AUTOMATIC ITERATIVE OPERATION ON AN ANALOG COMPUTER

by

George Hannauer

INTRODUCTION

Recent developments in analog computers have aroused fresh interest in automatic iterative computation. While the technique is not new, it has not been widely used with analog computers in the past, and is therefore unfamiliar to many analog programmers. It is the purpose of this report to provide an introduction to the techniques involved and to suggest representative applications.

While the analog computer is, in many ways, well-suited for this mode of operation, many of the basic concepts (logical decision, storage, discrete variables) are more familiar to users of digital computers. Iterative analog computation is, in fact, a form of hybrid computation, employing both analog and digital techniques, and large and complex iterative problems are most effectively attached by combining digital devices (flip-flops, shift registers, gates, etc.) with conventional analog components. Nevertheless, a good many problems can be solved with analog components alone, and this report will deal almost exclusively with this type of problem. Specifically, the report will be concerned with what can be done with existing components on a medium sized general purpose analog computer such as the PACE® TR-48.

Printed in U.S.A. 103

Bulletin Number ALAC6302-1ab

TABLE OF CONTENTS

AUTOMATIC ITERATIVE OPERATION ON AN ANALOG COMPUTER

by

George Hannauer

1. DEFINITION OF ITERATIVE OPERATION

By iterative operation of an analog computer, I will mean any sequence of operations that involves: (1) automatic cycling of the computer mode so that the computer goes through a number of OPERATE cycles, and (2) automatically changing some condition of the problem — an input, a parameter value, or an initial condition — between successive OPERATE cycles so that a slightly different problem is solved with each iteration.

The second condition is important. It distinguishes true iterative operation from repetitive operation, which satisfies only the first condition. Repetitive operation, in fact, is not a computational technique at all, but merely a form of readout, or display. No special techniques are involved in programming a rep-op problem; in fact, programming a problem is the same for rep-op as for "real time" readout, with the exception that the programmer must restrict himself to high-speed components (e.g. quarter-square multipliers instead of servos).

In repetitive operation, much of the information generated is redundant; except for occasional manual parameter changes, the same problem is solved again and again, and the same curve generated repeatedly. In contrast, each OPERATE cycle in iterative computation investigates a slightly different set of parameter values, and produces slightly different results. In a given number of cycles, therefore, iterative operation generates considerably more information, utilizing more fully and efficiently the analog computer's ability to produce a large amount of data in a short period of time.

The word "automatic" is also an important distinction in the above definition. In a loose sense, almost all operations of an analog computer may be classed as iterative, since rarely if ever is an operator satisfied with a single run. More typically, he will make many runs, varying parameter values from one run to the next, and observing the effect on the solution. What distinguishes iterative computation, as defined above, is the possibility of automating this procedure.

2. TYPICAL APPLICATIONS

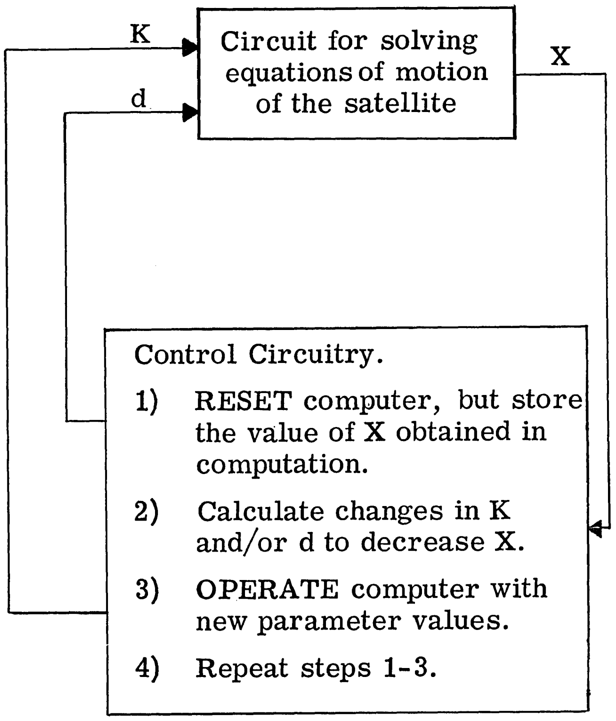
a) Parameter Sweep. For a simple example of automatic iterative operation, consider the effect of an adjustable parameter α on an analog computer solution. For example, we might want to study the steady-state error of a position servo as a function of controller gain or some other control parameter. Or we might be interested in the miss distance of a missile as a function of initial launching error. In either case, we are interested in a single number, X, (steady-state error, miss distance, etc.) which is computed during an OPERATE cycle using a particular value of some parameter α (controller gain, launching error, etc.). A value of α is chosen at the beginning of the OPERATE cycle, and the corresponding value of X is computed (usually X will be a final value, i.e. the output of some component at the end of the OPERATE cycle). It is desired to obtain a plot of X versus α.

Obtaining such a plot manually can be quite time-consuming. The operator has to choose several values of α, changing pots manually between runs and tabulating the result X after each run. He can then make a point-plot of X versus α and connect the points by a smooth curve. The procedure outlined below replaces this tedious work with a single computer run, in which the desired graph is automatically produced by an X-Y plotter. The necessary circuit is described in Appendix 2.

b) Optimization. Optimization problems arise quite frequently in design studies. In many cases, a long sequence of computer runs is made, with parameter adjustments between runs to achieve some optimum condition such as maximum profit, minimum cost, minimum error, etc.

For example, consider the design of a control system to stabilize the orientation of a satellite. Suppose the system is characterized by two parameters: a gain K and a dead-zone d. The problem is to correct a given orientation error with minimum fuel consumption. The general approach is given in Figure 2-1.

The computer is placed in the OPERATE mode for a sufficient length of time to correct the initial error, and the amount of fuel used is "remembered" by a storage device. The values of X from several successive OPERATE cycles may be stored if needed. On the basis of these


K
d
Circuit for solving equations of motion of the satellite
X
Control Circuitry.
1) RESET computer, but store the value of X obtained in computation.
2) Calculate changes in K and/or d to decrease X.
3) OPERATE computer with new parameter values.
4) Repeat steps 1-3.

d = width of deadzone in error sensing device

K = controller gain

X = amount of fuel used in the correcting maneuver

Figure 2-1

results, changes are made in d or K or both, and the problem is re-run. The process is continued until a minimum is reached.

This type of problem differs from parameter sweep in that the former is purely "open-loop" i.e. the changes in parameter values are determined in advance, while in optimization problems the parameter changes are allowed to depend on the results of previous runs. The circuitry for performing the required storage, parameter changes, etc. will be discussed in Sections 3 and 4.

c) Boundary-Value Problems. The analog computer is essentially an initial-value device, that is, from the many possible solutions to a given set of differential equations, a single one is determined by specifying initial values for the variables, i.e. values at time zero. In contrast, many problems are stated in terms of boundary values, which must be met by trial and error, i.e. by some sort of iteration process.

The most common examples of boundary value problems arise in the solution of partial differential equations which are discussed below. However, occasionally one runs into the same problem with ordinary differential equations, as in the following example:

A bullet is fired from a cannon at an inclination θ from the horizontal, and with a muzzle velocity V_0. In the absence of air resistance, the trajectory of the bullet is described by the equations:

$$\dot{x} = V_0 \cos\theta \qquad x(o) = 0 \tag{2-1}$$

$$\ddot{y} = -g \qquad y(o) = 0;\ \dot{y}(o) = V_0 \sin\theta \tag{2-2}$$

where x and y are the horizontal and vertical coordinates of the bullet respectively, as in Figure 2-2.

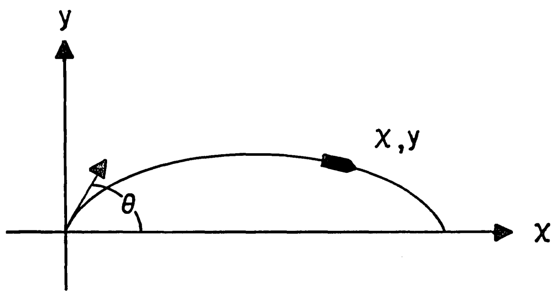


Figure 2-2

The problem is to find the firing angle θ that will make the bullet strike a target at ground level, at a distance R from the point of firing. That is, we are interested in the value of x(t) when y(t) becomes zero. Figure 2-3 gives a general flow diagram for solving this type of problem. Sections 3 and 4 contain descriptions of the necessary circuits.

d) Curve-Fitting. In many applications, a mathematical model with free parameters must be fitted to empirical data, i.e. a theoretical curve must be passed through a number of experimental points. As an example, suppose bench-scale experiments provide data on the decomposition of a component in a chemical reaction. Chemical theory and/or experience suggest that the decay should satisfy the differential equation

$$\dot{C} = -kC^n$$

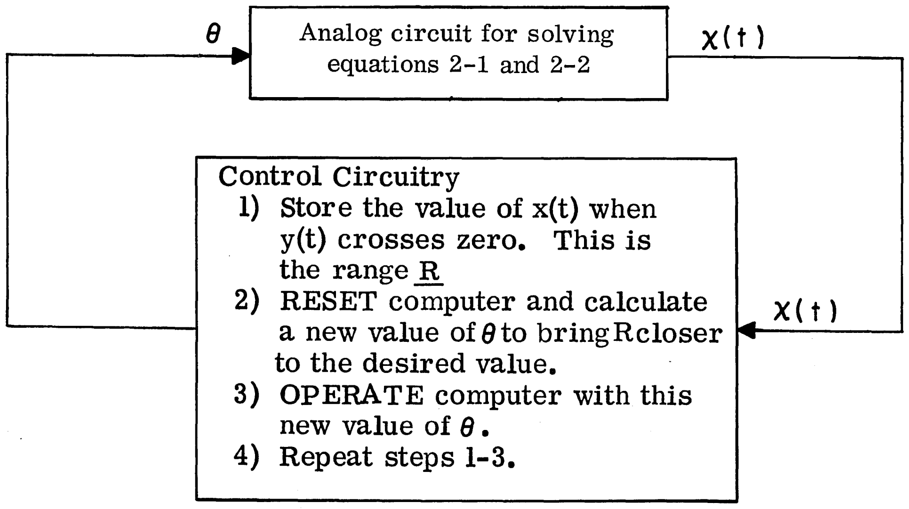
where C (t) is the concentration of the chemical at time t. The problem is to find numerical values of k and n so as to provide the closest agreement between the theoretical curve and the experimental data.

Let C (t) be the solution to the equation (with given values of k and n) and C*(t) be the experimental curve. Then, we would like to choose k and n so as to make C (t) and C* (t) as close as possible.

To automate this procedure, we need to define some measure of the error. If we define

$$\epsilon = \int \left[C(t) - C^*(t)\right]^2 dt$$

then we can program the computer to make ϵ as small as possible. Thus, curve-fitting problems can be treated as a special case of optimization problems.



Control Circuitry
1) Store the value of x(t) when y(t) crosses zero. This is the range R
2) RESET computer and calculate a new value of θ to bring R closer to the desired value.
3) OPERATE computer with this new value of θ.
4) Repeat steps 1-3.

Figure 2-3

e) Serial Solution of Partial Differential Equations. The solution of partial differential equations on a general-purpose analog computer by the usual finite-difference techniques involves a large number of similar circuits operating simultaneously. If the mesh size is small or the number of independent variables is large, then the duplication of equipment may make this approach prohibitively expensive. The simularity of circuits leads to the idea of using a single "time-shared" circuit, that is, a single circuit which performs sequentially the calculations that would otherwise be performed simultaneously by many similar circuits.

Consider the heat equation

$$\frac{\partial^2 T}{\partial x^2} = K \frac{\partial T}{\partial t}$$

with boundary conditions

$$\begin{aligned} T\,(o,\,t) &= f\,(t) \\ T\,(L,\,t) &= g\,(t) \\ T\,(x,\,o) &= h\,(x) \end{aligned}$$

This system describes the temperature T(x, t) at position x and time t in a thin, homogeneous rod of length L with an initial temperature distribution h(x) and time-varying temperatures f(t) and g(t) established at the ends.

The usual approach to analog solution of this problem is to treat x as a discrete variable, replace partial derivatives with respect to x by finite-difference approximations, and solve the resulting ordinary differential equations in the conventional manner.

For serial solution, the best approach is to treat t as a discrete variable and work with a system of ordinary differential equations in x. (If the reader doubts this, he may attempt serial solution with discrete x and continuous t and struggle with the boundary conditions.)

Replacing the time-derivatives with finite-difference approximations, we obtain the following equation:

$$T''\,(x,\,t) = K\,\frac{T\,(x,\,t) - T\,(x,\,t - \Delta\,t)}{\Delta\,t} \qquad 2.2$$

where T', T'' refer to derivatives with respect to x (not t). Since the integrators on an analog computer integrate with respect to time, we must represent distance in the heat equation by time on the machine. The circuit for solving equation 2-2 is given in Figure 2-4.

The fact that computer time represents distance in the original equation must be taken into account in interpreting the results.

The length of the OPERATE cycle is proportional to the length of the rod. The graph produced by the computer during a single OPERATE cycle is an instantaneous temperature profile. Establishing boundary values T (o, t) and T (L, t) for T at both ends of the rod means establishing initial and final values for the output of integrator 2.

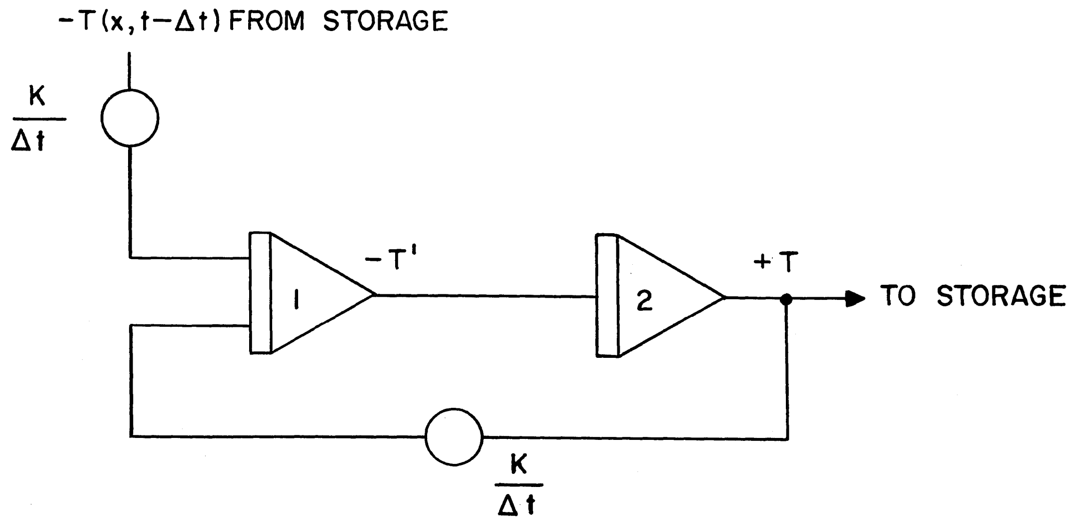


Figure 2-4. Circuit for Solving Equation 2-2

Establishing the initial value is no problem, since this means putting an initial condition directly on the integrator #2. To establish a final value, we must pick an appropriate initial condition for integrator #1. This requires an iterative procedure similar to the one used for the missile trajectory problem described above. The sequence of steps would be as follows:

1) Use the circuit in Figure 2-4 to calculate $T(x, \Delta t)$ with the given function $h(x) = T(x, o)$ as an input. Use $f(\Delta t) = T(o, \Delta t)$ as an initial condition on amplifier 2, and guess at an initial condition for amplifier 1.

2) At the end of the OPERATE cycle, compare the final value $T(L, \Delta t)$ with the desired value $g(\Delta t)$, and change the IC on amplifier #1 accordingly.

3) Iterate until the condition $T(L, \Delta t) = g(\Delta t)$ is met.

4) When this condition is met, store the complete curve $T(x, \Delta t)$ and repeat steps 1 - 3 with $T(x, \Delta t)$ as an input to generate $T(x, 2\Delta t)$.

Continuing in this manner, we obtain a sequence of temperature profiles $T(x, o)$ (given), $T(x, \Delta t)$, $T(x, 2\Delta t)$, $T(x, n\Delta t)$.

NOTE that this application involves an iteration <u>within</u> an iteration. Such computations are referred to as <u>nested</u> computations.

All of these problems, with the possible exception of the last, can be solved on the TR-48, provided the number of adjustable parameters involved is not too great. What distinguishes the last example is the need for curve storage — the entire curve $T(x, t)$ must be stored and used as an input to generate $T(x, t + \Delta t)$ in the next OPERATE cycle. In contrast, only a few numbers need be remembered at any one time to solve the other problems.

Curve storage is possible on the TR-48, but the equipment requirements are severe. At present, it is more practical to solve problems requiring curve storage on a larger machine, such as the 100 amplifier 231-R. A number of circuits are available, (References 2, 3, and 4).

In each of the preceding applications the conventional analog circuit that solves the dynamic equations of the basic problem is supplemented by a number of additional components to perform the iteration. These additional components must be capable of performing the following functions:

Storage: The computer must "remember" the results of previous computations, using them, if necessary, in the computation of new parameter values for successive runs.

Logic: The computer must decide which parameter(s) to alter, and must route signals into and out of storage in response to the results of past computations.

Automatic Parameter Adjustment: The computer must be able to change the values of important parameters and initial conditions.

Mode Control: The computer must be automatically cycled through a sequence of modes. Normally, the RESET and OPERATE modes will alternate, but more complicated sequences are sometimes desirable.

The ability of a computer to store information, make logical decisions, and alter its own program has long been a familiar concept to digital computer users. What is not so well known is that an analog computer can also perform these functions, as the following pages will indicate.

This section will deal with the above requirements and how they can be met on an analog computer. The circuits given below are not necessarily the best or most economical. They have been chosen with one overriding consideration in mind — they can be mechanized by ordinary general-purpose analog components, without special-purpose external equipment.

a) Storage. Since an analog signal is a voltage, it may be stored on a charged capacitor. Readout of the stored information must take place without drawing appreciable current from the capacitor, as drawing current will partially discharge it, resulting in an erroneous reading. This requirement for "nondestructive readout" implies that an amplifier will be necessary.

Consider the circuit in Figure 3-1 in which an amplifier and a capacitor are combined. Were it not for the presence of the capacitor, this circuit would simply be an inverter, and we would have $V_{out} = -V_{in}$.

The effect of the capacitor is to make V_{out} lag slightly behind $-V_{in}$, but this lag will be negligible provided V_{in} is not changing too rapidly, and the capacitor is sufficiently small. (The transfer function of the circuit is

$$\frac{V_{out}}{V_{in}} = -\frac{1}{1 + RSC}$$

and this is nearly -1 as long as the time constant RC is small.)

If the relay is opened, the amplifier behaves like an integrator in the HOLD mode, and V_{out} remains constant at the value it had when the relay opened. The amplifier is said to track, or follow its input voltage when the relay is closed, and store, or hold its value when the relay is opened.

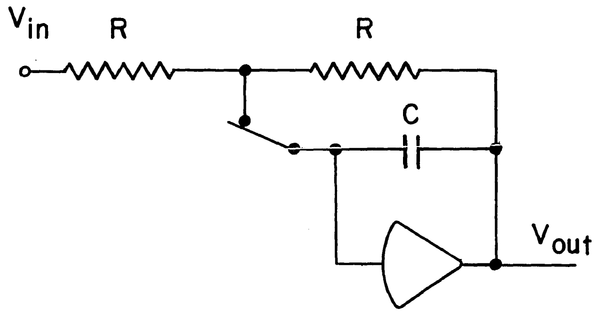
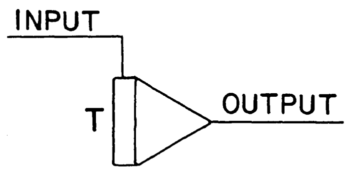


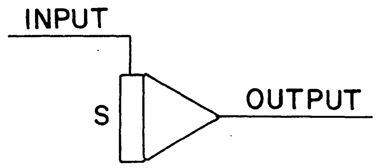
Figure 3-1

The circuit in Figure 3-1 is essentially the initial condition circuit of an integrator. If an input variable is patched into the IC terminal of an integrator, the integrator will track that variable in the RESET mode and store it in the HOLD mode (and even in the OPERATE mode if the integrator has no other inputs).

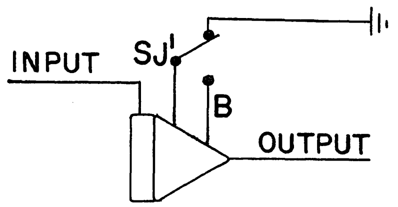
Since switching an integrator from tracking to storing is accomplished by changing its mode, it is necessary to be able to control the integrator modes individually. This can be done on the TR-48, as the integrator relay coils appear on the patch panel. A detailed description of TR-48 mode control is given in Appendix 1. The devices described below all use the track-store principle outlined above but they differ in their mode control.



TRACKING AMPLIFIER
Patch RESET coil to OPERATE bus
Patch OPERATE coil to RESET bus
Remove β plug
(Tracks when computer is in OPERATE; stores when computer is in RESET)



STORAGE AMPLIFIER
Patch OPERATE and RESET coils normally
Remove β plug
(Stores when computer is in OPERATE; tracks when computer is in RESET)



COMPARATOR CONTROLLED TRACK-STORE UNIT
Omit mode control connections
Connect SJ′ (<u>not</u> SJ) to B through comparator contacts
(Stores when comparator is open, tracks when comparator is closed)
Remove β plug

Notes:

1) The T and S amplifiers are named after what they do in OPERATE — T tracks and S stores. In RESET, they do the opposite.

2) Input resistor networks need not be patched.

3) Removal of β plug reduces the tracking error by shortening the time constant.

In order to store a result from a previous computation cycle we use a tracking amplifier and a storage amplifier in cascade, as in Figure 3-2. The problem variable X (t) is tracked during an OPERATE cycle, and its final value is stored between OPERATE cycles. The result is shown in Table 3-1; after every OPERATE cycle except the first, the output of amplifier S is the final value of X from the previous computer run. If it is desired to store a value of X other than a final value, the T amplifier can be replaced with a comparator-controlled track-store unit.

In OPERATE, T tracks and S stores. In RESET, their functions are reversed.

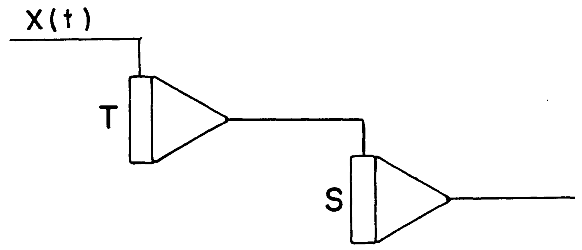


Figure 3-2

TABLE 3-1

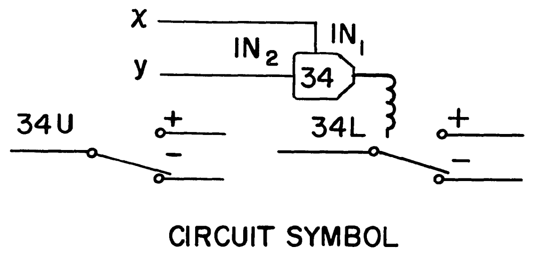
Computer Mode	Behavior of Amplifier T	Behavior of Amplifier S
R	Stores (output = O)	Tracks (output = O)
O	Tracks - X_1	Stores (output is still O)
R	Stores - X_1 (final value)	Tracks X_1 (final value)
O	Tracks - X_2	Stores X_1 (final value)
R	Stores - X_2 (final value)	Tracks X_2 (final value)
O	Tracks - X_3	Stores X_2 (final value)

Table 3-1 indicates the behavior of T and S amplifiers, where X_1, X_2, X_3 are the solutions produced during the 1st, 2nd, 3rd OPERATE cycle.

b) Logic. Logical decisions on the analog computer are usually made by relay comparators. The relay comparator consists of a high-gain amplifier whose output drives the coil of a DPDT relay. The relay contacts normally make to the side marked "+"; they transfer to the side marked "-" when the sum of the input voltages becomes negative. (See Figure 3-3.)

Let α be the parameter that we wish to maintain constant during a computer run and vary between runs by an amount $\Delta\alpha$. The value of $\Delta\alpha$, i.e. the magnitude and direction of the change in α, can be made to depend on the results of previous computations. Hence $\Delta\alpha$ will also be an amplifier output voltage. Methods of calculating $\Delta\alpha$ are discussed in Section 4.

The easiest way to change α in the desired manner is to use an integrator, as in Figure 3-4.



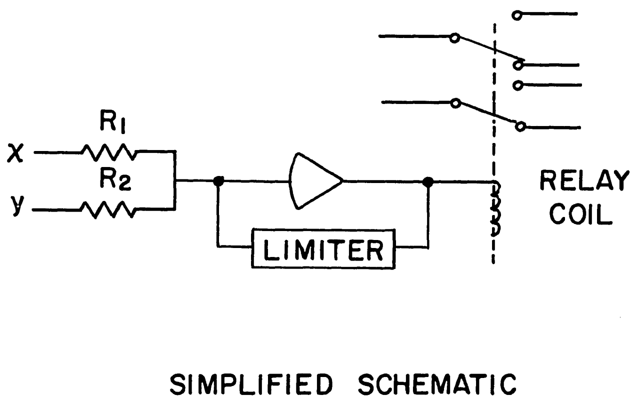


Figure 3-3. The Relay Comparator

It should be noted that the input resistors on the TR-48 comparator are not equal. Therefore a pot may be necessary to attenuate one of the inputs and assure that the switching occurs exactly when X + Y crosses zero.

The comparator is numbered with the number of the amplifier immediately to its left on the patch panel; this convention makes it easy to locate the comparator when patching. The letters "U" and "L" on circuit diagrams refer to the upper and lower sets of contacts respectively.

c) Automatic Parameter Adjustment. A parameter in an analog computer circuit is a number which is constant during any single OPERATE cycle, but which is allowed to change value between successive OPERATE cycles. Usually such parameters are represented by pot-settings, and the pots are adjusted manually between computations. However, in automatic iterative computation, the parameter value is changed by the computer itself, which means that the parameter must be regarded as a computer variable, that is, as an amplifier output voltage.

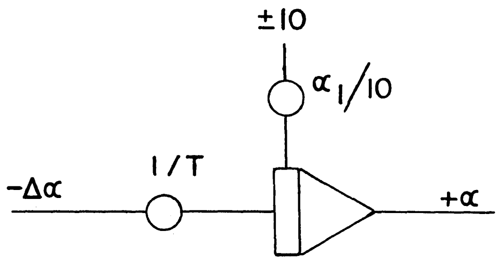


Figure 3-4

Since α is to be constant during a computer run, the integrator must be in the HOLD mode when the rest of the computer is in the OPERATE mode. Since α is supposed to change between computer runs, the integrator must be in the OPERATE mode when the rest of the computer is in RESET and the problem is being reset for the next run. This type of mode behavior can be achieved on a TR-48 by patching the OPERATE coil to the RESET bus. (See Appendix 1 for an explanation of TR-48 mode control.)

If the computer resets for T seconds between OPERATE cycles, the integrator in Figure 3-4 will integrate the constant $-\Delta\alpha/T$ for T seconds, and its output will change by an amount $\Delta\alpha$ for the next computation.

An "initial condition" must be chosen for α, that is a value for the first run. This is, of course, an initial condition on the integrator. The mode control must be arranged so that the integrator is in the RESET mode at the beginning of the iteration. A method of accomplishing this is given in Section 3d.

The three most common uses of the signal α are:

1) as an initial condition

2) as an additive constant, or bias voltage

3) as a coefficient

In cases 1) and 2) the voltage α may be fed directly to the problem, as a voltage input is what is required. In the third case, we have α as a voltage, but we need a coefficient. This means that a quarter-square multiplier must be used, as shown below.

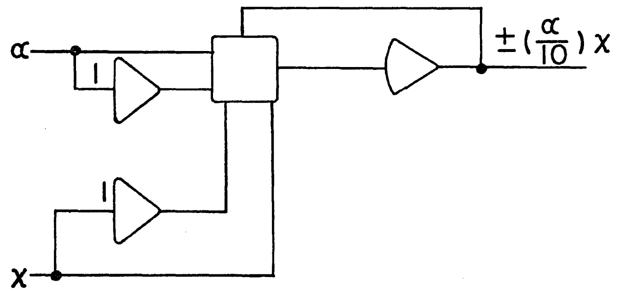


Figure 3-5

The net result is that an input is multiplied by a constant coefficient $\alpha/10$ which is a number between zero and one (in magnitude). This circuit is therefore equivalent to feeding x through a pot, except that we "reset the pot" automatically by changing the value of α.

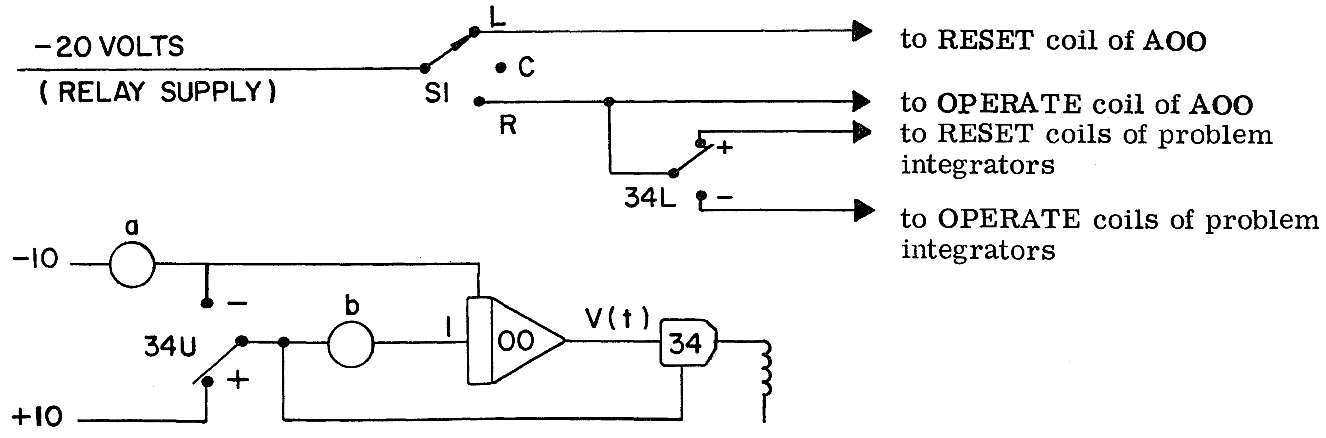
d) Mode Control. For the purposes of iterative computation, it is necessary to cycle the computer mode automatically between RESET and OPERATE. The repetitive operation unit does exactly this, and can be used in some forms of iterative computation. However, its use suffers from a number of limitations:

1) For some purposes it is too fast — the longest OPERATE cycle obtainable is 200 milliseconds, which makes readout on an X-Y plotter impractical. Also, many programs rely heavily on relay comparators to make the necessary logical decisions, and a 200 millisecond OPERATE cycle followed by a 10 millisecond RESET cycle does not always allow these relays sufficient throwing time.

2) When using the rep-op drive, there is some uncertainty about the first and last cycles. Switching transients may destroy the accuracy of the first computation. In "open-loop" iteration, (parameter sweep) this is not a serious defect, but in "closed loop" iteration, in which the parameter values for a given OPERATE cycle depend on the stored results from previous cycles, the effect can be serious.

3) It is desirable to have a means of cycling the computer mode on machines not equipped with rep-op.

The above circuit meets all these requirements. The RESET and OPERATE times are individually adjustable over a wide range. The iteration is started and stopped by a manual switch. This switch allows the entire iteration to be switched through RESET, HOLD, and OPERATE modes.



Finally, the circuit, which involves only one integrator, two pots, and a comparator, can be used on machines not equipped for repetitive operation.

When using this form of relay switching, the individual integrator coils should be patched directly to the switch and relay contacts. The regular mode busses are not used.

Switch S1 is initially in the LEFT position putting A00 in RESET and all integrators which are patched to the relay contacts in HOLD.

Putting switch S1 to the CENTER position puts every integrator in HOLD. When S1 is thrown to the RIGHT position, A00 integrates and produces an asymmetrical sawtooth, as shown in Figure 3-6. A00 starts from an initial condition + 10a and integrates down at the rate of -10b volts per second. When it reaches -10 volts, comparator 34 switches, and A00 integrates up at the rate of + 10 ab volts per second until it reaches + 10a, when comparator 34 switches again. The integrators patched to the comparator contacts are cycled between RESET and OPERATE.

The output of A00 varies from + 10a to -10 volts. The total voltage variation is 10a + 10 or 10 (a+1) volts. With Comparator 34 in the "+" position (which puts the computer on RESET) the rate of integration is -10b volts per second, and hence if T_R is the length of the RESET cycle we have $10b\ T_R = 10\ (a+1)$ or

$$T_R = \frac{a+1}{b}$$

Similarly, the integration rate of A00 is + 10ab volts per second when comparator 34 is in the "-" position and the computer is in OPERATE. We therefore have $10ab\ T_O = 10\ (a+1)$ or

$$T_O = \frac{a+1}{ab}$$

. . .where T_O is the length of the OPERATE cycle.

Solving for a and b, we get

$$a = T_R/T_O$$

$$b = \frac{T_R+T_O}{T_R\ T_O}$$

If we want a ten-second OPERATE cycle followed by a one-second RESET cycle, then we must have a = 0.100 and b = 1.100. Since a pot-setting greater than 1 is impossible, we use a gain of 10 on the integrator and set the "b" pot for b/10 = 0.110.

The operational procedure is as follows: put the manual switch to the left, and select the HOLD mode on the pushbutton mode-control panel. We may regard the L, C, and R switch positions as representing RESET, HOLD, and OPERATE cycles for the entire iteration; RESET and OPERATE for the individual runs are obtained by the switching of comparator 34.

In section 3C an integrator was used for automatic parameter adjustment. This integrator must cycle between HOLD and OPERATE during the iteration but should be reset at the beginning of the iteration. This object can be achieved by patching its OPERATE coil to the + contacts and its RESET coil to the left contact on the manual switch. The integrator will then be reset with A00.

Since the iteration commences with a RESET cycle, the parameter α will be updated by an amount $\Delta\alpha$ before the first run. If α is the desired value of α for the first run the IC on the integrator should be $\alpha_1 - \Delta\alpha$.

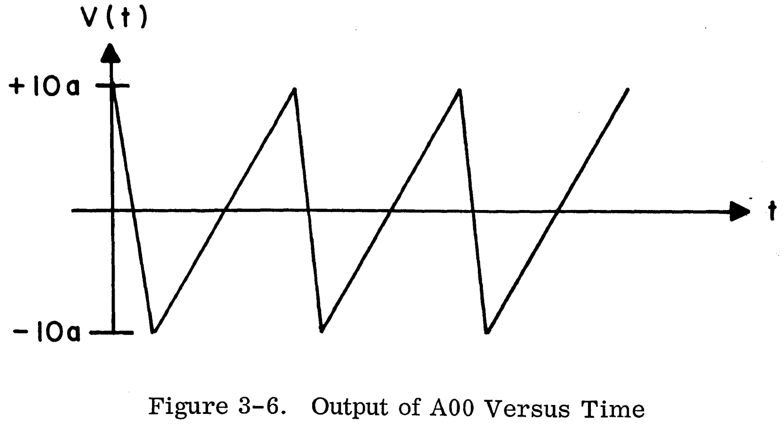


Figure 3-6. Output of A00 Versus Time

4. ITERATION SCHEMES

In cases where we are varying a parameter α to meet a boundary value or a minimum condition, we must decide what change to make in α for the next run. A formula for calculating the parameter value for the next run from the results of previous runs is called an iteration scheme. Two iteration schemes will be considered here. For simplicity we will limit ourselves to the case of a single parameter α. Iteration with several parameters is considered in References 6 and 7.

a) Correction Proportional to Error. Suppose we have to vary a parameter or initial condition α to meet the boundary condition $X_f = a$, where X_f is the final value of a variable X, such as the horizontal displacement of the bullet in the missile trajectory problem or the temperature T in the heat equation.

If we define an error by $\epsilon = X_f - a$, then a reasonable change in α would be

$$\Delta\alpha = G\epsilon$$

where G is a constant. This scheme makes large corrections for large errors and small corrections for small ones, which is reasonable. If α eventually settles down to a steady value, then $\Delta\alpha$ will approach zero, and therefore ϵ will approach zero, which is what we want.

If ϵ is an increasing function of α, then the constant G should be negative so as to change α in the right direction to decrease ϵ. On the other hand, if ϵ is a decreasing function of α, then G should be positive. This means that the method is most useful in cases where we know a priori whether ϵ is increasing or decreasing with α. This is the case with the heat equation. Figure 4-1 shows how Figure 2-4 may be modified by adding storage and parameter change circuits to effect the iteration scheme. The parameter α to be varied is $\alpha = T'(0)$ and the computer effects the change in α by integrating the error ϵ during the RESET cycle, whose length is T_R.

In cases where ϵ is sometimes an increasing function of α and sometimes decreasing, the above scheme is less practical. In these cases an alternative is to minimize the absolute error $|\epsilon|$. This approach reduces the boundary value problem to a problem in optimization.

b) Fixed - Step - Size Optimization. Consider the problem of choosing a value for the parameter α to minimize the result X. X may be the absolute error in a boundary value problem, as described in Section 4a above, it may be the error in a curve fitting problem, as in Section 2d, or the amount of fuel consumed during a satellite maneuver, as in Section 2b. Many iteration schemes are possible, but the following one is probably the simplest to mechanize. (In the following paragraph, subscripts refer to computer runs. Thus α_1 is the value of α used in the first run, X_1 is the result obtained from the first run, etc.)

1) Pick a starting value for α, say α_1

2) Make a run with this value and store the result X_1

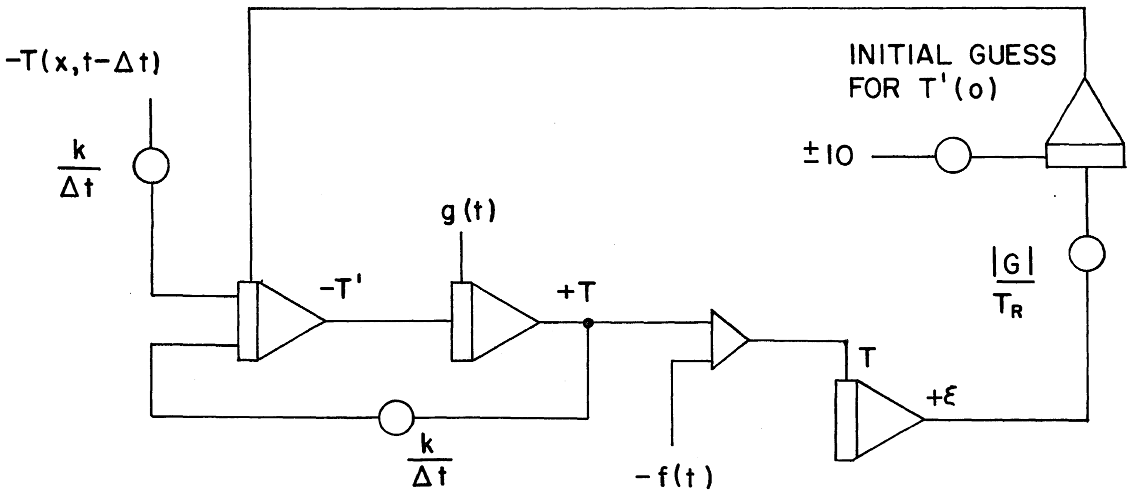


Figure 4-1. Iterative Circuit for Solving the Boundary-Value Problem in Section 3e, Using the Iteration Scheme $\Delta\alpha = G\epsilon$

3) Change α by a fixed amount $\Delta\alpha$ and make a new run with the value $\alpha_2 = \alpha_1 + \Delta\alpha$. Store the result X_2.

4) Compare the values X_1 and X_2. If $X_2 < X_1$, increase α by the same amount $\Delta\alpha$ and repeat.

5) Continue as long as X decreases with each successive change in α.

6) When finally an increase in X is obtained, this is an indication that the minimum value has been overshot. When this happens, cut the step-size $\Delta\alpha$ in half, and reverse its sign.

The following table gives an indication of what to expect when this procedure is followed. In this table, X, α, and $\Delta\alpha$ are all computer variables and are given in volts.

Run #	α	X	$\Delta\alpha$ for next run
1	1.00	+ 9.82	+ 1.00
2	2.00	+ 7.17	+ 1.00
3	3.00	+ 6.61	+ 1.00
4	4.00	+ 6.29	+ 1.00
5	5.00	+ 6.73	- 0.50
6	4.50	+ 6.10	- 0.50
7	4.00	+ 6.29	+ 0.25
8	4.25	+ 6.07	+ 0.25
9	4.50	+ 6.10	- 0.125
10	4.375	+ 6.02	etc.---

Note that some of the runs are redundant; run 7 duplicates run 4 and run 9 duplicates run 6, but this duplication is not a serious drawback. To mechanize the reversal of sign, we can use the following circuit:

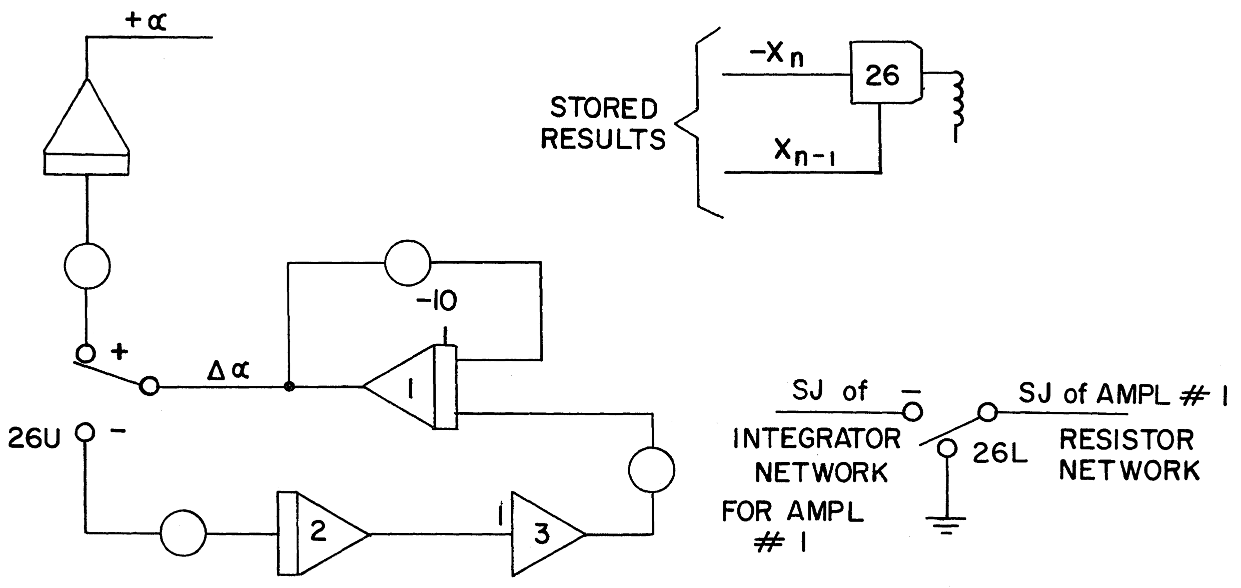

+α
STORED RESULTS
$-X_n$
26
X_{n-1}
-10
+
Δα
1
26U
-
2
3
SJ of INTEGRATOR NETWORK FOR AMPL # 1
SJ of AMPL # 1 RESISTOR NETWORK
26L

Figure 4-2

The mode control relays on integrators 1 and 2 are patched so that they integrate in RESET and hold in OPERATE. The comparator contacts assure that the integrators will receive inputs only when $X_n > X_{n-1}$; that is, only when the last parameter change made things worse instead of better. When $X_n > X_{n-1}$, the integrators will integrate for one RESET cycle, and then HOLD.

The system in Figure 4-2 produces a damped sinewave output. The pot-settings can be chosen so that the half-period of this damped sinewave is equal to the RESET time, and the amplitude decays by a factor of 1/2 in this time. At the end of the RESET cycle, the integrators hold the new value of $\Delta \alpha$, which is - 1/2 times the previous value. Hence, in a given RESET cycle, one of two things will happen: If $X_n < X_{n-1}$ the value of α will be updated by an amount $\Delta\alpha$. If $X_n > X_{n-1}$, the step-size $\Delta \alpha$ is halved in magnitude and reversed in direction.

Note that in this second case, the value of $\Delta \alpha$ is changed, but the value of α is not. Hence, the next run will be made with the same value of α, and should yield the same value of X.

The comparator will then be comparing two equal voltages, and it becomes a matter of chance whether it will decide to update α or reverse $\Delta \alpha$ during the next RESET cycle. To avoid this difficulty, the comparator is slightly biased so that when in doubt, it updates α.

5. PROGRAM FOR ITERATIVE SOLUTION OF A BOUNDARY-VALUE PROBLEM

In this section the trajectory problem given in section 2c is solved. The circuit is described in detail, and a typical set of results is given.

The dynamic equations of the bullet, which were given in Section 2c, are repeated here for reference:

$$\dot{x} = V_O \cos\theta \quad x(o) = 0 \qquad \text{Eq 5-1}$$

$$\ddot{y} = -g \qquad y(o) = 0;\ \dot{y}(o) = V_O \sin\theta \quad \text{Eq 5-2}$$

The solution of these equations on an analog computer is straightforward. Equation 5-1 requires one integrator and equation 5-2 requires two. An inspection of figure 5-1 indicates that an additional fifteen amplifiers and three multipliers are required to effect the iteration scheme. In view of the triviality of the differential equations (Eq. 5-1 and Eq. 5-2 may be solved analytically by direct integration) it appears ridiculous (and indeed it is) to use such a sophisticated and expensive iteration scheme. However, the circuit is intended as a general illustration, not as a solution of a specific practical problem. The following points should be noted:

1) The circuit was chosen to illustrate a number of interesting techniques. A number of modifications could have been made in the interest of equipment economy, but the present circuit illustrates the individual operations better.

2) The interest here is in the iteration scheme, not in the differential equations. Hence a trivial set of differential equations, which leads to an interesting and complex iteration scheme, was deliberately chosen. The dynamic equations could have been made considerably more complex by the addition of aerodynamic forces on the projectile. Some of these aerodynamic terms are quite complex and nonlinear, and would result in the bulk of the computing capacity being used to to solve the differential equations. The iteration scheme would, of course, remain the same. However, from the point of view of illustrating the iterative techniques nothing would be gained by making the differential equations more complex.

3) Most of the amplifiers (and two of the three multipliers) are performing essentially digital functions — storage, gating, logic, etc. The use of a few digital devices (AND gates, flip-flops, etc.) would materially simplify the program. Such components are available for larger computers, such as the 100-amplifier PACE 231-R.

4) In many practical cases, it is possible to use a far simpler iteration scheme than the one shown here. The iteration circuit in Figure 4-1, for solving the heat equation, should be regarded as more typical of a large number of one-point boundary-value problems. In this example also, the differential equation would be made more complex if appropriate non-linearities were introduced—such as a temperature-dependent thermal conductivity. Such a nonlinearity would result in the addition of a diode function generator and a multiplier to the circuit for solving the differential equation, but the iteration scheme would remain the same.

ANALYSIS OF THE CIRCUIT

The parameter to be controlled is the firing angle θ. However, the equations require $\sin\theta$ and $\cos\theta$. If θ were chosen as the iteration parameter, it would be necessary to use two DFG's to generate $\sin\theta$ and $\cos\theta$. The equipment requirement is reduced if we define $\alpha = \cos\theta$ and iterate on α. We can then obtain $\sin\theta$ from the relation $\sin\theta = \sqrt{1-\cos^2\theta}$, which holds when θ is in the first quadrant. This relation can be mechanized by a single amplifier and one multiplier (see amplifier 18, figure 5-1). This circuit makes use of the possibility of separating the TR-48 quarter-square multiplier into two independent X^2 generators. The iteration scheme used is proportional correction:

$$\Delta\alpha_{n+1} = \pm G\epsilon_n$$

where α_n and ϵ_n are the values of α and ϵ for the nth run,

$$\Delta\alpha_{n+1} = \alpha_{n+1} - \alpha_n \quad ✓$$

$$\Delta\epsilon_n = \epsilon_n - \epsilon_{n-1} \quad ✓$$

$$\Delta\alpha_n = \epsilon_n - \alpha_{n-1}$$

Nonsense

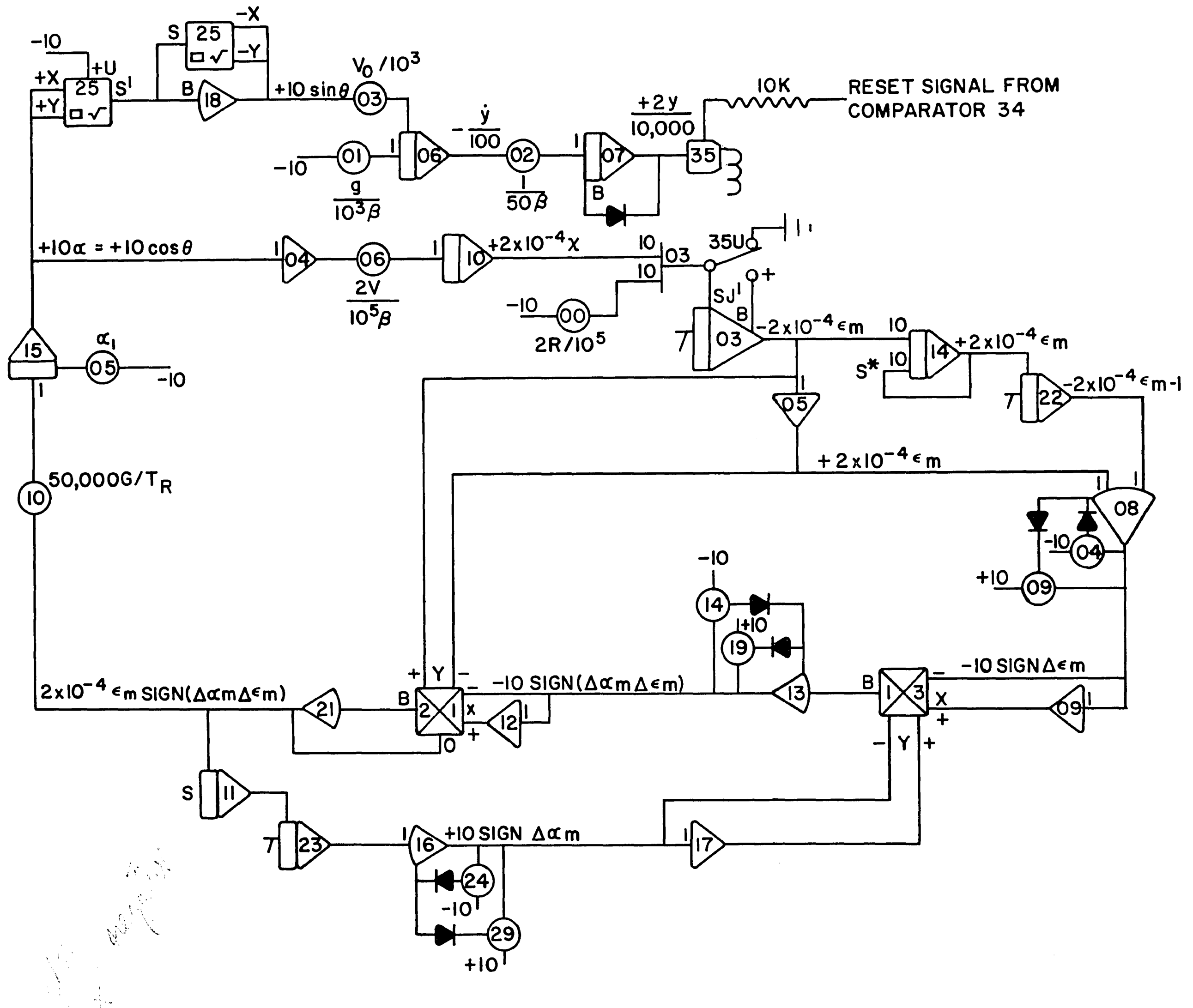


Figure 5-1. Program for Iterative Solution of Boundary-Value Problem

The use of the plus or minus sign is determined by the following rule: The signs of $\Delta\alpha_n$ and $\Delta\epsilon_n$ are compared; if they are alike, this is taken as an indication that $d\alpha/d\epsilon$ is positive, and the - sign is used. If they are unlike, the + sign is used.

Figure 5-1 gives the appropriate circuit. Amplifiers 06, 07, and 10 solve the dynamic equations. The value of x is to be sampled when y crosses zero. This will not happen at the end of an operate cycle, but at some time during the cycle. Hence a comparator-controlled track-store unit is used. When y crosses zero, comparator 35 flips to the minus side, switching amp 03 from TRACK to STORE. Amplifier 07 continues to integrate downward (this means a negative value of y, which has no physical significance). A feedback diode is used to prevent the output of amp 07 from becoming more negative than about half a volt. This is to prevent overload in case the OPERATE cycle continues for a long time beyond the collision point (the length of the OPERATE cycle is fixed).

Amplifier 03 is a combination summer and track-store unit (see Appendix 3). The desired value of the range R is set in on pot 00, and amplifier 03 tracks and stores the error. At some point in the OPERATE cycle amplifier 03 is switched from the TRACK mode to the STORE mode, and it should remain in this mode throughout the rest of the OPERATE cycle and the following RESET cycle. However, during this RESET cycle, the control signal, which is derived from amplifier 07, will be reset to zero, which will flip the comparator to "+" and return amplifier 03 to the TRACK mode prematurely. To prevent this, the second input to comparator 35 is connected to the RESET signal from comparator 34 (see figure 5-2) which is performing the mode cycling. When the main problem integrators 06, 07, and 10 are being reset, a -20 volt signal appears at this terminal, latching the comparator in the "-" state and assuring that amplifier 03 remains in the STORE mode throughout the RESET cycle. Since the -20 volt relay power supply is a larger signal than is normally used on the TR-48 patch panel, an input resistor of 10K is used to cut down comparator sensitivity. This resistor may be a "gain 10" resistor from any of the unused amplifier networks.

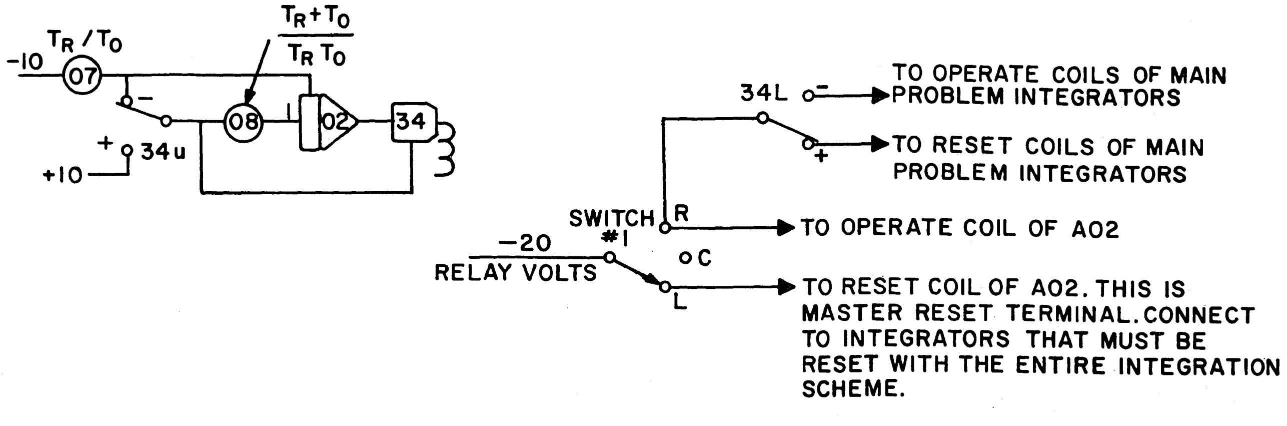
Amplifiers 14 and 22 delay the stored error by one cycle (see Table 3-1 and Figure 3-2). Amplifier 14 (marked S*) behaves like an ordinary S (Storage) amplifier, but its mode control is a bit different from that described in section 3a. It cycles between OPERATE and HOLD instead of OPERATE and RESET. An ordinary Storage amplifier, as described in section 3a, could have been used, but this trick enables the amplifier to share a dual integrator network with amplifier 15, reducing the number of integrator networks required.

Amplifier 08 is a high-gain amplifier. Instead of a feedback resistor, it has a feedback limiter circuit designed to limit it to approximately ±10 volts. The output of amp 08 may be thought of as essentially a digital signal, having the value -10 when $\epsilon_n > \epsilon_{n-1}$ and +10 when $\epsilon_n < \epsilon_{n+1}$, or in more concise terms, its output is -10 sign ($\Delta\epsilon_n$). Similarly, the output of amplifier 16 is +10 sign$\Delta\alpha_n$, which has been computed from the value of $\Delta\alpha$ for the last run, delayed one cycle. The signs of $\Delta\alpha_n$ and $\Delta\epsilon_n$ are compared by multiplier 13, which acts as a "coindidence gate" or "exclusive OR" circuit. The output of amplifier 13 is -10 volts if $\Delta\alpha_n$ and $\Delta\epsilon_n$ have the same sign, and +10 volts if they have opposite signs. Multiplier 21 multiplies this logic signal by ϵ_n, so that the output of amplifier 21 is proportional to $\pm\epsilon_n$, the sign depending on the relative signs of $\Delta\alpha_n$ and $\Delta\epsilon_n$. The signs on the multipliers are chosen so that the output is proportional to $-\Delta\alpha_{n+1}$. This signal is delayed by amplifiers 11 and 23 and converted to a binary signal by amplifier 16, so that it can be used to determine the direction of change for the next run.

The output of amplifier 21 is just what is needed for the iteration scheme $\Delta\alpha_{n+1} = \pm G\epsilon_n = -G\epsilon_n$ SIGN ($\Delta\alpha_n \Delta\epsilon_n$). Pot 10 introduces the constant G and the appropriate scale factor. Its output is $-10\,\Delta\alpha_{n+1}/T_R$. Amplifier 15, the parameter updating integrator, is connected so that it operates when the main problem is in RESET and holds when the main problem is in the OPERATE mode. It integrates the constant $-10\Delta\alpha_{n+1}/T_R$ for a constant interval T_R (the (RESET time) and its output changes by an amount $+10\Delta\alpha_{n+1}$ in this time. Thus its output, $+10\alpha = +10\cos\theta$ is appropriately updated during the RESET cycle and its new value is held during the following OPERATE cycle. Multiplier 25 and amplifier 18, as described above, provide the necessary squaring, subtraction, and square root functions to generate 10 sin θ from the input 10 cos θ.

Figure 5-2 gives the circuit for the mode control cycling. The operation of this circuit was analyzed in detail in section 3-d. Note that the MASTER RESET terminal on switch #1 may be patched to the RESET coil of the parameter-updating integrator 15, to enable a first-run value α_1 to be established on the integrator. The computer mode busses are completely unused, and all mode control signals are obtained from the manual switch and the comparator. The -20 volt relay power supply is terminated on the TR-48 patch panel in a terminal marked RV (relay voltage). This terminal is in the lower left corner of the readout panel.

Since the RESET and OPERATE busses are not used in this circuit, the operation of the circuit will be independent of the mode pushbuttons on the control panel, except for the POT-SET mode, which closes



-10
T_R/T_O
07
$\frac{T_R+T_O}{T_R T_O}$
08
02
34
+
34u
+10
34L
TO OPERATE COILS OF MAIN PROBLEM INTEGRATORS
TO RESET COILS OF MAIN PROBLEM INTEGRATORS
SWITCH #1
R
C
L
-20 RELAY VOLTS
TO OPERATE COIL OF A02
TO RESET COIL OF A02. THIS IS MASTER RESET TERMINAL. CONNECT TO INTEGRATORS THAT MUST BE RESET WITH THE ENTIRE INTEGRATION SCHEME.

Figure 5-2. Mode Control Circuitry for Boundary-Value Problem

a relay around every amplifier. The POT-SET mode is therefore a convenient one for setting all storage amplifiers to zero before the start of the iteration. The sequence of operations to be performed by the operator is as follows:

1) Put the computer in the POT-SET mode to clear all storage to zero.

2) Put SWITCH #1 to left position.

3) Depress the HOLD pushbutton to bring the computer out of the POT-SET mode.

4) Put SWITCH #1 to the right position.

The computer will now cycle back and forth between RESET and OPERATE, updating the value of the parameter α each time. The results may be recorded on an X-Y plotter, and the convergence observed.

Results for a typical series of runs are given in figure 5-3. The target was placed at 2,000 feet, and a first-run value of α was chosen that gives an entremely large initial overshoot. Note that the results of successive runs approach the target monotonically, and that the amount of the correction decreases as the target is approached.

It is worthwhile to pay some attention to the meaning of the constant G. The entire iteration scheme may be thought of as a feedback control system controlling the parameter α . The correction is proportional to the error ϵ , and the constant G may be thought of as the <u>gain</u> of the iteration schmeme. It is interesting to run the iteration scheme several times and observe the effect of the gain G on the convergence. As expected, low gain leads to slow convergence. Increasing the gain speeds the convergence, up to a point, but excessive gain produces divergence.

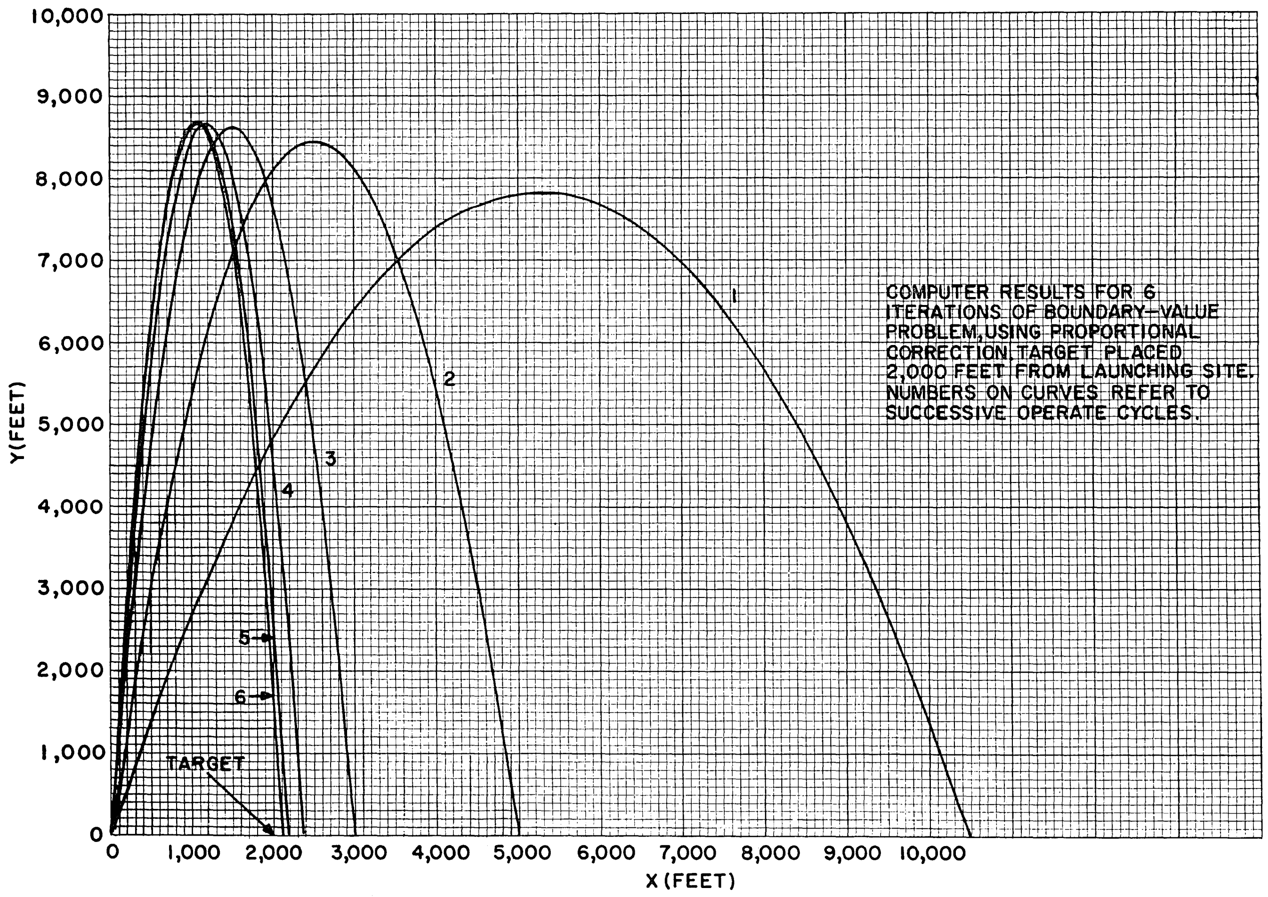


Figure 5-3. Results for a Typical Series of Runs

Table 5-1. Potentiometer Settings

Pot Number	Parameter Description	Numerical Setting
00	$2R/10^5$	0.040
01	$g/10^3/\beta$	0.161
02	$1/50\beta$	0.100
03	$V_o/10^3$	0.750
04	Limiter	0.470
05	α_1	0.100
06	$2V_o/10^5\beta$	0.075
07	T_R/T_{op}	0.500
08	$(T_R + T_{op})/T_R T_{op}$	0.300
09	Limiter	0.470
10	$50,000G/T_R$	0.040
14	Limiter	0.470
19	Limiter	0.470
24	Limiter	0.470
29	Limiter	0.470

Table 5-2. Summary of Integrator Mode Patching

02, 03	From SWITCH 1.
06, 07 10, 11	From lower contacts on comparator 34 RESET coils from + terminal OPERATE coils from - terminal
14, 15	RESET coil patched to SWITCH 1, left terminal.
	OPERATE coil from + terminal, comparator 35
21, 23	From lower contacts on comparator 34
	RESET coils from - terminal.
	OPERATE coils from + terminal. (The mode of these integrators is complementary to the mode of the main problem integrators)

APPENDIX 1

TR-48 MODE CONTROL

The mode control on the TR-48 is accomplished with relays which are energized from a -20 volt DC power supply. The individual relay coils appear on the patch panel, which allows for independent control of integrator mode.

The relays normally receive inputs from two mode control busses, called the RESET bus and OPERATE bus. Each bus is energized from the corresponding pushbutton on the mode control panel:

1) When the RESET mode is selected with a pushbutton, the RESET bus is energized with -20 volts and the OPERATE bus is de-energized.

2) When the OPERATE mode is selected, the OPERATE bus is energized, and the RESET bus de-energized.

3) In the REP-OP mode, the RESET and OPERATE busses are energized alternately, thus cycling the computer between RESET and OPERATE.

4) When the HOLD mode is selected, both busses are de-energized.

Each dual integrator network has two input terminals on the patch panel which go to relay coils. These terminals will be referred to as the RESET coil and the OPERATE coil. (This is an over simplification — actually, one of the terminals energizes two relay coils, but this is not important from a programmer's point of view):

1) When the RESET coil is energized, the integrator is in the RESET mode.

2) When the OPERATE coil is energized, the integrator is in the OPERATE mode.

3) When neither coil is energized, the integrator is in HOLD.

4) If both coils are energized simultaneously, the integrator may be in any of several modes. Because of this uncertainty, the programmer should avoid simultaneously energizing both coils.

In normal operation, each coil is patched to the corresponding bus. The mode of the integrator is then selected by the pushbuttons in the normal manner. This standard mode control connection can be made with a double horizontal bottle plug. If this plug is removed, the coils can be patched in many different ways. For example, reversing the connections (RESET coil to OPERATE bus and OPERATE coil to RESET bus) gives a tracking unit, as described in Section 3a. Integrator coils may also be energized directly from the -20 volt relay power supply or through comparators or manual switches. The relay power supply appears on the patch panel in the lower left-hand terminal of the readout panel, at the terminal marked RV (relay volts).

For some applications, it is desirable to have some integrators running in Rep-op and some in real time simultaneously. This result can be achieved on the TR-48 as follows: a shaded portion of the integrator network contains two holes marked NORMAL and two holes marked SPECIAL.

When the holes marked SPECIAL are patched together, the integrator operates normally. (When a bottle plug is used for this connection, the word SPECIAL is covered up. Hence the operator <u>sees</u> the word NORMAL when the integrator is operating normally). When the holes marked NORMAL are covered with a bottle plug and the word SPECIAL is exposed, the integrator operates in a special manner, i.e., it goes into the OPERATE mode as soon as the REP-OP mode button is depressed, and remains in OPERATE (with a real-time capacitor, not a rep-op capacitor) as long as the computer is in repetitive operation. An application of such an integrator is given in Appendix 2.

The mode control relays on the TR-48 are shared by the two integrators in a dual integrator network. Hence the two integrators in a dual network must be in the same mode at any given time. The programmer must keep this in mind when assigning numbers to various components. Occasionally an integrator network will be wasted (if an odd number of integrators with a special nonstandard mode logic are required, one extra network will have this same logic, whether it is needed or not). However, the waste will never amount to more than a few integrator networks. Note that an S-amplifier (storage amplifier) has the same mode logic as a conventional integrator. Hence if three S-amplifiers are needed in an iteration, the odd network can be used in the problem itself.

One other feature of the TR-48 integrator network has been used in this article. This is the possibility of changing the feedback capacitor on an integrator to increase its integration rate. This feature allows a wider range of time — scale factors than would otherwise be the case. A bottle plug, called the β plug, is normally patched into each integrator. Removing this plug replaces the feedback capacitor by a capacitor of 1/10 the value, increasing the integration rate by a factor of 10. This feature operates independently of mode control patching, rep-op, NORMAL/SPECIAL connections or anything else. Whatever feedback capacitor the integrator would have with the β plug (a rep-op or "real-time" capacitor) is replaced by a capacitor of 1/10 the value. In contrast to the mode control connections, the dual integrator has two separate β plugs — one for each integrator in the network.

APPENDIX 2

PARAMETER SWEEP CIRCUIT

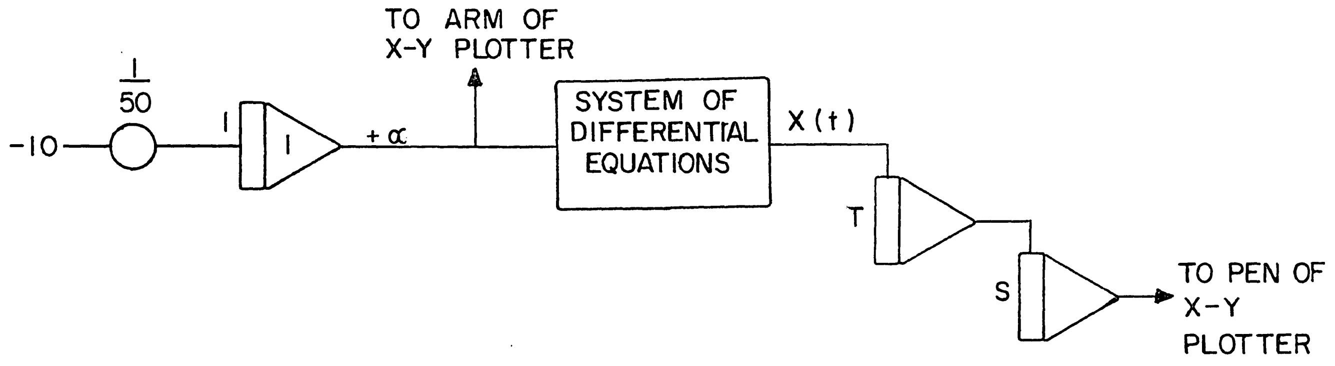


Figure A2-1

The NORMAL/SPECIAL option on the TR-48 integrator enables us to construct a very simple circuit for the parameter sweep problem in Section 3a. In Figure A2-1, integrator #1 is a SPECIAL integrator, while all the others, including the T and S (track and store) integrators are operating in rep-op.

After every rep-op cycle, the S-amplifier stores the final value of the variable X. This final value will change gradually as α changes. Since α is changing very slowly (it takes 50 seconds to go from zero to +10 volts) it will appear very nearly constant from the point of view of the rep-op circuit. In particular, if the rep-op rate is 20 solutions per second, α will change by only 0.01 volt during a single computation. This is the smallest change that can be detected on the TR-48.

Thus, the rep-op circuit will solve the differential equations of the system for a virtually constant value of α, and the S-amplifier will store the final value X. After the next solution, the slightly different value of α will cause a small change in the value of X, and the output of the S-amplifier will change accordingly. With the α - integrator connected to the arm and the S-amplifier connected to the pen, the X-Y plotter will produce a graph of the final value X versus the parameter α.

The graph should look something like Figure A2-2, that is, the graph will not be smooth, but will consist of many small steps. However, if α varies over the desired range in 50 seconds, and the rep-op rate is 20 solutions per second, the graph should consist of 1,000 steps, and they should be so small as to be barely noticeable.

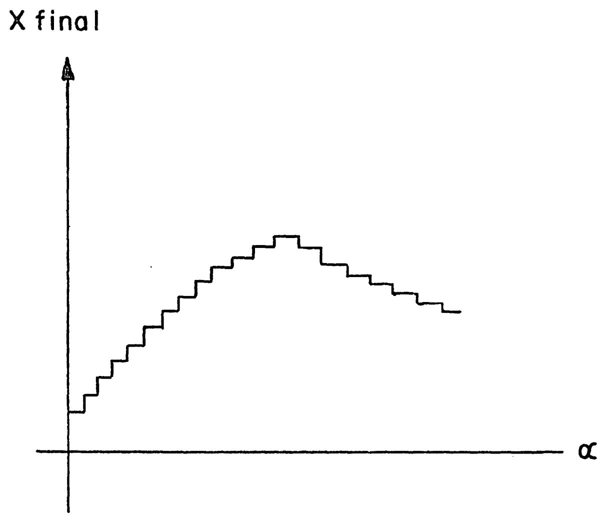


Figure A2-2

APPENDIX 3

THE COMBINATION SUMMER AND TRACK-STORE UNIT

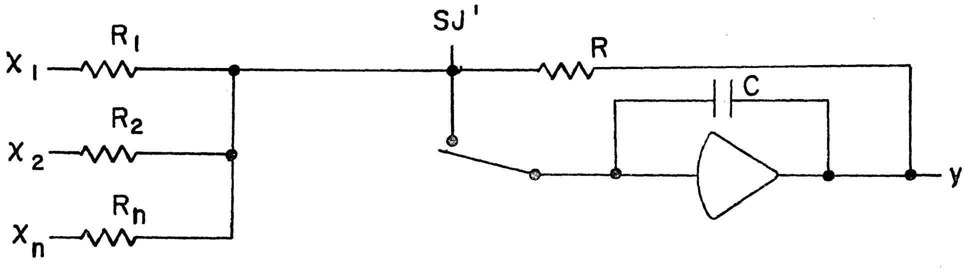


Figure A3-1

In many applications, it is necessary to store the sum of two or more signals. This can, of course, be accomplished with two amplifiers — a summer, and a track-store unit. However, the functions of both may be combined in a single track-store unit with several input resistors (Figure A3-1).

When the switch is closed, we have

$$y = -\frac{1}{1+RCS}\sum_{1}^{n}\frac{R}{R_i}x_i$$

. . .and the unit acts as a combination track-summer.

The point marked SJ′ in Figure A3-1 appears on the patch panel area of a TR-48 integrator network, so that additional IC input resistors can be patched in to take advantage of this capability. In many cases, this feature saves one or more amplifiers.

Since the input resistor network of a track-store integrator is not normally used, it can be patched to the SJ′ and used for multiple inputs. A symbol for this type of patching is given in Figure A3-2.

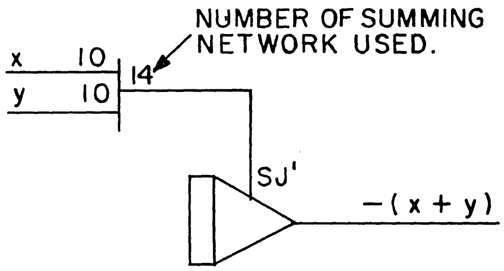


Figure A3-2

Note that a "gain of 10" on a TR-48 means a 10K input resistor, which produces an actual gain of one in this case, since the feedback resistor is also 10K. (Some TR-48's have 50K input and feedback resistors in the integrator IC networks. The individual computer should be checked for this before the circuit is used.)

REFERENCES

1) Wadel, Louis B. (Chance Vought Aircraft, Inc.). Automatic Iteration on an Electronic Analog Computer. Presented at Wescon, 1954.

2) Landauer, J. P. Automatic Storage PCC#130, July 1958.

3) Brunner, Walter. Boundary — Value Problems PCC#154, April, 1960.

4) Berthiaume, Paul. Analog Computer Storage Techniques and Applications PCC#170, 1961.

5) Rogers, A.E. and Connolly, T.W. Analog Computation in Engineering Design McGraw-Hill, New York, 1960.

6) Witsenhausen, Hans. Fixed Step-Size Optimization with Several Parameters PCC#169, August, 1961.

7) Witsenhausen, Hans. Hybrid Techniques Applied to Optimization Problems. Presented at Spring Joint Computer Conference, San Francisco, May, 1962.

References 2, 3, 4, and 6 are PCC Reports and may be obtained from Electronic Associates, Inc., Princeton Computation Center Box 582, Princeton, New Jersey.