HERISTIC LEARNING PARAMETER IDENTIFICATION FOR SURVEILLANCE AND DIAGNOSTICS OF NUCLEAR POWER PLANTS

A Dissertation

by

Eduardo Lavenère Machado

December 1983

Nuclear Engineering Department

THE UNIVERSITY OF TENNESSEE
Knoxville, Tennessee

HEURISTIC LEARNING PARAMETER IDENTIFICATION FOR SURVEILLANCE AND DIAGNOSTICS OF NUCLEAR POWER PLANTS

A Dissertation

by

Eduardo Lavenère Machado

December 1983

Nuclear Engineering Department

THE UNIVERSITY OF TENNESSEE

Knoxville, Tennessee

HEURISTIC LEARNING PARAMETER IDENTIFICATION FOR SURVEILLANCE AND DIAGNOSTICS OF NUCLEAR POWER PLANTS

Orientador
Rafael B. Perez
(informação
Eng. Mecânica)

A Dissertation

Presented for the

Doctor of Philosophy

Degree

The University of Tennessee, Knoxville

Eduardo Lavenère Machado

December 1983

To Marcia, Paulo and Barbara.

ACKNOWLEDGMENTS

The work was performed for the Instrumentation and Controls Division of the Oak Ridge National Laboratory which is operated by the Union Carbide Corporation for the United States Department of Energy. This research was sponsored by the Instrumentation and Control Branch, Division of Facility Operations of the Office of Nuclear Regulatory Research, U. S. Regulatory Commission under Interagency Agreement 40-551-75 with the U. S. Department of Energy under Contract W-7405-eng-26 with the Union Carbide Corporation.

The author wishes to express his gratitude to all who contributed their time and talent for the completion of this work, in particular to:

Rafael B. Perez, from UT and ORNL, for his advice, patience and encouragement, which made this work possible.

P. F Pasqua, head of the Nuclear Engineering Department at UT, and D. N. Fry, head of the Surveillance and Diagnostics Group at ORNL, for making the completion of this work possible through a research assistantship.

Willie T. King, Jose March-Leuba and James A. Mullens, from ORNL, for developing part of the computer programs utilized in this work.

Richard T. Wood, from UT and ORNL, for his patient work in reviewing and helping prepare the final draft.

M. E. Buchanan, from ORNL, for his help with the pressure loop experiment setup.

ABSTRACT

A new method of heuristic reinforcement learning has been developed for parameter identification purposes. In essence, this new parameter identification technique is based on the idea of breaking a multidimensional search for the minimum of a given functional into a set of unidirectional searches in parameter space. Each search situation is associated with one block in a memory organized into cells, where the information learned about the situations is stored (e.g. the optimal directions in parameter space). Whenever the search falls into an existing memory cell, the system chooses the learned direction. For new search situations, the system creates additional memory cells. This algorithm imitates the following cognitive process: (i) characterize a situation, (ii) select an "optimal" action, (iii) evaluate the consequences of the action, and (iv) memorize the results for future use. As a result, this algorithm is "trainable" in the sense that it can learn from previous experience within a specific class of parameter identification problems. From the mathematical point of view, the algorithm utilizes the inversion of a newly introduced concept of "fuzzy maps" between two spaces: the feature space and the parameter correction space. However, this operation is performed by the cognitive process described above rather than by mathematical manipulations. The main advantages of the new method are: (i) because of the minimum amount of computations, the

parameter identifications proceed faster than by the usual methods, and (ii) the parameter search can proceed automatically without the user-program interaction. The new algorithm was validated both analytically and via extensive computer simulations, utilizing a model of a U-tube steam generator of the type used in pressurized water reactors. A "real world" application was implemented whereby analog signals from an experimental pressure loop were utilized as inputs to the present learning algorithm. After a training period, during which the algorithm was shown the normal behavior of the loop, the system was able to diagnose failures in the loop and to accurately determine the parameters characterizing its normal behavior.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

FIGURE PAGE

CHAPTER 1

INTRODUCTION

1.1 Parameter identification and Surveillance and Diagnostic Methods

Since the inception of Nuclear Power Plant Technology and because of its "unforgiving" characteristics, a large effort has been devoted to the development of control, surveillance and diagnostic methods. Continuous surveillance of large dynamic systems such as Nuclear Power Plants can improve the system availability by early detection of incipient failure and by avoiding unnecessary periodic maintenance (1).

Reactor Noise methods (2-4) do not depend on the introduction of external stimulii and hence do not interfere with routine plant operations, making such methods ideal tools for surveillance purposes. The reactor noise method for surveillance and diagnostics is based on the utilization of a set of noise descriptors which characterize the signature or the state of a power plant that is in a given configuration and mode of operation. A noise descriptor is a functional of the stochastic fluctuations exhibited by state variables such as the Power Spectral Density (PSD) and the Cross Power Spectral Density (CPSD). Typically, a noise descriptor (the Fourier components

describing the time-evolution of the state variables) exhibits a considerable amount of structure as a function of the frequency. These features (e.g. peaks and valleys) are related to specific causative mechanisms such as fuel element vibrations, core barrel motion and thermal-hydraulic processes. In the normal operation mode, the location and strength of the features in the noise descriptors defines the plant signature. Plant surveillance methods are based on the detection of changes in the plant signature and the utilization of diagnostic algorithms to correlate the observed discrepancies with the altered plant conditions.

To implement the above surveillance and diagnostic methods, one must parameterize the noise descriptors in terms of the system parameters contained in a suitable model of the dynamic behavior of the plant. Ultimately, the plant signature is in fact associated to a set of parameters (heat transfer coefficients, Doppler coefficients of reactivity, fuel temperature, and so on). In principle, whenever an alteration of normal plant operation arises, the set of altered parameters will have to be identified . Essentially, it is then seen that surveillance methodologies do in fact reduce to parameter identification problems. Not very surprisingly, this problem is germane to the development of optimal control algorithms since, in both instances, the extrema of error functionals must be determined.

1.2 Survey of Control Optimization and Parameter Identification Methods

Historically, the development of control optimization and parameter identification techniques since the early days of nuclear technology can be divided into two periods. The first period covers the late 40's up to the early 60's - it was the era of control systems theory as a precise analytic discipline (5), with the goal of developing servo-mechanisms that would drive the system to follow a desired output trajectory. The fundamental method was the minimization of an error functional (i.e. a measure of the distance between the desired and the actual output), subject to various constraints. Very powerful analytical tools were either already available or were developed. Among such tools were the classical Gauss-Newton least squares methods (6) in their many variations and dynamic programming (7). The common feature of these methods was their deterministic character.

The second period covers the early 60's up to now. During this period there was the realization that either because of the inherent stochastic nature of physical systems or because of the fluctuations and uncertainties induced in the system by the environment, one required more sophisticated decision-making algorithms (8). System optimization methods experienced substantial new developments during this period. Amazingly enough, the new methods, which are based on probabilistic techniques and/or the analysis of cognitive processes, do utilize much fewer analytical manipulations than the ones so

profusely needed in the old deterministic methods. In the early 60's, work on the new "less-analytical" algorithms was performed by Rosenbrook (9), Powell (10), and Nelder and Mead (11). The new feature in this work was that one could find the extrema of functionals without calculating time-consuming derivatives of the error functional by some varied, though still "analytical", techniques.

Also in the early 60's, an important event took place with the development by Hooke and Jeeves (12) of the "Direct Search" method. In this technique, the minimization of the functional (i.e. the determination of the optimal parameter set) is accomplished by a series of trial searches in parameter space. Each trial performance is kept in memory and strategies are developed to select new trial points in parameter space. Again the analytical manipulations were reduced to the calculation of the error functional at each trial point. The new method lacked mathematical rigor but could solve optimization problems which were either very difficult to solve or practically untreatable by the well stablished deterministic techniques.

The next step in the evolutionary process of control and optimization methods developed when Waltz and Fu (13), Fu (14) and Saridis (8) endowed the control system with learning capabilities. This development led to the design of "Trainable Controllers" based on the concept of heuristic "Reinforcement Learning" borrowed from the behavioral sciences (8). New and more sophisticated techniques in

Learning Controls did quickly appear based on Pattern Recognition Methods (8,15,16).

1.3 Parameter Identification by Heuristic* Learning

As it was previously mentioned, the parameter identification problem is equivalent to the process of minimization of a given functional. The aim of Heuristic Learning techniques is to perform this process by replacing many analytical tools with heuristic guidelines. In a general sense, Heuristic Learning is the method which endeavors to perform a given function or reach a desired goal by deciding the best action to be taken for each particular situation. Implicit in the method is a decision-making algorithim to select from among a large set of possible actions the ones which expedite the achievement of the set goal. This policy-making facet of heuristic learning can be implemented in many ways, such as the use of "rules of thumb" based on previous experience relating to the behavior of the system under investigation. Upon evaluation of the consequences following a given action, the heuristic learning method stores in memory the best action as well as the situation which was encountered.

* The word heuristic is derived from the Greek heuriskein - to discover, to find. The Webster's Third New International Dictionary defines heuristic as an adjective meaning "serving to guide, discover or reveal". Something that is heuristic is described as "providing aid or direction in the solution of a problem but otherwise unjustified" in the Webster's New Collegiate Dictionary.

In this way, when faced with a similar situation, the system has "learned" the appropriate action to be taken to reach the desired goal.

As mentioned in Section 1.2, a form of the Heuristic Learning method was developed as learning behavior models of living organisms, whereby learning is accomplished by a stimulus-response scheme so that actions with high performances are rewarded and actions with low performances punished. The main components in a heuristic learning algorithm are: (i) the characterization of the "situation", (ii) the selection of the strategy based on heuristic arguments and rules, (iii) the evaluation of the consequences of actions and the reinforcement of optimal decisions, and (iv) the memory organization where the past experiences are stored.

1.4 Previous Applications of Heuristic Learning Methods in the Field of Nuclear Power Plants

Although the importance of heuristic methods in the area of control systems was quickly recognized in the early 70's (14), very few applications in the field of Nuclear Power Plants can be found in the literature. The first application to digital control of nuclear reactors by heuristic methods (to the author's knowledge) was implemented by MacDonald and Koen (17), who addressed the problem of making the reactor follow a desired power output trajectory by the formulation of simple heuristic rules on how to activate the control rods. Although a highly simplified model of the reactor was utilized,

MacDonald succeeded in showing the potential advantages of heuristic learning methods. Bubak and Moscinski (18) and Kitowski and Moscinski (19) utilized a much more sophisticated reactor model to produce a heuristically based digital power-following algorithm. Finally, Hoshino (20) developed a heuristic learning based, optimal strategy for the refueling of a reactor with the goal of maximizing the average discharge burnup.

1.5 Motivation and Objectives

For the past few years, the Nuclear Engineering Deptartment at The University of Tennessee and the Instrumentation and Controls Division at the Oak Ridge National Laboratory have been heavily involved in the development of Surveillance and Diagnostic techniques for Nuclear Power Plants (21-27). This work has resulted in the formulation of surveillance techniques which require on-line parameter identification for assessment of abnormalities in the plant operation. To satisfy this requirement, a least squares program was produced for parameter identification purposes (26). Because of the fact that the plant parameters enter in a highly non-linear fashion into the models for the noise descriptors, the fitting procedure was both time consuming and very sensitive to the initial guesses for the parameters. As a result user-program interaction was needed to "guide" the algorithm to convergence, and on-line operation had to be ruled out. During the fitting process, it was noticed that the user with experience developed a series of "rules of thumb" for the

selection of the initial parameters and their subsequent changes, which greatly facilitate the fitting process. Motivated by these findings, it was decided to explore the possibility of performing on-line parameter identification by replacing the human operator with a learning program.

The following objectives were set for this dissertation:

a. To implement and evaluate heuristic reinforcement learning techniques already developed for other purposes as a tool for a parameter identification algorithm. This step was necessary as previous studies of heuristic methods dealt with essentially "trajectory-following" problems rather than with the specific problem at hand.

b. To investigate the possibility of developing new, more efficient heuristic learning techniques.

c. To validate the new methodology by means of extensive computer simulation studies and the utilization of statistical tests. The motivation behind this objective was that heuristic methods do not guarantee that an optimal solution is obtained (with the exception of particular situations studied later in this work).

d. To apply the newly developed parameter identification method to a "real world" surveillance and diagnostic problem. The requirements for this objective were to perform on-line surveillance with analog signals from a real dynamic system.

1.6 Original Contributions

The original contributions included in this dissertation refer to two different areas: (i) heuristic learning in general and (ii) the field of surveillance and diagnostics relevant to Nuclear Power Plants.

In the area of heuristic learning, the present work contains the first application (known to the author) of heuristic reinforcement learning to the problem of parameter identification, in contrast with the previous applications of this technique to trajectory-following algorithms.

A new learning method, the "Single-Direction Learning" (SDL) parameter identification method, has been developed which is clearly superior to the reinforcement-learning method, the "Fixed-Directions Learning" (FDL), which was developed by adapting the heuristic learning control techniques to the parameter identification problem. During the development of the learning algorithms the new concept of "fuzzy" maps has been introduced, which is of fundamental importance for the implementation and use of the learning methods.

In the area of surveillance and diagnostics, a learning-program has been developed which can be used for on-line diagnostics and which learns the particularities of the parameter identification at hand, thereby increasing its performance with the experience.

1.7 Organization of the Text

In Chapter 2, the general equations for the spectral densities of a linear noise model are presented, and the parameter identification problem is defined and discussed. The two heuristic learning methods developed (the Fixed-Directions Learning and the Single-Direction Learning parameter identification methods) are described in Chapter 3. The computer simulation results with the two heuristic learning methods, as well as comparison with the "Direct Search" method (12), are presented in Chapter 4. The results of the application of the SDL method to the "real world" on-line automatic surveillance and diagnosis of an experimental pressure loop are given in Chapter 5. Finally, a summary of the accomplishments and recommendations for further research are given in Chapter 6.

CHAPTER 2

DEFINITION OF THE PROBLEM AND GENERAL THEORY

2.1 Introduction

In this chapter, the general noise model equations, the definition of parameter identification, and a discussion of parameter identification solutions and the difficulties associated with parameter identification itself are presented in order to introduce notation and to establish a mathematical basis for subsequent chapters.

In Section 2.2, the equations for the frequency domain noise descriptors of a linear, dynamic, stochastic system are presented. These general equations are applied to particular systems in Chapters 4 and 5.

The mathematical definition of parameter identification, as used in the context of this work, is presented in Section 2.3, followed by a discussion of the difficulties associated with parameter identification and their diagnostic implications in Section 2.4.

Although in this chapter, and throughout this work, parameter identification is always discussed in terms of frequency domain noise models and a weighted sum of the squares error functional, it is

important to point out that the learning methods developed in this work are not restricted to these particular applications.

2.2 Noise Model

Noise models are derived from linearized dynamic models of a system, often in lumped parameters form. Using the Langevin source technique (2, 3), the stochastic parameters of a model are separated into steady state values and fluctuating components. The fluctuating parts of the parameter set are treated as extra inputs (or sources) to the system.

In general, the Laplace-transformed noise model can be written in the form

$$B(s)\underline{X}(s) = H(s)\underline{U}(s) \quad , \qquad 2\text{-}1$$

where $\underline{X}(s)$ is an N-dimensional vector whose components are fluctuations of the system variables about steady state values,

$\underline{U}(s)$ is an L-dimensional vector whose components are the inputs to the system, as well as the fluctuating components of the stochastic parameters,

$B(s)$ is an (N x N) system matrix, and

$H(s)$ is an (N x L) forcing matrix.

Left multiplying Equation 2-1 by the inverse of the system matrix yields

$$\underline{X}(s) = B^{-1}(s)H(s)\underline{U}(s) \quad , \qquad 2\text{-}2$$

where the transfer function matrix, G(s), can be identified as

$$G(s) = B^{-1}(s)H(s) \quad . \qquad 2\text{-}3$$

A generic element of the transfer function matrix, $g_{ik}(s)$, is the transfer function from input "k" to the system variable "i". The frequency response matrix is obtained by substituting for "s" by "jw" in Equation 2-3, where "j" is the square root of negative one and "w" is the frequency in radians per second.

The noise descriptors measured in frequency domain noise analysis are usually Power Spectral Densities (PSDs) and Cross Power Spectral Densities (CPSDs) between the available signals. Other common descriptors can, in general, be expressed in terms of the PSDs and CPSDs. The equations for the PSDs and CPSDs of a system described by Equation 2-2 are derived in Appendix A.

The CPSD between two system variables in terms of the frequency response and the input spectral densities is given by

$$S_{\ell m}(w) = \sum_{i=1}^{L} \sum_{k=1}^{L} g^{*}_{\ell i}(w) g_{mk}(w) S_{ik}(w) \quad , \qquad 2\text{-}4$$

where $S_{\ell m}(w)$ is the CPSD between system variables X_ℓ and X_m ,

$g^{*}_{\ell i}(w)$ is the complex conjugate of the frequency response of the system variable X_ℓ to input u_i,

$g_{mk}(w)$ is the the frequency response of the system variable X_m to input u_k , and

$S_{ik}(w)$ is the CPSD between inputs u_i and u_k.

The expression for the PSD is obtained by making "ℓ" equal "m" in Equation 2-4. A common simplifying assumption is to assume that the inputs are uncorrelated. In that case, the CPSDs between the inputs vanish, and Equation 2-4 reduces to

$$S_{\ell m}(w) = \sum_{i=1}^{L} g^{*}_{\ell i}(w) g_{mi}(w) S_{ii}(w) \quad . \qquad 2\text{-}5$$

2.3 Parameter Identification

In general, parameter identification involves the estimation of parameters for a system model from measured experimental data points. Typically, the estimated values of the parameters are the values that minimize an error functional which measures the error between experimental points and model predicted points.

Several forms of the error functional have been used in the past. In this work, the weighted least squares error functional was always used, although the parameter identification methods developed are not restricted to this error functional.

The weighted least squares functional is defined as:

$$Q(\underline{P}) = \sum_{i=1}^{N} W_i (Y_i - F_i(\underline{P}))^2 \qquad 2\text{-}6$$

where Y_i $(i=1,\ldots,N)$ are experimentally obtained components of an N-dimensional measurement vector;

$F_i(\underline{P})$ $(i=1,\ldots,N)$ are components of an N-dimensional,

model-calculated vector that depends on the parameter vector, $\underline{P}$, i.e. $F_i(\underline{P})$ is a model prediction of the experimental point Y_i;

$\underline{P}$ is a K-dimensional parameter vector, with K<N; and

W_i $(i=1,\ldots,N)$ are the components of an N-dimensional weight vector usually taken as the inverse of the variances of the experimental data points, i.e.

$$W_i = \frac{1}{\text{var}[Y_i]} \quad . \qquad 2\text{-}7$$

Parameter identification, within the context of this work, can be defined as the process of finding the value of the parameter vector, $\underline{P}^*$, that minimizes the weighted least squares error functional defined by Equation 2-6. The identified parameter vector, $\underline{P}^*$, is usually called the optimum parameter vector.

When the functions $F_i(\underline{P})$ are continous with continuous first derivatives, the following set of equations must be satisfied at the minimum:

$$\left[\frac{dQ(\underline{P})}{dP_i}\right]_{\underline{P}=\underline{P}^*} = 0 \quad , \qquad \text{for } i=1,2,\ldots,K \quad . \qquad 2\text{-}8$$

An explicit solution for $\underline{P}^*$ from this set of equations exists only when the functions $F_i(\underline{P})$ are linear. A solution of Equations 2-8 has been obtained through either iterative-analytical methods (6, 28,

29) or search methods (9-12). The learning parameter identification methods presented in Chapter 3 belong to the search method category.

2.4 Difficulties Associated with Parameter Identification

There are several types of difficulties associated with parameter identification related to the model formulation and to the experiment design. Some of these difficulties can be eliminated by a careful analysis of the model equations and by sensitivity studies, and others can be avoided by limiting the range of the parameters as discussed below.

The first, and simplest difficulty appears when the sensitivity of the model predictions $F_i(\underline{P})$ is zero for some parameter. That is

$$\frac{dF_i(\underline{P})}{dP_k} = 0 \qquad \text{for } i=1,2,\ldots,K \quad . \tag{2-9}$$

This may happen when some cancellation effect makes the model predictions not a function of that parameter. Obviously, in that case, that parameter can not be identified.

A second type of difficulty occurs when a group of parameters always appears in the model equations in a particular expression. For example, assume that parameters P_k and P_ℓ always appear as the product $P_k P_\ell$ in the model equations. In that case, their sensitivities are proportional, i.e.

$$\frac{dF(\underline{P})}{dP_k} \alpha \frac{dF(\underline{P})}{dP_\ell} , \qquad 2\text{-}10$$

and these parameters can not be simultaneously identified.

The third type of difficulty arises when the error functional in Equation 2-6 is multimodal, i.e. there is more than one point where Equations 2-8 are satisfied. There are no parameter identification methods that are guaranteed to converge to "the minimum" of Equation 2-6 when the error functional is multimodal. The recommended procedure, when there is a possibility that multimodality exists, is to restart the parameter identification from different initial parameter values with the hope that eventually it will converge to the minimum.

There are two distinct types of multimodality:

a. When there are two or more points in parameter space for which Equations 2-8 are satisfied yielding exactly the same model prediction, i.e. $\underline{F}(\underline{P}_1) = \underline{F}(\underline{P}_2)$. In that case, $Q(\underline{P}_1) = Q(\underline{P}_2)$ and the parameter identification is undefined. For example, this happens when a parameter appears squared in the model equations, such that the negative value and the positive value of this parameters give the same answer. It is possible to eliminate some undesirable solutions by limiting the range of the parameters in view of physical considerations.

b. When the minimum is well defined (i.e. there is a point $\underline{P}^*$ where $Q(\underline{P}^*) < Q(\underline{P})$ for all $\underline{P} \neq \underline{P}^*$, and for all other points

satisfying Equations 2-8 the error functional is much greater than $Q(\underline{P}^*)$), a hypothesis test can be performed to accept a particular solution as the optimum parameters, as will be discussed later.

Even when the mathematical parameter identification problem is well defined, numerical problems may cause the parameter identification method to converge far from the true solution. Those problems are caused by the finite precision arithmetic of digital computers and by the convergence criteria utilized. These problems are largely dependent on the geometric characteristics of the error functional surface. These numerical problems can be detected by restarting the parameter identification from different initial parameter values. In addition, the use of higher precision arithmetic can avoid these problems with the penalty of increasing computational time and memory requirements.

The last type of difficulty is caused by inperfections in the model. In that case, a mathematically well defined parameter identification problem can lead to identified parameters with little physical meaning. Two hypothesis tests were used in this work to evaluate the suitability of the model to predict the experimental data:

a. The χ^2 test. When the model predicts well the experimental data, the error functional is randomly distributed according to a χ^2 probability distribution with N-K degrees of freedom (N is the number of data points, and K is the number of

parameters).

b. The Wald-Wolfowitz test(30). This test checks the hypothesis that the experimental data are independently, randomly distributed about the model predictions at the solution point. This method is usually more sensitive than the χ^2 test.

In summary, there are some difficulties associated with parameter identification but most of them can be eliminated by sensitivity studies and by appropriate selection of the model parameters. The multimodality difficulty can be eliminated by either choosing a range for the parameters or by performing hypothesis tests to reject solutions for which the model predictions do not represent well the experimental data. Numerical problems can be minimized by using double precision arithmetic and by judiciously choosing the convergence criteria. The agreement between model and experiment is important, and must be studied before trying to use model predictions for diagnosis. After each parameter identification, the hypothesis test should be performed to accept only solutions for which the model predicts well the experimental data.

CHAPTER 3

HEURISTIC LEARNING PARAMETER IDENTIFICATION

3.1 Introduction

Two heuristic learning parameter identification methods, the fixed-directions learning method and the single-direction learning method, were developed to introduce learning capabilities into the basic parameter identification algorithm. In this basic algorithm, the search for the optimum value of the parameters is performed in a sequence of unidirectional searches in parameter space. The computational efficiency of this algorithm depends on the sequence of directions selected. The basic parameter identification algorithm is explained in Section 3.2.

The ability to characterize and memorize the search situations encountered is fundamental to learning parameter identification methods. Consequently, the experience gained during previous parameter identifications can be remembered to help perform other parameter identifications. The search situation and the simplifications made in its characterization for each learning parameter identification methods are described in Section 3.3.

In Section 3.4, the Fixed-Directions Learning parameter identification method (FDL) is described. In the FDL method, the directions are selected from a set of fixed directions. In the beginning, equal weights are assigned to all directions. The weight of the direction selected, in the current situation, is modified according to the performance of the direction in minimizing the error functional. The weight of the direction selected is either reinforced or penalized depending on whether the direction produces a high or low performance. With this scheme, the method learns the weights of the directions thereby permitting it to improve its computational efficiency by selecting the directions with higher average performances.

In the Single-Direction Learning parameter identification method (SDL), for the first identification, the directions are selected from the set of basic directions (as in the basic algorithm), and all points in the search path are memorized. At the end of each parameter identification, the direction pointing from each point in the search path to the optimum parameter values is stored in a memory cell corresponding to the search situation. This extra learned direction is the one selected when the same search situation is encountered again. The SDL method is described in Section 3.5.

3.2 Basic Parameter Identification Algorithm

In this section, the basic parameter identification algorithm used with both learning parameter identification methods is described. The algorithm is based on breaking the search for the minimum error functional, in the multidimensional parameter space, into a sequence of much simpler, unidimensional searches. A block diagram of this algorithm is shown in Figure 3-1. The algorithm can be divided into four steps:

a. Initialization. In this step, the experimental points are obtained as well as appropriate initial parameter values, from which the iterative search for the minimum error functional starts.

b. Direction selection and unidirectional search. In this step, a direction is selected from a subset of basic directions containing all directions not included in the list of unsuccessful directions. A search along the line defined by the selected direction and by the current value of the parameters is performed. This search is continued until the minimum error functional on the line is found. Details of the unidirectional search are given in Appendix C.

c. Checking the search progress. This step checks whether or not the parameter values have changed significantly during the last search along the line. If they have not changed significantly, then the selected direction is included in a list of unsuccessful directions and step "d" is executed.

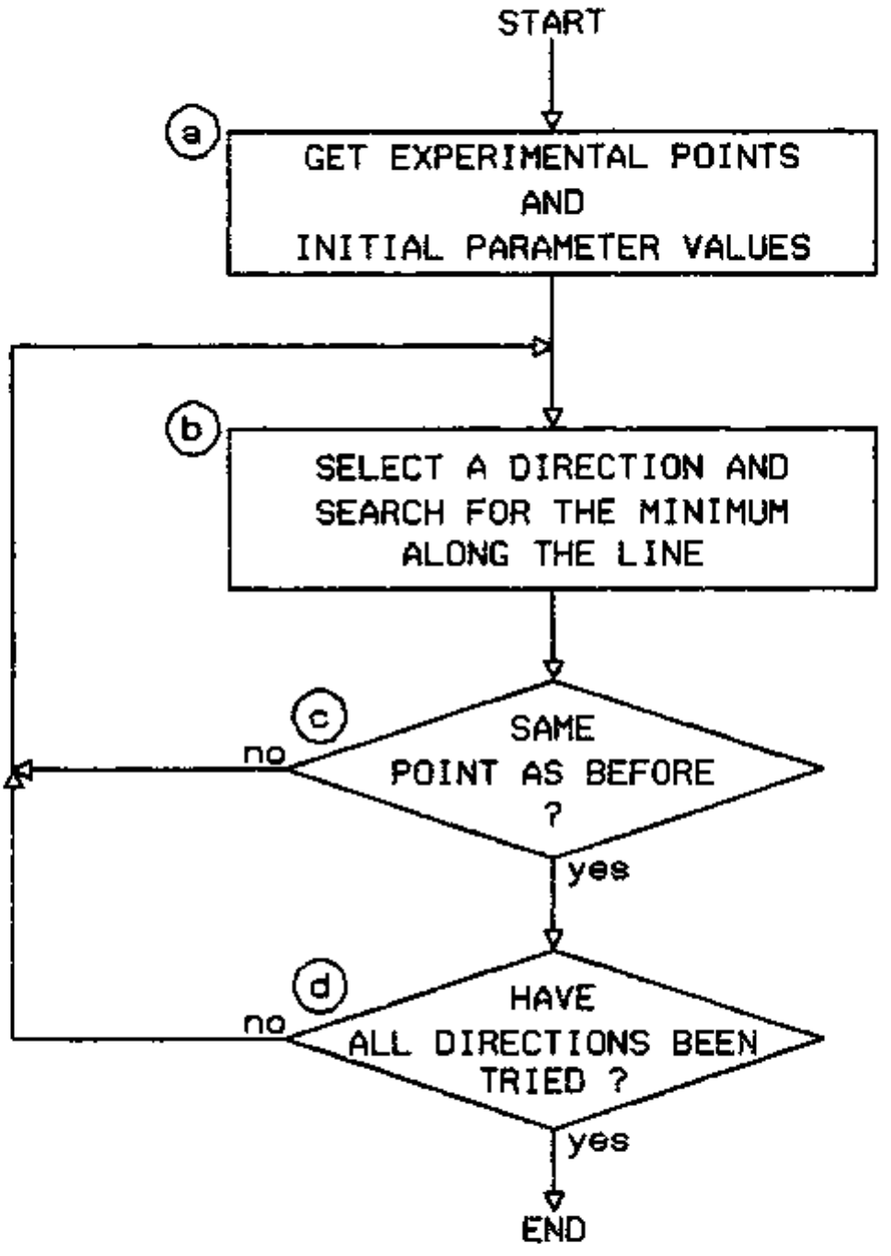

FIGURE 3-1. Block diagram of the basic parameter identification algorithm.

Otherwise, the list of unsuccessful directions is changed to contain only the last selected direction. The last direction is always included in the list of unsuccessful directions because the parameter values after the unidirectional search are already at the minimum along that direction.

d. Convergence check. This step verifies whether unsuccessful searches have been performed along all available directions. In this event, the iterative process has converged to the optimum parameters and the search is ended. Otherwise, step "b" is repeated by selecting a direction among those not included in the unsuccessful list.

It can be shown (see Appendix B) that if the set of directions chosen spans a complete set in parameter space, and if the error functional is a unimodal function of the parameters, then this method always converges to the minimum. However, the efficiency of this method will depend on the set of directions available and on the selection of the direction in each iteration.

3.3 Search Situation Characterization

A learning parameter identification method requires: (a) the ability to characterize the situation and (b) a memory associated to each situation where the information learned is stored. The concept of regions in a feature space, used to characterize the search situation for the two parameter identification methods, is described

in this section.

For a given class of parameter identification problems, where the same model, $\underline{F}(\underline{P})$, is used to represent the experimental points, "$\underline{Y}$", the state of the search (or search situation) is completely characterized by the experimental points, "$\underline{Y}$", and the current value of the parameters, "$\underline{P}$".

Let the situation space be defined as the (N+K)-dimensional space whose coordinates are the "N" experimental points and the "K" components of the current parameter vector. Let $\underline{\delta P}$, the correction vector, be defined by

$$\underline{\delta P} = \underline{P}^* - \underline{P} \quad , \qquad 3\text{-}1$$

where $\underline{P}^*$ is the optimum parameter vector for the given set of experimental points. Since the optimum parameter vector, $\underline{P}^*$, is uniquely defined by the experimental points (except for pathological cases discussed in Section 2.4), each point in situation space corresponds to one, and only one, point in parameter correction space. This map of the situation space into the parameter correction space is described by the unknown functional relationship

$$\underline{\delta P} = \underline{Z}(\underline{Y},\underline{P}) \quad . \qquad 3\text{-}2$$

Although in principle one could consider a scheme to learn this functional relationship directly, the high dimensionality of the situation space makes this approach altogether impractical. Instead, a lower-dimensional space, the feature space, will be used to

characterize the search situation.

Let the feature space be defined as an I-dimensional space whose coordinates, called features, are functions of the experimental points and of the current parameter values. These features can be expressed by the following functional relationship

$$\underline{f} = \underline{W}(\underline{Y},\underline{P}) \quad , \tag{3-3}$$

where $\underline{f}$ is an I-dimensional feature vector ($I < N+K$), and

$\underline{W}(\underline{Y},\underline{P})$ is some functional relationship that extracts the relevant features from the situation space.

Equation 3-3 defines a map connecting each point in situation space to one, and only one, point in feature space. The maps of the situation space into the parameter correction space and into the feature space are illustrated in Figure 3-2.

On the basis of the newly defined feature space, one endeavors to construct a functional relationship between the feature space and the parameter correction space in the form

$$\underline{\delta P} = \underline{M}(\underline{f}) \quad , \tag{3-4}$$

which affords the advantage of being a much lower-dimensional mapping than the one defined by Equation 3-2. However, because of the reduction in dimensionality from the situation space to the feature space, the map defined by Equation 3-3 is not an invertible map, in the sense that two or more points in situation space may map into the same point in feature space. Therefore, as illustrated in Figure 3.2,

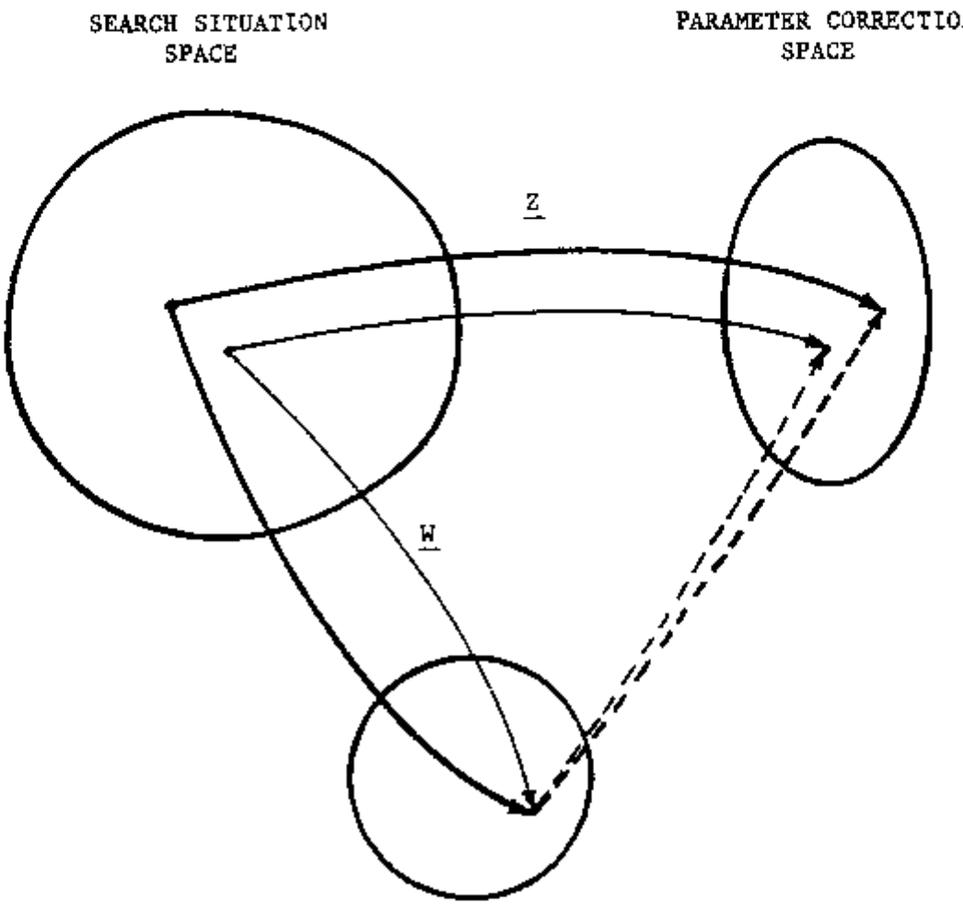

FIGURE 3-2. Maps of the situation space into the feature space and into the parameter correction space.

each point in feature space may correspond to more than one point in parameter correction space.

To construct the functional relationship in Equation 3-4, the concept of "fuzzy-map" is introduced, which is defined as follows.

Two spaces, "A" and "B", are said to be related by a fuzzy-map when to a generic point "$\underline{a}$" in "A" there corresponds a subset $\{ \underline{b}_i \}$ of points in space "B", that share some common characteristics (i.e. have a degree of similarity among themselves) as measured by some heuristically determined consistency index, "c". Then a map can be constructed that associates a generic point "$\underline{a}$" to one, and only one, point in "B" which is defined by the mean vector of the subset $\{ \underline{b}_i \}$.

The application of the fuzzy-map concept to the feature and parameter correction spaces leads to the following mapping relationship:

To each generic point in feature space, "$\underline{f}$", one associates a vector , $\underline{\delta P}$, in parameter correction space defined as the mean vector of the set $\{ \underline{\delta P}_j \}$ (j=1,...,J) which corresponds to the point "$\underline{f}$".

The consistency index, "c", is defined by

$$c = \frac{| \overline{\underline{\delta P}} |}{\overline{| \underline{\delta P} |}} \quad , \qquad 3\text{-}5$$

with

$$\overline{\underline{\delta P}} = \frac{1}{J} \sum_{j=1}^{J} \underline{\delta P}_j \qquad 3\text{-}6$$

and

$$\overline{|\underline{\delta P}|} = \frac{1}{J} \sum_{j=1}^{J} |\underline{\delta P}_j| \quad . \qquad 3\text{-}7$$

On account of Equation 3-5, the consistency index is a number between zero and one that quantifies the degree of similarity between the vectors in the set $\{ \underline{\delta P}_j \}$. Consistency values close to one correspond to a high degree of similarity between the correction vectors in the set $\{ \underline{\delta P}_j \}$, while values close to zero correspond to a low similarity between the vectors.

There is not a systematic way to choose the feature space coordinates, but the consistency is a criteria to rank different feature spaces.

Two types of features were used throughout this work:

a. Feature spaces where the coordinates are moments of the residues; i.e.

$$\rho_n = \sum_{i=1}^{N} w_i^n \, (Y_i - F_i) \quad , \qquad 3\text{-}8$$

where ρ_n is the n^{th} moment of the residues,

w_i is the independent variable, and

$(Y_i - F_i)$ is the residue at the point w_i.

b. The coordinates are differences between some relevant characteristics of the experimental points and of the

calculated points. As relevant characteristics, the center frequency of resonances and the root mean square over certain frequency ranges have been used.

For practical reasons, the continuous feature space was coarse-grained by grouping similar feature vectors into hyperspheric regions defined by center coordinates and a radius. The current state of the search is said to belong to a feature region when the calculated features fall within a distance from the region center less than or equal to the region radius. In the event that the calculated features fall within more than one region, the state of the search is said to belong to the region whose center is closer.

Let "R" be a generic region in feature space and let $\underline{\delta P}^R$ be the mean value of all correction vectors corresponding to any point in the region "R". The mapping defined by

$$R \Longrightarrow \underline{\delta P}^R \qquad 3\text{-}9$$

is the subject learned by the two learning methods described in the following sections. In the FDL method, the mean correction vector is approximated by one of the fixed directions, while in the SDL, an estimate of the mean correction vector is learned directly.

The definition of consistency of a point in feature space can be extended to consistency of a region by defining the mean values in Equation 3-5 over the set of all correction vectors corresponding to points within the feature region.

The consistency of a region can be used as a criteria for

choosing the region radius. The selection of the region radius is a compromise between memory requirements and the region consistency. In Section 3.5.3, a way to estimate the consistency for the SDL method is described.

3.4 Fixed-Directions Learning Parameter Identification (FDL)

The FDL method introduces learning to the basic parameter identification algorithm described in Section 3.2. The term fixed-directions is used because the selection of directions is made among an initial set of directions, in contrast with the SDL method where an extra direction is learned for each different search situation.

3.4.1 Overview of the FDL method

In Figure 3-3, a simplified block diagram of the FDL method is shown. After the initialization step, an iterative procedure starts. At the beginning of each iteration, the feature vector is calculated to characterize the search situation, and a check is performed to verify whether or not the calculated features belong to an existing region. If they do not belong to an existing region, then a new region with center coordinates equal to the calculated features is created. A direction is selected as explained in Section 3.4.4, and a search for the minimum error functional, along the line defined by the current value of the parameters and the selected direction, is performed (see Appendix C for details). At the end of the

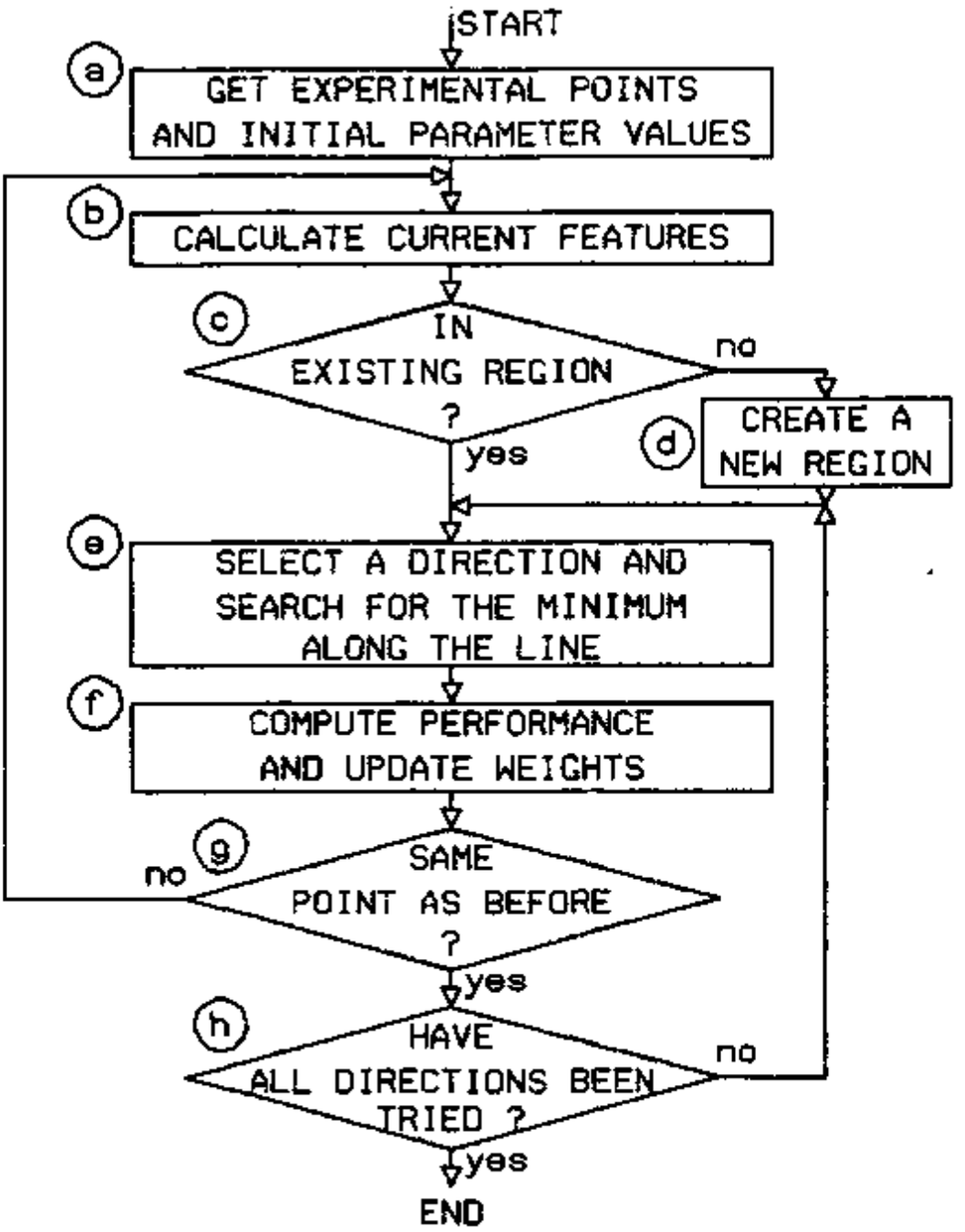

FIGURE 3-3. Block diagram of the Fixed-Directions Learning Parameter Identification Method.

unidirectional search, the index of performance is evaluated (Section 3.4.2), and the information in the memory cell corresponding to the feature region is updated (Section 3.4.3). The convergence check is performed as explained in Section 3.2.

On the average, the parameter identification efficiency increases as more parameter identifications are performed because (a) the probability increases that the features fall in an existing region, which permits selection of the directions according to their weights and (b) the uncertainty in the weights decreases as more updates are performed. Eventually, the whole feature space gets partitioned in regions, all calculated features fall in existing regions, and all directions are selected according to their weights.

3.4.2 Index of performance

Let $Q^{\ell-1}$ and Q_i^{ℓ} be, respectively, the error functional at the beginning of iteration "ℓ" and the error functional at the end of the same iteration; in which the unidirectional search was performed along the direction "i". The index of performance for direction "i" in the situation existing at the beginning of iteration "ℓ" can be defined as:

$$I_i^{\ell} = \frac{Q^{\ell-1} - Q_i^{\ell}}{Q_i^{\ell}} \quad . \qquad 3\text{-}10$$

The best direction for that situation, "j", can be defined as the direction whose index of performance satisfies

$$I_j^{\ell} = \underset{i}{\text{MAX}}\ I_i^{\ell} \quad . \qquad 3\text{-}11$$

The goal of the FDL method is to learn how to select the best direction for each search situation.

3.4.3 Memory organization and updating

For each region in feature space, i.e. each search situation, a memory cell is assigned. The memory cell corresponding to a generic region, "R", contains the following information:

- f_i^R (i=1,2,...,I) - coordinates of the region center,
- W_j^R (j=1,2,...,M) - the weights of each direction in the search situation corresponding to this region,
- n_j^R (j=1,2,...,M) - the number of times each direction was used within this region,
- I_L^R - the lowest index of performance of any direction when used within this region, and
- d^R - The average distance moved in parameter space independent of the direction used.

After performing each unidirectional search, the index of performance is evaluated, and the information in the memory cell corresponding to the feature region at the start of the unidirectional search is updated. If direction "j" was selected to define the line for the unidirectional search, its weight is modified by the linear reinforcement scheme

$$(W_j^R)_{up} = \alpha.W_j^R + (1-\alpha).I_j^R \quad , \qquad 3\text{-}12$$

where W_j^R is the weight of direction j, in the feature region at the start of the unidirectional search, before updating,

$(W_j^R)_{up}$ is the corresponding updated weight,

I_j^R is the direction index of performance during the last unidirectional search, and

α is the learning factor.

The updated weight will increase when the index of performance is larger than the weight before updating, and it will decrease when the index of performance is smaller. The learning factor, "α", is a number between zero and one that controls how fast the weight is allowed to change during each update. When the learning factor is close to one the change is very slow, and when close to zero the change is very fast. A value of the learning factor given by

$$\alpha = \frac{n_j^R}{n_j^R + 1} \qquad 3\text{-}13$$

was typically used. This value makes the weight of a direction equal to the average index of performance in the region.

The remaining information in the cell is updated as follows:

$$(n_j^R)_{up} = n_j^R + 1 \quad , \qquad 3\text{-}14$$

$$(I_L^R)_{up} = I_j^R \text{ if } I_j^R < I_L^R \text{ , otherwise remains the same,} \qquad 3\text{-}15$$

and $$(d^R)_{up} = \alpha.d^R + (1-\alpha).|\underline{P}_o - \underline{P}| \quad , \qquad 3\text{-}16$$

where $\underline{P}_o$ is the parameter vector before the unidirectional search, and

$\underline{P}$ is the parameter vector after the unidirectional search.

3.4.4 Direction selection

Two schemes for direction selection were used. Before all directions have been used in a region, a direction not yet utilized is selected either sequentially or at random. After all directions have been selected at least one time in the region, a direction is statistically selected from a selection probability defined as

$$S_j^R = \frac{(W_j^R - I_L^R + 1/N^R)^k}{\sum_{i=1}^{M} (W_i^R - I_L^R + 1/N^R)^k} \quad , \qquad 3\text{-}17$$

where S_j^R is the selection probability of direction j,

N^R is the total number of times the memory cell has been updated, and

k is the probability enhancement parameter, which should reflect the degree of confidence in the weights (typically $k = N^R/M$ has been used).

3.5 Single Direction Learning Parameter Identification (SDL)

In the previous section, the FDL method was described in which the efficiency of the parameter identification is improved by learning how to select the direction that maximizes the index of performance at each iteration from a set of fixed directions. In the present method, the basic parameter identification is performed in the same way, but an extra direction is learned for each region in feature space (or each search situation). The learned direction is an estimate of the region mean correction vector defined in Section 3.3.

Since, for this method, a new direction vector is defined for each region in feature space, the basic set of directions can be reduced to the minimum necessary to assure convergence. Typically, a number of basic directions equal to the number of parameters was used, where each direction is aligned with one of the coordinate axes in parameter space.

3.5.1 Overview of the SDL method

In Figure 3-4, a simplified block diagram of the SDL method is shown. As in the FDL method, an iteration is started by calculating the features and checking whether or not the calculated features belong to an existing region in feature space. If not, then a new region is created whose center coordinates are the calculated features. After that, the region number and the current value of the parameters are saved for posterior updating of the memory (see Section 3.5.2). In step "f", as in the FDL method, a direction is selected,

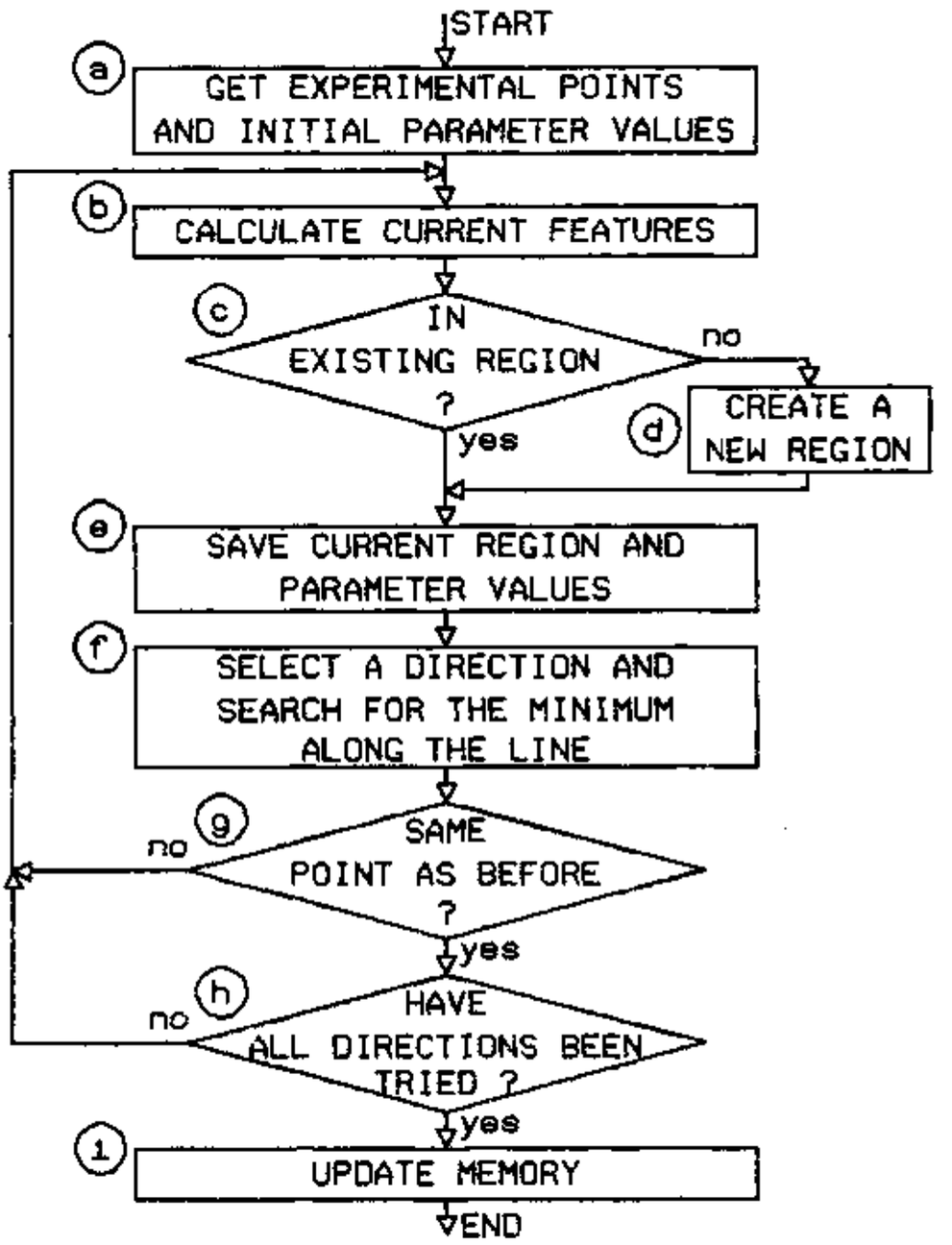

FIGURE 3-4. Block diagram of the Single-Direction Learning Parameter Identification Method.

and an unidirectional search is performed. In contrast with the FDL method, no memory updating is performed during the iterations, but only after the parameter identification has been concluded.

When the search is finished, the region number and the parameter values at the beginning of each iteration are known, as well as the optimum parameter values. The differences between the optimum parameters and the value of the parameters at the beginning of each iteration are samples of the correction vector defined in Section 3.3. These samples are used to update the estimate of the region mean correction vector in the corresponding memory cells (see Section 3.4.2).

Besides the region dependent information, this method keeps the directions in a priority queue with the highest priority direction in front. When the learned direction is not available, the direction in the front of the queue is selected. The priority of the directions are updated according to the index of performance and according to the direction of the last direction selected (see Section 3.5.5). This priority queue helps to improve the parameter identification efficiency both during the first parameter identifications (when the learned directions are available for a few regions only) and at the end of each parameter identification, when the basic directions must be selected to assure convergence.

As in the FDL method, the parameter identification efficiency increases, on the average, as more parameter identifications are performed because (a) the probability increases that the features fall

in existing regions, which permits selection of the estimate of the mean correction vector for the unidirectional search, and (b) the uncertainty in the estimate of the mean correction vector decreases as more updates are performed. Eventually, the whole feature space gets partitioned into regions, all calculated features fall in existing regions, and an estimate of the mean correction vector is available for all regions.

3.5.2 Memory organization and updating

The information stored in memory for region "R" in parameter space is:

f_i^R $(i=1,\ldots,I)$	- coordinates of the region center,
$\underline{\Delta P}^R$	- average correction vector,
A^R	- average magnitude of the correction vector, and
N^R	- total number of times the region has been updated.

Let R^ℓ and $\underline{P}^\ell$ be the region number and the parameter vector at iteration "ℓ" of a parameter identification, where "L" iterations were necessary for convergence. For each of the "L" iterations, the information in the memory cell corresponding to the region at the beginning of the iteration is updated as follows:

$$(\underline{\Delta P}^{R})^{\ell}_{up} = \alpha . \underline{P}^{R\,\ell} + (1-\alpha).(\underline{P}^{*} - \underline{P}^{\ell}) \quad , \qquad 3\text{-}18$$

$$(A^{R})^{\ell}_{up} = \alpha . A^{R\,\ell} + (1-\alpha).|\underline{P}^{*} - \underline{P}^{\ell}| \quad , \qquad 3\text{-}19$$

and $$(N^{R})^{\ell}_{up} = N^{R\,\ell} + 1 \quad , \qquad 3\text{-}20$$

where $$\alpha = \frac{N^{R\,\ell}}{N^{R\,\ell} + 1} \quad , \qquad 3\text{-}21$$

and $\underline{P}^{*} = \underline{P}^{L}$, the optimum parameter vector.

3.5.3 Region consistency estimation

The average magnitude of the correction vector and the average correction vector are used to estimate the consistency of a region, "R", as:

$$C^{R} = \frac{|\underline{\Delta P}^{R}|}{A^{R}} \quad . \qquad 3\text{-}22$$

The consistency gives an idea of how much the correction vector varies in a region. If the direction of the correction vector is the same every time the region is updated, "C^{R}" will be equal to one. In the extreme case where the directions are completely random, the consistency will asymptotically converge to zero as more updates are performed.

The consistency is very useful in determining the adequacy of the feature space and region sizes chosen. It is used to attach a confidence index to the learned direction, in order to save computational effort during the unidirectional search (see

Appendix C), and to avoid selecting the learned direction when the consistency is lower than a threshold.

3.5.4 Direction selection

The direction selected is always the learned direction unless one of the following conditions exist:

a. The current region is a new region (no learned direction available).

b. The current region is the same as the one in the previous iteration.

c. The region consistency is less than 0.1.

In the event that one of the above conditions arise, the direction selection is performed with the help of a direction priority queue. The direction in the first place of the queue, i.e. the direction having highest priority, is selected.

In the beginning of a parameter identification, the priority of the basic directions are initialized giving higher priority to the directions that have been used more frequently in past parameter identifications. However, for the first iteration, the direction in front of the queue is the direction that has the highest average performance in the first iterations of previous parameter identifications.

3.5.5 Direction priority updating

The priority of the direction selected is updated according to its index of performance (see Section 3.4.2), as follows:

$$(T_i)_{up} = \alpha . T_i + (1-\alpha) I_i \quad , \qquad 3\text{-}23$$

where T_i is the priority of direction i before updating,

$(T_i)_{up}$ is the updated priority, and

I_i is the index of performance of the selected direction.

It is desirable to update the priority swiftly, therefore, a low value for the learning factor "α" (typically 0.1) is used.

The priorities of all directions are updated to reinforce the directions that are very different from the selected one and to penalize the directions that are similar to the selected one. The following updating scheme is used:

$$(T_j)_{up} = T_j . \left[\frac{2}{1 + 3.(\cos\theta_{ij})^2}\right]^2 \quad , \qquad 3\text{-}24$$

where θ_{ij} is the angle between the selected direction and the one whose priority is being updated.

With this scheme, the priority of the directions orthogonal to the selected one are doubled, while the priority of the directions very similar to the selected one are halved.

CHAPTER 4

COMPUTER SIMULATION RESULTS

4.1 Introduction

During the development of the learning parameter identification methods described in Chapter 3, a large number of computer simulations were performed to compare different strategies for direction selection, weight updating techniques, and different feature space definitions; as well as to check the validity of the methods. In this chapter some basic results,as well as a comparison between the FDL and the SDL methods, is presented. Also presented is a comparison between the SDL method and the Direct Fit method of Hooke and Jeeves (12).

Two system models, which are described in Section 4.2, were used for the computer simulations. The techniques used in the simulation of experimental data are described in Section 4.3. In Section 4.4, a graphical technique to visualize the mapping between the feature space and the parameter correction space is described. In Sections 4.5 and 4.6, the results of some simulations with the FDL method and with the SDL method are presented.

4.2 System Models Used for Parameter Identification

Two system models were used in the simulations presented in this chapter: (a) a simple dynamic model represented by a second order ordinary differential equation, and (b) a fourth order state variable model for a U-tube steam generator.

The second order ordinary differential equation model used was

$$\frac{d^2x}{dt^2} + a_1\frac{dx}{dt} + a_0x = b_1\frac{du}{dt} + b_0u \quad , \qquad 4\text{-}1$$

where x is the system variable considered measurable,

u is the system single input, and

a_0, a_1, b_0 and b_1 are constant coefficients.

Laplace transforming Equation 4.1, assuming zero initial conditions, and rearranging yields

$$\frac{x(s)}{u(s)} = \frac{b_0 + sb_1}{s^2 + a_1s + a_0} \quad , \qquad 4\text{-}2$$

which is by definition the system transfer function, g(s). Applying Equation 2-5 for a single input, single output system yields

$$S_{xx}(w) = \frac{b_0^2 + w^2b_1^2}{(a_0 - w^2)^2 + w^2a_1^2}\, S_{uu}(w) \quad , \qquad 4\text{-}3$$

where $S_{xx}(w)$ is the system variable PSD, and

$S_{uu}(w)$ is the input PSD.

Two parameters were identified with this model: a_1 and S_{uu} (which was considered independent of the frequency). The typical value of the coefficients and parameters were

$$a_0 = 2 \ , \quad a_1 = 1 \ , \quad b_0 = 1 \ , \quad b_1 = 1 \ , \text{ and } \quad S_{uu} = 1 \quad .$$

The PSD of the system variable for two values of parameter a_1 is shown in Figure 4-1; in Figure 4-2, the same PSD is shown for two different values of S_{uu}.

The noise model for the U-tube steam generator was derived from basic conservation equations (see Appendix F) in a similar way to that described by Ali (31). The state variables in this model are:

1. δTp - variation from equilibrium of the temperature of the primary water inside the tubes,
2. δTm - variation from equilibrium of the temperature of the metal tubes,
3. δP - variation from equilibrium of the secondary pressure, and
4. δMw - variation from equilibrium of the mass of secondary water.

The input variables are:

1. δTpi - variation from equilibrium of the primary water inlet temperature,
2. δHfi - variation from equilibrium of the enthalpy of the feedwater,
3. δWf - variation from equilibrium of the feedwater mass flow rate,

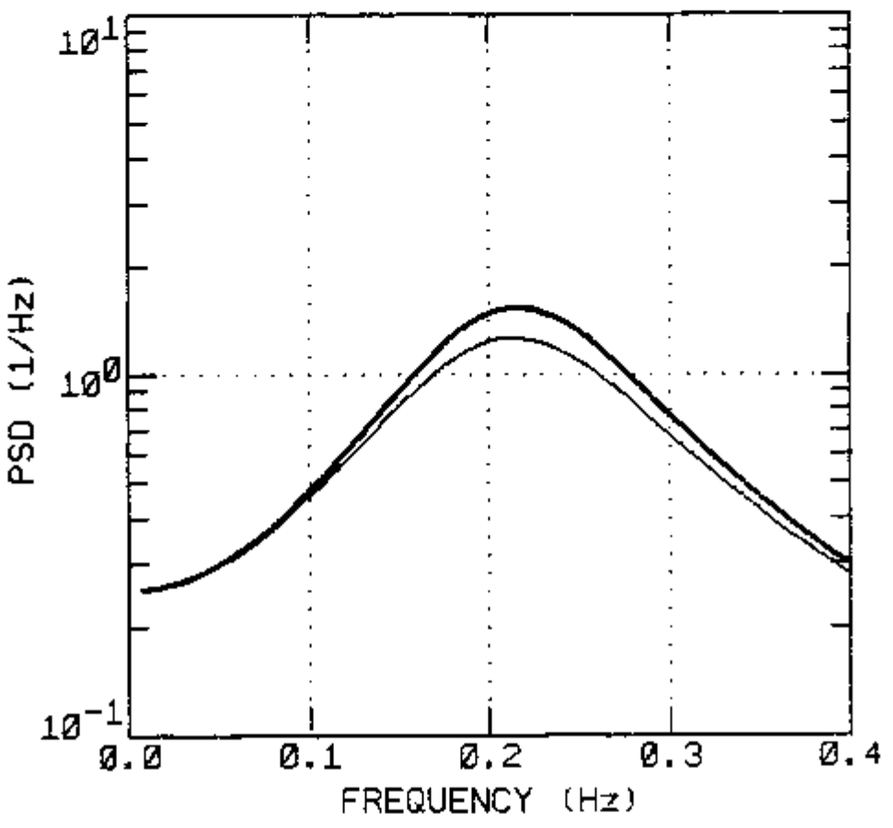

FIGURE 4-1. System Variable PSD for $a_1 = 1$ (thick line) and for $a_1 = 1.1$ (thin line).

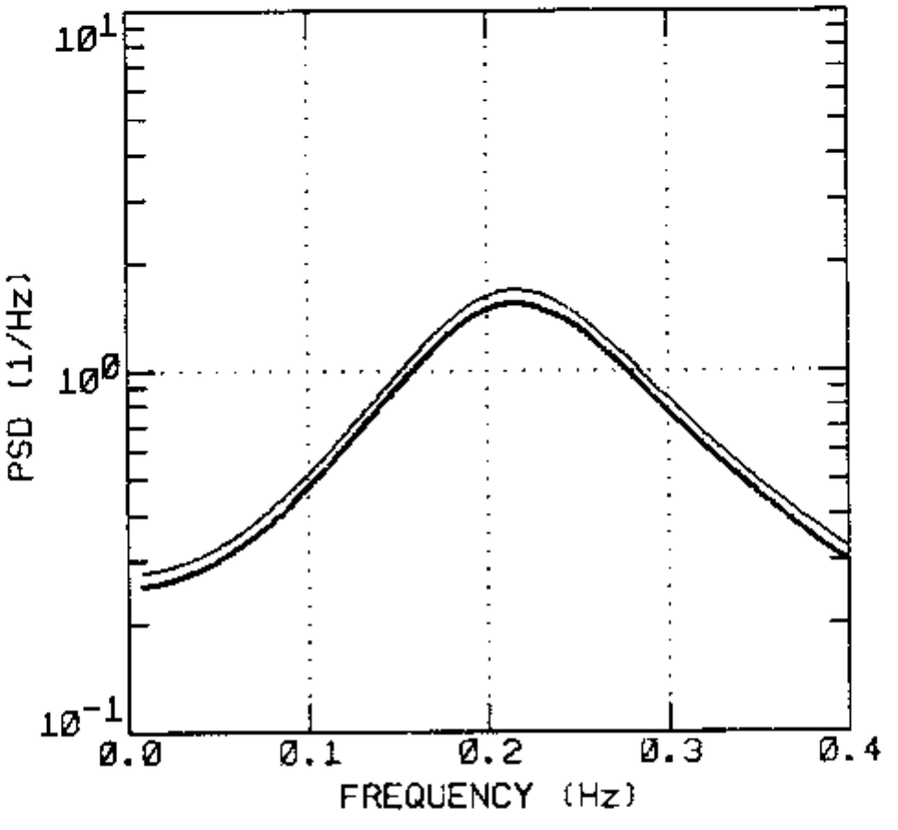

FIGURE 4-2. System Variable PSD for $S_{uu} = 1$ (thick line) and for $S_{uu} = 1.1$ (thin line).

4. δW_s - variation from equilibrium of the steam mass flow rate, and

5. δU_{ms} - variation from equilibrium of the overall heat transfer coefficient between the metal tubes and the secondary side.

The detailed model equations for a PWR steam generator are given in Appendix F. In Figures 4-3 through 4-7, the frequency responses of the primary water temperature, δT_p, with respect to each of the input variables are shown.

4.3 Simulation of Experimental Data

The experimental data were simulated by adding experimental errors to the exact values calculated using the model equations. The probability distribution of the errors added was similar to the probability distribution of experimentally measured data, as explained below.

For experimental PSD data, it is known (32) that the relative standard deviation of the PSD estimates, $S^{(n)}$, obtained by averaging "n" estimates computed from independent time records (n-block estimate) is given by

$$\frac{\sigma[S^{(n)}]}{S} = n^{-1/2} \quad . \qquad 4\text{-}4$$

Furthermore, the probability distribution function of the PSD estimates can be well approximated by a Gaussian distribution for

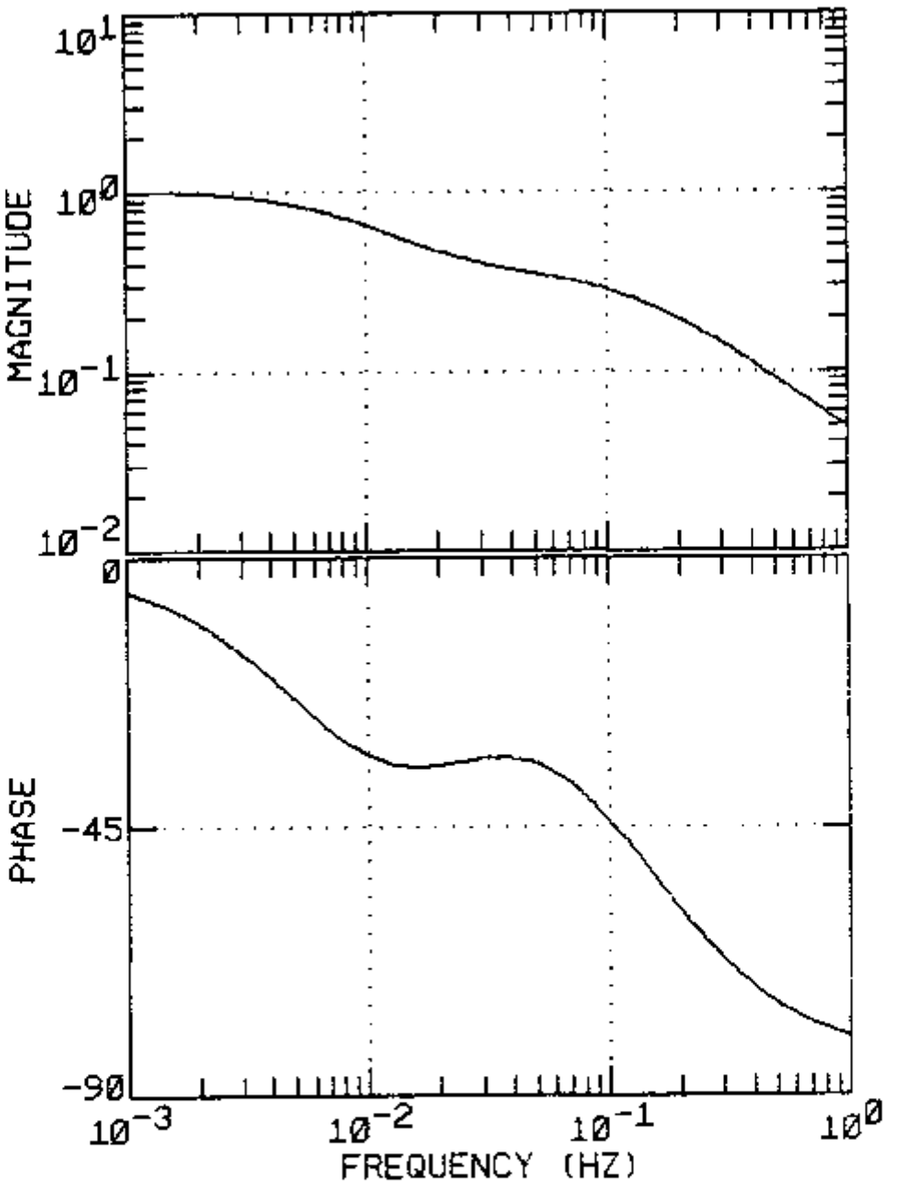

FIGURE 4-3. Frequency response of the primary water temperature to inlet temperature.

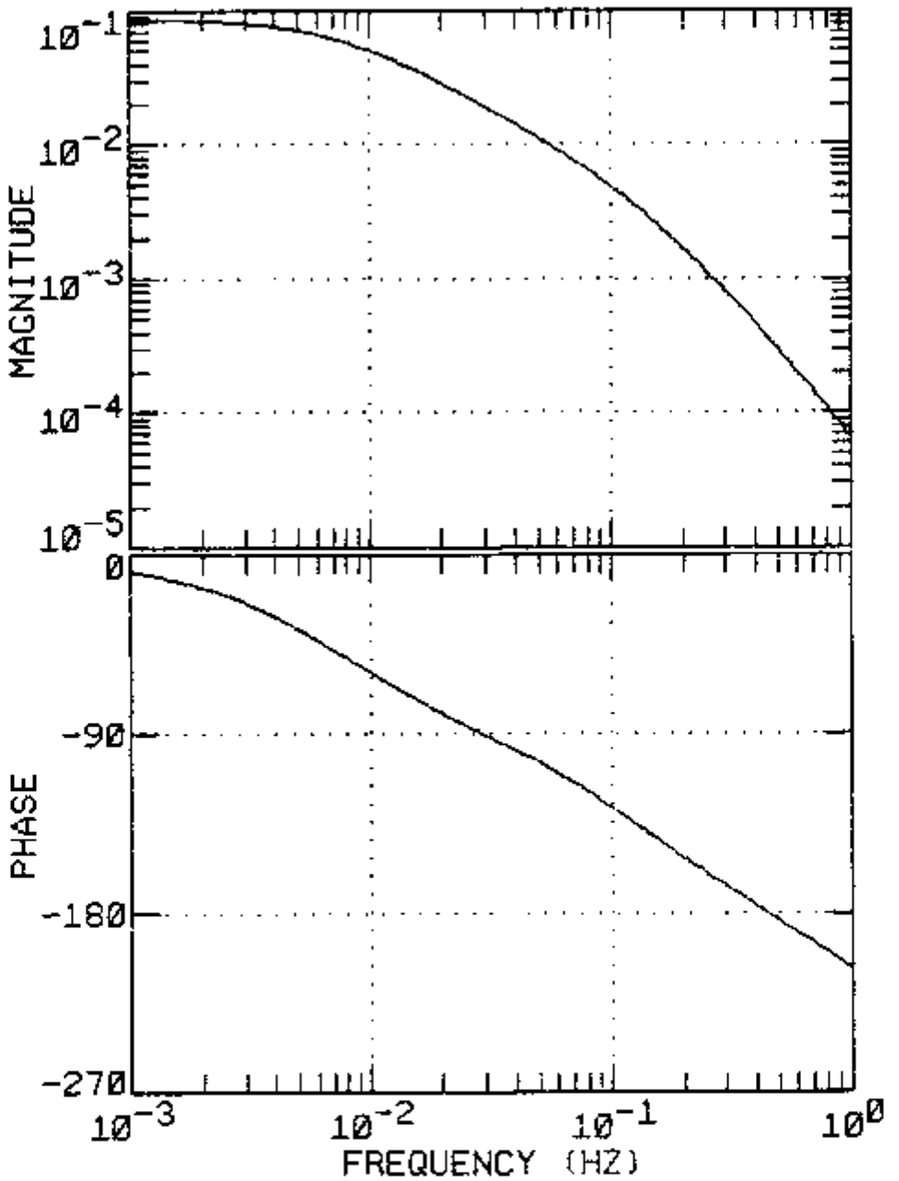

FIGURE 4-4. Frequency response of the primary water temperature to feedwater temperature.

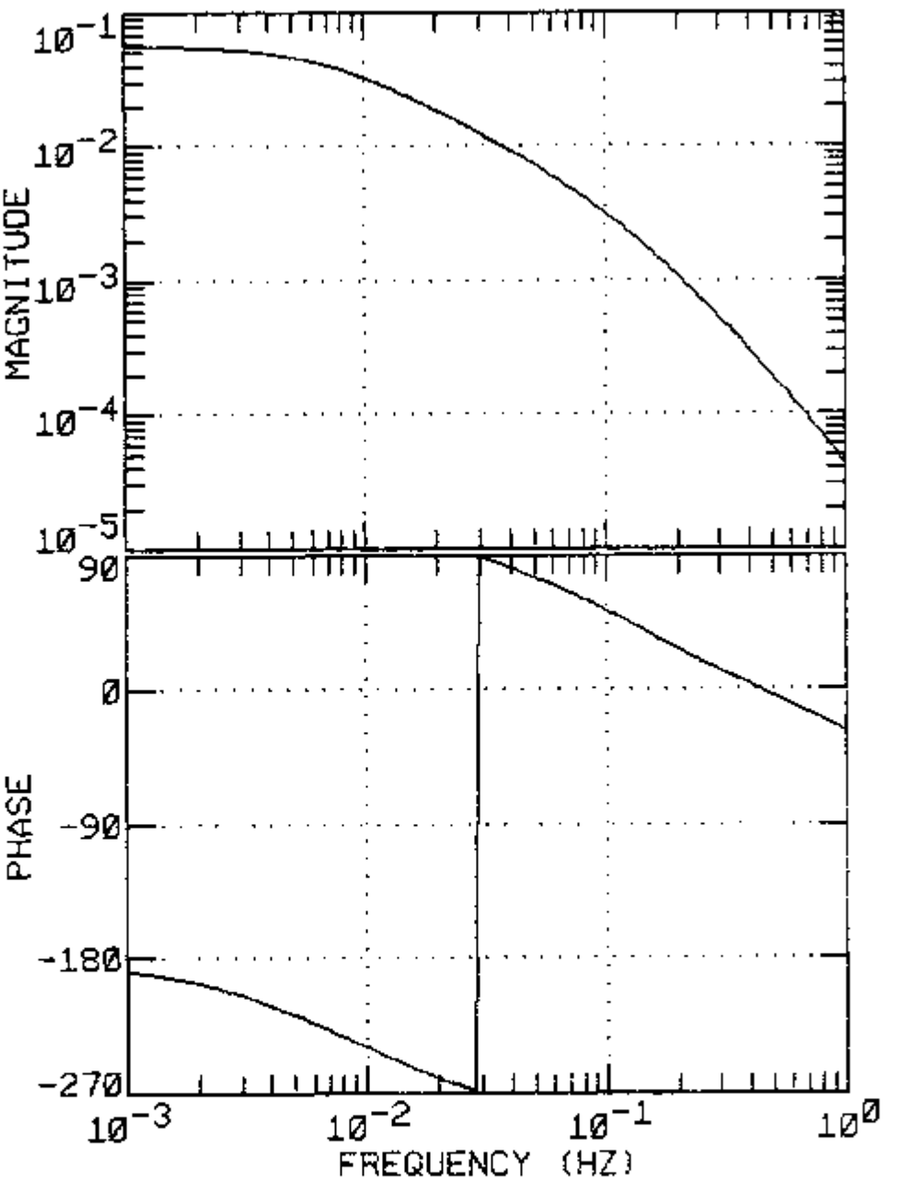

FIGURE 4-5. Frequency response of the primary water temperature to feedwater mass flow rate.

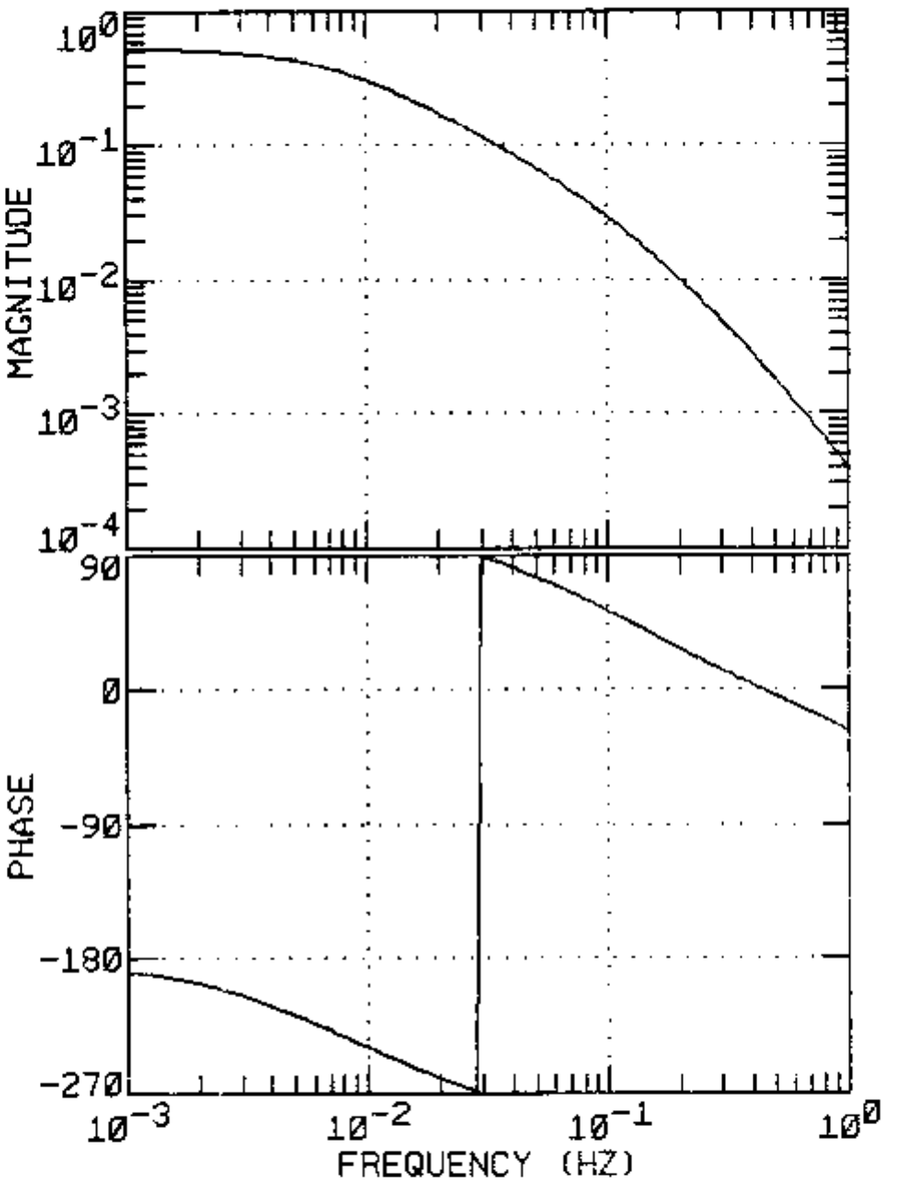

FIGURE 4-6. Frequency response of the primary water temperature to steam mass flow rate.

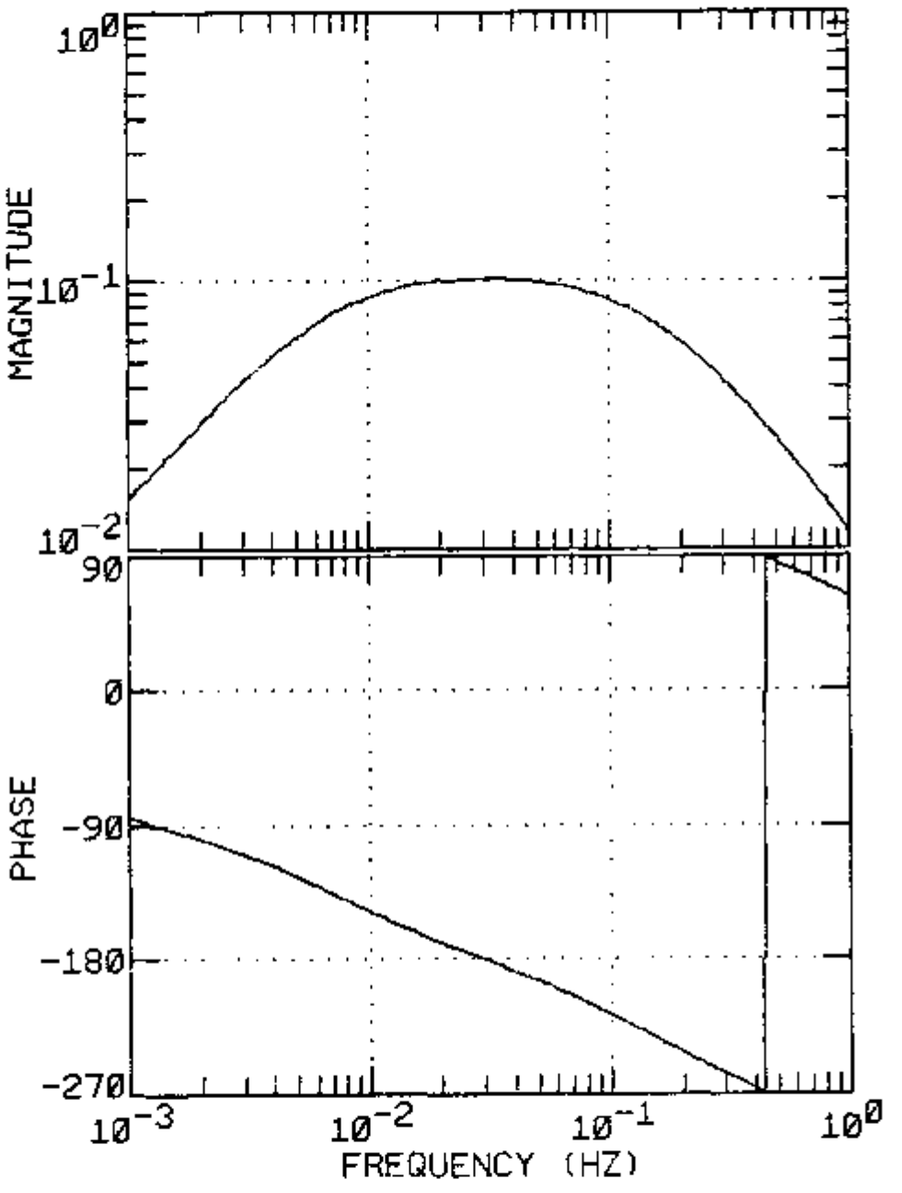

FIGURE 4-7. Frequency response of the primary water temperature to metal-secondary heat transfer coefficient.

number of blocks greater than 15 (32).

To simulate an n-block experimental PSD with the above distribution, the following expression was used:

$$S_s^{(n)} = F + F\varepsilon n^{-1/2} \quad , \qquad 4\text{-}5$$

where $S_s^{(n)}$ is the simulated experimental PSD,

F is the exact value of the PSD calculated with the model equations, and

ε is a random variable with a normal probability distribution.

The normal random variable, ε, was generated using the expression (33)

$$\varepsilon = (-2.\ln\tau_1)^{1/2} \cos(2\pi\tau_2) \quad , \qquad 4\text{-}6$$

where τ_1 and τ_2 are random numbers uniformly distributed in the interval from zero to one.

4.4 Features and Feature Space Consistency

As discussed in Section 3.3, an essential aspect of the learning methods is the utilization of a suitable feature space. For a two parameter, two feature, parameter identification, the adequacy of the feature space selected can be verified by graphical techniques. The "fuzzy map" from feature space to parameter correction space can be visualized using the following procedure:

1. Simulate the experimental data for an arbitrary true parameter vector, $\underline{P}^T$.

2. Generate a random parameter vector, $\underline{P}^R$, and evaluate the two components of the feature vector.

3. Draw at the position corresponding to the calculated features a line segment making an angle with the axis f1 equal to the angle that the correction vector, $\underline{P}^T - \underline{P}^R$, makes with the P_1 axis.

4. Repeat steps 1 through 3 until the map becomes well defined.

Figure 4-8 shows a graph created using the above procedure. This graph was constructed using the second order differential equation model described in Section 4.2. The two dimensional feature space components were proportional to the area of the residuals and to the first moment of the residuals, defined by the following expresions:

$$f_1 = \frac{1}{A} \sum_{i=1}^{N} W_i^{1/2} (Y_i - F_i) \quad , \text{ and} \qquad 4\text{-}7$$

$$f_2 = \frac{1}{A\bar{w}} \sum_{i=1}^{N} W_i^{1/2} (Y_i - F_i) (w_i - \bar{w}) \quad , \qquad 4\text{-}8$$

with $$A = \sum_{i-1}^{N} W_i^{1/2} Y_i \quad , \text{ and} \qquad 4\text{-}9$$

$$\bar{w} = \frac{1}{N} \sum_{i=1}^{N} w_i \quad . \qquad 4\text{-}10$$

It can be noticed in Figure 4-8 that the directions of the correction vectors are well defined within regions in feature space. This indicates that this feature space has a high (close to one) consistency index (see Section 3.3). It can also be noticed that the

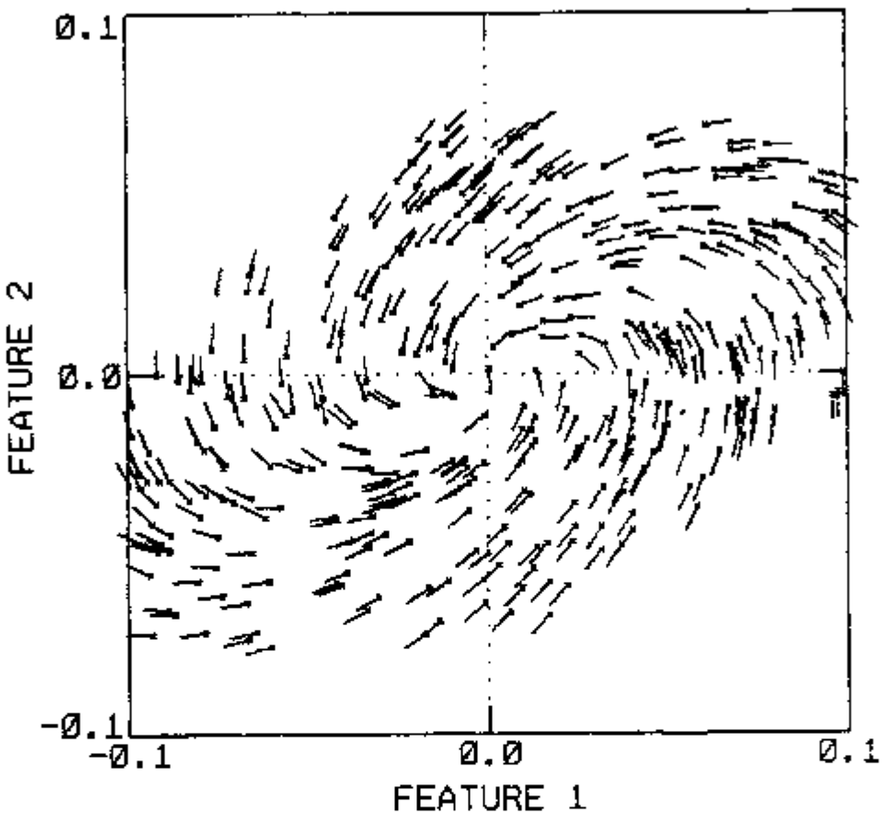

FIGURE 4-8. Visualization of feature space into parameter correction space mapping.

direction of the correction vector differs more between neighboring points close to the origin than for neighboring points in the peripheric regions. It was for this reason that the regions radii in feature space were chosen as a linearly increasing function of the distance from the origin.

This graphical technique is very useful in deciding the size of the regions and the adequacy of the feature space. Unfortunately, it is not possible to use this technique for higher-dimensional feature spaces or parameter spaces.

4.5 Fixed-Directions Learning (FDL) Method Results

4.5.1 Results with the second order differential equation model

Many tests were performed with this model because of its intrinsic simplicity. In this section, a detailed presentation is given of the results of a series of parameter identifications performed under the conditions specified in Table 4-1. For these results, a two dimensional feature vector, whose components are specified, respectively, by Equation 4-7 and Equation 4-8, was used.

The learning ability of a parameter identification method can be shown as a learning curve, which is a plot of the computational time required per parameter identification as a function of the number of identifications performed. The learning curve presents statistical fluctuations due to the random nature of the parameter identification procedure. To decrease the amount of fluctuations, the learning curve is repeated several times and the results are averaged. Typically,

TABLE 4-1. Simulation conditions used to generate the results presented in Figures 4-9 through 4-13.

Model	Second order differential eq.
Parameters identified	a_1 and S_{uu}
True parameters interval (uniformly distributed)	0.9 to 1.1 (all parameters)
Initial parameter values	1.0 (all parameters)
Experimental points	S_{xx}
Number of frequency points	50
Frequency range (uniform)	0.02 Hz to 1 Hz
Experimental error	1%
Number of features	2
Region size	(radius) =0.003 + 0.2(distance)
Number of directions	13
Precision criteria	0.0001
Number of parameter identifications per series	50
Number of series	100

the computational time required to perform the first parameter identification is high but it decreases rapidly for the subsequent parameter identifications, reaching an approximately constant level after a certain number of parameter identifications.

The learning curve shown in Figure 4-9 was obtained by averaging 100 series of 50 parameter identifications each. The number of unidirectional searches executed during each parameter identification was used as a measure of the computational time. It can be noticed that the number of unidirectional searches per parameter identification decreases from an average of 90, for the first parameter identification, to an asymptotic value of approximately 20.

The corresponding total average number of regions in feature space for these series of parameter identifications are shown in Figure 4-10. Predictably, the rate at which the regions are created per parameter identification decreases with the number of parameter identifications performed. The regions created in feature space during the last series of parameter identifications are shown in Figure 4-11.

It can be seen in this figure that most of the space near the origin has been partitioned into regions. Each existing region has a learned direction; therefore, when the features fall in existing regions, the search proceeds along learned directions (which point toward the minimum in the ideal case). When the parameters are close to the optimum values, the features are close to the origin (i.e. in existing regions) from the feature definition. As a result, part of

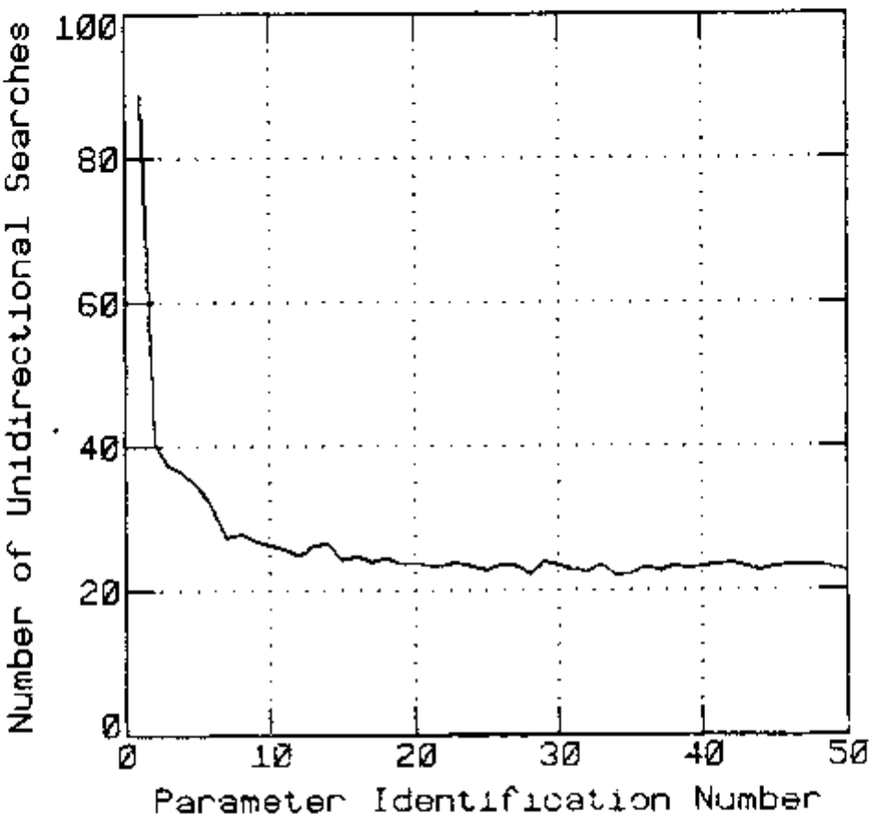

FIGURE 4-9. Learning Curve (conditions in Table 4-1).

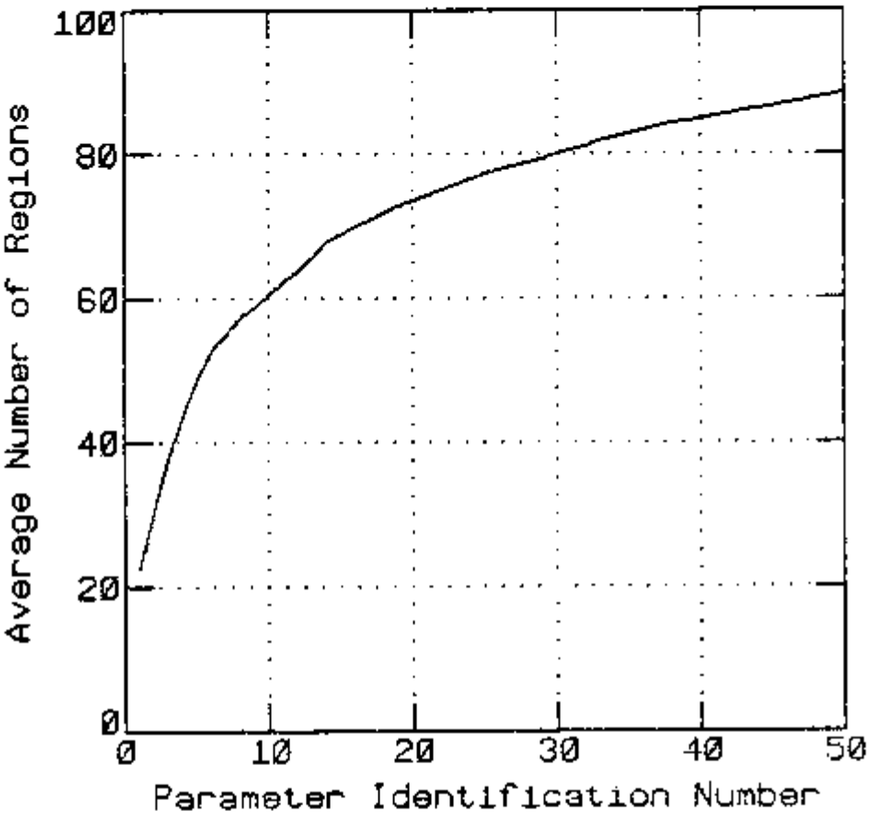

FIGURE 4-10. Average number of regions existing as a function of the number of parameter identifications performed (conditions in Table 4-1).

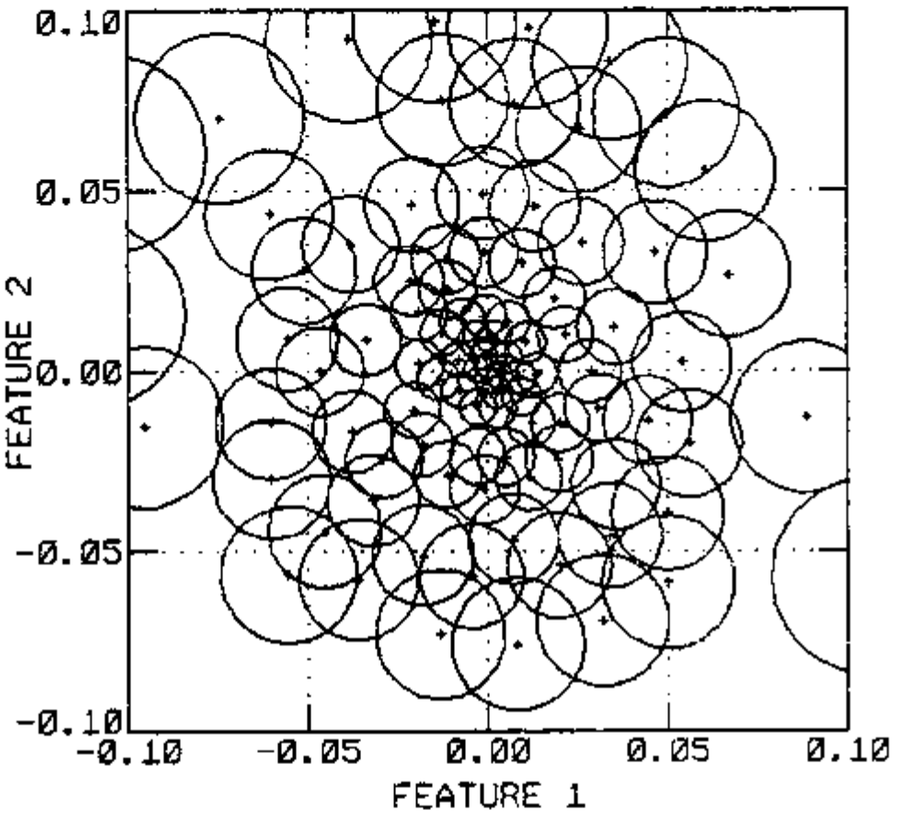

FIGURE 4-11. Regions in feature space, after 50 parameter identifications.

the search path for all parameter identifications is close to the origin of the feature space. These facts makes subsequent parameter identifications converge faster since their paths will pass near the origin, which is in a set of regions containing optimal learned directions.

When the experimental data are computer simulated, the true value for each parameter is known. Thus, it is possible to plot the true value of the parameters versus the identified values to verify the variance of the identified parameters and any convergence problems of the parameter identifications. In Figure 4-12, a plot of the true value of parameter 1 versus the identified value of parameter 1 is shown for all the parameter identifications in these series. In Figure 4-13, the corresponding plot for parameter 2 is shown. The good agreement between the true parameters and the corresponding identified values can be noticed from these figures. The difference between the true parameters and the identified parameters is caused by the experimental error and, in the limit, when the experimental error goes to zero, the identified parameters become equal to the true parameters.

4.5.2 Results with the steam generator model

Tests were also performed using simulated data from the steam generator model with either two or three parameters. For the two parameter case, the results were, in general, similar to the results obtained using the second order differential equation model, with the significant exception that, for the steam generator model, the error

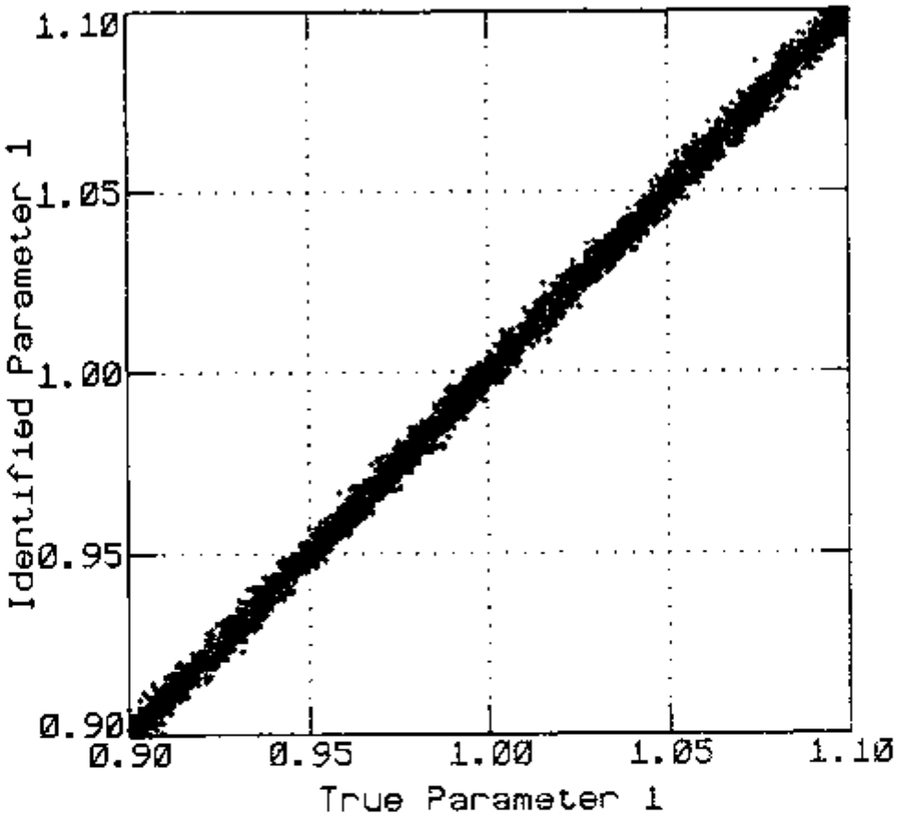

FIGURE 4-12. Plot of true-versus-identified Parameter 1.

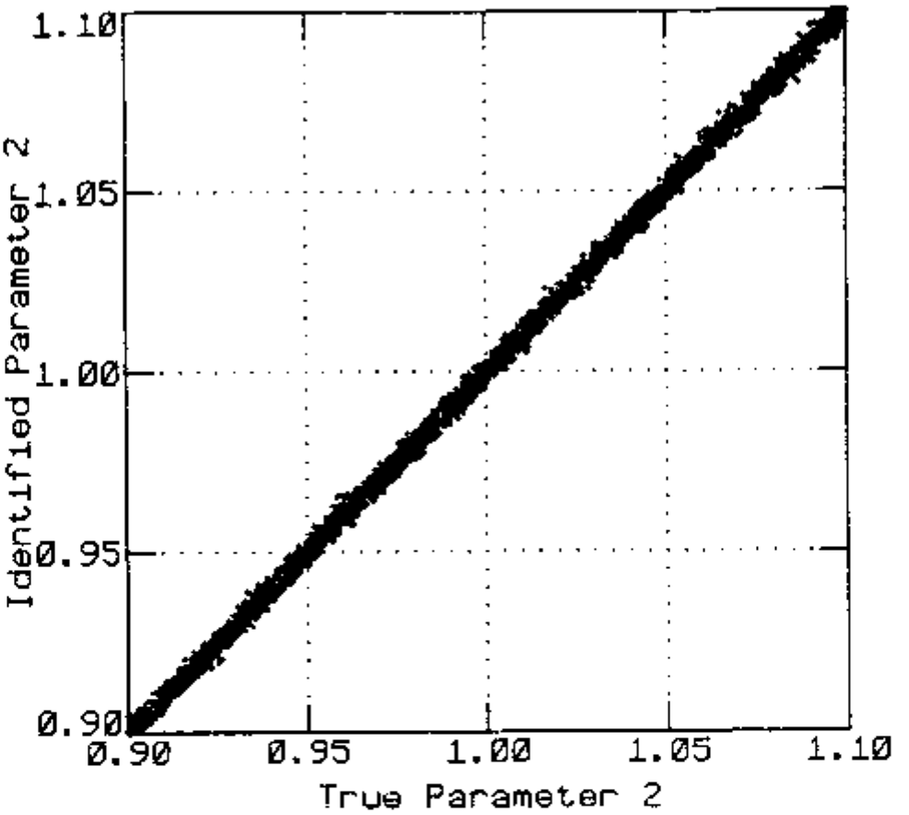

FIGURE 4-13. Plot of true-versus-identified Parameter 2.

functional surface is not unimodal, i.e. it exhibits more than one minimum.

Figure 4-14 is an isometric contour representation of the error functional surface for the PSD of the primary water temperature between 0.001 and 0.1 Hz as a function of the primary water mass flow rate and the overall heat transfer coefficient between the metal tubes and the secondary side. It can be seen that this surface has four minimum points in the range plotted. The true minimum occurs when both parameters equal one, which is the normalized reference value for each parameter. The local minima correspond to non-physical solutions where at least one of the two parameters is negative. Sometimes during the parameter identifications, the search converged to the local minimum with positive mass flow rate and negative heat transfer coefficient, but these solutions were rejected by the hypothesis tests. The parameter identification, in that case, was restarted until convergence to the true minimum was obtained.

Several tests were performed to verify the influence of various factors on the learning curve. Figures 4-15 through 4-17, for example, show the learning curves obtained under the conditions specified in Table 4-2, but for different numbers of directions, i.e. 3, 13 and 25, respectively. It should be noticed from the comparison of the three curves that (a) three directions are obviously too few, hence hardly any learning is achieved, (b) the 13-direction case showed a substantial improvement in both learning speed and final performance, and (c) the 25-direction case, although it showed a

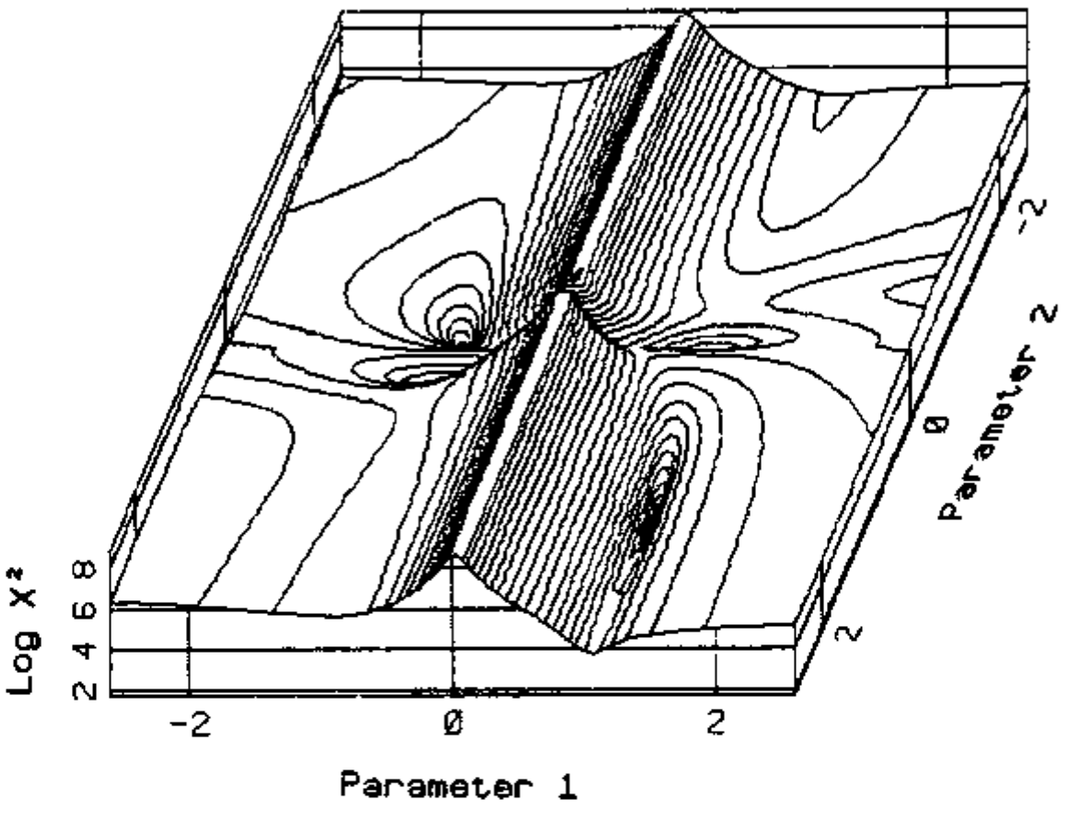

FIGURE 4-14. Isometric contour representation of the error functional surface (for one PSD).

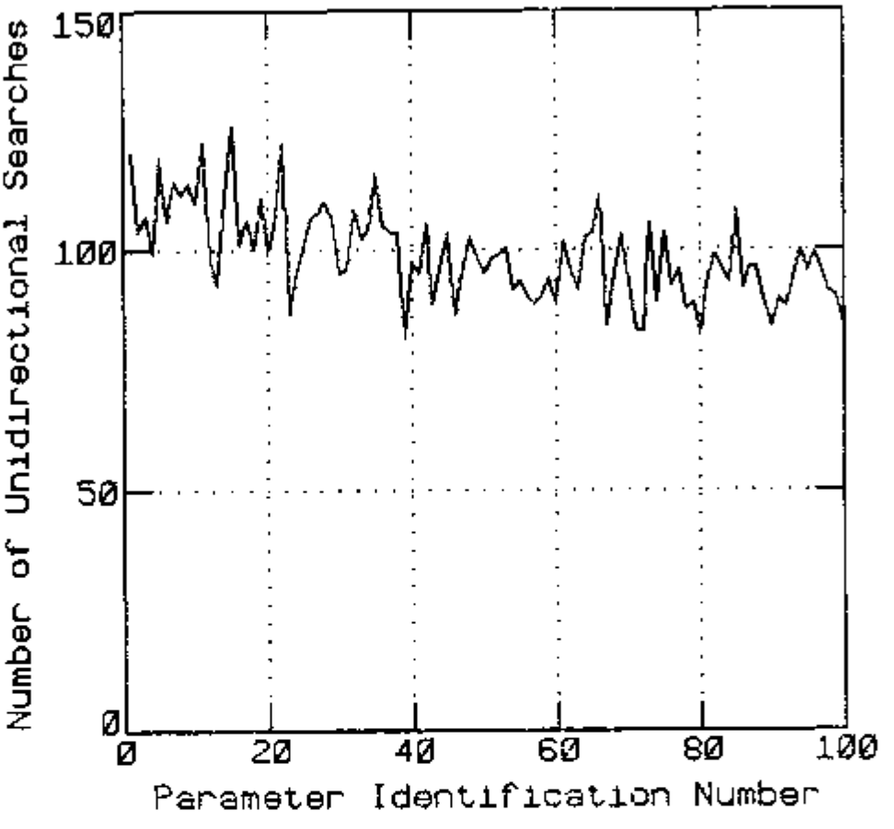

FIGURE 4-15. Learning Curve (FDL method; 3-directions).

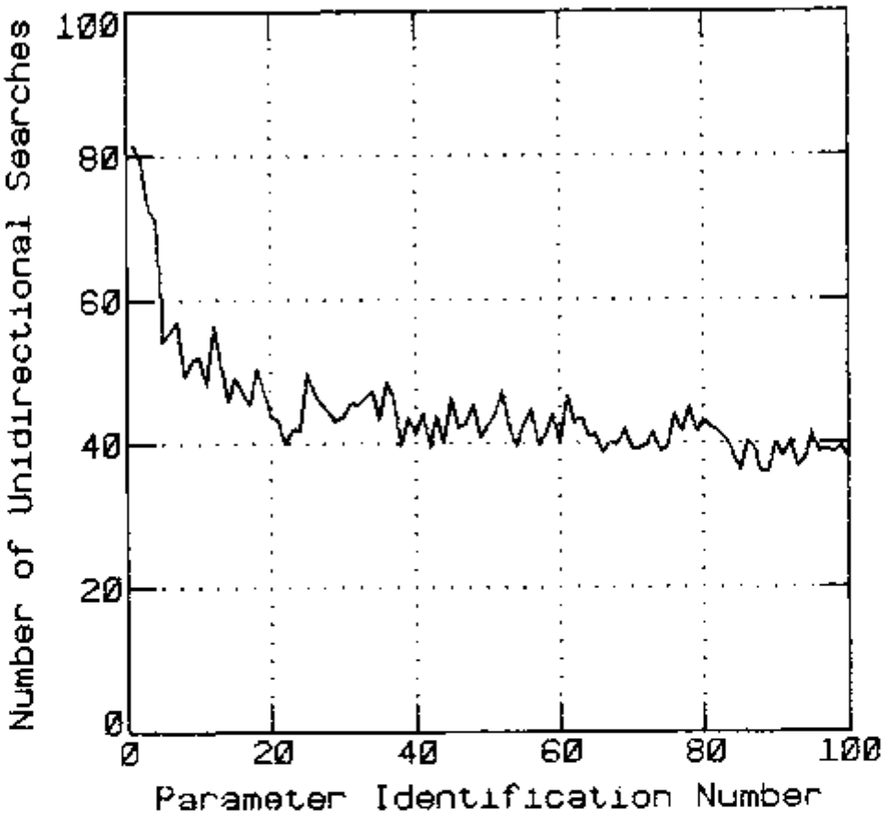

FIGURE 4-16. Learning Curve (FDL method; 13-directions).

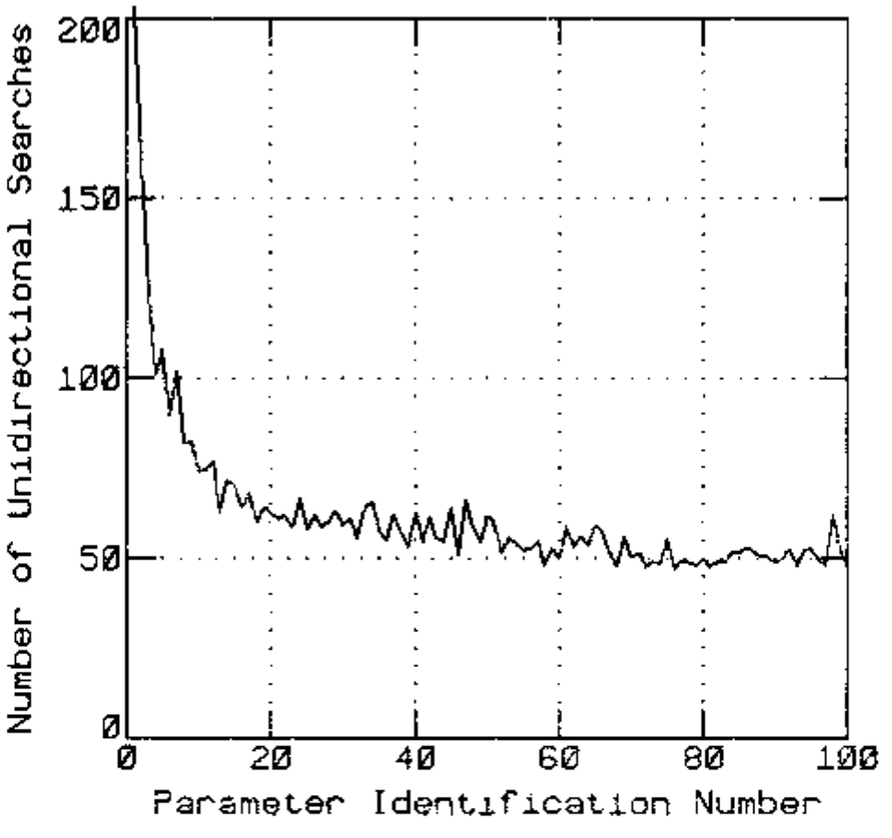

FIGURE 4-17. Learning Curve (FDL method; 25-directions).

TABLE 4-2. Simulation conditions used to generate the results presented in Figures 4-15, 4-16, 4-17 and 4-19.

Model	Steam generator model
Parameters identified	Primary water mass flow rate Metal-secondary heat transfer coefficient
True parameters interval (uniformly distributed)	0.5 to 1.5 (all parameters)
Initial parameter values	1.0 (all parameters)
Experimental points	Primary water temperature PS0
Number of frequency points	20
Frequency range (uniform)	0.001 Hz to .1 Hz
Experimental error	1%
Number of features	2
Region size	(radius) =0.0005 + 0.3(distance)
Number of directions	3, 13 and 25 (see text)
Precision criteria	0.0001
Number of parameter identifications per series	100
Number of series	50

longer learning period due to the increased number of directions and the resulting increased number of associated manipulations, produced better estimates of the parameters than the 13-direction case.

In the three parameter tests, the PSD of the variation from equilibrium of the steam mass flow rate was used as the third parameter identified. This PSD was considered independent of the frequency (white) in the frequency range used. A three dimensional feature vector was used in these tests, with the first two components given by Equations 4.7 and 4.8, and the third component given by

$$f_3 = \frac{4}{3A\bar{w}^2} \sum_{i=1}^{N} W_i^{1/2} (Y_i - F_i)(w_i - \bar{w})^2 \quad , \qquad 4\text{-}11$$

which is the second moment of the residuals. A set of simulations were performed to compare different strategies in the direction selection:

a. Normal selection. This is the selection strategy normaly used (see Section 3.3.4),

b. Random selection. In this case, any available direction can be selected with the same probability,

c. Sequential Selection. The directions are selected sequentially for this case, and

d. Best direction selection. In this strategy, a search is performed in each direction and the parameter values are returned to the initial value after each search. The direction for which the search results in the lowest error functional

is selected. For the purpose of the comparison below, the unidirectional searches performed to find the best direction are not counted.

The same series of parameter identifications were accomplished using each direction selection strategy under the conditions specified in Table 4-3. For the random selection strategy, the series of parameter identifications was performed twice with different random numbers for the direction selection. The results of these simulations are summarized in Figure 4-18.

As expected, on the average, the best direction selection presented the best performance followed by the normal selection (which should, in the ideal case, converge to best direction as more parameter identifications are performed). The sequential selection ranked next in performance and the random selection demonstrated the worst performance. However, it may happen that a sequence of randomly selected directions perform better than a sequence of best directions, as in parameter identification number 8 (see Figure 4-18). This behavior is due to the discretization of the directions, and it can appear at some points in the search path when none of the directions point from the search point toward the minimum.

Looking in detail at the directions selected during the parameter identifications for the best direction selection strategy, it was noticed that, for many parameter identifications, a sequence of three directions were consistently repeated. It was then hypothesized that the four dimensional error functional surface presented the equivalent

TABLE 4-3. Simulation conditions used to generate the results presented in Figure 4-18.

Model	Steam generator model
Parameters identified	Primary water mass flow rate, Metal-secondary heat transfer coefficient Steam flow PSD
True parameters interval (uniformly distributed)	0.9 to 1.1 (all parameters)
Initial parameter values	1.0 (all parameters)
Experimental points	Primary water temperature PSD
Number of frequency points	20
Frequency range (log-uniform)	0.001 Hz to .1 Hz
Experimental error	0.3%
Number of features	3
Region size	(radius) =0.003 + 0.3(distance)
Number of directions	13
Precision criteria	0.0001

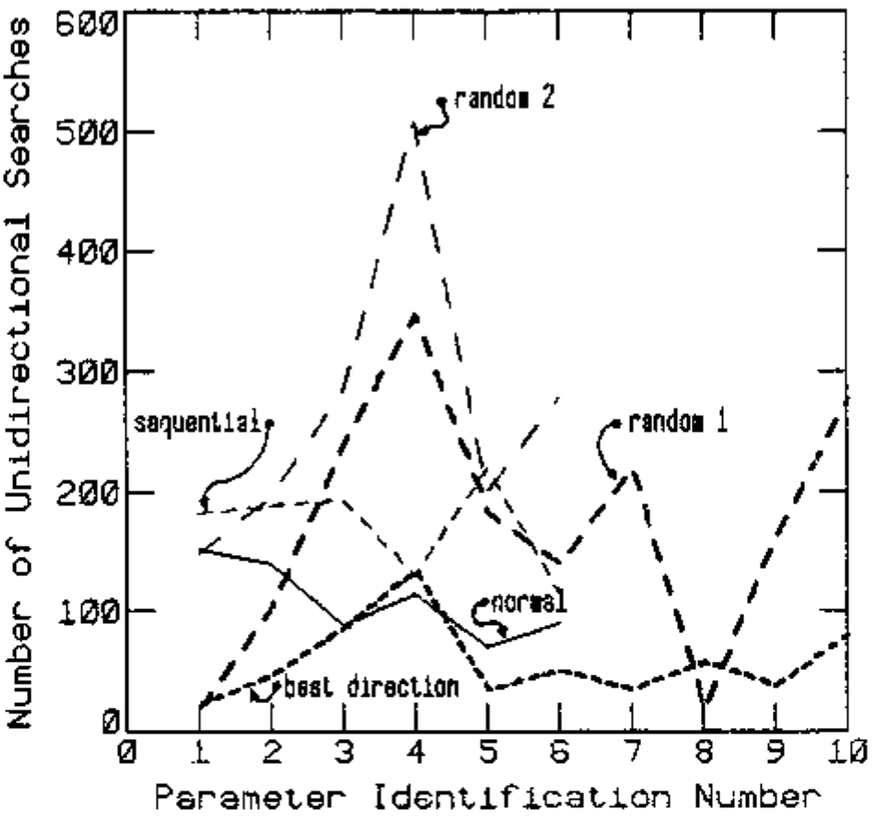

FIGURE 4-18. Comparison of several direction selection strategies (FDL method; conditions in Table 4-3).

of a valley in a three dimensional surface. This valley was not aligned with any of the available directions, so that the search had to proceed in small steps.

To test this hypothesis, the average direction, after several sequences had repeated, was calculated. This extra direction was added to the set of 13 directions, and the parameter identifications were repeated. After adding the extra direction, no more sequence repetition was observed, and the number of unidirectional searches decreased drastically.

To quantify this improvement, a series of 100 parameter identifications were performed using the normal selection strategy, both with 13 directions and with 14 directions. The average number of unidirectional searches required for convergence in the last 10 parameter identifications were: 156 for the 13 directions, and 32 for the 14 directions. This vast improvement confirmed the hypothesis set forth previously and triggered the idea of the SDL method.

4.6 Single-Direction Learning (SDL) Method Results

In this section, some results of the SDL method for the steam generator model are presented. Some of the parameter identifications solved by the FDL method were also solved with the SDL method, so that a comparison of the two learning methods could be performed. A comparison with the Direct Fit method of Hooke and Jeeves (12) is also presented.

4.6.1 Results with the steam generator model

The set of parameter identifications for the conditions specified in Table 4-2 were repeated using the SDL method with three basic directions. The learning curve obtained is shown in Figure 4-19. This learning curve can be compared directly with the learning curves for the FDL method shown in Figures 4-15 through 4-17, which were obtained under the same conditions. For the best of the three cases with the FDL method (13-directions; see Figure 4-16), the average number of unidirectional searches, for parameter identifications 91 through 100, is 39. The corresponding number for the SDL method is 13, which gives a reduction of approximately a factor of three in the number of required unidirectional searches.

In Figure 4-20, a learning curve is shown for a 3-parameter identification problem obtained under the conditions specified in Table 4-4. A four dimensional feature vector was used whose components were the first four normalized moments of the residuals, given by

$$f_j = \left[\sum_{i=1}^{N} (w_i-\bar{w})^{2(j-1)}\right]^{-1/2} \sum_{i=1}^{N} W_i^{1/2}(Y_i-F_i)(w_i-w)^{j-1} \quad , \qquad 4\text{-}12$$

for $j = 1, 2, 3$ and 4.

These features are normalized so that the variance at the convergence point is one.

Another series of tests was performed using as experimental data two noise descriptors: the PSD of the primary water temperature and

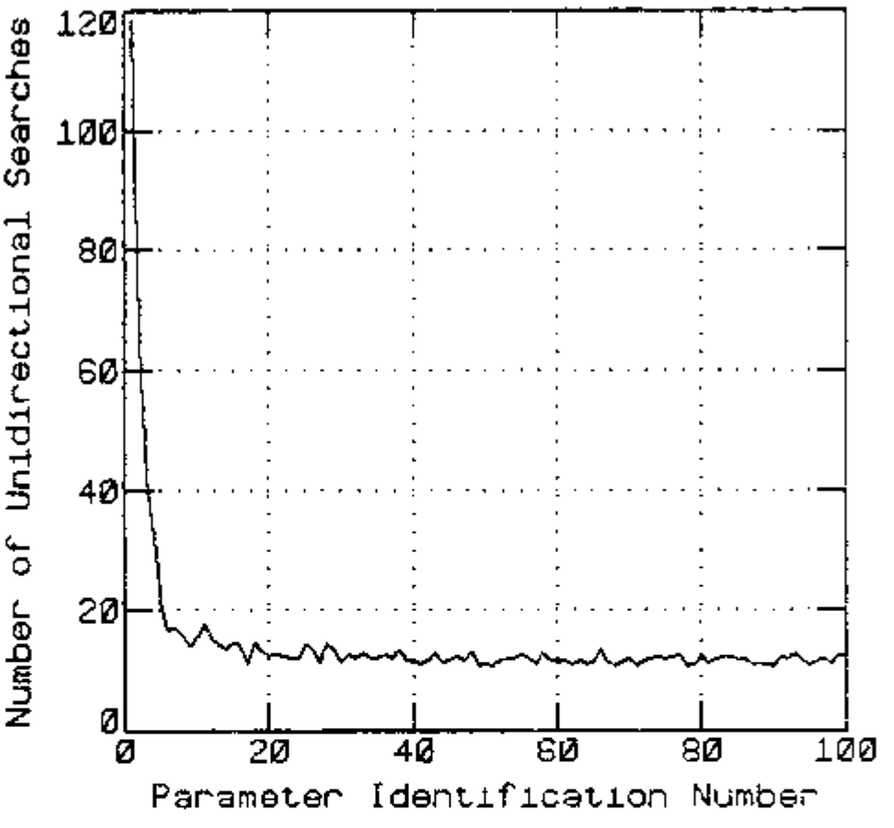

FIGURE 4-19. Learning Curve (SDL method; conditions in Table 4-2).

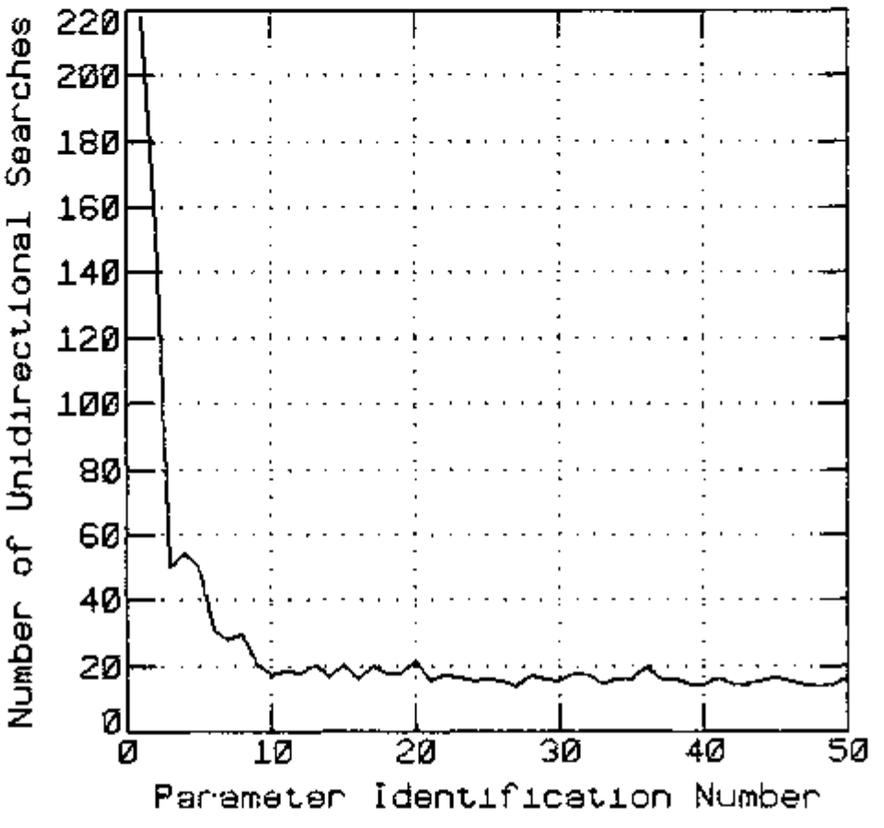

FIGURE 4-20. Learning Curve (SDL method; conditions in Table 4-4 with one PSD).

TABLE 4-4. Simulation conditions used to generate the results presented in Figures 4-20, 4-22 and 4-23.

Model	Steam generator
Parameters identified	Primary water mass flow rate Metal-secondary heat transfer coefficient Steam flow PSD
True parameters interval (uniformly distributed)	0.9 to 1.1 (all parameters)
Initial parameter values	1.0 (all parameters)
Experimental points	Primary water temperature PSD (and steam pressure PSD)
Number of frequency points	20
Frequency range (log-uniform)	0.001 Hz to 0.1 Hz
Experimental error	0.3%
Number of features	4
Region size	(radius) =0.5 + 0.5(distance)
Number of basic directions	3
Precision criteria	0.0001
Number of parameter identifications per series	50
Number of series	50

the PSD of the steam pressure. The minimization of the combined squared error of both PSDs improves the identifiability of the parameters. For two parameters, this fact can be visualized by plotting the error functional surface. In Figure 4-21, the combined error functional surface is shown as a function of the parameters: the PSD of primary water mass flow rate and the metal-secondary mass heat transfer coefficient. Comparing this surface with the one associated with the PSD of the primary water temperature only (see Figure 4-14), it can be noticed that, using the two PSDs, a new ridge appeared, separating the true minimum from the local minima. Hence, the likelihood of convergence to the local minima is decreased.

In Figure 4-22, a learning curve is shown for the case of two PSDs and three parameters. The other conditions are the same as the ones specified in Table 4-4. A four dimensional feature vector was used, where the features were the zeroth and the first normalized moments of each PSD.

It can be seen, comparing Figures 4-20 and 4-22, that minimizing the error in both PSDs simultaneously significantly decreased the required number of unidirectional searches per parameter identification. This decrease is predominant for the first parameter identifications, i.e. before a significant learning is achieved. This means that for the case of one PSD only, the parameter identification is more difficult to perform. However, the SDL method still achieves comparable asymptotic numbers of unidirectional searches in both cases.

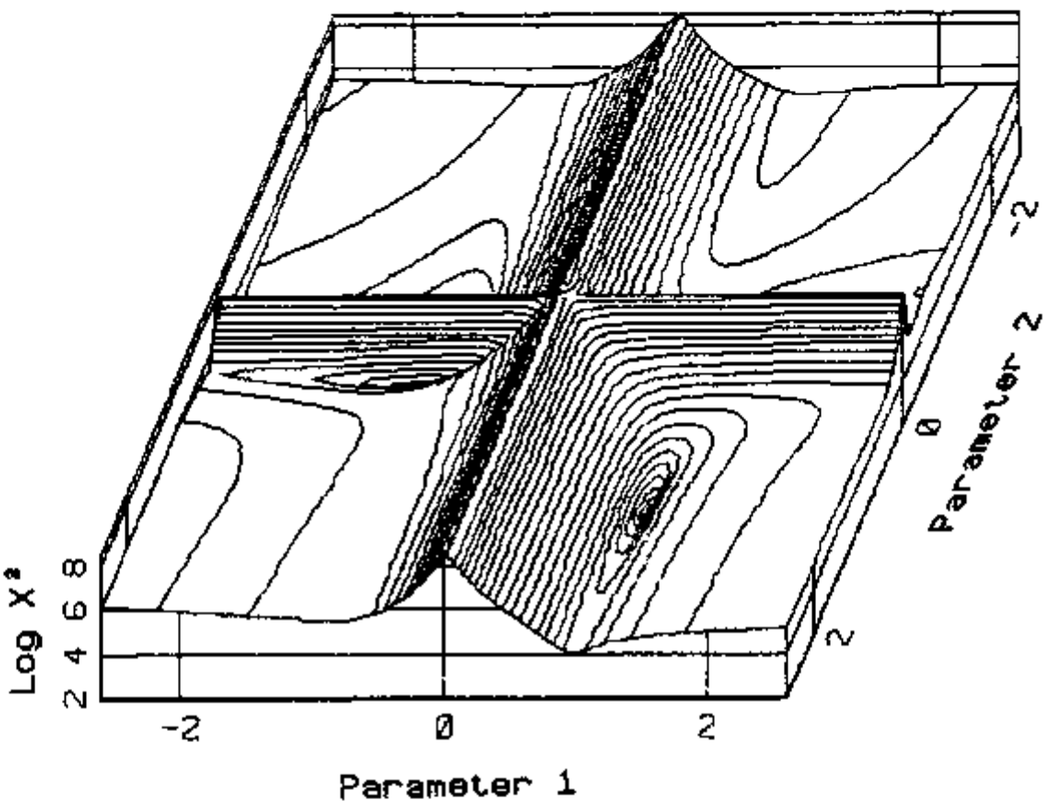

FIGURE 4-21. Isometric contour representation of the error functional surface (for two PSDs).

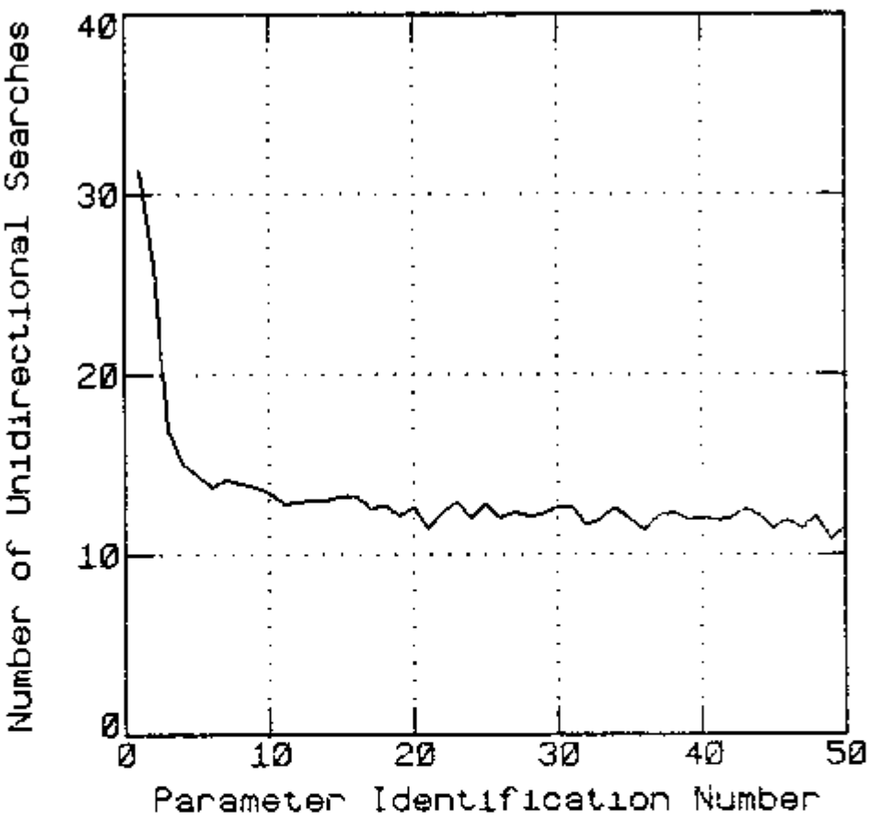

FIGURE 4-22. Learning Curve (SDL method; conditions in Table 4-4 with two PSDs).

4.6.2 Comparison with the Direct Search method

Comparison of the SDL method with the Direct Search method (12) was made by solving some series of parameter identifications using both methods under the same conditions, i.e. the same initial value for the parameters and the same accuracy in the determination of the optimum parameters. The comparison was done for several cases using the steam generator model under the conditions specified in Table 4-4. Either the primary water temperature PSD only or the primary water temperature PSD and the steam pressure PSD were used as experimental data. The following cases were used:

1. Primary water temperature PSD as experimental data, and initial value for all parameters equal to 1.0.
2. Primary water temperature PSD as experimental data, and initial value for all parameters equal to 4.0.
3. Primary water temperature PSD as experimental data, and initial value for the parameters: parameter one equal to 2.0, parameter two equal to 3.0, and parameter three equal to 4.0.
4. Primary water temperature PSD and steam pressure PSD as experimental data, and initial value for all parameters equal to 1.0.
5. Primary water temperature PSD and steam pressure PSD as experimental data, and initial value for all parameters equal to 5.0.

In these comparisons, the total number of error functional evaluations

was used as a common measure of computational time.

The number of error functional evaluations required by each method for a series of 50 parameter identifications of Case 1 is shown in Figure 4-23. It can be seen from this figure that even though the Direct Search method required less functional evaluations than the SDL method during its learning period (first five parameter identifications), on the average, the SDL method required considerably less error functional evaluations than the Direct Search method. For instance, the average number of error functional evaluations for parameter identifications 21 through 50 was: 46 for the SDL method, and 148 for the Direct Search method.

A series of 50 parameter identifications were solved by both methods also for the other four cases. The average number of error functional evaluations required by each method to perform the parameter identifications 21 through 50 of each series is given in Table 4-5. It can be seen in this table that the SDL method required less error functional evaluations for all cases. The advantage of the SDL method over the Direct Search method varied from a factor of three, for Case 4, to a factor of 10, for Case 5.

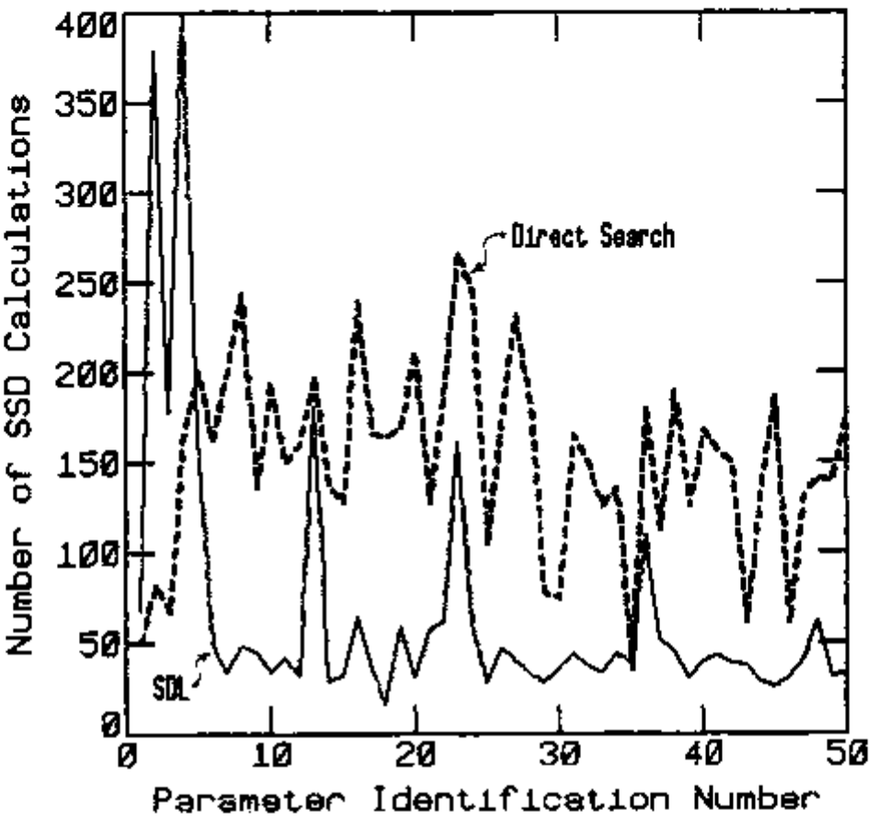

FIGURE 4-23. Number of sum-of-the-squares-of-deviations (SSD) calculations per parameter identification by the Direct Search and by the SDL methods.

TABLE 4-5. Comparison between the Single-Direction Learning (SDL) method and the Direct Search method.

Case Number	CONDITIONS (other conditions in Table 4-4)		Number of error functional evaluations	
	Number of PSDs as experimental points	parameter initial values	SDL	Direct Search
1	1	1, 1, 1	46.2	147.8
2	1	4, 4, 4	65.3	234.2
3	1	2, 3, 4	44.3	275.5
4	2	1, 1, 1	34.3	99.9
5	2	5, 5, 5	41.0	390.9

CHAPTER 5

APPLICATION OF THE HEURISTIC LEARNING METHOD TO THE DIAGNOSTICS OF AN EXPERIMENTAL PRESSURE LOOP

5.1 Introduction

After the development and the testing of the learning methods using computer simulated data, a "real world" application was achieved by developing an automatic surveillance system for diagnostics of a experimental pressure loop.

The description of the pressure loop is given in Section 5.2, and the model for the loop pressure noise is described in Section 5.3. In Section 5.4, an overview of the automatic surveillance system is given. The results obtained are presented in Section 5.5.

5.2 Description of the Experimental Pressure Loop

The pressure loop is a small experimental device designed to study pressure noise, pressure sensor responses and sensing line responses. Although the loop is quite scaled down and operating at low pressure (typically 15 psi), it shows most of the pressure noise characteristics observed in the primary system of PWR plants.

A diagram of the experimental loop is shown in Figure 5-1, where the components were given labels corresponding to the analogous PWR components. The reactor vessel and the pressurizer are plexiglass tanks, and the lines are copper tubes. The dimensions of these components are given in Table 5-1. The pump shown is a centrifugal pump driven by a 1/2 HP eletric motor. The three pressure sensors are wide band pressure sensors, Validyne model DP-7. These sensors present flat responses in the range of frequencies used in these experiments (below 10 Hz).

In Figure 5-2 through 5-7, the PSDs of the three pressure signals and the CPSDs between these signals are shown. A resonance at 1.2 Hz is observed in those spectral densities, which is the Helmholtz resonance of the pressurizer--surge line--reactor vessel system. It arises from the oscillation of the water in and out of the pressurizer through the pressurizer surge line. This resonance can be modeled in terms of the oscillations of a spring loaded oscillator, whose inertial mass is the mass of water in the surge line, with an spring constant determined by the compressibility of the water in the reactor vessel. This type of resonance is observed in PWR plants in the range 0.5 Hz to 1.1 Hz (27). A similar resonance is observed at 3.9 Hz, which is the Helmholtz resonance of the sensing line. This type of resonance has also been observed in PWR plants (27).

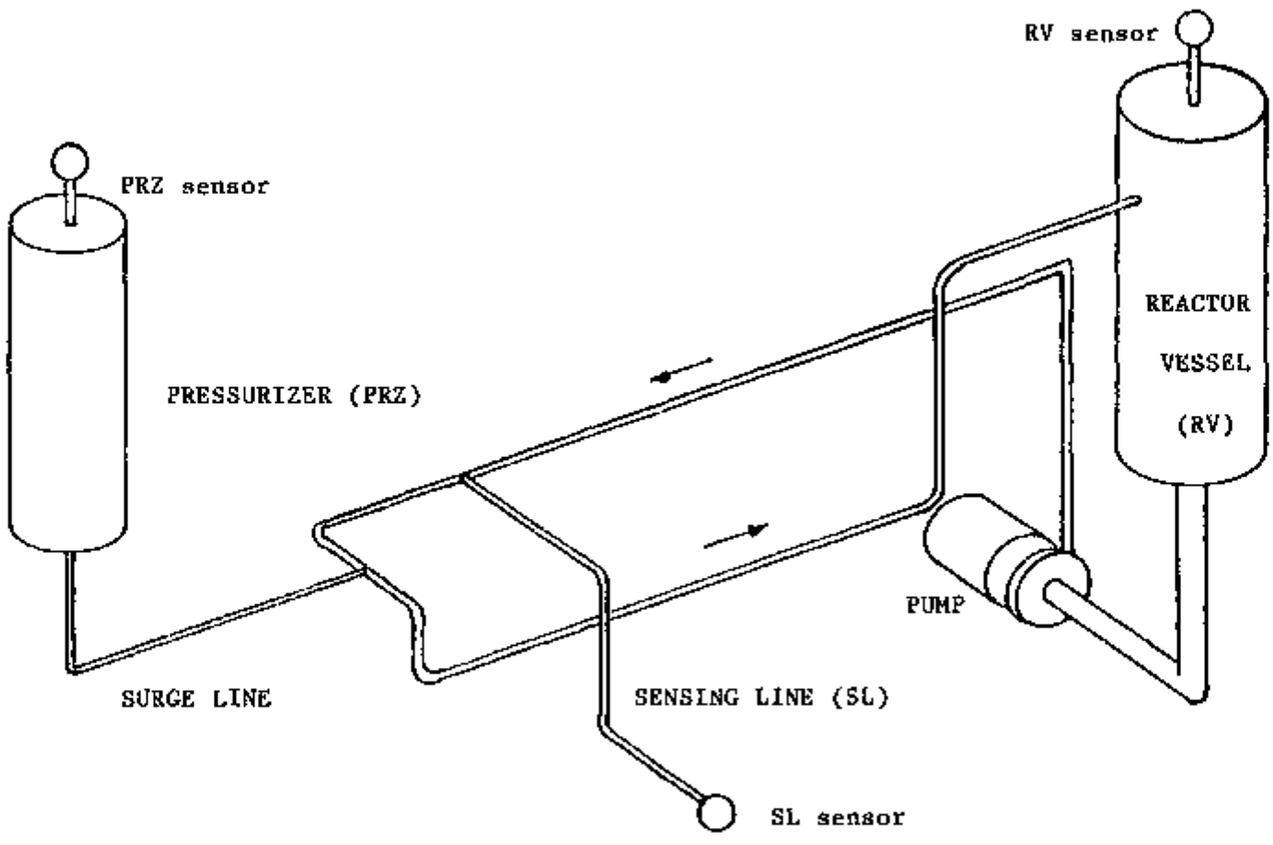

FIGURE 5-1. Experimental Pressure Loop

TABLE 5-1. Dimensions of the experimental pressure loop components.

Component	Dimensions		
	type	value	unit
Reactor vessel	diameter:	0.548	ft
	height:	2.417	ft
Pressurizer	diameter:	0.333	ft
	total height:	1.687	ft
	water level:	0.917	ft
Sensing line	area:	$2.01*10^{-4}$	ft^2
	length:	17.58	ft
Surge line	area:	$2.01*10^{-4}$	ft^2
	length:	3.46	ft
Flow lines:			
pump to sensing line connection	area:	$1.115*10^{-3}$	ft^2
	length:	7.08	ft
sensing line connection to surge line connection	area:	$1.115*10^{-3}$	ft^2
	length:	2.21	ft
surge line connection to reactor vessel	area:	$1.115*10^{-3}$	ft^2
	length:	7.50	ft
reactor vessel to pump	area:	$3.588*10^{-3}$	ft^2
	length:	2.58	ft

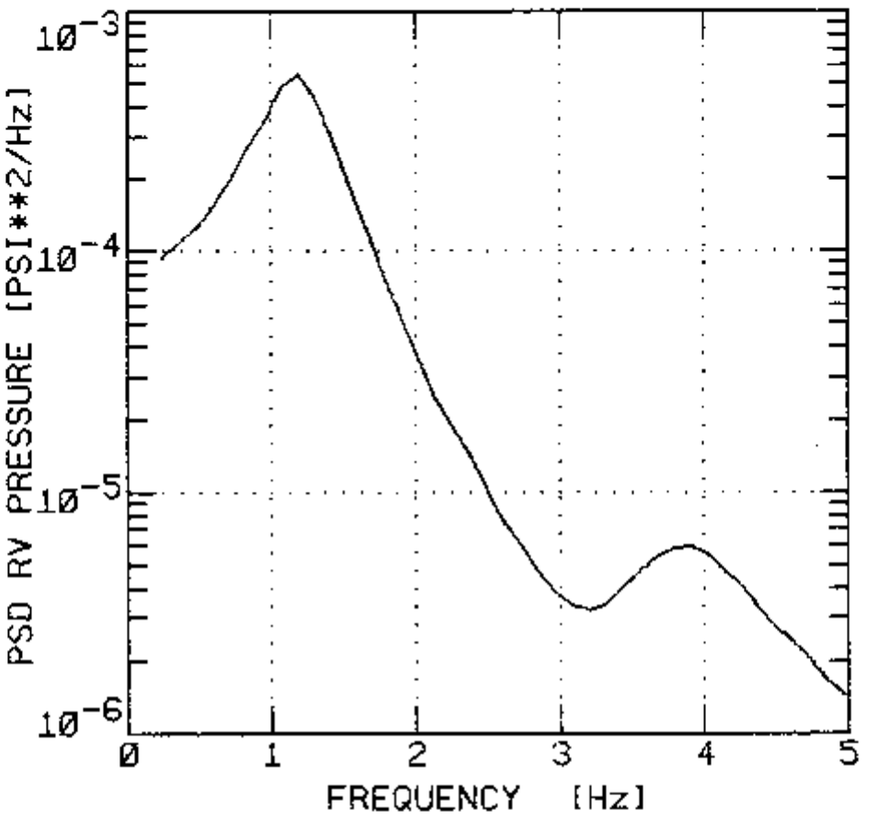

FIGURE 5-2. Experimental Reactor Vessel pressure PSD.

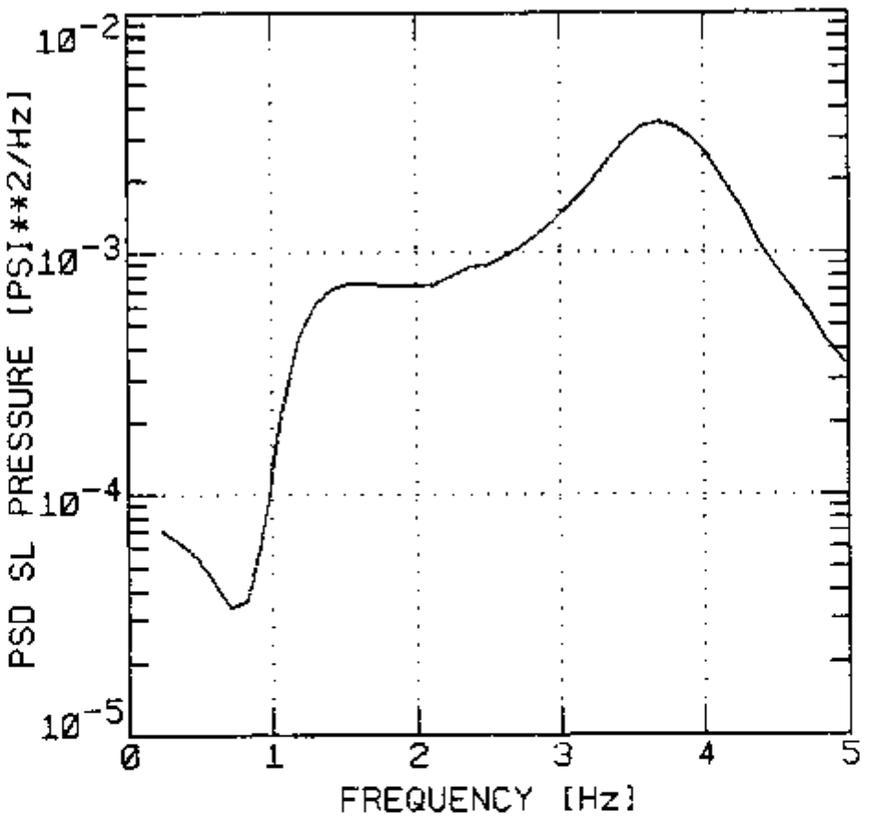

FIGURE 5.3. Experimental Sensing Line pressure PSD.

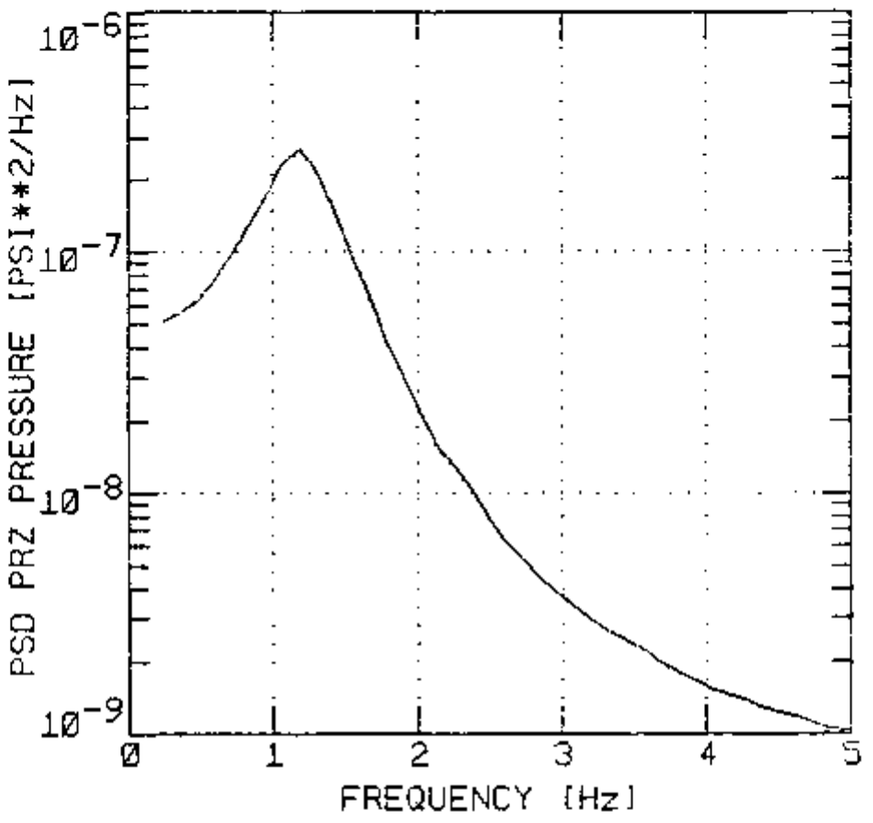

FIGURE 5-4. Experimental Pressurizer pressure PSD.

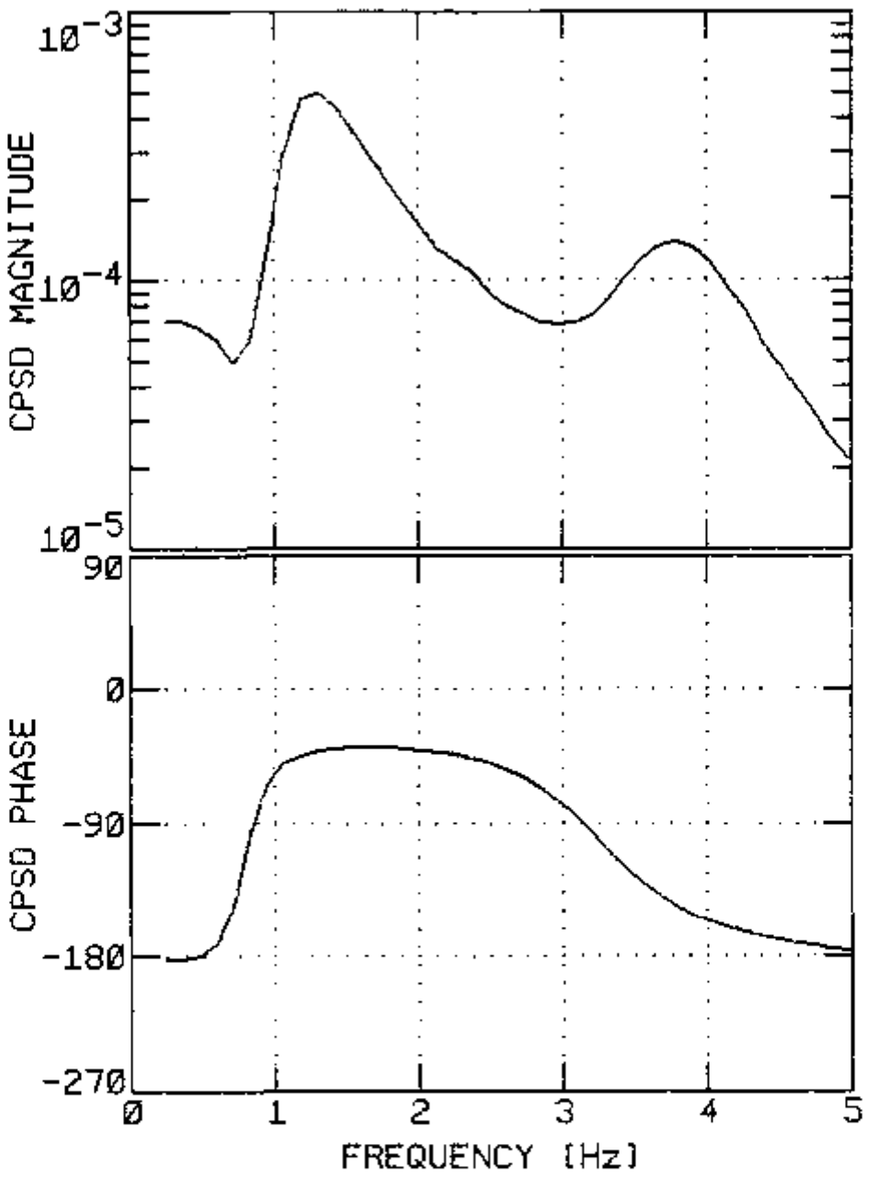

FIGURE 5-5. Experimental Reactor Vessel - Sensing Line CPSD.

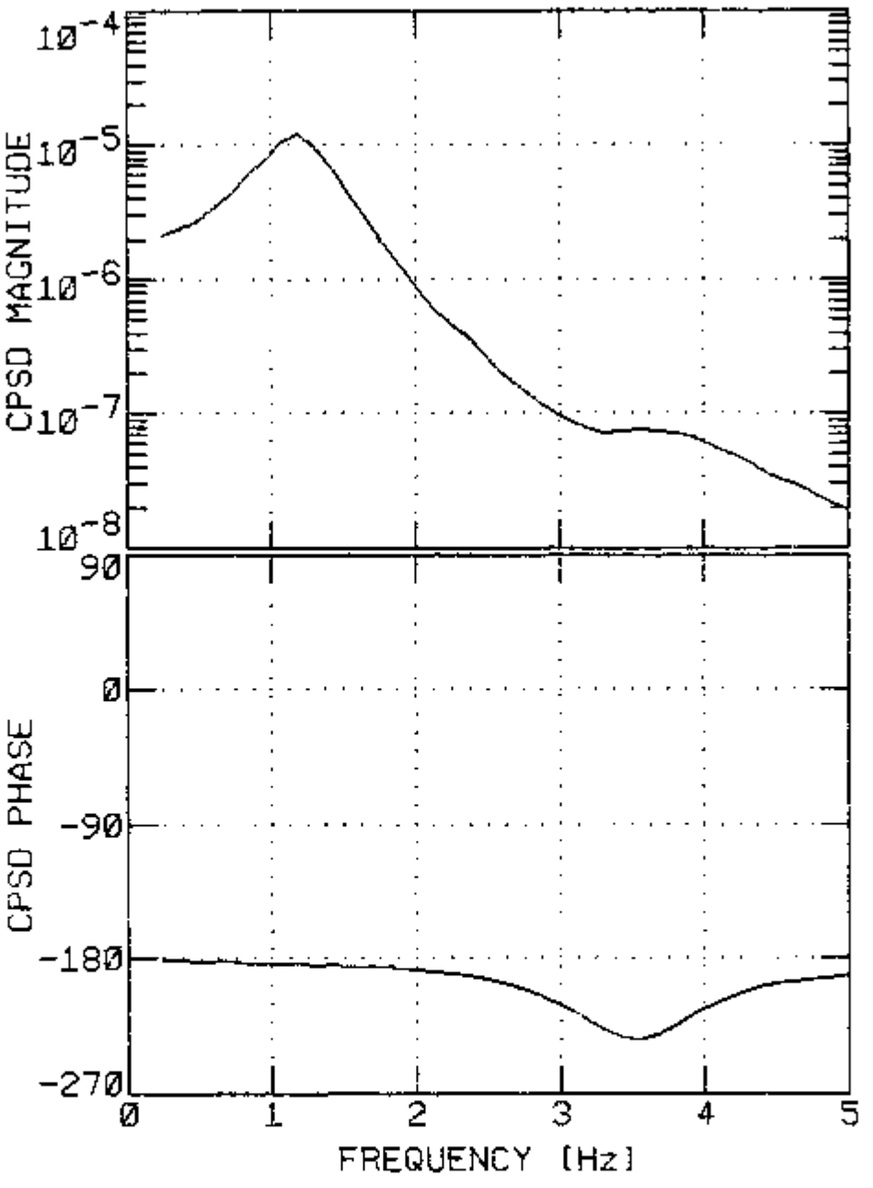

FIGURE 5-6. Experimental Reactor Vessel - Pressurizer CPSD.

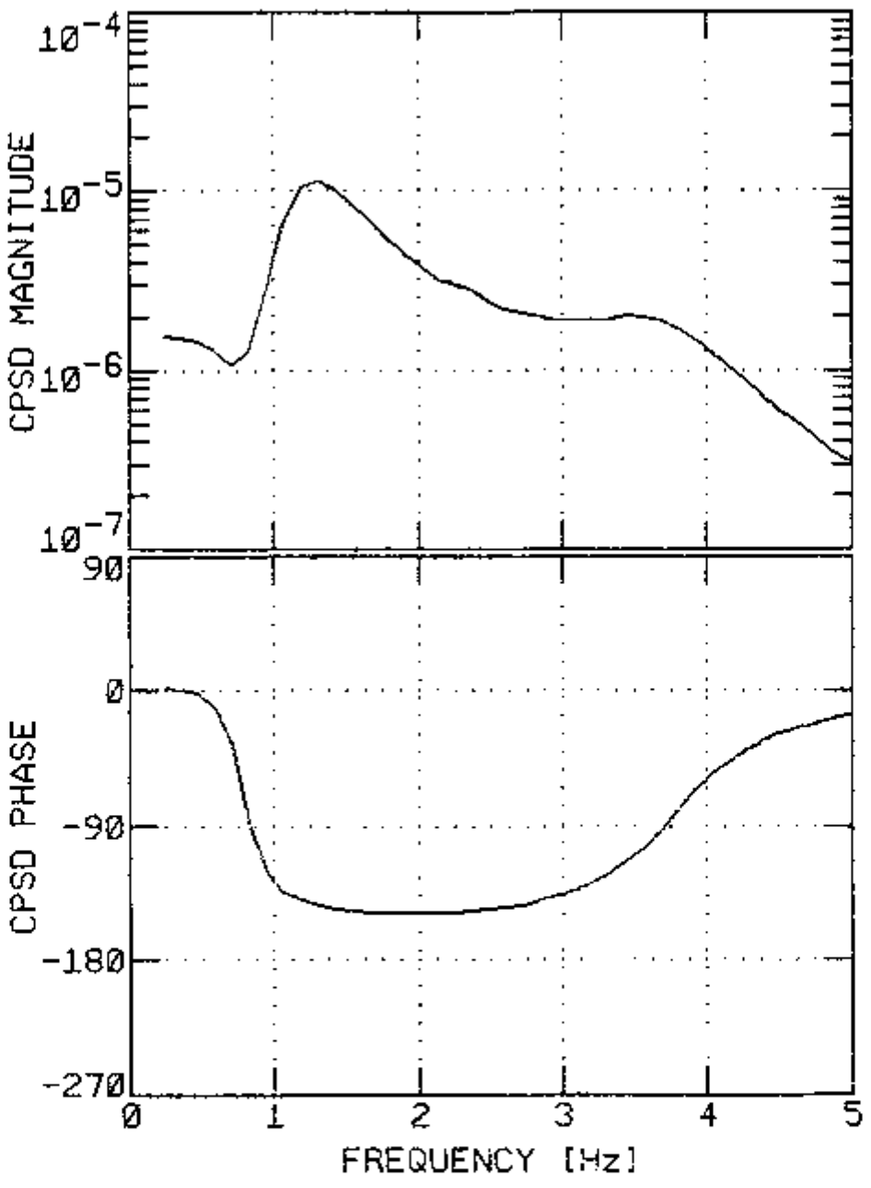

FIGURE 5-7. Experimental Sensing Line - Pressurizer CPSD.

5.3 Loop Pressure-Noise Model

The model for the pressure noise at low frequencies used in this work was developed by Mullens and Thie (27). In this model, the water in the lines was considered incompressible, and the main physical phenomenon for the pressure noise in the tanks was considered to be the compressibility of the fluid inside. The electric analog circuit representing the pressure loop model is shown in Figure 5-8, where the inertia of the water inside the lines is analogous to inductance, the compressibility of the fluid in the the tanks is analogous to capacitance, and the deviation from equilibrium of the volumetric water flow is analogous to current.

The inductances of the lines are given by (34)

$$L_i = \frac{\rho_i \ell_i}{A_i} \quad , \qquad 5\text{-}1$$

where ρ_i is the density of the fluid inside line "i",

ℓ_i is the length of line "i", and

A_i is the flow area of line "i".

The capacitances of the fluid inside the tanks are given by (34)

$$C_i = \frac{V_i}{\rho_i v_i^2} \quad , \qquad 5\text{-}2$$

where V_i is the volume of tank "i", and

v_i is the speed of sound in the fluid inside tank "i".

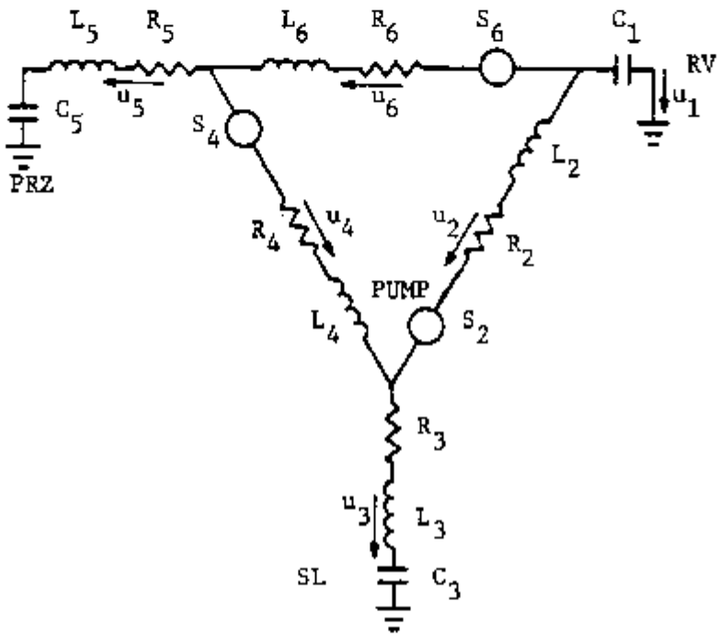

FIGURE 5-8. Electric analog model of the preasure loop noise.

In Table 5-2, the values for all the pressure loop model parameters are given. The inductances, and the capacitance of the pressurizer tank were calculated using Equations 5-1 and 5-2, respectively. The remaining parameters were obtained from a least square fit of the model to the measured spectral densities.

The equations for the analog circuit in Figure 5-8 are obtained using the classical Kirchhoff electric network laws. Applying the first law to the three nodes yields the following equations:

$$X_1(s) + X_2(s) + X_6(s) = 0 \quad , \tag{5-3}$$

$$X_2(s) - X_3(s) + X_4(s) = 0 \quad , \text{ and} \tag{5-4}$$

$$X_4(s) + X_5(s) - X_6(s) = 0 \quad , \tag{5-5}$$

where $X_i(s)$ is the Laplace transform of the deviation from equilibrium of the volumetric flow rate in line "i".

The application of the second Kirchhoff law to the closed path formed by lines 1, 2 and 3 (see Figure 5-8), with the path closed through the ground, yields

$$- X_1(s)Z_1(s) + X_2(s)Z_2(s) + X_3(s)Z_3(s) - u_2(s) = 0 \quad , \tag{5-6}$$

where $Z_i(s)$ is the complex impedance of line "i", given in general by

$$Z_i(s) = sL_i + R_i + 1/(sC_i) \quad . \tag{5-7}$$

The corresponding equations for the closed paths formed by lines 3, 4 and 5 , and by the lines 1, 5 and 6 are given, respectively, by

TABLE 5-2. Experimental pressure loop parameter values.

Parameter[1]	value	Parameter[2]	value	Parameter[3]	value
-	-	C_1	$1.24*10^{-8}$	-	-
L_2	$4.41*10^{5}$	-	-	R_2	$3.75*10^{6}$
L_3	$5.46*10^{6}$	C_3	$3.20*10^{-1}$	R_3	$3.27*10^{7}$
L_4	$1.24*10^{5}$	-	-	R_4	$2.57*10^{6}$
L_5	$1.07*10^{6}$	C_5	$5.74*10^{-7}$	R_5	$2.79*10^{6}$
L_6	$4.20*10^{3}$	-	-	R_6	$2.71*10^{7}$

[1]Units for inductances are: $lbm.ft^{-4}$.

[2]Units for capacitances are: $ft^{4}.s^{2}.lbm^{-1}$.

[3]Units for resistances are: $lbm.ft^{-4}.s^{-1}$.

$$X_3(s)Z_3(s) + X_4(s)Z_4(s) - X_5(s)Z_5(s) - u_4(s) = 0 \quad , \text{ and} \qquad 5\text{-}8$$

$$- X_1(s)Z_1(s) + X_5(s)Z_5(s) + X_6(s)Z_6(s) - u_6(s) = 0 \quad . \qquad 5\text{-}9$$

The signals measured are the deviations from equilibrium of the pressure in the reactor vessel, δP_1; of the sensing line pressure, δP_3; and of the pressurizer pressure, δP_5. These pressures are related to the corresponding currents in the capacitors by the following expressions:

$$X_1(s) = sC_1\delta P_1(s) \quad , \qquad 5\text{-}10$$

$$X_3(s) = sC_3\delta P_3(s) \quad , \text{ and} \qquad 5\text{-}11$$

$$X_5(s) = sC_5\delta P_5(s) \quad . \qquad 5\text{-}12$$

By substituting Equations 5-10 through 5-12 into Equations 5-3 through 5-6 and into Equations 5-8 and 5-9, and then eliminating $X_2(s)$, $X_4(s)$ and $X_6(s)$, one obtains

$$B(s) \begin{bmatrix} \delta P_1(s) \\ \delta P_2(s) \\ \delta P_3(s) \end{bmatrix} = H(s) \begin{bmatrix} \delta u_1(s) \\ \delta u_2(s) \\ \delta u_3(s) \end{bmatrix} \quad , \qquad 5\text{-}13$$

where B and H are complex matrices with the non-zero elements given by

$$b_{11}(s) = -sC_1[Z_1 + Z_6 + Z_1Z_6/Z_2] \quad , \qquad 5\text{-}14$$

$$b_{12}(s) = sC_3Z_3Z_6/Z_2 \quad , \qquad 5\text{-}15$$

$$b_{13}(s) = sC_5Z_5 \quad , \tag{5-16}$$

$$b_{21}(s) = sC_1Z_1Z_4/Z_2 \quad , \tag{5-17}$$

$$b_{22}(s) = sC_3[Z_3 + Z_4 + Z_3Z_4/Z_2] \quad , \tag{5-18}$$

$$b_{23}(s) = -sC_5Z_5 \quad , \tag{5-19}$$

$$b_{31}(s) = sC_1 \quad , \tag{5-20}$$

$$b_{32}(s) = sC_3 \quad , \tag{5-21}$$

$$b_{33}(s) = sC_5 \quad , \tag{5-22}$$

$$h_{11}(s) = Z_6/Z_2 \quad , \tag{5-23}$$

$$h_{13}(s) = 1 \quad , \tag{5-24}$$

$$h_{21}(s) = Z_4/Z_2 \quad , \text{ and} \tag{5-25}$$

$$h_{22}(s) = 1 \quad . \tag{5-26}$$

Equation 5-13 is in the same format as Equation 2-1, the general noise model equation. Therefore, Equation 2-5 can be used directly to calculate the spectral densities for the loop model.

This model, although simple, predicts very well the loop pressure noise in the range from 0 to 5 Hz. In Figures 5-9 through 5-14, the model calculated PSDs and CPSDs of the three pressure signals are shown. These spectral densities were evaluated with the parameter values given in Table 5-2. The comparison of these calculated spectral densities with the corresponding experimental spectral

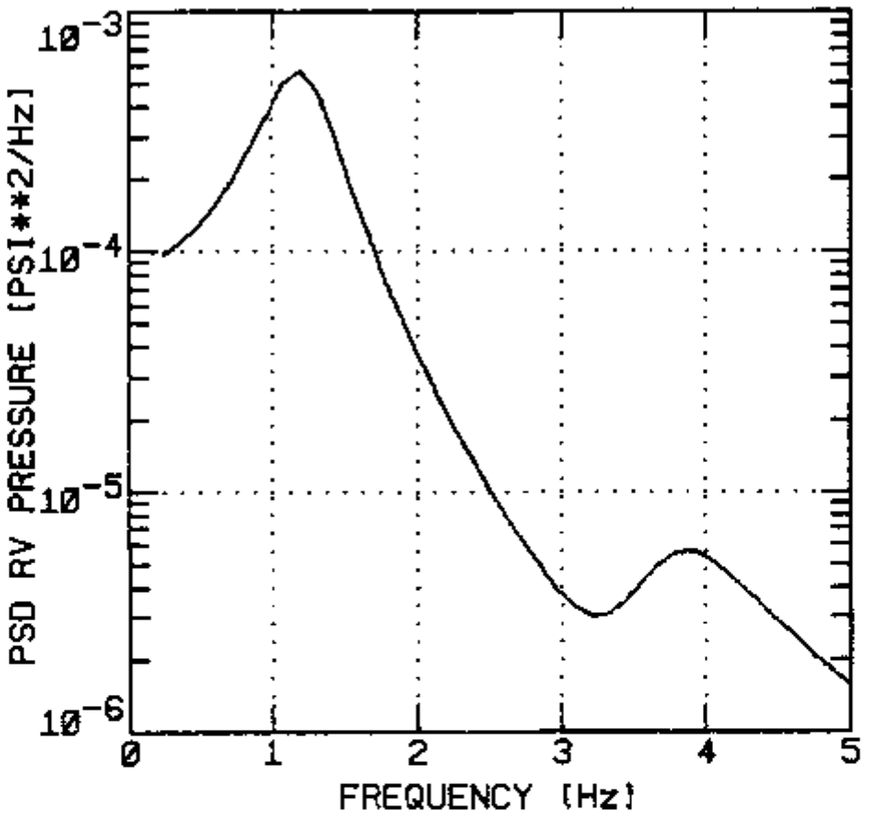

FIGURE 5-9. Calculated Reactor Vessel pressure PSD.

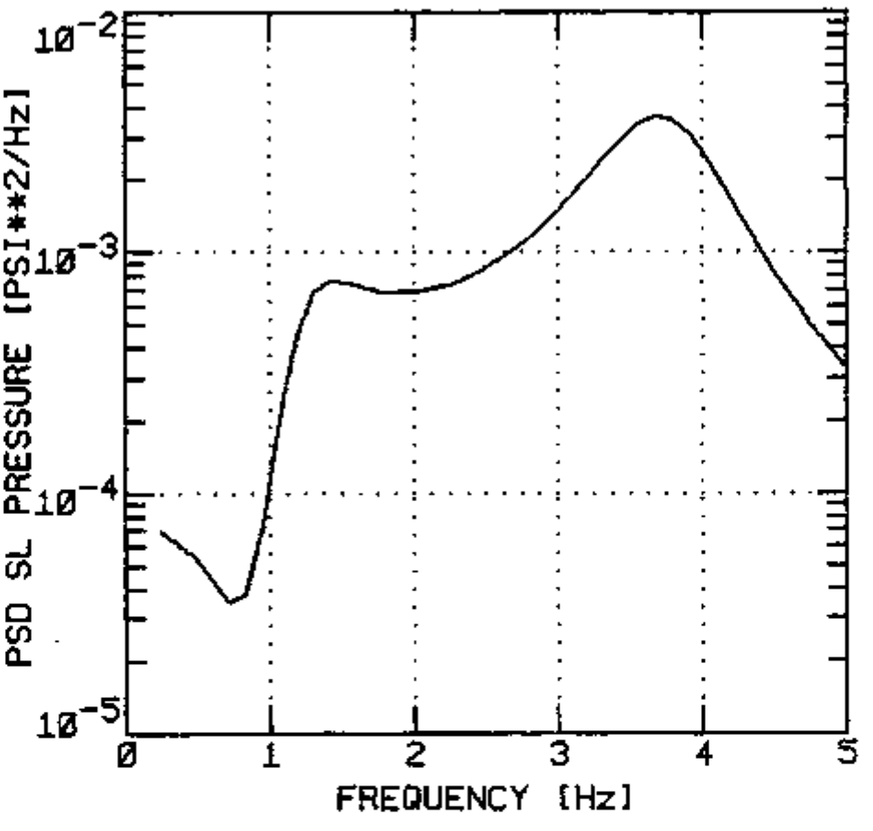

FIGURE 5-10. Calculated Sensing Line pressure PSD.

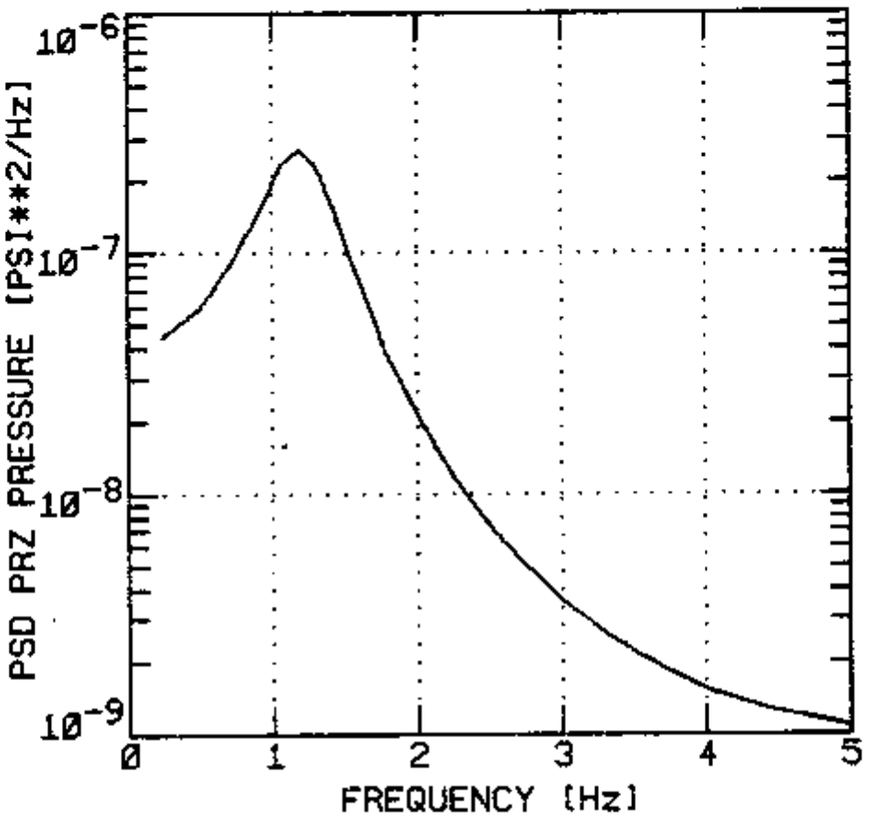

FIGURE 5-11. Calculated Pressurizer pressure PSD.

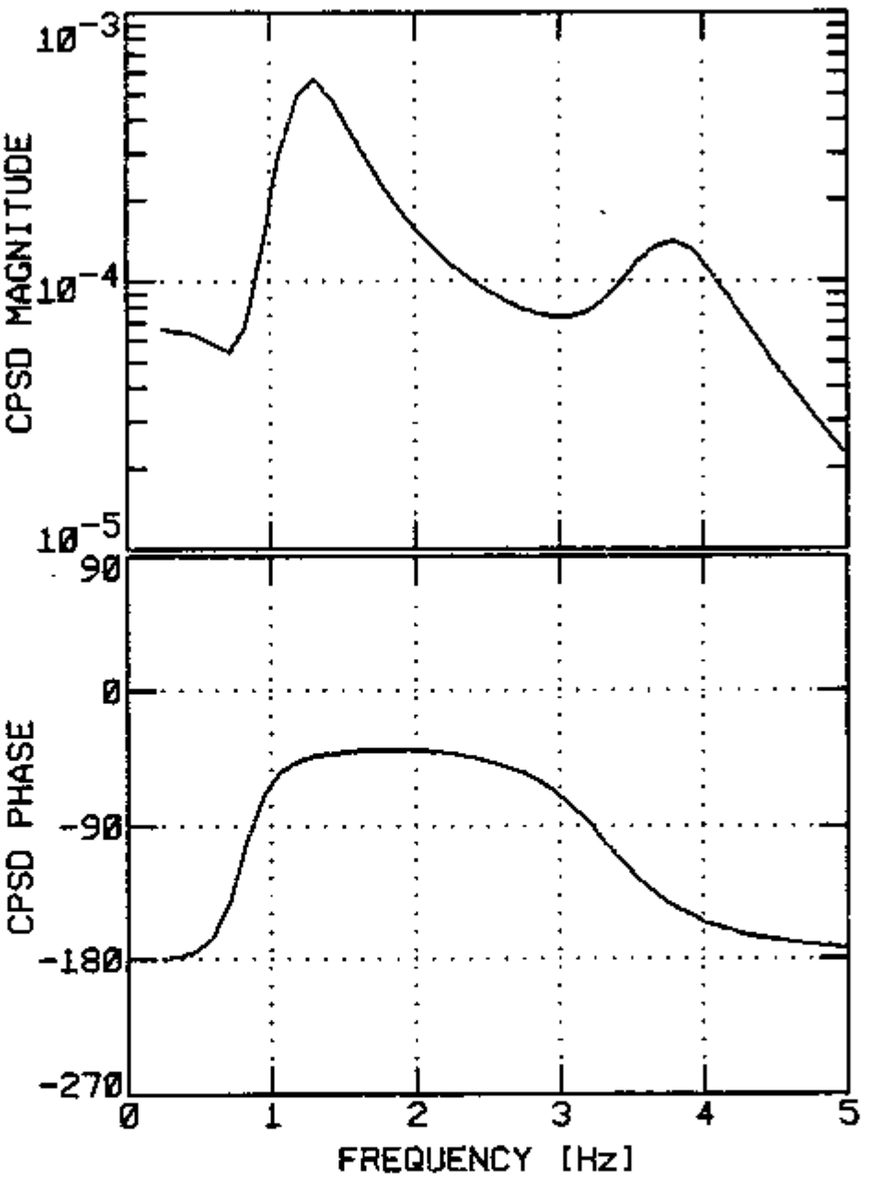

FIGURE 5-12. Calculated Reactor Vessel - Sensing Line CPSD.

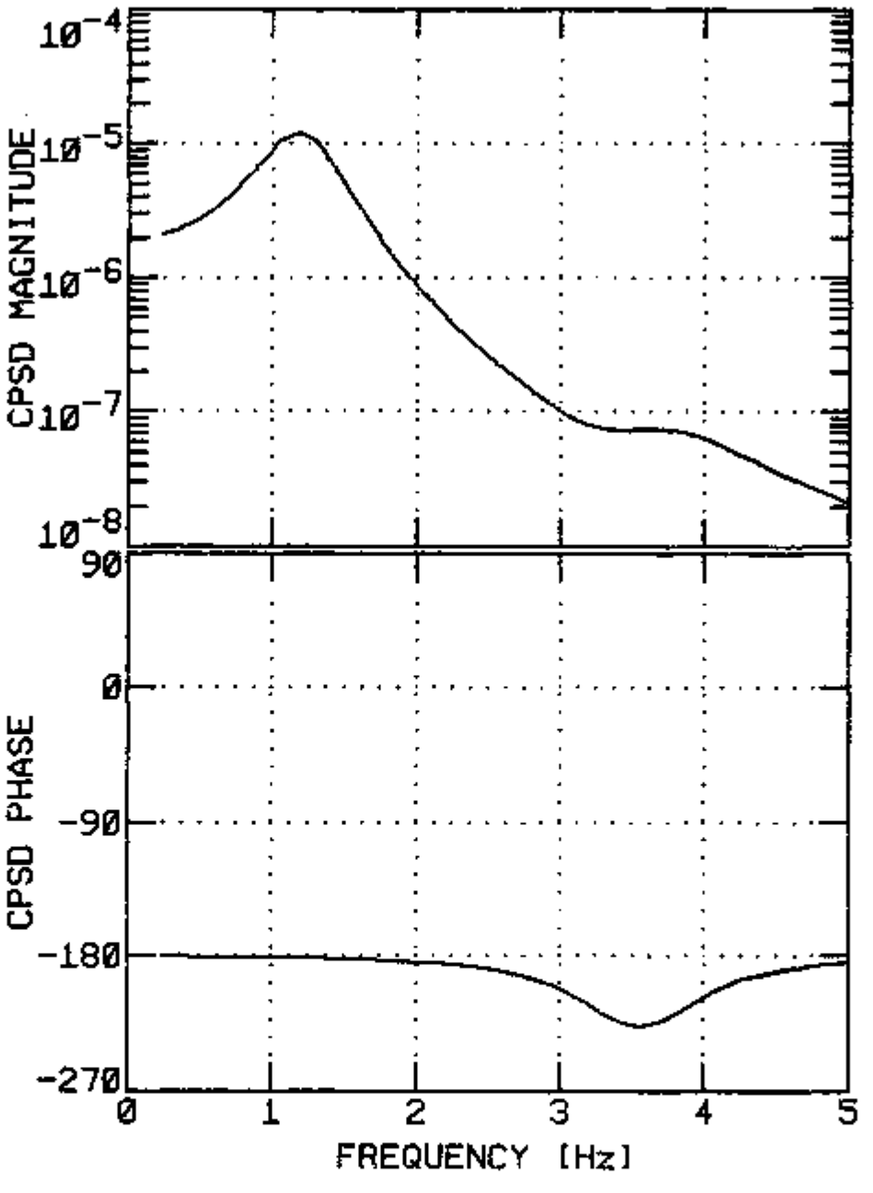

FIGURE 5-13. Calculated Reactor Vessel - Pressurizer CPSD.

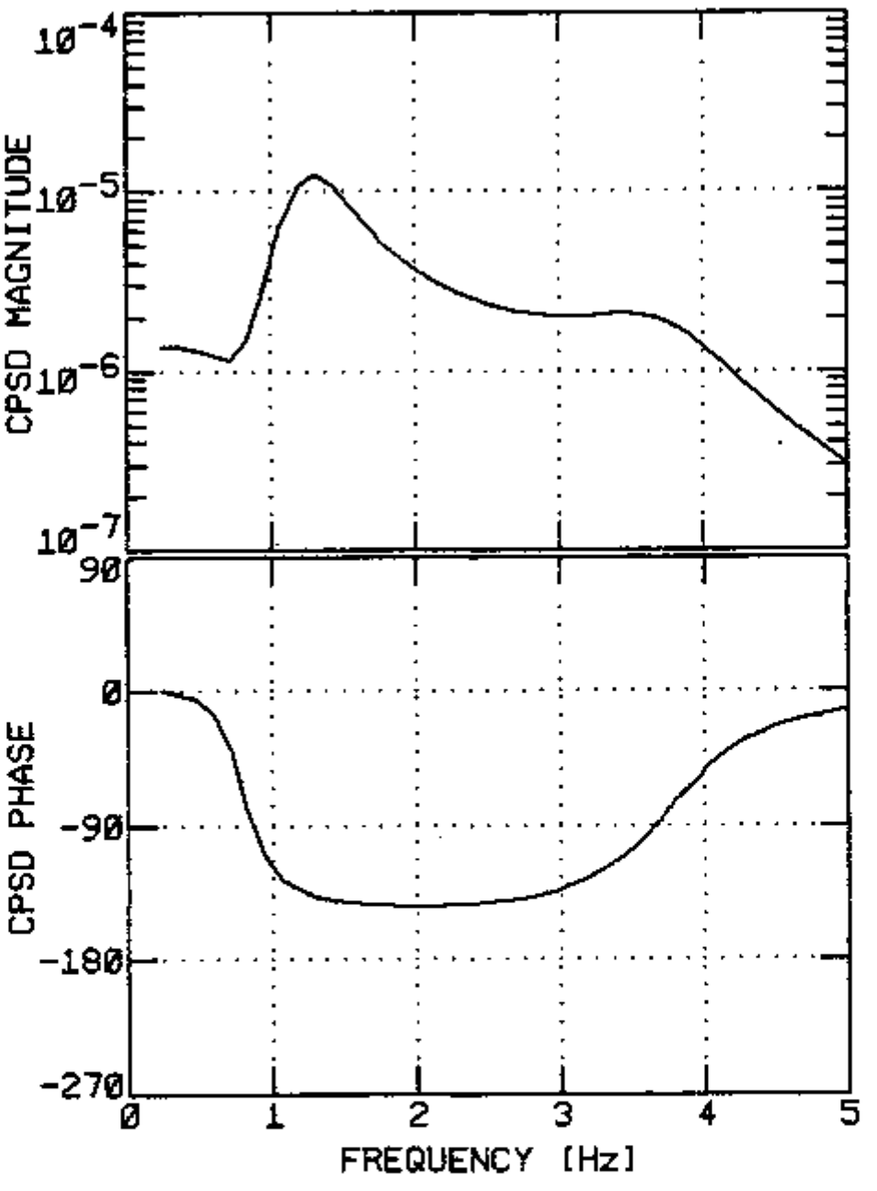

FIGURE 5-14. Calculated Sensing Line - Pressurizer CPSD.

densities in Figures 5-2 through 5-7, shows very good agreement between the model and the experiment.

5.4 Automatic Surveillance and Diagnostic System

The analog signals from the loop pressure sensors were connected to the analog to digital converter of a PDP-11/44 computer with a RSX-11M operating system. The automatic surveillance system was implemented as a multi-task system supervised by a command file program. The information flow diagram between the different tasks (or programs) is shown in Figure 5-15, where the boxes represent the programs and the arrows represent the flow of information. The stacks are last-in-first-out storage areas.

The four major programs in the automatic surveillance system are described below.

SAMPLR. This program digitizes the signals from the three pressure sensors and stores them in time data files. This program also checks for overloading of the analog to digital converter. When an overload is encountered, the partial time data file is deleted, a message is issued, and digitization is restarted. At the end of the data sampling period, the digitized time data file is stored at the top of the time data stack.

SPECTR. This program removes a time data file from the top of the time data stack, evaluates the spectral densities and stores the spectral densities data file at the top of the spectral data stack.

DOCTOR. This program removes a spectral densities file from the

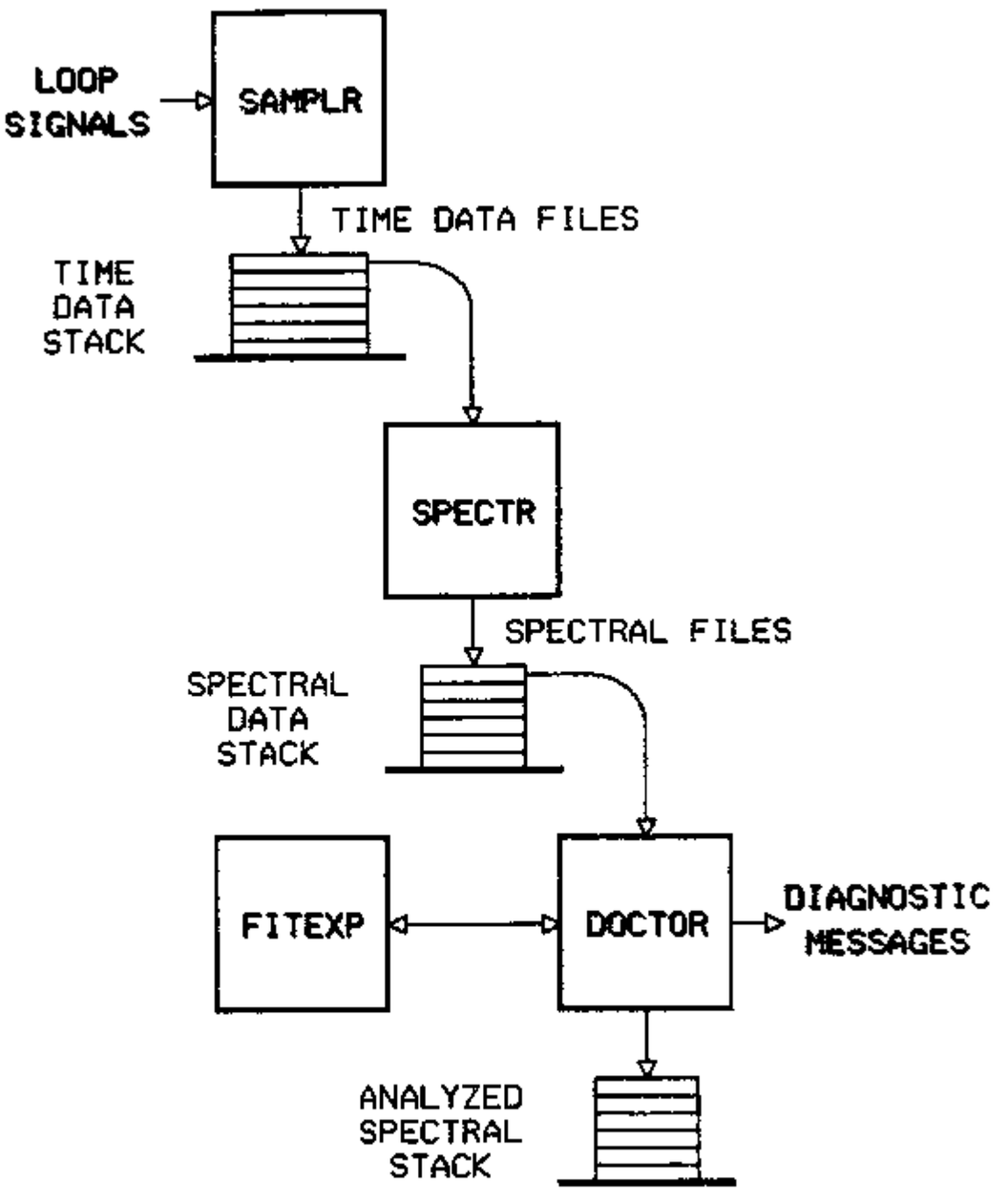

FIGURE 5-15. Information flow diagram for the task coordinator in the Automatic Surveillance and Diagnostic System.

top of the spectral data stack and analyzes the spectral densities looking for possible anomalies. It calculates some spectral characteristics (e.g. center frequency of resonances and power within certain frequency ranges) and compares them with normal characteristics. When the evaluated characteristics compare favorably with the normal characteristics, the diagnosis is ended with the statement that the loop is in normal operation. When the characteristics do not compare well with the normal ones, a warning message is issued, the FITEXP program is activated, and the DOCTOR program waits for FITEXP to perform the parameter identification. At the end of the parameter identification, the DOCTOR program receives the values of the identified parameters and issues the final diagnosis. The spectra that have been analyzed are stored in the analyzed spectral stack.

FITEXP. This program receives the experimental spectral densities from the DOCTOR program and performs a parameter identification using the SDL method (more details are given in the next section). At the end of the parameter identification, it passes to the DOCTOR program the values of the parameters identified, the final error functional, and the number of runs from the Wald-Wolfowitz test.

The overall control of the above programs is accomplished through a command file program. This control program is responsible for starting the other programs when input data are available for them and for keeping the current information displayed on the terminal screen.

This display shows the current date and time, which programs are active, and the number of data files in the stack.

In Figure 5-16, a typical screen display is shown. The screen is divided into three major areas with contrasting backgrounds to aid the user in locating and assimilating the information provided.

The top area is where the command file program displays the current information. On the left side of this area, the current time and date is displayed. The central part displays the active programs. Finally, the number of data files in each data stack is displayed on the right side.

The central area is used to interact with the computer and to display temporary information. For instance, in Figure 5-16 the calculated characteristics of the current set of spectra being analyzed is displayed.

The bottom area is used by the program DOCTOR to display the diagnostic messages. Each message begins with the time that the data was obtained, followed by the diagnosis of the loop operational condition. The information concerning the current set of data being analyzed is displayed on the top (of the bottom area). When the final diagnosis is made, the information in this area is moved down to make space for the information on the next set of data to be analyzed. With this procedure, the most recent analysis information is kept on the screen. For instance, in Figure 5-16, the data which was sampled starting at 21:13 is being analysed, and at 21:34 an anomaly was diagnosed as voids in the reactor vessel.

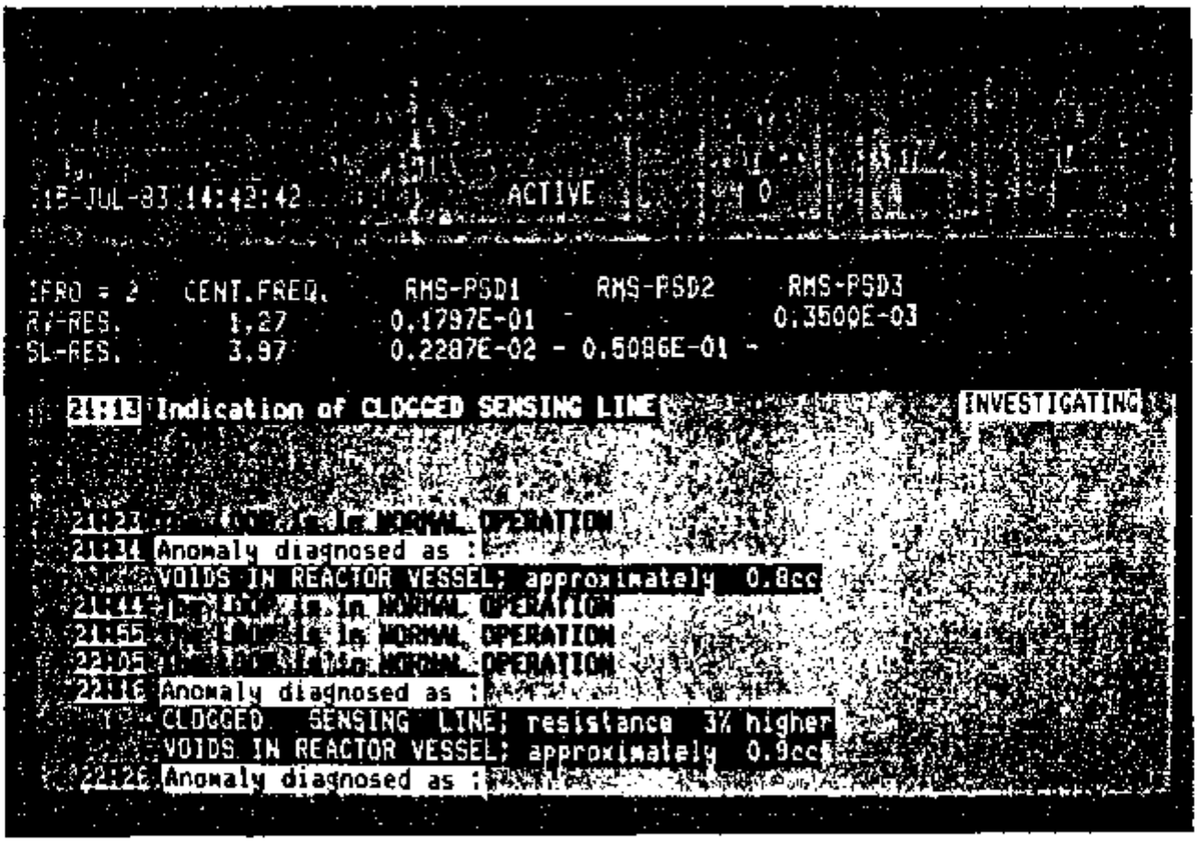

FIGURE 5-16. Sample screen display From the Automatic Surveillance and Diagnostic System.

5.5 Learning Parameter Identification in the Diagnostics System

The FITEXT program performs the parameter identifications by the Single-Direction Learning Method. This program is a version of the SDLPI program (listed in Appendix E) which was adapted to operate with the other programs in the automatic surveillance system. The objective of parameter identification in this system is to detect and quantify three types of anomalies, either separately or simultaneously. The three types of anomalies are:

1. Air in the sensing line. This kind of anomaly usually occurs during the installation of the sensor, and it reduces the response of the system to high frequencies. In terms of the parameters of the model, the sensing line capacitance is increased. The most important effect of this anomaly seen in the spectral densities is the shift of the sensing line resonance to a lower frequency than normal. In Figure 5-17, the sensing line PSD for the normal operation condition is compared with the air in the sensing line anomaly.
2. Clogged (or frozen) sensing line. This anomaly happens when deposits partially or totally block the sensing line internal area. Frozen sensing lines have also been observed in PWR reactors. The effect of this anomaly on the loop parameters is an increase of the sensing line resistance. In the spectral densities, this anomaly lowers the amplitude of the sensing line resonance. In Figure 5-18, this effect is

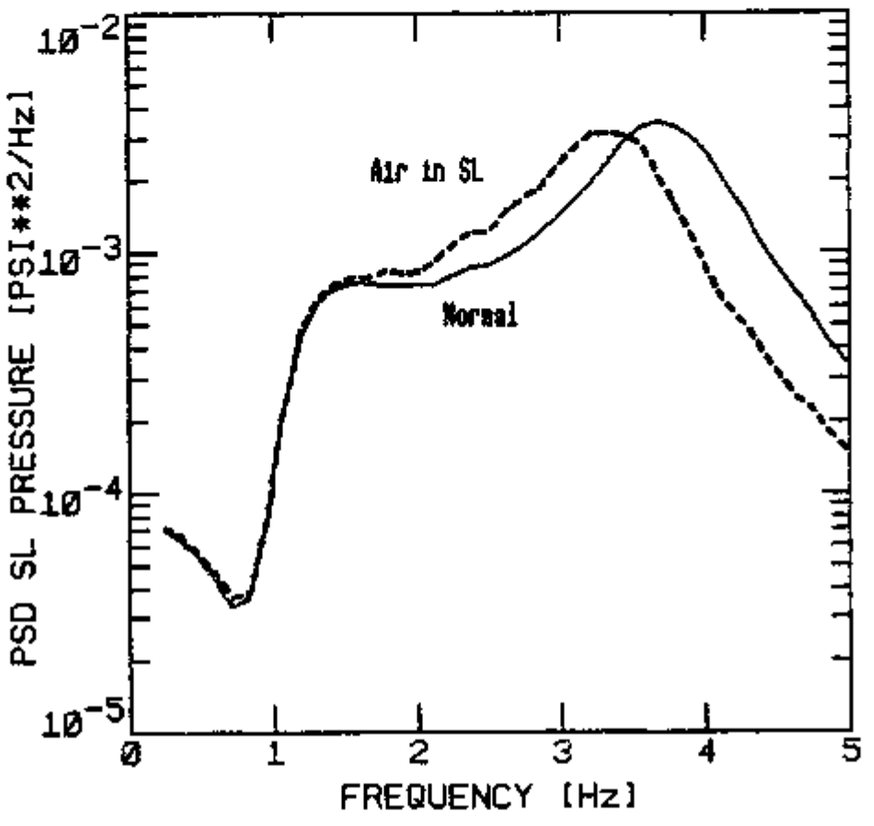

FIGURE 5-17. Experimental Sensing Line PSD for normal operation (solid line) and for Air in Sensing Line anomaly (dashed line).

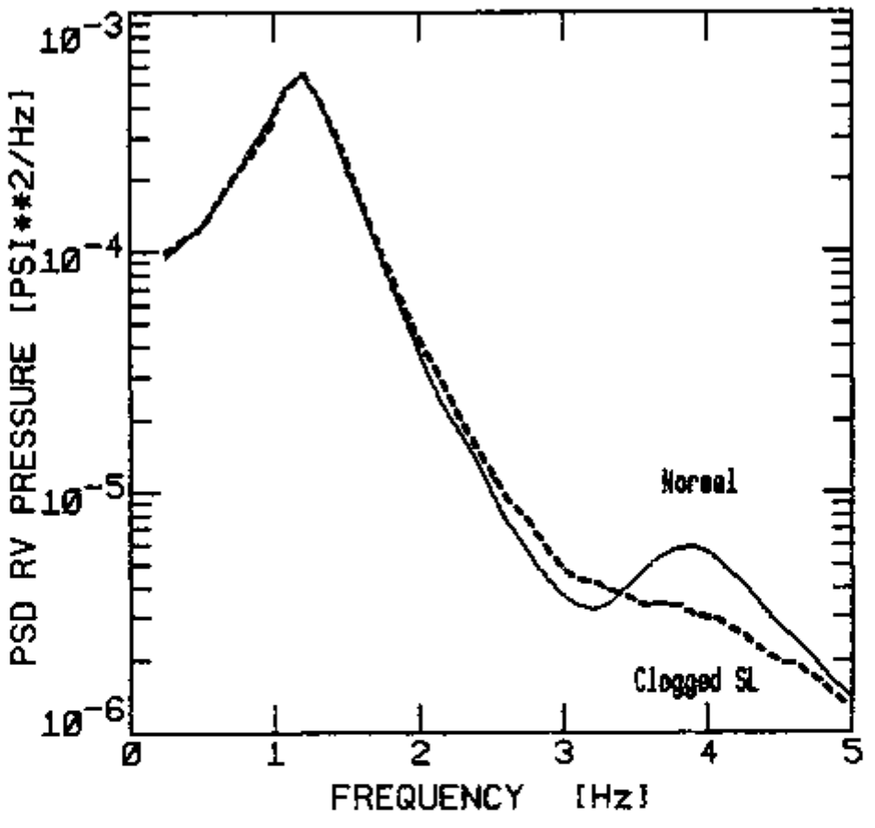

FIGURE 5-18. Experimental Reactor Vessel PSD for normal operation (solid line) and for Clogged Sensing Line anomaly (dashed line).

shown in the reactor vessel PSD. This anomaly is implemented in the loop by partially closing a valve in the sensing line.

3. Voids in the reactor vessel. This type of anomaly happened during the Three Mile Island accident, when steam, and later hydrogen, accumulated in the reactor vessel. The capacitance of the reactor vessel increases when any kind of fluid more compressible than the water is present. Figure 5-19 shows the effect of this type of anomaly on the pressurizer pressure PSD. This anomaly was implemented by injecting air into the reactor vessel.

It should be noted that, although these anomalies can be observed in the pressure spectral densities, they do not cause significant changes in the steady state pressure.

The reactor vessel capacitance, the sensing line resistance, and the sensing line capacitance were identified by the SDL method. The error functional included the PSDs of the three pressure signals and the three CPSDs, in the range from 0 to 5 Hz. The real and the imaginary parts of the CPSDs were treated independently for the error functional evaluation. The statistical weights (Equation 2-7) utilized upper limits for the variance of the estimates (32) given by

$$\sigma^2[\ C_{xy}\ ] = \frac{S_{xx}(w)\ S_{yy}(w)}{n} \quad , \qquad 5\text{-}27$$

where C_{xy} is either the real or the imaginary part of the CPSD variables "x" and "y",

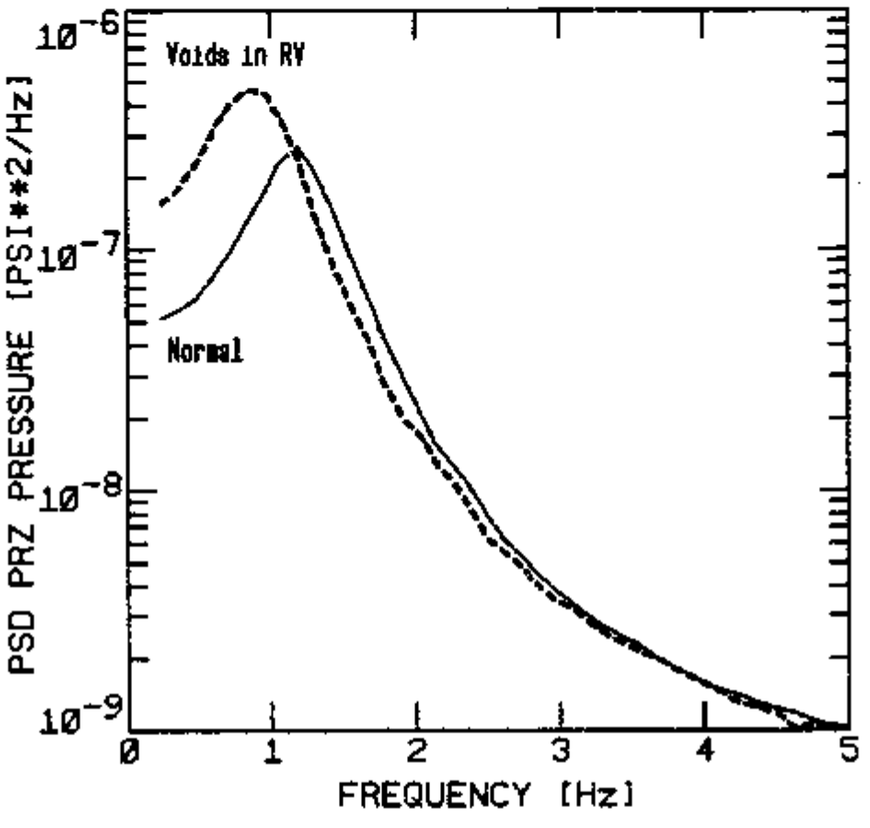

FIGURE 5-19. Experimental Pressurizer PSD for normal operation (solid line) and for Voids in the Reactor Vessel anomaly (dashed line).

S_{xx} is the PSD of variable "x",

S_{yy} is the PSD of variable "y", and

n is the number of independent blocks in the CPSD estimate.

For the pressure loop parameter identifications, a different type of features than the ones utilized in the previous work was used. As it can be seen in Figures 5-2 through 5-7, the pressure noise spectral densities present some characteristic resonances. The center frequencies and the amplitudes of these resonances change with the loop operational condition (see Figures 5-17 through 5-19). These characteristics were used to define the six dimensional feature vector components as follows:

f1 is the difference between the experimental and the model calculated reactor vessel resonance center frequency.

f2 is the difference between the experimental and the model calculated sensing line resonance center frequency.

f3 is the difference between the experimental and the model calculated amplitude of the reactor vessel resonance observed in the reactor vessel pressure PSD.

f4 is the difference between the experimental and the model calculated amplitude of the reactor vessel resonance observed in the pressurizer pressure PSD.

f5 is the difference between the experimental and the model calculated amplitude of the sensing line resonance observed in the reactor vessel pressure PSD.

f6 is the difference between the experimental and the model

calculated amplitude of the sensing line resonance observed in the sensing line pressure PSD.

In Figure 5-20, a learning curve is shown which was obtained under the conditions specified in Table 5-3. For the first 38 parameter identifications in this curve, the experimental data were measured with the pressure loop in normal operation. For the other 12 parameter identifications in the curve, one of the anomalies was implemented in the loop, as follows:

Parameter identification	Operational condition
1 through 38	Normal operation
39 through 42	Air in the Sensing line
43 through 46	Clogged sensing line
47 through 50	Voids in the reactor vessel.

It can be seen in this figure that, for the parameter identifications under normal operational conditions (1 through 38), the number of error functional evaluations necessary for convergence decreased from aproximately 100, for the first parameter identifications, to aproximately 30, for parameter identifications 20 through 38 (after learning). For the anomalous data, the number of error functional evaluations increased, but to a level substantially lower than the initial level. This fact shows that part of the information learned with the experimental data under normal operational conditions can be used to help perform the parameter identifications under anomalous operational conditions.

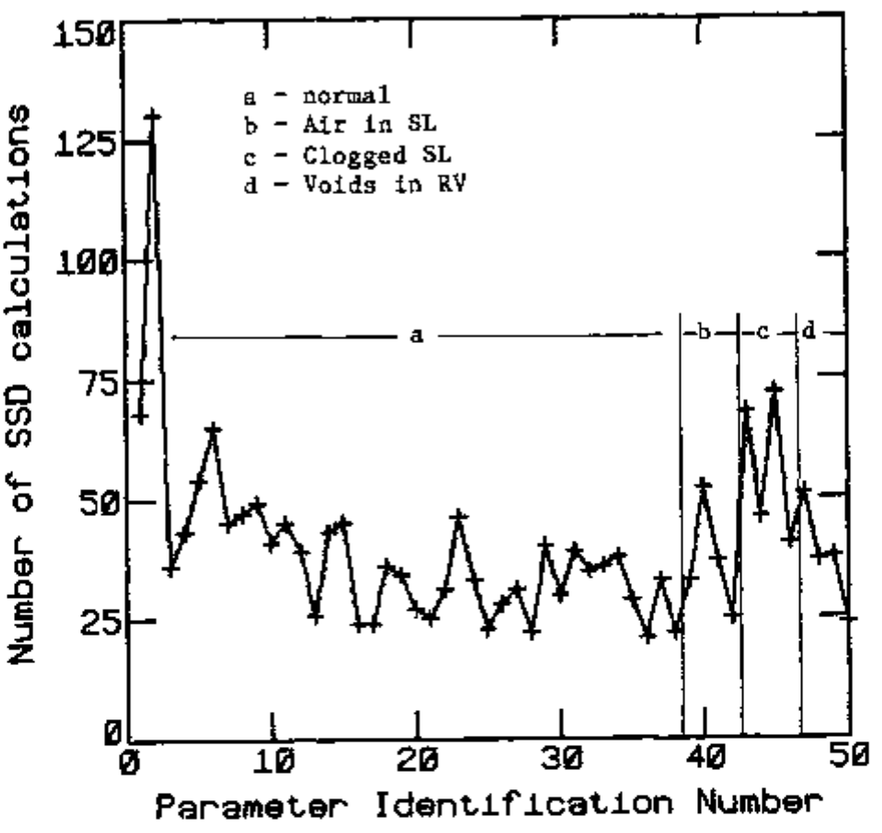

FIGURE 5-20. Learning curve (SDL method in the Automatic Surveillance and Diagnostic System).

TABLE 5-3. Parameter identification conditions used to obtain the results presented in Figure 5-20.

Model	Pressure loop noise
Parameters identified	Sensing line resistance Reactor vessel capacitance Sensing line capacitance
Initial parameter values	P1 = $0.50*10^{8}$ P2 = $0.20*10^{-7}$ P3 = $0.15*10^{-9}$
Experimental points	Reactor vessel pressure PSD Sensing line pressure PSD Pressurizer pressure PSD CPSD between RV and SL CPSD between RV and PRZ CPSD between SL and PRZ
Number of frequency points	42
Frequency range (log-uniform)	0 to 5 Hz
Experimental error	10%
Number of features	6
Region size	(radius) =0.05 + 0.2*(distance)
Number of basic directions	3
Precision criteria	0.0001
Number of parameter identifications per series	50
Number of series	1

The average value for the three parameters identified is given in Table 5-4, for each operational condition. It can be seen in this table that the only parameters that changed significantly following system anomalies were indeed the ones that should have varied during the anomalies. This shows that the FITEXP program did correctly identify the pertinent parameters.

In Table 5-5, the volume of air injected, either in the reactor vessel or in the sensing line, is compared with the value of the the volume calculated from Equation 5-2 with the values of the parameters given in Table 5-4. It can be seen that the identified values for the parameters give a reasonably good approximation for the true volume of air injected.

TABLE 5-4. Value for the identified loop parameters for four operational conditions.

	Parameter value			
Type	Normal	Air in SL	Clogged SL	Voids in RV
R_3	$3.27*10^{7}$	$3.06*10^{7}$	$6.68*10^{7}$	$3.23*10^{7}$
C_1	$1.24*10^{-8}$	$1.24*10^{-8}$	$1.23*10^{-8}$	$2.18*10^{-8}$
C_3	$3.20*10^{-10}$	$4.04*10^{-10}$	$3.06*10^{-10}$	$3.16*10^{-10}$

TABLE 5-5. Comparison between the identified and the measured volumes of air.

Anomaly	Volume of air	
	Measured	Identified
Air in SL	$1.9*10^{-5}$ ft^3	$1.5*10^{-5}$ ft^3
Voids in RV	$1.15*10^{-3}$ ft^3	$1.20*10^{-3}$ ft^3

CHAPTER 6

CONCLUSIONS AND RECOMMENDATIONS

6.1 Introduction

A new Heuristic Reinforcement Learning Method has been developed and proven to be successful for the on-line detection of anomalies in an experimental pressure loop. This technique has far-reaching implications for on-line surveillance and diagnostics of dynamic systems such as Nuclear Power Plants, Fossil Power Plants, and many other industrial and manufacturing facilities.

The accomplishments of this dissertation are highlighted in Section 6.2. Suggestions for further research are included in Section 6.3.

6.2 Accomplishments

A general heuristic learning algorithm for parameter identification purposes has been implemented. During the implementation process, a completely new heuristic learning method (the Single-Direction Learning Method) has been developed, which learns faster and provides more accurate results than previous

algorithms, such as the Direct Search method.

The success of heuristic parameter identification techniques hinges on the utilization of an appropriate low dimensional space, the feature space, to characterize the search situation. For the development of such a suitable feature space, the new concept of "fuzzy" maps was introduced. The introduction of this rather unorthodox concept highlights two characteristics of heuristic algorithms: (i) the need for minimizing computer memory requirements and (ii) the intrinsic "fuzziness" of the input data themselves, due to statistics and fluctuations imposed by the environment. Because of the "non-analytical" nature of heuristic learning methods (in the sense that analytical manipulations are drastically reduced), it is, in general, very difficult to prove the existence of optimal solutions in an entirely rigorous manner. This issue has been addressed in this dissertation both analytically and by computer simulation methods. In Appendix B, it is shown rigorously that, for unimodal error functionals, the present heuristic learning algorithm yields the optimal solution. For non-unimodal error functionals (i.e. for those functionals exhibiting several minima) the validation was performed via computer simulations utilizing as input data the results of a steam generator noise model (see Chapter 4).

In the realm of applications to "real world" problems, an original development has taken place with the implementation of the present heuristic learning parameter identification method to an experimental pressure loop. This development has resulted in the

creation of a trainable surveillance and diagnostics system, which is capable of detecting and quantifying anomalies in dynamic system.

In summary, the initial objective of implementing Heuristic Reinforcement Learning techniques for parameter identification purposes and applying these techniques to an actual physical system have been accomplished.

6.3 Recommendations for Further Research.

There are at least two main areas of future development which should be pursued; the first is related to the method itself and the second to its applications. Concerning the first area of development, one should (i) do extensive work on the relationship between the errors in the identified parameters and the uncertainties of the input experimental data and (ii) study the issue of the dimensionality of feature space vis-a-vis the parameter space dimension and the limitations in the available computer memory.

The second area of development refers to the fascinating possibilities for the minimization of variational principles, whose optimal trajectories (representing neutron flux profiles or trajectories in dynamic problems) could be obtained by the present heuristic learning method. The solution of the Euler-Lagrange equations associated with a given variational principle by this method would be of extraordinary interest in fields like Reactor Design and Robotics.

LIST OF REFERENCES

LIST OF REFERENCES

1. L. F. Pau, Failure Diagnosis and Performance Monitoring, (New York and Basel: Marcel Dekker Inc., 1981).

2. Robert E. Uhrig, Random Noise Techniques in Nuclear Reactor Systems, (New York: The Ronald Press Company, 1970).

3. M. M. R. Williams, Random Processes in Nuclear Reactors, (Oxford: Pergamon Press, 1974).

4. Joseph A. Thie, Power Reactor Noise, (La Grange Park, Illinois: American Nuclear Society, 1981).

5. George N. Saridis, Self-Organizing Control of Stochastic Systems, (New York: Marcel Dekker Inc., 1977).

6. F. W. Stallmann, "Theory and Practice of General Adjustment and Model Fitting Procedures," ORNL/TM-7896, Oak Ridge National Laboratory, (1981).

7. R. Bellman, Adaptative Control Process, A Guided Tour, (Princeton: Princeton University Press, 1961).

8. George N. Saridis, "Application of Pattern Recognition Methods To Control Systems," IEEE Transactions on Automatic Control, Vol. AC-26, (1981), pp. 638-45.

9. H. H. Rosenbrock, "An Automatic Method for Finding the Greatest or Least Value of a Function," Computer Journal, Vol. 3, no. 3, (1960), pp.175-84.

10. M. J. D. Powell, "An Efficient Method for Finding the Minimum of a Function of Several Variables without Calculating Derivatives," Computer Journal, Vol. 7, (1964), pp. 155-62.

11. J. A. Nelder and R. Mead, "Simplex Method for Minimization," Computer Journal, Vol. 7, (1965), pp. 308-13.

12. Robert Hooke and T. A. Jeeves, " 'Direct Search' Solution of Numerical and Statistical Problems," J. Assoc. Comp. Mach., Vol. 8, (1961), pp. 212-29.

13. M. D. Waltz and K. S. Fu, "A Heuristic Approach to Reinforcement Learning Control Systems," IEEE Transactions on Automatic Control, Vol. AC-10, (1965), pp. 390-98.

14. King-Sun Fu, "Learning Control Systems--Review and Outlook," IEEE Transactions on Automatic Control, Vol. AC-15, (1970), pp. 210-21.

15. Julius T. Tou and R. C. Gonzalez, Pattern Recognition Principles, (London: Addison-Wesley Publishing Company, 1974).

16. Bernard Dubuisson and Patrice Lavison, "Surveillance of a Nuclear Power Plant by Use of a Pattern Recognition Methodology," IEEE Transactions on Systems, Man, and Cibernetics, Vol. SMC-10, (1980), pp. 603-909.

17. James L. Macdonald and Billy V. Koen, "Application of Artificial Intelligence to Digital Computer Control of Nuclear Reactors," Nuclear Science and Engineering, Vol. 56, (1975), pp. 142-51.

18. Marian Bubak and Jacek Moscinski, "A Heuristic Approach to Reinforcement Learning Control of Point-Model Nuclear Reactor," Atomkernenergie Vol. 31, (1978), p. 63.

19. Jacek Kitowski and Jacek Moscinski, "Computer Simulation of Heuristic Reinforcement-Learning Systems for Nuclear Power Plant Load Change Control," Computer Physics Communications, Vol. 18, (1979), pp. 339-352.

20. Tsutomu Hoshino, "In-Core Fuel Management Optimization by Heuristic Learning Technique," Nuclear Science and Engineering, Vol. 49, (1972), pp. 59-71.

21. D. N. Fry, "Experience in Reactor Malfunction Diagnosis Using On-Line Noise Analysis," Nuclear Technology, Vol. 10, (1971), pp. 273-82.

22. R. C. Gonzalez, D. N. Fry and R. C. Kryter, "Results in the Application of Pattern Recognition Methods to Nuclear Reactor Core Component Surveillance," IEEE Transactions on Nuclear Science, Vol. 21, (1974).

23. K. R. Piety, "A Statistical Algorithm to Perform Automated Signature Analysis on Power Spectral Density Data," Progress in Nuclear Energy Vol. 1, (Oxford: Pergamon Press, 1977), pp. 781-802.

24. C. M. Smith and F. J. Sweeney, "Demonstration of an Automatic On-Line Surveillance System at a Commercial Nuclear Power Plant," Proceedings of The Fifth Power Plant Dynamics, Control and Testing Symposium, (The University of Tennessee, 1983), Paper no. 31.

25. B. R. Upadhyaya and F. J. Sweeney, " Theoretical and Experimental Stochastic Modeling Analysis of PWR Core Heat Transfer," Proceedings of The Fifth Power Plant Dynamics, Control and Testing Symposium, (The University of Tennessee, 1983), Paper no. 46.

26. J. A. March-Leuba, G. deSaussure and R. B. Perez, "CARDIOGRAMA: A Stochastic, Semi-Empirical Methodology for Power Reactor Surveillance and Diagnostics," Transactions of the American Nuclear Society, Vol. 39, (1981), pp. 951-2.

27. J. A. Mullens and J. A. Thie, "Understanding Pressure Dynamics Phenomena in PWRs for Surveillance and Diagnostic Applications," Proceedings of The Fifth Power Plant Dynamics, Control and Testing Symposium, (The University of Tennessee, 1983), Paper no. 44.

28. R. Fletcher and M. J. D. Powell, "A Rapid Descent Method for Minimization," Computer Journal, Vol. 6, (1983), pp.163-8.

29. J. B. Rosen, "The Gradient Projection Method for Nonlinear Programing," J. Soc. Indust. Appl. Math., Vol. 8, (1960), pp. 181-217.

30. A. Wald and J. Wolfowitz, "On a Test Rather Samples Are From the Same Population," Annals. of Math. Stat., Vol XI, (1940), pp. 147-62.

31. Mohamed R. Ali, "Lumped Parameter, State Variable Dynamic Models for U-Tube Recirculation Type Nuclear Steam Generator," Ph.D. Dissertation, The University of Tennessee, Knoxville, (August 1976).

32. Julius S. Bendat and Allan G. Piersol, Random Data: Analysis and Measurement Procedures, (New York: Wiley-Interscience, 1971).

33. Donald E. Knuth, The Art of Computer Programming, (London: Addison-Wesley, 1969).

34. J. A. Mullens, J. A. Thie and M. E. Buchanan, "Model for ORNL-SDC Pressure Loop," ORNL, Surveillance and Diagnostics Group, Internal Report, (January, 1983).

35. R. B. Perez, NE-6120: Selected Topics in Reactor Theory, Course Notes, The University of Tennessee, (January, 1982).

APPENDICES

APPENDIX A

SPECTRAL DENSITY EQUATION

Equations for the spectral densities in terms of the transfer functions of the system and of the input spectral densities for particular dynamic systems have been derived in several references, but the derivation for a general linear dynamic system has not been found. In this appendix, the equation for the spectral density between two system variables is derived for a general multiple input, multiple output, linear dynamic system. The derivation is performed in a similar way to that used by Perez (35) for a dynamic model of a nuclear reactor.

The equations for a linear, lumped parameter, dynamic system can be written as

$$A_I \frac{d^I}{dt^I}\underline{x}(t) + A_{I-1}\frac{d^{I-1}}{dt^{I-1}}\underline{x}(t) + \ldots + A_0\underline{x}(t) =$$

$$C_J \frac{d^J}{dt^J}\underline{u}(t) + C_{J-1}\frac{d^{J-1}}{dt^{J-1}}u(t) + \ldots \; C_0 u(t) \quad , \qquad \text{A-1}$$

where A_i and C_j are constant matrices,

$\underline{x} = (x_1, x_2, \ldots x_n)^T$ are the system variables, and

$\underline{u} = (u_1, u_2, \ldots u_m)^T$ are the system inputs.

Laplace transforming Equation A-1 and assuming zero initial conditions yields

$$B(s)\underline{X}(s) = H(s)\underline{U}(s) \quad , \qquad \text{A-2}$$

where $B(s) = s^n A_n + s^{n-1} A_{n-1} + \ldots + A_0$, and

$H(s) = s^m C_m + s^{m-1} C_{m-1} + \ldots + C_0$.

Left multiplying Equation A-2 by $B^{-1}(s)$, one obtains

$$\underline{X}(s) = G(s)\underline{U}(s) \quad , \qquad \text{A-3}$$

where $G(s) = B^{-1}(s)H(s)$.

A generic element of Equation A-3 is

$$x_\ell(s) = \sum_{k=1}^{M} g_{\ell k}(s) u_k(s) \quad , \qquad \text{A-4}$$

where $g_{\ell k}(s)$ is the transfer function from input "k" to system variable "ℓ". Inverse laplace transforming Equation A-4 using the Faltung theorem gives

$$x_\ell(t) = \sum_{k=1}^{M} \int_0^t g_{\ell k}(t_1) u\ (t-t_1) dt_1 \quad , \qquad \text{A-5}$$

where $g_{\ell k}(t) = L^{-1}\{ g_{\ell k}(s) \}$, and

$$u_k(t) = L^{-1}\{ u_k(s) \} \quad .$$

Observing that $g_{\ell k}(t)$ and $u_k(t)$ are equal to zero for negative time, the limits of integration in Equation A-5 can be extended to infinity, i.e.

$$x_\ell(t) = \sum_{k=1}^{M} \int_{-\infty}^{\infty} g_{\ell k}(t_1) u_k(t-t_1) dt \qquad \text{A-6}$$

The definition of cross correlation function (autocorrelation is the same as cross correlation when the two variables are the same) is given by

$$S_{x_\ell x_m}(\tau) = S_{\ell m}(\tau) = E\left[x_\ell(t) x_m(t-\tau) \right] \quad , \qquad \text{A-7}$$

where $E\left[\ \right]$ means the expectation or ensemble average.

Substituting Equation A-6 into A-7, one obtains

$$S_{\ell m}(\tau) = E\left[\sum_{k=1}^{M} \int_{-\infty}^{\infty} g_{\ell k}(t_1) u_k(t-t_1) dt_1 \sum_{j=1}^{M} \int_{-\infty}^{\infty} g_{mj}(t_2) u_j(t+\tau-t_2) dt_2 \right] \quad . \qquad \text{A-8}$$

Rearranging dummy indexes and variables and taking in account the deterministic nature of the transfer functions, Equation A-8 can be rewritten in the form

$$S_{\ell m}(\tau) = \sum_{k=1}^{M}\sum_{j=1}^{M}\int_{-\infty}^{\infty}dt_1\int_{-\infty}^{\infty}dt_2 g_{\ell k}(t_1)g_{mj}(t_2)E\left[u_k(t-t_1)u_j(t+\tau-t_2)\right] \quad . \qquad \text{A-9}$$

In view of the definiton of crosscorrelation in Eq. A-7, Eq. A-9 can be written as

$$S_{\ell m}(\tau) = \sum_{k=1}^{M}\sum_{j=1}^{M}\int_{-\infty}^{\infty}dt_1\int_{-\infty}^{\infty}dt_2 g_{\ell k}(t_1)g_{mj}(t_2)\, S_{kj}(t_1+\tau-t_2) \quad . \qquad \text{A-10}$$

This provides the relationship between the input and the output correlation functions.

In order to obtain the corresponding expresion for the spectral densities, the functions inside the integrals in Equation A-10 are written in terms of their respective Fourier transforms as

$$g_{\ell k}(t_1) = \frac{1}{2\pi}\int_{-\infty}^{\infty}dw_1 g_{\ell k}(w_1)e^{iw_1 t_1} \quad , \qquad \text{A-11}$$

$$g_{\ell j}(t_2) = \frac{1}{2\pi}\int_{-\infty}^{\infty}dw_2 g_{\ell j}(w_2)e^{iw_2 t_2} \quad , \text{ and} \qquad \text{A-12}$$

$$S_{kj}(t_1+\tau-t_2) = \frac{1}{2\pi}\int_{-\infty}^{\infty}dw_3 S_{kj}(w_3)e^{iw_3(t_1+\tau-t_2)} \quad . \qquad \text{A-13}$$

Substituting Equations A-11, A-12 and A-13 into Equation A-10, and Fourier transforming Equation A-8 yields

$$S_{\ell m}(w)=\frac{1}{8\pi^3}\sum_{k=1}^{M}\sum_{j=1}^{M}\int_{-\infty}^{\infty}d\tau\int_{-\infty}^{\infty}dw_1\int_{-\infty}^{\infty}dw_2\int_{-\infty}^{\infty}dw_3\int_{-\infty}^{\infty}dt_1\int_{-\infty}^{\infty}dt_2 g_{\ell k}(w_1)g_{mj}(w_2)S_{kj}(w_3)$$

$$* \quad e^{i\tau(w_3-w)}\, e^{it_1(w_1+w_3)}\, e^{it_2(w_2-w_3)} \quad . \qquad \text{A-14}$$

To deal with the various integrals in Equation A-14, the following definition of Dirac's delta function is utilized,

$$\delta(y) = \frac{1}{2\pi}\int_{-\infty}^{\infty} e^{ivy}\, dv \quad . \qquad \text{A-15}$$

Applying the above definition, Equation A-14 can be written as

$$S_{\ell m}(w)=\sum_{k=1}^{M}\sum_{j=1}^{M}\int_{-\infty}^{\infty}dw_1\int_{-\infty}^{\infty}dw_2\int_{-\infty}^{\infty}dw_3 g_{\ell k}(w_1)g_{mj}(w_2)S_{kj}(w_3)\delta(w_2-w_3)\delta(w_1+w_3)\delta(w_3-w) \qquad \text{A-16}$$

In view of the properties of Dirac's delta function, straightforward integration of Equation A-16 yields

$$S_{\ell m}(w) = \sum_{k=1}^{M}\sum_{j=1}^{M} g_{\ell k}(-w)g_{mj}(w)S_{kj}(w) \qquad \text{A-17}$$

Utilizing the following property of transfer functions,

$$g^{*}_{kj}(-w) = g_{kj}(w) \qquad \text{A-18}$$

Equation A-17 reduces to

$$S_{\ell m}(w) = \sum_{k=1}^{M} \sum_{j=1}^{M} g^{*}_{\ell k}(w) g_{mj}(w) S_{kj}(w) \quad , \qquad \text{A-19}$$

which is the general equation for the spectral density.

APPENDIX B

PROOF OF CONVERGENCE OF THE PARAMETER IDENTIFICATION METHODS

The convergence criteria used to stop the iterative search in the learning parameter identification methods requires that an unsuccessful unidirectional search be performed along each direction $\underline{d}_1, \underline{d}_2, \underline{d}_3, \cdots \underline{d}_m$, where $\underline{d}_j=(d_{j1}, d_{j2}, d_{j3}, \ldots d_{jn})^T$ is a directional vector in parameter space. Let the error functional, $Q(\underline{P})$, be a continuous, unimodal function of the parameter vector, $\underline{P}=(P_1, P_2, \ldots P_n)^T$, with continuous derivatives. In that condition, the optimum parameter vector, $\underline{P}^*$, is the only solution to the set of equations

$$\left[\frac{\partial Q}{\partial P_i}\right]_{\underline{P}^*} = 0 \quad , \qquad i=1,2,3,\ldots n \quad . \qquad \text{B-1}$$

A unidirectional search starting from a parameter vector $\underline{P}^+$, along the line $\underline{d}_j$ is defined by the solution for the equation

$$\frac{dQ(\underline{P}^{+} + s\underline{d}_j)}{ds} = 0 \qquad \text{B-2}$$

and the unidirectional search is said to be unsuccessful when Equation B-2 is satisfied for s=0. In that condition, Equation B-2 can be written as

$$\left[\frac{\partial Q}{\partial P_1}\right]_{\underline{P}^{+}} d_{j1} + \left[\frac{\partial Q}{\partial P_2}\right]_{\underline{P}^{+}} d_{j2} + \cdots \left[\frac{\partial Q}{\partial P_n}\right]_{\underline{P}^{+}} d_{jn} = 0 \quad . \qquad \text{B-3}$$

From the convergence criteria, Equation B-3 is satisfied for all directions at the convergence point, $\underline{P}^{+}$. In a matrix form, this condition can be written as

$$D\,\underline{Q}' = \underline{0} \quad , \qquad \text{B-4}$$

where

$$D = \begin{bmatrix} \underline{d}_1^{T} \\ \underline{d}_2^{T} \\ \underline{d}_3^{T} \\ \cdot \\ \cdot \\ \cdot \\ \underline{d}_m^{T} \end{bmatrix} ,$$

$$\underline{Q}' = \left[\left[\frac{\partial Q}{\partial P_1} \right]_{\underline{P}^+}, \left[\frac{\partial Q}{\partial P_2} \right]_{\underline{P}^+}, \quad \cdots, \quad \left[\frac{\partial Q}{\partial P_u} \right]_{\underline{P}^+} \right]^T , \text{ and}$$

$\underline{0}$ is an m-dimensional zero vector.

If the set of directions contains a complete set (i.e. there are "n" linearly independent vectors in the set), the rank of matrix "D" is "n". Therefore the only solution of Equation B-5 is

$$\underline{Q}' = \underline{0} \qquad \text{B-6}$$

or

$$\left[\frac{\partial Q}{\partial P_i} \right]_{\underline{P}^+} = 0 \quad , \text{ for } i=1,2,\ldots n. \qquad \text{B-7}$$

Comparing with Equation B-1 and considering the unimodalidaty of the error functional, it can be concluded that

$$\underline{P}^+ = \underline{P}^* \quad . \qquad \text{B-8}$$

Therefore, the learning parameter identification methods do converge to the optimum parameter vector whenever $Q(\underline{P})$ is a continuous, unimodal function of the parameter vector.

APPENDIX C

THE UNIDIRECTIONAL SEARCH

In this appendix, the unidirectional search used with both learning parameter identification methods is described. A few small differences exist between the versions of the unidirectional search used with the FDL method and with the SDL method. The description that follows is for the version used with the SDL method, which is more general. The version for the FDL method is, for all practical purposes, equivalent to the one described with the confidence index always equal to zero.

The unidirectional search is equivalent to a unidimensional search where the parameter vector, $\underline{P}$, is allowed to vary on a straight line defined by an initial parameter vector , $\underline{P}_o$, and a directional vector, $\underline{d}$. Any point on this line can be defined by a single scalar, s, as

$$\underline{P}(s)=\underline{P}_o + s\underline{d} \quad . \qquad \text{C-1}$$

The unidirectional search seeks for the parameter vector along the line defined by Equation C-1, that minimizes the error functional,

$Q(\underline{P})$. Implicit in the method is the assumption that the error functional is a continuous, positive, unimodal function of the parameters within the domain defined by a minimum and maximum value for each parameter.

A method similar to the one described by Powell (10) was used for the unidirectional search. The method is divided into three steps: (a) the generation of the three first points, (b) the iterative selection of a fourth test point, and (c) the convergence check.

(a) Given an initial parameter vector, $\underline{P}_o$, the correspondent error functional, $q_o=Q(\underline{P}_o)$, and a guess for the optimum value of s, $\hat{s}$; the first and second test points are always: the origin, $s_1=0$, and the guess, $s_2=\hat{s}$. As specified in Table C-1, the third test point is selected on the basis of the error functional, q_2, calculated at the second test point, s_2, and a confidence index, Ic. Once the third test point, s_3, is selected, the correspondent error functional, q_3, is evaluated.

(b) The selection of the fourth test point, s_4, is performed by calculating the second order polynomial, M(s), that passes through the three test points: (s_1,q_1), (s_2,q_2) and (s_3,q_3). If the second derivative of the polynomial, M"(s), is positive, the fourth test point, s_4, is selected as the point that minimizes the polynomial M(s) ($M'(s)=0$; $M''(s)>0$). The three possible cases when the second derivative is non-positive are illustrated in Figure C-1. In the first two

TABLE C-1. Selection of the third test point.

Confidence index	Third trial point	
	q2 < q1	q2 > q1
0	$2*\hat{g}$	$-\hat{g}$
1	$2*\hat{g}$	$0.5*\hat{g}$
2	$1.6*\hat{g}$	$0.66*\hat{g}$
3	$1.3*\hat{g}$	$0.75*\hat{g}$

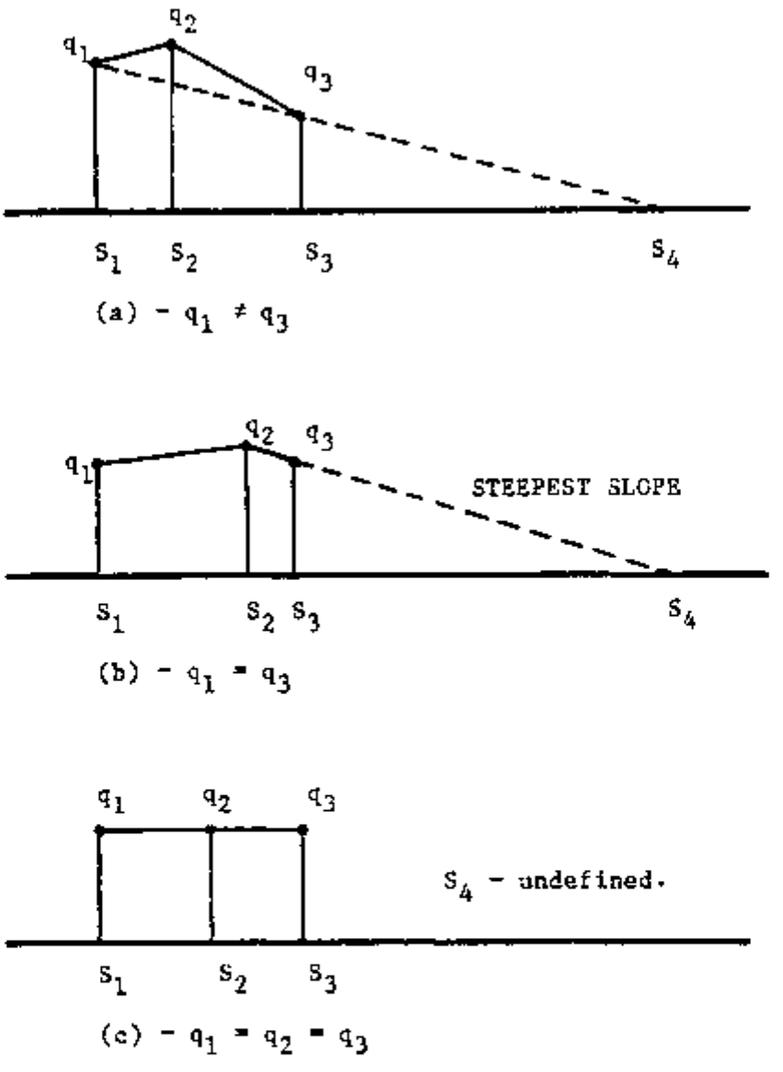

FIGURE C-1. Selection of the fourth trial point when the second derivative of the interpolant polynomial is negative or zero.

cases, the fourth test point is chosen as the point where the linear extrapolation reaches zero. In the third case the direction that minimizes the functional is completely undefined. After evaluating the error functional at the fourth point, s_4, this step is repeated with the current best point and either its left and right neighbors, if they exist, or its two nearest neighbors.

(c) The above iterative procedure is terminated when the distance between the fourth point, s_4, and the previous best point is smaller than a variable convergence criteria, whose values are shown in Figure C-2. This variable convergence criteria was adopted to reduce the total number of error functional evaluations, since a minimum far from the origin indicates that the overall search is still far from converging; in which case, an accurate determination of the unidirectional minimum is not necessary.

To assure that the unidirectional search is performed within the allowed domain, before accepting a test point, the correspondent parameter values are compared with their respective limits. If one or more parameters are outside the range, the value of the test point is modified such that all parameters are inside the range or at the limit.

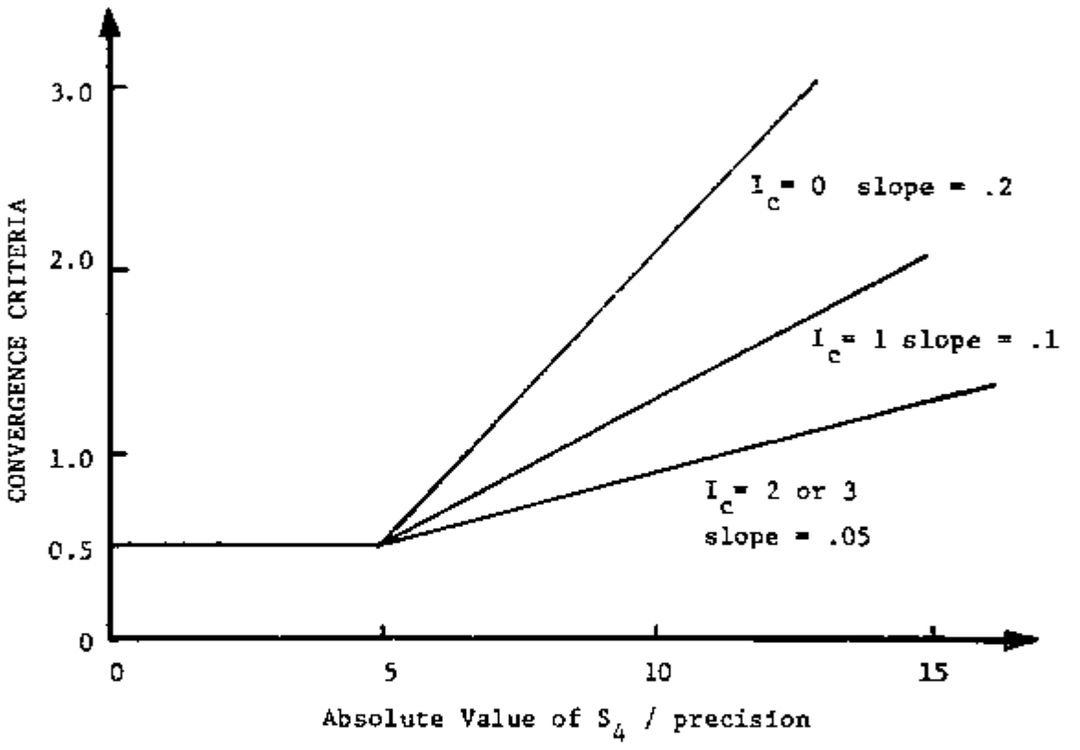

FIGURE C-2. Convergence criteria as a function of the distance from the origin and confidence index.

APPENDIX D

LISTING OF THE "FDLPI" COMPUTER CODE

In this appendix, the listings of the FDLPI program is presented. This program is the final version of a series of programs used during the development of the Fixed-Directions Learning method described in Chapter 3, Section 3-4.

```
CHH-
C       Module name: FOLPI
C
C       Version X01.00  Last edit: 01-JUL-83 13:52
C       Status: Development/Debugging
C
C       Revision history:
C
C               Version X01.00 23-JUN-83 10:43 - 01-JUL-83 13:52
C               Created by: E.BACHARD
C
CHH-
C
C   SUBROUTINES CALLED:
C       GETEP - SUBROUTINE TO GET THE EXPERIMENTAL POINTS (USER SUPLIED)
C       GETPA - SUBROUTINE TO GET PARAMETER INFORMATION   (USER SUPLIED)
C
C   FUNCTIONS:FUNC - RETURNS THE VALUE OF THE FUNCTION
C
C
C
C
        PARAMETER MEDIM=50      ! # EXPERIMENTAL POINTS
        PARAMETER NPDIM=2       ! # OF PARAMETERS TO FIT
        PARAMETER NRDIM=1700    ! # OF REGIONS IN FEATURE SPACE
        PARAMETER NHDIM=13      ! # OF HEURISTICS RULES
        PARAMETER NFDIM=2       ! # OF FEATURE SPACE VARIABLES
        DIMENSION Y(MEDIM)                  ! DEPENDENT VARIABLE ARRAY
        DIMENSION X(MEDIM)                  ! INDEPENDENT VARIABLE ARRAY
        DIMENSION W(MEDIM)                  ! WEIGHTS (1/VARIANCE) OF Y
        DIMENSION P(NPDIM)                  ! PARAMETER ARRAY
        DIMENSION PN(NPDIM)                 ! NEW PARAMETER ARRAY
        DIMENSION PT(NPDIM)                 ! TRUE PARAMETERS
        DIMENSION CENT(NFDIM,NRDIM)         ! REGION CENTER COORDINATES
        DIMENSION HT(NRDIM)                 ! AVERAGE PERFORMANCE
        DIMENSION NTR(NRDIM)                ! # OF TIMES IN REG USED IN IREG
        REAL*4 BP                           ! LARGEST PERFORMANCE
        REAL*4 SP                           ! SMALLEST PERFORMANCE
        REAL*4 AP                           ! AVERAGE PERFORMANCE IN IREG
        REAL*4 AVDT(NHDIM)                  ! AVERAGE DELTA
        REAL*4 FNT                          ! # OF TIMES IN REGION
        DIMENSION D(NPDIM,NHDIM)            ! DIRECTIONS
        DIMENSION IHR(NHDIM)                ! LAST USED RULES
        DIMENSION PLV(NHDIM)                ! PERFORMANCE OF LAST USED RULES
        DIMENSION IBEST(NHDIM)              ! BEST PERFORMANCE COUNTER
        DIMENSION RS(NFDIM)                 ! FEATURE COORDINATES
        DIMENSION PI(NPDIM)                 !# INITIAL PARAMETER ARRAY
        DIMENSION RSI(NFDIM)                !# INITIAL FEATURE ARRAY
        LOGICAL*1 TITC(70),SDATE(10),TDATE(10),STIME(8),TTIME(8)
        LOGICAL*1 ICON,IBEG,TBST
        COMMON /TDATA/PT
        COMMON /SEEDEP/ISEP1,ISEP2
        COMMON /SEEDSH/ISSH1,ISSH2
C
C   GET VERTION CHARACTER FOR FILE NAMES
C
        LOGICAL*1 HEAD(10),HEAD(12),FITL(12),LCHR(12),IVCHAR
        DATA HEAD/'H','E','A','D',' ',' ','N','L','X',' ',0/
        DATA HEAD/'H','E','H','D','H','Y',' ','N','L','X',' ',0/
        DATA FITL/'F','I','T','L','D','S',' ','N','L','X',' ',0/
        DATA LCHR/'L','C','H','R','V','E',' ','N','L','X',' ',0/
        OPEN(UNIT=1,NAME='VERSION.CHR',ACCESS='SEQUENTIAL',
        1  FORM='UNFORMATTED',TYPE='OLD',READONLY,RECORDTYPE='VARIABLE')
        READ(1)IVCHAR
        CLOSE (UNIT=1)
        HEAD(8)=IVCHAR
        HEAD(10)=IVCHAR
        FITL(10)=IVCHAR
        LCHR(10)=IVCHAR
```

```
C
C  DETERMINE DIRECTIONS
C
      DO 1 I=1,MNDIR
      Q(1,I)=COS(3.1415927*FLOAT(I-1)/FLOAT(MNDIR))
1     Q(2,I)=SIN(3.1415927*FLOAT(I-1)/FLOAT(MNDIR))
      IEOFLG=46
      IBCK=0
      MPA=MPDIR
      MEP=MEDIR
      MNE=MNDIR
      ISEP1=14199
      ISEP2=13960
      ISSH1=0223
      ISSH2=6209
      WRITE(5,1000)
1000  FORMAT(' ENTER (ST) TO START A NEW RUN',/,
     1       '      OR  1   TO RESTART AN OLD RUN : ',$)
      READ(4,2650)ISTART
      IF(ISTART.EQ.0)GO TO 30
C
C  RESTART OLD RUN
C
      OPEN(UNIT=1,NAME=HEAD,TYPE='OLD',ACCESS='SEQUENTIAL',
     1 FORM='UNFORMATTED',SHARED)
      READ(1)TIT
      READ(1)SDATE,STIME,TDATE,TTIME
      READ(1)NEP,NPA,NREST,NNE,NFE,NBFIT,NBREP,NNEG,NBF,NBR,TOFIT,
     1  ISEP1,ISEP2,ISSH1,ISSH2
      READ(1)PRE,RAD,RNIN,NIT,NIIT,IPRI,IPLT,ISEL,IUPD,
     1  LC,LOOP1,LOOP2
      CLOSE (UNIT=1)
      OPEN(UNIT=1,NAME=MEMO,TYPE='OLD',ACCESS='DIRECT',
     1 FORM='UNFORMATTED',SHARED)
      DO 20 II=1,NNEG
20    READ(1'II)UT,WTH,BP,SP,AP,AVBT,FNT,IBEST,
     1  (CENT(JJ,II),JJ=1,NFE)
      CLOSE (UNIT=1)
      NREST=NREST+1
      GO TO 49
C
C  NEW RUN
C
30    CALL TIME(STIME)
      CALL DATE(SDATE)
      WRITE(5,1100)
1100  FORMAT(' ENTER TITLE [TO CH.]')
      READ(4,1200)TIT
1200  FORMAT(70A1)
C
C  INITIALS
C
      PRE=1.E-4              ! RELATIVE PRECISION
      RAD=.05
      RNIN=.001
      NFE=NFDIR
      NIT=2000               ! MAX. # OF ITERATIONS
      NIIT=2000              ! MAX # OF INNER ITERATIONS
      NBREP=100
      NBFIT=50
      IPLT=0                 ! PLOT LEVEL
      IPRI=1                 ! PRINT LEVEL
      ISEL=1                 ! HEURISTIC SELECTION MODE (1-4)
      IUPD=1                 ! UPDATE FLAG (0 - NO UPDATING)
      LC=1
      NREST=0          ! NUMBER OF RESTARTS
      NBR=0            ! ***
      LOOP1=1          ! ***
```

```
C
C   OPEN FILES AND STORE INITIAL DATA
C
        IREC=18         ! FOR TITTLE
        OPEN(UNIT=1,NAME=HEAD,TYPE='NEW',ACCESS='SEQUENTIAL',
        1 FORM='UNFORMATTED',RECORDSIZE=IREC,SHARED)
        WRITE(1)TIT
        CALL DATE(TDATE)
        CALL TIME(TTIME)
        WRITE(1)SDATE,STIME,TDATE,TTIME
        WRITE(1)NEP,NPA,NREST,NME,NFE,NOFIT,NOREP,NRES,NBF,NBR,TNFIT,
        1  ISEP1,ISEP2,ISSN1,ISSN2
        WRITE(1)PRE,RAD,RMIN,WIT,HIIT,IPRI,IPLT,ISEL,IUP0,
        1  LC,LOOP1,LOOP2
        CLOSE (UNIT=1)
        IREC=(6*NME+2*NFE+9)/2
        OPEN(UNIT=1,NAME=MEAN,TYPE='NEW',ACCESS='DIRECT',
        1 FORM='UNFORMATTED',RECORDSIZE=IREC,SHARED)
        CLOSE (UNIT=1)
        IREC=(6*NPOIN+6*NFOIN+14+1)/2
        OPEN(UNIT=1,NAME=FITL,TYPE='NEW',ACCESS='SEQUENTIAL',
        1  FORM='UNFORMATTED',RECORDSIZE=IREC,SHARED)
        CLOSE (UNIT=1)
        IF(LC.EQ.0)GO TO 40
        OPEN (UNIT=1,NAME=LCOR,TYPE='NEW',ACCESS='DIRECT',
        1 SHARED,RECORDSIZE=10)
        CLOSE (UNIT=1)
C
C   RESTART LOOP
C
40      TNFIT=0.
        TIME=0.
        NRES=0
        NBF=0   ! ***
        IF(LC.EQ.0)GO TO 50
        NBR=NBR+1       ! ***
45      WRITE(5,1300)NBR
1300    FORMAT(' NBR = ',I4)
        IF(LC.NE.0 .AND. NBR.GT.NBREP)GO TO 910 ! ***
C
C  GET EXPERIMENTAL POINTS
C
50      NBF=NBF+1       ! ***
        IF(LC.NE.0 .AND. NBF.GT.NOFIT)GO TO 40  ! ***
        ISEP1I=ISEP1
        ISEP2I=ISEP2
        CALL GETEP(NEP,Y,X,U)
        TNFIT=TNFIT+1
C
C  SET FUNCTION PARAMETERS
C
60      CALL GETP0(NPA,P,PRI,PDA,IPF,1)
        ISFL=0
        TIME=0.
C
C  CALCULATE INITIAL SSD
C
        CALL STATUS(NEP,X,Y,R,P,SSD0,NFE,RO,0)
        IF(IPRI.GE.1)WRITE(6,1350)TNFIT,NEP,SSD0,RO,
        1  ((J,P(J)),J=1,NPA)
1350    FORMAT(//,'  FIT NUMBER : ',F10.0,/,' NUMBER OF POINTS =',
        1  I4,/,' INITIAL WEIGHTED SUM OF DEVIATIONS SQUARED =',G10.4,//,
        1  ' FEATURES : ',<NFE>(F7.4,2X),//,
        1  ' PAR. #   INIT.VALUE',//
        1  20(4X,I2,3W,G13.7,/))
        SSOI=SSD0
        DO 65 I=1,NFE
65      ROI(I)=RO(I)
```

```
      DO 97 I=1,NPA
97    PI(I)=P(I)
      SST=0.0
      NITI=1
      IBEG=0
      IROLD=0
      DELY0=.1
      NNN=0
C
C*****************************************************************************
C                                                                             *
C                    OUTER ITERATION LOOP                                     *
C                                                                             *
150   DO 900 IT=NIT1,NIT
      IF(IPLT.EQ.-1 .AND. ISFL.EQ.0)CALL SPLACNFE,RO,NPA,P,0)
      IF(IPLT.EQ.-1)ISFL=1
      IF(IT.EQ.1)IROLD=0
      CALL READEF(IEVFLG,IEFLAG)
      IF(IEFLAG.EQ.0)GO TO 160
      CALL CLREF(IEVFLG)
      GO TO 910
C
C   IN THE OLD REGION ?
C
160   CONTINUE
      IF(IBEG.LT.0)GO TO 350
      IF(NREG.EQ.0)GO TO 250
C
C  CHECK IF IN AN EXISTENT REGION
C
      RAD2=0.
      DO 170 II=1,NFE
170   RAD2=RAD2+RO(II)*RO(II)
      RAD1=RADK*SQRT(RAD2)+RMIN
      RAD2=RAD1**2
      DIST0=RAD2+.1
      IREG=-1
      DO 190 IR=1,NREG
      DIST=0.0
      DO 180 IFE=1,NFE
      IF(ABS(RO(IFE)-CENT(IFE,IR)).GT.RAD1)GO TO 190
      DIST=DIST+(RO(IFE)-CENT(IFE,IR))**2
180   IF(DIST.GT.RAD2)GO TO 190
      IF(DIST.GE.DIST0)GO TO 190
      IREG=IR
      DIST0=DIST
190   CONTINUE
      IF(IREG.EQ.IROLD)GO TO 300
      OPEN(UNIT=1,NAME='MEM0',TYPE='OLD',ACCESS='DIRECT',
     1 FORM='UNFORMATTED',SHARED)
      IF(IPRI.GE.4 .AND. IROLD.GT.0)WRITE(6,1400)IROLD,BP,SP,AP,FNT,
     1  (CNT(II),NTN(II),ANDT(II)),II=1,NNE)
1400  FORMAT(' REG=',I3,' BP=',F7.3,' SP=',F7.3,' AP=',F7.3,' FNT=',F7.0,
     1  /,(' NT=',F7.3,' NTN=',I5,' ANDT=',F7.5))
      IF(IROLD.GT.0)WRITE(1'IROLD)NT,NTN,BP,SP,AP,ANDT,FNT,IBEST,
     1  (CENT(II,IROLD),II=1,NFE)
C
C   RETRIEVE INFO FROM MEMORY
C
      IF(IREG.GT.0)READ(1'IREG)NT,NTN,BP,SP,AP,ANDT,FNT,IBEST
      IF(IPRI.GE.4 .AND. IREG.GT.0)WRITE(6,1400)IREG,BP,SP,AP,FNT,
     1  (CNT(II),NTN(II),ANDT(II)),II=1,NNE)
      CLOSE (UNIT=1)
      IF(IREG.GT.0)GO TO 300
C
C  CREATE A NEW REGION
C
250   IF(NREG.LT.MAXIN)GO TO 270
```

```
      WRITE(6,1500)
1500  FORMAT(' NO MORE MEMORY SPACE')
      GO TO 910
210   IF(IPRI.GE.3)WRITE(6,1600)NREG+1,A0
1600  FORMAT(' CREATING REGION ',I3,' AT: ',G13.7,<NFE>(' ; ',G13.7))
      NREG=NREG+1
      IREG=NREG
      SP=1.
      BP=0.
      DO 200 IFE=1,NFE
200   CENT(IFE,IREG)=A0(IFE)
      FNT=0          ! NUMBER OF TIMES IN THIS REGION
      DO 290 IN=1,NNE
      NTR(IN)=0.
      AVDT(IN)=0.0
      IBEST(IN)=0
290   NT(IN)=0.
C
C  FIND THE BEST HEURISTIC
C
300   IF(IT.EQ.1)IREGI=IREG
      INB=0
      IF(IBCK.EQ.0 .AND. ISEL.NE.4)GO TO 350
      SSBO=SSBO+.1
      DO 330 IN=1,NNE
      IF(NNB.EQ.0)GO TO 320
      DO 310 II=1,NNB
310   IF(IN.EQ.INB(II))GO TO 330
      DELTA=.1
      IF(FNT.GT.NNE)DELTA=AVDT(IN)*(FNT+FLOAT(NNE))/FNT
320   CALL SBGOAL(NEP,X,Y,U,PRE,DELTA,NPA,P,PN,D(1,IN),IPRI,NFE,
     1  A0,INNER,NIIT,SS,SSD,SSDO,ISBGU)
      IF(SSB.LT.SSBO)INB=IN
      IF(SSD.LT.SSDO)SSDO=SSD
330   CONTINUE
      IF(IBCK.GT.0)IBEST(INB)=IBEST(INB)+1
C
C  SELECT A HEURISTIC
C
350   IF(ISEL.GT.3)GO TO 360
C--350  IF(FNT.GE.NNE .AND. ISEL.GT.3)GO TO 360
      FFNT=FNT
      IF(IREG.EQ.IREGLO .AND. FNT.GT.NNE)FFNT=NNE
      CALL SELHEU(NNB,INB,NNE,NPA,B,FFNT,NTR(1),NT,
     1  NP,BP,SP,IPRI,JSVER,IN,ISEL)
      GO TO 400
360   IN=INB
C
C   GET MINIMUM SSD USING HEURISTIC
C
400   DELTA=.1
      IF(FNT.GT.NNE)DELTA=AVDT(IN)*(FNT+FLOAT(NNE))/FNT
      CALL SBGOAL(NEP,X,Y,U,PRE,DELTA,NPA,P,PN,D(1,IN),IPRI,NFE,
     1  A0,INNER,NIIT,SS,SSB,SSBO,ISBGU)
      TIME=TIME+FLOAT(INNER)
C
C  CALCULATE PERFORMANCE
C
      FIPS=(SSDO-SSB)/SSB
C
C  UPDATE WEIGHTS
C
      IF(IUPD.EQ.0)GO TO 740
C--   IF(FNT.EQ.NNE)GO TO 540
      IF(FIPS.GT.BP)BP=FIPS
      IF(FIPS.LT.SP)SP=FIPS
      BP=(FNT*BP+FIPS)/(FNT+1.)
      AVDT(IN)=(FLOAT(NTR(IN))*AVDT(IN)+SS)/(FLOAT(NTR(IN))+1.)
```

```
      IF(FNT.EQ.0)WUT(IN)=SS
      TIMES=WTN(IN)
      WT(IN)=(TIMES*WUT(IN)+FIPS)/(TIMES+1.)
      IF(WTN(IN).LT.32000)WTN(IN)=WTN(IN)+1.
540   FNT=FNT+1.
740   CONTINUE
C--740  IF(FNT.LE.NHE)GO TO 750
C
C  KEEP TRACK OF USED HEURISTICS
C
      DO 750 II=NHE,2,-1
      PLU(II)=PLU(II-1)
750   INU(II)=INU(II-1)
      PLU(1)=FIPS
      INU(1)=IN
C
C  CHECK IF AT THE SAME POINT
C
      IF(ABS(SS).GT.PRE)GO TO 760
      IF(IPRI.GE.7)WRITE(6,1700)IN
1700  FORMAT(' IT IS AT A MINIMUM ACCORDING TO RULE:',I3)
      NNU=NNU+1
      GO TO 770
764   NNU=1
770   IREG=IREG
      DO 780 J=1,NPR
780   P(J)=PN(J)
      IF(IPLT.EQ.-1)CALL SPLA(NFE,AO,NPR,P,2)
      CALL STATUS(NEP,X,Y,W,P,SSDO,NFE,AO,2)
790   ICON=32
      IF(NNU.GT.1)ICON=35
      TEST=32
      IF(IN.EQ.INO)TEST=43
      IREG=32
      IF(FNT.LE.NHE)IREG=42
      IF(IPRI.GE.1)WRITE(6,1000)IT,SSDO,IREG,IBEG,FNT,
     1  TEST,IN,SS,ICON,FIPS
1000  FORMAT(' ',I4,' RE=',F10.3,
     1 ' RE=',I4,A1,'(',F8.0,')',A1,'H=',I2,' SS=',FT.5,A1,' IP=',F9.4)
      IF(IPRI.GE.2)WRITE(6,1900)INNER,SSO,
     1  ((J,PN(J),(P(J)-PN(J))),J=1,NPR)
1900  FORMAT(' INNER ITERATIONS:',I5,'  SSD = ',G13.7,/,
     1  ' PAR.#    NEW VAL.    LAST CHANGE',/
     1  ,99(3X,I2,2X,G13.7,1X,G13.7,/))
C--   IF(FNT.LE.NHE)IBEG=-1
      IF(NNU.EQ.NHE)GO TO 810
      SST=SST+ABS(SS)
840   CONTINUE
C                                                                       *
C               END OF OUTER LOOP                                       *
C                                                                       *
C***********************************************************************
C
      WRITE(6,2000)NIT
2000  FORMAT(' NOT CONVERGED AFTER ',I4,' ITERATIONS')
      WRITE(5,2000)NIT
      GO TO 910
C
C  EXIT ON CONVERGENCE
C
810   IF(IPRI.GE.0)WRITE(6,2400)TNFIT,IT,TIME,SSD,NAE6,IAE6
2400  FORMAT(/,
     1  ' A FIT ',F7.0,' CONVERGED AFTER ',I3,' OUTER & ',
     1  F5.0,'INNER ITERATIONS',/,
     1  ' SUM OF THE SQUARE OF DEVIATIONS :',G13.7,/,
     1  ' TOTAL # OF REGIONS: ',I4,'   CONVERGED IN REGION ',I4,/)
      IRUN=0
      DO 840 I=1,NEP
```

```
      ISI=SIGN(1.,CY(I)-FUNCCX(I),P)))
      IF(I.EQ.1)GO TO 830
      IF(ISI.EQ.ISAVE)GO TO 840
830   ISAVE=ISI
      IRUN=IRUN+1
840   CONTINUE
      RUN=FLOAT(NEP)/2.+1.
      STRUN=SQRT(FLOAT(NEP)*FLOAT(NEP-2)/(4.*FLOAT(NEP-1)))
      IF(IPRI.GE.0)WRITE(6,2500)IRUN,RUN,STRUN
2500  FORMAT(' # OF RUNS:',I4,'  EXPECTED # RUNS:',F5.1,' +/- ',F5.2)
      IF(IPRI.GE.0)WRITE(6,2600)(J,P(J),PT(J)),J=1,NPQ)
2600  FORMAT('  PAR.#        VALUE          TRUE VAL',/,
     1  20(3X,I3,4X,G13.7,2X,G13.7,/))
      OPEN(UNIT=1,NAME=HEAD,TYPE='UNKNOWN',ACCESS='SEQUENTIAL',
     1 FORM='UNFORMATTED',RECORDSIZE=IREC,SHARED)
      WRITE(1)TIT
      CALL DATE(TDATE)
      CALL TIME(TTIME)
      WRITE(1)SDATE,STIME,TDATE,TTIME
      WRITE(1)NEP,NPR,NREST,NWE,NFE,NBFIT,NBREP,NRE6,NBF,NBR,TNFIT,
     1  ISEP1,ISEP2,ISSN1,ISSN2
      WRITE(1)NPRE,NAB,NRUN,NIT,NIIT,IPRI,IPLT,ISEL,IUMB,
     1  LC,LOOP1,LOOP2
      CLOSE (UNIT=1)
      CONTINUE
      OPEN(UNIT=1,NAME=NEW0,TYPE='OLD',ACCESS='DIRECT',
     1 FORM='UNFORMATTED',SHARED)
      IF(IPRI.GE.4)WRITE(6,1000)IRE6,BP,SP,AP,FNT,
     1  ((NT(II),NTN(II),AVDT(II)),II=1,NWE)
      WRITE(1'IRE6)NT,NTN,BP,SP,AP,AVDT,FNT,IBEST,
     1  (CENT(II,IRE6),II=1,NFE)
      CLOSE (UNIT=1)
      OPEN(UNIT=1,NAME=FITL,TYPE='OLD',ACCESS='APPEND',
     1  FORM='UNFORMATTED',SHARED)
      NIIT=IFIX(TIME)
      WRITE(1)IRE6I,ISEP1I,ISEP2I,PI,PT,AQI,SSDI,IRE6,P,AQ,SSD,IRUN,
     1  IT,NIIT,SST,NRE6
      CLOSE (UNIT=1)
      IF(LC.EQ.0)GO TO 820
      OPEN (UNIT=1,NAME=LCOR,TYPE='OLD',ACCESS='DIRECT',
     1 SHARED,RECORDSIZE=10)
      STSS=0.0
      DO 825 JJ=1,NPQ
825   STSS=STSS+(P(JJ)-PI(JJ))**2
      STSS=SQRT(STSS)
      SST=SST/STSS
      NTNS=0
      AVIT=0
      AVIT2=0
      AVIIT=0
      AVIIT2=0
      ASST=0
      ASST2=0
      IF(NBR.GT.1)READ(1'NBF)NTNS,AVIT,AVIT2,AVIIT,AVIIT2,ASST,ASST2
      NTNS=NTNS+1
      AVIT=(AVIT*FLOAT(NTNS-1)+FLOAT(IT))/FLOAT(NTNS) ! ***
      AVIT2=(AVIT2*FLOAT(NTNS-1)+(FLOAT(IT)**2))/FLOAT(NTNS) ! ***
      AVIIT=(AVIIT*FLOAT(NTNS-1)+TIME)/FLOAT(NTNS) ! ***
      AVIIT2=(AVIIT2*FLOAT(NTNS-1)+(TIME**2))/FLOAT(NTNS) ! ***
      ASST=(ASST*FLOAT(NTNS-1)+SST)/FLOAT(NTNS) ! ***
      ASST2=(ASST2*FLOAT(NTNS-1)+(SST**2))/FLOAT(NTNS) ! ***
      WRITE(1'NBF)NTNS,AVIT,AVIT2,AVIIT,AVIIT2,ASST,ASST2
      CLOSE (UNIT=1)
820   CONTINUE
900   IF(LOOP1.GT.0)GO TO 50
      IF(LOOP2.GT.0)GO TO 80
910   NIIT=IT
      WRITE(5,2610)TIT
```

```
2610    FORMAT(/,1X,7A1,//,' ENTER:  0- TO GET NEW DATA POINTS',/,
       1        '          1- TO CHANGE PRECISION, PRINT-PLOT LEVEL...',/,
       1        '          2- TO TO CHANGE PARAMETERS',/,
       1        '          3- TO CONTINUE ITERATION',/,
       1        '          4- TO CHANGE DEVICE FOR LU6',/,
       1        '          5- TO START OVER',/,
       1        '          6- TO STOP  ..................?',$)
       READ(4,2650,ERR=910)IANS
2650   FORMAT(I15)
       GO TO (50,920,80,150,970,40,990),IANS+1
       GO TO 910
920    WRITE(5,2700)TIT,RAD,XMIN,PRE,NIT,IPRI,IPLT,LOOP1,LOOP2,IBCK,
       1  ISEL,IUPD,MIIT
2700   FORMAT(/,1X,70A1,/
       1  ,' 1. RAD  =',G11.5,'2. XMIN =',G11.5,2X,'3. PRE =',G11.5,/,
       1        ' 4. NIT  =',I4,7X,'5. IPRI =',I4,9X,'6. IPLT=',I4,/,
       1        ' 7. LOOP1=',I4,7X,'8. LOOP2=',I4,9X,'9. IBCK=',I5,/,
       1        ' 10.ISEL =',I4,7X,'11.IUPD =',I4,9X,'12.MIIT =',I4,/,
       1        ' ENTER PARAMETER # TO CHANGE:',$)
       READ(4,2650,ERR=920)IANS
       IF(IANS.LT.1 .OR. IANS.GT.12)GO TO 920
930    WRITE(5,2800)
2800   FORMAT(' ENTER NEW VALUE: ',$)
       IF(IANS.EQ.1)READ(4,2900,ERR=930)RAD
       IF(IANS.EQ.2)READ(4,2900,ERR=930)XMIN
       IF(IANS.EQ.3)READ(4,2900,ERR=930)PRE
       IF(IANS.EQ.4)READ(4,2650,ERR=930)NIT
       IF(IANS.EQ.5)READ(4,2650,ERR=930)IPRI
       IF(IANS.EQ.6)READ(4,2650,ERR=930)IPLT
       IF(IANS.EQ.7)READ(4,2650,ERR=930)LOOP1
       IF(IANS.EQ.8)READ(4,2650,ERR=930)LOOP2
       IF(IANS.EQ.9)READ(4,2650,ERR=930)IBCK
       IF(IANS.EQ.10)READ(4,2650,ERR=930)ISEL
       IF(IANS.EQ.11)READ(4,2650,ERR=930)IUPD
       IF(IANS.EQ.12)READ(4,2650,ERR=930)MIIT
2900   FORMAT(F15.0)
       GO TO 910
970    WRITE(5,3000)
3000   FORMAT(' ENTER DEVICE [ EX: TTY0,C6]]] :',$)
       READ(4,3100,ERR=970)II,III,IIII
3100   FORMAT(A2,I10,I10)
       IF(IIII.EQ.0)IIII=6
       CALL ASMLUN(IIII,II,III)
       GO TO 910
990    STOP
       END
```

```
      SUBROUTINE STATUS(NEP,X,V,W,P,SSD,NFE,RO,IFL)
C
C   IFL=0  CALCULATE EVERYTHING
C      =1  CALCULATE SSD ONLY
C      =2  CALCULATE SSD, RO AND R1
C
      DIMENSION X(NEP),V(NEP),W(NEP),P(1),RO(NFE)
      IF(IFL.GT.0)GO TO 200
      XBAR=0.0
      WBAR=0.0
      AREA=0.0
      DO 100 I=1,NEP
      XBAR=XBAR+X(I)
      WBAR=WBAR+SQRT(V(I))
      AREA=AREA+V(I)*SQRT(W(I))
100   CONTINUE
      XBAR=XBAR/FLOAT(NEP)
      XBAR2=XBAR*XBAR
200   SSD=0.0
      DO 300 I=1,NEP
      FI=FUNC(X(I),P)
      DEV=(V(I)-FI)
      SSD=SSD+DEV*DEV*W(I)
      IF(IFL.EQ.1)GO TO 300
      WDEV=DEV*SQRT(W(I))
      DO 290 I1=1,NFE
      IF(I.EQ.1)RO(I1)=0.0
290   RO(I1)=RO(I1)+WDEV*(X(I)-XBAR)**(I1-1)
300   CONTINUE
      IF(IFL.EQ.1)RETURN
      RO(1)=RO(1)/AREA
      RO(2)=4.*WBR(2)/(AREA*XBAR)
      IF(NFE.GT.2)RO(3)=1.333*RO(3)/(AREA*XBAR2)
      RETURN
      END
```

```
C++-
C       Module Name: FDLPIS1
C
C       Version X01.00  Last edit: 30-JUN-83 15:05
C       Status: Development/Debugging
C
C       Revision History:
C
C               Version X01.00 23-JUN-83 10:37 - 30-JUN-83 15:05
C               Created by: E.MACHADO
C
C++-
        SUBROUTINE SETEP(N,Y,X,M)
        DATA IS1,IS2/0,0/
        DIMENSION Y(N),X(N),U(N)
        DIMENSION PT(2)
        COMMON /TDATA/PT
        M=50
        PT(1)=.9+.2*RAN(IS1,IS2)
        PT(2)=.9+.2*RAN(IS1,IS2)
        BL=10000.
        DO 100 I=1,N
        X(I)=.02*FLOAT(I)
        X1=X(I)
        X2=X1*X1
        Y(I)=FUNC(X(I),PT)
70      XR1=RAN(IS1,IS2)
        IF(XR1.LE.0.)GO TO 70
        XR2=RAN(IS1,IS2)
        XR=SQRT(-2.*ALOG(XR1))*COS(6.28318*XR2)
        FACT=1.+XR/SQRT(BL)
        U(I)=BL/(Y(I)*Y(I))
        Y(I)=FACT*Y(I)
100     CONTINUE
        RETURN
        END
```

```
      SUBROUTINE GETPA(K,P,PMI,PMA,IPF,IFL)
      DIMENSION P(2),PMI(1),PMA(1),IPF(1)
      DATA IS1,IS2/0,0/
      P(1)=1. !.5+RAN(IS1,IS2)
      P(2)=1. !.5+RAN(IS1,IS2)
      IF(IFL.GT.0)RETURN
      K=2
      DO 100 J=1,K
      WRITE(5,1000)J
1000  FORMAT(' ENTER VALUE, MIN., MAX. & FLAG FOR PAR. # ',I2,' :',$)
100   READ(4,1100)P(J),PMI(J),PMA(J),IPF(J)
1100  FORMAT(3F15.0,I15)
      RETURN
      END
```

```
      FUNCTION FUNC(X,P)
      DIMENSION P(2)
      A0=1.
      B1=1.
      A2=2.
      A1=P(1)
      A02=A0*A0
      B12=B1*B1
      A12=A1*A1
      AM=P(2)
      X1=AM*2.*3.1415927
      X2=X1*X1
      FUNC=(A02+X2*B12)*AM/((A0-X2)*(A0-X2)+A2*B12)
      RETURN
      END
```

```
      SUBROUTINE SPLA(NFE,AO,NPA,P,IFL)
      DIMENSION AO(NFE),P(NPA)
      IF(IFL.GT.0)GO TO 100
      OPEN (UNIT=1,NAME='SPLA.DLZ',TYPE='NEW',ACCESS='SEQUENTIAL',
     1  FORM='UNFORMATTED',RECORDSIZE=(NPA+NFE+1)/2)
      WRITE(1)NFE,NPA
      GO TO 200
  100 OPEN (UNIT=1,NAME='SPLA.DLZ',TYPE='OLD',ACCESS='APPEND',
     1  FORM='UNFORMATTED',RECORDSIZE=(NPA+NFE+1)/2)
  200 WRITE(1)AO,P
      CLOSE(UNIT=1)
      RETURN
      END
```

```
CNH-
C       Module name: FBLPIS2
C
C       Version X01.00  Last edit: 29-JUN-83 19:01
C       Status: Development/Debugging
C
C       Revision history:
C
C               Version X01.00 23-JUN-83 10:24 - 29-JUN-83 19:01
C               Created by: E.MACHADO
C
CNH- ,
        SUBROUTINE SELHEU(NHU,IHU,NHDIN,KDIN,D,FNT,
       1  NTH,WT,AP,BP,SP,IPRI,IPER,IN,ISEL)
C
C   THIS SUBROUTINE SELECTS A HEURISTIC ( IH )
C
C   ISEL = 1  - NORMAL SELECTION
C        = 2  - RANDOM (EQUAL PROBABILITIES) SELECTION
C        = 3  - NEXT RULE SELECTION
C
        PARAMETER MHDIN=25
        DIMENSION IHU(MHDIN),NTH(MHDIN),WT(MHDIN),D(KDIN,MHDIN)
        DIMENSION SELH(MHDIN),IRUL(MHDIN)
        COMMON /SEEDS0/IS1,IS2
        IPER=1
        IF(NHDIN.GT.MHDIN)WRITE(6,1000)
1000    FORMAT(' DIMENSION ERROR IN SELHEU ')
        IF(NHDIN.GT.MHDIN)STOP
        IF(ISEL.EQ.3)GO TO 300
        IF(ISEL.EQ.2)GO TO 410
        IF(FNT.GE.NHDIN)GO TO 410
C ** BEFORE USING ALL RULES **
300     IH=IHU(1)+1
        IF(IN.LE.0)IH=1
390     IH=MOD(IH-1,NHDIN)+1
        IF(ISEL.EQ.3)RETURN
        IF(NTH(IH).EQ.0)RETURN
        IH=IH+1
        GO TO 390
C **  AFTER ALL RULES HAVE BEEN USED **
C
C ** STOCHASTIC SELECTION **
410     JHU=0
        FK=FNT/FLOAT(NHDIN)
        SUM=0.
        DO 470 II=1,NHDIN
        IF(NHU.EQ.0)GO TO 430
        DO 420 III=1,NHU
420     IF(IHU(III).EQ.II)GO TO 470
430     SHD=0
460     JHU=JHU+1
        IF(ISEL.NE.2)SELH(JHU)=(1.-SHD)*(WT(II)-SP+1/FNT)**FK
        IF(ISEL.EQ.2)SELH(JHU)=1.
        IRUL(JHU)=II
        SUM=SUM+SELH(JHU)
470     CONTINUE
        RAND=SUM*RAN(IS1,IS2)
        IF(IPRI.EQ.3)WRITE(6,1300)NHU,JHU,SUM,RAND,
       1  ((JJ,IHU(JJ),IRUL(JJ),WT(JJ),SELH(JJ)),JJ=1,NHDIN)
        CP=0.0
        DO 480 II=1,JHU
        IH=IRUL(II)
        CP=CP+SELH(II)
        IF(RAND.LE.CP)RETURN
480     CONTINUE
        WRITE(5,1300)NHU,JHU,SUM,RAND,
       1  ((JJ,IHU(JJ),IRUL(JJ),WT(JJ),SELH(JJ)),JJ=1,NHDIN)
```

```
1300    FORMAT(' NMO =',I5,' IMU=',I5,' SUM=',G12.5,' RAND=',G12.5,/,
        1  100(1X,I2,2X,
        1  ' IMU:',I5,' IMUL:',I5,' MT:',G12.5,' SELN:',G12.5,/))
        WRITE(5,2010)
2010    FORMAT('NMO RULE SELECTED. ENTER RULE NUMBER : ')
        READ(4,2020)IM
2020    FORMAT(I15)
        RETURN
        END
```

```
      SUBROUTINE SDGOAL(Q,X,Y,W,PRE,AUGT,KDIM,P,PN,D,IPRI,NFDIM,RO,
     1  INNER,NIIT,SS,SSQ,SSQO,IVER)
C
C  GET MINIMUM SSD USING HEURISTIC
C
      DIMENSION P(KDIM)
      DIMENSION PN(KDIM)
      DIMENSION D(KDIM)
      DIMENSION RO(NFDIM)
      DIMENSION Q(3),S(3)
      IVER=1
      INNER=0
      DELTA=AUGT
      IF(DELTA.LT.PRE)DELTA=1.1*PRE
      S(1)=0
      Q(1)=SSQO
      S(2)=DELTA
      DO 510 J=1,KDIM
510   PN(J)=P(J)+S(2)*D(J)
      CALL STATUS(N,X,Y,W,PN,SSD,NFDIM,RO,1)
      Q(2)=SSD
      IF(Q(2).LE.Q(1))S(3)=S(2)+DELTA
      IF(Q(2).GT.Q(1))S(3)=S(1)-DELTA
      DO 520 J=1,KDIM
520   PN(J)=P(J)+S(3)*D(J)
      CALL STATUS(N,X,Y,W,PN,SSD,NFDIM,RO,1)
      Q(3)=SSD
C**********************************
C
C     INNER LOOP
C
C  REORDER
C
530   INNER=INNER+1
      DO 560 II=1,2
      IMIN=II
      DO 550 III=II+1,3
550   IF(S(IMIN).GT.S(III))IMIN=III
      IF(IMIN.EQ.II)GO TO 560
      SS=S(II)
      QQ=Q(II)
      S(II)=S(IMIN)
      Q(II)=Q(IMIN)
      S(IMIN)=SS
      Q(IMIN)=QQ
560   CONTINUE
      IQMAX=1
      IQMIN=1
      DO 570 II=2,3
      IF(Q(IQMAX).LT.Q(II))IQMAX=II
570   IF(Q(IQMIN).GT.Q(II))IQMIN=II
      DELTA=S(3)-S(1)
      D1=(Q(2)-Q(1))/(S(2)-S(1))
      D2=(Q(3)-Q(2))/(S(3)-S(2))
      IF(D2.LE.D1)GO TO 580
      S1=(S(1)+S(2))/2.
      S2=(S(2)+S(3))/2.
      SS=(D2*S1-D1*S2)/(D2-D1)
      GO TO 590
580   IF(Q(3).NE.Q(1))SS=S(1)-Q(1)*(S(3)-S(1))/(Q(3)-Q(1))
      IF(Q(3).EQ.Q(1))SS=S(IQMIN)
590   IF(S(1)-SS.GT.DELTA)SS=S(1)-DELTA
      IF(SS-S(3).GT.DELTA)SS=S(3)+DELTA
      DO 600 J=1,KDIM
600   PN(J)=P(J)+SS*D(J)
      CALL STATUS(N,X,Y,W,PN,SSD,NFDIM,RO,1)
      QQ=SSD
      IF(IPRI.GE.9)WRITE(6,1450)(S(JJ),JJ=1,3),SS,(Q(JJ),JJ=1,3),QQ
```

```
1450    FORMAT('     S1          S2          S3          SS',//,
        1  4(1X,G13.7,1X),/,4(1X,G13.7,1X))
C       IF(IPLT.GE.5)CALL DISPLA(N,V,X,P,2)
        IF(QQ.GE.Q(IQMAX))GO TO 660
C
C  CHECK CONVERGENCE
C
        IF(INNER.GE.NIIT)GO TO 660
        IF(ABS(SS-S(IQMIN)).LT.PRE/10.)GO TO 660
        IF(ABS(SS-S(IQMIN)).LT.PRE .AND. DELTA.LT.20.*PRE)GO TO 660
        GO TO (630,640,650),IQMIN
630     S(3)=SS
        Q(3)=QQ
        GO TO 530
640     IF(SS.LT.S(2))GO TO 630
650     S(1)=SS
        Q(1)=QQ
        GO TO 530
C
C  CONVERGED INNER ITERATION
C
C         END INNER LOOP
C
C**********************************
660     IF(QQ.LT.Q(IQMIN))GO TO 680
        SS=S(IQMIN)
        SSQ=Q(IQMIN)
680     DO 690 II=1,NDIM
690     PM(II)=P(II)+SS*D(II)
        RETURN
        END
```

APPENDIX E

LISTING OF THE "SDLPI" COMPUTER CODE

In this appendix, the listings of the SDLPI program is presented. This program is the final version of a series of programs used during the development of the Single-Direction Learning method described in Chapter 3, Section 3-5.

```
CHH-
C       Module name: Single Direction Learning Parameter Identification
C
C       Version X01.00  Last edit: 07-JUL-83 11:12
C       Status: Development/Debugging
C
C       Revision history:
C
C              Version X01.00 03-JUL-83 14:09 - 07-JUL-83 11:12
C              Created by: E.L.Machado
C
CHH-
C
C   SUBROUTINES CALLED:
C
C       GETEP - SUBROUTINE TO GET THE EXPERIMENTAL POINTS (USER SUPLIED)
C       GETPA - SUBROUTINE TO GET PARAMETER INFORMATION   (USER SUPLIED)
C       STATUS- EVALUATES THE SUM SQUARED DEVIATION AND FEATURES
C       SBOUND- FINDS THE MINIMUMS ALONG A DIRECTION
C       POPQ  - REMOVE FIRST ELEMENT FROM QUEUE
C       PLACEQ- PLACE AN ELEMENT IN QUEUE
C       REORDQ- REORDER QUEUE
C
C   FILES:      VERSION.CHA     -    input  : The character for file names (*)
C               HEAD.QL*        -    output : General information
C               MEMORY.QL*      -    output : Region memory
C               FITLOG.QL*      -    output : Log of all PI
C               LCURVE.QL*      -    output : Learning curve (if requested)
C
        PROGRAM SDLPI
C
        PARAMETER NDBIN=2       ! # DEPENDENT VARIABLES
        PARAMETER NEDIN=20      ! # EXPERIMENTAL POINTS
        PARAMETER NPBIN=3       ! # OF PARAMETERS TO FIT
        PARAMETER NRBIN=1000    ! # OF REGIONS IN FEATURE SPACE
        PARAMETER NBDIN=3       ! # OF BASIC DIRECTIONS
        PARAMETER NFBIN=4       ! # OF FEATURE SPACE VARIABLES
        PARAMETER MAXIT=800     ! MAXIMUM # ITERATIONS
        PARAMETER MAXST=1000    ! MAXIMUM # SSD CALC'S
        DIMENSION X(NEDIN)                      ! INDEPENDENT VARIABLE ARRAY
        DIMENSION Y(NEDIN,NDBIN)                ! EXPERIMENTAL DEPENDENT VAR. ARRAY
        DIMENSION FI(NDBIN)                     ! CALCULATED DEPENDENT VAR. ARRAY
        DIMENSION ISIGN(NDBIN)                  ! RUN SIGN (for # runs calculation)
        DIMENSION W(NEDIN,NDBIN)                ! WEIGHTS (1/VARIANCE) OF Y
        DIMENSION RSSD(NDBIN)                   ! DEPENDENT VARIABLE RELATIVE SSD
        DIMENSION IRUN(NBDIN)                   ! DEPENDENT VARIABLE # RUNS
        DIMENSION P(NPDIN)                      ! PARAMETER ARRAY
        DIMENSION PN(NPDIN)                     ! NEW PARAMETER ARRAY
        DIMENSION PT(NPDIN)                     ! TRUE PARAMETERS
        DIMENSION PMIN(NPDIN)                   ! MINIMUM PARAMETER LIMIT
        DIMENSION PMAX(NPDIN)                   ! MAXIMUM PARAMETER LIMIT
        DIMENSION PAVE(NPDIN)                   ! AVERAGE PARAMETER ARRAY
        DIMENSION PSDV(NPDIN)                   ! PARAMETER STANDARD DEVIATION
        DIMENSION PPRE(NPDIN)                   ! PARAMETER PRECISION
        DIMENSION IPF(NPDIN)                    ! not used
        VIRTUAL CENT(NFDIN,NRDIN)               ! REGION CENTER COORDINATES
        REAL*4 FNT                              ! # OF TIMES IN REGION
        DIMENSION BB(NPDIN,NBDIN+1)             ! BASIC DIRECTIONS
        DIMENSION AO(NFDIN)                     ! FEATURE COORDINATES
        DIMENSION PI(NPDIN)                     !N INITIAL PARAMETER ARRAY
        DIMENSION BOI(NFDIN)                    !N INITIAL FEATURE ARRAY
        DIMENSION IQUEUE(NBDIN+1)               ! DIRECTION QUEUE ARRAY
        DIMENSION QPRIOR(NBDIN+1)               ! DIRECTION PRIORITY ARRAY
        DIMENSION SPRIOR(NBDIN)                 ! DIRECTION PRIORITY FOR START (IT=1)
        DIMENSION CPRIOR(NBDIN)                 ! DIRECTION PRIORITY FOR IT>1
        DIMENSION BDEL(NBDIN)                   ! DIRECTIONAL DELTA
        LOGICAL*1 TIT(70),SDATE(10),TDATE(10),STIME(8),TTIME(8)
        LOGICAL*1 ICON
```

```
      VIRTUAL ISRE6(MAXST),SPAR(NPDIM,MAXST)
      DIMENSION AUPEC(NPDIM)
      DIMENSION BOSIGN(NBDIM)
      COMMON /TDATA/PT
      COMMON /SEEDEP/ISEP1,ISEP2
      COMMON /SEEDPA/ISPA1,ISPA2
      LOGICAL*1 HEAD(10),MEMO(12),FITL(12),LCUR(12),IDCHAR
      DATA HEAD/'H','E','A','D','.','D','L','A',' ',0/
      DATA MEMO/'M','E','M','O','R','Y','.','D','L','A',' ',0/
      DATA FITL/'F','I','T','L','E','S','.','D','L','A',' ',0/
      DATA LCUR/'L','C','U','R','V','E','.','D','L','A',' ',0/
      DATA PAPE/1.,1.,1./              ! DEFAUT VALUES
      DATA PMIN/.0001,.0001,.0001/     ! DEFAUT VALUES
      DATA PMAX/2.,2.,2./              ! DEFAUT VALUES
      DATA PPRE/.0001,.0001,.0001/     ! DEFAUT VALUES
C2    DATA PAPE/1.,1./              ! DEFAUT VALUES
C2    DATA PMIN/.0001,.0001/     ! DEFAUT VALUES
C2    DATA PMAX/2.,2./              ! DEFAUT VALUES
C2    DATA PPRE/.0001,.0001/     ! DEFAUT VALUES
C
C   GET VERTION CHARACTER FOR FILE NAMES
C
      OPEN(UNIT=1,NAME='VERSION.CHA',ACCESS='SEQUENTIAL',
     1  FORM='UNFORMATTED',TYPE='OLD',READONLY,RECORDTYPE='VARIABLE')
      READ(1)IDCHAR
      CLOSE (UNIT=1)
      HEAD(8)=IDCHAR
      MEMO(10)=IDCHAR
      FITL(10)=IDCHAR
      LCUR(10)=IDCHAR
C
C  DETERMINE BASIC DIRECTIONS AND INITIALIZE QUEUE
C
C##-  LOOP FOR NPOIN=NBDIM  -###
C##-  ie. basic directions = one parameter at a time -###
C
      DO 1 I=1,NBDIM
      DO 1 J=1,NBDIM
      BO(J,I)=0.0
1     IF(I.EQ.J)BO(J,I)=1.0
C
C##   LOOP FOR 2 PARAMETERS ONLY
C     DO 1 I=1,NBDIM
C     BO(1,I)=COS(3.1415927*FLOAT(I-1)/FLOAT(NBDIM))
C1    BO(2,I)=SIN(3.1415927*FLOAT(I-1)/FLOAT(NBDIM))
      JREG=1
      JRAD=0
      JRADRA=1
      IPAFL=4
      WRITE(5,1000)
1000  FORMAT(' ENTER <RT> TO START A NEW RUN',/,
     1       '    OR  1   TO RESTART AN OLD RUN : ',$)
      READ(4,2650)ISTART
      IF(ISTART.EQ.0)GO TO 30
C
C  RESTART OLD RUN
C
      OPEN(UNIT=1,NAME=HEAD,TYPE='OLD',ACCESS='SEQUENTIAL',
     1 FORM='UNFORMATTED',SHARED)
      READ(1)TIT
      READ(1)SDATE,STIME,TDATE,TTIME
      READ(1)NEP,NBV,NFA,NREST,NBD,NFE,NBFIT,NBREP,NNRES,NREG,NBF,NBR,
     1  TNFIT,ISEM1,ISEP2,ISPA1,ISPA2
      READ(1)PRE,RAD,RMIN,WIT,WASSD,IPRI,ISEL,IUPD,
     1  LC,LOOPC,LOOP2,JEUFLG,JRAD,NAVE,IPAFL
      READ(1)PAVE
      READ(1)PSDV
      READ(1)PMIN
```

```
      READ(1)PMAX
      READ(1)PPRE
      READ(1)BPRIOR
      READ(1)CPRIOR
      READ(1)DELOLO,DDEL
      CLOSE (UNIT=1)
      OPEN(UNIT=1,NAME=NEED,TYPE='OLD',ACCESS='DIRECT',
     1 FORM='UNFORMATTED',SHARED)
      DO 20 II=1,NREG
20    READ(1'II)FNT,AVVEC,AVMAG,
     1  ((CENT(JJ,II),JJ=1,NFE)
      CLOSE (UNIT=1)
      NREST=NREST+1
      NOIN=NOO+1
      IF(TNFIT.EQ.0.0)GO TO 40
      GO TO 49
C
C********************  NEW RUN  ****************************
C
30    CALL TIME(STIME)
      CALL DATE(SDATE)
      WRITE(5,1100)
1100  FORMAT(' ENTER TITLE (70 CH.)')
      READ(4,1200)TIT
1200  FORMAT(70A1)
C
C  INITIALS
C
      IEOFLG=45
      NOO=NOOIN
      NOIN=NOO+1
      ISEP1=31013
      ISEP2=-19579
      ISPA1=-1230
      ISPA2=17504
C     PRE=2.E-4              ! RELATIVE PRECISION
      RAD=.5                 ! REGION RADIUS IS GIVEN BY:
      RMIN=.5                ! (DISTANCE)*RAD+RMIN
      NEP=NEDIN
      NOV=NOOIN
      NFE=NFDIN
      NAMEG=NAGIN
      NIT=MAXIT              ! MAX. # OF ITERATIONS
      NNSSO=MAXIT            ! MAX # OF INNER ITERATIONS
      NOREP=50
      NBFIT=50
      IPRI=1                 ! PRINT LEVEL
      ISEL=2                 !** NOT USED **HEURISTIC SELECTION MODE (1-4)
      IUPD=1                 !** NOT USED **UPDATE FLAG (0 - NO UPDATING)
      LC=1
      NREST=0          ! NUMBER OF RESTARTS
      NOV=0            ! NUMBER OF REPETITIONS (FOR LEARNING CURVE)
      LOOPC=1          ! LOOP CONTROL ( GET A NEW SPECTRA EVERY PASS)
      DELOLO=100.
      CALL GETPA(NPA,P,PMIN,PMAX,PAVE,PSDV,NAVE,PPRE,EPF,0)
      DO 35 I=1,NPA
35    PAVE(I)=P(I)
C
C   OPEN FILES AND STORE INITIAL DATA
C
      OPEN(UNIT=1,NAME=NEAD,TYPE='NEW',ACCESS='SEQUENTIAL',
     1 FORM='UNFORMATTED',SHARED)
      WRITE(1)TIT
      CALL DATE(TDATE)
      CALL TIME(TTIME)
      WRITE(1)SDATE,STIME,TDATE,TTIME
      WRITE(1)NEP,NOV,NPA,NREST,NOO,NFE,NBFIT,NOREP,NOREG,NAEG,NBF,NOO,
     1  TNFIT,ISEP1,ISEP2,ISPA1,ISPA2
```

```
      WRITE(1)MPRE,MAD,MMIN,MIT,MMSSD,IPRI,ISEL,IWPD,
     1  LC,LOOPC,LOOP2,IEVFLG,JSAD,MAVE,IPAFL
      WRITE(1)MAVE
      WRITE(1)MSSD
      WRITE(1)MMIN
      WRITE(1)MMAX
      WRITE(1)MPRE
      WRITE(1)BPRIOR
      WRITE(1)CPRIOR
      WRITE(1)BBEL,DBEL
      CLOSE (UNIT=1)
      IREC=NFE+NPDIM+2
      OPEN(UNIT=1,NAME=MEM0,TYPE='NEW',ACCESS='DIRECT',
     1 FORM='UNFORMATTED',RECORDSIZE=IREC,SHARED)
      CLOSE (UNIT=1)
      IREC=(6*NPDIM+4*NFDIM+16+1)/2+1
      OPEN(UNIT=1,NAME=FITL,TYPE='NEW',ACCESS='DIRECT',
     1  FORM='UNFORMATTED',RECORDSIZE=IREC,SHARED)
      CLOSE (UNIT=1)
      IF(LC.EQ.0)GO TO 40
      OPEN (UNIT=1,NAME=LCUR,TYPE='NEW',ACCESS='DIRECT',
     1 SHARED,RECORDSIZE=10)
      CLOSE (UNIT=1)
C
C   START OVER LOOP
C
40    TNFIT=0.
      TNSSD=0.
      NREG=0
      DO 41 I=1,NBB
      BBEL(I)=1.
      BPRIOR(I)=1.
41    CPRIOR(I)=1.
      MAVE=0
      NBF=0    ! ***
      IF(LC.EQ.0)GO TO 50
      NBB=NBB+1        ! ***
49    IF(JDEG.NE.0)WRITE(6,1250)TIT,NPDIM,NDDIM,NEDIM,NBB,NFDIM,IEVFLG
1250  FORMAT(/,1X,70A1,/,
     1  ' # PARAMETER ',I3,'  # DEP. VAR. ',I2,'  # EXP. POI. ',I5,/,
     1  ' # BAS. DIR. ',I3,'  # FEATURES  ',I2,'  EVENT FLAG  ',I2,/)
      WRITE(6,1300)NBB
1300  FORMAT(' NBB = ',I4)
      IF(LC.NE.0 .AND. NBB.GT.NBMEP)GO TO 910 ! ***
C
C ------------------- LOOP --------------------
C
C  GET EXPERIMENTAL POINTS
C
50    JDES=0
      NBF=NBF+1        ! ***
      IF(LC.NE.0 .AND. NBF.GT.NBFIT)GO TO 40  ! ***
      ISEP1I=ISEP1
      ISEP2I=ISEP2
      ISPA1I=ISPA1
      ISPA2I=ISPA2
      TNFIT=TNFIT+1
      IEPFL=1
60    CALL GETEP(NEP,NDV,Y,X,W,IEPFL)
C
C  GET FUNCTION PARAMETERS
C
80    CALL GETPA(NPA,P,PMIN,PMAX,PAVE,PSDV,NAME,PPRE,IPF,IPAFL)
C
C  CALCULATE INITIAL SSD
C
      CALL STATUS(NEP,NDV,X,Y,W,P,SSD0,RSSD,NFE,ND,0)
      TNSSD=1.
```

```
      RDIST=0.
      DO 81 I=1,NFE
81    RDIST=RDIST+RD(I)*RD(I)
      RDIST=SQRT(RDIST)
      IF(IPRI.GE.1)WRITE(6,1350)NRF,SSD0,RSS0,RDIST,RB,P,PAVE
1350  FORMAT(/,
     1  '  FIT # : ',I5,' SSD= ',G13.7,/,(NDV)(1X,FT.5),/,
     1  '                 DIST= ',FT.2,/,(NPE)(1X,FT.2),/,
     1  ' INI. PAR. ',(NPR)(2X,G13.7),/,
     1  ' AVE. PAR. ',(NPR)(2X,G13.7),/)
      SSDI=SSD0
      DO 85 I=1,NFE
85    RDI(I)=RD(I)
      DO 87 I=1,NPR
87    PI(I)=P(I)
      CPMAX=0.
      DO 88 I=1,NBD
      IF(CPRIOR(I).GT.CPMAX)CPMAX=CPRIOR(I)
88    NBSIGN(I)=1
      NQUEUE=0
      DO 89 I=1,NBD
      QPRIOR(I)=CPRIOR(I)/CPMAX
      CALL PLACEQ(I,0,IQUEUE,NQUEUE,NBIN,QPRIOR,NBD,IERR)
      IF(IPRI.GE.5 .AND. NQUEUE.GT.0)WRITE(5,990)1,NQUEUE,
     1 ((II,IQUEUE(II),QPRIOR(IQUEUE(II))),II=1,NQUEUE)
990   FORMAT(' ',I1,2X,'QUEUE ',I2,/,(1X,I4,2X,I4,2X,F15.4))
89    IF(IERR.GT.0)WRITE(5,999)IERR,I,NQUEUE
999   FORMAT(' QUEUE ERROR ',I2,2X,I2,2X,I4)
      QPRIOR(NBIN)=1.
      SST=0.0
      NITI=1
      IROLD=0
C
      DELOLD=DELOLD*3.
      ICON=32
      IPATH=0
      NHH=-1
C
C*****************************************************************************
C                                                                             *
C                  SEARCH ITERATION LOOP                                      *
C                                                                             *
150   DO 800 IT=NITI,NIT
      IF(IT.EQ.1)IROLD=0
      CALL READEF(IEVFLG,IEFLAG)
      IF(IEFLAG.EQ.0)GO TO 160
      CALL CLREF(IEVFLG)
      GO TO 910
C
C  CHECK IF IN AN EXISTENT REGION
C
160   CONFID=0.
      IF(NREG.EQ.0)GO TO 250
      IF(ICON.NE.32)IREG=IROLD
      IF(ICON.NE.32)GO TO 300
      RAD2=0.
      DO 170 II=1,NFE
170   RAD2=RAD2+RD(II)*RD(II)
      RDIST=SQRT(RAD2)
      RAD1=RADM*RDIST+RDTH
      RAD2=RAD1**2
      DISTD=RAD2+.1
      IREG=-1
      DO 190 IR=1,NREG
      DIST=0.0
      DO 180 IFE=1,NFE
      IF(ABS(RD(IFE)-CENT(IFE,IR)).GT.RAD1)GO TO 190
      DIST=DIST+(RD(IFE)-CENT(IFE,IR))**2
```

```
180     IF(DIST.GT.RAD2)GO TO 190
        IF(DIST.GE.DIST0)GO TO 190
        IREG=IR
        DIST0=DIST
190     CONTINUE
        IF(IREG.LT.0)GO TO 250
C
C   RETRIEVE REGION INFORMATION
C
        IF(IREG.EQ.IREG0)GO TO 300
        CONSIS=0.0
        CALL POPQ(NDIR,0,IQUEUE,NQUEUE,IERR)
        IF(IPRI.GE.5 .AND. NQUEUE.GT.0)WRITE(5,990)2,NQUEUE,
        1 ((II,IQUEUE(II),APRIOR(IQUEUE(II))),II=1,NQUEUE)
        IF(IPATH.LE.1)GO TO 194
        DO 193 II=1,IPATH-1
        IF(ISREG(II).EQ.IREG)GO TO 300
193     CONTINUE
194     OPEN(UNIT=1,NAME=MEM0,TYPE='OLD',ACCESS='DIRECT',
        1 FORM='UNFORMATTED',SHARED)
        READ(1'IREG)FNT,AVVEC,AVMAG
        CLOSE (UNIT=1)
        IF(FNT.EQ.0.)GO TO 300
        VMAG=0.
        VMAGN=0.
        DO 195 II=1,NPA
        VMAGN=VMAGN+(AVVEC(II)/PPNE(II))**2
195     VMAG=VMAG+AVVEC(II)**2
        IF(VMAG.EQ.0)WRITE(5,9876)IREG,FNT,AVVEC,AVMAG
9876    FORMAT(' REG.',I4,' # OF TIMES ',F7.0,3(2X,E10.3))
        IF(VMAG.EQ.0.)GO TO 300
        VMAGN=SQRT(VMAGN)
        VMAG=SQRT(VMAG)
        CONSIS=VMAGN/AVMAG
        IF(CONSIS.LT..1)GO TO 300
        DO 200 II=1,NPA
200     BB(II,NDIR)=AVVEC(II)/VMAG
        APRIOR(NDIR)=0.
        CALL PLACEQ(NDIR,0,IQUEUE,NQUEUE,NDIR,APRIOR,1,IERR)
        IF(IPRI.GE.5 .AND. NQUEUE.GT.0)WRITE(5,990)3,NQUEUE,
        1 ((II,IQUEUE(II),APRIOR(IQUEUE(II))),II=1,NQUEUE)
        IF(CONSIS .GT. 0.2)CONFID=1.
        IF(CONSIS .GT. 0.5 .AND. FNT.GT.10.)CONFID=2.
        IF(CONSIS .GT. 0.85 .AND. FNT.GT.20.)CONFID=3.
        GO TO 300
C
C  CREATE NEW REGION
C
250     CALL POPQ(NDIR,0,IQUEUE,NQUEUE,IERR)
        CONSIS=0.0
        FNT=0           ! NUMBER OF TIMES IN THIS REGION
        IF(NREG.LT.NUMREG)GO TO 270
        IF(IPRI.GE.2 .AND. NNREG.GT.0)WRITE(6,1500)
1500    FORMAT(' NO MORE MEMORY SPACE')
        GO TO 350
270     IF(IPRI.GE.5 .AND. NQUEUE.GT.0)WRITE(5,990)4,NQUEUE,
        1 ((II,IQUEUE(II),APRIOR(IQUEUE(II))),II=1,NQUEUE)
        IF(IPRI.GE.3)WRITE(6,1600)NREG+1,A0
1600    FORMAT(' CREATING REGION ',I3,' AT: ',613.7,/,(NFE)(' ;',F7.2))
        NREG=NREG+1
        IREG=NREG
        DO 280 IFE=1,NFE
280     CENT(IFE,IREG)=A0(IFE)
        OPEN(UNIT=1,NAME=MEM0,TYPE='OLD',ACCESS='DIRECT',
        1 FORM='UNFORMATTED',SHARED)
        WRITE(1'IREG)FNT,AVVEC,AVMAG,
        1  (CENT(II,IREG),II=1,NFE)
        CLOSE (UNIT=1)
```

```
C
C---------- STORE REGION AND PARAMETERS FOR UPDATING ----------
C
300     IF(IREG.EQ.0 .OR. IPATH.GE.MAXST)GO TO 350
        IPATH=IPATH+1
        ISREG(IPATH)=IREG
        DO 310 II=1,NP
310     SPAR(II,IPATH)=P(II)
        IF(IPATH.EQ.1)IREG1=IREG
C
C------------------------ SELECT A DIRECTION ------------------------
C
350     CALL POPQ(IQ,1,IQUEUE,NQUEUE,IERR)
        IF(IPRI.GE.5 .AND. NQUEUE.GT.0)WRITE(5,998)S,NQUEUE,
        1 ((II,IQUEUE(II),QPRIOR(IQUEUE(II))),II=1,NQUEUE)
        IF(IERR.GT.0)WRITE(5,999)IERR,I,NQUEUE
        PMX=0.0
        DO 360 II=1,NP
360     PMX=PMX+(PD(II,ID)/PPRE(II))**2
        PRE=1./SQRT(PMX)
        DELOLD=DELOLD/3.
        IF(ID.LE.NBD)DELTA=SIGN(ID)*DELOLD*PRE*DDEL(ID)
        IF(ID.GT.NBD)DELTA=PMAG
        IF(ABS(DELTA).LT.PRE .AND. DELTA.NE.0)
        1  DELTA=1.1*PRE*DELTA/ABS(DELTA)
        IF(ABS(DELTA).LT.PRE .AND. DELTA.EQ.0)DELTA=1.1*PRE
        IF(IPRI.GE.3)WRITE(6,8145)ID,DELOLD,PRE,DELTA
8145    FORMAT(' ID=',I3,' DELOLD=',G12.6,' PRE=',G12.6,' DEL=',G12.6)
C
C---------------------- UNIDIRECTIONAL SEARCH ----------------------
C
        CALL SUGOAL(NEP,NDP,X,Y,N,PRE,DELTA,NPR,P,PMIN,PMAX,PM,
        1  PD(1,ID),IPRI,NFE,
        1  CONF ID,RO,MSSD,MMSSD,SSP,SSD,MSSD,SSDO,ISDGV)
        SS=SSP/PRE
C------------- LEARN SIGN AND DELTA FOR EACH BASIC DIRECTION ---
        IF(ID.LE.NBD .AND. SS.NE.0.)BDSIGN(ID)=SS/ABS(SS)
C
        IF(IT.EQ.1 .AND. ID.LE.NBD)DELOLD=.1*ABS(SS)/DDEL(ID)+.9*DELOLD
        IF(IT.EQ.1 .AND. ID.GT.NBD)DELOLD=.1*ABS(SS)+.9*DELOLD
C
        IF(ID.LE.NBD)DELOLD=.8*ABS(SS)/DDEL(ID)+.2*ABS(DELTA/PRE)
        IF(ID.GT.NBD)DELOLD=.8*ABS(SS)+.2*DELOLD
C
C======IF(ID.LE.NBD .AND. IT.GT.1)DDEL(ID)=.95*DDEL(ID)+.05*ABS(SSP/DELTA)
C
        IF(IT.EQ.1 .AND. ID.LT.NDIR)
        1  QPRIOR(ID)=.9*QPRIOR(ID)+.1*(SSDO-SSO)/SSDO
        IF(IT.GT.1.AND.ID.LT.NDIR.AND.SS.NE.0.)CPRIOR(ID)=CPRIOR(ID)+.01
        QPRIOR(ID)=.1*QPRIOR(ID)+.9*(SSDO-SSD)/SSDO
C
C---------- PENALIZE DIRECTIONS SIMILAR TO LAST ONE ----------
C
        DO 366 II=1,NQUEUE+1
        PM=0.0
        PMX=0.0
        PMY=0.0
        DO 364 JJ=1,NP
        PMX=PMX+(PD(JJ,II)/PPRE(JJ))**2
        PMY=PMY+(PD(JJ,ID)/PPRE(JJ))**2
364     PM=PM+PD(JJ,II)*PD(JJ,ID)/(PPRE(JJ)**2)
        PM=(PM/PMX)*(PM/PMY)
366     QPRIOR(II)=QPRIOR(II)**2./(1.+3.*PM)
C
C---------------------- CHECK IF CONVERGING ----------------------
C
369     IF(ABS(SS).GT.1.)GO TO 370
C
```

```
C------------- CONVERGING ---------------
C
      NNN=NHU+1
      CALL POPQ(NNIR,0,IRREVE,NQUEUE,IERR)
      IF(IPRI.GE.5 .AND. NQUEUE.GT.0)WRITE(5,990)2,NQUEUE,
     1 ((II,IQUEUE(II),QPRIOR(IQUEUE(II))),II=1,NQUEUE)
      GO TO 700
C
C-------------------- NOT CONVERGING -----------
C
370   NNN=0
      IF(ABS(SS).LT.1.)NNN=1
C-----------------REORDER DIRECTIONS IN QUEUE--------
      CALL REORDR(0,IQUEUE,NQUEUE,QPRIOR,NDIR,IERR)
      IF(IPRI.GE.5 .AND. NQUEUE.GT.0)WRITE(5,990)6,NQUEUE,
     1 ((II,IQUEUE(II),QPRIOR(IQUEUE(II))),II=1,NQUEUE)
      IF(IERR.GT.0)WRITE(5,999)IERR,1,NQUEUE
C------------- PUT LAST DIRECTION IN THE QUEUE END ------
700   IPL=NQUEUE
      CALL PLACE0(ID,IPL,IQUEUE,NQUEUE,NDIR,QPRIOR,NDIR,IERR)
      IF(IPRI.GE.5 .AND. NQUEUE.GT.0)WRITE(5,990)7,NQUEUE,
     1 ((II,IQUEUE(II),QPRIOR(IQUEUE(II))),II=1,NQUEUE)
      IF(IERR.GT.0)WRITE(5,999)IERR,1,NQUEUE
C
C---------------------  BOOKKEEPING ------------------
C
      TNSSD=TNSSD+FLOAT(NSSD)
      SST=SST+ABS(SS)
      IROL0=IREG
      SSD0=SSD
      DO 780 J=1,NPR
780   P(J)=PW(J)
C
C-------------------  ITERATION PRINTOUTS -------------
C
      ICON=32
      IF(ABS(SS).LT.1.)ICON=35
      IF(IPRI.GE.1)WRITE(6,1000)IT,SSD0,IFIX(TNSSD),ADIST,IREG,
     1  IFIX(FNT),CONSIS,ID,SS,ICON
1000  FORMAT(1X,I3,' K2=',F10.0,' N=',I4,' R=',F5.2,' R=',I3,'(',I5,')',
     1  ' C=',F4.2,2X,'H=',I2,' S=',F8.1,A1)
      IF(IPRI.GE.2)WRITE(6,1900)(PW(J),J=1,NPA)
1900  FORMAT(' PAR. VAL. ',<NPA>(2X,G13.7))
      IF(NNN.EQ.NDD)GO TO 810
800   CONTINUE
C
C-----------------------------------------------------------------*
C                                                                 *
C                END OF SEARCH LOOP                               *
C                                                                 *
C*****************************************************************
C
      WRITE(6,2000)NIT
2000  FORMAT(' NOT CONVERGED AFTER ',I4,' ITERATIONS')
      GO TO 910
C
C  EXIT ON CONVERGENCE
C
810   LREM=0
      IFIRST=1
      DO 820 J=1,NDV
820   IRGEN(J)=0
        DO 850 I=1,NEP
        IF(I.EQ.NEP)IFIRST=-1
        CALL FUNCIC(X(I),P,FI,IFIRST)
          DO 840 J=1,NDV
          IFIRST=0
          ISI=SIGN(1.,(YK(I,J)-FI(J)))
```

```
            IF(I.EQ.1)GO TO 830
            IF(ISI.EQ.ISRUN(J))GO TO 840
830         ISRUN(J)=ISI
            IRRUN(J)=IRRUN(J)+1
840         CONTINUE
850       CONTINUE
          DO 860 J=1,NDV
860       LRUN=LRUN+IRRUN(J)
          RUN=FLOAT(NEP)*(FLOAT(NEP)/2.+1.)
          STRUN=SQRT(FLOAT(NEP)*FLOAT(NEP-2)/(4.*FLOAT(NEP-1)))
          STRUN=STRUN*SQRT(FLOAT(NDV))
          IF(IPRI.GE.0)WRITE(6,2400)NBR,NBF,IT,IFIX(TNSSD),RDIST,RD,SSD,LRUN,
          1  RSSD,IRRUN,P,PT
2400      FORMAT(/,
          1  ' NBR= ',I5,2X,'FIT=',I5,2X,' IT ',I5,'  NSSD ',I5,' DIST ',F8.3,/,
          1  (NFE)(1X,F7.3),/,
          1  ' SSD= ',G13.7,' RUNS ',I3,/,
          1  (RRR)(1X,F7.5),/,
          1  (NDV)(2X,I3,3X),/,
          1  ' FIN. PAR. ',(NPA)(2X,G13.7),/,
          1  ' TRUE PAR. ',(NPA)(2X,G13.7),/)
C
C  CHECK IF THE FIT IS ACCEPTABLE
C
C    CHECK # RUNS
C    CHECK SUM OF SQUARES
C
          IF(SSD .LT. 5.*FLOAT(NEP*NDV))GO TO 865
          JBAD=JBAD+1
          WRITE(6,2001)NBR,NBF,SSD,JBAD
2001      FORMAT(
          1  ' XXXXXXXXXXXXXXXXXX',/,
          1  ' XXXX  BAD FIT XXXX NBR=',I3,' FIT=',I3,' SSD=',F8.3,/,
          1  ' XXXXXXXXXXXXXXXXXX',/,'  ',I6,' TIMES',/)
          IPAFLS=IPAFL
          IPAFL=2
          IF(JBAD.LT.JBADMA)GO TO 86
865       IF(JBAD.GT.0)IPAFL=IPAFLS
          JBAD=0
C    CHECK FINAL PARAMETERS VALUES
C
C    KEEP AVERAGE PARAMETERS
C
          IF(NAVE.LT.3)NAVE=NAVE+1
          DO 870 I=1,NPA
          IF(NAVE.GE.2)S2=FLOAT(NAVE-2)*PSDV(I)*PSDV(I)+
          1                FLOAT(NAVE-1)*PAVE(I)*PAVE(I)
          PAVE(I)=(FLOAT(NAVE-1)*PAVE(I)+P(I))/FLOAT(NAVE)
          IF(NAVE.GE.2)PSDV(I)=SQRT((S2+P(I)*P(I)-
          1           FLOAT(NAVE)*PSDV(I)*PAVE(I))/FLOAT(NAVE-1))
C
C   UPDATE MEMORY
C
          IF(IPATH.EQ.0)GO TO 890
          OPEN(UNIT=1,NAME=MEMO,TYPE='OLD',ACCESS='DIRECT',
          1 FORM='UNFORMATTED',SHARED)
          DO 885 II=1,IPATH
          IREC=ISREC(II)
          READ(1'IREC)FNT,AVVEC,AVRMS
          VRMS=0.
          DO 880 I=1,NPA
          DECI=P(I)-SPAR(I,II)
          AVVEC(I)=(FNT*AVVEC(I)+DECI)/(FNT+1)
880       VRMS=VRMS+(DECI/PPRE(I))**2
          VRMS=SQRT(VRMS)
          IF(VRMS.LT.1.0)GO TO 885
          AVRMS=(FNT*AVRMS+VRMS)/(FNT+1)
          FNT=FNT+1
```

```
      WRITE(1'IRE6)FNT,AUVEC,AVNAG,
     1  (CENT(JJ,IRE6),JJ=1,NFE)
805   CONTINUE
      CLOSE (UNIT=1)
C
C   STORE UPDATED HEAD
C
850   OPEN(UNIT=1,NAME=HEAD,TYPE='OLD',ACCESS='SEQUENTIAL',
      1 FORM='UNFORMATTED',SHARED)
      WRITE(1)TIT
      CALL DATE(TDATE)
      CALL TIME(TTIME)
      WRITE(1)SDATE,STIME,TDATE,TTIME
      WRITE(1)NEP,NDV,NPA,NREST,NRB,NFE,NBFIT,NBREP,NNAE6,NRE6,NBF,NBR,
     1  TNFIT,ISEP1,ISEP2,ISPA1,ISPA2
      WRITE(1)PRE,RAD,RMIN,KIT,DNSSD,IPRI,ISEL,IBPB,
     1  LC,LOOPC,LOOP2,IEVFLG,JBAD,NAVE,IPAFL
      WRITE(1)PAVE
      WRITE(1)PSDV
      WRITE(1)PMIN
      WRITE(1)PMAX
      WRITE(1)PPRE
      WRITE(1)BPRIOR
      WRITE(1)CPRIOR
      WRITE(1)DELDLO,DOEL
      CLOSE (UNIT=1)
      CONTINUE
C
C   LOG THIS PARAMETER IDENTIFICATION
C
      IREC=(NBR-1)*NBFIT+NBF
      OPEN(UNIT=1,NAME=FITL,TYPE='OLD',ACCESS='DIRECT',
     1  FORM='UNFORMATTED',SHARED)
      WRITE(1'IREC)IRE6I,ISEP1I,ISEP2I,ISPA1I,ISPA2I,PI,PT,ROI,SSDI,IRE6,
     1  P,RA,SSD,LAMA,IT,IFIX(TNSSD),SST,NRE6,JBAD
      CLOSE (UNIT=1)
      IF(LC.EQ.0)GO TO 855
C
C   UPDATE LEARNING CURVE FILE
C
      OPEN (UNIT=1,NAME=LCUR,TYPE='OLD',ACCESS='DIRECT',
     1 SHARED,RECORDSIZE=10)
      STSS=0.0
      DO 892 JJ=1,NPA
892   STSS=STSS+((P(JJ)-PI(JJ))/PPRE(JJ))**2
      STSS=SQRT(STSS)
      SST=SST/STSS
      NTNS=0
      APIT=0
      APIT2=0
      AVIIT=0
      AVIIT2=0
      ASST=0
      ASST2=0
      IF(NBR.GT.1)READ(1'NBF)NTNS,APIT,APIT2,AVIIT,AVIIT2,ASST,ASST2
      NTNS=NTNS+1
      APIT=(APIT*FLOAT(NTNS-1)+FLOAT(IT))/FLOAT(NTNS) ! ***
      APIT2=(APIT2*FLOAT(NTNS-1)+(FLOAT(IT)**2))/FLOAT(NTNS) ! ***
      AVIIT=(AVIIT*FLOAT(NTNS-1)+TNSSD)/FLOAT(NTNS) ! ***
      AVIIT2=(AVIIT2*FLOAT(NTNS-1)+(TNSSD**2))/FLOAT(NTNS) ! ***
      ASST=(ASST*FLOAT(NTNS-1)+SST)/FLOAT(NTNS) ! ***
      ASST2=(ASST2*FLOAT(NTNS-1)+(SST**2))/FLOAT(NTNS) ! ***
      WRITE(1'NBF)NTNS,APIT,APIT2,AVIIT,AVIIT2,ASST,ASST2
      CLOSE (UNIT=1)
855   CONTINUE
900   IF(LOOPC.EQ.1)GO TO 50
      IF(LOOPC.EQ.2)GO TO 80
      IF(LOOPC.EQ.3)GO TO 40
```

```
910     NITI=IT
        WRITE(5,2610)TIT
2610    FORMAT(/,1X,70A1,/,
        1  ' ENTER:  0- TO GET NEW DATA POINTS',/,
        1  '         1- TO CHANGE PRECISION, PRINT-PLOT LEVEL...',/,
        1  '         2- TO TO CHANGE PARAMETERS',/,
        1  '         3- TO CONTINUE ITERATION',/,
        1  '         4- TO CHANGE DEVICE FOR LU6',/,
        1  '         5- TO START OVER',/,
        1  '         6- TO FORCE CONVERGENCE',/,
        1  '         7- TO STOP  ..................... ?',$)
        READ(4,2650,ERR=910)IANS
2650    FORMAT(I15)
        GO TO (50,920,80,150,970,40,900,990),IANS+1
        GO TO 910
920     WRITE(5,2700)TIT,RAB,RMIN,PRE,NIT,IPRI,NNREG,LOOPC,NBREP,IEVFLG,
        1  ISEL,IUP0,NNSSQ,NBFIT,IPAFL
2700    FORMAT(/,1X,70A1,/,
        1   ' 1. RAB  =',G11.5,'2. RMIN =',G11.5,2X,'3. PRE =',G11.5,/,
        1   ' 4. NIT  =',I4,7X,'5. IPRI =',I4,9X,'6. NNREG=',I4,/,
        1   ' 7. LOOPC=',I4,7X,'8. NBREP=',I4.9X,'9.IEVFLG=',I5,/,
        1   ' 10.ISEL =',I4,7X,'11.IUP0 =',I4,9X,'12.NNSSQ=',I4,/,
        1   ' 13.NBFIT=',I4,7X,'14.IPAFL=',I4,/,
        1   ' ENTER PARAMETER # TO CHANGE:',$)
        READ(4,2650,ERR=920)IANS
        IF(IANS.LT.1 .OR. IANS.GT.14)GO TO 920
930     WRITE(5,2900)
2900    FORMAT(' ENTER NEW VALUE: ',$)
        IF(IANS.EQ.1)READ(4,2900,ERR=970)RAB
        IF(IANS.EQ.2)READ(4,2900,ERR=930)RMIN
        IF(IANS.EQ.3)READ(4,2900,ERR=930)PRE
        IF(IANS.EQ.4)READ(4,2650,ERR=930)NIT
        IF(IANS.EQ.5)READ(4,2650,ERR=930)IPRI
        IF(IANS.EQ.6)READ(4,2650,ERR=930)NNREG
        IF(IANS.EQ.7)READ(4,2650,ERR=930)LOOPC
        IF(IANS.EQ.8)READ(4,2650,ERR=930)NBREP
        IF(IANS.EQ.9)READ(4,2650,ERR=930)IEVFLG
        IF(IANS.EQ.10)READ(4,2650,ERR=930)ISEL
        IF(IANS.EQ.11)READ(4,2650,ERR=930)IUP0
        IF(IANS.EQ.12)READ(4,2650,ERR=930)NNSSQ
        IF(IANS.EQ.13)READ(4,2650,ERR=930)NBFIT
        IF(IANS.EQ.14)READ(4,2650,ERR=930)IPAFL
2900    FORMAT(F15.4)
        GO TO 910
970     WRITE(5,3000)
3000    FORMAT(' ENTER DEVICE C EX: TTE%,C6]]] :',$)
        READ(4,3100,ERR=970)II,III,IIII
3100    FORMAT(A2,I10,I10)
        IF(IIII.EQ.0)IIII=6
        CALL ASNLUN(IIII,II,III)
        GO TO 910
990     STOP
        END
```

```
CMS-
C       Module name: SOLPIS1 :  GETEP, GETPR & FUNCI
C
C       Version X01.00  Last edit: 07-JUL-83 16:09
C       Status: Development/Debugging
C
C       Revision history:
C
C               Version X01.00 03-JUL-82 13:34 - 07-JUL-83 16:09
C               Created by: E. MACHADO
C
CMS-
        SUBROUTINE GETEP(N,NDU,Y,X,W,IFL)
C
C   IFL = 1 Generate parameters at random for experimental points generation
C       = 2 Use parameters in common for experimental points generation
C
        PARAMETER NDOIN=1               ! DEP. VAR.
        PARAMETER NEPOM=20              ! # EXPERIMENTAL POINTS
        PARAMETER NPOM=3                ! # OF PARAMETERS
        COMMON /SEEDEP/IS1,IS2
        DIMENSION Y(N,NDOIN),X(N),W(N,NDOIN)
        DIMENSION FI(NDOIN)
        DIMENSION P(NPOM)
        COMMON /TOETA/P
        DIMENSION FEA(6)
        COMMON /FEATOR/FEA
        LOGICAL*1 $TIME(6)
        ERR=.1
        BL=100000
        IFIRST=1
        N=NEPOM
        IF(IFL.EQ.2)GO TO 80
        DO 50 I=1,NPOM
50      P(I)=(1.-ERR)+2.*ERR*RAN(IS1,IS2)
80      DELTF=.001
        DO 100 I=1,N
C
C  LOG SPACED  .001 TO .1 HZ
C
        X(I)=DELTF*1.274**(I-1)
        CALL FUNCI(X(I),P,FI,IFIRST)
        IFIRST=0
        DO 90 J=1,NDOIN
        Y(I,J)=FI(J)
70      XR1=RAN(IS1,IS2)
        IF(XR1.LE.0)GO TO 70
        XR2=RAN(IS1,IS2)
        XR=SQRT(-2.*ALOG(XR1))*COS(2.*3.141592*XR2)
        FACT=1.+XR/SQRT(BL)
        Y(I,J)=FACT*Y(I,J)
        W(I,J)=BL/(Y(I,J)*Y(I,J))
90      CONTINUE
100     CONTINUE
        RETURN
        END
```

```
      SUBROUTINE GETPA(K,P,PMIN,PMAX,PAVE,PSDV,NAME,PPRE,IPF,IFL)
C
C   CHANGES:
C   22-DEC-92 (a) INCLUDED MAX,MIN AND AVE (b) initial guess modified
C   10-jan-93 use of IFL as described below:
C   IFL = 0 READ initial guess from TI
C       = 1 initial guess equal to the average
C       = 2 initial guess selected at random within interval
C       = 3 initial guess selected around moving average value
C       = 4 KEEP INITIAL GUESS CONSTANT , EQUAL TO LAST INITIAL GUESS
C
      PARAMETER NPOIN=3              ! # OF PARAMETERS
      DIMENSION P(NPOIN),PMIN(NPOIN),PMAX(NPOIN),PAVE(NPOIN),IPF(NPOIN)
      DIMENSION PSDV(NPOIN),PPRE(NPOIN),PT(NPOIN)
      DIMENSION POLD(NPOIN)
      LOGICAL*1 IANS
      DATA ICOUNT/3/
      COMMON /TDATA/PT
      COMMON /SEEDPA/ISPA1,ISPA2
      K=NPOIN
      GO TO (90,300,500,400,900),IFL+1
90    DO 100 I=1,NPOIN
      P(I)=PAVE(I)
      WRITE(5,1000)I,P(I),PMIN(I),PMAX(I),PPRE(I),IPF(I)
1000  FORMAT(' ',I2,' P:',G13.6,' MIN:',G13.6,' MAX:',G13.6,
     1  ' PRE:',G13.6,' FL:',I3,/,' DO YOU WANT TO CHANGE [N] ?',$)
      READ(5,1100)IANS
1100  FORMAT(A1)
      IF(IANS.NE.'Y')GO TO 100
      WRITE(5,1200)P(I)
1200  FORMAT(' P:[',G13.6,'] NEW VALUE:',$)
      READ(5,1300)RANS,ANS
1300  FORMAT(E,F15.0)
      IF(RANS.GT.0)P(I)=ANS
      WRITE(5,1400)PMIN(I)
1400  FORMAT(' PMIN:[',G13.6,'] NEW VALUE:',$)
      READ(5,1300)RANS,ANS
      IF(RANS.GT.0)PMIN(I)=ANS
      WRITE(5,1500)PMAX(I)
1500  FORMAT(' PMAX:[',G13.6,'] NEW VALUE:',$)
      READ(5,1300)RANS,ANS
      IF(RANS.GT.0)PMAX(I)=ANS
      WRITE(5,1600)PPRE(I)
1600  FORMAT(' PPRE:[',G13.6,'] NEW VALUE:',$)
      READ(5,1300)RANS,ANS
      IF(RANS.GT.0)PPRE(I)=ANS
      WRITE(5,1700)IPF(I)
1700  FORMAT(' IPF:[',I3,'] NEW VALUE:',$)
      READ(5,1300)RANS,ANS
      IF(RANS.GT.0)IPF(I)=ANS
100   CONTINUE
      GO TO 990
300   DO 350 I=1,NPOIN
350   P(I)=PAVE(I)
      GO TO 990
500   DO 550 I=1,NPOIN
550   P(I)=PMIN(I)+(PMAX(I)-PMIN(I))*RAN(ISPA1,ISPA2)
      GO TO 990
400   CONTINUE
C     NAVE=MIN0(NAVE,100)
C ---- FIXED TO REPEAT SAME INITIAL GUESS 3 TIMES ------
      ICOUNT=ICOUNT+1
      IF(ICOUNT.LT.3)GO TO 900
      ICOUNT=0
      DO 850 I=1,NPOIN
C     P(I)=(PAVE(I)*FLOAT(NAVE)+(PMAX(I)+PMIN(I))/2.)/FLOAT(NAVE+1)
      P(I)=PAVE(I)
850   P(I)=P(I)*(.5+RAN(ISPA1,ISPA2))
```

```
      GO TO 990
900   DO 910 I=1,NPOIN
910   P(I)=POLD(I)
      RETURN
990   DO 995 I=1,NPOIN
995   POLD(I)=P(I)
      END
```

```
      SUBROUTINE FUNCI(X,P,FI,IFIRST)
      IMPLICIT REAL*4 L
      PARAMETER NDBIM=1                 ! # OF DEP. VARIABLES
      PARAMETER NPOIN=9                 ! # OF PARAMETERS
      DIMENSION P(NPOIN),FI(NDBIM)
      DIMENSION ALP(10)
      DATA ALP/-.29704854E-4,-.12004583E+1,-.43661313E+1,-.47873907E+1,
     1  .61125893,.22234778E+1,-.31128698E+0,.15462272E-2,
     1  -.55530403E-4,.28279161E-4/
      DATA A12,A21/1.20046,4.78739/
      DATA B32,B33,B34/5.2133E-2,-3.39324E-2,-3.17917E-1/
      DATA PSDTI,PSDTF,PSDNF,PSDNN/1.,1.,75.,1.88144E-2/
      DATA WP,WNS,PSDNS/10941.6,.6858206,75./
      COMPLEX*8 DELT,L11,L12,L13,L21,L22,L23,L31,L32,L33
      COMPLEX*8 H15,H35
C------ P3=1 -----
      IF(IFIRST.EQ.0)GO TO 100
C**  N-INDEPENDENT CALCULATION ***
      A11=P(1)*WP*ALP(1)+ALP(2)
      A22=P(2)*WNS*ALP(3)+ALP(4)
      A23=P(2)*WNS*ALP(5)
      A32=P(2)*WNS*ALP(6)
      A33=P(2)*WNS*ALP(7)+ALP(8)
      B11=-P(1)*WP*ALP(1)
      B112=B11*B11
      B25=P(1)*WP*ALP(9)
      B252=B25*B25
      B322=B32*B32
      B332=B33*B33
      B342=B34*B34
      B35=P(1)*WP*ALP(10)
      B352=B35*B35
      B25B35=B25*B35
      L13=CMPLX(A12*A23,0.)
      L132=REAL(L13*CONJG(L13))
      L31=CMPLX(A21*A32,0.)
      L312=REAL(L31*CONJG(L31))
C**  N-DEPENDENT CALCULATIONS ***
100   X1=XN6.2831853
      DELT=CMPLX(-A11,X1)*CMPLX(-A22,X1)*CMPLX(-A33,X1)
     1  -A23*A32*CMPLX(-A11,X1)-A12*A21*CMPLX(-A33,X1)
      DELT2=REAL(DELT*CONJG(DELT))
      L11=(CMPLX(-A22,X1)*CMPLX(-A33,X1)+CMPLX(-A23*A32,0.))
      L112=REAL(L11*CONJG(L11))
      L21=CMPLX(A21,0.)*CMPLX(-A33,X1)
      L212=REAL(L21*CONJG(L21))
      L12=CMPLX(A12,0.)*CMPLX(-A33,X1)
      L122=REAL(L12*CONJG(L12))
      L22=CMPLX(-A11,X1)*CMPLX(-A33,X1)
      L222=REAL(L22*CONJG(L22))
      L32=CMPLX(A32,0.)*CMPLX(-A11,X1)
      L322=REAL(L32*CONJG(L32))
      L23=CMPLX(A23,0.)*CMPLX(-A11,X1)
      L232=REAL(L23*CONJG(L23))
      L33=(CMPLX(-A11,X1)*CMPLX(-A22,X1)+CMPLX(-A12*A21,0.))
      L332=REAL(L33*CONJG(L33))
C     AL1213=REAL(L12*CONJG(L13))
C     AL3233=REAL(L32*CONJG(L33))
      H15=CMPLX(B25,0.)*L12+CMPLX(B35,0.)*L13
      H152=REAL(H15*CONJG(H15))
      H35=CMPLX(B25,0.)*L32+CMPLX(B35,0.)*L33
      H352=REAL(H35*CONJG(H35))
      FI(1)=(L112*B112*PSDTI*L132*(B322*PSDTF+B332*PSDNF+B342*PSDNS*P(3))+
     1  H152*PSDNN)/DELT2
C     FI(2)=(B112*L312*PSDTI*L332*(B322*PSDTF+B332*PSDNF+B342*PSDNS*P(3))+
C     1  H352*PSDNN)/DELT2
      RETURN
      END
```

```
C**-
C       Module name: SOLPIS2 : STATUS & SUBGRL
C
C       Version X01.00  Last edit: 04-JUL-83 10:09
C       Status: Development/Debugging
C
C       Revision history:
C
C             Version X01.00 03-JUL-83 15:04 - 04-JUL-83 10:10
C             Created by: E.L.Machado
C
C**-
        SUBROUTINE STATUS(NEP,NDV,X,Y,W,P,SSQ,RSSQ,NFE,RQ,IFL)
C
C   IFL=0  CALCULATE EVERYTHING
C      =1  CALCULATE SSQ ONLY
C      =2  CALCULATE SSQ AND FEATURES
C
        PARAMETER NDDIM=2               ! # OF DEP. VARIABLES
        PARAMETER NFQDP=4               ! INT(NFDIM/NDDIM)
        DIMENSION FI(NDDIM),CAL(NFQDP)
        DIMENSION X(NEP),Y(NEP,NDV),W(NEP,NDV),P(NEP),RSSQ(NDV),RQ(NFE)
        IF(IFL.GT.0)GO TO 200
C
C----- PARAMETER INDEPENDENT CALCULATIONS -----
C
        XBAR=0.0
          DO 100 I=1,NEP
100       XBAR=XBAR+X(I)
        XBAR=XBAR/FLOAT(NEP)
        II=(NFE-1)/NDV
        IF(II.EQ.0)GO TO 120
          DO 110 I=1,NEP
            DO 105 J=1,II
            IF(I.EQ.1)CAL(J)=0.
            CAL(J)=CAL(J)+(X(I)-XBAR)**(J+2)
105         CONTINUE
110       CONTINUE
120     CONTINUE
C
C----- PARAMETER DEPENDENT CALCULATIONS -----
C
200     SSQ=0.0
        IFIRST=1
        DO 280 J=1,NFE
280     RQ(J)=0
        DO 290 J=1,NDV
290     RSSQ(J)=0
          DO 400 I=1,NEP
          IF(I.EQ.NEP)IFIRST=-1
          CALL FUNCTX(X(I),P,FI,IFIRST)
          IFIRST=0
            DO 390 J=1,NDV
C**- EVALUATE SSQ -**
            DEVJ=Y(I,J)-FI(J)
            SQJ=SQRT(W(I,J))
            RSSQ(J)=RSSQ(J)+DEVJ*DEVJ*W(I,J)
            IF(IFL.EQ.1)GO TO 390
C**- EVALUATE FEATURES -**
            III=0
              DO 350 II=J,NFE,NDV
              RQ(II)=RQ(II)+DEVJ*SQJ*(X(I)-XBAR)**III
350           III=III+1
390         CONTINUE
400       CONTINUE
        SSQ=0
        DO 410 J=1,NDV
410     SSQ=SSQ+RSSQ(J)
```

```
      DO 420 J=1,NDV
420   NSSQ(J)=FLOAT(NDP)*NSSQ(J)/SSQ
      IF(IFL.EQ.1)RETURN
C**- CALIBRATE FEATURES -**
        DO 500 II=1,NFE
        I=(II-1)/NDV
        J=I-INDV
        IF(I.EQ.0)RQ(II)=RQ(II)/SQRT(FLOAT(NEP))
        IF(I.GE.1)RQ(II)=RQ(II)/SQRT(CNL(I))
500     CONTINUE
      RETURN
      END
```

```
      SUBROUTINE SRCHAL(N,MDV,X,Y,U,PRE,APDT,KDIM,P,PMIN,PMAX,PN,
     1  D,IPRI,NFDIM,
     1  CONFID,A0,NSSB,NNSSD,SS,SSB,RSSD,SSDB,IVER)
C
C  SEARCH MINIMUM SSB USING ALONG "U"
C
      PARAMETER NFDIM=6                    ! FEATURES
      DIMENSION P(KDIM),PMIN(KDIM),PMAX(KDIM),PN(KDIM),D(KDIM),A0(NFDIM)
      DIMENSION RSSB(MDV)
      DIMENSION Q(3),S(3)
      DIMENSION A0SAVE(NFDIM)
      IVER=1
      NSSB=0
      DELTA=APDT
      IQUIT=1
      DO 100 J=1,NFDIM
100   A0SAVE(J)=A0(J)
      S(1)=0
      Q(1)=SSDB
      DO 110 J=1,KDIM
      PNJ=P(J)+DELTA*D(J)
      IF(PNJ .LE.PMAX(J))GO TO 105
      IF(P(J).GE.PMAX(J))DELTA=-1.1*PRE*DELTA/ABS(DELTA)
      IF(P(J).LT.PMAX(J))DELTA=(PMAX(J)-P(J))/D(J)
      IF(IPRI.GE.1)WRITE(6,1900)NSSB,J
1900  FORMAT('  NSSB=',I3,' PARAMETER ',I2,' REACHED UPPER LIMIT')
      GO TO 110
105   IF(PNJ .GE.PMIN(J))GO TO 110
      IF(P(J).LE.PMIN(J))DELTA=-1.1*PRE*DELTA/ABS(DELTA)
      IF(P(J).GT.PMIN(J))DELTA=(PMIN(J)-P(J))/D(J)
      IF(IPRI.GE.1)WRITE(6,1950)NSSB,J
1950  FORMAT('  NSSB=',I3,' PARAMETER ',I2,' REACKED LOWER LIMIT')
110   CONTINUE
      S(2)=DELTA
      DO 210 J=1,KDIM
      PN(J)=P(J)+S(2)*D(J)
      IF(PN(J).LT.PMIN(J))PN(J)=PMIN(J)
210   IF(PN(J).GT.PMAX(J))PN(J)=PMAX(J)
      CALL STATUS(N,MDV,U,Y,U,PN,SSD,RSSB,NFDIM,A0,2)
      IF(IPRI.GE.2)WRITE(6,1000)S(2)/PRE,SSD,SSDB/SSD
1000  FORMAT(' S:',F10.1,' SSB:',F12.3,' RATIO:',F12.3)
      NSSB=1
      Q(2)=SSD
      IF(SSD.GT.Q(IQUIT))GO TO 250
      IQUIT=2
      DO 220 J=1,NFDIM
220   A0SAVE(J)=A0(J)
C
C   THIRD POINT STRATEGY
C
250   CONTINUE
      IF(CONFID.LT.3)GO TO 260
      IF(Q(2).LT.Q(1))S(3)=1.33*DELTA
      IF(Q(2).GE.Q(1))S(3)=.75*DELTA
      GO TO 290
260   IF(CONFID.LT.2)GO TO 270
      IF(Q(2).LT.Q(1))S(3)=1.6*DELTA
      IF(Q(2).GE.Q(1))S(3)=.66*DELTA
      GO TO 290
270   IF(CONFID.LT.1)GO TO 280
      IF(Q(2).LT.Q(1))S(3)=2.*DELTA
      IF(Q(2).GE.Q(1))S(3)=.5*DELTA
      GO TO 290
280   IF(Q(2).LT.Q(1))S(3)=2.0*DELTA
      IF(Q(2).GE.Q(1))S(3)=-1.0*DELTA
290   CONTINUE
      DO 320 J=1,KDIM
      PNJ=P(J)+S(3)*D(J)
```

```
      IF(PNJ.LE.PMAX(J))GO TO 310
      S(3)=(PMAX(J)-P(J))/D(J)
      IF(IPRI.GE.1)WRITE(6,1980)NSSD,J
      GO TO 320
310   IF(PNJ.GE.PMIN(J))GO TO 320
      S(3)=(PMIN(J)-P(J))/D(J)
      IF(IPRI.GE.1)WRITE(6,1950)NSSD,J
320   CONTINUE
      IF(S(3).EQ.S(1) .OR. S(3).EQ.S(2))GO TO 670
      DO 350 J=1,KDIM
      PN(J)=P(J)+S(3)*D(J)
      IF(PN(J).LT.PMIN(J))PN(J)=PMIN(J)
350   IF(PN(J).GT.PMAX(J))PN(J)=PMAX(J)
      CALL STATUS(N,NDP,X,Y,N,PN,SSD,NSSD,NFDIM,ND,2)
      IF(IPRI.GE.2)WRITE(6,1000)S(3)/PNE,SSD,SSD0/SSD
      NSSD=2
      A(3)=SSD
      IF(SSD.GT.A(IORIN))GO TO 400
      IORIN=3
      DO 380 J=1,NFDIM
380   DDSAVE(J)=DD(J)
C
C     ------- INNER LOOP ------
C
C  REORDER
C
400   CONTINUE
      DO 420 II=1,2
      ININ=II
      DO 410 III=II+1,3
410   IF(S(ININ).GT.S(III))ININ=III
      IF(ININ.EQ.II)GO TO 420
      SS=S(II)
      QQ=A(II)
      S(II)=S(ININ)
      A(II)=A(ININ)
      S(ININ)=SS
      A(ININ)=QQ
420   CONTINUE
      IAMAX=1
      IORIN=1
      DO 430 II=2,3
      IF(A(IAMAX).LT.A(II))IAMAX=II
430   IF(A(IORIN).GT.A(II))IORIN=II
C
C   CALCULATE FOURTH POINT
C
      DELTA=S(3)-S(1)
      B1=(A(2)-A(1))/(S(2)-S(1))
      B2=(A(3)-A(2))/(S(3)-S(2))
      IF(B2.LE.B1)GO TO 500
      S1=(S(1)+S(2))/2.
      S2=(S(2)+S(3))/2.
      SS=(B2*S1-B1*S2)/(B2-B1)
      IF(S(1)-SS.GT.3.0*DELTA)SS=S(1)-3.0*DELTA
      IF(SS-S(3).GT.3.0*DELTA)SS=S(3)+3.0*DELTA
      GO TO 505
500   IF(IPRI.GE.2)WRITE(6,2200)
2200  FORMAT(' NEGATIVE SECOND DERIVATIVE')
      IF(Q(3).NE.Q(1))SS=S(1)-Q(1)*(S(3)-S(1))/(Q(3)-Q(1))
      IF(Q(3).EQ.Q(1) .AND. Q(3).EQ.Q(2))SS=S(IORIN)
      IF(Q(3).EQ.Q(1) .AND.
     1  Q(3).NE.Q(2) .AND. D1.GT.ABS(B2))SS=S(1)-Q(1)/D1
      IF(Q(3).EQ.Q(1) .AND.
     1  Q(3).NE.Q(2) .AND. D1.LE.ABS(B2))SS=S(3)-Q(3)/D2
      IF(S(1)-SS.GT.2.0*DELTA)SS=S(1)-2.0*DELTA
      IF(SS-S(3).GT.2.0*DELTA)SS=S(3)+2.0*DELTA
505   CONTINUE
```

```
      DO 520 J=1,KDIM
      PNJ=P(J)+SSN*D(J)
      IF(PNJ.LE.PMAX(J))GO TO 510
      SS=(PMAX(J)-P(J))/D(J)
      IF(IPRI.GE.1)WRITE(6,1500)NSSD,J
      GO TO 520
510   IF(PNJ.GE.PMIN(J))GO TO 520
      SS=(PMIN(J)-P(J))/D(J)
      IF(IPRI.GE.1)WRITE(6,1550)NSSD,J
520   CONTINUE
C
C   CHECK IF WITHIN CONVERGENCE DISTANCE
C
      IF(ABS(S(IQMIN)).GT.PRE .AND. ABS(SS).LT.PRE)GO TO 600
      PRECK=.5*PRE
      IF(CONFID.EQ.0. .AND.
     1  ABS(S(IQMIN)).GT.5.*PRE)PRECK=.2*ABS(S(IQMIN))+0.5*PRE
      IF(CONFID.EQ.1. .AND.
     1  ABS(S(IQMIN)).GT.5.*PRE)PRECK=.1*ABS(S(IQMIN))
      IF(CONFID.GT.1. .AND.
     1  ABS(S(IQMIN)).GT.5.*PRE)PRECK=.05*ABS(S(IQMIN))+.25*PRE
C
      IF(IPRI.GE.2 .AND.
     1  ABS(SS-S(IQMIN)).LT.PRECK)WRITE(6,2000)SS/PRE
2000  FORMAT(' SS=',F8.1,' TOO CLOSE')
      IF(ABS(SS-S(IQMIN)).LT.PRECK)GO TO 670
C
C     NOT CONVERGED
C
600   CONTINUE
      DO 610 J=1,KDIM
      PN(J)=P(J)+SSN*D(J)
      IF(PN(J).LT.PMIN(J))PN(J)=PMIN(J)
610   IF(PN(J).GT.PMAX(J))PN(J)=PMAX(J)
      CALL STATUS(N,MSV,M,V,W,PN,SSD,RSSB,NFDIM,RO,2)
      IF(IPRI.GE.2)WRITE(6,1000)SS/PRE,SSD,SSDO/SSD
      NSSD=NSSD+1
      QQ=SSD
      IF(SSD.GT.Q(IQMIN))GO TO 620
      DO 615 J=1,NFDIM
615   RBSAVE(J)=RO(J)
620   IF(IPRI.GE.9)WRITE(6,1450)(S(JJ)/PRE,JJ=1,3),SS/PRE,(Q(JJ),JJ=1,3),QQ
1450  FORMAT('      S1          S2          S3          SS',//,
     1  4(1X,G13.7,1X),/,4(1X,G13.7,1X))
C
C  CHECK MAXIMUM NUMBER OF SSD CALCULATIONS
C
      IF(NSSD.GE.NMSSD)GO TO 660
C
C   KEEP POINTS AROUND BEST AND TRY AGAIN
C
      GO TO (631,640,651),IQDIM
631   S(3)=SS
      Q(3)=QQ
      GO TO 400
640   IF(QQ.LE.Q(IQMIN))GO TO 645
      IF(QQ.GE.Q(IQMAX))GO TO 644
      S(IQMAX)=SS
      Q(IQMAX)=QQ
      GO TO 400
644   IF(SS.GT.S(2))GO TO 631
      GO TO 651
645   IF(SS.LT.S(2))GO TO 631
651   S(1)=SS
      Q(1)=QQ
      GO TO 400
C
C  CONVERGED
```

```
C
660     IF(RR.LT.R(IRMIN))GO TO 680
670     SS=S(IRMIN)
        SSR=R(IRMIN)
680     DO 690 J=1,KOIN
        PN(J)=P(J)+SS*D(J)
        IF(PN(J).LT.PMIN(J))PN(J)=PMIN(J)
690     IF(PN(J).GT.PMAX(J))PN(J)=PMAX(J)
        DO 700 J=1,NFDIM
700     RR(J)=ROSAVE(J)
        RETURN
        END
```

APPENDIX F

STEAM GENERATOR NOISE MODEL

The noise model for the U-tube steam generator used in this work was derived using basic conservation equations in a way similar to that used by Ali (31) for his Model A. In the noise model, the heat transfer coefficient between the tube metal and the secondary side was considered stochastic and treated as one of the noise sources.

The model considers the steam generator devided into three lumps, as shown in Figure F-1. Lump 1 is the primary side water inside the tubes, considered as an well stirred tank. Lump 2 is the tube metal, which conducts heat from the primary water to the secondary water and steam. Lump 3 is the secondary side water and steam assumed to be at the saturation temperature.

The energy balance for Lump 1 can be written as

$$\frac{d}{dt}(MpCplTpo) = WpCpl(Tpi-Tpo) - UpmSpm(Tpo-Tm) \quad , \qquad F-1$$

where Mp is the mass of primary water inside the tubes,

Cpl is specific heat of the primary water,

Tpo is the temperature of the primary water,

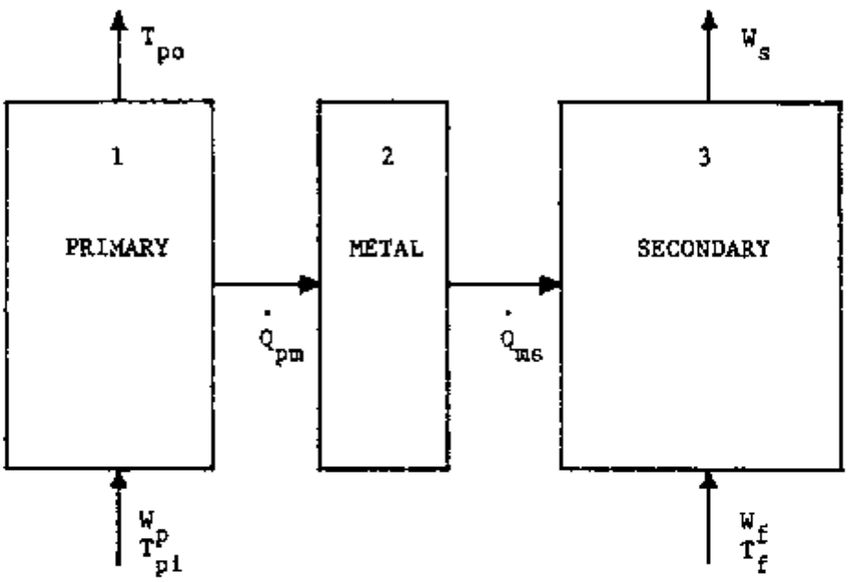

FIGURE F-1. Diagram of the lump parameter steam generator model.

Wp is the primary water mass flow rate,

Tpi is the primary water inlet temperature,

Upm is the primary water to tube metal overall heat tranafer coefficient,

Spm is the internal surface area of the tubes, and

Tm is the temperature of the metal tubes.

Considering Mp and Cpl constant during transients and writing Equation F-1 in terms of deviations from equilibrium yields

$$\frac{d\delta Tpo}{dt} = \frac{Wp}{Mp} + \frac{UpmSpm}{MpCpl}\,\delta Tpo + \frac{UpmSpm}{WpCpl}\delta Tm + \frac{Wp}{Mp}\delta Tpi \quad , \qquad F-2$$

where the "δ" indicates the deviation from equilibrium.

The energy balance for Lump 2 can be written as

$$\frac{Mmd\,(CmTm)}{dt} = UpmSpm(Tpo-Tm) - UmsSms(Tm-Ts) \quad , \qquad F-3$$

where Mm is the mass of the metal tubes,

Cm is the specific heat of the tube metal,

Ums is the overall heat transfer coefficient between the metal tubes and the secondary side, and

Ts is the secondary side temperature (saturation temperature).

Considering Cm constant during transients, writing Equation F-3 in terms of the deviation from equilibrium, and disregarding second order terms, yields

$$\frac{d\,\delta Tm}{dt} = \frac{UpmSpm}{MmCm}\,\delta Tpo - \frac{(UpmSpm + UmsoSms)}{MmCm}\,\delta Ums + \frac{UmsoSms}{MmCm}\frac{dTs}{dP}\,\delta P +$$

$$+ \frac{Sms(Tso-Tm)}{MmCm}\,\delta Ums \quad , \qquad \text{F-4}$$

where Ums was considered stochastic and was separated into a steady state part, Umso, and a fluctuating part, δUms; and the following linear approximation was used for the deviation from equilibrium of the saturation temperature as a function of the deviation from equilibrium of the secondary pressure, δP,

$$\delta Ts = \frac{dTs}{dP}\,\delta P \quad . \qquad \text{F-5}$$

For the secondary side, Lump 3, three basic concervation equations were used:

(a) Mass balance -

$$\frac{d}{dt}\,(Mw + Ms) = Wf - Ws \quad , \qquad \text{F-6}$$

where Mw is the mass of secondary water,

Ms is the mass of secondary steam,

Wf is the feedwater mass flow rate, and

Ws is the steam mass flow rate.

(b) Volume balance -

$$MwVf + MsVg = V \quad , \qquad \text{F-7}$$

where Vf is the specific volume of the secondary water,

Vg is the specific volume of the secondary steam, and

V is the total internal volume of the secondary side of the steam generator.

(c) Energy balance -

$$\frac{d}{dt}(MwHf+MsHg) = UmsSms(Tm-Ts) + WfHfi - WsHg \quad , \qquad \text{F-8}$$

where Hf is the enthalpy of the secondary water,

Hg is the enthalpy of the secondary steam, and

Hfi is the enthalpy of the feedwater.

After some algebraic manipulations of Equations F-6 through F-8, and also using the expressions

$$\frac{dVf}{dt} = \frac{dVf}{dP}\frac{dP}{dt} \quad , \text{ and} \qquad \text{F-9}$$

$$\frac{dVg}{dt} = \frac{dVg}{dP}\frac{dP}{dt} \quad ; \qquad \text{F-10}$$

one obtains the following equations:

$$\frac{dP}{dt} = \frac{UmsSms}{K}(Tm-Ts) + \frac{1}{K}\left[Hfi-Hg+\frac{VgHfg}{Vfg}\right]Wf - \frac{Vg\ Hfg}{KVfg}Ws \quad , \text{ and} \qquad \text{F-11}$$

$$\frac{dMw}{dt} = \frac{J}{k}\ UmsSms(Tm-Ts) + \left[\frac{J}{K}\left(Hfi-Hg+\frac{VgHfg}{Vfg}\right) + \frac{Vg}{Vfg}\right]Wf$$

$$-\left[\frac{JVgHfg}{KVfg} + \frac{Vg}{Vfg}\right]Ws \quad , \qquad \text{F-12}$$

$$\text{with} \quad J = \frac{1}{Vg}\left[\frac{V-MwVf}{Vg}\frac{dVg}{dP} + Mw\frac{dVf}{dP}\right] \quad , \text{ and} \qquad \text{F-13}$$

$$K = \left[\frac{dHg}{dP} - \frac{Hfg}{Vfg}\frac{dVg}{dP}\right]\frac{V-MwVf}{Vg} + \left[\frac{dHf}{dP} - \frac{Hg}{Vfg}\frac{dVf}{dP}\right]Mw \quad . \qquad \text{F-14}$$

Writing Equations F-11 and F-12 in terms of the deviations from equilibrium, disregarding the second order terms and considering that at the equilibrium the steady state feedwater and steam mass

flow rate are equal (i.e Wfo=Wso), one obtains

$$\frac{d\delta P}{dt} = \frac{UmsSms}{K}\delta Tm - \left[\frac{UmsSms}{K}\frac{dTsat}{dp} + \frac{Wfi}{K}\frac{dHg}{dP}\right]\delta P + \frac{Sms}{K}(Tm-Ts)\delta Ums$$

$$+\frac{Wfi}{K}\delta Hfi + \frac{1}{K}\left[Hfi - Hg + \frac{VgHfg}{Vfg}\right]\delta Wfi - \frac{VgHfg}{KVfg}\delta Wso \text{ , and} \qquad \text{F-15}$$

$$\frac{dMw}{dt} = \frac{JUmsSms}{K}\delta Tm - \frac{J}{K}\left[UmsSms\left(\frac{dTsat}{dP}\right) + Wfi\left(\frac{dHg}{dP}\right)\right]\delta P + \frac{JSms(Tm-Ts)}{K}\delta Ums$$

$$+\frac{JWfi}{K} + \left[\frac{J(Hfi-Hg)}{K} + \frac{Vg}{Vfg}\frac{Ms\left(\frac{dHg}{dP}\right) + Mw\left(\frac{dHf}{dP}\right)}{K}\right]\delta Wfi$$

$$-\left[\frac{Vg}{Vfg}\frac{Ms\left(\frac{dHg}{dP}\right) + Mw\left(\frac{dHf}{dP}\right)}{K}\right]\delta Ws \text{ ,} \qquad \text{F-16}$$

where all the parameters are calculated at the equilibrium value.

Equations F-2 , F-4 , F-15 and F-16 are the steam generator noise model equations.

VITA

Eduardo Lavenère Machado was born in Maceió, Alagoas, Brazil in 1946. He attended the college of engineering at the Universidade de Federal de Pernambuco, in Recife; majoring in Electrical Engineering. Since 1970 he has been employed by the Instituto de Pesquisas Energéticas e Nucleares in São Paulo. In 1976, Eduardo obtained his Master Degree in Nuclear Engineering from the Universidade de São Paulo. He came to the United States in 1978, to perform research in the area of Noise Analysis at the Oak Ridge National Laboratory, Instrumentation and Controls Division. He married the former Marcia M. O. Lopes in 1965, with whom he has two children: Paulo and Barbara.