THE UNIVERSITY OF MICHIGAN

COLLEGE OF ENGINEERING
Department of Mechanical Engineering

Progress Report

DIGITAL COMPUTER USE IN POWER PLANT DESIGN

G. V. Edmonson
J. S. Squire
J. L. Stokes
F. H. Westervelt
R. E. Hohmeyer
R. W. Albrecht

ORA Project 03760

under contract with:

CONSUMERS POWER COMPANY
JACKSON, MICHIGAN

COMMONWEALTH ASSOCIATES, INC.
JACKSON, MICHIGAN

administered through:

OFFICE OF RESEARCH ADMINISTRATION ANN ARBOR

December 1961

ensm

UMR0850

PREFACE

On February 1, 1959, Consumers Power Company and Commonwealth Associates Incorporated began a contract for research with The University of Michigan. The objective of this research program was to develop the art of utilizing digital computer equipment in an analytical manner to enable the optimization of power plane designs.

The research program is under the general supervision of Associate Dean Glenn V. Edmonson and under the technical direction of Assistant Professor Franklin H. Westervelt assisted by Dr. Robert Albrecht, Mr. Jon Squire, Mr. Robert Hohmeyer, and Mr. James Stokes.

Four major computer programs have been devised. They may be used separately, or linked together, to enable optimization of power plant designs. This report explains the computer programs as presently developed and indicates areas in which further work remains to be done.

The details of the logic involved in the computer analysis are presented by Professor Westervelt in a separate publication entitled Automatic System Simulation Programming. Details of the SIMULATOR PROGRAM are explained by Mr. Jon Squire in Appendix A. Details of the STEPWISE REGRESSION PROGRAM WITH SIMPLE LEARNING and of nonlinear estimation are explained by Mr. Robert Hohmeyer in Appendix B. Details of the COST ANALYSIS PROGRAM are explained by Mr. James Stokes in Appendix C. Details of the OPTIMIZATION PROGRAM are explained by Mr. Jon Squire in Appendix D.

During the past year, the research group conducted a training program for personnel of Consumers Power Company and Commonwealth Associates Incorporated. An outline of the portion of the training program designed to provide the background of mathematical statistics necessary for understanding the computational logic of the computer program is presented by Professor Westervelt in Appendix E. The material covered in other portions of the training program is presented in Appendices A, B, and D.

TABLE OF CONTENTS

TABLE OF CONTENTS (Concluded)

1. THE PROBLEM TO BE SOLVED

1.1 THE PROBLEM

The designer of a steam power plant is faced with a formidably complex task when he must decide upon the specifications of the components of the system. Only a few of the specifications—such as those of size, power, capacity, and tolerance of a particular component in a particular application—can be selected by following standard and well-accepted procedures. When the designer comes to such specifications as efficiency, loss, and operating point, he is faced with a range of possible values, or "free choices." He must now "play it by ear" and trust his own experience and his knowledge of the design of previous power plants.

Why does this create a problem? Simply because each of the values the designer selects will affect the operation of all the rest of the system and thus modify the over-all efficiency and the capital investment of the entire system. It is a very rare case when the designer can select an efficiency, a loss, or an operating point knowing that it will not affect the efficiency, the initial cost, and the operating cost of the whole system.

Until now there has been no practical way to solve this problem, to make the design of a power plant a science, rather than an art. The designer cannot select one "free choice" and then design the rest of the system to match. He has no way of determining that the particular value he selected will result in the greatest efficiency for the complete design. What value of what "free choice" should be select to get the greatest system efficiency?

The logical method of solving this problem would be to look over all possible combinations of all "free choices," try out those combinations which appear desirable, and finally select that combination which gives us the lowest total cost.

Such a method is clearly beyond the capability of a single designer, or even a whole group of them. It would take years to perform such an analysis. But this method can well be adapted to a modern high-speed digital computer which can perform a very large number of computations in a very short time.

1.2 THE SOLUTION

This problem has been solved through the creation of a series of computer programs:

a. SIMULATOR PROGRAM—translates a verbal description of the flow diagram of the system into a computational procedure, in this case the Heat-Balance Program, which enables us to solve for the efficiency of the system.

b. STEPWISE REGRESSION PROGRAM WITH SIMPLE LEARNING—derives formulas for predicting costs and efficiencies from given sets of data. These formulas are used in the Cost Analysis Program and the Heat-Balance Program.

c. COST ANALYSIS PROGRAM—uses the efficiencies calculated by the Heat-Balance Program and the cost formulas derived by the Stepwise Regression Program to predict the equipment and operating costs of the design.

d. OPTIMIZATION PROGRAM—selects various combinations of "free choices" for the Heat-Balance Program to determine the combination that will give us the most economical design as defined by the Cost Analysis Program.

These computer programs may be used separately to solve a variety of problems, or they may be linked together to optimize the design of a whole system.

Provided information from several sources is available, it is possible that by the time this report is distributed, each of these programs will have been tried out separately. We will soon try them linked together by running a Heat-Balance Program created by the Simulator Program with the Cost Analysis and Optimization Program. Two "free choices" and only a few cost formulas from the Stepwise Regression Program will be employed in an attempt to analyze the design which will provide maximum economy over a thirty-five year period. Additional work on the writing of element descriptions, and refinements in the Cost Analysis Program, will enable us to use many more "free choices" in the design analysis.

This report explains in very general terms how a digital computer can be instructed to perform such an analysis. The computer needs a completely detailed description of this analysis for this purpose—so detailed that the essential principle involved is easily lost in the forest of instructions. Because this report is intended for a wide audience, rather than for computer specialists, the body of the report deliberately omits these details and concentrates on the principles involved.

The techniques of preparing instructions for the computer and the details of its operation are discussed in the Appendices attached to the body of the report.

2. THE SIMULATOR PROGRAM

This particular program produces a computational process, or another computer program, to enable us to analyze a system. This can be any system that we can specify. Let us describe how this can be used to analyze a portion of a steam power plant design, in other words, to produce a Heat-Balance Program for a steam-water system.

2.1 FLOW DIAGRAM

To design a power plant, one ordinarily starts with a flow diagram. The same thing is done when we use a computer to analyze the efficiency, or the economy, of a particular power plant design. On this flow diagram we must indicate all the components essential to the operation which add energy or decrease it. Thus, pipes are included because energy is lost in them, but pipe hangers and turbine foundations are not. Such a diagram might resemble the one in Fig. 2.1, but there is no theoretical limit to the number of energy components. We may make the plant as complicated as we wish.

The computer, of course, cannot read a flow diagram. Therefore, the flow diagram must be described for it. Ordinary words typed with a keypunch onto IBM cards will do. There is only one limitation; the words must not be longer than six alphanumeric characters, i.e., combinations of letters and digits. For this reason TURBINE must be shortened to TURBIN or T1, T2, or T3.

There are frequently several components in the system with the same designation, such as PUMP, but they come in different sizes and capacities and are connected to different parts of the system. To differentiate one pump, or one condensor, from another, the name may be followed by a comma and an additional six-character identifying name or symbol, for example, PUMP, A or PUMP, ALT—for alternate pump.

But how do we describe the links, or connections, between components? This is done simply by typing, in parentheses, following the component name, an additional name or symbol identifying the point of attachment of the connection or link. For example, a pump inlet, outlet, and shaft are points of attachment. They would be described as PUMP,A(INLET), PUMP,A(OUTLET) and PUMP,A(SHAFT).

These points of attachment must be connected to points of attachment on another component. Thus PUMP,A(OUTLET) might lead to a pipe which in turn leads to a condensor inlet. These connections are indicated by the simple

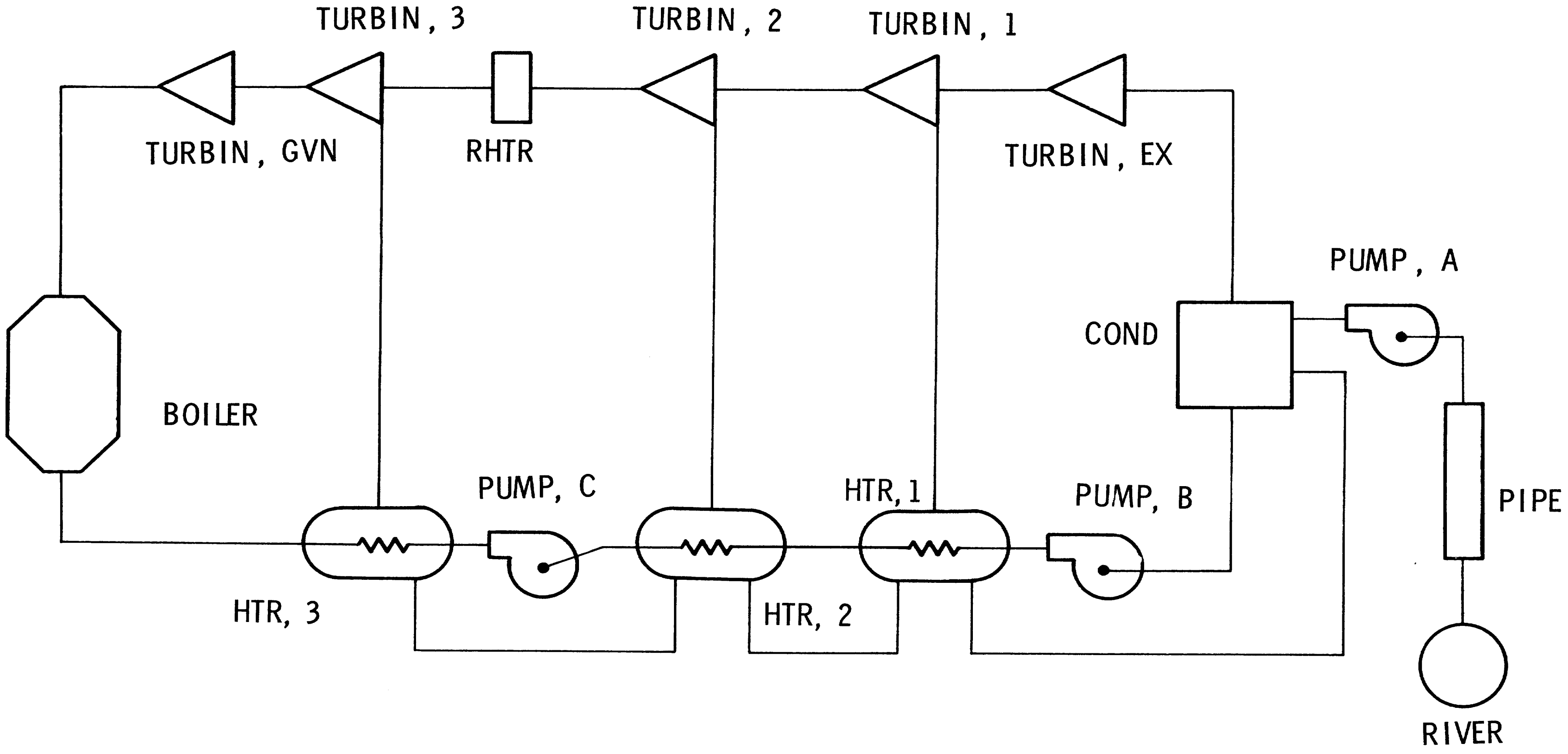

TURBIN, 3
TURBIN, 2
TURBIN, 1
TURBIN, GVN
RHTR
TURBIN, EX
PUMP, A
COND
BOILER
PUMP, C
HTR, 1
PUMP, B
PIPE
HTR, 3
HTR, 2
RIVER

Fig. 2-1.

word TO set off with commas on either side, or PUMP,A(OUTLET),TO,PIPE,TO, COND,B(INLET).

But what if the connection of a component is multiple; what if the outlet of a pump is divided between two condensors? All we need to do is to differentiate between the points of attachment, just as we identified the components, by adding a comma and an identifying word or symbol. Thus PUMP, A(OUTLET,2),TO,COND,B(INLET,1).

Now the flow diagram of Fig. 2.1 might be described for the computer as follows:

RIVER,TO,PIPE,TO,PUMP,A(INLET)
PUMP,A(OUTLET,1),TO,COND(IN), etc, etc.

Have we omitted anything? Forgotten any connections or attachments? We can ask the computer to check the description of the flow diagram for consistency as it prints out the description. The computer will do this by simply sorting through these connection statements looking for duplicate terms. That is, if PUMP,A(OUTLET,1) appears twice in the flow diagram we have either inadvertently typed out PUMP,A(OUTLET,1),TO,BOILER,A(INLET) twice, which is merely redundant, and of no importance, or we have PUMP,A(OUTLET,1) also connected to another attachment on another component, which is illogical. In case the computer discovers this second type of duplication, it adds AMBIGUOUS to the printed sheet it is producing.

2.2 PARAMETER LOGIC

All we have done so far is to describe the flow diagram of the power plant design to the computer. We must now go further and describe to the computer what performance data, taken at numerous points over the operating range, we can measure on each component, at which attachment on the component we take this measurement, and how this measure of performance is related to other measurements at other attachments. For example, on a constant-speed motor we can measure the horsepower at the shaft, or HP(SHAFT). The HP(SHAFT) is, of course, related to the torque at the shaft, or TORQUE(SHAFT).

If we are given the value of one of these measurements, logic indicates that we can compute the other. The computer needs to be told this. It has no intuition and can perform no inferential leaps. We must explain to it exactly what the logical relationships are. Thus, we tell it that if we know the horsepower at the shaft, then we can compute the torque at the shaft. Or, HP(SHAFT),THEN,TORQUE(SHAFT).

Further, we need to explain to it how we can find the torque at the shaft from the horsepower at the shaft. Therefore, we follow the general logical satement with the more specific one, ♁TORQUE(SHAFT)♁ = 26.4*♁HP(SHAFT)♁, or the torque at the shaft equals 26.4 times the horsepower at the shaft. The symbols ♁ serve as arbitrary brackets and indicate that real number values will be provided or will be computed to provide the value of the word within these symbols. The asterisk (*) stands for the operation "multiply."

2.3 ELEMENT DESCRIPTION

Now we must describe for the computer each component, or element in the system, including its name, its attachments, and the logical relationships of the parameters, or performance data, that we can measure on the component.

First we must tell the computer what we are about to do. Therefore we write, ELEMENT DESCRIPTION.

Next, obviously enough, NAME OF ELEMENT = MOTOR.

Next we list the attachments of the element. For example,

```
ATTACHMENT NAMES = SHAFT, INPUT, OUTPUT
```

If there are multiple connections at a single attachment, some of the parameters are likely to be identical. For example, the pressures at multiple outlets of a pump are likely to be the same. To simplify the statement collections we can indicate these identical parameters under the heading BROAD SCOPE PARAMETERS = PRESS. But we must be careful; if the parameter is declared "broad scope" for one element, it must be so declared in every element description in which it appears.

In some cases we will want to try out various types of components in conjunction with a single design arrangement. It will save a great deal of time if we can indicate to the computer that we plan to utilize the element being described in several design arrangements. We can do this by next writing PERMANENT. When the computer reads this, it will preserve the element description in its memory to use it in another analysis rather than discarding it.

Now we will tell the computer each of the steps of parameter logic that we have for this element. This follows the form discussed in Section 2.2, but we must precede each general and specific logical statement with the phrase STATEMENT COLLECTION. For the motor we are describing, these statement collections might appear as follows:

```
STATEMENT COLLECTION
     HP(SHAFT),THEN,TORQUE(SHAFT)
     ♁TORQUE(SHAFT)♁ = 26.4*♁HP(SHAFT)♁
```

```
STATEMENT COLLECTION
     TORQUE(SHAFT),THEN,HP(SHAFT)
     ♁HP(SHAFT)♁ = .038*♁TORQUE(SHAFT)♁
```

And in a number of statement collections, especially those dealing with the designer's "free choices," we must indicate that the parameter depends on the values of other parameters and must be computed from a formula we may not yet have. For example, we can compute the electric power at the terminals of a motor if we know the horsepower and the torque of the shaft and have a formula for the efficiency of the motor. A statement collection of this type might appear as:

```
STATEMENT COLLECTION
     HP,TORQUE(SHAFT),THEN,ELECPR(TERM)(electric power at terminals)
     ♁ELECPR(TERM)♁ = ♁HP(SHAFT)♁/EFFMTR.(♁TORQUE(SHAFT)♁)
```

The formula indicated by the term EFFMTR. will be devised by the computer itself from the Stepwise Regression Program described in Section 3.

If an single statement collection is going to appear under all similar attachments of components in the system (as the relationship of torque, rpm, and shaft horsepower of all shafts in the system will) we can save a great deal of time by typing this statement collection only once under NAME OF ELEMENT = UNIVERSAL.

Finally, we must tell the computer either DESCRIPTION FINISHED, or ANOTHER ELEMENT FOLLOWS.

2.4 INPUT PARAMETERS

Now that the flow diagram, the parameter logic, and the elements have all been described to the computer, we can indicate to the computer what data, or parameters, are known for each component and at which point of attachment of the component these data are measured. Thus we may know the pressure at the outlet of a pump, or the temperatures to be expected of the river water.

These given parameters are described to the computer by beginning with the statement INPUT PARAMETERS. This statement is followed by a name for the parameter, such as FLOW, or TEMP, or PRESS (for pressure). The parameter name must be followed by the name, in parentheses, of the specific attachment of the specific component where this is measured, for example, PRESS (PUMP,B(OUTLET,2)).

If the particular attachment is not specified, as in PRESS(PUMP(INLET)), the computer will assign pressure as an input parameter to the inlet of every pump in the system. This can save time in describing the input parameters, but if the input parameter is described this way by accident, the program produced by the computer (described in Section 2.5) will be wrong.

2.5 DESIRED RESULTS

Ultimately we should like to ask the computer to give us the answer to the question, how many kilowatt-hours of electricity will a power plant of this design produce for each ton of coal? In other words, what is the unit-net-heat-rate at the generator terminals? We ask the computer this by typing,

```
DESIRED RESULTS
     UNHR(GENER(TERMIN))
```

The computer cannot give us an answer to this question in numbers because we have not yet provided it with numbers to work with. We have, however, told the computer, as we explained in Section 2.4, what number values we can supply for certain parameters. We have also told it, as explained in Section 2.2, how these parameter values are logically related. Thus, we have told the computer, in the STATEMENT COLLECTION under the element description of the generator, BTU(FUEL),THEN,UNHR(TERMIN). If we work our way back down the flow diagram from the desired result toward the given INPUT PARAMETERS we can eventually describe how to calculate the desired result from the given input parameters. In other words, we have created a computational technique, or computer program.

This sounds like a very simple procedure. In principle it is simple; but in actual practice it is tedious and exasperating. It is very much like working through a maze with a great many alternate alleys. For a simple system it would take days to do it; for a complex system, weeks or months. And by the time we got through we would be able to calculate only a few of the many different results we desire.

Fortunately, the computer can do this sort of maze-running very rapidly. If we ask a digital computer to produce one single program to calculate all the parameters that can be calculated at all the points in the system, it can do this in a matter of minutes. And it learns as it runs; that is, whenever it runs into a blind alley, it backs up and starts again. But this time, by assigning a lower probability of selection to that particular alley, it remembers that this alley led only to a dead end. The method by which the computer is told how to do this is described in detail in Professor Frank Westervelt's dissertation.*

*Westervelt, Franklin H., Automatic System Simulation Programming, The University of Michigan, 1960.

The output of this maze-running is a series of logical steps leading from the INPUT PARAMETERS to the DESIRED RESULTS. In other words, the computer produces a computational process, or Heat-Balance Program, completely ready for the computer—no more hand work needs to be done. This is a very desirable answer to our original question. We can now insert real number values in place of the INPUT PARAMETERS and get a real number answer back in a matter of minutes.

A more detailed description of this Simulator Program is presented in Appendix A.

3. THE STEPWISE REGRESSION PROGRAM WITH SIMPLE LEARNING

In analyzing either the efficiency or the economy of a power plant design it is necessary to describe in computational terms the relationship between a number of interacting variables. For example, the efficiency of a single stage of a turbine varies with the torque of the output shaft, the temperature of the steam at the output, the pressure of the steam at the input and the output, and the rate of flow of steam at the input, the extraction point, and the output, and the number and arrangement of turbine stages. In the same way the cost of the turbine depends not only on its efficiency, but also, among other things, on its material, and its size. Each of these variables influences the cost in a different way.

For this reason we cannot insert in the STATEMENT COLLECTION a simple mathematical equality such as:

```
HP(SHAFT),THEN,TORQUE(SHAFT)
♁TORQUE(SHAFT)♁ = 26.4*♁HP(SHAFT)♁
```

The specific logical statement appearing in the second line is a simple regression equation, or an equation which enables us to predict the value of the torque if we know the value of the horsepower at the input. We could plot this relationship with a simple straight line on a graph.

But when several variables interact, we cannot use a simple regression equation. If we superimpose the curves of efficiency of our turbine versus each of the variables that affect it, we get a very complicated-looking graph. The computer cannot, of course, interpret graphs as such. It can, however, predict the efficiency of the turbine from these variables if it can generate a multiple regression equation, which is a mathematical model of this curve. This equation must be derived from the data.

We now give the computer data in the form of coordinates of selected points along each curve of each graph of a real turbine whose efficiency is already known. From these coordinates the computer now derives a multiple regression equation which will enable it to predict the efficiency of a new and different turbine from any given set of values of the parameters of pressure, temperature, torque, and flow.

The computer derives this multiple regression equation by another form of maze-running. It begins by selecting a given parameter, such as pressure, and computes the coefficient (or multiplier, like the coefficient 26.4 by which horsepower must be multiplied to get the value of torque). If this one parameter and its coefficient enables us to predict efficiency from pressure with reasonable accuracy, the computer adds another parameter and computes the coefficients of both parameters at once to attempt to provide a prediction with less error. In this way, step by step, the computer tries out the combinations of parameters and their computed coefficient values, and, by rejecting the less successful combinations and keeping the more successful, it learns which combination produces the best prediction of the real efficiency of the real turbine. This combination is then used as the equation to predict the efficiency characteristic of a new turbine throughout its range. This equation is the mathematical model of the complicated curve.

If, when a parameter is added to the combination, the prediction becomes less accurate, the computer goes back to the best previous combination and adds a different parameter to it. In computer language this technique is called "branching."

A more detailed description of this program is presented in Professor Frank Westervelt's dissertation.* Further refinements in this program to provide for nonlinear estimation are present in Appendix B.

*Westervelt, Franklin H., Automatic System Simulation Programming, The University of Michigan, 1960.

4. THE COST ANALYSIS PROGRAM

We can now find the efficiency of a given design very easily. However, an efficient design may be so costly to construct that it would be uneconomical. Further, we will not be running this plant at 100% of its capacity all the time, and if the cost of operating the plant at less than capacity looms too large, we may find that a design of less efficiency is nevertheless the better design in terms of economy. Therefore, we must find out how much the equipment of our design will cost and how much it will cost to operate a plant of this design.

4.1 EQUIPMENT COST

Since most of the important components of a power plant are specially built items, we cannot merely look up the price in a manufacturer's catalogue. Just as the efficiency of a component depends on a set of interacting variables, so the cost depends on a set of interacting specifications. Again we are faced with the problem of deriving a multiple-regression equation.

Once again we can call on the computer to do this for us through the Stepwise Regression Program described in Section 3. As we did before, we give the computer coordinates along the graphs of cost versus a variety of specifications of existing components. And, as before, the computer tries out various combinations of specifications and computes the coefficients until it arrives at the combination of specification parameters and coefficient values which will best predict the cost of the existing component. By inserting numerical values for the specification parameters, we can predict the cost of each item of new equipment, and by adding these costs, get the total equipment cost of the design.

4.2 OPERATING COST

Now that we know how to find out how much the equipment will cost, in what way can we find out the cost of operating the plant we are designing? The best way to compare the operating costs of two different designs is to determine the costs of the coal required to run each plant during its expected lifetime. We will arbitrarily choose a number of years as the expected lifetime of a power plant.

As the power plant grows older it will probably be operated less and less at capacity. Newer and more efficient plants will have been added to the over-all distribution system and will be providing a larger and larger share of the power. Therefore, we will now give the computer the number of hours we think this plant will operate at 100%, 85%, 70%, 55%, 40%, 25%, and 0% (shut down for maintenance) of its capacity during each five-year interval of its expected lifetime. We expect to develop a formula which will enable the computer to predict the number of hours of operation at various capacities in each year.

As we explained in Section 2.5, we already have a computational process, or computer program, which will enable us to find the unit-net-heat-rate, or the number of Btu's of fuel required for each kilowatt-hour at each percentage of capacity. This makes it easy to compute the number of Btu's of fuel required for each five-year interval of the expected life of the plant.

But the cost of fuel is likely to change with time. So we determine the cost of coal in past periods, and by projecting these costs into the future, we can compute the probable cost of coal used by the plant.

Other operating and maintenance costs must be taken into account. By adding these to the fuel costs, we get the operating cost for each five-year interval of the expected life of the plant.

4.3 TOTAL COST

The total cost of the equipment, must be multiplied by a specified fixed charge rate to account for interest, taxes, insurance, and the return on capital investment. The product of this multiplication, added to the operating cost for a given year, will give us the annual revenue required in that year to operate and pay for the plant at a predetermined rate of return.

These annual revenue requirements are then translated into "present worth" and summed to give us the total cost of the design. This enables us to compare the total costs of two different designs on the same basis of "present worth."

A more detailed description of the Cost Analysis Program is presented in Appendix C.

5. THE OPTIMIZATION PROGRAM

Thus far we have only created computational techniques, or computer programs, which tell us how to find various answers. How now do we combine these programs to use the numerical values we can supply and solve for the numerical results we desire?

First, of course, we can given real numerical values to the Stepwise Regression Program so that the computer can derive the formulas that tell us how to predict the efficiencies of individual components and their costs.

These formulas we then can add to the element descriptions of Section 2.3 in the form of STATEMENT COLLECTIONS. Then through the Simulator Program we can have the computer devise its own Heat-Balance Program to determine the efficiency and the equipment cost of the entire design from the various INPUT PARAMETERS that we intend to supply.

In addition we have devised a Cost Analysis Program for predicting the operating costs from a given number of hours of operation at various percentages of capacity. However, we need to supply the unit-net-heat-rate of the design. This value, of course, can be provided by the Heat-Balance Program. The operating costs and equipment costs added together will give us the economy of the total design.

All we need to do is to choose the values of the input parameters. But this is where we began. There are so many choices available to us! Of course we do have a big advantage because now we can compute the efficiency and economy of our combination of "free choices" very rapidly, but there still remains a host of "free choice" combinations we could try out.

The computer is still available to us; why not let it try out these combinations in some logical fashion? We can do this with Optimization Program.

To do this we give the computer, not a set of fixed INPUT PARAMETER values, but a range of values. We can specify the range of values within maximum and minimum limits, as PRESS(PUMP(OUTLET)) = 100PSI,MIN,TO,150PSI, MAX. Or we can set the limit of a combination of values, as pressure times temperature must not exceed a certain value, PRESS*TEMP = 90000MIN,TO, 150000MAX. Or we can set a limited number of permissible values, as TEMP (TURBIN(INLET)) = 900,950,1000,1050. Some of the input parameters are obviously going to fixed, that is, only a single value will be supplied, because of certain design features.

Following the procedure set up by the Optimization Program, the computer will now select a random combination of parameter values and solve for the efficiency and total cost of the entire design. Why a random combination? Simply because we wish to investigate various possible combinations of parameter values, and, contrary to what seems to be common sense, a random selection of values, combined with simple learning, is a more efficient way of hunting for the best combination than is systematic selection. Selecting the values systematically would eventually lead us to the best combination, but random selection enables us to explore interactions between variables more efficiently because we cover a greater range of value combinations in less time.

Once the combination of parameter values has been selected, the computer will then increase the value of one parameter and solve for the total cost again. If the cost is lower, the computer will increase the value of this parameter the next time it changes it; if the cost is higher, it will decrease the value. The computer tries out each parameter in turn, learning whether its value should be increased or decreased to get a lower total cost.

With the direction of change of each parameter value determined, the computer now changes all the parameter values at once by a small increment and solves for the total cost again. If this cost is lower than the first cost, it changes all the parameter values again by a larger increment, and solves again. It keeps changing the parameter values by larger and larger steps until the cost is no longer reduced.

The computer then selects another random combination of parameter values, but this time it refuses to select values that it has already tried, and repeats the same process over again. It keeps repeating this process with new random combinations of values until it has tried out a reasonable sample of all the values within each range of parameter values we have given it, and has found the combination of these values that will produce the lowest total cost.

In this way the computer works closer and closer to that combination of "free choices" which will give us the most efficient and economical power plant from this particular flow diagram.

There is an important advance being comtemplated for this Optimization Program. If the element descriptions themselves can be modified in this program, a completely new flow diagram can be tried out. This technique is being developed by writing the element descriptions in a modified way.

Further details of the Optimization Program are presented in Appendix D.

6. FUTURE APPLICATIONS

Each of the programs described in this report can be used separately or linked with the other programs to solve a variety of problems. To what sorts of problems might they be applied?

It is clear that the Simulator Program can be used to analyze any system that can be adequately described to it. It should be easy to apply it to transmission networks, either gas or electric. But the system might just as well be a biological or social, rather than a physical, system. If the complex of a river system can be described, this program could well produce a computational scheme for analyzing the effects of population growth and industrial development on the water purification, sewage treatment, and conservation requirements of the entire drainage area. If a customer relations system can be described, an analysis of the effects of various changes of policy should be possible.

In like manner, the Stepwise Regression Program with Simple Learning, with its proposed refinement to handle nonlinear estimation, can be used, where mass data are available, to predict the effects of a number of interacting variables. Demand figures and maintenance and repair costs may be analyzed in this way. The program has already been used to predict the incidence of disease from a mass of biological data and to predict personality characteristics from a mass of psychological data.

And, as explained in this report, where management is faced with a number of alternatives, the Optimization Program can be a very useful tool.

These computer programs have other potential advantages. It is quite clear that they can exert a powerful educational influence. They enable one to perceive relationships that were formerly obscure and thus provide a new understanding of complex phenomena. If the element descriptions discussed in this report could be made a standard part of every manufacturer's catalogue, designers would be able to analyze a variety of configurations very swiftly and economically. Finally, these programs, and especially the element descriptions that they employ, preserve for the benefit of a new generation of designers the accumulated knowledge and skill which has heretofore been lost to a concern as an experienced man retires from the organization.

APPENDIX A

DETAILED INSTRUCTIONS ON THE USE OF THE SYSTEM SIMULATOR PROGRAM

APPENDIX A. TABLE OF CONTENTS

Page

1. INTRODUCTION

Many fields of engineering are concerned with problems of analysis as a guide to design. When dealing with a particular component, this analysis can be carried out by combining physical laws with the aid of mathematics and a slide rule. If the problem is expanded to the analysis of a group of components operating together as a system, it becomes necessary to find more powerful techniques. Simulation is one such technique. To analyze by simulation one chooses a set of operating conditions in such a way that the behavior of a system is completely determined. The behavior of the system can then be evaluated for this particular set of operating conditions. By specifying various operating conditions, the general behavior of the system can be determined.

Once a step-by-step procedure has been set up for determining the behavior of the system, many sets of operating conditions must be calculated. Here a digital computer is of tremendous aid. The computer is an accurate and fast tool for doing the calculations required for simulation.

Specific attention will now be given to the problem of simulation. By simulation we mean the process of representing physical parameters by numerical quantities and using these numerical quantities in mathematical expressions of physical laws. In other words, any physical parameter such as temperature, pressure, flow, voltage, or current can be represented by a number which is read off the appropriate type of meter or gage.

There are three basic things that must be known in order to do a simulation. First, the parameters for which values will be given must be listed. Second, the parameters which are to be calculated must be listed. Third, an algorithm or calculation procedure must be found which yields the values of the unknown parameters when known parameters are given.

One of the objectives accomplished by this research project was the development of a computer program called The System Simulator to make simulation of systems relatively easy. The philosophy used during the development of the System Simulator Program (hereafter referred to as The Simulator) was to require the user to do as little as possible yet keep the structure of The Simulator independent of the type of problem. Thus The Simulator can work on problems involving mechanical systems, electrical systems, chemical systems, or even economic systems.

The average user of The Simulator will not have to know a computer language, nor will this user be required to specify any form of algorithm for doing the simulation. The Simulator does both of these tasks. The basic information required by The Simulator is as follows: (1) A list of the parameters for which values will be specified (the specific name "Input Parameters" is given to this type of parameter). (2) A list of parameters for which values are to be calculated (the specific name "Desired Results" is given to this type of parameter). (3) An accurate description of how the various components of the system are tied together (the name "Connection" will be used for any statement which describes a point where two components are attached).

Having been given Imput Parameters, Desired Results and Connections, The Simulator will produce an algorithm or procedure for calculating the values of Desired Results when given numerical values of the Input Parameters. The Simulator then produces a computer program to perform the algorithm. The computer program produced by The Simulator plus a set of data consisting of numerical values for the Input Parameters is submitted to a computer. The results of this program computation are the numerical values for the Desired Results. The same program may be used with different values for the Input Parameters. The relation between the values of the Input Parameters and the values of the Desired Results can be interpreted to determine the behavior of the system.

Before using The Simulator to generate simulation programs, a library of mathematical models of various components or elements must be prepared for The Simulator. (The more general name "Element" will be used instead of component.) Typical Elements are turbines, pumps, pipes, resistors, generators, etc. The Element is the basic building block of a system that is to be simulated. The mathematical model as it appears in The Simulator's library is called an "Element Description." An Element Description once written may be used any number of times in any number of different systems. More detail about writing Element Descriptions is given later.

2. CONNECTION STATEMENTS

To define unambiguous connections between elements, specific points on each element (called "attachments") are given names. For example Fig. A-1 shows an element called PUMP. This element has three attachments: INLET, OUTLET, SHAFT. To define a connection of this pump to some other element we may state an element name, attachment name, the word TO, another element name, and another attachment. An example of this type of connection statement is PUMP(OUTLET)TO,COND(CIRWIN). See Fig. A-2. Note that the order of stating the connected points is unimportant.

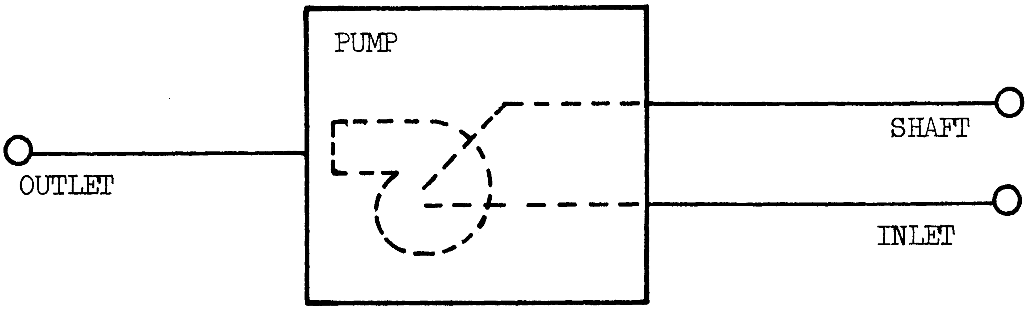
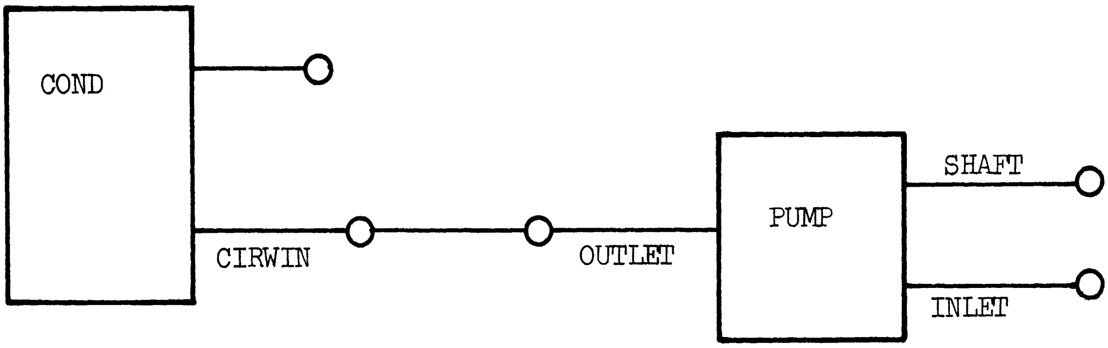


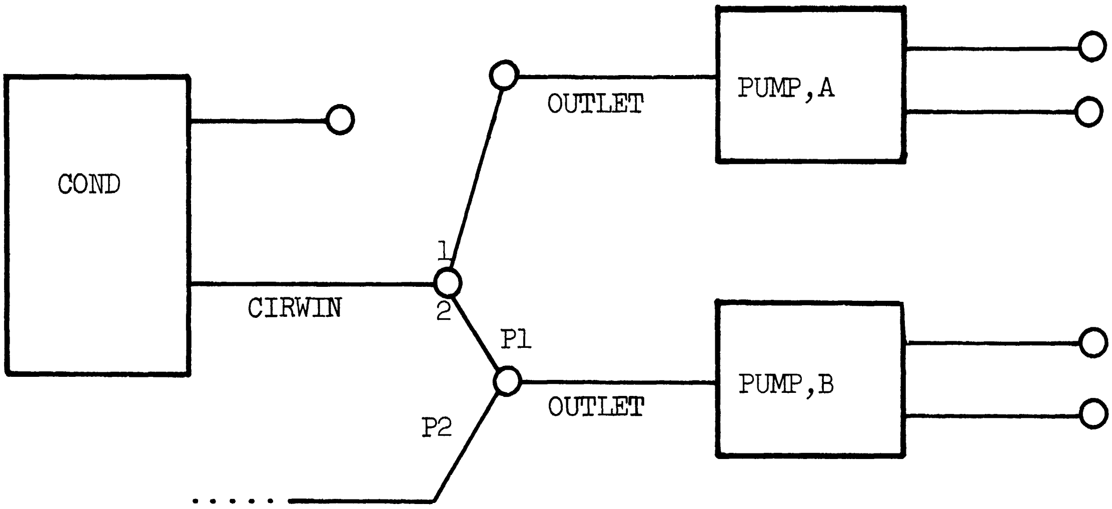
Fig. A-1.



COND(CIRWIN)TO,PUMP(OUTLET)

Fig. A-2.

Since it is possible to use the same element more than once in a system, a second name called an "Element Identifier" may be used. The Element Identifier is placed after the element name with a separating comma. See Fig. A-3. If more than one connection is to be made to a particular attachment of a particular element, an "Attachment Identifier" must be used for each connection. The Attachment Identifier follows the attachment name in brackets, with a separating comma.

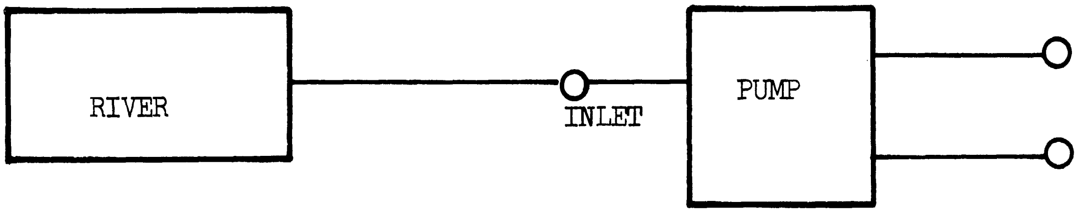


PUMP,A(OUTLET)TO,COND(CIRWIN,1)
PUMP,B(OUTLET,P1)TO,COND(CIRWIN,2)

Fig. A-3.

There is a slight restriction on all types of names mentioned above. For ease of manipulation we require all names to be six or less alphanumeric characters. Alphanumeric characters are the decimal numbers 0 through 9 and the letters A through Z.

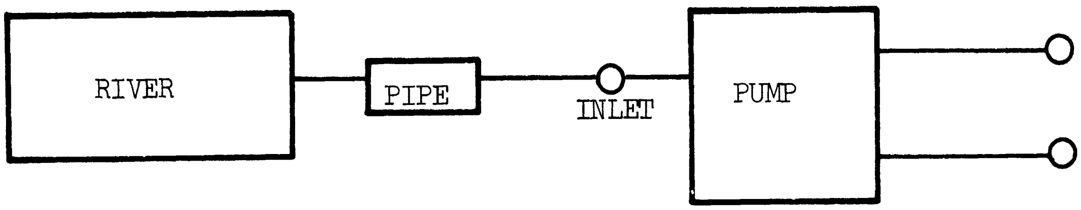
There are two types of elements for which a shorthand notation may be used. They are elements with one or two attachments. Typical unary elements are pure sources or sinks such as a river or a coal mine. Unary elements are recognized by the fact that no attachment name is given in the connection statement involving this element. If an attachment name was used it would be the single number 1 (see Fig. A-4). Binary elements such as pipes or wires



RIVER,TO,PUMP(INLET)

Fig. A-4.

have two attachments, named 1 and 2. A binary element may be used without specifying attachment names if it appears between the TO connectives. See Fig. A-5.



RIVER,TO,PIPE,TO,PUMP(INLET)

Fig. A-5.

A summary of all possible connections (CONN) is given below. The following abbreviations are used:

Element name = EL
Element identifier name = EID
Attachment name = AT
Attachment identifier name = AID

Let CONN be any form below:

```
EL
EL,EID
EL(AT)
EL,EID(AT)
EL(AT,AID)
EL,EID(AT,AID)
```

then allowable connection statements are,

CONN,TO,CONN

and

$\mathrm{CONN}_1,\mathrm{TO},\mathrm{CONN}_2,\mathrm{TO}\ \ldots\ \mathrm{TO},\mathrm{CONN}_n$

This completes the techniques for specifying the connections between the elements of the system. See Fig. A-6 for general example.

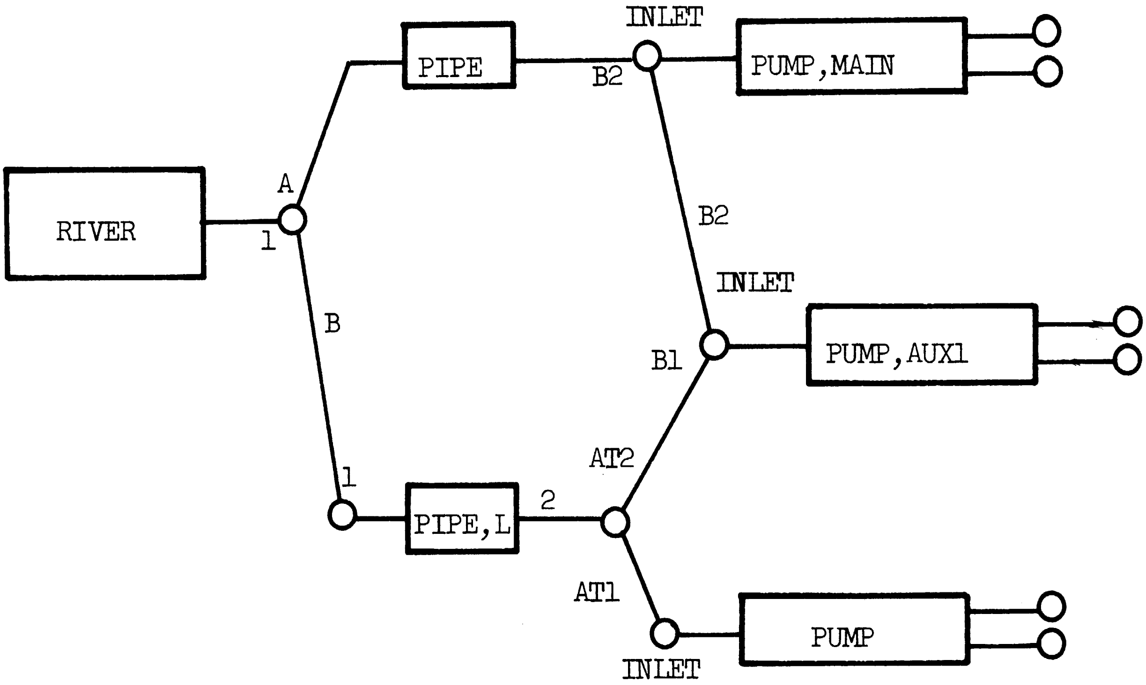


Fig. A-6.

```
CONNECTIONS
  RIVER(1,A)TO,PIPE,TO,PUMP,MAIN(INLET)
  RIVER(1,B)TO,PIPE,L(1)
  PIPE,L(2,AT2)TO,PUMP,AUX1(INLET,B1)
  PIPE,L(2,AT1)TO,PUMP(INLET)
  PUMP,AUX1(INLET,B2)TO,PUMP,MAIN(INLET,B2)
```

3. INPUT PARAMETERS AND DESIRED RESULTS

Once the system has been specified, further information must be given about the physical parameters which are known as "Input Parameters," and about the parameters which are to be calculated "Desired Results." Examples of known parameters are output voltage of a generator, pressure at the inlet of a turbine, temperature of a river, etc. Each input parameter and desired result consist of a parameter name (VOLT, PRES, TEMP, etc.) and a specific point in the system (as given in the connection statements). Input parameters and desired results have exactly the same form. When the simulator encounters the declaration: INPUT PARAMETERS, it interprets the succeeding statements as input parameters until another declaration is found. The declaration for desired results is DESIRED RESULTS. Examples of single input parameters, or desired results, are:

```
TEMP(COND(CIRWIN))
PRES(PUMP,A(OUTLET))
FLOW(PUMP,B(OUTLET,P1))
TEMP(RIVER)
```

Notice that the exact location of the place where the parameter is to be measured is given by any form: CONN, which was defined on page A-7. Let PAR be any parameter name, then:

```
PAR(CONN)
```

is a statement of an input parameter, or desired result.

There are several shorthand forms which are allowed for the convenience of the user. First, up to twenty parameters can be specified at the same point in the system with the connection point given only once. For example:

```
PAR1, PAR2, PAR3,...,PARN(CONN),
```

where PAR1,...,PARN are parameter names.

The second shorthand notation allows a parameter to be specified at many different points in the system while writing the parameter only once. This is a convenient way of specifying a desired parameter at every attachment of an element. For example:

```
PRES(TURBIN)
```

would specify the pressure at every attachment of the element TURBIN. In fact, if there were more than one TURBIN in the system, distinguished by element identifiers, the above parameter statement would apply to every attachment of every element called TURBIN. The general rule is stated as follows.

When specifying the location of a parameter, any designation which is omitted will be considered as if it matched every corresponding designation in the connection statements. A designation refers to element name, element identifier, attachment name, or attachment identifier. A parameter may be given at every point in the system by the single statement

```
PAR( ).
```

These two shorthand notations can be combined in any way that the user may desire.

A summary of some possible input parameter and desired result statements is given below:

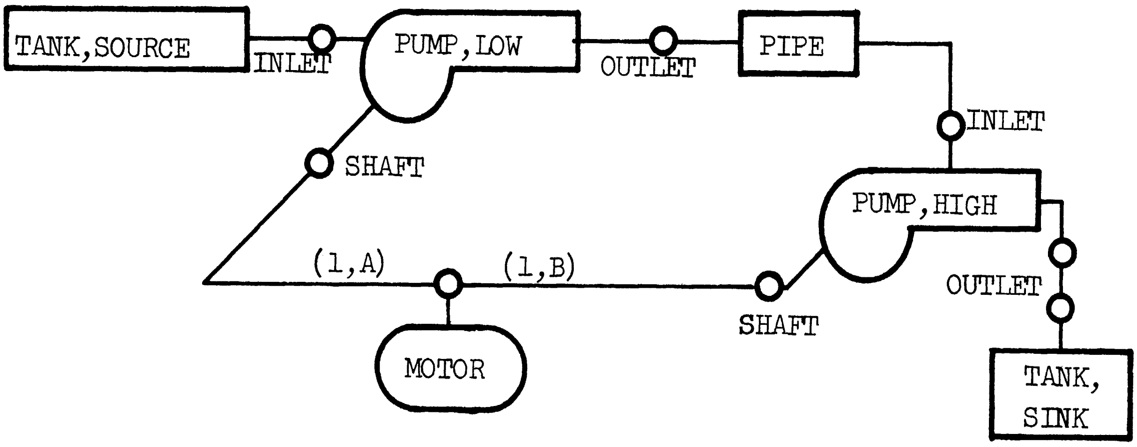
```
PAR 1,...,PARN( )
PAR 1,...,PARN(EL)
PAR 1,...,PARN(,EID)
PAR 1,...,PARN((AT))
PAR 1,...,PARN((,AID))
PAR 1,...,PARN(EL,EID)
PAR 1,...,PARN(EL(AT))
PAR 1,...,PARN(EL,EID(AT))
PAR 1,...,PARN(EL,EID(AT,AID))
PAR 1,...,PARN(EL(,AID))
PAR 1,...,PARN(EL,EID(,AID))
PAR 1,...,PARN(,EID(,AID))
```

PARN can be the first to the twentieth parameter name.

A simple example using the features discussed thus far is given below. This example is in a form as it would be punched on IBM cards and used as data for the System Simulator Program.

> (Declarations and statements may start in any column. Only columns one to seventy-two may be used. Blank columns are ignored; therefore, spacing is unimportant. If more than one declaration and/or statement are on the same card, they must be separated by periods (.). If the declaration or statement must be continued on the succeeding card, a dollar sign ($) must be placed in column 1 of the continuation card. There is no limit to the number of continuation cards that may be used.)

We will assume that the following element descriptions are available: TANK, MOTOR, PIPE, and PUMP. TANK and MOTOR have a single attachment named 1. PIPE is a binary element with attachment names 1 and 2. PUMP has three attachments INLET, OUTLET, and SHAFT. The parameter names are PRES, TEMP, FLOW, RPM, and TORQUE. The physical setup of the system is shown in the diagram.



```
CONNECTIONS
  MOTOR(1,A)TO,PUMP,LOW(SHAFT)
  MOTOR(1,B)TO,PUMP,HIGH(SHAFT)
  TANK,SOURCE,TO,PUMP,LOW(INLET)
  PUMP,LOW(OUTLET)TO,PIPE,TO,PUMP,HIGH(INLET)
  PUMP,HIGH(OUTLET)TO,TANK,SINK
INPUT PARAMETERS
  PRES,TEMP(TANK)
  RPM(MOTOR)
DESIRED RESULTS
  FLOW(TANK,SINK)
  PRES(PUMP,LOW(OUTLET))
  PRES(PUMP,HIGH(INLET))
  TORQUE((SHAFT))
```

4. SYNONYM STATEMENTS

A special type of statement which can be very useful has been provided in The Simulator. These statements allow the user to declare several names to be synonymous. One possible application of synonyms is the ability to use one short name in place of several long names. For example: the connection statements,

```
CONNECTIONS
  TURBIN,STAGE3(EXTR)TO,...
  TURBIN,STAGE3(OUT,1)TO,...
  TURBIN,STAGE3(OUT,2)TO,...
```

could be replaced by the following connection statements plus a synonym statement,

```
CONNECTIONS
  T3(EXTR)TO,...
  T3(OUT,1)TO,...
  T3(OUT,2)TO,...
SYNONYM
  (TURBIN,STAGE3) = (T3)
```

thus saving considerable writing.

Notice that the true name or names are written first, followed by any names which are to be synonymous to the true name. By true name we mean the name that would have been used if synonyms were not available.

Synonyms may also be used to write out the connection, input parameter, and desired result statements when we do not know the exact element, attachment, and parameter names. Then, just prior to giving these statements to The Simulator, we supply a few synonym statements which specify the true names as they appear in the available element descriptions.

Great flexibility has been provided in the use of synonyms. Since only a few applications have been mentioned here, a detailed description of what the synonym statement does is given below so that the user may find his own applications.

RULE I—all synonym statements apply to all other types of statements except to those in element descriptions, i.e., connection, imput parameter, desired result, equivalence, and substitution statements may be modified by the use of synonym statements.

RULE II—a true name (TN) or its synonym (SN) can refer to any of the following designations: parameter name, element name, element identifier, attachment name, attachment identifier.

In a synonym statement the designations are in the same positions used for input parameters and desired results.

Let TN and SN be any of the following forms

```
PAR
PAR(EL)
PAR(,EID)
PAR(EL,EID)
PAR(EL(AT))
PAR(EL(AT,AID))
PAR(EL,EID(AT))
PAR(EL,EID(AT,AID))
(EL)
(,EID)
(EL,EID)
((AT))
((,AID))
((AT,AID))
```

then allowable statements following the declaration SYNONYMS, are of the form

```
TN = SN
TN = SN1 = SN2 = SN3 =...=SNN
```

Here TN represents a true name and SN the synonymous name which will be replaced by the true name. The TN and SN of a given statement may have different forms, i.e.,

```
PAR1(EL,EID) = PAR2
(,AID) = PAR(EL(AT) = PAR(EL2)
```

are allowable forms for synonym statements.

RULE III—a true name will replace a portion of a statement in some other declaration under the following conditions:

SN is equal to a statement in every designation or the designation in SN has not been given. For example, if no element identifier is given in the SN part of a synonym statement, this synonym statement could apply to all statements in all other declarations even if they had element identifiers. It should be noted that any SN for which a parameter name is given can not apply to any connection statement since the form of the connection statement does not contain a parameter designation.

RULE IV—Only the designations of TN which are given will be used to replace the corresponding designations in a statement which matches an SN. In other words, if the SN part of a synonym statement matches a connection statement, for instance, as prescribed in Rule III, then the connection statement will be modified. The modification is governed by the structure of the TN part of the synonym statement. Wherever a name has been specified in a designation of TN, this name will replace the corresponding designation of the connection statement.

There are three cases which do not exactly fit the above description. First, in the connection statement no parameter is given; we therfore ignore any parameter name given in TN. Second, there is no parameter, attachment name, or attachment identifier given in substitution statements; therefore, these are ignored if they are present in TN. Third, the true name of the substitution and equivalence statements may not be modified by synonyms.

Due to the flexibility of the synonym statement, the preceding rules are difficult to comprehend. Therefore, a number of specific examples will be given. In each example the first group of statements will specify the same data as the second group, the first group containing synonyms, the second with no synonym statements.

SYNONYM EXAMPLE NO. 1

```
Part A

  CONNECTIONS
    PA(I2)TO,PM(I2)
    PA(INLET,B1)TO,PIPE,TO,RIVER
  DESIRED RESULTS
    P,T,F(PA(I2))
    P,T,F(PM)
  SYNONYMS
    (PUMP,MAIN) = (PM)
    (PUMP,AUX1) = (PA)
    ((INLET,B2)) = ((I2))


Part B

  CONNECTIONS
    PUMP,AUX1(INLET,B2)TO,PUMP,MAIN(INLET,B2)
    PUMP,AUX1(INLET,B1)TO,PIPE,TO RIVER
  DESIRED RESULTS
    P,T,F(PUMP,AUX1(INLET,B2))
    P,T,F(PUMP,MAIN)
```

Note that part A has more statements, but they are shorter. It is usually easier to detect errors in shorter statements and more convenient to correct errors in one synonym statement rather than many other statements.

SYNONYM EXAMPLE NO. 2

Part A

```
CONNECTIONS
  S(POS)TO,RES,TO,S(NEG)
  S(1)TO,CAP,TO,S(2)
SYNONYM
  (SOURCE,A(1)) = ((POS))
  (SOURCE) = S
  (SOURCE,A(2)) = ((NEG))
```

Part B

```
CONNECTION
  SOURCE,A(1)TO,RES,TO,SOURCE,A(2)
  SOURCE(1)TO,CAP,TO,SOURCE(2)
```

Here an attachment was used to change the element name, add an element identifier, and change the attachment name.

SYNONYM EXAMPLE NO. 3

Part A

```
INPUT PARAMETER
  P,T(TURBIN,1)
  F(TURBIN,2)
  P,T(TURBIN,3)
SYNONYMS
  (TRB) = (,1)
  (TRB) = (,2)
```

Part B

```
INPUT PARAMETERS
  P,T(TRB,1)
  F(TRB,2)
  P,T(TURBIN,3)
```

Here the element identifier is used to change the element name. This type of change might be necessary if a slightly different system was to be studied, and several stages of the turbine were to be described by a new element description.

5. CALCULATION STATEMENTS

The ability to insert additional MAD statements in the Simulator-generated program in provided through the use of the CALCULATION declaration. Any statement allowed in the MAD language may be used as calculation statement. This includes READ, PRINT, DIMENSION, and INTERNAL FUNCTION type MAD statements.

It is often desirable to supplement the capability of element descriptions through use of calculation. For example, if one wanted to compute the total power loss of a system, this would involve the power loss from many elements. It would be very difficult to include this computation within an element description, but this computation is easily implemented with the CALCULATION declaration.

Another potential use of this declaration is to set up special READ statements for which it is more convenient to prepare data. Or, one may desire more elegant output formats which can easily be inserted as CALCULATIONS.

The details of using calculation-type data are as follows:

(1) Calculation statements are in the standard MAD format.

(2) The last statement which is to be interpreted as a calculation statement must have a plus sign (+) in column one. The plus sign is removed before the simulator punches this calculation statement.

(3) Calculations come in two parts. These are separated by one calculation statement which has a minus sign (-) in column one.
The first set of calculation statements, up to and including the statement with the minus in column one, are inserted in The Simulator-generated program upon encountering the first minus of the prolog. The second set of calculation statements are inserted when the second minus is encountered in the prolog.

SUBSTITUTION-type statements apply to the calculation statements where the ō symbols are used.

6. EQUIVALENCE STATEMENTS

It is often desirable to know the exact name that will be assigned to a parameter at a specific point in the system. For instance, if all input parameters and desired results were made equivalent to known names, the simplified input—output of the MAD language could be used. In this way data could be prepared for the execution run at the same time the data were being prepared for The Simulator. This would minimize the chance of making errors in preparing data and interpreting results. Another use of the equivalence statement is for additional communication between elements. This is very convenient when one element computes a value needed by many other elements.

These statements are prepared for The Simulator as follows:

Give the declaration

```
EQUIVALENCE
```

followed by statements of the form

```
NAME = PAR(CONN),
```

where NAME is a valid MAD variable, PAR is a parameter name, and CONN is a connection point as described in Section 2. CONN must appear as a part of one of the connection statements in the set of data where the equivalence statement is used.

A specific example might be:

```
MAINFL = FLOW(TURBIN,1(STMIN))
```

7. FUNCTION-SUBSTITUTION STATEMENTS

It is usually more economical to set up a general procedure that can be easily adapted to a special case rather than treat each case individually. This is possible with three types of data: calculations, the prolog, and element descriptions. The statements in these types of data may contain variables, constants, and portions of variables or constants, which are set off by a pair of ō symbols. This means that the characters between the ō symbols may be modified by function-substitution statements. If no statement is available which causes modification, then the ō symbols are removed and the characters are left unchanged.

Since function, or subroutine, names are most often modified, the declaration

```
FUNCTION SUBSTITUTION
```

is usually used. But since the technique applies to many other modifications the declaration

```
SUBSTITUTION
```

may also be used.

The statements which follow either of these declarations are of the form:

```
NEW,OLD(EL,EID)
NEW,OLD(EL)
NEW,OLD
```

where NEW represents the new characters that are to replace the characters between the ō which are represented by OLD. The above statements which refer to elements (EL) and/or element identifiers (EID) apply only to element descriptions, while the

```
NEW,OLD
```

statement refers to calculations, the prolog, and element descriptions.

An example of how the procedure works follows.

Given

```
          SUBSTITUTION
            2,1(TURBIN,2)
            3,1(TURBIN,3)
            COUNT,NO
            GRAPH,PLOT
```

and

```
          LOSS(ō1ō) = 1.-EFFō1ō.(TOTAL)
```

at turbine 1,2, and 3, and

```
          EXECUTEōPLOTō.(ōNOō,ōVARō,ōTYPEō)
```

The result would be:

```
          LOSS(1) = 1.-EFF1.(TOTAL)
          LOSS(2) = 1.-EFF2.(TOTAL)
          LOSS(3) = 1.-EFF3.(TOTAL)
          EXECUTE GRAPH.(COUNT,VAR,TYPE).
```

Since statement labels may appear in calculations, prolog, or element descriptions, a special combination of symbols can be used to prevent duplication of the same label on two or more statements. By writing ō** in front of the label and ō after the label, the simulator will prevent duplication. This is a form of substitution, but the computer handles all phases of the modification.

8. NEW ELEMENT TAPE

The over-all form of the program which is generated by The Simulator is shown in Fig. A-7. The majority of the work done by The Simulator is to produce the section of cards called the generated program in Fig. A-7. This is the procedure for computing the values of the desired results from values of the input parameters. Most of this section of cards came from the logical organization of the information in element descriptions. To get a complete program which can be executed to obtain numerical results more than just the generated program section is needed.

There are two methods for making the additional information available to The Simulator. The first, special calculation, has been described in Section 5. The second, the prolog, is described later in this section.

Since an element tape must be available to The Simulator there must be some way of originating such a tape. This is done by giving the delcaration.

NEW ELEMENT TAPE.

This declaration is then followed by the prolog that will be used until another element tape is written. The prolog consists of valid MAD statements, with a few special indicators for The Simulator.

The indicators in the prolog are used to specify where the cards from the two parts of the calculations, and the cards from the generated program are to be inserted in the prolog. This allows the prolog to have control over each section of the program by using the appropriate MAD statements.

The first type of indicator is a plus (+) or minus (-) sign in column 1. The first minus sign in column 1 is placed on the card which is to precede directly the first part of the special calculations. (See Section 5 of this Appendix.) The minus sign is removed before the card is punched so that the card can be a valid MAD statement. Following the last card of the first part of the special calculations is the next card of the prolog. Similarly, the card with the second minus sign is followed by the second part of the special calculations. The card with the third minus sign is followed by the cards of the generated program.and finally the last card of the prolog, which is the last card of the total program produced by The Simulator, has the plus sign in column 1. The 3 minus signs and the plus sign must appear in every prolog.

For example, a simple prolog to set up the stimulator output as a subrouting could appear as:

```
NEW ELEMENT TAPE
$       COMPILE MAD, PRINT OBJECT, PUNCH OBJECT
        EXTERNAL FUNCTION (A,B,C,D)
-       ENTRY TO HB.
        TRANSFER TO ALG
-CALC2  CONTINUE
        FUNCTION RETURN
-ALG    CONTINUE
        TRANSFER TO CALC2
+       END OF FUNCTION
```

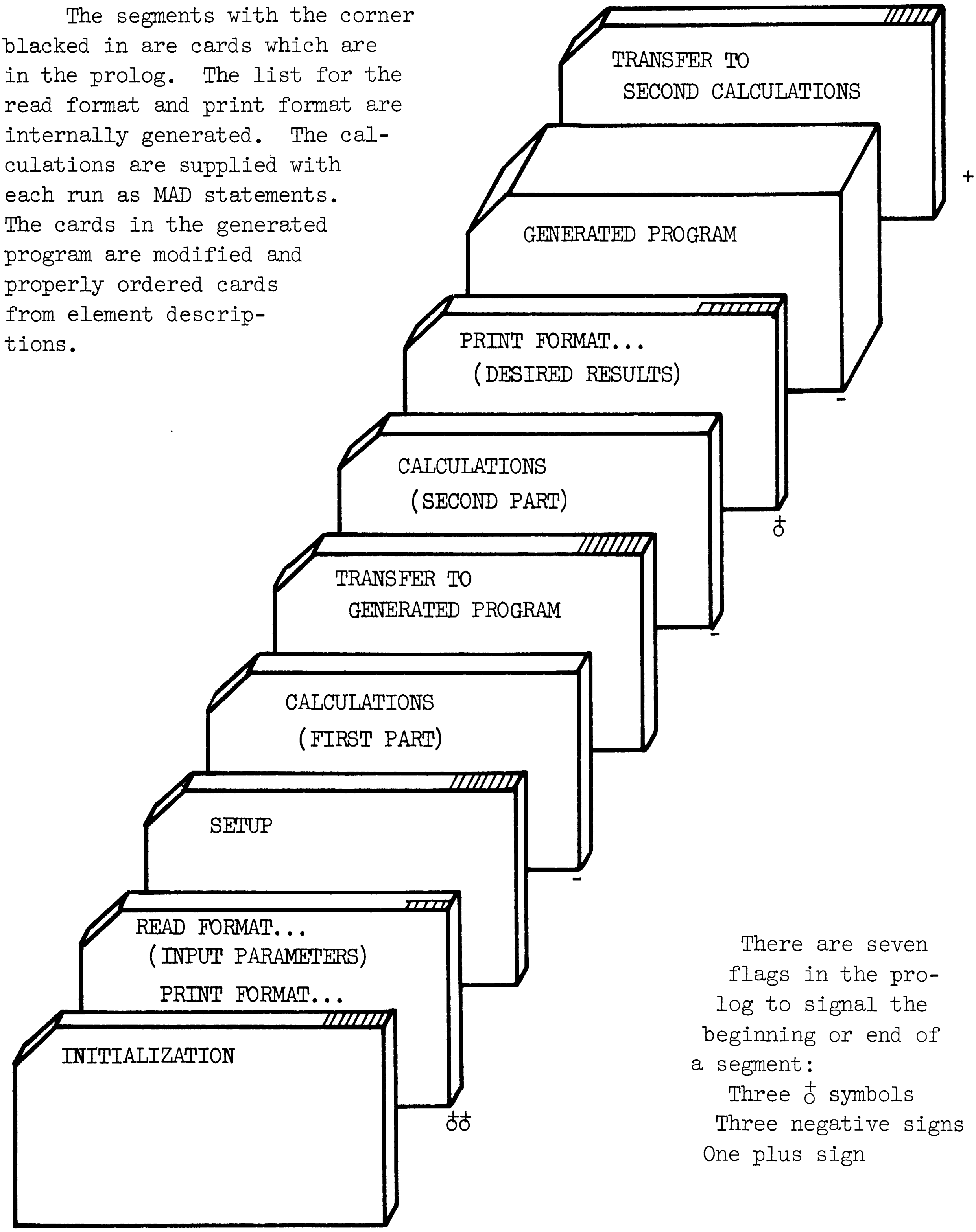
The arguments A, B, C, and D are defined equivalent to their respective input parameters and desired results through the use of equivalence statements, as described in Section 6 of this Appendix.

The symbol ♁ is another indicator in the prolog. The first occurrence of this symbol causes the list of MAD names which correspond to the input parameters to be punched on cards and inserted at the place where the ♁ is found. The ♁ is removed. This list is the form required by the MAD READ FORMAT or PRINT FORMAT statements. The second occurrence of the ♁ causes the same list to be punched again. This policy was adopted since it is desirable to print out data immediately after they are read in. The third and final ♁ causes a list of the desired results to be punched so that the results may be printed through the use of a PRINT FORMAT MAD statement. The ♁ indicator does not have to appear in a prolog if no list is needed.

The substitution ability defined in Section 7 of this Appendix may be used in the prolog.*

*For more examples of prologs see Westervelt, Franklin H., Automatic System Simulation Programming, The University of Michigan, 1960.

The segments with the corner blacked in are cards which are in the prolog. The list for the read format and print format are internally generated. The calculations are supplied with each run as MAD statements. The cards in the generated program are modified and properly ordered cards from element descriptions.


TRANSFER TO SECOND CALCULATIONS
+
GENERATED PROGRAM
-
PRINT FORMAT... (DESIRED RESULTS)
$\overset{+}{o}$
CALCULATIONS (SECOND PART)
-
TRANSFER TO GENERATED PROGRAM
CALCULATIONS (FIRST PART)
-
SETUP
READ FORMAT... (INPUT PARAMETERS) PRINT FORMAT...
$\overset{+}{o}\overset{+}{o}$
INITIALIZATION

There are seven flags in the prolog to signal the beginning or end of a segment:
Three $\overset{+}{o}$ symbols
Three negative signs
One plus sign

Typical deck make up of the program that has been produced by the simulator.

Fig. A-7.

9. ELEMENT DESCRIPTION

The purpose of an element description is to supply a computational procedure which defines the operation and/or physical characteristics of an element to The Simulator. If the operation of an element can be described completely by a number of simple computations, it is relatively easy to write the element description. Consider, for example, a synchronous electric motor. The parameters of interest in defining the operation of the motor are torque, horsepower, and efficiency. The characteristics for the motor, as given in catalogs, might be an efficiency vs. torque characteristic. This can be converted into a formula which can be used on a computer to compute the efficiency for any specific numerical value of torque. The other relations are torque = 26.4*horsepower or horsepower = .038*torque (assuming a 1200 rpm motor).

The actual preparation of an element description requires some specific declarations. These declarations are:

```
ELEMENT DESCRIPTION
NAME OF ELEMENT = .....
BROAD-SCOPE PARAMETERS = ...,...,....
ATTACHMENT NAMES = ...,...,...,....
PERMANENT
STATEMENT COLLECTION
DESCRIPTION FINISHED
ANOTHER ELEMENT FOLLOWS
```

The first declaration of every element description must be ELEMENT DESCRIPTION and the last declaration must be either DESCRIPTION FINISHED or ANOTHER ELEMENT FOLLOWS. Every element must have a name, and this name must be six or less alphanumeric characters. (The one exception is the special element name UNIVERSAL which will be described later.) In the preceding example the element name might be MOTOR and the statement for The Simulator would read:

```
NAME OF ELEMENT = MOTOR
```

Since elements are either of permanent or temporary status, a selection must be made. If the declaration PERMANENT appears in an element description (before the first STATEMENT COLLECTION and ATTACHMENT NAMES declarations), this element will go on the permanent element description library tape, thus making this element description available until the element tape is destroyed or modified. If no PERMANENT declaration appears, the element is assumed to be of temporary status, i.e., this element description will be available for the current set of data being processed but will effectively be erased before any other set of data (new connection statements, Imput Parameters, etc.) is processed.

Above we defined the general structure of an element as a component which connects to other components at its attachment points. In an element description the names of these attachments must be given with the declaration:

ATTACHMENT NAMES = ...,....,....

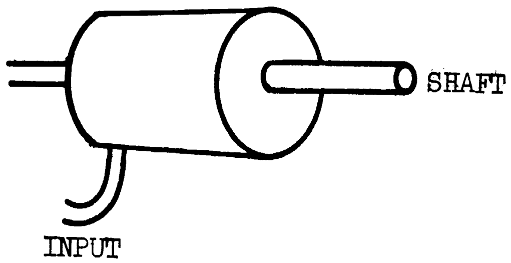
Again using the previous example, this declaration might read

ATTACHMENT NAMES = SHAFT, INPUT (Input refers to the point where horsepower is measured.)

At each attachment there may be more than one connection. Multiple connections to the same attachment are generated by specifying attachment identifiers as well as the attachment. We define a parameter whose value is not dependent on the number of identifiers or on a specific identifier as a broad-scope parameter. None of the parameters in the example fit into this class. Typical broad-scope parameters are voltage, or temperature, or pressure. The other classes are narrow-scope parameters like current, flow, torque, etc. Since most parameters are narrow-scope no special declaration is needed for these. If a parameter is declared to be broad-scope in any element, it must be declared to be broad-scope in every element where it occurs.

The statement collections are the main body of the element description. Each statement collection consists of a computational procedure (usually in the MAD language) which describes some operation and/or physical characteristic of the element.

Consider the following picture to go with our example:



A typical computational procedure might be written:

♁HP(INPUT)♁ = .038*♁TORQUE(SHAFT)♁,
shere the symbols ♁--♁ act as brackets to set off parameters and attachment names. (The information between the ♁--♁ is automatically condensed to 6 or less alphanumeric characters to form an allowable MAD variable name.)

Along with this computational procedure, there must be a logical statement pertaining to the information required and the information yielded. In this case,

```
TORQUE(SHAFT),THEN,HP(INPUT)
```

would be the appropriate capability statement. On other words, this statement says that if the torque at the shaft is known then the horsepower at the input can be computed. The actual form is given below:

```
STATEMENT COLLECTION
TORQUE(SHAFT),THEN,HP(INPUT)
♁HP(INPUT)♁ = .038*TORQUE(SHAFT)♁
```

There can be only one capability statement per STATEMENT COLLECTION declaration, but there can be as much computational procedure as necessary for the indicated capability. There can be as many statement collections as necessary to describe completely the element.

For example, the remaining statement collections might appear as follows:

```
STATEMENT COLLECTION
  HP(INPUT)THEN,TORQUE(SHAFT)
  ♁TORQUE(SHAFT)♁ = ♁HP(INPUT)♁
STATEMENT COLLECTION
  TORQUE(SHAFT)THEN,EFF(INPUT)
  ♁EFF(INPUT)♁ = MTREFF.(♁TORQUE(SHAFT)♁)
```

This completely describes the motor since, if horsepower or torque is given, everything else can be calculated. (Note that MTREFF is a function or subroutine which could be produced by The Stepwise Regression Program from the data given in an efficiency-torque characteristic curve.)

Since there is usually more than one parameter at each attachment, a shorthand notation for this is provided. The parameters associated with the same attachment are listed (separated by commas), followed by the attachment name in parentheses.

For example a capability statement might appear as follows:

```
PRES,TEMP,FLOW(INLET),FLOW(LEAK),THEN,PRES,FLOW(OUTLET)
```

This statement says that if the values of the parameters PRES, TEMP, and FLOW are known at the attachment INLET and if the value of the parameter FLOW is known at the attachment LEAK then there exists a computational procedure which can calculate the value of PRES and FLOW at the attachment OUTLET.

There is no limit to the number of parameters or the number of attachments which can be used to describe a capability. There are four over-all limits which are necessary due to storage limitations. The number of distinct attachments may not exceed 20 in any given element, and the total number of distinct parameters in all elements may not exceed 70. The total number of elements in the library, temporary plus permanent, may not exceed 94 and the total number of elements referred to by a single run may not exceed 59.

There is one special element name, UNIVERSAL, which has the following feature. The statement collections in this element can be used anywhere in the system. Thus general, or universal, relationships defined in the UNIVERSAL element may be omitted from many elements each of which would normally require a copy of the relationships. Other than this feature the UNIVERSAL element description is written just as any other element description.

It was mentioned that an attachment may have many other attachments connected to it, each with a unique identifier. The method of preparing capability and calculational procedure for the statement collection is as follows.

(1) To compute the total flow into a junction use:

```
STATEMENT COLLECTION
  FLOW(INLET)THEN,FLOW(OUTLET)
    T=0.
    T=T+♁FLOW(INLET,I)♁
    ♁FLOW(OUTLET)♁=T
```

(Note that the I in the attachment identifier position indicates this statement is to be repeated, substituting the specific name of the parameter flow at every identifier.)

(2) To compute the total flow into a junction and the pressure at the inlet to the junction use:

```
STATEMENT COLLECTION
  FLOW(INLET),PRES(OUTLET)THEN,FLOW(OUTLET)PRES(INLET)
    T=0.
    T=T+♁FLOW(INLET)♁
    T=T+♁FLOW(INLET,I)♁
    ♁FLOW(OUTLET)♁=T
    ♁PRES(INLET)♁=♁PRES(OUTLET)♁-.02*T
```

Note that an additional statement

```
T=T+♁FLOW(INLET)♁
```

is required to get all possible identifiers because the attachment INLET appears to the right of the THEN.

(3) To computethe flow at one attachment use:

```
STATEMENT COLLECTION
  FLOW(INLET),FLOW(OUTLET)THEN,FLOW(INLET,I)
    T=♁FLOW(OUTLET)♁
    T=T-♁FLOW(INLET,I)♁
    ♁FLOW(INLET)♁=T
```

Again note that FLOW(INLET) applies to the point where a result is sought while FLOW(INLET,I) applies everywhere else. The (INLET,I) in the capability statement puts the restriction on this collection that it be used only once. The need for this is evident because, if the flow is not given at two points, repeated use of this collection results in a never-terminating cycle.

There are occasions when a statement collection may apply to a modification of a basic element, for instance, an element with an attachment (AT1) removed. In this case, the capability statement must contain the statement

```
...,WITHOUT AT1,...
```

It is also useful to have a capability that requires no input, i.e., nothing to the left of the THAT. In this case, the word COMPUTE replaces the ..,THEN, in the capability statement.

To be able to handle completely general algorithms an iterative procedure must be available. This is accomplished by using ESTIMATE to replace the ...,THEN, in the capability statement. This means that an estimate of the values of the parameters listed after the ESTIMATE is available, but there must be some other algorithm found which can be used to improve the estimate and determine when the iteration should terminate. A typical example is available in Section 11 in the element RES.

10. MACRO DECLARATIONS

The situation sometimes arise where a minor change is to be made in a system for the purpose of comparison. To facilitate a change of this type several macro declarations have been provided.

The declaration which must appear just after the basic system description (this includes connection statements, input parameters, element descriptions, etc.) is written as:

CONFIGURATION CHANGE NO. X

where X is any alphanumeric word of six characters or less. Essentially, this statement begins a completely new set of data for The Simulator, except that some of the data has already been read in and partially processed. This statement will also be used as a heading for the appropriate output. Following this declaration, there must be one of the following declarations.

DELETION(S)
ADDITION(S)
ELEMENT TO BE DELETED
ELEMENT DESCRIPTION
CALCULATIONS

The (S), indicating the plural, may be used or omitted. The Element description declaration and the form of an element description have been described earlier.

The declaration

ELEMENT TO BE DELETED, Y

where Y is the name of an element in the permanent or temporary library, is used to remove the element Y. Another element description with the name Y may then be added to the library.

The use of the Calculations declaration causes the previous calculation statements to be ignored and the new statements to be used. The Calculation declaration was discussed in detail above.

The remaining Addition and Deletion declarations apply to the following sections: Connection, Input Parameter, Desired Result, Synonym, Substitution, and Equivalence. The word Addition implies that the statements which follow are to be added to the basic system data. The word Deletion implies that the statements which follow are part of the basic system data and should be removed.

The result of the changes is a new set of data for The Simulator. To clarify the discussion in this section an example follows.

Example of a portion of a basic configuration, configuration change statements, and the new configuration which results.

```
BASIC CONFIGURATION
  CONNECTIONS
    RIVER(1,A)TO,PUMP,A(INLET)
    PUMP,A(OUTLET)TO,COND(INLET,1)
    MOTOR,P1(SHAFT)TO,PUMP,A(SHAFT)
  INPUT PARAMETERS
    EFF(PUMP,A(SHAFT))
    TEMP(RIVER)
  DESIRED RESULTS
    POWER(MOTOR(SHAFT))

CONFIGURATION CHANGE NO. 2 PUMPS
  DELETIONS
    CONNECTIONS
      MOTOR,P1(SHAFT)TO,PUMP,A(SHAFT)
  ADDITIONS
    CONNECTIONS
      MOTOR,P1(SHAFT)TO,PUMP,A(SHAFT)
  ADDITIONS
    CONNECTIONS
      RIVER(1,B)TO,PUMP,B(INLET)
      PUMP,B(OUTLET)TO,COND(INLET,2)
      MOTOR,P1(SHAFT,1)TO,PUMP,A(SHAFT)
      MOTOR,P1(SHAFT,2)TO,PUMP,B(SHAFT)
    INPUT PARAMETERS
      EFF(PUMP,B(SHAFT))
```

The new configuration which results from changing the basic configuration is not actually generated as shown below, but the effect is the same as if these statements were given as a second basic configuration.

```
CONNECTIONS
  RIVER(1,A)TO,PUMP,A(INLET)
  PUMP,A(OUTLET)TO,COND(INLET,1)
  RIVER(1,B)TO,PUMP,B(INLET)
  PUMP,B(OUTLET)TO,COND(INLET,2)
  MOTOR,P1(SHAFT,1)TO,PUMP,A(SHAFT)
  MOTOR,P1(SHAFT,2)TO,PUMP,B(SHAFT)
```

```
INPUT PARAMETERS
  EFF(PUMP,A(SHAFT))
  TEMP(RIVER)
  EFF(PUMP,B(SHAFT))
DESIRED RESULTS
  POWER(MOTOR(SHAFT))
```

14. CONTROL INFORMATION

When preparing data it is often desirable to insert comments so that future reference will be easy. For this purpose a slash or divide symbol (/) may be used in column one of a data card. With a slash in column one, the card image is printed along with the rest of the input data, but the information on the card is not interpreted.

There is one control card which can be used to modify the processing of data. This card has an asterisk (*) in column one. Columns 7 through 14 may contain identification which will be punched on all cards produced by The System Simulator Program. The cards will be consecutively numbered, providing the last several characters of the identification are numeric. The first card produced by The System Simulator Program has the information, given in columns 7 through 14 on the control card, punched in columns 73 through 80. One is added to column 80 for each succeeding card. Carries are propagated as needed until a non-numeric character is encountered.

A one (1) in columns 19 through 21 can be used to suppress various processing. The column and corresponding process suppressed are given in the table which follows.

Column	
19	Punching input-output statements for input parameters and desired results
20	Punching prolog and epilog
21	Generating algorithm

A one in columns 25 through 28 can be used to get extra output, as listed in the table below.

25	Print element tape
26	Print collections even if no algorithm has been found
27	Punch collections even if no algorithm has been found
28	Print table of collections used

Many of the above items will be better understood after rereading the section on element descriptions.

12. COMPLETE EXAMPLE

A complete example is presented to pull together the information given in preceding sections. This example uses most of the features that are available.

Except for the headings and page numbers the following are the data exactly as they are punched on cards.

The example shows the data that were given to The System Simulator Program in order to get a program that could simulate the operation of a feedback amplifier on a computer.

The circuit diagram for the feedback amplifier is given below. The data given to The System Simulator follow, with some subroutines needed by the simulator-generated program. Finally, two of these subroutines, which define the operation of vacuum tubes, were produced by the Stepwise Regression Program.

The declaration CHECK RUN is for use while checking out the simulator program. This should be used only on small sets of data since a very large amount of extra printing is initiated by this declaration.

The declaration NEXT SET OF DATA is used at the end of a data set if another data set follows.

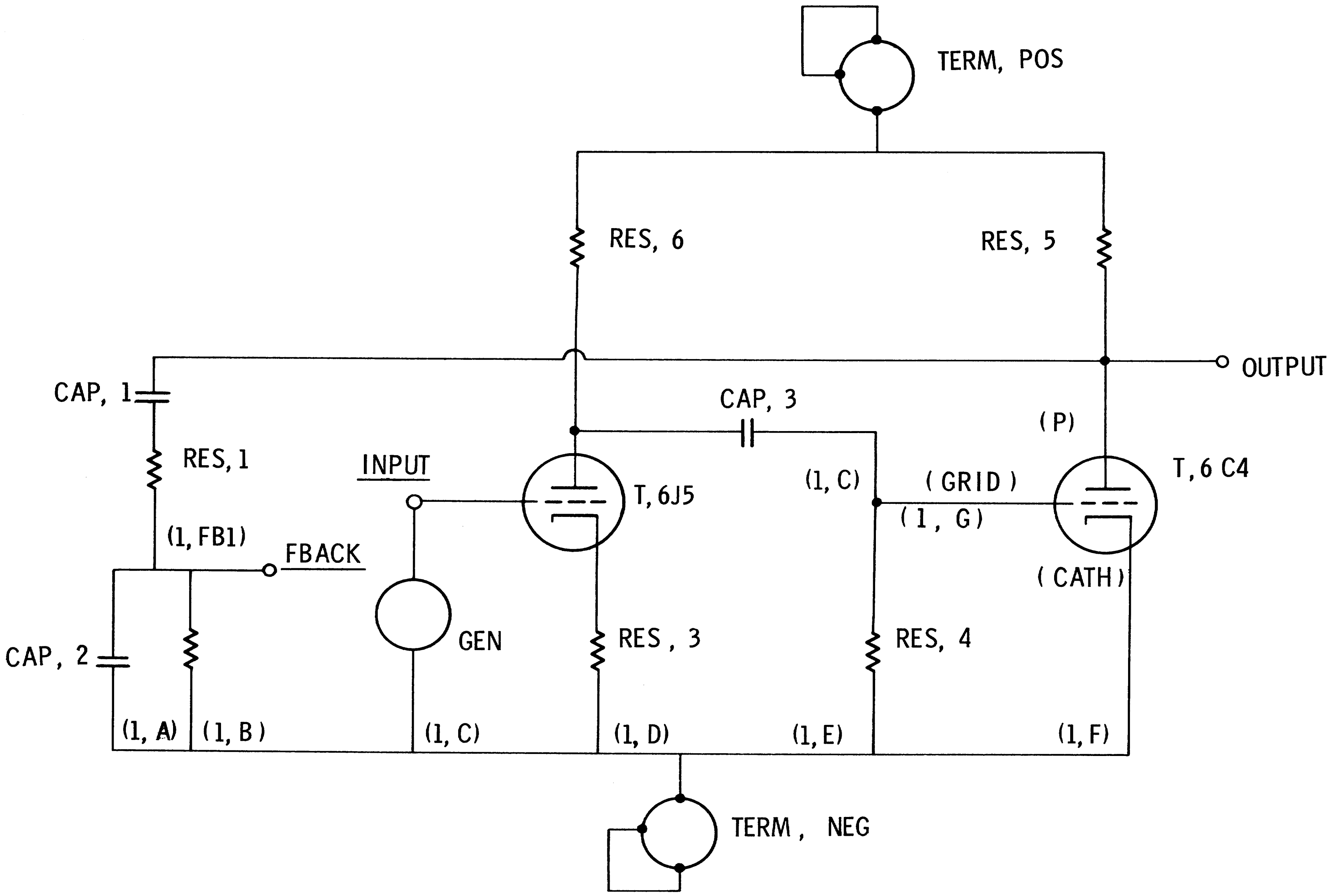

TERM, POS
RES, 6
RES, 5
OUTPUT
CAP, 1
CAP, 3
(P)
T, 6 C4
RES, 1
INPUT
T, 6J5
(1, C)
(GRID)
(1, G)
(1, FB1)
FBACK
(CATH)
GEN
CAP, 2
RES, 3
RES, 4
(1, A)
(1, B)
(1, C)
(1, D)
(1, E)
(1, F)
TERM, NEG

Fig. A-8.

XII COMPLETE EXAMPLE

DATA AS PREPARED FOR INPUT TO THE SYSTEM SIMULATOR PROGRAM

THE DIAGRAM FOR THIS DATA IS ON THE PRECEEDING PAGE

```
$ DATA

 CONNECTIONS
          TN(1,A) TO, CAP,2, TO, RES,1(1,FB1)
          TN(1,B) TO, RES,2, TO, RES,1(1,FB2)
          TN(1,C) TO, GEN , TO, T,6J5(GRID)
          TN(1,D) TO, RES,3, TO,T,6J5(CATH)
          TN(1,E) TO, RES,4(2)
          TN(1,F) TO, T,6C4(CATH)
       RES,1(2) TO, CAP,1, TO, RES,5(1,OUTPUT)
     RES,6(1,C) TO, CAP,3, TO,RES,4(1,C)
       RES,4(1,G) TO, T,6C4(GRID)
                    RES,6(2) TO, TERM,POS(1,T1)
       T,6J5(P) TO, RES,6(1,P)
       RES,5(2)    TO, TERM,POS(1,T2)
       RES,5(1,P) TO, T,6C4(P)


 EQUIVALENCE
    FBACK = V(RES,1(1,FB2))
    INPUT = V(GEN(2))
    OUTPUT =V(RES,5(1,OUTPUT))


 DESIRED RESULTS
   V(RES,1(1,FB1))
   V(RES,5(1,OUTPUT))
       V(T,6J5(GRID))
   I(TERM,POS(OUTPUT))
       I(T,6J5)


 SYNONYMS
   (TERM,NEG) = (TN)
   (TRIODE) = (T)
   ((PLATE)) = ((P))


 FUNCTION SUBSTITUTIONS
        6J5,TUBE(TRIODE,6J5)
        6C4,TUBE(TRIODE,6C4)



  CONNECTIONS
      TN(INPUT) TO, TN(OUTPUT)
      TERM,POS(INPUT) TO, TERM,POS(OUTPUT)
 INPUT PARAMETERS
    V(TN(INPUT))
    V(TERM,POS(INPUT))
OHM (RES(2))
FARAD (CAP(2))


*     TEST2000


 NEW ELEMENT TAPE.
*          COMPILE MAD,PRINT OBJECT,PUNCH OBJECT
 START     READ FORMAT FORM,F(1)...F(12)
           F(13)=$*$
           F(0)=$1H ,$
           READ FORAMT F(1),⊕
-          PRINT FORMAT F,⊕
          R FIRST PART OF CALCULATIONS END HERE
           TRYCNT=1
           FIRST=1B
           REPEAT=0B
           TRANSFER TO BEGIN
  BACK     CONTINUE
           WHENEVER TRYCNT.L.10 .AND.REPEAT
           REPEAT=0B
           FIRST=0B
           TRYCNT=TRYCNT+1
           TRANSFER TO BEGIN
           END OF CONDITIONAL
-          WHENEVER REPEAT, PRINT FORMAT FORM,$0NO CONVERGENC IN 10 TRY$
          R SECOND PART OF CALCULATIONS END HERE
           TRANSFER TO START
- BEGIN    CONTINUE
           TRANSFER TO BACK
           INTEGER TRYCNT,F
           BOOLEAN FIRST,REPEAT
           DIMENSION F(13)
           VECTOR VALUES FORM=$12C6*$
+          END OF PROGRAM


 CALCULATIONS.
           THROUGH SETLP,FOR J=0,1,J.G.50
           I(J) = 0.
           IT (J) = 0.
           V(J)=0.
  SETLP    VT(J)=0.
           DT=.628E-4
           TM=.0314
           K=0
-          THROUGH MAINLP,FOR TMSEC=0.,DT,TMSEC.G.TM
           H=K/5
           WHENEVER H.GE.100,H=99
           X(H)=TMSEC
           X(H+100)=TMSEC
           X(H+200)=TMSEC
           Y(H) = FBACK
           Y(H+100)=OUTPUT
           Y(H+300)=INPUT
           K=K+1
           THROUGH RSET,FOR J=0,1,J.G.50
           IT(J) = I(J)
 RSET      VT(J)=V(J)
MAINLP     CONTINUE
           DIMENSION  V(50),VT(50),I(50),IT(50),X(299),Y(299)
           INTEGER J,K,H
           PRINT FORMAT FORM,$1    GRAPH OF VOLTAGES AS A FUNCTION OF TI
          1ME$
           PRINT FORMAT FORM,$0$
           EXECUTE GRAPH.(X,Y,NOPTS,NOPTS(3),3,YHEAD)
           PRINT FORMAT FORM,$0            TIME IN THOUSANDTHS OF A SECOND
          1$
           VECTOR VALUES NOPTS=100,100,100,$F$,$*$,$I$
           VECTOR VALUES YHEAD=$ VOLTAGE,  F=FEEDBACK,  *=OUTPUT,  I=INP
+         1UT            $
```

```
  ELEMENT DESCRIPTION                                                      TR    000
    NAME OF ELEMENT = TRIODE                                               TR    002
     PERMANENT                                                             TR    004
    ATTACHMENT NAMES = GRID,CATH,PLATE                                     TR    006
    BROAD SCOPE PARAMETERS = V                                             TR    008
/                                                                          TR    020
/ THIS ELEMENT REQUIRES TWO FUNCTIONS..  IC.(VKP,VKG) , IB.(VKP,VKG)       TR    021
/                                                                          TR    022
     STATEMENT COLLECTION                                                  TR    110
        V(GRID), V(CATH), V(PLATE),  THEN, I(GRID), I(CATH), I(PLATE)      TR    120
           VKP = V⊕PLATE⊕ - V⊕CATH⊕                                        TR    130
           VKG = V⊕GRID⊕ - V⊕CATH⊕                                         TR    131
           I⊕PLATE⊕ = IBθTUBEθ.(VKP,VKG)                                   TR    132
           I⊕GRID⊕  = ICθTUBEθ.(VKP,VKG)                                   TR    133
           I⊕CATH⊕ = - I⊕PLATE⊕ - I⊕GRID⊕                                  TR    134
           DESCRIPTION FINISHED
```

```
ELEMENT DESCRIPTION                                                        G     000
   NAME OF ELEMENT = GEN                                                   G     002
   PERMANENT                                                               G     004
   ATTACHMENT NAMES = 1,2                                                  G     006
   BROAD SCOPE PARAMETERS = V                                              G     008
     STATEMENT COLLECTION                                                  G     110
       COMPUTE V(2)                                                        G     120
          V⊕2⊕ = GEN.(TMSEC)                                               G     130
     STATEMENT COLLECTION                                                  G     210
       I(2) THEN, I(1)                                                     G     220
          I⊕1⊕ = I⊕2⊕                                                      G     230


          DESCRIPTION FINISHED
```

```
ELEMENT DESCRIPTION.                                                    TERM  00
/                                                                       TERM  01
NAME OF ELEMENT = TERM                                                  TERM  02
/                                                                       TERM  03
BROAD SCOPE PARAMETERS = V                                              TERM  04
/                                                                       TERM  05
PERMANENT                                                               TERM  06
/                                                                       TERM  07
 ATTACHMENT NAMES = 1, INPUT, OUTPUT                                    TERM  08
/                                                                       TERM  10
/ ONLY ATTACHMENT  1  SHOULD BE CONNECTED TO OTHER ELEMENTS             TERM  11
/                                                                       TERM  12
/ USE    TERM(INPUT) TO, TERM(OUTPUT)   IF VOLTAGE IS TO BE GIVEN       TERM  13
/                                       OR CURRENT IS TO BE CALCULATED  TERM  14
/                                                                       TERM  15
/     REQUEST TOTAL CURRENT AT OUTPUT                                   TERM 16
/     SPECIFY VOLTAGE AT INPUT                                          TERM 17
/                                                                       TERM 18
    STATEMENT COLLECTION                                                TERM 110
       V(INPUT) THEN,V(1)                                               TERM 120
           EQUIVALENCE (V0INPUT0,V01,A0)                                TERM 130
           EQUIVALENCE (V0INPUT0,V010)                                  TERM 131
    STATEMENT COLLECTION                                                TERM 210
       I(1) THEN,I(OUTPUT)                                              TERM 220
           IT=0.                                                        TERM 230
           IT=IT + I⊕1,A⊕                                               TERM 231
           I⊕OUTPUT⊕ = IT                                               TERM 232
    STATEMENT COLLECTION                                                TERM 510
        I(1)  WITHOUT (OUTPUT)  THEN,  I(1,A)                           TERM 520
           IT=0.                                                        TERM 530
           IT=IT + I⊕1,A⊕                                               TERM 531
            I⊕1⊕ = -IT                                                  TERM 532
           DESCRIPTION FINISHED
```

```
  ELEMENT DESCRIPTION                                                   DCAP 000
    NAME OF ELEMENT = CAP                                               DCAP 002
    PERMANENT                                                           DCAP 004
    ATTACHMENT NAMES = 1,2                                              DCAP 006
    BROAD SCOPE PARAMETERS = V,FARAD,C                                  DCAP 008
/                                                                       DCAP 020
/ THIS ELEMENT MUST BE USED AS SINGLE ATTACHMENT CONNECTIONS            DCAP 021
/                                                                       DCAP 022
    STATEMENT COLLECTION                                                DCAP 110
      FARAD(2) THEN, C(1)                                               DCAP 120
    STATEMENT COLLECTION                                                DCAP 210
      FARAD(1) THEN, C(1)                                               DCAP 220
           EQUIVALENCE (V⊕1⊕,V⊕1,A⊕)                                    DCAP 230
           EQUIVALENCE (V⊕2⊕,V⊕2,A⊕)                                    DCAP 231
           EQUIVALENCE (⊕FARAD(1)⊕,⊕FARAD(1,A)⊕,C⊕1,A⊕)                 DCAP 232
           EQUIVALENCE (⊕FARAD(1)⊕,⊕FARAD(2,A)⊕,C⊕2,A⊕)                 DCAP 233
           EQUIVALENCE (⊕FARAD(1)⊕,C⊕1⊕)                                DCAP 234
    STATEMENT COLLECTION                                                DCAP 310
      C(1) THEN, C(2)                                                   DCAP 320
           EQUIVALENCE (V⊕1⊕,V⊕1,A⊕)                                    DCAP 330
           EQUIVALENCE (V⊕2⊕,V⊕2,A⊕)                                    DCAP 331
```

```
          EQUIVALENCE (⊕FARAD(1)⊕,⊕FARAD(1,A)⊕,C⊕1,A⊕)               DCAP 332
          EQUIVALENCE (⊕FARAD(1)⊕,⊕FARAD(2,A)⊕,C⊕2,A⊕)               DCAP 333
          EQUIVALENCE (⊕FARAD(1)⊕,C⊕1⊕)                              DCAP 334
   STATEMENT COLLECTION                                               DCAP 410
     C,V,I(1), V(2) THEN, I(2)                                        DCAP 420
          V(⊕1⊕) = .ABS.(V⊕1⊕ - V⊕2⊕)                                DCAP 430
          I ⊕1⊕ = C⊕1⊕ *(VT(⊕1⊕)-V(⊕1⊕))/DT - I⊕2⊕                   DCAP 431
   STATEMENT COLLECTION                                               DCAP 510
     C,V,I(2), V(1) THEN, I(1)                                        DCAP 520
          V(⊕1⊕) = .ABS.(V⊕1⊕ - V⊕2⊕)                                DCAP 530
          I⊕2⊕ = C⊕2 ⊕ *(VT(⊕1⊕)-V(⊕1⊕))/DT - I⊕1⊕                   DCAP 531
   STATEMENT COLLECTION                                               DCAP 610
     ESTIMATE I(1)                                                    DCAP 620
          WHENEVER FIRST                                              DCAP 630
          REPEAT = 1B                                                 DCAP 631
          TI⊕1⊕ = I⊕1⊕                                                DCAP 632
          OTHERWISE                                                   DCAP 633
           I⊕1⊕ = ITTR.( I⊕1⊕, TI⊕1⊕, REPEAT)                         DCAP 634
          END OF CONDITIONAL                                          DCAP 635
   STATEMENT COLLECTION                                               DCAP 710
     ESTIMATE I(2)                                                    DCAP 720
          WHENEVER FIRST                                              DCAP 730
          REPEAT = 1B                                                 DCAP 731
          TI⊕2⊕ = I⊕2⊕                                                DCAP 732
          OTHERWISE                                                   DCAP 733
           I⊕2⊕ = ITTR.( I⊕2⊕, TI⊕2⊕, REPEAT)                         DCAP 734
          END OF CONDITIONAL                                          DCAP 735
   STATEMENT COLLECTION                                               DCAP 810
     C,V,I(1), I(2) THEN, V(2)                                        DCAP 820
          V(⊕2⊕) = VT(⊕2⊕) + DT*(I⊕1⊕+I⊕2⊕)/C⊕1⊕                     DCAP 830
          V⊕1⊕ = V⊕2⊕ + V(⊕2⊕)                                        DCAP 831
   STATEMENT COLLECTION                                               DCAP 910
     C,V,I(2), I(1) THEN, V(1)                                        DCAP 920
          V(⊕2⊕) = VT(⊕2⊕) + DT*(I⊕1⊕+I⊕2⊕)/C⊕2⊕                     DCAP 930
          V⊕2⊕ = V⊕1⊕ - V(⊕2⊕)                                        DCAP 931
          DESCRIPTION FINISHED


ELEMENT DESCRIPTION.                                                  RES  000
NAME OF ELEMENT  =    RES.                                            RES  010
/                                                                     RES  011
/  ** NO RESTRICTIONS ON THE CONNECTION OF THIS ELEMENT.              RES  012
/                                                                     RES  013
 BROAD SCOPE PARAMETERS = V,OHM,R                                     RES  020
PERMANENT.                                                            RES  030
 ATTACHMENT NAMES = 1,2                                               RES  040
/                                                                     RES  100
/                                                                     RES  104
   STATEMENT COLLECTION.                                              RES  110
      R(1) THEN, R(2).                                                RES  120
   STATEMENT COLLECTION.                                              RES  210
        OHM(2) THEN, R(1)                                             RES  220
          EQUIVALENCE (OHM⊕1⊕,R⊕1⊕)                                   RES  230
          EQUIVALENCE (V⊕1⊕,V⊕1,A⊕)                                   RES  231
          EQUIVALENCE (V⊕2⊕,V⊕2,A⊕)                                   RES  232
          EQUIVALENCE (OHM⊕1⊕,OHM⊕1,A⊕,R⊕1,A⊕)                        RES  233
          EQUIVALENCE (OHM⊕1⊕,OHM⊕2,A⊕,R⊕2,A⊕)                        RES  234
```

```
STATEMENT COLLECTION.                                                   RES  310
    I(1)  I(2) THEN,  I(1,A).                                           RES  320
        T=0.                                                            RES  330
        T = T +  I⊕1,A⊕                                                 RES  331
        S=0.                                                            RES  340
        S = S +  I⊕2,A⊕                                                 RES  341
         I⊕1⊕ = -S -T                                                   RES  350
STATEMENT COLLECTION                                                    RES  410
      OHM(1) THEN, R(1)                                                 RES  420
        EQUIVALENCE (OHM⊕1⊕,R⊕1⊕)                                       RES  430
        EQUIVALENCE (V⊕1⊕,V⊕1,A⊕)                                       RES  431
        EQUIVALENCE (V⊕2⊕,V⊕2,A⊕)                                       RES  432
        EQUIVALENCE (OHM⊕1⊕,OHM⊕1,A⊕,R⊕1,A⊕)                            RES  433
        EQUIVALENCE (OHM⊕1⊕,OHM⊕2,A⊕,R⊕2,A⊕)                            RES  434
STATEMENT COLLECTION.                                                   RES  510
    I(1)  I(2) THEN,  I(2,A).                                           RES  520
        T=0.                                                            RES  530
        T = T +  I⊕1,A⊕                                                 RES  531
        S=0.                                                            RES  540
        S = S +  I⊕2,A⊕                                                 RES  541
         I⊕2⊕ = -S -T                                                   RES  550
STATEMENT COLLECTION                                                    RES  610
   ESTIMATE I(1)                                                        RES  620
        WHENEVER FIRST                                                  RES  630
        REPEAT = 1B                                                     RES  631
        TI⊕1⊕ = I⊕1⊕                                                    RES  632
        OTHERWISE                                                       RES  633
         I⊕1⊕ = ITTR.( I⊕1⊕, TI⊕1⊕, REPEAT)                             RES  634
        END OF CONDITIONAL                                              RES  635
STATEMENT COLLECTION.                                                   RES  710
   R, I, V(1)  V(2) THEN,  I(1,A).                                      RES  720
        T=0.                                                            RES  730
        T = T +  I⊕1,A⊕                                                 RES  731
        I⊕1⊕ = ( V⊕1⊕ - V⊕2⊕ ) / R⊕1⊕ -T                                RES  750
STATEMENT COLLECTION                                                    RES  810
   ESTIMATE I(2)                                                        RES  820
        WHENEVER FIRST                                                  RES  830
        REPEAT = 1B                                                     RES  831
        TI⊕2⊕ = I⊕2⊕                                                    RES  832
        OTHERWISE                                                       RES  833
         I⊕2⊕ = ITTR.( I⊕2⊕, TI⊕2⊕, REPEAT)                             RES  834
        END OF CONDITIONAL                                              RES  835
STATEMENT COLLECTION.                                                   RES  910
   R, I, V(2)  V(1) THEN,  I(2,A).                                      RES  920
        T=0.                                                            RES  930
        T = T +  I⊕2,A⊕                                                 RES  931
        I⊕2⊕ = ( V⊕2⊕ - V⊕1⊕ ) / R⊕2⊕ -T                                RES  950
STATEMENT COLLECTION                                                    RES 1010
   ESTIMATE V(1)                                                        RES 1020
        WHENEVER FIRST                                                  RES 1030
        REPEAT = 1B                                                     RES 1031
        TV⊕1⊕ = V⊕1⊕                                                    RES 1032
        OTHERWISE                                                       RES 1033
         V⊕1⊕ = ITTR.( V⊕1⊕, TV⊕1⊕, REPEAT)                             RES 1034
        END OF CONDITIONAL                                              RES 1035
STATEMENT COLLECTION.                                                   RES 1110
   R, I, V(1) THEN,  V(2).                                              RES 1120
        T=0.                                                            RES 1130
        T = T +  I⊕1,A⊕                                                          RES 1131
         V⊕2⊕ =  V⊕1⊕ - T * R⊕1⊕                                                 RES 1150
STATEMENT COLLECTION                                                              RES 1210
   ESTIMATE V(2)                                                                  RES 1220
        WHENEVER FIRST                                                            RES 1230
        REPEAT = 1B                                                               RES 1231
        TV⊕2⊕ = V⊕2⊕                                                              RES 1232
        OTHERWISE                                                                 RES 1233
         V⊕2⊕ = ITTR.( V⊕2⊕, TV⊕2⊕, REPEAT)                                       RES 1234
        END OF CONDITIONAL                                                        RES 1235
STATEMENT COLLECTION.                                                             RES 1310
   R, I, V(2) THEN,  V(1).                                                        RES 1320
        T=0.                                                                      RES 1330
        T = T +  I⊕2,A⊕                                                           RES 1331
         V⊕1⊕ =  V⊕2⊕ - T * R⊕2⊕                                                  RES 1350
STATEMENT COLLECTION.                                                             RES 1510
   R, V(1)  I(2) THEN,  V(2).                                                     RES 1520
        T=0.                                                                      RES 1530
        T = T +  I⊕2,A⊕                                                           RES 1531
         V⊕2⊕ =  V⊕1⊕ + ( T +  I⊕2⊕ ) * R⊕1⊕                                      RES 1550
STATEMENT COLLECTION.                                                             RES 1710
   R, V(2)  I(1) THEN,  V(1).                                                     RES 1720
        T=0.                                                                      RES 1730
        T = T +  I⊕1,A⊕                                                           RES 1731
         V⊕1⊕ =  V⊕2⊕ + ( T +  I⊕1⊕ ) * R⊕2⊕                                      RES 1750
        DESCRIPTION FINISHED
```

SUBROUTINES THAT WILL BE REQUIRED BY THE SIMULATOR GENERATED PROGRAM

```
$ COMPILE MAD                                                   GEN  000
          EXTERNAL FUNCTION (T)
          ENTRY TO GEN.
          FUNCTION RETURN SIN.(1000.*T)
          END OF FUNCTION
```

```
$ COMPILE MAD                                                   ITTR 00
          EXTERNAL FUNCTION (ARG1,ARG2,NC)
          ENTRY TO ITTR.
          WHENEVER .ABS.(ARG1-ARG2).G..01*(.ABS.ARG1+.ABS.ARG2),NC=1.
          ARG2=(ARG1+ARG2)/2.
          FUNCTION RETURN ARG2
          END OF FUNCTION
```

THESE SUBROUTINES WERE PRODUCED BY THE STEPWISE REGRESSION PROGRAM

```
$ COMPILE MAD                                                   IB6J5000
          EXTERNAL FUNCTION (EB,EC)
          INTEGER I
          DIMENSION T(14)
          ENTRY TO IC6J5.
          ENTRY TO IC6C4.
          X1=EB/100.
          X2=EC
          T(1) =  0.56313861E-02
          T( 1) = T( 1) * X 1 .P. ( -2)
          T( 2) =  -0.32877611E-02
          T( 2) = T( 2) * .ABS.X 1 .P. ( -0.500000)
          T( 2) = T( 2) * X 2
          T( 3) =  -0.10737944E 01
          T( 3) = T( 3) * .ABS.X 1 .P. (  0.142857)
          T( 4) =   0.16821247E-02
          T( 4) = T( 4) * X 1 .P. ( -1)
          T( 4) = T( 4) * X 2
          T( 5) =  -0.90962253E-05
          T( 5) = T( 5) * X 1 .P. ( -3)
          T( 5) = T( 5) * X 2 .P. (  3)
          T( 6) =  -0.15554141E-03
          T( 6) = T( 6) * X 1 .P. (  2)
          T( 6) = T( 6) * .ABS.X 2 .P. (  0.500000)
          T( 7) = T( 7) * X 1 .P. ( -1)
          T( 7) =  -0.12864274E-04
          T( 7) = T( 7) * X 2 .P. (  3)
          T( 8) =   0.54098414E 00
          T( 8) = T( 8) * .ABS.X 1 .P. ( -0.166667)
          T( 9) =  -0.66028692E-01
          T( 9) = T( 9) * X 1 .P. ( -1)
          T(10) =   0.40204109E-02
          T(10) = T(10) * .ABS.X 1 .P. (  0.142857)
          T(10) = T(10) * X 2
          T(11) =   0.60386404E 00
```

```
           T(11) = T(11) * .ABS.X 1 .P. (  0.333333)
           T(12) =   0.94997922E-07
           T(12) = T(12) * X 1 .P. (  7)
           T(13) =  -0.12147852E-06
           T(13) = T(13) * X 1 .P. ( -1)
           T(13) = T(13) * X 2 .P. (  4)
           T(14) =  -0.20375654E-04
           T(14) = T(14) * X 1 .P. ( -4)
           T(0) = 0.
           THROUGH SUM, FOR I = 1,1,I .G. 14
       SUM T(0) = T(0) + T(I)
           FUNCTION RETURN T(0)
           END OF FUNCTION


$ COMPILE MAD                                                   IC6J500
           EXTERNAL FUNCTION (EB,EC)
           DIMENSION T( 9)
           INTEGER I
           ENTRY TO IB6J5.
           X1=EB
           X2=EC
           TRANSFER TO IN
           ENTRY TO IB6C4.
           X1 = EB/85.
           X2 = EC*.95
IN         WHENEVER X2.L.0., FUNCTION RETURN 0.
           T( 1) =   0.56313861E-02
           T( 1) = T( 1) * X 1
           T( 1) = T( 1) * .ABS.X 2 .P. (  0.250000)
           T( 2) =   0.63145097E-07
           T( 2) = T( 2) * .ABS.X 1 .P. ( -0.500000)
           T( 2) = T( 2) * X 2 .P. (  5)
           T( 3) =   0.24672532E-02
           T( 3) = T( 3) * .ABS.X 1 .P. ( -0.250000)
           T( 3) = T( 3) * X 2
           T( 4) =  -0.13613705E-07
           T( 4) = T( 4) * X 1 .P. ( -2)
           T( 4) = T( 4) * X 2 .P. (  6)
           T( 5) =   0.99646928E-09
           T( 5) = T( 5) * X 2 .P. (  5)
           T( 6) =  -0.25777743E 00
           T( 6) = T( 6) * X 1 .P. ( -3)
           T( 7) =  -0.12208366E-08
           T( 7) = T( 7) * .ABS.X 1 .P. ( -0.333333)
           T( 7) = T( 7) * X 2 .P. (  6)
           T( 8) =   0.39028671E-05
           T( 8) = T( 8) * X 1 .P. ( -6)
           T( 8) = T( 8) * X 2 .P. (  6)
           T( 9) =  -0.54927094E-02
           T( 9) = T( 9) * .ABS.X 1 .P. (  0.250000)
           T( 9) = T( 9) * X 2 .P. ( -2)
           T(0) = 0.
           THROUGH SUM, FOR I = 1,1,I .G.  9
       SUM T(0) = T(0) + T(I)
           FUNCTION RETURN T(0)
           END OF FUNCTION
```

13. INDEX TO DECLARATIONS

APPENDIX B

REFINEMENTS OF THE STEPWISE REGRESSION PROGRAM TO PROVIDE FOR NONLINEAR ESTIMATION

APPENDIX B. TABLE OF CONTENTS

1. THE LOGIC OF STEPWISE REGRESSION

1.1 STATISTICAL REGRESSION

Regression is a method of statistical analysis. This form of analysis attempts to find a relationship among several variables and represent it in the form of an equation. In using regression analysis, we are, in effect, attempting to draw a single line through a set of data points in such a way that this line is the "best" line that describes the set of points. Actually, instead of drawing this single line through the points, we are here interested in forming a single mathematical equation to represent this line. This equation is called the regression equation.

In general, the regression analysis is used in finding relationships among several variables. Where a sufficient amount of data already exists, or can be collected, this method will prove valuable in finding the existing relationships.

1.2 STEPWISE REGRESSION

Stepwise Regression was developed by Dr. Franklin H. Westervelt in connection with this research project at The University of Michigan and is a sophistication of the regular regression discussed above. Stepwise Regression differs from regular regression in the following way.

Regular regression first requires that the form of the final regression equation be assumed and given. Then analysis is carried out to see if this equation, which was assumed, is really an adequate representation of the data. Stepwise Regression, on the other hand, does not first assume the final form of the regression equation. By trying terms out one by one in the equation, Stepwise Regression finds the form of the regression equation in a stepwise manner. Thus there is an obvious advantage to using Stepwise Regression over regular regression. When the form of the regression equation is unknown, it is advantageous to let Stepwise Regression find the equation rather than to assume it.

Stepwise Regression has direct implications for the methods of approaching and solving certain problems. In those problems where relationships among a large number of variables are to be determined, and where little or no knowledge exists of the form of the relationship, the Stepwise Regression procedure will be a valuable aid in the analysis.

Within the scope of this research project, Stepwise Regression has two important applications: (1) finding physical performance relationships to describe the technical behavior of an element in the system, and (2) finding the cost of an element in terms of its engineering design parameters.

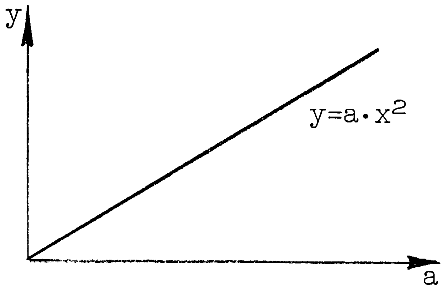
2. THE COMBINATION OF STEPWISE REGRESSION AND NONLINEAR ESTIMATION

2.1 DIFFERENCE BETWEEN THE TWO TECHNIQUES

We are also concerned with the improvement of Stepwise Regression so that it will make available more terms for our use. Particularly in optimization work we find a need for certain kinds of nonlinear terms. We have taken steps to improve this process by finding ways to introduce Nonlinear Estimation into the technique which we have found valuable in developing regression equations.*

Nonlinear Estimation is another method of statistical analysis. It is similar to Stepwise Regression, discussed above. The basic difference between the two methods of analysis lies in the kinds of terms that each method is able to utilize in its final predicting equation.

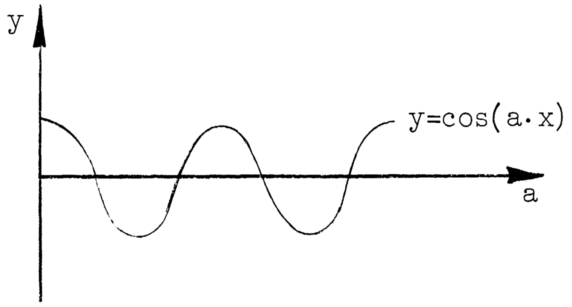
Stepwise Regression utilizes terms that are linear in their coefficients. This linear concept is displayed graphically below. For example, let $y = a \cdot x^2$, where a is the coefficient of x-squared (x^2). If we double a and leave x fixed, then the value of y doubles. This is simply a straight-line relationship and is called linear in the coefficient a.



A linear relationship in the coefficient a.

On the other hand, Nonlinear Estimation utilizes terms that are nonlinear in their coefficients. For a contrasting example to the linear case, let $y = \cos(a.x)$; the graph of y versus the coefficient a is shown below:

*Forecasting by Generalized Regression Methods, by G. E. Box, Princeton, IBM Report , 1960.



This relationship is not a straight line; hence it is called "nonlinear" in the coefficient a.

Another difference between the two methods is that Stepwise Regression can actually find the predicting relationships, as mentioned before. Nonlinear Estimation can at best analyze a known relationship. It cannot predict it.

2.2 NEED FOR COMBINED TECHNIQUES

Suppose we are confronted with a problem of relating one observed variable to several other observable quantities. We have also found after preliminary analysis that what is happening is more aptly described by a formulation in which we must remove the restriction of linearity. Instead of each coefficient being linearly related to the dependent quantity, some are formulated so that they have an exponential, logarithmic, or, in general a nonlinear relationship to the dependent variable. The problem can no longer be solved by Stepwise Regression economically.

We have then entered the domain for application of Nonlinear Estimation. By using the program we can evaluate these coefficients, which will be termed "parameters," and assess the worth of our mathematical model.

However, to use Nonlinear Estimation we need to have a mathematical model to predict the relationship. Herein lies an area of proposed future research. We plan to develop a learning mechanism that will be used in the selection of the terms for consideration in the mathematical model. The terms will be tried out in a predicting equation using both Nonlinear Estimation and parts of the Stepwise Regression procedure.

The combined program would be a strong tool of analysis in investigating both linear and nonlinear relationships. Thus the restriction of linearity would no longer exist. Secondly, this type of analysis would require no prior knowledge of the relationship between the variables. The combined procedure would not only have the applications that exist presently for Stepwise Regression, but would also be able to extend its applications into areas of more complex relationships. It is expected that this method would be valuable in basic research and applied studies in the physical and social sciences. It would also be valuable as a tool for studying and forecasting complex subtle marketing relationships in the business cycles. In our own work it will be valuable in the optimization of the design of physical systems.

The geometrical logic and the mathematical description of Nonlinear Estimation are presented in Sections 3 and 4 of this Appendix.

2.3 METHOD OF COMBINATION

The procedure or program would consist of three distinct parts: Stepwise Regression, Nonlinear Estimation and a Monitor. Stepwise Regression would attempt to find a linear relationship from among the independent variables considered. Nonlinear Estimation would attempt to find the best estimates of the nonlinear parameters in the equation. The Monitor would be charged with both the analysis of the work done by Stepwise Regression and Nonlinear Estimation and learning in the simple learning mechanism.

The simple learning mechanism would contain the knowledge of what functions of independent variable and parameters were useful in explaining the relationship between the dependent variable and the independent variables.

Basically the procedure would be first to arbitrarily select a set of functions with nonlinear coefficients. The nonlinear coefficients would be assigned a value determined by some preliminary analysis of the data. The resulting set of functions would be considered as a new set of independent variables by Stepwise Regression. Stepwise Regression would attempt to find a minimal set of these functions which described the relationships present in the data. Then Nonlinear Estimation would attempt to get the best fit of the nonlinear coefficients from the minimal set of functions. This basic process would be repeated until the termination criteria setup by the user are satisfied.

A three-cornered diagram shown below would characterize the combined program.

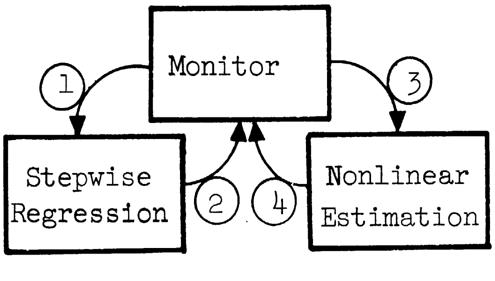


Fig. B-1. The combined program.

The Monitor would first select a library of functions. Preliminary analysis of the data would be done to obtain a best guess for the values of the nonlinear coefficients in the selected functions. The values of best guesses would be assigned to the nonlinear coefficients. The Monitor would edit the data to make the nonlinear functions appear as regular independent variables for Stepwise Regression.

(1) The Stepwise Regression Program would select those independent variables for the prediction equation that enabled it to predict the relationship and delete those variables that did not enable it to predict the relationship. Stepwise Regression would also find the meaningful interactions among the independent variables.

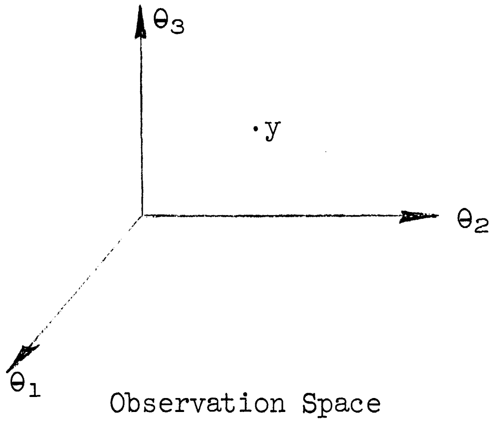
(2) Control would be returned to the Monitor. The results of Stepwise Regression would be analyzed, and the termination criteria would be tested for the existence of a satisfactory predicting equation.

(3) If the termination criteria were not all satisfied, control would go to Nonlinear Estimation. With the minimal set of terms left from Stepwise Regression, Nonlinear Estimation would find the best values for both the remaining nonlinear and linear coefficients and then return to the Monitor for analysis.

(4) The termination criteria would again be tested. If the procedure was not finished, the Monitor would observe what happened to each of the functions in the equations along with the coefficients. The learning mechanism would be updated in the Monitor. Then the process would repeat with another set of functions chosen probabilistically from the library of possible functions.

3. THE GEOMETRICAL LOGIC OF NONLINEAR ESTIMATION

To illustrate the procedures that Nonlinear Estimation employs, let us consider a general experiment in which we have made three observations of the dependent or response variable. Corresponding to these three observations we also have three observations of a single independent variable. Let each observation assume a separate dimension in a hypothetical "observation space," the set of observations becoming a single point in the observation space. Our set of three observations determines one point in this three-dimensional observation space.

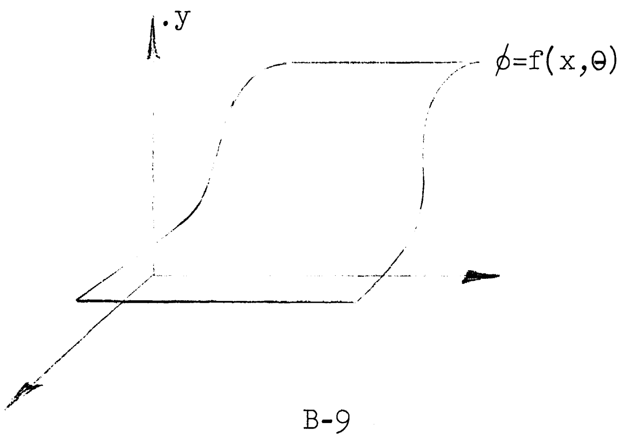


where θ_i is the ith observation and y is the observed point of the dependent variable

Observation Space

We have assumed that we can mathematically relate the observations of the dependent variable in terms of the independent (or predictor) variables. Let us call this mathematical relationship a function. For this discussion let us assume that our function has two parameters and the property that when we assign numerical values to each parameter the value of the function is uniquely determined.

If the values of these parameters of the function are allowed to vary through all their possible values, a surface of the function will be generated in our observation space. Let the parameter of the function be denoted by the single variable θ, a vector. The surface of the function might



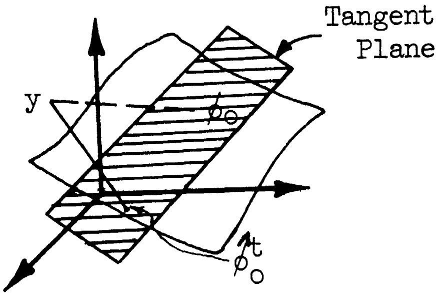
look like this for a particular set of observations on the independent variable x. Let the observed point on the dependent variable be the point y. Recall that we are trying to estimate those values of the parameters, denoted by θ, such that the distance between ϕ, the function surface, and the observed point y is a minimum.

3.1 ITERATION PROCEDURE

The method by which the values of these parameters are estimated is as follows:

(1) Select an initial set of values for the parameters. Denote them by θ_o. This determines a value for the function ϕ, called ϕ_o. Consequently, this point ϕ_o lies on the function surface in the observation space.

(2) At that point ϕ_o, construct a plane tangent to the function surface. We will use this plane as a linear approximation of the function about the point ϕ_o.

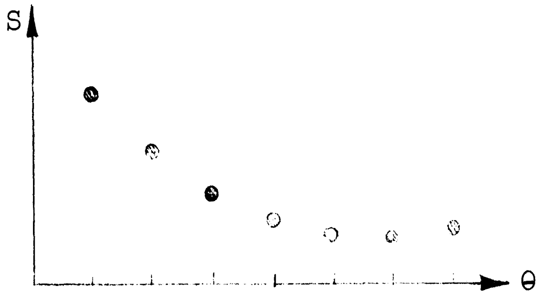


(3) Now we use a multiple linear regression analysis to find the point in the tangent plane which is closest to the point y. Call this closest point $\hat{\phi}^t_o$. We use this new point in the tangent plane to estimate a new set of values θ_1 for our parameters. At this point, $\hat{\phi}^t_o$, we construct another tangent plane and repeat the procedure of (2).

(4) We now repeat steps 2 and 3 over and over until a set of values for θ has been found such that the value of the sum of squares is a minimum.

The procedure described above, termed "iteration," is the basic mechanism of the first part of the program. In this manner the process repeats itself using successive estimates of the parameters ultimately to determine the best estimate. Since this method estimates parameters of nonlinear functions, it was given the name "Nonlinear Estimation."

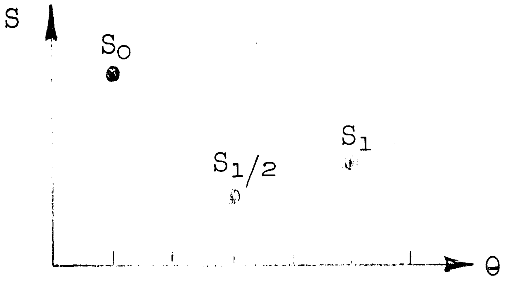
This is a convergent process on most classes of functions. The sum of the squares, S, which we are trying to minimize might look like the graph below after six steps. For example:



3.2 INTERPOLATION PROCEDURE

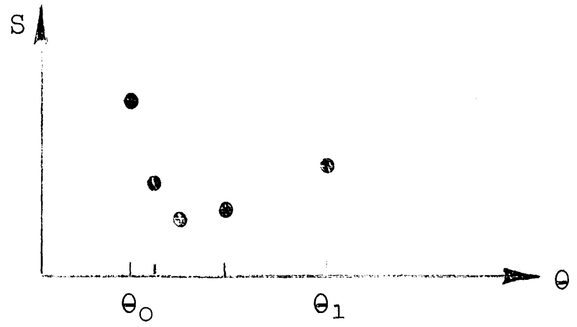
A modification to the iteration procedure is optionally available to the user. This modification is called "interpolation." Steps (1) and (2) proceed as before, but after finding the new estimates of the parameters from the regression, the following procedure is instituted.

(a) Consider the segment between θ_0 and θ_1, the difference between the first values of the parameters and our last values of the parameters. Halve this segment and at the point $\theta_{1/2}$, find the value of the function $\phi_{1/2}$. Determine the distance from the point y to the function surface at the point $\phi_{1/2}$. Let the squares of the distances from y to ϕ_0, $\phi_{1/2}$, and ϕ_1, be S_0, $S_{1/2}$, and S_1 respectively. Here S refers to the square of the distance from y to the function surface, not to the approximating tangent plane.

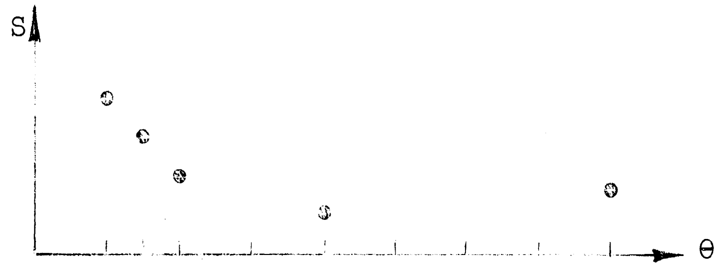


Example of Interpolation

(b) Contracting type of search: interpolation. Compare $S_{1/2}$ with S_1. If $S_{1/2}$ is less than or equal to S_1 as shown above, the segment between θ_0 and θ_1 is halved again. This half interval process is continued as long as S decreases.



(c) Expanding type of search: extrapolation. Alternatively if $S_{1/2}$ is greater than S_1, the segment between θ_0 and θ_1 is doubled. The square of the distance is determined and this process continues as long as S decreases.



(d) When S no longer decreases, the last three S points are fitted by a quadratic to find the values of the parameters corresponding to the minimum sum of squares. This new value of θ constitutes the new value of ϕ_0 for repeating steps (2) and (3) of the normal iteration procedure.

3.3 LOCAL EXPLORATION PROCEDURE

Having terminated the procedures of iteration and interpolation, we now assess the worth of the estimates of the parameters. We accomplish this by exploring the region in the neighborhood of the estimated parameters.

A confidence region can be determined which defines a contour surrounding the least squares estimates of the parameters. The formulation of the confidence region is based on an approximation to the function which is linear in parameters. If the approximation were correct and the errors were independently and identically normally distributed, the confidence region would include the true values of the parameters with a confidence coefficient of (1-α). The symbol α is the significance level of the Fisher F-distribution at the appropriate number of degrees of freedom.

The sum of squares on the confidence contour is generated from

$$S = S_m \cdot [1 + \frac{p}{n-p} \cdot F_{p,n-p}(\alpha)] \qquad (3.3\text{-}1)$$

where

S = contour sum of squares
S_m = minimum sum of squares
p = number of parameters
n = number of observations
F = Fisher's F-distribution value
α = significance level.

The geometrical significance of this confidence contour is as follows:

Consider the observation space. The value $\hat{\phi}$, of the function determined by least squares lies on the function surface. Construct a tangent plane, ϕ^t, to the surface at the point $\hat{\phi}$. Now, any point in the tangent plane will be a greater distance from y than $\hat{\phi}$. Applying the F-test, the contours at fixed distances from y in the ϕ plane are swept out. Call these contours F^t. (See Fig. B-2).

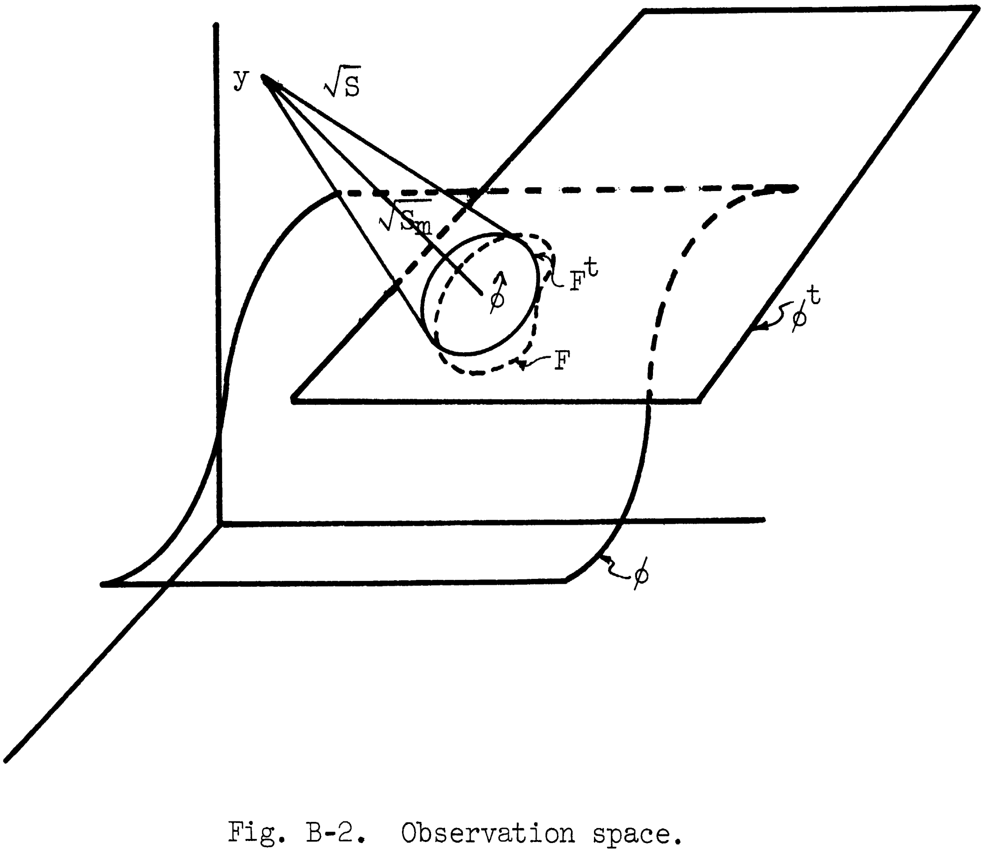


Fig. B-2. Observation space.

This contour also sweeps out a contour in the parameter space θ^t, which will be an ellipse. In general, however, the projection of the confidence contour on the function surface will not be an ellipse, but will be some nonelliptical contour. Likewise, the contour in the linearized parameter space θ^t will be an ellipse, while the contour in parameter space will be nonelliptical. (See Figs. B-3 and B-4).

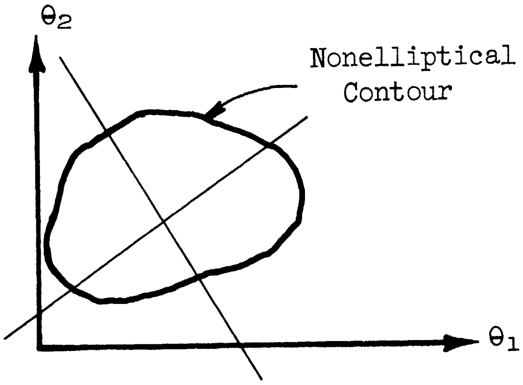


Fig. B-3. Parameter space.

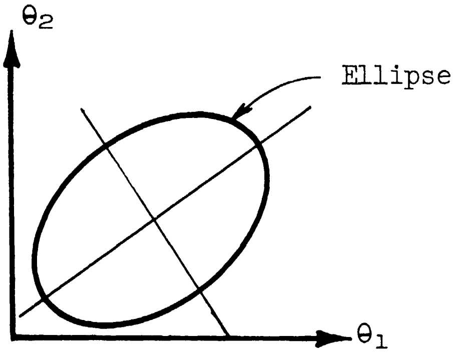


Fig. B-4. Linearized parameter space.

The contours found in the ϕ^t tangent plane are used to establish the region over which the parameters may be varied for a specified confidence in their estimation.

The relative confidence which one can place in the estimation of parameters can be judged by the elongation of the ellipse in normalized, linearized parameter space. That is, all the parameters are given the same dimensions by a normalizing factor so that if the two parameters θ_1 and θ_2 are equally well determined, the confidence contour in linearized parameter space will be a circle. If one parameter or one linear combination of parameters is better determined than another, the contour in parameter space will become elliptical. The most poorly determined parameter, or linear combination of parameters, will have the largest projection of the ellipse. The most well-determined parameter, or linear combination of parameters, will have the smallest projection of the ellipse.

Determining whether the distance $\sqrt{S_m}$ from y to $\hat{\phi}$ is a true absolute minimum is, in principle, only possible by calculating the sum of squares corresponding to each point on the surface. This task is nearly impossible in most cases, thus alternative methods are employed.

One method of gaining information about other possible minima is to redo the problem several times, each time starting from a different initial guess of parameters and seeing if we always return to the same minimum. If another local minimum is found, the values of the sums of squares are compared to ascertain which is better.

Another method of attacking this problem is to generate larger and larger contours in the tangent ϕ^t plane about the point $\hat{\phi}^t$ and calculate the sum of squares along the projections, F, of these contours on the function surface, looking for a sum of squares smaller than $\hat{S}$.

If by several applications of the previous methods there is a consistent preference for one minimum, one may infer with some degree of certainty that that is the true minimum.

With how much certainty can we use these methods of linear approximations to answer the following questions?

(1) For a certain degree of confidence, over what range may the parameters estimates vary?

(2) What is the relative confidence that can be placed in the determination of several parameters?

Information concerning the success of linear methods is obtained by comparing the contour in the normal parameter space to the elliptical contour in the linearized parameter space. If the contours are similar, the function is approximately linear in the parameters in the region of the best estimate, and these linear methods are successful. On the other hand, if the two contours are highly dissimilar, one should be suspicious of the success of these linear methods.

4. MATHEMATICAL DESCRIPTION OF NONLINEAR ESTIMATION

The following will be a fairly concise mathematical development of the same problem dealt with in the geometrical development. Except that now we have generalized the problem to include k independent variables, n observations, and p parameters. In this section little descriptive material will be included because the preceding geometrical development should have sufficiently illuminated the concepts.

Mathematical Model

$$\phi = f(x_1,\ldots,x_k;\ \theta_1,\ldots,\theta_p) \tag{4.0-1}$$

Experimental Data

Observed values of dependent variables

$$y = \begin{bmatrix} y_1 \\ y_2 \\ \cdot \\ \cdot \\ \cdot \\ y_n \end{bmatrix} \tag{4.0-2}$$

Observed values of independent variables

$$x = \begin{bmatrix} x_{11} & \cdots & x_{1k} \\ \cdot & & \\ \cdot & & \\ \cdot & & \\ x_{n1} & \cdots & x_{nk} \end{bmatrix} \tag{4.0-3}$$

Computed Values

$$\phi = \begin{bmatrix} f_1 \\ f_2 \\ \cdot \\ \cdot \\ \cdot \\ f_n \end{bmatrix} = \begin{bmatrix} f(x_{11},\ldots,x_{1k};\ \theta_1,\ldots,\theta_p) \\ \cdot \\ \cdot \\ \cdot \\ f(x_{n1},\ldots,x_{nk};\ \theta_1,\ldots,\theta_p) \end{bmatrix} \qquad (4.0\text{-}4)$$

Algorithm

$$f^t = \begin{bmatrix} f_1^t \\ \cdot \\ \cdot \\ \cdot \\ f_n^t \end{bmatrix} = \begin{bmatrix} f_1^o+(\theta_1-\theta_1^o)\left(\frac{\Delta f_1}{\Delta\theta_1}\right)^o+\ldots+(\theta_p-\theta_p^o)\left(\frac{\Delta f_1}{\Delta\theta_p}\right)^o \\ \cdot \\ \cdot \\ \cdot \\ f_n^o+(\theta_1-\theta_1^o)\left(\frac{\Delta f_n}{\Delta\theta_1}\right)^o+\ldots+(\theta_p-\theta_p^o)\left(\frac{\Delta f_n}{\Delta\theta_p}\right)^o \end{bmatrix} \qquad (4.0\text{-}5)$$

Definitions

$$\theta^o = \begin{bmatrix} \theta_1^o \\ \cdot \\ \cdot \\ \cdot \\ \theta_p^o \end{bmatrix} \qquad (4.0\text{-}6)$$

$$f^o = \begin{bmatrix} f_1^o \\ \cdot \\ \cdot \\ \cdot \\ f_n^o \end{bmatrix} \qquad (4.0\text{-}7)$$

$$D_o = \begin{bmatrix} \left(\frac{\Delta f_1}{\Delta\theta_1}\right)^o & \ldots & \left(\frac{\Delta f_1}{\Delta\theta_p}\right)^o \\ \cdot & & \\ \cdot & & \\ \cdot & & \\ \left(\frac{\Delta f_n}{\Delta\theta_1}\right)^o & \ldots & \left(\frac{\Delta f_n}{\Delta\theta_p}\right)^o \end{bmatrix} \qquad (4.0\text{-}8)$$

$$\beta^o = \theta - \theta^o \qquad (4.0\text{-}9)$$

$$f^t - f^o = D_o\beta^o \qquad (4.0\text{-}10)$$

$$W^o = y - f^o \qquad (4.0\text{-}11)$$

Multiple Regression

$$\text{Fit } (f^t - f^o) \text{ to } (y - f^o)$$

$$D_o'D_o\beta^o = D_o'(y-f^o)$$

$$D_o'D_o\beta^o = D_o'W^o$$

$$\beta^o = (D_o'D_o)^{-1}D_o'W^o \qquad (4.0\text{-}12)$$

Iteration Formula

$$\theta^{i+1} = \theta^i + (D_i'D_i)^{-1}D_i'W^i \qquad (4.0\text{-}13)$$

Objective

Minimize

$$S = \Sigma(y_i - f_i)^2 \qquad (4.0\text{-}14)$$

5. COMMUNICATING THE MATHEMATICS TO THE COMPUTER

5.1 DATA

The data must be on IBM cards with each observation of the dependent variable followed by the corresponding observations of the independent variables. The form of the numbers presented to the program is specified on a control card.

5.2 CONTROL CARDS

Two control cards are required with this program. One of the control cards specifies the control parameters for the operation of the regression portion of the program. The other control card specifies the Fisher F-values to be used in generating confidence contours. Both control cards must be present (whether or not confidence contours are to be generated).

Table 5.2-1 describes the information which must be supplied on the first control card. This table should be self-explanatory. Examples of control cards used in actual problems can be found in Section 6.3.

TABLE 5.2-1

CONTROL CARD NO. 1

Card Columns	Significance	Standard MAD Input Format
1 6	Problem Identification	C6
7 10	Trial Number	I4
11 14	Maximum Iteration Count 0⟹No Limit Otherwise ⩾ 3	I4
15	Interpolation + Activate - Suppress	
16 18	Maximum Interpolation Count 0⟹No Limit Otherwise ⩾ 3	I4
19	Preliminary Eigenvalue Analysis + Activate - Suppress	
20 22	Number of Observations	I4
23	Correlation Matrix + Compute - Suppress Computation	
24 25	Number of Independent Variables	I2
26	Parameter Increments + Compute - Supplied on Data Cards	
27 28	Number of Parameters	I2
29	Contours to be Based on + F-Values - Sums of Squares	
30	Number of Contours	I1
31	+ Print Input Data and Intermediate Stages - Suppress this Printing	
32 37	Number Used to Compute Δp's	F9.7
40	Termination Based on Relative Change in + Parameters - Sum of Squares	
41 48	Termination Error	F9.7
49 60	Output Format	2C6
61 72	Input Format	2C6
73 80	Card I.D.	---

CONTROL CARD NO. 2

Card No. 2 can contain a maximum of seven floating point numbers, punched according to the standard "MAD" input format 7F 10.4. There are three cases:

(1) Contours based on sums of squares values: A maximum of seven of these may be punched.

(2) Contours based on F-values with internal estimate of error variance: A maximum of five may be punched. Use the first five fields.

(3) Contours based on F-values with external estimate of error variance: A maximum of five F-values may be punched in the first five fields. If there are N contours desired, the estimate of the error variance must be punched in the (N+1)st field and its corresponding degrees of freedom must be punched (as a floating point number) in the (N+2)nd field.

NOTE: If local exploration is suppressed, Card No. 2 must be supplied, but it may be blank.

5.3 FUNCTION SUBROUTINE

The function subrountine may be either a "MAD" or "Fortran" subroutine. Here only "MAD" subroutines will be discussed.

A typical function subroutine might be one which computes the function

$$\sum_{i=1}^{4} A_i \left[1 - \frac{1-e^{-\alpha_i \tau}}{\alpha_i \tau} \right],$$

where τ is the independent variable, and A_i and α_i are parameters.

The calling sequence for the subroutine is EXTERNAL FUNCTION (X,K,P,LP, F,S,). The significance of each of the arguments is given below:

X is the first independent variable
K is the number of independent variables
P is the first parameter. All parameters must be stored sequentially.
LP is the number of parameters
F is the value of the function to be returned
S is a statement label for error return.

Using the above nomenclature the function subroutine is written.

```
          EXTERNAL FUNCTION (X,K,P,LP,F,S)
          STATEMENT LABEL S
          INTEGER K,LP
          ENTRY TO FUNCTN.
          F = 0.0
          THROUGH ONE, FOR I = 0,2,I.G.6
     ONE  F = P(I)*(1.0-((1.0-EXP.(-P(I+1)*X,S))/(P(I+1)*X)))+F
          FUNCTION RETURN
          END OF FUNCTION
```

It is important to note that all the parameters must have the same name, P. To accomplish this we simply assign the even locations of the P-list to A, and assign the odd locations of the P list to α.

Thus,

P(0),P(2),P(4),P(6), are assigned to $A_{1,2,3,4}$,

and

P(1),P(3),P(5),P(7), are assigned to $\alpha_{1,2,3,4}$.

We must now be sure to recall this identification of parameters when looking at the print-out from any problem performed with the use of the above function.

5.4 INITIAL PARAMETER ESTIMATES

The program must be supplied with initial parameter estimates in order to have some place from which to begin the iteration. The closer the initial estimates are to the "best" values, the less computer time will be spent in getting the answer.

The initial parameter estimates must conform to the "MAD" input format specification on control Card No. 1 and are included at the end of the data deck.

5.5 SUPPLIED PARAMETER INCREMENTS

One may elect to supply values of parameter increments rather than allowing the program to compute these increments from $\Delta p_i = 0.01 \times p_i$. If parameter increments are supplied, they must be included after the initial parameter estimates in the input deck.

5.6 ARRANGEMENT OF DECK

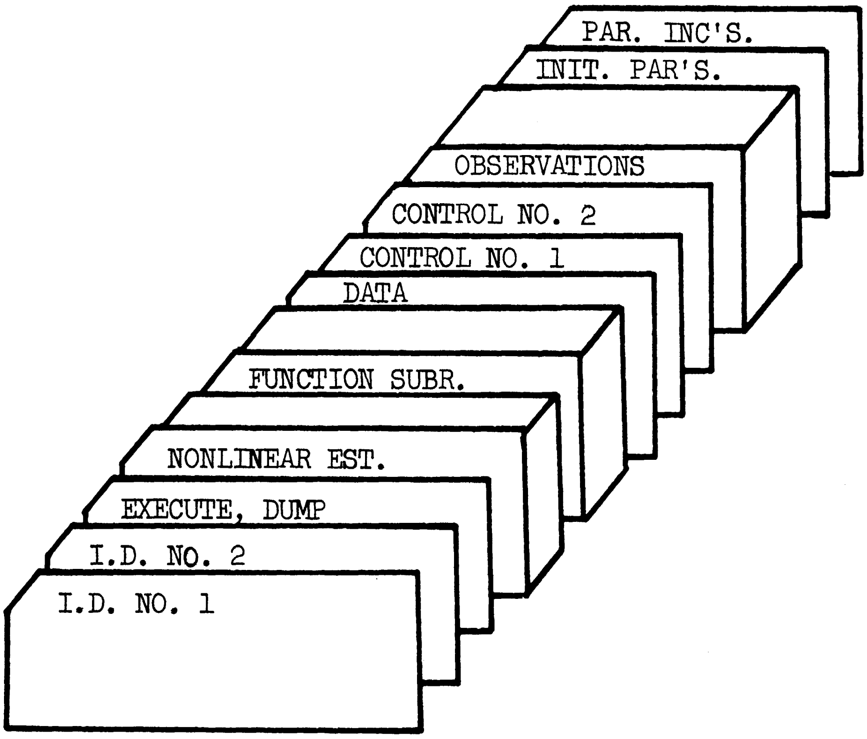


Fig. B-5.

5.7 TECHNIQUE OF PROGRAM CONTROL

At present, the nonlinear estimation program requires that the investigator supply the program with certain control information. This information consists of the maximum allowable number of iteration stages, the maximum allowable number of interpolations, information about the function and the data, printing control, and format information. Of course, each problem is different and will require different control information. However, one may make some generalizations about the control parameters which are applicable to a wide range of problems.

Maximum number of iteration stages.—The amount of machine time required for an average iteration stage is a function of the size of the problem (the number of observations, number of parameters, and complexity of the function).

For small problems the time per iteration stage may be as short as fifteen seconds, but for large problems the amount of time required per iteration stage could be as much as a minute or two.

Obviously, a problem will take less machine time and will have a smaller likelihood of going astray if the initial parameter estimates are as close as possible to the final value. This means that it is advisable for the investigator to supply the best parameter estimates available to minimize machine time and avoid the possibility of finding a minimum which may not be the true minimum because of local minima. Also, as the example problems point out, it is advisable to supply initial parameter estimates that do not require the program to move parameters through singularities in parameter space.

Thus, we see that the maximum number of iteration stages allowed depends upon how well we can estimate the initial parameters and how complicated the problem is.

Another consideration which may influence the number of allowable iteration stages is the confidence which we have in the ability of the program to find a solution. For potentially difficult problems in which we anticipate possible inadequacies in either the data or the function, it may be advisable to run only a few (say three) iteration stages and then examine the print-out to see if the program is making progress. If satisfactory progress is being made, the input control should be modified to correspond to the last stage to date and allow the program to continue for (perhaps) a larger number of stages. If it is observed that difficulty is being encountered, the information which is causing the difficulty should be modified and then another check run should be made.

Maximum number of interpolation stages.—This is at present a rather subjective matter. One may feel that he is not going to derive much benefit from interpolation and therefore suppresses it (-col. 15) or he may feel that interpolation will aid convergence and activate it. Because of the nature of the interpolation algorithm, if interpolation is activated the number of allowed interpolation stages must be $\geqslant 3$. One way to judge objectively whether interpolation will be of much help is to observe the output from the program when it is run for only three iteration stages and see if interpolation and/or extrapolation of the results from one stage to another would yield results quickly. If it is found that such an interpolation or extrapolation would help, then it is advisable to activate interpolation.

Occasionally, when the function surface is particularly rough, iteration may lead to an increasing sum of squares. In these cases, it has been found that interpolation helps to keep the program under control and not allow it to go too far afield when searching for a minimum sum of squares.

Print control.—One may call for intermediate print-out if he so desires. It is recommended that this print-out be called for in the check-out stages of solving a problem, but be deleted for "production runs." During the check-out phase the intermediate print-out may be valuable in finding sources of difficulty, however, the print-out is time consuming and should be avoided when not needed.

Format specifications.—The form of the information for the form specification is described in the "MAD" and also in "Fortran" manuals for the computer.

Termination criteria.—Along with the problem, the user must supply the criteria by which the program decides when it has solved the problem. The criterion may be either (1) that the change in parameter values should be less than E from one iteration stage to the next; or (2) that the sum of squares should be less than S. The usual criterion is the less-than-E criterion; however, when the user has an independent estimate of the sum of squares the less-than-S criterion may be useful.

Statistical control.—The user must supply information to the program for use in performing the analysis of the results obtained. This information specifies whether or not the program computes the correlation matrix, how many confidence contours are to be generated, and for what values of confidence these contours are to be generated.

To make the preceding specifications, the user must find the Fisher F-value for the confidence regions desired, using the degrees-of-freedom specifications of Eq. (3.3-1).

Summary of control.—With the nonlinear estimation program that is presently in existence one must exercise considerable judgement in specifying the control information for each problem. The format for the required control cards is discussed in general in Section 5.1 with specific control cards illustrated in Section 6.3.

Error procedure.—Several types of errors can be encountered during execution of nonlinear estimation. Several subroutines used contain error checking, and the program is equipped to take action when a subroutine encounters an error. Floating trapping mode is on during execution, and overflow during a calculation will cause an error. If underflow occurs, the contents of the register in which the underflow occurred will be replaced by a zero, and computation will proceed.

When an error is encountered the comment "Error type N at loc L" is printed. N is an integer identifying the error type. (See Error Table.) L is an octal number giving the location in core storage at which the error occurred.

ERROR TABLE

N	Cause
0	Floating trap error-check location to determine point at which trap occurred.
1	Negative argument to SQRT during computation of standard deviation.
2	Negative argument to SQRT during computation of normalizing element.
3	Negative argument to SQRT during computation of grid parameters in local exploration.
4	Function error-nonincrementing procedure.
5	Function error-incrementing procedure.
6	Error return from EVV (P.E.A).
7	Error return from INV (singular matrix).
8	Diagonal element of correlation matrix differs significantly from 1(machine error).
9	Error return from EVV (local exploration).
10	Attempt to divide by zero. Check location to determine where.

6. SAMPLE PROBLEM NO. 1

In the following example, the formulation, communication, and solution of the problem will be discussed. The sequence of events is designed to follow the method in which such problems might be solved in practice. This example problem was worked by R. W. Albrecht.

6.1 DESCRIPTION OF THE PROBLEM

Consider the problem of the compound decay of an isotope into a second isotope which then decays into a third isotope. (A case of particular interest in reactors might be $I^{135} \longrightarrow Xe^{135} \longrightarrow Cs^{135}$.) An equivalent problem might be a problem in chemical reaction kinetics, where a feed material, A, produces a product, B, which in turn produces a product, C. Symbolically, either of these cases can be represented by

$$A \xrightarrow{k_1} B \xrightarrow{k_2} C,$$

where k_1 and k_2 are rate constants. The differential equations are

$$\frac{dA}{dt} = - k_1 A$$

$$\frac{dB}{dt} = k_1 A - k_2 B$$

$$\frac{dC}{dt} = k_2 B.$$

If only the concentration of B is observed, the solution to the set of equations for the concentration of B may be written

$$B = \frac{k_1}{k_1 - k_2} (e^{-k_2 t} - e^{-k_1 t}).$$

Now, if the experimentor observes the concentration of B at several times t, and wishes to infer the rate constants k_1 and k_2 from these measurements, he has a nonlinear problem to solve which is ideal for nonlinear estimation.

6.2 DATA

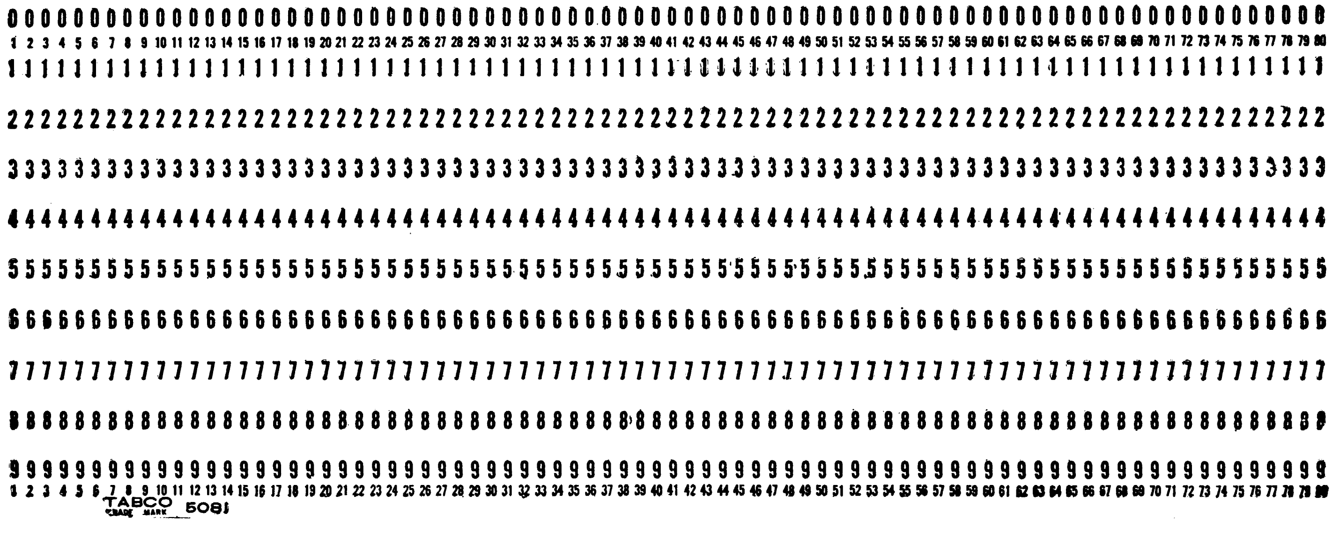
Concentration of B (dependent variable)	Time (independent variable)
0.166	10
0.192	20
0.3655	40
0.413	80
0.4355	160
0.247	320

6.3 CONTROL CARDS

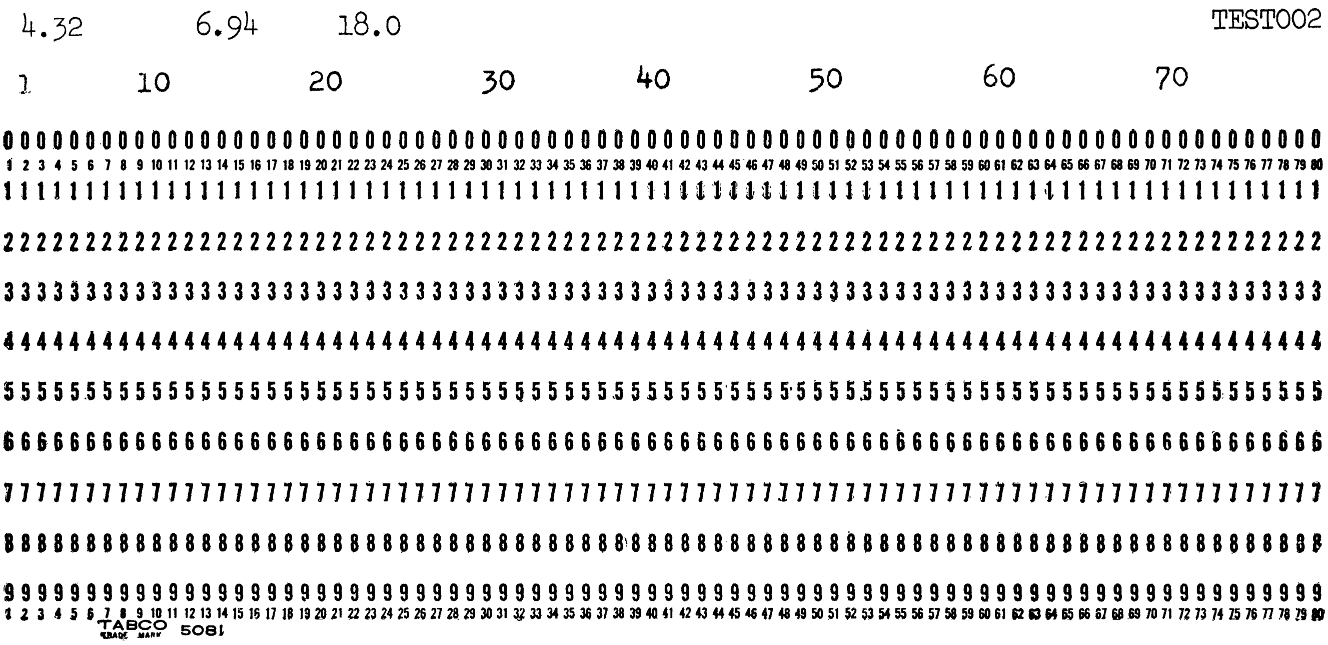
The control cards used are shown below. Refer to Table 5.2-1 for the meaning of the entries.

Card No. 1

```
 INTERP   3  10+   0+  6+ 1+ 2+3+   .01   +    .0011H1 P6E12.4*2F10.5*      TEST001
1        10        20        30        40        50        60        70
```



Card No. 2



4.32 6.94 18.0 TEST002

1
10
20
30
40
50
60
70
TABCO
5081

6.4 FUNCTION SUBROUTINE

The function subroutine which computes the desired function is listed below.

```
EXTERNAL FUNCTION (X,K,P,LP,F,S)
STATEMENT LABEL S
INTEGER K, LP
ENTRY TO FUNCTN.
F = P(0)/(P(0)-P(1))*(EXP.(-P(1)*X,S)-EXP.(-P(0)*X,S))
FUNCTION RETURN
END OF FUNCTION
```

This function subroutine is compiled and placed at the rear of the program deck. (Fig. B-5.)

6.5 INITIAL PARAMETER ESTIMATES

In this problem it is apparent that k_1 and k_2 should be positive since they are reaction rates. We, therefore, expect the resultant values of k_1 and k_2 to be positive.

We observe that when k_1 approaches k_2 we have a region in the parameter space which is nearly singular. The value of the function becomes very large.

It is, therefore, evident that if we pick initial parameters such that in the iteration process we must cross this region go get to the desired point, we will encounter difficulty. Thus, we should do some preliminary analysis of the data to ascertain whether $k_1 > k_2$ or $k_2 > k_1$ is more likely to give initial parameter estimates which cause no machine problems.

To converge to an answer in the shortest time, we pick parameters k_1 and k_2 as close to the correct values as a preliminary analysis allows. This implies making some kind of preliminary judgement of the values of k_1 and k_2 usually by an "eye fit" to the data.

Let us suppose that we have exercised the decisions required and have made the initial parameter estimates below.

6.6 RESULTS AND DISCUSSION

We now arrange the program deck according to Fig. B-5 and submit this to the computer for the actual execution of the procedure. The numerical results for this example are found at the end of this Appendix.

Some knowledge of statistics is required to analyse the results that are concerned with the quality of the final best estimate. The results indicate that there is a contour No. 1 with a contour value of 4.32 (in this case, the F-value for ($1-\alpha = 90\%$). Later on there is the minimum sum of squares, contour sum of squares, and grid sum of squares, to which attention will now be directed.

Nonlinearity considerations.—Inferences can be drawn if the nonlinear situation is linearized and a knowledge of linear statistics is brought to bear on the problem. If the particular problem under study were truly linear in the parameters, the contours which outline the confidence regions would be ellipses. As soon as a problem, such as this one, in chemical reaction kinetics becomes nonlinear in the parameters, the confidence regions become distorted. No longer are they true ellipses, but the approximation might still be good depending on how nonlinear the problem is.

If the problem is very nearly linear the sum of squares characterizing the ellipse in linearized space will be close to the sum of squares in nonlinear space. To compare these sums of squares the print-out gives the contour sum of squares and the sums of squares at the end points of the ellipse.

Notice that in this problem the contour sum of squares is larger than the minimum sum of squares. It represents an ellipse which corresponds to the range over which the parameters may vary for 90% confidence. Note also that the grid sums of squares are not greatly different from the contour sum of squares, so that we expect that the problem is approximately linear in the parameters in the region of the best estimate.

Contours No. 1, 2, and 3 are shown in the print-out. They correspond to the 90%, 95%, and 99% confidence and are thus increasingly larger ellipses. Using the grid parameters listed, Fig. B-6 was constructed.

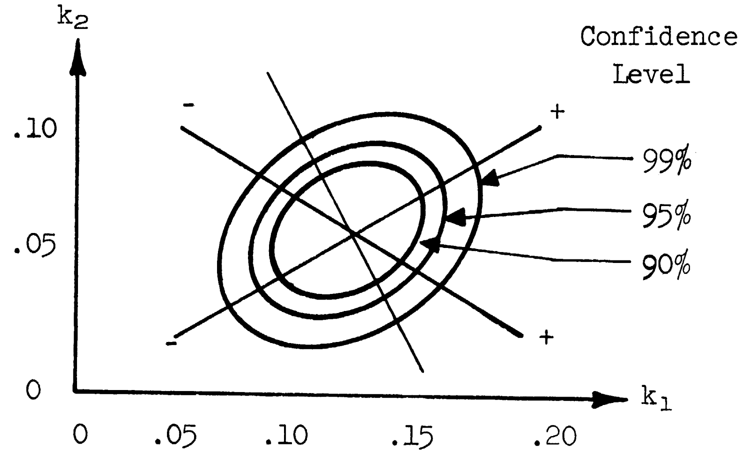


Fig. B-6. Parameter space.

Significance of parameter estimates.—The shape of the ellipse gives indications about the relative confidence placed in the estimates of the parameters. The linear combination of parameters corresponding to the longest axis is most poorly determined and the linear combination corresponding the shortest axis is best determined when the shape of the ellipse is examined in normalized parameter space.

The approximately equal eigenvalues indicate that in this problem the parameters are nearly equally well determined.

PRINT-OUT OF EXAMPLE 1

```
PROBLEM INTERP

TRIAL NO. 3

NO. OF OBSERVATIONS 6

NO. OF INDEPENDENT VARIABLES 1

NO. OF PARAMETERS 2

OBSERVATIONS
  1.6600E-01  1.9200E-01  3.6550E-01  4.1300E-01  4.3550E-01  2.4700E-01

INDEPENDENT VARIABLES
  1.0000E 01  2.000E 01  4.0000E 01  8.0000E 01  1.6000E 02  3.2000E 02

PARAMETERS
  1.2500E-02  7.0000E-03

INCREMENTS
  1.2500E-04  7.0000E-05

EIGENVALUES OF MOMENT MATRIX-PRELIMINARY
  1.0771E 03  3.7544E 03

CORRELATION MATRIX
  1.0000E 00  e.3669E-01  2.366oE-01  1.0000E 00

NORMALIZING ELEMENTS
  2.8525E-02  1.6071E-02

EIGENVALUES OF CORRELATION MATRIX
  7.6331E-01  1.2367E 00

EIGENVECTORS OF CORRELATION MATRIX
  7.0711E-01  -7.0711E-01  7.0711E-01  7.0711E-01

TRANSFORMS OF EIGENVECTORS OF CORRELATION MATRIX
  2.4789E 01  -4.3998E 01  2.4789E 01  4.3998E 01
```

CONTOUR NO. 1

Confidence Level 90%

```
CONTOUR VALUE
  4.3200E 00

CENTER OF GRID PARAMETERS - FINAL BEST ESTIMATE
  1.1857E-02  6.5755E-03

GRID PARAMETERS-(POSITIVE GRID)
  1.4083E-02  5.3214E-03  1.4690E-02  1.1717E-03

GRID PARAMETERS-(NEGATIVE GRID)
  9.6310E-03  7.8295E-03  9.0237E-03  4.9792E-03

CENTER OF NORMALIZED GRID PARAMETERS
  4.1566E-01  4.0914E-01

NORMALIZED GRID PARAMETERS-(POSITIVE GRID)
  4.9370E-01  3.3111E-01  5.1499E-01  5.0846E-01

NORMALIZED GRID PARAMETERS-(NEGATIVE GRID)
  3.3763E-01  4.8717E-01  3.1634E-01  3.0982E-01

MINIMUM SUM OF SQUARES
  7.3860E-03

CONTOUR SUM OF SQUARES
  2.3340E-02

GRID SUMS OF SQUARES-(POSITIVE GRID)
  2.4362E-02  1.9222E-02

GRID SUMS OF SQUARES-(NEGATIVE GRID)
  2.2533E-02  2.5136E-02
```

CONTOUR NO. 2

Confidence Level 95%

```
CONTOUR VALUE
  6.9400E 00

CENTER OF GRID PARAMETERS - FINAL BEST ESTIMATE
  1.1857-0e  6.5755E-03

GRID PARAMETERS-(POSITIVE GRID)
  1.4678E-02  4.9860E-03  1.5448E-02  8.5987E-03

GRID PARAMETERS-(NEGATIVE GRID)
  9.0357E-03  8.1650E-03  8.2659E-03  4.5523E-03

CENTER OF NORMALIZED GRID PARAMETERS
  4.1566E-01  4.0914E-01

NORMALIZED GRID PARAMETERS-(POSITIVE GRID)
  5.1457E-01  3.1024E-01  5.4155E-01  5.3503E-01

NORMALIZED GRID PARAMETERS-(NEGATIVE GRID)
  3.1676E-01  5.0804E-01  2.8978E-01  2.8325E-01

MINIMUM SUM OF SQUARES
  7.3860E-03

CONTOUR SUM OF SQUARES
  3.3015E-02

GRID SUMS OF SQUARES-(POSITIVE GRID)
  3.5454E-02  2.5366E-02

GRID SUMS OF SQUARES-(NEGATIVE GRID)
  3.1640E-02  3.7202E-02
```

CONTOUR NO. 3

Confidence Level 99%

```
CONTOUR VALUE
  1.8000E 01

CENTER OF GRID PARAMETERS - FINAL BEST ESTIMATE
  1.1857E-02  6.5755E-03

GRID PARAMETERS-(POSITIVE GRID)
  1.6400E-02  4.0156E-03  1.7640E-02  9.8338E-03

GRID PARAMETERS-(NEGATIVE GRID)
  7.3134E-03  9.1353E-03  6.0737E-03  3.3171E-03

CENTER OF NORMALIZED GRID PARAMETERS
  4.1566E-01  4.0914E-01

NORMALIZED GRID PARAMETERS-(POSITIVE GRID)
  5.7494E-01  2.4986E-01  6.1841E-01  6.1188E-01

NORMALIZED GRID PARAMETERS-(NEGATIVE GRID)
  2.5638E-01  5.6842E-01  2.1292E-01  2.0640E-01

MINIMUM SUM OF SQUARES
  7.3860E-03

CONTOUR SUM OF SQUARES
  7.3860E-02

GRID SUMS OF SQUARES-(POSITIVE GRID)
  8.8013E-02  4.7264E-02

GRID SUMS OF SQUARES-(NEGATIVE GRID)
  7.0716E-02  9.1996E-02
```

7. SAMPLE PROBLEM NO. 2

7.1 DESCRIPTION OF THE PROBLEM

Let us assume that we are required to design a rectangular cooling fin. We have the specifications that the steady state temperature distribution on one edge is sinusoidal and that the temperature on the other three edges are to be held constant (at T = 0, for example). Neither length (a or b) is sp cified, but the desired temperature distribution is specified.

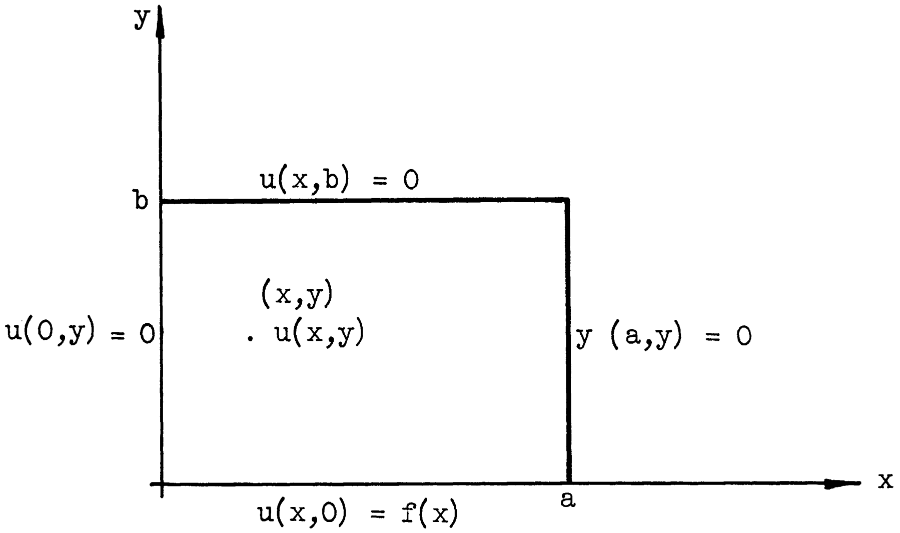


Fig. B-7.

The problem is to find the optimum lengths, a and b, of the fin (assuming that only two dimensions are important) to satisfy the specified temperature distribution in the least squares sense.

Let u(x,y) describe the steady-state temperature at point x,y and satisfy Laplace's equation in two dimensions.

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

The following equations describe the boundary conditions:

$$u(+0,y) = 0 \qquad u(z-0,y) = 0$$

$$u(x,b-0) = 0 \qquad u(x,+0) = f(x)$$

By separating the variables in the usual way, that is, setting u(x,y) = X(x)·Y(y), the functions

$$\sin \frac{n\pi x}{a} \sinh[\frac{n\pi}{a} (y-c)] \quad n = 1,2,\dots$$

are found to be solutions of the heat equation which satisfy boundary conditions in Eq. (7-1) for every constant C. When C = b, they satisfy the first condition of Eq. (7-2) and the series

$$u(x,y) = \sum_{n=1}^{\infty} A_n \sin \frac{n\pi x}{a} \sinh[\frac{n\pi}{a} (y-b)]$$

satisfies the second condition of Eq. (7-2) provided that

$$f(x) = - \sum_{n=1}^{\infty} A_n \sinh \frac{n\pi b}{a} \sin \frac{n\pi x}{a} \quad (0 < x < a).$$

According to the Fourier sine series, this is true if the coefficients A_n are determined so that

$$- A_n \sinh \frac{n\pi b}{a} = \frac{2}{a} \int_0^a f(x) \sin \frac{n\pi x}{a} dx.$$

Using this, the formal solution can be written

$$u(x,y) = \frac{2}{a} \sum_{n=1}^{\infty} \frac{\sinh[\frac{n\pi}{a} (b-y)]}{\sinh(\frac{n\pi b}{a})} \sin \frac{n\pi x}{a} \int_0^a f(x) \sin \frac{n\pi x}{a} dx .$$

Now, if the face temperature, along the x-axis is sinusoidal;

$$u(x,0) = 500 \sin \frac{\pi x}{a} ,$$

then the solution may be written

$$u(x,y) = \frac{500 \cdot \sinh[\frac{\pi}{a} (b-y)]}{\sinh (\frac{\pi b}{a})} \sin \frac{\pi x}{a} ,$$

since

$$\int_0^a \sin\frac{\pi x}{a}\sin\frac{n\pi x}{a}\,dx = \begin{cases} \frac{a}{2} & n=1 \\ 0 & n\neq 1. \end{cases}$$

At this point, the problem is specifically to find the "best" parameters, a and b, which give the least-squares fit of the data for x,y, and u(x,y).

Other information that might be useful is the degree of confidence which may be placed in these estimates, the difference between the supplied temperature distribution and the temperature distribution which would exist in the fin with the parameters a and b, and the amount that a and b could change and not effect the temperature distribution more than some pre-determined amount.

7.2 DATA

Temperature at the point

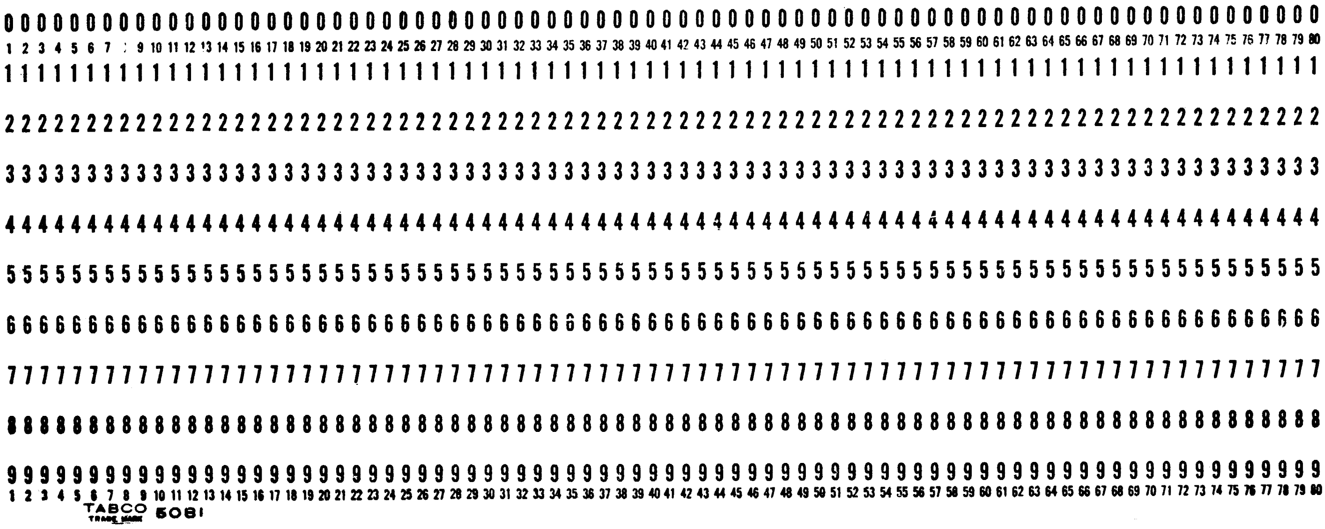
(x,y)	x	y
45	.1	.3
72.5	.2	.4
86	.3	.5
86	.4	.6
74.5	.5	.7
55.5	.6	.8
35	.7	.9
1.5	.8	1
410	.9	.1
344	1.0	.2
286	1.1	.3
225	1.2	.4
168	1.3	.5
118	1.4	.6
75	1.5	.7
40.5	1.6	.8
15.5	1.7	.9
.5	1.8	1
29.5	1.9	.5
1.0	2.0	.3

7.3 CONTROL CARDS

The control cards used in this example are shown below.

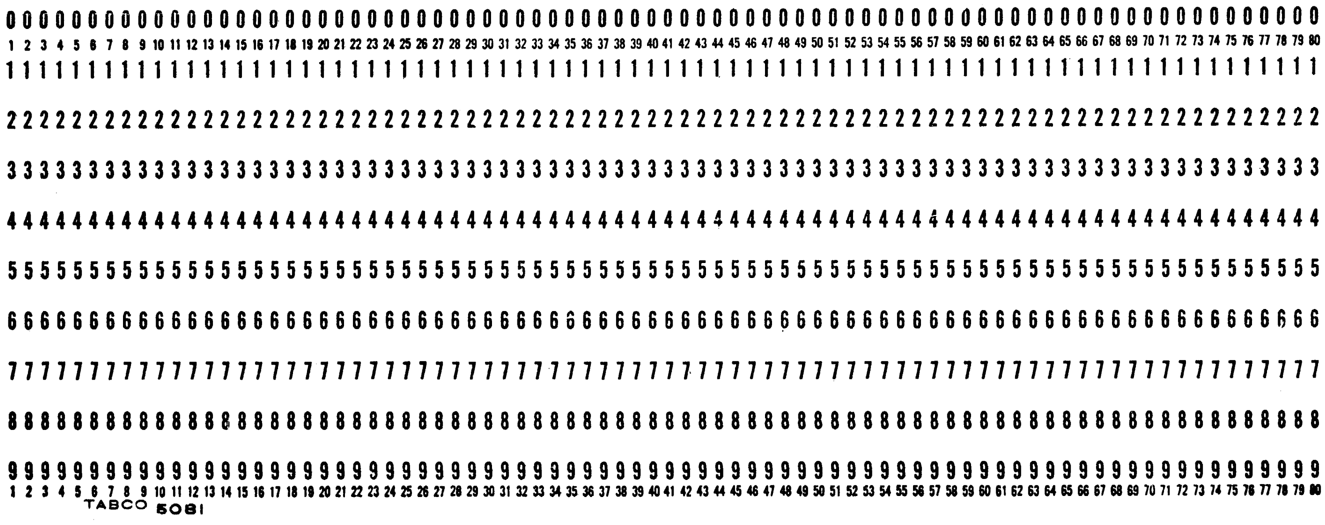
Card No. 1

```
 TEMPS   1   5   20 20   2  2  3    .01     .0011H  1P6E12.4*2F10.5*       Card 1

1        10        20        30        40        50        60        70
```



Card No. 2

```
 2.0     5.0     10.0                                                       Card 2

1        10        20        30        40        50        60        70
```



7.4 FUNCTION SUBROUTINE

The MAD Function used to evaluate u(x,y) for this problem is shown below.

```
EXTERNAL FUNCTION (X, K, P, LP, F, S)
STATEMENT LABEL S
INTEGER K, LP
ENTRY TO FUNCTN.
21=(3.1416/P(0))*(P(1)-X(1))
22=3.1416*P(1)/P(0)
23=3.1416*X(0)/P(0)
S1=(EXP.(21,5)-EXP.(-21,5))/2.0
S2=(EXP.(22,5)-EXP.(-22,5))/2.0
S3=SIN.(23.5)
F=500.*S1*53/52
FUNCTION RETURN
END OF FUNCTION
```

7.5 INITIAL PARAMETER ESTIMATE

The parameters "a" and "b" are initially estimated to be, 2.5 and 1.5 respectively.

7.6 RESULTS AND DISCUSSION

The final best estimates of "a" and "b" are 2.0 and 1.0 respectively.

From the print-out under Example 2 we see that the correlation matrix has small off diagonal elements and that the eigenvalues of the correlation matrix are not very different. Thus we are able to have some confidence in these estimates.

The confidence contours obtained indicate the tolerance one may have in designing the fin close to the specifications. The comparison of grid sums of squares with contour sums of squares indicates that the function surface is nearly linear in the region of the final best estimates of "a" and "b."

PRINT-OUT OF EXAMPLE 2

```
PROBLEM TEMPS

TRIAL NO. 1

NO. OF OBSERVATIONS 20

NO. OF INDEPENDENT VARIABLES 2

NO. OF PARAMETERS 2

OBSERVATIONS
  4.5000E 01  7.2500E 01  8.6000E 01  8.6000E 01  7.4500E 01  5.5500E 01
  3.5000E 01  1.5000E 00  4.1000E 02  3.4400E 02  2.8600E 02  2.2500E 02
  1.6800E 02  1.1800E 02  7.5000E 01  4.0500E 01  1.5500E 01  5.000E-01
  2.9500E 01  1.0000E 00

INDEPENDENT VARIABLES
  10.0000E-02  3.0000E-01  2.0000E-01  4.0000E-01  3.0000E-01  5.0000E-01
   4.0000E-01  6.0000E-01  5.0000E-01  7.0000E-01  6.0000E-01  8.0000E-01
   7.0000E-01  9.0000E-01  8.0000E-01  1.0000E 00  9.0000E-01 10.0000E-02
   1.0000E 00  2.0000E-01  1.1000E 00  3.0000E-01  1.2000E 00  4.0000E-01
   1.3000E 00  5.0000E-01  1.4000E 00  6.0000E-01  1.5000E 00  7.0000E-01
   1.6000E 00  8.0000E-01  1.7000E 00  9.0000E-01  1.8000E 00  1.0000E 00
   1.9000E 00  5.0000E-01  2.0000E 00  3.0000E-01

PARAMETERS - INITIAL ESTIMATES
  2.5000E 00  1.5000E 00

INCREMENTS
  2.5000E-02  1.5000E-02

EIGENVALUES OF MOMENT MATRIX-PRELIMINARY
  1.8618E 05  2.8940E 04

CORRELATION MATRIX
  1.0000E 00  -2.3758E-01  -2.3758E-01  1.0000E 00

NORMALIZING ELEMENTS
  1.7112E-03  1.7859E-03
```

```
EIGENVALUES OF CORRELATION MATRIX
  7.6242E-01  1.2376E 00

EIGENVECTORS OF CORRELATION MATRIX
  7.0711E-01  7.0711E-01  -7.0711E-01  7.0711E-01

TRANSFORMS OF EIGENVECTORS OF CORRELATION MATRIX
  4.1322E 02  3.9595E 02  -4.1322E 02  3.9595E 02
```

```
                              CONTOUR NO. 1


CONTOUR VALUE
  2.0000E 00

CENTER OF GRID PARAMETERS - FINAL BEST ESTIMATE
  2.0000E 00  1.0027E 00

GRID PARAMETERS-(POSITIVE GRID)
  2.0049E 00  1.0078E 00  1.9939E 00  1.0091E 00

GRID PARAMETERS-(NEGATIVE GRID)
  1.9952E 00  9.9769E-01  2.0062E 00  9.9631E-01

CENTER OF NORMALIZED GRID PARAMETERS
  1.1688E 03  5.6148E 02

NORMALIZED GRID PARAMETERS-(POSITIVE GRID)
  1.1716E 03  5.6430E 02  1.1652E 03  5.6507E 02

NORMALIZED GRID PARAMETERS-(NEGATIVE GRID)
  1.1660E 03  5.5866E 02  1.1724E 03  5.5789E 02

MINIMUM SUM OF SQUARES
  9.3723E 01

CONTOUR SUM OF SQUARES
  1.1455E 02

GRID SUMS OF SQUARES-(POSITIVE GRID)
  1.1527E 02  1.1443E 02

GRID SUMS OF SQUARES-(NEGATIVE GRID)
  1.1547E 02  1.1576E 02
```

```
                              CONTOUR NO. 2


CONTOUR VALUE
  5.0000E 00

CENTER OF GRID PARAMETERS - FINAL BEST ESTIMATE
  2.0000E 00  1.0027E 00

GRID PARAMETERS-(POSITIVE GRID)
  2.0077E 00  1.0107E 00  1.9903E 00  1.0129E 00

GRID PARAMETERS-(NEGATIVE GRID)
  1.9924E 00  9.9477E-01  2.0097E 00 9.9258E-01

CENTER OF NORMALIZED GRID PARAMETERS
  1.1688E 03  5.6148E 02

NORMALIZED GRID PARAMETERS-(POSITIVE GRID)
  1.1732E 03  5.6594E 02  1.1631E 03  5.6716E 02

NORMALIZED GRID PARAMETERS-(NEGATIVE GRID)
  1.1643E 03  5.5703E 02  1.1745E 03  5.5580E 02

MINIMUM SUM OF SQUARES
  9.3723E 01

CONTOUR SUM OF SQUARES
  1.4579E 02

GRID SUMS OF SQUARES-(POSITIVE GRID)
  1.4739E 02  1.4558E 02

GRID SUMS OF SQUARES-(NEGATIVE GRID)
  1.4831E 02  1.4875E 02
```

CONTOUR NO. 3

```
CONTOUR VALUE
  1.0000E 01

CENTER OF GRID PARAMETERS - FINAL BEST ESTIMATE
  2.0000E 00  1.0027E 00

GRID PARAMETERS-(POSITIVE GRID)
  2.0108E 00  1.0140E 00  1.9863E 00  1.0171E 00

GRID PARAMETERS-(NEGATIVE GRID)
  1.9893E 00  9.9147E-01  2.0138E 00  9.8839E-01

CENTER OF NORMALIZED GRID PARAMETERS
  1.1688E 03  5.6148E 02

NORMALIZED GRID PARAMETERS-(POSITIVE GRID)
  1.1751E 03  5.6778E 02  1.1607E 03  5.6951E 02

NORMALIZED GRID PARAMETERS-(NEGATIVE GRID)
  1.1625E 03  5.5518E 02  1.1768E 03  5.5345E 02

MINIMUM SUM OF SQUARES
  9.3723E 01

CONTOUR SUM OF SQUARES
  1.9786E 02

GRID SUMS OF SQUARES-(POSITIVE GRID)
  2.0060E 02  1.9719E 02

GRID SUMS OF SQUARES-(NEGATIVE GRID)
  2.0335E 02  2.0419E 02
```

APPENDIX C

COST ANALYSIS

APPENDIX C. TABLE OF CONTENTS

1. EQUIPMENT COSTS

The ultimate goal of this project is to design the most economical power plant possible. To do this the cost of the equipment in the plant must be calculated along with the cost of operating the plant once it has been built. It will then be possible to compare the total cost of one design with the total cost of another.

Two possible methods were considered feasible for determining the cost function of the different equipment components of a power plant. One method uses the actual cost of power plants which have already been built; the other method is based on a manufacturer's price book, making it possible to compute the price of a component much as an estimator would calculate the price. Both these procedures could be used for any piece of equipment in a power plant if past costs of the same component in different plants can be found or if a manufacturer's price book can be used. However, for purposes of explanation, the following example will be limited to the procedures used by the procedures used by the author when studying condensers and condensing equipment.

The reader should keep in mind what is being attempted. The purpose of this study is to find a variable or group of variables that will best predict the cost of the condensing equipment for a new power plant (for feedwater heater equipment or boiler equipment if such a component were being studied). At this junction in the study the investigator has no idea which or how many of the hundreds of variables that have been collected will most affect cost. Obviously, most of the variables, if not all, may affect the cost, at least slightly, but the purpose here is to find those few variables which most affect the cost.

Another restriction which must be kept in mind is that we must be able to compute from the heat balance every one of the variables that are even possibilities for the cost equation. This restriction is imposed because, in the final computer program, all the cost equations use, as inputs, values of variables which are computed by the heat balance which is generated by the simulator. Fortunately, some variables, such as condenser surface, may still be considered, even though this variable does not come directly from a heat balance, since it is a function of other variables which are computed in the heat balance equations.

1.1 CALCULATION OF COST FUNCTIONS USING ACTUAL DATA

First, all the data that might be used had to be compiled. This was done by referring to the contracts of completed jobs and all the data that might be considered pertinent were punched onto IBM cards. Such data included all the data on costs which were available, any performance data which were considered pertinent, and miscellaneous information such as delivery date, proposal date, escalation information, etc. Well over 500 different pieces of data were collected for each group of condensing equipment which had been designed in the past fifteen years by Commonwealth Associates, Inc.

Once this had been completed, the author had to decide which of these jobs would be used. Every past job was included if it was felt that the data for that job were sufficiently complete for the purposes of the analysis. Nineteen different jobs were thus included in the study.

Once the initial step of collecting the data was completed, the cost figures had to be analyzed. For most of the jobs in the study, one lump-sum cost was given for all the condensing equipment; the major pieces included were the condenser, the condensate and circulating water pumps, the tubes, and the air-removal equipment. However, variations were found: some jobs did not include the tubes since they were contracted for separately, others did; a few jobs included some motors, most did not; all jobs included some kind of air-removal equipment but some of them used vacuum pumps while others used air and hogging ejectors. Because these variations would make any cost comparison meaningless, it was necessary to find a common group of equipment whose cost would be included in each job. This was done by including all the tube costs that had been excluded, using estimates where actual figures were unavailable. The cost of air-removal equipment was taken out of each job because it was felt that this cost difference would not be reflected by the variables studied. Motors were also excluded, but only for the sake of convenience; it was easier to eliminate such cost from the few jobs where they were included rather than including them in the many jobs where they were not originally included. The cost remaining for each job accurately reflected the cost of a common group of equipment, the condensers, tubes, and pumps. One other cost that was included in each job was the cost of supervision of erection.

One may wonder how a tube price could be estimated today for equipment that was built in 1950. Or he might wonder how an accurate comparison can be made between the prices of equipment, some of which were bought in 1950 and some in 1960. The answer to both of these problems lies in the use of escalation techniques which will be explained in the following paragraphs.

The first problem that must be considered arose because of the practice of sometimes quoting a firm price, one that is fixed at the beginning of the contract even though delivery will not be made for two or three years, rather than an escalated price, which means that one amount is fixed with the possibility of raising or lowering that price depending upon whether some inflationary characteristic (like the cost of living) rises or falls during the period. In the cases where a price which was to be escalated was quoted, a maximum escalation amount was also named, such as 20% of that price. This meant that the highest price to be paid for the equipment could not exceed 120% of the base price. However, there was no information given to help determine whether the maximum escalation figure was used or whether prices did not rise as rapidly as had been expected, and therefore, a lesser figure was used. To help solve this dilemma, the Handy Cost Index numbers were used.

This cost index has been prepared for the electric light and power industry and compares prices for any given kind of equipment from turbines to untreated 35 ft chestnut poles using the 1911 price as the base price. This index works identically like the cost of living index; in fact it really is a cost of living index, but for a power plant rather than consumer goods. Since there was an index to be used specifically for condensers and tubes, it was an easy task to make the necessary adjustment for the prices of equipment bought in different years.

On the problem of firm and escalated prices, there was a lack of information about the exact amount of escalation actually used when only a maximum amount was given. (For example a contract might state that the price was $100,000 subject to a maximum possible escalation of 10% which would be determined by the value of the Metals and Metals Products Index at some future date.) There is a possibility that no escalation would be used since the index might not rise; then, the assumption of a maximum escalation would give a clearly erroneous result. To solve this problem, the Handy Cost Index at the beginning of the escalation period was compared with the same index at the end of the period. It was felt that the index actually used in the contract and the Handy Cost Index would be comparable. Therefore, if the Handy Cost Index rose more than the maximum allowed in the contract, the contract maximum was used, since it was known that the index actually used could not exceed the maximum figure given in the contract. On the other hand, if the percent rise of the Handy Cost Index was less than allowed in the contract, the percent rise given in the Index was used as the "best guess" of what was actually done.

The value of this procedure can best be seen by a simple illustration. Suppose there are two contracts which are bid on at the same time and scheduled for completion at the same time. But one bid uses a firm price, giving the price to be paid two years from now, when the job is completed. The other uses an escalation price, with a maximum escalation percent given. For example, the firm price could be $115,000, payable two years in the future when the equipment is delivered. The price subject to escalation could be $100,000 subject to a maximum possible escalation of 20% over a two-year period. If the index used rises only 10% during that time, then the price to be paid will be $100,000 plus 10%, = $110,000. But if the index rises 20% or more, then the amount due will be $100,000 plus 20%, = $120,000. Regardless of the amount of escalation, the price computed at the end of the period is comparable to the firm price since both are equal to the amount actually paid by the buyer.

By the same token, the cost data actually used were adjusted in this manner so that all the costs would reflect the amount actually paid by the buyer at the time of delivery.

One further problem with escalation remains. The costs which have thus far been computed are the cost at the delivery date. But some jobs were delivered in 1948, others in the 1950's and some in 1961. To make an accurate comparison of these prices, it was necessary to adjust all these prices to a 1961 figure, by once again using the Handy Cost Index. For example, the condensing equipment for one plant might have cost $200,000 in 1950. Between 1950 and 1961 the Handy Cost Index rose from 500 to 750 (figures assumed for ease of explanation), a rise of 150%. Therefore the 1961 price of this equipment would be $200,000 plus 50% = $300,000. This 1961 figure, the $300,000, is the cost which would then be used in all the future analyses of the condensing equipment costs.

In this same way, all the different condensing equipment costs were recomputed to a 1961 figure. Obviously the equipment in the oldest plant was escalated the most, the equipment in the newest plant, the least. The real question that is to be answered by this procedure is, "How much would a given piece of equipment cost if delivered in 1961?"

When, for instance, a 1952 unit did not include tubes, the tube price was estimated and added into the cost. Here the reverse procedure of escalation was used, again using the Handy Cost Index.* In this manner, all the different units included the same pieces of equipment with a price estimated for the time of delivery.

Now that the same pieces of equipment are included in each unit and all the prices are based on a 1961 level, one of the few steps remaining is to choose the variables which might measurably affect cost. It was decided that four variables would merit study here: condenser surface, circulating water flow through the condenser, loss (the steam condensed times heat rejected), and saturation temperature. The equation, therefore, was expected to be of the form, $ = f(surface, flow, loss, temp). These data were fed into the regression program (explained in Appendix B), and an equation generated of the above form which can now be used to predict any future condenser costs, given the values of surface, flow, loss, and temperature.

In spite of the fact that such a formula is based on today's prices, it will be valid in the near future for the very important reason that the formula

*Obviously this procedure was not really necessary, since the price started out as a 1961 price, was escalated back to an earlier year, added to the total price, and then the total escalated forward to 1961. This was done, however, not only to give any future investigator the choice of using different escalation techniques, but of also having at least a base price with which to work.

is used only for comparing two different condensers to determine which is the most economical. (It will not be accurate for predicting the cost of any single condenser.) This formula will still show which of the two condensers is more expensive since, in most instances, it is a valid assumption that prices will be raised for all sizes of condensers, not just in isolated cases.

In the same manner that a cost function has been developed for condensers, cost functions can be developed for the other kinds of equipment, whether they are feedwater heaters or steam turbines.

1.2 CALCULATION OF COST FUNCTIONS USING MANUFACTURER'S PRICE BOOK

There is still one problem, however, that cannot be solved by the method explained above. Such a problem arises when a condenser is bought either from a manufacturer who pads his bid because he is so busy that he does not care if he gets the job or from one who undercuts his normal profit margin because the company needs the business badly. In both instances, the price is out of line, either high or low, and when such a price is included in the data used by the regression program, large errors can result. If enough data were available, this would not be a problem, since the unusual prices would be such a small percentage of the prices used. But in the study here, three or four condenser units were felt to have unrealistic prices, either high or low and therefore the value of the final equation was questioned.

For this reason, a condenser manufacturer's price book was used instead of the actual data used above. Obviously, the use of a price book eliminates not only this problem of price fluctuations which cannot be explained by inflation but also the inflation problem which was solved previously by the use of the Handy Cost Index. The reason for this is obvious: any cost in the price book is based on the same time scale as any other cost. Therefore any price computed using the price book is a 1961 price.

The method used was the same procedure that an estimator would use in estimating a condenser bid. The condenser size was chosen along with the other variables of flow, loss, and temperature used in the earlier study with actual cost data. Then the procedure outlined in the price book for calculating cost was followed. This was repeated for ten different sizes of condensers and a regression run made on these data, giving an equation again with cost being a function of the same four variables used in the other analysis. It was shown statistically that this method of using price book data to generate a cost equation gave a much better fit to the data points than did the other method, using actual data, thus substantiating the hypothesis that there were some cost factors which could not be isolated and eliminated when the actual data were used.

2. OPERATING COSTS

In addition to the equipment cost, the cost of operating the plant must be predicted. This operating cost will be primarily reflected by the amount of coal used; a second, lesser cost which will also be included is the cost of maintainance.

The first step is to decide how much coal the design being studied will use in any given year. This is done by the use of the load duration data, which was supplied by Consumers Power Company. These data, which have been compiled in tabular form in Table 2.0-1, gives the percent of hours in a year that the plant will operate at any given load. It is then an easy task to use the heat balance to determine the amount of coal used in terms of Btu/KWH for each KW load during the year. Then, by multiplying by the KW load being studied and the hours from the proper place in the load duration table, the total Btu's can be calculated for each load at which the design is expected to operate. The total Btu's used during the year will be the sum of the Btu's at each load.

Now that the amount of coal has been calculated, the cost of that coal must be computed. Since the operating cost is being predicted for each year of a 35 year period, it is obvious that some account must be taken of the increase in the cost of coal during that time. This has been done using a formula computed by Consumers Power Company, which increases the cost of coal for each future year of operation. Therefore, the total cost of coal for a given year is generated by giving this coal cost function the amount of coal used during the year. To this cost will be added the maintainance cost, which will be calculated from data received from Commonwealth Associates. The total of these two costs is the total operating cost for the year.

TABLE 2.0-1

LOAD DURATION DATA

Percent	Year of Operation**							
of Full Load	1960	1965	1970	1975	1980	1985	1990	1995
100	55*	50*	45*	35*	20*	10*	5*	0*
85	15	15	15	20	20	15	10	5
70	5	10	10	15	20	15	10	5
55	5	5	5	5	10	10	5	5
40	5	5	5	5	5	5	5	0
25	5	5	10	10	5	0	0	0
0	10	10	10	10	20	45	65	85

*Hours of operation in percent of yearly total.

**Year of operation - these figures are based on a plant completed in 1960.

3. TOTAL COSTS

Once the total equipment cost and total operating costs have been calculated, it is necessary to combine these two costs to determine the total revenue required for any given design. It should be kept in mind that overhead costs have not been included because these are the same irrespective of the design studied.

The total cost of the equipment must be multiplied by the fixed charge rate, (11.9% was used in this study*). This fixed charge rate represents the income return required on a capital investment necessary to earn a return of 6.75% on the capitalization of the operation. The product of the equipment cost and fixed charge rate is referred to as the carrying charge and must be earned every year throughout the life of the plant.

The operating cost for a given year must then be added to the carrying charge to determine the total revenue requirement for that year. To make a valid comparison, the present worth of the annual revenue requirements must then be calculated. These can all be added together to get the total revenue required for this design based on a present worth basis.

It will then be an easy task to compare the costs of any two designs. The design which requires the least total revenue, using the methods described above, will be the most economical one to build and operate for the next thirty-five years.

*This 11.9% figure, which has been levelized, is broken down as follows: return - 6.75%; depreciation - 1.01%; federal income tax - 2.79%; property tax - 1.10%; insurance 0.06%; and franchise tax - 0.18%.

APPENDIX D

OPTIMIZATION PROGRAM

APPENDIX D. TABLE OF CONTENTS

ABSTRACT

This Appendix describes a computer program capable of finding the values for the arguments of a function which make the value of the function a minimum. There is no restriction on the function other than being able to program it as a subroutine. The program was written mainly for nonlinear functions with nonlinear constraints on the arguments. A function may have false or local minima and may have an arbitrary number of finite discontinuities.

1. INTRODUCTION

Most design problems can be reduced to the following statement: "I have a number of parameters. There are functional relations which put constraints on the parameters. There is some function of the parameters which has a value representing the desirability of the design. Find the values of the parameters which yield the most desirable design." In general the direct solution of this problem is not attempted because either the functional relations are not known or no mathematical simplification of the functions is possible.

In those cases where the constraints on the parameters are known and some function of the parameters can be defined such that the value of the function decreases when a more desirable design is obtained from a new choice of parameter values, a computer can be used to find the best values for the parameters.

This Appendix describes a computer program that has been tested on a variety of functions for which finding the minimum is considered difficult, i.e., functions the values of which have some or all of the following properties: (1) finite discontinuities for continuous parameter values, (2) combinations of nonlinear functions of the parameters, (3) complicated interactions of the parameters, (4) almost flat or exponential behavior, and (5) many local or false minima. The constraint relations between the parameters may also have any or all of the first four properties given above. The program was able to find an acceptable approximation to the minimum for each test in a reasonable amount of time.

This program was developed from research on electric power plant design where the problem was to find the optimum design—hence the name optimization program.

2. GENERAL COMMENTS ABOUT THE PROGRAM

The following must be supplied by the user:

(1) Control cards which give the number of parameters, the tolerance for each parameter, the maximum and minimum value for each parameter, and a limit to the number of times the function may be evaluated.

(2) A function, in the form of a subroutine, which will return a single value for any set of parameter values within the specified constraints.

(3) If the parameters are to have constraints other than those given on the control card, a subroutine must be supplied that will compute the maximum and minimum value of each parameter. The current values of all other parameters will be available to the subroutine.

The amount of computer time required to find the minimum depends on many factors. First, the length of time required to evaluate the function is important since this must be done many times. The optimization program requires on the average less than .2 second between evaluations of the function. Second, the number of times the function must be evaluated increases rapidly as the number of parameters is increased. Currently the program allows ten parameters, but it seems reasonable to consider up to 50 parameters for some relatively simple functions. The tolerance, or resolution, of the parameter values greatly affects the number of times the function must be evaluated. A tolerance of 1 part in 10^6 is about the best that can be achieved. This means that if the value of a parameter ranged from -10 to 20 then its value could be determined to within $\pm$ 30 x 10^{-6} of its true best value. Again due to time considerations, tolerances from 1 part in 100 to 1 part in 10^4 are usually used. The final factor which influences the time required to get a solution is the general behavior of the function. If the function has many false or local minima, computation time may be greatly increased. If the function has many discontinuities, or is highly nonlinear, or has many high-order interactions, some additional evaluations will be required. But the program was especially written to handle such cases and therefore, in general, will not require much more time.

If functional constraints are used, there is generally very little difference in running time. But this could become a factor if many parameters have constraints which are much more severe than those given on the control cards.

The general flow diagram is given in Fig. D-5, and the discussion of various procedures used is given in the following sections.

3. WEIGHTED RANDOM SELECTION

Let the range of values of each parameter be divided into n equal intervals. Also let a weight, W_{ij}, be associated with the j^{th} interval of the i^{th} parameter such that

$$\sum_{j=1}^{n} W_{ij} = 1$$

and

$$W_{ij} > 0. \qquad (3\text{-}1)$$

Now consider the sequence

$$S_o, S_1, S_2, \ldots, S_n,$$

where

$$S_k = \sum_{j=0}^{k} W_{ij}, \quad S_o = W_{io} \equiv 0, S_n = 1, \qquad (3\text{-}2)$$

which segments the range from zero to one into n intervals. If a random number is between zero and one it is easily seen that the probability this number lies in the interval

$$S_{k-1} \text{ to } S_k$$

is just W_{ik}. This means the probability of choosing the value of the i^{th} parameter from the k^{th} interval is also W_{ik}. A procedure to determine those weights is given in a later section.

The procedure for determining the value of a parameter by a random weighted selection is used to generate initial points for other procedures, which are described later. The computation of the initial point is as follows.

Let a_i be the minimum value and b_i be the maximum value of the i^{th} parameter, then define

$$c_i = (b_i - a_i)/n. \qquad (3\text{-}3)$$

Generate a random number r_i.

The value of the i^{th} parameter V_i is defined by

$$V_i = (k-1)c_i + (r_i - S_{k-1})c_i/W_{ik}, \qquad (3\text{-}4)$$

where k is such that

$$S_{k-1} < r_i < S_k$$

and

$$(V_1, V_2, \ldots, V_m)$$

is the required initial point.

4. PROCEDURE FOR FINDING "PREFERRED DIRECTION"

The initial point (X_o,Y_o) has been selected by some other means. Z_o, the value of the function which is to be minimized at the point (X_o,Y_o), is computed from

$$Z_o = f(X_o, Y_o). \qquad (4\text{-}1)$$

The function is now evaluated at two other points,

$$Z_1 = f(X_o + \Delta X, Y_o) \qquad (4\text{-}2)$$

and

$$Z_2 = f(X_o, Y_o + \Delta Y), \qquad (4\text{-}3)$$

where the values of ΔX and ΔY have been determined previously. The difference between the value of the function at the initial point and the other points is given by

$$D_1 = Z_o - Z_1 \qquad (4\text{-}4)$$

and

$$D_2 = Z_o - Z_2. \qquad (4\text{-}5)$$

The "preferred direction" is now expressed as the line formed by the points (X_k, Y_k),

$$(X_k,Y_k) = (X_o + I_1 k\Delta X, Y_o + I_2 k\Delta Y) \qquad (4\text{-}6)$$

where

$$I_1 = \frac{D_1}{\sqrt{D_1^2 + D_2^2}} \qquad (4\text{-}7)$$

$$I_2 = \frac{D_2}{\sqrt{D_1^2 + D_2^2}} \quad \text{(see Fig. D-1)} \qquad (4\text{-}8)$$

The development is easily extended to a function of n variables

$$Z = f(V_1, V_2, \ldots, V_n) \qquad (4\text{-}9)$$

where

$$(V_{1k}, V_{2k}, \ldots, V_{nk}) = (V_{1o} + I_1 k \Delta V_1,\ V_{2o} + I_2 k \Delta V_2, \ldots, V_{no} + I_n k \Delta V_n)$$

$$I_j = \frac{Z_o - Z_j}{\sqrt{(Z_o - Z_1)^2 + (Z_o - Z_2)^2 + \ldots + (Z_o - Z_n)^2}} \qquad (4\text{-}10)$$

and

$$Z_j = f(V_{1o},\ V_{2o}, \ldots, V_{jo} + \Delta V_j, \ldots, V_n). \qquad (4\text{-}11)$$

This procedure requires n evaluations of the function at an incremental distance from the initial point plus one evaluation of the function at the initial point.

5. PROCEDURE FOR THE "EXPANDING SEARCH"

Having determined a "preferred direction" it is now necessary to find the minimum value of the function along this line. The first step is to do an "expanding search," which is defined as follows.

Given the normalized increments I_1 and I_2 from the previous section, let

$$DX = I_1 \Delta X \quad \text{and} \quad DY = I_2 \Delta X. \tag{5-1}$$

Now define a sequence of points such that the distance between successive points doubles.

$$(X_i, Y_i) = [X_o + (2^i - 1)DX,\ Y_o + (2^i - 1)DY] \text{ for } i = 1,2,\ldots \tag{5-2}$$

The function is evaluated at successive points,

$$Z_i = f(X_i, Y_i) \tag{5-3}$$

until

$$Z_{i+1} \geqslant Z_i. \quad \text{(see Fig. D-2).} \tag{5-4}$$

6. PROCEDURE FOR THE "CONTRACTING SEARCH"

When no further minimization is possible by the expanding search, the last three points are used as end points for a half interval or contracting search. This procedure consist of evaluating the function at points such that the distance between successive points is halved. This procedure starts by working back from the last point used by the expanding search. The contracting search stops when the distance to the next point would be less than the initial increment of the expanding search.

Consider the case where the last three points of the expanding search are

$$(X_{i-1},Y_{i-1}),\ (X_i,Y_i),\ (X_{i+1},Y_{i+1}),$$

where

$$Z_i < Z_{i-1} \qquad (6\text{-}1)$$

$$Z_i < Z_{i+1} \qquad (6\text{-}2)$$

The next point evaluated would be at

$$(X_{i+2},Y_{i+2}) = [(X_{i+1} + X_i)/2,\ (Y_{i+1} + Y_i)/2]. \qquad (6\text{-}3)$$

Successive points would be closer to X_i unless a point is found where the value of the function, Z_j, is less than Z_i. In this case, successive points would get closer to Z_j. For either case, the procedure stops where the next point would be closer than the initial increment to the current minimum (see Fig. D-3).

7. LAGRANGIAN FIT TO FIND MINIMUM

The Lagrange interpolation formula is applied to the three best points found in either expanding or contracting search. The minimum of the resulting second-order equation is then easily calculated. The calculation proceeds as follows:

Let

$$Z_a = f(\mu_1) \tag{7-1}$$

$$Z_b = f(\mu_2) \tag{7-2}$$

$$Z_c = f(\mu_3) \tag{7-3}$$

Then, in general, the reduced Lagrangian polynomial will be of the form

$$Z = C_o + C_1\mu + C_2\mu \tag{7-4}$$

Now if

$$Z_b < Z_a, \; Z_b < Z_c \tag{7-5}$$

and

$$\mu_2 < \mu_1, \; \mu_2 < \mu_3, \tag{7-6}$$

then the solution of

$$\frac{dZ}{d\mu} = 0, \quad \mu_m = -\frac{1}{2}\frac{C_1}{C_2}, \tag{7-7}$$

will be an explicit expression for the μ, which is the best approximation of the minimum using only the best three points.

Let

$$T_1 = (\mu_3-\mu_2)(Z_a-Z_b) \tag{7-8}$$

$$T_2 = (\mu_1-\mu_2)(Z_c-Z_b), \tag{7-9}$$

then

$$\mu_m = -\frac{1}{2}\frac{C_1}{C_2} = \frac{(\mu_3+\mu_2)T_1-(\mu_1+\mu_2)T_2}{T_1-T_2} \tag{7-10}$$

This is easily generalized to functions of n variables, where

$$(\mu_1) = (V_1+\mu_1\Delta V_1,\ V_2+\mu_1\Delta V_2,\ldots,V_n+\mu_1\Delta V_n) \tag{7-11}$$

$$(\mu_2) = (V_1+\mu_2\Delta V_1,\ V_2+\mu_2\Delta V_2,\ldots,V_n+\mu_2\Delta V_n) \tag{7-12}$$

$$(\mu_3) = (V_1+\mu_3\Delta V_1,\ V_2+\mu_3\Delta V_2,\ldots,V_n+\mu_3\Delta V_n) \tag{7-13}$$

and where the minimum point along the preferred direction is

$$(V_1+\mu_m\Delta V_1,\ V_2+\mu_m\Delta V_2,\ldots,V_n+\mu_m\Delta V_n) \tag{7-14}$$

with μ_m as it is defined above.

8. WEIGHTING PROCEDURE

Assume an initial point was chosen, the preferred direction obtained, and the minimum along the preferred direction found. Since each parameter is divided into n equal intervals and each interval has a weight associated with it, the weights can be used to guide the random selection procedure to relatively unexplored areas. The weighting procedure consist of two steps.

(1) For each interval that each variable passes through along the line from the initial point to the local minimum, the corresponding weight is divided by a positive factor greater than unity. This will have the effect of reducing the probability of choosing an initial random point in regions already investigated.

(2) The weights must then be normalized such that the sum of the weights for each variable is 1. This is done by replacing W_{ij} by

$$W_{ij} \quad \sum_{j=1}^{n} W_{ij} \qquad \text{(see Fig. D-4).} \qquad (8\text{-}1)$$

9. SOLUTION CRITERIA

There are several criteria for determining when a suitable mainimum has been found. There are also several limits which terminate the process even if the expected solution criteria have not been met.

In some problems a sufficient condition for solution is any set of parameter values satisfying all constraints and having a function value less than some minimum. Each evaluation of the function is checked against this minimum, but processing does not stop until there is no further improvement along the current preferred direction.

In other problems it may be desirable to have a high degree of confidence that a good approximation to the minimum point has been found. By observing the behavior of the weights, two things can be noted. First, a very small weight means that the local minimum was found in the same range of a parameter on most of the searches. Since the small weight also means that there was a very small probability of a random point originating in this range, there is a large probability the true minimum is in this range. Second, a very small weight indicates many evaluations of the function in a range, thus giving a good chance that the local minimum in this range has been determined to within the specified accuracy. The requirement for having a high degree of confidence and a good approximation is that at least one region for each parameter is less than some specified value.

Since it is possible that neither of the previously mentioned conditions will be satisfied, a number of limits are placed on the amount of computation that may be done on a given problem. The most important is the limit on the total number of times the function may be evaluated. Limits are also placed on the number of random initial points that can be generated and the number of preferred directions that can be investigated.

If any limit is exceeded or a solution criteria satisfied computation on the problem ceases and the next set of data, if one exists, is processed.

10. ADAPTIVE FEATURE

The progress being made is continuously monitored by the program. The information thus obtained is used to modify a control constant which tends to adapt the program to the particular function being minimized.

Specifically, if the first step of the expanding search finds a value greater than the value at the initial point, it is assumed that the step size is too large, and the control constant which determines the size of the increment is reduced by a factor. On the other hand, if all steps, including the last, of the expanding search, yield successively lower values for the function, the size of the increment constant is increased by a factor. By adjusting the basic increment, the program is able to improve the efficiency of the procedures described earlier for the particular function being minimized.

11. CONTROL

The selection of the sequence of procedures is not an easy task. The behavior of the function at a limited number of points is known, but there is no sure way of predicting its over-all behavior. Therefore, care must be taken not to decide on a specific sequence of procedures which may be suited to only a few types of functions. The use of a weighted random selection provides a means by which the user can specify what the most probable selection should be. The program is still flexible enough to find a solution even if the most probable selection does not work well in a particular case.

12. DETAILS OF USING THE OPTIMIZATION PROGRAM

Full control of the program has been left to the user. Below is an explanation of what each control constant does and what value it is initially assigned by the program. Any constant given below can be set by the user as follows:

> Punch the name given below, an equal sign, and the value the program is to use for each constant that is to be changed. Separate these statements by commas if more than one is punched per card. Symbols, meanings, and initial values of control constants may be changed for each data set. The values remain the same as they were for the previous data set unless changed. Three dots (...) indicate these must be supplied with the first data set (see p. D-19).

The function which is to be minimized must be prepared as follows. The calculation of the function is programmed as subroutine or external function. The name of the subroutine is VALUE. The values of the parameters to be used in the calculation are in consecutive locations with the base location given the subroutines as a single argument. The user may choose any name for the argument. Assume the name PAR is used, then the parameters values would be in PAR(1), PAR(2),...,PAR(N). The result of the calculation based on the current values of the parameters is returned as the single value of the function.

If constraints other than the independent limits given by PMX and PMN are to be placed on the parameters, two additional subroutines must be prepared. These must be named MIN and MAX. The arguments for these functions are as follows:

PAR — The base location of all parameter values except the one for which the maximum or minimum is requested.

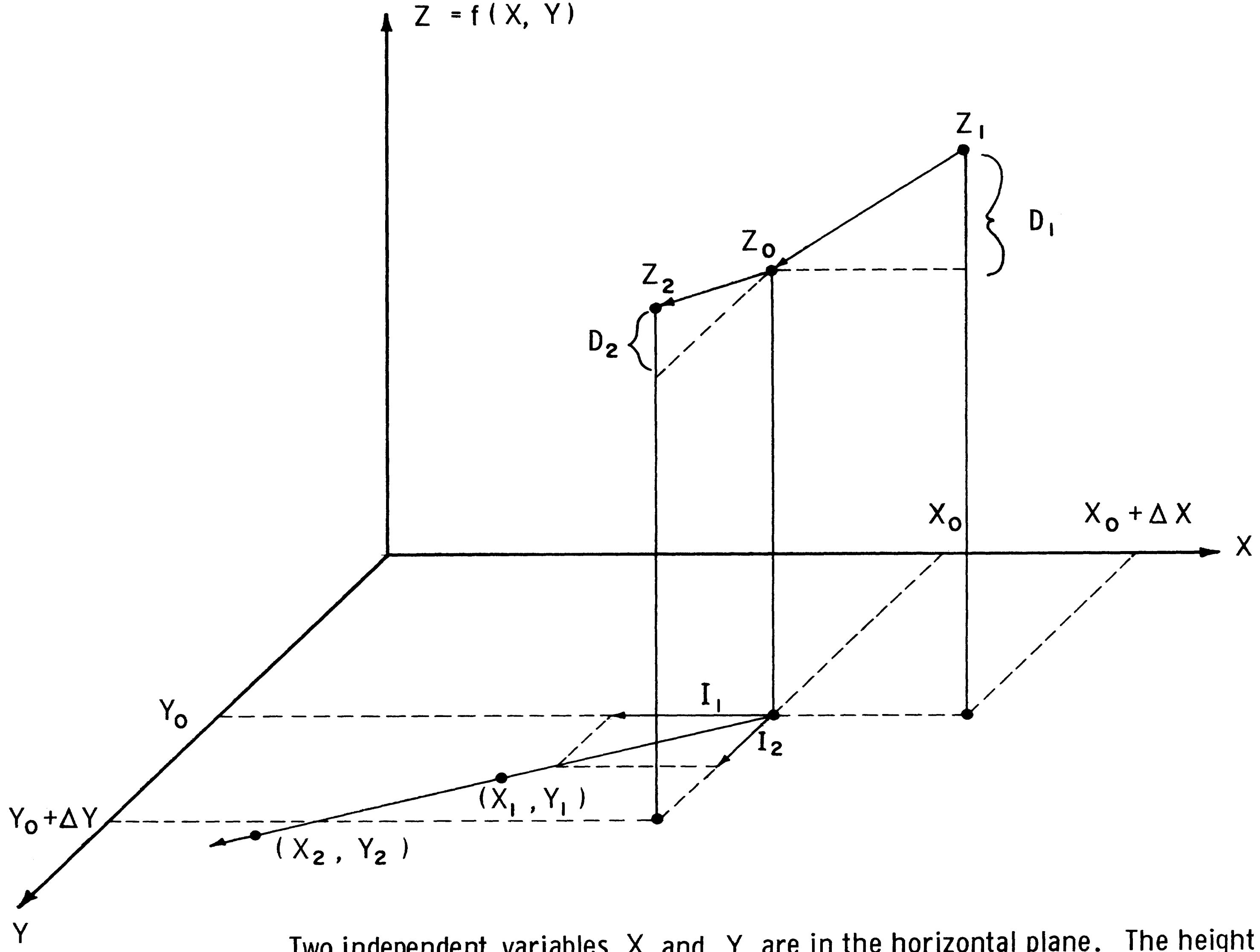
I — The index giving the number of the parameter whose limit is requested.

ST — The location to which control should be transferred if no possible limit exists.

PMX(I) or PMN(I) — The absolute limit which the parameter value must not exceed. The subroutine must return a limit which does not exceed this fixed limit.

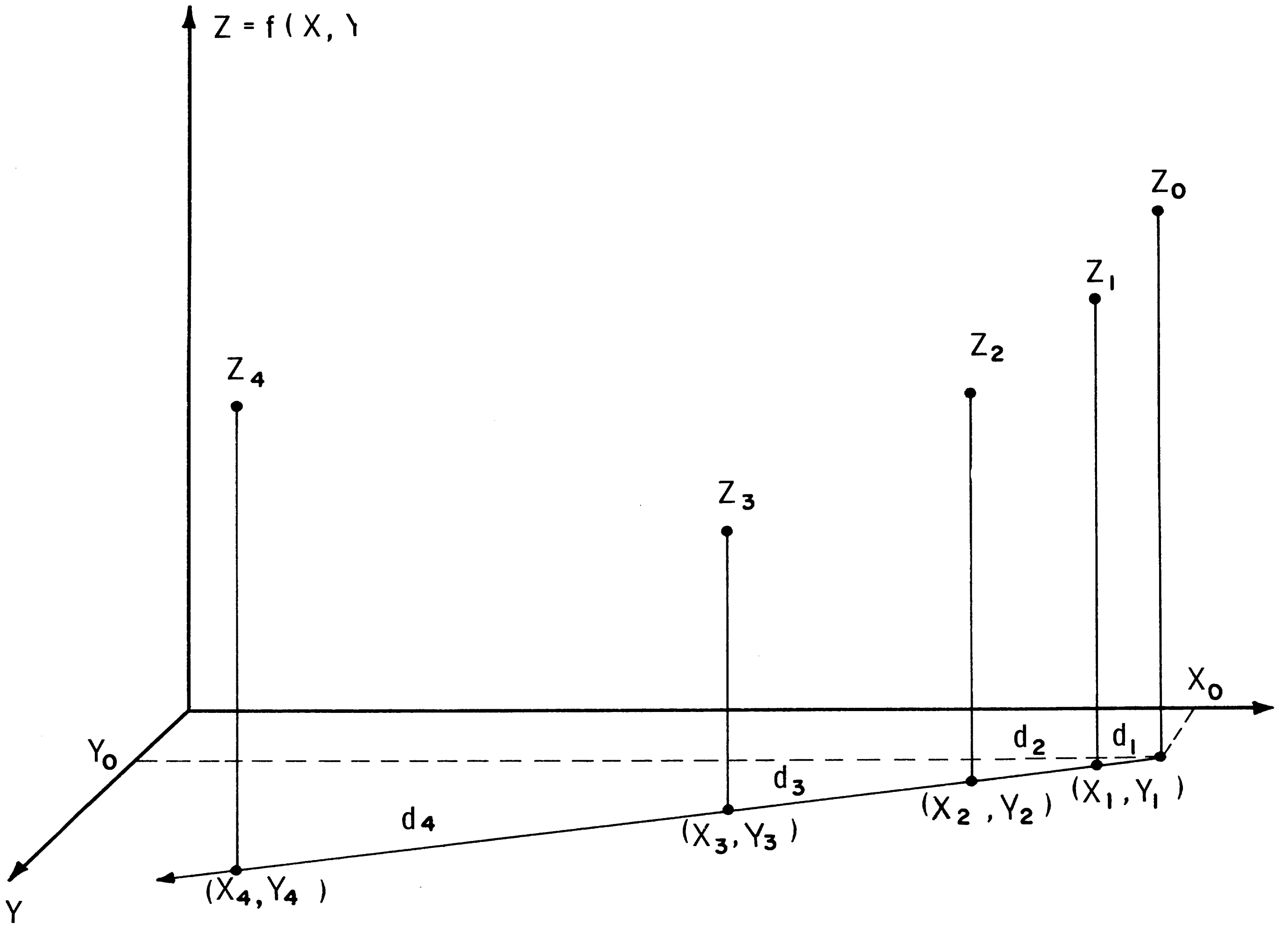
Again, both of these subroutines return a single value, which is the required limit.

Symbol	Initial Value	Definition
N	...	The number of parameters.
PMX(1)...PMX(N)	...	The maximum value of each parameter.
PMN(1)...PMN(N)	...	The minimum value of each parameter.
IFINIT	0B	If initial choice of parameter values are to be given IFINIT = 1B.
P(1)...P(N)	0	Initial choice of parameter values.
IFINC	0B	If initial choice of increments are to be given IFINC = 1B.
IP(1)...IP(N)	0	Initial choice of increments.
IFPRA	1B	Set to 0B if value of function and parameter values are not to be printed each time the function is evaluated.
KLIM	12	The maximum number of steps the expanding search may take before another preferred direction is found. Usually 6 $\leqslant$ KLIM $\leqslant$ 16.
BINC	.0001	The initial increment-to-parameter-range ratio. Usually .01 $\leqslant$ BINC $\leqslant$ .00001 and .1 $\leqslant$ 2KLIM* BINC $\leqslant$ 1.
MXINEX	50.	Maximum factor by which BINC can be increased.
MNINCN	.1	Maximum factor by which BINC can be decreased. The two preceeding constants limit the amount that the program can adapt itself to the function.
DRREP	3	The maximum numbers of successive preferred direction calculations before choosing a new random point.
MXF	200	The maximum number of evaluations of the function.
MXRND	10	The maximum number of random points that may be used as initial points.
MXDR	40	The maximum number of direction numbers that may be calculated.
LRNMN	10^{-10}	Solution condition if one weight of each parameter is less than this value. 0 $\leqslant$ LRNMN $\leqslant$ .01.
MINVAL	-10^{37}	Solution condition if any value of the function is less than this value.
DRMIN	.02	The minimum of the absolute value of the direction numbers. 0 $\leqslant$ DRMIN $\leqslant$ .1.
LRNFCT	5.	The amount by which the weights are divided during the weighting procedure. 1. < LRNFCT < 20.
LINCON	.5	The factor by which each increment is decreased when required by the adaptive procedure. .1 $\leqslant$ LINCOL < 1.
LINEXP	2.	The factor by which each increment is increased when required by the adaptive procedure. 1 < LINEXP $\leqslant$ 10.
PRBLP	.8	The probability of the last point being used as an initial point when a control decision is to be made. 0 < PRBLP < 1-PRBLP.
PRBRND	.1	The probability of a random point being used as an initial point when a control decision is to be made. 0 < PRBRN < 1-PRBLP.
PRBBP	.7	The probability of the best point found to date being used as an initial point when other procedures are not giving sufficient improvement or when various limits are exceeded. 0 < PRBBP < 1.



Two independent variables X and Y are in the horizontal plane. The height of a point directly above some coordinant (X, Y) is the valueof f (X,Y).

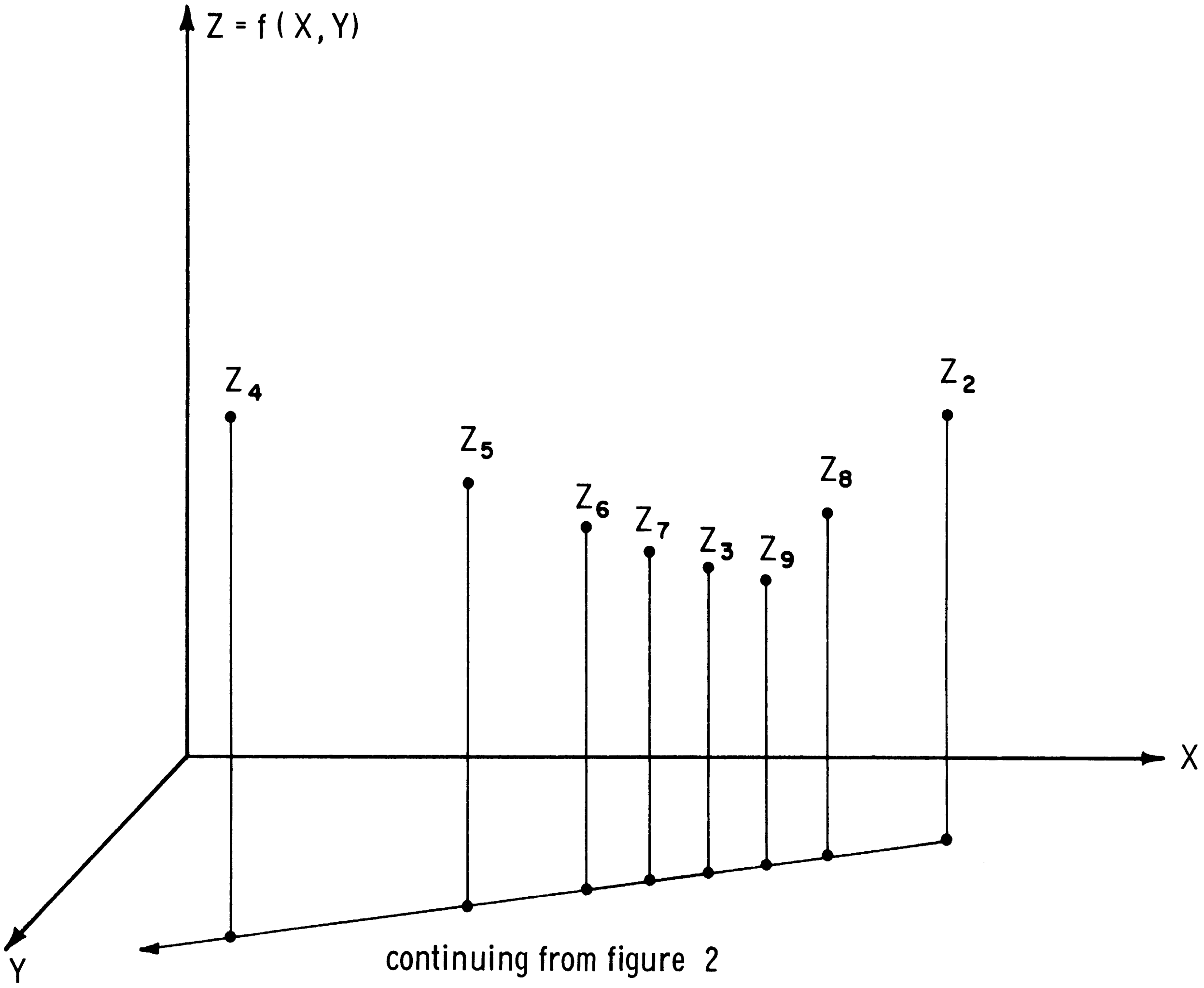
Fig. D-1.



$$d_2 = 2\,d_1$$
$$d_3 = 2\,d_2$$
$$\vdots$$
$$d_i = 2^{i-1}\,d_1$$

$$(X_1 , Y_1) = (X_0 + DX,\ Y_0 + DY)$$
$$(X_2 , Y_2) = (X_0 + 3\,DX,\ Y_0 + 3\,DY)$$
$$\vdots$$
$$(X_{i+1}, Y_{i+1}) = (X_0 + (2^{i+1} - 1)\,DX,\ Y_0 + (2^{i+1} - 1)\,DY)$$

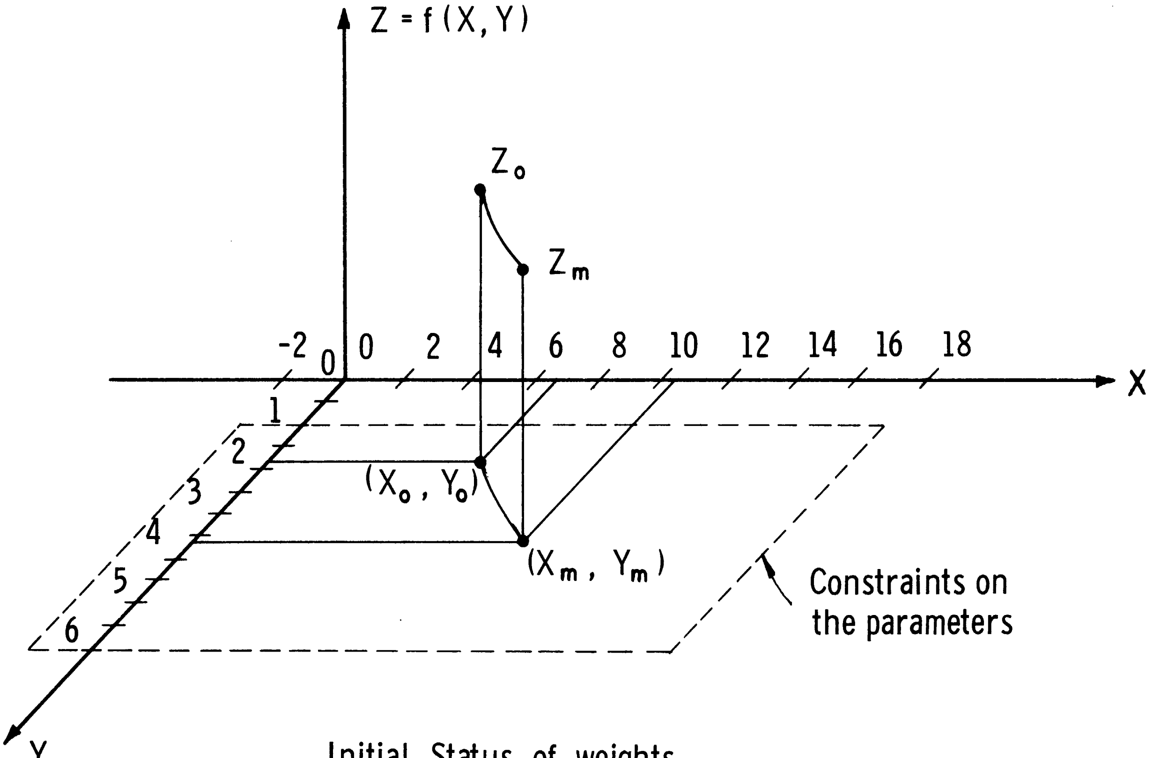
Fig. D-2.



continuing from figure 2

The points $(X_5, Y_5) \ldots (X_9, Y_9)$ are evaluated in the $Z_5, Z_6, \ldots, Z_9$. This is only one of the many possible cases. For another example consider $Z_5 < Z_6 < Z_3$. Here the point where Z_7 is to be evaluated lies between (X_5, Y_5) and (X_6, Y_6) Also Z_9 and Z_9 would not even be evaluated.

Fig. D-3.



Initial Status of weights

-2 0 2 4 6 8 10 12 14 16 18

X	.1	.1	.1	.1	.1	.1	.1	.1	.1	.1

1 1.5 2 2.5 3 3.5 4 4.5 5 5.5 6

Y	.1	.1	.1	.1	.1	.1	.1	.1	.1	.1

Modification of weights due to the search for a minimum from (X_o , Y_o) to (X_m , Y_m). Weighting factor = 5.

X	.1	.1	.1	.1	.02	.02	.1	.1	.1	.1
Y	.1	.02	.02	.02	.02	.02	.1	.1	.1	.1

Normalization to make the sum of the weights for each parameter equal to unity.

X	.119	.119	.119	.119	.024	.024	.119	.119	.119	.119
Y	.1667	.0333	.0333	.0333	.0333	.0333	.1667	.1667	.1667	.1667

Fig. D-4.

GENERAL FLOW DIAGRAM

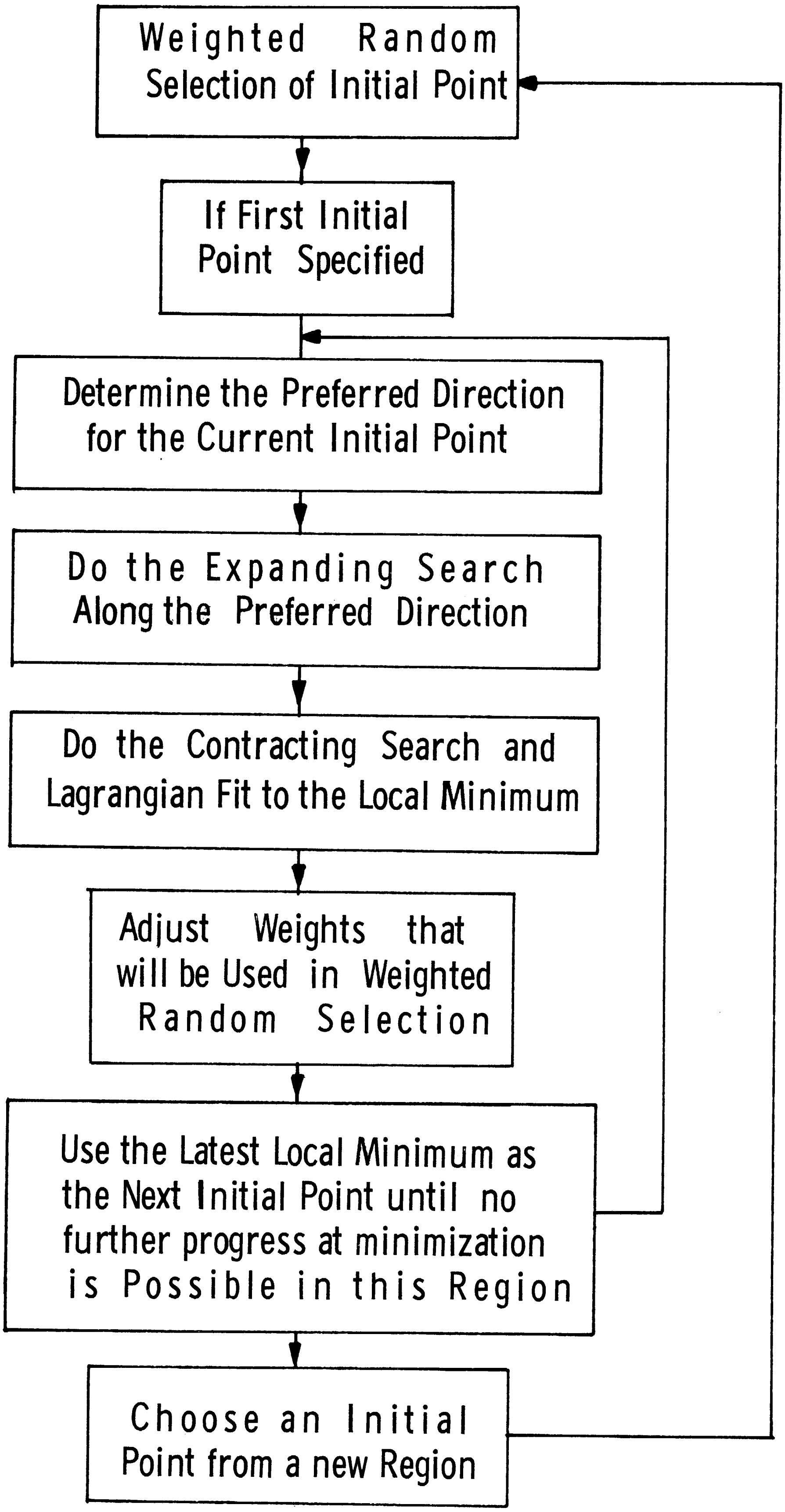

Weighted Random Selection of Initial Point
If First Initial Point Specified
Determine the Preferred Direction for the Current Initial Point
Do the Expanding Search Along the Preferred Direction
Do the Contracting Search and Lagrangian Fit to the Local Minimum
Adjust Weights that will be Used in Weighted Random Selection
Use the Latest Local Minimum as the Next Initial Point until no further progress at minimization is Possible in this Region
Choose an Initial Point from a new Region

Fig. D-5.

APPENDIX E

OUTLINE OF A TRAINING PROGRAM IN STATISTICAL METHODS

APPENDIX E. TABLE OF CONTENTS

1. BASIC TERMS AND FORMULAS

1.1 ESSENTIAL MATHEMATICS

For persons who need review the following text is recommended.

Helen M. Walker, Mathematics Essential for Elementary Statistics, Henry Holt and Co., New York, 1949.

1.2 STATISTICAL TERMS AND FORMULAS

n	number of observations
$X_1,\ldots,X_n,Y_1,\ldots,Y_n$	observed data
$\sum X = X_1+X_2+\ldots+X_n$	
$\sum X^2 = X_1^2+X_2^2+\ldots+X_n^2$	
$\sum XY = X_1Y_1+X_2Y_2+\ldots+X_nY_n$	
$\bar{X} = \sum X/n$	arithmetic mean
$x = X-\bar{X}$	deviation of observed data from mean
$\sum x^2 = \sum X^2-(\sum X)^2/n$	sum of squared deviations
$S = \sqrt{\sum x^2/n-1}$	standard deviation
$S' = \sqrt{\sum x^2/n}$	approximation used when n is large
$S^2 = \sum x^2/n-1$	variance
$D = \sum x^2/n$	mean (average)deviation
$b_y = \sum xy/\sum x^2$	regression coefficient of y on x
$b_x = \sum xy/\sum y^2$	regression coefficient of x on y
$\tilde{y} = b_y x$	predicting equation for y

$s_y = \sqrt{\sum(y-\tilde{y})^2/n-2}$ standard error of estimate

$r = \sum xy/\sqrt{\sum x^2 \sum y^2}$ correlation coefficient

$\binom{n}{k} = n!/(n-k)!k!$ the number of ways to pick k items from a set of n items

$n!$ the number of ways n items can be ordered

$P = 1/n$ the probability that any one of n equally likely events will occur

$P_n = 1-P_o$ the probability that an event will not happen if P_o is the probability the event will happen

$P_b = P_1 \cdot P_2$ the probability of two independent events happening is the product of the probabilities of each event happening individually

$P_\ell = P_1+P_2$ the probability of either of two mutually exclusive events occurring is the sum of the probabilities that each will occur

$\sum P = 1$ the sum of the probabilities of occurrance of all mutually exclusive events is always unity

$0 \leqslant P \leqslant 1$ a probability may take on only positive values in the range zero to one.

2. POPULATIONS, SAMPLES AND THEIR MEASURES

Based upon appropriate chapters in the following texts:

Allen L. Edwards, Statistical Analysis, Rinehart and Co., New York, 1946, Chapters 3, 4, 11, and 15.

Harold Cramer, The Elements of Probability, John Wiley and Sons Inc., New York, 1955, Chapters 13, 14, and 15.

3. DISTRIBUTIONS

Based upon appropriate chapters in the above texts.

3.1 BINOMIAL DISTRIBUTIONS

Edwards, Chapter 8
Cramer, Chapter 6

3.2 POISSON DISTRIBUTIONS

Cramer, Chapter 6

3.3 GAUSSIAN OR NORMAL DISTRIBUTIONS

Edwards, Chapter 10
Cramer, Chapter 7

4. CORRELATION

Based upon appropriate chapters in the above texts.

Edwards, Chapter 6
Cramer, Chapter 10

5. REGRESSION

Based upon appropriate chapters in the above texts.

Edwards, Chapter 6
Cramer, Chapter 10

Additional material from the following texts:

K. A. Brownlee, Industrial Experimentation, Chemical Publishing Co., Cleveland, Ohio, 1953.

Harry H. Goode and Robert E. Machol, System Engineering, McGraw-Hill Book Co., New York, 1957.

6. TRANSLATING PROBLEMS FOR THE COMPUTER

6.1 SAMPLE PROBLEM IN LINEAR CURVE FITTING

GIVEN Experimental data:

x,y,z independent variables or functions of independent variables, i.e., $y = x^3$, $z = x^{1/2}$, where x is independent

w dependent variable

Where w is measured for many values of x,y, and z

COMPUTE The values of a,b,c, and d

such that $w^* = a*x+b*y+c*z+d$, predicts w as least squares fit.

S = sum of squares of differences between actual value of w and predicted value of w (w^*).

Let

$$S = \sum_{i=1}^{n} \{w_i - w_i^*\}^2$$

or

$$S = \sum_{i=1}^{n} \{w_i - a*x_i - b*y_i - c*z_i - d*1_i\}^2$$

By least-squares we mean the choice of a,b,c and d which makes S a minimum.

Note: x,y,z and w are known (data)
i indicates the i^{th} observation of x,y,z and w
n is the number of observations
a,b,c and d are to be determined

To find the minimum of S: Take derivatives of S with respect to a,b,c, and d. Set the derivatives equal to zero and solve for a,b,c, and d. These are the values of a,b,c, and d for which S is a minimum.

Thus,

$$\frac{\partial S}{\partial a} = \sum_{i=1}^{n} - 2x_i\{w_i-ax_i-by_i-cz_i-dl_i\} = 0 \tag{E-1}$$

$$\frac{\partial S}{\partial b} = \sum_{i=1}^{n} - 2y_i\{w_i-ax_i-by_i-cz_i-dl_i\} = 0 \tag{E-2}$$

$$\frac{\partial S}{\partial c} = \sum_{i=1}^{n} - 2z_i\{w_i-ax_i-by_i-cz_i-dl_i\} = 0 \tag{E-3}$$

$$\frac{\partial S}{\partial d} = \sum_{i=1}^{n} - 2\{w_i-ax_i-by_i-cz_i-dl_i\} = 0 \tag{E-4}$$

We multiply both sides of all equations by -1/2, therefore drop the -2 in each equation. Also, it is convenient to write Eq. (E-4) first.

Thus,

$$\sum_{i=1}^{n} \{ax_i+by_i+cz_i+dl_i-w_i\} = 0 \tag{E-5}$$

$$\sum_{i=1}^{n} x_i\{ax_i+by_i+cz_i+dl_i-w_i\} = 0 \tag{E-6}$$

$$\sum_{i=1}^{n} y_i\{ax_i+by_i+cz_i+dl_i-w_i\} = 0 \tag{E-7}$$

$$\sum_{i=1}^{n} z_i\{ax_i+by_i+cz_i+dl_i-w_i\} = 0 \tag{E-8}$$

Now, do the indicated multiplication and place the summation sign "Σ" with the individual terms to be summed. The indices will not be written for brevity.

$$a\Sigma x_i+b\Sigma y_i+c\Sigma z_i+d\Sigma l_i-\Sigma w_i = 0 \tag{E-9}$$

$$a\Sigma x_i^2+b\Sigma x_iy_i+c\Sigma x_iz_i+d\Sigma x_i-\Sigma x_iw_i = 0 \tag{E-10}$$

$$a\Sigma x_iy_i+b\Sigma y_i^2+c\Sigma y_iz_i+d\Sigma y_i-\Sigma y_iw_i = 0 \tag{E-11}$$

$$a\Sigma x_iz_i+b\Sigma z_iy_i+c\Sigma z_i^2+d\Sigma z_i-\Sigma z_iw_i = 0 \tag{E-12}$$

By writing the terms with d first, and adding terms with w to both sides of each equation, we obtain a system of simultaneous linear equations which completely determine a,b,c, and d. (The least-squares coefficients.)

$$d\sum 1_i+a\sum x_i+b\sum y_i+c\sum z_i \quad = \quad \sum w_i \qquad (E\text{-}13)$$

$$d\sum x_i+a\sum x_i^2+b\sum x_iy_i+c\sum x_iz_i \quad = \quad \sum x_iw_i \qquad (E\text{-}14)$$

$$d\sum y_i+a\sum x_iy_i+b\sum y_i^2+c\sum y_iz_i \quad = \quad \sum y\ w \qquad (E\text{-}15)$$

$$d\sum z_i+a\sum x_iz_i+b\sum y_iz_i+c\sum z_i^2 \quad = \quad \sum z_iw_i \qquad (E\text{-}16)$$

6.2 SAMPLE PROBLEM IN MAD LANGUAGE

Reference - B. Arden, B. Galler, and R. Graham, Michigan Algorithm Decoder, The University of Michigan Computing Center, Ann Arbor, Mich., 1961.

```
$   COMPILE MAD, EXECUTE, DUMP
          R
          R   PROGRAM TO SET UP MATRIX FOR CURVE FITTING
          R
NEXT       READ FORMAT F1,N
           DIM(2) = N+1
           DIM(1) = N+2
           READ FORMAT F2,F3(1)...F3(12)
           THROUGH SET, FOR I = 0,1,I.E.(N+1)*(N+1)
SET        A(I) = 0.
READ       READ FORMAT F3(1),X(0)...X(N)
           WHENEVER X(0).L.0., TRANSFER TO PROCES
           PRINT FORMAT F3,X(0)...X(N)
           X(0) = 1.
           THROUGH L1, FOR I = 0,.,I.G.N
           THROUGH L1, FOR J = 0,1,J.G.I
L1         A(I,J) = A(I,J)+X(I)*X(J)
           TRANSFER TO READ
PROCES     THROUGH L2, FOR I = 0,1,I.G.M
           THROUGH L2, FOR J = 0,1,J.E.I
L2         A(J,I) = A(I,J)
          R
          R   NOW SOLVE SYSTEM OF EQUATIONS
          R
           DIMENSION A(2500,DIM), X(50), F3(12), C(50)
           VECTOR VALUES DIM = 2,0,0
           VECTOR VALUES F3 = $1H, $
           INTEGER I,J,N
          R
           EXECUTE SIMSLN.(A,N,C)
```

```
            R
             PRINT FORMAT F4, C(0)...C(N-1)
             TRANSFER TO NEXT
             VECTOR VALUES F1 = $I10*$
             VECTOR VALUES F2 = $12C6*$
             VECTOR VALUES F4 = $1H0, 1P5E20.7*$
             END OF PROGRAM
```

6.3 SUBROUTINE SIMSLN USED BY PROGRAM

```
$   COMPILE MAD, PUNCH OBJECT
             EXTERNAL FUNCTION (A,N,S)
             ENTRY TO SIMSLN.
             THROUGH SET, FOR K = 0,1,K.E.N
SET          V(K) = K
             ZC = 0
             THROUGH LP1, FOR K = 0,1,K.E.N.
             WHENEVER A(V(K),K).E.0.
             THROUGH LP2, FOR I = K+1,1,I.E.N
             WHENEVER A(V(I),K).NE.0.
             J = V(I)
             V(I) = V(K)
             V(K) = J
             TRANSFER TO ON
             END OF CONDITIONAL
LP2          CONTINUE
             THROUGH LP3, FOR J = K+1,1,J.G.N
LP3          A(V(K),J) = 0.
             ZC = ZC+1
             TRANSFER TO LP1
             END OF CONDITIONAL
ON           THROUGH LP4, FOR J = N,-1,J.L.K
LP4          A(V(K),J) = A(V(K),J)/A(V(K),K)
             THROUGH LP1, FOR I = 0,1,I.E.N
             WHENEVER I.E.V(K), TRANSFER TO LP1
             THROUGH LP1, FOR J = N,-1,J.E.K
             A(I,J) = A(I,J) - A(I,K)*A(V(K),J)
LP1          CONTINUE
             THROUGH RESULT, FOR I = 0,1,I.E.N
RESULT       S(V(I)) = A(I,N)
             FUNCTION RETURN ZC
             INTEGER I,J,K,V,ZC,N
             DIMENSION V(40)
             END OF FUNCTION
```

6.4 EXAMPLE OF COMPUTATION PERFORMED BY SUBROUTINE SIMSLN

GIVEN:

$$2y_1+4y_2+4y_3+4y_4 = 4$$

$$0y_1+1y_2+3y_3+3y_4 = 0$$

$$0y_1+0y_2+5y_3+5y_4 = 0$$

$$2y_1+4y_2+2y_3+4y_4 = 2$$

Solve for y_1, y_2, y_3, and y_4 by Jordan elimination

2	4	4	4	4	Initial			
0	1	3	3	0				
0	0	5	5	0				
2	4	2	4	2				

1	2	2	2	2	÷2	k=1		
0	1	3	3	0			I=2	$a_{k,j} = a_{k,j}/a_{kk}$
0	0	5	5	0			I=3	
0	0	-2	0	-2			I=4	$a_{i,j}=a_{i,j}-a_{i,k}*a_{k,j}$ j=5,4,3,2,1

1	0	-4	-4	2			I=1	
0	1	3	3	0	÷1	k=2		
0	0	5	5	0			I=3	
0	0	-2	0	-2			I=4	j=5,4,3,2

1	0	0	0	2			I=1	
0	1	0	0	0			I=2	
0	0	1	1	0	÷5	k=3		
0	0	0	2	-2			I=4	j=5,4,3

1	0	0	0	2			I=1	
0	1	0	0	0			I=2	
0	0	1	0	1			I=3	
0	0	0	1	-1	÷2	k=4		j=5,4

↑ SOLUTION $y_1=2$, $y_2=0$, $y_3=1$, and $y_4=-1$