In certain cases in system modeling, parameter search problems are complicated by the availability of scanty and corrupt physical system data and a large number of model parameters. An algorithm using man-machine interaction is presented to attack this type of parameter search problem with the objective of efficiency in the parameter search. The man-machine interaction allows model response evaluation by a man by providing the man with a convenient display of the data and the model responses.

The large number of parameters in the system model contributes to a slow parameter search. The algorithm presented uses computer learning to improve the parameter search efficiency. The algorithm is divided into a learning phase and a learned phase. In the learning phase the man teaches the computer his preferences for the model responses. Then in the learned phase the computer proceeds with the parameter search independently of the man by
means of its experience acquired during the learning phase. After a period of learned model response evaluation by independent action of the computer in the learned phase the control of the parameter search process is returned to the man. This terminates one iteration of the algorithm.

The problem of extracting features from model responses to train the computer is studied. A method to compare different feature extraction operations using entropy is presented.

The algorithm is tested on an example oceanographic problem. It is found that the success of the computer learning technique depends on the step size of the model parameter changes during the learned (computer directed) phase of operation and on the size of a learned region in feature space (a quantity calculated for response classification in the learned phase). Also, the algorithm is more efficient when the parameters are not too close to their optimum.
Parameter Estimation in System Modeling
Using Man-Machine Interaction and
Computer Learning

by

John Arthur Wagner

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Doctor of Philosophy

June 1970
APPROVED:

Redacted for Privacy

Associate Professor of Electrical and Electronics Engineering
in charge of major

Redacted for Privacy

Head of Department of Electrical and Electronics Engineering

Redacted for Privacy

Dean of Graduate School

Date thesis is presented ____________

Typed by Opal Grossnicklaus for ____________
ACKNOWLEDGEMENT

I would like to thank my major advisor Professor J. L. Saugen for his ideas on this research subject and for the assistance which he extended to me during my studies. My appreciation for assistance in the experimental portion of this thesis is given to my friends Carlton Cross, Rudy Frank and Rich Iverson. This research was supported in part by a research assistantship of the Engineering Experiment Station and in part by the Sea Grant Project. Also I would like to thank my instructors, the staff members, and my friends and acquaintances who have made my educational stay here so enjoyable. And, of course, thank you, Mary Jean, Peter, and Mark.
TABLE OF CONTENTS

I INTRODUCTION

Statement of the Problem 7
The Parameter Search Algorithm 8
Background 11
Discussion of the Algorithm 18
Results 23

II THE ALGORITHM

Definitions 27
The Functional Divisions of the Algorithm 29
Operation of the First Part of the Learning Phase 31
Operation of the Learned Phase 36
Operation of the Second Part of the Learning Phase 40
Transformations of the Algorithm 43
Miscellaneous Considerations 46
Example Problem 48
Statement of the Problem 48
The Original Solution 50
Results Pertaining to the Algorithm 50
Results Pertaining to the Parameters 55

III EVALUATING FEATURE EXTRACTION TECHNIQUES 62

Introduction 62
Description of the Problem 63
Feature Reduction: Subspace Projection 67
Quantization of the Response Space 71
Quantization of the Feature Space 74
A Means of Feature Space Reduction 75
Application to a Second Order Linear System 78

IV CONCLUSIONS 82

BIBLIOGRAPHY 85

APPENDIX 88
PARAMETER ESTIMATION IN SYSTEM MODELING
USING MAN-MACHINE INTERACTION
AND COMPUTER LEARNING

I INTRODUCTION

The problem attacked in this thesis is the determination of the parameters of a mathematical system model by comparing real system data with the system model response. The structure (or form) of the system model is given by a set of ordinary differential equations containing parameters of unknown value. An algorithm is developed to find these parameter values from badly corrupted data.

An example of such a system model is the set of ordinary differential equations given in (1).

\[
\begin{align*}
\dot{r}_1 &= a_{11} r_1 + a_{12} r_1 r_3 + a_{13} r_1 r_2 - m(t) r_1 \\
\dot{r}_2 &= a_{21} r_1 + a_{22} r_1 r_2 + a_{23} m(t) - m(t) r_2 \\
\dot{r}_3 &= a_{31} r_1 + a_{32} r_3 + a_{33} r_3^2
\end{align*}
\]

(1)

where \( r_i \) are state variables \( i = 1, 2, 3 \)

\( r_1 = \) plants (phytoplankton)

\( r_2 = \) nutrients (phosphates)

\( r_3 = \) animals (herbivores)

\( a_{ij} = \) parameters

\( m(t) = \) uncontrollable input (mixing coefficient)
This set of equations describes an oceanographic system. The model is derived from physical considerations [24].

It is hoped that ultimately predictions can be made from this system model about the supply of food fish in the sea. The microscopic organisms serve as the basic food for the larger fish. A good system model of the microscopic organisms is a necessary step toward the ultimate goal.

In (1) the \( a_{ij} \) parameters are initially unknown. The technique used here to find these parameters is an iterative process (an algorithm) called parameter search. The parameter search consists of finding a solution of the system model equations (system response) for some parameter values and comparing the system response with data obtained from the physical system. Based on the comparison of the model response and the system data a new set of parameter values are found. The algorithm provides directed search of these parameters so that the iterative process continually improves the fit of the model response to real data.

This algorithm is especially useful for, although not limited to, comparisons involving physical data that are scanty and corrupted. Such data is present in the example problem (1). The fact that the data is of "poor" quality is an important consideration in the choice of a method of comparing model responses and physical data. Figure 1.1 is an example of this type of data. Note that the data is scanty
and has a "large" range of error. The problem is to find the $a_{ij}$ parameters of (1) which produces model responses which best fit this data.

![Figure 1.1. Physical data obtained from an oceanographic system.](image)

This problem can be attacked using a technique of comparison which minimizes an index of performance (IP). This IP measures the error between the model response and the physical system data. For example, an IP commonly used is the integral of the squared error. For systems where there is a greater quantity of data available and less erroneous data a parameter search that uses an IP to evaluate the model response is an appropriate technique.

Since the type of parameter search problems under consideration here have "poor" data available from the physical system a man-machine interaction technique is considered more amenable to the problem than an IP technique. The man-machine interaction
technique allows the man who is directing the parameter search to use his intuitive knowledge of "reasonable" model responses and parameter values. Also during the parameter search he can make allowances for the physical system data.

In the man-machine interaction approach, shown in Figure 1.2, the model response and the physical system data are shown to the man on a computer output display. He first judges the model response in comparison to the physical system data. Then the man notifies the computer of his evaluation by means of a cueing box connected to the digital computer using sense lines. Sense lines carry cue information to the computer and are interrogated during execution of the algorithm. Decisions on parameter changes are made partly by the man's cue and partly by the computer. The parameter value change is carried out and a new model response is generated for display. This process is continued iteratively until in the man's judgement the response and data are best fitted.

The algorithm developed here is designed for a parameter search problem which has a large number of parameters in the system model. For instance, in the equations (1) the parameter search must find the values of the nine model parameters \( a_{ij} \). Since the algorithm must search over a large number of parameters it is desired to incorporate into the algorithm a technique for making the parameter search an efficient operation. This is done by allowing
Figure 1.2. Man-machine interaction.

- Parameters of the model
- Model response
- Physical data
- Storage oscilloscope display
- Decision on parameter changes
- Cueing box
- Error evaluation by man

Man's function
the parameter search to enter a computer directed phase of operation which is described in the following paragraphs.

Man-machine interaction has an inherent inefficiency. This inefficiency is the delay in the progress of the parameter search while the computer waits for the man to evaluate the model responses. The computer learning technique renders the parameter search more efficient because the computer directed action takes advantage of the speed of the computer. Moreover, as is explained in following sections, the computer directed action culls out model responses which are unlikely to be preferential responses. Thus, the algorithm operates more efficiently subsequent to the computer directed action because the man must view relatively fewer "bad" model responses.

Computer learning is a technique, which, as included in this algorithm, incorporates both a man directed parameter search and a computer directed parameter search (Figure 1.3). After a period of man directed parameter searching the computer assumes control of the parameter search and carries on the searching independently of the man. The resultant parameter search can be more efficient because advantage is taken of the computer's speed.

Figure 1.3 illustrates the concept of computer learning as used in this thesis. The parameter search becomes a two phase process. In the man directed phase the computer "learns" from the man's cues of the characteristics of the model responses which satisfy the man.
In the computer directed phase the computer acts independently of the
man and makes judgements on the model responses based on the char-
acteristics learned from the man. The two phases alternate in the
algorithm as the search proceeds. The search terminates upon
command from the man.

Two phase
parameter
search

Man directed
parameter search.
Computer "learns."

Computer directed
parameter search.
Computer acts indepen-
dently of man.

Figure 1.3. Computer learning.

Statement of the Problem

The problem attacked in this thesis is stated as follows. A set
of equations whose form is fixed is given as a model of a physical
system. ¹ This set of equations has a "large" number of parameters

¹For example, these equations can be algebraic, ordinary dif-
ferential, or partial differential equations. The only limitation is the
ease with which the equations can be implemented and the correspond-
ing responses displayed.
of unknown value. The data available from the physical system is scanty and corrupted by noise of unknown statistics. The problem is to design an algorithm that will perform an efficient search for the parameter values, notwithstanding the "large" number of parameters and the "poor" physical system data.

For example, the equations of (1) are given as a model of a physical system of fixed form. This set of equations has nine unknown parameters. Figure 1.1 illustrates data from this system. This data is "sparsely" distributed in time and the data values are "largely" unreliable. The problem in this example is to find in an efficient manner the "best" values of the $a_{ij}$ parameters of (1).

The Parameter Search Algorithm

Figure 1.4 shows the salient features of the algorithm. They are 1) man-machine interaction which is used to attack the problem of poor data available for the parameter search and 2) the computer learning technique which is employed to improve the efficiency of the parameter search in the presence of a large number of parameters in the system model.

Man-machine interaction includes model response display, the man's evaluation of the model responses, and the sending of information on the evaluation of the responses to the computer by the cues of the man. The computer finds the next model parameters,
Start

Man selects from a display preferential responses. Computer learns.

Computer learning sufficient for computer directed research?

no

Man directed phase

yes

Determines possible preferential responses

Man evaluates by a display possible preferential responses. Computer learns.

Computer directed phase

Man directed phase

Figure 1.4. General concepts of the algorithm.
performs computational and bookkeeping operations, and presents the model response display to the man terminating one iteration.

The computer learning technique is composed of a man directed phase where the computer learns and of a computer directed phase where the computer operates on its experience gained from the man. In the man directed phase the computer learns from information extracted from the model responses. This information consists of a set of characteristics of the response. The man knows of these characteristics and trains the computer based in part on this knowledge.

These general concepts are illustrated in Figure 1.4. The man selects from a display of model responses his preferential response. The computer "learns" from the repeated judgement of the man.

The operation of the algorithm proceeds to a branch point where the man must decide whether the computer has sufficient information to enter the computer directed phase of the algorithm. He bases this decision on his knowledge of the algorithm and its details of operation. For example, he knows a priori what characteristics of the response the algorithm uses in training. An example of a detail of algorithmic operation is that he learns how the parameter step size affects the optimum model response.

If the computer learning is sufficient for a computer directed search the operation leaves the man directed phase and enters the computer directed phase. In this phase the computer determines
from the experience taught to it by the man any possible preferential responses (Figure 1.4). The computer does this independently of the man by iteratively changing the model parameters, finding model responses, and testing them for their possibility to rate as preferential responses.

After the algorithm has iteratively obtained and tested an arbitrary a priori number of test responses the operation returns to the man directed phase. Upon re-entering this phase the man evaluates the possible preferential responses found by the computer. The computer learning is updated again based on the man's cues. The operation of the algorithm returns to the branch point (Figure 1.4) where the operation repeats itself.

**Background**

A **system model** is defined as a set of equations which describe a real physical system. That is, when the solution to the equations sufficiently matches the behavior of the physical system then the equations are said to be a system model.

A **parameter** is a real time-invariant coefficient in the system model that can be changed to affect the solution of the equations (model response). That is, the solution of the model equations is a function of the parameters. The parameters of the system model are assumed to be independent of each other.
Two basic problems which must be solved in order to obtain a final set of equations which constitute the system model are 1) the formulation of the form or structure of the equations in terms of unknown parameters and 2) the determination of the values of the parameters of the model (parameter search). In order to solve the problem of formulating the model the use of physical considerations are of paramount importance. The problem of finding the parameter values can be often solved by methods which optimize a mathematical quantity. The optimization methods usually minimize iteratively some kind of index of performance (IP), a measure of model quality.

One method of obtaining the form of the system model separates the physical system into smaller pieces from which "submodels" are formulated [16]. Physical considerations lead to the separation of the physical system into components. Each component becomes a submodel for which equations can be written. By connecting the submodels together the whole system model is obtained.

Rather than break the whole system into smaller submodels another approach in determining the form or structure of the system model is to view the physical system as one unit with a known set of inputs and outputs. With the information on the physical system inputs and outputs the set of model equations are chosen to behave like the physical system regardless of the internal model structure. The individual terms of the model equations do not have any physical
meaning in this input-output approach to finding the model form.

The two approaches to finding the model form, the internal, submodel approach and the input-output approach, can be illustrated by a simple example. Consider the physical system a tightly wound coil of wire. The data of the physical system is a plot of current $i$ through the coil versus time $t$ as a result of a voltage impulse being applied to the coil. The submodel approach uses the reasoning that the coil has both the properties of resistance and inductance inherent in it. This approach separates the coil into submodels of an ideal inductor $L$ and a series resistor $R$. The system model that yields a response similar to the physical system is given by the differential equation $\frac{di}{dt} + \frac{R}{L}i = 0$. The second approach does not attempt to separate the real coil into submodels but just yields the differential equation $\frac{di}{dt} + ai = 0$ where $a$ is a constant coefficient. In the two approaches the parameters are $R$ and $L$, and $a$ respectively. After finding the model form and the parameters the next step is to find the values of the model parameters.\(^2\)

Reference [5] presents a comparison of some methods used in system modeling. It points out that a model gives a means of predicting the behavior of the process over a short enough interval of time.

\(^2\)Determining a system model form and finding appropriate parameter values is often called system identification [1].
Reference [10] presents a linear time varying "learning model configuration." It states that in a great majority of cases, the modeling problem is formulated into one of parameter estimation.

Several references [1, 22, 25] show that stochastic systems require an approach to system modeling that is somewhat different than the one used in this thesis.

The so-called optimum solutions to the parameter search problem found using IP methods are still often subject to the designer's judgement. Currently, techniques to find the values of the model parameters range from analysis to synthesis and from trial-and-error to automatic adjustment based on measurements of functions of the system variables. Commonly techniques rely on a scalar measure of performance (an IP) to find the set of parameter values, e.g. the integral of the squared error. Even though a system has optimum performance in terms of a given IP, the goodness of the system has to be measured by its time behavior [18]. Furthermore, the final judgement on this time behavior of the system model is often the designer's.

The fact that the final judgement on the goodness of the model often depends on the designer's judgement of the model response lends credence to the use of a man-machine interaction algorithm. That is, for some parameter search problems a man-machine interaction technique where judgement of the model response is entirely
the designer's is an appropriate approach.

Reference [21] is on the use of man-machine interaction as an aid in network design. It is pointed out that the benefits are obvious if a designer could "converse" with a computer in a language closely related to the design process. A good system would have the characteristics of swift computer response to user commands, graphical output and perhaps graphical input.

Work involving man-machine interaction has been done [2] on modeling what a man does when he is sitting in front of a computer console. The man-machine interaction technique in this thesis is briefly described in the Introduction (and in more detail in Chapter II). This description uses Figure 1.2 which contains a box labeled "man's functions." This thesis does not attempt to describe the man in functional or mathematical terms as in [2]. Rather, an attempt is made to determine which characteristics of the model response the man uses in his evaluation activity and how the computer can use this information as an aid in the parameter search. Further discussion of this point which relates to computer learning and feature extraction techniques is in Chapters II and III.

For another viewpoint on man-machine interaction, reference [18] contains a discussion on the interaction between the man, the machine and the implementation of an algorithm. There it is found that the algorithm serves as the interpreter between the man and the
system model. The inputs to the algorithm are cues from the man. The algorithm translates the cues into various computer actions. Also the algorithm supervises the system model.

References [17, 18] describe a parameter search process. This parameter search includes a "global" investigation to find a starting point in the parameter search and a "regional" investigation of parameters. The algorithm in [17, 18] using these techniques can effectively handle about four parameters.

In the algorithm presented in this thesis no "global" search of the parameters is made as in [17, 18] to find a starting point in the parameter search. Rather, engineering judgement is used to find the starting point of the parameter search. In the algorithm presented in this thesis a parameter search technique is designed (quite differently than the parameter search technique in [17, 18], for details see Chapter II) for a system model with a larger number of parameters than four as in [17, 18]. For instance the parameter search technique can handle the nine parameters in the example problem (1).

In previous man-machine interaction work [17] the display shown to the man contained one model response which included three state variables. The model equations were solved by an analog computer repetitively in order to display the responses on a non-storage type oscilloscope. The operator was required to retain in
his memory the responses shown to him previously.

In comparison, the algorithm here is written to utilize the advantages of a storage oscilloscope display. Namely, that on this type of display several responses and several state variables can be displayed. Also data from the physical system can be displayed to the man. Figure 1.6 is an example of the display shown to the man for the example problem (1). On this display there are two responses with three state variables each. Also on the display are data from the physical system that have been entered a priori into the algorithm.

In this thesis concepts have been borrowed from pattern recognition theory [9, 14, 19]. A pattern classification system consists of a pattern to be classified, a feature extraction operation on the pattern, and a classification operation on the features. Many pattern recognition systems are first "taught" and then operate by classifying patterns based on their gained experience.

System model responses can be regarded as patterns. Removing characteristics from the responses during the man operated phase of the algorithm is equivalent to a feature extraction operation on a pattern. The object of removing these features from these patterns is to obtain information to be used in the computer controlled phase of the algorithm.

The man directed phase is equivalent to the "teaching" operation that many pattern recognition systems undergo. The computer directed phase is essentially a pattern classification system in
operation. This pattern classification takes place when the algorithm classifies a pattern as a possible preferential response or otherwise. Chapter II contains more discussion on this subject.

**Discussion of the Algorithm**

The two techniques, man-machine interaction and computer learning, have been included in the algorithm as methods to solve the parameter search problem. The reason for using the man-machine interaction technique is that it can manage the poor data from the physical system more suitably than an IP technique. The reason for using the computer learning technique is an aid for an efficient parameter search. The need for the efficiency arises from the fact that the system model has a large number of parameters, as in example (1).

A parameter search of increased efficiency is the objective of incorporating the computer learning technique with the parameter search. The point is that a computer directed parameter search has the potential to make model response evaluations faster than the man directed parameter search. The reason for this is the computer's high speed of operation compared to a man's.

In the man directed phase the computer learns from the cues of the man (Figure 1.3). In the computer directed phase the computer acts independently of the man and makes judgements on test
model responses based on the characteristics learned from the man. Following learning system terminology in the literature [11] the computer directed phase will be referred to as the learned (trained) phase and the man directed phase will be referred to as the learning (training) phase.

"Learning" generically speaking involves experience and some future action based on the experience. Computer learning [28] is the process by which the computer acquires experience from the man during the learning (man directed) phase of the algorithm. The computer takes action based on this acquired experience during the learned (computer directed) phase of the algorithm.

The computer learns as follows (Figure 1.5). During the learning phase of the algorithm the computer displays the model response to the man. Interacting with the computer the man evaluates the information displayed to him. He then makes a judgement on the information and informs the computer (by pressing a push button). The information on the man's evaluation is processed in such a way that the computer acquires experience, i.e. becomes trained. The computer training is accomplished by the algorithm storing certain quantities for use later during the trained phase of

---

3 "Learn" and "train" and their derivatives are used interchangeably in this thesis.

4 Two other terms appearing in the literature corresponding to the term "learning phase" are "training procedure" and "adaptation" [6].
Figure 1.5. The elements of computer learning.
the algorithm operation. Through his own experience the man makes the decision whether the algorithm should proceed to the learned phase.

Entering the learned phase the computer establishes a criterion based on its learning which it will use next to evaluate system model responses independently of the man. A set of model responses are generated iteratively by computer variation of the parameters. The computer acting independently of the man then evaluates the set of model responses using its learned criterion.

The display used in man-machine interaction is discussed in the following two paragraphs. In Figure 1.6 the response on the left shown to the man (consisting of three state variable responses) is the best model response $r_b$ found so far by the man. The model response on the right is called the next model response $r_n$. The man compares the next model response $r_n$ with the best model response $r_b$.

Each of the responses in Figure 1.6 is the solution of the model equations for a different set of parameter values. That is, the best model response $r_b$ is found for one set of parameter values, then the computer, according to the algorithm, changes the parameter values and solves the model equations again producing the next model response $r_n$. 
Figure 1.6. The display shown to man.
Results

The main contribution of this thesis is the development of an algorithm that efficiently solves the parameter search problem which is complicated by a large number of model parameters and poor physical system data. The unique characteristic of this algorithm is that it has the ability to gain experience from the response evaluation activity of the man and then proceed in the parameter search independently of the man. It proceeds in the parameter search based on the experience acquired from the man.

It was found that the computer learning technique aided the parameter search process most effectively when the parameter values were still not too close to the optimum. That is, experience with the algorithm shows that when the model response still requires considerable improvement the learned phase of the algorithm helps to speed up the parameter search process. In the parameter search when the model response fits the data more closely then the learned phase of the algorithm does not effectively aid the parameter search process. The reason for this result is due to a detail of the algorithm involving the size of the parameter variation in the learned phase of operation.

Several secondary results or attributes of the algorithm have been found. These attributes are features of the algorithm that
make it useful for parameter search problems.

The first attribute of the algorithm is that a good evaluation of the model responses by the man is possible. A good evaluation is possible because of the convenience of the display. Three items are displayed to the man, 1) the physical system data, 2) the best model response to date, and 3) the next model response (Figure 1.6). He can easily compare the next model response with the best model response to date against the physical system data and judge whether the next model response fits the data better than the best model response to date. Another factor contributing to a facile evaluation of the model response is the fact that the man does not have to rely on his memory for the data or the best response to date. Consequently, since many items may be displayed simultaneously the storage oscilloscope display is superior to the method using an ordinary oscilloscope as the display device.

The second attribute of the algorithm is that the display allows the man to use his intuitive knowledge in judging the reasonableness of the model response. The man observing the display may know things like the validity of certain data points or the importance of one time segment of a response relative to another time segment of the response. This type of consideration can be made with the man-machine interaction technique whereas this type of consideration is difficult to include in a parameter search algorithm using
During the parameter search the algorithm provides for the man the current parameter values of the system model. At any point in the parameter search the man can cease his response evaluation activity and check the current parameter values corresponding to the best response to date.

The third attribute is that the man can determine the reasonableness of the parameter values. The man from experience can determine quantities like upper and lower bounds on parameter values, signs of parameters, and orders of magnitude of the parameters. Hence he can monitor the parameter values during the parameter search progress.

For example, an oceanographer observing the responses of the system model (1) has knowledge of reasonable responses for the plants, animals and nutrients of the oceanographic system. This particular model (1) is derived from physical consideration so that the parameters $a_{ij}$ have physical meaning to the oceanographer. Also, for instance, if the oceanographer had been responsible for gathering the data from the sea then he knows his own level of confidence in the data points.

The fourth attribute is that experience using the algorithm shows that the success of the learning technique depends on two fixed quantities. The two quantities, which will be explained in
detail in Chapter II, are the parameter step size and the shape of the learned region. For now, both of these quantities are used during the trained phase and influence the trained computer's evaluation of model responses.

The values of these two quantities are arbitrary and are selected according to the judgement of the man. Conceivably the algorithm could be modified so that automatic adjustment of these quantities would be possible. This modification would vary these two quantities in a suitable way for the parameter search.

The results of the learning technique are mixed depending on the values assigned these two mentioned quantities. That is, experience with the algorithm shows that at times the computer learning does in fact fulfill its objective, namely, to aid the computer search process. At other times experience seems to show that the algorithm is not made more efficient by the man exercising the learned phase option. That is, when the parameter search is close to the optimum, the algorithm is not made more efficient in terms of either the number of cues the man is required to make or the amount of time he must spend in the searching process.
II THE ALGORITHM

Some terms must be defined to permit a precise explanation of the algorithm.

Definitions

The set of equations that are the system model are assumed to be of the form \( \dot{r} = f(r, \pi, u(t)) \). \( r \) is the response vector of the system model. \( r \) is of arbitrary dimension. \( \pi \) is a parameter vector of finite dimension. \( t \) is time. \( u(t) \) is an input vector of arbitrary dimension. The parameter vector \( \pi \) is assumed to be a member of the parameter space \( \mathcal{P} \) where the dimension of \( \mathcal{P} \) is equal to the number of parameters [8]. The dimension of \( \mathcal{P} \) is finite because of physical and implementational considerations. These considerations also dictate that the parameter vector \( \pi \) must lie in a finite volume of parameter space i.e. the magnitudes of the parameter vector \( \pi \) must have finite limits. This volume in \( \mathcal{P} \) is called the allowed parameter sector \( \mathcal{P}^s \).

Each response \( r \) (or pattern)\(^5\) is a vector in response (or pattern) space \( \mathcal{R} \) of arbitrary dimension. For instance, if the

---

\(^5\)The output of the model is referred to as either a response or a pattern depending on the context.
model response is a continuous curve then the response space $\mathcal{R}$ must be of infinite dimension in order to completely describe the curve. Of course in all practical applications the dimension of $\mathcal{R}$ must be finite.

A feature $f_i$ of a response is a characteristic of a response $r$ obtained by a mathematical operation [26] on $r$ where $i$ is an integer. For example, a feature $f_i$ of a response $r$ can be the rise time or the overshoot of the response, or $f_i$ can be a sample of $r$ at a particular time, or $f_i$ can be one Fourier coefficient of a Fourier series expansion of $r$. A feature vector $f$ of a response $r$ is a set of features $f_i$ of the response, i.e. $f = (f_1, f_2, \ldots, f_n)$ where $n$ is an integer. A feature vector $f$ is said to be a member of a feature space $\mathcal{F}$ where the dimension of $\mathcal{F}$ is equal to the number of features of the response $r$. The process of obtaining features of a response is called feature extraction [19].

For example a feature vector $f$ of a response $r$ can be a set of samples of $r$, or it can be a finite number of the Fourier coefficients from a Fourier series expansion of $r$. Each dimension of the feature space $\mathcal{F}$ would then correspond to one sample, in the first case, and to one Fourier coefficient, in the second case. Furthermore, feature extraction in the first case would correspond to the process of sampling the response $r$, and feature extraction in the second case would correspond to the mathematical operation of calculating the Fourier series coefficients of $r$. 
Before proceeding to a more detailed explanation of the algorithm an explanation of the functional parts of the algorithm is presented. This explanation attempts to correlate the man-machine interaction technique and the learning technique with the logical flow of the algorithm.

**The Functional Divisions of the Algorithm**

Functionally the algorithm is separated into three divisions, 1) the first part of the learning phase, 2) the learned phase, and 3) the second part of the learning phase. Figure 2.1 shows this separation. A more detailed description of each part is given in the next three sections of this thesis.

Man-machine interaction is present in the first and second parts of the learning phase. Consequently, the parameter search is man directed in these two parts of the learning phase.

Figure 2.1 shows that a branch point separates the learning phase from the learned phase. At this branch point a decision must be made whether or not the computer is sufficiently trained. This decision is made by the man who uses his knowledge of the algorithm and its details of operation. For example, he knows which characteristics of the responses the algorithm uses in training and he knows other algorithm details, like storage space available, etc. Also an intuitive "feel" for the amount of information that is sufficient for a
satisfactory computer directed phase is obtained by experience using the algorithm.

In the learned phase the computer acts independently of the man. In this phase the parameter search is computer directed rather than man directed.

The chief difference between the first and second parts of the learning phase is the method by which the parameters are varied. In the first part the step size of the parameters can be changed. In the second part of the learning phase the step size remains fixed.
Additional explanation of the method of changing parameter step size is given in a later section and in the Appendix.

The next three sections present the operation in more detail of each of the three functional divisions of the algorithm.

**Operation of the First Part of the Learning Phase**

Two parameter vectors that are used by the learning phase are defined. The **best parameter vector** $\pi_b$ is the value of the parameter vector in the model equations when the solution to the model equations $\dot{r} = f(r, \pi_b, u(t))$ is the best model response $^6 r_b$. Similarly, the **next parameter vector** $\pi_n$ is the value of the parameter vector in the model equations when the solution to the model equations $\dot{r} = f(r, \pi_n, u(t))$ is the next model response $r_n$ where $\pi_b, \pi_n \in \mathbb{P}^s$.

Figure 2.2 illustrates the first part of the learning phase. Three items are displayed to the man, 1) the best response $r_b$, 2) the next response $r_n$, and 3) the physical system data. The algorithm sets the parameter vector $\pi$ in the model equations to the value of the best parameter vector $\pi_b$, solves the equations and displays the response $r_b$. Similarly, the algorithm sets the parameter vector $\pi$ to the value of the next parameter vector $\pi_n$, solves the equations and displays the response $r_n$.

---

6 The subscript b stands for best and the subscript n stands for next.
Figure 2.2. The first part of the learning phase.
The man observes the display and evaluates the model responses \( r_b \) and \( r_n \). His objective is to find a response to best fit the data. With this objective in mind he can make three judgements, \( r_n \) fits the data 1) better than \( r_b \), 2) the same as \( r_b \), or 3) worse than \( r_b \).

From each next response \( r_n \) the algorithm extracts a feature vector \( f \). If the response \( r_n \) is judged to be better than the best response \( r_b \) then the feature vector \( f \) is stored and is called the stored feature vector \( f_s \). The stored feature vector \( f_s \) contains the information on the response \( r_n \). This information is the "experience" that is required for learning. There is no provision in the algorithm to train the computer from cues on responses \( r_n \) judged to be the same as, or worse than, the best response \( r_b \). Hence, whenever the operator judges \( r_n \) to be better than \( r_b \) computer learning is taking place.

The next step in the algorithm (Figure 2.2) is the calculation of the next parameter vector \( \pi_n \). This calculation depends on the cues of the man and on several quantities internal to the algorithm. Briefly, for the example problem (1) one dimension of \( \pi_b \) is changed and this vector becomes \( \pi_n \). If the man judges \( \pi_n \) better than or the same as \( \pi_b \) then \( \pi_n \) is stepped in the same direction. If the man judges \( \pi_n \) as worse than \( \pi_b \) then \( \pi_n \) is stepped in the opposite direction.\(^7\)

\(^7\)The subscript \( s \) stands for stored.
direction. Refer to the Appendix for a complete description of the way \( \pi_n \) is changed.

Following the selection of \( \pi_n \), the man must determine whether the learning is sufficient for a satisfactory learned phase (Figure 2.2). His determination is dependent on his own experience in using the algorithm. If the learning is sufficient the operation passes on to the learned phase, otherwise the operation returns to the start and another display is presented.

If the man judges \( r_n \) the same as, or worse than \( r_b \) the algorithm changes the old \( \pi_n \) to a new \( \pi_n \). On the next display a new \( r_n \) appears along with the old \( r_b \). On the other hand, if the man judges \( r_n \) better than \( r_b \) the algorithm replaces the old \( r_b \) with \( r_n \). On the next display a new \( r_n \) appears with the new \( r_b \). This process repeats iteratively so that the so called "best response" on the display continually appears to fit to the data better.

The algorithm generates by this iterative process a series of responses that are temporarily called the best response \( r_b \). This series of responses is called the preferentially ordered responses \( r_b^k \) where \( k \) is a finite integer. That is, the preferentially ordered responses are a series of vectors \( r_b^1, r_b^2, \ldots, r_b^k, \ldots, r_b^m \) where \( r_b^k \) is a general term and \( m \) is a finite integer. According

\[ ^8 \text{The superscript on a vector denotes different vectors in the same vector space.} \]
to the man's judgement $r_b^i$ is better than $r_b^j$ where $i$ is an integer greater than $j$.

For each response $r_b$ a feature vector $f_s$ is stored. Hence, the preferentially ordered responses $r_b^k$ generate a series of preferentially ordered feature vectors $f_s^k$ where $k$ is a finite integer. These vectors $f_s^k$ are used in calculations in the learned phase.

Figure 2.3 shows an example of a series of preferentially ordered responses $r_b^k$ and the corresponding feature vectors $f_s^k$. 

Figure 2, 3. An example of a series of preferentially ordered responses and the corresponding feature vectors.
Four response vectors $r_b^1, r_b^2, r_b^3, r_b^4$ are illustrated in response space $\mathcal{R}$ along with a next response vector $r_n$. Note that $r_b^4$ is better than $r_b^3$, $r_b^3$ is better than $r_b^2$, and $r_b^2$ is better than $r_b^1$.

In the feature space $\mathcal{F}$, the four corresponding preferentially ordered feature vectors $f_s^k, k = 1, 2, 3, 4$, are shown. The feature extraction operation is represented by the arrows and the symbol $T_2$. More is said about $T_2$ in a later section.

**Operation of the Learned Phase**

The training information the man gives the computer during the learning phase consists of the set of preferentially ordered feature vectors $f_s^k$. This set of vectors in feature space $\mathcal{F}$ shows the man's tendency for what he prefers about the observed changes in the responses. For instance, consider the response of a second order system. Characteristics of this system's response which are good candidates for features are rise time, overshoot, and settling time. These characteristics would make good features because these quantities are the ones the man evaluates when judging the response.

The parameter space $\mathcal{P}$ is another space where it is possible to find a tendency for what the man prefers about the responses in a manner similar to the tendency in feature space $\mathcal{F}$. However, the point here is that the tendency in $\mathcal{F}$ is more directly related to the choice of feature improvements of the man than is the corresponding
tendency in $\mathcal{F}$. This idea of using the feature space $\mathcal{F}$ of the responses as the medium for storing information on the tendency of the man's preferences to be used later in training the computer is felt to be a new contribution in the area of parameter searching in system modeling.

From the preferentially ordered feature vectors $f_s^k$, a vector called the preferential direction $g$ is calculated. Figure 2.4 shows a set of $f_s^k$, $k = 1, 2, 3, 4$, and a preferential direction $g$. For instance, $g$ can be calculated as the average of the $f_s^k$ differences, i.e., for Figure 2.4 $g = 1/3((f_s^2 - f_s^1) + (f_s^3 - f_s^2) + (f_s^4 - f_s^3))$.

The preferentially ordered feature vectors $f_s^k$ and the preferential direction $g$ are used to calculate an area in feature space called the learned region $F^l$. For example, Figure 2.4 shows the learned region $F^l$ as a cone with its base near the last preferential feature vector.

The learned region $F^l$ is a man-taught volume in feature space $\mathcal{F}$. This region marks off an area in $\mathcal{F}$ where it seems reasonable to expect the next preferential feature vector to occur. This learned region $F^l$ is used in the basic operation of the algorithm in the learned phase.

---

The superscript $^l$ stands for learned.
Figure 2.4. An example in 2-dimensional feature space.

Figure 2.5 illustrates the operation of the learned phase. The first operation computes the preferential direction \( g \) and the learned region \( F^g \). The next operation arbitrarily selects a test parameter vector \( \pi_t \). This test parameter vector is used in the model equations to produce a test response \( r_t \). Since the learned phase is computer controlled the test response \( r_t \) is not displayed to the man.

The next operation of the algorithm (Figure 2.5) is the extraction of a test feature vector \( f_t \) from \( r_t \). The algorithm then determines whether \( f_t \in F^g \). If \( f_t \in F^g \) then \( f_t \) is called a learned feature vector \( f^g_\lambda \). In this operation the computer acts as a two class pattern recognition system. More is said about the pattern classification in following sections.

\(^{10}\) The subscript \( t \) stands for test.
From the first part or the second part of the learning phase (Figure 2.2, 2.6)

1. Compute a preferential direction $g$ and a learned region $F$ in feature space $\mathcal{G}$
2. Select a test parameter vector $\pi_t$
3. Solve the system equations for the test response $r$
4. Extract from $r_t$ a test feature vector $f_t$
5. **Decision**: $f_t \in F$?
   - **Yes**: Save $\pi_t$ as a learned parameter vector $\pi_{\mathcal{G}}$.
   - **No**: Parameter variation complete?
     - **Yes**: To the second part of the learned phase (Figure 2.6)
     - **No**: Go back to step 4.

Figure 2.5. The learned phase.
If the test feature vector $f_t$ lies in the learned region $F^\lambda$ then the test parameter vector $\pi_t$ is saved as a learned parameter vector $\pi^\lambda$. Otherwise $\pi_t$ is not saved. $\pi^\lambda$ is used in the second part of the learning phase.

The learned parameter vector $\pi^\lambda$ produces a solution of the model equations which is likely to be a preferential response. This likely preferential response is called a learned response $r^\lambda$.

Since the responses generated in the learned phase are not displayed it is not known at this point whether the learned response $r^\lambda$ is in fact better than the best response $r_b$. This decision takes place in the second part of the learning phase when $r^\lambda$ and $r_b$ are displayed to the man.

The next operation in the learned phase is a branch point (Figure 2.5) that queries whether the parameter variation is complete. The parameter variation in the learned phase is under computer control. If the parameter variation is not complete a new test parameter vector $\pi_t$ is selected. Otherwise the algorithm operation goes on to the second part of the learning phase.

**Operation of the Second Part of the Learning Phase**

The algorithm displays the best response $r_b$ and one of the learned responses $r^\lambda$ (Figure 2.6). The responses $r_b$ and $r^\lambda$ are solutions to the model equations for the best parameter vector $\pi_b$. 
and one of the learned parameter vectors $\pi^x$.

The man judges the learned response $r^x$ as better, the same, or worse than the best response $r_b$. The difference between this display and the one in the first part of the learning phase is that the learned response $r^x$ is more likely to be a better response than the next response $r_n$ of the first part. That is, $r^x$ has already undergone some selection by the learned phase whereas $r_n$ has not been selected in a special way.

When $r^x$ is judged better than $r_b$ two actions are taken by the algorithm, 1) the stored feature vectors $f_s$ are updated and 2) the learned response $r^x$ replaces the best response $r_b$. Otherwise no action is taken by the algorithm.

The next operation in the algorithm is a branch point (Figure 2.6). The man decides whether he wants the operation of the algorithm to return to the first part of the learning phase. For instance, one reason for returning to the first part of the learning phase is a need for a change in the size of the parameter variation.

The next branch point in the algorithm is computer controlled. If all the learned responses $r^x$ are displayed then the operation proceeds to the learned phase.

The main difference in the first part and the second part of the learning phases is the way in which the parameters are varied. In the first part the parameter step size is partially under the man's
From the learned phase (Figure 2.5)

Display the best response \( r_b \) and the learned response \( r_A \)

Man judges the learned response \( r_A \) as better, the same, or worse than the best response \( r_b \)

Is \( r_A \) better than \( r_b \)?

- Yes: Update the saved features \( f_s \). \( r_b \) becomes \( r_A \)
- No: Return to first part?

Return to first part?

- Yes: To the first part of the learning phase (Figure 2.2)
- No: All \( r_A \) displayed?

- Yes: To the learned phase (Figure 2.5)
- No:

Figure 2.6. The second part of the learning phase.
control and partially under a rather complicated routine in the algorithm. In the second part of the learning phase the parameter step sizes remain fixed at the values found in the first part, although the parameter vector $\pi$ itself of course varies. If none or only a few of $f_t \in F^t$ then the man must return to the first part of the learning phase to update the training and change the parameter step size.

**Transformations of the Algorithm**

The purpose of mentioning the transformations of the algorithm is a way of clarification of the underlying principles of the algorithm. This discussion should lead to a better understanding of the algorithm.

The solution of the system model equations, the operation of feature extraction and the operation of the trained computer can each be regarded as a transformation between vector spaces [6]. These three transformations are illustrated in Figure 2.7.

![Figure 2.7. Transformations of the algorithm.](image-url)
Parameter space $\mathcal{P}$ is a vector space where each dimension corresponds to one of the parameters in the system model equations. Response space $\mathcal{R}$ is a vector space of arbitrary dimension whose member vectors are the responses of the model equations. The solution to the model equations can be regarded as a transformation $T_1: \mathcal{P} \rightarrow \mathcal{R}$, i.e. $T_1$ maps each parameter vector $\pi$ in the allowed parameter sector $P^s$ into a response $r \in \mathcal{R}$.

Feature space is a vector space where each dimension corresponds to one feature extracted from a response $r \in \mathcal{R}$. The feature extraction operation is a transformation $T_2: \mathcal{R} \rightarrow \mathcal{F}$. Classification space $\mathcal{C}$ in this thesis is a one dimensional vector space. The trained computer performs the transformation $T_3: \mathcal{F} \rightarrow \mathcal{C}$ on each feature vector $f$, i.e. the trained computer classifies each vector in $\mathcal{F}$ as either a learned feature vector $f$ or otherwise.

As an example of transformation $T_1$ consider the example problem of (1). The parameter vector $\pi$ corresponds to the coefficients in the model equations $a_{ij}$. The allowed parameter sector $P^s$ is determined by the practical limits set upon the $a_{ij}$ parameters by both physical and implementational considerations. The model responses $r$ are illustrated in Figure 1.6.

The method of extracting features in the example problem (1) demonstrates the transformation $T_2$. In the algorithm written to solve this problem the system model equations provide a solution for
100 days in time. Samples are taken of each state variable $r_1$, $r_2$, $r_3$, at 10, 30, and 90 days. Hence $T_2$ maps $r$ into a feature vector of nine dimensions, i.e. $f = (f_1, f_2, \ldots, f_9)$ where $f_i$, $i = 1, 2, \ldots, 9$, is the value of one of the samples.

The example problem (1) provides an example of the pattern classification transformation $T_3$. Whenever a test feature vector $f_t$ is in the learned region $F^g$ (a hypercone) (Figure 2.8) then this fact corresponds to a "1" in classification space $C$. Otherwise $f_t$ corresponds to a "0" in $C$.

Figure 2.8. An example of the classification transformation $T_3$. 

\[ \text{Hypercone } F^g \]

\[ \text{Classification space } C \]
The two special areas, the allowed parameter sector $P^s$ and the learned region $F^l$, are illustrated in Figure 2.7.

**Miscellaneous Considerations**

This section contains consideration of several miscellaneous topics of the algorithm. These include a discussion of the opposing requirements of feature vectors, a mention of the new contribution regarding the preferentially ordered feature vectors, an illustration of a series of learned regions, and a remark on algorithm implementation.

The requirements of 1) extracting sufficient information from the responses and 2) the limitation of feature vector storage space are often opposing requirements [19]. That is, sufficient information on the responses is desired to avoid equivocation, and on the other hand, greater information extracted from the responses demands larger storage requirements. Hence, because of computer storage limitation the series of stored feature vectors $f_{sk}$ must be finite both in feature space dimension and in the number of feature vectors.

The series of preferentially ordered responses found in the learning phase produces a series of parameter vectors in $\mathcal{P}$ which are indeed similar to the preferentially ordered feature vectors $f_{sk}^l$. It would be possible to use the series of parameter vectors in $\mathcal{P}$ to find a preferential direction and a learned region in $\mathcal{P}$ rather than
in $\mathcal{F}$. However, the man does not observe the parameters, he observes the response and evaluates the features of the response. The series $f^k$ are more representative of the man's tendency for preferential responses. Until now this area has been an unexplored area of research.

The algorithm functions more smoothly if the features are chosen to represent those characteristics by which the man evaluates responses. Otherwise an incorrect preferential direction can be calculated. Consequently the user of the algorithm must make an intelligent decision in choosing the features of the responses.

It is possible to envision a series of iteratively produced learned regions $F^k$. Figure 2.9 shows the series beginning at a starting point where computer learning takes place. The series extends through $\mathcal{F}$ finally reaching an optimum region $F^0$ in $\mathcal{F}$. $F^0$ is a region wherein lies feature vectors from model responses which appear identical to the man. This series of $F^k$ illustrate the iterative progress of the algorithm toward an optimum response.

A brief remark is made here on implementation of the algorithm. The analog portion of the hybrid computer performs the function of simulating the model equations and acts as an output device to the display and the recorder. The digital portion of the hybrid computer performs mainly the tasks of controlling the analog computer and computing various quantities required for parameter
variation and training. The display is a storage oscilloscope. The only limitation to the number of responses which may be drawn on the display is the physical dimension of the display and the response curves.

\[ f_2 \]

\[ \text{Feature space } \mathcal{F}_i \]

\[ F^0 \]

\[ \text{Series of learned regions} \]

\[ \text{Starting point} \]

\[ X \]

\[ \text{Series of learned regions in } \mathcal{F}_i \text{ progressing toward } F^0. \]

**Example Problem**

**Statement of the Problem**

The given set of equations with a fixed form is a model of an oceanographic system which was derived from physical considerations [24]. (This is the same problem presented at the beginning of Chapter 1.)

\[
\begin{align*}
\dot{r}_1 &= a_{11} r_1 + a_{12} r_1 r_2 + a_{13} r_1 r_2 - m(t) r_1 \\
\dot{r}_2 &= a_{21} r_1 + a_{22} r_1 r_2 + a_{23} m(t) - m(t) r_2 \\
\dot{r}_3 &= a_{31} r_1 + a_{32} r_3 + a_{33} r_3^2
\end{align*}
\]  

(1)
where \( r_i \) are state variables, \( i = 1, 2, 3 \)

\[
\begin{align*}
    r_1 & = \text{plants (phytoplankton)} \\
    r_2 & = \text{nutrients (phosphates)} \\
    r_3 & = \text{animals (herbivores)}
\end{align*}
\]

\( m(t) \) - uncontrollable input (a water mixing coefficient)

A model response \( r \) is composed of the individual state variable time responses \( r_1, r_2, \) and \( r_3 \). The coefficients \( a_{ij} \) become the parameter vector \( \pi = (a_{11}, a_{12}, \ldots, a_{33}) \). This \( \pi \) has nine components. Scanty and erroneous data is available from the oceanographic system. Figure 1.1 is a representation of this data [24].

In (1) the parameter vector \( \pi \) is initially unknown. The problem is to find the value of \( \pi \) which produces a model response \( r \) that best fits this data. This parameter search is carried out by the algorithm presented here. Details of the algorithm for (1) are given in the Appendix.

A value of the parameter vector \( \pi \) used in [24] is taken as the starting point in the algorithm. Initially, the model response corresponding to the starting parameter vector fits the data poorly. An example of a solution of the model response for the starting point is response \( r_1^1 \) in Figure 2.10.

Responses \( r_2^2 \) and \( r_3^3 \) in Figure 2.10 are model responses obtained to fit the data. This was done by using the algorithm.
The Original Solution

Reference [24] is the original work done on fitting the model response $r$ to the data. However, in [24] the parameter $\pi$ is arbitrarily made a function of one of the state variables of the system. This is done, apparently unjustifiably, to obtain a good match between the physical data and the model response $r$. In this thesis the parameter $\pi$ is assumed to remain constant with the state variables $r_i$ and with time.

Results Pertaining to the Algorithm

Recall that the object of the algorithm is to perform an efficient search for the parameter vector $\pi$, notwithstanding the nine parameters and the scanty and corrupted data from the oceanographic system. Furthermore, this algorithm uses man-machine interaction to solve the poor data problem and computer learning to render more efficient the parameter search.

It is possible to conjecture on several ways to measure this efficiency. One measure of efficiency is the ratio of the number of times the man pushes the "better" button to the sum of the number of times the man pushes the "same" or the "worse" button. A phrase is coined for this ratio, the "push button ratio." This ratio is reasonable because finding responses which are better means
progress toward the optimum response. When the push button ratio in the first part of the learning phase is compared with the push button ratio in the second part of the learning phase then a statement can be made about the relative rates of progress toward the optimum response.

Comments can be made on another type of rate of progress. This is the amount the responses step toward the optimum on each iteration as observed by the man on the display. For example the better responses in one case could be "creeping" toward an optimum and in another case the better responses could be "jumping" toward an optimum. This sort of rate of progress toward an optimum is a subjective observation in man-machine interaction.

Another factor that must be considered when evaluating the learning technique is the possibility that the computer finds responses in the learned phase in a manner by which it was taught but these responses found by the computer are not actually improvements of the model response. For example, the man could be observing carefully the slope of a response which is taken as a feature of the response. The man observes a series of preferential responses in which the slope increases towards a desired slope. Consider the case where the algorithm enters the learned phase with the slope of the curve already at its optimum. The learned phase finds a learned response with a different slope as it was taught. However the learned response
is not better than the last best response because the slope is now no longer optimum. A phenomenon such as this one is subjective and hence cannot be evaluated quantitatively.

The algorithm was tested by three graduate students\textsuperscript{11} who used the algorithm to conduct the parameter search of example problem (1). Everyone easily adapted to the man-machine interaction technique. Since of course the "optimum" response is a subjective entity each man terminated the parameter search with a slightly different model response fit to the data. In all three cases the push button ratio is higher in the learned phase of operation indicating a more efficient parameter search. The figures are shown below.

<table>
<thead>
<tr>
<th>man</th>
<th>C. Cross</th>
<th>R. Iverson</th>
<th>R. Frank</th>
</tr>
</thead>
<tbody>
<tr>
<td>first part of learning phase</td>
<td>0.20</td>
<td>0.29</td>
<td>0.102</td>
</tr>
<tr>
<td>second part of learning phase</td>
<td>0.45 or 0.78</td>
<td>0.33 or 1.0</td>
<td>0.2</td>
</tr>
</tbody>
</table>

The model responses corresponding to these push button ratios are discussed in the next section.

In the case of Cross there are two quantities for the push button ratio in the second part of the learning phase. When he was conducting the parameter search the circumstances were such that the first

\textsuperscript{11} Acknowledgement is extended to C. Cross, R. Frank, and R. Iverson.
time the algorithm entered the learned phase the parameter step size was too small. On the second and third times through the learned phase the parameter step size was increased. This results in the two push button ratios. The smaller number (0.45) is calculated using the man's judgements for all three entries into the learned phase. The larger number (0.78) is calculated using the man's judgements for only the second and third entries into the learned phase. The first number is smaller because some of the learned responses found the first time through the learned phase were so similar to the best response $r_b$ that they could have been called either better than or the same as $r_b$. Nevertheless, both push button ratios in the second part of the learning phase are larger than the push button ratio in the first part of the learning phase indicating a more efficient parameter search.

In the case of Iverson there are two push button ratios in the second part of the learning phase. Iverson went through the learned phase three times in succession. By the time the algorithm had progressed through the learned phase the third time the best response $r_b$ was rather close to the optimum. Hence, there are two push button ratios given. The higher number (1.0) is the push button ratio calculated from the judgements made after the first and second times through the learned phase. Similarly, the lower number (0.33) is calculated based on all the judgements made (after the first, second
and third times through the learned phase). The interesting point is that in the third time through the learned phase the computer found learned responses \( r^A \) which were tending in the direction taught to the computer. However, since \( r^b \) was near optimum the \( r^A \)'s stepped past the optimum response. Thus, they were rejected by the man.

The push button ratios both in the first and second parts of the learning phase are smaller in the case of Frank than in the other two cases. In this case the test parameter step sizes were allowed to become large resulting in oscillations around the feature space and a slower progression toward an optimum response. Nevertheless, the push button ratio in the second part of the learning phase is larger than the pushbutton ratio in the first part of the learning phase again indicating a more efficient parameter search.

One of the problems encountered in example problem (1) is that the starting point in feature space \( \mathcal{F} \) was too close to the optimum region \( F^O \) (Figure 2.9). In the test cases run, typically the man ran through part one of the learning phase once and then cycled through the learned phase and part two of the learning phase one, two, or three times. Hence in terms of Figure 2.9 the algorithm ran through only the last one to three learned regions \( F^L \). After this the learned phase was not useful because the parameter step size was too large resulting in the features stepping past the optimum
Some comments are made on procedures for using the algorithm. It is beneficial for computer training for the man to know the features which are being extracted from the model responses by the algorithm. The man should know how the parameters are changed in the learning phase as an aid in deciding which cueing button to push. If the man is familiar with the physics of the system he should be aware of the current values of the parameters. The man should avoid large parameter changes since this could lead to overloading difficulties in the learned phase. This can be done by the man observing the current parameter values.

Results Pertaining to the Parameters

Several parameter searches were conducted for example problem (1). The terminating point of the parameter searches resulted in model responses, called optimum responses, which are improved responses when compared to the model response at the start of the parameter search. The term "optimum" response is used here to mean a response that fits the data "reasonably" well.

For each optimum response $r^i$ there corresponds a parameter vector $\pi^i \in \mathcal{P}$ where $i$ is an integer. These $\pi^i$ are recorded as well as the $r^i$. (The superscript denotes different vectors in the vector space.)
Starting point responses $r^1$ and $r^6$, and optimum responses
$r^2$, $r^3$, $r^4$, $r^5$, $r^7$ are shown in Figure 2.10 to 2.12. The parameter vectors $\pi_i$, $i = 1, 2, 3, 4, 5, 7$ corresponding to the responses $r^i$ are listed in Table 2.1. The parameter vector $\pi^1$ (corresponding to starting response $r^1$) is obtained from reference [24].

Response $r^1$ of Figure 2.10 is the model response for a starting point $\pi^1$ of the parameter search. Both $r^2$ and $r^3$ are optimum responses found for two different parameter searches starting at $\pi^1$. Responses $r^2$ and $r^3$ are both different in shape. $r^3$ is probably a better fit than $r^2$. The difference can partly be explained by the fact that the water mixing coefficient $m(t)$ is 0.025 for $r^2$ and 0.01 for $r^3$. (In the physical system $m(t)$ is not a constant.) However, $r^2$ and $r^3$ are different optimum responses mainly because the operator terminated the search process at a different point in $\Phi$ based on the judgement that $r^2$ and $r^3$ are respectively reasonable fits to the data.

The parameter searches indicated by Figures 2.10 and 2.11 both began with starting response $r^1$. In Figure 2.11 the optimum response $r^4$ was found by operator R. Frank and the optimum response $r^5$ was found by operator R. Iverson. A good fit of the state variable $r_3$ (animals) is less important than the fit of the other two state variables because the error in the data points of $r_3$ are especially bad. Hence for the response $r^4$ of Figure 2.11 the operator
Figure 2.10. Starting response $r_1$, optimum responses $r_2$ and $r_3$. 
Figure 2.11. Starting response $r_1$, optimum responses $r^4$ and $r^5$. 
Figure 2.12. Starting response $r^6$, optimum response $r^7$. C. Cross.
Table 2.1. Parameter values for various responses of problem (1).

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Fig. 2.10</th>
<th>Fig. 2.11</th>
<th>Fig. 2.12</th>
<th>IP method</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>π1</td>
<td>π2</td>
<td>π3</td>
<td>π4</td>
</tr>
<tr>
<td>$a_{11}$</td>
<td>-.11</td>
<td>-.252</td>
<td>-.203</td>
<td>-.252</td>
</tr>
<tr>
<td>$a_{12}$</td>
<td>.024</td>
<td>.0593</td>
<td>.002</td>
<td>.0</td>
</tr>
<tr>
<td>$a_{13}$</td>
<td>.75</td>
<td>1.051</td>
<td>.688</td>
<td>.676</td>
</tr>
<tr>
<td>$a_{21}$</td>
<td>.027</td>
<td>.0</td>
<td>.016</td>
<td>.026</td>
</tr>
<tr>
<td>$a_{22}$</td>
<td>-.58</td>
<td>-.81</td>
<td>-.58</td>
<td>-.340</td>
</tr>
<tr>
<td>$a_{23}$</td>
<td>-.7</td>
<td>-.308</td>
<td>-.84</td>
<td>-.280</td>
</tr>
<tr>
<td>$a_{31}$</td>
<td>4.0</td>
<td>1.572</td>
<td>1.625</td>
<td>2.40</td>
</tr>
<tr>
<td>$a_{32}$</td>
<td>-.026</td>
<td>-.041</td>
<td>-.041</td>
<td>.0</td>
</tr>
<tr>
<td>$a_{33}$</td>
<td>-.01</td>
<td>-.011</td>
<td>-.011</td>
<td>-.010</td>
</tr>
</tbody>
</table>
tended to emphasize the fit of \( r_1 \) and \( r_2 \) rather than \( r_3 \).

The operator R. Iverson is an oceanography graduate student. He used his knowledge of the physical system as an aid in obtaining a reasonable fit of the response \( r^5 \) to the data. Both \( r^4 \) and \( r^5 \) in Figure 2.11 are improvements over the starting response \( r^1 \).

Figure 2.12 has a starting response \( r^6 \) that is different than starting response \( r^1 \) of Figures 2.10 and 2.11. The operator terminated the parameter search with optimum response \( r^7 \).

Example problem (1) was solved elsewhere\(^{12}\) by a parameter search using an IP rather than man-machine interaction. Also in this solution the water mixing coefficient \( m(t) \) was not held constant with time. The parameter values found by these investigators are listed as \( \pi^8 \) in Table 2.1. It is interesting to note that for the optimum responses presented here that there is a variance in parameter values. This is to be expected for two reasons 1) the optimum responses were found by different operators and 2) the sensitivity is different for different parameters. The more sensitive parameters are the ones with the smaller variances. This indicates that there exists not one parameter vector for one optimum response but a region in parameter space containing equivalent optimum parameter vectors.

III EVALUATING FEATURE EXTRACTION TECHNIQUES

Introduction

A major problem in pattern recognition is the extraction of features which characterize a given situation. For example, in a power distribution system a set of voltage monitoring devices represents a pattern from which features can be extracted to characterize the power system situation. Similarly, in a character recognition system a set of measurements of a character also represents a pattern from which useful features can be extracted. In general the information contained in a set of feature measurements decreases as the feature extraction process becomes less accurate. Consequently, entropy is a natural candidate for evaluating feature extraction methods. In this Chapter the use of entropy to compare different methods of feature extraction is described. In particular, we consider the problem of extracting features which describe a physical system response. The notion of feature extraction in characterizing a system's response is borrowed from pattern recognition theory [6, 14, 19]. References [26] and [27] on feature extraction inspired the use of entropy in this study.

The main result of this Chapter is the application of the concept of entropy to evaluate feature extraction techniques. Necessary
and sufficient conditions for a decrease in entropy due to quantization and the reduction of feature dimensionality in the feature extraction process are derived. Three feature extraction techniques for a 2nd order system response are compared. The results of this Chapter provided insight into the selection of the features which were used in the example problem (1).

**Description of the Problem**

Consider a system modeled by a set of equations. The response of this mathematical model is a function of model parameters and fixed initial conditions. It is desired to extract and store in a finite size memory features of the system responses for several different sets of parameter values. The problem is to evaluate different feature extraction methods so that a maximum of information about the system response can be retained within the memory capability. It is shown that this problem is equivalent to finding the feature extraction technique that maximizes the entropy of the extracted features.

The feature extraction process is shown in Figure 3.1. System responses are a result of a transformation $T_{pr}$ whose domain is the set of parameter vectors in the parameter space $\mathcal{P}$ and whose range is in the pattern space $\mathcal{R}$. A feature $f$ is a real number obtained by processing a vector in pattern space according to a specified operation or transformation $T_{rf}$. The domain of $T_{rf}$ is in the pattern space $\mathcal{R}$.
and its range is in feature space \( \mathcal{F} \).

**Figure 3.1.** Feature extraction process.

Generally the feature extraction process results in a loss of pattern information. Hence, there is the possibility of the equivocation of patterns, e.g. the transformation \( T_{rf} \) could map two different pattern vectors into the same feature vector. The limitations on memory used to store feature vectors increases the possibility of pattern vector equivocation. For a given transformation \( T_{pr} \) and domain of \( T_{pr} \), we seek the transformation \( T_{rf} \) that minimizes pattern vector equivocation subject to feature vector storage limitation.

From pattern recognition theory the transformation \( T_{rf} \) is the process of feature extraction from a pattern. Two dependent and opposing considerations that affect the selection of a transformation \( T_{rf} \) are the "distance" between two feature space vectors that correspond to two parameter space vectors and the dimensionality of the feature space [19].

For simplicity assume there is no loss of entropy in \( T_{pr} \) and any loss of entropy occurs subsequent to \( T_{pr} \).

The entropy in \( H_\Pi \) in parameter space \( \mathcal{P} \) and the entropy \( H_F \).
in feature space $\mathcal{F}$ is defined as follows [23]:

$$H_{\Pi} = -E_{\Pi} \{ \log p_{\Pi}(\pi) \} = -\Sigma_{\Pi} p_{\Pi}(\pi) \log p_{\Pi}(\pi)$$

$$H_{F} = -E_{F} \{ \log p_{F}(f) \} = -\Sigma_{F} p_{F}(f) \log p_{F}(f)$$

where $\pi$, an $n_{\Pi}$-vector, is a parameter vector in $\mathcal{P}$; $f$, an $n_{F}$-vector, is a feature vector in $\mathcal{F}$; and $p_{\Pi}(\pi)$, $p_{F}(f)$ are discrete probability densities. In this paper probability densities conform to the notation $p_{X}(x) = \text{Prob}(X=x/\text{sample space of } X)$ where $X$ is a random vector and $x$ is a specific value of $X$ [20].

Note that in some real problems $p_{\Pi}(\pi, i)$ can be found. For example, this is done by measuring attributes of the physical system and from these quantities establishing $p_{\Pi}(\pi, i)$ by histograms.

Let $W$ be a class of feature extraction processes $T_{rf}$ with $w$ a member of $W$. Let the physical capacity to store feature vectors be called the storage space, e.g. a computer memory. The storage space for feature vectors is limited. For a given $T_{pr}$, a domain of $T_{pr}$, a class $W$ and a finite storage space we desire to find $w$ in order to $\max_{w \in W} H_{F}$.

For the purpose of stating and proving a simple result let

$$\pi_{i} \in \mathcal{F}, \quad i = 1, 2$$

$$\pi_{1} \neq \pi_{2}$$

$$T_{rf}(T_{pr}(\pi_{i})) = f_{i} \in \mathcal{F}, \quad i = 1, 2$$

$$P_{\Pi}(\pi_{1}) + P_{\Pi}(\pi_{2}) = 1$$

$$P_{\Pi}(\pi_{1}) \geq 0, \quad i = 1, 2$$
Theorem 1: \( H_F < H_{\Pi} \) if and only if \( f^1 = f^2 \).

Proof: (1) If \( f^1 = f^2 = f^* \) then \( p_F(f^*) = 1 \). Following from this we have \( H_F = 0 \). But \( H_{\Pi} > 0 \). Therefore \( H_F < H_{\Pi} \).

(2) If \( f^1 \neq f^2 \) then \( p_F(f^i) = p_{\Pi}(\pi^i), \ i = 1, 2 \). This implies \( H_F = H_{\Pi} \). QED

This theorem is easily extended to the more general case. Let \( \pi^i \in \mathcal{P}, \ i = 1, 2, \ldots, n_F; \ \pi^i \neq \pi^j \) for \( i \neq j \); \( T_{pr}(T_{pr}(\pi^i)) = \pi^j \in \mathcal{F}, j = 1, 2, \ldots, n_F; \ \sum_i p_{\Pi}(\pi^i) = 1 \) and \( p_{\Pi}(\pi^i) > 0 \); then \( H_F < H_{\Pi} \) if and only if for at least one pair of vectors \( f_1^i, f_2^j, i, j = 1, 2, \ldots, n_F \), the condition \( f_1^i = f_2^j \) is satisfied.

Example: Let \( \mathcal{P} \) be of 2-dimensions and \( \mathcal{F} \) be of 1-dimension. Define graphically in Figure 3.2 two transformations \( T_1 \) and \( T_2 \):

\[ T_1: \mathcal{P} \rightarrow \mathcal{F}_1, \quad T_2: \mathcal{P} \rightarrow \mathcal{F}_2. \]

The transformation \( T_2 \) provides a loss in entropy. We would choose \( T_1 \) in preference to \( T_2 \) because \( T_1 \) provides no loss in entropy and the dimensions of \( F_1 \) and \( F_2 \) are equal.

![Figure 3.2. Definition of two transformations \( T_1 \) and \( T_2 \).](image-url)
In the study of a feature extraction transformation $T_{rf}$ it is desirable to know the effect of reducing the dimensionality of the feature space $F$ because of the finite storage requirement. Without loss of generality $F$ can be separated into the direct sum [29] of two subspaces, $F_1$ and $F'$, of $F$, such that $F = F_1 \oplus F'$ where $F_1$ is a 1-dimensional subspace of $F$ along the first dimension of $F$, i.e. $f_1 \in F_1$, and $(f_2, f_3, \ldots, f_{n_F}) = (f'_1, f'_2, \ldots, f'_{n_F-1}) \in F'$ where $f = (f_1, f_2, \ldots, f_{n_F}) \in F$. $F'$ is a space of dimension $n_F - 1$ called the reduced feature space.

Define the projection transformation $E$ so that for any $f \in F$, $f_1 \in F_1$ and $f' \in F'$ we have $E(f) = f'$. A measure of quality of $E$ is the comparison of the entropy $H_F$ of $F$ with the entropy $H_{F'}$ of $F'$. A simple theorem can now be stated giving conditions for the loss of entropy by projection.

**Theorem 2:** Assume the vectors $f^i \in F$, $i = 1, 2, \ldots, n$, are distinct and the vectors $f^i$ are projected on $F'$ along $F_1$. $H_{F'} < H_F$ if and only if at least one pair of vectors $f^i, f^j, i \neq j$, satisfy the conditions that $f^i_1 \neq f^j_1$ and $f^i_q = f^j_q$ for every $q = 2, 3, \ldots, n_F$.

**Proof:** (1) Let $H_{F'} < H_F$. Then we know from the uniqueness of all $f^i \in F$ and from the extension of Theorem 1, that at least one
pair of \( f^i_1 = f^j_1 \). These facts imply that \( f^i_1 \neq f^j_1 \) by definition of \( E \).

Thus the necessity is demonstrated.

(2) Let at least one pair of vectors \( f^i, f^j \), \( i \neq j \), satisfy the conditions that \( f^i_1 \neq f^j_1 \) and \( f^i_q = f^j_q \), \( q = 2, 3, \ldots, n_F \). For every vector \( f'^{k} \in \mathcal{F} \), \( k = 1, 2, \ldots, m \), that does not satisfy the given condition it follows from the definitions of \( E \) and \( p_F(f) \) that

\[
p_F(f^k) = p_F(f'^{k}).
\]

However for a pair of vectors \( f^i, f^j \) that satisfy the given conditions we have \( f^* = E(f^i) = E(f^j) \) and

\[
p_{F^n}(f^*) = p_F(f^i) + p_F(f^j).
\]

The definition of entropy provides

\[
H_F = - \sum_{i=1}^{m} p_F(f^i) \log p_F(f^i)
\]

(4)

\[
H_{F^n} = - \sum p_{F^n}(f^i) \log p_{F^n}(f^i)
\]

(5)

The terms in (4) and (5) corresponding to the vectors \( f'^k \) are identical. However for the vectors \( f^i, f^j \) that \( E \) projects into \( f^* \) we have two terms in (4), namely,

\[
p_F(f^i) \log p_F(f^i) + p_F(f^j) \log p_F(f^j),
\]

(6)

and one term in (5), namely,

\[
p_{F^n}(f^*) \log p_{F^n}(f^*).
\]

(7)
The two terms in (6) and the one term in (7) correspond to each other in (4) and (5), respectively. Then from (3) - (7) it follows that

$H_f < H_f$ because (7) is greater than (6) by virtue of (3). QED

This theorem provides a means of evaluating the effect of reducing the dimensionality of feature space $\mathcal{F}$.

Example: in feature space $\mathcal{F}$ assume

\[
\begin{align*}
  f^1 &= (1, 2, 3) \\
  f^2 &= (2, 2, 3) \\
  f^3 &= (3, 2, 5)
\end{align*}
\]

Then $H_f = \log 3$.

Let $E(f) = (f^2, f^3) = (f^1, f^2)$.

Then $f'^1 = E(f^2) = (2, 3)$

\[
\begin{align*}
  f'^2 &= E(f^3) = (2, 5) \\
  p_{F'}(f'^1) &= 2/3 \\
  p_{F'}(f'^2) &= 1/3
\end{align*}
\]

The entropy in $\mathcal{F}'$ space is

\[
H_{F'} = \log 3 - 2/3 \log 2 < H_F.
\]

Recall that the problem, as stated previously, requires one
to find the maximum of $H_F$ over the given class of transformations $W$ from $\mathcal{R}$ to $\mathcal{F}$. An extended problem statement is now made which incorporates the objective of reducing the dimensionality of the feature space $\mathcal{F}$. For this purpose $\mathcal{F}$ is separated into a direct sum $\mathcal{F} = \mathcal{M}_1 \oplus \mathcal{M}_2$. A transformation $E_c$ is defined such that for any $f \in \mathcal{F}$, $f^1 \in \mathcal{M}_1$ and $f^2 \in \mathcal{M}_2$ we have $E_c(f) = f^1$ which is the projection of $f$ on $\mathcal{M}_1$ along $\mathcal{M}_2$. Let a nonzero positive integer $c$ be called the projection index. More specifically if $\mathcal{M}_1$ is a non-fixed subspace of $\mathcal{F}$, $\dim (\mathcal{M}) = n$, $\dim (\mathcal{M}_1) = k$ and the basis vectors of $\mathcal{M}_1$ are chosen to coincide with any $k$ basis vectors of $\mathcal{F}$, then the number of possible transformations which can be employed to take $\mathcal{F}$ into $\mathcal{M}_1$ is $\binom{n}{k}$. Consequently $c$ can take on values from 1 to $\binom{n}{k}$ where each value of $c$ identifies one of the possible subspaces.

The extended problem statement follows. For a class $W$ of transformations $T_{rf}: \mathcal{R} \rightarrow \mathcal{F}$, a given $T_{pr}$ and a domain of $T_{pr}$, and for a projection transformation $E_c: \mathcal{F} \rightarrow \mathcal{F}'$ whose domain is the range of $T_{rf}$ find $c$ and $w$ in order to max $\max_c \max_{w \in W} H_F(T_{rf}, E_c)$.

Figure 3.3 shows the transformations in the extended problem.

Figure 3.3. Transformations in the problem.
Quantization of the Response Space

In most realistic situations the recording of a response of a physical system contains some form of quantization. Furthermore for practical considerations in storing the feature vectors, it is necessary to intentionally quantize the feature vectors. In this section the effect of the familiar concept of quantization in this feature extraction problem is investigated.

To include quantization, a practical system can be separated into two parts: 1) an ideal system model with exact measurable attributes followed in series by 2) a transformation that quantizes measured variables [13]. This technique is employed as follows. A quantization transformation $Q_1$ from $\mathbb{R}$ to another space called quantized response space $\mathcal{Q}$ is defined in the following way: Let $h_i$ be a scalar constant called the quantization level in the $i$th dimension, $i = 1, 2, \ldots, n_q$. For vectors $r = (r_1, r_2, \ldots, r_{n_r}) \in \mathcal{R}$ and $k = (k_1, k_2, \ldots, k_{n_q}) \in \mathbb{Z}$, whenever $u(r_i)h_i - h_i/2 < r_i < u(r_i)h_i + h_i/2$ then $k_i = u(r_i)h_i$ where $u(r_i)$ is an integer. Note that $\dim(\mathcal{Q}) = \dim(\mathbb{R})$, hence $n_{\mathcal{Q}} = n_q$.

Theorem 1 is applied to the transformation $Q_1$ as follows.

Assume the entropy $H_R$ of $\mathcal{R}$ is known. Let $r_1, r_2 \in \mathcal{R}$ on which $p_R(r)$ is defined such that $p_R(r_1) + p_R(r_2) = 1$ and $p_R(r_i) \neq 0$, $i = 1, 2$. 


\( Q_1 \) defines a probability density \( p_K(k) \) in \( \mathcal{Q} \). Also \( Q_1 \) defines
\( k^1 = Q_1(r^1) \), \( k^2 = Q_1(r^2) \). Then Theorem 1 yields

**Assertion 1:** \( H_Q < H_R \) if and only if \( k^1 = k^2 \).

As Theorem 1 can be extended to a larger number of vectors
Assertion 1 can likewise be extended: There is a decrease in entropy by quantization if and only if at least two vectors in the range of \( Q_1 \) in \( \mathcal{Q} \) are equal.

An assertion can be made about the quantization level \( h_i \) for a transformation \( T_{pr} \) defined on a domain in \( \mathcal{P} \) consisting of a finite number of vectors \( r^i, i = 1, 2, \ldots, m_p \). It has been assumed that there is no loss of entropy in the transformation \( T_{pr} \), hence

\[ H_P = H_R. \]

**Assertion 2:** There exists a scalar \( \epsilon \) such that when \( h_i \leq \epsilon \),

\[ i = 1, 2, \ldots, n_q, H_Q = H_P. \]

Proof: The validity of Assertion 2 is demonstrated graphically in Figure 3.4. By hypothesis the vectors \( r^i, i = 1, 2, \ldots, m_r \), are discrete and unequal. Hence an \( \epsilon \) exists for which a grid can be constructed that is fine enough to enclose each \( r^i \) in its unique quantization cell. This is equivalent to saying that for transformation \( Q_1 \) each vector \( k^i = Q_1(r^i), i = 1, 2, \ldots, m_q \), is unique. Applying the extension to Theorem 1 we can say then that \( H_Q = H_R \). But \( H_R = H_P \) by hypothesis therefore \( H_Q = H_P \). QED
Example: Consider the responses \( r(\pi, t) = \pi t \) where 
\[ \pi \in (0, 1/19, 2/19, \ldots, 18/19, 1) \] and \( t = (0, 1/2, 1) \) for each response. Hence, the patterns become 
\[ r_1 = (0, 0, 0), \]
\[ r_2 = (0, 1/38, 1/19), \ldots, r_{20} = (0, 1/2, 1). \] Assume \( p(\pi) = 1/20. \) This implies \( H_P = \log 20. \) Let \( h_i = 1/m, \quad m = 1, 2, \ldots, 20. \) Data obtained by computer for a plot of \( H_Q \) vs. \( m \) is displayed in Figure 3.5. The plot shows that when \( m \geq 18 \) or equivalently \( \epsilon \leq 1/18, \) the equality \( H_Q = H_P \) holds.
Quantization of the Feature Space

Because of storage space limitations it is necessary in a practical problem to intentionally quantize the feature vectors extracted from pattern vectors. In this section assertions similar to those in the previous section are made.

A quantization transformation $Q_2$ from $\mathcal{F}$ to a space called quantized feature space $\mathcal{F}'$ is defined in a manner identical to the definition of $Q_1$. Similar to Assertion 1 we state
Assertion 3: $H_{F'} < H_F$ if and only if $f_1^1 = f_2^2$, where $H_F$ is the entropy of $\mathcal{F}$, $H_{F'}$ is the entropy of $\mathcal{F}'$, and $f_1^1$ and $f_2^2$ are defined in a way identical to the definition of $k_1^1$ and $k_2^2$ (page 72). The proof of Assertion 3 is identical to that of Assertion 1 except for notation.

Analogous to Assertion 2 the following assertion is made.

Assertion 4: There exists a scalar $\epsilon$ such that when $h_i < \epsilon$, $i = 1, 2, \ldots, n_q$, $H_{F'} = H_F$, where $\dim(\mathcal{F}') = \dim(\mathcal{F}) = n_q$ and $h_i$ is the quantization level in $Q_2$. The proof of Assertion 4 follows Assertion 2 except for notation.

A Means of Feature Space Reduction

In order to conserve the storage space available to store the given feature space vectors it is desirable to investigate the feasibility of reducing the dimensionality of the feature space. For a quantized feature space $\mathcal{F}_{n_c}^n$, $\dim(\mathcal{F}_{n_c}^n) = n_c$, and a subspace $\mathcal{F}_{n_c}^k$ of $\mathcal{F}_{n_c}^n$, $\dim(\mathcal{F}_{n_c}^k) = k$, $k < n_c$, there exists $\binom{n_c}{k}$ different ways to project $\mathcal{F}_{n_c}^n$ along $\mathcal{F}_{n_c}^k$. A means of reduction of feature space dimensionality from $n_c$ to $m$ proceeds as follows (Figure 3.6). A series of projection transformations $E_{n_c}^1, E_{n_c-1}^1, \ldots, E_{m+2}, E_{m+1}'$, in which each transformation reduces the dimension by one, is performed sequentially upon a series of feature spaces $\mathcal{F}_{n_c}^n, \mathcal{F}_{n_c-1}^n, \ldots, \mathcal{F}_{m+2}^n, \mathcal{F}_{m+1}^n$ where $Q_2: \mathcal{F} \rightarrow \mathcal{F}_{n_c}^n$, i.e. $E_i: \mathcal{F}^i \rightarrow \mathcal{F}^i_{k-1}$. 
In words this series of projection transformations yields a series of feature spaces where each space is one dimension smaller than the preceding space. The entropy $H_{F_i}$ of each $F_i$ is calculated. The series $H_{F_i}$, $i = n, \ldots, m$, provides a numerical means to determine a reasonable dimensional size for a feature space.

Figure 3.6. Transformations in feature space reduction.
At each projection transformation $E_i$, the 1-dimensional subspace $\mathcal{F}_i^*$ of $\mathcal{F}_i^i$, which is being "eliminated" is chosen from the $\binom{1}{1}$ possible 1-dimensional subspace choices. This choice should be made so that a minimum of entropy is lost in the projection transformation $E_i$.

The heuristic approach taken in the following application to determine $\mathcal{F}_i^*$ is to assign $\mathcal{F}_i^*$ to the 1-dimensional subspace of $\mathcal{F}_i^i$, whose vector components have the least "spread" out of the $\binom{1}{1}$ possible 1-dimensional subspace choices. For a measure of "spread" in the $j$th dimension of $\mathcal{F}_n^c$, we use the variance of the $j$th component of $f \epsilon \mathcal{F}_n^c$, $\text{var}(f_j)$, where for the set of vectors $\pi^i \epsilon \mathcal{P}$, $i = 1, 2, \ldots, n_p$, transformations yield (Figure 3.6)

$$f_i^i = Q_2(T_k(Q_1(T_{pr}(\pi^i))))$$

$k$ is an integer,

$$f_i' = (f_1^i, f_2^i, \ldots, f_n^i)$$ and

$$\text{var}(f_j') = \sum_{i=1}^{n_p} \left( (f_j'^i - \sum_{i=1}^{n_p} f_j'^i p_{F_i^i}(f_j'^i))^2 p_{F_i^i}(f_j'^i) \right).$$

In the application in the next section $n_c$ is taken as $\text{dim}(\mathcal{F}_n^c)$ and $n_c = 10$ or 9. The variances $\text{var}(f_j')$, $j = 1, 2, \ldots, n_c$, are calculated and arranged in the order of increasing magnitude. Let the variances in their increasing order be denoted $(v_1^c, \ldots, v_3^c, v_2^c)$. Then $\mathcal{F}_1^*$ at projection transformation $E_i$, $i = n_c, \ldots, 3, 2$, is
the 1-dimensional subspace of $\mathfrak{H}$ corresponding to the variance $v_i$.

**Application to a Second Order Linear System**

The transformations of feature extraction, quantization, and dimensional reduction of feature space are represented in Figure 3.6. The problem is to evaluate three different feature extraction transformations.

$$\pi^i \in \mathcal{O}, i = 1, 2, \ldots, 100$$

$$\pi^i = (\xi^j, \omega^k)$$

$$\xi^j = .6, .6222, \ldots, .8, j = 1, 2, \ldots, 10$$

$$\omega^k = .5, .5555, \ldots, 1.0, k = 1, 2, \ldots, 10$$

$$\pi^u \neq \pi^v \text{ for } u \neq v$$

$$T_{pr}(\pi^i) = r((\xi, \omega), t) = e^{-\xi \omega t} \cos \omega t$$

$$Q_1 = \text{transformation that samples } r((\xi, \omega), t) \text{ at } t = 0, 10/9, 20/9, \ldots, 10 \text{ and quantizes the samples into levels of } 0.02.$$

$$T_1 = \text{feature extraction transformation, } i = 1, 2, 3$$

$$T_1 = 1, \text{ unity transformation, } i.e. \text{ the samples are the features}$$

$$T_2 = \text{transformation that calculates 9 Fourier coefficients of a cosine series which are components of the quantized pattern vector}$$
\[ k \in \mathcal{Q} [4, 7, 12]. \]

\[ T_3 = \text{transformation that calculates 10 orthogonal polynomial coefficients of } k \in \mathcal{Q} [4, 7, 15]. \]

\[ \mathcal{Q}_2 = \text{transformation that quantizes } f \in \mathcal{F} \text{ into levels of } 0.02. \]

\[ \mathcal{E}_c = \text{projection transformation where} \]
\[ c = 1, 2, \ldots, n_c \text{ and } n_c = 9 \text{ for } T_2, \]
\[ n_c = 10 \text{ for } T_1 \text{ and } T_3, \text{ such that } \mathcal{E}_{i+1} \]
projects an \((i+1)\text{st}\) dimensional space \(\mathcal{F}_{i+1}\) into an \(i\text{th}\) dimensional space \(\mathcal{F}_i'.\)

Assume \(p(\pi^i) = 1/100, i = 1, 2, \ldots, 100. \) Note that \(T_{pr}\) represents the familiar 2nd order system with damping \(\zeta\) and natural frequency \(\omega.\)

As mentioned in the previous section the choice of the \(1\)-dimensional subspace \(\mathcal{F}_1^*\) of \(\mathcal{F}_1'\) along which \(\mathcal{E}_i\) projects is arbitrary and is determined by the variance of the corresponding vector components in \(\mathcal{F}_n^{n_c}.\)

The entropies \(H_{P}, H_{F_{n_c}}, \ldots, H_{F_2}, H_{F_1}\) are calculated by computer and plotted in Figure 3.7 for \(T_1, T_2, T_3.\) The difference \(H_{P} - H_{F_{n_c}}\) is the entropy lost because of the properties of \(\mathcal{Q}_1, T_1, \) and \(\mathcal{Q}_2\) and not because of dimensional reduction.

The results shown in Figure 3.7 reveals that in this case the
choice of either $T_1$ or $T_2$ and a 6-dimensional feature space would be best in meeting the objectives of maximizing entropy and minimizing storage space of the features, i.e. determining $\max_c \max_w H_{F'}$. 

Figure 3.7. Plot of entropy of reduced feature spaces $H_{F'}$ vs. number of features $m$. 

- Samples $T_1$
- Fourier $T_2$
- Polynomial $T_3$
It may be of interest to note the order in which the basis vectors of $\mathcal{F}_{n_c}$ were eliminated. For $T_1$, the 1st sample is taken at $t = 0$ and the 10th sample at $t = 10$. The order, beginning with the dimension eliminated 1st, is 1, 10, 9, 8, 6, 7, 5, 4, 2, 3. For $T_2$, the order of the Fourier coefficients, beginning with the dimension eliminated 1st, is 9, 2, 8, 7, 1, 6, 3, 5, 4. For $T_3$, the order of the polynomial coefficients, beginning with the dimension eliminated 1st, is 1, 2, 10, 3, 9, 4, 8, 5, 7, 6. Interestingly, for $T_1$, where each sample is a feature, the features eliminated last are those samples taken where the waveform has a steeper slope.
IV CONCLUSIONS

The algorithm presented here attacks the problem of conducting an efficient parameter search for parameter values of a given system model. The algorithm uses man-machine interaction and computer learning to solve the problem of poor physical system data and the problem of a large number of system model parameters.

The example problem presented (1) has been implemented using a hybrid computer and a storage oscilloscope display device. Experience with this example problem emphasized the obvious advantages of a computer aided parameter search. The main advantage a man has who is familiar with the physical system being modeled is the capability for him to use his intuitive knowledge of the system. The man observes on line the model responses and the parameter values for those model responses. From this observation he makes an evaluation of the model response in a way which is more closely related to the real problem than an algorithm that uses only an index of performance.

Experience with the algorithm used on example problem (1) also pointed out the shortcomings of the techniques employed in the algorithm. One shortcoming is the fact that a man-machine interaction algorithm written for the hybrid computer is a more difficult algorithm to design than would be a parameter search algorithm using
an index of performance written for a digital computer. Another shortage of a man-machine interaction algorithm is the necessity of a good display device. Display devices and the requisite software are not readily available in the ordinary computer laboratory. There is however a trend toward interactive display devices in computer facilities.

The system model equations were programmed on an analog computer for example problem (1). A shortage in this particular instance was the necessity to use potentiometers for the parameters. This required the man to wait for the potentiometers to be set for each different model response. A solution to this difficulty, albeit more expensive in terms of equipment, is the use of electronic multipliers in place of the parameter potentiometers.

In the example problem (1) the push button ratio showed that the parameter search was more efficient in the learned phase of the algorithm. That is, it is more efficient in the learned phase than in the learning phase in the sense that a higher proportion of the responses displayed to the man were judged "better."

The example problem showed that the push button ratios indicated a more efficient learned phase when the parameter search was not too close to its optimum. That is, for this particular algorithm the learned phase did not contribute to the efficiency of the parameter search when the parameters are close to optimum.
A possible cure for this problem is to vary the parameter step size or the size of the learned region as a function of "distance" in feature space of the current feature vector from the optimum feature vector.

In the example algorithm the shape of the learned region was a hypercone with its base on the last best feature vector. As an experiment to improve the algorithm one could orient the cone in the opposite direction, i.e., place the tip of the cone on the last best feature vector. This would allow the learned phase to take a "long" step in the preferential direction.

In the example problem one possible shortcoming is the length of time involved in training the computer the first time the algorithm passes through the learning phase, i.e., the amount of judgements necessary in order to store a set of preferentially ordered feature vectors. The algorithm is written so that computer learning takes place only when the man judges a model response "better." In order to improve the algorithm the time for training could be reduced by rewriting the algorithm to use the information available whenever the man makes a "worse" or "same" judgement. This information is now essentially wasted for training.


APPENDIX
APPENDIX

A detailed description of the logic flow diagrams of the main computer program and its subroutines are on file in the System Simulation Laboratory, Department of Electrical and Electronics Engineering.