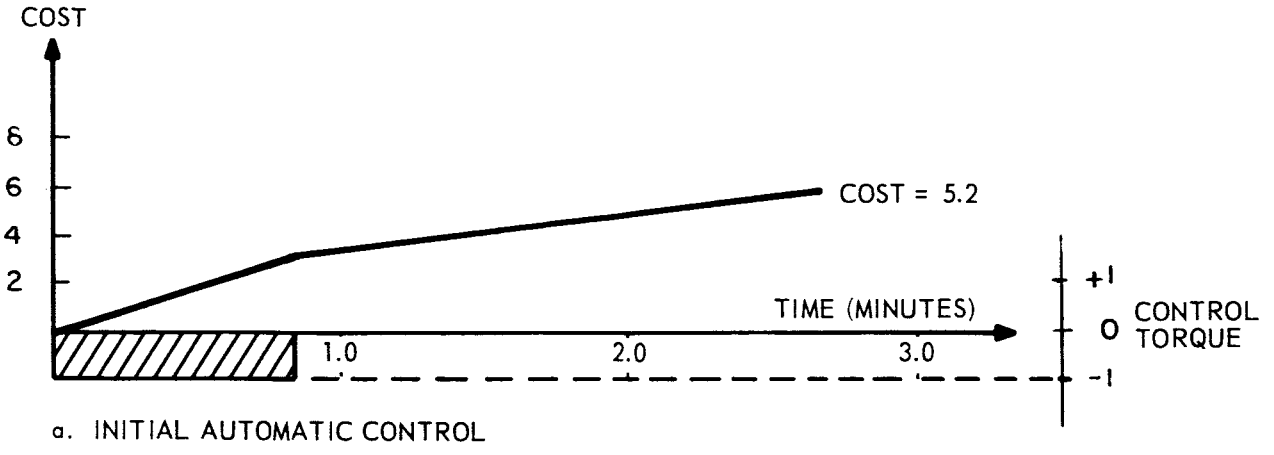
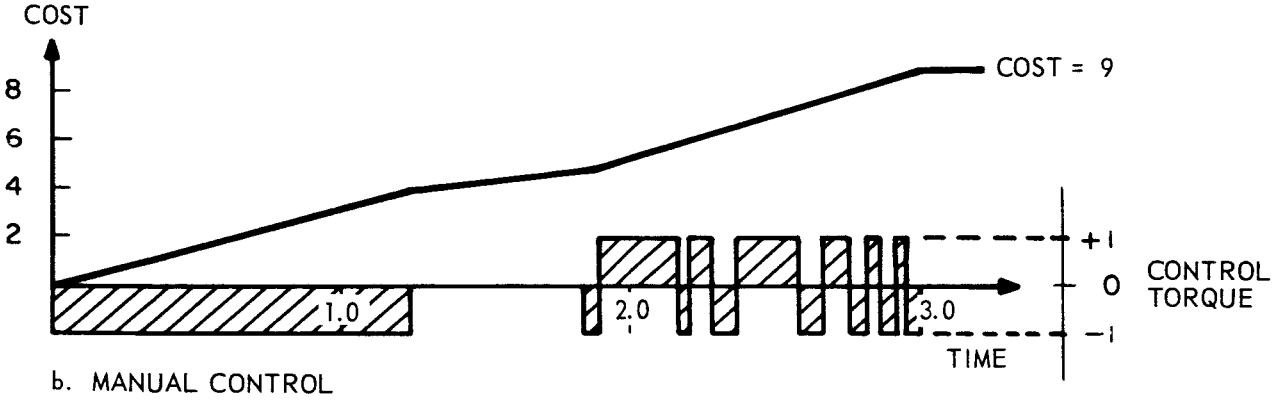


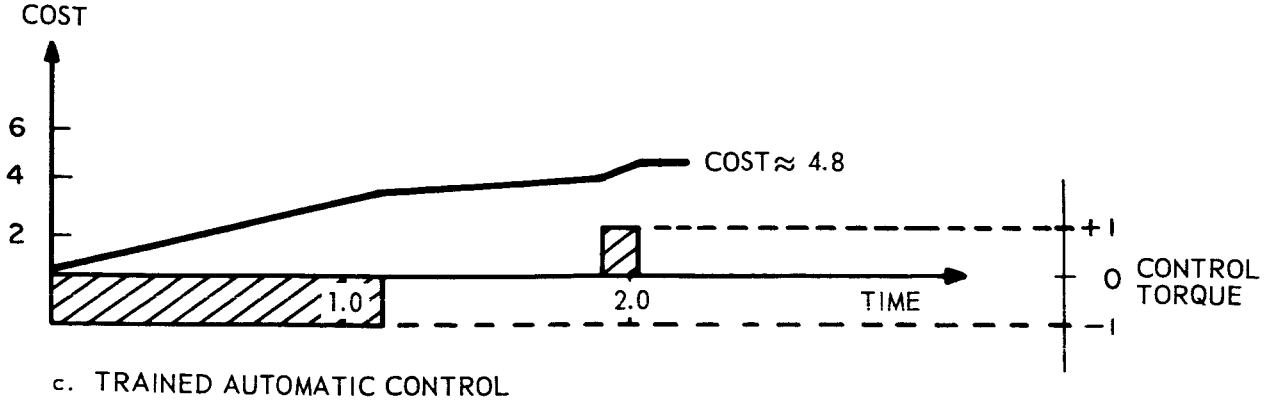
Figure 10. Experiment Number 1 (Phase Trajectory of Trained Controller)



a. INITIAL AUTOMATIC CONTROL



b. MANUAL CONTROL



c. TRAINED AUTOMATIC CONTROL

Figure 11. Experiment Number 2 (C_p = 2.5)

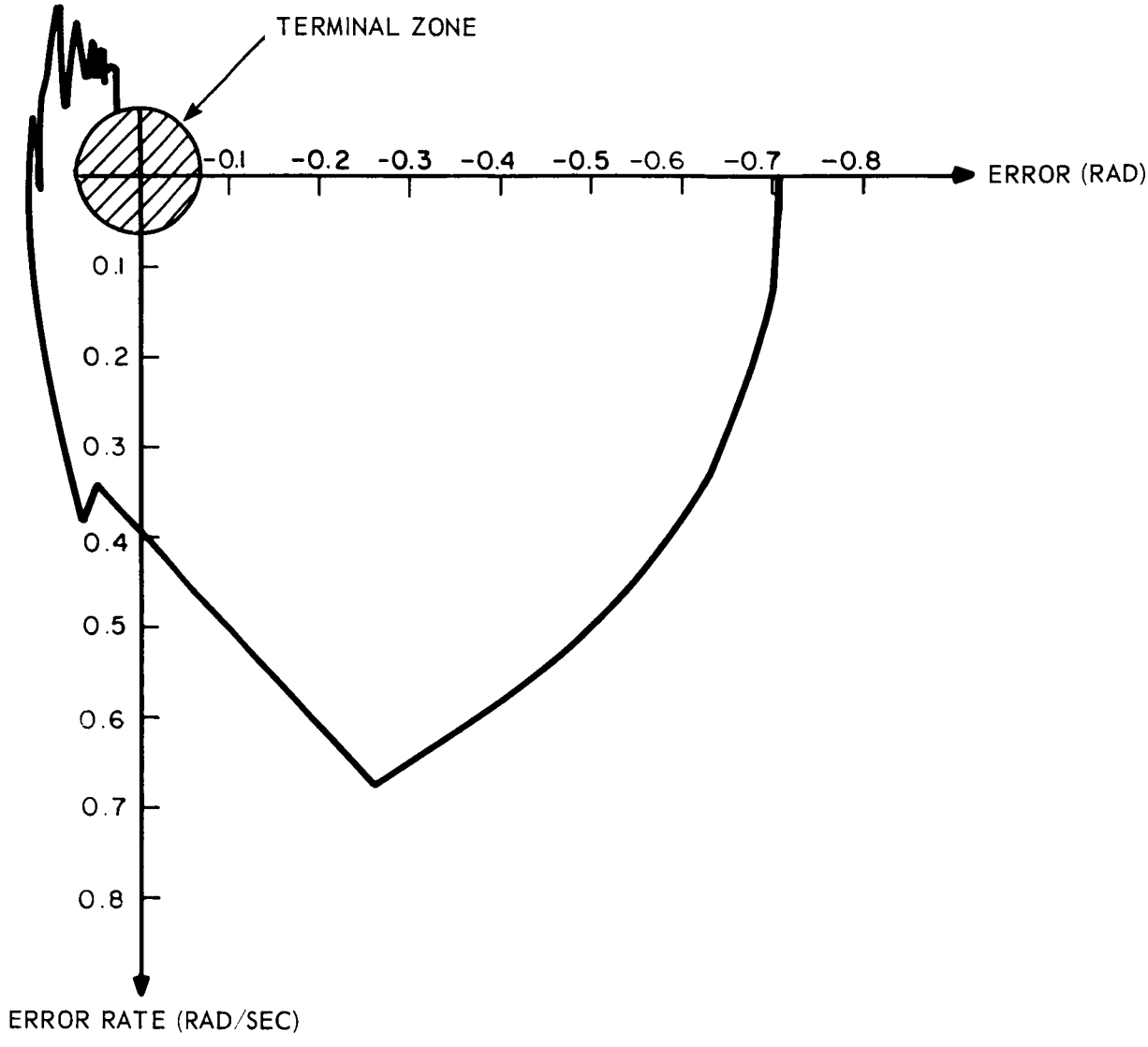


Figure 12. Experiment Number 2 (Phase Plane Trajectory During Training)

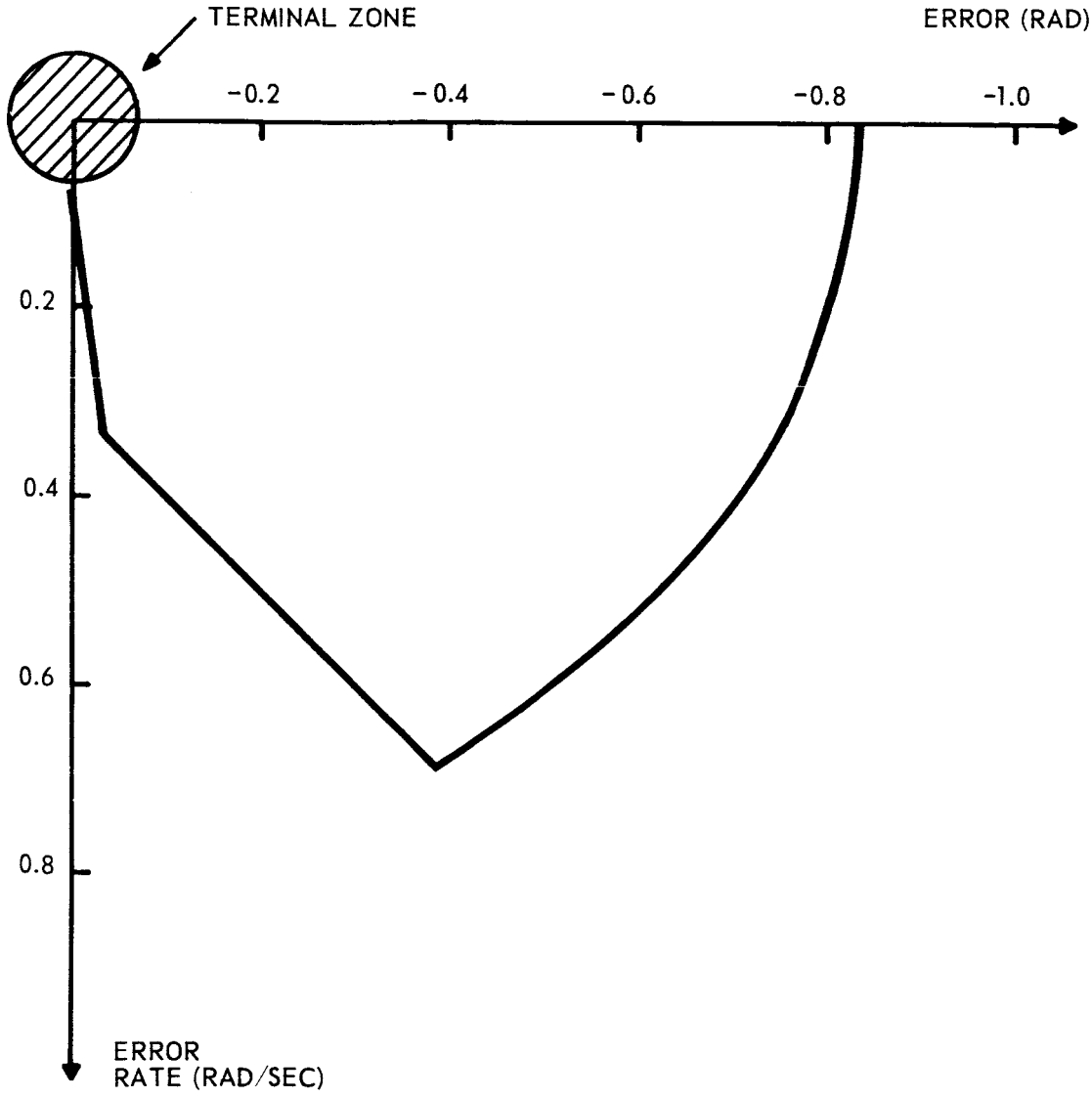
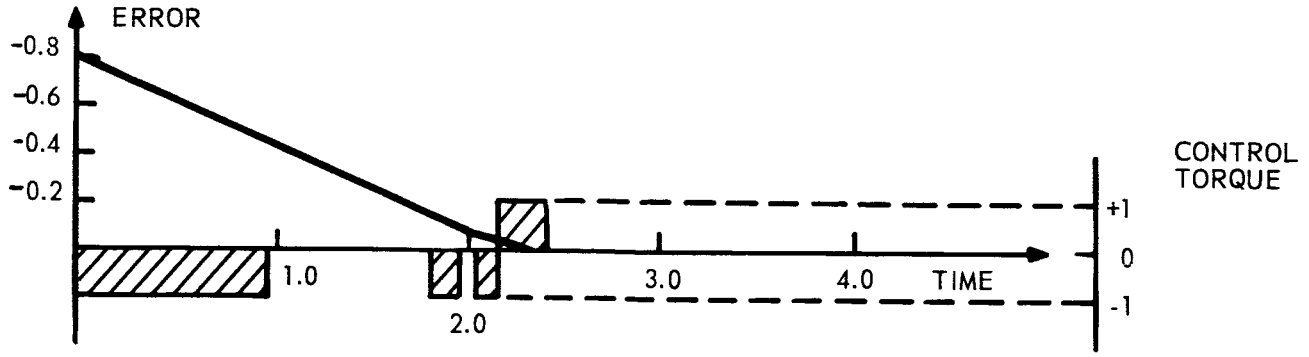


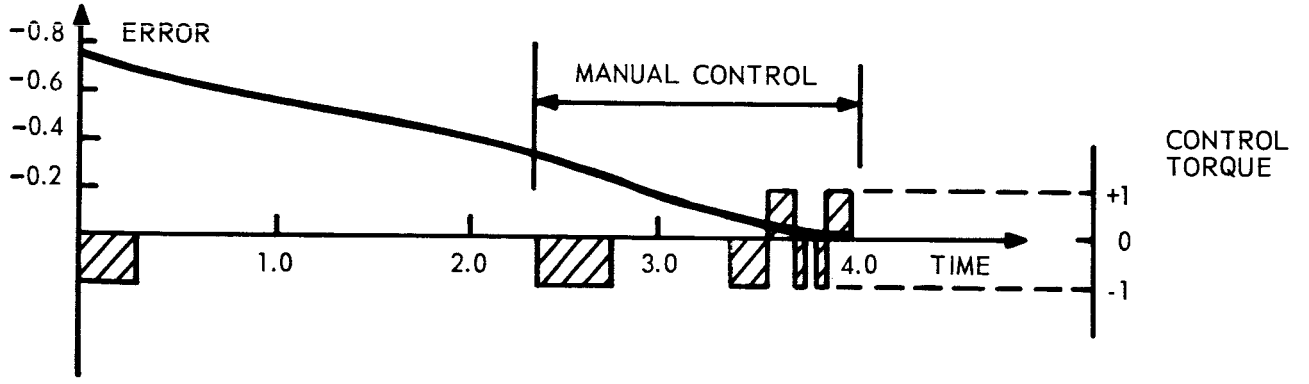
Figure 13. Experiment Number 2 (Phase Trajectory of Trained Controller)

Finally, figures 14 and 15 show training to a policy which is not optimal but very close to that of the human controller. Training was accomplished over a period of two step inputs. After the initial training, the control was returned to automatic but, before convergence occurred, the human operator became dissatisfied with the performance and selected manual control again. The trained policy closely approximates the human policy, because the human operator did not change control values frequently within given phase space regions. A summary of the experimental work is given in table 2.

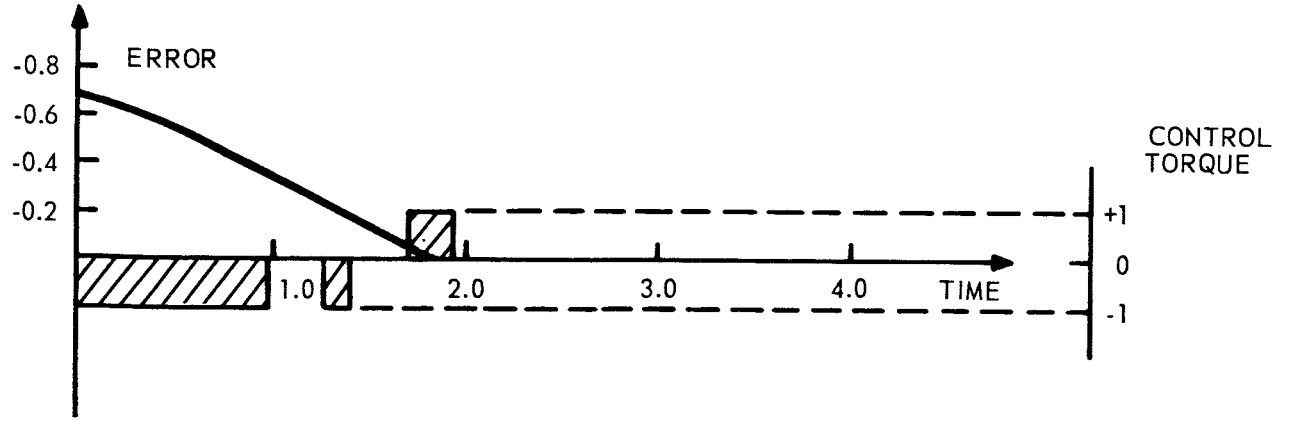
It was observed that the human operator tended to switch control torque far more often than necessary to keep costs low. Consequently, the automatic control to which the operator trained frequently came up with lower costs, since the trainable controller tends to integrate the type of control used within a control region.



a. MANUAL CONTROL



b. AUTOMATIC/MANUAL CONTROL



c. TRAINED AUTOMATIC CONTROL

Figure 14. Experiment Number 3 (Training to Arbitrary Human Control)

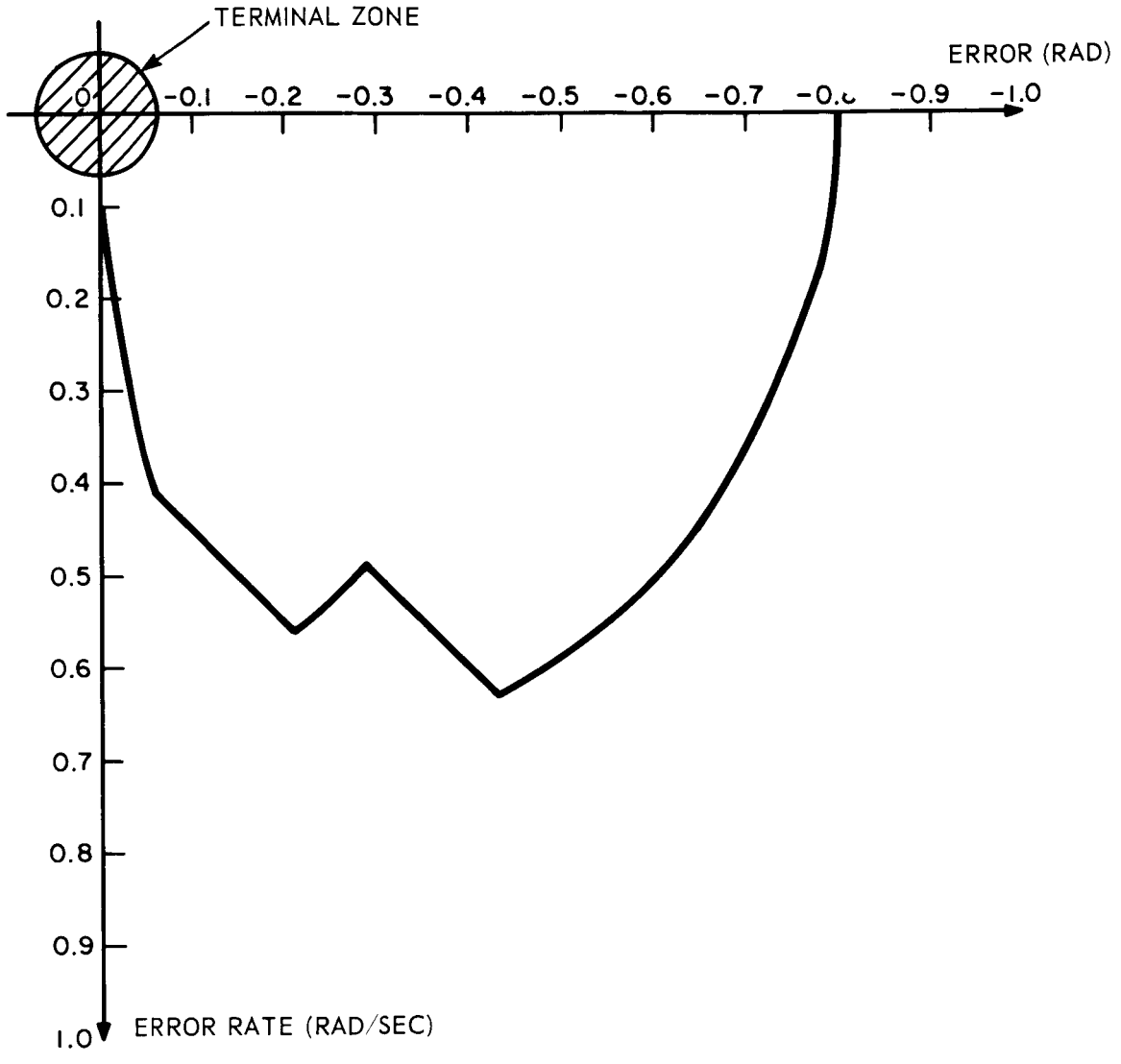


Figure 15. Experiment Number 3 (Phase Trajectory Resulting from Training in Experiment Number 3)

TABLE 2. DESCRIPTION OF GRAPHS

Figure	Description of Graphs
5	Switching Boundaries in Phase Space
6	Reference Data for $C_p = 5.0$
7	Reference Data for $C_p = 5.0$
8	Experiment Number 1: Training of Controller from Time Optimal to a Fuel Conservative Policy; Error, Cost, and Control Values
9	Experiment Number 1: Phase Plane Trajectory During Training
10	Experiment Number 1: Phase Trajectory of Trained Controller
11	Experiment Number 2: Training of Controller from Minimum Fuel Policy to Fuel Conservative Policy; Cost and Control Values
12	Experiment Number 2: Phase Plane Trajectory During Training
13	Experiment Number 2: Phase Trajectory of Trained Controller
14	Experiment Number 3: Training of a Controller to Arbitrary Human Control Policy
15	Experiment Number 3: Phase Trajectory Resulting from Training in Experiment Number 3

4. CONCLUSIONS AND RECOMMENDATIONS

The work done under the PCMS program has extended the trainable logical network concept into a tool for adaptive decision-making. Types of decision processes and their associated decision criteria were identified. A mathematical model of the adaptive decision process was developed and evaluated by applying it to a problem of control. The results were very encouraging for the sample plant and performance index chosen. The approach adopted made maximum use of a priori information about the plant and input waveforms.

It is recommended that a study be undertaken which directly attacks the two problems: (1) choice of performance measurements, and (2) teaching a human to control a system in accordance with predefined performance criteria.

APPENDIX A

LEAST SQUARES PREDICTION

Let q_i be the performance measured at time t_{i+1}. It is taken as a random variable, having mean value $E(q_i)$. Let this mean value be given by

$$E(q_i) = f(\bar{x}^i, \bar{\theta})$$

where $\bar{x}^i$ is the measurement vector at time t_i, having components $x_j(t_i)$ for $j = 1,2, \ldots, r$, and where $\bar{\theta}$ is a vector of parameters having components $\theta_j(t_i)$. Knowing this relationship enables one to establish the mean value of performance one time unit in advance of measurements $\bar{x}^i$. The problem posed is one where we are given the form of the function f. We must establish the best estimate of the parameters, $\bar{\theta}$, its distributional properties, and some computing algorithm for its updating with time.

Let the measured performance over n-time indices be represented by the vector

$$\bar{q} = (q_1, \ldots, q_n)$$

The problem becomes especially simple now, if we take the above referenced functional form as [13]

$$E(\bar{q}) = \bar{\theta} A^T$$

where

$$A = \left[a_{ij} = g_j(x^i)\right] ; \quad \begin{matrix} i = 1, 2, \ldots, n \\ j = 1, 2, \ldots, r \end{matrix}$$

with $g_j(x^i)$ being independent functions of the measurement vector $\bar{x}$ at time index i, and where the parameter vector is of the form

$$\bar{\theta} = (\theta_1, \theta_2, \ldots, \theta_r)$$

That is, the performance predicted for time index i+1 is given by

$$E(q_i) = f(\bar{x}^i, \bar{\theta}) = \sum_{j=1}^{r} a_{ij}\ \theta_j$$

$$= a_{i1}\ \theta_1 + a_{i2}\theta_2 + \cdots + a_{ir}\theta_r$$

$$= g_1(\bar{x}^i)\theta_1 + g_2(\bar{x}^i)\theta_2 + \cdots + g_n(\bar{x}^i)\theta_r$$

for i = 1, 2, ..., n. Select the "best" estimate of the parameter, $\bar{\theta}$, $\hat{\bar{\theta}}$, as one which minimizes the sum of the squares of the deviation of E(q) from the

actual measurements, q; i.e., let

$$R^2 = \left[\bar{q} - E(\bar{q})\right] \left[\bar{q} - E(\bar{q})\right]^T$$
$$= \left(\bar{q} - \bar{\theta}A^T\right) \left(\bar{q} - \bar{\theta}A^T\right)^T$$

To minimize R^2, let the rank of A be r, whereby it can be shown in a straightforward manner (setting its derivative with respect to $\bar{q}$ equal to the vector $\bar{\theta}$) that one must select

$$\hat{\theta} = \bar{q}A\ (A^T A)^{-1}$$

Note that matrix A is not a square matrix, and that $A^T A$ is a nonsingular symmetric r-by-r matrix. In this case, one obtains

$$R^2 = \left(\bar{q} - \hat{\theta} A^T\right) \left(\bar{q} - \hat{\theta} A^T\right)^T$$
$$= \bar{q}\ \left[I - A(A^T A)^{-1}\ A^T\right] \bar{q}^T$$

A theorem by Markov states that if we take the components of q as normally distributed with common variance σ^2 (a restriction that is convenient rather than necessary), then

(1) $\hat{\theta} \cap N\left[\bar{\theta},\ \sigma^2\ (A^T A)^{-1}\right]$

(2) $\frac{R^2}{\sigma^2} \cap \chi^2_{n-r}$

(3) $\hat{\theta}$ and $\frac{R^2}{\sigma^2}$ are independent,

where $\cap$ is used to denote "distributed as" and $N\left[a, B\right]$ is used to denote "normal with mean a and variance-covariance B". This result is of interest to us here in that one obtains the distributional properties of the prediction required for decision theoretic considerations.

As an illustration, let $f(\bar{x},\bar{\theta})$ be some arbitrary function of, say, three variables. Let these be the three measurements: x_1, x_2, and x_3. The best second-order fit of $E(q_i)$ is of the form

$$E(q_i) = x_1\ {}^{(i)}\hat{\theta}_1 + x_2\ {}^{(i)}\hat{\theta}_2 + \ldots + x_2^{(i)}\ x_3^{(i)}\ \hat{\theta}_{23}$$

where

$$\hat{\theta} \equiv (\hat{\theta}_1, \hat{\theta}_2, \ldots, \hat{\theta}_{23})$$

is evaluated as above using the matrix A given by

$$A = \begin{bmatrix} x_1^{(1)}, x_2^{(1)}, x_3^{(1)}, x_1^{(1)2}, x_2^{(1)2}, x_3^{(1)2}, x_1^{(1)}x_2^{(1)}, x_1^{(1)}x_3^{(1)}, x_2^{(1)}x_3^{(1)} \\ \\ \\ x_1^{(n)}, x_2^{(n)}, x_3^{(n)}, \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \end{bmatrix}$$

The previously described procedure now yields the least squares fit of the specified form.

With n>r collections of data (consisting of the n-rows of matrix A), one can readily establish $\hat{\theta}$. If the distribution of $\bar{q}$ is known, one can also establish the distribution of $\hat{\theta}$. For the time being, however, we will consider $\bar{q}$ as normally distributed. Consequently, as n gets larger, the elements of the variance-covariance matrix associated with θ decreases approximately as $\frac{1}{n}$. Whereas this property is desired when the process being considered is stationary, it may not be desired in the adaptive prediction techniques considered here.

Consider, first, the case where N-sets of data are to be considered in conjunction with weighting factors which depend only on their age. The solution is obtained by using a modified R^2, Q^2, given by

$$Q^2 = \left(\bar{q} - \theta A^T\right) \begin{bmatrix} w_1 & 0 & 0 \ldots \\ 0 & w_2 & 0 \ldots \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & w_n \end{bmatrix} \left(\bar{q} - \theta A^T\right)^T$$

where the w_j are positive weighting factors. By denoting the weighting matrix, S, we obtain

$$Q^2 = qWq^T - qWA\theta^T - \theta A^T Wq + \theta A^T WA\theta^T$$

wherein

$$\left.\frac{\partial Q^2}{\partial \theta}\right|_{\theta = \hat{\theta}} = -2qWA + 20A^T WA = 0$$

Hence,

$$\hat{\theta} = \bar{q}\,(WA)\,(A^T WA)^{-1}$$

where $(A^T W\,A)^{-1}$ exists whenever $(A^T\,A)^{-1}$ exists (i.e., whenever A is of rank r, and $\theta = \theta_1, \ldots, \theta_r$). One can again obtain the distribution in $\hat{\theta}$ by knowing the distribution of $\bar{q}$.

We shall now consider the computational aspects of the above. Designate the performance measurements by the vector

$$\bar{q}_o = (q_1, q_2, \ldots, q_n)$$

Let the parameters that have been estimated after the above n-measurements be designated $\hat{\theta}_n$. Let the system measurement matrix be arranged in the (n-by-r) matrix, A_n. We have indicated that the simple unweighted case gives rise to the solution

$$\hat{\theta}_n = q_n\,A_n\,(A_n^T\,A_n)^{-1}$$

If the additional data obtained during the (n+1)st interval are designed by the (vector) row matrix $B_{n+1,} = a_{n+1,1}, a_{n+1,2}, \ldots, a_{n+1,r})$, then the updated "best" estimate of parameter θ is representable in terms of the partitioned matrices given in the equation

$$\theta_{n+1} = (q_o, q_{n+1}) \begin{pmatrix} A_n \\ B_{n+1} \end{pmatrix} \left[\begin{pmatrix} A_n^T & B_{n+1}^T \end{pmatrix} \begin{pmatrix} A_n \\ B_{n+1} \end{pmatrix} \right]^{-1}$$

wherein

$$\hat{\theta}_{n+1} = \left(\bar{q}_o A_n + q_{n+1}\,B_{n+1}\right) \left(A_n^T\,A_n + B_{n+1}^T\,B_{n+1}\right)^{-1}$$

The total system memory required for this updating process resides in the (1-by-r) vector $(\bar{q}_o A_n)$ and the (r-by-r) matrix, $A_n^T A_n$.

For the weighted case, we established that for n-sets of data,

$$\hat{\theta}_n = \bar{q}_o W_n A_n (A_n W_n A_n)^{-1}$$

where

$$W_n = \begin{bmatrix} w_1 & 0 & 0 & . & . & . \\ 0 & w_2 & 0 & . & . & . \\ . & . & . & & & \\ . & . & & . & & \\ . & . & & & . & \\ & & . & . & . & w_n \end{bmatrix}$$

This implies that the weight, w_j, is given to the square of the jth deviation, relative to the minimization procedure. Assume, for the sake of simplicity, that the relative importance of one sample to another sample remains fixed once it is established; i.e., once the weight, w_j, is established, the ratio w_j/w_k for all $k \leq j$ remains fixed (independent of assignments of weights w_i for $i > j$). Under this assumption, we see that for time index, n+1, we can write

$$\hat{\theta}_{n+1} = (q_o, q_{n+1}) \begin{pmatrix} W_n & 0 \\ 0 & w_{n+1} \end{pmatrix} \begin{pmatrix} A_n \\ B_{n+1} \end{pmatrix} \left[(A_n^T, B_{n+1}^T) \begin{pmatrix} W_n & 0 \\ 0 & w_{n+1} \end{pmatrix} \begin{pmatrix} A_n \\ B_{n+1} \end{pmatrix} \right]^{-1}$$

wherein our weighted updating procedure would compute

$$\hat{\theta}_{n+1} = \left(q_o W_n A_n + w_{n+1} q_{n+1} B_{n+1} \right) \left(A_n^T W_n A_n + w_{n+1} B_{n+1}^T B_{n+1} \right)^{-1}$$

Noting that the updated parameters are represented as

$$\bar{q}_o = (\bar{q}_o, q_{n+1})$$

$$A_{n+1} = \begin{pmatrix} A_n \\ B_{n+1} \end{pmatrix}$$

$$W_{n+1} = \begin{pmatrix} W_n & 0 \\ 0 & w_{n+1} \end{pmatrix}$$

we see that

$$q_o W_{n+1} A_{n+1} = q_o W_n A_n + w_{n+1} q_{n+1} B_{n+1}$$

and

$$A_{n+1}^T W_{n+1} A_{n+1} = A_n^T W_n A_n + w_{n+1} B_{n+1}^T B_{n+1}$$

These parameters are then used in the succeeding cycles of computations, yielding θ_{n+m} for all m.

The above scheme can be readily modified to perform the computation of predicted performance on the basis of previous data without updating. This would be useful for establishing control modifications of the man-machine process.

APPENDIX B

SYMBOL

AND

PROGRAM LISTINGS

FORTRAN Name	Meaning
A	Constant used in y, $\dot{y}$, and performance equations
ALPH(J)	Slope of line J
BETA(J)	Intercept of line J
CE	Tolerance for desired output region
CP	Constant used in performance evaluation
ER	e = error
ERDAB	Absolute value of $\dot{e}$
ERDOT	$\dot{e}$
EXPR	Expected performance
ICTR(M)	Counter for region M
IFLAG	Flag to denote change in y_{in}
IRU(M)	Monitored control for region M
L	Number of steps in time function for y_{in}
NUMST	Number of steps needed for training
OFLAG	Flag to denote actual output within tolerance region
PERF	Performance evaluation
PFLAG	Flag to denote poor performance
RAND	Random number subroutine
SIER	Sign function of $\dot{e} = \frac{\lvert\dot{e}\rvert}{\dot{e}}$
STP	Distance from desired output
TAU	Increment of time
TIME	Time
TYIN(J),J=1,L	Time for values of y_{in}
YINP(J),J=1,L	Values of y_{in} as a function of time
TOL	Performance tolerance
UU	Control value before incorporating gain
V(J)	Value of regional function J at some point
Y	y = actual output
YDOT	$\dot{y}$
YIN	Value of y_{in} (desired output)
ZK	System gain constant

```
00000  0 00 00000          1 $RAND  PZE
*00001  0 43 0 00000        2        BRM     201SYS
00002  1 00 0 00041         3        XSD     TEM
*00003  0 43 0 00000        4        BRM     202SYS
                            5 *
00004  0 71 0 00033         6        LDX     N12
00005  0 41 0 00016         7 A      BRX     BB
00006  0 71 0 00034         8        LDX     N121
00007  0 76 0 00036         9 AA     LDA     RANDM
00010  0 72 0 00026        10        SKA     N1
00011  0 01 0 00013        11        BRU     $+2
00012  0 76 0 00035        12        LDA     RN1
00013  1 40 0 00037        13        XMP     K
00014  0 35 0 00036        14        STA     RANDM
00015  0 35 0 00043        15        STA     R2BIT
00016  0 75 0 00043        16 BB     LDB     R2BIT
00017  0 76 0 00023        17        LDA     ZERO
00020  0 6700 002          18        LSH     2
00021  0 36 0 00043        19        STB     R2BIT
00022  0 73 0 00024        20        SKG     ONE
00023  0 01 0 00030        21        BRU     OUT
00024  0 75 0 00040        22        LDB     THRE
00025  0 70 0 00040        23        SKM     THRE
00026  0 01 0 00005        24        BRU     A
00027  0 76 0 00026        25        LDA     N1
00030  0 35 1 00041        26 OUT    STA     *TEM
00031  0 37 0 00033        27        STX     N12
00032  0 51 0 00000        28        BRR     RAND
                            29 *
00033  77777777           30 N12    DATA    -1
00034  77777764           31 N121   DATA    -12
00035  37145213           32 RN1    DATA    037145213
00036  37145213           33 RANDM  DATA    037145213
00037  07346545           34 K      DATA    07346545
00040  00000003           35 THRE   DATA    3
00041                     36 TEM    RES     2
00043  0 00 00000         37 R2BIT  PZE
       00000026           38 N1     EQU     026
       00000023           39 ZERO   EQU     023
       00000024           40 ONE    EQU     024
                          41 XMP    OPD     014000000
                          42 XSD    OPD     010000000
                          43        END

00001                        201SYS
00003                        202SYS
```

```
      1  *     HUMAN PERFORMANCE CONTROL AND MONITORING SYSTEM
      2  C     CONTRACT NO. NASW 1085
      3        DIMENSION TYIN(75), YINP(75), ALPH(5), BETA(5), V(5), IRU(0/31),
      4       1ICTR(0/31),CPIN(75)
      5        COMMON TYIN,YINP,ALPH,BETA,V,IRU,ICTR,CPIN
      6  C     INPUT INITIAL DATA
      7  C     L=NO. OF VALUES YIN CAN ASSUME
      8  C     TYIN,YINP=TIME,F(TIME) FOR YIN
      9     25 READ 126,L
     10    126 FORMAT (I3)
     11    130 READ 131,(TYIN(J),YINP(J),CPIN(J),J=1,L)
     12    131 FORMAT(3F10.2)
     13        TYPE 134
     14    134 FORMAT(//,7X,4HTIME,7X,3HYIN,7X,2H CP)
     15    135 TYPE 136,(J,TYIN(J),YINP(J),CPIN(J),J=1,L)
     16    136 FORMAT (I3,3F10.2)
     17    137 FORMAT (I3,2F10.2)
     18    138 FORMAT (2F10.2)
     19  C     READ SLOPES AND INTERCEPTS OF LINES DETERMINING REGIONS
     20    140 READ 138,(ALPH(J),BETA(J),J=1,4)
     21        TYPE 141
     22    141 FORMAT(//,7X,4HALPH,7X,4HBETA)
     23    145 TYPE 137,(J,ALPH(J),BETA(J),J=1,4)
     24  C     READ NO. OF STEPS NEEDED FOR TRAINING
     25        READ 126, NUMST
     26    146 TYPE 147,NUMST
     27    147 FORMAT(//,5X,6HNUMST=,I3)
     28  C     IRU(M)=INITIAL CONTROL FOR REGION M
     29        READ 126,(IRU(M),M=0,31)
     30        TYPE 127
     31    127 FORMAT(//,2X,$REGION  CONTROL$)
     32    148 TYPE 149,(M,IRU(M),M=0,31)
     33    149 FORMAT(5X,I3,7X,I3)
     34    150 TIME=0.0
     35    151 DO 154 M=0,31
     36        ICTR(M)=0
     37    154 CONTINUE
     38        Y=0.0
     39        YDOT=0.0
     40        PERF=0.0
     41        IFLAG=0
     42        OFLAG = 0
     43        I=1
     44    155 YIN=YINP(I)
     45        ERDOT = 0.0
     46    100 READ 110,A
     47        READ 110,ZK
     48        READ 110,CP
     49        READ 110,TAU
     50        READ 110,CE
     51        READ 110, TOL
     52        TYPE 115, A,ZK,CP,TAU,CE,TOL
     53        TYPE 120
     54    110 FORMAT(F13.3)
     55    115 FORMAT(/,2HA=,F7.3,6H---ZK=,F7.3,6H---CP=,F7.3,7H---TAU=,F7.3,
```

```
=    56        16H---CE=,F7.3,7H---TOL=,F7.3]
=    57     120 FORMAT[//,3X,4HTIME,8X,1HU,9X,1HY,7X,5HY DOT,6X,1HE,8X,5HE DOT,
=    58        16X,1HP,8X,4HY IN]
=    59  C      COMPUTE EXPECTED PERFORMANCE
=    60     900 DL=1/[A*CP]
=    61         ER= YIN-Y
=    62         IPFLAG=0
=    63         CYT=ELOG[1+A*DL]
=    64         ABE=ABS[ER]
=    65         ACYT=ABS[CYT]
=    66         W=-A*[ABE+ACYT]
=    67         B=[1-EXP[A*W]]/DL
=    68         YDS=[1-EXP[-A*ABE]]/A
=    69         S=SQRT[YDS]
=    70         IF[YDS-DL**2] 1000,1000,1001
=    71    1000 ESCP=[1+CP]*[1/A]*ELOG[[1+S]/[1-S]]
=    72    1002 TYPE 160, ESCP
=    73         TACE=[1/A]*ELOG[[1+S]/[1-S]]+TIME
=    74         TYPE 161, TACE
=    75     160 FORMAT[///,$EXPECTED PERFORMANCE=$,F7.3]
=    76     161 FORMAT [$EXPECTED TIME OF CONVERGENCE =$, F 7.3]
=    77         EXPR=ESCP
=    78         GO TO 499
=    79    1001 EXPR=[1+CP]*A*[2*ACYT+ABE]+[1/A]*ELOG[B]+TOL
=    80         TRAP=A*[2*ACYT+ABE]+TIME+TOL/2+ELOG[B]
=    81         TYPE 160, EXPR
=    82         TYPE 161, TRAP
=    83     499 IF[ERDOT] 500,505,510
=    84     500 SIER = -1
=    85         GO TO 520
=    86     505 SIER = 0
=    87         GO TO 520
=    88     510 SIER = 1
=    89     520 ERDAB = ABS[ERDOT]
=    90  C       DETERMINE REGION M
=    91     600 M=0
=    92         DO 620 J=1,5
=    93         IF[J-5] 602,601,602
=    94     601 V[J]=A**2*ER - SIER *ELOG[1.0+A*ERDAB]+A*ERDOT
=    95         GO TO 603
=    96     602 V[J]=ERDOT-ALPH[J]*ER-BETA[J]
=    97     603 IF[V[J]] 605,605,610
=    98     605 V[J]=0
=    99         GO TO 615
=   100     610 V[J]=1
=   101     615 M=M+2**[J-1]*V[J]
=   102     620 CONTINUE
=   103  C      DETERMINE AUTO OR MANUAL CONTROL
=   104  C
=   105     170 IF[SENSE SWITCH 4] 200,250
=   106  C          UNDER MANUAL CONTROL
=   107     200 IF[SENSE SWITCH 1] 205,215
=   108     205 IF[SENSE SWITCH 2] 210,220
=   109     210 UU=1.0
=   110         GO TO 700
```

```
=   111    215 IF[SENSE SWITCH 2] 220,230
=   112    220 UU=0.0
=   113        GO TO 700
=   114    230 UU= -1.0
=   115  C       MONITOR MANUAL CONTROL
=   116    700 IF[UU-IRU[M]] 730,710,730
=   117    710 IF[ICTR[M]-NUMST] 720,300,300
=   118    720 ICTR[M]=ICTR[M]+1
=   119        GO TO 300
=   120    730 IF[ICTR[M]] 760,740,760
=   121    740 CALL RAND[NEWU]
=   122        IRU[M]=NEWU
=   123        IF[UU-NEWU] 300,750,300
=   124    750 ICTR[M]=2
=   125    760 ICTR[M]=ICTR[M]-1
=   126        GO TO 300
=   127  C         UNDER AUTOMATIC CONTROL
=   128    250 IF[OFLAG] 270,270,260
=   129    260 U=0.0
=   130        GO TO 310
=   131    270 UU=IRU[M]
=   132    300 U=UU*7K
=   133  C         COMPUTE Y AND YDOT
=   134    310 Y=U/A*[TAU-1.0/A]+[YDOT+A*Y]/A+[U/A**2-YDOT/A]*EXP[-A*TAU]
=   135        YDOT=U/A-[U/A-YDOT]*EXP[-A*TAU]
=   136    350 ER=YIN-Y
=   137        ERDOT=-YDOT
=   138        STP=ERDOT**2+ER**2
=   139    450 OFLAG=1
=   140        GO TO 375
=   141    455 OFLAG=0
=   142    355 IF[IFLAG] 360,360,370
=   143    360 PERF=PERF+[CP*ABS[U]+1.0]*TAU
=   144        GO TO 375
=   145    370 PERF=[CP*ABS[U]+1.0]*TAU
=   146  C     OUTPUT DATA FOR THIS LOOP
=   147    375 IF[SENSE SWITCH 3] 401,400
=   148    400 TYPE 380, TIME,U,Y,YDOT,ER,ERDOT,PERF,YIN
=   149    380 FORMAT[8F10.4]
=   150    401 TIME=TIME+TAU
=   151        IF[TIME-TYIN[I+1]] 420,410,410
=   152    410 YIN=YINP[I+1]
=   153        CP=CPIN[I+1]
=   154        I=I+1
=   155        IFLAG=1
=   156        GO TO 900
=   157    420 IFLAG=0
=   158    430 IF [PERF-EXPR]  499,470,470
=   159    470 IPFLAG=IPFLAG+1
=   160        IF[IPFLAG-1] 499,480,499
=   161    480 TYPE 485
=   162    485 FORMAT [//,$-PERFORMANCE IS POOR$]
=   163        GO TO 499
=   164        STOP
=   165  *END
```

COMMON ALLOCATION

77552 TYIN	77324 YINP	77312 ALPH	77300 BETA
77266 V	77226 IRU	77166 ICTR	76740 CPIN

PROGRAM ALLOCATION

00007 L	00010 J	00011 NUMST	00012 M
00013 IFLAG	00014 I	00015 IPFLAG	00016 NEWU
00017 TIME	00021 Y	00023 YDOT	00025 PERF
00027 OFLAG	00031 YIN	00033 ERDOT	00035 A
00037 ZK	00041 CP	00043 TAU	00045 CE
00047 TOL	00051 DL	00053 ER	00055 CYT
00057 ABE	00061 ACYT	00063 W	00065 B
00067 YDS	00071 S	00073 ESCP	00075 TACE
00077 EXPR	00101 TRAP	00103 SIER	00105 ERDAB
00107 UU	00111 U	00113 STP	

SUBPROGRAMS REQUIRED

FLOG	ABS	EXP	SQRT	RAND

APPENDIX C

TRUNCATED SEQUENTIAL DECISIONS

BAYES TRUNCATED SEQUENTIAL DECISION SOLUTION

Let the loss function associated with one of a finite set of possible states of nature, $\omega \epsilon \Omega$, and one of a finite set of possible actions, $a \epsilon A$, be denoted $L(a,\omega)$. If one selects the action which minimizes the expected loss after k-experiments have been performed, then this expected loss can be shown[8] to be given by

$$U_k = \sum_{i=k}^{k} C_i(x_1, \ldots, x_i) + \underset{a \epsilon A}{\text{Min}} \; E_{k\xi}\left[L(a,\omega)\right]$$

where C_i is the cost of the ith experiment which yielded measurement x_i. (If A is not finite, one simply replaces "minimum" with "infimum" over A.) In this expression, $\xi(\omega)$ is the distribution over the states of nature and

$$E_{k\xi}\left[L(a,\omega)\right] \equiv \frac{\sum_{F(\vec{x})} \sum_{\omega \epsilon \Omega} L(a,\omega)\; p\,(\vec{x} \,|\, x_1, x_2, \ldots, x_k, \omega)\xi(\omega)}{\sum_{F(\vec{x})} \sum_{\omega \epsilon \Omega} p(\vec{x} \,|\, x_1, \ldots, x_k, \omega)\; \xi\;(\omega)}$$

where $F(\vec{x})$ is the set of all x whose first k-coordinates are $x_1, \ldots, x_k$.

To establish whether to continue experimenting or to make a decision after a given experiment, one compares the expected loss associated with each of these possibilities. The lesser of these two expected losses (after k-experiments have been performed) is given by α_k. After the complete set of N-experiments has been run, the minimum loss would be

$$\alpha_N = U_N = \sum_{i=1}^{N} C_i(x_1, x_2, \ldots, x_i) + \underset{a \epsilon A}{\text{Min}} \; E_{N\xi}\left[L(a,\omega)\right]$$

Hence, after N-1 experiments it would be

$$\alpha_{N-1} = \text{smaller of the two numbers} \left[\begin{array}{c} U_{N-1} \\ F_{N-1,\,\xi}\left(\alpha_N\right) \end{array}\right]$$

Continuing in this way, one obtains the complete set of minimum risks at each stage of experimentation to be

$$\alpha_N = U_N$$

$$\alpha_{N-1} = \text{smaller}\left[U_{N-1},\ E_{N-1,\xi}\left(\alpha_N\right)\right]$$

$$\vdots$$

$$\alpha_j = \text{smaller}\left[U_j,\ E_{j\xi}\left(\alpha_{j+1}\right)\right]$$

$$\vdots$$

$$\alpha_o = \text{smaller}\left[U_o,\ E_{\xi}\left(\alpha_1\right)\right]$$

where U_o is the expected loss associated with making a decision without experimentation, given by

$$U_o = \underset{a \in A}{\text{Min}} \sum_{\omega \in \Omega} L(a,\omega)\ \xi(\omega)$$

The Bayes optimal procedure requires the computation of α_N, α_{N-1}, ..., α_o, in that order. At each stage of experimentation, say j, one makes a decision if

$$\alpha_j = U_j$$

Otherwise, one continues with experiment j + 1. This is represented schematically by the tree structure shown in figure C-1.

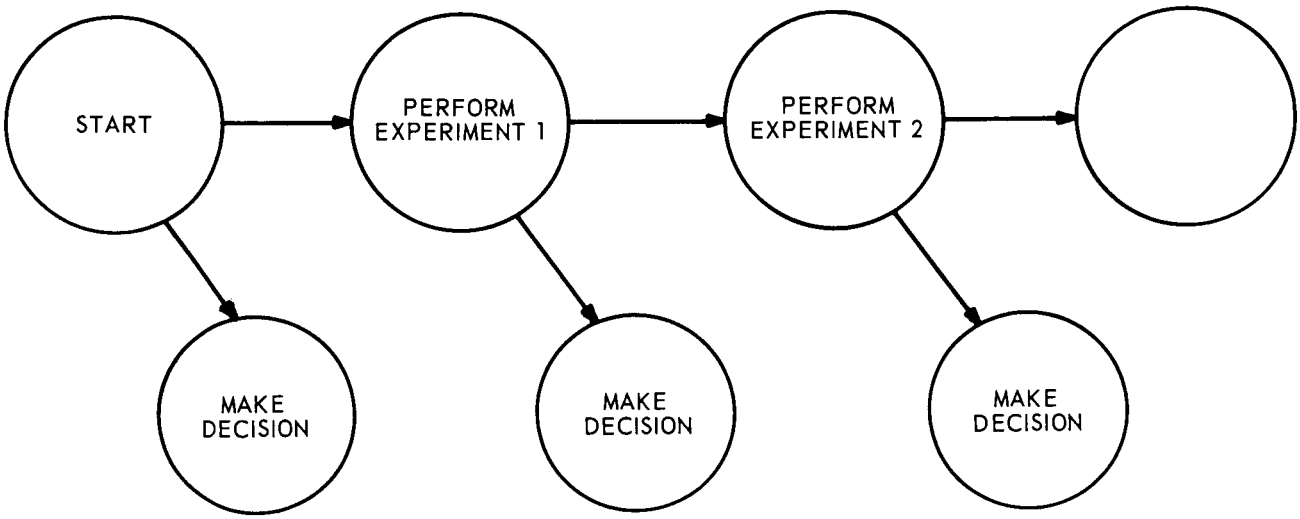


Figure C-1. Decision Tree (Sequential Theory with Fixed-Ordered Experiments)

TRUNCATED SEQUENTIAL DECISIONS WITH DETERMINATION OF THE ORDER OF EXPERIMENTATION

Because notation becomes cumbersome, the procedure is described for the case of three experiments: e_1, e_2, e_3. Generalization to N-experiments follows directly, as does a rigorous proof. The description given below, however, makes for greater clarity.

Basically, the process is like that described previously. The loss matrix will be given by $\left[L(a,\omega)\right]$. Let the states of nature, $\omega\epsilon\Omega$, and the set of possible actions, $a\epsilon A$, be finite. Let the set of observations resulting from the experiments be denoted by the vector, $\vec{x} = (x_1, x_2, x_3)$, where coordinate x_i corresponds to experiment e_i. The minimum expected loss without experimentation is then given by[1,2]

$$U_\phi = \underset{a\epsilon A}{\text{Min}} \; E_\xi \left[L(a,\omega)\right]$$

where $\xi(\omega)$ is the a priori distribution over the states of nature and

$$E_\xi \left[L(a,\omega)\right] = \sum_{\omega\epsilon\Omega} L(a,\omega)\, \xi\,(\omega)$$

After experiment e_i has been performed, yielding the result, $x_i = x_i^o$, the minimum expected loss associated with making a decision is given by

$$U_i = \underset{a\epsilon A}{\text{Min}} \; E_{\xi i} \left[L(a,\omega)\right] + C_i(x_i^o)$$

where $C_i\,(x_i^o)$ is the cost of performing experiment e_i and having the results be $x_i = x_i^o$, and where

$$E_{\xi i} \left[L(a,\omega)\right] = \frac{\sum_{F(\vec{x}\,|\,x_i^o)} \sum_{\omega\epsilon\Omega} L(a,\omega) p(\vec{x}\,|\,x_i = x_i^o, \omega)\, \xi\,(\omega)}{\sum_{F(\vec{x}\,|\,x_i^o)} \sum_{\omega\epsilon\Omega} p(\vec{x}\,|\,x_i = x_i^o, \omega)\, \xi\,(\omega)}$$

with the set $F\,(\vec{x}\,|x_i^o)$ used to indicate summation taken over the set of all possible $\vec{x}$ whose ith coordinate is $x_i = x_i^o$. (The symbol, p, is used generically to represent "probability.") In a like manner, the minimum expected loss associated with making a decision after performing experiment e_i and then performing e_j, obtaining results x_i and x_j, is given by

$$U_{ij} = \underset{a \epsilon A}{\text{Min}}\ E_{\xi ij}\left[L(a,\omega)\right] + C_i(x_i^{o}) + C_{ij}(x_i^{o}, x_j^{o})$$

where $C_{ij}(x_i^{o}, x_j^{o})$ is the cost of experiment e_j after e_i has been performed, and where

$$E_{\xi ij}\left[L(a,\omega)\right] = \frac{\sum_{F(\vec{x}\,|\,x_i^{o}, x_j^{o})} \sum_{\omega \epsilon \Omega} L(a,\omega) p(\vec{x}\,|\,x_i^{o}, x_j^{o}, \omega)\ \xi(\omega)}{\sum_{F(\vec{x}\,|\,x_i^{o}, x_j^{o})} \sum_{\omega \epsilon \Omega} p(\vec{x}\,|\,x_i^{o}, x_j^{o}, \omega)\ \xi(\omega)}$$

For the general case considered here, one should note that

$$p(\vec{x}\,|\,x_i^{o}, x_j^{o}, \omega) \neq p(\vec{x}\,|\,x_j^{o}, x_i^{o}, \omega)$$

That is, the order in which experiments are performed can be expected to be different if, for example, the experiments alter the state of the system considered. However, this procedure is actually required (in general) whenever these probabilities are not independent.

The procedure at each stage in experimentation is to compare the expected loss associated with stopping experimentation and the expected loss for the various continuations. As before, the various expected losses are established by first computing

$$\alpha_{123} = U_{123} \qquad \alpha_{231} = U_{231}$$

$$\alpha_{132} = U_{132} \qquad \alpha_{312} = U_{312}$$

$$\alpha_{213} = U_{213} \qquad \alpha_{321} = U_{321}$$

From this, one computes

$$\alpha_{12} = \text{smaller}\left[U_{12},\ E_{\xi 12}(\alpha_{123})\right]$$

$$\alpha_{13} = \text{smaller}\left[U_{13},\ E_{\xi 13}(\alpha_{132})\right]$$

$$\alpha_{21} = \text{smaller}\left[U_{21},\ E_{\xi 21}(\alpha_{213})\right]$$

$$\alpha_{23} = \text{smaller}\left[U_{23},\ E_{\xi 23}(\alpha_{231})\right]$$

$$\alpha_{31} = \text{smaller}\left[U_{31},\ E_{\xi 31}(\alpha_{312})\right]$$

$$\alpha_{32} = \text{smaller}\left[U_{32},\ E_{\xi 32}(\alpha_{321})\right]$$

Then one computes

$$\alpha_1 = \text{smaller}\left[U_1 \; E_{\xi^1}(\alpha_{12}),\; E_{\xi^1}(\alpha_{13})\right]$$

$$\alpha_2 = \text{smaller}\left[U_2,\; E_{\xi^2}(\alpha_{21}),\; E_{\xi^2}(\alpha_{23})\right]$$

$$\alpha_3 = \text{smaller}\left[U_3,\; E_{\xi^3}(\alpha_{31}),\; E_{\xi^3}(\alpha_{32})\right]$$

Finally, one establishes

$$\alpha_\phi = \text{smaller}\left[U_\phi,\; E_\xi(\alpha_1),\; E_\xi(\alpha_2),\; E_\xi(\alpha_3)\right]$$

One can note that α_{ijk} is the expected loss associated with performing experiments i, j, k -- in that order. The expression, α_{ij}, is the smaller of the expected losses associated with stopping or with continuing. Hence, it is the minimum expected loss (corresponding to the minimum expected loss procedure). This argument is repeated for α_i and for α_ϕ. By this argument, one sees that α_ϕ is the minimum expected risk prior to experimenting.

To utilize this procedure, one should decide without experimentation if

$$\alpha_\phi = U_\phi$$

If this is not so, and

$$\alpha_\phi = E_\xi\left[\alpha_i\right]$$

then one should perform experiment e_i. Having done this, one obtains the result $x_i = x_i^o$, and inquires if

$$\alpha_i(x_i^o) = U_i(x_i^o)$$

wherein one should decide without further experimentation. If this is not so, and

$$\alpha_i(x_i^o) = E_{\xi i}\left[\alpha_{ij}\right]$$

then the procedure calls for continuation by performing e_j, etc. This is illustrated by the tree shown in figure C-2.

A much more formal proof can be argued on the basis of showing that the expected loss for any other partitioning of the outcome space into actions or experiments will be higher than that described above. This has been accomplished but, due to its lack of heuristic appeal, is not included in this report.

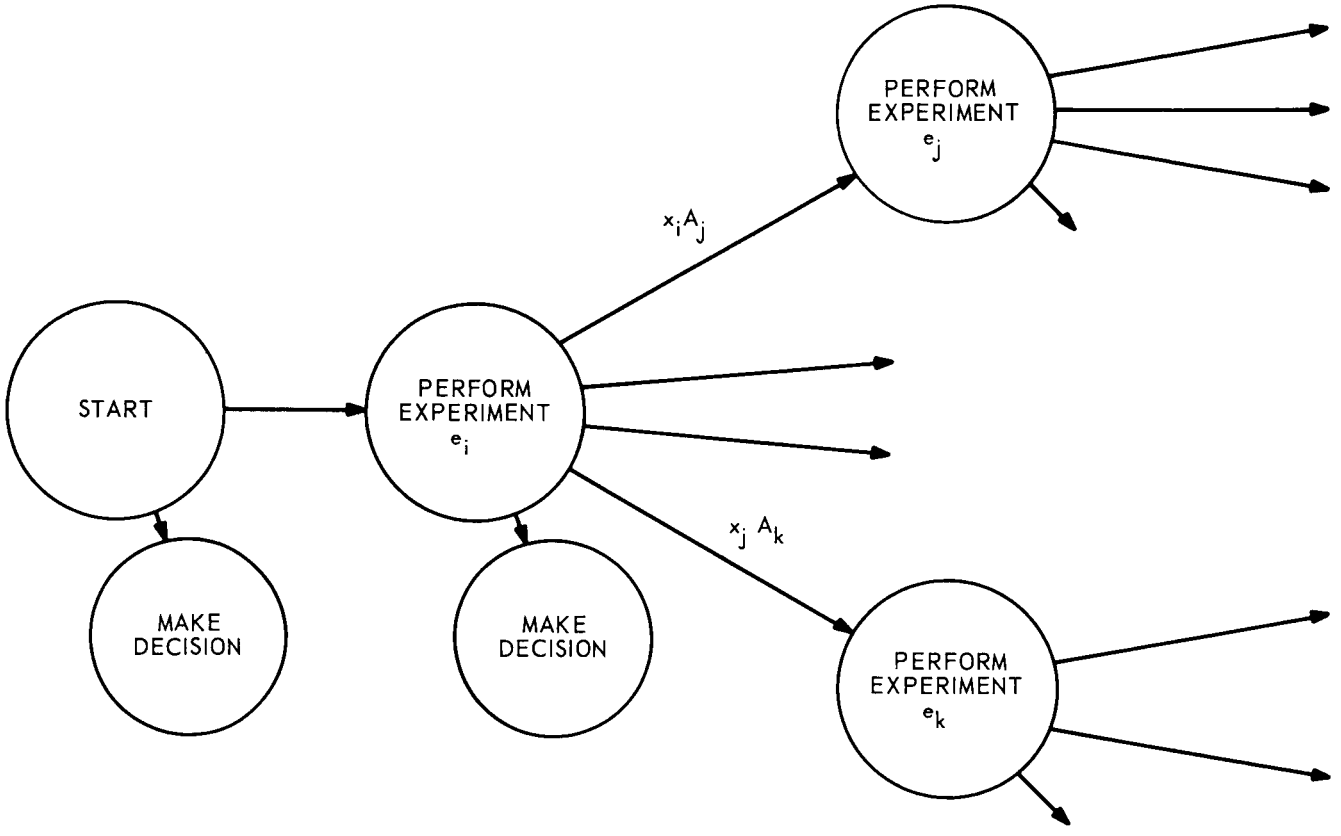


Figure C-2. Decision Tree (Path Depending on Experiment Outcome)

APPENDIX D

BIBLIOGRAPHY

1. Cooper, P. W., The Hyperplane in Pattern Recognition, Melpar Technical Note 61/6, Melpar Applied Science Division (1961).

2. Cooper, P. W., The Hypersphere in Pattern Recognition, Melpar Technical Note 62/1, Melpar Applied Science Division (1962).

3. Sebestyen, G. S., Decision Making Process in Pattern Recognition, MacMillan Company, New York (1962).

4. Cooper, P. W., "The Hyperplane in Pattern Recognition," Cybernetics, 5, 215-238.

5. Cooper, P. W., "Hyperplanes, Hyperspheres, and Hyperquadrics as Decision Boundaries," ONR-COINS Symposium at Northwestern University, appearing in Computers and Information Sciences, Spartan Books, Washington, D.C. (1963).

6. Sommerville, D. M. Y., An Introduction to the Geometry of N-Dimensions, Dover Publications, New York (1958).

7. Cooper, P. W., and Cooper, D. B., "Nonsupervised Adaptive Signal Detection and Pattern Recognition," Information and Control, 7, 416-444.

8. Blackwell, D., and Girshick, M. A., Theory of Games and Statistical Decisions, John Wiley and Sons (1954).

9. Luce, R. D., and Raiffa, H., Games and Decision, Introduction and Critical Survey, John Wiley and Sons (1957).

10. Thomas, J. B., and Wolfe, J. K., "On the Statistical Detection Problem for Multiple Signals," IEEE Trans on Information Theory, Vol IT-8, pp 274-280 (July 1962).

11. Ogg, F. C., Jr., "A Note on Bayes Detection of Signals," IEEE Trans on Information Theory, Vol IT-10, pp 57-60 (Jan. 1964).

12. Wald, A., Sequential Analysis, John Wiley and Sons (1947).

13. Lee, R. C. K., Optimal Estimation and Control, MIT Press (1964).

14. Beckenbach, E. F., Modern Mathematics for the Engineer, McGraw-Hill Book Co. (1956).

15. Kemeny, J. G., and Snell, J. L., Finite Markov Chains, D. Van Nostrand Co., Inc. (1960).

16. Jackson, A. S., Analog Computation, McGraw-Hill Book Co. (1960).

17. Tou, J. T., Modern Control Theory, McGraw-Hill Book Co. (1964).

18. Freeman, H., Discrete Time Systems, John Wiley and Sons (1965).

19. Howard, R. A., Dynamic Programming and Markov Process, MIT Press (1960).

20. Kunz, K. S., Numerical Analysis, McGraw-Hill Book Co. (1957).

21. Ralston, A., and Wilf, H., Mathematical Methods for Digital Computers, John Wiley and Sons, p. 253 (1964).