Dynamic Simulation of Inverse-Time Overcurrent Relays

by

Tim Yuen Eng

A thesis
presented to the University of Manitoba
in partial fulfillment of the
requirements for the degree of
Master of Science
in
The Department of Electrical Engineering

Winnipeg, Manitoba

DYNAMIC SIMULATION OF INVERSE-TIME OVERCURRENT RELAYS

BY

TIM YUEN ENG

A thesis submitted to the Faculty of Graduate Studies of the University of Manitoba in partial fulfillment of the requirements of the degree of

MASTER OF SCIENCE

I hereby declare that I am the sole author of this thesis.

I authorize the University of Manitoba to lend this thesis to other institutions or individuals for the purpose of scholarly research.

Tim Yuen Eng

I further authorize the University of Manitoba to reproduce this thesis by photocopying or by other means, in total or in part, at the request of other institutions or individuals for the purpose of scholarly research.

Tim Yuen Eng

ABSTRACT

A new concept of describing the characteristic of inverse-time overcurrent relays is introduced in this thesis. The motion of a rotating induction disc unit can be represented accurately as a function of the relay time dial setting and multiple of pickup current. Such dynamic characteristics are developed based on the time-current characteristic curves published by relay manufacturers.

Piece-wise linearization and searching for suitable non-linear continuous models are the two approaches recommended to implement the relay dynamic curves. Optimization techniques were employed to accomplish these goals. Results of the study are found to be satisfactory indicating that both of these recommended approaches are feasible.

Three simple trip simulations are also presented as application test cases. The test cases show that the concept of simulating the dynamic behavior of overcurrent relays is a very practical one.

ACKNOWLEDGEMENTS

I wish to express my sincere gratitude to professor G.W. Swift, my advisor, for his guidance and encouragement.

The financial support by the National Science and Engineering Research council of Canada, through a Project Research Assistance to Industry (PRAI) grant and also through the University of Manitoba in the form of university fellowship are appreciated.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

Chapter I

INTRODUCTION

In recent years, computer-aided-design (CAD) approach has been used by electric power system engineers to solve their problems. One of the efforts toward applying computers to the protection engineers' problems has been oriented toward components of the problem such as representing the time delay characteristics of inverse-time overcurrent relays.

An inverse-time overcurrent relay is a device which detects the current flow in one phase of an electric power system component and generates a trip signal after a delay that is short for heavy fault currents and long for lighter fault currents.

The inverse-time characteristic is adjustable by selecting a suitable time dial setting (TDS) such that relays may coordinate with each other, as shown in figure 1.1

Figure 1.2 shows a typical family of time-current curves for a particular overcurrent relay, namely CO-8. Theoretically, their time ordinates should be proportional to the time dial setting (contact travel) so that, if the times for a given current were divided by the time dial setting, all the curves should be coincident. Unfortunately, the inertia

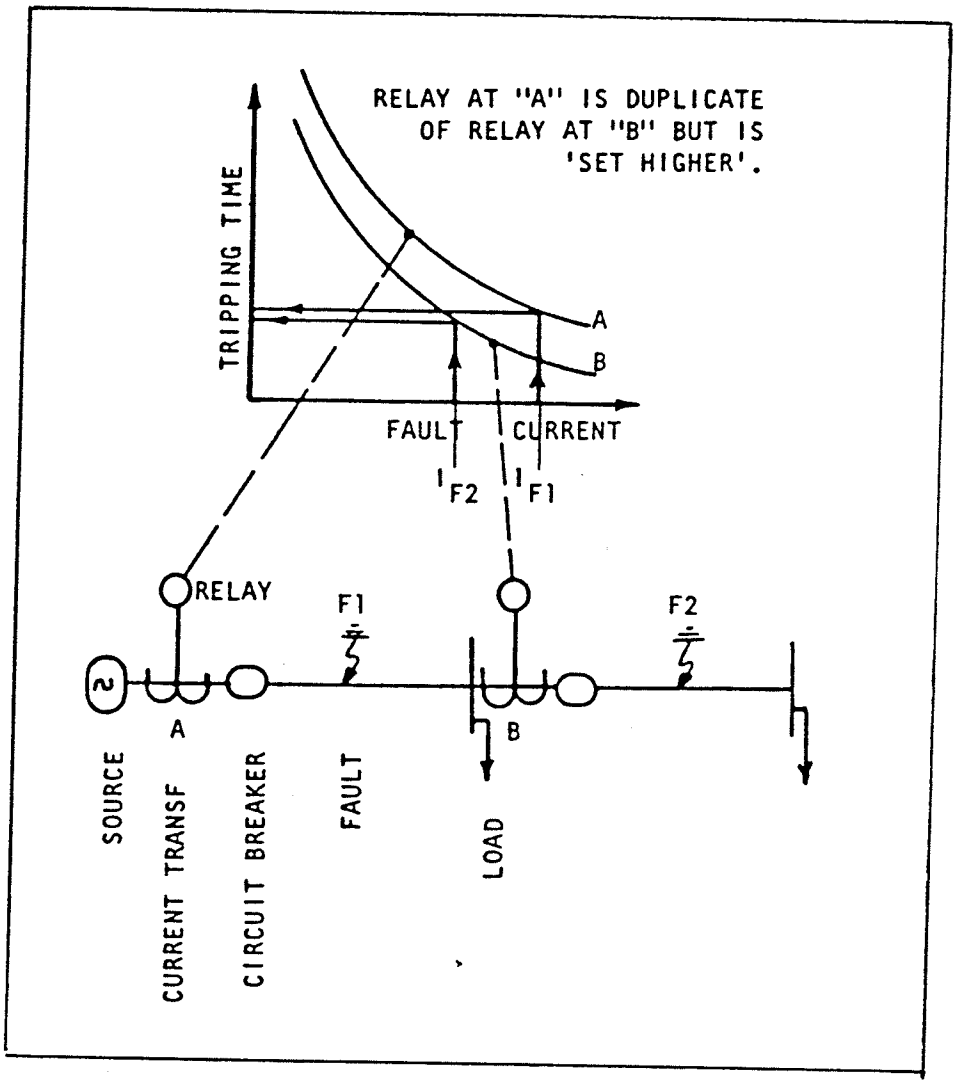


Figure 1.1: Relay Coordination .

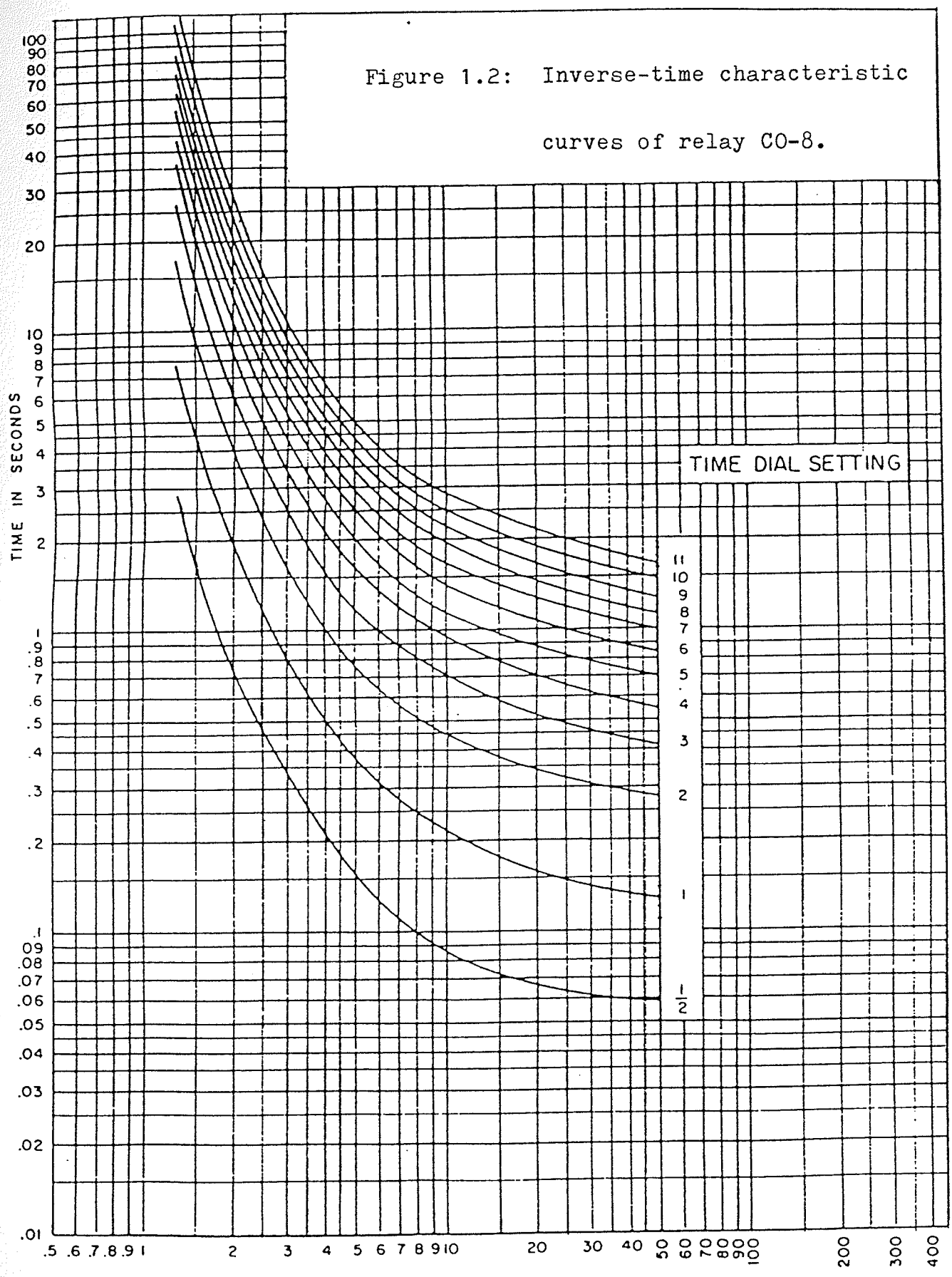


Figure 1.2: Inverse-time characteristic curves of relay CO-8.

of the disc makes this impossible at low current values because it takes a little time for the disc to accelerate from standstill to its steady speed. This is taken care of by publishing a family of curves such as is shown in figure 1.2

To represent this family of curves mathematically, several mathematical models have been proposed in the past. Some of the typical ones are those suggested by Warrington, Radke and Sachdev.

A.R.Van C. Warrington [1] proposed the following equation for expressing the contact closing time of an inverse-time overcurrent relay as a function of the operating current.

$$T = C + \frac{K \cdot D}{M^n + 1}$$

where : T is the operating time in seconds.

C is a constant which accounts for the effect of friction and hysteresis of the magnetic circuit.

K is a design constant.

D is the time dial setting. (Note : D=TDS)

M is the multiple of pick up current.

n is a constant for the relay.

The approach suggested by G.E. Radke [2] is based on the method of templates which is very popular with relay engi-

neers. Assuming that for different TDS the relay characteristic are identical and are displaced only vertically, the characteristic of each TDS is modelled by assigning a suitable value to the constant a_0 in the following equation.

$$\log(T-DC) = a_0 + a_1(\log M) + a_2(\log M)^2 + a_3(\log M)^3 + a_4(\log M)^4$$

where : a_p's are constants.

DC is a delay term included to improve the accuracy of the equation.

Noting the fact that the time-current characteristics of the overcurrent relays are asymtotic to the minimum pickup current. Sachdev introduced the following equation [3].

$$T = \left[a_0 + \frac{a_1}{(I-1)} + \frac{a_2}{(I-1)^2} + \frac{a_3}{(I-1)^3} + \frac{a_4}{(I-1)^4} \right] \cdot F$$

where : $F = b_0 + b_1(TDS) + b_2(TDS)^2 + b_3(TDS)^3$

In Sachdev's model, the a_p's are calculated for the maximum TDS, and then a scaling factor (F) which represents the relationship between the operating time at different TDS and the operating time at the maximum TDS is introduced. F is

expressed as a function of TDS. The product of F and the polynomial, which contains coefficients a_p ,will give the time-current characteristic of a given time dial setting.

The models proposed in the past have emphasized the time-current characteristic of the overcurrent relays. Although they do represent the time-current curves quite faithfully, there are three serious drawbacks. Firstly, when a time-current model is incorporated in a large computer program for system simulation study, the model cannot produce a correct relay operating time if the fault current amplitude changes before the relay trips, because the model tells nothing about the change of angular velocity of the induction disc unit when the fault current fluctuates. Secondly, the time-current models cannot predict a correct relay operating time in the case of circuit breaker reclosure, that is because the models cannot provide information regarding the angle traversed by the disc unit at the time of reclosure. The third deficiency is that it is not suitable for application in implementing a microprocessor based overcurrent relay. That is due to the fact that the models are too mathematically complicated and that it will take a long time for a microprocessor to do calculation with such models.

The objective of this thesis is to propose a new approach to represent the characteristics of overcurrent relays such that the above mentioned drawbacks can be eliminated. Precisely speaking, it is desirable to express the rotating an-

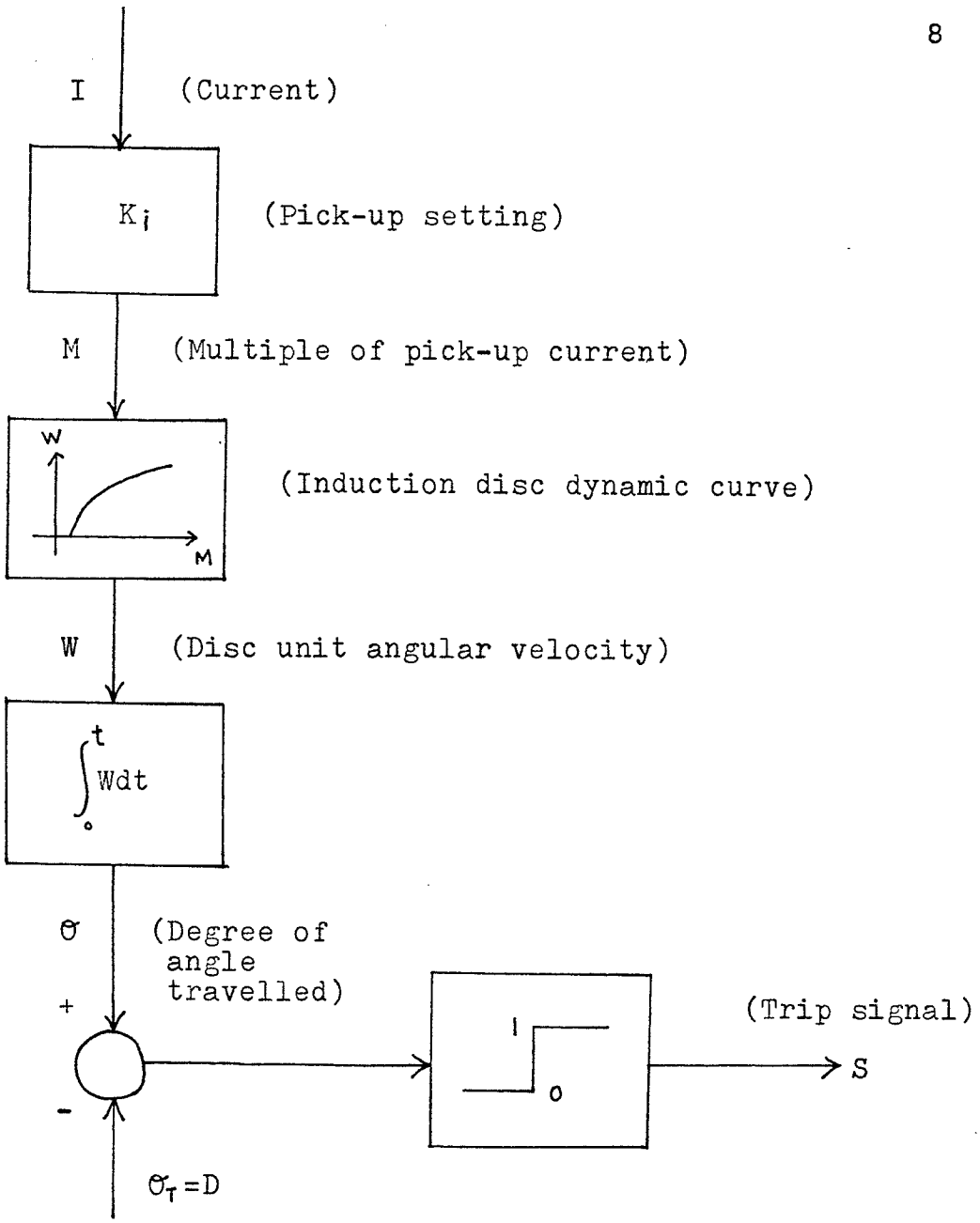
gular velocity of the relay disc mathematically and accurately for all current levels.

Let W be the relay disc angular velocity. Then the desired model will represent the relationship between W and the multiple of relay pickup current (M). Because the time-current characteristics are expressed as a family of non-linear curves with each curve associated with a different TDS, one would expect to see that the W-current characteristics of the overcurrent relays can also be expressed as a family of curves and these curves are also non-linear. We shall call these curves the dynamic curves of inverse-time overcurrent relays.

Once new models representing the dynamic curves are available and tested to be satisfactory, the models can be used to simulate relay operations. Figure 1.3 illustrates the concept of this simulation.

Four types of commercial inverse-time overcurrent relays, specifically the CO-8, CO-11, IAC51 and IAC77 will be investigated. Different models representing the dynamic curves of these relays will be introduced and methods of developing these models will be discussed in this thesis.

In addition to figure 1.2, The manufacturer's inverse-time characteristic curves of CO-11, IAC51 and IAC77 are also given in appendix N for reference.



where : θ_T is the the angle at which the relay will trip and is set to be equal to the TDS, D.

Figure 1.3: Overcurrent simulation using a dynamic curve.

Chapter II

THE DYNAMIC CURVES

2.1 THE DEVELOPMENT OF DYNAMIC CURVES

The approach of developing the overcurrent relay dynamic curves is to work backward from the existing time-current characteristic curves published by the manufacturers.

Assuming the angular velocity (W) of relay disc is a function of the multiple of pickup current (M) and TDS (D). Thus,

$$W = f(M,D) \qquad \text{(eq. 2.1)}$$

Note that W has a slight dependence on the disc angle (θ) because the reset spring reverse torque increases slightly with increasing θ. However, the forward torque due to the current is assumed to be much larger than this reverse torque.

A simplified induction-disc relay is illustrated in figure 2.1 showing the interaction between the reverse torque and forward torque.

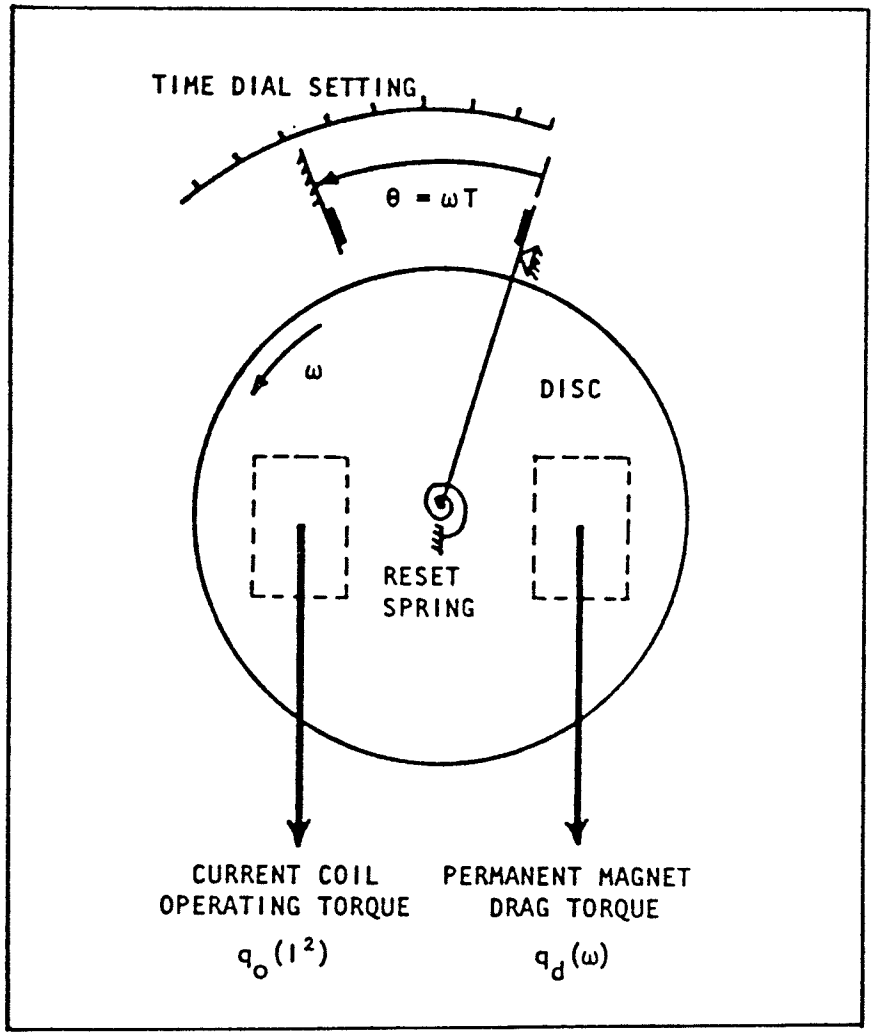


Figure 2.1: Induction-disc relay.

Referring to figure 1.3, Trip signal (S=1) will be generated when the integral of W reaches the TDS (D), i.e.

$$S=1 \quad \text{if} \quad \int_0^T W\,dt \geqslant D \qquad \text{(Operated State)}$$

$$S=0 \quad \text{if} \quad \int_0^T W\,dt < D \qquad \text{(Unoperated State)}$$

Referring to figure 1.2, the operating time of an overcurrent relay is a function of M as well as the TDS (D). One can work backward from

$$T = g(M,D) \qquad \text{(eq. 2.2)}$$

to get equation 2.1.

Let critical trip condition be :

$$\int_0^T W\,dt = D \qquad \text{(eq. 2.3)}$$

For constant W (a sufficient condition),

$$\int_0^T W\,dt = W \cdot T = D \qquad \text{(eq. 2.4)}$$

Therefore we have the following relationship for T and W :

$$W = D/T = f(M,D) \qquad \text{(eq. 2.5)}$$

Whether the disc angular velocity (W) is a function of TDS (D) or not depends on the characteristic of the time-current curves of the relay. For example, for an IEC type C relay,

$$T = \frac{80}{M^2-1} D \qquad \text{(eq. 2.6)}$$

i.e. at a fixed value of current (M), the time ordinate is always proportional to the time dial setting (D),

$$T \propto D$$

now, if we substitue equation 2.6 into equation 2.5,

$$W = \frac{M^2-1}{80 \cdot D} D = \frac{M^2-1}{80} = f(M)$$

and W is not a function of D.

However, for real electromechanical relays, the relay operating times are not quite proportional to the TDS (D) for a given value of current, and W will be a function of both M and D. An example is the CO-8 relay. Referring back to figure 1.2, if the operating times for a given current are divided by the time dial settings, the curves do not coincide. Therefore, T is not proportional to D and it implies that W is a function of both M and D.

To get an idea of the shape of a typical dynamic curve, let us use the CO-8 as our example and apply equation 2.5 to convert one of the curves shown in figure 1.2 into a dynamic curve, i.e. a plot of W versus M. such curve is shown in figure 2.2. Notice that the TDS (D) is 6 for this curve.

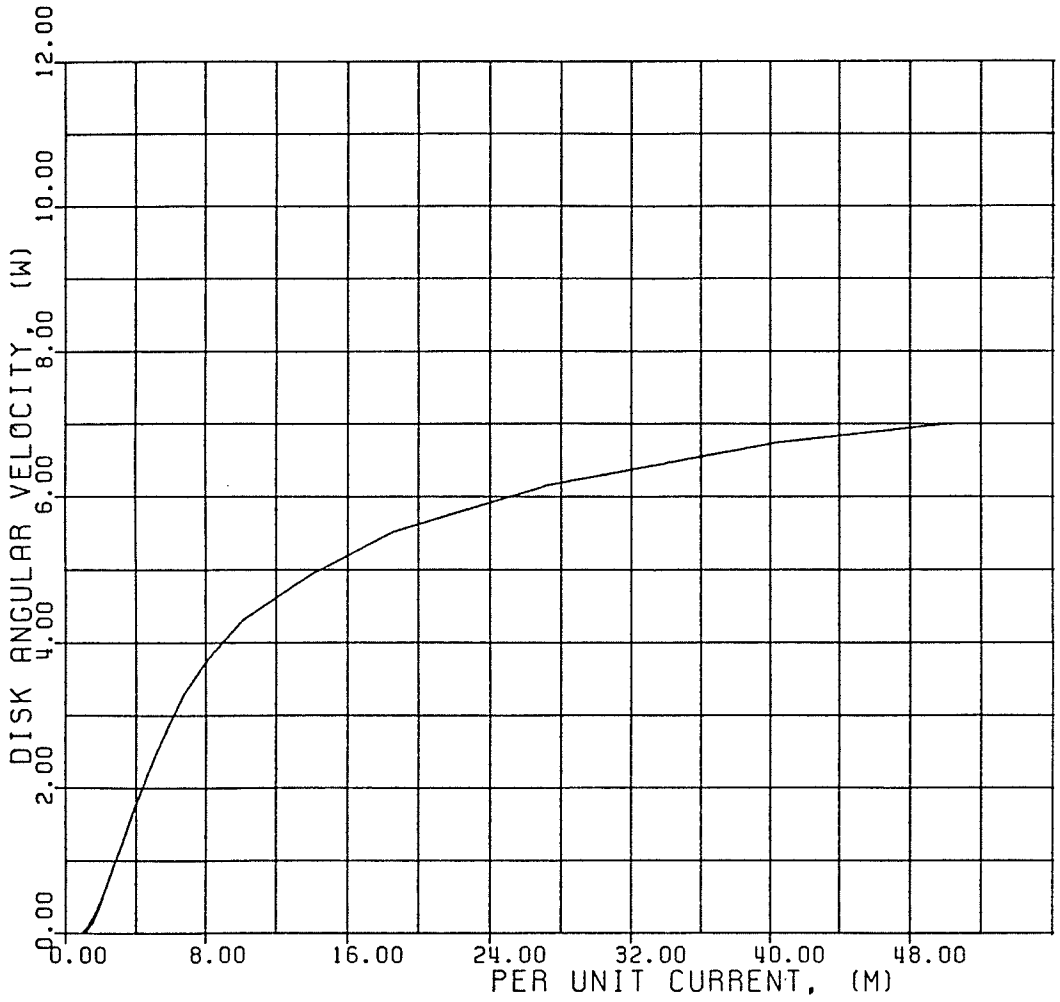


Figure 2.2: The dynamic curve for D=6 of CO-8 overcurrent relay.

2.2 METHODS OF IMPLEMENTING THE DYNAMIC CURVES

It is desirable that the dynamic curves are suitable for application in microprocessor based relay designs as well as for main frame computer relaying system study. For application in digital relay design, the mathematical model representing the dynamic curves has to be simple and reasonably accurate. One possible way to describe a non- linear curve in the simplest mathematical terms is to express the curve in terms of straight line equations. The method for implementing the dynamic curves using this idea is called the piece-wise linear approximation, and it will be pursued further in the next chapter.

If the dynamic curve model is to be used in main frame computer relaying system studies, then a more complex mathematical model might be acceptable. However, the degree of accuracy is also expected to be higher. This will be the second approach adopted to implement the dynamic curves and a relatively more complex non-linear model will be searched for.

The third possibility of representing the dynamic curves is the straight forward table-look-up method. The idea is to form a large table, which consists of enough data to represent the dynamic curves for all the time dial settings of a particular relay.

In the following two chapters, effort will be spent on implementing the dynamic curves by using the piece-wise linear approximation and the non-linear continuous model approach. The table-look-up method involves little calculation in finding the disc angular velocity (W), but it takes a large memory space to store all the data. It is simple to implement but the cost is high in terms of computer memory capacity; therefore it will not be investigated further.

2.3 ACCURACY CRITERIA FOR DYNAMIC CURVES IMPLEMENTATION

The suggested accuracy criteria are based on the International Electromechanical commission (IEC) standard "class index" 5, which implies the tolerance listed in table 2.1 .

M	2	5	10	20
Maximum allowable %error in T	±12.5%	±7.5%	±5%	±5%

TABLE 2.1

Maximum allowable %error in relay operating time (T).

The region where the multiple of current (M) is less than 2 is considered to be a region of uncertainty. In this region no accuracy criteria are applicable. The ±12.5% tolerance at M=2 is supposed to extend up to M<5, ± 7.5% is applicable between M=5 and M<10, and 5% for all M greater than and equal to 10. The distribution of such criteria is illustrated more clearly in figure 2.3.

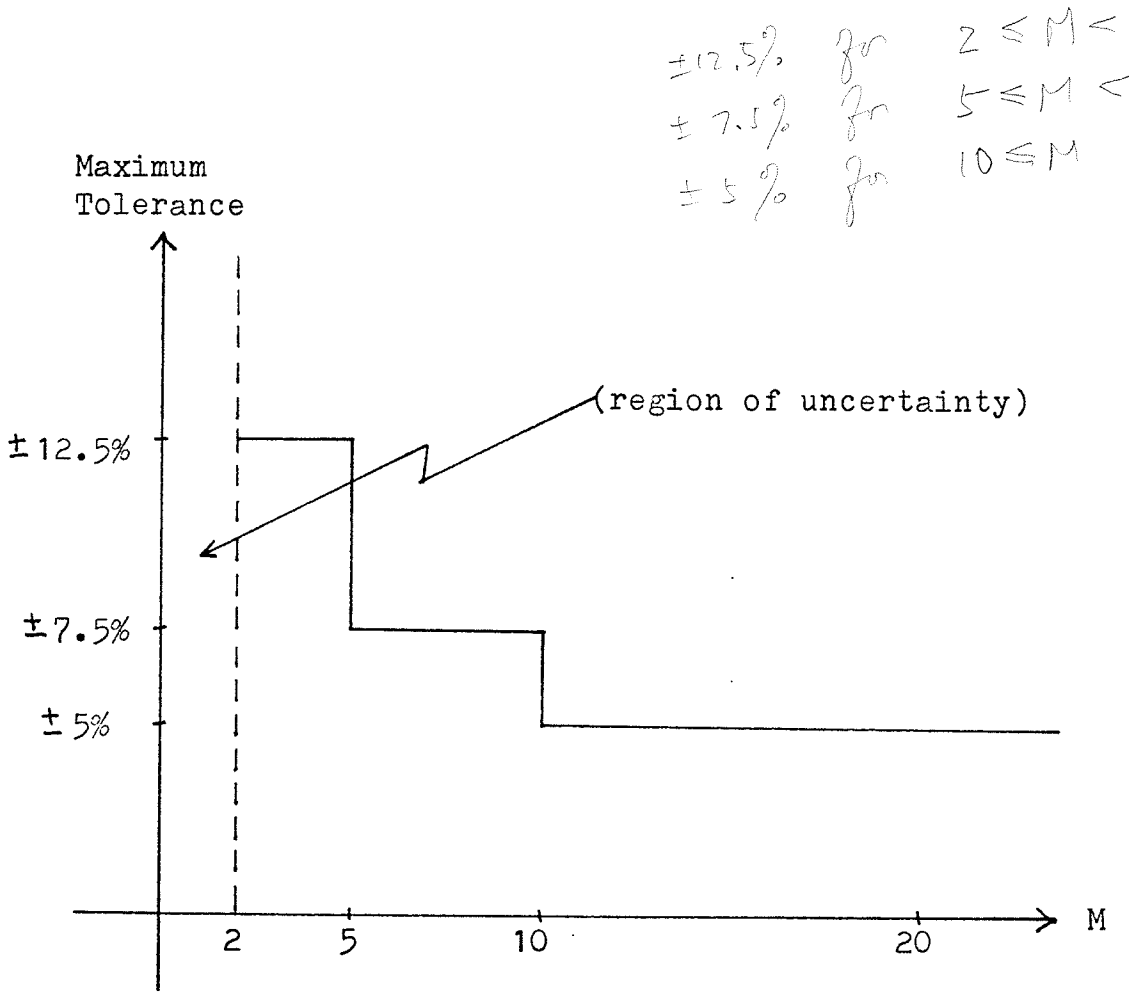


Figure 2.3: Graphical illustration of table 2.1.

Notice that the above tolerance is set for the curves of T versus M. However, our objective is to implement the dynamic curves which are plotted as W versus M. Some adjustments are necessary to convert them to suit the dynamic curves.

Let E_T be the %error in T.

Let E_W be the %error in W.

Let $T'=T(1+E_T)$.

Since W=D/T (equation 2.5), we can define

$$W' = D/T' = D/[T(1+E_T)] \qquad \text{(eq. 2.7)}$$

we can also define

$$E_W = \frac{(W'-W)}{W} 100\% \qquad \text{(eq. 2.8)}$$

Subtituting equation 2.5 and equation 2.7 into equation 2.8, E becomes

$$E_W = \frac{D/[T(1+E_T)] - D/T}{D/T} 100\%$$

which implies

$$E_W = \frac{-E_T}{1 + E_T} 100\% \qquad \text{(eq. 2.9)}$$

Using equation 2.9, the tolerance for relay operating time can be converted into tolerance for disc angular velocity (W). The results are listed in table 2.2.

M	2	5	10	20
Allowable %error Upper Bound (S_u)	+14.29%	+8.11%	+5.26%	+5.26%
Allowable %error Lower Bound (S_ℓ)	-11.11%	-6.98%	-4.76%	-4.76%

TABLE 2.2

Maximum allowable %error in disc angular velocity (W).

Chapter III

PIECE-WISE LINEAR APPROXIMATION

3.1 INTRODUCTION

The idea of piece-wise linear approximation is to use a number of straight lines to describe the behavior of the over-current dynamic curves. Let's use the dynamic curve shown in figure 2.1 as an example. To start with, three straight lines are assumed to be enough to approximate this curve. The straight lines and the dynamic curve itself are shown in figure 3.1.

The time-current characteristic curves of CO-8, CO-11 and IAC51 published by the relay manufacturers cover information between M=1 and M=50, and M=1 to M=40 for the IAC77. The dynamic curves discussed in this thesis will also span from M=1 to M=40 or 50 accordingly.

The objective of the method described here is to find the coordinates for the points $A(M_a, W_a)$, $P_1(M_1, W_1)$, $P_2(M_2, W_2)$, $B(M_b, W_b)$ Such that this piece-wise linear curve will produce the best approximation of the experimental curves with all the accuracy criteria specified in table 2.2 fulfilled.

In order to achieve this objective, it is necessary to use the theories and techniques of optimization.

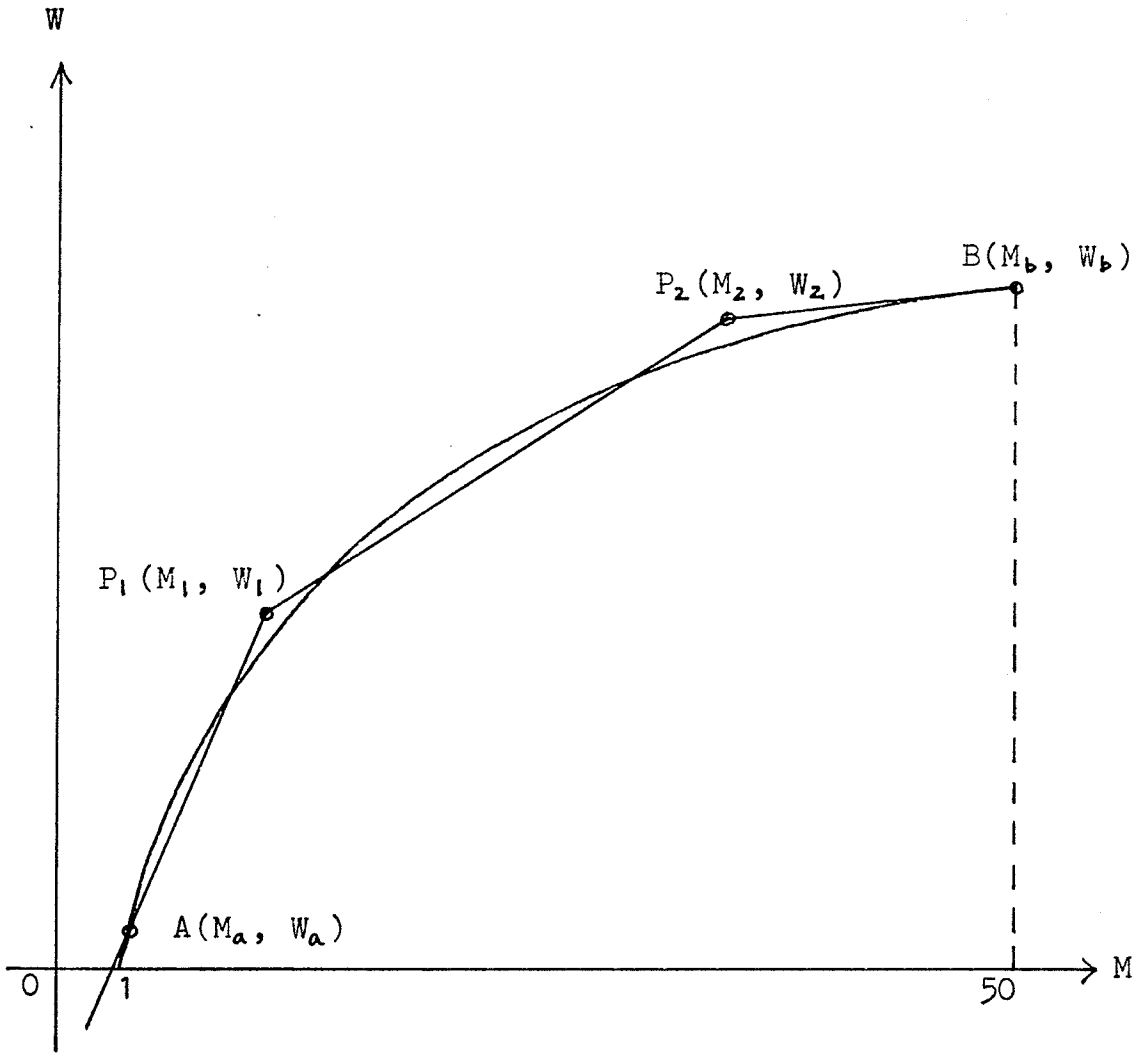


Figure 3.1: Piece-wise linear approximation of dynamic curve.

3.2 SAMPLE POINTS

In most scientific and engineering problems the quantity to be minimized is a function not only of controlled parameters but also of one or more independent variables. The objective function F then takes on a series of values as the independent variables vary. Physical measurement of such an objective function can only be entered into a digital computer as a series of values at specific sample values of independent variables. In addition the number of samples may be limited, due either to practical difficulties in making measurements, or to the extremely large number of available samples.

A similar situation is found in our optimization problem here. The objective function F, which will be defined in detail in the following sections, is a function not only of the controlled variables [$A(M_a, W_a)$, $P_1(M_1, W_1)$, $P_2(M_2, W_2)$, $B(M_b, W_b)$] but also of the independent variable (M). Since F can only be evaluated by the computer for specific sets of values of the function arguments, the process is handled by the use of a number of sample points sufficient to describe the graph of the function (dynamic curve) involved.

A program called the DIGITIZER was prepared in the electrical engineering department of the University of Manitoba. This program receives data samples and stores them in memory. This is done by driving a cursor along the manufacturer's time-current characteristic curves. Once the digitiz-

ing process of a time-current curve is finished, the data in time versus current can be converted to angular velocity (W) versus current (equation 2.5). In the research conducted here, thirty sample points were digitized for each relay characteristic curve under study.

In a later stage of solving the optimization problem in this thesis, it was found necessary to depict the behaviors of the dynamic curves for total error calculation. A very popular curve fitting method called the Cubic Spline was employed to find a mathematical description of the dynamic curves. An IMSL routine called ICSICU which computes an interpolatory approximation to a given set of points by cubic splines was used and the algorithm description is given in appendix A.

3.3 DIMENSIONALITY OF THE PROBLEM

If three straight lines are being searched for to represent the experimental dynamic curves, then as shown in figure 3.1, there will be four points to be identified, specifically points A, P_1 ,P_2 and B. Four points imply that the dimension of this search problem is eight, because each point has an ordinate and abscissa. If four straight lines were needed then the number of dimensions would increase to ten.

In principle it is just as easy to apply optimization techniques for a two-dimensional function (which have been proved to be very effective) to an n-dimensional function

(where n is a large number). Unfortunately this is not true in practise since multivariable problems have a structure entirely different from that of a single variable one. One must recognize high dimensionality as a practical difficulty. One deleterious effect of multidimensionality is that it makes the unimodality assumption less plausible as the number of dimensions increases, hence it overpowers the search techniques that were so effective when we only had lines (single variables) to deal with. The other difficulty is the very size of multidimensional spaces, it forces us to seek regions of uncertainty which are but tiny fractions of the original experimental region. Therefore it is desirable to reduce the number of dimensions of the function to a minimum.

Reduction of dimensionality can be achieved by assigning fixed values to the two terminal points, namely $A(M_a, W_a)$ and $B(M_b, W_b)$. By doing that, the three-straight-line representation will become a four-dimensional problem and the four-straight-line representation will become six-dimensional.

3.4 THE CONSTRAINTS OF THE PROBLEM

Let $W_{ex}(M)$ be the experimental curve, and let $W_{ap}(M)$ be the piece-wise linear curve. According to table 2.2, the problem presented in section 3.1 is subjected to the following constraints.

$$C_1: (1-11.11\%)W_{ex} \leqslant W_{ap} \leqslant (1+14.29\%)W_{ex} \quad \text{for } 2 \leqslant M < 5$$

C_2: $(1-6.98\%)W_{ap} \leqslant W_{ex} \leqslant (1+8.11\%)W_{ap}$ for $5 \leqslant M < 10$

C_3: $(1-4.76\%)W_{ap} \leqslant W_{ex} \leqslant (1+5.26\%)W_{ap}$ for $10 \leqslant M \leqslant 50$

The constraints C_1, C_2 and C_3 are specified as the error tolerance of the optimization problem, and are referred to as the "explicit constraints". However, there is one more constraint needed to be taken into account. This constraint is inherent in the problem and is called the "implicit constraint" C_4 as listed below.

C_4: $M_a \leqslant M_1 \leqslant M_2 \leqslant M_b$

Where M_a, M_1, M_2 and M_b are the abscissas of points $A(M_a, W_a)$, $P_1(M_1, W_1)$, $P_2(M_2, W_2)$ and $B(M_b, W_b)$ respectively. (They are shown in figure 3.1).

Any solution to the problem that can satisfy the above four constraints is called a feasible solution. The best solution among all the feasible ones is called the optimum solution. It is the objective of this chapter to find such a solution.

3.5 HANDLING THE CONSTRAINTS

A general aim when dealing with a constrained optimization problem is to reduce it to an unconstrained problem. Often it is possible to eliminate one or more of the constraints by a change of variables. In solving the problem here, both methods will be used.

The fourth constraint, C_4, can be eliminated by changing variables. The constraint C_4 is equivalent to

$$M_1 = M_a+(M_b-M_a)\sin^2(m_1) \qquad \text{(eq. 3.1)}$$

$$M_2 = M_1+(M_b-M_1)\sin^2(m_2) \qquad \text{(eq. 3.2)}$$

with m_1 and m_2 unconstrained.

The nature of these two equations can be understood by illustrating equation 3.2 graphically as shown in figure 3.2

Hence C_4 can be eliminated by substituting for M_1 and M_2 wherever they appear in the problem, and the solution will be searched over (m_1, W_1), (m_2, W_2) instead of (M_1, W_1) and (M_2, W_2). The number of constraints has been reduced to three, and a method to reduce the optimization problem to an unconstrained one is needed.

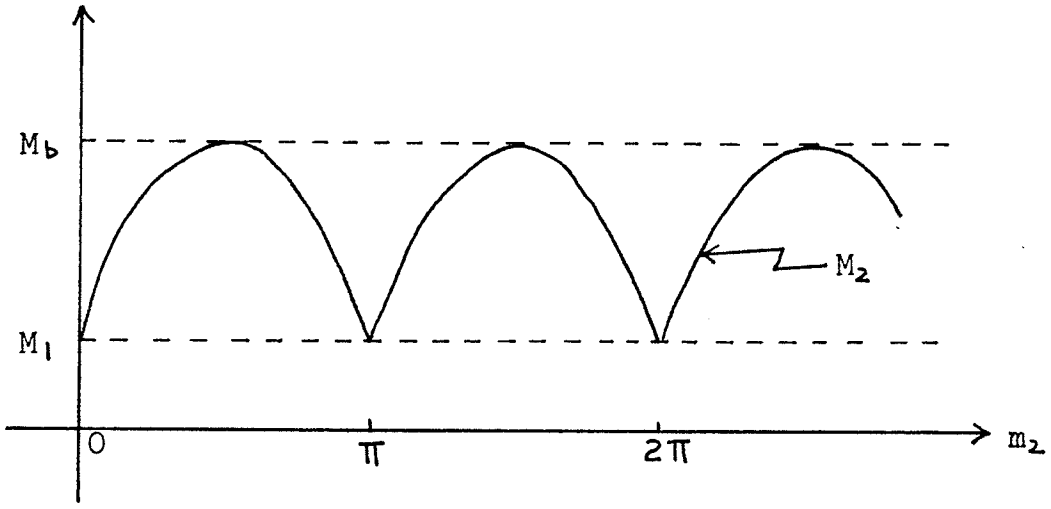


Figure 3.2: Graphical illustration of equation 3.2 .

3.5.1 Sequential Unconstrained Minimization Technique

One of the two methods that were tried to transform the problem to an unconstrained optimization problem is the Sequential Unconstrained Minimization Technique (SUMT)[4].

The idea of SUMT is to penalize constraint violation by adding a sequence of penalty functions to the objective function in such a way that the solutions to the resulting sequence of unconstrained problems tend to a constrained minimum. (A description of SUMT is given in appendix B).

The newly created functions have a form given in equation 3.3. where Z_K is the newly created function, $F(\underset{\sim}{x})$ is the ob-

jective function, $\underset{\sim}{x}$ is a vector of the objective function parameters, g_i are the constraints and w_i are weighting factors.

$$Z_K(\underset{\sim}{x}, r_k) = F(\underset{\sim}{x}) + r_k \sum_i w_i / g_i(\underset{\sim}{x}) \qquad \text{(eq. 3.3)}$$

where $k = 1, 2, 3, \ldots\ldots$

$r_k > 0$ for all k

$w_i > 0$ for all i

The objective function for minimization by SUMT was defined to be the total integration of the area bounded by the piece-wise linear approximation curve and the original experimental curve. Graphically speaking, the problem is to minimize the total shaded area shown in figure 3.3 within the constraints.

This method was tested and was found to be not very suitable for this problem. Choosing the initial value of r is a difficulty task, a wrong choice leading to minimization of a function other than the objective. It was found that the created functions (Z_K) were extremely sensitive to r_K. The technique was eventually abandoned due to the fact that suitable values of r_K could not be found in several cases. A different method called Least Pth Approximation (or Minimax Approximation) was pursued instead.

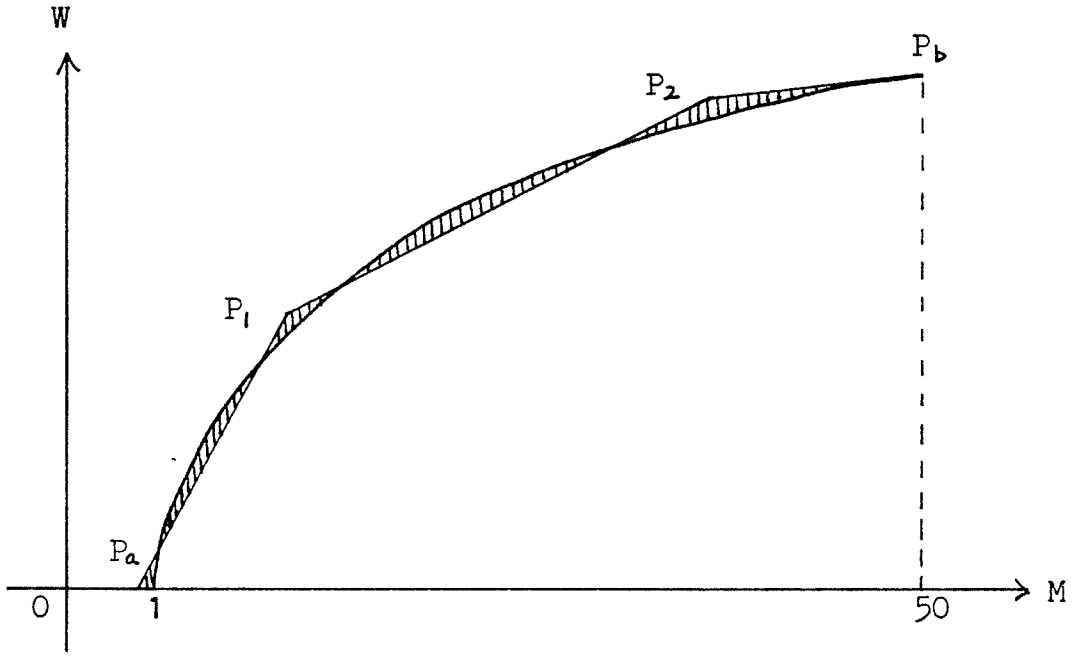


Figure 3.3: Objective for minimization by SUMT.

3.5.2 Least Pth Approximation [5]

The introduction of real error functions related to the upper and lower specifications allows the Least Pth Approximation method to achieve near minimax approxmation for a wide class of system-design problems.

With a properly formulated problem, the optimization process by the least pth method can be started with an unfeasible design : once the design became feasible, the optimization process will still continue and push the design further away from the specifications until convergence occurs.

Notation

The following notation is used :

$W(\underset{\sim}{\Phi}, M)$	The approximating function (The piece-wise linear curve).
$S_u(M)$	An upper specified boundary.
$S_\ell(M)$	A lower specified boundary.
$w_u(M)$	An upper positive weighting function.
$w_\ell(M)$	A lower positive weighting function.
$\underset{\sim}{\Phi}$	A vector containing the k controlled parameters (They are the break points of the approximating curves).
M	An independent variable (The multiple of pickup current).

Referring to figure 3.1 and assuming that only three straight lines are used to approximate the curve, W can be defined as follows,

$$W = [(W_1 - W_a)/(M_1 - M_a)](M - M_a) + W_a \quad \text{for } M_a \leq M \leq M_1 \quad \text{(eq. 3.4)}$$

$$W = [(W_2 - W_1)/(M_2 - M_1)](M - M_1) + W_1 \quad \text{for } M_1 < M \leq M_2 \quad \text{(eq. 3.5)}$$

$$W = [(W_b - W_2)/(M_b - M_2)](M - M_2) + W_2 \quad \text{for } M_2 < M \leq M_b \quad \text{(eq. 3.6)}$$

and S_u and S_ℓ are already given in table 2.2.

Error functions

The error functions related to the upper and lower specifications are defined as follow,

$$e_u(\underset{\sim}{\Phi}, M) = w_u(M)[W(\underset{\sim}{\Phi}, M)-S_u(M)] \qquad \text{(eq. 3.7)}$$

$$e_\ell(\underset{\sim}{\Phi}, M) = w_\ell(M)[W(\underset{\sim}{\Phi}, M)-S_\ell(M)] \qquad \text{(eq. 3.8)}$$

We will evaluate all the functions at a finite discrete set of values of M. It is assumed that a sufficient number of sample points have been chosen so that the discrete approximation problem adequately approximates the continuous problem. As already mentioned in section 3.2, thirty sample points are used in this particular problem. Therefore we will define the functions

$$e_{ui}(\underset{\sim}{\Phi}) = e_u(\underset{\sim}{\Phi}, M_i) \qquad \text{(eq. 3.9)}$$

$$e_{\ell j}(\underset{\sim}{\Phi}) = e_\ell(\underset{\sim}{\Phi}, M_j) \qquad \text{(eq. 3.10)}$$

where $i = 1, 2, 3, \ldots . 29, 30.$

$j = 1, 2, 3, \ldots . 29, 30.$

The objective function

Let $$MX(\underset{\sim}{\phi}) = \max_{i,j}[e_{ui}(\underset{\sim}{\phi}) , -e_{\ell j}(\underset{\sim}{\phi})] \qquad \text{(eq. 3.11)}$$

then the objective function is

$$F(\underset{\sim}{\phi}) = MX(\underset{\sim}{\phi})\left\{\sum_{i\in K_u}\left[\frac{e_{ui}(\underset{\sim}{\phi})}{MX(\underset{\sim}{\phi})}\right]^q + \sum_{j\in K_\ell}\left[\frac{-e_{\ell j}(\underset{\sim}{\phi})}{MX(\underset{\sim}{\phi})}\right]^q\right\}^{1/q} \qquad \text{(eq. 3.12)}$$

where

$$K_u(\underset{\sim}{\phi}) = \begin{cases} i \mid e_{ui} \geqslant 0 \ ; \ i \in 1,2,\ldots 30. & \text{if } MX(\underset{\sim}{\phi}) > 0 \\ 1,2,\ldots 30. & \text{if } MX(\underset{\sim}{\phi}) \leqslant 0 \end{cases} \qquad \text{(eq. 3.13)}$$

$$K_\ell(\underset{\sim}{\phi}) = \begin{cases} j \mid -e_{\ell j} \geqslant 0 \ ; \ j \in 1,2,\ldots 30. & \text{if } MX(\underset{\sim}{\phi}) > 0 \\ 1,2,\ldots 30. & \text{if } MX(\underset{\sim}{\phi}) \leqslant 0 \end{cases} \qquad \text{(eq. 3.14)}$$

and $$q = P\,\frac{MX(\underset{\sim}{\phi})}{|MX(\underset{\sim}{\phi})|} \qquad 1 < P < \infty$$

This method was found to be very effective for solving the problem. A computer program was written to search for

the solutions. In this program, the constraints are handled by the least pth approximation method and the optimization process is performed by a method called the simplex search method. The choice of optimization method will be explained in the next section. The flow chart of the program is shown in figure 3.4. For details of the program listing please refer to appendix C.

In the program mentioned above, seven iterations with seven different sets of starting points are used to find a solution. Different results are then compared with each other. Those results with a negative value for $MX(\underset{\sim}{\phi})$ are called feasible solutions because a negative $MX(\underset{\sim}{\phi})$ indicates that the solution satisfies all the constraints. Among all these feasible solutions, the one that has the smallest total area bounded by the original experimental curve and the piecewise linear curve will be chosen as the best solution.

The reason for using several iterations with different sets of starting points to find the solution is that any particular function may possess several local minima. One of these will be the global minimum but it is usually impossible to determine if a local minimum is also the global minimum. Even extensive testing of all minima found is not a complete answer. Therefore, the suggestion here is to spread out the locations of several starting points and choose the one that best meets the minimum criterion, i.e. the smallest total error area.

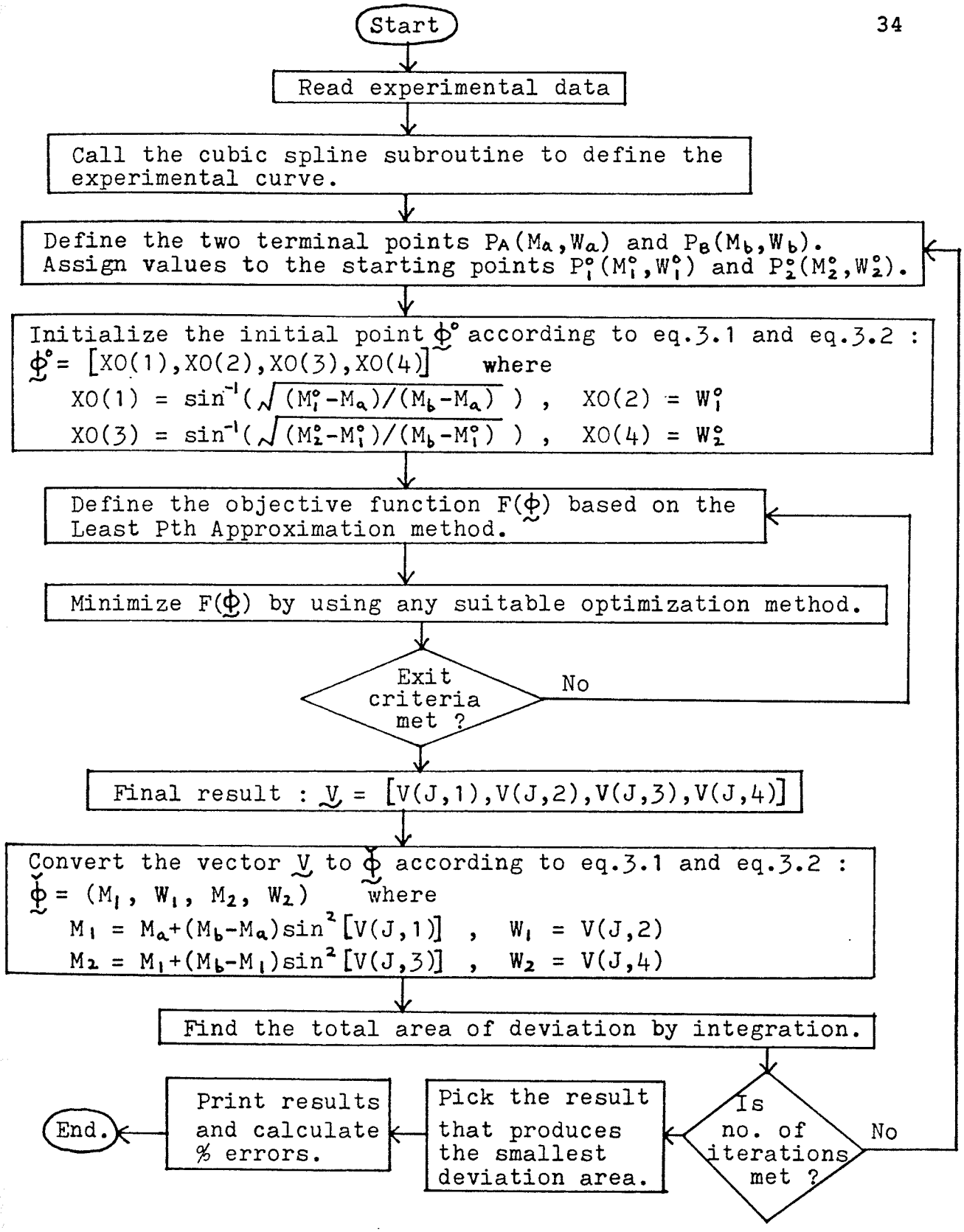


Figure 3.4: Program flow chart using the least Pth method.

3.6 THE CHOICE OF OPTIMIZATION METHOD (THE SIMPLEX SEARCH METHOD)

In section 3.5, the problem was reduced from a constrained one to an unconstrained one. It is necessary now to choose a suitable optimization method to find the optimal solutions.

Methods for multi-variable optimization fall naturally into two classes. Although these classes are not completely separate, methods can be divided into direct search methods, and gradient methods which in addition require gradient information in the form of the Jacobian gradient vector as well as the Hessian matrix H.

A direct search method is a method which relies only on evaluating $f(\underset{\sim}{x})$ at a sequence of points $\underset{\sim}{x}_1$, $\underset{\sim}{x}_2$, $\underset{\sim}{x}_3$, ... and comparing values, in order to reach the optimal point x^*. Direct search methods are commonly used in the following circumstances :

(a) The function $f(\underset{\sim}{x})$ is not differentiable, or is subject to random errors.

(b) The derivatives $\partial f/\partial \underset{\sim}{x}_j$ are discontinuous, or their evaluation is much more difficult than the evaluation of the function $f(\underset{\sim}{x})$ itself.

(c) An approximate solution may be required at any time during the course of the calculation.

(d) Computer facilities are inadequate for the more complicated algorithms.

The fundamental problem in the design of a direct search method is to determine the point $\underset{\sim}{x}_{r+1}$, given the points $\underset{\sim}{x}_1$, $\underset{\sim}{x}_2$, ... $\underset{\sim}{x}_r$ and the function values $f(\underset{\sim}{x}_1)$, $f(\underset{\sim}{x}_2)$, ... $f(\underset{\sim}{x}_r)$.

Due to the nature of the unconstrained problem described in section 3.5.2, it is uneconomic or impossible to compute the derivatives of the objective function. It is more suitable to use the direct search method.

Two different direct search methods, namely the Pattern search method and the Simplex search method of Nelder and Mead were tried and compared. The results showed that both of these methods were effective for this problem. However Nelder and Mead's simplex method was found to be more efficient than the pattern search method, i.e. it required fewer function evaluations and hence it gave a faster convergence than the pattern search method. A comparison of the two methods is given in table 3.1.

A digital computer program using the simplex method was prepared at the University of Manitoba. This program receives the starting points coordinates, termination criteria, maximum allowable number of function evaluations, simplex side length reduction factor, simplex expansion factor and simplex contraction factor. The program then computes the minimum approximation until the exit criteria are met.

	Method used	Search results for coefficients of eq.4.1 (P.49)	Total area of deviation	No. of function evaluations
1st run	Simplex	A = 13.01 B = 1.38 C = 8.98	43.63	62
	Pattern	A = 13.07 B = 1.38 C = 9.00	43.43	747
2nd run	Simplex	A = 14.45 B = 1.34 C = 9.69	15.33	80
	Pattern	A = 12.90 B = 1.41 C = 9.08	12.12	443
3rd run	Simplex	A = 13.51 B = 1.37 C = 9.23	13.88	46
	Pattern	A = 12.78 B = 1.40 C = 9.06	13.10	446
4th run	Simplex	A = 13.20 B = 1.37 C = 9.09	17.37	40
	Pattern	A = 12.55 B = 1.40 C = 9.02	15.23	427

TABLE 3.1

comparison of simplex and pattern search methods.

A detailed algorithm description and program listing of the simplex method and the pattern search method are given in appendix D and E respectively.

3.7 THE RESULTS

The dynamic characteristics of four different types of inverse-time overcurrent relays, namely CO-8, CO-11, IAC51 and IAC77 were chosen to be the subject of this study. The piece-wise linear approximation for each time dial setting of these relays was computed.

Three straight lines were found to be enoungh for the CO-8 and IAC51. However, due to the more extreme curvature of the dynamic curves of the CO-11 and IAC77, a minimum of four straight lines must be used.

The piece-wise linearization method works very well to approximate the dynamic curves. The results for all time dial settings of the four types of relays fall well within the error specifications. It is of interest to see a graphical comparison between the original experimental curves and the approximating curves. In figure 3.5, several piece-wise linearized dynamic curves of relay CO-8 were converted back to time versus current and plotted together with the manufacturer's curves. The original curves are shown in dotted lines.

Tables 3.2, 3.3, 3.4 and 3.5 show the percentage errors as some typical examples, these examples were calculated for D=5 of the four relays. Tables 3.6, 3.7, 3.8 and 3.9 summarize the results for all the time dial settings of the four different types of relays. Where (M_a, W_a), (M_1, W_1), (M_2, W_2), (M_3, W_3) and (M_b, W_b) are the coordinates of P_a, P_1, P_2, P_3 and P_b respectively.

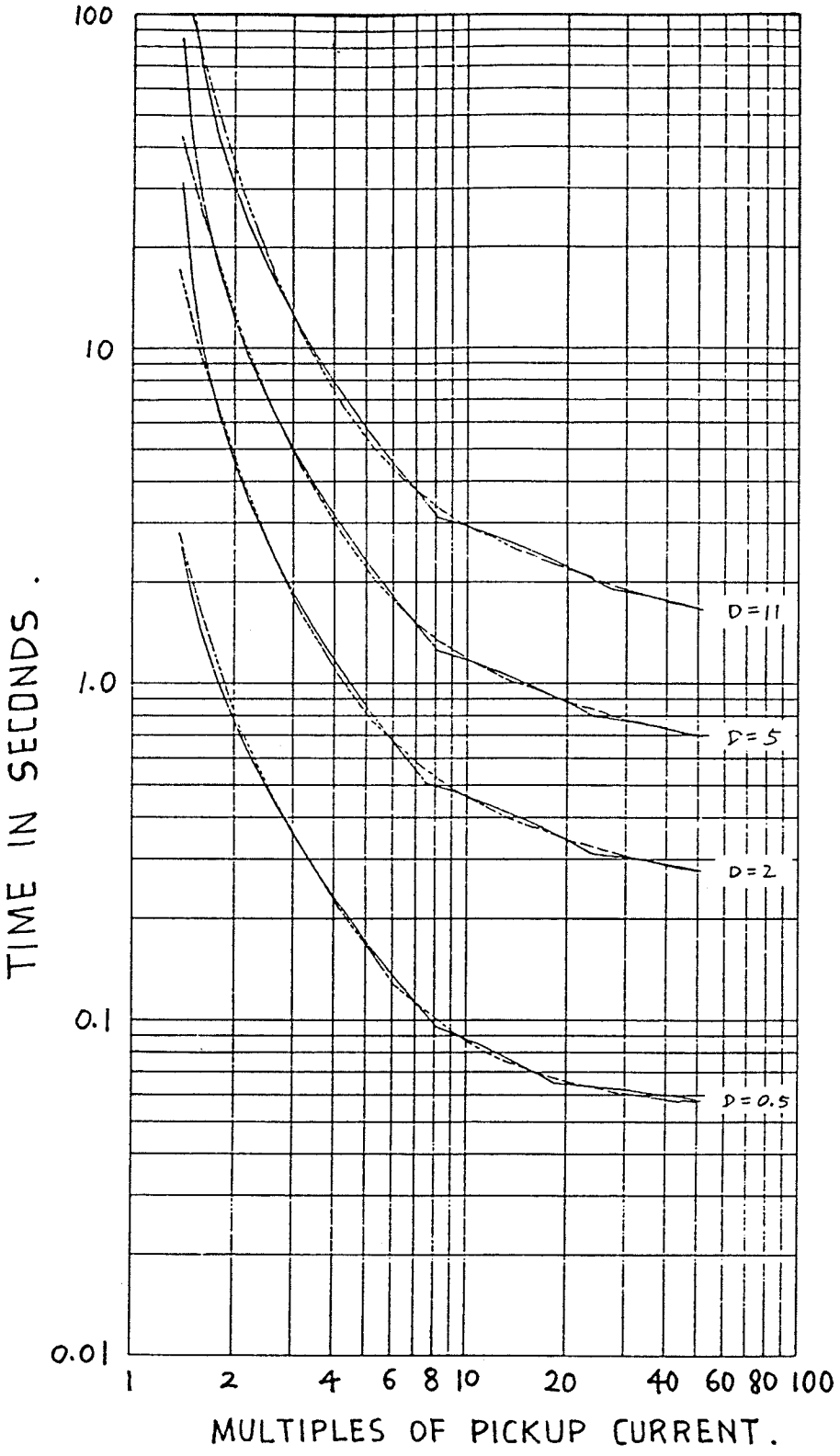


Figure 3.5: Comparison between the original and the approximate inverse-time characteristic curves. Relay : CO-8 .

(MA,WA) = (1.000000, -0.200000)

(M1,W1) = (7.726334, 3.832138)

(M2,W2) = (23.243301, 6.208254)

(MB,WB) = (51.549194, 7.173299)

MX = -0.031195

M	ORIG W	APPX W	ERROR	%ERROR
1.39328	0.11523	0.03575	-0.07948	-68.97211
1.60976	0.19749	0.16552	-0.03197	-16.18721
2.12915	0.46477	0.47687	0.01210	2.60328
2.64677	0.78844	0.78716	-0.00128	-0.16229
3.19148	1.14159	1.11370	-0.02789	-2.44352
4.22888	1.79925	1.73557	-0.06368	-3.53944
5.31750	2.47176	2.38815	-0.08361	-3.38267
6.28414	2.95959	2.96761	0.00802	0.27093
7.35007	3.43313	3.60658	0.17345	5.05220
8.33245	3.77726	3.92495	0.14769	3.90997
9.44705	4.06912	4.09563	0.02651	0.65150
10.48887	4.33711	4.25517	-0.08194	-1.88934
11.89249	4.62189	4.47010	-0.15179	-3.28423
13.62613	4.92838	4.73557	-0.19281	-3.91216
15.61398	5.14402	5.03997	-0.10405	-2.02267
16.98315	5.31147	5.24964	-0.06184	-1.16425
18.85262	5.54308	5.53591	-0.00717	-0.12941
20.71623	5.72380	5.82128	0.09748	1.70313
23.24670	5.91098	6.20837	0.29739	5.03111
25.81406	6.10399	6.29590	0.19191	3.14402
28.36581	6.30300	6.38290	0.07990	1.26765
30.84605	6.43988	6.46746	0.02757	0.42817
33.19470	6.51037	6.54753	0.03716	0.57077
35.72047	6.65144	6.63364	-0.01779	-0.26750
38.44028	6.72425	6.72637	0.00213	0.03161
41.36517	6.86994	6.82609	-0.04385	-0.63833
43.59059	6.94448	6.90196	-0.04251	-0.61218
46.41789	7.09461	6.99835	-0.09625	-1.35668
48.98355	7.13374	7.08583	-0.04791	-0.67161
51.54919	7.17330	7.17330	-0.00000	-0.00001

TABLE 3.2

%error found for CO-8 at D=5 by piece-wise approximation method.

(MA,WA) = (1.000000, -0.400000)

(M1,W1) = (4.326900, 2.256432)

(M2,W2) = (18.979355, 24.151245)

(M3,W3) = (36.360214, 32.501343)

(MB,WB) = (51.902557, 33.743027)

MX.= -0.042609

M	ORIG W	APPX W	ERROR	%ERROR
1.44691	0.11230	-0.04316	-0.15546	-138.43002
1.70092	0.21074	0.15966	-0.05108	-24.23816
2.21442	0.52511	0.56968	0.04457	8.48717
2.74484	0.91625	0.99320	0.07696	8.39909
3.26717	1.38031	1.41027	0.02996	2.17078
4.35713	2.51307	2.30161	-0.21146	-8.41455
5.34918	3.86707	3.78399	-0.08308	-2.14843
6.30621	5.30149	5.21407	-0.08742	-1.64894
7.36058	6.89677	6.78958	-0.10719	-1.55423
8.32964	8.51225	8.23763	-0.27462	-3.22619
9.52727	10.28948	10.02722	-0.26226	-2.54883
10.56330	12.04992	11.57533	-0.47459	-3.93855
11.83294	14.41139	13.47252	-0.93887	-6.51478
13.82318	17.06271	16.44647	-0.61624	-3.61163
15.65500	19.36803	19.18372	-0.18431	-0.95162
17.37216	21.30122	21.74962	0.44839	2.10502
19.06992	22.93996	24.19475	1.25479	5.46989
20.72145	24.70239	24.98817	0.28578	1.15690
23.47858	26.61037	26.31274	-0.29762	-1.11845
25.78522	28.06393	27.42090	-0.64304	-2.29132
28.61914	29.29153	28.78238	-0.50916	-1.73823
31.10951	30.24879	29.97879	-0.27000	-0.89261
33.82603	31.23735	31.28386	0.04651	0.14889
36.00468	31.58398	32.33054	0.74655	2.36370
38.73396	32.27354	32.69098	0.41743	1.29343
41.24023	32.63168	32.89120	0.25952	0.79531
43.90866	32.99382	33.10439	0.11057	0.33511
46.26305	33.35681	33.29248	-0.06433	-0.19286
48.74368	33.72377	33.49066	-0.23311	-0.69123
51.90256	33.74303	33.74301	-0.00002	-0.00005

TABLE 3.3

%error found for CO-11 at D=5 by piece-wise approximation method.

(MA,WA) = (2.000000, 1.250000)

(M1,W1) = (4.566688, 2.803676)

(M2,W2) = (21.241760, 5.285306)

(MB,WB) = (50.595093, 6.682507)

MX = -0.048715

M	ORIG W	APPX W	ERROR	%ERROR
1.11967	0.19840	0.71712	0.51872	261.45654
1.23586	0.37052	0.78745	0.41693	112.52689
1.35225	0.57367	0.85790	0.28423	49.54578
1.52797	0.80957	0.96427	0.15471	19.10973
1.78198	1.10739	1.11803	0.01064	0.96109
2.03739	1.35085	1.27263	-0.07822	-5.79011
2.53139	1.71909	1.57166	-0.14743	-8.57592
3.05076	1.99176	1.88605	-0.10571	-5.30741
4.03783	2.38214	2.48355	0.10141	4.25698
5.07433	2.73323	2.87922	0.14599	5.34123
6.11893	2.97529	3.03468	0.05939	1.99621
7.15237	3.17125	3.18848	0.01723	0.54344
8.10253	3.37916	3.32989	-0.04927	-1.45802
9.08442	3.56312	3.47602	-0.08711	-2.44470
10.07951	3.75675	3.62411	-0.13265	-3.53088
12.28205	4.08984	3.95190	-0.13795	-3.37287
15.28172	4.45330	4.39832	-0.05498	-1.23458
17.67715	4.74482	4.75481	0.00999	0.21048
20.23952	4.95372	5.13615	0.18242	3.68257
22.69437	5.16940	5.35445	0.18505	3.57974
25.18501	5.33817	5.47300	0.13483	2.52576
27.66133	5.45494	5.59087	0.13593	2.49192
30.37537	5.69136	5.72006	0.02870	0.50419
33.01547	5.81531	5.84573	0.03042	0.52306
35.14671	5.87940	5.94717	0.06777	1.15272
37.80113	6.06964	6.07352	0.00388	0.06395
40.24133	6.13653	6.18967	0.05314	0.86599
43.28050	6.33509	6.33434	-0.00076	-0.01194
47.53152	6.54131	6.53668	-0.00463	-0.07078
50.59509	6.68251	6.68251	-0.00000	-0.00001

TABLE 3.4

%error found for IAC51 at D=5 by piece-wise approximation method.

(MA,WA) = (2.000000, 0.510606)

(M1,W1) = (4.358624, 3.501557)

(M2,W2) = (14.726746, 33.687408)

(M3,W3) = (25.613647, 48.680969)

(MB,WB) = (39.984314, 51.365112)

MX = -0.000001

M	ORIG W	APPX W	ERROR	%ERROR
1.27440	0.16804	-0.40952	-0.57756	-343.70630
1.59282	0.31261	-0.00573	-0.31835	-101.83391
1.86831	0.50320	0.34362	-0.15958	-31.71339
2.12176	0.66524	0.66501	-0.00023	-0.03501
2.62276	1.17482	1.30032	0.12549	10.68195
3.17383	1.87166	1.99913	0.12747	6.81042
3.68107	2.66035	2.64235	-0.01800	-0.67645
4.13623	3.55399	3.21954	-0.33445	-9.41046
4.64604	4.50789	4.33834	-0.16955	-3.76122
5.16350	5.54305	5.84487	0.30181	5.44487
6.11654	8.21136	8.61956	0.40821	4.97122
7.16684	11.31313	11.67743	0.36430	3.22017
8.21910	14.19956	14.74098	0.54142	3.81295
9.03984	17.46149	17.13048	-0.33101	-1.89565
10.15044	21.02899	20.36389	-0.66510	-3.16278
11.63829	25.58575	24.69563	-0.89012	-3.47897
13.48112	30.17419	30.06085	-0.11334	-0.37563
15.28941	34.14426	34.46231	0.31805	0.93150
16.98262	37.06149	36.79422	-0.26727	-0.72116
18.46272	39.84085	38.83263	-1.00822	-2.53063
20.28484	41.93472	41.34206	-0.59267	-1.41331
21.82562	43.68941	43.46405	-0.22536	-0.51582
23.23843	45.05106	45.40979	0.35873	0.79628
25.00185	46.45187	47.83838	1.38651	2.98482
27.75305	47.88612	49.08055	1.19443	2.49431
30.16750	48.35809	49.53152	1.17343	2.42655
32.79425	49.34384	50.02216	0.67831	1.37467
34.91216	49.83717	50.41772	0.58055	1.16490
37.55873	50.85664	50.91205	0.05540	0.10894
39.98431	51.36511	51.36510	-0.00002	-0.00003

TABLE 3.5

%error found for IAC77 at D=5 by piece-wise approximation method.

TDS = 11		TDS = 10	
(Ma,Wa)=(	1.000000, -0.200000)	(Ma,Wa)=(	1.000000, -0.200000)
(M1,W1)=(	7.912584, 3.466780)	(M1,W1)=(	8.030608, 3.659065)
(M2,W2)=(28.144363,	5.915689)	(M2,W2)=(20.213516,	5.498960)
(Mb,Wb)=(52.297333,	6.644351)	(Mb,Wb)=(51.721924,	6.855216)

TDS = 9		TDS = 8	
(Ma,Wa)=(	1.000000, -0.200000)	(Ma,Wa)=(	1.000000, -0.200000)
(M1,W1)=(	7.930999, 3.642295)	(M1,W1)=(	7.044111, 3.336432)
(M2,W2)=(27.442230,	6.141142)	(M2,W2)=(19.634323,	5.654264)
(Mb,Wb)=(52.232391,	7.152041)	(Mb,Wb)=(51.660172,	7.139319)

TDS = 7		TDS = 6	
(Ma,Wa)=(	1.000000, -0.200000)	(Ma,Wa)=(	1.000000, -0.200000)
(M1,W1)=(	7.438444, 3.558677)	(M1,W1)=(	7.816092, 3.698848)
(M2,W2)=(23.026642,	5.899845)	(M2,W2)=(22.826859,	6.087655)
(Mb,Wb)=(51.633026,	7.015601)	(Mb,Wb)=(51.593567,	7.119135)

TDS = 5		TDS = 4	
(Ma,Wa)=(	1.000000, -0.200000)	(Ma,Wa)=(	1.000000, -0.200000)
(M1,W1)=(	7.726334, 3.832138)	(M1,W1)=(	7.534724, 3.664346)
(M2,W2)=(23.243301,	6.208254)	(M2,W2)=(20.429993,	5.913383)
(Mb,Wb)=(51.549194,	7.173299)	(Mb,Wb)=(51.490112,	7.390114)

TDS = 3		TDS = 2	
(Ma,Wa)=(	1.000000, -0.200000)	(Ma,Wa)=(	1.000000, -0.200000)
(M1,W1)=(	7.060617, 3.499621)	(M1,W1)=(	7.587606, 3.885877)
(M2,W2)=(23.519455,	6.362973)	(M2,W2)=(19.854980,	6.075693)
(Mb,Wb)=(51.968460,	7.294279)	(Mb,Wb)=(51.335312,	7.184522)

TDS = 1		TDS = 0.5	
(Ma,Wa)=(	1.000000, -0.200000)	(Ma,Wa)=(	1.000000, -0.200000)
(M1,W1)=(	7.030705, 3.753448)	(M1,W1)=(	7.630236, 5.076595)
(M2,W2)=(19.357574,	6.457766)	(M2,W2)=(16.741150,	7.509706)
(Mb,Wb)=(51.693527,	7.842308)	(Mb,Wb)=(51.508575,	8.650692)

TABLE 3.6

Piece-wise linear approximation results summary, CO-8.

TDS = 11

(Ma,Wa)=(1.000000, -0.400000)
(M1,W1)=(4.252516, 1.970047)
(M2,W2)=(14.954429, 18.026810)
(M3,W3)=(28.615555, 31.114029)
(Mb,Wb)=(52.229492, 37.207275)

TDS = 10

(Ma,Wa)=(1.000000, -0.400000)
(M1,W1)=(4.241950, 2.042159)
(M2,W2)=(15.045345, 18.522629)
(M3,W3)=(27.939392, 31.268829)
(Mb,Wb)=(51.636612, 37.542557)

TDS = 9

(Ma,Wa)=(1.000000, -0.400000)
(M1,W1)=(4.239755, 2.126500)
(M2,W2)=(18.690826, 24.196060)
(M3,W3)=(36.411453, 34.628647)
(Mb,Wb)=(52.140121, 36.752762)

TDS = 8

(Ma,Wa)=(1.000000, -0.400000)
(M1,W1)=(3.870703, 1.789781)
(M2,W2)=(18.523590, 23.839920)
(M3,W3)=(36.833115, 33.622238)
(Mb,Wb)=(52.105392, 35.152283)

TDS = 7

(Ma,Wa)=(1.000000, -0.400000)
(M1,W1)=(4.254417, 2.182305)
(M2,W2)=(18.874710, 24.390427)
(M3,W3)=(37.803864, 34.162125)
(Mb,Wb)=(52.036072, 35.611755)

TDS = 6

(Ma,Wa)=(1.000000, -0.400000)
(M1,W1)=(4.349241, 2.293500)
(M2,W2)=(19.124908, 24.427017)
(M3,W3)=(35.185120, 33.399384)
(Mb,Wb)=(51.976654, 34.608932)

TDS = 5

(Ma,Wa)=(1.000000, -0.400000)
(M1,W1)=(4.326900, 2.256432)
(M2,W2)=(18.979355, 24.151245)
(M3,W3)=(36.360214, 32.501343)
(Mb,Wb)=(51.902557, 33.743027)

TDS = 4

(Ma,Wa)=(1.000000, -0.400000)
(M1,W1)=(3.943371, 1.968444)
(M2,W2)=(14.614110, 18.607422)
(M3,W3)=(27.297302, 30.000534)
(Mb,Wb)=(51.784164, 34.702225)

TDS = 3

(Ma,Wa)=(1.000000, -0.400000)
(M1,W1)=(3.842682, 1.874901)
(M2,W2)=(18.793060, 23.378220)
(M3,W3)=(36.011887, 33.839935)
(Mb,Wb)=(51.651337, 34.525284)

TDS = 2

(Ma,Wa)=(1.000000, -0.400000)
(M1,W1)=(4.013988, 2.324063)
(M2,W2)=(18.445160, 23.248672)
(M3,W3)=(38.283295, 30.248672)
(Mb,Wb)=(51.509003, 31.178497)

TDS = 1

(Ma,Wa)=(1.000000, -0.400000)
(M1,W1)=(3.632900, 1.911270)
(M2,W2)=(14.621381, 15.627641)
(M3,W3)=(28.846863, 23.190984)
(Mb,Wb)=(51.269455, 26.033752)

TDS = 0.5

(Ma,Wa)=(1.000000, -0.400000)
(M1,W1)=(4.240770, 2.982196)
(M2,W2)=(9.306017, 9.737250)
(M3,W3)=(25.004184, 19.640701)
(Mb,Wb)=(50.370163, 28.833405)

TABLE 3.7

Piece-wise linear approximation results summary, CO-11.

TDS = 10		TDS = 9	
(Ma,Wa)=(2.000000,	1.250000)	(Ma,Wa)=(2.000000,	1.250000)
(M1,W1)=(5.585267,	2.720755)	(M1,W1)=(5.594883,	2.803534)
(M2,W2)=(21.347855,	4.877323)	(M2,W2)=(22.925507,	5.202778)
(Mb,Wb)=(50.416916,	6.257117)	(Mb,Wb)=(50.878601,	6.514183)
TDS = 8		**TDS = 7**	
(Ma,Wa)=(2.000000,	1.250000)	(Ma,Wa)=(2.000000,	1.250000)
(M1,W1)=(5.530257,	2.882948)	(M1,W1)=(4.919003,	2.773321)
(M2,W2)=(22.961365,	5.305884)	(M2,W2)=(21.018326,	5.196920)
(Mb,Wb)=(50.292923,	6.559031)	(Mb,Wb)=(50.748672,	6.708193)
TDS = 6		**TDS = 5**	
(Ma,Wa)=(2.000000,	1.250000)	(Ma,Wa)=(2.000000,	1.250000)
(M1,W1)=(4.671986,	2.763935)	(M1,W1)=(4.566688,	2.803676)
(M2,W2)=(20.442245,	5.251792)	(M2,W2)=(21,241760,	5.285306)
(Mb,Wb)=(50.140732,	6.860616)	(Mb,Wb)=(50.595093,	6.682507)
TDS = 4		**TDS = 3**	
(Ma,Wa)=(2.000000,	1.250000)	(Ma,Wa)=(2.000000,	1.250000)
(M1,W1)=(4.248973,	2.702619)	(M1,W1)=(3.726619,	2.573116)
(M2,W2)=(21.774567,	5.564614)	(M2,W2)=(20.488693,	5.273116)
(Mb,Wb)=(49.927490,	7.301662)	(Mb,Wb)=(50.879807,	6.813467)
TDS = 2		**TDS = 1**	
(Ma,Wa)=(2.000000,	1.250000)	(Ma,Wa)=(2.000000,	1.250000)
(M1,W1)=(3.422415,	2.288331)	(M1,W1)=(6.434784,	2.899257)
(M2,W2)=(15.833755,	4.618190)	(M2,W2)=(22.330627,	4.471057)
(Mb,Wb)=(50.721024,	6.401083)	(Mb,Wb)=(50.505325,	5.109439)

TDS = 0.5

(Ma,Wa)=(2.000000, 1.250000)
(M1,W1)=(6.603162, 2.251510)
(M2,W2)=(20.118668, 3.233689)
(Mb,Wb)=(50.328644, 3.751941)

TABLE 3.8

Piece-wise linear approximation results summary, IAC51.

TDS = 10

(Ma,Wa)=(2.000000, 0.516391)
(M1,W1)=(4.366284, 2.808301)
(M2,W2)=(10.212614, 16.256134)
(M3,W3)=(18.585648, 37.164871)
(Mb,Wb)=(40.180832, 45.745651)

TDS = 9

(Ma,Wa)=(2.000000, 0.493339)
(M1,W1)=(4.350057, 2.882337)
(M2,W2)=(15.505034, 29.110794)
(M3,W3)=(24.287720, 41.769379)
(Mb,Wb)=(40.220627, 47.604660)

TDS = 8

(Ma,Wa)=(2.000000, 0.488560)
(M1,W1)=(4.222894, 2.795351)
(M2,W2)=(15.461820, 29.639359)
(M3,W3)=(23.425797, 43.708618)
(Mb,Wb)=(40.266174, 49.953033)

TDS = 7

(Ma,Wa)=(2.000000, 0.499816)
(M1,W1)=(4.186208, 2.909414)
(M2,W2)=(15.506436, 30.222321)
(M3,W3)=(22.511963, 44.636536)
(Mb,Wb)=(40.300354, 49.501572)

TDS = 6

(Ma,Wa)=(2.000000, 0.505331)
(M1,W1)=(4.183851, 3.240717)
(M2,W2)=(14.737992, 31.686752)
(M3,W3)=(25.660522, 44.210022)
(Mb,Wb)=(39.522354, 51.678604)

TDS = 5

(Ma,Wa)=(2.000000, 0.510606)
(M1,W1)=(4.358624, 3.501557)
(M2,W2)=(14.726746, 33.687408)
(M3,W3)=(25.613647, 48.680969)
(Mb,Wb)=(39.984314, 51.365112)

TDS = 4

(Ma,Wa)=(2.000000, 0.544569)
(M1,W1)=(4.305526, 3.675531)
(M2,W2)=(9.971620, 20.781296)
(M3,W3)=(18.600113, 41.496826)
(Mb,Wb)=(39.617493, 49.018311)

TDS = 3

(Ma,Wa)=(2.000000, 0.541541)
(M1,W1)=(4.242242, 3.474843)
(M2,W2)=(14.572657, 31.416092)
(M3,W3)=(29.217758, 45.435791)
(Mb,Wb)=(39.284576, 49.154602)

TDS = 2

(Ma,Wa)=(2.000000, 0.556664)
(M1,W1)=(4.148196, 3.421675)
(M2,W2)=(14.349888, 28.086136)
(M3,W3)=(29.027725, 42.017441)
(Mb,Wb)=(39.373581, 45.667084)

TDS = 1

(Ma,Wa)=(2.000000, 0.575853)
(M1,W1)=(4.101241, 3.604546)
(M2,W2)=(14.289413, 22.074783)
(M3,W3)=(28.579865, 34.454819)
(Mb,Wb)=(38.695496, 37.960693)

TDS = 0.5

(Ma,Wa)=(2.000000, 0.679314)
(M1,W1)=(5.086073, 4.522317)
(M2,W2)=(14.228571, 15.576043)
(M3,W3)=(23.528015, 21.362854)
(Mb,Wb)=(38.772186, 25.375717)

TABLE 3.9

Piece-wise linear approximation results summary, IAC77.

Chapter IV

NON-LINEAR CONTINUOUS MODEL

4.1 PROPOSED MODEL

The second alternative to implement the dynamic curves is to find a mathematical expression to represent the curves. A possible model is recommended as follows :

$$W(M) = \frac{C(M-1)^B}{A+(M-1)^B} \qquad \text{(eq. 4.1)}$$

where A, B, C are constants and M is the multiple of pickup current.

As M increases, W will increase exponentially but gradually approaches zero as M decreases to the minimum pickup value.

Equation 4.1 is derived from equation 4.2, which is proposed to represent the time-current characteristics of over-current relays. Using equation 2.5 (W=D/T), we can convert T(M) into W(M) and therefore we have equation 4.1.

$$T(M) = \frac{A+(M-1)^{B}}{(M-1)^{B}} D \quad ; \quad D = 1/2,1,2,3,.... \qquad \text{(eq. 4.2)}$$

where D are the different time dial settings.

Ideally, A, B and C are constant for different values of D. However, as explained in section 2.1, the dynamic curves for different Ds of the same type of relay do not coincide. This implies that for each D there will be a set of A, B, C. We may restate equation 4.1 as the following :

$$W_K(\underset{\sim}{\phi}_K, M) = \frac{C_K(M-1)^{B_K}}{A_K+(M-1)^{B_K}} \quad ; \quad K = D \qquad \text{(eq. 4.3)}$$

Where $\underset{\sim}{\phi}_K$ is the controlled parameter vector :

$$\underset{\sim}{\phi}_K = (A_k \ , \ B_k \ , \ C_K) \quad \text{for} \quad k=D.$$

The curve fitting method taken to find the constants A, B, C is the same optimization technique used in chapter 3, with $W_K(\underset{\sim}{\phi}_K, M)$ as the approximating function. The least pth method is used to handle the constraints and the simplex technique is used to perform the minimization part.

4.2 STARTING POINTS

To start the process of optimization, a starting point is needed. To find a starting point $\underset{\sim}{\Phi}_0$, three points are picked from the thirty sample points to set up three equations. The three equations are then solved simultaneously to find a starting point. The following is an example, three points are picked from sample data of relay IAC77 at D=0.5.

(IAC77, D=0.5) M1 = 2.1 M2 = 15.0 M3 = 37.0

W1 = 0.77 W2 = 16.0 W3 = 35.0

Substituting these three points into equation 4.1, one can have the following equations :

$$0.77 = \frac{C^\circ (2.1-1)^{B^\circ}}{A^\circ + (2.1-1)^{B^\circ}} \qquad \text{(eq. 4.4)}$$

$$16 = \frac{C^\circ (15-1)^{B^\circ}}{A^\circ + (15-1)^{B^\circ}} \qquad \text{(eq. 4.5)}$$

$$37 = \frac{C^\circ (37-1)^{B^\circ}}{A^\circ + (37-1)^{B^\circ}} \qquad \text{(eq. 4.6)}$$

Combining equation 4.4, 4.5, 4.6 yields :

$$56.03(2.57)^{B^\circ} - (32.73)^{B^\circ} - 55.03 = 0 \qquad \text{(eq. 4.7)}$$

Therefore B° can be found by solving equation 4.7. The approach that was taken to solve for B° in equation 4.7 is to treat equation 4.7 as a function of B°, i.e.

$$F(B^\circ) = 56.03(2.57)^{B^\circ} - (32.73)^{B^\circ} - 55.03 \qquad \text{(eq. 4.8)}$$

and then apply an optimization method to minimize $\left|F(B^\circ)\right|$. The desired value of B° will be found when the exit criteria is met, and the initial point can be found easily once B° is obtained.

The starting point $\underset{\sim}{\phi}^\circ$ of the example shown above was found to be :

$$\underset{\sim}{\phi}^\circ = (A^\circ, B^\circ, C^\circ)$$

where $A^\circ = 42.068634$

$B^\circ = 1.3875494$

$C^\circ = 32.293488$

A single variable optimization method called the Golden Section Search was used to find B°. For a description of this method, please refer to appendix F.

4.3 AN IMPROVED MODEL

The model suggested by equation 4.3 was found to be only satisfactory in representing the dynamic curves of relay CO-8. It performs poorly for CO-11, IAC51 as well as IAC77. Tables 4.1, 4.2, 4.3 and 4.4 show the percentage errors resulted from this model. The excessive percentage errors suggest that a better model is needed.

The idea of improving the performance of equation 4.3 is to develop a power series based on equation 4.3. The following equation is recommended and investigated.

$$Q_K = a_K^1 W_K(1, \underset{\sim}{\Phi}_K, M) + a_K^2 W_K(2, \underset{\sim}{\Phi}_K, M) + a_K^3 W_K(3, \underset{\sim}{\Phi}_K, M) + a_K^4 W_K(4, \underset{\sim}{\Phi}_K, M) \qquad \text{(eq. 4.9)}$$

$$\text{where} \quad W_K(J, \underset{\sim}{\Phi}_K, M) = \left[\frac{C_K (M-1)^{B_K}}{A_K + (M-1)^{B_K}}\right]^{(J-1)} \qquad \text{(eq. 4.10)}$$

$$J = 1, 2, 3, 4.$$

Coefficients of equation 4.9, namely a_K^1, a_K^2, a_K^3 and a_K^4, can be computed via any suitable curve fitting method. The criterion chosen for this purpose is the well known least error squares method. A digital computer program named IFLSQ was prepared at the University of Manitoba; this program receives the basis functions $W_K(J, \underset{\sim}{\Phi}_K, M)$ and sample data. It then computes the coefficients (a_K's). A description of the algorithm is given in appendix G.

A,B,C = 17.29149 1.39777 7.52718

MX = -0.013820

M	ORIG W	APPX W	ERROR	%ERROR
1.393282	0.115235	0.116286	0.001051	0.911850
1.609759	0.197492	0.211885	0.014393	7.287752
2.129146	0.464773	0.482775	0.018001	3.873178
2.646770	0.788445	0.783224	-0.005221	-0.662191
3.191483	1.141590	1.111009	-0.030581	-2.678848
4.228885	1.799254	1.726559	-0.072696	-4.040324
5.317500	2.471760	2.324408	-0.147352	-5.961429
6.284144	2.959590	2.800669	-0.158921	-5.369704
7.350065	3.433131	3.265074	-0.168057	-4.895164
8.332455	3.777263	3.640527	-0.136736	-3.619973
9.447052	4.069121	4.012332	-0.056789	-1.395617
10.488870	4.337108	4.314616	-0.022491	-0.518582
11.892490	4.621895	4.663693	0.041799	0.904360
13.626130	4.928380	5.019899	0.091519	1.856986
15.613980	5.144021	5.349243	0.205222	3.989527
16.983154	5.311475	5.537777	0.226302	4.260627
18.852615	5.543080	5.755696	0.212616	3.835700
20.716232	5.723799	5.936756	0.212957	3.720559
23.246704	5.910981	6.138181	0.227200	3.843685
25.814056	6.103989	6.302847	0.198858	3.257841
28.365814	6.302998	6.436725	0.133727	2.121642
30.846054	6.439883	6.545008	0.105124	1.632397
33.194702	6.510372	6.631949	0.121577	1.867438
35.720474	6.651436	6.711926	0.060490	0.909422
38.440277	6.724245	6.785455	0.061210	0.910283
41.365173	6.869943	6.852849	-0.017094	-0.248818
43.590591	6.944475	6.897505	-0.046970	-0.676370
46.417892	7.094606	6.947440	-0.147166	-2.074339
48.983551	7.133738	6.987228	-0.146509	-2.053750
51.549194	7.173299	7.022603	-0.150696	-2.100787

TABLE 4.1

%error found by using equation 4.3. Relay : CO-8, D=5

A,B,C = 99.50446 1.68417 38.96318

MX = -0.000777

M	ORIG W	APPX W	ERROR	%ERROR
1.446908	0.112301	0.100600	-0.011701	-10.419226
1.700918	0.210742	0.214041	0.003299	1.565219
2.214423	0.525114	0.535665	0.010551	2.009365
2.744842	0.916247	0.974911	0.058664	6.402588
3.267174	1.380306	1.494586	0.114280	8.279300
4.357134	2.513072	2.794531	0.281459	11.199790
5.349176	3.867074	4.158868	0.291794	7.545596
6.306211	5.301485	5.576804	0.275319	5.193244
7.360576	6.896769	7.199624	0.302855	4.391266
8.329641	8.512248	8.707743	0.195495	2.296627
9.527273	10.289480	10.551165	0.261684	2.543222
10.563300	12.049920	12.100731	0.050811	0.421669
11.832940	14.411390	13.917836	-0.493554	-3.424749
13.823180	17.062714	16.548111	-0.514603	-3.015948
15.655000	19.368027	18.716553	-0.651474	-3.363657
17.372162	21.301224	20.532837	-0.768387	-3.607242
19.069916	22.939957	22.135895	-0.804062	-3.505071
20.721451	24.702393	23.527542	-1.174850	-4.756019
23.478577	26.610367	25.527802	-1.082565	-4.068209
25.785217	28.063934	26.936829	-1.127106	-4.016207
28.619141	29.291534	28.397766	-0.893768	-3.051285
31.109512	30.248795	29.479507	-0.769287	-2.543199
33.826035	31.237350	30.484085	-0.753265	-2.411425
36.004684	31.583984	31.180099	-0.403885	-1.278765
38.733963	32.273544	31.937866	-0.335678	-1.040102
41.240234	32.631683	32.540100	-0.091583	-0.280657
43.908661	32.993820	33.099350	0.105530	0.319847
46.263046	33.356812	33.533539	0.176727	0.529809
48.743683	33.723770	33.940002	0.216232	0.641187
51.902557	33.743027	34.393524	0.650497	1.927797

TABLE 4.2

%error found by using equation 4.3. Relay : CO-11, D=5

A,B,C = 6.27389 0.76627 8.33802

MX = 0.026551

M	ORIG W	APPX W	ERROR	%ERROR
1.119671	0.198396	0.253289	0.054893	27.668304
1.235863	0.370518	0.417360	0.046842	12.642413
1.352247	0.573671	0.557487	-0.016184	-2.821136
1.527974	0.809566	0.742142	-0.067424	-8.328425
1.781983	1.107386	0.972376	-0.135010	-12.191736
2.037389	1.350847	1.174388	-0.176459	-13.062859
2.531386	1.719088	1.508880	-0.210208	-12.227875
3.050763	1.991762	1.805366	-0.186397	-9.358376
4.037830	2.382138	2.267183	-0.114955	-4.825704
5.074326	2.733234	2.656860	-0.076374	-2.794273
6.118927	2.975291	2.982977	0.007686	0.258316
7.152373	3.171250	3.257996	0.086745	2.735364
8.102529	3.379158	3.478868	0.099710	2.950720
9.084425	3.563125	3.681531	0.118406	3.323102
10.079510	3.756755	3.865245	0.108490	2.887864
12.282050	4.089842	4.211627	0.121785	2.977747
15.281720	4.453296	4.586813	0.133517	2.998167
17.677155	4.744824	4.830265	0.085441	1.800712
20.239517	4.953725	5.050684	0.096959	1.957296
22.694366	5.169397	5.232029	0.062632	1.211583
25.185013	5.338172	5.392541	0.054369	1.018494
27.661331	5.454940	5.533225	0.078285	1.435124
30.375366	5.691364	5.669770	-0.021594	-0.379418
33.015472	5.815310	5.787977	-0.027332	-0.470006
35.146713	5.879400	5.874550	-0.004850	-0.082498
37.801132	6.069640	5.972933	-0.096707	-1.593295
40.241333	6.136532	6.055387	-0.081144	-1.322315
43.280502	6.335093	6.148909	-0.186184	-2.938930
47.531525	6.541312	6.265343	-0.275970	-4.218870
50.595093	6.682507	6.340499	-0.342008	-5.117954

TABLE 4.3

%error found by using equation 4.3. Relay : IAC51, D=5

A,B,C = 119.47008 1.82242 62.61217

MX = 0.345688

M	ORIG W	APPX W	ERROR	%ERROR
1.274401	0.168038	0.049609	-0.118429	-70.477554
1.592821	0.312615	0.201452	-0.111163	-35.558975
1.868314	0.503197	0.402570	-0.100627	-19.997589
2.121762	0.665245	0.639559	-0.025686	-3.861085
2.622756	1.174823	1.241287	0.066454	5.657399
3.173832	1.871662	2.085703	0.214041	11.435860
3.681067	2.660346	3.009947	0.349601	13.141172
4.136233	3.553988	3.942894	0.388906	10.942786
4.646039	4.507890	5.087106	0.579216	12.848937
5.163496	5.543055	6.338115	0.795050	14.343357
6.116539	8.211359	8.820730	0.609371	7.421070
7.166844	11.313130	11.726337	0.413207	3.652454
8.219102	14.199560	14.709953	0.510393	3.594429
9.039840	17.461487	17.031708	-0.429779	-2.461296
10.150440	21.028992	20.106491	-0.922501	-4.386803
11.638290	25.585754	24.021912	-1.563843	-6.112162
13.481120	30.174194	28.451263	-1.722931	-5.709948
15.289410	34.144257	32.302658	-1.841599	-5.393581
16.982620	37.061493	35.472595	-1.588898	-4.287192
18.462723	39.840851	37.922729	-1.918121	-4.814458
20.284836	41.934723	40.569702	-1.365021	-3.255108
21.825623	43.689407	42.526642	-1.162766	-2.661435
23.238434	45.051056	44.122391	-0.928665	-2.061361
25.001846	46.451874	45.881470	-0.570404	-1.227945
27.753052	47.886124	48.192184	0.306061	0.639143
30.167496	48.358093	49.865402	1.507309	3.116973
32.794250	49.343842	51.386734	2.042892	4.140116
34.912155	49.837173	52.428986	2.591812	5.200560
37.558731	50.856644	53.543503	2.585859	5.283201
39.984314	51.365112	54.414215	3.049103	5.936135

TABLE 4.4

%error found by using equation 4.3. Relay : IAC77,D=5

4.4 THE COMPUTER PROGRAM

A computer program was written to receive thirty sample points of a relay dynamic curve and produce all the coefficients of equations 4.9 and 4.10.

The program starts with picking up three arbitrary points from the sample data and using them as inputs to set up an equation similar to equation 4.8. The single variable optimization method, namely the golden section search, is then applied to find B° and hence A° and C°. The point $\underset{\sim}{\Phi}^\circ = (A^\circ, B^\circ, C^\circ)$ is then used as a starting point. The main program can call the search routine now to search for values of A_K, B_K, and C_K at which equation 4.3 will best fit the dynamic curve. The approximating curve expressed by equation 4.3 will be further improved by incorporating it into a polynomial as described by equations 4.9 and 4.10. To find the coefficients a_K^1, a_K^2, a_K^3 and a_K^4, the subroutine IFLSQ is called to apply the least error squares method to compute these coefficients.

The same constraint handling technique and optimization method, namely the least pth approximation technique and the simplex search method, are used again here and they once again proved to be very effective. Figure 4.5 shows the flow chart of the main program and the program listing can be found in appendix H.

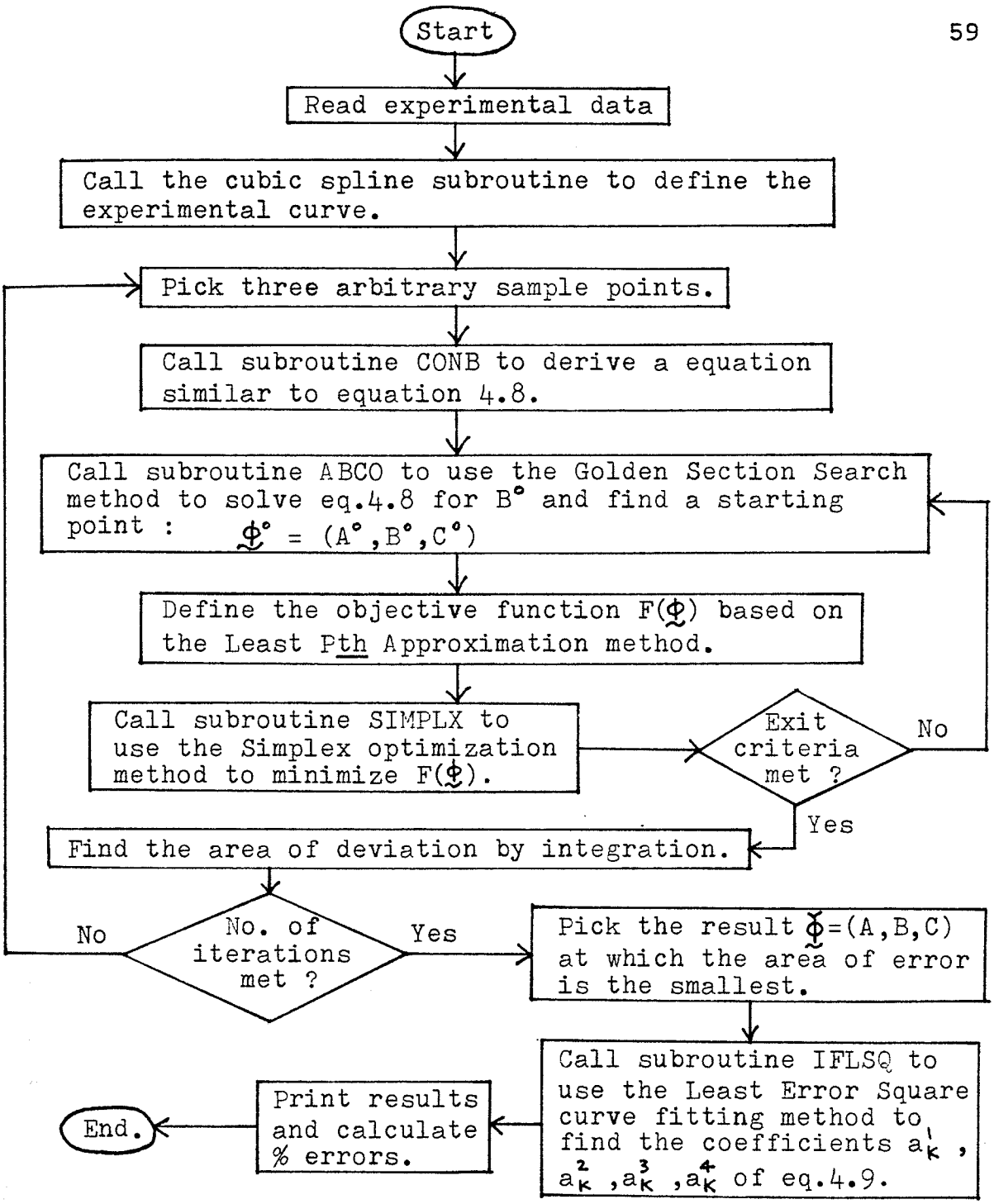


Figure 4.1: Flow chart of program for finding coefficients of the non-linear model (equation 4.9).

4.5 THE RESULTS

The results produced by equation 4.9 indicated that it is superior to equation 4.3. Although the dynamic curves of the CO-8 can be represented adequately by equation 4.3, equation 4.9 can improve the percentage errors by a great deal. Using the IAC77 and D=5, a comparison between the two results is shown in table 4.5.

A series consisting of only three terms was investigated in hopes that a smaller number of coefficients would be adequate to delineate the dynamic curves. Deleting the last term from equation 4.9, we get the following equation :

$$Q_K=a_K^1 W_K(1,\underset{\sim}{\phi}_K,M)+a_K^2 W_K(2,\underset{\sim}{\phi}_K,M)+a_K^3 W_K(3,\underset{\sim}{\phi}_K,M) \qquad \text{(eq. 4.11)}$$

However, the performance of this equation is not satisfactory. A comparison shows that the last term in equation 4.9 has a significant effect and therefore equation 4.9 is recommended. In tables 4.6 and 4.7, a comparison is given to show the effect of the last term of equation 4.9.

Some typical percentage errors produced by equation 4.9 are listed in tables 4.8 and 4.9. It is noticed that the percentage errors at M<2 are large, but errors in the vicinity of M=2 or M<2 are not significant in actual relay applications.

Tables 4.10, 4.11, 4.12, 4.13 summarize the values of the coefficients in equation 4.9 and 4.10 for all the time dial settings of the four different relays. All these values are computed by the computer program described in section 4.4.

A,B,C = 119.47008 1.82242 62.61217

A1,A2,A3,A4 =

0.053273 0.834564 0.014136 -0.000226

M	(eq. 4.3) %ERROR	(eq. 4.9) %ERROR
1.274401	-70.477554	-43.638184
1.592821	-35.558975	-28.995956
1.868314	-19.997589	-22.193710
2.121762	-3.861085	-10.897654
2.622756	5.657399	-5.470532
3.173832	11.435860	-0.977438
3.681067	13.141172	1.008396
4.136233	10.942786	-0.118257
4.646039	12.848937	2.816792
5.163496	14.343357	5.594931
6.116539	7.421070	1.804782
7.166844	3.652454	0.937545
8.219102	3.594429	3.308962
9.039840	-2.461296	-1.200936
10.150440	-4.386803	-1.507593
11.638290	-6.112162	-1.793548
13.481120	-5.709948	-0.452188
15.289410	-5.393581	0.010636
16.982620	-4.287192	0.809721
18.462723	-4.814458	-0.324855
20.284836	-3.255108	0.378425
21.825623	-2.661435	0.105510
23.238434	-2.061361	-0.131687
25.001846	-1.227945	-0.363010
27.753052	0.639143	-0.139822
30.167496	3.116973	0.932034
32.794250	4.140116	0.546014
34.912155	5.200560	0.546182
37.558731	5.283201	-0.527402
39.984314	5.936135	-0.857776

TABLE 4.5

Comparison of %errors produced by equation 4.3 and 4.9.
Relay : IAC77, D=5

A,B,C = 17.29149 1.39777 7.52718

A1,A2,A3,A4 = -0.167412 1.441585 -0.158246 0.014221

A1,A2,A3 = 0.054856 0.982351 0.000194

M	(eq. 4.11) %ERROR	(eq. 4.9) %ERROR
1.393282	46.73633	-101.64365
1.609759	33.17474	-33.63370
2.129146	13.85232	6.13041
2.646770	4.55715	10.52505
3.191483	0.42956	10.22969
4.228885	-2.65306	6.87921
5.317500	-5.35950	1.42691
6.284144	-5.13499	-0.62297
7.350065	-4.91566	-2.49535
8.332455	-3.80072	-2.85093
9.447052	-1.71112	-2.00019
10.488870	-0.92636	-2.03539
11.892490	0.40156	-1.41763
13.626130	1.27145	-0.97247
15.613980	3.32841	0.94498
16.983154	3.56519	1.25204
18.852615	3.10856	1.01162
20.716232	2.96771	1.14252
23.246704	3.06250	1.64052
25.814056	2.46025	1.45831
28.365814	1.31700	0.71172
30.846054	0.81940	0.56321
33.194702	1.04309	1.08805
35.720474	0.08445	0.42212
38.440277	0.07779	0.70032
41.365173	-1.07837	-0.19215
43.590591	-1.50664	-0.43876
46.417892	-2.89760	-1.63403
48.983551	-2.88082	-1.44435
51.549194	-2.93065	-1.33788

TABLE 4.6

Comparison of %errors produced by equation 4.9 and 4.11.
Relay : CO-8, D=5

A,B,C = 119.47008 1.82242 62.61217

A1,A2,A3,A4 = 0.053273 0.834564 0.014136 -0.000226

A1,A2,A3 = -0.840558 1.193598 -0.003986

M	(eq. 4.11) %ERROR	(eq. 4.9) %ERROR
1.274401	-564.98755	-43.638184
1.592821	-292.01465	-28.995956
1.868314	-171.68092	-22.193710
2.121762	-111.84702	-10.897654
2.622756	-45.95799	-5.470532
3.173832	-12.82658	-0.977438
3.681067	2.09182	1.008396
4.136233	7.02635	-0.118257
4.646039	13.76163	2.816792
5.163496	18.42709	5.594931
6.116539	14.20422	1.804782
7.166844	11.44466	0.937545
8.219102	11.65634	3.308962
9.039840	4.98647	-1.200936
10.150440	2.46373	-1.507593
11.638290	-0.21070	-1.793548
13.481120	-0.93426	-0.452188
15.289410	-1.72093	0.010636
16.982620	-1.55838	0.809721
18.462723	-2.88451	-0.324855
20.284836	-2.17438	0.378425
21.825623	-2.24051	0.105510
23.238434	-2.19078	-0.131687
25.001846	-1.97890	-0.363010
27.753052	-0.96448	-0.139822
30.167496	0.84649	0.932034
32.794250	1.26758	0.546014
34.912155	1.89594	0.546182
37.558731	1.54341	-0.527402
39.984314	1.83212	-0.857776

TABLE 4.7

Comparison of %errors produced by equation 4.9 and 4.11.
Relay : IAC77, D=5

A,B,C = 99.50446 1.68417 38.96318

A1,A2,A3,A4 = 0.077857 0.828792 0.018909 -0.000418

M	ORIG W	APPX W	ERROR	%ERROR
1.446908	0.112301	0.161424	0.049124	43.742950
1.700918	0.210742	0.256115	0.045372	21.529785
2.214423	0.525114	0.527173	0.002060	0.392284
2.744842	0.916247	0.903440	-0.012807	-1.397780
3.267174	1.380306	1.357399	-0.022907	-1.659577
4.357134	2.513072	2.532480	0.019408	0.772291
5.349176	3.867074	3.821654	-0.045420	-1.174522
6.306211	5.301485	5.215392	-0.086093	-1.623940
7.360576	6.896769	6.868863	-0.027905	-0.404616
8.329641	8.512248	8.452310	-0.059938	-0.704143
9.527273	10.289480	10.436242	0.146762	1.426329
10.563300	12.049920	12.134357	0.084437	0.700729
11.832940	14.411390	14.147739	-0.263651	-1.829461
13.823180	17.062714	17.074982	0.012268	0.071900
15.655000	19.368027	19.470901	0.102875	0.531157
17.372162	21.301224	21.445602	0.144379	0.677795
19.069916	22.939957	23.151398	0.211441	0.921715
20.721451	24.702393	24.595581	-0.106812	-0.432393
23.478577	26.610367	26.597580	-0.012787	-0.048052
25.785217	28.063934	27.945938	-0.117996	-0.420455
28.619141	29.291534	29.281433	-0.010101	-0.034485
31.109512	30.248795	30.224670	-0.024124	-0.079752
33.826035	31.237350	31.062805	-0.174545	-0.558771
36.004684	31.583984	31.620712	0.036728	0.116286
38.733963	32.273544	32.205841	-0.067703	-0.209779
41.240234	32.631683	32.653580	0.021896	0.067101
43.908661	32.993820	33.055115	0.061295	0.185776
46.263046	33.356812	33.357117	0.000305	0.000915
48.743683	33.723770	33.631927	-0.091843	-0.272338
51.902557	33.743027	33.929306	0.186279	0.552053

TABLE 4.8

%errors produced by equation 4.9. Relay : CO-11, D=5

A,B,C = 6.27389 0.76627 8.33802

A1,A2,A3,A4 = -0.182587 1.499941 -0.214229 0.023226

M	ORIG W	APPX W	ERROR	%ERROR
1.119671	0.198396	0.183965	-0.014431	-7.273763
1.235863	0.370518	0.407801	0.037283	10.062420
1.352247	0.573671	0.591054	0.017383	3.030173
1.527974	0.809566	0.822084	0.012518	1.546259
1.781983	1.107386	1.094714	-0.012671	-1.144268
2.037389	1.350847	1.321081	-0.029756	-2.203511
2.531386	1.719088	1.672690	-0.046397	-2.698943
3.050763	1.991762	1.963778	-0.027985	-1.405017
4.037830	2.382138	2.387560	0.005422	0.227595
5.074326	2.733234	2.725922	-0.007313	-0.267550
6.118927	2.975291	3.001955	0.026664	0.896174
7.152373	3.171250	3.233486	0.062236	1.962501
8.102529	3.379158	3.420694	0.041536	1.229191
9.084425	3.563125	3.594853	0.031729	0.890475
10.079510	3.756755	3.755697	-0.001058	-0.028153
12.282050	4.089842	4.069774	-0.020068	-0.490683
15.281720	4.453296	4.431595	-0.021701	-0.487298
17.677155	4.744824	4.681786	-0.063039	-1.328580
20.239517	4.953725	4.920759	-0.032966	-0.665472
22.694366	5.169397	5.127337	-0.042061	-0.813651
25.185013	5.338172	5.318404	-0.019758	-0.370310
27.661331	5.454940	5.492692	0.037752	0.692073
30.375366	5.691364	5.668336	-0.023028	-0.404619
33.015472	5.815310	5.825824	0.010514	0.180803
35.146713	5.879400	5.944503	0.065103	1.107299
37.801132	6.069640	6.082928	0.013288	0.218918
40.241333	6.136532	6.201948	0.065416	1.066014
43.280502	6.335093	6.340364	0.005271	0.083202
47.531525	6.541312	6.517942	-0.023370	-0.357265
50.595093	6.682507	6.635755	-0.046752	-0.699617

TABLE 4.9

%errors produced by equation 4.9. Relay : IAC51, D=5

TDS	11	10	9	8	7
A	18.09709	18.16992	18.04555	17.89734	17.68565
B	1.37901	1.42629	1.37495	1.39379	1.39580
C	6.94451	6.97664	7.36946	7.33908	7.28394
A1	-0.236098	-0.182102	-0.198192	-0.173041	-0.177021
A2	1.625359	1.544777	1.557767	1.502666	1.517089
A3	-0.250449	-0.220914	-0.217780	-0.203328	-0.196429
A4	0.025212	0.022125	0.021126	0.020186	0.018656

TDS	6	5	4	3	2
A	18.15140	17.29149	17.24532	17.90462	15.47383
B	1.40001	1.39777	1.37962	1.46749	1.41182
C	7.48478	7.52718	7.60017	7.33642	7.41377
A1	-0.153112	-0.167412	-0.183747	-0.143101	-0.154400
A2	1.441457	1.441585	1.524651	1.540100	1.405449
A3	-0.165675	-0.158246	-0.204056	-0.228115	-0.153965
A4	0.015338	0.014221	0.019465	0.022869	0.014629

TDS	1	0.5
A	16.88931	13.51051
B	1.35743	1.37005
C	8.43328	9.23443
A1	-0.120720	-0.006208
A2	1.294505	0.908987
A3	-0.098391	-0.040877
A4	0.008222	-0.003409

TABLE 4.10

A summary of the non-linear model coefficients, CO-8.

TDS	11	10	9	8	7
A	126.09720	121.21257	110.89185	98.63873	105.78397
B	1.65280	1.66167	1.67200	1.66223	1.69108
C	44.83141	44.60661	42.48439	40.71584	40.98219
A1	-0.039366	-0.062055	-0.004563	0.054820	0.081924
A2	0.923148	0.960367	0.926934	0.822993	0.850700
A3	0.010984	0.007422	0.010062	0.018227	0.014390
A4	-0.000246	-0.000176	-0.000218	-0.000374	-0.000287

TDS	6	5	4	3	2
A	99.58469	99.50446	95.23657	100.48656	78.86342
B	1.65171	1.68417	1.67292	1.63816	1.67282
C	41.40608	38.96318	39.67676	42.16138	35.35660
A1	0.139786	0.077857	-0.030545	0.155789	0.187914
A2	0.764057	0.828792	0.930639	0.832886	0.739048
A3	0.023050	0.018909	0.008455	0.014605	0.026589
A4	-0.000482	-0.000418	-0.000189	-0.000303	-0.000599

TDS	1	0.5
A	60.56042	51.85222
B	1.57835	1.36545
C	29.54970	34.32295
A1	0.104708	-0.218491
A2	0.887913	1.270957
A3	0.012425	-0.034158
A4	-0.000294	0.000966

TABLE 4.11

A summary of the non-linear model coefficients, CO-11.

TDS	10	9	8	7	6
A	6.74318	7.08752	6.99625	6.58141	6.67272
B	0.79984	0.75034	0.76694	0.79500	0.79880
C	7.73154	8.50984	8.46690	8.26262	8.46484
A1	-0.134187	-0.128684	-0.149324	-0.174547	-0.137702
A2	1.476746	1.416822	1.456946	1.511990	1.454188
A3	-0.231218	-0.190031	-0.198332	-0.233222	-0.209094
A4	0.027314	0.021701	0.021886	0.025910	0.022947

TDS	5	4	3	2	1
A	6.27389	7.57916	6.86438	5.42290	3.85957
B	0.76627	0.74642	0.75545	0.79795	0.84286
C	8.33802	9.79151	9.08887	7.52279	5.58421
A1	-0.182587	-0.119703	-0.152673	-0.140537	-0.074692
A2	1.499941	1.396555	1.460973	1.475412	1.400798
A3	-0.214229	-0.178701	-0.198454	-0.235836	-0.262254
A4	0.023226	0.018935	0.020293	0.027881	0.039216

TDS	0.5
A	2.86762
B	0.78195
C	4.09319
A1	-0.087035
A2	1.367002
A3	-0.312450
A4	0.063373

TABLE 4.12

A summary of the non-linear model coefficients, IAC51.

TDS	10	9	8	7	6
A	118.58736	139.83804	76.58092	117.22205	119.38005
B	1.77378	1.85560	1.41456	1.78282	1.77673
C	57.59761	56.71538	77.45924	60.94965	63.18912
A1	0.214309	0.283418	-0.111659	0.544437	-0.053865
A2	0.639545	0.729506	0.532422	0.507076	0.913601
A3	0.025954	0.018798	0.032393	0.029124	0.008574
A4	-0.000412	-0.000289	-0.000470	-0.000400	-0.000147

TDS	5	4	3	2	1
A	119.47008	115.26006	86.41020	83.56050	65.53865
B	1.82242	1.87595	1.64714	1.70616	1.48896
C	62.61217	57.98434	63.12338	52.73680	48.97885
A1	0.053273	0.291785	-0.286559	-0.218029	-0.215360
A2	0.834564	0.762282	0.931607	1.026611	1.108391
A3	0.014136	0.017783	0.009207	-0.001218	-0.004767
A4	-0.000226	-0.000278	-0.000176	0.000019	0.000054

TDS	0.5
A	43.10860
B	1.41279
C	32.48772
A1	0.019119
A2	0.980973
A3	-0.000956
A4	0.000032

TABLE 4.13

A summary of the non-linear model coefficients, IAC77.

Chapter V

APPLICATION TO RELAY SETTING DETERMINATION AND TRIP SIMULATION

5.1 APPLICATION TO RELAY SETTING DETERMINATION

When applying an overcurrent relay to a protective relay system, it is necessary to select the right time dial setting for the relay so that the desired operating time can be obtained and reliable relay coordination can be achieved. To calculate a time dial setting, a program is needed to function as follows :

It will receive the following information as input,

Input : (1) Relay type (e.g. CO-8).
(2) M (multiple of pickup current).
(3) T (desired operating time).

and it will give the appropriate time dial setting as output,

Output : (1) D (time dial setting).

The dynamic curves of D equal to 0.5, 1, 2, . . . and up to 10 or 11 are assumed to be available from the computer memory. Linear interpolation is suggested to be used to calculate the wanted D, which may not be directly available from the computer memory.

For example, the following input is given,

(1) Relay type is CO-8.

(2) $M = M_f$.

(3) $T = T_{op}$.

The first thing the program will do is to realize the specific relay type wanted and retrieve the right data from the computer storage. Suppose that T_{op} is found to be located between the time-current curves of two adjacent time dial settings, say D_a and D_b.

The situation is illustrated in figure 5.1, where T_a and T_b are the operating time at $M=M_f$ for TDS equal to D_a and D_b respectively. (Notice that the relay characteristics are stored in terms of dynamic behavior, i.e. dynamic curves, T_a and T_b are calcuated from equation 2.5, i.e. $T=D/W$).

Applying linear interpolation, the time dial setting (D) can be calculated as follow :

$$D = D_a+(D_b-D_a)(T_{op}-T_a)/(T_b-T_a) \qquad \text{(eq. 5.1)}$$

A program using the non-linear dynamic curve model was prepared to do this calculation. Figure 5.2 shows the program flow chart, and the program listing is listed in appendix I. Using the CO-8 as an example, several sample calculation results are listed in table 5.1.

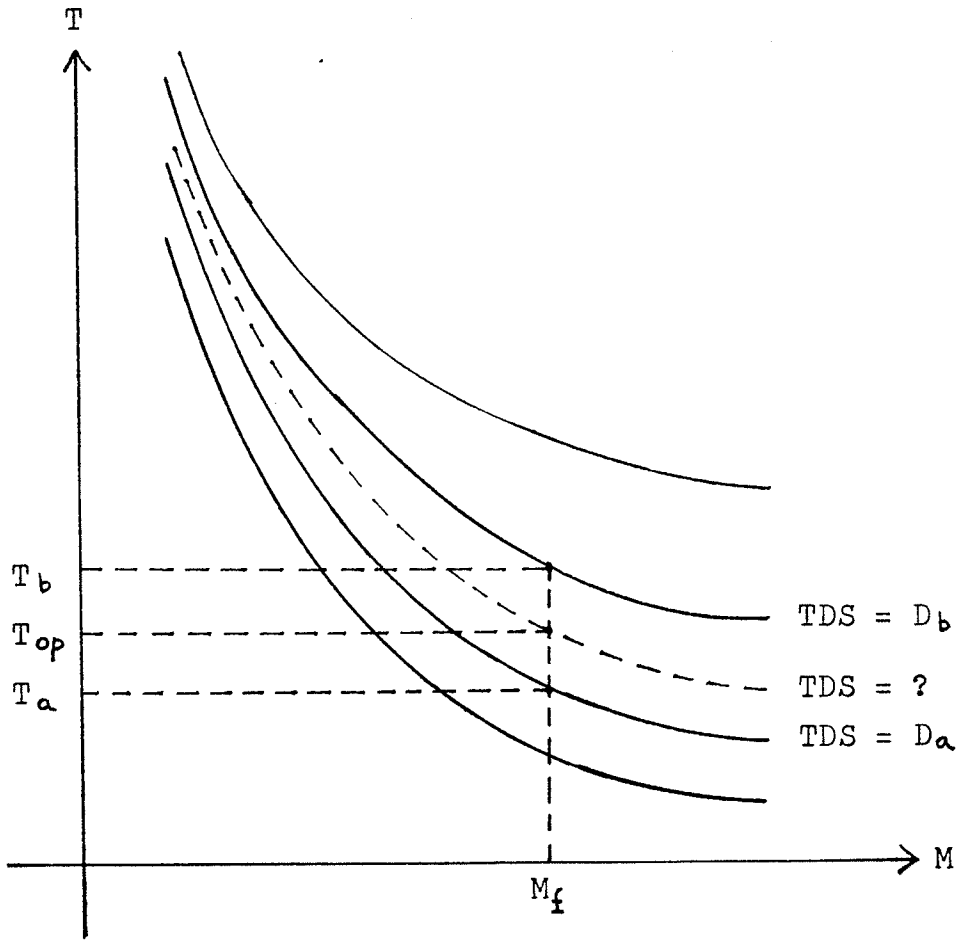


Figure 5.1: Finding TDS (D) by linear interpolation.

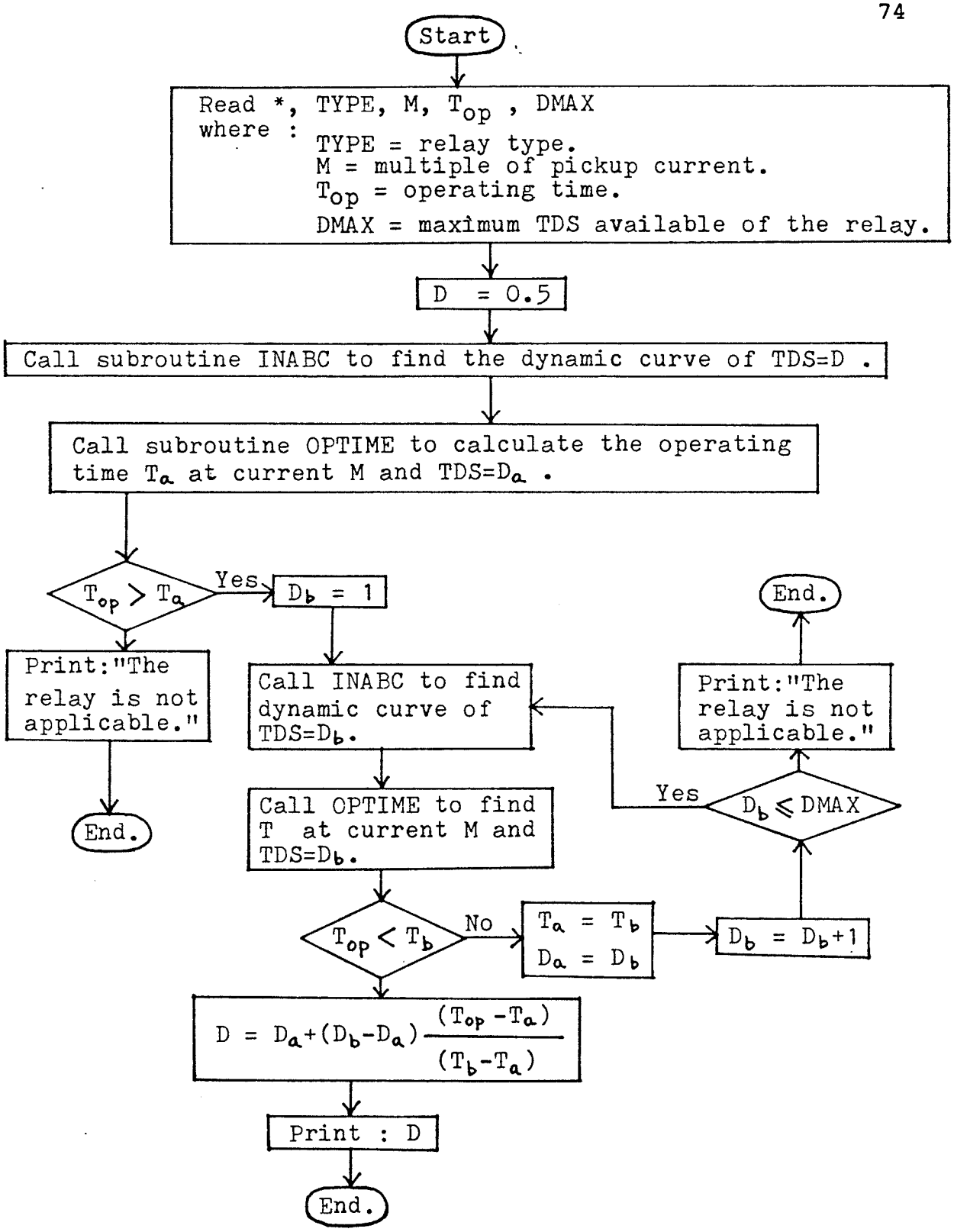


Figure 5.2: Program flow chart for finding relay time dial setting.

Input Information			Output
Relay type	M	T_{op} (seconds)	TDS
CO-8	10	0.3	1.32
CO-8	20	0.4	2.32

TABLE 5.1

Sample calculations of time dial setting determination.

5.2 APPLICATION TO OPERATING TIME CALCULATION

The time dial setting selection switch of a conventional electromechanical overcurrent relay operates on a continuous scale. This implies that there are literally an infinite number of time dial settings for one to choose. However, it is impossible to store all the relay characteristics of an infinite number of time dial settings.

If only the published relay operating characteristics are stored in the computer, then it is necessary to find a way which can produce the desired operating characteristic for

any given time dial setting based on the available information. In other words, it is desirable to have a program to accomplish the following :

Input : (1) Relay type (e.g. IAC51).
(2) $M = M_f$ (multiple of pickup current).
(3) D (time dial setting).

Output : (1) T_{op} (Relay operating time).

Once the specified relay type is realized, the correponding family of dynamic curves will be retrieved from the computer memory. The program will then find the time dial settings D_a and D_b such that $D_a < D < D_b$. D_a and D_b are shown in figure 5.1.

To find T_{op} at $M=M_f$, the method of linear interpolation will be used the same way as in the previous section, except this time we want to calculate T_{op} instead of D.

At $M=M_f$, T_a and T_b will be calculated first by the program, then by applying linear interpolation, T_{op} can be obtained by the following equation :

$$T_{op} = T_a+(T_b-T_a)(D-D_a)/(D_b-D_a) \qquad \text{(eq. 5.2)}$$

A flow chart description of the program is given in figure 5.3. Appendix J gives the program listing. Using the IAC51 as an example, several sample calculations were carried out and the results are listed in table 5.2.

Input Information			Output
Relay type	M	TDS	T_{op} (seconds)
CO-8	8	5.5	1.57
CO-8	20	2.322	0.40

TABLE 5.2

Sample calculations of relay operating time determination.

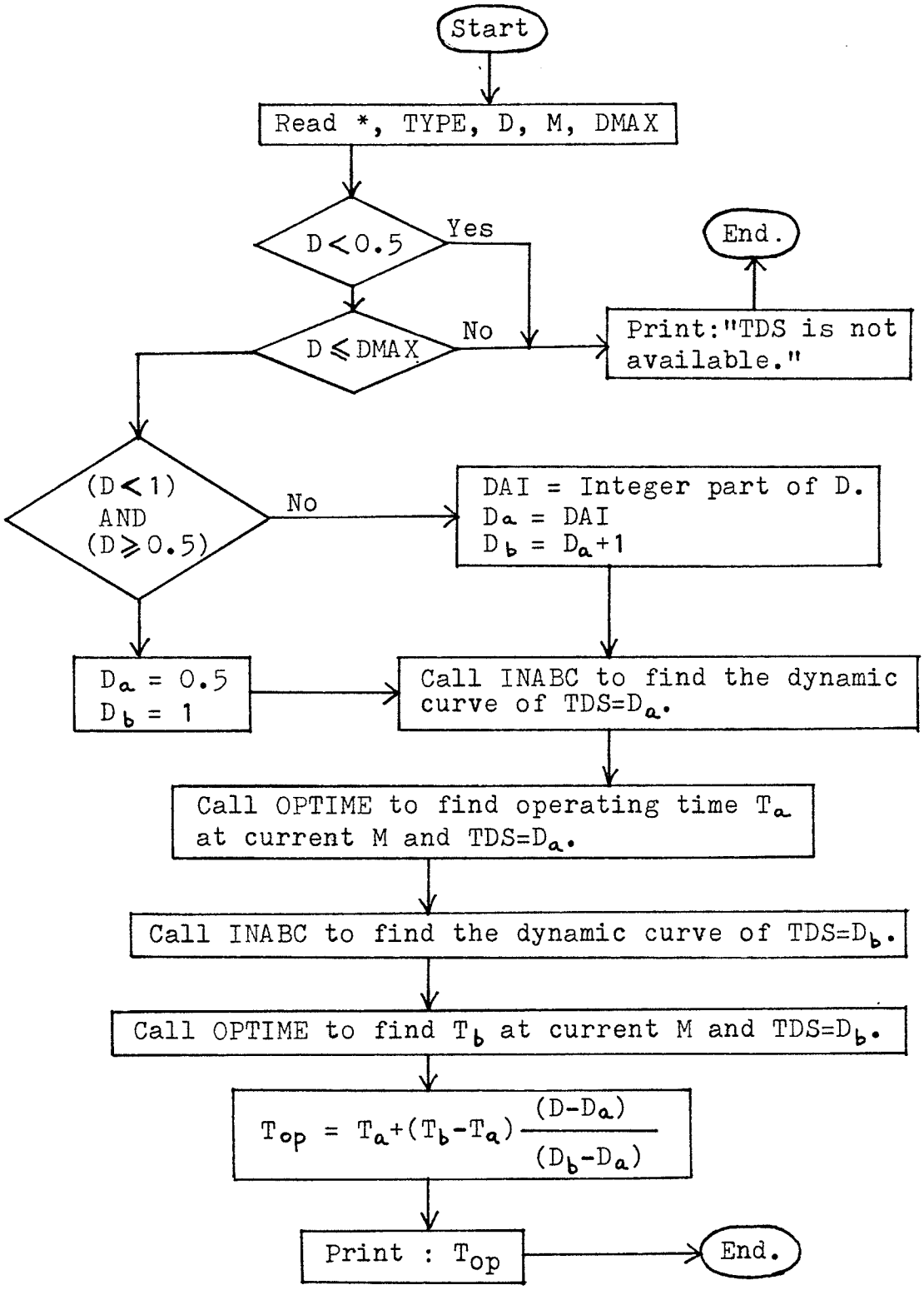


Figure 5.3: program flow chart for finding relay operating time.

5.3 RELAY SIMULATION

The whole idea of the dynamic curves discussed in this thesis is to represent accurately the motion of the disc of an induction disc relay. This section concerns both the operating and reset behavior of such a relay.

5.3.1 Relay disc reset time

On the surface, fast reset of overcurrent relays appears to be a highly desirable quality, but when one considers the fact that conductors, transformers and fuses do not reset fast thermally, one needs to exercise caution in the use of fast reset overcurrent devices.

The reset action of several types of relays are shown in figure 5.4. This figure is published by the Westinghouse Electric Corp.

Although the curves shown in figure 5.4 are not exactly linear curves, they are close enough to linear that an acceptable approximation can be obtained.

When simulating the reset action, the reset time (T_R) can be computed by adding a small time increment (ΔT_R) to T_R and the disc backward action can be simulated by decrementing a small change of angular distance ($\Delta\Theta$) from the angle (Θ) at which the disc start resetting, i.e.

$$T_R = T_R + \Delta T_R \qquad \text{(eq. 5.3)}$$

$$\Theta = \Theta - \Delta\Theta$$

If we recall equation 2.3, the time dial setting (D) was also defined as the maximum angle that the disc can travel; hence T can be expressed as :

$$\Delta T_R = (T_{FR}/D)\Delta\theta \qquad \text{(eq. 5.4)}$$

where T_{FR} is the total reset time from the fully closed position (refer to figure 5.4).

Using the CO-8 as an example and referring to figure 5.4, the reset time is 70 seconds for for D=10.

$$\Delta T_R = (70/10)\Delta\theta$$

A flow chart describing the algorithm of reset time simulation is given in figure 5.5.

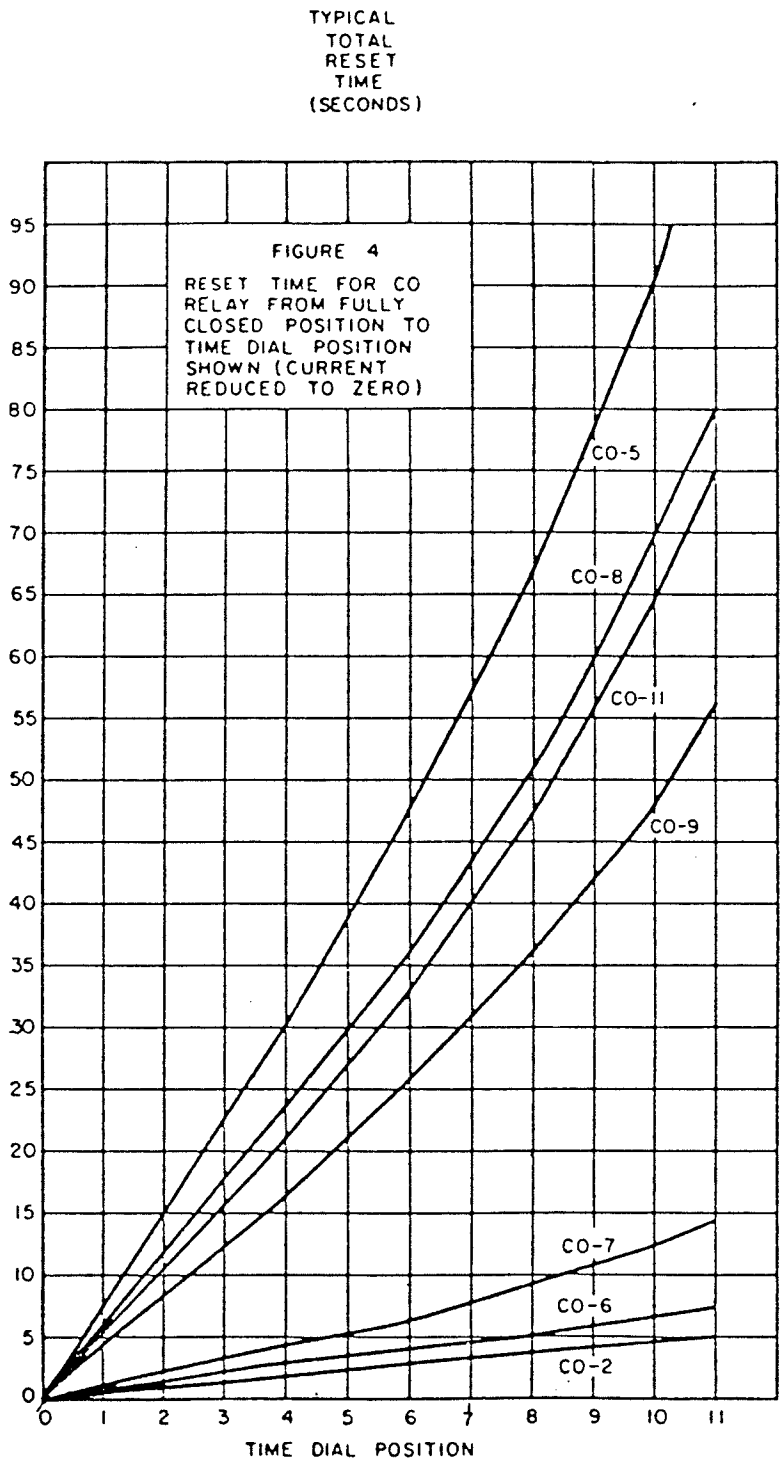


Figure 5.4: Reset actions of overcurrent relays.

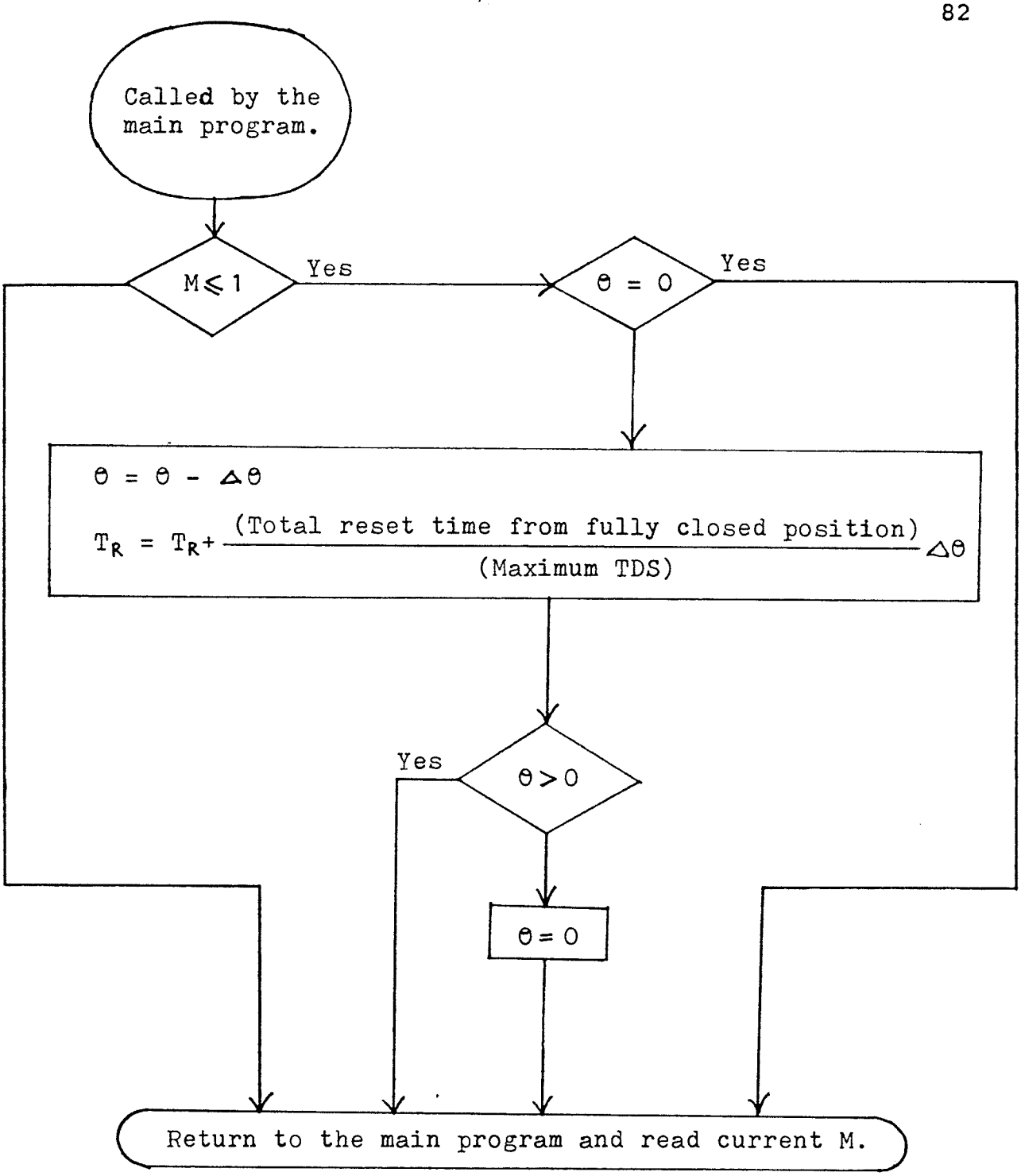


Figure 5.5: Induction disc relay reset action simulation.

5.3.2 Relay operation simulation

The simulation of relay operation includes the disc forward motion and the disc reset motion. Opposite to the reset motion, the disc forward motion can be simulated by adding an incremental angle ($\Delta\theta$) to θ, and $\Delta\theta$ depends on the disc angular velocity (W), i.e.

$$\Delta\theta = W \cdot \Delta T$$
$$\theta = \theta + \Delta\theta$$

where ΔT is a small time increment.

and the operate time (T_{op}) can be computed by adding ΔT to T_{op} :

$$T_{op} = T_{op} + \Delta T$$

Using the non-linear model (eq. 4.9) to represent the dynamic curves, a complete algorithm was developed and is shown in figure 5.6.

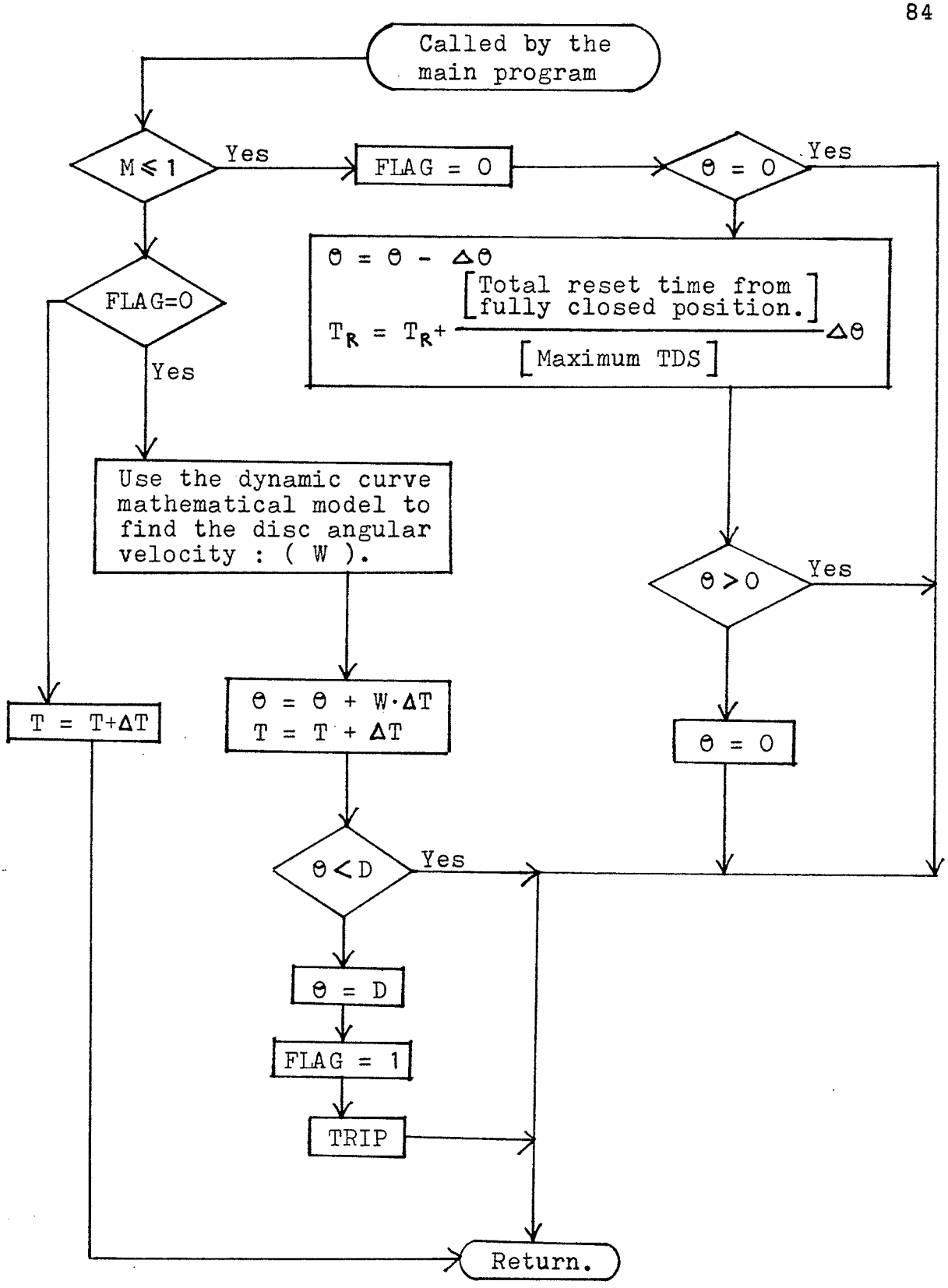


Figure 5.6: Induction disc overcurrent relay operation simulation.

5.4 TRIP SIMULATION

Three different trip simulations will be performed in order to show the feasibility of the OC relay dynamic curve models described in the previous sections. Since the principle of applying the piece-wise linear approximation model and the non-linear continuous model of the OC dynamic curves to relay operation simulation is the same, only the non-linear model will be used in the following test cases, i.e. equation 4.9 will be used. The CO-8 overcurrent relay will be used as our example and the time dial setting is set equal to 6 for all three cases.

5.4.1 Test case one

In this test, the fault current will stay constant until the fault is cleared. Circuit breaker time is considered and is assumed to last about six cycles or approximately equal to 0.1 seconds. Relay trip time, fault clearance time, and the time required by the disc to reset are calculated.

The input information and the output of the program are listed in figure 5.7, and a graphical illustration of the operation is also given. The algorithm is explained via a flow chart shown in figure 5.8, and the program listing can be found in appendix K.

The fault current (M_F) is set equal to 10.5 times the pickup current. The relay operating time (T_{op}) and reset time (T_R) can be predicted as follows :

T_{op} = 1.41 seconds (refer to figure 1.2)

T_R = 42.0 seconds (refer to figure 5.4)

The test program, using the relay simulation model shown in figure 5.6, received the fault current M_F and produced the following :

T_{op} = 1.45187 seconds

T_R = 41.9408 seconds

The relay operating time (T_{op}) has a deviation of about 3%, which implies that the results are very close to those predicted values.

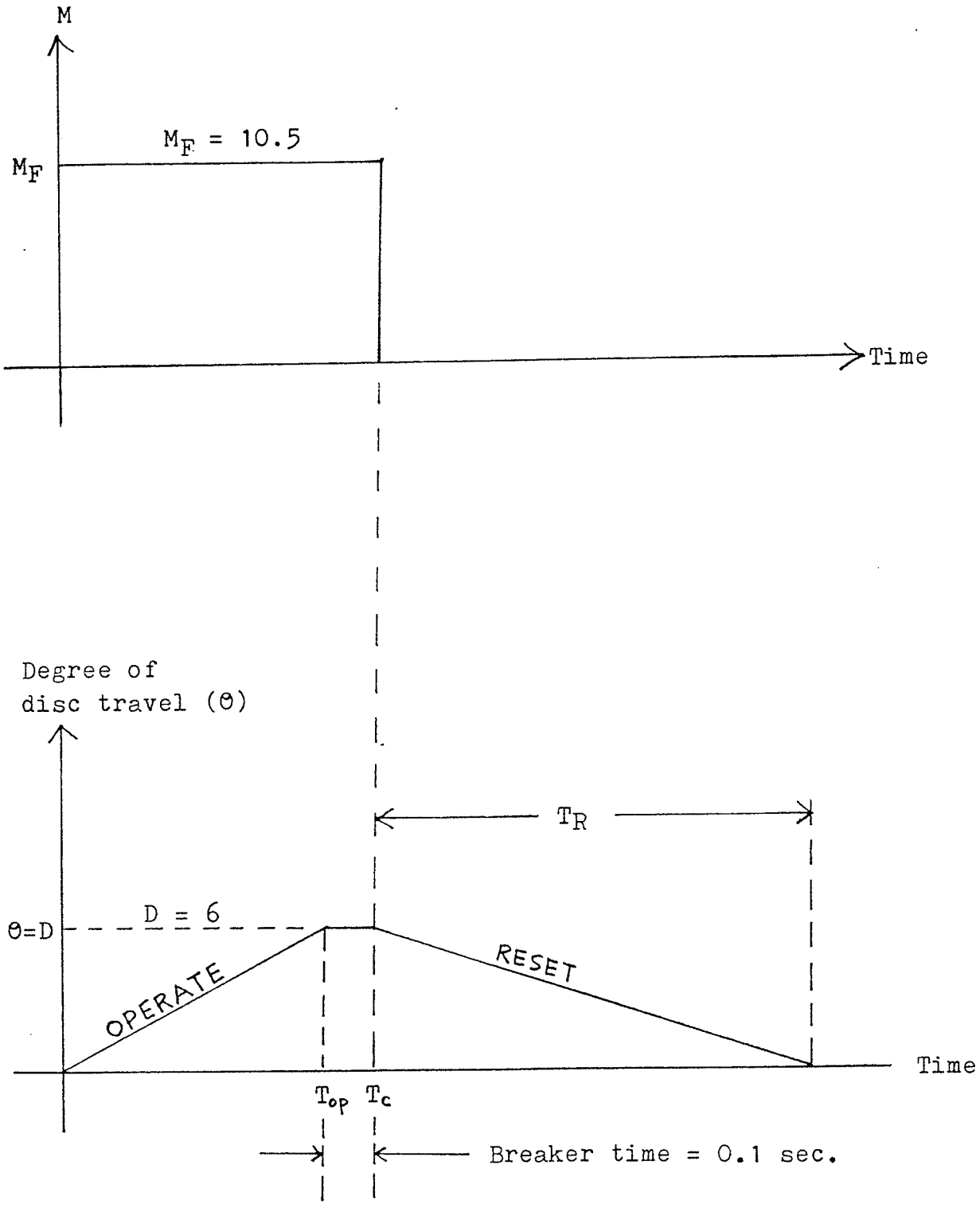


Figure 5.7: Trip simulation one : fault current is stable and C.B. action is included.

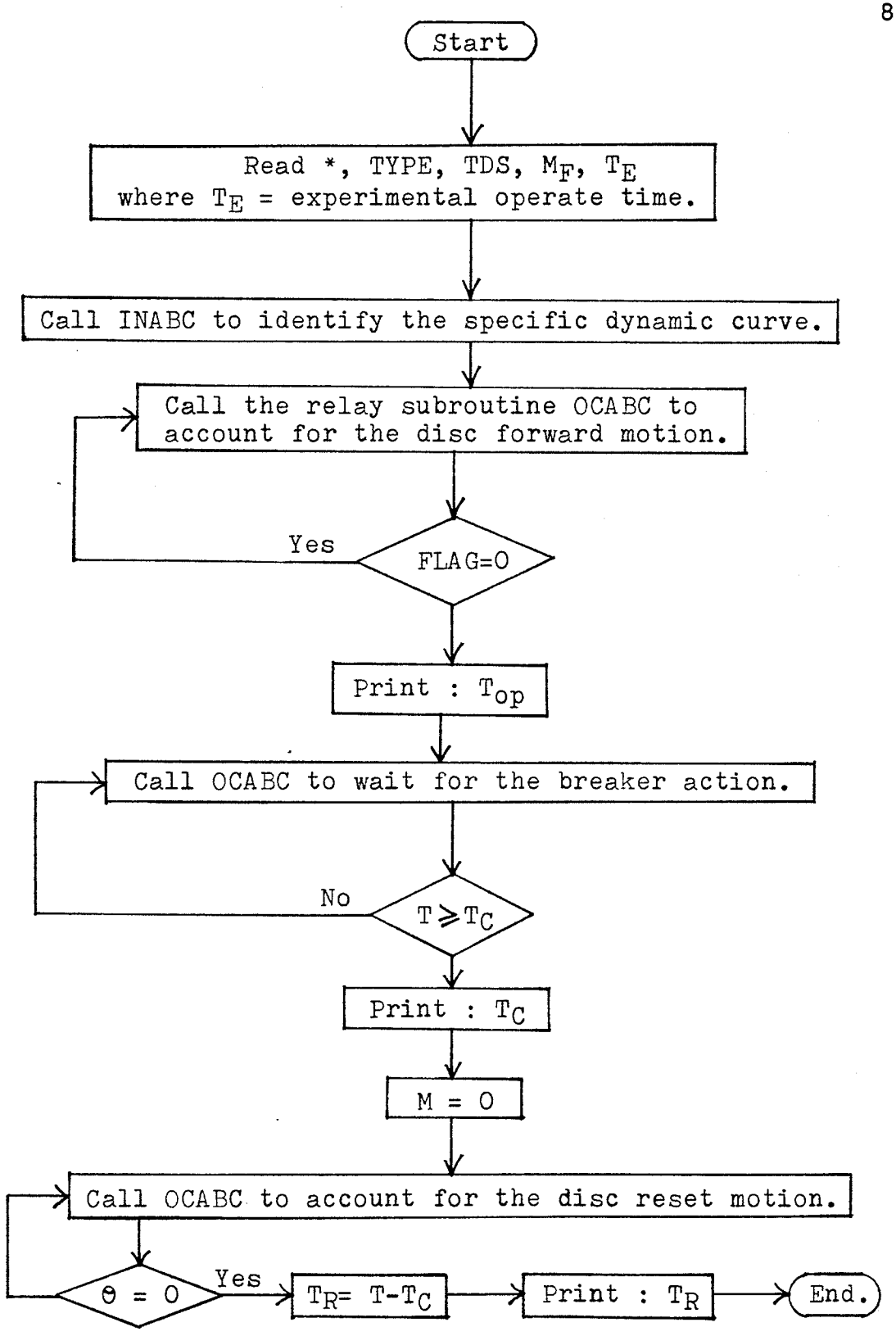


Figure 5.8: Program flow chart for trip simulation one.

5.4.2 Test case two

The second test case consists of increasing the fault current suddenly while the relay is operating. In this case, the C.B. action and the relay reset action are not included. Only the forward action of the induction disc unit is of interest here.

The input and output information of the program are shown in figure 5.9. Again, a graphical interpretation of the simulation is given in the same figure. Figure 5.10 shows the algorithm flow chart and the program listing is in appendix L.

We first let the initial fault current (M_1) be 5.0. The fault current is then increased to M_2=9.0 after 0.5 seconds. Referring to figure 1.2, the OC relay operating time at M=5.0 and M=9.0 are predicted as follows, with the relay initially reset :

T_{op1} = 1.5 seconds , at M=5.0

T_{op2} = 2.5 seconds , at M=9.0

The test program received values of M_1, M_2 and the time at which the magnitude of the current changed. The operating time of the OC relay under such an unstable fault current was found to be 1.8 seconds.

The calculated T_{op} is between the values of T_{op1} and T_{op2} as is expected. the result indicates that the relay simulation model performs very well.

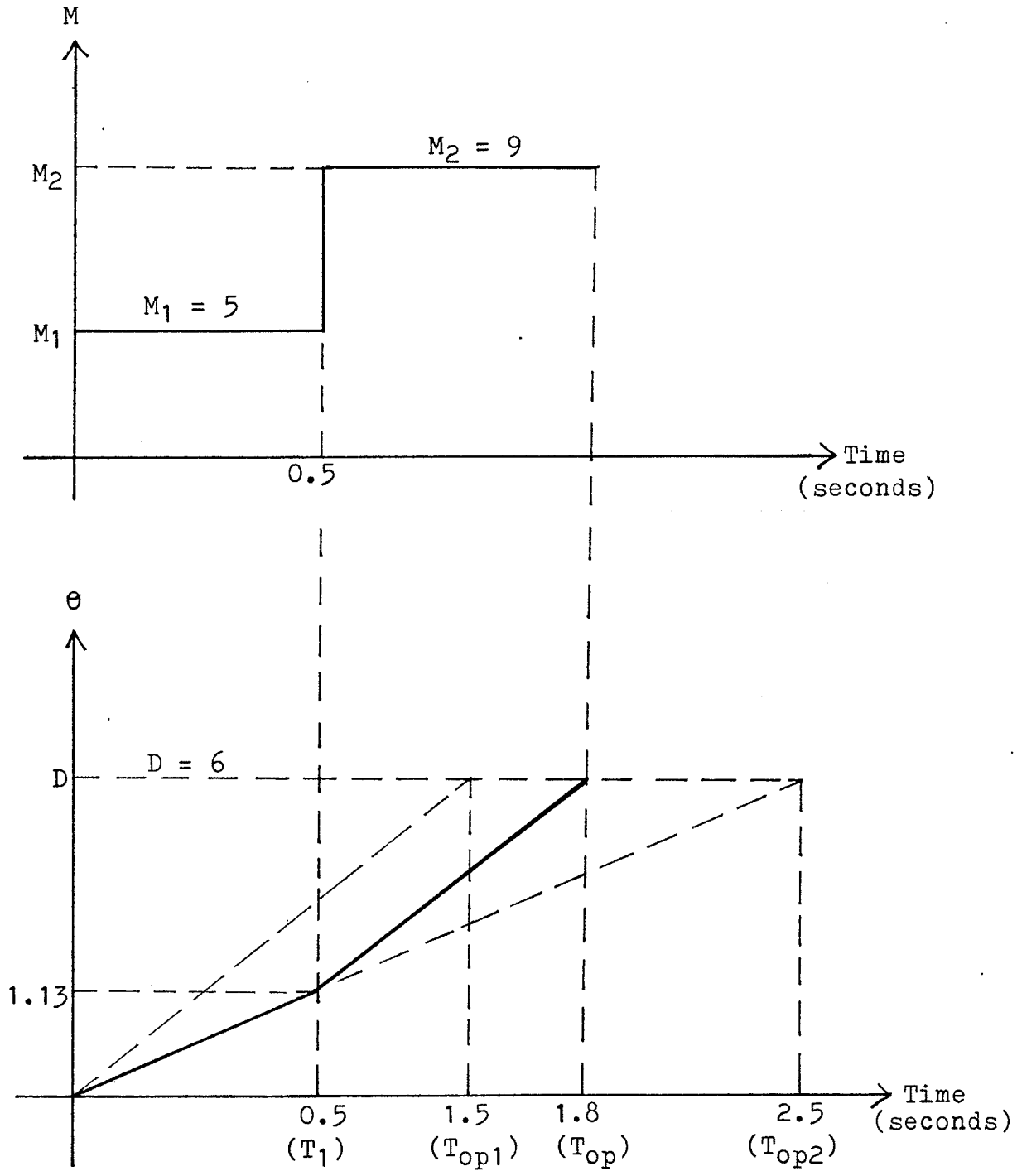


Figure 5.9: Trip simulation two : unstable fault current, breaker action not shown.

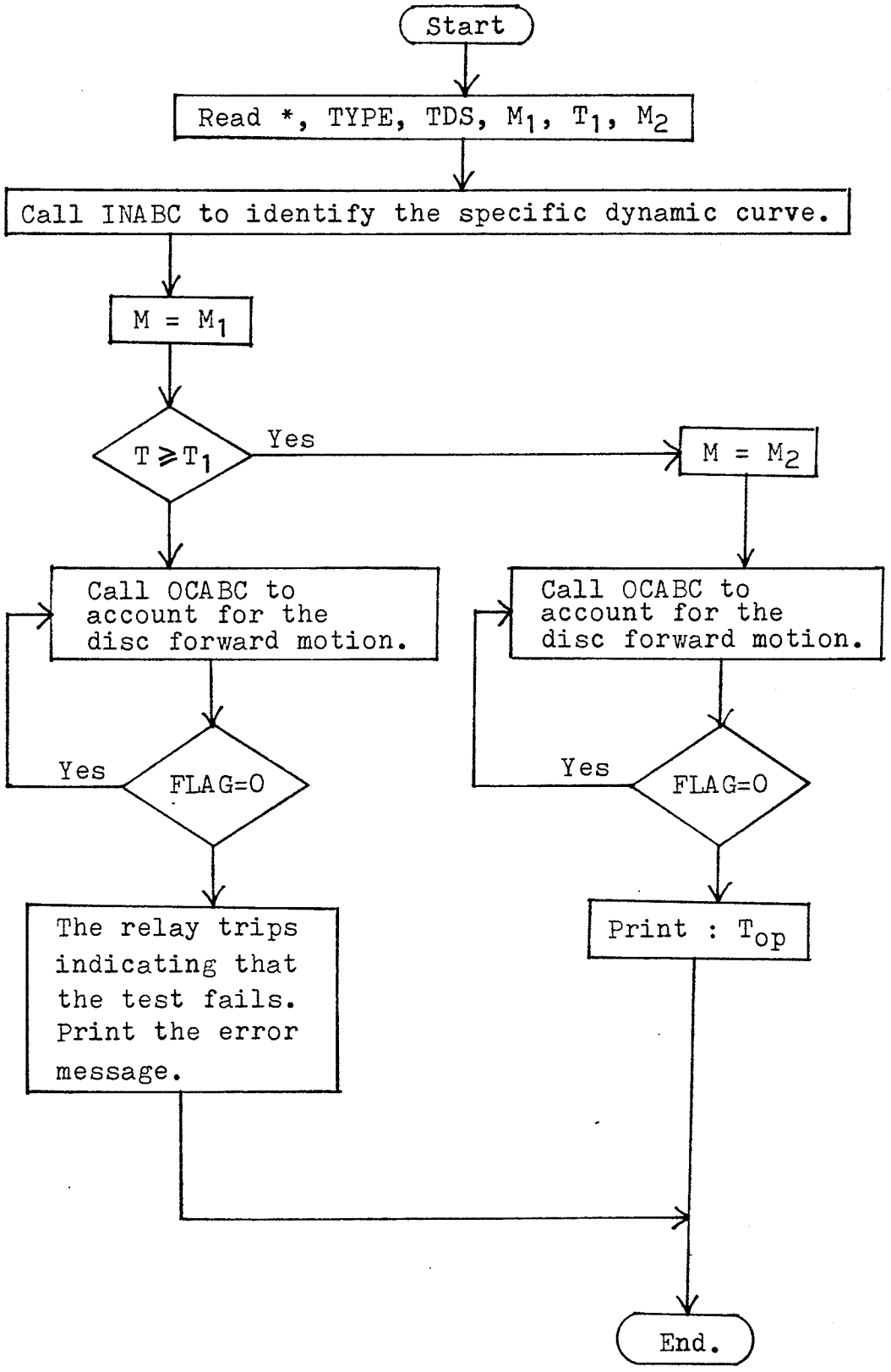


Figure 5.10: Program flow chart for trip simulation two.

5.4.3 Test case three

In the third test case, it is assumed that instantaneous tripping occurs initially at time T_I in figure 5.11, and the reclosing relay cuts out further instantaneous trip action and produces a breaker trip command. The objective here is to calculate the relay operating time following reclosure (T_o) by using the relay model shown in figure 5.6. The calculated T_o is then compared with a predicted value.

In figure 5.11, with D representing full travel to the contact-closed position, the duration of application of current in excess of pickup (T_I) establishes the degree of closure. Let this degree of closure be called θ_I. The complete expression for approximate operating time following a previous trip and reclosure (derived from figure 5.11) is :

$$T_o = (1 - T_I/T_{CO} + T_D/T_R) T_{CO} \qquad \text{(eq. 5.5)}$$

where T_o = Operate time of OC induction disc unit following reclosure.

T_I = Initial trip time, C.B. time negligible.

T_{CO} = Full operate time at current above pickup considered, with relay initially reset.

T_D = Dead time prior to reclosing.

T_R = Relay total reset time from fully closed position. (refer to figure 5.4)

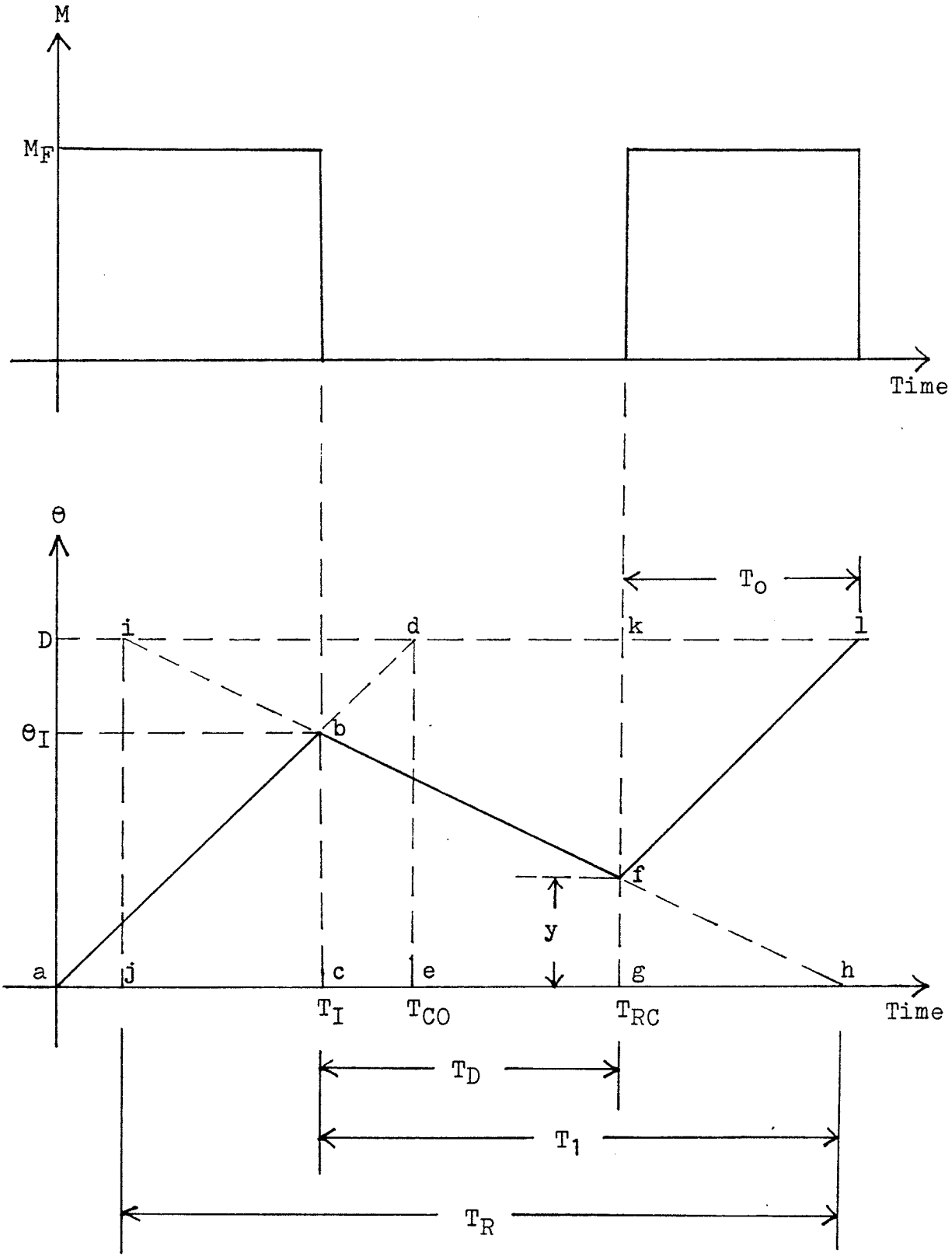


Figure 5.11: Trip simulation three : Initial tripping and relay reclosure are incorporated.

Referring to figure 5.11 and assuming operating and reset characteristic is linear, equation 5.5 can be derived in the following way :

Comparing △abc and △ade :

$$\frac{\theta_I}{D} = \frac{T_I}{T_{CO}} \quad , \quad \text{hence} \quad \theta_I = \left(\frac{T_I}{T_{CO}}\right)D$$

Comparing △bch and △ijh :

$$\frac{T_1}{T_R} = \frac{\theta_I}{D}$$

$$= \frac{T_I}{T_{CO}} \quad , \quad \text{hence} \quad T_1 = \left(\frac{T_I}{T_{CO}}\right)T_R$$

Comparing △fgh and △bch :

$$\frac{y}{\theta_I} = \frac{T_1 - T_D}{T_1} \quad , \quad \text{hence} \quad y = \left(\frac{T_1 - T_D}{T_1}\right)\theta_I$$

$$y = \frac{(T_I/T_{CO})T_R - T_D}{(T_I/T_{CO})T_R}\left(\frac{T_I}{T_{CO}}\right)D$$

$$= \frac{D(T_I/T_{CO})T_R - DT_D}{T_R}$$

Comparing △kfl and △eda :

$$\frac{T_O}{(D - y)} = \frac{T_{CO}}{D} \quad , \quad \text{hence} \quad T_o = \frac{(D - y)}{D}T_{CO}$$

$$T_o = \frac{1}{D}\left[D - \frac{D(T_I/T_{CO})T_R + DT_D}{T_R}\right]T_{CO}$$

$$= (1 - T_I/T_{CO} + T_D/T_R)T_{CO}$$

Let us use the CO-8 as an example again. Setting D=6, the following data are available as initial information :

let T_I = 3.0 seconds

let T_{RC} = 18.0 seconds

then $T_D = T_{RC} - T_I$ = 15.0 seconds

also T_{CO} = 5.3 seconds, at M_F = 3
(refer to figure 1.2)

and T_R = 42.0 seconds (refer to figure 5.4)

Substituting the above data into equation 5.5, T_O can be predicted as :

$$T_O = (1 - 3/5.3 + 15/42) \, 5.3$$
$$= 4.19 \text{ seconds}$$

Now, using the relay simulation model shown in figure 5.6, a program was written to calculate T_O. The program receives values of T_I and T_{RC} . It then computes the degree of disc travel at the relay's initial tripping as well as at relay reclosure. The operate time between reclosure and final tripping (T_O) will also be produced. The program output is given in figure 5.12. Figure 5.13 shows the program flow chart. The program listing can be found in appendix M.

The operate time of the relay following reclosure (T_o) generated by the test program is equal to 4.16 seconds which comes to a very good agreement with the predicted value : 4.19 seconds.

INPUT INFORMATION :

RELAY TYPE : CO-8 ; TDS = 6.0

A , B , C = 18.4663086 1.3976603 7.5571299

A1 , A2 , A3 , A4 = -0.1530450 1.4485979 -0.1658980 0.0150190

FAULT CURRENT, MF = 3.0000000

INITIAL TRIP TIME BY INSTANTANEOUS RELAY, TI = 3.000 SECONDS.

RECLOSURE TIME, TRC = 18.000 SECONDS.

OUTPUT INFORMATIUN :

DEGREE OF DISC TRAVEL, PHATA = 3.2517490 AT INITIAL TRIP.

DEGREE OF DISC TRAVEL, PHATA = 1.1068821 AT RECLOSURE.

TIME BETWEEN RECLOSURE AND FINAL TRIP, TO = 4.1616974 SECONDS.

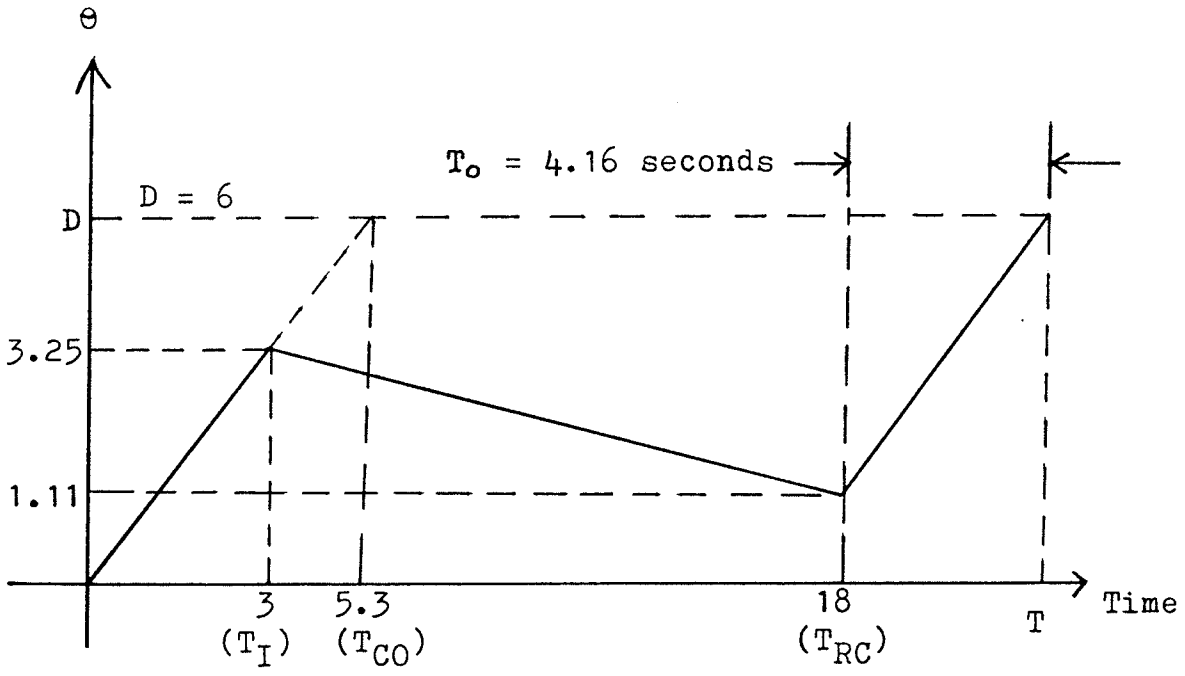


Figure 5.12: Results of trip simulation test case three.

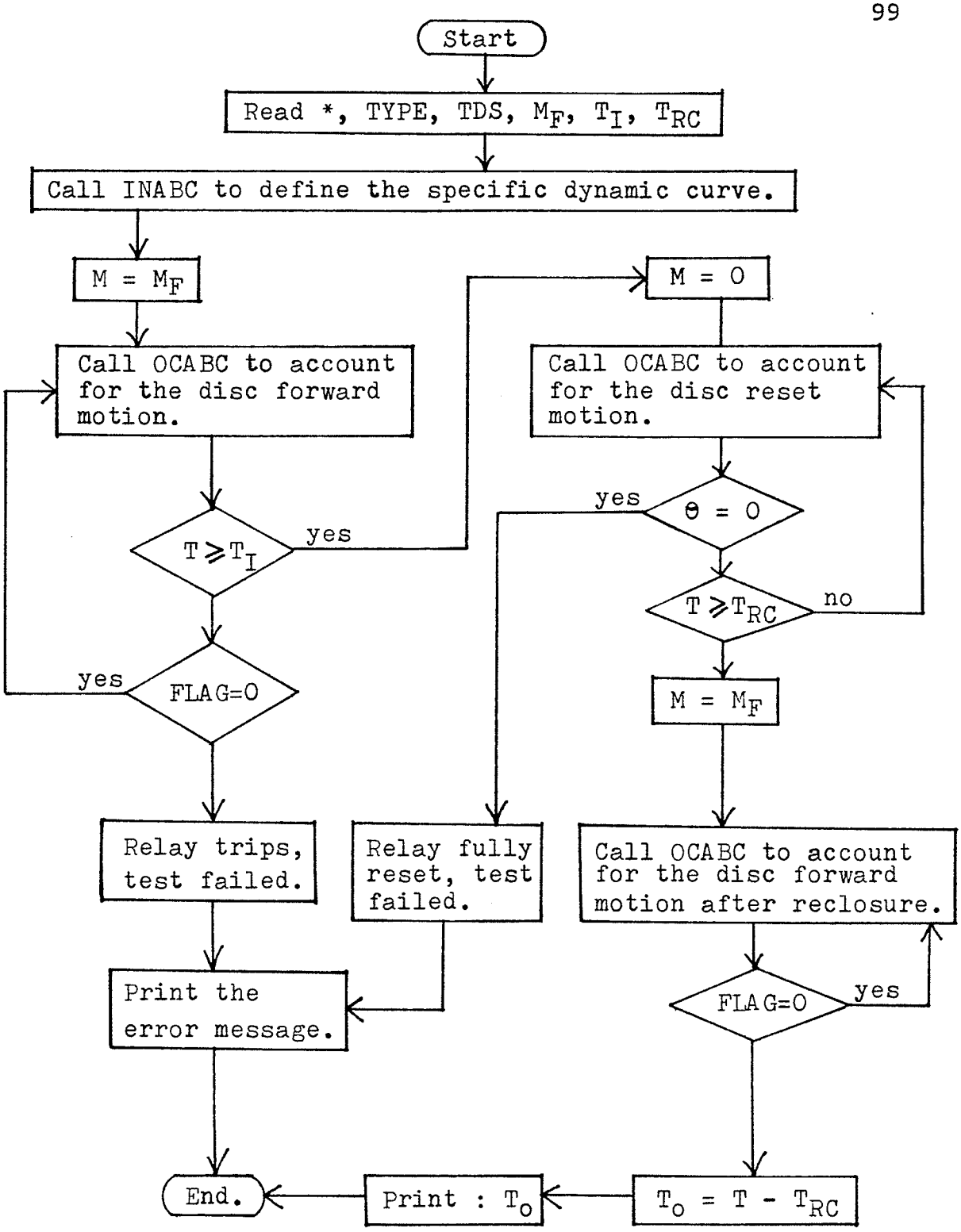


Figure 5.13: Test case three program flow chart.

Chapter VI

CONCLUSIONS

The investigation results presented in this thesis indicate that dynamic curve implementation is a practical approach to represent the operation of induction disc overcurrent relays. The optimization methods employed are very successful in finding a close approximation to represent the disc rotation angular velocity with acceptable deviations.

The non-linear continuous model, due to its complex mathematical expression, can provide a better approximation to the induction disc dynamic behavior for fault currents greater than two times the minimum pickup current (M>2). The region where M is less than two is an uncertain region and the deviations found in this region are non-material.

The non-linear continuous model requires seven coefficients to express the dynamic curves of each relay time dial setting. For a particalar type of relay, say the CO-8, the time-current curves published by manufacturers contain curves for eleven time dial settings. If the non-linear model is used for each time dial settings, it implies that 77 coefficients are needed for this relay. If the dynamic characteristics of several types of relay are to be stored in computer memory, it would be nice if a simpler mathematical

model could be developed so that data storage space could be saved. Another possible alternative is not to store all the curves of every time dial setting, but instead only several critical ones. Curves other than these critical ones could be calculated by using interpolation methods.

The piece-wise linear model also observes the error criteria faithfully, although the model must use more straight lines to represent the dynamic characteristics for some OC relays such as the CO-11 and IAC77. The search revealed that the CO-11 and IAC77 need four straight lines instead of three.

In actual application of this piece-wise linear model to a micro-processor based relay, it has been found that micro-processors are not very efficient with non-integers. Therefore, it is suggested that some kind of adjustment to the model might be necessary so that, if possible, the straight line equations might involve only integer numbers.

BIBLIOGRAPHY

1. Warrington, A.R.Van C. Protective Relays Their Theory and Practice, Vol.II. 3rd edition. London, England : Chapman and Hall Ltd., 1977.

2. Radke, G.E. A Method for Calculating Time-Over-Current Relay Setting by Digital Computer. IEEE Transaction on Power Apparatus and Systems, Vol.82 1963 Special Supplement, P.189-205; Discussion, Ibid, Vol.85, NO.3, 1966 P.303-307.

3. Sachdev, M.S.; Singh, J. and Fleming, R.J. Mathematical Models Representing Time-Current Characteristics of Overcurrent Relays for Computer Applications. presented to The IEEE Winter Meeting, New York, N.Y., January 29 - February 3, 1978. No. A 78 131-5.

4. Walsh, G.R. Methods of Optimization. New York : John Wiley & Sons, 1975.

5. Bandler, John W. and Charalamous, C. Practical Least pth Optimization of Networks. IEEE Transaction on Microwave Theory and Techniques, vol. MTT-20, No.12, P.834-840, December 1972.

6. Adby, P.R. and Dempster, M.A.H. Introduction to Optimization Methods. London, England : Chapman and Hall Ltd., 1974.

7. Elmore, W.A. and Hoinowski, E.R. Consideration in The Application of Modern Overcurrent Relays. Presented to The 3rd Annual Western Protective Relay Conference, Spokane, Washington, October 19-21, 1976. U.S.A : Westinghouse Electric Corp. Relay-Instrument Division.

8. Fox, Richard L. Optimization Methods for Engineering Design. Massachusettes : Addison-Wesley Publishing Company, Inc., 1971.

9. Swift, Glenn W. and Small, Robert C. Inverse-Time Overcurrent Relay. IEEE 1982 IECON proceeding, P.183.

10. Wilde, Douglass J. Optimum Seeking Methods. Englewood Cliffs, N.J. : Prentice-Hall, Inc., 1964.

Appendix A

THE CUBIC SPLINE CURVE FITTING METHOD - ALGORITHM DESCRIPTION

IMSL ROUTINE NAME - ICSCCU

PURPOSE - CUBIC SPLINE INTERPOLATION
(EASY-TO-USE VERSION)

USAGE - CALL ICSCCU (X,Y,NX,C,IC,IER)

ARGUMENTS

X - VECTOR OF LENGTH NX CONTAINING THE ABSCISSAE OF THE NX DATA POINTS (X(I),Y(I)) I=1,..., NX. (INPUT) X MUST BE ORDERED SO THAT X(I) .LT. X(I+1).

Y - VECTOR OF LENGTH NX CONTAINING THE ORDINATES (OR FUNCTION VALUES) OF THE NX DATA POINTS. (INPUT)

NX - NUMBER OF ELEMENTS IN X AND Y. (INPUT) NX MUST BE .GE. 2.

C - SPLINE COEFFICIENTS. (OUTPUT) C IS AN NX-1 BY 3 MATRIX. THE VALUE OF THE SPLINE APPROXIMATION AT T IS
S(T) = ((C(I,3)*D+C(I,2))*D+C(I,1))*D+Y(I)
WHERE X(I) .LE. T .LT. X(I+1) AND
D = T-X(I).

IC - ROW DIMENSION OF MATRIX C EXACTLY AS SPECIFIED IN THE DIMENSION STATEMENT IN THE CALLING PROGRAM. (INPUT)

IER - ERROR PARAMETER. (OUTPUT)
TERMINAL ERROR
IER = 129, IC IS LESS THAN NX-1.
IER = 130, NX IS LESS THAN 2.
IER = 131, INPUT ABSCISSA ARE NOT ORDERED SO THAT X(1) .LT. X(2) ... .LT. X(NX).

PRECISION/HARDWARE - SINGLE AND DOUBLE/H32
- SINGLE/H36,H48,H60

REQD. IMSL ROUTINES - UERTST,UGETIO

NOTATION - INFORMATION ON SPECIAL NOTATION AND CONVENTIONS IS AVAILABLE IN THE MANUAL INTRODUCTION OR THROUGH IMSL ROUTINE UHELP

Algorithm

ICSCCU computes an interpolatory approximation to a set of points by a cubic spline, with end point conditions determined automatically by the program. The endpoint conditions used correspond to the "not-a-knot" condition (see Reference) which requires that the third derivative of the spline be continuous at the second and penultimate knots.

See reference:

de Boor, Carl, "A Practical Guide to Splines," Springer-Verlag, N.Y, 1978.

The routine requires the X(I) is less than X(I+1) for I=1,2,...,NX-1. If X is not in ascending order, the user can reorder the vector by using IMSL sorting routines. For additional information, see the Chapter I introduction.

Example

The following example illustrates the usage of ICSCCU to interpolate the spline coefficients, ICSEVU to evaluate the spline at selected points, and USPLT to plot the results.

Input:

```
      INTEGER   NX,IC,IER,M,I,NXM1,NIN,NOUT
      REAL      X(9),Y(9),C(8,3),RI,RM,XPLOT(100),YPLOT(100),RANGE(4)
      DATA      X/0.0,.1,.23,.34,.47,.59,.73,.92,1.0/,
     1          IC/8/,NX/9/,M/100/,RM/100./,
     2          RANGE/0.0,1.0,0.0,0.0/
C                                F(X(I))=X(I)*EXP(-4.0*X(I))
      DO 5 I=1,NX
         Y(I) = X(I)*EXP(-4.0*X(I))
    5 CONTINUE
C                                CALL ICSCCU TO INTERPOLATE
      CALL ICSCCU (X,Y,NX,C,IC,IER)
C                                CALL UGETIO TO OBTAIN INPUT/OUTPUT UNITS
      CALL UGETIO (1,NIN,NOUT)
      WRITE (NOUT,20) IER
      IF (IER.NE.0) STOP
      WRITE(NOUT,25)
      NXM1 = NX-1
C                                OUTPUT ORDINATES AND
C                                SPLINE COEFFICIENTS
      DO 10 I=1,NXM1
          WRITE(NOUT,30)I,Y(I),C(I,1),C(I,2),C(I,3)
   10 CONTINUE
      WRITE(NOUT,30)NX,Y(NX)
C                                EVALUATE THE CALCULATED SPLINE
C                                  AT M POINTS AND GRAPH THE RESULTS
C                                CALCULATE THE ABSCISSAE OF THE
C                                  POINTS TO BE EVALUATED
      DO 15 I=1,M
         RI=I
         XPLOT(I)=RI/RM
   15 CONTINUE
C                                CALL ICSEVU TO EVALUATE
      CALL ICSEVU (X,Y,NX,C,IC,XPLOT,YPLOT,M,IER)
C                                CALL USPLO TO GRAPH
```

```
      CALL USPLO (XPLOT,YPLOT,M,M,1,1,19HICSCCU SPLINE GRAPH,19,
     1 6HX-AXIS,6,6HY-AXIS,6,RANGE,1H1,1,IER)
      STOP
C                                FORMAT STATEMENTS
   20 FORMAT(7H IER = ,I3)
   25 FORMAT(1X,1HI,4X,4HY(I),2X,6HC(I,1),2X,6HC(I,2),2X,6HC(I,3))
   30 FORMAT(IX,I1,4F8.4,/)
      END
```

Output:

IER = 0

I	Y(I)	C(I,1)	C(I,2)	C(I,3)
1	0.0	.9641	-3.3061	3.6827
2	.0670	.4134	-2.2013	3.6827
3	.0917	.0278	-.7650	1.3594
4	.0873	-.0912	-.3164	.7539
5	.0717	-.1352	-.0224	.3141
6	.0557	-.1271	.0907	-.1186
7	.0394	-.1086	.1566	-.1717
8	.0232	-.0677	.0852	-.0324
9	.0183			

The graphical output appears on the following page.

Appendix B

THE SEQUENTIAL UNCONSTRAINED MINIMIZATION TECHNIQUE (SUMT)

This method was proposed by Carroll in 1961 and was developed by Fiacco and McCormick. Carroll named it the 'Created Response Surface Technique'. The method replaces a constrained minimization problem by a sequence of unconstrained minimization problems. The basic idea is to attach different penalty functions to the given objective function in such a way that the optimal solutions of successive unconstrained problems approach the optimal solution of the given constrained problem.

Consider the constrained optimization problem :

$$\text{Minimize} \qquad z = f(\underset{\sim}{X})$$

$$\text{Subject to} \qquad g_i(\underset{\sim}{X}) \geqslant 0$$

This problem is replaced by the sequence of unconstrained problems :

$$\text{Minimize} \quad Z_K(\underset{\sim}{X}, r_K) = f(\underset{\sim}{X}) + r_K \sum_i w_i / g_i(\underset{\sim}{X}) \qquad \text{(eq. B.1)}$$

where $k = 1,2,3, \ldots\ldots\ldots$

$r_K \geqslant 0$ for all k

$w_i > 0$ for all i.

In equation B.1, the w_i are weighting factors that remain fixed throughout the calculation, while the r_K are parameters that decrease from one iteration to the next. From a feasible initial point, and using any unconstrained opimization technique, problem with k=1 is solved. The minimizing point for this problem becomes the initial point for the next, with k=2 and $r_2 > r$. This process is repeated until the desired accuracy is attained.

The exit criterion can be specified as :

when $f[\underset{\sim}{X}(r_j)] - f[\underset{\sim}{X}(r_{j-1})] < \epsilon$

If necessary, r_K is finally set equal to zero and a search is made for a minimum of the original objective function z, until a further decrease in z is prevented by a critical constraint or by limitations of accuracy.

It should be noted that the current point must remain feasible throughout the calculations. If a non-feasible point is reached at any time, then the calculations continue with a reduced step length from the last feasible point.

The simplest way to deal with the weighting factors w_i is to set them all equal to unity. However, it is preferable to choose them in such a way that the initial values of the functions $w_i / g_i(\underset{\sim}{X})$ are all of the same order of magnitude.

It has been suggested that the overall computational effort is closely associated with the choice of r_1 . Since the

computation of finding r_1 is complicated and time-consuming, in spite of all theory for predicting r_1 , equally good values of r_1 can be found by trial and error.

Appendix C

THE PIECE-WISE LINEAR MODEL - PROGRAM LISTING (THREE LINES)

Piece-Wise Linearization (Three Straight Lines)

```
 1. //TNG JOB '1222335,,T=40'
 2. //  EXEC FORTXCLG,OPT=2,USERLIB='MENZIES.MAIN.LOAD'
 3. //FORT.SYSIN DD *
 4. C
 5. C FILENAME : LNMS3
 6. C 3-STRAIGHT-LINE APPROXIMATION OF INVERSE TIME OVERCURRENT RELAY
 7. C DYNAMIC CURVES WITH 7 SELF-GENERATED STARTING POINTS.
 8. C RANGE OF M : 2<M<50
 9. C METHOD USED : LEASE PTH & SIMPLEX
10. C
11.         COMMON /IOUT/IN,IO,ID,P,MA,WA,MB,WB,MX,EMAX
12.         COMMON /ARAY/MD(15),WD(15),C(14,3)
13.         REAL V(11,10),VR(10),F(11),VE(10),VK(10),XO(10),VC(10)
14.         REAL Y(4),G(4),H(4),MD,WD,C,BPAR(4),MA,MB,MX,MXI,MDD(30),WDD(30)
15.         REAL MXM,MXX(15),TOAREA(15),BESTP(4),AREAM,ERR,ERROR
16.         INTEGER IC,NX,IER,J,IB,JB,KB,I,II
17.         INTEGER P
18. C
19.         IN=5
20.         IO=6
21.         NMX=10
22.         NMXP1=NMX+1
23. C
24.         P=10
25.         NX=15
26.         IC=NX-1
27. C
28.         READ *,ID,K,NFEMAX,A,ALPHA,BETA,GAMMA,EXIT
29. C
30.         DO 10 I=1,30
31.         READ *,MDD(I),WDD(I)
32.  10     CONTINUE
33. C
34.         DO 20 I=1,NX
35.            II=I*2
36.            MD(I)=MDD(II)
37.            WD(I)=WDD(II)
38.  20     CONTINUE
39. C
40.         DO 30 I=1,4
41.            BPAR(I)=0.0
42.  30     CONTINUE
43.         CALL ICSICU(MD,WD,NX,BPAR,C,IC,IER)
44. C
45.         MA=1.0
46.         WA=-0.2
47. C
48.         MB=MD(NX)
49.         WB=WD(NX)
50. C
51.         Y(1)=10.0
52.         Y(3)=15.0
53.         II=1
54.         DO 160 IB=1,2
55.         KB=5-IB
56.         DO 150 JB=1,KB
57.         Y(3)=Y(3)+5.0
58.         CALL EXAC(Y(1),FEX)
59.         Y(2)=FEX
60.         CALL EXAC(Y(3),FEX)
61.         Y(4)=FEX
62. C
63.         CALL ERRT(Y,ERR)
64.         ERRI=ERR
65. C
66.         XO(1)=ARSIN(SQRT((Y(1)-MA)/(MB-MA)))
67.         XO(2)=Y(2)
68.         XO(3)=ARSIN(SQRT((Y(3)-Y(1))/(MB-Y(1))))
69.         XO(4)=Y(4)
70. C
71.         CALL FUNCT(F,XO,NFEVAL)
72.         MXI=MX
```

```
C
      ID=0
      NFEMAX=400
      CALL SIMPLX(V,VC,VE,VK,VR,F,XO,A,ALPHA,BETA,GAMMA,EXIT,K,NFEMAX,
     &            NMX,NMXP1)
C
      IF (F(1).GT.F(2)) GO TO 40
      J=1
      SMAL=F(1)
      GO TO 50
 40   J=2
      SMAL=F(2)
 50   L=K+1
      DO 60 I=3,L
      IF (SMAL.LT.F(I)) GO TO 60
      J=I
      SMAL=F(I)
 60   CONTINUE
C
      G(1)=MA+(MB-MA)*(SIN(V(J,1)))**2
      G(2)=V(J,2)
      G(3)=G(1)+(MB-G(1))*(SIN(V(J,3)))**2
      G(4)=V(J,4)
C
      WRITE(IO,70) (Y(I),I=1,4)
 70   FORMAT(' ',/////,5X,'STARTING POINTS M1,W1,M2,W2 = ',
     &       4(2X,F10.7))
C
      WRITE(IO,80) ERRI
 80   FORMAT(' ',/,5X,'TOTAL AREA BETWEEN THE THREE STRAIGHT LINES AND
     &THE SPLINE CURVE, INITIAL VALUE = ',F10.5)
C
      WRITE(IO,90) MXI
 90   FORMAT(' ',/,5X,'INITIAL MAXIMUM ERROR FUNCTION VALUE, MX = '
     &       ,F10.5)
C
      WRITE(IO,100) (G(I),I=1,4)
 100  FORMAT(' ',/////,15X,'FINAL POINTS M1,W1,M2,W2 = ',4(2X,F10.7))
C
      CALL ERRT(G,ERR)
      WRITE(IO,110) ERR
 110  FORMAT(' ',/,15X,'TOTAL AREA BETWEEN THE THREE STRAIGHT LINES AND
     & THE SPLINE CURVE, FINAL VALUE = ',F10.6)
C
      H(1)=V(J,1)
      H(2)=V(J,2)
      H(3)=V(J,3)
      H(4)=V(J,4)
C
      CALL FUNCT(F,H,NFEVAL)
C
      CALL EXAC(EMAX,FEX)
      CALL FAPPX(G,EMAX,FAPX)
      ERRP=(FAPX-FEX)/FEX*100
C
      MXX(II)=MX
      TOAREA(II)=ERR
C
      IF (II.GT.1) GO TO 112
      AREAM=ERR
      MXM=MX
      DO 111 I=1,4
         BESTP(I)=G(I)
 111  CONTINUE
      GO TO 116
C
 112  IF (MXM.GT.0.AND.MX.GT.0) GO TO 113
      IF (MXM.LT.0.AND.MX.LT.0) GO TO 113
      IF (MXM.LT.0.AND.MX.GT.0) GO TO 116
      AREAM=ERR
      GO TO 114
C
 113  AREAM=AMIN1(AREAM,ERR)
      IF (AREAM.NE.ERR) GO TO 116
 114  MXM=MX
      DO 115 I=1,4
         BESTP(I)=G(I)
 115  CONTINUE
```

```
C
 116    II=II+1
        WRITE(IO,120) MX
 120    FORMAT(' ',/,15X,'FINAL MAXIMUM ERROR FUNCTION VALUE, MX = ',
     &         F10.6)
        WRITE(IO,130) EMAX
 130    FORMAT(/,15X,'MAXIMUM ERROR FUNCTION VALUE FOUND AT M= ',F10.7)
C
        WRITE(IO,140) ERRP
 140    FORMAT(' ',/,15X,'%ERROR AT THIS POINT = ',F10.6,' %')
C
 150    CONTINUE
        Y(1)=Y(1)+5.0
        Y(3)=Y(1)+5.0
 160    CONTINUE
C
        WRITE(IO,170)
 170    FORMAT('1',//////////)
        II=II-1
C
        DO 180 I=1,II
        WRITE(IO,175) MXX(I),TOAREA(I)
 175    FORMAT('0',10X,'MX = ',F10.6,5X,'FINAL AREA = ',F10.6)
 180    CONTINUE
C
        WRITE(IO,190) MA,WA
 190    FORMAT('1',/////,13X,'(MA,WA) = ','(',F10.6,',',F10.6,' )')
        WRITE(IO,191) BESTP(1),BESTP(2)
 191    FORMAT(/,13X,'(M1,W1) = ','(',F10.6,',',F10.6,' )')
        WRITE(IO,192) BESTP(3),BESTP(4)
 192    FORMAT(/,13X,'(M2,W2) = ','(',F10.6,',',F10.6,' )')
        WRITE(IO,193) MB,WB
 193    FORMAT(/,13X,'(MB,WB) = ','(',F10.6,',',F10.6,' )')
        WRITE(IO,194) MXM,AREAM
 194    FORMAT(///,13X,'MX = ',F10.6,5X,'FINAL AREA = ',F10.6)
C
        WRITE(6,210)
 210    FORMAT(/////,13X,'M',12X,'ORIG W',7X,'APPX W',7X,'ERROR',8X,
     &         '%ERROR',/)
C
        DO 230 I=1,30
        CALL FAPPX(BESTP,MDD(I),FAPX)
        ERROR=FAPX-WDD(I)
        PERC=ERROR/WDD(I)*100
        WRITE(IO,220) MDD(I),WDD(I),FAPX,ERROR,PERC
 220    FORMAT(10X,5(F10.5,3X))
 230    CONTINUE
C
        STOP
        END
C
C*********************************************************************
C
        SUBROUTINE FUNCT(F,X,NFEVAL)
        COMMON /IOUT/IN,IO,ID,P,MA,WA,MB,WB,MX,EMAX
        COMMON /ARAY/MD(15),WD(15),C(14,3)
        REAL X(1),EU(500),EL(500),Y(4)
        REAL M,MX,MA,MB,NEL
        INTEGER P,Q
C
        M=2.0
        EUT=0.0
        ELT=0.0
C
        Y(1)=MA+(MB-MA)*(SIN(X(1)))**2
        Y(2)=X(2)
        Y(3)=Y(1)+(MB-Y(1))*(SIN(X(3)))**2
        Y(4)=X(4)
C
        I=0
 5      IF (M.GT.50) GO TO 10
        CALL FAPPX(Y,M,FAPX)
        CALL ULSP(M,SU,SL)
        I=I+1
        EU(I)=FAPX-SU
        EL(I)=FAPX-SL
        NEL=EL(I)*(-1)
```

```
228.          IF (M.GT.2.0) GO TO 7
229.          MX=AMAX1(EU(I),NEL)
230.          XE=MX
231.          EMAX=M
232.   7      MX=AMAX1(MX,EU(I),NEL)
233.          IF (MX.NE.XE) EMAX=M
234.          XE=MX
235.          M=M+0.1
236.          GO TO 5
237.  C
238.   10     Q=P*MX/ABS(MX)
239.          IF (MX.GT.0) GO TO 20
240.          IF (MX.LE.0) GO TO 40
241.  C
242.   20     DO 30 J=1,I
243.          NEL=EL(J)*(-1)
244.          IF (EU(J).GE.0) EUT=EUT+(EU(J)/MX)**Q
245.          IF (NEL.GE.0) ELT=ELT+(NEL/MX)**Q
246.   30     CONTINUE
247.          GO TO 60
248.  C
249.   40     DO 50 J=1,I
250.          NEL=EL(J)*(-1)
251.          EUT=EUT+(EU(J)/MX)**Q
252.          ELT=ELT+(NEL/MX)**Q
253.   50     CONTINUE
254.  C
255.   60     F=MX*(EUT+ELT)**(1/Q)
256.          NFEVAL=NFEVAL+1
257.          RETURN
258.          END
259.  C
260.  C*************************************************************************
261.  C
262.          SUBROUTINE FAPPX(X,M,FAPX)
263.          COMMON /IOUT/IN,IO,ID,P,MA,WA,MB,WB,MX,EMAX
264.          COMMON /ARAY/MD(15),WD(15),C(14,3)
265.          REAL X(1),MD,WD,C
266.          REAL E,M,M1,M2,MA,MB
267.  C
268.          WAPX1(E)=(W1-WA)/(M1-MA)*(E-MA)+WA
269.          WAPX2(E)=(W2-W1)/(M2-M1)*(E-M1)+W1
270.          WAPX3(E)=(WB-W2)/(MB-M2)*(E-M2)+W2
271.  C
272.          M1=X(1)
273.          W1=X(2)
274.          M2=X(3)
275.          W2=X(4)
276.  C
277.          IF (M.LE.M1) FAPX=WAPX1(M)
278.          IF (M.GT.M1.AND.M.LE.M2) FAPX=WAPX2(M)
279.          IF (M.GT.M2) FAPX=WAPX3(M)
280.  C
281.          RETURN
282.          END
283.  C
284.  C*************************************************************************
285.  C
286.          SUBROUTINE ULSP(M,SU,SL)
287.          COMMON /IOUT/IN,IO,ID,P,MA,WA,MB,WB,MX,EMAX
288.          COMMON /ARAY/MD(15),WD(15),C(14,3)
289.          REAL MD,WD,C
290.          REAL M,MA,MB,UPB,LOB,SU,SL
291.  C
292.          IF (M.GE.2.AND.M.LT.5) GO TO 10
293.          IF (M.GE.5.AND.M.LT.10) GO TO 20
294.          IF (M.GE.10) GO TO 30
295.  C
296.   10     UPB=14.285710/100
297.          LOB=-11.11111/100
298.          GO TO 40
299.  C
300.   20     UPB=8.1081000/100
301.          LOB=-6.976740/100
302.          GO TO 40
303.  C
304.   30     UPB=5.2631500/100
305.          LOB=-4.761900/100
```

```
306. C
307.  40     CALL EXAC(M,FEX)
308.         SU=FEX*(1+UPB)
309.         SL=FEX*(1+LOB)
310. C
311.         RETURN
312.         END
313. C
314. C*******************************************************************************
315. C
316.         SUBROUTINE ERRT(Y,ERR)
317.         COMMON /IOUT/IN,IO,ID,P,MA,WA,MB,WB,MX,EMAX
318.         COMMON /ARAY/MD(15),WD(15),C(14,3)
319.         REAL Y(4),MD,WD,C,H,LAST
320.         REAL M,M1,M2,MA,MB
321. C
322.         N=100
323. C
324.         M1=Y(1)
325.         W1=Y(2)
326.         M2=Y(3)
327.         W2=Y(4)
328. C
329.         H=(M1-MA)/N
330.         SUM=0.0
331.         M=MA
332.         LAST=M1-H
333.  10     IF (M.GT.LAST) GO TO 20
334.         CALL EXAC(M,FEX)
335.         FM=ABS(FEX-(W1-WA)/(M1-MA)*(M-MA)-WA)
336.         SUM=SUM+FM
337.         M=M+H
338.         GO TO 10
339.  20     ERR1=H*SUM
340. C
341.         H=(M2-M1)/N
342.         SUM=0.0
343.         M=M1
344.         LAST=M2-H
345.  30     IF (M.GT.LAST) GO TO 40
346.         CALL EXAC(M,FEX)
347.         FM=ABS(FEX-(W2-W1)/(M2-M1)*(M-M1)-W1)
348.         SUM=SUM+FM
349.         M=M+H
350.         GO TO 30
351.  40     ERR2=H*SUM
352. C
353.         H=(MB-M2)/N
354.         SUM=0.0
355.         M=M2
356.         LAST=MB-H
357.  50     IF (M.GT.LAST) GO TO 60
358.         CALL EXAC(M,FEX)
359.         FM=ABS(FEX-(WB-W2)/(MB-M2)*(M-M2)-W2)
360.         SUM=SUM+FM
361.         M=M+H
362.         GO TO 50
363.  60     ERR3=H*SUM
364. C
365.         ERR=ERR1+ERR2+ERR3
366.         RETURN
367.         END
368. C
369. C*******************************************************************************
370. C
371.         SUBROUTINE EXAC(M,FEX)
372.         COMMON /IOUT/IN,IO,ID,P,MA,WA,MB,WB,MX,EMAX
373.         COMMON /ARAY/MD(15),WD(15),C(14,3)
374.         REAL MD,WD,C
375.         REAL M,A
376. C
377.         SP(A)=((C(I,3)*(A-MD(I))+C(I,2))*(A-MD(I))+C(I,1))*(A-MD(I))+
378.        &       WD(I)
379. C
380.         IF (M.GE.MD(1).AND.M.LE.MD(2)) I=1
381.         IF (M.GE.MD(2).AND.M.LE.MD(3)) I=2
382.         IF (M.GE.MD(3).AND.M.LE.MD(4)) I=3
383.         IF (M.GE.MD(4).AND.M.LE.MD(5)) I=4
384.         IF (M.GE.MD(5).AND.M.LE.MD(6)) I=5
```

```
      IF (M.GE.MD(6).AND.M.LE.MD(7)) I=6
      IF (M.GE.MD(7).AND.M.LE.MD(8)) I=7
      IF (M.GE.MD(8).AND.M.LE.MD(9)) I=8
      IF (M.GE.MD(9).AND.M.LE.MD(10)) I=9
      IF (M.GE.MD(10).AND.M.LE.MD(11)) I=10
      IF (M.GE.MD(11).AND.M.LE.MD(12)) I=11
      IF (M.GE.MD(12).AND.M.LE.MD(13)) I=12
      IF (M.GE.MD(13).AND.M.LE.MD(14)) I=13
      IF (M.GE.MD(14).AND.M.LE.MD(15)) I=14
C
      FEX=SP(M)
C
      RETURN
      END
//GO.SYSIN DD *
0   4   400   1.000   1.0   0.5   2.0   0.0001000
   1.3900900        0.1147262
   1.6059940        0.1987057
   2.1236590        0.4929591
   2.6680420        0.8188379
   3.1835620        1.1855410
   4.1730020        1.8882500
   5.2492010        2.5398610
   6.2697440        2.9465270
   7.2577410        3.3113320
   8.2277910        3.6432560
   9.2306030        3.9661930
  10.4658400        4.2278080
  11.8663700        4.5066810
  13.7389200        4.8553650
  15.5774600        5.1756330
  17.3026700        5.3446290
  18.6131400        5.5168060
  20.4521000        5.7587000
  22.9503100        5.9470190
  25.4837000        6.2063400
  28.2994600        6.3417330
  30.7738900        6.4794550
  33.1154700        6.6198530
  36.0110000        6.7636160
  38.7510600        6.9101700
  41.2664400        6.9854730
  43.9451200        7.0615970
  46.3093400        7.1382080
  49.3129800        7.2925280
  51.9684600        7.2942790
```

Piece-Wise Linearization (Four Straight Lines)

```
1. //TNG JOB '1222335,,T=30'
2. //  EXEC FORTXCLG,OPT=2,USERLIB='MENZIES.MAIN.LOAD'
3. //FORT.SYSIN DD *
4. C
5. C FILENAME : LNMS4
6. C 4-STRAIGHT-LINE APPROXIMATION OF INVERSE TIME OVERCURRENT RELAY
7. C DYNAMIC CURVES WITH MULTIPLE STARTING POINTS.
8. C RANGE OF M : 2<M<50
9. C METHOD USED : LEASE PTH AND SIMPLEX
10. C
11.       COMMON /IOUT/IN,IO,ID,P,MA,WA,MB,WB,MX,EMAX
12.       COMMON /ARAY/MD(15),WD(15),C(14,3)
13.       REAL V(11,10),VR(10),F(11),VE(10),VK(10),XO(10),VC(10)
14.       REAL Y(6),G(6),H(6),MD,WD,C,BPAR(4),MA,MB,MX,MXI,MDD(30),WDD(30)
15.       REAL MXM,MXX(15),TOAREA(15),BESTP(6),AREAM,ERR,ERROR,INPT(5,3)
16.       INTEGER IC,NX,IER,J,IB,I,II,NTRY
17.       INTEGER P
18. C
19.       IN=5
20.       IO=6
21.       NMX=10
22.       NMXP1=NMX+1
23. C
24.       P=10
25.       NX=15
26.       IC=NX-1
27. C
28.       READ *,ID,K,NFEMAX,A,ALPHA,BETA,GAMMA,EXIT
29.       READ *,NTRY
30. C
31.       DO 5 I=1,NTRY
32.       READ *,INPT(I,1),INPT(I,2),INPT(I,3)
33.  5    CONTINUE
34. C
35.       DO 10 I=1,30
36.       READ *,MDD(I),WDD(I)
37.  10   CONTINUE
38. C
39.       DO 20 I=1,NX
40.          II=I*2
41.          MD(I)=MDD(II)
42.          WD(I)=WDD(II)
43.  20   CONTINUE
44. C
45.       DO 30 I=1,4
46.          BPAR(I)=0.0
47.  30   CONTINUE
48.       CALL ICSICU(MD,WD,NX,BPAR,C,IC,IER)
49. C
50.       MA=2.0
51.       CALL EXAC(MA,FEX)
52. C     WA=FEX*1.1428550
53. C     WA=FEX
54.       WA=FEX*0.8888900
55. C
56.       MB=MD(NX)
57.       WB=WD(NX)
58. C
59.       II=1
60.       DO 160 IB=1,NTRY
61.       Y(1)=INPT(IB,1)
62.       Y(3)=INPT(IB,2)
63.       Y(5)=INPT(IB,3)
64.       CALL EXAC(Y(1),FEX)
65.       Y(2)=FEX
66.       CALL EXAC(Y(3),FEX)
67.       Y(4)=FEX
68.       CALL EXAC(Y(5),FEX)
69.       Y(6)=FEX
70. C
71.       CALL ERRT(Y,ERR)
72.       ERRI=ERR
```

```
C
      XO(1)=ARSIN(SQRT((Y(1)-MA)/(MB-MA)))
      XO(2)=Y(2)
      XO(3)=ARSIN(SQRT((Y(3)-Y(1))/(MB-Y(1))))
      XO(4)=Y(4)
      XO(5)=ARSIN(SQRT((Y(5)-Y(3))/(MB-Y(3))))
      XO(6)=Y(6)
C
      CALL FUNCT(F,XO,NFEVAL)
      MXI=MX
C
      ID=0
      NFEMAX=400
      CALL SIMPLX(V,VC,VE,VK,VR,F,XO,A,ALPHA,BETA,GAMMA,EXIT,K,NFEMAX,
     &            NMX,NMXP1)
C
      IF (F(1).GT.F(2)) GO TO 40
      J=1
      SMAL=F(1)
      GO TO 50
 40   J=2
      SMAL=F(2)
 50   L=K+1
      DO 60 I=3,L
      IF (SMAL.LT.F(I)) GO TO 60
      J=I
      SMAL=F(I)
 60   CONTINUE
C
      G(1)=MA+(MB-MA)*(SIN(V(J,1)))**2
      G(2)=V(J,2)
      G(3)=G(1)+(MB-G(1))*(SIN(V(J,3)))**2
      G(4)=V(J,4)
      G(5)=G(3)+(MB-G(3))*(SIN(V(J,5)))**2
      G(6)=V(J,6)
C
      WRITE(IO,70) (Y(I),I=1,6)
 70   FORMAT(' ',/////,5X,'STARTING POINTS M1,W1,M2,W2,M3,W3 = ',
     &       6(2X,F10.7))
C
      WRITE(IO,80) ERRI
 80   FORMAT(' ',/,5X,'TOTAL AREA BETWEEN THE FOUR STRAIGHT LINES AND
     &THE SPLINE CURVE, INITIAL VALUE = ',F10.5)
C
      WRITE(IO,90) MXI
 90   FORMAT(' ',/,5X,'INITIAL MAXIMUM ERROR FUNCTION VALUE, MX = '
     &       ,F10.5)
C
      WRITE(IO,100) (G(I),I=1,6)
 100  FORMAT(' ',/////,15X,'FINAL POINTS M1,W1,M2,W2,M3,W3 = ',
     &       6(2X,F10.7))
C
      CALL ERRT(G,ERR)
      WRITE(IO,110) ERR
 110  FORMAT(' ',/,15X,'TOTAL AREA BETWEEN THE FOUR STRAIGHT LINES AND
     & THE SPLINE CURVE, FINAL VALUE = ',F10.6)
C
      H(1)=V(J,1)
      H(2)=V(J,2)
      H(3)=V(J,3)
      H(4)=V(J,4)
      H(5)=V(J,5)
      H(6)=V(J,6)
C
      CALL FUNCT(F,H,NFEVAL)
C
      CALL EXAC(EMAX,FEX)
      CALL FAPPX(G,EMAX,FAPX)
      ERRP=(FAPX-FEX)/FEX*100
C
      MXX(II)=MX
      TOAREA(II)=ERR
C
      IF (II.GT.1) GO TO 112
      AREAM=ERR
      MXM=MX
      DO 111 I=1,6
         BESTP(I)=G(I)
 111  CONTINUE
      GO TO 116
```

```
153. C
154.  112   IF (MXM.GT.0.AND.MX.GT.0) GO TO 113
155.        IF (MXM.LT.0.AND.MX.LT.0) GO TO 113
156.        IF (MXM.LT.0.AND.MX.GT.0) GO TO 116
157.        AREAM=ERR
158.        GO TO 114
159. C
160.  113   AREAM=AMIN1(AREAM,ERR)
161.        IF (AREAM.NE.ERR) GO TO 116
162.  114   MXM=MX
163.        DO 115 I=1,6
164.           BESTP(I)=G(I)
165.  115   CONTINUE
166. C
167.  116   II=II+1
168.        WRITE(IO,120) MX
169.  120   FORMAT(' ',/,15X,'FINAL MAXIMUM ERROR FUNCTION VALUE, MX = ',
170.     &         F10.6)
171.        WRITE(IO,130) EMAX
172.  130   FORMAT(/,15X,'MAXIMUM ERROR FUNCTION VALUE FOUND AT M= ',F10.7)
173. C
174.        WRITE(IO,140) ERRP
175.  140   FORMAT(' ',/,15X,'%ERROR AT THIS POINT = ',F10.6,' %')
176. C
177.  160   CONTINUE
178. C
179.        WRITE(IO,170)
180.  170   FORMAT('1',//////////)
181.        II=II-1
182. C
183.        DO 180 I=1,II
184.        WRITE(IO,175) MXX(I),TOAREA(I)
185.  175   FORMAT('0',10X,'MX = ',F10.6,5X,'FINAL AREA = ',F10.6)
186.  180   CONTINUE
187. C
188.        WRITE(IO,190) MA,WA
189.  190   FORMAT('1',/////,13X,'(MA,WA) = ','(',F10.6,',',F10.6,' )')
190.        WRITE(IO,191) BESTP(1),BESTP(2)
191.  191   FORMAT(/,13X,'(M1,W1) = ','(',F10.6,',',F10.6,' )')
192.        WRITE(IO,192) BESTP(3),BESTP(4)
193.  192   FORMAT(/,13X,'(M2,W2) = ','(',F10.6,',',F10.6,' )')
194.        WRITE(IO,193) BESTP(5),BESTP(6)
195.  193   FORMAT(/,13X,'(M3,W3) = ','(',F10.6,',',F10.6,' )')
196.        WRITE(IO,194) MB,WB
197.  194   FORMAT(/,13X,'(MB,WB) = ','(',F10.6,',',F10.6,' )')
198.        WRITE(IO,195) MXM,AREAM
199.  195   FORMAT(///,13X,'MX = ',F10.6,5X,'FINAL AREA = ',F10.6)
200. C
201.        WRITE(6,210)
202.  210   FORMAT(/////,13X,'M',12X,'ORIG W',7X,'APPX W',7X,'ERROR',8X,
203.     &         '%ERROR',/)
204. C
205.        DO 230 I=1,30
206.        CALL FAPPX(BESTP,MDD(I),FAPX)
207.        ERROR=FAPX-WDD(I)
208.        PERC=ERROR/WDD(I)*100
209.        WRITE(IO,220) MDD(I),WDD(I),FAPX,ERROR,PERC
210.  220   FORMAT(10X,5(F10.5,3X))
211.  230   CONTINUE
212. C
213.        STOP
214.        END
215. C
216. C*****************************************************************
217. C
218.        SUBROUTINE FUNCT(F,X,NFEVAL)
219.        COMMON /IOUT/IN,IO,ID,P,MA,WA,MB,WB,MX,EMAX
220.        COMMON /ARAY/MD(15),WD(15),C(14,3)
221.        REAL X(1),EU(500),EL(500),Y(6)
222.        REAL M,MX,MA,MB,NEL
223.        INTEGER P,Q
224. C
225.        M=2.0
226.        EUT=0.0
227.        ELT=0.0
228. C
229.        Y(1)=MA+(MB-MA)*(SIN(X(1)))**2
230.        Y(2)=X(2)
231.        Y(3)=Y(1)+(MB-Y(1))*(SIN(X(3)))**2
232.        Y(4)=X(4)
233.        Y(5)=Y(3)+(MB-Y(3))*(SIN(X(5)))**2
234.        Y(6)=X(6)
```

```
C
      I=0
 5    IF (M.GT.50) GO TO 10
      CALL FAPPX(Y,M,FAPX)
      CALL ULSP(M,SU,SL)
      I=I+1
      EU(I)=FAPX-SU
      EL(I)=FAPX-SL
      NEL=EL(I)*(-1)
      IF (M.GT.2.0) GO TO 7
      MX=AMAX1(EU(I),NEL)
      XE=MX
      EMAX=M
 7    MX=AMAX1(MX,EU(I),NEL)
      IF (MX.NE.XE) EMAX=M
      XE=MX
      M=M+0.1
      GO TO 5
C
 10   Q=P*MX/ABS(MX)
      IF (MX.GT.0) GO TO 20
      IF (MX.LE.0) GO TO 40
C
 20   DO 30 J=1,I
      NEL=EL(J)*(-1)
      IF (EU(J).GE.0) EUT=EUT+(EU(J)/MX)**Q
      IF (NEL.GE.0) ELT=ELT+(NEL/MX)**Q
 30   CONTINUE
      GO TO 60
C
 40   DO 50 J=1,I
      NEL=EL(J)*(-1)
      EUT=EUT+(EU(J)/MX)**Q
      ELT=ELT+(NEL/MX)**Q
 50   CONTINUE
C
 60   F=MX*(EUT+ELT)**(1/Q)
      NFEVAL=NFEVAL+1
      RETURN
      END
C
C*********************************************************************
C
      SUBROUTINE FAPPX(X,M,FAPX)
      COMMON /IOUT/IN,IO,ID,P,MA,WA,MB,WB,MX,EMAX
      COMMON /ARAY/MD(15),WD(15),C(14,3)
      REAL X(1),MD,WD,C
      REAL E,M,M1,M2,M3,MA,MB
C
      WAPX1(E)=(W1-WA)/(M1-MA)*(E-MA)+WA
      WAPX2(E)=(W2-W1)/(M2-M1)*(E-M1)+W1
      WAPX3(E)=(W3-W2)/(M3-M2)*(E-M2)+W2
      WAPX4(E)=(WB-W3)/(MB-M3)*(E-M3)+W3
C
      M1=X(1)
      W1=X(2)
      M2=X(3)
      W2=X(4)
      M3=X(5)
      W3=X(6)
C
      IF (M.LE.M1) FAPX=WAPX1(M)
      IF (M.GT.M1.AND.M.LE.M2) FAPX=WAPX2(M)
      IF (M.GT.M2.AND.M.LE.M3) FAPX=WAPX3(M)
      IF (M.GT.M3) FAPX=WAPX4(M)
C
      RETURN
      END
C
C*********************************************************************
C
      SUBROUTINE ULSP(M,SU,SL)
      COMMON /IOUT/IN,IO,ID,P,MA,WA,MB,WB,MX,EMAX
      COMMON /ARAY/MD(15),WD(15),C(14,3)
      REAL MD,WD,C
      REAL M,MA,MB,UPB,LOB,SU,SL
C
      IF (M.GE.2.AND.M.LT.5) GO TO 10
      IF (M.GE.5.AND.M.LT.10) GO TO 20
      IF (M.GE.10) GO TO 30
```

```
315. C
316.   10    UPB=14.285710/100
317.         LOB=-11.11111/100
318.         GO TO 40
319. C
320.   20    UPB=8.1081000/100
321.         LOB=-6.976740/100
322.         GO TO 40
323. C
324.   30    UPB=5.2631500/100
325.         LOB=-4.761900/100
326. C
327.   40    CALL EXAC(M,FEX)
328.         SU=FEX*(1+UPB)
329.         SL=FEX*(1+LOB)
330. C
331.         RETURN
332.         END
333. C
334. C*****************************************************************************
335. C
336.         SUBROUTINE ERRT(Y,ERR)
337.         COMMON /IOUT/IN,IO,ID,P,MA,WA,MB,WB,MX,EMAX
338.         COMMON /ARAY/MD(15),WD(15),C(14,3)
339.         REAL Y(6),MD,WD,C,H,LAST
340.         REAL M,M1,M2,M3,MA,MB
341. C
342.         N=100
343. C
344.         M1=Y(1)
345.         W1=Y(2)
346.         M2=Y(3)
347.         W2=Y(4)
348.         M3=Y(5)
349.         W3=Y(6)
350. C
351.         H=(M1-MA)/N
352.         SUM=0.0
353.         M=MA
354.         LAST=M1-H
355.   10    IF (M.GT.LAST) GO TO 20
356.         CALL EXAC(M,FEX)
357.         FM=ABS(FEX-(W1-WA)/(M1-MA)*(M-MA)-WA)
358.         SUM=SUM+FM
359.         M=M+H
360.         GO TO 10
361.   20    ERR1=H*SUM
362. C
363.         H=(M2-M1)/N
364.         SUM=0.0
365.         M=M1
366.         LAST=M2-H
367.   30    IF (M.GT.LAST) GO TO 40
368.         CALL EXAC(M,FEX)
369.         FM=ABS(FEX-(W2-W1)/(M2-M1)*(M-M1)-W1)
370.         SUM=SUM+FM
371.         M=M+H
372.         GO TO 30
373.   40    ERR2=H*SUM
374. C
375.         H=(M3-M2)/N
376.         SUM=0.0
377.         M=M2
378.         LAST=M3-H
379.   50    IF (M.GT.LAST) GO TO 60
380.         CALL EXAC(M,FEX)
381.         FM=ABS(FEX-(W3-W2)/(M3-M2)*(M-M2)-W2)
382.         SUM=SUM+FM
383.         M=M+H
384.         GO TO 50
385.   60    ERR3=H*SUM
386. C
387.         H=(MB-M3)/N
388.         SUM=0.0
389.         M=M3
390.         LAST=MB-H
391.   70    IF (M.GT.LAST) GO TO 80
392.         CALL EXAC(M,FEX)
393.         FM=ABS(FEX-(WB-W3)/(MB-M3)*(M-M3)-W3)
394.         SUM=SUM+FM
```

```
395.          M=M+H
396.          GO TO 70
397.   80     ERR4=H*SUM
398.  C
399.          ERR=ERR1+ERR2+ERR3+ERR4
400.          RETURN
401.          END
402.  C
403.  C*************************************************************************
404.  C
405.          SUBROUTINE EXAC(M,FEX)
406.          COMMON /IOUT/IN,IO,ID,P,MA,WA,MB,WB,MX,EMAX
407.          COMMON /ARAY/MD(15),WD(15),C(14,3)
408.          REAL MD,WD,C
409.          REAL M,A
410.  C
411.          SP(A)=((C(I,3)*(A-MD(I))+C(I,2))*(A-MD(I))+C(I,1))*(A-MD(I))+
412.         &        WD(I)
413.  C
414.          IF (M.GE.MD(1).AND.M.LE.MD(2)) I=1
415.          IF (M.GE.MD(2).AND.M.LE.MD(3)) I=2
416.          IF (M.GE.MD(3).AND.M.LE.MD(4)) I=3
417.          IF (M.GE.MD(4).AND.M.LE.MD(5)) I=4
418.          IF (M.GE.MD(5).AND.M.LE.MD(6)) I=5
419.          IF (M.GE.MD(6).AND.M.LE.MD(7)) I=6
420.          IF (M.GE.MD(7).AND.M.LE.MD(8)) I=7
421.          IF (M.GE.MD(8).AND.M.LE.MD(9)) I=8
422.          IF (M.GE.MD(9).AND.M.LE.MD(10)) I=9
423.          IF (M.GE.MD(10).AND.M.LE.MD(11)) I=10
424.          IF (M.GE.MD(11).AND.M.LE.MD(12)) I=11
425.          IF (M.GE.MD(12).AND.M.LE.MD(13)) I=12
426.          IF (M.GE.MD(13).AND.M.LE.MD(14)) I=13
427.          IF (M.GE.MD(14).AND.M.LE.MD(15)) I=14
428.  C
429.          FEX=SP(M)
430.  C
431.          RETURN
432.          END
433.  //GO.SYSIN DD *
434.  0  6  400  1.000  1.0  0.5  2.0  0.0001000
435.  3
436.  3.0   6.00   15.0
437.  4.0   10.0   20.0
438.  4.0   15.0   30.0
439.    1.2870800       0.1435514
440.    1.6103750       0.3120120
441.    1.8498220       0.4971154
442.    2.1025450       0.7443022
443.    2.5984500       1.2745290
444.    3.0784890       1.9276300
445.    3.5707380       2.7684610
446.    4.0964520       3.6597480
447.    4.5000000       4.5454530
448.    5.0076740       5.5916160
449.    6.1163940       7.5414220
450.    7.0134300       9.2738210
451.    8.0408900      11.1669500
452.    9.0268510      13.0364500
453.   10.0251300      14.4508900
454.   11.3722600      16.8689700
455.   13.0336700      19.2861000
456.   14.9367500      21.8220600
457.   16.4143600      23.6948200
458.   18.2257600      25.4610400
459.   19.8183400      27.0804700
460.   21.3252400      28.5077200
461.   22.9450600      29.6922600
462.   24.6879500      30.9433400
463.   27.1225100      32.2335600
464.   29.4904100      33.9300200
465.   32.0627800      35.3472700
466.   34.5030000      36.0703200
467.   36.7261600      37.1945100
468.   38.6954900      37.9606900
```

Appendix D

THE SIMPLEX METHOD OF NELDER AND MEAD

A method based on the geometrical design known as a regular simplex was devised in 1962 by Spendley, Hext and Himsworth. A simplex in a n-dimemsional hyperspace E^n consists of (n+1) points $\underset{\sim}{X}_t$ (t=1,2,..,n+1) which do not lie on a hyperplane, together with every convex combination of these points. The simplex is regular if its vertices are equally spaced. Examples of regular simplices are an equilateral triangle in E^2 and a regular tetrahedron in E^3 . The basic ideas of Spenley, Hext and Himsworth's method will be explained for the two-dimensional case; the extension to n dimensions is straightforward.

Two-dimensional case

Consider the problem of minimizing $f(\underset{\sim}{X})$. Let $\underset{\sim}{X}_1$ be an initial estimate of $\underset{\sim}{X}^*$ and let $\underset{\sim}{X}_1$, $\underset{\sim}{X}_2$, $\underset{\sim}{X}_3$ be the vertices of a regular simplex in E^2. The reflection (Figure D.1) of the vertex $\underset{\sim}{X}_1$ in the line segment joining $\underset{\sim}{X}_2$ and $\underset{\sim}{X}_3$ is defined in a natural way as

$$\underset{\sim}{X}_4 = \underset{\sim}{X}_2 + \underset{\sim}{X}_3 - \underset{\sim}{X}_1 \qquad \text{(eq. D.1)}$$

it is obvious that the points $\underset{\sim}{X}_2$, $\underset{\sim}{X}_3$, $\underset{\sim}{X}_4$ are also the vertices of a regular simplex. Similarly, if $\underset{\sim}{X}_2$ or $\underset{\sim}{X}_3$ is reflected, then further regular simplices are formed.

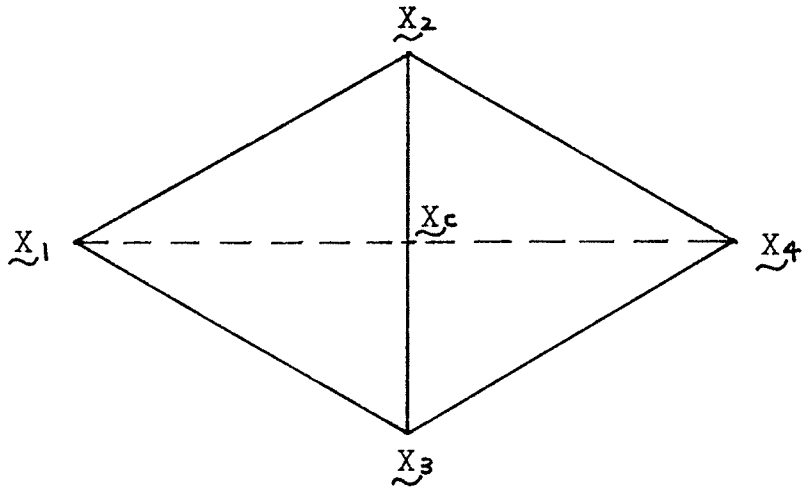


Figure D.1: Reflection of vertex $\underset{\sim}{X}_1$ in regular simplex

If $f(\underset{\sim}{X})$ is a linear function of $\underset{\sim}{X}$ in the neighbourhood of the initial simplex, then

$$f(\underset{\sim}{X}_4) = f(\underset{\sim}{X}_2+\underset{\sim}{X}_3-\underset{\sim}{X}_1) = f(\underset{\sim}{X}_2)+f(\underset{\sim}{X}_3)-f(\underset{\sim}{X}_1) \qquad \text{(eq. D.2)}$$

suppose that

$$f(\underset{\sim}{X}_1) > \max[f(\underset{\sim}{X}_2), f(\underset{\sim}{X}_3)]$$

then equation D.2 show that

$$f(\underset{\sim}{X}_4) < f(\underset{\sim}{X}_2) \quad \text{and} \quad f(\underset{\sim}{X}_4) < f(\underset{\sim}{X}_3)$$

since $f(\underset{\sim}{X})$ is to be minimized, it is advantageous to replace $\underset{\sim}{X}_1$ by $\underset{\sim}{X}_4$. However, when $f(\underset{\sim}{X})$ is not linear in the neighbourhood of the current simplex, there is no guarantee that $f(\underset{\sim}{X}_4)<f(\underset{\sim}{X}_1)$. Nevertheless, the method of Spendley, Hext and Himsworth is based entirely on the above results for locally linear functions.

The recommended procedure for the minimization of $f(\underset{\sim}{X})$ when $\underset{\sim}{X}$ is two-dimensional is therefore as follows :

1. Estimate the coordinates of the optimal point $\underset{\sim}{X}^*$ and let this estimated point be one of the vertices of the initial simplex. Scale the variables if necessary and choose the remaining vertices of the initial simplex.
2. Evaluate $f(\underset{\sim}{X})$ at the vertices of the initial simplex.
3. Replace the vertex with the highest function value by its reflection and evaluate $f(\underset{\sim}{X})$ at the new vertex.
4. Repeat step 3 for each new simplex.
5. If the highest function value occurs at the new vertex, reflect the vertex with the second highest function value. This rule avoids oscillations between new and old vertices with the highest function values.
6. If one vertex persists for four iterations, reduce the size of the most recently formed simplex by halving the distances of the remaining vertices from that vertex. This step is called a contraction.
7. Stop after a prescribed number of contractions have been carried out.

N-dimensional case

Let the vertices of a regular simplex in E^n be $\underset{\sim}{X}_1$, $\underset{\sim}{X}_2$,..., $\underset{\sim}{X}_{n+1}$. The reflection of the vertex $\underset{\sim}{X}_P$ is defined as

$$\underset{\sim}{X}_q = 2\underset{\sim}{X}_C - \underset{\sim}{X}_P \qquad \text{(eq. D.3)}$$

where

$$\underset{\sim}{X}_c = (\underset{\sim}{X}_1 + \ldots + \underset{\sim}{X}_{P-1} + \underset{\sim}{X}_{P+1} + \ldots + \underset{\sim}{X}_{n+1})/n$$

is the centroid of the remaining vertices. Equation D.3 replaces equation D.1 in steps 3 and 5 above. A contraction about a persistent vertex X of a simplex by

$$\underset{\sim}{X}_q = 1/2(\underset{\sim}{X}_P + \underset{\sim}{X}_K)$$

that is, the vertex $\underset{\sim}{X}_K$ remains fixed while the linear dimensions of the simplex are halved.

A major disadvantage of Spendley, Hext and Himsworth's method is that the use of regular simplices limits the movement of the current point in each iteration. For example, once a favourable direction has been found, it would be more efficient to move further in that direction than the regular simplex pattern allows.

Nelder and Mead's method

In 1965, Nelder and Mead increased the efficiency of Spendley, Hext and Himsworth's method by allowing the simplices to become non-regular.

Consider again the problem of minimizing $f(\underset{\sim}{X})$. Let $\underset{\sim}{X}_1$ be an initial estimate of $\underset{\sim}{X}^*$ and let the vertices of the initial simplex be $\underset{\sim}{X}_1, \ldots, \underset{\sim}{X}_{n+1}$, where

$$\underset{\sim}{X}_{j+1} = \underset{\sim}{X}_1 + hj\underset{\sim}{e}j \qquad (j=1,\ldots,n)$$

the e_j are the usual unit coordinate vectors and the scalars h_j are chosen so as to equalize, as far as possible, the quantities

$$\left| f(\underset{\sim}{X}_l + h_j e_j) - f(\underset{\sim}{X}_l) \right|$$

in the current simplex, let

$\underset{\sim}{X}_h$ be the vertex with the highest function value,

$\underset{\sim}{X}_s$ be the vertex with the second highest function value,

$\underset{\sim}{X}_l$ be the vertex with the lowest function value,

$\underset{\sim}{X}_c$ be the centroid of all the vertices except $\underset{\sim}{X}_h$, i.e.

$$\underset{\sim}{X}_c = \frac{1}{n} \sum_{\substack{j=1 \\ j \neq h}}^{n+1} \underset{\sim}{X}_j$$

Also, let $y=f(\underset{\sim}{X})$, $y_h=f(\underset{\sim}{X}_h)$, etc. Then the procedure recommended by Nelder and Mead for minimization of $f(\underset{\sim}{X})$ is as follows :

(1) Choose the vertices of the initial simplex as described above and evaluate $f(\underset{\sim}{X})$ at each vertex.

(2) Reflection. Reflect $\underset{\sim}{X}_h$ (Figure D.2) using a reflection factor $\alpha > 0$, i.e. find $\underset{\sim}{X}_o$ such that

$$\underset{\sim}{X}_o - \underset{\sim}{X}_c = \alpha(\underset{\sim}{X}_c - \underset{\sim}{X}_h)$$

(3) If $y_l < y_o < y_s$, replace $\underset{\sim}{X}_h$ by $\underset{\sim}{X}_s$ and return to step (2).

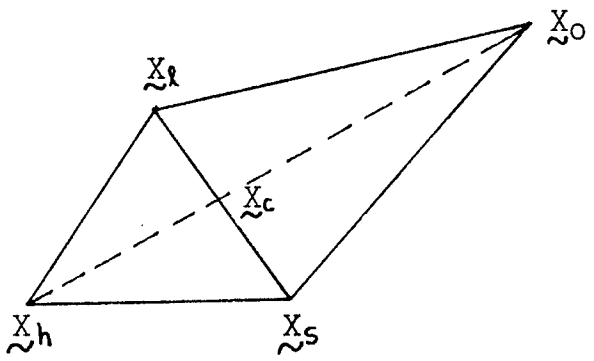


Figure D.2: Reflection of vertex $\underset{\sim}{X}_h$ in non-regular simplex

(4) Expansion. If $y_o < y_l$, expand the simplex (Figure D.3) using an expansion factor $\gamma > 1$, i.e. find $\underset{\sim}{X}_{oo}$ such that

$$\underset{\sim}{X}_{oo} - \underset{\sim}{X}_c = \gamma(\underset{\sim}{X}_o - \underset{\sim}{X}_c)$$

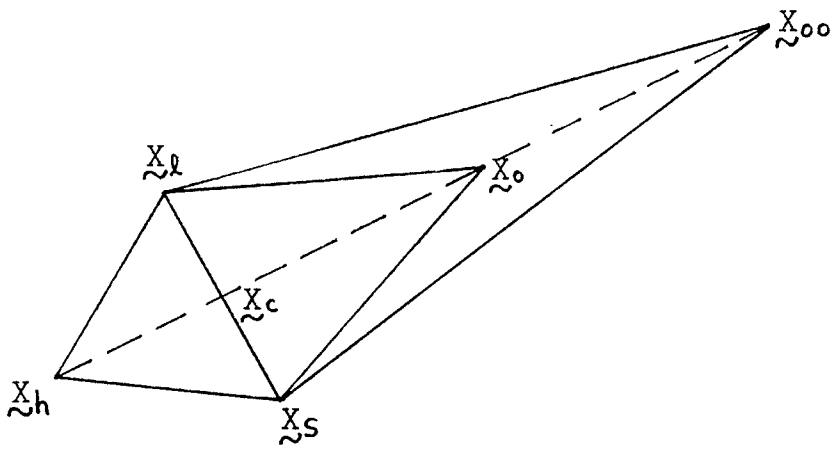


Figure D.3: Expansion of non-regular simplex

(a) If $y_{oo} < y_\ell$, replace $\underset{\sim}{X}_h$ by $\underset{\sim}{X}_{oo}$ and return to step (2).

(b) If $y_{oo} \geq y_\ell$, replace $\underset{\sim}{X}_h$ by $\underset{\sim}{X}_o$ and return to step (2).

(5) Contraction. If $y_o > y_s$, contract the simplex, using a contraction factor $\beta (0 < \beta < 1)$. There are two cases to consider :

(a) If $y_o < y_h$ (Figure D.4), find $\underset{\sim}{X}_{oo}$ such that

$$\underset{\sim}{X}_{oo} - \underset{\sim}{X}_c = \beta(\underset{\sim}{X}_o - \underset{\sim}{X}_c)$$

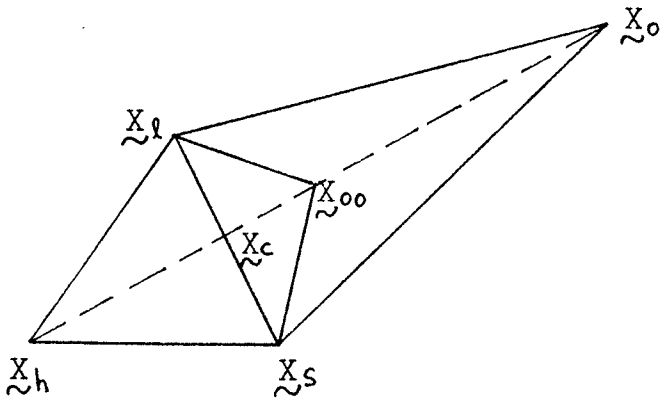


Figure D.4: Contraction of non-regular simplex when $y_s < y_o < y_h$

(b) If $y_o \geq y_h$ (Figure D.5), find $\underset{\sim}{X}_{oo}$ such that

$$\underset{\sim}{X}_{oo} - \underset{\sim}{X}_c = \beta(\underset{\sim}{X}_h - \underset{\sim}{X}_c)$$

Whether 5(a) or 5(b) is used, there are again two cases to consider :

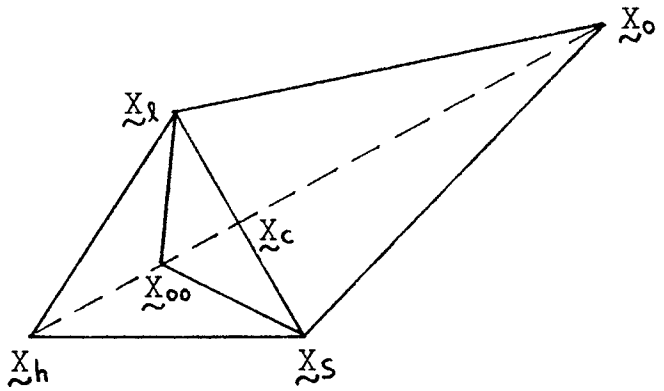


Figure D.5: Contraction of non-regular simplex when $y_o \geq y_h$

(c) If $y_{oo} < y_h$ and $y_{oo} < y_o$, replace $\underset{\sim}{X}_h$ by $\underset{\sim}{X}_{oo}$ and return to step (2).

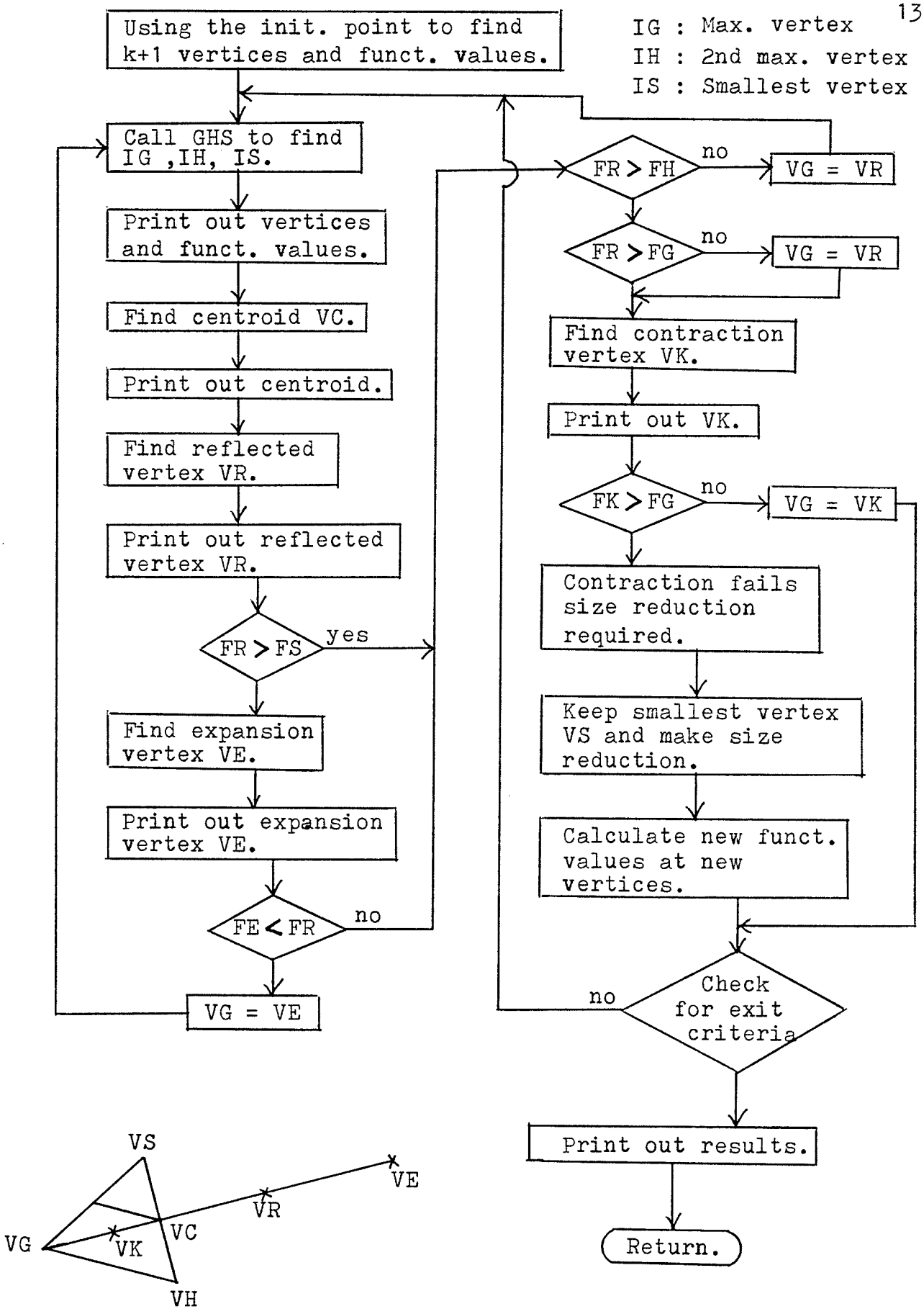
(d) If $y_{oo} > y_h$ or $y_{oo} > y_o$, reduce the size of the simplex by halving the distances from $\underset{\sim}{X}_l$ and return to step (2).

Nelder and Mead suggest values of $\alpha=1$, $\beta=0.5$, $\gamma=2$ for the reflection, contraction, and expansion factors, respectively. A convenient convergence criterion is that the calculations terminate when the standard deviation of $y_l, \ldots, y_{n+l}$ is less than some prescribed value $\epsilon > 0$, i.e. when

$$S = \sum_{k=1}^{n+1} \left\{ (y_k - \bar{y})^2 / n \right\}^{1/2} < \epsilon$$

where

$$\bar{y} = \left(\sum_{k=1}^{n+1} y_k \right) / (n+1) .$$



Flow chart of simplex.

```
      1. CSJOB  WATFIV  TURANLI,RUN=FREE,NOEXT
      2.        COMMON /IOUT/IN,IO,ID
      3.        DIMENSION V(11,10),VR(10),F(11),VE(10),VK(10),XO(10),VC(10)
      4.        IN=5
      5.        IO=6
      6.        NMX=10
      7.        NMXP1=NMX+1
      8.        READ (IN,10) ID,K,NFEMAX,A,ALPHA,BETA,GAMMA,EXIT
      9. C
     10. C      ------------------------------------------------------------
     11. C      WE HAVE JUST READ IN VALS. OF THE SPACE DIMENSION,SIMPLEX SIDE
     12. C      LENGTH AND THE REFLEXION,CONTRAXION AND EXPANSION COEFFS.
     13. C      WE NOW READ THE SIMPLEX STARTING VERTEX VALUE.
     14. C      ------------------------------------------------------------
     15. C
     16.        READ (IN,20) (XO(I),I=1,K)
     17.        CALL SIMPLX(V,VC,VE,VK,VR,F,XO,A,ALPHA,BETA,GAMMA,EXIT,K,NFEMAX,
     18.       &NMX,NMXP1)
     19. C
     20.        STOP
     21. 10     FORMAT (3I5/4F5.0,F10.0)
     22. 20     FORMAT (8F10.0)
     23.        END
     24.        SUBROUTINE SIMPLX(V,VC,VE,VK,VR,F,XO,A,ALPHA,BETA,GAMMA,EXIT,K,
     25.       &NFEMAX,NMX,NMXP1)
     26. C
     27. C      ------------------------------------------------------------
     28. C      PROGRAM FOR MINIMIZATION USING THE SIMPLEX METHOD.
     29. C      ------------------------------------------------------------
     30. C
     31. C      ID : DEBUGGING PARAMETER.
     32. C           0 NO PRINT-OUT WILL BE GIVEN,
     33. C           1 LIMITED PRINT-OUT WILL BE GIVEN,
     34. C           2 FULL PRINT-OUT WILL BE GIVEN.
     35. C
     36. C      A  : SIDE LENGTH.
     37. C      ALPHA,BETA,GAMMA VALUES ARE RECOMMENDED AS 1.0,0.5,2.0.
     38. C      EXIT CRITERIA IS ROOT SUM SQUARED OF THE DIFFERENCE OF
     39. C      VERTEX FUNCTION VALUE AND CENTROID FUNCTION VALUE OVER K.
     40. C
     41.        COMMON /IOUT/IN,IO,ID
     42.        DIMENSION V(NMXP1,NMX),VE(1),VK(1),XO(1),VC(1),F(1),VR(1)
     43.        DATA PI,PR,PE,PC,PS/'I','R','E','C','S'/
     44.        WRITE (IO,270) ID,K,NFEMAX,A,ALPHA,BETA,GAMMA,EXIT,(XO(I),I=1,K)
     45.        K1=K+1
     46.        POP=PI
     47.        NFEVAL=0
     48. C
     49. C      ------------------------------------------------------------
     50. C      K1 IS THE NO. OF VERTICES AND NFEVAL THE NO. OF FUNC. EVALUATIONS.
     51. C      CREATION OF STARTING SIMPLEX:THE VERTEX VECTORS ARE ROWS OF THE
     52. C      MATRIX V
     53. C      ------------------------------------------------------------
     54. C
     55.        Q=A*(SQRT(K1*1.0)-1)/K/SQRT(2.0)
PAUSE:
```

```
      P=Q+A/SQRT(2.0)
      DO 10 J=1,K
10    V(1,J)=XO(J)
      DO 20 I=2,K1
      DO 20 J=1,K
      V(I,J)=Q+XO(J)
20    IF (I.EQ.(J+1)) V(I,J)=P+XO(J)
C
C     ---------------------------------------------------------
C     FUNCTION EVALUATIONS THRU SUBROUTINE FUNC.
C     ---------------------------------------------------------
C
      DO 40 I=1,K1
      DO 30 J=1,K
30    XO(J)=V(I,J)
      CALL FUNCT(FDD,XO,NFEVAL)
40    F(I)=FDD
C
C     ---------------------------------------------------------
C     WE NOW FIND THE MAX,MAX BUT ONE AND MIN VERTICES,V(IG,J),V(IH,J)
C     V(IS,J) FOR VARYING J.
C     ---------------------------------------------------------
C
50    IF (NFEVAL.GE.NFEMAX) ID=2
      IF (NFEVAL.GE.(NFEMAX+20)) GO TO 250
      CALL GHS(F,K1,IG,IH,IS)
      IF (ID.EQ.0) GO TO 70
      WRITE (IO,280) NFEVAL,POP,(I,I=1,K1)
      DO 60 J=1,K
60    WRITE (IO,290) (V(I,J),I=1,K1)
      WRITE (IO,300) (F(I),I=1,K1)
      IF (ID.GE.2) WRITE (IO,310) IG,IH,IS
C
C     ---------------------------------------------------------
C     FINDING THE CENTROID AFTER ISOLATING LARGEST VALUED VERTEX
C     ---------------------------------------------------------
C
70    DO 90 J=1,K
      VC(J)=0.0
      DO 80 I=1,K1
80    IF (I.NE.IG) VC(J)=V(I,J)+VC(J)
90    VC(J)=VC(J)/K
      IF (ID.GE.2) WRITE (IO,320) (VC(J),J=1,K)
C
C     ---------------------------------------------------------
C     WE NOW REFLECT VG INTO VR AND EVALUATE FUNC(VR)
C     ---------------------------------------------------------
C
      DO 100 J=1,K
100   VR(J)=(1+ALPHA)*VC(J)-ALPHA*V(IG,J)
      CALL FUNCT(FR,VR,NFEVAL)
C
      IF (ID.GE.2) WRITE (IO,330) (VR(J),J=1,K),FR
C
C     ---------------------------------------------------------
PAUSE:
```

```
111. C     WE NOW CHECK FOR SUCESSFUL REFLEXION.
112. C     ------------------------------------------------------------
113. C
114.       IF (FR.GT.F(IS)) GO TO 130
115. C
116. C     ------------------------------------------------------------
117. C     IF THE REFLEXION SUCCEEDS WE ACCELERATE TO VE.
118. C     ------------------------------------------------------------
119. C
120.       DO 110 J=1,K
121. 110   VE(J)=(1.0-GAMMA)*VC(J)+GAMMA*VR(J)
122.       CALL FUNCT(FE,VE,NFEVAL)
123.       IF (ID.GE.2) WRITE (IO,340) (VE(J),J=1,K),FE
124. C
125. C     ------------------------------------------------------------
126. C     IF STEP'S A SUCCESS,VE IS THE NEW VERTEX,ELSE VR IS
127. C     ------------------------------------------------------------
128. C
129.       IF (FE.GT.FR) GO TO 130
130.       DO 120 J=1,K
131. 120   V(IG,J)=VE(J)
132.       POP=PE
133.       F(IG)=FE
134.       GO TO 50
135. 130   IF (FR.GT.F(IH)) GO TO 150
136.       DO 140 J=1,K
137. 140   V(IG,J)=VR(J)
138.       POP=PR
139.       F(IG)=FR
140.       GO TO 50
141. C
142. C     ------------------------------------------------------------
143. C    THIS MEANS THAT A CONTRAXION IS CALLED FOR.
144. C     ------------------------------------------------------------
145. C
146. 150   IF (FR.GT.F(IG)) GO TO 170
147.       DO 160 J=1,K
148. 160   V(IG,J)=VR(J)
149.       POP=PR
150.       F(IG)=FR

152. C
153. C     ------------------------------------------------------------
154. C     HERES THE CONTRAXION
155. C     ------------------------------------------------------------
156. C
157. 170   DO 180 J=1,K
158. 180   VK(J)=BETA*V(IG,J)+(1-BETA)*VC(J)
159.       CALL FUNCT(FK,VK,NFEVAL)
160.       IF (ID.GE.2) WRITE (IO,350) (VK(J),J=1,K),FK
161. C
162. C     ------------------------------------------------------------
163. C     IF THE CONTRACTION'S A SUCCESS USE VK AS NEW VERTEX.
164. C     ------------------------------------------------------------
165. C
PAUSE:
```

```
      IF (FK.GT.F(IG)) GO TO 200
      DO 190 J=1,K
190   V(IG,J)=VK(J)
      POP=PC
      F(IG)=FK
      GO TO 230
C
C     ------------------------------------------------------------
C     IF THE CONTRACION FAILS REDUCE SIZE OF SIMPLEX
C     ------------------------------------------------------------
C
200   IF (ID.GE.2) WRITE (IO,360)
      DO 220 I=1,K1
      IF (I.EQ.IS) GO TO 220
      DO 210 J=1,K
      V(I,J)=(V(I,J)+V(IS,J))/2.0
210   XO(J)=V(I,J)
      CALL FUNCT(FDD,XO,NFEVAL)
      F(I)=FDD
220   CONTINUE
      POP=PS
C
C     ------------------------------------------------------------
C     TIME TO EXIT? FOLLOWING STEPS CHECK FOR EXIT CRITERION
C     ------------------------------------------------------------
C
230   SUM=0.0
      DO 240 J=1,K1
240   SUM=SUM+(F(J)-F(IS))*(F(J)-F(IS))
      SUM=SQRT(SUM/K1)
      IF (SUM.GT.EXIT) GO TO 50
      WRITE (IO,370) NFEVAL
      GO TO 260
250   WRITE (IO,390) NFEVAL
260   WRITE (IO,380) (V(IS,J),J=1,K),F(IS)
      RETURN
270   FORMAT (1H1/1X,70('-')/2X,'PROGRAM FOR MINIMIZATION USING THE ','S
     &IMPLEX METHOD'/1X,70('-')///5X,'INPUT VALUES:'//5X,'ID   =',I4/5X,
     &'K    =',I4/5X,'NFEMX=',I4/5X,'A    =',F8.3/5X,'ALPHA=',F8.3/5X,'B
     &ETA =',F8.3/5X,'GAMMA=',F8.3/5X,'EXIT =',F12.7//5X,10F10.5//)
280   FORMAT (1X,70('-')/1X,I3,1X,A1,5X,9('V',I1,9X))
290   FORMAT (7X,9(F10.6,1X))
300   FORMAT (/7X,9(F10.6,1X))
310   FORMAT (1H  ,' IG=',I2,' IH=',I2,' IS=',I2)
320   FORMAT (1H  ,'COORDS. OF CENTROID='/1H  ,10F10.4)
330   FORMAT (1H  ,'REFLECTED POINT AND FUNC VALUE THERE'/1H  ,8F10.4,
     &F10.3)
340   FORMAT (1H  ,'ACCELERATED POINT='/10F8.2,'F=',F10.2)
350   FORMAT (1H  ,'CONTRAXION CALLED FOR'/1H  ,8F10.4,F10.4)
360   FORMAT (1X,'CONTRACTION FAILS.SIZE REDUCTION REQUIRED')
370   FORMAT (/////1H  ,'SYSTEM CONVERGED,NUMBER OF FUNCTION EVALUATIONS
     &  =',I3)
380   FORMAT (/2X,11F10.5)
390   FORMAT (///2X,'**** ERROR ****'/2X,'NUMBER OF FUNCTION EVALUATIONS
     & EXCEEDED,',2X,'NFEVAL=',1X,I4//)
PAUSE:
```

```
      END
      SUBROUTINE GHS(F,K1,IG,IH,IS)
      DIMENSION F(1)
      IG=1
      IS=1
      DO 10 I=2,K1
      IF (F(I).LT.F(IS)) IS=I
      IF (F(I).GT.F(IG)) IG=I
10    CONTINUE
      IH=IS
      DO 20 I=1,K1
      IF (I.EQ.IG) GO TO 20
      IF (F(I).GT.F(IH)) IH=I
20    CONTINUE
      RETURN
      END
      SUBROUTINE FUNCT(F,X,NFEVAL)
C
      DIMENSION X(1)
      F=100*(X(2)-X(1)*X(1))*(X(2)-X(1)*X(1))+(1-X(1))*(1-X(1))
      NFEVAL=NFEVAL+1
      RETURN
      END
:
```

Appendix E

THE PATTERN SEARCH METHOD OF HOOKE AND JEEVES (1961)

The Pattern Search Method of Hooke and Jeeves attempts to align a search direction with the principal axis of the objective function. In this attempt, two strategies, known as exploratory moves and pattern moves are used alternately.

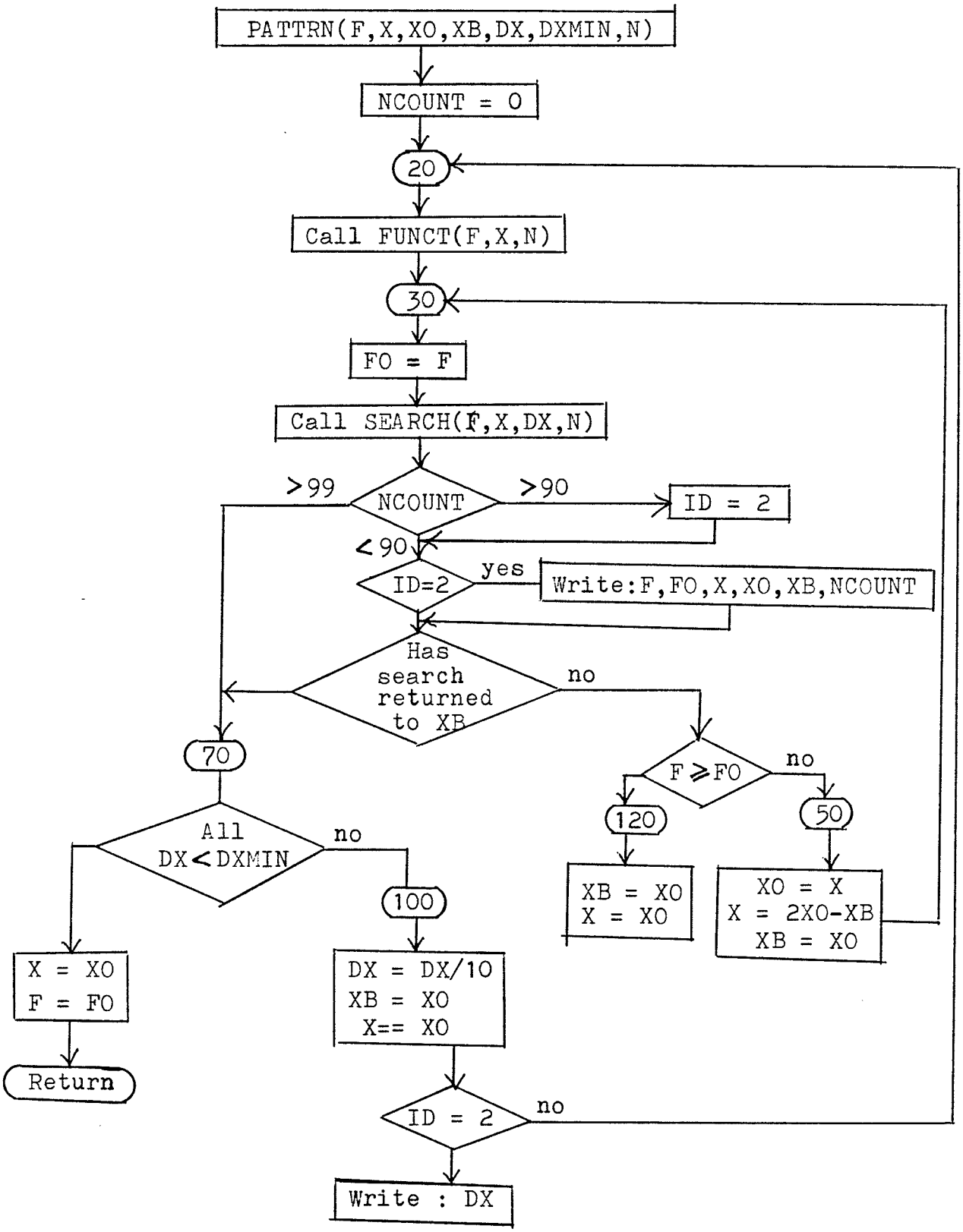
The initial point provided forms the first base point b_1, at which the function value is f_1. In the exploratory moves, each variable is considered in turn, with the ith variable x_i being perturbed by an amount d_i. If a smaller function value results from this step, this trial point becomes the current point and the next variable is considered. If a greater function value is obtained, the trial point is rejected, the sign of d_i is changed, and a perturbation applied to the ith variable in the opposite sense. Again only if a smaller function value is obtained does this trial point become the current point, and in either case the next variable is then considered. The exploratory moves are completed when all n variables have been considered in turn. The final current point becomes a new base point b_2, with corresponding function value f_2.

The exploratory moves have resulted in the current point travelling from b_1 to b_2. Hooke and Jeeves' idea is to investigate whether any further progress is prossible in the direction b_2-b_1 , in the hope that this direction approximates to the local principal axis of the objective function. Accordingly a pattern move is made to the point $2b_2 - b_1$. Hooke and Jeeves do not compare the function value at this point with f_2, but instead endeavour to improve on this pattern direction by performing another set of exploratory moves from this point, to yield a new base point b_3 at which the function value is f_3.

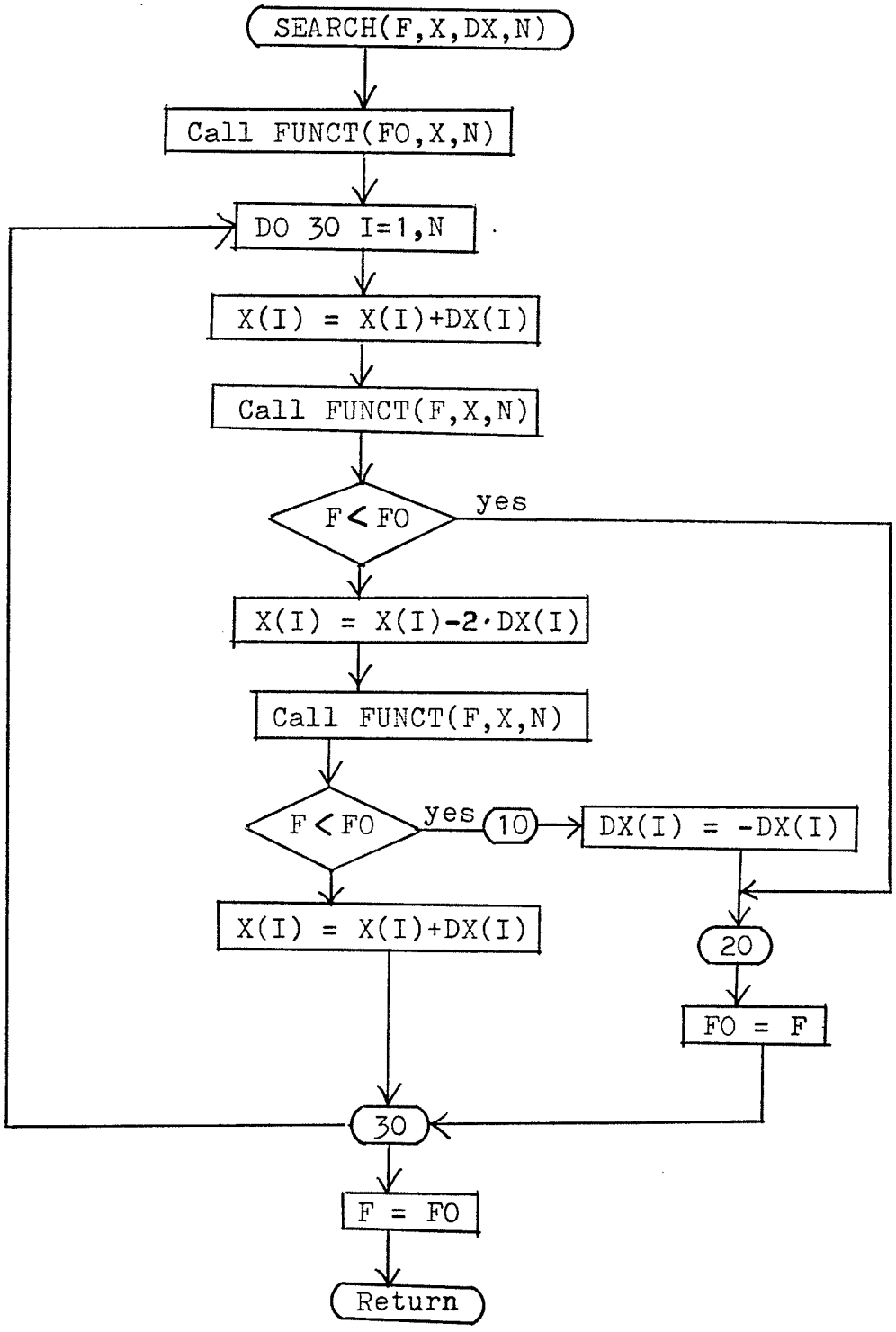
If $f_3 > f_2$, the pattern move plus exploratory moves are deemed a failure, and the whole procedure is recommenced with exploratory moves from the base point b_2. If however $f_3 \leq f_2$, then the direction from b_2 to b_3 is considered worthy of further investigation, and a new pattern move is made to the point $2b_3-b_2$, and the process continued with a set of exploratory moves starting from this point, as before.

The above rules are employed until a set of exploratory moves about a base point (as opposed to a point obtained from a pattern move) all fail. This is taken to indicate that either the minimum has been located to the accuracy of the d_i, or the search has descended into a steep skew valley which cannot be negotiated using the current step sizes. The remedy in either cases is to reduce the step sizes d_i, and recommence the whole procedure from the current point. Con-

vergence is assumed when the step-lengths have been reduced by pre-assigned factors.



Flow chart of Pattern Search


SEARCH(F,X,DX,N)
Call FUNCT(FO,X,N)
DO 30 I=1,N
X(I) = X(I)+DX(I)
Call FUNCT(F,X,N)
F < FO
yes
X(I) = X(I)-2·DX(I)
Call FUNCT(F,X,N)
F < FO
yes
10
DX(I) = -DX(I)
X(I) = X(I)+DX(I)
20
FO = F
30
F = FO
Return

```
 1. C$JOB  WATFIV  TURANLI,RUN=FREE,NOEXT
 2.        COMMON /IOUT/IN,IO,ID
 3.        DIMENSION X(16),XO(16),XB(16),DX(16)
 4. C
 5. C
 6.        IN=5
 7.        IO=6
 8.        READ (IN,10) N,ID,DXMIN
 9.        READ (IN,20) (X(I),I=1,N)
10.        READ (IN,20) (DX(I),I=1,N)
11. C
12.        CALL PATTRN(F,X,XO,XB,DX,DXMIN,N)
13. C
14.        STOP
15. 10     FORMAT (2I5,F5.0)
16. 20     FORMAT (16F5.0)
17.        END
18.        SUBROUTINE PATTRN(F,X,XO,XB,DX,DXMIN,..,
19. C
20.        COMMON /IOUT/IR,IO,ID
21.        DIMENSION X(1),XO(1),XB(1),DX(1)
22. C
23.        NCOUNT=0
24.        NFEVAL=0
25. C
26.        IF (ID.GE.0) WRITE (IO,180) N,ID,DXMIN
27.        IF (ID.GE.0) WRITE (IO,140) (X(I),I=1,N)
28.        DO 10 I=1,N
29.        XO(I)=X(I)
30. 10     XB(I)=X(I)
31. 20     CALL FUNCT(F,X,NFEVAL)
32. 30     FO=F
33.        CALL SEARCH(F,X,DX,N,NFEVAL)
34.        NCOUNT=NCOUNT+1
35.        IF (NCOUNT.GT.90) ID=2
36.        IF (NCOUNT.GT.99) GO TO 70
37.        IF (ID.EQ.2) WRITE (IO,140) F,FO,(X(I),I=1,N)
38.        ICH=3
39.        DO 40 I=1,N
40.        AA=(XO(I)-X(I))/DX(I)
41. 40     IF (ABS(AA).GT.0.9) ICH=1
42.        IF (ID.GE.2) WRITE (IO,160) ICH,NCOUNT,(X(I),XO(I),XB(I),I=1,N)
43.        IF (ICH.EQ.3) GO TO 70
44.        IF (F.GE.FO) GO TO 120
45. 50     DO 60 I=1,N
46.        XO(I)=X(I)
47.        X(I)=2.*X(I)-XB(I)
48. 60     XB(I)=XO(I)
49.        GO TO 30
50. 70     ICH=0
51.        DO 30 I=1,N
52. 80     IF (ABS(DX(I)).GT.DXMIN) ICH=1
53.        IF (ICH.EQ.1) GO TO 100
54.        DO 90 I=1,N
55. 90     X(I)=XO(I)
```

```
      F=FO
      IF (ID.GE.0) WRITE (IO,170) ICH,NFEVAL,F,(X(I),I=1,N)
      RETURN
100   DO 110 I=1,N
      DX(I)=0.1*DX(I)
      XB(I)=XO(I)
110   X(I)=XO(I)
      NCOUNT=0
      IF (ID.GE.2) WRITE (IO,150) (DX(I),I=1,N)
      GO TO 20
120   DO 130 I=1,N
      XB(I)=XO(I)
130   X(I)=XO(I)
      GO TO 20
140   FORMAT (10E15.7/8E15.7)
150   FORMAT (/' REDUCED DX TO:',8E15.6/8E15.6)
160   FORMAT (2I4,8E15.7/8E15.7)
170   FORMAT (///2X,'ICH=',I2,3X,'NFEVAL=',I4,3X,'VALUE OF FUNCTION ','F
     &OR X VARIABLES BELOW=',F13.7//8E15.7/8E15.7)
180   FORMAT (1H1///1X,70('-')/1X,'PATTERN-SEARCH METHOD ','FOR NONLINEA
     &R MINIMIZATION'/,1X,'PROGRAMMED BY R.W.MENZIES'/1X,70('-')///5X,'I
     &NPUT DATA:',/5X,11('-')//5X,'N    =',I3/5X,'ID   =',I3/5X,'DXMIN='
     &,F8.4/)
      END
      SUBROUTINE SEARCH(F,X,DX,N,NFEVAL)
C
      DIMENSION X(1),DX(1)
C
      CALL FUNCT(FO,X,NFEVAL)
      DO 30 I=1,N
      X(I)=X(I)+DX(I)
      CALL FUNCT(F,X,NFEVAL)
      IF (F.LT.FO) GO TO 20
      X(I)=X(I)-2.*DX(I)
      CALL FUNCT(F,X,NFEVAL)
      IF (F.LT.FO) GO TO 10
      X(I)=X(I)+DX(I)
      GO TO 30
10    DX(I)=-DX(I)
20    FO=F
30    CONTINUE
      F=FO
      RETURN
      END
      SUBROUTINE FUNCT(F,X,NFEVAL)
C
      DIMENSION X(1)
      F=100*(X(2)-X(1)*X(1))*(X(2)-X(1)*X(1))+(1-X(1))*(1-X(1))
      NFEVAL=NFEVAL+1
      RETURN
      END
```

Appendix F

THE GOLDEN SECTION SEARCH METHOD

Golden section search is a single variable search technique which, while nearly as effective as the Fibonacci method, is completely independent of the number of experiments available.

Let j represent the number of experiments already run. The experimental plan should place successive experiments such that :

$$L_{j-1} = L_j + L_{j+1} \qquad \text{(eq. F.1)}$$

Let the ratio of successive lengths be constant. Calling this ratio τ , we have

$$L_{j-1}/L_j = L_j/L_{j+1} = \tau$$

by dividing equation F.1 throughout by L_{j+1} and noting that

$$L_{j-1}/L_{j+1} = \tau^2$$

we obtain $\tau^2 = \tau + 1$ as shown in figure F.1. Only one root of this quadratic equation is positive, and so we see that

$$\tau = (1+\sqrt{5})/2 = 1.618033989...$$

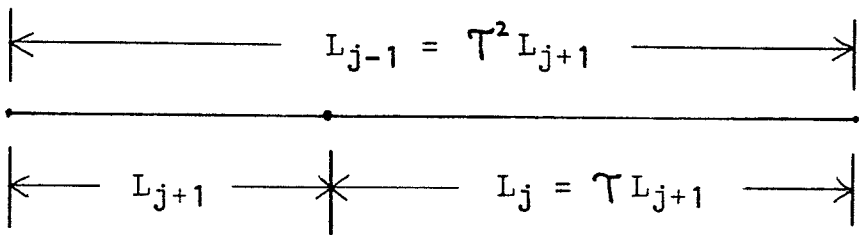
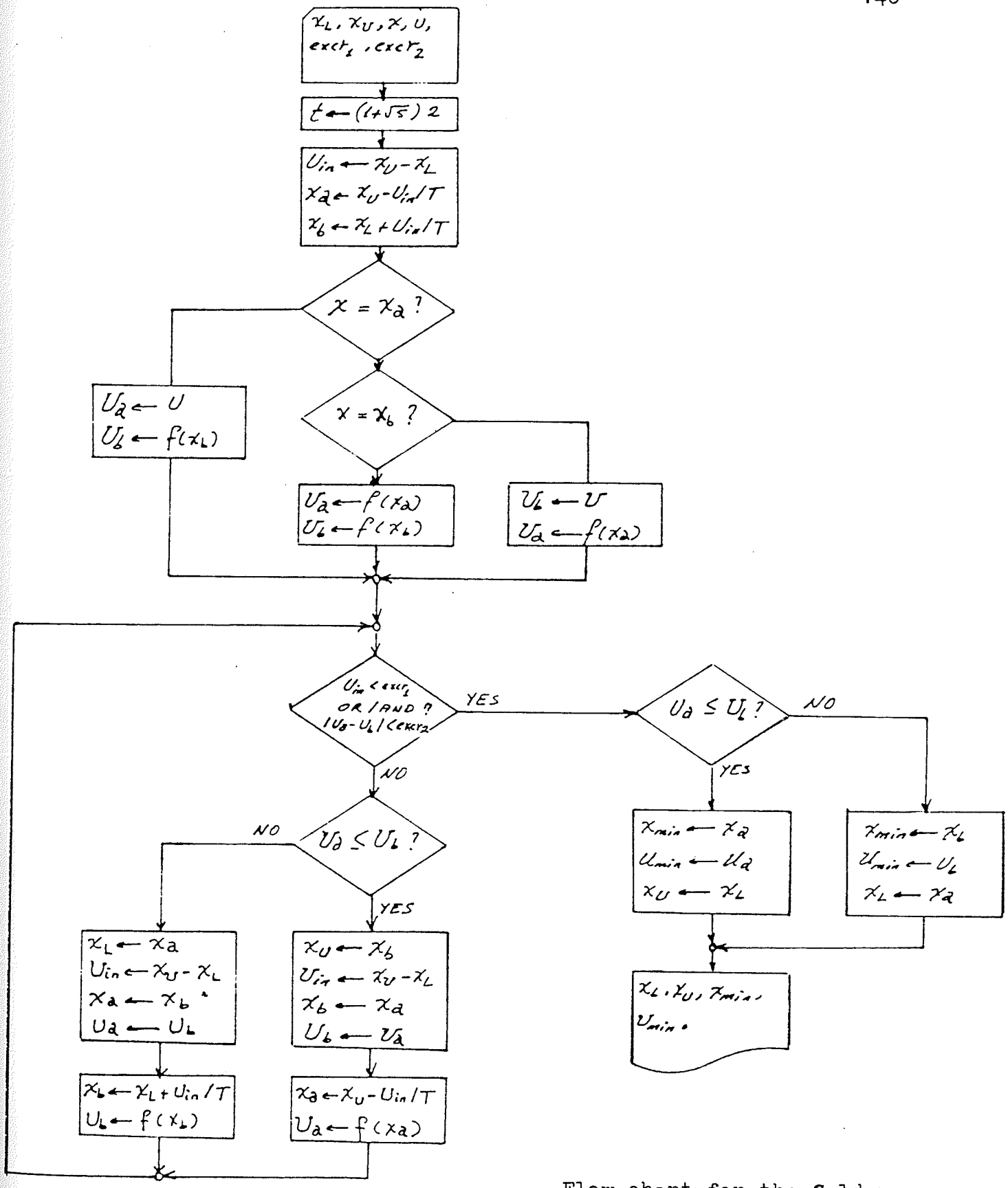


Figure F.1: The Golden Section.

The results of the two experiments will determine which segment is to be explored further. This remaining segment will contain one of the previous trails, and to continue the search one merely places the next experiments symmetrically in the interval. Once begun, this process may be continued as long as desired. After n experiments the interval L_n remaining is given by

$$L_n = (1/\tau^{n-1}).$$



Flow chart for the Golden Section Search method.

```
      SUBROUTINE GOLDEN(XL,XU,X,U,NMAX,EXCR1,EXCR2,KEXIT,ID)
C
C     ***************************************************************
C
C     DESCRIPTION OF THE PARAMETERS
C
C     XL     : LOWER LIMIT OF THE INITIAL UNIMODAL INTERVAL
C              CARRIES THE LOWER LIMIT OF THE FINAL INTERVAL ON RETURN
C     XU     : UPPER LIMIT OF THE INITIAL UNIMODAL INTERVAL
C              CARRIES THE UPPER LIMIT OF THE FINAL INTERVAL ON RETURN
C     X      : CARRIES THE ESTIMATED VALUE FOR XMIN ON RETURN
C              EQUATE X TO 0.0 WHEN CALLING SUBR. GOLDEN
C     U      : CARRIES THE MINIMUM FUNCTION VALUE ON RETURN
C              EQUATE U TO 0.0 WHEN CALLING SUBR. GOLDEN
C     NMAX   : MAXIMUM NUMBER OF FUNCTION EVALUATIONS
C     EXCR1  : EXIT CRITERIA FOR THE MINIMUM LENGTH OF THE FINAL INTERVAL
C     EXCR2  : EXIT CRITERIA FOR THE DIFFERENCE IN THE TWO
C              CONSECUTIVE FUNCTION EVALUATIONS
C     KEXIT  : PARAMETER TO SPECIFY THE REQUIRED EXIT CONDITIONS
C              1 : MINIMUM FINAL INTERVAL
C              2 : MINIMUM CHANGE IN FUNCTION VALUES
C              3 : BOTH
C     ID     : DEBUGGING PARAMETER
C              0 : NO PRINTOUT
C              1 : ONLY FINAL VALUES ARE PRINTED
C              2 : DETAILED PRINTOUT
C
C     ***************************************************************
C
      N=0
      IO=6
      T=0.5*(1.0+SQRT(5.0))
      IF(ID.GE.1)WRITE(IO,61)XL,XU,T,X,U,NMAX,EXCR1,EXCR2,KEXIT
      IF(ID.GE.2)WRITE(IO,60)
      DIN=XU-XL
      XA=XU-DIN/T
      XB=XL+DIN/T
      IF((XA.EQ.0.0).OR.(XB.EQ.0.0))GO TO 82
      IF(ABS((X-XA)/XA).LT.1.E-4)GO TO 10
      IF(ABS((X-XB)/XB).LT.1.E-4)GO TO 20
   82 CALL FUNCT(XA,UA,N)
      CALL FUNCT(XB,UB,N)
      GO TO 29
   10 CALL FUNCT(XB,UB,N)
      UA=U
      GO TO 29
   20 CALL FUNCT(XA,UA,N)
```

```
      UB=U
   29 IS1=0
      IS2=0
   30 IF(ID.GE.2)WRITE(IC,62)XL,XA,XB,XU,UA,UB,N
      IF(N.GT.NMAX)GO TO 999
      GO TO(1,2,3),KEXIT
    1 IF(UIN.LT.EXCB1)GO TO 90
      GO TO 15
    2 IF(ABS(UA-UB).LT.EXCB2) GO TO 90
      GO TO 15
    3 IF(IS1)11,11,12
   11 IF(UIN-EXCB1)13,12,12
   13 IS1=1
      IF(ID.GE.2)WRITE(IC,25)EXCB1
   12 IF(IS2)14,14,16
   14 IF(ABS(UA-UB)-EXCB2)17,16,16
   17 IS2=1
      IF(ID.GE.2)WRITE(IC,35)EXCB2
   16 IF((IS1+IS2).GE.2)GO TO 90
C
   15 IF(UA.LT.UB)GO TO 40
      XL=XA
      UIN=XU-XL
      XA=XB
      UA=UB
      XB=XL+UIN/T
      CALL FUNCT(XB,UB,N)
      GO TO 30
```

```
C
   40 XU=XB
      DIN=XU-XL
      XB=XA
      UB=UA
      XA=XU-DIN/T
      CALL FUNCT(XA,UA,N)
      GO TO 30
   90 IF(UA.LT.UB)GO TO 91
      U=UB
      X=XB
      XL=XA
      GO TO 99
   91 U=UA
      X=XA
      XU=XB
      GO TO 99
C
   99 IF(ID.GE.1)WRITE(IO,63)U,X,N,XL,XU
      WRITE(IO,65)
      RETURN
  999 WRITE(IO,64)XL,XA,XB,XU,UA,UB
      STOP
C
   25 FORMAT(//,30X,'EXCR1 =',E12.5,3X,'IS SATISFIED',//)
   35 FORMAT(//,30X,'EXCR2 =',E12.5,3X,'IS SATISFIED',//)
   60 FORMAT(////,10X,'XI', 14X,'XA', 14X,'XB', 14X,'XU', 14X,'UA', 14X,
     A 'UB',12X,'N',/,10X,5(2('_'), 14X),2('_'),12X,'_',//)
C
   61 FORMAT(1H1,/////,25X,55('_')///,30X,'GOLDEN SECTION  INTERVAL',
     A ' REDUCTION SUBROUTINE',//,25X,55('_'),///,25X,'XL =',E12.5,8X,
     A 'XU =',E12.5,8X,'T =',E12.5,//,25X,'X =',E12.5,9X,'U =',E12.5,
     A 9X,'NMAX =',I4,//,25X,'EXCR1 =',E12.5,5X,'EXCR2 =',E12.5,5X,
     A 'KEXIT =',I3)
C
   62 FORMAT( 5X,6(E12.5,4X),I4,/)
C
   63 FORMAT(////,15X,'MINIMUM FUNCTION VALUE IS',E12.5,2X,'AT  X =',
     A E12.5,2X,'IN',I4,2X,'FUNCTION EVALUATIONS',////,15X,'FINAL ',
     A 'INTERVAL IS',2X,E12.5,' < XMIN <',E12.5)
C
   64 FORMAT(//,25X,'MAX. NO. OF FUNCT EVALUATIONS ARE EXCEEDED FINAL',
     A'VALUES ARE :',//,25X,'XL =',E12.5,5X,'XA =',E12.5,5X,'XB =',E12.5
     A,5X,'XU =',E12.5,//,44X,'UA =',E12.5,5X,'UB =',E12.5)
   65 FORMAT(1H1)
      END
```

Appendix G

THE LEAST ERROR SQUARE CURVE FITTING METHOD - ALGORITHM DESCRIPTION

IMSL ROUTINE NAME - IFLSQ

PURPOSE - LEAST SQUARES APPROXIMATION WITH USER SUPPLIED FUNCTIONS

USAGE - CALL IFLSQ (F,X,Y,M,A,N,WK,IER)

ARGUMENTS

F - NAME OF THE REAL FUNCTION SUBPROGRAM FOR EVALUATING THE BASIS FUNCTIONS. (INPUT) THE FUNCTION ITSELF MUST BE SUPPLIED BY THE USER AND IT SHOULD BE OF THE FOLLOWING FORM

```
REAL FUNCTION F(K,X)
INTEGER K
REAL X
   .
   .
   .
```

IT IS ASSUMED THAT THE USER WANTS TO APPROXIMATE THE DATA BY A SERIES OF THE FORM

A(1)*F(1,X)+A(2)*F(2,X)+...+A(N)*F(N,X)

F MUST APPEAR IN AN EXTERNAL STATEMENT IN THE CALLING PROGRAM AND K AND X MUST NOT BE ALTERED BY F.

X - VECTOR OF LENGTH M CONTAINING THE ABSCISSAE OF THE DATA POINTS (X(I),Y(I)),I=1,...,M. (INPUT)

Y - VECTOR OF LENGTH M CONTAINING THE ORDINATES (OR FUNCTION VALUES) OF THE DATA POINTS. (INPUT)

M - NUMBER OF DATA POINTS. (INPUT)

A - VECTOR OF LENGTH N CONTAINING THE COEFFICIENTS OF THE BASIS FUNCTIONS. (OUTPUT)

N - NUMBER OF BASIS FUNCTIONS. (INPUT) N MUST BE LESS THAN OR EQUAL TO M.

WK - WORK ARRAY OF LENGTH N*(N+3).

IER - ERROR PARAMETER. (OUTPUT) TERMINAL ERROR

IER = 129, THE PROBLEM DOES NOT HAVE A UNIQUE SOLUTION. THIS WILL ALWAYS OCCUR IF N IS GREATER THAN M.

IER = 130, THE PROBLEM IS ILL-CONDITIONED. THAT IS, THE BASIS FUNCTIONS (RESTRICTED TO THE DATA SET) ARE ALMOST LINEARLY DEPENDENT.

PRECISION/HARDWARE - SINGLE AND DOUBLE/H32
- SINGLE/H36,H48,H60

REQD. IMSL ROUTINES - SINGLE/LEQT2P,LUDECP,LUELMP,LUREFP,UERTST, UGETIO
- DOUBLE/LEQT2P,LUDECP,LUELMP,LUREFP,UERTST, UGETIO,VXADD,VXMUL,VXSTO

NOTATION - INFORMATION ON SPECIAL NOTATION AND CONVENTIONS IS AVAILABLE IN THE MANUAL INTRODUCTION OR THROUGH IMSL ROUTINE UHELP

Algorithm

IFLSQ finds coefficients a_i, i=1,2,...,n such that

$$\sum_{j=1}^{m} \{a_1 f_1(x_j)+\ldots+a_n f_n(x_j)-y_j\}^2$$

is minimized.

Programming Notes

If polynomial approximation is desired, an orthogonal polynomial basis should normally be used, since the basis set F(K,X) = X**(K-1) leads to a very ill-conditioned linear system for moderate or large N. The following Fortran function defines a set of Legendre polynomials which are orthogonal over the interval (XA,XB) (set XA=X(1), XB=X(M) normally).

```
      REAL FUNCTION F(K,X)
      INTEGER   K,I
      REAL      X,XA,XB,T,PKM2,PKM1,RI
      DATA      XA,XB/.../
      T = (2.*X-XA-XB)/(XB-XA)
      F = 1.0
      IF(K.EQ.1)RETURN
      F = T
      IF(K.EQ.2)RETURN
      PKM2 = 1.0
      PKM1 = T
      DO 5 I = 3,K
         RI = I
         F = ((2.*RI-3.)*T*PKM1-(RI-2.)*PKM2)/(RI-1.)
         PKM2 = PKM1
         PKM1 = F
    5 CONTINUE
      RETURN
      END
```

Example

The data points are $(x_j,g(x_j)),j=1,20$ where $g(x)=xe^{-4x}$ and $x_j = j/20$. A linear combination of the functions sin (PI·x), sin(2·PI·x),...,sin (18·PI·x) is to be found which approximates the data where PI=3.14159.

Input:

```
      INTEGER    M,N,IER,I
      REAL       F,X(20),Y(20),A(18),WK(378),RM,RI
      EXTERNAL   F
      M  = 20
      N  = 18
      RM = M
      DO 5 I=1,M
         RI=I
         X(I) = RI/RM
    5 CONTINUE
C                                G(X(I))=X(I)*EXP(-4.0*X(I))
      DO 10 I = 1,M
         Y(I) = X(I)*EXP(-4.0*X(I))
   10 CONTINUE
      CALL IFLSQ (F,X,Y,M,A,N,WK,IER)
         .
         .
      END

      REAL FUNCTION F(K,X)
      INTEGER     K
      REAL        X,RK
      RK  =  K
      F   =  SIN(RK*3.14159*X)
      RETURN
      END
```

Output:

```
      IER = 0

      A   = (.0809,.0280,.0172,.0040,.0058,.0004,
             .0029,-.0003,.0017,-.0004,.0011,-.0004,
             .0008,-.0003,.0005,-.0002,.0003,-.0001)
```

Appendix H

THE NON-LINEAR CONTINUOUS MODEL - PROGRAM LISTING

```
1. //TNG JOB '1222335,,T=05'
2. //  EXEC FORTXCLG,OPT=2,USERLIB='MENZIES.MAIN.LOAD'
3. //FORT.SYSIN DD *
4. C
5. C FILENAME : MODL2M
6. C FINDING COEFFICIENCES OF THE FOLLOWING MODEL BY MULTIPLE TRIALS
7. C WW=AA(1)+AA(2)*(C*(M(I)-1)**B/(A+(M(I)-1)**B))
8. C          +AA(3)*(C*(M(I)-1)**B/(A+(M(I)-1)**B))**2
9. C          +AA(4)*(C*(M(I)-1)**B/(A+(M(I)-1)**B))**3
10. C
11.         COMMON AU,BU,CU,CM(15),CW(15),C(14,3)
12.         COMMON DP,EP,FP,GP,FLAG
13.         COMMON /IOUT/IN,IO,ID,P,MX,EMAX,NP
14.         COMMON /ARAY/M(30),WD(30)
15.         REAL V(11,10),VR(10),FF(11),VE(10),VK(10),XO(10),VC(10)
16.         REAL M,WD,MX,Z(3),M1,M2,M3,PERR,F,AA(4),WK(28),MXX(10),MXM
17.         REAL TOAREA(10),BPAR(4)
18.         INTEGER P,EMAX,FLAG,N
19.         EXTERNAL F
20. C
21.         IN=5
22.         IO=6
23.         NMX=10
24.         NMXP1=NMX+1
25. C
26.         P=10
27.         NX=15
28.         IC=NX-1
29. C
30.         READ *,ID,K,NFEMAX,AL,ALPHA,BETA,GAMMA,EXIT
31.         READ *,NP,SPT,NSERH
32. C
33.         DO 10 I=1,NP
34.         READ *,M(I),WD(I)
35.   10    CONTINUE
36. C
37.         DO 20 I=1,NX
38.            II=I*2
39.            CM(I)=M(II)
40.            CW(I)=WD(II)
41.   20    CONTINUE
42. C
43.         DO 30 I=1,4
44.            BPAR(I)=0.0
45.   30    CONTINUE
46.         CALL ICSICU(CM,CW,NX,BPAR,C,IC,IER)
47. C
48.         III=SPT
49.         DO 144 II=1,NSERH
50. C
51.         ID=0
52.         NFEMAX=500
53. C
54.         M1=M(1)
55.         W1=WD(1)
56.         M2=M(III)
57.         W2=WD(III)
58.         M3=M(30)
59.         W3=WD(30)
60. C
61.         CALL CONB(M1,W1,M2,W2,M3,W3)
62.         FLAG=0
63.         CALL ABCO(M1,W1,M2,W2,M3,W3,AO,BO,CO)
64. C
65.         XO(1)=AO
66.         XO(2)=BO
67.         XO(3)=CO
68. C
69.         FLAG=1
70.         CALL SIMPLX(V,VC,VE,VK,VR,FF,XO,AL,ALPHA,BETA,GAMMA,EXIT,K,NFEMAX
71.        &                ,NMX,NMXP1)
72. C
73.         IF (FF(1).GT.FF(2)) GO TO 40
74.         J=1
75.         SMAL=FF(1)
76.         GO TO 50
77.   40    J=2
78.         SMAL=FF(2)
79.   50    L=K+1
```

```
      DO 60 I=3,L
      IF (SMAL.LT.FF(I)) GO TO 60
      J=I
      SMAL=FF(I)
60    CONTINUE
C
      WRITE(IO,80) M2,W2
80    FORMAT(//////////,15X,'(M2,W2) = ','( ',F10.7,' , ',F10.7,' )')
C
      WRITE(IO,100) V(J,1),V(J,2),V(J,3)
100   FORMAT(/,15X,'FINAL POINTS A,B,C =',3(2X,F10.5))
C
      Z(1)=V(J,1)
      Z(2)=V(J,2)
      Z(3)=V(J,3)
C
      CALL FUNCT(FP,Z,NFEVAL)
C
      CALL ERRT(Z,ERR)
      WRITE(IO,110) ERR
110   FORMAT(/,15X,'TOTAL AREA BETWEEN THE APPROXIMATE CURVE AND THE SP
     &LINE CURVE, FINAL VALUE = ',F10.6)
      WRITE(IO,120) MX
120   FORMAT(/,15X,'FINAL MAXIMUM ERROR FUNCTION VALUE, MX = ',
     &       F10.6)
C
      WRITE(IO,130) M(EMAX)
130   FORMAT(/,15X,'MAXIMUM ERROR FUNCTION VALUE FOUND AT M= ',F10.7)
C
      FAPX=Z(3)*(M(EMAX)-1)**Z(2)/(Z(1)+(M(EMAX)-1)**Z(2))
      ERRP=(FAPX-WD(EMAX))/WD(EMAX)*100
      WRITE(IO,140) ERRP
140   FORMAT(/,15X,'%ERROR AT THIS POINT = ',F10.5,' %')
C
      MXX(II)=MX
      TOAREA(II)=ERR
      IF (II.GT.1) GO TO 142
      AREAM=ERR
      MXM=MX
      AU=Z(1)
      BU=Z(2)
      CU=Z(3)
      GO TO 143
142   AREAM=AMIN1(AREAM,ERR)
      IF (AREAM.NE.ERR) GO TO 143
      MXM=MX
      AU=Z(1)
      BU=Z(2)
      CU=Z(3)
143   III=III+2
144   CONTINUE
C
      WRITE (IO,145)
145   FORMAT('1',//////////)
C
      DO 147 I=1,NSERH
      WRITE (IO,146) M(SPT),MXX(I),TOAREA(I)
146   FORMAT('0',10X,'M2 = ',F10.7,3X,'MX = ',F10.7,3X,'FINAL AREA = ',
     & F10.6)
      SPT=SPT+2
147   CONTINUE
C
      WRITE(IO,148) AU,BU,CU
148   FORMAT('1',/////,20X,'A,B,C =',3(2X,F10.5))
      WRITE(IO,149) MXM,AREAM
149   FORMAT(//,20X,'MX = ',F10.6,5X,'FINAL AREA = ',F10.6)
C
      WRITE(6,150)
150   FORMAT(////,15X,'M',10X,'ORIG W',7X,'APPX W',7X,'ERROR',8X,
     &       '%ERROR',/)
C
      DO 170 I=1,NP
         W=CU*(M(I)-1)**BU/(AU+(M(I)-1)**BU)
         ERR=W-WD(I)
         PERR=ERR/WD(I)*100
         WRITE(6,160) M(I),WD(I),W,ERR,PERR
160      FORMAT(' ',10X,5(F10.6,3X))
170   CONTINUE
C
      N=4
      CALL IFLSQ(F,M,WD,NP,AA,N,WK,IER)
```

```
C
      WRITE(IO,175)
 175  FORMAT('1',/////,10X,'<<<<<<<<<<  MODEL 2  >>>>>>>>>>')
      WRITE(6,180) AA(1),AA(2),AA(3),AA(4)
 180  FORMAT(/////,10X,'A1,A2,A3,A4 = ',4(F10.6,3X))
C
      WRITE(6,190)
 190  FORMAT(////,15X,'M',10X,'ORIG W',7X,'APPX W',7X,'ERROR',8X,
     &       '%ERROR',/)
C
      DO 200 I=1,NP
C
         WW=AA(1)+AA(2)*(CU*(M(I)-1)**BU/(AU+(M(I)-1)**BU))
     &           +AA(3)*(CU*(M(I)-1)**BU/(AU+(M(I)-1)**BU))**2
     &           +AA(4)*(CU*(M(I)-1)**BU/(AU+(M(I)-1)**BU))**3
C
         ERR=WW-WD(I)
         PERC=ERR/WD(I)*100
         WRITE(6,210) M(I),WD(I),WW,ERR,PERC
 210     FORMAT(' ',10X,5(F10.6,3X))
 200  CONTINUE
      STOP
      END
C
C*********************************************************************
C
       REAL FUNCTION F(K,M)
       COMMON AU,BU,CU,CM(15),CW(15),C(14,3)
       INTEGER K
       REAL M
       F=(CU*(M-1)**BU/(AU+(M-1)**BU))**(K-1)
       RETURN
       END
C
C*********************************************************************
C
      SUBROUTINE CONB(M1,W1,M2,W2,M3,W3)
      COMMON AU,BU,CU,CM(15),CW(15),C(14,3)
      COMMON DP,EP,FP,GP,FLAG
      REAL M1,M2,M3
C
      X1=(1-W1/W2)*(W2/W3)+(1-W2/W3)
      C1=(M1-1)*(M2-1)*(M3-1)
      X2=(1-W1/W2)
      C2=(M1-1)*(M2-1)**2
      X3=(1-W2/W3)*(W1/W2)
      C3=(M2-1)**2*(M3-1)
C
      DP=X1/X3
      EP=C1/C2
      FP=C3/C2
      GP=X2/X3
C
      RETURN
      END
C
C*********************************************************************
C
      SUBROUTINE ABCO(M1,W1,M2,W2,M3,W3,AO,BO,CO)
      COMMON AU,BU,CU,CM(15),CW(15),C(14,3)
      COMMON DP,EP,FP,GP,FLAG
      REAL M1,M2,M3
      X=0.0
      U=0.0
      NMAX=200
      EXCR1=0.000001
      EXCR2=0.1E-03
      KEXIT=1
      ID=1
      XL=1.0
      XU=2.0
C
      CALL GOLDEN(XL,XU,X,U,NMAX,EXCR1,EXCR2,KEXIT,ID)
C
      BO=X
      AO=(1-W1/W2)*((M1-1)*(M2-1))**BO/((W1/W2)*(M2-1)**BO-(M1-1)**BO)
      AP=(1-W2/W3)*((M2-1)*(M3-1))**BO/((W2/W3)*(M3-1)**BO-(M2-1)**BO)
      CO=W2*(AO+(M2-1)**BO)/(M2-1)**BO
      CP=W3*(AO+(M3-1)**BO)/(M3-1)**BO
C
```

```
241.          WRITE(6,5) M1,W1,M2,W2,M3,W3
242.    5     FORMAT(/////,15X,'(M1,W1), (M2,W2), (M3,W3) = ',3('(',F6.3,' , '
243.         & ,F6.3,')',2X))
244.          WRITE(6,10) AO,BO,CO
245.    10    FORMAT(/////,15X,'INITIAL A,B,C =',3(2X,F10.5))
246.          WRITE(6,20) AP,BO,CP
247.    20    FORMAT(//,15X,'INITIAL AP,B,CP =',3(2X,F10.5),///)
248.          RETURN
249.          END
250.  C
251.  C*********************************************************************
252.  C
253.          SUBROUTINE FUNCT(F,X,NFEVAL)
254.          COMMON AU,BU,CU,CM(15),CW(15),C(14,3)
255.          COMMON DP,EP,FP,GP,FLAG
256.          COMMON /IOUT/IN,IO,ID,P,MX,EMAX,NP
257.          COMMON /ARAY/M(30),WD(30)
258.          REAL X(1),EU(30),EL(30),M,WD,MX,NEL
259.          INTEGER P,Q,EMAX,FLAG
260.  C
261.          IF (FLAG.EQ.1) GO TO 3
262.          FF=DP*EP**F-FP**F-GP
263.          X(1)=ABS(FF)
264.          RETURN
265.  C
266.    3     EUT=0.0
267.          ELT=0.0
268.  C
269.          AA=X(1)
270.          BB=X(2)
271.          CC=X(3)
272.  C
273.          L=1
274.          DO 10 I=1,NP
275.          FAPX=CC*(M(I)-1)**BB/(AA+(M(I)-1)**BB)
276.          CALL ULSP(I,SU,SL)
277.          EU(I)=FAPX-SU
278.          EL(I)=FAPX-SL
279.          NEL=EL(I)*(-1)
280.          IF (L.GT.1) GO TO 7
281.          MX=AMAX1(EU(I),NEL)
282.          XE=MX
283.          EMAX=I
284.    7     MX=AMAX1(MX,EU(I),NEL)
285.          IF (MX.NE.XE) EMAX=I
286.          XE=MX
287.          L=L+1
288.    10    CONTINUE
289.  C
290.          Q=P*MX/ABS(MX)
291.          IF (MX.GT.0) GO TO 20
292.          IF (MX.LE.0) GO TO 40
293.  C
294.    20    DO 30 J=1,NP
295.          NEL=EL(J)*(-1)
296.          IF (EU(J).GE.0) EUT=EUT+(EU(J)/MX)**Q
297.          IF (NEL.GE.0) ELT=ELT+(NEL/MX)**Q
298.    30    CONTINUE
299.          GO TO 60
300.  C
301.    40    DO 50 J=1,NP
302.          NEL=EL(J)*(-1)
303.          EUT=EUT+(EU(J)/MX)**Q
304.          ELT=ELT+(NEL/MX)**Q
305.    50    CONTINUE
306.  C
307.    60    F=MX*(EUT+ELT)**(1/Q)
308.          NFEVAL=NFEVAL+1
309.          RETURN
310.          END
311.  C
312.  C*********************************************************************
313.  C
314.          SUBROUTINE ULSP(I,SU,SL)
315.          COMMON AU,BU,CU,CM(15),CW(15),C(14,3)
316.          COMMON DP,EP,FP,GP,FLAG
317.          COMMON /IOUT/IN,IO,ID,P,MX,EMAX,NP
318.          COMMON /ARAY/M(30),WD(30)
319.          REAL UPB,LOB,SU,SL,M,WD
320.  C
```

```
      IF (M(I).GE.2.AND.M(I).LT.5) GO TO 10
      IF (M(I).GE.5.AND.M(I).LT.10) GO TO 20
      IF (M(I).GE.10) GO TO 30
C
 10   UPB=14.285710/100
      LOB=-11.11111/100
      GO TO 40
C
 20   UPB=8.1081000/100
      LOB=-6.976740/100
      GO TO 40
C
 30   UPB=5.2631500/100
      LOB=-4.761900/100
C
 40   SU=WD(I)*(1+UPB)
      SL=WD(I)*(1+LOB)
C
      RETURN
      END
C
C ***************************************************************************
C
      SUBROUTINE ERRT(Z,ERR)
      COMMON AU,BU,CU,CM(15),CW(15),C(14,3)
      COMMON DP,EP,FP,GP,FLAG
      COMMON /IOUT/IN,IO,ID,P,MX,EMAX,NP
      COMMON /ARAY/M(30),WD(30)
      REAL MM,M,H,LAST,Z(3)
C
      AA=Z(1)
      BB=Z(2)
      CC=Z(3)
C
      N=100
C
      H=(M(30)-M(1))/N
      SUM=0.0
      MM=M(1)
      LAST=M(30)-H
 10   IF (MM.GT.LAST) GO TO 20
      CALL EXAC(MM,FEX)
      FM=ABS(FEX-CC*(MM-1)**BB/(AA+(MM-1)**BB))
      SUM=SUM+FM
      MM=MM+H
      GO TO 10
 20   ERR=H*SUM
      RETURN
      END
C
C ***************************************************************************
C
      SUBROUTINE EXAC(M,FEX)
      COMMON AU,BU,CU,CM(15),CW(15),C(14,3)
      COMMON DP,EP,FP,GP,FLAG
      REAL CM,CW,C
      REAL M,A
C
      SP(A)=((C(I,3)*(A-CM(I))+C(I,2))*(A-CM(I))+C(I,1))*(A-CM(I))+
     &      CW(I)
C
      IF (M.GE.CM(1).AND.M.LE.CM(2)) I=1
      IF (M.GE.CM(2).AND.M.LE.CM(3)) I=2
      IF (M.GE.CM(3).AND.M.LE.CM(4)) I=3
      IF (M.GE.CM(4).AND.M.LE.CM(5)) I=4
      IF (M.GE.CM(5).AND.M.LE.CM(6)) I=5
      IF (M.GE.CM(6).AND.M.LE.CM(7)) I=6
      IF (M.GE.CM(7).AND.M.LE.CM(8)) I=7
      IF (M.GE.CM(8).AND.M.LE.CM(9)) I=8
      IF (M.GE.CM(9).AND.M.LE.CM(10)) I=9
      IF (M.GE.CM(10).AND.M.LE.CM(11)) I=10
      IF (M.GE.CM(11).AND.M.LE.CM(12)) I=11
      IF (M.GE.CM(12).AND.M.LE.CM(13)) I=12
      IF (M.GE.CM(13).AND.M.LE.CM(14)) I=13
      IF (M.GE.CM(14).AND.M.LE.CM(15)) I=14
C
      FEX=SP(M)
C
      RETURN
```

```
400.          END
401. //GO.SYSIN DD *
402. 0    3    500    1.0    1.0    0.5    2.0    0.1E-03
403. 30   9    6
404.     1.4280360        0.1505281
405.     1.6450540        0.2624752
406.     2.0975210        0.6677244
407.     2.6253080        1.2277890
408.     3.1574670        1.8687510
409.     4.1678040        3.2295040
410.     5.1769400        4.4761320
411.     6.1720640        5.6441460
412.     7.2081140        6.8956560
413.     8.1625420        7.9096680
414.     9.1479870        8.9774800
415.    10.2543500        9.9783860
416.    11.7365300       11.2097000
417.    13.7157600       12.7278600
418.    15.2159400       13.9982800
419.    17.0658200       15.0738400
420.    18.7354200       16.0689500
421.    20.3599200       17.1233600
422.    22.8266100       18.6382400
423.    25.0691900       19.6563200
424.    27.5268800       21.1685100
425.    30.2312400       22.3248100
426.    32.8556500       23.5420300
427.    34.9748900       24.3067700
428.    37.6225500       25.0987500
429.    40.0493000       25.9140400
430.    42.6325300       26.7558400
431.    44.4466500       27.3321800
432.    47.3180500       27.9262300
433.    50.3701600       28.8334000
```

Appendix I

TIME DIAL SETTING DETERMINATION - PROGRAM LISTING

```
1. //TNG JOB '1222335,,T=05'
2. //  EXEC FORTXCLG,OPT=2,USERLIB='MENZIES.MAIN.LOAD'
3. //FORT.SYSIN DD *
4. C
5. C FILENAME : DSETTNG
6. C APPLICATION OF DYNAMIC CURVE TO RELAY SETTING DETERMINATION.
7. C
8.         REAL M
9.         READ *,TYPE,TOP,M
10. C
11.        WRITE(6,10)
12.  10    FORMAT('1',//,5X,'INPUT INFORMATION :')
13. C
14.        IF (TYPE.EQ.8) GO TO 20
15.        IF (TYPE.EQ.11) GO TO 30
16.        IF (TYPE.EQ.51) GO TO 40
17.        IF (TYPE.EQ.77) GO TO 50
18. C
19.  20    WRITE(6,21)
20.  21    FORMAT(//,15X,'RELAY TYPE : CO-8 ')
21.        DMAX=11.0
22.        GO TO 60
23.  30    WRITE(6,31)
24.  31    FORMAT(//,15X,'RELAY TYPE : CO-11 ')
25.        DMAX=11.0
26.        GO TO 60
27.  40    WRITE(6,41)
28.  41    FORMAT(//,15X,'RELAY TYPE : IAC51 ')
29.        DMAX=10.0
30.        GO TO 60
31.  50    WRITE(6,51)
32.  51    FORMAT(//,15X,'RELAY TYPE : IAC77 ')
33.        DMAX=10.0
34.        GO TO 60.
35. C
36.  60    WRITE(6,80)
37.  80    FORMAT(//,15X,'FIND THE TIME DIAL SETTING SUCH THAT THE RELAY ')
38.        WRITE(6,85) TOP,M
39.  85    FORMAT(15X,'WILL TRIP AT ',F6.3,' SECONDS WHEN M = ',F6.3)
40.        WRITE(6,87)
41.  87    FORMAT(/////,5X,'OUTPUT INFORMATIOM :')
42. C
43.        DA=0.5
44.        CALL INABC(TYPE,DA,PHATA,FLAG,T,A,B,C,A1,A2,A3,A4)
45.        CALL OPTIME(DA,M,T,A,B,C,A1,A2,A3,A4)
46.        TA=T
47.        IF (TOP.GT.TA) GO TO 100
48.        WRITE(6,90)
49.  90    FORMAT(//,15X,'!!! THE SPECIFIED RELAY IS NOT APPLICABLE !!!')
50.        GO TO 160
51.  100   DB=1.0
52.  110   CALL INABC(TYPE,DB,PHATA,FLAG,T,A,B,C,A1,A2,A3,A4)
53.        CALL OPTIME(DB,M,T,A,B,C,A1,A2,A3,A4)
54.        TB=T
55.        IF (TOP.LT.TB) GO TO 130
56.        TA=TB
57.        DA=DB
58.        DB=DB+1
59.        IF (DB.LE.DMAX) GO TO 110
60.        WRITE(6,120)
61.  120   FORMAT(//,15X,'!!! THE SPECIFIED RELAY IS NOT APPLICABLE !!!')
62.        GO TO 160
63.  130   DP=DA+(DB-DA)*(TOP-TA)/(TB-TA)
64.        WRITE(6,150) DP
65.  150   FORMAT(//,15X,'THE REQUIRED TDS IS = ',F10.7)
66.  160   STOP
67.        END
68. C
69. C ***************************************************************************
70. C
71.        SUBROUTINE OPTIME(D,M,T,A,B.C,A1,A2,A3,A4)
72.        REAL M
73.        F=C*(M-1)**B/(A+(M-1)**B)
74.        W=A1+A2*F+A3*F**2+A4*F**3
75.        T=D/W
76.        RETURN
77.        END
78. C
79. C ***************************************************************************
80. C
```

```
      SUBROUTINE INABC(TYPE,TDS,PHATA,FLAG,T,A,B,C,A1,A2,A3,A4)
      INTEGER R,FLAG
      REAL CO8(12,7),CO11(12,7)
      REAL IAC51(11,7),IAC77(11,7)
C
C TYPE : CO-8 ; TDS=0.5
C
      CO8(1,1)=13.51051
      CO8(1,2)= 1.37005
      CO8(1,3)= 9.23443
      CO8(1,4)=-0.006208
      CO8(1,5)= 0.908987
      CO8(1,6)= 0.040877
      CO8(1,7)=-0.003409
C
C TYPE : CO-8 ; TDS=1
C
      CO8(2,1)=16.88931
      CO8(2,2)= 1.35743
      CO8(2,3)= 8.43328
      CO8(2,4)=-0.120720
      CO8(2,5)= 1.294505
      CO8(2,6)=-0.098391
      CO8(2,7)= 0.008222
C
C TYPE : CO-8 ; TDS=2
C
      CO8(3,1)=15.47383
      CO8(3,2)= 1.41182
      CO8(3,3)= 7.41377
      CO8(3,4)=-0.154400
      CO8(3,5)= 1.405449
      CO8(3,6)=-0.153965
      CO8(3,7)= 0.014629
C
C TYPE : CO-8 ; TDS=3
C
      CO8(4,1)=17.90462
      CO8(4,2)= 1.46749
      CO8(4,3)= 7.33642
      CO8(4,4)=-0.143101
      CO8(4,5)= 1.540100
      CO8(4,6)=-0.228115
      CO8(4,7)= 0.022869
C
C TYPE : CO8 ; TDS=4
C
      CO8(5,1)=17.24532
      CO8(5,2)= 1.37962
      CO8(5,3)= 7.60017
      CO8(5,4)=-0.183747
      CO8(5,5)= 1.524651
      CO8(5,6)=-0.204056
      CO8(5,7)= 0.019465
C
C TYPE : CO8 ; TDS=5
C
      CO8(6,1)=17.29149
      CO8(6,2)= 1.39777
      CO8(6,3)= 7.52718
      CO8(6,4)=-0.167412
      CO8(6,5)= 1.441585
      CO8(6,6)=-0.158246
      CO8(6,7)= 0.014221
C
C TYPE : CO8 ; TDS=6
C
      CO8(7,1)=18.15140
      CO8(7,2)= 1.40001
      CO8(7,3)= 7.48478
      CO8(7,4)=-0.153112
      CO8(7,5)= 1.441457
      CO8(7,6)=-0.165675
      CO8(7,7)= 0.015338
C
C TYPE : CO8 ; TDS=7
C
      CO8(8,1)=17.68565
      CO8(8,2)= 1.39580
      CO8(8,3)= 7.28394
      CO8(8,4)=-0.177021
      CO8(8,5)= 1.517089
      CO8(8,6)=-0.196429
      CO8(8,7)= 0.018656
C
C TYPE : CO8 ; TDS=8
C
```

```
      CO8(9,1)=17.89734
      CO8(9,2)= 1.39379
      CO8(9,3)= 7.33908
      CO8(9,4)=-0.173041
      CO8(9,5)= 1.502666
      CO8(9,6)=-0.203328
      CO8(9,7)= 0.020186
C
C TYPE : CO8 ; TDS=9
C
      CO8(10,1)=18.04555
      CO8(10,2)= 1.37495
      CO8(10,3)= 7.36964
      CO8(10,4)=-0.198192
      CO8(10,5)= 1.557767
      CO8(10,6)=-0.217780
      CO8(10,7)= 0.021126
C
C TYPE : CO8 ; TDS=10
C
      CO8(11,1)=18.16992
      CO8(11,2)= 1.42629
      CO8(11,3)= 6.97664
      CO8(11,4)=-0.182102
      CO8(11,5)= 1.544777
      CO8(11,6)=-0.220914
      CO8(11,7)= 0.022125
C
C TYPE : CO8 ; TDS=11
C
      CO8(12,1)=18.09709
      CO8(12,2)= 1.37901
      CO8(12,3)= 6.94451
      CO8(12,4)=-0.236098
      CO8(12,5)= 1.625359
      CO8(12,6)=-0.250449
      CO8(12,7)= 0.025212
C
      R=0
      R=TDS+1
      IF (TYPE.EQ. 8) GO TO 20
      IF (TYPE.EQ.11) GO TO 30
      IF (TYPE.EQ.51) GO TO 40
      IF (TYPE.EQ.77) GO TO 50
      WRITE(6,10)
 10   FORMAT(/////,5X,'!!!!! RELAY TYPE NOT AVAILABLE !!!!!')
      STOP
C
 20   A=CO8(R,1)
      B=CO8(R,2)
      C=CO8(R,3)
      A1=CO8(R,4)
      A2=CO8(R,5)
      A3=CO8(R,6)
      A4=CO8(R,7)
      GO TO 70
 30   A=CO11(R,1)
      B=CO11(R,2)
      C=CO11(R,3)
      A1=CO11(R,4)
      A2=CO11(R,5)
      A3=CO11(R,6)
      A4=CO11(R,7)
      GO TO 70
 40   A=IAC51(R,1)
      B=IAC51(R,2)
      C=IAC51(R,3)
      A1=IAC51(R,4)
      A2=IAC51(R,5)
      A3=IAC51(R,6)
      A4=IAC51(R,7)
      GO TO 70
 50   A=IAC77(R,1)
      B=IAC77(R,2)
      C=IAC77(R,3)
      A1=IAC77(R,4)
      A2=IAC77(R,5)
      A3=IAC77(R,6)
      A4=IAC77(R,7)
      GO TO 70
C
 70   PHATA=0.0
      FLAG=0
      T=0.0
      RETURN
      END
//GO.SYSIN DD *
8     0.4     20.0
```

Appendix J

OPERATE TIME DETERMINATION - PROGRAM LISTING

```
1.  //TNG JOB '1222335,,T=05'
2.  //  EXEC FORTXCLG,OPT=2,USERLIB='MENZIES.MAIN.LOAD'
3.  //FORT.SYSIN DD *
4.  C
5.  C FILENAME : OPERTME
6.  C APPLICATION OF DYNAMIC CURVES TO RELAY OPERATE TIME CALCULATION
7.  C
8.          INTEGER DAI
9.          REAL M
10.         READ *,TYPE,D,M
11. C
12.         WRITE(6,10)
13.  10     FORMAT('1',//,5X,'INPUT INFORMATION :')
14. C
15.         IF (TYPE.EQ.8) GO TO 20
16.         IF (TYPE.EQ.11) GO TO 30
17.         IF (TYPE.EQ.51) GO TO 40
18.         IF (TYPE.EQ.77) GO TO 50
19. C
20.  20     WRITE(6,21) D
21.  21     FORMAT(//,15X,'RELAY TYPE : CO-8 ',2X,';    TDS = ',F4.1)
22.         DMAX=11.0
23.         GO TO 60
24.  30     WRITE(6,31) D
25.  31     FORMAT(//,15X,'RELAY TYPE : CO-11 ',2X,';    TDS = ',F4.1)
26.         DMAX=11.0
27.         GO TO 60
28.  40     WRITE(6,41) D
29.  41     FORMAT(//,15X,'RELAY TYPE : IAC51 ',2X,';    TDS = ',F4.1)
30.         DMAX=10.0
31.         GO TO 60
32.  50     WRITE(6,51) D
33.  51     FORMAT(//,15X,'RELAY TYPE : IAC77 ',2X,';    TDS = ',F4.1)
34.         DMAX=10.0
35.         GO TO 60
36. C
37.  60     WRITE(6,80) M,D
38.  80     FORMAT(//,15X,'FIND THE OPERATE TIME AT M = ',F6.3,' AND D = ',
39.      & F6.3)
40. C
41.         WRITE(6,83)
42.  83     FORMAT(/////,5X,'OUTPUT INFORMATION :')
43. C
44.         IF (D.LT.0.5) GO TO 84
45.         IF (D.LE.DMAX) GO TO 87
46.  84     WRITE(6,85)
47.  85     FORMAT(//,15X,'THE SPECIFIED TIME DIAL SETTING IS NOT AVAILABLE.'
48.      & )
49.         GO TO 130
50.  87     IF (D.LT.1.0.AND.D.GE.0.5) GO TO 90
51. C
52.         DAI=D
53.         DA=DAI
54.         DB=DA+1
55.         GO TO 100
56. C
57.  90     DA=0.5
58.         DB=1
59. C
60.  100    CALL INABC(TYPE,DA,PHATA,FLAG,T,A,B,C,A1,A2,A3,A4)
61.         CALL OPTIME(DA,M,T,A,B,C,A1,A2,A3,A4)
62.         TA=T
63. C
64.         CALL INABC(TYPE,DB,PHATA,FLAG,T,A,B,C,A1,A2,A3,A4)
65.         CALL OPTIME(DB,M,T,A,B,C,A1,A2,A3,A4)
66.         TB=T
67. C
68.         TOP=TA+(TB-TA)*(D-DA)/(DB-DA)
69. C
70.         WRITE(6,120) TOP
71.  120    FORMAT(//,15X,'THE OPERATE TIME = ',F10.6)
72.  130    STOP
73.         END
74. C
75. C ***************************************************************************
76. C
77.         SUBROUTINE OPTIME(D,M,T,A,B,C,A1,A2,A3,A4)
78.         REAL M
79.         F=C*(M-1)**B/(A+(M-1)**B)
80.         W=A1+A2*F+A3*F**2+A4*F**3
81.         T=D/W
82.         RETURN
83.         END
```

```
C
C ***************************************************************************
C
      SUBROUTINE INABC(TYPE,TDS,PHATA,FLAG,T,A,B,C,A1,A2,A3,A4)
      INTEGER R,FLAG
      REAL CO8(12,7),CO11(12,7)
      REAL IAC51(11,7),IAC77(11,7)
C
C TYPE : CO-8 ; TDS=0.5
C
      CO8(1,1)=13.51051
      CO8(1,2)= 1.37005
      CO8(1,3)= 9.23443
      CO8(1,4)=-0.006208
      CO8(1,5)= 0.908987
      CO8(1,6)= 0.040877
      CO8(1,7)=-0.003409
C
C TYPE : CO-8 ; TDS=1
C
      CO8(2,1)=16.88931
      CO8(2,2)= 1.35743
      CO8(2,3)= 8.43328
      CO8(2,4)=-0.120720
      CO8(2,5)= 1.294505
      CO8(2,6)=-0.098391
      CO8(2,7)= 0.008222
C
C TYPE : CO-8 ; TDS=2
C
      CO8(3,1)=15.47383
      CO8(3,2)= 1.41182
      CO8(3,3)= 7.41377
      CO8(3,4)=-0.154400
      CO8(3,5)= 1.405449
      CO8(3,6)=-0.153965
      CO8(3,7)= 0.014629
C
C TYPE : CO-8 ; TDS=3
C
      CO8(4,1)=17.90462
      CO8(4,2)= 1.46749
      CO8(4,3)= 7.33642
      CO8(4,4)=-0.143101
      CO8(4,5)= 1.540100
      CO8(4,6)=-0.228115
      CO8(4,7)= 0.022869
C
C TYPE : CO8 ; TDS=4
C
      CO8(5,1)=17.24532
      CO8(5,2)= 1.37962
      CO8(5,3)= 7.60017
      CO8(5,4)=-0.183747
      CO8(5,5)= 1.524651
      CO8(5,6)=-0.204056
      CO8(5,7)= 0.019465
C
C TYPE : CO8 ; TDS=5
C
      CO8(6,1)=17.29149
      CO8(6,2)= 1.39777
      CO8(6,3)= 7.52718
      CO8(6,4)=-0.167412
      CO8(6,5)= 1.441585
      CO8(6,6)=-0.158246
      CO8(6,7)= 0.014221
C
C TYPE : CO8 ; TDS=6
C
      CO8(7,1)=18.15140
      CO8(7,2)= 1.40001
      CO8(7,3)= 7.48478
      CO8(7,4)=-0.153112
      CO8(7,5)= 1.441457
      CO8(7,6)=-0.165675
      CO8(7,7)= 0.015338
C
C TYPE : CO8 ; TDS=7
C
      CO8(8,1)=17.68565
      CO8(8,2)= 1.39580
      CO8(8,3)= 7.28394
```

```
      CO8(8,4)=-0.177021
      CO8(8,5)= 1.517089
      CO8(8,6)=-0.196429
      CO8(8,7)= 0.018656
C
C TYPE : CO8 ; TDS=8
C
      CO8(9,1)=17.89734
      CO8(9,2)= 1.39379
      CO8(9,3)= 7.33908
      CO8(9,4)=-0.173041
      CO8(9,5)= 1.502666
      CO8(9,6)=-0.203328
      CO8(9,7)= 0.020186
C
C TYPE : CO8 ; TDS=9
C
      CO8(10,1)=18.04555
      CO8(10,2)= 1.37495
      CO8(10,3)= 7.36964
      CO8(10,4)=-0.198192
      CO8(10,5)= 1.557767
      CO8(10,6)=-0.217780
      CO8(10,7)= 0.021126
C
C TYPE : CO8 ; TDS=10
C
      CO8(11,1)=18.16992
      CO8(11,2)= 1.42629
      CO8(11,3)= 6.97664
      CO8(11,4)=-0.182102
      CO8(11,5)= 1.544777
      CO8(11,6)=-0.220914
      CO8(11,7)= 0.022125
C
C TYPE : CO8 ; TDS=11
C
      CO8(12,1)=18.09709
      CO8(12,2)= 1.37901
      CO8(12,3)= 6.94451
      CO8(12,4)=-0.236098
      CO8(12,5)= 1.625359
      CO8(12,6)=-0.250449
      CO8(12,7)= 0.025212
C
      R=0
      R=TDS+1
      IF (TYPE.EQ. 8) GO TO 20
      IF (TYPE.EQ.11) GO TO 30
      IF (TYPE.EQ.51) GO TO 40
      IF (TYPE.EQ.77) GO TO 50
      WRITE(6,10)
 10   FORMAT(/////,5X,'!!!!! RELAY TYPE NOT AVAILABLE !!!!!')
      STOP
C
 20   A=CO8(R,1)
      B=CO8(R,2)
      C=CO8(R,3)
      A1=CO8(R,4)
      A2=CO8(R,5)
      A3=CO8(R,6)
      A4=CO8(R,7)
      GO TO 70
 30   A=CO11(R,1)
      B=CO11(R,2)
      C=CO11(R,3)
      A1=CO11(R,4)
      A2=CO11(R,5)
      A3=CO11(R,6)
      A4=CO11(R,7)
      GO TO 70
 40   A=IAC51(R,1)
      B=IAC51(R,2)
      C=IAC51(R,3)
      A1=IAC51(R,4)
      A2=IAC51(R,5)
      A3=IAC51(R,6)
      A4=IAC51(R,7)
      GO TO 70
```

```
246.   50     A=IAC77(R,1)
247.          B=IAC77(R,2)
248.          C=IAC77(R,3)
249.          A1=IAC77(R,4)
250.          A2=IAC77(R,5)
251.          A3=IAC77(R,6)
252.          A4=IAC77(R,7)
253.          GO TO 70
254. C
255.   70     PHATA=0.0
256.          FLAG=0
257.          T=0.0
258.          RETURN
259.          END
260. //GO.SYSIN DD *
261. 8      2.322      20
```

Appendix K

RELAY SIMULATION TEST CASE ONE - PROGRAM LISTING

```
1. //TNG JOB '1222335,,T=05'
2. //  EXEC FORTXCLG,OPT=2,USERLIB='MENZIES.MAIN.LOAD'
3. //FORT.SYSIN DD *
4. C
5. C FILENAME :'TEST1
6. C TRIP SIMULATION WITH A CONSTANT FAULT CURRENT. C.B. TIME IS ASSUMED
7. C TO BE 6 CYCLES (=0.1 SECONDS).
8. C
9.         COMMON DELTAT,DELPHA
10.        INTEGER TYPE,FLAG
11.        REAL M,MF,A,B,C,A1,A2,A3,A4,PHATA,DELTAT,DELPHA
12. C
13.        DELTAT=0.0001
14.        DELPHA=0.001
15. C
16.        READ *,TYPE,TDS,MF,TE
17.        CALL INABC(TYPE,TDS,PHATA,FLAG,T,A,B,C,A1,A2,A3,A4)
18. C
19.        WRITE(6,10)
20.   10   FORMAT('1',//,5X,'INPUT INFORMATION :')
21. C
22.        IF (TYPE.EQ.8) GO TO 20
23.        IF (TYPE.EQ.11) GO TO 30
24.        IF (TYPE.EQ.51) GO TO 40
25.        IF (TYPE.EQ.77) GO TO 50
26. C
27.   20   WRITE(6,21) TDS
28.   21   FORMAT(//,5X,'RELAY TYPE : CO-8 ',2X,';   TDS = ',F4.1)
29.        GO TO 60
30.   30   WRITE(6,31) TDS
31.   31   FORMAT(//,5X,'RELAY TYPE : CO-11 ',2X,';   TDS = ',F4.1)
32.        GO TO 60
33.   40   WRITE(6,41) TDS
34.   41   FORMAT(//,5X,'RELAY TYPE : IAC51 ',2X,';   TDS = ',F4.1)
35.        GO TO 60
36.   50   WRITE(6,51) TDS
37.   51   FORMAT(//,5X,'RELAY TYPE : IAC77 ',2X,';   TDS = ',F4.1)
38.        GO TO 60
39. C
40.   60   WRITE(6,61) A,B,C
41.   61   FORMAT(/,5X,'A , B , C  =',3(2X,F10.7))
42.        WRITE(6,70) A1,A2,A3,A4
43.   70   FORMAT(/,5X,'A1 , A2 , A3 , A4  =',4(2X,F10.7))
44.        WRITE(6,80) MF
45.   80   FORMAT(/,5X,'FAULT CURRENT, MF = ',F10.7,'  AT T = 0.0 SECONDS.
46.       & ')
47. C
48.        M=MF
49.   90   CALL OCABC(TDS,M,T,PHATA,FLAG,A,B,C,A1,A2,A3,A4)
50. C
51.        IF (FLAG.EQ.0) GO TO 90
52.        TOP=T
53.        WRITE(6,95)
54.   95   FORMAT(///,5X,'OUTPUT INFORMATION :')
55.        WRITE(6,100) TOP
56.   100  FORMAT(//,5X,'RELAY TRIP TIME, TOP = ',F10.7,'  SECONDS.')
57.        TC=TOP+0.1
58. C
59.   110  CALL OCABC(TDS,M,T,PHATA,FLAG,A,B,C,A1,A2,A3,A4)
60. C
61.        IF (T.LT.TC) GO TO 110
62.        WRITE(6,120) T
63.   120  FORMAT(/,5X,'FAULT CLEARANCE TIME, TC = ',F10.7,'  SECONDS.')
64. C
65.        M=0.0
66.   130  CALL OCABC(TDS,M,T,PHATA,FLAG,A,B,C,A1,A2,A3,A4)
67. C
68.        IF (PHATA.NE.0.0) GO TO 130
69.        TR=T-TC
70.        WRITE(6,140) TR
71.   140  FORMAT(/,5X,'RESET TIME, TR = ',F10.6,' SECONDS.')
72.        ER=(TOP-TE)/TE*100
73.        WRITE(6,150) ER
74.   150  FORMAT(/,5X,'%ERROR OF RELAY TRIP TIME =',2X,F10.6,2X,'%')
75.        STOP
76.        END
77. C
78. C*****************************************************************************
79. C
80.        SUBROUTINE OCABC(D,M,T,PHATA,FLAG,A,B,C,A1,A2,A3,A4)
81.        COMMON DELTAT,DELPHA
82.        INTEGER FLAG
83.        REAL M
84. C
85.        IF (M.LE.1.0) GO TO 20
86.        IF (FLAG.EQ.0) GO TO 10
87.        T=T+DELTAT
```

```
 88.          GO TO 30
 89.  C
 90.    10    F=C*(M-1)**B/(A+(M-1)**B)
 91.          W = A1 + A2*F + A3*F**2 + A4*F**3
 92.          PHATA=PHATA+W*DELTAT
 93.          T=T+DELTAT
 94.  C
 95.          IF (PHATA.LT.D) GO TO 30
 96.          PHATA=D
 97.          FLAG=1
 98.          GO TO 30
 99.  C
100.    20    FLAG=0
101.          IF (PHATA.EQ.0.0) GO TO 30
102.          PHATA=PHATA-DELPHA
103.          T=T+(70.0/10.0)*DELPHA
104.          IF (PHATA.GT.0.0) GO TO 30
105.          PHATA=0.0
106.    30    RETURN
107.          END
108.  C
109.  C*************************************************************************
110.  C
111.          SUBROUTINE INABC(TYPE,TDS,PHATA,FLAG,T,A,B,C,A1,A2,A3,A4)
112.          INTEGER R,TYPE,FLAG
113.          REAL CO8(12,7),CO11(12,7)
114.          REAL IAC51(11,7),IAC77(11,7)
115.  C
116.  C TYPE : CO-8 ; TDS=6
117.  C
118.          CO8(7,1)=18.46631
119.          CO8(7,2)= 1.39766
120.          CO8(7,3)= 7.55713
121.          CO8(7,4)=-0.153045
122.          CO8(7,5)= 1.448598
123.          CO8(7,6)=-0.165898
124.          CO8(7,7)= 0.015019
125.  C
126.          R=0
127.          R=TDS+1
128.          IF (TYPE.EQ. 8) GO TO 20
129.          IF (TYPE.EQ.11) GO TO 30
130.          IF (TYPE.EQ.51) GO TO 40
131.          IF (TYPE.EQ.77) GO TO 50
132.          WRITE(6,10)
133.    10    FORMAT(/////,5X,'!!!!! RELAY TYPE NOT AVAILABLE !!!!!')
134.          STOP
135.  C
136.    20    A=CO8(R,1)
137.          B=CO8(R,2)
138.          C=CO8(R,3)
139.          A1=CO8(R,4)
140.          A2=CO8(R,5)
141.          A3=CO8(R,6)
142.          A4=CO8(R,7)
143.          GO TO 70
144.    30    A=CO11(R,1)
145.          B=CO11(R,2)
146.          C=CO11(R,3)
147.          A1=CO11(R,4)
148.          A2=CO11(R,5)
149.          A3=CO11(R,6)
150.          A4=CO11(R,7)
151.          GO TO 70
152.    40    A=IAC51(R,1)
153.          B=IAC51(R,2)
154.          C=IAC51(R,3)
155.          A1=IAC51(R,4)
156.          A2=IAC51(R,5)
157.          A3=IAC51(R,6)
158.          A4=IAC51(R,7)
159.          GO TO 70
160.    50    A=IAC77(R,1)
161.          B=IAC77(R,2)
162.          C=IAC77(R,3)
163.          A1=IAC77(R,4)
164.          A2=IAC77(R,5)
165.          A3=IAC77(R,6)
166.          A4=IAC77(R,7)
167.          GO TO 70
168.  C
169.    70    PHATA=0.0
170.          FLAG=0
171.          T=0.0
172.          RETURN
173.          END
174.  //GO.SYSIN DD *
175.     8    6     1.0498402E+01   1.4087193E+00
```

Appendix L

RELAY SIMULATION TEST CASE TWO - PROGRAM LISTING

```
1. //TNG JOB '1222335,,T=05'
2. //  EXEC FORTXCLG,OPT=2,USERLIB='MENZIES.MAIN.LOAD'
3. //FORT.SYSIN DD *
4. C
5. C FILENAME :  TEST2
6. C TRIP SIMULATION WITH A SUDDEN INCREASE OF FAULT CURRENT.
7. C
8.         COMMON DELTAT,DELPHA
9.         INTEGER TYPE, FLAG
10.        REAL M,M1,M2,A,B,C,A1,A2,A3,A4,PHATA,DELTAT,DELPHA
11. C
12.        DELTAT=0.0001
13.        DELPHA=0.001
14. C
15.        READ *,TYPE,TDS,M1,T1,M2
16.        CALL INABC(TYPE,TDS,PHATA,FLAG,T,A,B,C,A1,A2,A3,A4)
17. C
18.        WRITE(6,10)
19.  10    FORMAT('1',//,5X,'INPUT INFORMATION :')
20. C
21.        IF (TYPE.EQ.8) GO TO 20
22.        IF (TYPE.EQ.11) GO TO 30
23.        IF (TYPE.EQ.51) GO TO 40
24.        IF (TYPE.EQ.77) GO TO 50
25. C
26.  20    WRITE(6,21) TDS
27.  21    FORMAT(//,5X,'RELAY TYPE : CO-8 ',2X,';   TDS = ',F4.1)
28.        GO TO 60
29.  30    WRITE(6,31) TDS
30.  31    FORMAT(//,5X,'RELAY TYPE : CO-11 ',2X,';   TDS = ',F4.1)
31.        GO TO 60
32.  40    WRITE(6,41) TDS
33.  41    FORMAT(//,5X,'RELAY TYPE : IAC51 ',2X,';   TDS = ',F4.1)
34.        GO TO 60
35.  50    WRITE(6,51) TDS
36.  51    FORMAT(//,5X,'RELAY TYPE : IAC77 ',2X,';   TDS = ',F4.1)
37.        GO TO 60
38. C
39.  60    WRITE(6,61) A,B,C
40.  61    FORMAT(/,5X,'A , B , C  =',3(2X,F10.7))
41.        WRITE(6,70) A1,A2,A3,A4
42.  70    FORMAT(/,5X,'A1 , A2 , A3 , A4  =',4(2X,F10.7))
43.        WRITE(6,80) M1
44.  80    FORMAT(/,5X,'INITIAL FAULT CURRENT, M1 = ',F10.7,'  AT T = 0.0
45.       & SECONDS.')
46.        WRITE(6,90) M2,T1
47.  90    FORMAT(/,5X,'INCREASED FAULT CURRENT, M2 = ',F10.7,'  AT T = ',
48.       &        F10.7,' SECONDS.')
49. C
50.        M=M1
51.  100   IF (T.GE.T1) GO TO 110
52.        CALL OCABC(TDS,M,T,PHATA,FLAG,A,B,C,A1,A2,A3,A4)
53.        IF (FLAG.EQ.0) GO TO 100
54.        WRITE(6,105)
55.  105   FORMAT(//////////,5X,'!!!! TEST FAILED !!!!')
56.        STOP
57.  110   M=M2
58.        WRITE(6,112)
59.  112   FORMAT(///,5X,'OUTPUT INFORMATION :')
60.        WRITE(6,115) PHATA,T1
61.  115   FORMAT(//,5X,'DEGREE OF DISC TRAVEL, PHATA = ',F10.7,' AT T = ',
62.       & F6.3,' SECONDS.')
63.  120   CALL OCABC(TDS,M,T,PHATA,FLAG,A,B,C,A1,A2,A3,A4)
64.        IF (FLAG.EQ.0) GO TO 120
65.  130   WRITE(6,131) T
66.  131   FORMAT(/,5X,'TRIP COMMAND EMITTED AT ',F10.6,
67.       & ' SECONDS AFTER FAULT INCURSION.')
68.        WRITE(6,140) PHATA
69.  140   FORMAT(/,5X,'FINAL DEGREE OF DISC TRAVEL, PHATA = ',F10.7)
70.        STOP
71.        END
72. C
73. C*********************************************************************
74. C
75.        SUBROUTINE OCABC(D,M,T,PHATA,FLAG,A,B,C,A1,A2,A3,A4)
76.        COMMON DELTAT,DELPHA
77.        INTEGER FLAG
78.        REAL M
79. C
80.        IF (M.LE.1.0) GO TO 20
81.        IF (FLAG.EQ.0) GO TO 10
82.        T=T+DELTAT
83.        GO TO 30
84. C
```

```
 10     F=C*(M-1)**B/(A+(M-1)**B)
        W = A1 + A2*F + A3*F**2 + A4*F**3
        PHATA=PHATA+W*DELTAT
        T=T+DELTAT
C
        IF (PHATA.LT.D) GO TO 30
        PHATA=D
        FLAG=1
        GO TO 30
C
 20     FLAG=0
        IF (PHATA.EQ.0.0) GO TO 30
        PHATA=PHATA-DELPHA
        T=T+(70.0/10.0)*DELPHA
        IF (PHATA.GT.0.0) GO TO 30
        PHATA=0.0
 30     RETURN
        END
C
C*************************************************************************
C
        SUBROUTINE INABC(TYPE,TDS,PHATA,FLAG,T,A,B,C,A1,A2,A3,A4)
        INTEGER R,TYPE,FLAG
        REAL CO8(12,7),CO11(12,7)
        REAL IAC51(11,7),IAC77(11,7)
C
C TYPE : CO-8 ; TDS=6
C
        CO8(7,1)=18.46631
        CO8(7,2)= 1.39766
        CO8(7,3)= 7.55713
        CO8(7,4)=-0.153045
        CO8(7,5)= 1.448598
        CO8(7,6)=-0.165898
        CO8(7,7)= 0.015019
C
        R=0
        R=TDS+1
        IF (TYPE.EQ. 8) GO TO 20
        IF (TYPE.EQ.11) GO TO 30
        IF (TYPE.EQ.51) GO TO 40
        IF (TYPE.EQ.77) GO TO 50
        WRITE(6,10)
 10     FORMAT(/////,5X,'!!!!! RELAY TYPE NOT AVAILABLE !!!!!')
        STOP
C
 20     A=CO8(R,1)
        B=CO8(R,2)
        C=CO8(R,3)
        A1=CO8(R,4)
        A2=CO8(R,5)
        A3=CO8(R,6)
        A4=CO8(R,7)
        GO TO 70
 30     A=CO11(R,1)
        B=CO11(R,2)
        C=CO11(R,3)
        A1=CO11(R,4)
        A2=CO11(R,5)
        A3=CO11(R,6)
        A4=CO11(R,7)
        GO TO 70
 40     A=IAC51(R,1)
        B=IAC51(R,2)
        C=IAC51(R,3)
        A1=IAC51(R,4)
        A2=IAC51(R,5)
        A3=IAC51(R,6)
        A4=IAC51(R,7)
        GO TO 70
 50     A=IAC77(R,1)
        B=IAC77(R,2)
        C=IAC77(R,3)
        A1=IAC77(R,4)
        A2=IAC77(R,5)
        A3=IAC77(R,6)
        A4=IAC77(R,7)
        GO TO 70
C
 70     PHATA=0.0
        FLAG=0
        T=0.0
        RETURN
        END
//GO.SYSIN DD *
8    6    5.0    0.5    9.0
```

Appendix M

RELAY SIMULATION TEST CASE THREE - PROGRAM LISTING

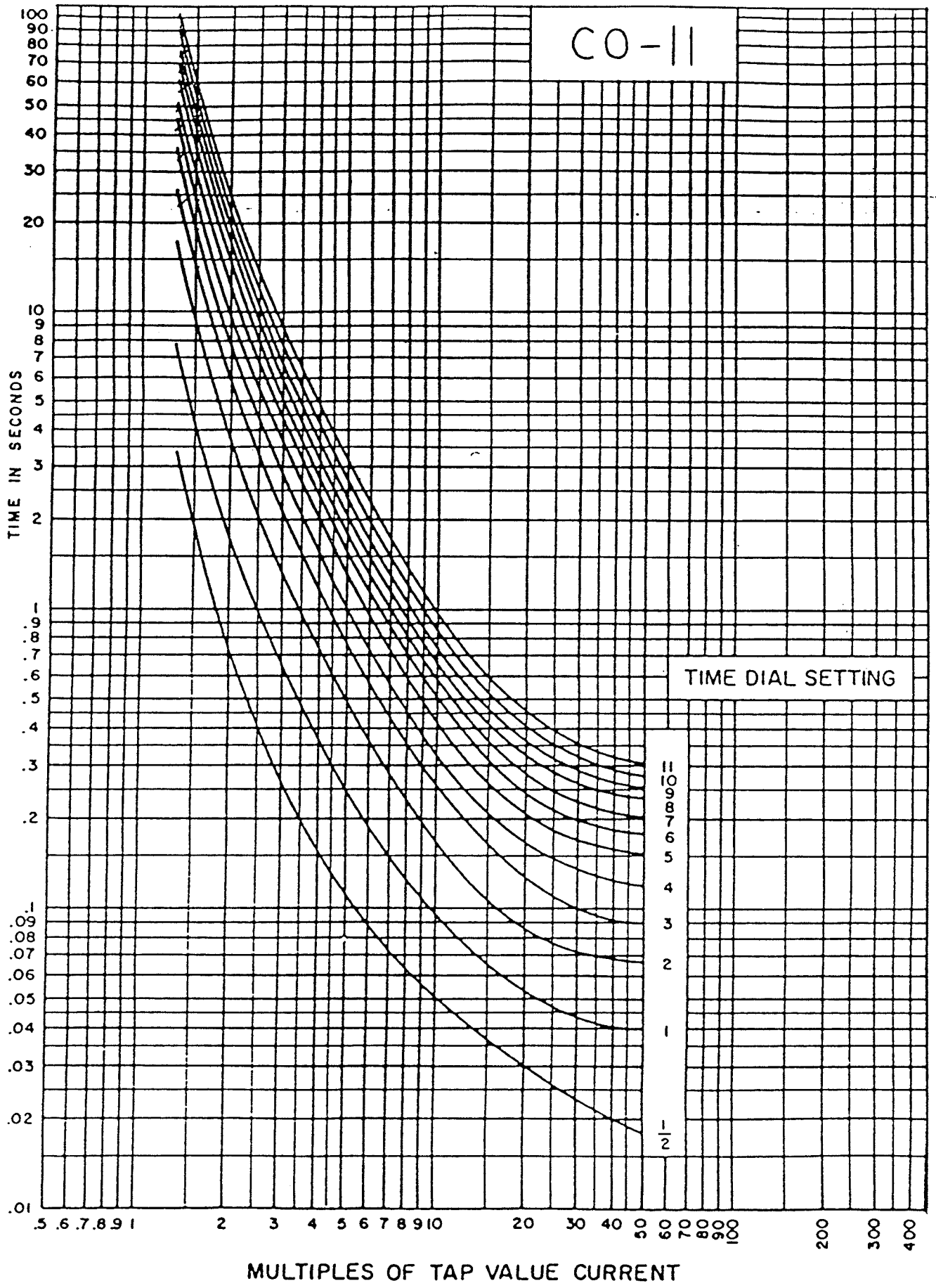
```
 1. //TNG JOB '1222335,,T=05'
 2. //  EXEC FORTXCLG,OPT=2,USERLIB='MENZIES.MAIN.LOAD'
 3. //FORT.SYSIN DD *
 4. C
 5. C FILENAME : TEST3
 6. C TRIP SIMULATION WITH AN INITIAL INSTANTANEOUS TRIP AND RECLOSURE.
 7. C
 8.         COMMON DELTAT,DELPHA
 9.         INTEGER TYPE,FLAG
10.         REAL M,MF,A,B,C,A1,A2,A3,A4,PHATA,DELTAT,DELPHA
11. C
12.         DELTAT=0.0001
13.         DELPHA=0.001
14. C
15.         READ *,TYPE,TDS,MF,TI,TRC
16.         CALL INABC(TYPE,TDS,PHATA,FLAG,T,A,B,C,A1,A2,A3,A4)
17. C
18.         WRITE(6,10)
19.  10     FORMAT('1',//,5X,'INPUT INFORMATION :')
20. C
21.         IF (TYPE.EQ.8) GO TO 20
22.         IF (TYPE.EQ.11) GO TO 30
23.         IF (TYPE.EQ.51) GO TO 40
24.         IF (TYPE.EQ.77) GO TO 50
25. C
26.  20     WRITE(6,21) TDS
27.  21     FORMAT(//,5X,'RELAY TYPE : CO-8 ',2X,';   TDS = ',F4.1)
28.         GO TO 60
29.  30     WRITE(6,31) TDS
30.  31     FORMAT(//,5X,'RELAY TYPE : CO-11 ',2X,';   TDS = ',F4.1)
31.         GO TO 60
32.  40     WRITE(6,41) TDS
33.  41     FORMAT(//,5X,'RELAY TYPE : IAC51 ',2X,';   TDS = ',F4.1)
34.         GO TO 60
35.  50     WRITE(6,51) TDS
36.  51     FORMAT(//,5X,'RELAY TYPE : IAC77 ',2X,';   TDS = ',F4.1)
37.         GO TO 60
38. C
39.  60     WRITE(6,61) A,B,C
40.  61     FORMAT(/,5X,'A , B , C  =',3(2X,F10.7))
41.         WRITE(6,70) A1,A2,A3,A4
42.  70     FORMAT(/,5X,'A1 , A2 , A3 , A4  =',4(2X,F10.7))
43.         WRITE(6,80) MF
44.  80     FORMAT(/,5X,'FAULT CURRENT, MF = ',F10.7)
45.         WRITE(6,90) TI
46.  90     FORMAT(/,5X,'INITIAL TRIP TIME BY INSTANTANEOUS RELAY, TI = ',
47.       & F6.3,' SECONDS.')
48.         WRITE(6,100) TRC
49.  100    FORMAT(/,5X,'RECLOSURE TIME, TRC = ',F6.3,' SECONDS.')
50. C
51.         M=MF
52.  110    CALL OCABC(TDS,M,T,PHATA,FLAG,A,B,C,A1,A2,A3,A4)
53.         IF (T.GE.TI) GO TO 125
54.         IF (FLAG.EQ.0) GO TO 110
55.  120    WRITE(6,121)
56.  121    FORMAT(/////,5X,'!!!!! TEST FAILED !!!!!')
57.         STOP
58.  125    WRITE(6,130)
59.  130    FORMAT(///,5X,'OUTPUT INFORMATION :')
60.         WRITE(6,131) PHATA
61.  131    FORMAT(//,5X,'DEGREE OF DISC TRAVEL, PHATA = ',F10.7,
62.       & '  AT INITIAL TRIP.')
63.         M=0.0
64.  140    CALL OCABC(TDS,M,T,PHATA,FLAG,A,B,C,A1,A2,A3,A4)
```

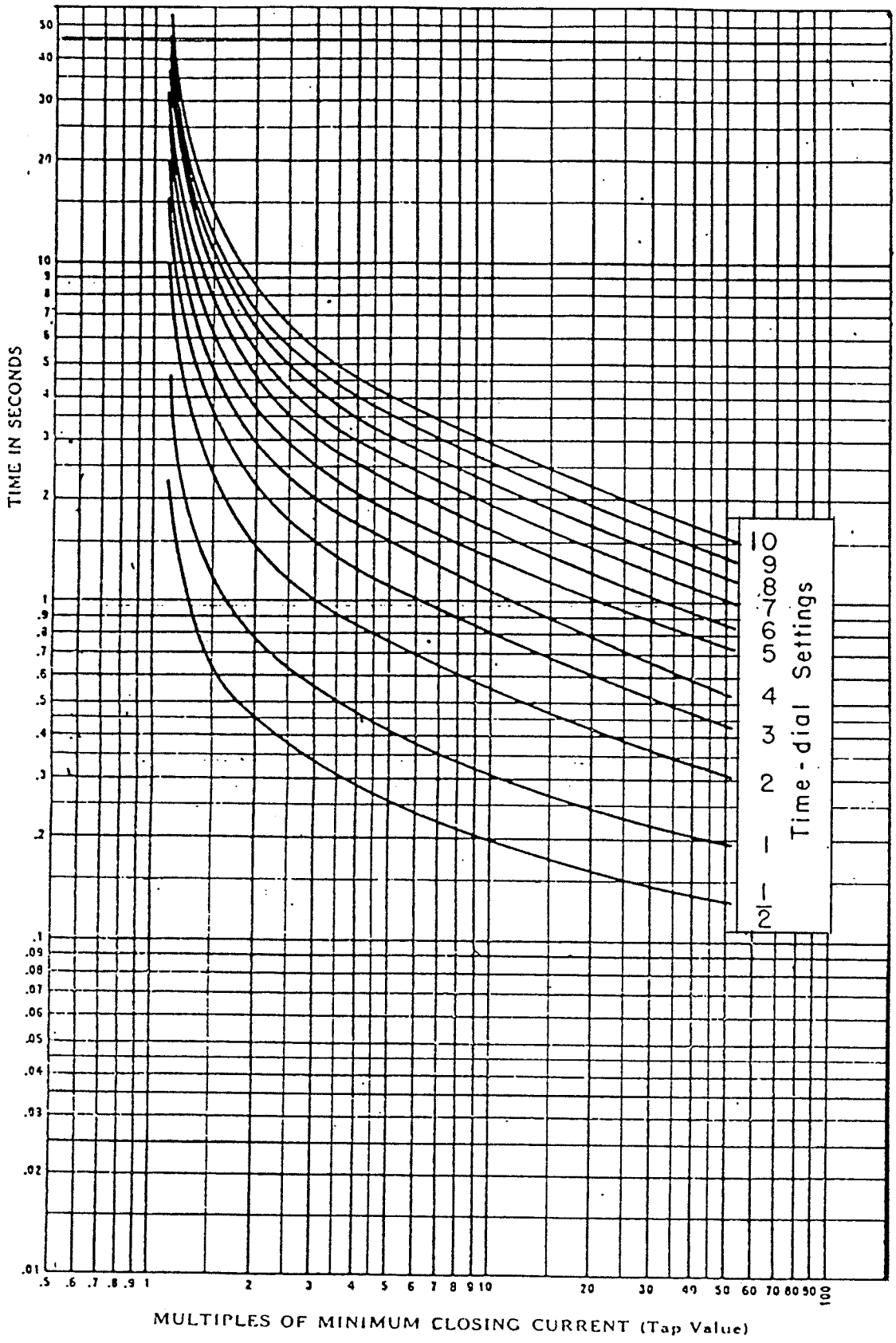
```
      IF (PHATA.EQ.0.0) GO TO 120
      IF (T.GE.TRC) GO TO 150
      GO TO 140
150   WRITE(6,151) PHATA
151   FORMAT(/,5X,'DEGREE OF DISC TRAVEL, PHATA = ',F10.7,
     & '  AT RECLOSURE.')
      M=MF
160   CALL OCABC(TDS,M,T,PHATA,FLAG,A,B,C,A1,A2,A3,A4)
      IF (FLAG.EQ.0) GO TO 160
      TO=T-TRC
      WRITE(6,170) TO
170   FORMAT(/,5X,'TIME BETWEEN RECLOSURE AND FINAL TRIP, TO = ',
     & F10.7,' SECONDS.')
      STOP
      END
C
C*************************************************************************
C
      SUBROUTINE OCABC(D,M,T,PHATA,FLAG,A,B,C,A1,A2,A3,A4)
      COMMON DELTAT,DELPHA
      INTEGER FLAG
      REAL M
C
      IF (M.LE.1.0) GO TO 20
      IF (FLAG.EQ.0) GO TO 10
      T=T+DELTAT
      GO TO 30
C
 10   F=C*(M-1)**B/(A+(M-1)**B)
      W = A1 + A2*F + A3*F**2 + A4*F**3
      PHATA=PHATA+W*DELTAT
      T=T+DELTAT
C
      IF (PHATA.LT.D) GO TO 30
      PHATA=D
      FLAG=1
      GO TO 30
C
 20   FLAG=0
      IF (PHATA.EQ.0.0) GO TO 30
      PHATA=PHATA-DELPHA
      T=T+(70.0/10.0)*DELPHA
      IF (PHATA.GT.0.0) GO TO 30
      PHATA=0.0
 30   RETURN
      END
C
C*************************************************************************
C
      SUBROUTINE INABC(TYPE,TDS,PHATA,FLAG,T,A,B,C,A1,A2,A3,A4)
      INTEGER R,TYPE,FLAG
      REAL CO8(12,7),CO11(12,7)
      REAL IAC51(11,7),IAC77(11,7)
C
C TYPE : CO-8 ; TDS=6
C
      CO8(7,1)=18.46631
      CO8(7,2)= 1.39766
      CO8(7,3)= 7.55713
      CO8(7,4)=-0.153045
      CO8(7,5)= 1.448598
      CO8(7,6)=-0.165898
      CO8(7,7)= 0.015019
C
```

```
129.          R=0
130.          R=TDS+1
131.          IF (TYPE.EQ. 8) GO TO 20
132.          IF (TYPE.EQ.11) GO TO 30
133.          IF (TYPE.EQ.51) GO TO 40
134.          IF (TYPE.EQ.77) GO TO 50
135.          WRITE(6,10)
136.   10     FORMAT(/////,5X,'!!!!! RELAY TYPE NOT AVAILABLE !!!!!')
137.          STOP
138.  C
139.   20     A=CO8(R,1)
140.          B=CO8(R,2)
141.          C=CO8(R,3)
142.          A1=CO8(R,4)
143.          A2=CO8(R,5)
144.          A3=CO8(R,6)
145.          A4=CO8(R,7)
146.          GO TO 70
147.   30     A=CO11(R,1)
148.          B=CO11(R,2)
149.          C=CO11(R,3)
150.          A1=CO11(R,4)
151.          A2=CO11(R,5)
152.          A3=CO11(R,6)
153.          A4=CO11(R,7)
154.          GO TO 70
155.   40     A=IAC51(R,1)
156.          B=IAC51(R,2)
157.          C=IAC51(R,3)
158.          A1=IAC51(R,4)
159.          A2=IAC51(R,5)
160.          A3=IAC51(R,6)
161.          A4=IAC51(R,7)
162.          GO TO 70
163.   50     A=IAC77(R,1)
164.          B=IAC77(R,2)
165.          C=IAC77(R,3)
166.          A1=IAC77(R,4)
167.          A2=IAC77(R,5)
168.          A3=IAC77(R,6)
169.          A4=IAC77(R,7)
170.          GO TO 70
171.  C
172.   70     PHATA=0.0
173.          FLAG=0
174.          T=0.0
175.          RETURN
176.          END
177.  //GO.SYSIN DD *
178.  8    6    3.0    3.0    18.0
```

Appendix N

INVERSE TIME CHARACTERISTIC CURVES OF CO-11, IAC51 AND IAC77

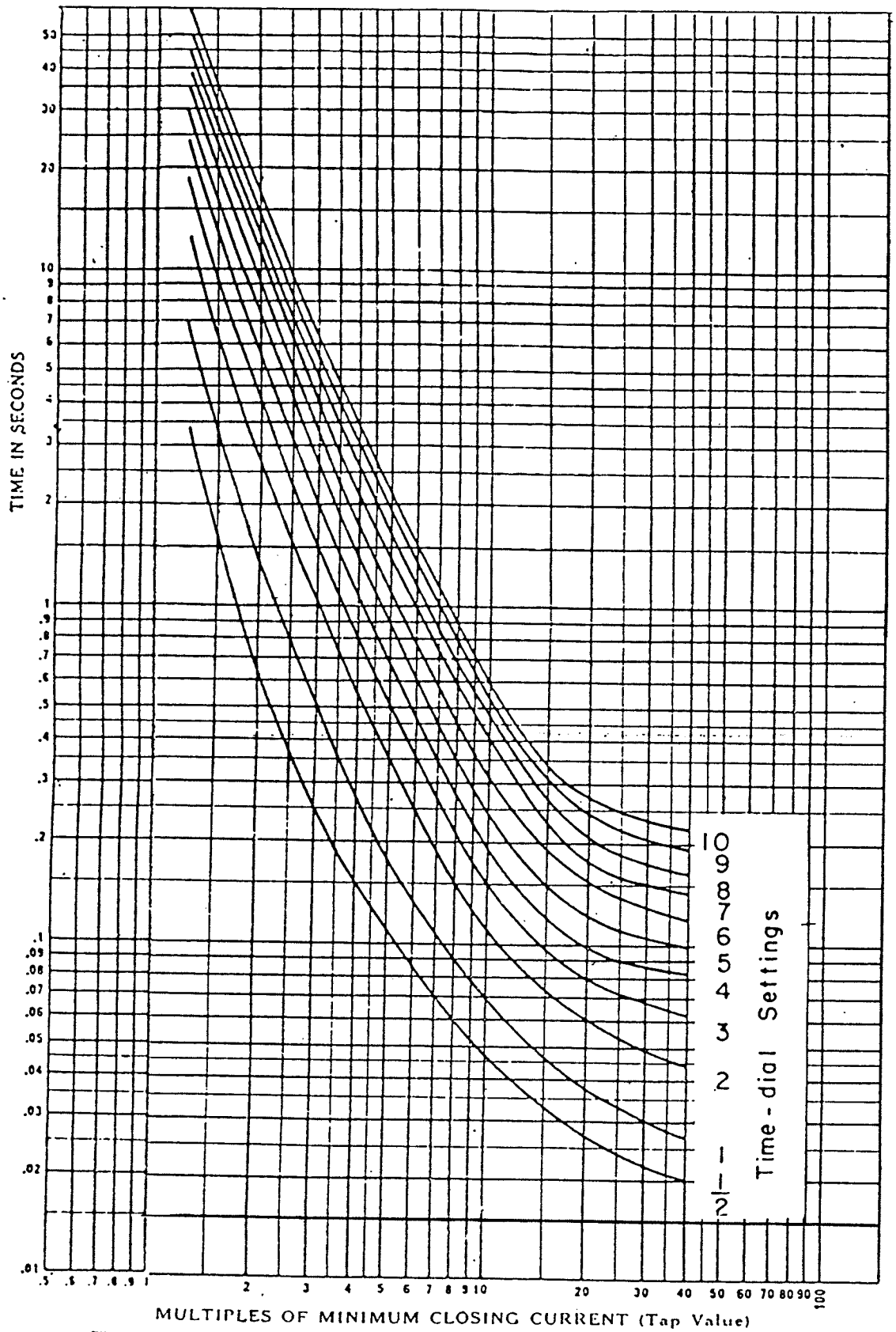

CO-11
TIME IN SECONDS
100
90
80
70
60
50
40
30
20
10
9
8
7
6
5
4
3
2
1
.9
.8
.7
.6
.5
.4
.3
.2
.1
.09
.08
.07
.06
.05
.04
.03
.02
.01
.5 .6 .7 .8 .9 1 2 3 4 5 6 7 8 9 10 20 30 40 50 60 70 80 90 100 200 300 400
TIME DIAL SETTING
11
10
9
8
7
6
5
4
3
2
1
1/2
MULTIPLES OF TAP VALUE CURRENT



TIME-CURRENT CHARACTERISTIC CURVES

Types IAC51, 52, 60, or 61 Relays

INVERSE



TIME-CURRENT CHARACTERISTIC CURVES

Type IAC77 Relay

EXTREMELY INVERSE