PARAMETER OPTIMIZATION OF A THIRD-ORDER DIFFERENTIAL EQUATION

by

KENNETH WAYNE SWITZER

B. S., Kansas State University, 1966

A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1968

Approved by:

W. N. Gruesel
Major Professor

TABLE OF CONTENTS

Chapter		Page
I.	THE PROBLEM	1
	Introduction	1
	Statement of the Problem	2
II.	THEORETICAL ANALYSIS	4
	Discussion	4
	Analysis of Methods Used	6
III.	PROCEDURE	9
	Input	10
	Phase 1	11
	Phase 2	13
	Phase 3	17
	Output	20
	System Simulation	20
IV.	RESULTS AND CONCLUSION	22
	Results	22
	Conclusion	27
BIBLIOGRAPHY		28
APPENDIX A		29
APPENDIX B		30

LIST OF TABLES

Table — Page

I. Comparison of Distance Functions and Error Definitions for Several Performance Measures. 24

LIST OF FIGURES

Figure — Page

1. Plot of Distance Function DF1 as a Function of Parameters A_1 and A_2. 5
2. Sample Curves Showing Cases 1A, 1B, 1C, and 1D. 11
3. Flowchart of Phase 1. 12
4. Sample Curves Showing Cases 2A, 2B, 2C, and 2D. 15
5. Flowchart of Phase 2. 16
6. Flowchart of Phase 3. 18
7. Plot of Response and Input Versus Time for DF1. 25
8. Plot of Response and Input Versus Time for DF2. 26

CHAPTER I

THE PROBLEM

INTRODUCTION

This paper is concerned with the following type of optimal control problem:

> For a given topological configuration of a linear control system and a given input f(t), the parameter values are to be found which optimize some performance measure.

Traditionally problems of this type have been solved by trial and error (Bach, 1965). However this is time consuming, costly, and impractical on large systems. In recent years considerable interest has developed in using computers as a design tool to simulate the control system and find the most desirable system parameters. This is done by defining a performance measure or measures in terms of a distance function. A distance function relates quality of performance to the least distance from the origin. Any number of performance measures can be combined in a single distance function. An example of a commonly used distance function is the square root of the sum of the squares of the performance criterion (Levine, 1964).

In control systems the performance measure usually involves the integral of some form of error between the input and the output. Other criterion that may be used are overshoot, undershoot, time delay, and settling time (Kuo, 1962). These measures are defined in Appendix A.

Most other treatments of this problem are restricted to step inputs. This approach allows a more general f(t) that has a constant final value after

some time T_1. The assumption is made in this paper that the input f(t) never exceeds the final value. If the input f(t) is larger than the final value for some time T_0 less than T_1, then the performance measures must be redefined to take this into account.

This paper describes a general FORTRAN program to solve the above problem. The program is written to find the optimal parameters of any control system that can be described by an N^{th} order $(2 < N < 8)$ linear differential equation with constant coefficients. The computation time increases very rapidly however, as the order of the system increases. To demonstrate the program, a specific problem is considered and the performance measure criterion are evaluated for two distance functions and three different error definitions.

STATEMENT OF THE PROBLEM

The program to be solved is as follows. The parameters of a third order linear differential equation are to be found which minimize a distance function involving settling time T_s and the integral of some form of error. The input to the system is a modified step function consisting of a unit ramp to time one and unit step from one to infinity. Thus the system has the following form.

$$\dddot{y} + A_1\ddot{y} + A_2\dot{y} + y = f(t)$$

$$\begin{array}{lll} f(t) = 0 & -\infty > t > 0 \\ \quad\ t & 0 \geq t > 1 \\ \quad\ 1 & 1 \geq t \end{array}$$

The distance function is the square root of the sum of the squares of T_s and ERROR where T_s is defined as the time required for the response to decrease to and stay within five percent of the final value. Three different

ERROR definitions are used and compared.

1. $IAE = \int_0^T ERR\ dt$

   Integral of the absolute value of error.

   Note: ERR is the difference between the input and output. T can either be a fixed value designated by the user or the value of T_s.

2. $IES = \int_0^T ERR^2 dt$

   Integral of the error squared.

3. $ITES = \int_0^T t*ERR^2 dt$

   Integral of time multiplied error squared.

CHAPTER II

THEORETICAL ANALYSIS.

DISCUSSION

Because of the difficulties in generalizing analytical methods of optimization, search techniques are used to find the optimal parameter values. The distance function is evaluated each time the parameters are changed by solving the differential equation that describes the system. The distance function is then minimized by adjusting the parameters in some optimal way.

For efficiency two search methods, steepest descent and relaxation, are used. Steepest descent works best in adjusting the parameters from the initial values to somewhere near the optimum values. This method has problems with step sizes near the minimum and doesn't work when there are discontinuities in the solution space.

Due to the definition of T_s, the solution space for the distance function is discontinuous. Near the optimal parameter values, a slight change will move the solution outside the allowed range and markedly increase the value of T_s. This in turn increases the ERR and because it is integrated over a longer time.

In this problem the minimum values occur right next to a discontinuity in the solution space. (See Fig. 1) This is quite understandable when one looks at the definition of the distance function. The program will first try and make the response converge as soon as possible to minimize both T_s and ERR. Then once T_s is relatively constant, the program will move the overshoot

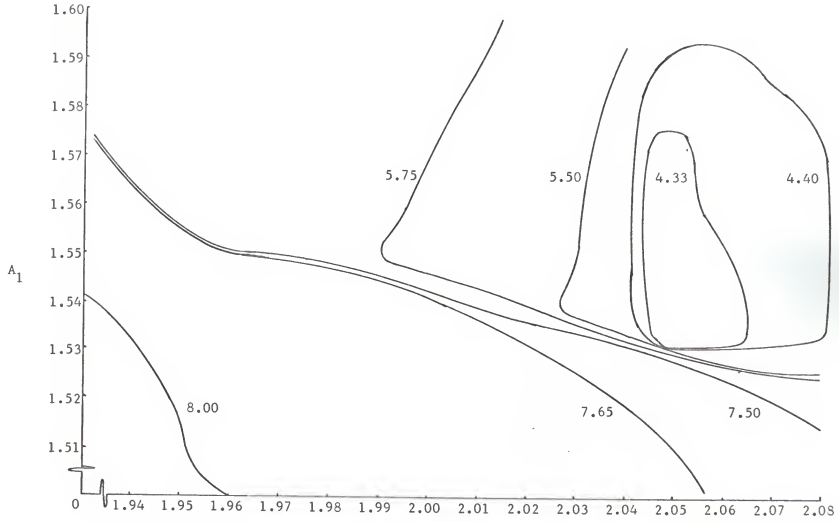

Fig. 1 Plot of distance function $\sqrt{T_s^2+ERR^2}$ as a function of parameters A_1 and A_2.
ERR=IAE DELTX=.04

as close to the limit as possible without exceeding it. Thus it can be seen why a very small increment will suddenly cause a great change in the value of the distance function. This causes problems in the search technique because proper adjustments must be made when this happens. It is also easy to get caught on a relative minimum near a discontinuity.

This brings up questions as to the validity of the choice of performance measure. One of the requirements for an optimum system is mathematical tractability (Hancock, 1966). Obviously the choice of the performance measure is purely subjective. Therefore to be effective it must be based on good engineering judgement. This particular performance measure should only be used when necessary. In most problems, other measures that have continuous solution spaces would be more satisfactory.

As noted above the optimum parameter values represent a relative minimum and not necessarily an absolute minimum. When there are several minima it is necessary to run the program several times from different starting points to determine the absolute minimum.

Because the parameters represent physical components the accuracy is limited to one part in a thousand. It would be senseless to obtain optimal parameter values with greater accuracy than this because the components would change with time or temperature. The parameters should not be picked too near a discontinuity in the solution space for the same reason.

This paper assumes there should be no steady state error. This constrains the parameter on y to the value 1. Thus for an N^{th} order differential equation there will be N-1 adjustable parameters.

Analysis of Methods Used

Three numerical methods are used in this paper. Two are search tech-

niques and the other is a method for finding the solution to a differential equation. The methods are reviewed briefly here.

Relaxation

In this search technique a single parameter is varied with all others held constant until a local minimum is reached. Slope is used to determine the direction of descent. Each time the slope changes, the stepsize is reduced by an order of magnitude. This continues until some termination criterion is satisfied. When the solution space is discontinuous some additional checks are made to insure convergence.

Steepest Descent

The distance function is defined as $D(A_1,\ldots,A_n)$. This function can be minimized in a region R of the N- dimensional rectilinear parameter space if one starts from an arbitrarily selected point in the region and follows a path which leads to decreasing values of D. The rate of change dD/dt is greatest if the path chosen in the parameter space is tangential to the gradient vector in this same space (Levine, 1964). This may be written as an inner product as follows:

$$dD/dt = (\underline{\text{grad}}\ D) \cdot (\underline{\dot{A}})$$

where

$$\underline{\dot{A}} = \dot{A}_1\underline{u}_1+\dot{A}_2\underline{u}_2+ \ldots \dot{A}_n\underline{u}_n$$

The u_i's are unit vectors along the coordinate axis. The minimum occurs when the two vectors are parallel and point in opposite directions (Levine, 1964). This may be written as follows:

$$dA_i/dt = -(\partial D/\partial A_i) \qquad i = 1,2,\ldots n$$

This may be approximated by the following difference equation:

$$\Delta A_i = -(\Delta D/\Delta A_i)\Delta t \qquad i = 1,2,\ldots n$$

One variation on steepest descent that saves some computer time is to

calculate the steepest descent stepsizes at a point and increment with those same stepsizes until a local minimum is reached. The local minimum is taken as the new starting point and the process is repeated.

Euler's Method

Euler's Method is used to solve the differential equation because of its simplicity and minimal computation time. If higher accuracy would be needed some other method such as Runge-Kutta could be used (Conte, 1965). The choice of the time increment has an important part in the accuracy of the solution, the accuracy of the settling time T_s, and the computation time required to find the solution. The value 0.01 seems to be a good tradeoff between time and accuracy. Euler's Method can be stated in the following general form.

$$y_{n+1} = y_n + \dot{y} \cdot \Delta t$$

To solve the differential equation $\dot{y} + y + f(t)$ where $f(t)$ is the input, the following difference equations are solved repeatedly.

$$t_{n+1} = t_n + \Delta t$$

$$\dot{y}_n = -y_n + f(t)$$

$$y_{n+1} = y_n + \dot{y}_n \cdot \Delta t$$

The method can be generalized to solve an N^{th} order differential equation by changing it to a set of N first order differential equations.

CHAPTER III

PROCEDURE

The description of the program can be divided into six main parts. These are Input, Phase 1, Phase 2, Phase 3, Output, and System Simulation. Through the input the user can specify the order of the differential equation, the initial parameter values and the initial stepsize values. By changing the appropriate FUNCTION Subroutines the user can specify the input f(t) and the distance function. Phases 1 and 2 are used to insure the convergence of the response so there will be a solution for the settling time. Phase 3 makes the final parameter adjustments and makes adjustments to discontinuities in the solution space. A simplified distance function defined as the integral from zero to twenty-five seconds of the absolute value between the input and output is used in Phases 1 and 2. This is a continuous function that simplifies the search procedure. Phase 1 uses steepest descent search technique. Phases 2 and 3 use a relaxation search method. Each time the distance function is evaluated pertinent information is printed out so the user can know where the search procedure has been. When the optimal parameters are found they are printed out with the message PARAMETERS OPTIMIZED. The system simulation obtains the necessary information for calculating the distance function and uses a generalized Euler's Method for solving N^{th} order differential equations.

The program is basically written in FORTRAN II. The single exception to this is the IMPLICIT statement which makes all floating point variables

double precision. This means that every time the ABS, or SQRT Library Subroutines are used they must be specified as DABS and DSQT. The program was run on an IBM 360/50 using the FORTRAN IV Level G compiler on Release 12. The execution time for a typical run on the third order problem described in this paper is 6 minutes. This time increases with increasing order of the differential equation and decreasing stepsize.

INPUT

The user must specify the order of the differential equation, the intial parameter values, and the initial stepsizes. The program has a default so that if the user has no idea what values the parameters should be the program will arbitrarily set them to 1 and the initial stepsize to one tenth. JJ represents the order of the differential equation and KJ is the default code. JJJ is set equal to JJ-1. This is the number of adjustable parameters. IERR is a code that determines which ERROR definition is to be used (see page 33). AA is the time of integration in seconds used in Phase 1 and 2. DELTX is the time increment used in Phase 1 and 2. The input list JJ,KJ,IERR,AA,DELTX is read in on a 3I1,2X,2F5.0 format. JJ must be assigned a value ($2 < JJ < 8$) but all the other variables will take on a default value if left blank. The default values are KJ=0, IERR=1, AA=25., DELTZ=.05. If JJ is 9 the program will terminate. If KJ is positive two more cards are read with a 8F10.5 format. The first card should have parameter values A_i in ascending order. The second card should have stepsizes $DELT_i$ in ascending order. If KJ is positive the program goes directly to Phase 3. The default occurs if KJ is zero or blank. The input f(t) and distance function are modified by changing the appropriate FUNCTION Subroutines. (see Appendix B)

PHASE 1

This Phase used steepest descent to adjust the parameters A_i from their initial value of 1 to a set of parameters whose response converges to the input with minimal error. All parameters are adjusted simultaneously along a straight line tangent to the gradient at the starting point. This continues until a local minimum is reached or the parameters are adjusted 15 times. The parameter values are taken as a new starting point and the process is repeated two times. $DELT_i$ is defined to be the stepsize. The interval of integration, AA, is 25 seconds. When the $DELT_i$ are calculated only one parameter is varied at a time. ERR is the absolute value of the difference between the input and output $|y - f(t)|$.

STEP 1

The following quantities are evaluated.

H1= $\int_0 ERR \; dt$ for $A_i - DELT_i /10$

H2= $\int_0 ERR \; dt$ for A_i

H3= $\int_0 ERR \; dt$ for $A_i + DELT_i /10$

From these values four cases are obtained as follows:

The resulting curves are shown in Fig. 2.

CASE	
1A	H1 > H2 > H3
1B	H1 < H2 < H3
1C	H2 > H1, H3
1D	H2 < H1, H3

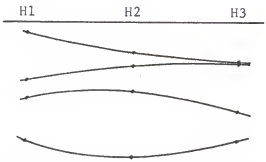

Fig. 2 Sample Curves Showing Cases 1A, 1B, 1C, and 1D.

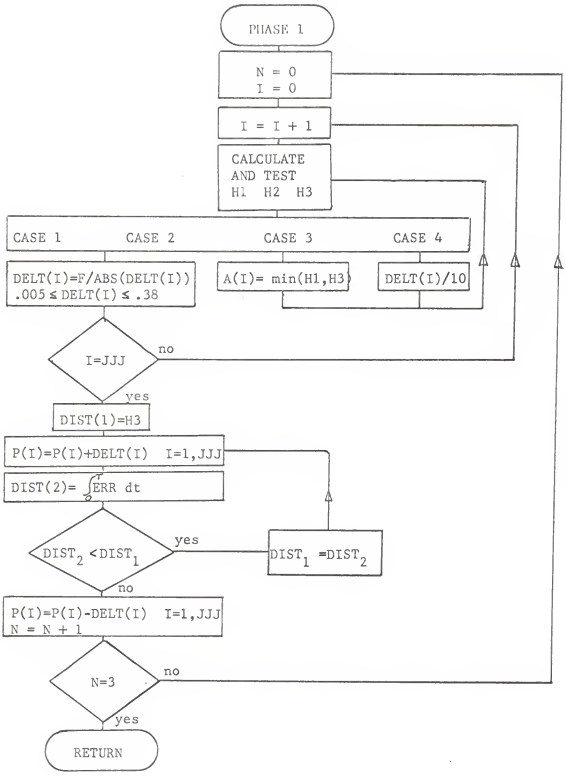

Fig. 3 Flowchart of Phase 1

F is defined as (H1 - H2).

After testing to see which case exists the following action is taken.

Case 1A and 1B	$DELT_i$ is redefined as F/ $DELT_i$. The program then goes to STEP 2.
Case 1C	A_i is redefined to that parameter value corresponding to the smaller of H1 or H2. STEP 1 is then repeated.
Case 1D	$DELT_i$ is decreased by an order of magnitude and STEP 1 is repeated

STEP 2

The resulting $DELT_i$ is tested as follows:

If	$DELT_i > 0.38$	$DELT_i$ is set equal to 0.38
If	$DELT_i < 0.005$	$DELT_i$ is set equal to 0.005

If all the $DELT_i$ have been calculated the program goes to STEP 3. If not I is incremented by 1 and the program goes to STEP 1.

STEP 3

After all the $DELT_i$ are calculated each A_i is incremented by the corresponding $DELT_i$. $\int_0^{AA} ERR\ dt$ is then evaluated repeatedly, incrementing the parameters each time. When the value of the integral is larger than the previous value a local minimum has been reached. All A_i are then decremented by $DELT_i$, I is set to one, and the counter KJJ is incremented by one. If KJJ is three the program goes to Phase 2. Otherwise the program goes to STEP 1. See Fig. 3 for a flow diagram of Phase 1.

PHASE 2

This phase uses a relaxation search method to make the parameter response converge even more quickly. A single parameter is varied with all others held fixed. The parameter is varied until one of three things occur.

First, the difference between the two distance functions is negligible, second, the stepsize may get too small, or third, the parameter has been varied more than 15 times. If one of these criterion is true the next parameter is then varied. Each parameter is varied in turn until the change in the distance function for JJJ successive parameter changes is negligible. The distance function used is the integral from zero to AA seconds of the absolute value of the difference between the input f(t) and the output. AA is 25 seconds in this report.

The initial parameter values A_i are those calculated in Phase One. The initial stepsize $DELT_i$ are one tenth. All changes in stepsize are by orders of magnitude. CODE is plus or minus one corresponding to positive or negative slope of the distance function. DIST(K) corresponds to the most recent value of the distance function, DIST(K-1) the previous value, etc. Each time the simplified distance function is evaluated it is done twice, once for A_i, and once for $A_i + DELT_i * CODE/10$. By this means the slope can be determined and CODE set to the proper value. The parameter A_i is then incremented by $DELT_i * CODE$. Two more distance functions are then evaluated corresponding to the new A_i and $A_i + DELT_i * CODE/10$. When this is done one of the following Cases will result. CODE is taken to be plus one in this example. This implies that DIST(1) is greater than DIST(2). See Fig. 4 for sample curves.

CASE 2A DIST(1) > DIST(2) > DIST(3) > DIST(4)

This case is a desirable result. It means the search is continuing down a slope in the parameter space.

CASE 2B DIST(1) > DIST(3), DIST(4) > DIST(3)

This case shows the minimum has been crossed and that it is nearest DIST(3).

CASE 2C DIST(3) > DIST(1), DIST(4) > DIST(3)

This case shows the minimum has been crossed and that it is probably nearest DIST(2).

CASE 2D DIST(3) > DIST(1), DIST(3) > DIST(4)

This case indicates that a discontinuity has been crossed in the solution space or that the stepsize is very large.

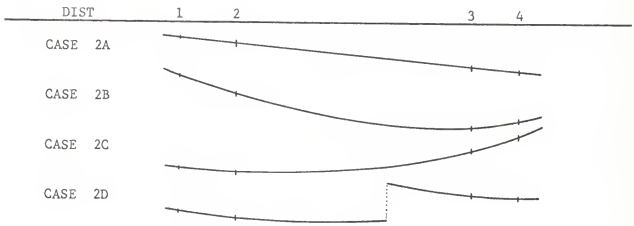

Fig. 4 Sample Curves Showing Cases 2A, 2B, 2C, and 2D.

The above cases are tested for and the following action is taken.

CASE 2A A_i is incremented by $DELT_i$.

CASE 2B $DELT_i$ is decreased by an order of magnitude and 3*CODE*$DELT_i$ is subtracted from A_i.

CASE 2C $DELT_i$ is decreased by an order of magnitude and 7*CODE*$DELT_i$ is subtracted from A_i.

CASE 2D $DELT_i$ is decreased by an order of magnitude and A_i is set to the parameter value corresponding to DIST(2).

If one of the following criterion are not satisfied, the program sets DIST(1)= DIST(3) and DIST(2)= DIST(4). The program then sets CODE to the proper value or finds DIST(3) and DIST(4) corresponding to the new A_i and A_i DELT(I)*CODE/10.

1. $|DIST(3)-DIST(4)| < DMIN$ This means negligible improvement from the change in parameter.

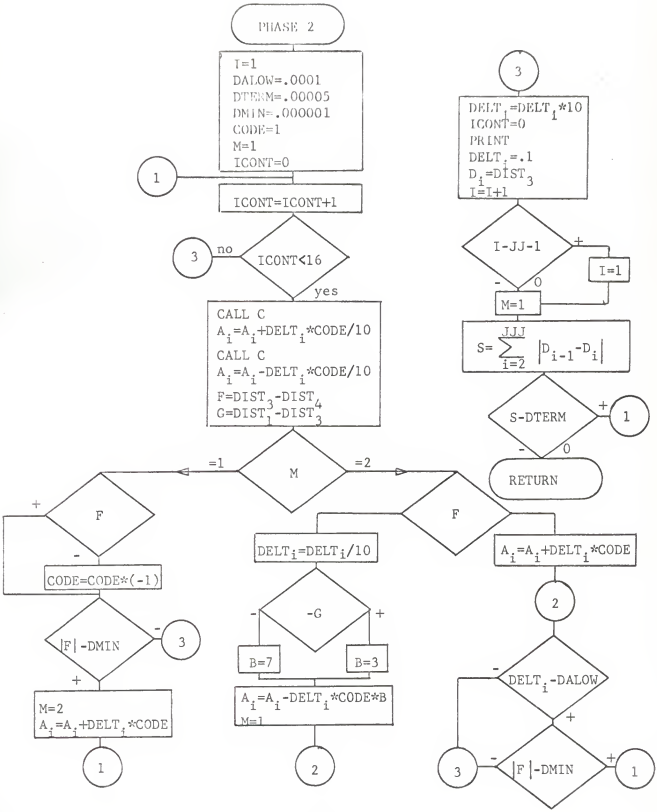

Fig. 5 Flowchart of Phase 2

2.	$\lvert DELT_i \rvert <$ DALOW	This indicates that the stepsize is less than that allowed by the user.
3.	ICONT > 15	This means a single parameter has been adjusted more than 15 times successively. Levine(1965) shows that by limiting the number of adjustments of a single parameter the minimum is found more quickly.

When one of the above criterion is satisfied D_i is defined to be equal to the last $DIST_3$ calculated. The following overall termination criterion is then tested.

If $\sum_{i=2}^{JJJ} \lvert D_{i-1}-D_i \rvert <$ DTERM The program goes to Phase 3.

If this inequality is not satisfied I is changed so the next parameter will be adjusted. The program returns to set CODE to the proper value and the process is repeated until the termination criteria is satisfied. See Fig. 5 for a flow diagram of Phase 2.

PHASE 3

This phase is very similar to Phase 2. However there are three important differences. First the required difference function $\sqrt{T_s^2+ERROR^2}$ is used, second a smaller stepsize DELTX is used, and third some additional tests are made to determine when the search has crossed a discontinuity. When a discontinuity is detected, the stepsize is decreased by an order of magnitude and the parameter is adjusted so it is on the small side of the discontinuity. If the termination criterion is satisfied while on the wrong side of a discontinuity, A_i is set to its previous value and D_i is set equal to $DIST_1$ instead of $DIST_3$. The action taken after testing for the four cases (see Phase 2) is different for cases 3B and 3C.

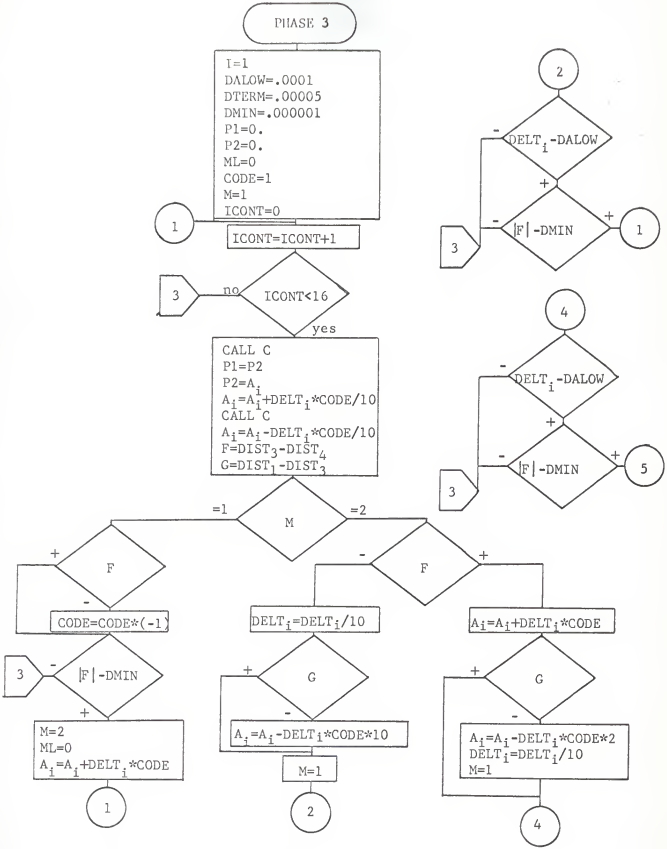

Fig. 6 Flowchart of Phase 3
continued on next page

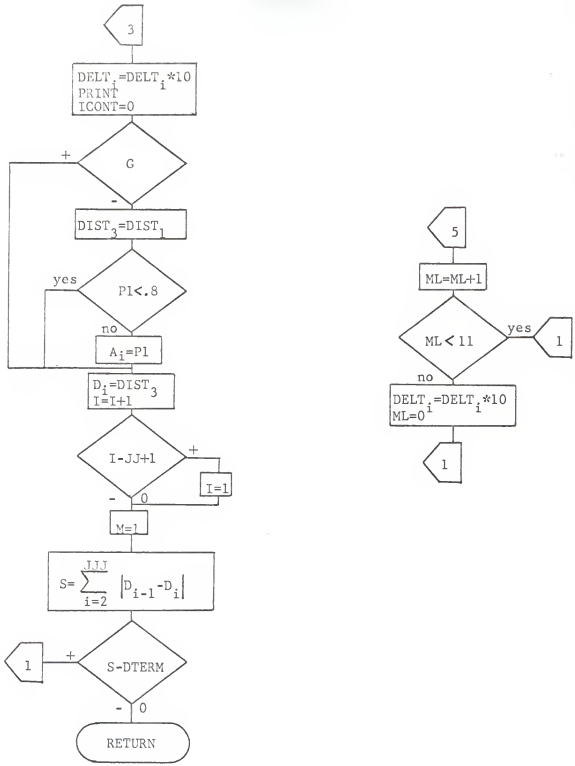

Fig. 6 Continued Flowchart of Phase 3

Case 3B $DELT_i$ is decreased by an order of magnitude.

Case 3C $DELT_i$ is decreased by an order of magnitude and 10*CODE*$DELT_i$ is subtracted from A_i.

When Phase 3 is terminated the program prints PARAMETERS OPTIMIZED and prints out the optimal parameter values. See Fig. 6 for a flow diagram of Phase 3.

OUTPUT

The program will terminate in one of three ways. The desired termination is taken when the parameters are optimized. If the parameters have a response that does not converge within 25 seconds the program will also terminate. If the distance function $\sqrt{T_s^2+ERROR^2}$ is evaluated more than 400 times for the same input data, the program will terminate and print out the message PROGRAM TERMINATED AFTER 401 LOOPS. Because of the possibility of these undesired terminations the user needs to know what the program has done up to the time of the termination. Each phase prints pertinent information applicable to that Phase.

In Phase 1 the new $DELT_i$ are printed each time they are evaluated. Each time the parameters are incremented the program prints out the corresponding distance function and the parameter values. In Phase 2 the program prints $DIST_3$, $DIST_4$, CODE, I, A_i, and $DELT_i$. In Phase 3 the program prints the same as in Phase 2 plus the values for AREA and T. Whenever I is incremented in Phases 2 and 3, the parameter values and $DELT_i$ are printed. At the end of the program if the proper termination criterion is satisfied the program prints PARAMETERS OPTIMIZED and prints the parameter values.

SYSTEM SIMULATION

The actual mathematics of the simulation is described in the theoretical analysis. Interwoven with this are all the necessary checks to evaluate the distance functions. Some problems are encountered here because of round off error and the inability of a binary computer to represent decimal numbers exactly. Double precision is used on all floating point variables to try and improve the accuracy. The round off error can be most readily seen in the value representing time. Because it is determined by repeatedly adding small increments a large number of times the roundoff error becomes very apparent. This also affects the accuracy of the solution to the differential equation.

If the value of time exceeds 30 seconds when using the required distance function, T_s is set equal to 25, the distance function calculated, and control is returned to Phase 3.

LLL is the key used to determine which distance function is to be evaluated. LLL is equal to one for Phases 1 and 2. This causes the subroutine to evaluate the continuous distance function. LLL is equal to two in Phase 3. This causes the assumed distance function to be used. LLL is equal to three when a listing is made of the response of the differential equation versus time.

CHAPTER IV

RESULTS AND CONCLUSION

RESULTS

The problem to be solved is repeated as follows. The parameters of a third order differential equation are to be found which minimize the distance function $\sqrt{T_s^2+ERROR^2}$ with a modified step input. T_s is the time required for the response to decrease to and stay within five percent of the final value of the input. ERR is defined to be the difference between the input and the output. Three different ERROR definitions are used as follows.

1. IAE Integral of the absolute value of ERR.
2. IES Integral of ERR squared.
3. ITES Integral of time times ERR squared.

In addition to solving the above problem, optimal parameter values were found for the simplified distance function ERROR for each of the three definitions. This was done for a comparison of the performance measures of the distance functions for each ERROR definition. See Table 1 for the tabulation of the performance data and optimal parameter values.

For clarity the distance functions $\sqrt{T_s^2+ERROR^2}$ and ERROR are hereafter referred to as DF1 and DF2 respectively.

Because of the definition of DF1 the optimal parameters are the same for all three ERROR definitions. This result is caused by the high sensitivity of DF1 to the settling time T_s for this order differential equation. There is only one set of parameters that cause the output to respond most quickly

with no more than five percent overshoot or undershoot. Because of this the values of ERROR have little or no effect in determinimg the optimum parameter values with DF1. See Fig. 7 for a plot of the optimal response versus time for DF1.

When DF2 is used there are no constraints on overshoot and undershoot. The time of integration must be specified as twenty-five seconds because it is not implied in the definition of DF2 as it is in DF1. The performance values for DF2-IAE are nearly the same as those obtained from DF1. DF2-IES and DF2-ITES have faster response times but the settling time and overshoot are increased. From the data in Table 1 it can be seen that when one performance measure improves one or more of the others get worse. Thus there is a tradeoff between performance measures and good engineering judgment must be used to select the distance function that emphasizes the most desirable combination of performance measures. See Fig. 8 for a plot of the optimal response versus time for the three ERROR definitions for DF2.

A small change in the stepsize DELTX can cause rather significant changes in the calculated optimum parameter values. This is caused by the accuracy limitations of Euler's Method with large stepsize values. If higher accuracy is needed some other method such as Runge-Kutta should be used for solving the differential equation. In most cases however, high accuracy is not needed. With good judgment fairly large stepsizes can be used to determin rough optimum values with very little computation time.

The weakest part of the program is the time required to solve the differential equation. Since the system must be simulated many times in order to find the optimum values, most of the computer time is used in doing this simulation. In order to reduce the cost of finding the solution the simulation

TABLE 1

COMPARISON OF DISTANCE FUNCTIONS AND ERROR DEFINITIONS FOR SEVERAL PERFORMANCE MEASURES

DISTANCE FUNCTION	ERROR TYPE	OPTIMUM PARAMETERS A_1	A_2	OVERSHOOT percent	UNDERSHOOT percent	TIME DELAY seconds	RISE TIME seconds	SETTLING TIME seconds	ERROR VALUE
DF1	IAE	1.494	2.015	5.00	4.98	2.51	2.10	3.76	4.249
DF1	IES	1.494	2.015	5.00	4.98	2.51	2.10	3.76	4.001
DF1	ITES	1.494	2.015	5.00	4.98	2.51	2.10	3.76	2.039
DF2	IAE	1.470	2.080	2.64	5.31	2.51	2.15	3.84	2.156
DF2	IES	0.979	1.989	7.04	16.11	2.35	1.81	11.26	1.306
DF2	ITES	1.218	1.992	5.77	9.80	2.43	1.92	7.57	2.010

DF1 = $\sqrt{T_s^2+ERROR^2}$

DF2 = ERROR The error type is integrated from zero to fifteen seconds.

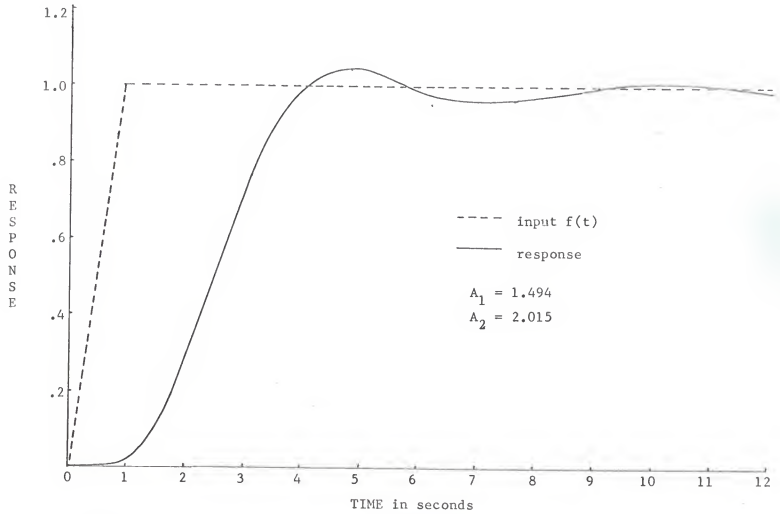

Fig. 7 Plot of Response and Input versus Time for DF1

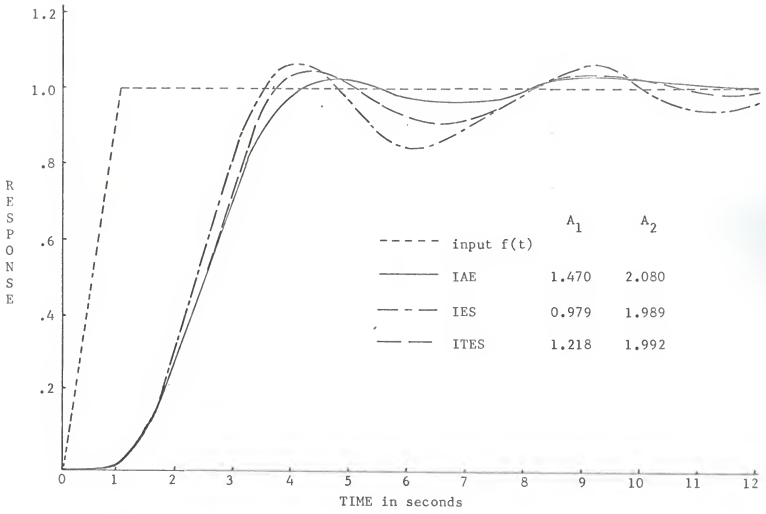

Fig. 8 Plot of Response and Input versus Time for DF2

routine must either be used less or a cheaper method of simulation used. Better prediction or extrapolation methods would help reduce the number of times the routine must be used. A high order system could be simulated more cheaply on a hybrid computer. For this reason, the program was written so it could be adapted to a hybrid computer.

CONCLUSION

The program presented in this paper is a practical approach to the optimal parameter design problem. With the use of the program and a judicious choice of a distance function that emphasizes the desired performance measures, the user can quickly determine the optimal parameter values for an N^{th} order ($2 \leq N \leq 8$) differential equation. The distance function should, if at all possible, be chosen so it is a continuous function. Only when the system must have special response characteristics should a discontinuous distance function be used.

The next step in the development of the program should be to generalize it so the program could simulate systems with both poles and zeros. This would greatly increase the usability of the program.

SELECTED BIBLIOGRAPHY

1. Bach, R. E. Jr. "A Practical Approach to Control System Optimization." Proceedings of IFAC Tokyo Symposium on Systems Engineering for Control System Design, Section 3. Tokyo, 1965.

2. Conte, S.D. Elementary Numerical Analysis, New York: McGraw-Hill, 1965.

3. Hancock, J. C. An Introduction to the Principles of Communication Theory, New York: McGraw-Hill, 1961.

4. Kuo, B. C. Automatic Control Systems, Englewood Cliffs: Prentice Hall Inc., 1962.

5. Levine, Leon. Methods for Solving Engineering Problems Using Analog Computers, New York: McGraw-Hill, 1964.

APPENDIX A

DEFINITION OF PERFORMANCE MEASURES

These terms are defined somewhat differently than usual because they are not limited to step inputs. Any input f(t) can be used that has a constant final value after some time T_1 and never exceeds the final value.

Overshoot	Percent overshoot is defined as the difference between the maximum response r(t) after T_1 and the final value of f(t) times 100 divided by the final value.
Undershoot	Percent undershoot is defined as the difference between the final value of f(t) and the minimum response after the first negative slope crossing of the final value line after T_1, multiplied times 100 and divided by the final value.
Time Delay	Time delay is defined as that time T_d required for the response to reach 50 percent of its final value.
Rise Time	The rise time T_r is defined as the time required for the response to rise from 10 percent of its final value to 90 percent of its final value.
Settling Time	The settling time T_s is defined as the time required for the response to decrease to and stay within 5 percent of its final value.

APPENDIX B

COMPUTER PROGRAM

The following pages contain a listing of the Fortran program described in this paper. The program consists of a main calling routine, four subroutines, and three FUNCTION subroutines. The program was written in this manner so the user could easily change it to meet his needs. If a faster execution time is desired, the FUNCTION subroutines should be incorporated into the calling routine. Comment statements are interspersed throughout the program to help the user understand what the program does.

The following are explanations of the three FUNCTION subroutines.

FUNCTION AINPUT(Z)

This routine calculates the value of the input as a function of time. Z is the variable that represents time. The user can change this routine as desired.

FUNCTION DISTFN(AREA,T)

This routine calculates the value of the distance function called for in the problem described in this paper. If this routine is changed to use other values in the distance function, SUBROUTINE C should be changed accordingly to supply needed values or delete uneeded values. The user can change this routine as desired.

FUNCTION RRR(ERR,Z)

This routine is used to allow several different ERROR definitions to be evaluated in the same program run. The program keys off the value of IERR to determine the ERROR definition. Care should be taken when changing this routine because of the COMMON statement.

```
//OPTIMIZE     JOB  03F40408G002,-SWITZER-,MSGLEVEL=1  010,040,000
//    EXEC   FTGCLGKS,PARM.FORT=-BCD-
//FORT.SYSIN   DD   *
C
C
C     ***************************************************************
C     ***PARAMETER OPTIMIZATION OF A JJ TH ORDER DIFFERENTIAL EQUATION**
C                    ***KENNETH SWITZER   1968***
C     ***************************************************************
C     **      THIS PROGRAM WILL FIND THE OPTIMUM PARAMETERS SUBJECT TO **
C     ** A SPECIFIED DISTANCE FUNCTION AND INPUT.  THE INPUT MUST HAVE**
C     ** A CONSTANT FINAL VALUE AFTER SOME TIME T(1) AND NEVER EXCEED **
C     ** THAT FINAL VALUE.  THE DISTANCE FUNCTION IS THE SQUARE ROOT  **
C     ** OF THE SUM OF THE SQUARES OF THE SETTLING TIME T(S)  AND THE **
C     ** VALUE OF THE INTEGRAL FROM ZERO TO T(S) OF SOME FORM OF THE  **
C     ** ERROR BETWEEN THE INPUT AND THE OUTPUT.  T(S) IS DEFINED AS  **
C     ** THE TIME REQUIRED FOR THE RESPONSE TO DECREASE TO AND STAY   **
C     ** WITHIN .05  OF THE FINAL VALUE.  IF IERR IS 1 EER IS EQUAL TO**
C     ** ERR. IF IERR IS 2 EER IS ERR SQUARED.  IF IERR IS 3 EER IS   **
C     ** T TIMES ERR SQUARED.  THE APPROPRIATE FUNCTION SUBROUTINES   **
C     ** CAN BE CHANGE TO SUIT THE USERS NEEDS.  EULERS METHOD IS USED**
C     ** TO SOLVE THE DIFFERENTIAL EQUATION.  THE PROGRAM GOES THROUGH**
C     ** TWO PHASES TO INSURE A SOLUTION TO T(S) IN PHASE 3.          **
C     **       INPUT                                                 **
C     ** THE ONLY VALUE REQUIRED IS JJ.  ALL OTHER VALUES WILL DEFAULT**
C     ** THE INPUT IS OF THE FORM JJ,KJ,IERR,AA,DELTX.  THIS IS READ  **
C     ** ON A 3I1,2F5.0 FORMAT.  IF JJ IS 9 THE PROGRAM WILL TERMINATE**
C     ** DEFAULT VALUES IF THE LOCATIONS ARE LEFT BLANK ARE AS FOLLOWS**
C     ** KJ=0    IERR=1     AA=25.    DELTX=.05.   KJ=1 IS A CODE USED IF**
C     ** THE STARTING POINT IS TO BE SPECIFIED.  TWO MORE CARDS ARE   **
C     ** READ IN WITH A 8F10.0 FORMAT.  THE FIRST CARD MUST CONTAIN   **
C     ** THE PARAMETER VALUES A(I) IN ASCENDING ORDER.  THE SECOND    **
C     ** CARD MUST CONTAIN THE INITIAL STEPSIZE IN A SIMILAR MANNER.  **
C     ***************************************************************
C
C     **                        PROGRAM VARIABLES                    **
C     ** AA       INTEGRATION TIME       DELT(I) STEPSIZE            **
C     ** JJ       ORDER OF DIFF EQ       KJ      CODE FOR DATA READ IN **
C     ** A(I)     PARAMETERS OF DIFFERENTIAL EQUATION                **
C     ** ALOW     ALLOWABLE DEVIATION FROM INPUT FOR SETTLING TIME   **
C     ** IERR     CODE FOR RRR FUNCTION SUBROUTINE                   **
C     ** LLL      CODE FOR WHICH PHASE IS USING SUBROUTINE C         **
C     ** LKK      CODE FOR THE STATUS OF SETTLING TIME               **
C     ** KKK      COUNTER FOR MAXIMUM NUMBER OF DISTANCE FN EVALUATIONS**
C     ** DELTX    TIME INCREMENT  THIS IS REDEFINED LATER FOR PHASE 3 **
C     ** DIST(I) THE FOUR STORED DISTANCE FUNCTION POINTS             **
C     ** D(I)     USED IN TERMINATION OF PHASES 2  AND 3             **
C     ***************************************************************
```

```
C     **       MAIN OR CALLING PROGRAM                                  **
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION Y(10),D(10),A(10),DELT(10),DIST(5)
      COMMON AA,ALOW,DELTX,D,DE,IERR,LKK,LLL,J,JJ,JJJ,KKK,K,I
C     **    INPUT AREA  AND DEFAULT CHECK                               **
   80 READ98,JJ,KJ,IERR,AA,DELTX
      PRINT105
      PRINT 99,JJ,IERR
   99 FORMAT(2I5)
   98 FORMAT(3I1,2X,2F5.0)
      IF(JJ)80,80,81
   81 IF(JJ-9)83,82,82
   82 STOP
   83 IF(IERR)84,84,85
   84 IERR=1
   85 IF(AA)88,88,89
   88 AA=25.
   89 IF(DELTX)92,92,87
   92 DELTX=.05
   87 DO 809 I=1,JJ
      A(I)=3.
      DELT(I)=.1
  809 D(I)=0.
      K=4
      ALOW=.05
      LLL=1
      LKK=1
      KKK=0
  105 FORMAT(1H1)
      DO 749 II=1,4
  749 DIST(II)=0.
      A(JJ)=1.
      IF(KJ)802,802,803
  803 READ100,(A(I),I=1,JJ)
      READ100,(DELT(I),I=1,JJ)
  100 FORMAT(8F10.5)
      GO TO 1
  802 J=JJ+1
      JJJ=JJ-1
      CALL PHAS1(A,DIST,DELT)
      DO 801I=1,J
  801 DELT(I)=.1
      CALL PHAS2(A,DIST,DELT)
      PRINT 300
  300 FORMAT(/,2X,31HSWITCH TO NEW DISTANCE FUNCTION,//)
C     ** RESET DELTX FOR PHASE 3                                        **
      DELTX=.01
    1 LLL=2
      CALL PHAS3(A,DIST,DELT)
      LLL=3
      DE=0.
C     ** CALL FOR LISTING OF RESPONSE VALUES                            **
      CALL C(A,DIST,DELT)
      GO TO 80
      END
```

```
      FUNCTION AINPUT(Z)
      IMPLICIT REAL*8(A-H,O-Z)
C     **  THIS FUNCTION SUBROUTINE SPECIFIES THE INPUT AS A FUNCTION  **
C     **     OF Z WHICH REPRESENTS TIME IN SECONDS                    **
      IF(Z-1.)1,2,2
    1 AINPUT=Z
      RETURN
    2 AINPUT=1.
      RETURN
      END



      FUNCTION DISTFN(AREA,T)
      IMPLICIT REAL*8(A-H,O-Z)
C     ** THIS FUNCTION SUBROUTINE SPECIFIES THE DISTANCE FUNCTION USED**
C     **     IN PHASE 3 IN TERMS OF SETTLING TIME T AND AREA          **
      DISTFN=DSQRT(AREA*AREA+T*T)
      RETURN
      END



      FUNCTION RRR(ERR,Z)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION Y(10),D(10),A(10),DELT(10),DIST(5)
      COMMON AA,ALOW,DELTX,D,DF,IERR,LKK,LLL,J,JJ,JJJ,KKK,K,I
C     ** THIS FUNCTION SUBROUTINE KEYS OFF IERR AND LLL TO SPECIFY THE**
C     **      VALUE OF RRR AND IN TURN EER.                          **
C     ** INTEGRAL OF ABSOLUTE VALUE OF ERROR     IAE                 **
C     ** INTEGRAL OF ERROR SQUARED                IES
C     ** INTEGRAL OF TIME MULTIPLIED ERROR SQUARED   ITES
      GO TO (1,2,2),LLL
    1 RRR=ERR
      RETURN
    2 GO TO (3,4,5),IERR
    3 RRR=ERR
      RETURN
    4 RRR=ERR*ERR
      RETURN
    5 RRR=ERR*ERR*Z
      RETURN
      END
```

```
      SUBROUTINE PHAS1(A,DIST,DELT)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION A(10),CODE(10),D(10),DELT(10),DIST(5),Y(10)
      COMMON AA,ALOW,DELTX,D,DF,IFRR,LKK,LLL,J,JJ,JJJ,KKK,K,I
C     **   PHASE 1 USES A STEEPEST DESCENT SEARCH TECHNIQUE                *
C     ** COUNTERS    KJJ  MAXIMUM ITERATIONS ALONG LINE  KJ MAX RUNS       *
  900 KJJ=0
      KJ=0
      I=1
C     ** EVALUATION OF H1 H2 H3                                             *
    1 CALL C(A,DIST,DELT)
      CC=DABS(DELT(I))/10.
      A(I)=A(I)+CC
      CALL C(A,DIST,DELT)
      A(I)=A(I)-2.*CC
      CALL C(A,DIST,DELT)
      A(I)=A(I)+CC
      H1=DIST(K)
      H3=DIST(K-1)
      H2=DIST(K-2)
      F=H1-H2
      G=H2-H3
      H=F-G
   99 FORMAT(I5)
      PRINT99,I
      PRINT71,A(I)
      PRINT72,H1,H2,H3
   71 FORMAT(5H A(I),E9.6)
   72 FORMAT(1X,2HH1,F12.8,2X,2HH2,F12.8,2X,2HH3,F12.8)
      IF(F)40,98,50
   50 IF(G)51,98,55
   51 DELT(I)=DELT(I)/10.
      GO TO 1
   40 IF(G)55,98,42
   42 IF(H1-H3)47,47,48
   47 A(I)=A(I)+CC
      GO TO 1
   48 A(I)=A(I)-CC
      GO TO 1
C     ** EVALUATION OF DELT(I)
   55 DELT(I)=F/DABS(DELT(I))
      PRINT80,DELT(I)
C     ** CONSTRAIN DELT(I)
      IF(DABS(DELT(I))-.38)61,61,60
   60 DELT(I)=.38*(F/DABS(F))
      GO TO 300
   61 IF(DABS(DELT(I))-.005)62,300,300
   62 DELT(I)=.005*(F/DABS(F))
  300 PRINT80,DELT(I)
      I=I+1
```

```
      IF(I-JJ)1,70,70
   70 I=1
C     ** INCREMENT ON STRAIGHT LINE
  172 DO 171 II=1,JJJ
  171 A(II)=A(II)+DELT(II)
      CALL C(A,DIST,DELT)
    8 PRINT80,DIST(K),(A(II),II=1,JJ)
   80 FORMAT(8F15.8)
      KJJ=KJJ+1
      IF(KJJ-15)284,273,273
  284 IF(DIST(K)-DIST(K-1))172,273,273
  273 KJJ=0
      DO 75 II=1,JJJ
   75 A(II)=A(II)-DELT(II)
C     ** TERMINATION COUNTER
      KJ=KJ+1
      IF(KJ-3)1,74,74
   74 PRINT81
   81 FORMAT(2X,6HSWITCH)
      RETURN
   98 STOP
      END
```

```
      SUBROUTINE PHAS2(A,DIST,DELT)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION DIST(5),DELT(10),A(10),D(10)
      COMMON AA,ALOW,DELTX,D,DE,IERR,LKK,LLL,J,JJ,JJJ,KKK,K,I
C     **    PHASE 2 USES RELAXATION FOR OPTIMIZING THE PARAMETERS      **
C     **  SEE PHASE THREE FOR AN EXPLANATION OF SOME OF THE VARIABLES **
      I=1
      DALOW=.0001
      DTERM=.00005
      DMIN=.000001
      CODE=1.
      M=1
      ICONT=0
    1 ICONT=ICONT+1
      IF(ICONT-16)3,30,30
    3 CALL C(A,DIST,DELT)
      A(I)=A(I)+DELT(I)*CODE/10.
      CALL C(A,DIST,DELT)
      A(I)=A(I)-DELT(I)*CODE/10.
  102 FORMAT(2X,2(F16.9,2X),5HCODE ,F3.0,I5,2X,F16.9,2X,F16.9)
   96 PRINT102,DIST(K-1),DIST(K),CODE,I,A(I),DELT(I)
      GO TO (10,20),M
   10 IF(DIST(K-1)-DIST(K))11,12,12
   11 CODE=CODE*(-1.)
   12 IF(DABS(DIST(K-1)-DIST(K))-DMIN)30,13,13
   13 M=2
      A(I)=A(I)+DELT(I)*CODE
      GO TO 1
   20 IF(DIST(K-1)-DIST(K))21,26,26
   21 DELT(I)=DELT(I)/10.
      IF(DIST(K-3)-DIST(K-1))23,22,22
   22 A(I)=A(I)-DELT(I)*CODE*3.
      GO TO 24
   23 A(I)=A(I)-DELT(I)*CODE*7.
   24 M=1
      IF(DELT(I)-DALOW)30,30,25
   25 IF(DIST(K)-DIST(K-1)-DMIN)30,30,1
   26 A(I)=A(I)+DELT(I)*CODE
      IF(DELT(I)-DALOW)30,27,27
   27 IF(DIST(K-1)-DIST(K)-DMIN)30,30,1
```

```
 90 DELT(I)=DELT(I)*10.
    ICONT=0
    PRINT101,DELT(I),(A(II),II=1,JJ)
101 FORMAT(2X,F16.6,6F12.6)
    DELT(I)=.1
    D(I)=DIST(K-1)
    I=I+1
    IF(I-JJ+1)41,41,40
 40 I=1
 41 M=1
    DO 2 II=1,JJ
  2 DELT(II)=.1
    SUM=0.
    DO 42 II=2,JJJ
 42 SUM=SUM+DABS(D(II-1)-D(II))
    IF(SUM-DTERM)43,1,1
 43 RETURN
 91 ML=ML+1
    IF(ML-9)1,92,92
 92 DELT(I)=DELT(I)*10.
    ML=0
    GO TO 1
    END
```

```
      SUBROUTINE PHAS3(A,DIST,DELT)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION DIST(5),DELT(10),A(10),D(10)
      COMMON AA,ALOW,DELTX,D,DF,IERR,LKK,LLL,J,JJ,JJJ,KKK,K,I
C     **   PHASE 3 USES A MODIFED RELAXATION METHOD                    **
      I=1
C     ** DALOW   ALLOWABLE MINIMUM STEPSIZE                           **
C     ** DTERM PHASE TERMINATION VALUE                                **
C     ** DMIN PARAMETER TERMINATION VALUE                             **
      DALOW=.0001
      DTERM=.00005
      DMIN=.000001
      P1=0.
      P2=0.
      ML=0
      CODE=1.
      M=1
C     ** COUNTER FOR MAXIMUM NUMBER OF INCREMENTS
      ICONT=0
    1 ICONT=ICONT+1
      IF(ICONT-16)3,30,30
    3 CALL C(A,DIST,DELT)
C     ** STORE PREVIOUS VALUE OF A(I)
      P1=P2
      P2=A(I)
      A(I)=A(I)+DELT(I)*CODE/10.
      CALL C(A,DIST,DELT)
      A(I)=A(I)-DELT(I)*CODE/10.
  102 FORMAT(2X,2(F16.9,2X),5HCODE ,F3.0,I5,2X,F16.9,2X,F16.9)
   96 PRINT102,DIST(K-1),DIST(K),CODE,I,A(I),DELT(I)
      GO TO (10,20),M
C     ** TEST FOR SLOPE
   10 IF(DIST(K-1)-DIST(K))11,12,12
   11 CODE=CODE*(-1.)
   12 IF(DABS(DIST(K-1)-DIST(K))-DMIN)30,13,13
   13 M=2
      ML=0
      A(I)=A(I)+DELT(I)*CODE
      GO TO 1
C     ** TEST FOR CASE 3A THROUGH 3D
   20 IF(DIST(K-1)-DIST(K))21,26,26
   21 DELT(I)=DELT(I)/10.
      IF(DIST(K-2)-DIST(K-1))23,24,24
   23 A(I)=A(I)-DELT(I)*CODE*10.
   24 M=1
      IF(DELT(I)-DALOW)30,30,25
```

```
   25 IF(DIST(K)-DIST(K-1)-DMIN)30,30,1
   26 A(I)=A(I)+DELT(I)*CODE
      IF(DIST(K-2)-DIST(K-1))31,31,32
   31 A(I)=A(I)-DELT(I)*CODE*2.
      DELT(I)=DELT(I)/10.
      M=1
   32 IF(DELT(I)-DALOW)30,27,27
   27 IF(DIST(K-1)-DIST(K)-DMIN)30,30,91
C     ** TERMINATE PARAMETER
   30 DELT(I)=DELT(I)*10.
      PRINT101,DELT(I),(A(II),II=1,JJ)
  101 FORMAT(2X,F16.6,6F12.6)
      ICONT=0
C     ** THIS CHECKS FOR A HIGH LAST VALUE OF THE DISTANCE FUNCTION    **
C     ** IF THIS IS SO D(I) IS SET EQUAL TO THE PREVIOUS VALUE
      IF(DIST(K-2)-DIST(K-1))50,51,51
   50 DIST(K-1)=DIST(K-2)
    52 A(I)=P1                    IF(P1-.8)51,51,52
   51 D(I)=DIST(K-1)
      I=I+1
      IF(I-JJ+1)41,41,40
   40 I=1
   41 M=1
C     ** PHASE TERMINATION TEST
      SUM=0.
      DO 42 II=2,JJJ
   42 SUM=SUM+DABS(D(II-1)-D(II))
      IF(SUM-DTERM)43,1,1
   43 PRINT 110
  110 FORMAT(2X,20HPARAMETERS OPTIMIZED)
      PRINT 101,CODE,(A(I),I=1,JJ)
      RETURN
C     ** THIS ROUTINE WILL INCREASE DELT(I) IF IT IS TOO SMALL         **
   91 ML=ML+1
      IF(ML-11)1,92,92
   92 DELT(I)=DELT(I)*10.
      ML=0
      GO TO 1
      END
```

```
      SUBROUTINE C(A,DIST,DELT)
      IMPLICIT REAL*8(A-H,O-Z)
      DIMENSION Y(10),D(10),A(10),DELT(10),DIST(5)
      COMMON AA,ALOW,DELTX,D,DF,IFRR,LKK,LLL,J,JJ,JJJ,KKK,K,I
      LKK=1
    1 AREA=0.
C     ** COUNTING LOOP FOR TERMINATION AFTER C IS CALLED 400 TIMES     **
      KKK=KKK+1
  816 IF(KKK-400)91,91,89
   89 PRINT911,KKK
  911 FORMAT(2X,25HPROGRAM TERMINATED AFTER,I4,6H LOOPS)
      STOP
C     **              EULERS METHOD
   91 Z=DELTX
C     ** X        REPRESENTS THE INPUT F(T)
      X=DELTX
      Y(J)=DELTX
      DO 925 II=1,JJ
  925 Y(II)=0.
C     ** Y(I)     I-1 DERIVATIVE OF THE DIFFERENTIAL EQUATION
    2 DO 926 II=1,JJ
  926 Y(II)=Y(II)+Y(II+1)*DELTX
      S=0.
      KL=J
      DO 927 II=1,JJ
      KL=KL-1
  927 S=S+Y(II)*A(KL)
      Y(J)=X-S
C     ** ERR      ABSOLUTE VALUE OF ERROR BETWEEN INPUT AND OUTPUT     **
      ERR=DABS(Y(1)-X)
C     ** EER      MODIFIED ERR USED IN CALCULATING AREA.  SEE RRR SUB  **
      EER=RRR(ERR,Z)
C     ** AREA     AREA UNDER CURVE DEFINED IN RRR FUNCTION SUBROUTINE  **
   76 AREA=AREA+EER*DELTX
      Z=Z+DELTX
      X=AINPUT(Z)
C     ** LLL      1 PHASE 1 OR 2.  2 PHASE 3.  3  PRINT OUT OF RESPONSE**
      GO TO (210,220,400),LLL
  220 GO TO (221,222),LKK
  221 IF(ERR-ALOW)230,230,223
C     ** T        SETTLING TIME T(S)
```

```
  230 T=Z
C     ** AREA2   STORES AREA UP TO TIME T IN CASE T IS T(S)
      AREA2=AREA
      LMM=0
      LKK=2
      GO TO 228
  222 IF(ERR-ALOW)231,231,223
  231 IF(DABS(Y(2))-.00105)232,232,228
  232 LMM=LMM+1
      IF(LMM-800)2,2,235
  235 AREA=AREA2
      GO TO 241
  223 LKK=1
  228 IF(Z-35.)2,2,234
  234 T=25.
C     ** EVALUATION OF DIATANCE FUNCTION FOR PHASE 3                    **
  241 DIS=DISTFN(AREA,T)
  127 FORMAT(2X,5HAREA ,F15.6,2X,2HT ,F10.6,2X,          F16.9)
      PRINT127,AREA,T,A(I)
      GO TO 847
  210 IF(Z-AA )2,2,4
C     ** EVALUATION OF DIATANCE FUNCTION FOR PHASES 1 AND 2             **
    4 DIS=AREA
  847 DO 867 II=2,4
  867 DIST(II-1)=DIST(II)
      DIST(4)=DIS
      RETURN
C     ** LISTING OF FINAL RESPONSE FOR OPTIMAL PARAMETERS               **
  400 DE=DE+DELTX
      IF(DE-1.)2,401,401
  401 DE=0.
      PRINT500,Z,AREA,ERR,Y(1)
  500 FORMAT(6(2X,F16.9))
      IF(20.-Z)600,2,2
  600 RETURN
      END
/*
//GO.SYSIN      DD   *
3 1  30.    .04
/*
```

ACKNOWLEDGEMENTS

I wish to acknowledge the counsel and support of my major adviser Prof. W. W. Koepsel, the suggestions and criticism of Prof. D. H. Lenhert, The assistance of D. R. Bentrup, and the patience of my wife. I also wish to thank Kansas State University and the Electrical Engineering Department for the computer time necessary to complete this report.

Kenneth Switzer

PARAMETER OPTIMIZATION OF A THIRD-ORDER DIFFERENTIAL EQUATION

by

KENNETH WAYNE SWITZER

B. S., Kansas State University, 1966

AN ABSTRACT OF A MASTER'S REPORT

submitted in partial fulfillment of the

requirements for the degree

MASTER OF SCIENCE

Department of Electrical Engineering

KANSAS STATE UNIVERSITY
Manhattan, Kansas

1968

Traditionally the design of linear control systems has been based on trial and error methods. In recent years considerable interest has developed in using computers as a design tool to simulate the control system and find the most desirable system parameters. This is done by defining performance measures in terms of a distance function that relates quality of performance to the least distance from the origin.

This paper presents a FORTRAN computer program that finds the control system parameters that minimize a distance function for a specified topological configuration and input f(t). The input f(t) must reach a constant final value after some time T_1 and never exceed that value. The system must be described by an N^{th} order (2 N 8) linear differential equation with constant coefficients.

The optimal parameters are found for a third order system to demonstrate the program. A modified step input is used with a performance measure that involves settling time and the integral of several forms of the error between the input and the output. The results are compared and analyzed.

The choice of the distance function is purely subjective and must be based on good engineering judgment. On this basis it is desirable that the distance function be chosen so that the solution space is continuous. This simplifies the search procedure and reduces the computation time required.