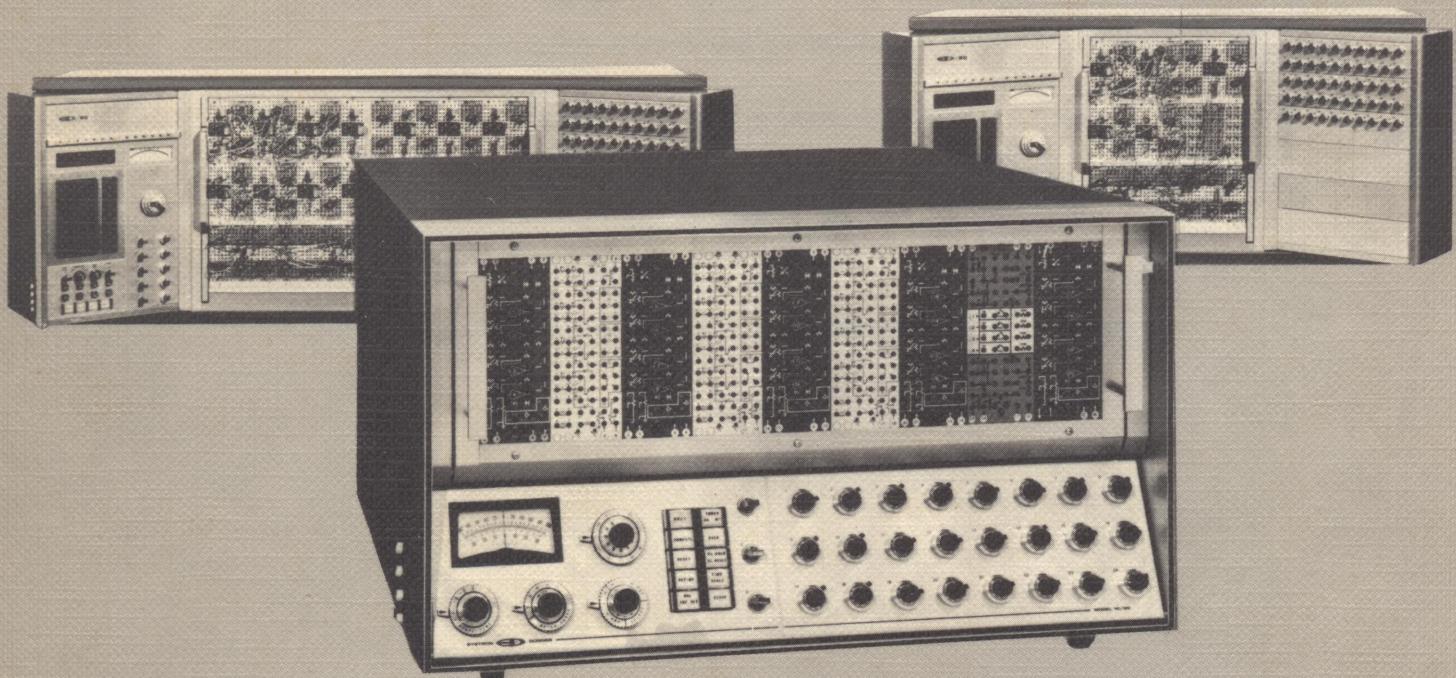


HANDBOOK
OF
A N A L O G
C O M P U T A T I O N



SYSTRON  DONNER
CORPORATION

The publication of this professionally prepared handbook could only be realized at a considerable investment in time and money. To defray some of the publication costs, a nominal charge of \$5.00 per copy is made. Of course, this handbook is supplied free of charge to all Systron-Donner Computer users and to persons who participate in a Systron-Donner Analog Computer Seminar.

To order your copy, please address your request to:

Analog Computer Operations Group

SYSTRON-DONNER CORPORATION
888 Galindo Street
Concord, California 94520

HANDBOOK OF

ANALOG COMPUTATION

(INCLUDING APPLICATION OF DIGITAL CONTROL LOGIC)

Prepared by

Maxwell C. Gilliland, Ph.D.
COMPUTER RESEARCH, INC.

and

Analog Computer Staff
SYSTRON-DONNER CORP.

JUNE 1967





Table of Contents

Chapter	Title
1	The Motivation for Analog Computers
2	Basic Analog Computing Elements
3	Elementary Analog Programming
4	Block Programming for Physical Systems
5	Scaling
6	Computer Operation
7	Logical Algebra
8	Basic Operation of Digital Logic Elements
9	Circuits for Simple Logical Functions
10	Circuits for Simple Linear and Non-Linear Functions
11	Simulation of Constant Coefficient Transfer Functions
12	Control System Simulation
13	Fundamentals of Vector Analysis
14	Partial Differential Equations – Part 1
15	Basic Iterative Programming
16	Sampled-Data System Simulation
17	Partial Differential Equations – Part II
18	Correlation Analysis
19	On-Line Data Analysis Programs
20	System Optimization
21	Medical Applications
22	A Practical Approach to Adaptive Control

APPENDIX

- I. Glossary of Abbreviations
- II. Uniform Graphics for Simulation



CHAPTER 1

THE MOTIVATION FOR ANALOG COMPUTERS

Analog Computation, based on the modern electronic analog computer, is of fairly recent date. The first commercially available general purpose electronic analog computers appeared on the market in the 1940's. These early machines were an outgrowth of an emerging electronics technology and a critical need for automatic computing machines that could solve complex dynamic problems. Slide Rule and manual equation solving could no longer be relied upon as a practical approach to seeking engineering solutions. Therefore, analog computers became important tools in the design of aircraft, jet engines, atomic reactors, oil refineries, chemical plants, etc.

Many types of analog computers have evolved over the years. The family has included the mechanical differential analyzer, electromechanical differential analyzer, and most recently, the iterative differential analyzer.

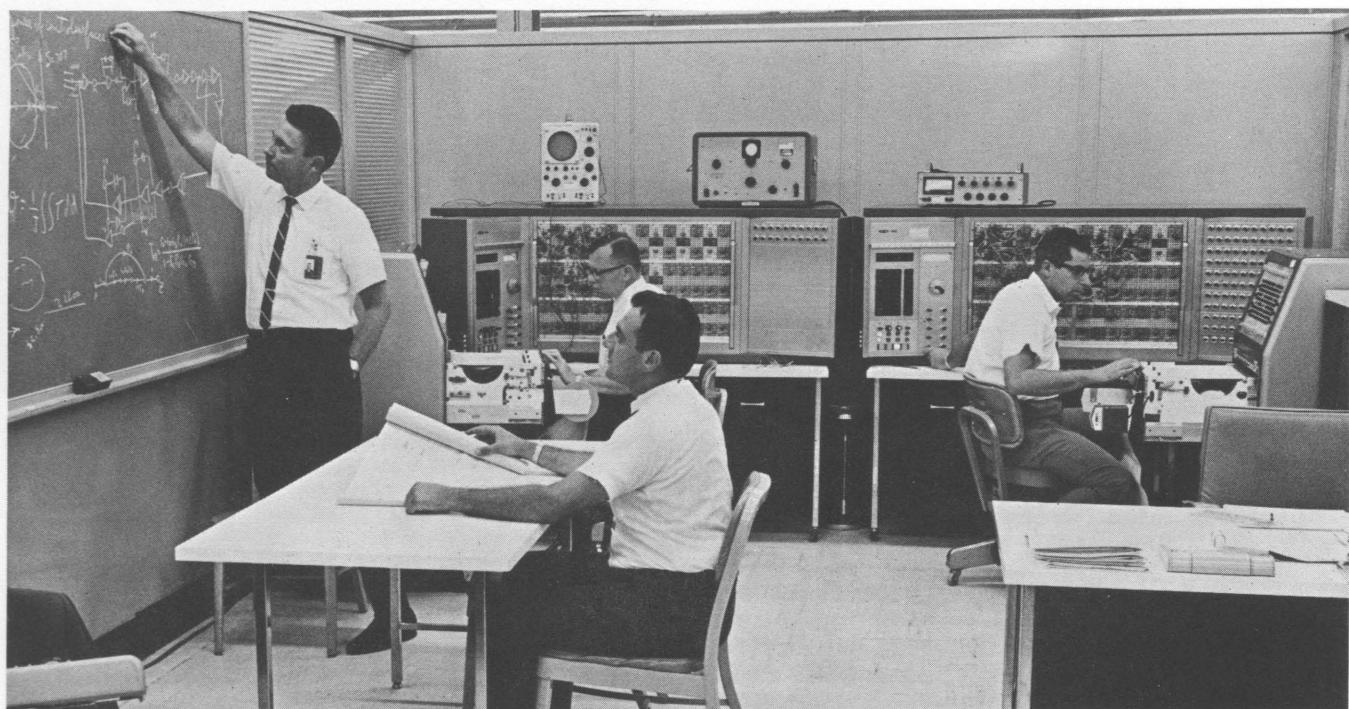
The analog computer has always had several advantages compared to a digital computer. These are primarily speed, more simulation capability per dollar, an ability to integrate, and an excellent man-machine interface.

The main feature of the analog computer is that it can integrate time-varying voltages. There is no easy way to differentiate. Consequently a mathematical model of a physical system which is expressed in terms of differential equations cannot be solved with the machine directly. It is necessary to reformulate the mathematical model in terms of integral equations, either implicitly or explicitly.

The analog computer can integrate only with respect to time. Thus, a mathematical model which contains partial integrals (corresponding to partial derivatives) with respect to several variables must be approximated by a set of ordinary integral equations with respect to time. Of course, computer time need not correspond to time in the physical world, although it usually does.

Before 1959 the analog computer was a synchronous machine; all its integrators operated in unison. In 1959 the DYSTAC¹ was introduced. The name was an acronym for 'dynamic storage analog computer'. This machine was the forerunner of the iterative differential

¹ DYSTAC is a registered trademark of CSI.



Two large SD 80 computers are used in this engineering laboratory to simulate the behavior of drone helicopters with different load configurations and various automatic flight control systems under a wide range of operating conditions. Simulation saved time and money, reduced the exposure of personnel and materiel to possible damage when actual flight tests were made, and it also eliminated the necessity of waiting for special environmental (weather) conditions.

(Photo courtesy, Gyrodyne Company of America, Inc.)



Analog computers have long served in the fields of chemistry and process control as a convenient, low cost means to observe, analyze, control, and predict the effect of varying parameters in a dynamic problem. Some of the problems may pertain to enzyme reaction, chemical kinetics, continuous distillation, heat transfer or transport delay -- just to mention a few basic applications of analog computers. The university student shown here is performing a research problem involving the effect of potential barriers on kinetic energy levels.

(Photo courtesy, University of California)

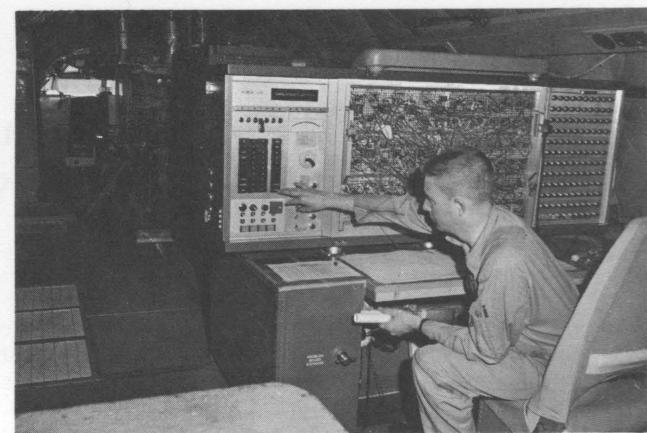
analyzer which appeared in 1960. The iterative differential analyzer is an asynchronous computer; the integrators need not be controlled in unison. They can operate independently either in groups or singly.

In 1962 the analog computer was augmented with digital logic. This innovation first appeared in a machine called the HYDAC.¹ The HYDAC had a very large quantity of synchronous digital logic. Since that time the inclusion of a smaller complement of asynchronous digital logic has become accepted practice.

Digital logic can be used for the implementation of logical decisions. These are based on results obtained from the analog portion of the computer during the solution of the problem. Digital logic also can be used for mode control of the analog computer. All of the Systron-Donner 10/20 and 40/80 series analog computers can be operated in the iterative mode and can be augmented with digital logic.

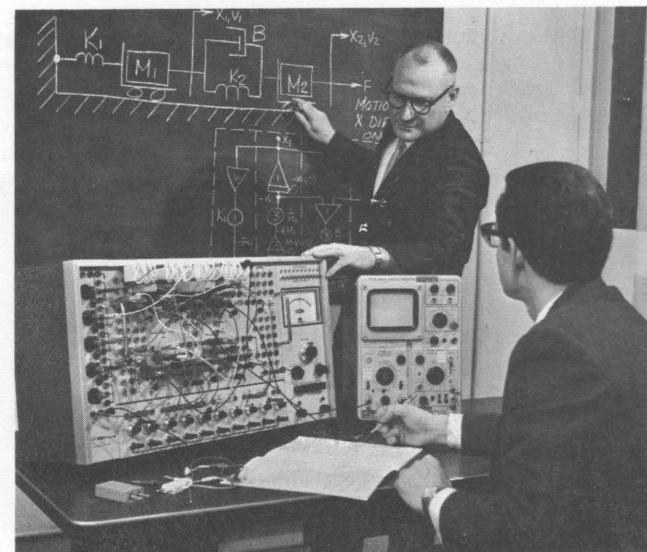
Analog computers have found widespread acceptance in virtually every area of scientific investigation. This growing interest in analog computers has created a need for complete software, specially designed for the beginner and less experienced user. Also, the recent addition of digital logic control has greatly improved the problem-solving capability of analog computers. How this new feature can be used in analog computation is thoroughly illustrated in this publication.

It is the purpose of this handbook to provide students as well as experienced computer users with comprehensive and up-to-date analog computer software. Chapters 2 to 5 develop the basic fundamentals of computer operation and illustrate the solution of elementary problems. Chapter 6 provides a detailed description of the operating controls and computer logic of the SD 10/20 and 40/80 series computers. This information serves as useful reference material to problem-solutions illustrated in the more advanced discussions



An SD 80 computer mounted inside a Boeing experimental jet transport, selected for the NASA sponsored Supersonic Transport (SST) Program. In this actual in-flight application, the SD 80 is inserted between the pilot's controls and the aerodynamic control surfaces of the jet plane. This permits the total control system to assume the dynamics of any of a wide variety of SST types.

(Photo courtesy, the Boeing Company)



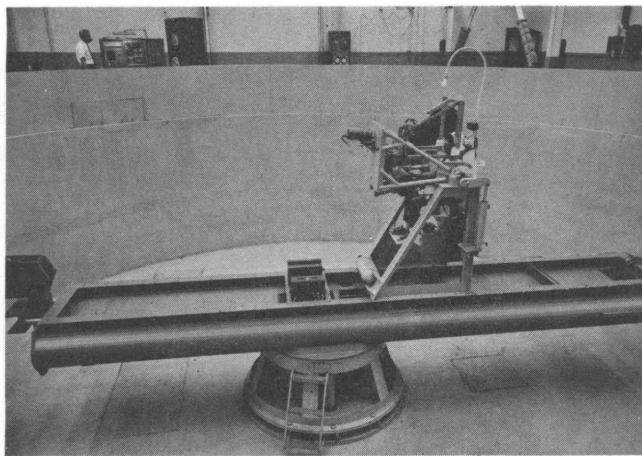
A student in mechanical engineering is shown how to simulate a mass-spring-damper system on an SD 3300 analog computer. Stating equations is unnecessary. Using block programming techniques, illustrated in Chapter 4, the student need only understand the basic relationship of physical variables and constants. Following the program block schematic for a given system, the student can easily program the problem on the computer, observe results on the oscilloscope, and make further parameter adjustments to seek an optimum solution.

which relate iterative programming techniques to the Systron-Donner computers. Chapters 7 to 22 develop more sophisticated programming techniques and applications on a progressive basis.

¹HYDAC is a registered trademark of EAI.

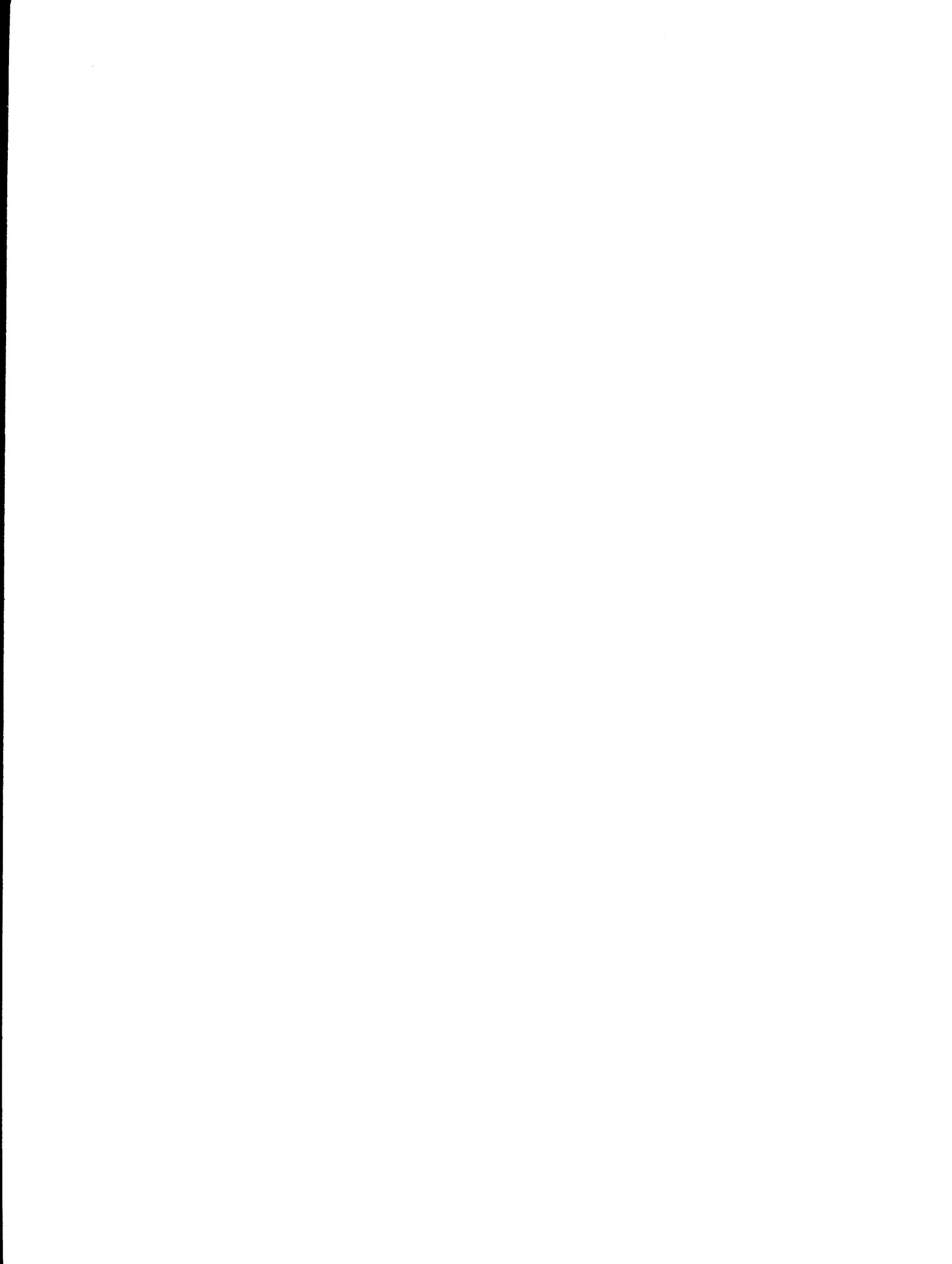
On the Apollo Program, Douglas Aircraft coupled an SD 40 computer to a large centrifuge. The computer calculated and integrated errors in human performance during Apollo lifting body reentry simulation studies. The Systron-Donner computer is seen next to the programmer, in upper left portion of picture.

(Photo courtesy, Douglas Aircraft Co.)



Analog computers are now standard computing equipment for classroom teaching and research work in colleges and universities. Students in the departments of Electrical Engineering, Mechanical Engineering, Chemistry and Biosciences receive instruction in the use of desk top analog computers as basic electronic model builders of dynamic problems. The ease and swiftness of presenting a solution on a readout (oscilloscope, XY recorder), and the ability to vary problem parameters and observe immediately their corresponding effects, have made the analog computer an important teaching aid.

(Photo courtesy, University of Santa Clara)



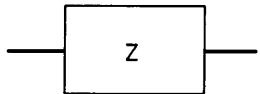
CHAPTER 2

BASIC ANALOG COMPUTING ELEMENTS

This chapter shows how electronic equipment and circuits are used to implement mathematical relationships in an analog computer.

OHM'S LAW

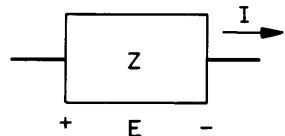
Ohm's law describes the relationship between the current through, and the voltage across a passive impedance. A passive impedance is a collection of passive elements such as resistors and capacitors connected together in an arbitrary way. Such an impedance is generally considered (in analog computing) to be a two-terminal network which can be denoted by



where Z is the dynamic impedance of the element. Ohm's law states

$$E = ZI$$

for



where I is the time-varying current (in the direction indicated) through the passive element generated by the impressed time-varying voltage, E (with the polarity indicated).

In order to simplify what follows, transform notation will be used where

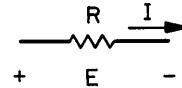
$$s f(t) = \frac{d}{dt} f(t)$$

and

$$\frac{1}{s} f = \int f(t) dt.$$

For a resistor $Z = R$ and Ohm's law is

$$E = RI$$



For a capacitor

$$C \frac{dE}{dt} = I$$

so that Ohm's law is

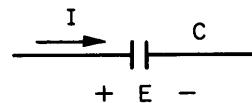
$$E = \frac{1}{Cs} I$$

or

$$E = ZI$$

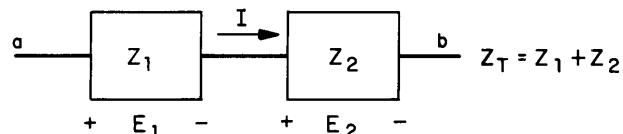
where

$$Z = \frac{1}{Cs}$$



SERIES and PARALLEL IMPEDANCE

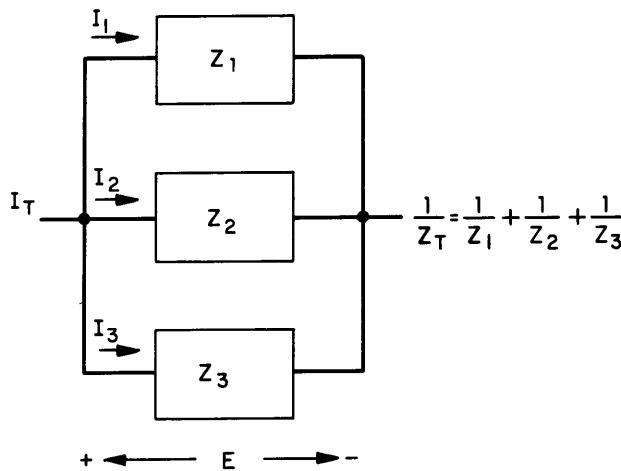
Impedances are additive in series:



This can be proved in a simple way. The total voltage from point a to point b is $E_T = E_1 + E_2$ and by Ohm's law

$$E_T = Z_1 I + Z_2 I = (Z_1 + Z_2) I = Z_T I.$$

When impedances are connected in parallel the total impedance can be found as the reciprocal of the sum of the reciprocals:



This again can be proved by simple application of Ohm's law:

$$I_T = I_1 + I_2 + I_3$$

$$I_T = \frac{E}{Z_1} + \frac{E}{Z_2} + \frac{E}{Z_3}$$

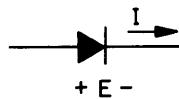
$$\frac{1}{Z_T} = \frac{I_T}{E} = \frac{1}{Z_1} + \frac{1}{Z_2} + \frac{1}{Z_3}$$

The total impedance of two parallel impedances has the simple formula:

$$Z_T = \frac{1}{\frac{1}{Z_1} + \frac{1}{Z_2}} = \frac{Z_1 Z_2}{Z_1 + Z_2}$$

DIODE

A diode is a non-linear resistor. Its resistance or impedance depends on the direction of the current flowing through it. It is denoted by



where

$$E = ZI$$

$$Z = R_1, E > 0$$

$$= R_2, E < 0$$

and generally

$$R_2 \gg R_1.$$

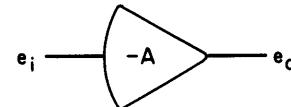
Typically, R_1 ranges between 1 ohm and 100 ohms; and R_2 between 100,000 ohms ($100K\Omega$) and 1,000,000 ohms ($1M\Omega$). Thus the diode is an approximation to a switch for which

$$R_1 = 0$$

$$R_2 = \infty$$

OPERATIONAL AMPLIFIER

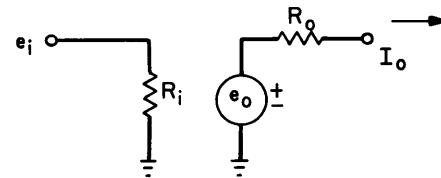
An operational amplifier is of the type that is called a d-c amplifier; it amplifies not only time-varying voltages, but also d-c or constant voltages. It is characterized by its excellent stability and extremely high low-frequency gain (amplification). It is denoted by



where

$$e_o = -A e_i$$

These voltages are measured with respect to ground (zero reference). The gain, A , is usually frequency dependent and will decrease with increasing frequency. As will be seen presently, this becomes a limiting factor in the use of a computer at high speeds. An equivalent circuit for the amplifier is



where, generally, R_i is greater than 10^6 ohms and R_o less than 10^{-4} ohms (closed-loop) at zero frequency. However, the amplifier is current limited. That is, it will only perform satisfactorily if the output current, I_o , is less than some value. The current limit for the S-D¹ amplifier is ± 25 milliamperes at ± 100 volt output. The amplifier is also voltage limited; it will not function satisfactorily if the output voltage, e_o , is greater, in absolute value, than some upper limit. The limit for the S-D solid-state 100 volt amplifier is 105 volts. Since the output of these amplifiers contains unwanted noise whose magnitude typically can be 10^{-2} volts, their effective useful range is about three and one-half decades (5×10^{-2} to 10^2 volts).

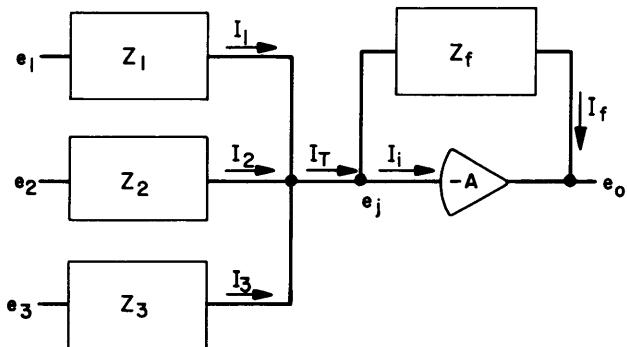
GENERATION OF TRANSFER FUNCTIONS

In what follows, it will be assumed that the input impedance of the operational amplifier is infinite, the output impedance is zero, and the gain is infinite.

¹ Abbreviation for Systron-Donner.

These assumptions introduce negligible error at zero frequency and are a good approximation at mid-frequencies. At high frequency, the assumptions cannot be made (particularly for gain).

Consider an amplifier with input and feedback (from output to input) impedances and applied input voltages as shown.



From Ohm's law

$$I_T = \frac{(e_1 - e_j)}{Z_1} + \frac{(e_2 - e_j)}{Z_2} + \frac{(e_3 - e_j)}{Z_3}$$

$$I_f = \frac{e_j - e_o}{Z_f}$$

$$I_T = I_f + I_i .$$

Now if the input impedance, R_i , of the amplifier is assumed to be infinite

$$I_i = \frac{e_j}{R_i} = 0$$

for finite e_j . Thus

$$I_T = I_f$$

and

$$\frac{(e_1 - e_j)}{Z_1} + \frac{(e_2 - e_j)}{Z_2} + \frac{(e_3 - e_j)}{Z_3} = \frac{e_j - e_o}{Z_f} \quad \dots \quad (1)$$

Further, if the gain, A , of the amplifier is assumed to be infinite, then

$$e_j = \frac{e_o}{-A} = 0$$

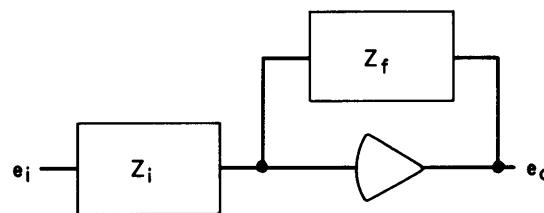
This is a reasonable assumption at low frequencies since as noted above e_o , at most, will be in the neighborhood of 100 volts in absolute value. Thus, equation (1) becomes

$$\frac{e_1}{Z_1} + \frac{e_2}{Z_2} + \frac{e_3}{Z_3} = -\frac{e_o}{Z_f}$$

and

$$e_o = - \left[\frac{Z_f}{Z_1} e_1 + \frac{Z_f}{Z_2} e_2 + \frac{Z_f}{Z_3} e_3 \right] . \quad \dots \quad (2)$$

The simplest case is



$$e_o = - \frac{Z_f}{Z_i} e_i$$

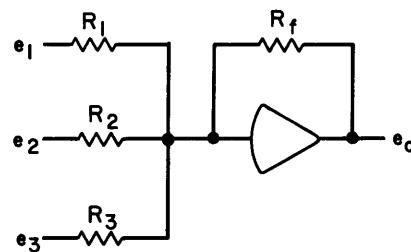
which can be expressed

$$\frac{e_o}{e_i} = - \frac{Z_f}{Z_i} . \quad \dots \quad (3)$$

By a suitable choice of impedances many desired transfer functions can be generated. (Henceforth, the units of megohm and microfarad will be used for resistance and capacitance respectively.)

SUMMER (ADDER)

Consider the configuration shown below, which is called a summer.



If, in equation (2), the substitutions

$$Z_1 = R_1$$

$$Z_2 = R_2$$

$$Z_3 = R_3$$

$$Z_f = R_f$$

are made, then

$$e_o = - \frac{R_f}{R_1} e_1 - \frac{R_f}{R_2} e_2 - \frac{R_f}{R_3} e_3 .$$

A set of typical values (in megohms) for these resistors in a summer in a computer is

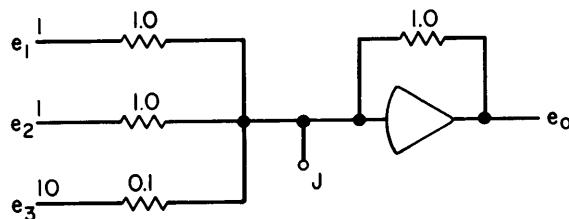
$$R_1 = R_2 = R_f = 1.0$$

$$R_3 = 0.1$$

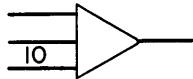
so that

$$e_o = -e_1 - e_2 - 10e_3.$$

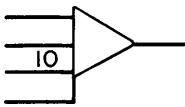
In a computer, access is usually provided to the input (summing) junction so that additional input resistors or feedback components can be added to the summer circuit externally. Thus, the summer circuit is



where the gains (multiplying factors) are indicated at each input, and the summing junction terminal by J. The program symbol for the summer is

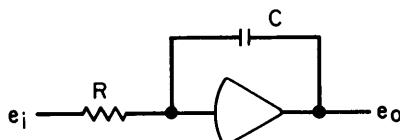


where the input gains are omitted if they are unity. If it is desired to indicate the junction or summing junction (high-gain input), the symbol becomes



INTEGRATOR

The high-gain input is labeled only if pertinent. Consider the configuration shown below, which is called an integrator



Substitution in equation (3) of

$$Z_i = R, Z_f = \frac{1}{Cs}$$

leads to the transfer function

$$\frac{e_o}{e_i} = -\frac{1}{RCs}$$

or

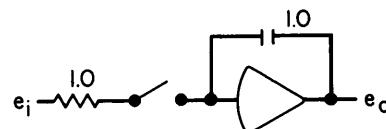
$$e_o = -\frac{1}{RCs} e_i = -\frac{1}{RC} \int e_i(t) dt$$

If $R=1$, $C=1$, then

$$e_o = - \int_0^t e_i(t) dt$$

so that the output voltage, e_o , of the amplifier will be the integral with respect to time of the time-varying input voltage, e_i .

If a switch is included as shown



and if

- 1) there is an initial charge stored on the capacitor which results in the voltage, $e_o(0)$

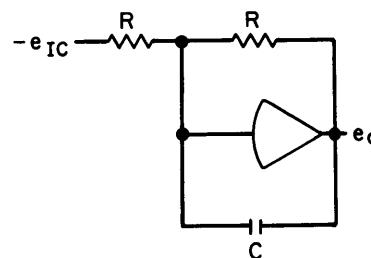
- 2) the switch closes at $t=0$,

then

$$e_o = e_o(0) - \int_0^t e_i(t) dt.$$

It is common practice to call this initial voltage the 'initial condition' for the integrator. The derivation of this terminology is obvious from mathematics.

It is necessary to find a practical way to establish the initial condition or initial voltage for the integrator. To accomplish this, the circuit below is used.



Here

$$\frac{1}{Z_f} = Cs + \frac{1}{R} = \frac{RCs + 1}{R}$$

$$Z_f = \frac{R}{RCs + 1}$$

$$Z_i = R$$

and substituting in equation (3)

$$e_o = \left(-\frac{1}{RCs + 1} \right) (-e_{IC}) = \frac{1}{RCs + 1} e_{IC}$$

or

$$RC \frac{de_o}{dt} + e_o = e_{IC}.$$

The solution of this differential equation is

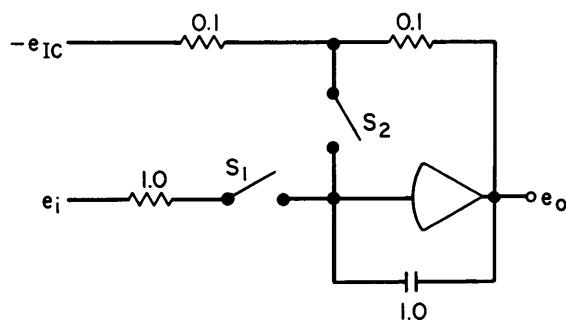
$$e_o = A \exp\left(\frac{-t}{RC}\right) + e_{IC},$$

where A is a constant depending on the initial voltage stored on the capacitor before e_{IC} was applied. Then

$$\lim_{t \rightarrow \infty} e_o = e_{IC},$$

For practical cases, it is only necessary that $t > 10RC$, since $e^{-10} < .0001$, which is compatible with the accuracy of the circuit. The result in simple terms is: to guarantee the establishment of the initial condition it is necessary to wait 10RC seconds after the voltage, $-e_{IC}$, has been applied to the circuit.

The two previous circuits, combined with appropriate switches as shown below, constitute a practical integrator



The operation is as follows:

With S_2 closed and S_1 open, $e_o = e_{IC}$ after 1.0 second (or 0.1 second if $C = 0.1$, etc.).

With S_2 open, e_o still equals e_{IC} , since the current through the capacitor is zero. (This must be true since both S_1 and S_2 are open and the input impedance of the amplifier is assumed to be infinite.)

With S_1 closed at, say, $t=0$, then

$$e_o(t) = e_{IC} - \int_0^t e_i(t) dt.$$

If at $t = T$, S_1 is again opened, e_o will stop changing and remain at the last value before S_1 opened, namely

$$e_o(T) = e_{IC} - \int_0^T e_i(t) dt.$$

If S_2 is closed again while S_1 is open, the output of the integrator will return to

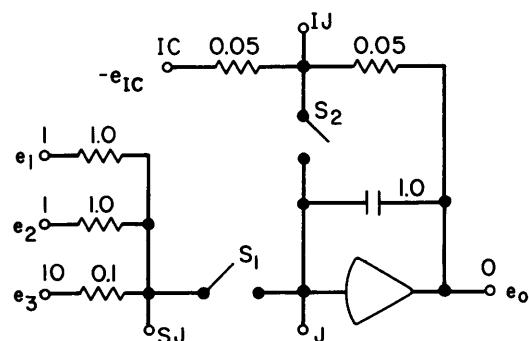
$$e_o = e_{IC}.$$

Each integrator in a computer has these switches as part of its circuit. They are open or closed depending on what the programmer wants the integrator to do. The state of these switches is called the integrator mode. If all the integrators are controlled in unison, the switch states are determined by the main computer mode.

The modes have simple names with obvious interpretations. During the R, or reset (IC) mode, initial voltages are impressed on the integrator capacitors. During the compute or C mode, the integrators integrate input voltages. During the hold or H mode, the integrator outputs remain constant at the last value achieved before entering the hold mode. Thus, the mode permits the programmer to stop the computation at any time, enabling him to evaluate what has happened thus far in the calculation. The table below shows the states of S_1 and S_2 in the various modes. The numeral 1 indicates the switch is closed (logical 1) while 0 indicates the switch is open (logical 0).

MODE	S_1	S_2
R(IC)	0	1
H	0	0
C	1	0

Finally, the general integrator circuit is shown below together with a definition of its transfer function for various modes.



MODE TRANSFER FUNCTION

R $e_o = e_{IC}$

C $e_o(t) = e_{IC}$

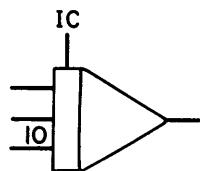
$$- \int_0^t [e_1(t) + e_2(t) + 10e_3(t)] dt$$

H $e_o(T) = e_{IC}$

$$- \int_0^T [e_1(t) + e_2(t) + 10e_3(t)] dt$$

Note that the gain (multiplying factor) of the computing inputs is noted next to the terminals and is determined by the value of the input resistor. The SJ terminal is provided to allow the other external input resistors to be added. The IJ terminal provides the ability to generate an initial condition which is the sum of several voltages. The J terminal permits the connection of additional feedback elements around the amplifier (to be discussed in chapter 10). Also, these terminals provide an external connection for the use of solid-state switches in place of the mechanical switches (relays) S_1 and S_2 . In this case S_1 and S_2 are constrained to be open regardless of integrator mode and the necessary switching is done by the external solid-state switches.

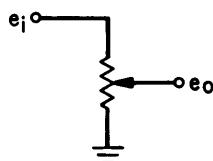
The program symbol for the integrator is



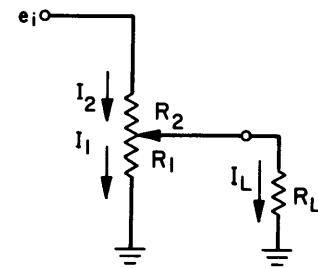
The input gains are usually omitted if they are unity.

POTENTIOMETERS

It is necessary to have a device for entering constant parameters in the computer program. This is accomplished with a potentiometer. The circuit for this element is shown below.



If the output voltage, e_o , of the potentiometer (pot) is applied to an input of another element of the computer, then the input impedance, R_L , of the other element is connected from the pot output to ground.



Then

$$I_2 = I_1 + I_L$$

so that

$$\frac{e_i - e_o}{R_2} = \frac{e_o}{R_1} + \frac{e_o}{R_L}$$

and

$$\frac{e_i}{R_2} = e_o \left[\frac{1}{R_1} + \frac{1}{R_L} + \frac{1}{R_2} \right].$$

Since

$$R_1 + R_2 = R_t$$

where R_t = total potentiometer resistance,

$$R_2 = R_t - R_1$$

and

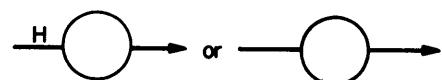
$$e_o = \left[\frac{\frac{1}{R_t} + \frac{1}{R_L} - \frac{1}{R_1}}{\frac{1}{R_L} + \frac{1}{R_L} - \frac{1}{R_L}} \right] e_i$$

$$e_o = \left[\frac{R_1 R_L}{R_t (R_L + R_1) - R_1^2} \right]$$

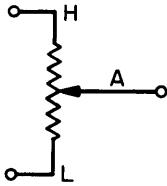
$$e_o = \alpha e_i.$$

Now, obviously, it would be time-consuming to determine R_1 , with the knowledge of R_t , R_L for each new α . Consequently, in practice, the pot is set with the load connected, by reading the output voltage, e_o , with a meter, for a known input voltage, which is usually 100 volts.

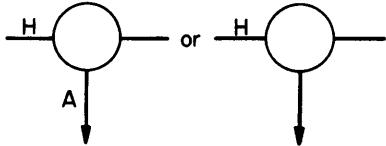
Thus the meter reading is equal to 100α and the pot is changed until the desired value for α is obtained. The symbol for the potentiometer is



where H is the notation for the high end, or input. The H is usually omitted since it is obvious from the computer program which side is the input to the pot. The element discussed above is called a two-terminal pot since it has two available terminals (or connections) on the computer program board (patchboard). Sometimes it is desirable to connect the bottom (low) end of the pot to some computing-element instead of ground. In this case the low end is made available at the patch-board. The circuit is



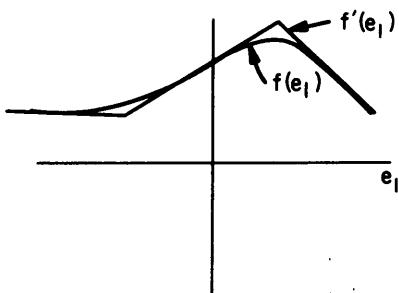
and the program symbol is



where notation for the arm, A, is omitted when it is obvious from the computer program.

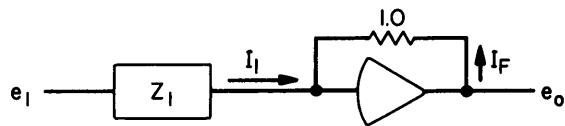
ARBITRARY FUNCTION GENERATOR

Many mathematical problems to be solved with a computer require the generation of an arbitrary function. This is accomplished with a device called an arbitrary function generator (or frequently, diode function generator, since the internal circuitry uses diodes). This device allows the programmer to approximate the desired function with straight line segments. An example is shown below



Here the function $f(e_1)$ is approximated by $f'(e_1)$ with three line segments. In general, each function generator, depending on how it is used, will provide either 10 or 11 line-segments with maximum slope changes of 2 or 2.5:1. Several of these devices can be used together if more segments are required. The location of the slope discontinuity is called the breakpoint and is adjustable. Detailed instructions for the setup of this element appear in Chapter 6.

The principle of operation of the function generator depends on the use of a non-linear input impedance for an operational amplifier. That is, the impedance generates an input current proportional to the function which in turn constrains the output voltage to have this functional relationship to the input voltage. A simplified circuit is shown below.



$$-I_1 = I_F$$

$$I_1 = -f'(e_1)$$

$$e_0 = f'(e_1)$$

The program symbol for an arbitrary function generator is



FIXED FUNCTION GENERATOR

Fixed function generators are used to generate often-used functions such as sine, cosine, log. etc. These are similar to arbitrary function generators in operation, but do not permit the programmer to change the parameters within the device. Generally, a fixed function generator is more accurate and has better frequency and noise specifications than an arbitrary function generator. The program symbol for a sine generator, for example, is



MULTIPLIER

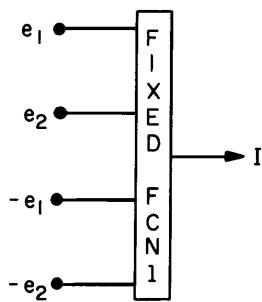
The multiplier used with the S-D computer is called a quarter-square multiplier. The name derives from the equation

$$XY = \frac{1}{4} [(X + Y)^2 - (X - Y)^2]$$

The actual equation to be implemented with hardware in order to provide multiplication is

$$e_1 e_2 = \frac{1}{400} [|e_1 + e_2|^2 - |e_1 - e_2|^2] \quad (4)$$

The multiplier module contains two fixed function generators. A block diagram for one of the generators is

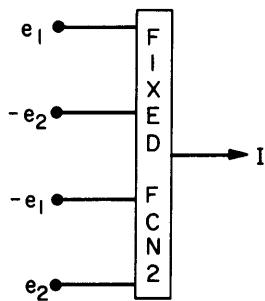


(Note that both polarities are required as inputs for both e_1 and e_2 .) The transfer function is

$$I = K \frac{|e_1 + e_2|^2}{400},$$

where K is a constant that determines the feedback resistor of the output amplifier (to be shown below).

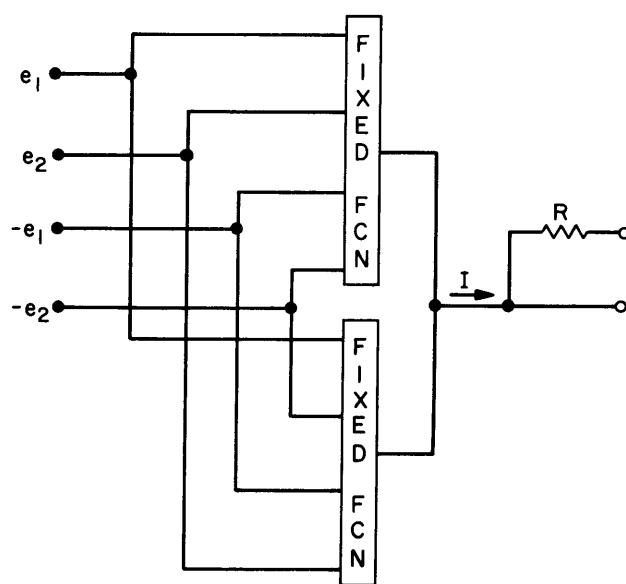
The block diagram for the other is



The transfer function is

$$I = -K \frac{|e_1 - e_2|^2}{400}$$

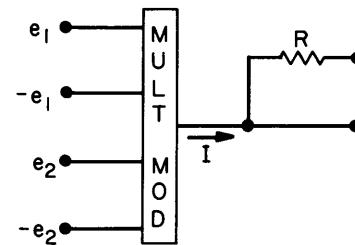
The multiplier module block diagram is



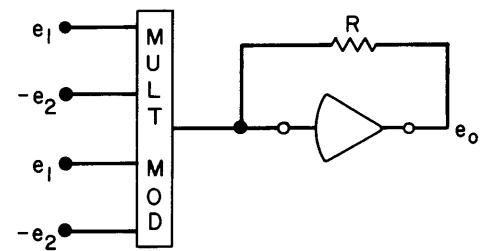
The transfer function is

$$I = \frac{K}{400} [|e_1 + e_2|^2 - |e_1 - e_2|^2] = K \frac{e_1 e_2}{100}$$

A simplified block diagram for the module is



If the module is connected to an output amplifier



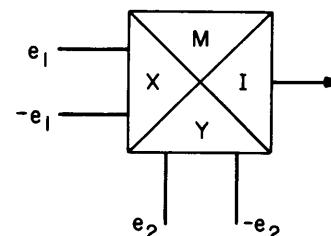
then

$$-\frac{e_0}{R} = \frac{K}{400} [|e_1 + e_2|^2 - |e_1 - e_2|^2] = K \frac{e_1 e_2}{100}$$

and if $K = \frac{1}{R}$

$$e_0 = \frac{-e_1 e_2}{100}$$

The program symbol for the multiplier module (without the output amplifier) is

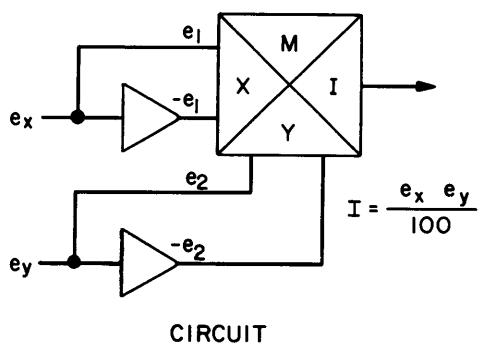


With the designations

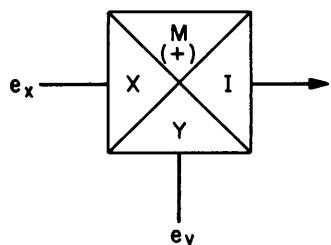
- M: The module is used as a multiplier
- I: The output is a current
- X, Y: Bipolar inputs

If the bipolar inputs are not naturally available from the program (e.g. other variables) then summers can be used to generate them. This can be done in two ways:

Case I:



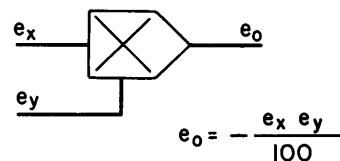
CIRCUIT



PROGRAM SYMBOL

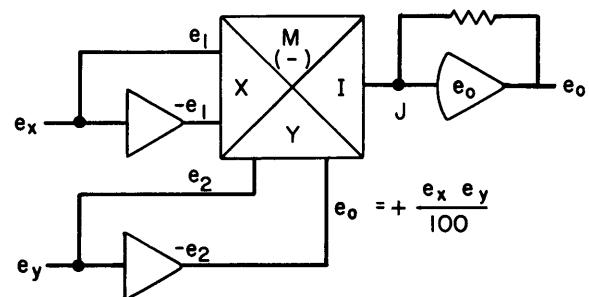
Note that the "minus sign" associated with the "M" indicates a negative current is generated for a positive product, $e_x e_y$. The sign is generally omitted since this is the normal condition.

The program symbol for the multiplier module together with an output amplifier is



As before, summers can be used to generate the bipolar inputs if they are not otherwise available.

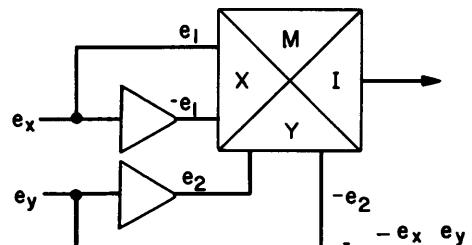
Case I:



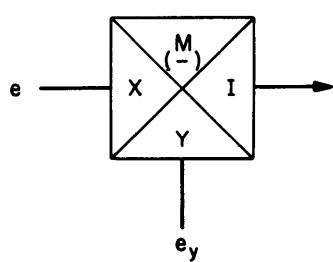
CIRCUIT

Note that the "plus sign" associated with the "M" indicates a positive current is generated for a positive product, $e_x e_y$. The sign is generally omitted since this is the normal condition.

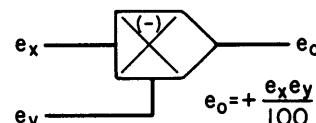
Case II:



CIRCUIT



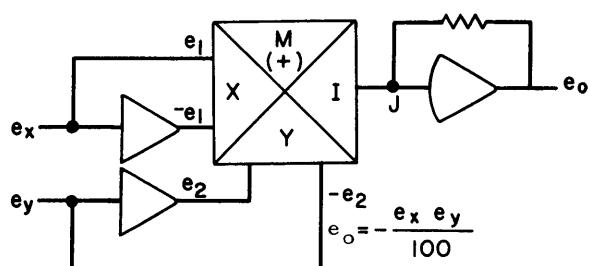
PROGRAM SYMBOL



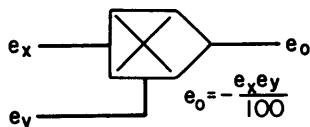
PROGRAM SYMBOL

(Note that here the sign inversion is due to the output amplifier.)

Case II:

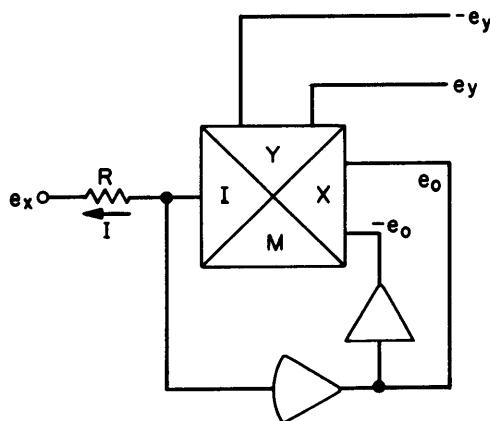


CIRCUIT



PROGRAM SYMBOL

A multiplier module can be used to make a divider with the same output amplifier. (This connection can be made conveniently at the patchboard.)



Here

$$I = K \frac{e_o e_y}{100} = -\frac{e_x}{R}$$

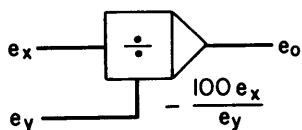
so that for $R = \frac{1}{K}$

$$e_x = -\frac{e_o e_y}{100}$$

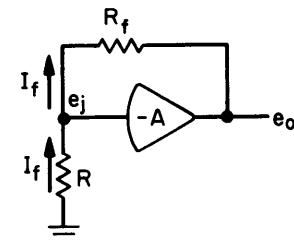
or

$$e_o = -\frac{100 e_x}{e_y}$$

The programmer's symbol is

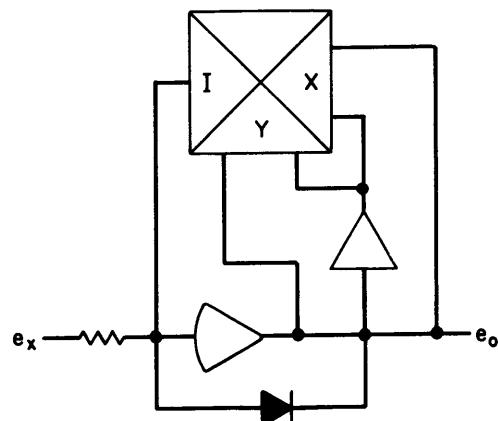


From the circuit it can be seen that the condition $e_y > 0$ must hold in order to avoid instability. This is easy to show. The multiplier module can be considered to be a non-linear feedback resistor, R_f . The sign of R_f is the same as the polarity of e_y . Suppose in the circuit



$R_f < 0$. Then, I_f will be in the direction shown, for $e_o > 0$, and I_f will produce an $e_j < 0$. Since the gain of the amplifier is negative and large, the circuit will be unstable. An additional restriction, $|e_x| \leq |e_y|$, is necessary to ensure that the output amplifier not exceed the voltage limit.

The divider can be used to generate square root in a simple way. If $e_y = e_o$,



CIRCUIT

then

$$e_o = -100 \frac{e_x}{e_o}$$

or

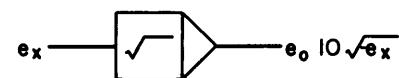
$$e_o^2 = -100 e_x$$

so that

$$e_o = 10\sqrt{-e_x}, e_x < 0.$$

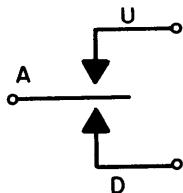
The diode insures that the system will not saturate in the wrong direction if e_x goes positive inadvertently.

This circuit has the program symbol



FUNCTION SWITCH

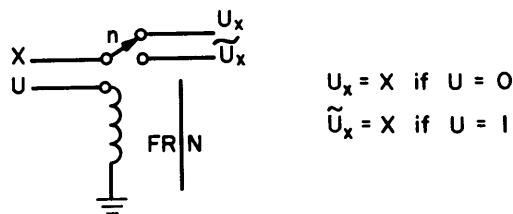
A function switch is a manual switch which can be operated from the control console. It is used to change the computer program during execution. The switch has three positions: up, down, and center (off). Its circuit is shown below.



This is the program symbol as well. The terminals of the switch are located at the patchboard.

FUNCTION RELAY

A function relay is used to make program changes automatically during execution. It is energized (logical 1) for an input equal to or greater than +28 volts and is de-energized (logical 0) for an input equal to or less than 0 volts. The circuit is shown below

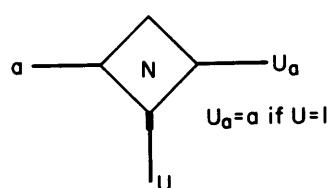


This is also the program symbol where N is the relay number.

ELECTRONIC SWITCH

An electronic switch is used in much the same way as a function relay; the only restriction being that it must be in series with the junction of an amplifier. Its speed of operation is much greater (10^{-5} sec) than a function relay (10^{-3} sec). It requires the same input voltages (logic levels) as the function relay (i.e. logical 1: $28 \leq e_{in} \leq 100$, logical 0: $-100 \leq e_{in} \leq 0$).

The program symbol is

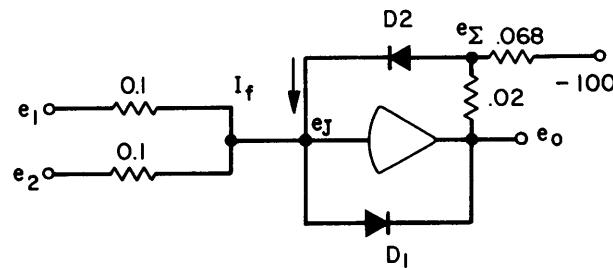


where N is the switch number.

COMPARATOR

Frequently, it is required to determine the sign of the sum of two variables.

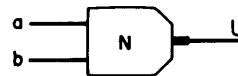
A circuit which accomplishes this is



If $(e_1 + e_2) > 0$, then e_o is limited to approximately 0 volts by D_1 because of its low resistance when it conducts (logical 1). If $(e_1 + e_2) < 0$, then e_o is limited to approximately +28 volts by D_2 (logical 0). The latter is true since

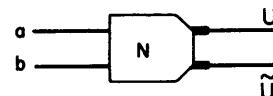
- 1) I_F must be in the direction shown ($e_1 + e_2 > 0$) and $e_\Sigma > e_J$.
- 2) $e_o = 28$ volts since e_Σ must be approximately 0 volts due to the high gain of the amplifier.

The program symbol for the comparator is

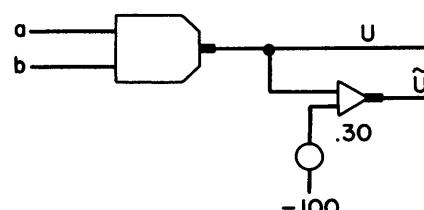


$$U = 1 \text{ if } (a+b) < 0 \\ U = 0 \text{ if } (a+b) > 0$$

If both logical comparator outputs are required, then e_o is complemented with a biased analog inverter. The program symbol is



and the analog circuit is





CHAPTER 3

ELEMENTARY ANALOG PROGRAMMING

All dependent variables in an analog computer are voltages. Consequently, it is necessary to equate these voltages to physical variables. Since the elements of the computer are voltage limited, scaling will be required. The art of scaling is discussed in Chapter 5. In this chapter scaling is ignored since the necessary concepts can be developed without reference to scaling.

As pointed out in Chapter 1, it is necessary to remember that an analog computer cannot differentiate easily. It is a machine whose main feature is its ability to integrate. Consequently, problems which are defined by differential equations must be reformulated in terms of integral equations either explicitly or implicitly. Consider the problem

$$y = y(x)$$

$$\frac{dy}{dx} + By = 0$$

$$y(0) = A.$$

This can be rewritten as the integral equation

$$y + B \int y dx = 0$$

$$y(0) = A.$$

For this problem computer time will represent x , and

$$y(t) + B \int y dt = 0, \quad t = x$$

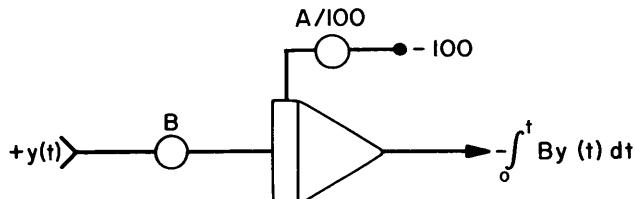
$$y(0) = A$$

An integrator, as shown in Chapter 2, integrates from $t = 0$ onward. It also requires an initial condition: namely the value of the integral at $t = 0$, which in this case is A . Thus, the problem is formulated for the computer as

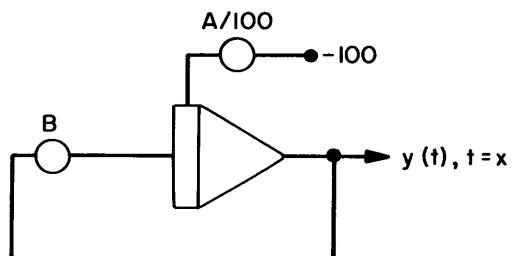
$$y(t) = - \int_0^t B y(t) dt, \quad (1)$$

$$y(0) = A$$

The right half of equation (1) can be generated by



where the constant B and the initial condition A are inserted by means of pots. Remember that the integrator inherently has a sign change. Equation (1) states that $y(t)$ is equal to the negative integral. All that is needed, to complete the computer program for this problem, is to connect the output of the integrator to the input of the pot set to B . Hence, the computer program is



The independent variable, y , can be recorded from the output of the integrator.

In this example the differential equation was reformulated explicitly in terms of an integral equation. It is frequently possible to do this in an implicit way, as the next example will show.

Consider the problem

$$\ddot{y} + A\dot{y} + By = f(t), \quad y = y(t) \quad (2)$$

$$y(0) = C$$

$$\dot{y}(0) = D$$

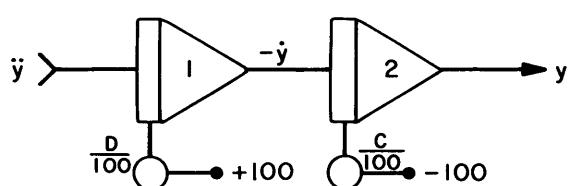
This is equivalent to

$$\ddot{y} = f(t) - A\dot{y} - By$$

$$y(0) = C$$

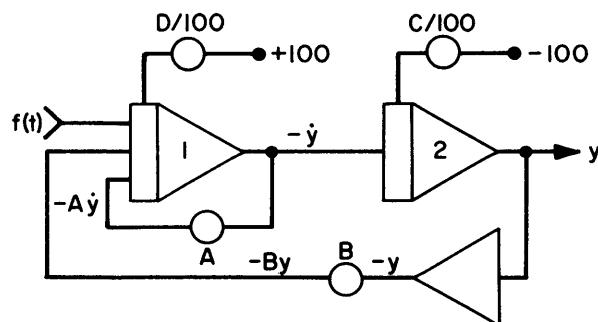
$$\dot{y}(0) = D$$

First, y can be generated from \ddot{y} by



(Note that again the pots are used to generate initial conditions.) Using this circuit, the complete program can be generated by

where equation (2) is satisfied by applying the correct inputs to integrator 1.



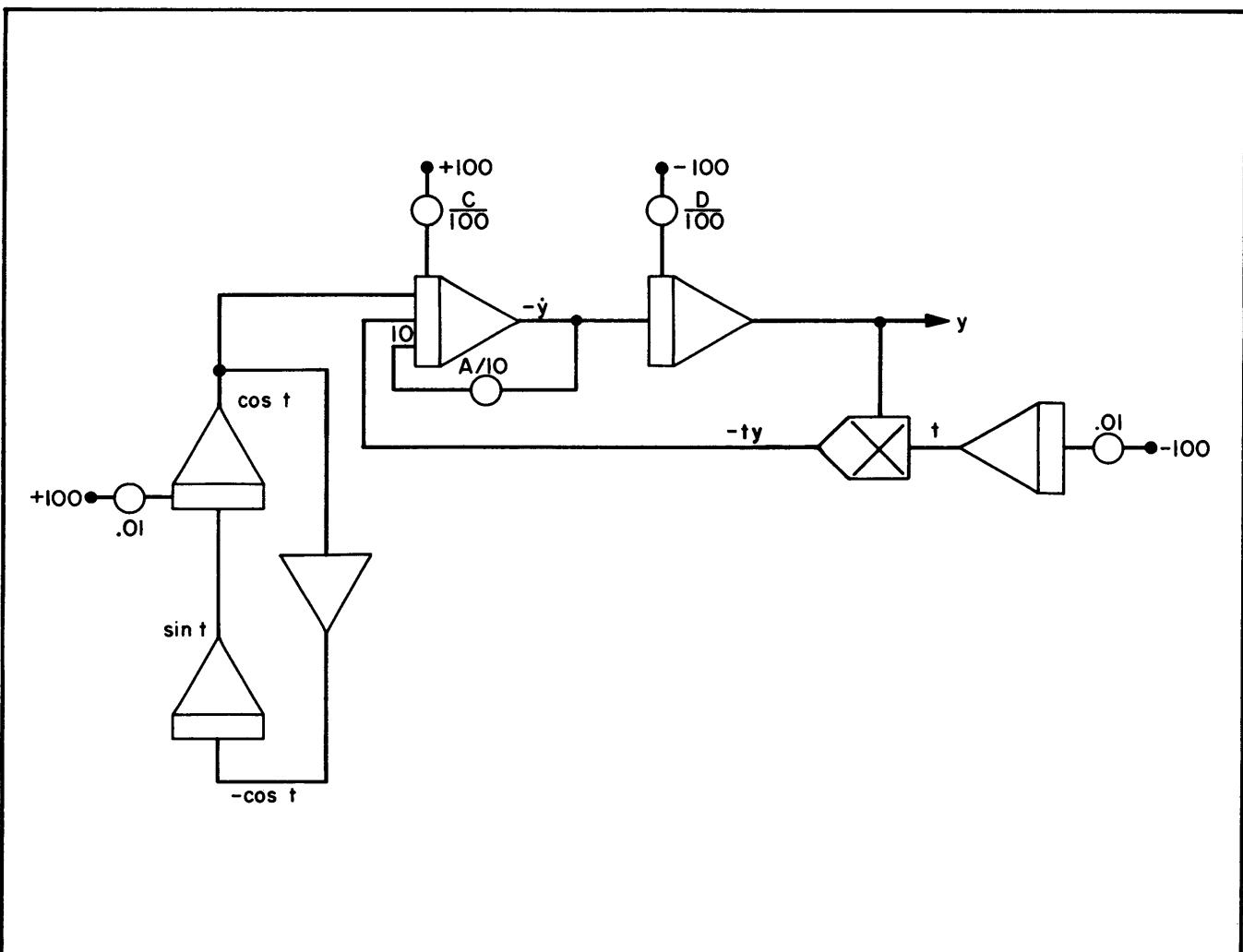
Next, the use of non-linear elements is demonstrated.
Consider

$$\ddot{y} + A\dot{y} + ty = \cos t, \quad 1 \leq A \leq 10,$$

$$\dot{y}(0) = C,$$

$$y(0) = D.$$

The computer program (except for scaling) is



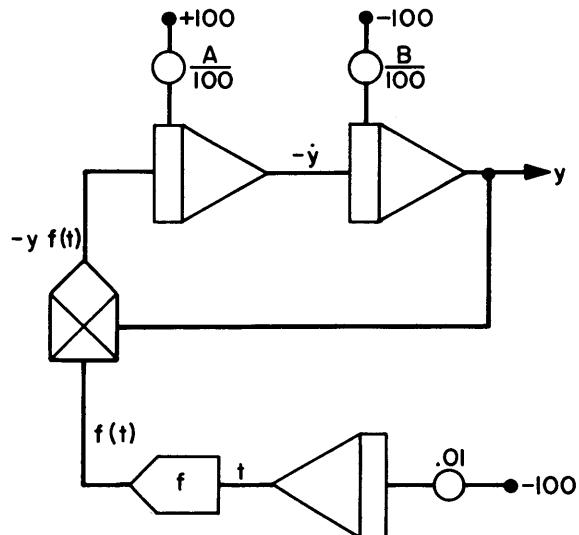
As another example, consider

$$\dot{y} + f(t)y = 0$$

$$y(0) = A$$

$$y(0) = B$$

The computer program is

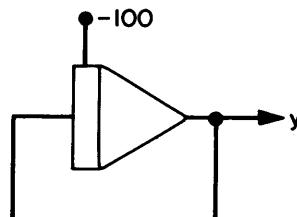


The next example shows how the DCU's (Chapter 8) can be used to determine the solution of a differential equation for a specified value of the independent variable. Suppose it is required to find $y(.17)$ for

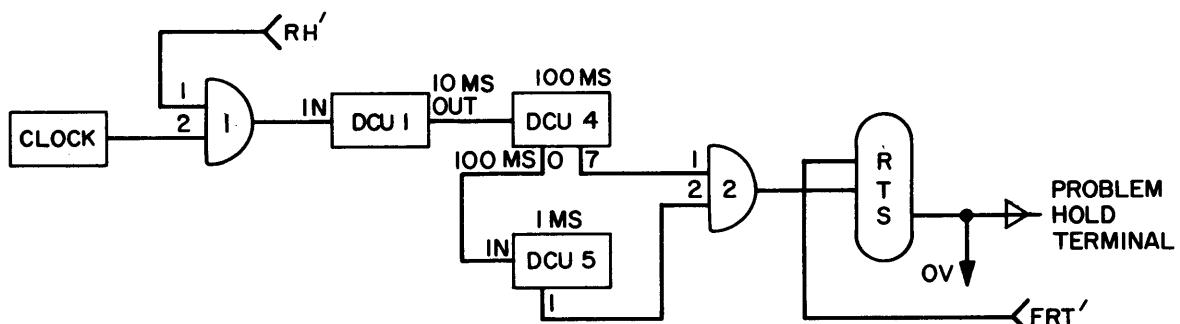
$$\dot{y} + y = 0$$

$$y(0) = 100.$$

From above, the analog program which generates y is



In order to determine $y(.17)$ the computer is allowed to compute for 0.17 seconds and then put into hold. The latter can be accomplished by applying +28 volts to the "problem hold" terminal at the patchboard after 0.17 seconds. This will halt the entire computer and in particular the computation carried out by the above program. The digital program which will do this is shown below:



The operation is as follows: The clock pulses are not applied to DCU 1 until the machine is in compute because the output of the RH' bus is a logical 0 in IC and this is an input to gate 1. The FF is initially in the R state when the machine is in compute having been reset by the FRT' Logic output. Divider 1 reduces the clock frequency of 1000 cycles/sec to 100 cycles/sec. The output of the 0 count of counter 4 is 10 cycles/sec and is connected to the input of counter 5. The 7 count output of counter 4 and the 1 count output of counter 5 are connected to gate 2. Thus, whenever $t = 0.17$ sec the output of gate 2 will change from 0 to 1, which will change the FF to the S state. At this time the computer will go to the hold mode due to the S output of the FF being connected to the problem hold terminal at the patchboard. Also, the clock and counters will be reset because of the logical 1 applied to the OV terminal. As soon as the machine goes to the reset mode the FRT' logic output will cause the FF to return to the R state. Thus, the logic circuit is ready to be used

again as the computer is manually put first in the reset mode and then the compute mode.

ALGEBRAIC EQUATIONS

The first examples were intended to illustrate the basic approach to programming. The following illustrates a more serious application of the analog computer. Suppose it is desired to solve a set of simultaneous algebraic equations which are expressed in matrix form by

$$AX = C \quad (3)$$

where the $n \times n$ matrix A and the column matrix C are known. For the purpose of illustration it will be assumed that A is of rank 2, although the derivation which follows is perfectly general. Consider

$$\dot{\mathbf{x}} + A\mathbf{x} = \mathbf{C} \quad (3a)$$

When the system has reached steady-state (assuming there is one)

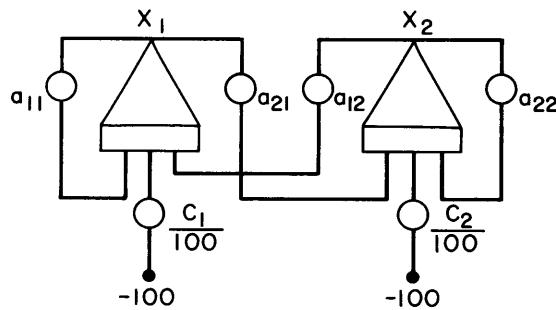
$$\dot{\mathbf{X}} = \mathbf{0}$$

$$\mathbf{AX} = \mathbf{C}.$$

Thus the solution to equation (3) is obtained from the steady-state solution of equation (3a). The latter can be written

$$\begin{aligned}\dot{x}_1 + a_{11}x_1 + a_{12}x_2 &= C_1 \\ \dot{x}_2 + a_{21}x_1 + a_{22}x_2 &= C_2\end{aligned}\quad (4)$$

and the analog program is



Generally, integrator capacitors are chosen equal to $0.001 \mu\text{fd}$ to decrease the solution time. However, there is no guarantee that the system of differential

equations, (4), is stable. A different approach will provide an unconditionally stable system of differential equations. If

$$\mathbf{AX} = \mathbf{C}$$

it follows that

$$\mathbf{A}'\mathbf{AX} = \mathbf{A}'\mathbf{C} \quad (5)$$

where \mathbf{A}' is the transpose of \mathbf{A} . The solution of equation (5) is the same as that for equation (3). Again at steady-state the solution of

$$\dot{\mathbf{X}} + \mathbf{A}'\mathbf{AX} = \mathbf{A}'\mathbf{C} \quad (6)$$

will be the solution of equation (5) and therefore equation (3). The stability of equation (6) is determined by

$$\dot{\mathbf{X}} + \mathbf{A}'\mathbf{AX} = 0. \quad (7)$$

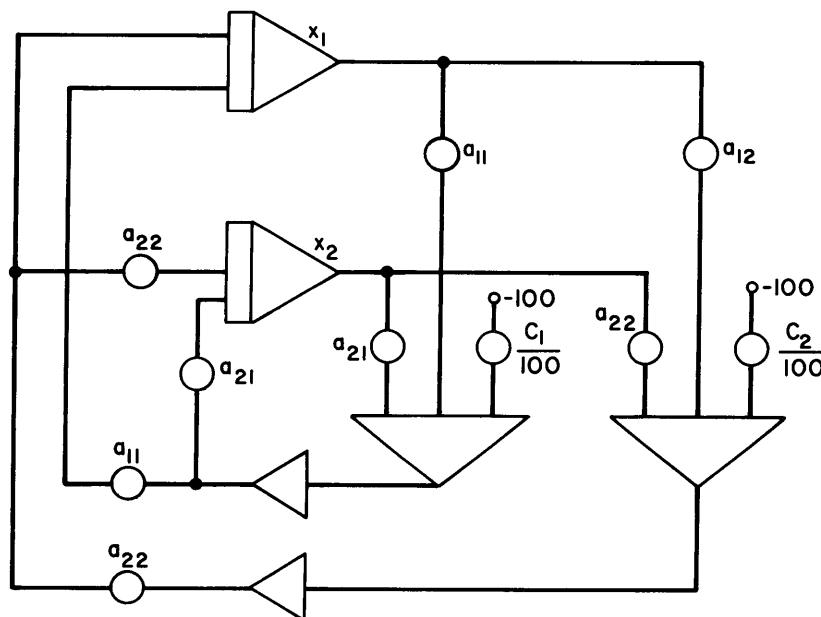
If equation (7) is multiplied by the row matrix \mathbf{X} , then the result is

$$\mathbf{X}\dot{\mathbf{X}} + \mathbf{X}\mathbf{A}'\mathbf{AX} = 0.$$

It is well known that $\mathbf{X}\mathbf{A}'\mathbf{AX}$ is positive definite (i.e. non-negative for all values of \mathbf{X}). Unless $\mathbf{X} = 0$,

$$\mathbf{X}\dot{\mathbf{X}} < 0$$

and stability is guaranteed. The computer program for equation (6) is



In this program it is easy to change individual coefficients because each pot is associated with only one coefficient. The program can be simplified so that it requires no more equipment than the previous program if the following coefficient changes are made.

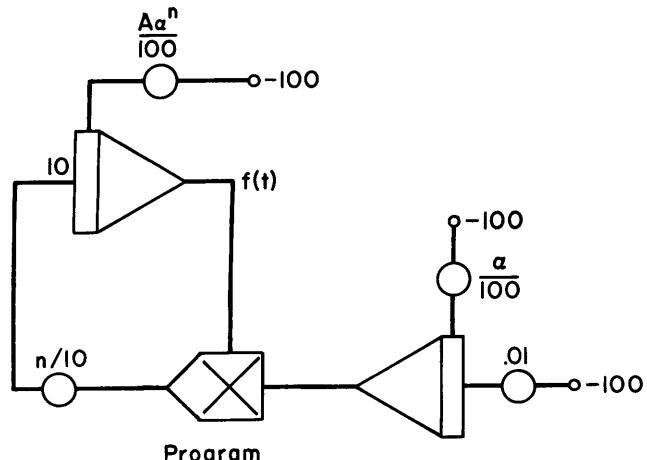
<u>OLD</u>	<u>NEW</u>
a_{11}	$a_{11}^2 + a_{21}^2$
a_{12}	$a_{11}a_{12} + a_{21}a_{22}$
a_{21}	$a_{11}a_{12} + a_{21}a_{22}$
a_{22}	$a_{22}^2 + a_{12}^2$
C_1	$a_{11}C_1 + a_{21}C_1$
C_2	$a_{12}C_2 + a_{22}C_2$

$$f(t) = A(t + \alpha)^n, t > 0:$$

$$\frac{df}{dt} = \frac{n}{t + \alpha} f$$

$$f(0) = A\alpha^n$$

Generating Equation



This simplified program saves equipment but is inconvenient to use when the solution is required for several different values of the coefficients.

ARBITRARY CLOSED FUNCTIONS

Many analytic or closed functions can be generated by representing them in terms of their generating differential equations. The following examples illustrate the technique.

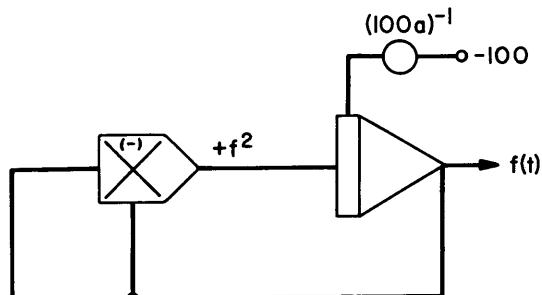
$$f(t) = 1/t, t > a:$$

This function has the generating equation

$$\frac{df}{dt} = -f^2$$

$$f(a) = \frac{1}{a}.$$

The program which solves the generating equation is

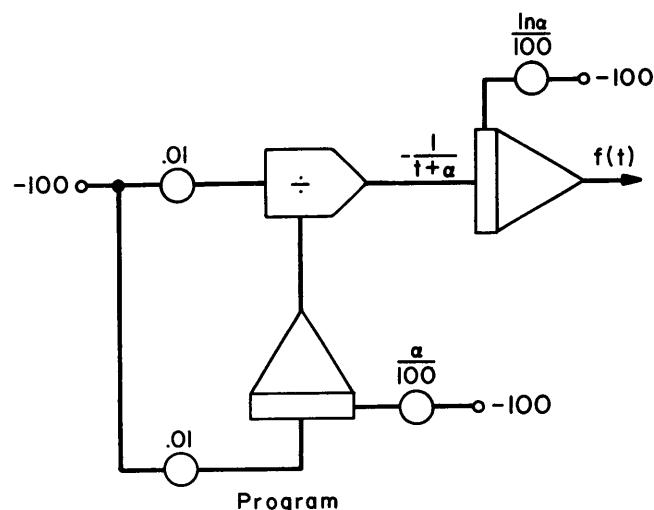


$$f(t) = \ln(t + \alpha), t > 0:$$

$$\frac{df}{dt} = \frac{1}{t + \alpha}$$

$$f(0) = \ln \alpha$$

Generating Equation

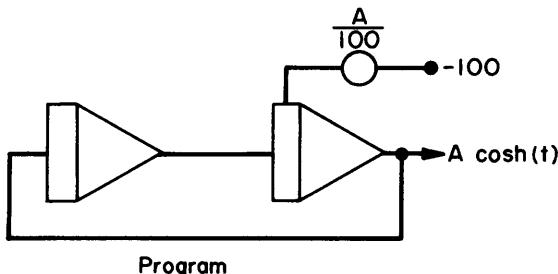


$$f(t) = A \cosh(t), t > 0:$$

$$\dot{f} = f$$

$$f(0) = A$$

Generating Equation



Program

Mean value:

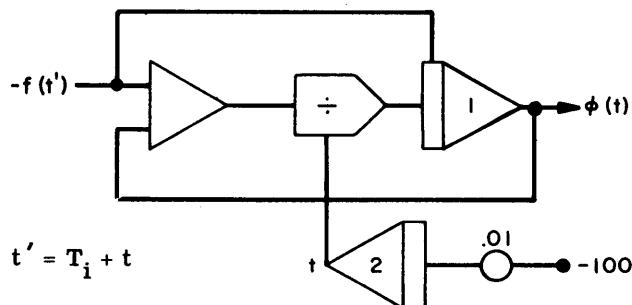
The mean value, $\phi(t)$, of a function, $f(t)$, over the interval $[T_1, T_1 + t]$ is defined by

$$\phi(t) = \frac{1}{t} \int_{T_1}^{T_1+t} f(\xi) d\xi, \quad t \geq 0. \quad (8)$$

The generating equation for ϕ can be obtained by differentiation:

$$\frac{d\phi}{dt} = \frac{1}{t} [f(T_1 + t) - \phi(t)] \quad (9)$$

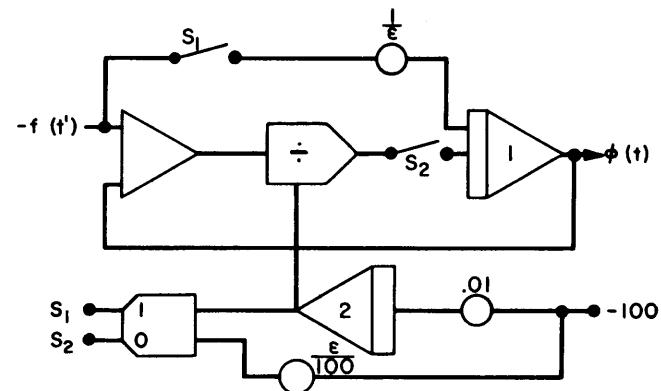
The program is



Suppose the integrators 1, 2 are in the initial condition mode until $t' = T_1$, at which time their mode is changed to compute. Then, theoretically, the program will generate the solution of equation (8). However, the program will not work because the right side of equation (9) is indeterminate for $t = 0$. Thus the divider, in the program will have an unstable output. Practical limitations require the Y-input to be at least 3 volts for useful operation. Consequently, it is necessary to choose a different initial condition:

$$\phi(\epsilon) = \frac{1}{\epsilon} \int_{T_1}^{T_1+\epsilon} f(\xi) d\xi,$$

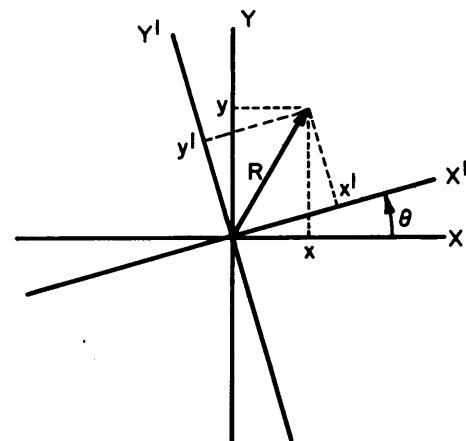
This program is



As before, when $t' = T_1$, the mode of integrators 1, 2 is changed from initial condition to compute. Initially S_1 is closed and S_2 is open. The output of integrator 1 will not be $\phi(t)$ until $t = \epsilon$. When $t = \epsilon$, the correct initial condition, namely $\phi(\epsilon)$, will have been established for integrator 1. At this time the comparator will be activated, which will open S_1 and close S_2 . Also the output of integrator 2 will be ϵ . Thus, the generating equation is implemented by the program starting when $t = \epsilon$. The output of integrator 1 will be $\phi(t)$, $t > \epsilon$.

COORDINATE TRANSFORMATION

Suppose two coordinate systems have the same origin and are displaced by an angular rotation, θ :



An X-coordinate point, say x , will have the coordinates

$$x' = x \cos \theta$$

$$y' = -x \sin \theta$$

in the x' , y' -coordinate system. Similarly, for a Y-coordinate, say y ,

$$x' = y \sin \theta$$

$$y' = y \cos \theta.$$

Thus, a vector, R , in the X, Y-system with components x, y , will have the components

$$x' = x \cos \theta + y \sin \theta$$

$$y' = -x \sin \theta + y \cos \theta$$

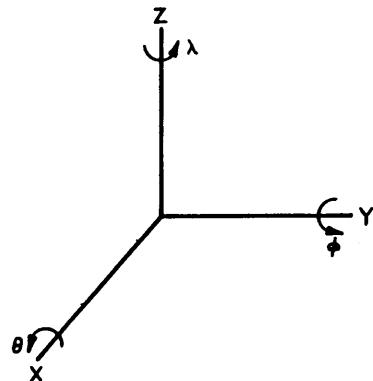
in the X' , Y' -system. In matrix notation

$$R' = TR$$

where R, R' are the representations of the vector in the unprimed and primed coordinate systems respectively, and where

$$T = \begin{bmatrix} \cos \theta & \sin \theta \\ -\sin \theta & \cos \theta \end{bmatrix}$$

In the three-dimensional coordinate system



positive rotations θ, ϕ, λ respectively about the X, Y, Z axes correspond to the transformations

$$R_\theta^X = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & \sin \theta \\ 0 & -\sin \theta & \cos \theta \end{bmatrix}$$

$$R_\phi^Y = \begin{bmatrix} \cos \phi & 0 & -\sin \phi \\ 0 & 1 & 0 \\ \sin \phi & 0 & \cos \phi \end{bmatrix}$$

$$R_\lambda^Z = \begin{bmatrix} \cos \lambda & \sin \lambda & 0 \\ -\sin \lambda & \cos \lambda & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Any arbitrary three-dimensional rotation can be represented as a product of these three matrices. The superscripts refer to the axis about which the coordinate system is rotated and the subscripts to the name of the angle of rotation.

The transformation of a vector, V , by R^X is

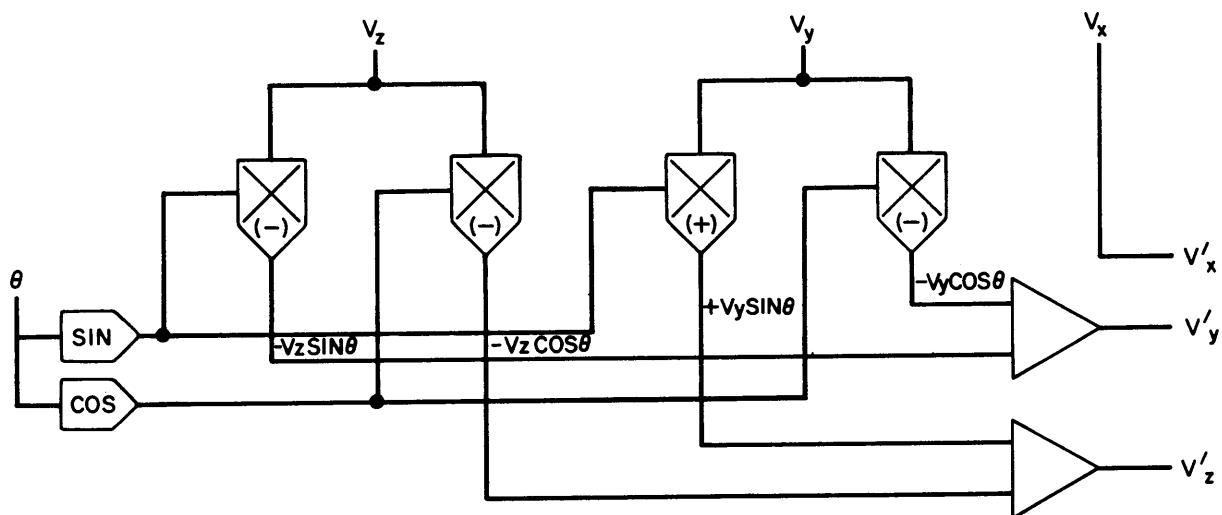
$$V' = RV$$

$$V'_x = V_x$$

$$V'_y = V_y \cos \theta + V_z \sin \theta$$

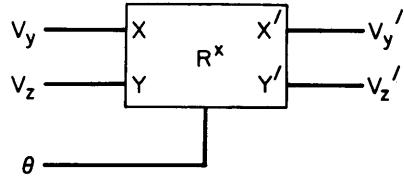
$$V'_z = -V_y \sin \theta + V_z \cos \theta.$$

The program is



Note that θ is limited by the range of the sine and cosine function generators.

The program symbol used frequently is



The transformations R^y , R^z have similar programs.

MATRICES

A matrix, A, is a collection of elements:

$$A = \{a_{ij}\} \quad i = 1, \dots, n \\ j = 1, \dots, m.$$

(in what follows it is assumed that $n = m$). Thus, if $n = 3$

$$A = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$$

Multiplication by a constant, α , follows the rule

$$\alpha A = \alpha \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} \alpha a_{11} & \alpha a_{12} & \alpha a_{13} \\ \alpha a_{21} & \alpha a_{22} & \alpha a_{23} \\ \alpha a_{31} & \alpha a_{32} & \alpha a_{33} \end{bmatrix}$$

which also can be denoted

$$\alpha A = \alpha a_{ij} \quad \alpha A = \alpha [a_{ij}] = [\alpha a_{ij}]$$

when it is understood that the a_{ij} are the elements of A.

Addition follows the rule

$$A + B = a_{ij} + b_{ij} = (a + b)_{ij}$$

or for, say, $n = 2$

$$A + B = \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} \\ b_{21} & b_{22} \end{bmatrix} = \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} \\ a_{21} + b_{21} & a_{22} + b_{22} \end{bmatrix}$$

Multiplication:

$$AB = \sum_{\alpha} a_{i\alpha} b_{\alpha j}$$

Thus for, say, $n = 2$

$$AB = \begin{bmatrix} a_{11}b_{11} + a_{12}b_{21} & a_{11}b_{12} + a_{12}b_{22} \\ a_{21}b_{11} + a_{22}b_{21} & a_{21}b_{12} + a_{22}b_{22} \end{bmatrix}$$

Note that $AB \neq BA$. The unit matrix, I, is

$$I = a_{ij} = I, i = j \\ = 0, i \neq j$$

Thus for, say, $n = 3$

$$I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

I, has the property

$$IA = AI = A$$

Differentiation:

$$\frac{d}{dx} A = \frac{d}{dx} (a_{ij}) = \frac{da_{ij}}{dx}$$

or for, say, $n = 2$

$$\frac{d}{dx} A = \frac{d}{dx} \begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix} = \begin{bmatrix} \frac{da_{11}}{dx} & \frac{da_{12}}{dx} \\ \frac{da_{21}}{dx} & \frac{da_{22}}{dx} \end{bmatrix}$$

Transpose: The transpose, A' , of A is

$$A' = a_{ji}.$$

or for, say, $n = 2$

$$A' = \begin{bmatrix} a_{11} & a_{21} \\ a_{12} & a_{22} \end{bmatrix}$$

Submatrices: The submatrix, S_{ij} , corresponding to the element a_{ij} in A is obtained by deleting the i-th row and j-th column of A. Thus for, say, $n = 3$

$$S_{13} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} = \begin{bmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{bmatrix}.$$

Determinant: The determinant, $|A|$, of A is a scalar quantity and can be found by induction. By definition

$$|A| = \sum_{j=1}^N (-1)^{i+j} a_{ij} |S_{ij}|, \text{ for any } i. \quad (10)$$

(Note that some particular choice is made for i).

For $n = 1$

$$|A| = a_{11}.$$

Thus, for $n = 2$,

$$\begin{aligned} |A| &= \sum_{j=1}^2 (-1)^{1+j} a_{1j} |S_{1j}| = a_{11} S_{11} - a_{12} S_{12} \\ &= a_{11} a_{22} - a_{12} a_{21} \end{aligned}$$

or

$$\begin{aligned} |A| &= \sum_{j=1}^2 (-1)^{2+j} a_{2j} |S_{2j}| = -a_{21} S_{21} + a_{22} S_{22} \\ &= -a_{21} a_{12} + a_{22} a_{11} \end{aligned}$$

and either calculation leads to the same result. For $n = 3$,

$$|A| = \sum_{j=1}^3 (-1)^{i+j} a_{ij} |S_{ij}|,$$

and in particular, any of the equations

$$|A| = \sum_{j=1}^3 (-1)^{1+j} a_{1j} |S_{1j}|$$

$$|A| = \sum_{j=1}^3 (-1)^{2+j} a_{2j} |S_{2j}|$$

$$|A| = \sum_{j=1}^3 (-1)^{3+j} a_{3j} |S_{3j}|$$

will lead to the same $|A|$. If $i = 1$

$$\begin{aligned} |A| &= a_{11} S_{11} - a_{12} S_{12} + a_{13} S_{13} \\ &= a_{11}(a_{22} a_{33} - a_{23} a_{32}) - a_{12}(a_{21} a_{33} - a_{23} a_{31}) \\ &\quad + a_{13}(a_{21} a_{32} - a_{22} a_{31}) \\ &= a_{11} a_{22} a_{33} - a_{11} a_{23} a_{32} - a_{12} a_{21} a_{33} + a_{12} a_{23} a_{31} \\ &\quad + a_{13} a_{21} a_{32} - a_{13} a_{22} a_{31} \end{aligned}$$

Alternatively, $|A|$ can be determined from

$$|A| = \sum_{i=1}^n (-1)^{i+j} a_{ij} |S_{ij}|, \text{ for any } j. \quad (11)$$

A determinant can be multiplied by a constant, α , in a simple way:

$$|\alpha A| = \alpha |A|.$$

From equations (10) and (11) it can be seen that if a matrix B can be generated by m adjacent row interchanges and p adjacent column interchanges starting with matrix A , then

$$|B| = (-1)^{m+p} |A|.$$

Cofactor: The cofactor, C_{ij} , corresponding to the element a_{ij} , in A is

$$C_{ij} = (-1)^{i+j} |S_{ij}|,$$

or for, say, $n = 3$

$$C_{13} = \begin{vmatrix} a_{21} & a_{22} \\ a_{31} & a_{32} \end{vmatrix}$$

and

$$C_{23} = - \begin{vmatrix} a_{11} & a_{12} \\ a_{31} & a_{32} \end{vmatrix}.$$

Inverse: The inverse, A^{-1} , of matrix, A , is

$$A^{-1} = \frac{C_{ji}}{|A|}$$

For example, if $n = 2$

$$A^{-1} = \frac{1}{a_{11} a_{22} - a_{12} a_{21}} \begin{bmatrix} a_{22} & -a_{21} \\ -a_{12} & a_{11} \end{bmatrix}.$$

Orthogonality: A matrix, A , is said to be orthogonal if

$$A' = A^{-1}.$$

Note that R^X , R^Y , R^Z are orthogonal matrices. The geometric interpretation of orthogonality is rotation.

Logical Matrix Operations:

$$(AB)' = B'A'$$

$$(AB)^{-1} = B^{-1}A^{-1}$$

DECOMPOSITION OF ORTHOGONAL MATRICES

Frequently the programmer is presented with a composite orthogonal (rotation) matrix which is a function of several angular variables. Usually these angular variables represent successive rotations about coordinate axes. The simplest way to generate the computer program is to decompose the matrix into a product of the R^X , R^Y , R^Z matrices and then generate subprograms for each of these matrices. This can be accomplished by following a simple set of rules. These are stated without proof and examples are given. R^X , R^Y , R^Z are said to be type 1, 2, 3 matrices respectively. Suppose the composite matrix, M , can be represented

$$M = A_1 A_2 \dots A_N.$$

The outer matrices are A_1 , A_N and the inner matrix M_I is defined by

$$M_I = A_2 A_3 \dots A_{N-1}.$$

Now suppose the simplest element (lowest degree homogeneity) of M is located in row i and column j . Then A_i is of type i and A_N is of type j . The angular variables corresponding to A_1 , A_N will appear in the products of columns j , i respectively. The sense of the rotation, A_1 , (positive or negative) can be found by setting all the variables corresponding to $A_2 \dots, A_N$ equal to zero. The sense of the rotation, A_N , is similarly found by setting the variables corresponding to $A_1 \dots, A_{N-1}$ equal to zero. If the simplest element of M is zero, then M has the representation

$$M = A_1 A_2$$

and there is no inner matrix. If the simplest element of M is not zero, then there is an inner matrix and M_I can be found by setting the variables corresponding to A_1, A_N equal to zero. The further decomposition of M_I is carried out with the same procedure as for M .

Example:

$$M = \begin{bmatrix} \cos \psi & \sin \psi & 0 \\ -\cos \theta \sin \psi & \cos \theta \cos \psi & \sin \theta \\ \sin \theta \sin \psi & -\sin \theta \cos \psi & \cos \theta \end{bmatrix}$$

The simplest element is zero and is in row 1 and column 3. Therefore there is no inner matrix and A_1, A_2 , are of types 1, 3 respectively. The products of column 3 contain the variable θ and the products of row 1 contain the variable for ψ . Therefore, the variable for A_1 is θ and the variable for A_2 is ψ . If $\psi = 0$

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & -\sin \theta \\ 0 & -\sin \theta & \cos \theta \end{bmatrix}$$

If $\theta = 0$

$$M = \begin{bmatrix} \cos \psi & \sin \psi & 0 \\ -\sin \psi & \cos \psi & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Finally,

$$M = R_\theta^X R_\psi^Z$$

Example:

$$M = \begin{bmatrix} \cos \phi \cos \psi + \sin \theta \sin \phi \sin \psi & & \\ -\cos \theta \sin \phi & & \\ \cos \phi \sin \psi + \sin \theta \sin \phi \cos \psi & & \\ & \sin \phi \cos \psi - \sin \theta \cos \phi \sin \psi & -\cos \theta \sin \psi \\ & \cos \theta \cos \phi & \sin \theta \\ & \sin \phi \sin \psi - \sin \theta \cos \phi \cos \psi & \cos \theta \cos \psi \end{bmatrix}$$

The simplest element of M is $\sin \theta$ and is in row 2 and column 3. The products of column 3 contain the variable ψ and those of row 2 the variable ϕ . Therefore A_1, A_3 are of types 2, 3 with variables ψ, ϕ , respectively.

If $\theta = \phi = 0$, then

$$A_1 = \begin{bmatrix} \cos \psi & 0 & -\sin \psi \\ 0 & 1 & 0 \\ \sin \psi & 0 & \cos \psi \end{bmatrix}.$$

If $\psi = \theta = 0$, then

$$A_3 = \begin{bmatrix} \cos \phi & \sin \phi & 0 \\ -\sin \phi & \cos \phi & 0 \\ 0 & 0 & 1 \end{bmatrix}.$$

If $\phi = \psi = 0$, then

$$A_2 = M_I = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos \theta & \sin \theta \\ 0 & -\sin \theta & \cos \theta \end{bmatrix}.$$

Finally,

$$M = R_\psi^Y R_\theta^X R_\phi^Z.$$

As can be seen from the previous example, it is not necessary to find all the elements of A_1 to determine the sense of ψ . Since it is known that A_1 is of type 2 it is only required to evaluate the element in row 1 and column 3 with $\theta = \phi = 0$. Similarly, if the element in row 1 and column 2 is found for $\psi = \theta = 0$, then the sense of ϕ is determined.

Example: (Here we use the abbreviations: $C_\alpha = \cos \alpha, S_\alpha = \sin \alpha$)

$$M = \begin{bmatrix} C_\phi C_\theta C_\psi + C_\phi S_\theta S_\lambda S_\psi + S_\phi S_\theta C_\lambda \\ C_\phi S_\theta C_\psi - C_\phi C_\theta S_\lambda S_\psi - S_\phi C_\theta C_\lambda \\ -C_\phi C_\lambda S_\psi + S_\phi S_\lambda \\ S_\phi C_\theta C_\psi + S_\phi S_\theta S_\lambda S_\psi - C_\phi S_\theta C_\lambda & C_\theta S_\psi - S_\theta S_\lambda C_\psi \\ S_\phi S_\theta C_\psi - S_\phi C_\theta S_\lambda S_\psi + C_\phi C_\theta C_\lambda & S_\theta S_\psi + C_\theta S_\lambda C_\psi \\ -S_\phi C_\lambda S_\psi - C_\phi S_\lambda & C_\lambda C_\psi \end{bmatrix}$$

The simplest element of M is $C_\lambda C_\psi$ which is in row 3 and column 3. The products of column 3 contain θ and those of row 3 contain ϕ . Thus, A_1 is type 3 and A_N is also type 3. θ, ϕ are the variables for A_1, A_N respectively. Consider the element in row 1 and column 2. For $\lambda = \psi = \phi = 0$ its value is $\sin \theta$ and consequently

$$A_1 = R_{-\theta}^Z.$$

For $\theta = \lambda = \psi = 0$ its value is $\sin \phi$ and consequently

$$A_N = R_\phi^Z.$$

To determine the inner matrix, set $\theta = \phi = 0$ so that

$$M_I = \begin{bmatrix} C_\Psi & 0 & S_\Psi \\ -S_\lambda S_\Psi & C_\lambda & S_\lambda C_\Psi \\ C_\lambda S_\Psi & -S_\lambda & C_\lambda C_\Psi \end{bmatrix}.$$

Finally,

$$A = R_{-\theta}^Z R_\lambda^X R_\Psi^Y R_\phi^Z.$$

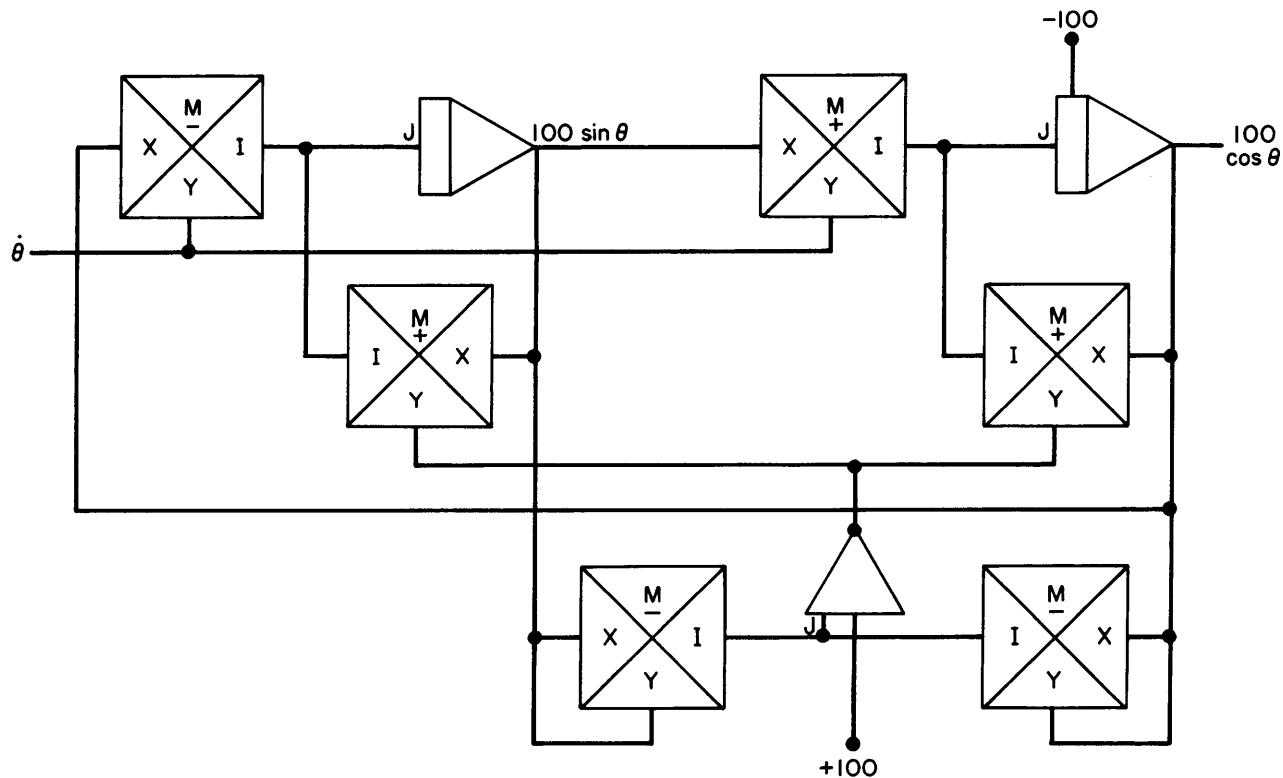
COORDINATE TRANSFORMATIONS WITH UNRESTRICTED ANGLES

The coordinate transformation program is the same as before, except for the use of Sine and Cosine function generators. For unrestricted range we use the variable $\dot{\theta}$ and the equations

$$\sin \theta = \int \dot{\theta} \cos \theta dt$$

$$\cos \theta = -\int \dot{\theta} \sin \theta dt$$

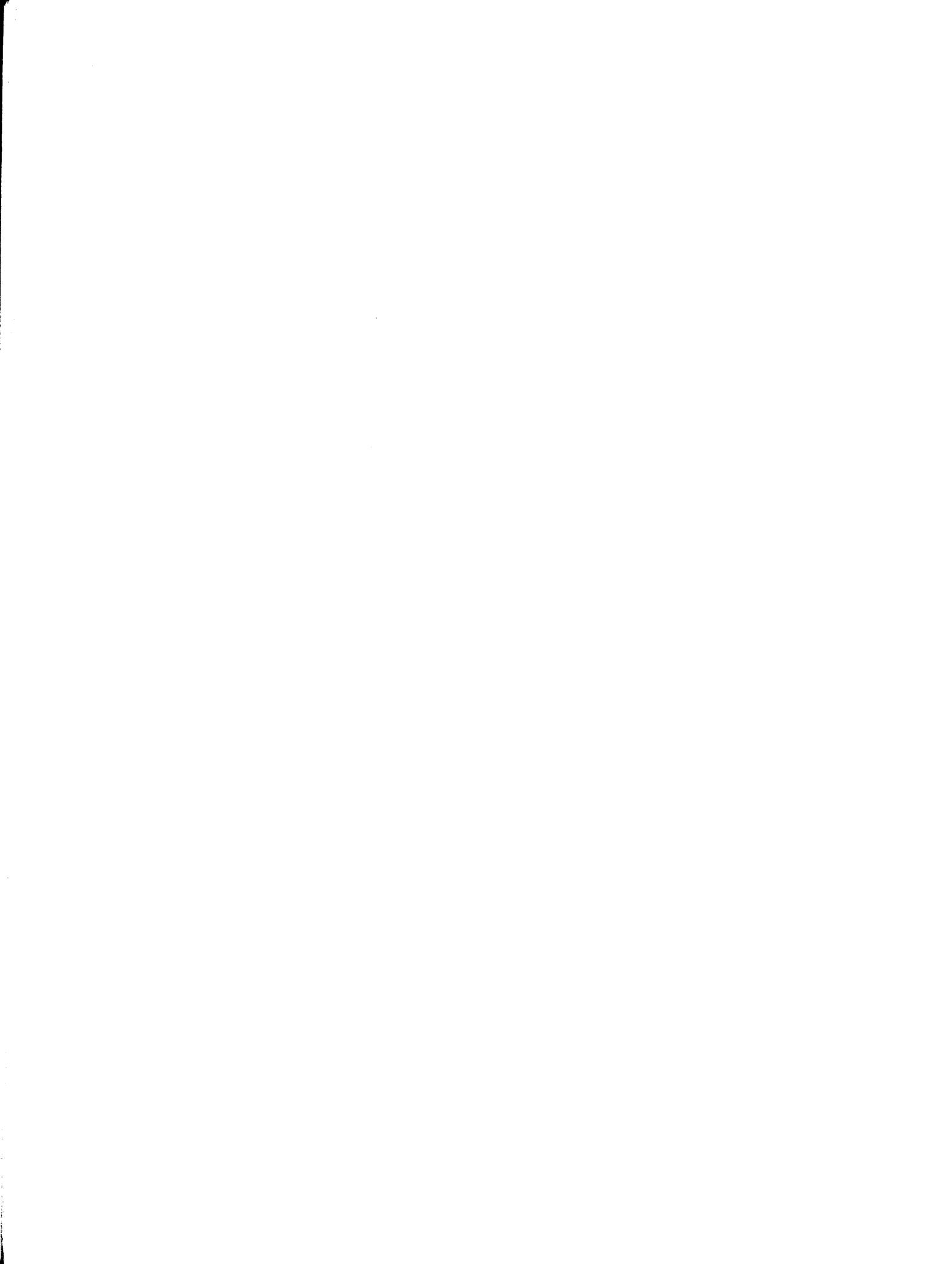
The program is,



The lower two multipliers and summer generate

$$E = -(100 - \sin^2 \theta - \cos^2 \theta).$$

This error is used to change the damping of the integrators to ensure orthonormality of $\sin \theta$, $\cos \theta$.



CHAPTER 4

BLOCK PROGRAMMING FOR PHYSICAL SYSTEMS

A Valuable Aid to Simulation of Lumped-Parameter Physical Systems

The analog computer, or differential analyzer, is a powerful tool for obtaining dynamic solutions to differential equations. Generally the equations to be solved relate to a physical system of particular interest to the computer programmer, in which case the programmer wishes to simulate his physical system by means of an electrical network whose defining equations are analogous (hence the name analog computer) to the subject physical system. It is desirable then, to arrange the computer mechanization to have a one-to-one correspondence between the problem board and the physical system, so that the programmer "sees" his system rather than an abstract electrical network.

The following discussion will illustrate a method of programming in which the physical elements are treated as blocks rather than as sets of differential equations. Certain useful basic building blocks will be developed, and methods for combining the basic blocks to form any desired physical system (simulation) will be discussed. Simulations generated by the methods to be described will have the advantage that any or all physical parameters appear as single controls in the computer so that any parameter may be changed without affecting any other setting. In addition, any or all variables appear as computer outputs and may be individually monitored.

A lumped-parameter system, by definition, consists of a number of interconnected discrete elements whose individual or collective responses to impressed forces or stimuli are of analytical interest. The forces and responses are related by a mathematical operator. Block programming starts with the selection of basic definition and rules.

MATHEMATICAL OPERATOR (O)

The operator, or combination of operators, describes the functional relationship between quantities. The most common operators are the following:

Summer

Constant Multiplier (potentiometer, commonly referred to as "pot")

Inverter - -1 Sign Changer

Integrator - $\frac{1}{s}$ (La Place Transform Notation)

Differentiator - s (This operator is never used in an analog simulation if it can be avoided, as it usually can).

Variable Multiplier

Arbitrary Function - f(x)

Analytical Function - Trigonometric, Log, Hyperbolic, etc.

Inverse Operator - $(O)^{-1}$ s, $\frac{1}{s}$ are inverses

ELEMENT

The element is the basic unit of the block representation of a system. It has a pair of terminals or nodes, with a value associated with each node. It also has a transference quantity between nodes, or "through" the element which is functionally related by some operator to the node-pair value. The transference quantity may also be referred to as the branch transference or transmission.

The elements required to depict the passive lumped-parameters of most physical systems are very few in number. In particular, five basic elements will suffice for many linear electrical, mechanical, and thermal systems. These are the Summer, Constant Multiplier, Sign Changer, Integrator, and Differentiator.

BLOCK

A block is the computer mechanization related to a physical element. The input/output variables of a computer block are voltages and currents, but the operator relationship is the same as the simulated element.

Table I gives examples of single elements and their corresponding blocks. Table II gives a few commonly occurring two-element combinations. Combinations of more than two elements can be generated as shown in the following examples. Useful tables of more complicated blocks or transfer-function simulations appear in many publications.¹

DRIVING SOURCE

A driving source is a source of energy or power for a network of elements.

SYSTEM STABILITY

A stable system is one whose responses are bounded (finite amplitude limit) for any finite input. An unstable system is one whose responses are not bounded.

NETWORK

A network is an interconnected set of elements and sources. Elements may be joined together at their nodes, providing the connected nodes have common dimensions and value, and provided the transference quantities have common dimensions.

This discussion concerns elements and networks in which the following two rules apply: (Kirchoff's laws)

1. The algebraic sum of the node values around any closed path (loop) in a network is zero.
2. The algebraic sum of the transmission quantities at any node is zero.

¹ Korn and Huskey, "Computer Handbook", Chapter 2, McGraw-Hill, New York, 1962.

TABLE I. Single-element Computing Blocks

PHYSICAL ELEMENT	SCHEMATIC SYMBOL	MATHEMATICAL RELATIONSHIP	PROGRAM BLOCK
		Summer Σ $Y = -(X_1 + X_2)$	
		Constant Multiplier (POT) $Y = KX$	
		Inverter $Y = -X$	
		Integrator $Y = \frac{X}{S}$	
		Differentiator $Y = S X$	
Resistor		$I = \frac{(E_1 - E_2)}{R}$	
Linear Spring		$F = K(X_1 - X_2)$	
Torsional Spring		$F = K(\theta_1 - \theta_2)$	
Heat Conductor		$\dot{Q} = K(T_1 - T_2)$	
Viscous Damper		$F = B(v_1 - v_2)$	

TABLE I (Continued)

PHYSICAL ELEMENT	SCHEMATIC SYMBOL	MATHEMATICAL RELATIONSHIP	PROGRAM BLOCK
Electrical Capacitor		$(E_1 - E_2) = \frac{I}{sC}$	
Viscous Damper		$(x_1 - x_2) = \frac{F}{sB}$	
Mechanical Mass		$(v_1 - v_2) = \frac{F}{sM}$	
Thermal Mass		$(T_1 - T_2) = \frac{\dot{Q}}{sM_T}$	
Electrical Inductor		$I = \frac{(E_1 - E_2)}{sL}$	
Mechanical Mass		$x = \frac{(v_1 - v_2)}{s}$	

COMPUTER PROGRAM MECHANIZATION COMMENTS

- Operational amplifiers operate with the junction at virtually zero potential; the currents into the junction through all the input and feedback paths must sum to zero; and there is a sign inversion between the amplifier input and output quantities.
- Any closed-loop in a problem mechanization simulating a stable physical system will generally have an odd number of sign inversions.
- In mechanizing an equation, set the highest order derivative only on the left of the equals sign to obviate the necessity for differentiation.
- The mechanization is accomplished by:

- assigning a position in the mechanization diagram to all necessary variables and operators.
- accomplishing the necessary interconnections so that each output has its proper inputs.
- checking to see that all input/output relationships in the mechanization are satisfied.

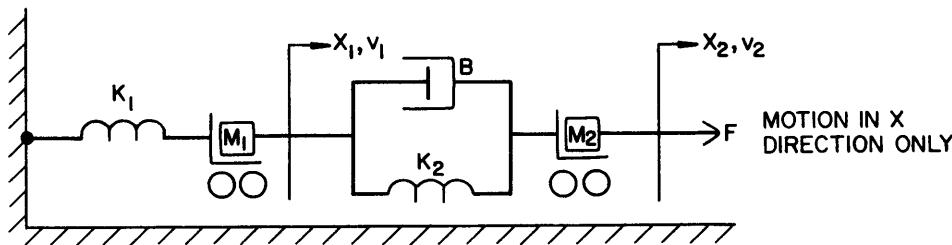
EXAMPLES OF NETWORK PROGRAMMING

The following examples illustrate some of the techniques used in applying the block programming methods. Examples are given for linear elements only. Non-linearities can be included simply by replacing the potentiometer corresponding to the non-linear element with a multiplier, function generator, or other suitable computing element. Backlash, hysteresis, stiction, etc can be included as required.

TABLE II. Two-Element Computing Blocks

SCHEMATIC	MATHEMATICAL RELATIONSHIP	PROGRAM BLOCK
	$(X_1 - X_2) = \frac{Y - K(X_1 - X_2)}{a \cdot s}$	
	$(E_1 - E_2) = \frac{I - \frac{1}{R}(E_1 - E_2)}{sC}$	
	$(X_1 - X_2) = \frac{F - K(E_1 - E_2)}{sB}$	
	$(v_1 - v_2) = \frac{F - B(v_1 - v_2)}{sM}$	
	$Y = \frac{(X_1 - X_2) - KY}{a \cdot s}$ $I = \frac{(E_1 - E_2) - RI}{sL}$ $F = \frac{(v_1 - v_2) - \frac{F}{B}}{sM}$	
	$Y = \frac{(X_1 - X_2) - \frac{Y}{a \cdot s}}{bs}$ $I = \frac{(E_1 - E_2) - \frac{I}{sC}}{sL}$ $v = \frac{(F_1 - F_2) - \frac{Kv}{s}}{sM}$	

MASS - SPRING - DAMPER SYSTEM



Element - force equations

$$\text{Spring } K_1: F_{s1} = K_1 (X_1 - 0)$$

$$\text{Spring } K_2: F_{s2} = K_2 (X_2 - X_1)$$

$$\text{Damper } B: F_D = B (v_2 - v_1)$$

Initial values at $t = 0$.

$$X_1(0), X_2(0), v_1(0), v_2(0)$$

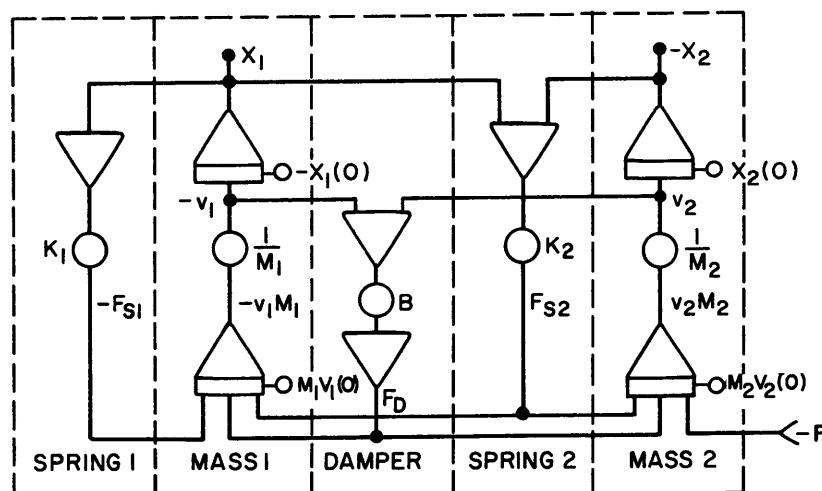
Forces acting on masses

$$F_{M1} = -F_{s1} + F_{s2} + F_D$$

$$F_{M2} = -F_{s2} - F_D + F$$

Steps to mechanize

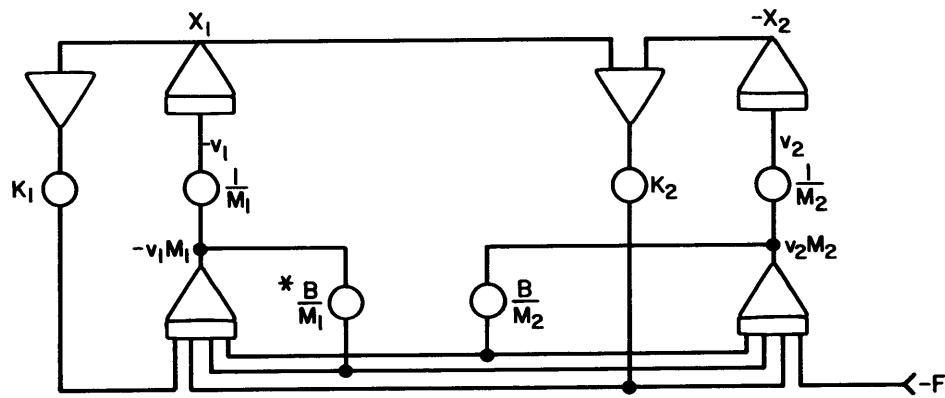
1. Assume $X_1, -X_2, -v_1, v_2$ and F are available
2. Draw blocks for M_1, M_2
3. Draw blocks for K_1, K_2, B
4. Make input connections to mass blocks to produce $X_1, -X_2, -v_1, v_2$



The above mechanization makes each parameter available as an individual adjustment. The programming is easily accomplished by generating the forces imposed on the elements and then operating on the forces to obtain velocity and displacement. Any or all of the initial conditions, $X_1(0)$, $X_2(0)$, $v_1(0)$ and $v_2(0)$, may be zero, or may also be variable.

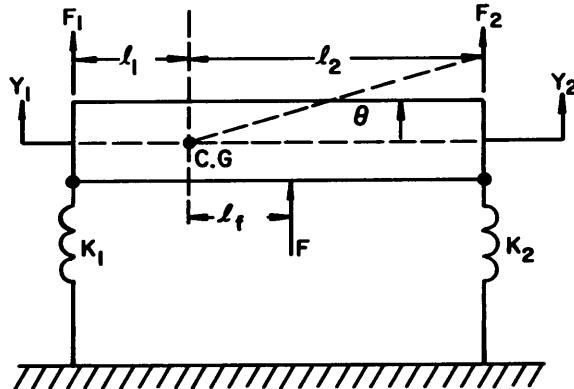
Note that the block representing the damper contains

two summers. The first summer produces $(v_1 - v_2)$, and the second summer acts only as a sign changer. The two-amplifier, one-pot combination can be reduced to a more simple two-pot combination by connecting v_1 and v_2 through B pots to the mass block inputs. The less complex alternate is shown below. This configuration requires fewer amplifiers, but variations in M_1 , M_2 , or B necessitate two or three adjustments.



* Do not connect to $-v_1$ to get pot setting of B instead of $\frac{B}{M_1}$. A coefficient pot output cannot be connected to another coefficient pot input. This is a practical limitation of the computer hardware.

Rigid Bar Supported on Springs



Rigid bar of mass M, moment of inertia about C. G. of I. Assume $\theta \approx \sin \theta$. Vertical motion only. Y_0 is displacement of C. G.

$$\ddot{\theta} = \frac{F_2 l_2 - F_1 l_1 + F l}{sI}$$

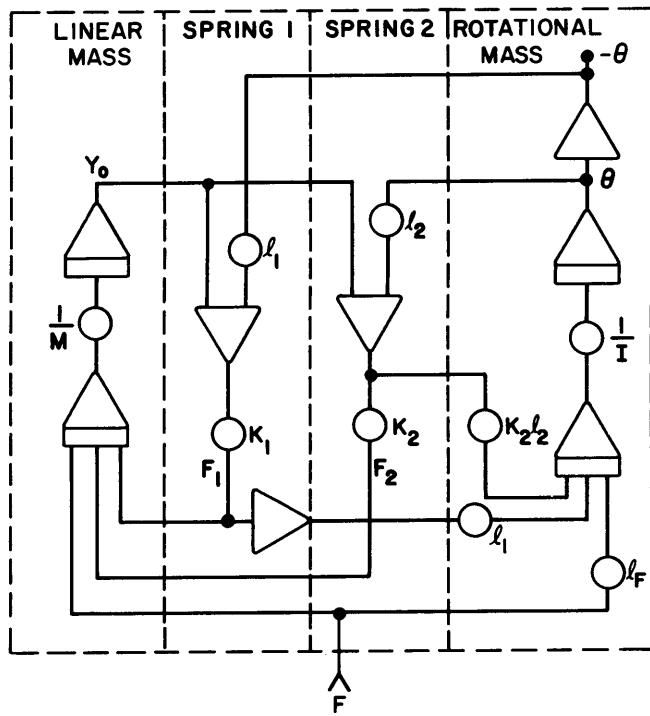
$$\dot{Y}_0 = \frac{F_1 + F_2 + F}{sM}$$

$$F_1 = -K_1 Y_1$$

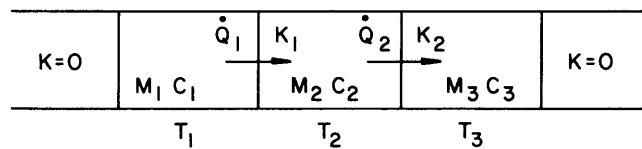
$$Y_1 = Y_0 - l_1 \theta$$

$$F_2 = -K_2 Y_2$$

$$Y_2 = Y_0 + l_2 \theta$$

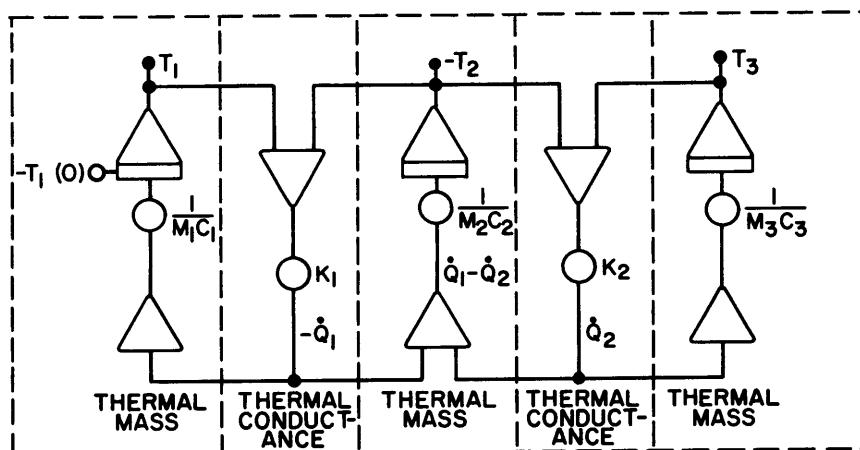


One-Dimensional Heat Transfer

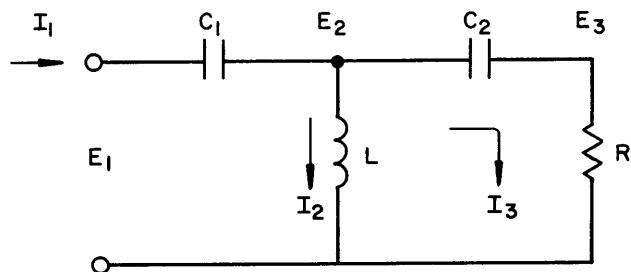
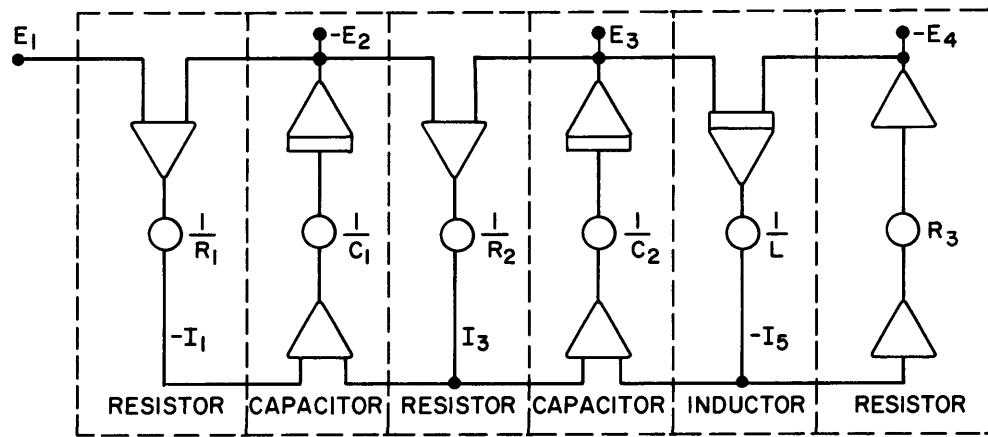
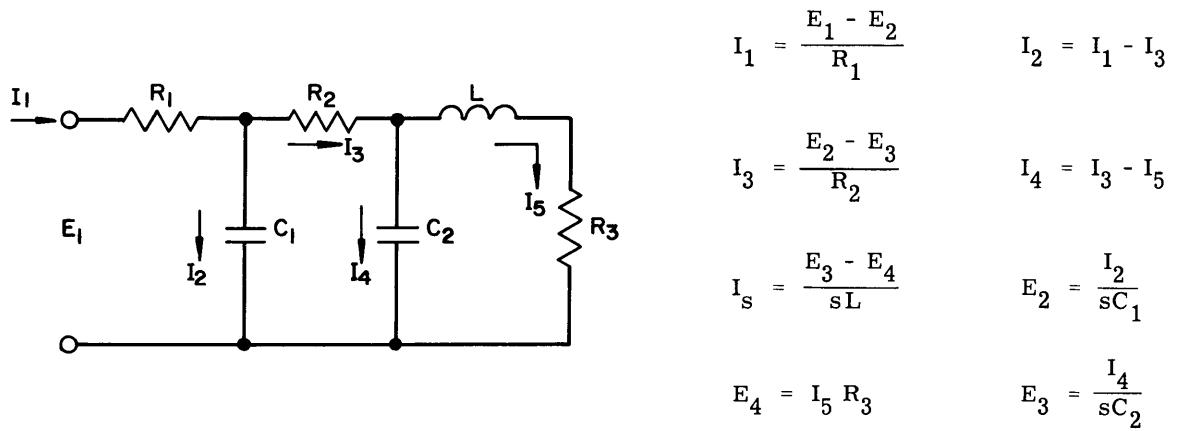


K - THERMAL CONDUCTIVITY
M - MASS
C - SPECIFIC HEAT

$$\begin{aligned}\dot{Q}_1 &= K_1 (T_1 - T_2) & \dot{Q}_2 &= K_2 (T_2 - T_3) \\ sT_1 &= \frac{\dot{Q}_1}{M_1 C_1} & sT_2 &= \frac{\dot{Q}_1 - \dot{Q}_2}{M_2 C_2} & sT_3 &= \frac{\dot{Q}_2}{M_3 C_3}\end{aligned}$$

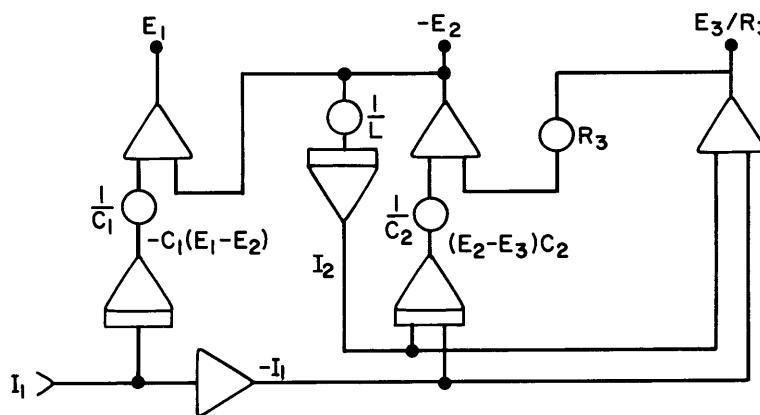


Electrical R-L-C Networks

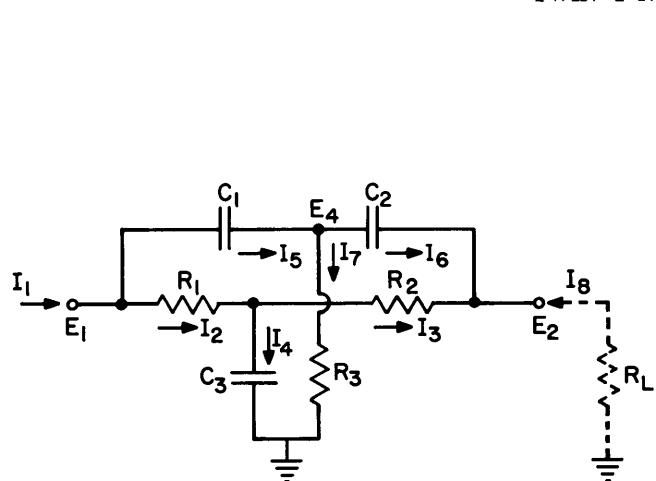


$$E_1 - E_2 = \frac{I_1}{sC_1} \quad E_2 - E_3 = \frac{I_3}{sC_2} \quad E_3 = I_3 R_3$$

$$I_2 = \frac{E_2}{sL} \quad I_3 = I_1 - I_2$$



TWIN-T NETWORK



$$I_2 = \frac{E_1 - E_3}{R_1}; \quad -E_3 = \frac{I_3 - I_2}{sC_3}$$

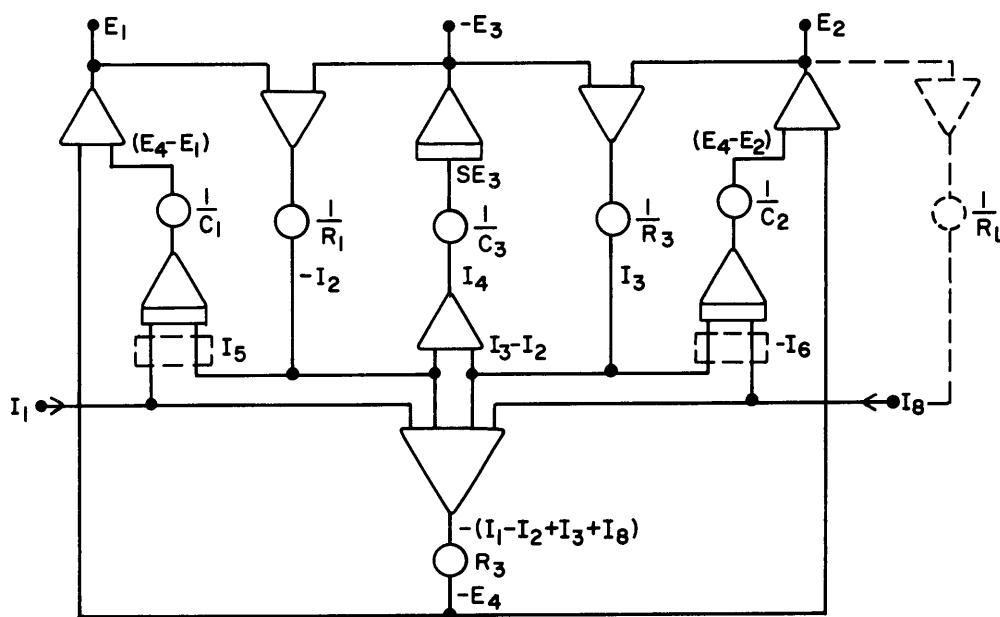
$$-I_3 = \frac{E_2 - E_3}{R_2}$$

$$(E_4 - E_2) = \frac{-(I_8 + I_3)}{sC_2}$$

$$(E_1 - E_4) = \frac{I_1 - I_2}{sC_1}$$

$$E_4 = (I_1 - I_2 + I_3 + I_8) R_3$$

$$E_1 = (E_1 - E_4) + E_4; \quad E_2 = E_4 - (E_4 - E_2)$$





CHAPTER 5

SCALING

The analog computer is voltage limited to a practical range of operation of ± 100 volts as a maximum and ± 1 volt or zero as a minimum. It is usually not possible to numerically equate analog voltages to physical variables. That is, the magnitude of physical variables must be scaled to fall within the useful range of the machine. This procedure is called amplitude scaling.

The analog computer is also limited as to the speed with which it will solve a problem. Practical bounds on solution time are 100 seconds as a maximum and 100 milliseconds as a minimum. Events in the physical world usually occur in time intervals which fall outside these limits. Thus, the simulation of real world phenomena with the analog computer is ordinarily faster or slower. The procedure for relating computer time to physical time is called time scaling.

Correct scaling is an important factor in reducing simulation errors. An error analysis of any particular program is about as complex as the problem which is being solved by the program. Consequently, except for unusual cases, an analysis of error is not made. An estimate of the accuracy of the program can be found from check cases for which the answer is already known. Error is reduced by scaling the program so that all potentiometer values and amplifier gains are reasonable. Frequently, in the process of scaling a problem, it will be found that some parts of the model are not significant and can be eliminated. Scaling is a good check on reasonability.

Scaling is an art, not a science. A good deal of experience is required to become proficient. Scaling can be done in several ways. The techniques discussed in this chapter are those in most prevalent use among analog programmers.

If X is a physical variable corresponding to computer voltage, V , then a scale factor, a , is chosen so that

$$V = a \cdot X$$

will fall within the practical range of the computer. It is good practice to make V as large as possible. If t_p is physical (real) time corresponding to t_c , the computing time, then these two can be related by a scale factor, N :

$$t_c = N t_p, \quad dt_c^m = N^m dt_p^m$$

It follows that

$$\int(F)dt_c = N \int(F)dt_p. \quad (1)$$

N is chosen so that the problem solution time on the analog computer falls within reasonable limits. Practically, this is accomplished by a choice of N which will not result in prohibitive amplifier gains.

The general procedure for scaling is:

1. Generate an unscaled program having a mathematical structure which agrees with the problem (model) to be solved.
2. Identify the location of all voltages corresponding to physical variables.
3. Associate a scale factor with each of these voltages (e.g. $V_x = a_x X$).
4. Label the program with the scaled physical variables (i.e. $a_x X$) rather than the voltages.
5. Estimate maximum values of the physical variables.
6. Choose the amplitude scale factors and the time scale factor, N , so that all pot settings and gains are reasonable.

In order to accomplish step (4) it is necessary to know how input and output scale factors are related for various computer elements.

POTENTIOMETER

$$\text{Physical equation: } Y = bX$$

$$\text{Scale factors : } V_x = a_x X, \quad V_y = a_y Y$$

$$\text{Scaled equation : } V_y = b \frac{a_y}{a_x} V_x$$

$$\text{Program : } a_x X \xrightarrow[b \frac{a_y}{a_x}]{\text{pot}} a_y Y$$

Note that a scale change can be made with a pot. Assuming either a_x or a_y has been established previously, the other is chosen so that $b a_x / a_y$ is a reasonable pot setting. The pot can be used solely for a scale change. In that case $b = 1$.

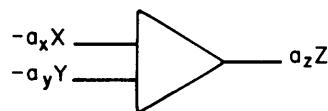
SUMMER

$$\text{Physical equation: } Z = X + Y$$

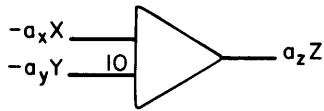
$$\text{Scale factors : } V_z = a_z Z, \quad V_x = a_x X, \quad V_y = a_y Y$$

$$\text{Scaled equation: } V_z = \frac{a_z}{a_x} V_x + \frac{a_z}{a_y} V_y$$

Now, if only one-gain inputs for the summer are used, then it must be true that $a_z = a_x = a_y$ and the program is



Thus, it is seen that input scale factors must be the same for a one-gain summer and that a change of scale cannot be made through a summer. However, a different input scale factor can be used for a 10-gain input. The program for this case is,



where it must be true that $a_x = 10a_y$.

INTEGRATOR

As with the summer, the input scale factors (except for the 10-gain input) must be the same. However, a change of scale can be made with an integrator. Both amplitude and time scaling can be done. Scaling for an integrator is derived below, where C is the value of the capacitor in μfd , and R is in megohms.

$$\text{Physical equation: } Y = \int X dt_p$$

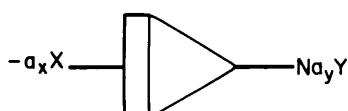
$$\text{Scale factors: } V_y = a_y Y, V_x = a_x X, t_c = Nt_p$$

$$\text{Scaled equation: } V_y = \frac{a_y}{a_x} \cdot \frac{1}{N} \int V_x dt_c$$

Thus, the integrator gain, $\frac{1}{RC}$, must be

$$\frac{1}{RC} = \frac{a_y}{a_x} \quad (2)$$

and the program is



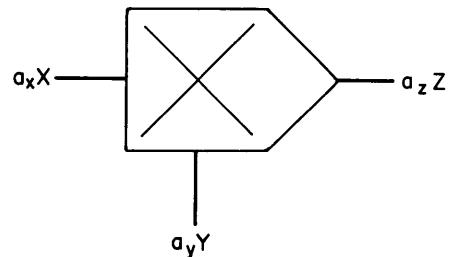
MULTIPLIER

$$\text{Physical equation: } XY = Z$$

$$\text{Scale factors: } V_x = a_x X, V_y = a_y Y, V_z = a_z Z$$

$$\text{Scaled equation: } \frac{V_x V_y}{100} \cdot \frac{100 a_z}{a_x a_y} = V_z$$

Program:



It must be true that

$$\frac{100 a_z}{a_x a_y} = 1.$$

From this requirement it is seen that the relationship among the scale factors is the same as that for input and output multiplier voltages:

$$a_z = \frac{a_x a_y}{100}$$

DIVIDER

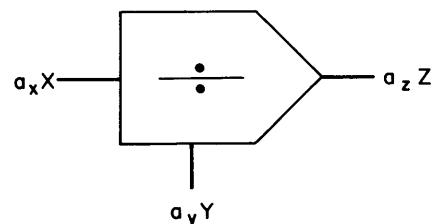
$$\text{Physical equation: } \frac{X}{Y} = Z$$

$$\text{Scale factors: } V_x = a_x X, V_y = a_y Y, V_z = a_z Z$$

$$\text{Scaled equation: } 100 \frac{V_x}{V_y} \cdot \frac{a_z a_y}{100 a_x} = V_z$$

$$\text{Scale factor constraint: } a_z = 100 \frac{a_x}{a_y}$$

Program:



SQUARE ROOT

Physical equation: $Y = \sqrt{X}$

Scale factors : $V_y = a_y Y, V_x = a_x X$

Scaled equation: $V_y = \frac{a_y}{10\sqrt{a_x}} \cdot 10\sqrt{V_x}$

Scale factor constraint: $a_y = 10\sqrt{a_x}$

Program:



FUNCTION GENERATOR

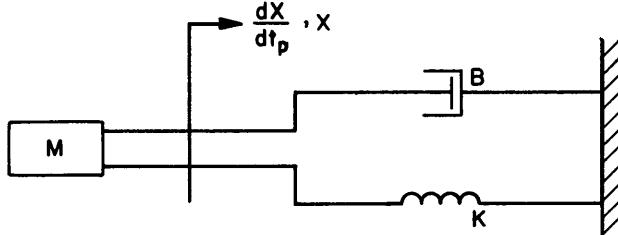
Physical equation: $Y = f(x)$

Scale factors : $V_y = a_y Y, V_x = a_x X$

Scaled equation: $V_y = a_y f(\frac{V_x}{a_x})$

EXAMPLE:

Mass-spring-damper system



Physical equations: Spring Force, $F_s = KX$

Damper Force, $F_d = B \frac{dX}{dt_p}$

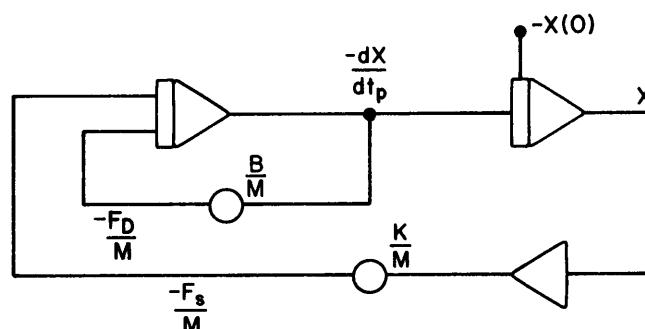
Mass Acceleration, $\frac{d^2X}{dt_p^2} = \frac{-(F_s + F_d)}{M}$

$$M = 1; B = 4 \times 10^{-4}; K = 6.4 \times 10^{-5}$$

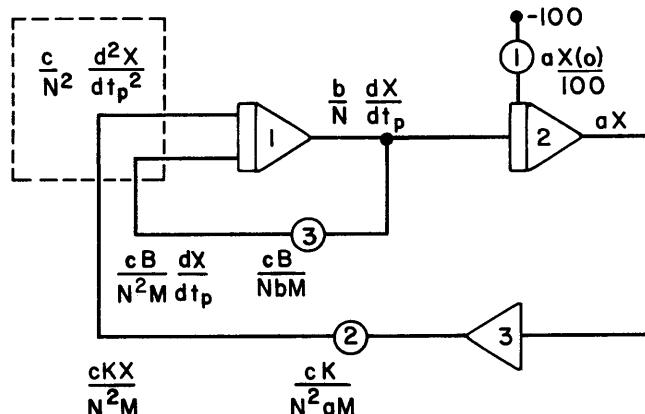
$$\frac{dX}{dt_p}(0) = 0, X(0) = .95$$

Estimates: $\left(\frac{dX}{dt_p}\right)_{max} = 8 \times 10^{-3}; X_{max.} = 1$

The unscaled program is:



The scaled program is obtained by assigning literal scale factors, determining their numerical values, and calculating the coefficient potentiometer settings.



The $\frac{1}{N^2}$ and $\frac{1}{N}$ factors appear due to the inherent gain of N through each integrator, from Eq. (1).

In general, the determination of the numerical values involves a trial and error approach. Any convenient parameter can be picked for the starting point. If a selected value leads to unreasonable potentiometer settings or unreasonable amplifier gains, then new values may be required. Reasonable pot gains are between .1 and 1. Reasonable amplifier gains are between 1 and 20. Amplifier gains should be integer values. Non-integer values can be obtained from a potentiometer, in combination with gain-of-10 inputs, if necessary.

1. Determination of a. In order to utilize the full dynamic range of integrator 2, set $aX_{max} = 100$ (volts).

$$a = 100$$

2. Potentiometer 1 setting.

$$P1 = \frac{a X(0)}{100} = .95 \quad (3)$$

3. Determination of $\frac{b}{N}$. At this time, a trial value for N may be selected. Analog solution times of .1 to 100 seconds, or computer natural frequencies from .1 to 100 radians per second are the preferable operating ranges. An estimate of the physical time or frequency ranges can be used to select a value for N.

For this example, a value of $N = .01$ appears reasonable.

$$\frac{b}{N} = \frac{100 \text{ (volts)}}{(dX/dt)_{\text{max}}} \quad (4)$$

Looking ahead, note that integrator 2 will require a gain of $\frac{aN}{b}$. The gain of N is inherent in the integrator from Eq. 1. Therefore, the $\frac{1}{RC}$ value for the integrator will be a . It is highly desirable that this factor be an integer, preferably either 1 or 10. Therefore, $b = 100$ will be used. The effect of this is to force integrator 1 to utilize something less than its full dynamic range, but the effect is not serious in this example. As a general rule, it is not possible to scale so that all amplifiers work over their full range.

4. Potentiometer 2 setting;

$$P2 = \frac{cK}{N^2 aM} = (6.4 \times 10^{-3})c \quad (5)$$

The value of c is chosen to assure that a reasonable value for P2 is obtained and to assure that $\frac{b}{c}$ is an integer.

$c = 100$ will satisfy both conditions.

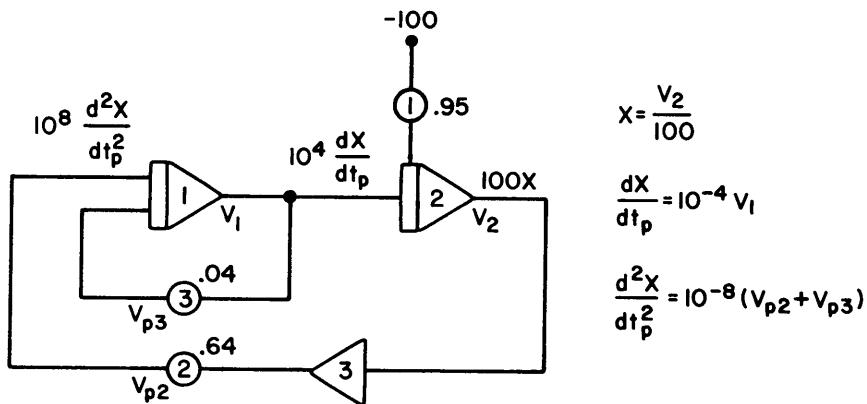
$$P2 = .64$$

5. Potentiometer 3 setting:

$$P3 = \frac{cB}{NbM} = .04$$

This setting for potentiometer 3 is below the desired minimum potentiometer setting value of .1. In this example, however, it is not possible to have a large setting for potentiometer 3. This follows from a consideration of the physical problem. The damper, B, was chosen to have a very slight effect on the mass-spring system, so it necessarily follows that the portion of the analog program related to the damper will have only a slight effect. It should be noted that abnormally low potentiometer settings will frequently arise in simulations of systems containing negligibly small elements. It should be recognized that the low settings stem from the actual system characteristics. The programmer, therefore, need not spend time in futile attempts to improve the scaling.

The final program with numerical values is:



$$X = \frac{V_2}{100}$$

$$\frac{dX}{dt_p} = 10^{-4} V_1$$

$$\frac{d^2X}{dt_p^2} = 10^{-8} (V_{p2} + V_{p3})$$

GENERAL COMMENTS ON TIME SCALING:

The following points concerning time scaling are of particular importance:

1. If all integrator gains are simultaneously changed by a factor k , the time scale changes from N to $\frac{N}{k}$, and all computer frequencies increase by k . The amplitude scaling, however, does not change. All numerical scale factors, all potentiometer settings, and all computer problem voltages remain the same. This property of amplitude invariance under time scale change allows the programmer to easily speed up or slow down the computer solution.

2. The most convenient way of changing all integrator gains simultaneously is by changing the capacitors by a fixed factor. Integrator capacitors can be simultaneously changed by means of the computer Time Scale control¹.
3. An event which occurs in a fixed real-time interval can occur in several different computer time intervals depending on the choice of integrator capacitors, that is, depending on the time scale factor.
4. Events of identical character which occur in different real-time intervals can all be made to occur in the same computer time interval by appropriate time scaling.

¹ On the SD 10/20 and SD 40/80 analog computers, a master Time Scale switch is a standard feature. This permits fast and convenient switching of capacitors that are patched -- up to 1000:1 time scale (x 10, x 100, x 1000, depending on patch panel connections).



CHAPTER 6

COMPUTER OPERATION

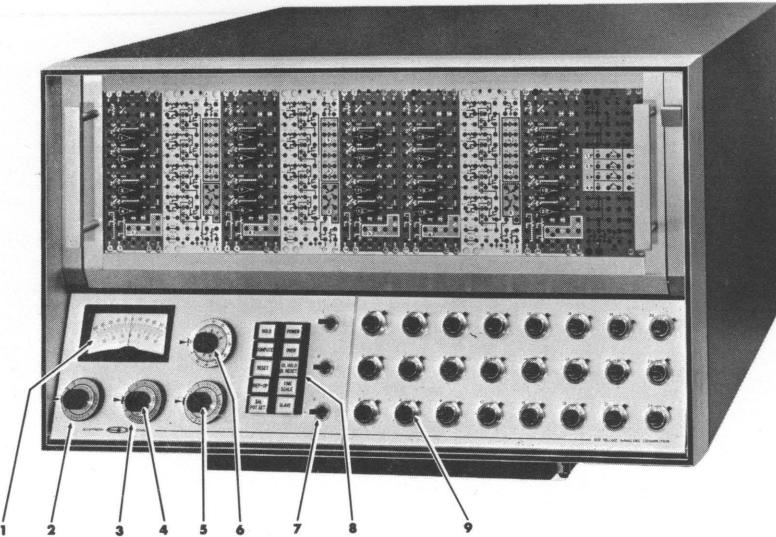
This chapter is based on materials from the Systron-Donner 10/20 and 40/80 Computer Operating Manuals. The descriptions of the computer mode controls, computer logic, and various computing modules permit the reader to relate all applications and programming examples given in this handbook to an existing computer. For example, Chapter 8 offers detailed information on the use of digital logic. The modules needed to implement the applications described in Chapter 8 are shown in this chapter.

Not all available S-D computing components are described here, only those which illustrate and serve to explain a basic function. For the latest, up-to-date availability of all computing elements and modules in the SD 10/20 and 40/80 series, please consult Systron-Donner or your nearest SD engineering sales representative.

SD 10/20 Computer

The Model SD 10/20 is an all solid state analog computer with an operating range of $\pm 100V$ and expansion capabilities for up to 24 operational amplifiers. The principle features of the computer are full $\pm 100V$ computing range, all solid-state design, compact modular construction of all computing components, removable problem board which mates directly with computing modules to eliminate all problem board cabling, a control center with readout selectors for amplifiers and potentiometers, high-speed reed relays and solid-state switches, complete logic capability for repetitive and iterative operations, 0.01% accuracy of computing resistor and capacitor networks, a panel voltmeter having a null mode of measurement with 0.02% (of full scale) resolution. The amplifiers are protected against accidental grounding of their outputs as are all power supplies.

All computing components in the Model SD 10/20 are modular plug-in units. They are identical to those used in the larger SD 80. This high degree of standardization between SD 10/20 and 40/80 computers maximizes usefulness of computing modules. Since all modules are standard plug-in units, the computer user is offered a wide choice of computing components. As new modules become available, the SD 10/20-40/80 series computers can thus be constantly up-graded to include the new modules. Furthermore, a user may start with an SD 10/20 and later use the same 10/20 modules in an SD 80. This common design feature results in important cost savings. As the need for more computer modules increases, the expansion cost from an SD 10/20 to an SD 80 is minimized.



- [1] Panel Meter Ranges:** 1, 3, 10, 30, 100, and 300 volts, and \pm null. Full scale accuracy: 3%. Null position provides 0.02% F.S. resolution with reference potentiometer having a $\pm 0.05\%$ linearity at $25^\circ C$.
- [2] Compute Time Selector** — compute time continuously variable from 5 msec to 10 sec. Reset time varies from 5 msec to 5 sec, depending upon coarse steps of compute range.
- [3] Meter Range Selector** — with positions for 300 v, 100 v, 30 v, 10 v, 3 v, and 1 v. Serves also as sensitivity adjustment for \pm null.
- [4] Function Selector** — for rapid choice of: + null, - null, Meter, External (connects selected bus to external jack).
- [5] Address Selector** — address of all amplifiers and potentiometers.
- [6] Null Reference Potentiometer** — provides high accuracy readout using null method with 0.02% F.S. resolution. Linearity is $\pm 0.05\%$ at $25^\circ C$.
- [7] Function Switches** — provide manual switching flexibility in problem solutions.

- [8] Mode Selection** (lighted pushbuttons):
 - Hold** — places problem solution on all integrators into hold position.
 - Compute** — applies problem voltages to all integrators.
 - Reset** — applies initial condition voltages to integrators.
 - Rep-Op** — places integrators into a repetitive operation cycle. Compute time variable from 5 msec to 10 sec.
 - Bal/Pot Set** — disconnects junction and grounds the input resistor summing junctions of all amplifiers. Each amplifier is converted to a gain of 2500 for precision monitoring of junction offset.
 - Pwer On/Off** — energizes and de-energizes computer.
 - Oven** — indicates + 28-volt oven power is on to maintain constant temperature of computing capacitors.
 - OL Hold, OL Reset** — lights up when any amplifier is overloaded. When depressed, computer goes into Hold; when released, normal operation is resumed.
 - Time Scale** — activates relays in each integrator module to change computing capacitor. ($x 10$, $x 100$, $x 1000$, depending on patchpanel connections.)
 - Slave** — permits operation of computer control circuitry from a second console.
- [9] Coefficient Potentiometers** — up to 24, available in groups of 6, featuring 10-turn wire-wound with lockable counting dials.

SD 40/80 Computer

The same compact design and high performance computing capability found in the small SD 10/20 computer is also available in the large SD 80 computer. With an expansion capability to 84 amplifiers (126 special order), 125 potentiometers, and a full set of digital logic control modules, the SD 80 provides a problem-solving power capable of handling large, complex problems normally associated with the simulation of dynamic electrical, mechanical, thermal and similar systems.

Yet, the large-capacity SD 80 fully retains the versatility of a desk-top computer. Mounted on a desk or table equipped with lockable casters, it can be easily moved to new locations. In addition to creating a highly compact and rugged computer, the all solid state design increases reliability, simplifies maintenance, and eliminates the need to operate the computer in an air conditioned laboratory environment.

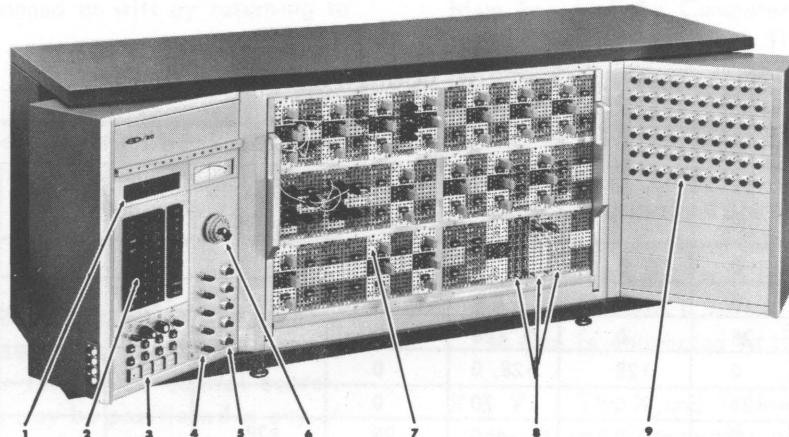
Programming a problem on a fully expanded SD 80 is virtually as simple as operating a small SD 10/20. Visual computer circuits in symbols everyone understands show at a glance how a circuit is patched. This feature saves time, reduces patching errors, and makes patching a task that the non-expert can perform.

To allow maximum operator control over a computer with a capacity as large as that of the SD 80, all computer controls are centralized in the left wing which

is movable to any convenient position to suit the operator. Major SD 80 controls that speed and simplify problem checking and solving include:

- A 199-point digital address pushbutton selector system for rapid access to amplifier outputs and potentiometer arms.
- Static and Dynamic checks by simple pushbutton control to verify patching and test time scaling rapidly.
- Instant alarm controls, both visual and audible, with automatic transfer to HOLD at the occurrence of an overload.
- Individual control of compute and reset times for precise adjustment in Iterative and REP-OP modes.
- Time scaling the problem with a single switch on the Control Wing.
- One pushbutton to check any amplifier's balance without repatching and to set any potentiometer.

The problem board is removable, just as it is with the SD 10/20. There is no restriction as to grouping of the modular patch panels in the overall problem board layout. In both the SD 10/20 and 40/80 computers, digital logic control is provided by modular plug-in units that slip into the central patchbay. They thus form an integral part of the computer and the main problem board, not a separate added-on element.



1. Digital Voltmeter — 5 digit readout, automatic ranging and polarity.
2. Address Selector Panel — with an amplifier and potentiometer address capability of 199 points.
3. Control Panel — for selection of:
Op. Modes: Hold, Compute, Reset and Rep-Op.
Master Time Scale Switch.
Compute Time: 5 msec to 5 sec.
Reset Time: 5 msec to 5 sec.
Slave: for remote control.
Oven and Power pushbuttons.
Check Mode switches for Static and Dynamic checks.
Overload Controls for: 1) audio alarm, 2) transfer to hold mode, 3) removal from Hold mode.
Amplifier Balance and Pot Set: changes all amplifiers to gain of 2500 for simplified monitoring of junction offset; and for proper loading of potentiometers.
4. Function Switches
5. Coefficient Potentiometers, No. 1-5
6. Null Meter Reference Potentiometer
7. Removable Patchboard. — An SD 80 board can hold up to 42 individual computing module panels. All modules are mounted in a universal patchbay directly behind the removable problem board. Nomenclature on patch panels matches textbook circuit diagrams. Operator can thus "see" the circuit while patching a problem.
8. Built-in Digital Logic Modules, consisting of Flip-flop, Gate, and Time/Event Control Modules.
9. Potentiometer Wing can hold up to 6 panels for a total of 125 potentiometers (first 5 pots are on Control Wing.)

Computer Logic

Each operating mode switch energizes a logic buss which is then distributed to the logic relays in the computing modules (primarily to the integrators). The logic levels in the Analog sections of the computer are:

$$\begin{aligned} +28V &= \text{True} = \text{logical 1} \\ 0V &= \text{False} = \text{logical 0} \end{aligned}$$

A logical one energizes a buss and places the computer in that mode of operation. All contradictory modes have their switches interlocked to prevent two modes of operation from being energized at the same time.

The table that follows outlines the logic busses and operating modes.

In most cases the nomenclature for each logic buss is self-explanatory. The following explanations should clarify the remainder of the logic nomenclature.

Logic Buss	Description
F. R. (R. O.)	<u>Forward Reset</u> (Repetitive Operation). This is the normal Reset buss and also is the Reset buss energized in the reset portion of Repetitive operation.
R. R. (R. O.)	<u>Reverse Reset</u> (Repetitive Operation). This buss is the logical complement of forward reset. It is used for complementary integrators and track/store circuits in iterative computation. Energized in the Compute mode of operation and in the Compute portion of Rep-Op.
F. R. (R. T.)	<u>Forward Reset</u> (Real Time). This buss is used to reset non-repetitive operation integrators. It is energized only when the reset switch is engaged.
F. H.	<u>Forward Hold</u> is the normal Hold buss. It is energized in the Hold mode of operation or if an overload occurs when the Overload Hold switch is engaged.
R. H.	<u>Reverse Hold</u> . Reverse Hold is the logical complement of F.R.R.T. It is used for complementary integrators and Track/Store circuits in iterative computation. Energized in the compute mode of operation and in Rep-Op.

Table 1. CONTROL LOGIC TABLE

LOGIC BUS	COMPUTER OPERATING MODE									
	POT-SET BALANCE	RESET	COMPUTE	REP-OP	MANUAL HOLD	PROBLEM HOLD	O.L. HOLD	STATIC CHECK	DYNAMIC CHECK	TIME SCALE CHANGE
Bal.	+28	0	0	0	0					
Pot-Set	+28	0	0	0	0		+28			
F.R.(R.O.)	0	+28	0	0, +28	0	0	0			
R.R.(R.O.)	0	0	+28	+28, 0	0	0	0			
F.R.(R.T.)	0	+28	0	0	0					
F.H. Bus	0	0	0	0	+28	+28				
R.H. Bus	0	0	+28	+28	0					
Static Ck.							+28	0		
Dyn. Ck.							+28	+28		
Time Scale										+28

(1) Blank indicates either 1 or +28 condition can exist, two values indicate cyclic operation.

(2) Manual Hold is actuated by pushbutton switch on Logic Control Panel.

(3) Problem Hold is actuated by application of +28V or greater Problem Hold Trunkline on Problem Board...

(Note: Will not be damaged by any voltage in computer range)

(4) O.L. Hold is actuated by operating "O.L. Hold" switch with an O.L. condition (either temporary or permanent) or any amplifier in the computer.

Use of the Mode Controls

The mode controls are of the push button type with interlocks to prevent two contradictory modes from being energized at the same time.

All control in the computer is accomplished in the integrators. Thus, when a mode is energized it opens or closes a relay in each integrator. Reference to the illustration, Integrator Block Diagram, on following page (6-6), while reading the following explanations will assist in understanding the control system.

Reset

The Reset or Initial Condition (IC) mode enables the integrators to be set to their initial values before starting computation. When the Reset relay is energized, an input and a feedback resistor are associated with the integrator amplifier making it, in effect, an inverter. Applying a voltage to the input resistor (in this case the IC terminal) will cause that input to appear at the output and to simultaneously charge the integrator capacitor to the same voltage.

Compute

In the compute mode there are no energized relays. The logic is such that if the computer is not in Hold and not in Reset, it is in compute.

Hold

In the Hold mode a normally closed relay contact is opened which disconnects the input signal from the amplifier. As there is no discharge path for the integrator capacitors the output will remain at its last computed value. In this manner outputs may be read or recorded and the problem resumed at will by returning to Compute.

Rep-Op

In the Rep-Op (Repetitive Operation) mode the computer is automatically switched between the Reset and Compute modes of operation. This mode enables the user to obtain repetitive solutions which are easily monitored on an oscilloscope.

The amount of time the computer stays in the compute mode is determined by the setting of the Compute Time switch located at the lower left of the control center. The outer dial of the switch may be positioned in any of three ranges: .005 to .1 sec, .05 to 1 sec, and .5 to 10 sec or in the External (EXT) position. The inner dial adjusts the compute time over the range selected.

The compute and reset times may be adjusted arbitrarily by observing the solution to the problem and adjusting the times until the desired display is obtained. If more precise compute and reset times are required the Compute-Reset cycle may be observed on an oscilloscope. Connect the oscilloscope to any of the FORWARD RESET terminals on the integrators or to Test Point 6 at the rear of the computer. A rectangular waveform will be observed. The positive (+28V) portion is Reset, the negative (0V) portion Compute. The Reset

and Compute time controls may then be adjusted to give the desired times.

If the Compute Time switch is placed in the External (EXT) position external circuits may be used to generate the Rep-Op cycle. The drive signals must be compatible with the logic used in the computer (+28V = True, 0V = False) and should be connected to the R (reset) and C(Compute) terminals located on the problem board.

OL Hold – OL Reset

When engaged, the Overload Hold portion of this switch will cause the computer to automatically switch to the Hold mode at the occurrence of an overload in any amplifier. When the overload is found and eliminated, the switch is re-engaged to reset the overload hold circuitry.

If an overload occurs, the light within the OL Hold /OL Reset will begin to blink on and off and will continue to do so until the overload is removed.

Time Scale

The Time Scale switch determines which of the integrator capacitors connected to the A and B terminals on the integrator modules is in the circuit. With the Time Scale switch not engaged, the capacitor connected to the A terminal is in the circuit. When the Time Scale switch is engaged the capacitor connected to the B terminal is in the circuit. Any two of the three available capacitors may be patched to the A and B terminals. If time scaling is not required the capacitors may be patched directly to the output.

Special Trunklines

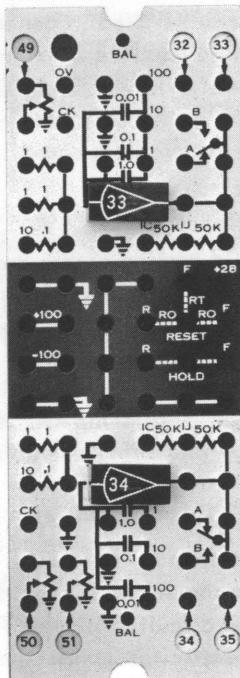
There are six special trunklines located on the problem board of the Computer. These are located on the top left row of modules. The first module contains the M, X and Y special trunk lines and the third module contains the H, C and R special trunklines. The special trunklines are used as follows:

M: This line is connected to the panel meter or the EXT jack as determined by the position of the Meter Select Switch. If the Meter Select Switch is in the METER function the M line is connected to the EXT jack; if the Meter Select Switch is in the EXT function, the M line is connected to the panel meter.

X & Y: The X and Y lines are connected directly to the X and Y terminals on the left side panel of the computer. These lines are intended for use with an X-Y plotter but may be used to connect signals to any external device or they may be used to connect external inputs into the computer.

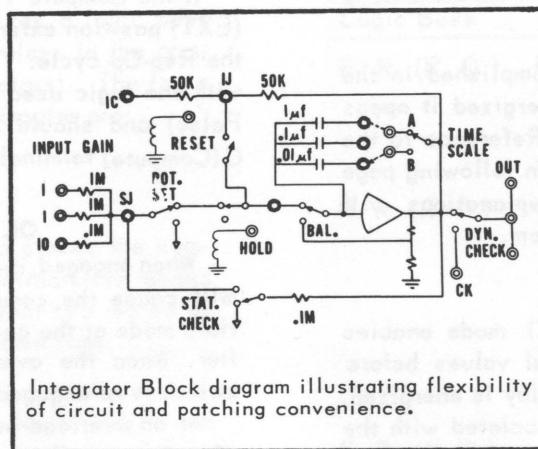
H, C & R: The H (Hold), C (Compute), and R (Reset) lines are used as external input lines to the Control Logic circuitry. A logical 1 applied to any of the lines will place the computer in that mode of operation. To use these lines the computer must be in REP-OP and the Compute Time switch must be in EXT.

Integrator Modules



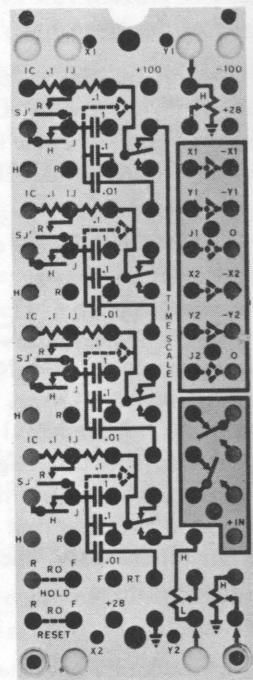
Model 3320

Dual integrator with two uncommitted operational amplifiers.



Integrator Block diagram illustrating flexibility of circuit and patching convenience.

Model 3329
Quad integrator, dual multiplier, operational relay. Contains four integrating networks, two four quadrant multipliers and one operational relay. Also available, less multipliers, Model 3329A.



Model 3320, Dual Integrator

The Model 3320 is divided into three sections: the top and bottom integrators and a logic section common to both. Basic patching of the integrator requires only a patch from the end of the integrating capacitor to the A terminal of the time scale relay. By patching the end of another capacitor to the B terminal a second time scale is set up and is controllable by the TIME SCALE switch on the control panel.

Initial conditions (IC) are applied at the IC terminal. The source of the IC voltage may be a pot, amplifier output, the reference voltage, or from an external source.

The logic section consists of a 7-terminal matrix near the center of the patching block. The nomenclature is the same as that explained in the Logic Control section of this manual. The two center terminals of the matrix are connected to the coils of the Reset and Hold relays. For normal and repetitive operation, patch from the F (forward) terminals to the relay coils. For reverse and iterative operation, patch from the R (reverse) terminals to the relay coils. For nonrepetitive operation, patch from the F(RT) terminal to the reset relay coil. (In this mode the integrators will reset ONLY when the reset button on the control panel is pushed.) Note that the logic patching controls both integrators in this module.

To use the integrator as a summer, patch from the input side of one of the resistors to the output of the integrator. This will connect a resistive feedback around the amplifier; the other resistors may be used as summer inputs.

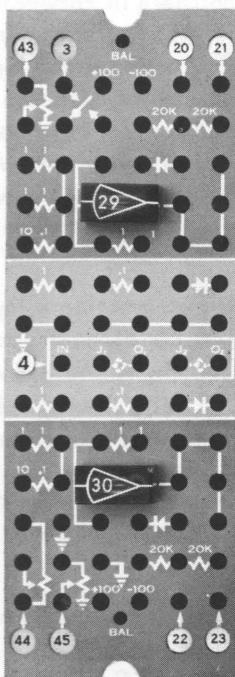
Model 3329 – Quad Integrator, Dual Multiplier, Function Relay

The Model 3329 has four integrating networks without operational amplifiers. The operational amplifiers necessary for integration are normally patched into the Model 3329 from an adjacent Model 3325. The input resistor network of the Model 3325 then becomes the input to the integrator. Relay logic connections are in the lower left hand section of the Model 3329 patchboard. Nomenclature is the same as that used on the Model 3320, Dual Integrator. Each Integrator Network has a Hold (H) and Reset (R) terminal associated with it, these terminals must be individually patched to the Relay Logic terminals. The Squid (multiple) patch cords should be used when more than one integrating network is to be used at the same time.

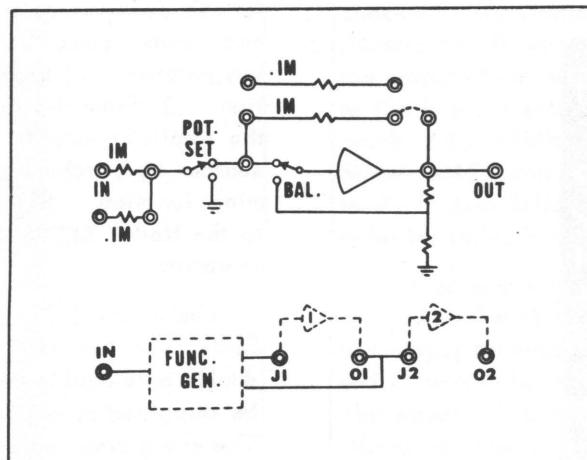
The Dual Multiplier section of the Model 3329 may be patched for division, squaring and square root functions as well as for multiplication. For multiplication or division operation, three operational amplifiers are required. In the division and square root functions arbitrary limitations as to polarity and relative amplitude of the inputs are imposed. For example, in division the Y input must always be positive and equal to or greater than the X input. Note that in square root an external diode must be patched into the circuit.

Patching and operation of the Function Relay in the Model 3329 is identical to that of the Function Relays in the Model 3322A. The relay is energized by a nominal +28 volt signal and will not be damaged by a signal of up to ± 100 volts.

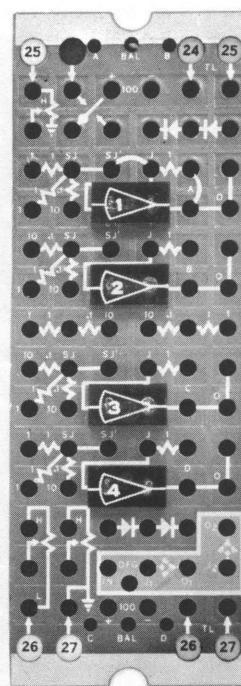
Summer Modules



Model 3321: Dual summer with two uncommitted operational amplifiers.



Model 3325: Quad summer, four summing networks, with four uncommitted operational amplifiers.



Model 3321 – Dual Summer

The Model 3321 contains two operational amplifiers and the components necessary for summing operations. Basic patching consists of connecting one end of the feedback resistor to the output. The top summer has three input resistors, the bottom summer two input resistors. There are four spare summing resistors located in the center section of the module which may be used as either input or feedback resistors with either of the summers.

The module also has two 20 kohm resistors and two uncommitted diodes associated with each summer. These may be used to patch the summer as a limiter-comparator or for any of the common diode circuits used in computation.

A function switch termination is provided on each summer module. The function switch is a single pole-double throw type switch and is physically located near the control center. Normally only the first three summing modules will have switches terminated in them.

The center section of the summing module has termination for a diode function generator. Use of these terminations and of the diode function generator is explained in that section of this manual.

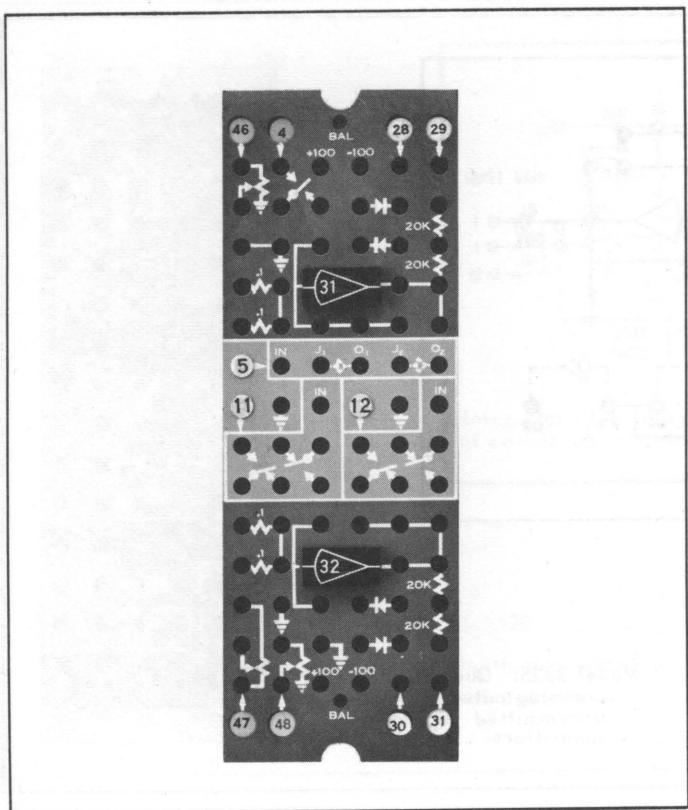
Model 3325 – Quad Summer

The Model 3325, Quad Summer is identical in performance to the Model 3321, Dual Summer. As illustrated, patching of the Model 3325 is, however, considerably different than that of the Model 3321. Basic patching for any of the four summers in the module consists of patching the SJ' terminal to the J terminal and patching the right end of the feedback resistor to any of the output terminals. Any of the resistor input networks may be used with any amplifier. Thus summers with six, nine or twelve inputs may be patched.

The Model 3325 also contains four spare resistors, four diodes and four trunk lines. In addition, it has terminations for a Function Switch and for a Diode Function Generator.

The Model 3325 is also used as a source of amplifiers for the Model 3329. When used in this manner, the Model 3325 furnishes the amplifier and input resistors for the integrator circuit.

Model 3322A – Dual Inverter, Dual Function Relay

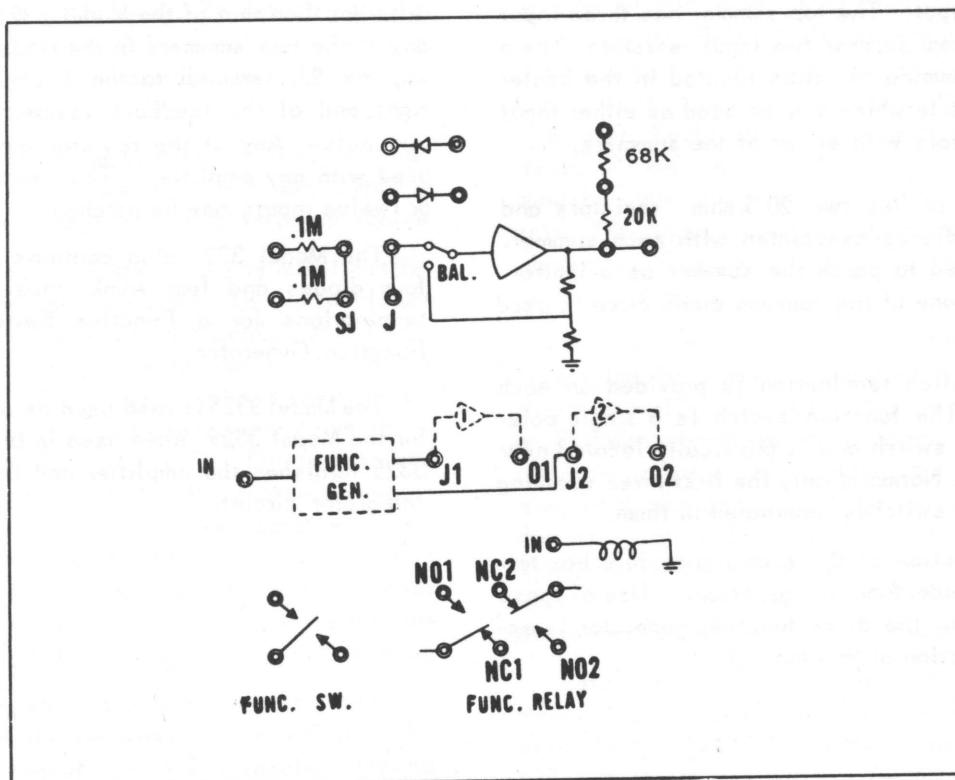


Model 3322A: Dual inverter, dual operational relay. Two uncommitted operational amplifiers with two operational relays.

The Dual Inverter contains two operational amplifiers for use as inverters or high gain amplifiers in comparator and limiter circuits. Operation as an inverter requires two patches: (1) From the summing junction to the junction. (2) From the input side of one of the resistors to the amplifier output. The inverter may be used as a summer by patching additional resistors into the summing junction. (Note: Potentiometers used as inputs to the Model 3322A should be set in the Reset mode of operation.)

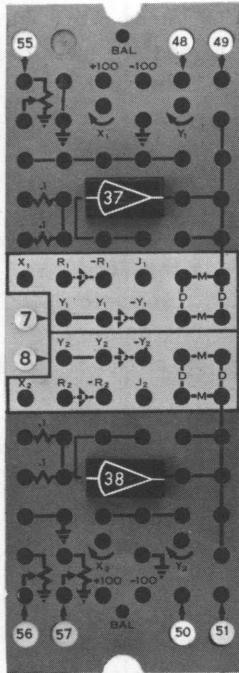
The Model 3322A also contains two function relays. Each is a two form C (2FC) relay corresponding to a double pole-double throw (DPDT) switch. The relays may be energized by any voltage from +28 volts to +100 volts. The energizing source is typically the output of an amplifier or a logic signal, but any source within the +28 volt to +100 volt range will energize the relays. Any voltage in the range plus and minus 100 may be applied without damage.

As in the Dual Summer, the Dual Inverter contains uncommitted diodes, limiter resistors, terminations for a Function Switch and Diode Function Generator terminations. The use and operation of these elements is the same as that described in the section on the Model 3321.



Block Diagram, Model 3322A

Model 3323 – Dual Inverter, Dual Multiplier



Model 3323: Dual multiplier, two four quadrant quarter square type multipliers with two uncommitted operational amplifiers.

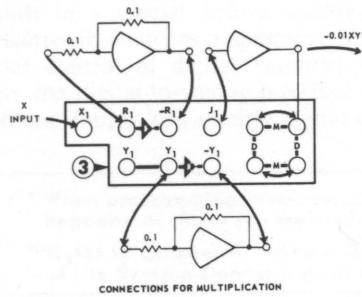
Multiplication Operation. The multiplier/divider is based upon the quarter-square principle, in which the following relationship holds:

$$\frac{1}{4} [(X + Y)^2 - (X - Y)^2] = XY$$

In ± 100 volt computers, the multiplier output voltage is always scaled to $-0.01XY$.

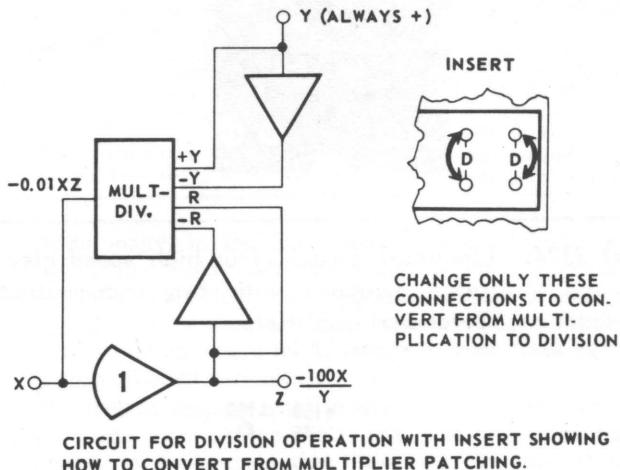
The circuitry consists mainly of two diode squaring networks, one based upon positive polarities and the other upon negative. For four-quadrant operation, voltages representing X , $-X$, Y , and $-Y$ must be present at the inputs to the squaring networks regardless of the polarity of input voltages.

The center section of the module contains the terminations for the multipliers. The section is divided into two identical halves, one for each multiplier. The multiplier will also perform division, squaring, and square root operations



Division Operation. Division is accomplished by the terminal connections shown in the figure below. The concept is illustrated by the circuit diagram in the same illustration. If the output of amplifier 1 is designated Z , the output of the multiplier must be $+0.01YZ$. By reference to the basic discussion on operational amplifiers (paragraph 2.8), it follows that $X = -0.01YZ$, or $Z = -100X/Y$. Since $+0.01YZ$ and Z must always be opposite in polarity to X , it also follows that Y must always be positive in order to maintain this relationship.

If Y occurs in the problem only as a negative voltage, apply it to the $-Y$ multiplier terminal and patch the input and output terminals of the inverting amplifier to the $-Y$ and Y terminals, respectively. Z still equals $-100X/Y$. The absolute value of Y must always be larger than X ; otherwise, the output would tend to be larger than 100 volts, causing an overload.



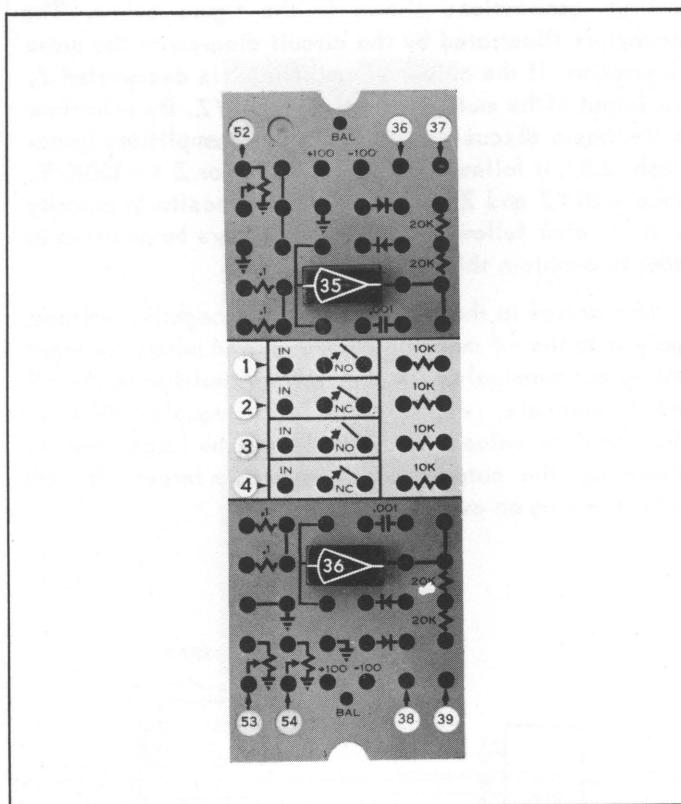
Squaring. Squaring is performed as a special case of multiplication where $X = Y$. Only one input inverter amplifier is required. Patch together the R and Y input terminals and the $-R$ and $-Y$ terminals. Leave the other connections as for multiplication. The output voltage taken at the amplifier output terminal represents $-0.01X^2$.

Squareroot Operation. The squareroot operation is a special case of division where $Z = Y$. The relationship $Z = -100X/Y$ of division becomes $Z^2 = -100X$ for $-100 \leq X < 0$, $Z = 10\sqrt{-X}$, for $0 < X \leq +100$, $Z = -10\sqrt{-X}$.

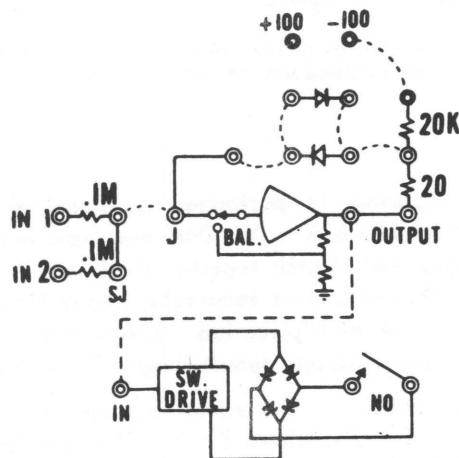
If X is always negative, patch together the R and Y terminals and the $-R$ and $-Y$. If X is always positive, patch R to $-Y$ and $-R$ to Y . Leave all other connections as for division.

The Inverter sections of the Model 3323 are similar to those described for the Model 3322A. The two basic patches required for inverter operation are: From the summing junction to the junction and from the input side of one of the resistors to the output of the amplifier.

Model 3324 – Dual Inverter, Quad Electronic Switch



Model 3324: Electronic switch, four high speed electronic switches with two uncommitted operational amplifiers.



Electronic Switch Block Diagram

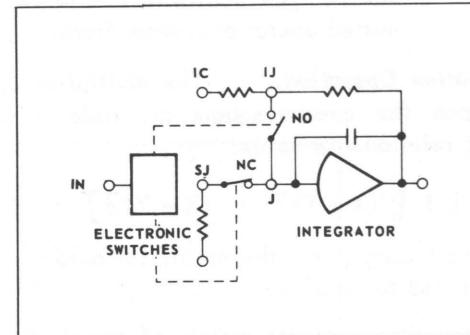
(Connections shown for comparator amplifier)

The center section of the Model 3324 contains four electronic switches. There are two normally open (NO) and two normally closed (NC) switches. The normal voltage required to excite the switch is a nominal +28 volts; however, voltages of up to ± 100 volts will not damage the switch. The current required to excite the switch is less than three milliamperes at 28 volts.

The Electronic switches were designed and are normally used to replace the reed type relays in the integrator modules. The Electronic switches may be used as normal switches as long as the input voltage and current through the switch limitations are observed.

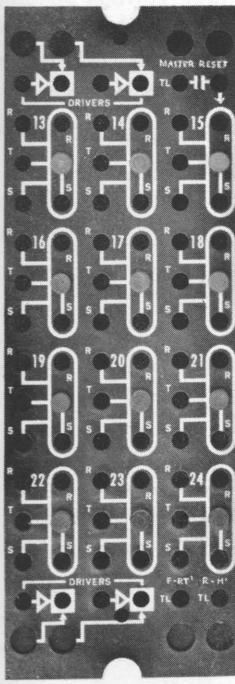
Patching to the Model 3320, Dual Integrator is shown in the accompanying illustration. Patching to the Model 3329, Dual Integrator is similar. Also included in the Model 3324 are two .001 mfd. capacitors and four .01%, 10 K ohm resistors for use in high speed Sample-Hold circuits.

The inverter sections of the Model 3324 are identical in operation and performance to those of the Model 3322A. Patching is identical to that of the Model 3322A and the instructions for that Model may be used.



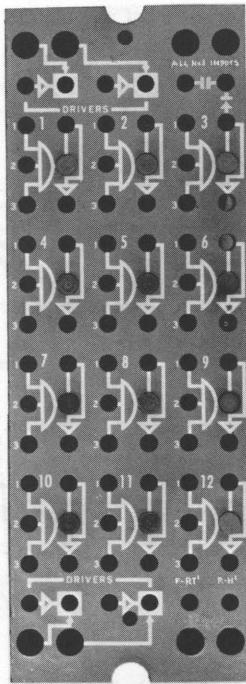
Integrator with Electronic Switches

Digital Logic Modules



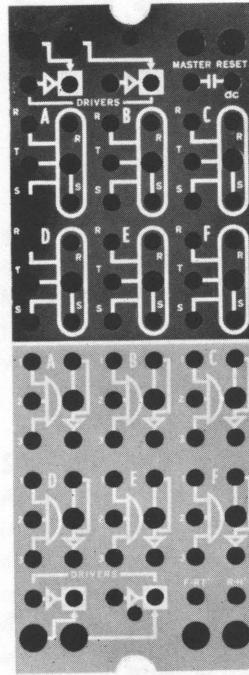
Model 3326

Twelve RST Flip-flops with four relay drivers.



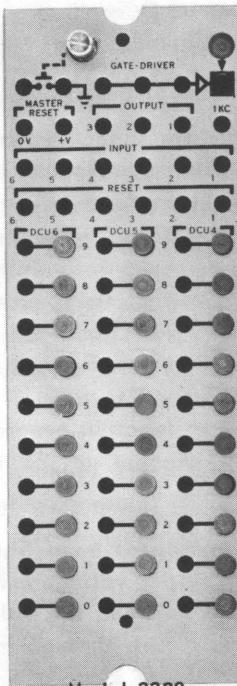
Model 3327

Twelve three-input AND Gates with four relay drivers.



Model 3326A

Six RST Flip-flops, six AND Gates and four relay drivers.



Model 3328

Time-event Control. 1 kHz clock with three decade dividers, three decade counters and one gated relay driver.

Digital control, the new way of multiplying the efficiency of an analog computer, can be included in the SD 40/80 as well as the small 10/20 computers. The SD hybrid computer expansion system is comprised of three types of compact plug-in modules.

The advantages gained by digital control in an analog computer are of far-reaching significance. Here are some important new advantages made possible by SD's Digital Logic Control:

1. Track and hold operation by individual integrators.
2. Sub-routines can be flexibly programmed at different speeds depending on decisions made by logical equations.
3. Program statements can be arranged into a flow chart quite similar to those used in digital computation.

The flexibility gained through this interplay of analog/digital equipment results in:

1. Better and greater problem-solving capacity.
2. Ability to solve a wide range of problems that before could not easily be handled by an analog computer.
3. Speed. Problem solving time is greatly reduced.

Through digital logic control, sub-routines start and terminate when the corresponding binary control variables change state as logical functions of:

1. External control (switches, relays controlled by external devices).
2. The states of timers or sub-routine counters.
3. Analog-comparator decisions.

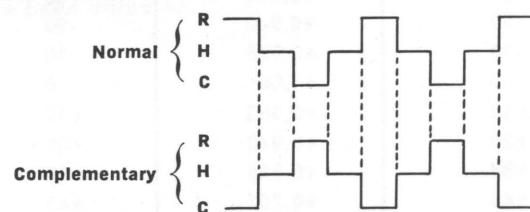
The interplay of binary control variables and analog computation results in a special *hybrid analog-digital* structure. Relays or electronic switches implement analog sub-routine changes under control of digital (binary) control variables and constitute the digital-to-analog interface of the computer. Analog solutions, in turn, can modify digital control.

Additionally, in the SD system it is possible to insert a delay of the Hold-mode command to the Complementary Integrators. This delay is important because it enables the C Integrators to store the final values of their inputs in a problem solution where the C Integrators are tracking rapidly changing problem variables.

The Reset-Compute-Hold modes are, of course, settable to any desired intervals, and it is this new degree of freedom imparted to integrators that illustrates the importance of digital logic control.

The combination of the three SD digital logic modules (Flip-Flops, Gates, Time/Event Control) results in a most flexible hybrid analog-digital structure. The operator can easily set all Reset-Compute-Hold intervals of integrators with the Time/Event Control Module.

A typical example that illustrates the use of SD's digital logic is mode control of iterative integrators. In iterative operation (IO), results obtained during or at the end of one solution of the problem are used to change parameters or the circuit configuration (switching) for the next solution. The following figure shows how in IO, integrators are paired into normal and complementary (opposite) logic to implement iterative solutions:



Mode Duty Cycle for Integrators in Iterative Operation

* When programming involves the use of digital control, logic levels for the digital logic elements are the opposite of those for the mode control and function relays.

* Refer to Chapter 8, "Basic Operation of Digital Logic Elements," for complete operating descriptions of the Systron-Donner digital logic control modules.

Function Generator Operation

Principle of Operation:

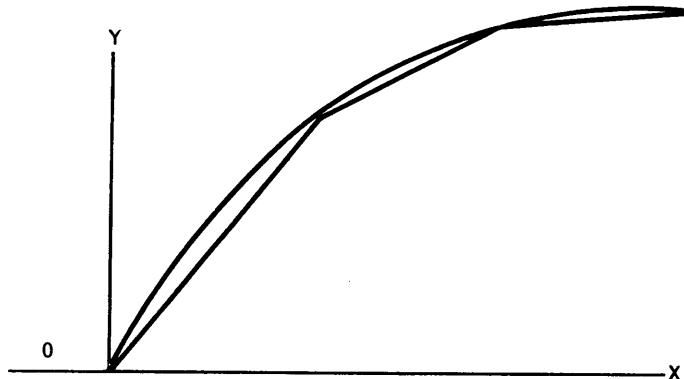
The Function Generator approximates the curve of an arbitrary function with a series of connected straight-line segments as shown in Figure 1A. This graph represents the sum of the individual line segments shown in Figure 1B: Each line segment corresponds to the output voltage of a diode network in the function generator. The point along the X axis where each line begins corresponds to the selected bias or "breakpoint" voltage. The diode networks are arranged so that six conduct when the X input is positive (0 to +100 volts) and six when the X input is negative (0 to -100 volts). For better accuracy, the input voltage may be biased to use all 12 networks above or below zero. Two function generator units can be operated as a single-channel of 24 segments for more accurate simulation.

The slope of each line segment corresponds to the ratio of output voltage per volt-change of input and can be varied between the values of +2 and -2 by adjusting the current flow through the diode network. For steeper changes of line slope at any point, two or more networks can be operated in parallel by shifting to the same breakpoint.

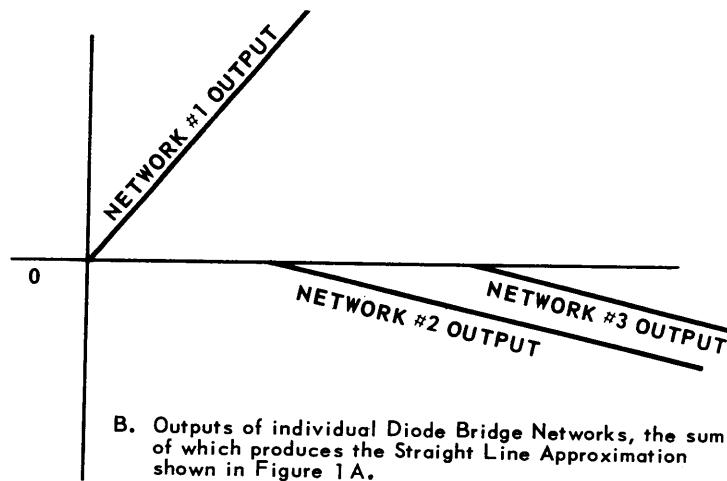
When the individual outputs of all diode networks are added together by an operational summing amplifier, the total output over the entire range of X-input will reproduce the original straight-line approximation. No output is produced when $X = 0$; therefore, a constant voltage, Y_0 , must be added at the output summing amplifier, when Y is not equal to 0 at $X = 0$.

Set Up Procedure: A function may be set up on the function generator either by the preparation of a table of values to which the individual slope and breakpoint potentiometers are adjusted, or by adjusting the poten-

meters so that a trace of the output voltage on an XY plotter will match a prepared graph as closely as possible (visual method). The second method may be faster but is limited in accuracy by the read-out instrument.



A. Straight Line Approximation of a Curve.



B. Outputs of individual Diode Bridge Networks, the sum of which produces the Straight Line Approximation shown in Figure 1A.

Figure 1. Simulation of a Function with Diode Network Outputs

TABLE 2. DATA FOR SETTING UP FUNCTION $Y = \text{COS}X$

X (DEGREES)	Y (COSINE)	X (VOLTS)	Y (VOLTS)	CONTROL* BREAK	SLOPE	POINT
-90	0	-90	0		-6	-6
-60	+0.500	-60	+50.0	-6	-5	-5
-45	+0.707	-45	+70.7	-5	-4	-4
-30	+0.866	-30	+86.6	-4	-3	-3
-20	+0.940	-20	+94.0	-3	-2	-2
-10	+0.985	-10	+98.5	-2	-1	-1
0	+1.00	0	+100	±1		
+10	+0.985	+10	+98.5	+2	+1	+1
+20	+0.940	+20	+94.0	+3	+2	+2
+30	+0.866	+30	+86.6	+4	+3	+3
+45	+0.707	+45	+70.7	+5	+4	+4
+60	+0.500	+60	+50.0	+6	+5	+5
+90	0	+90	0		+6	+6

* Arrows indicate sequence of adjustments.

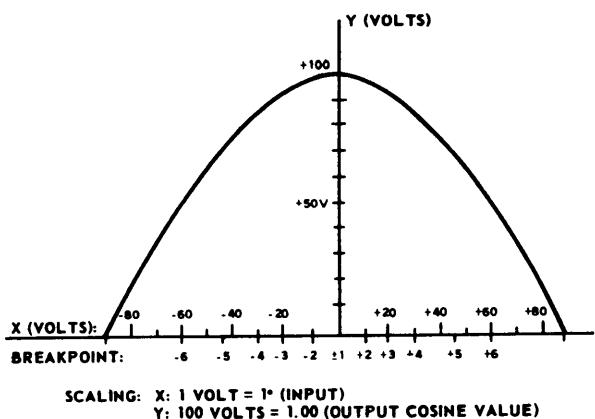


Figure 2. Typical Graph prepared for setting up the Function $Y = \cos X$.

Tabular Method (for Model 3352)

General: The maximum slope change for all segments, except the first, K_x, segment, is $\pm 2.5 : 1$. The first, K_x, segment has a maximum slope change of $\pm 5 : 1$. Breakpoints may be stacked to achieve greater slope changes.

Step 1: Set all Break-point/slope change (polarity) switches to the correct combination of polarities as determined by the function to be simulated. (See previous discussion on tabulating a function.)

Step 2: Set all Slope potentiometers to approximately center position. Set all Break-point potentiometers to one end as determined by the arrows associated with the polarity switches. (E.g., if the polarity switch is in the +BP, +ΔS or in the -BP, -ΔS position the break-point potentiometers should be set to the counter-clockwise end.) During this procedure the output amplifier may go into overload. Although no harm is being done, the computer may be placed in the Pot Set/Bal mode to prevent overload. The computer must be returned to Reset before proceeding with the adjustments.

Step 3: Set the K_x/Y_o switch as determined by the function and by the following instructions:

- The Y_o potentiometer sets the value of Y when X equals zero. This value may be set anywhere within the range of plus to minus 100 volts.
- The K_x potentiometer sets the slope of the first line segment. The K_x slope is adjustable within the range of plus to minus five (5).
- If the K_x/Y_o switch is placed in the Off position both Y_o and K_x will be zero.

- If the K_x/Y_o switch is placed in the K_x position Y_o will be zero, K_x will be adjustable over its range.
- If the K_x/Y_o switch is placed in the Y_o position, K_x will be zero, Y_o will be adjustable over its range.
- If the K_x/Y_o switch is placed in the K_x & Y_o position both K_x and Y_o are adjustable over their ranges.

When the switch has been set, the Y_o adjustment should be made. With X = 0, the potentiometer may be adjusted for the proper value of Y_o at the output of the Model 3352 (O₂). The Null voltmeter or a precision external meter should be used for this and all succeeding voltage adjustments in this procedure.

Step 4: The K_x adjustment should be made next. Apply the value of X at the first break-point (BP) to the input of the Model 3352. (If a potentiometer is used to obtain the X value its output should be applied through an inverter to the Model 3352. This will prevent errors due to the diode function generator loading the potentiometer.) Adjust the K_x potentiometer until the proper value of Y is obtained at the output of the Model 3352.

Step 5: The first break-point (BP) is now adjusted. Adjust the first break-point potentiometer until the K_x value previously set is offset by approximately 150 millivolts. (0.15 volts) This offset compensates for the diode characteristics and will help produce a more accurate function.

Step 6: Set the input to the value of X at the second break point. Adjust the slope potentiometer associated with the first break-point until the output is equal to the value of Y at the second break-point.

Step 7: Adjust the B.P. potentiometer associated with the next segment until the value of Y set in the preceding specification is offset approximately 150 millivolts. (See discussion in Step 5).

The remaining slope and break-point potentiometers are set alternately as indicated by the large arrows on the panel, by repeating Steps 6 and 7 until all segments have been adjusted.



CHAPTER 7

LOGICAL ALGEBRA

It is necessary to have a rudimentary understanding of logical algebra in order to use logic elements. The fundamentals of logical algebra are presented for this reason.

LOGICAL VARIABLES

A logical variable, L , has two values:

1 (true)

0 (false)

ADDITION (OR)

The logical sum, $L_3 = L_1 + L_2$, is defined by the following table:

L_1	L_2	L_3
0	0	0
0	1	1
1	0	1
1	1	1

This is commonly called the OR function and has the alternative symbology

$$L_3 = L_1 \cup L_2 .$$

MULTIPLICATION (AND)

The logical product, $L_3 = L_1 \cdot L_2$ is defined by the following table:

L_1	L_2	L_3
0	0	0
0	1	0
1	0	0
1	1	1

This is commonly called the AND function and has the alternative symbology

$$L_3 = L_1 \cap L_2 .$$

COMPLEMENT

The complement of L is denoted by \bar{L} . Loosely speaking, \bar{L} is the inverse of L . That is, whenever $L = 1$ or 0, then $\bar{L} = 0$ or 1, respectively. In manipulating logical formulas, the two obvious relations below are often useful.

$$L + \bar{L} = 1$$

$$L \cdot \bar{L} = 0$$

COMMUTATIVE LAWS

$$L_1 + L_2 = L_2 + L_1$$

$$L_1 \cdot L_2 = L_2 \cdot L_1$$

ASSOCIATIVE LAWS

$$L_1 + (L_2 + L_3) = (L_1 + L_2) + L_3$$

$$L_1 \cdot (L_2 \cdot L_3) = (L_1 \cdot L_2) \cdot L_3$$

DISTRIBUTIVE LAW

$$L_1 \cdot (L_2 + L_3) = L_1 \cdot L_2 + L_1 \cdot L_3$$

COMPLEMENTED SUM

$$\overline{L_1 + L_2} = \bar{L}_1 \cdot \bar{L}_2$$

This relationship can be seen from the following table:

L_1	L_2	$L_1 + L_2$	$\overline{L_1 + L_2}$	\bar{L}_1	\bar{L}_2	$\bar{L}_1 \cdot \bar{L}_2$
0	0	0	1	1	1	1
0	1	1	0	1	0	0
1	0	1	0	0	1	0
1	1	1	0	0	0	0

COMPLEMENTED PRODUCT

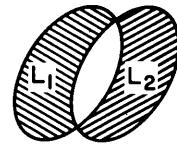
$$\overline{L_1 \cdot L_2} = \bar{L}_1 + \bar{L}_2$$

DE MORGAN FORMULAS

$$L_1 + L_2 = \overline{\overline{L}_1 \cdot \overline{L}_2}$$

$$L_1 \cdot L_2 = \overline{\overline{L}_1 + \overline{L}_2}$$

These can be obtained by complementing the complemented sum and product formulas.



In other words, the exclusive sum is the sum $L_1 + L_2$ minus the product $L_1 \cdot L_2$. A more convenient expression for \oplus can be derived.

EXCLUSIVE SUM

The exclusive sum, \oplus , of logical variables is defined by

$$L_1 \oplus L_2 = (L_1 + L_2) \cdot (\overline{L_1} \cdot \overline{L_2}).$$

This is also called the exclusive OR. $L_1 \oplus L_2$ is the shaded area of the "set" diagram below.

$$\begin{aligned} L_1 \oplus L_2 &= (L_1 + L_2) \cdot (\overline{L_1} \cdot \overline{L_2}) \\ &= (L_1 + L_2) \cdot (\overline{L_1} + \overline{L_2}) \\ &= L_1 \cdot \overline{L_1} + L_1 \cdot \overline{L_2} + L_2 \cdot \overline{L_1} \\ &\quad + L_2 \cdot \overline{L_2} \\ &= L_1 \cdot \overline{L_2} + L_2 \cdot \overline{L_1}. \end{aligned}$$

CHAPTER 8

BASIC DIGITAL LOGIC ELEMENTS

Digital logic can be included in an analog computer. It is used for making program decisions based on events occurring during the analog simulation. It is also used for program mode control. The application of these elements is discussed in other chapters of this handbook. This chapter describes the basic operation of the logic elements.

OPERATION OF DIGITAL LOGIC IN AN ANALOG COMPUTER.

Logic levels for the elements are

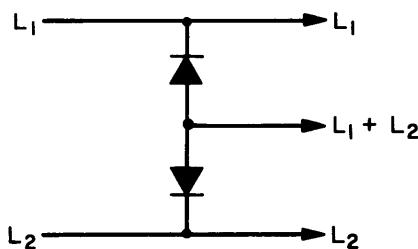
+28 volts = false (logical zero)

0 volts = true (logical one)

Most of the elements have a logically complemented output in addition to the normal output. All of the elements except the drivers and the dividers have status lights. When the light is on, the element is in the "true" state.

All of the elements, with the exception of the drivers, are designed so that their outputs can be connected together without damage. This feature is very important because it permits an OR operation by connecting two or more outputs to the same point on the patchboard. (See driver exception below.)

However, when two outputs are connected together in this way, they cannot be used as separate logical variables. To avoid this difficulty diodes can be used. Thus, if L_1 , L_2 are two logical outputs and $L_1 + L_2$, L_1 , L_2 are all required for logical inputs elsewhere in the program, the circuit below can be used



Each of the logic elements will drive any reasonable number of others. However, the logic elements are not capable of activating analog control relays or function relays. For this purpose drivers are supplied. The driver acts as both a power buffer and logical inverter. It should be noted that the logic levels for

mode control and function relays are the opposite or complement of those for digital logic. That is

0 volts = false (relay de-energized)

+28 volts = true (relay energized)

Thus, the driver not only delivers the current necessary to operate several relays but also provides the logic level inversion required. To use a driver, the output of the logic element is connected to the driver and the output of the driver to the relay input. These connections are made readily at the patchboard. It should be noted that the driver is a special element and its output cannot be used as an input to the flip-flops or DCU's. It can be used as a gate input.

Important: The driver outputs must never be connected to any other digital logic output. Connecting a driver output to ground (logical 1) will destroy the output transistor, that is, if connected to a gate, it should not be OR-ed with any other element.

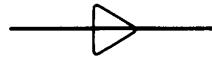
The remainder of this section describes the basic operation of the digital logic elements.

DRIVER

The driver is used as a power buffer and logical inverter from digital logic elements to analog mode control, function relays, and electronic switches. Its logic levels are

INPUT	OUTPUT
0 (false)	0 volts
1 (true)	+28 volts

The program symbol is

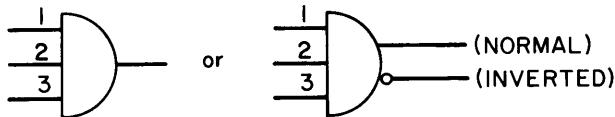


GATE

The logical gates are of the coincidence type. With the exception of input number 1, they operate as AND gates only on the inputs connected. That is, if no logical variable is connected to one of the inputs at the patchboard, then this input is essentially de-activated. Note that this is the same as connecting a logical 1 to the input.

Input number 1 is different because it must be connected; otherwise the gate will not operate. However, if only input 1 is connected, the gate will act as an inverter if its complemented output is used.

The program symbol for the gate is



where the input numbers are shown.

The truth tables below will clarify the operation of the gate.

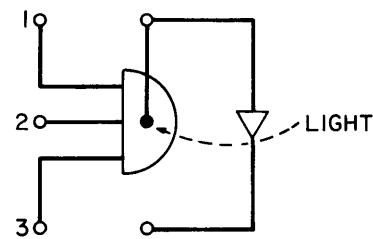
INPUTS			NORMAL OUTPUT	INVERTED OUTPUT
1	2	3		
none	any	any	0	1

INPUTS			NORMAL OUTPUT	INVERTED OUTPUT
1	2	3		
0	none	none	0	1
1	none	none	1	0

INPUTS			NORMAL OUTPUT	INVERTED OUTPUT
1	2	3		
0	0	none	0	1
0	1		0	1
1	0		0	1
1	1		1	0

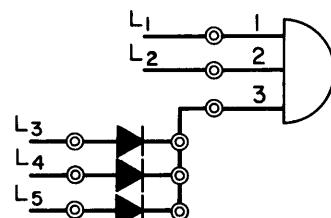
INPUTS			NORMAL OUTPUT	INVERTED OUTPUT
1	2	3		
0	0	0	0	1
0	0	1	0	1
0	1	0	0	1
0	1	1	0	1
1	0	0	0	1
1	0	1	0	1
1	1	0	0	1
1	1	1	1	0

The gate is shown on the patchboard as



The true state response to inputs 1 and 2 is delayed by 50 usec, while input 3 is not. As will be seen in Chapter 9, covering logic applications, this permits the gates to be used in connection with flip-flops to form a shift register. Also, this feature frequently obviates the necessity of using one-shots for delays in a logic program.

Each gate has three inputs on the patchboard. If the logical product of more than three variables is desired, the additional elements can be connected in through externally patched diodes, as shown below.

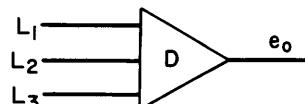


The additional elements can be connected to any of the three patchboard inputs, except number 1.

The gate can also be used for pulse-shaping. It is ON (logical 1 at the output) when no input voltage is greater than 4 volts and OFF when any input is greater than 6.5 volts. When the net input is between 4 and 6.5 volts, the output corresponds to the last state of the gate.

GATE-DRIVER

This is a special element found in the TIME/EVENT CONTROL MODULE. It is a power buffered, logically inverted AND gate which performs the combined function of an AND gate and driver. The program symbol is



where L_1 , L_2 and L_3 and e_o are respectively logical input variables and the output voltage. This gate is the same in logical operation as the AND gate except there is no delay introduced at any input. When $L_1 \cdot L_2 \cdot L_3 = 1$, the output of the gate is 28 volts; otherwise it is zero.

The primary use of this gate is for implementing analog mode control changes based on logic outputs. For example, in this way a multiple decade counter can be used either to stop the analog computer after a prescribed number of runs or in conjunction with a clock to specify precisely the length of time the analog computer is in one of several modes. This is accomplished by patching the output of a gate-driver to either the C, R or H inputs on the patchboard.

DECade Counting Unit (DCU)

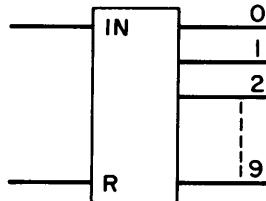
The DCU is either a decade divider or a decade counter. It contains ten internal binary states which correspond to integers from 0 to 9. Note that this is different than a binary counter which counts in the binary number system. There are two types of DCU's in the computer. Both of these are in the TIME/EVENT CONTROL MODULE (also referred to as Counter Module). One has all the integer binary states available as a "ten line" output. This element is called a decade counter. A status light is associated with each of the integer binary states. The other type has only the decade output available and is called a decade divider.

There are two master resets for each group of 3 decade counters together with 3 decade dividers. These are the 0V and +V terminals. The 0V input is directly coupled to the clock. As long as a logical one is applied to this terminal the clock will be stopped. A state change from zero to one will cause all the counters and dividers to be reset. As long as a logical one is maintained at this terminal, all other reset signals are inhibited.

A state change from logical 1 to 0 at the +V terminal will cause all the counters and dividers to be reset. (This is called a trailing edge trigger.) The +V input does not inhibit other reset signals, and it does not turn off the clock.

All counter or divider inputs (including reset inputs) require a change from a logical 0 to a logical 1 for operation. Thus, the application of a logical 1 to an input will cause it to operate once. It will not operate again until the logical one is removed and reapplied. This is true also for the master resets for each counter/divider group.

The program symbol for the decade counter is



The counter outputs are the ten integer counts from 0 to 9. Whenever a logical 1 is applied to the reset terminal, the counter will return to count 0. Whenever a logical 1 is applied to the input terminal, the counter will advance to the next higher count. The counter is cyclic: when it is in count 9 and receives an input pulse, it returns to count 0.

Note that the counters can be used in series. This is accomplished by connecting any output from the first counter to the input of the second counter. Then, whenever the transition from a logical 0 to a logical 1 occurs at the output of the first counter, this transition from 0 to 1 is applied to the input of the second counter. This causes the second counter to advance one count. Normally, the second counter receives its input from the "0" digit of the first. Then, when a transition from count 0 to count 1 occurs in the first counter, the output from count 0 goes from a logical 1 to a logical 0 and the second counter will not be advanced in count: its input is activated only with a logical change of state from zero to one. Note that when the first counter goes from count 9 to count 0, then the count 0 output goes from 0 to 1, and the second counter is advanced.

The program symbol for the decade divider is



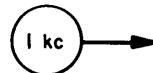
The divider has ten internal integer counts in the same way as the counter. However, these are not available as logical outputs. An input to the reset terminal will cause the divider to return to count 0 internally. The output will only return to state 0 if it is in state 1. If the logical variable applied to the input terminal changes state from 0 to 1, then the divider will count internally. The divider is cyclic: count 0 follows count 9.

With a sequence of input pulses, the output is a logical 1 except when the internal counter is in count 8 or count 9. Thus, when the internal transition from count 9 to count 0 occurs, there is a transition from a logical 0 to 1 at the output which can be used as input to other logic elements. However, it should be remembered that if the divider is reset during the time when internally it is in count 8 or count 9, the transition from 0 to 1 will occur at the output then.

CLOCK

Each TIME/EVENT CONTROL MODULE provides a 1000 cycle clock output at the patchboard. Maintaining a logical 1 at the 0V master reset terminal will stop the clock.

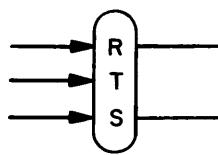
The program symbol for the clock is



The logical cycle of the clock is repetitively a logical 1 for 800 μ sec and logical 0 for 200 μ sec.

FLIP-FLOP

A flip-flop (FF) is a one-bit memory. It has a set (S) and a reset (R) state. The set state corresponds to true (logical 1) and the reset state to false. The program symbol is



There is a status light connected to the S output so that when the FF is in the S state the light will be on. The FF can be placed in the S state by applying a logical 1 to the S input. It can be placed in the R state by applying a logical 1 to the R input. Applying a logical state change from 0 to 1 to the T input will cause the FF to change to the opposite state. The flip-flop R and S inputs are direct-coupled. Each can be a-c coupled by inserting a 200 pF capacitor in series with the input so that these inputs will be activated by only logic state changes from 0 to 1 and not by logic levels.

Internally, the application of a logical 1 to the S input turns off the R output, and a logical 1 at the R input turns off the S output. The R and S inputs are level sensitive in contrast to the T input which is activated by state changes. Consequently, if a logical 1 is applied to both the R and S inputs simultaneously, both the R and S outputs will be a logical 0, and a state change at the T input will have no effect on the state of the FF. If both R and S inputs are a logical 1 and then both are changed to a logical 0 simultaneously, the final state of the FF will be either R or S with nearly equal probability. If the change to 0 does not occur simultaneously, the input which sees the logical 1 last will control the state of the FF.

If either the R or S input is energized by a logical 1 and if a pulse or logical variable state change is applied to the T input during this time, the FF will act like a one-shot mono-stable multivibrator. That is, it will make a transition to the opposite state for about $5 \mu\text{ sec}$.

Two master reset terminals are located at the patch-board for each FF module. One will put all the FFs in the R state for a transition from 0 to 1. The other reset is level sensitive. As long as a logical one is applied all the FFs are constrained to be in the R state.

COMPUTER CONTROL LOGIC OUTPUTS

The analog control logic outputs in the integrator module must not be used as an input to any logic element except a gate. Therefore, special F-RT and R-H outputs are located in the FF and gate modules. These can be used as an input to all of the logical elements. They are identified as F-RT' and R-H'. The logic levels for both F-RT' and R-H' are consistent with the digital logic elements. There is a logical inversion between F-RT and F-RT' as well as R-H and R-H'. For example, when F-RT' is connected to a flip-flop R input, the flip-flop will be in the reset state when the F-RT bus is energized. When the F-RT bus is de-energized the FF is free to respond to signals at its T and S inputs.

The F-RT and R-H busses are logical complements, which is the reason for their selection in logic control.

LOGIC SWITCH

There is a manual logic switch in the counter module which is convenient for single-stepping programs. The output of this logic switch (ungrounded terminal) cannot be connected to any other logic output. The reason for this is because the output of the logic switch is shunted internally by a large capacitor. Consequently, if this element is connected to another logical output it will cause degradation of the rise-time of that element.

CHAPTER 9

CIRCUITS FOR SIMPLE LOGICAL FUNCTIONS

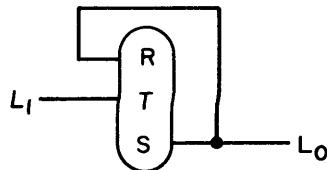
Several fundamental circuits which are used frequently for logical programming are presented below. In all cases the trigger is a negative-going voltage.

ONE-SHOT

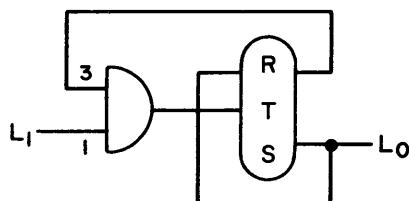
A one-shot circuit will produce a logical pulse of predetermined length whenever its input is triggered. That is, before the trigger, its output will be a logical zero. When the trigger occurs its output changes to a logical one for a fixed time interval. A one-shot can be constructed with logic elements in several ways.

5 μ sec:

The circuit below will provide a 5 μ sec logical pulse when triggered.



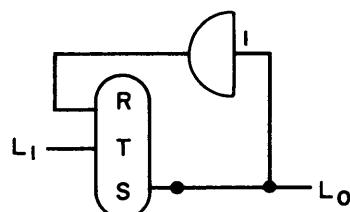
If a second trigger occurs during the pulse, then the pulse will last for 5 μ sec after the second trigger. In other words, after the one-shot is turned on by the trigger, it will stay on until 5 μ sec after the last pulse. The sensitivity to multiple triggering can be eliminated with a gate:



The logical pulse from this circuit will be delayed 50 μ sec due to the delayed input of the gate used for L_1 .

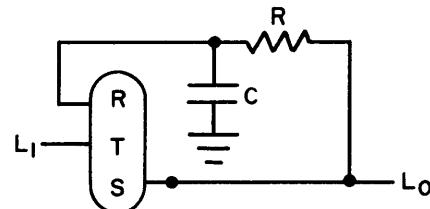
50 μ sec:

The circuit below will provide a 50 μ sec pulse when triggered.

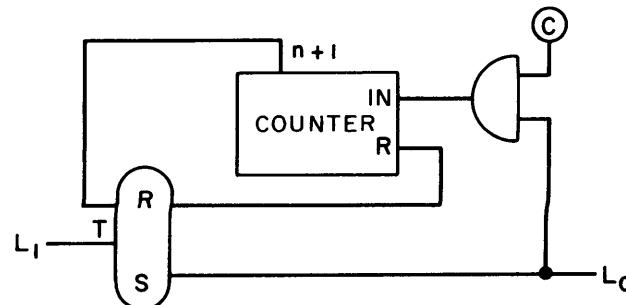


This is due to the 50 μ sec delay at input 1 of the gate. Its behavior for multiple triggering is the same as for the 5 μ sec one-shot.

Variable:



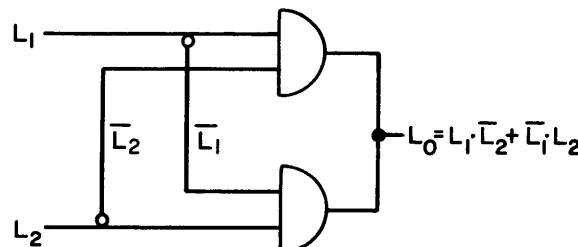
This one-shot will produce a logical pulse of fixed duration. The pulse duration depends on the choice of R, C. Its behavior for multiple triggering is sometimes erratic. The sensitivity to multiple triggering can be eliminated by gating.



This one-shot will produce a logical pulse whose duration is determined by both the clock frequency and the output line number from the counter. The pulse duration for the circuit shown is n times the clock period. Obviously several dividers may be inserted in series with the clock if desired and several counters may be used. This circuit is insensitive to multiple triggering.

HALF-ADDER

A half-adder is a circuit for implementing an exclusive sum, \oplus :



Although digital inverters are shown in this circuit, they are not usually necessary because all logical elements except for the counters have both complemented and non-complemented outputs.

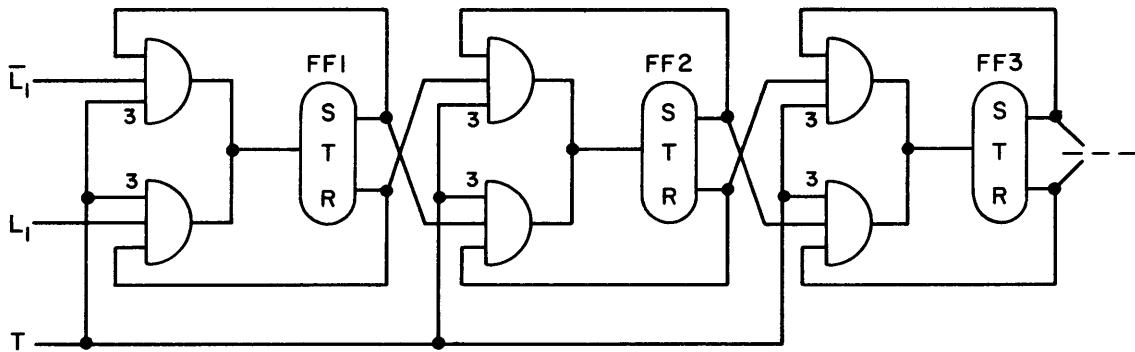
SHIFT REGISTER

A shift register is a binary register which is shifted one bit each time a trigger is applied. The triggered half-adder is needed for this circuit shown at right.

If $L_1 \oplus L_2 = 1$, then whenever a trigger, T, is applied, L_0 will be a logical pulse whose length is equal to the duration of T. Note that T is applied to input 3.

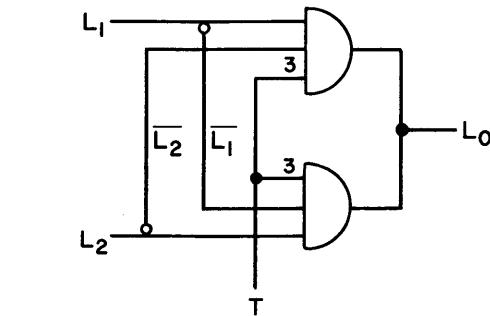
If the duration is less than 50 μ sec then changes of L_1 , L_2 after the application of T will have no influence on L_0 due to the built-in delay for gate inputs 1 and 2.

The circuit for a shift register is



Notice that the inverters for the triggered half-adder are not required since complemented inputs are available. An initial setting for this register can be entered by resetting the flip-flops individually, or with the common reset, and subsequently gating a logical one into each S input as required. This circuitry is not shown because it requires little explanation. Whenever a trigger, T, is applied which is less than 50 μ sec (a 5 μ sec one-shot is a convenient source) the following state transfers will be made

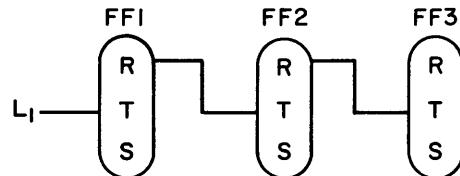
$$\begin{aligned} L_1 &\rightarrow FF_1 \\ FF_1 &\rightarrow FF_2 \\ &\vdots \\ &\vdots \\ FF_{n-1} &\rightarrow FF_n \end{aligned}$$



Thus, when each trigger occurs, the current value of L_1 is entered into the first flip-flop. The last flip-flop can be connected to the first flip-flop, instead of L_1 , to obtain a register with an end-around shift.

UP-COUNTER (BINARY)

The circuit for the binary up-counter is (shown for three bits)



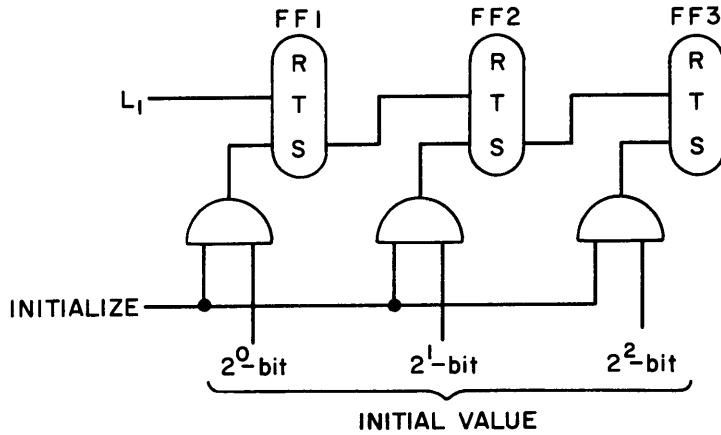
The counter is initialized with the common reset. Alternatively, an initial value can be entered in the same manner as for the shift register. This circuit is activated by a logically positive change of state. It will count the number of positive changes of state in the binary system. Each time FF_{n-1} returns to the R state, the state of FF_n is changed. The table below shows the states of the FF's for successive logically positive changes of state of L.

FF NUMBER			Number of State Changes for L ₁
3	2	1	
0	0	0	0
0	0	1	1
0	1	0	2

FF NUMBER			Number of State Changes for L ₁
3	2	1	
0	1	1	3
1	0	0	4
1	0	1	5
1	1	0	6
1	1	1	7
0	0	0	8

DOWN-COUNTER (BINARY)

The circuit for the binary down-counter is (shown for three bits)



This circuit is activated in the same way as the up-counter. The table below shows how it counts. (It is assumed that the initial value is the binary number 111.)

FF NUMBER			INPUT STATE CHANGES
3	2	1	
1	1	1	0
1	1	0	1
1	0	1	2
1	0	0	3
0	1	1	4
0	1	0	5
0	0	1	6
0	0	0	7
1	1	1	8

TIME INTERVAL MEASUREMENTS USING DIGITAL LOGIC

The digital logic modules, which are part of the Systron-Donner Analog Computer complement, can be used to obtain measurements of time intervals with a resolution of up to one millisecond. The overall timing accuracy depends on the number of digits displayed. It can be as good as .01%, which is a significant improvement over the typical 3% accuracy obtained through the use of an oscilloscope. Also, this particular capability of using digital logic modules to make accurate time interval measurements can eliminate the need for a frequency counter, which is generally an extra cost accessory.

The basic time interval measuring program is given in Figure 1A, and the timing relationships are shown in Figure 2. This program measures the duration of a logic signal, L. Initially, all flip-flops and DCUs (Decimal Counting Units) are in the Reset state. When, and only when L changes from 0 to 1, Flip-Flop 1 is toggled into the Set condition. Gate 3 is opened and the clock timing signals are sent to the Time Interval Indicator. This is a cascade of DCUs of sufficient quantity to measure the total time interval. At the end of the time interval being measured, L returns to the 0 state. \bar{L} from the inverted output of Gate 1 resets F-F1. F-F1, in turn, toggles F-F2 into the Set state and turns off Gate 1. The Time Interval DCUs stop counting and hold the elapsed time display. The Set output of F-F2 inhibits the output of Gate 2 through Diode 1, preventing any further changes in L from toggling F-F1. F-F2's Set output also opens Gate 4, and clock signals are sent into the Display Time DCUs.

The Display Time Generator is also a cascade of DCUs. Outputs from the Display Time Generator are logically summed in Gate 5. When the count reaches the value corresponding to the desired display time, Gate 5 is changed to the 1 state and toggles F-F3. F-F3 is connected as a 5 microsecond one-shot. That is, F-F3 goes into the Set state for 5 microseconds before returning to the Reset state. The Set output of F-F3 resets all flip-flops and DCUs, and the circuit is ready to measure another time interval. The automatic Display Time Generator can be dispensed with, and the system can be manually reset by using the logic switch on the clock module (Model 3328).

If L changes state before the end of the display time, nothing happens due to the Inhibiting action of F-F2 and D1. If the reset action at the end of the display time occurs while L is in the 1 state, the system remains in Reset. Interval measurement will not commence again until L changes from 0 to 1. This is illustrated in the Timing Sequence diagram, 3rd interval time.

To summarize, this program takes no action until L changes from 0 to 1. It then measures the current time interval, holds the display until automatically or manually reset, and then waits for the next state change from 0 to 1 in L.

An alternate program is shown in Figure 1B for measuring the time interval between two independent logic

signals. L_a initiates the timing action. Gate 2 acts as a buffer so that L_a is not affected by the inhibiting action through D1. L_b terminates the time interval when it goes into the 1 state. The interval terminates at that time independently of the state of L_a . If L_b causes trouble by being in the 1 state during part of the interval to be timed, it can be connected to a one-shot similar to F-F3 and then into the Reset input of F-F1.

The program of Figure 3 allows measuring of the Nth time interval after reset. It is identical to the program of Figure 1 except that a pre-set event counter is inserted between L and the toggle input of F-F1. The event counter can be made of DCUs, or a cascade of flip-flops.

The period of N intervals can also be measured, by inserting the pre-set circuit between L and the Reset input of F-F1. This is useful for measuring equal-period intervals with an effective resolution of 1/N milliseconds, or for averaging readings over several cycles.

Figure 4 shows how to convert analog signals to logic levels. The logic gates can be operated from analog inputs; the output will conform to logic states. However, the gates have hysteresis. That is, the gate is off for inputs greater than about +8 volts. Once opened, the gate will not turn on until the input becomes less than about +4 volts. Negative inputs to the gates cause no action. Also, the gates are not damaged by input voltages within the ± 100 volt computing range.

Gate B is turned on at the start of the time interval period, thereby toggling F-F A into the Set state. Gate B turns off during the interval, but this has no effect on F-F A. When Gate B turns on again, F-F A resumes its Reset state and the timing is terminated.

Figure 5 indicates the method used to convert integrator mode logic bus signals to logic levels. Gate input 1 must be grounded or connected to a logical 1. When the mode bus is energized, it turns the gate off. L then is taken from the inverted output of the gate since this is a logical 1 when the mode logic bus is energized.

Display of the less significant digits of a time interval (when the more significant digits are known), can appear on the Model 3328 which allows only a 3-digit display. A 3-digit display of the less significant digits will suffice when the more significant digits are known. For example, to read an 11 second interval to a resolution of 1 millisecond requires a 5-digit display. If the first two digits are already known, the last 3 can be indicated on the clock. The counters are cyclic, so overflows of the more significant-digit counters do not affect the count in the less significant digits.

As another example, the period of a 1 radian per second sine wave can be measured to 1 millisecond resolution, and .015% accuracy, without displaying the most significant digit since the period is known to be 6. + seconds. The display would indicate the .283 part of the 6.283 second period.

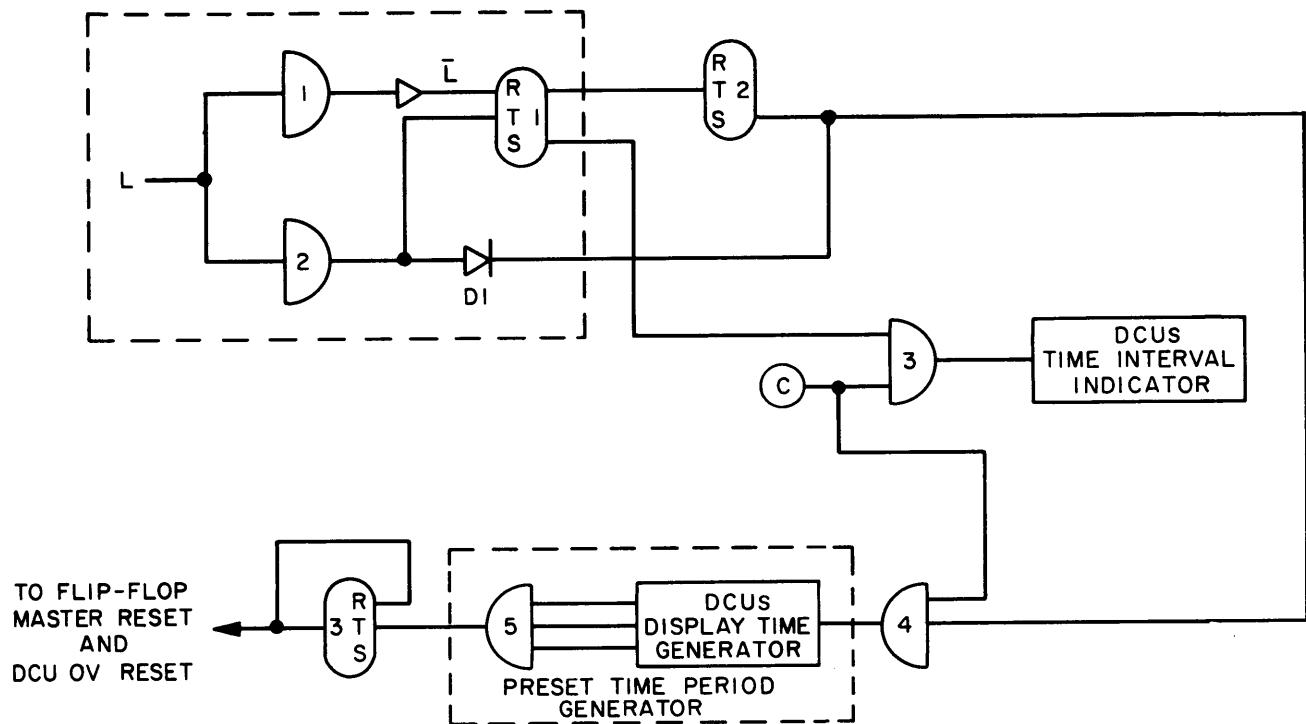


Figure 1A. Time Interval Measurement Program

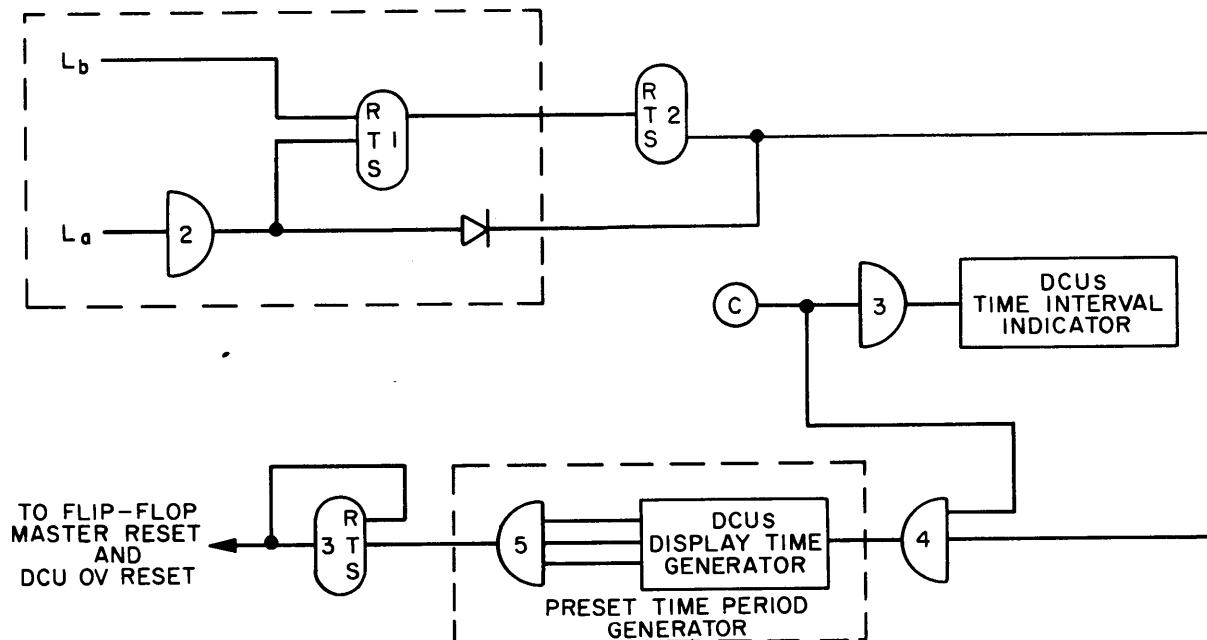


Figure 1B. Alternate for Interval Between Two Logical Events

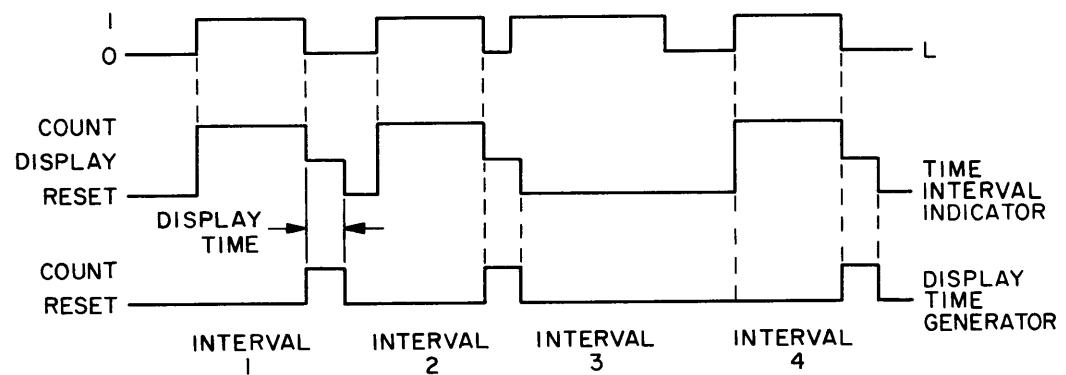


Figure 2. Timing Sequence

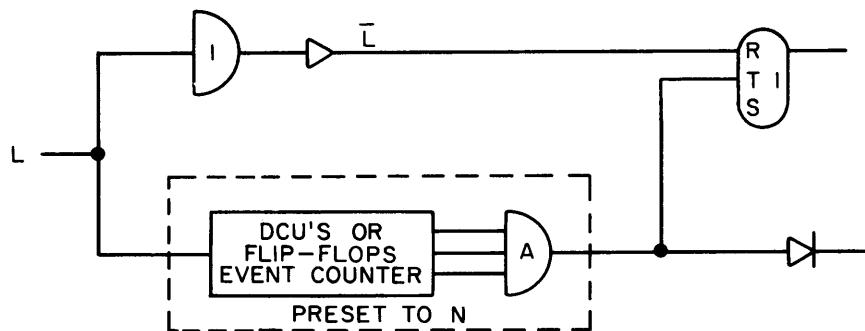


Figure 3. Alternate To Measure Nth Interval After Reset

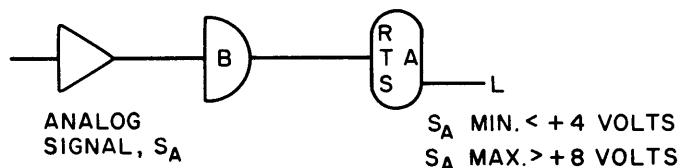


Figure 4. Program to Convert Analog Signal to Logic Signal

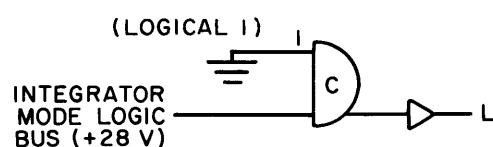


Figure 5. Program to Convert Integrator Mode Logic (+28V) Bus to Logic Signal

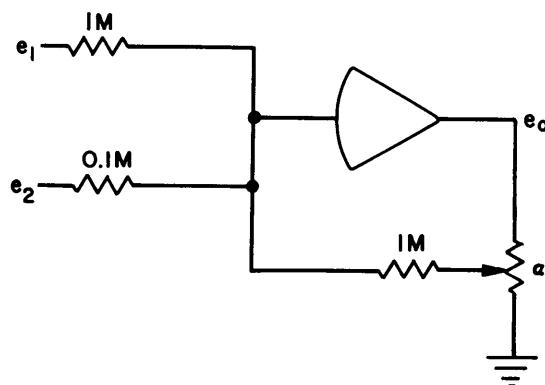
CHAPTER 10

CIRCUITS FOR SIMPLE LINEAR AND NON-LINEAR FUNCTIONS

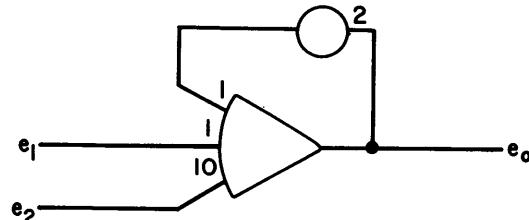
Several simple operations are used frequently for programming. They include division by a constant, differentiation, time delay, and a variety of sectionally linear transfer functions. These are described below. (It should be remembered that all electronic switches in this section must be used as J inputs in series with the input resistor.)

DIVISION BY A CONSTANT

The circuit



The symbol



satisfies the condition

$$\alpha e_0 + e_1 + 10e_2 = 0.$$

(The input gain labels are usually omitted if they are unity.) The transfer function is

$$e_0 = -\frac{1}{\alpha} e_1 - \frac{10}{\alpha} e_2.$$

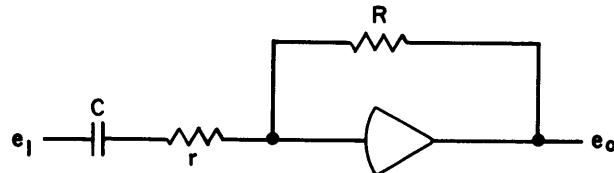
This circuit is useful for dividing a sum of variables by the same constant, α . Sometimes it is more convenient to set a pot to α , if α is a natural parameter in the program. Otherwise the pot would be put between the amplifier output and the following input and would require the calculation of α^{-1} for each new setting.

This configuration is also useful for obtaining gains of greater than 10 without requiring more input resistors or more amplifiers.

DIFFERENTIATION

Programs are usually written to avoid differentiation. This is done because differentiation inherently stresses noise and because the circuits used may lead to instability. However, in some instances it is difficult to avoid forming derivatives. This is especially true for on-line data processing or editing applications. For this reason, it is helpful to be familiar with available differentiation techniques.

A single amplifier program which can be used is

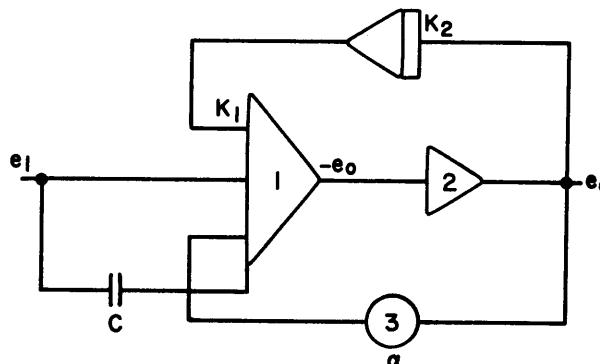


which has the transfer function

$$\frac{e_0}{e_1} = RCs \cdot \frac{1}{rCs + 1}.$$

r is usually about 10K producing reduced gain at frequencies above $(2\pi rC)^{-1}$ cps. Since C is generally large, the circuit tends to load unduly the preceding computer element. Also, a large C may lead to stability problems in the amplifier driving the differentiator. The small resistor, r , helps prevent drive circuit instability.

The more sophisticated program



has the transfer function

$$\frac{e_0}{e_1} = \frac{s}{K_1 K_2} \cdot \frac{1 + Ts}{1 + \left[\frac{(1-\alpha)}{K_1 K_2} \right] s}.$$

where $T = CR_{in}$. R_{in} , the input resistance at e_1 , is usually $1 M\Omega$. Thus, expressing C in microfarads, T (sec.) = $C(\mu F)$.

Ideally,

$$\alpha = 1 - K_1 K_2 C$$

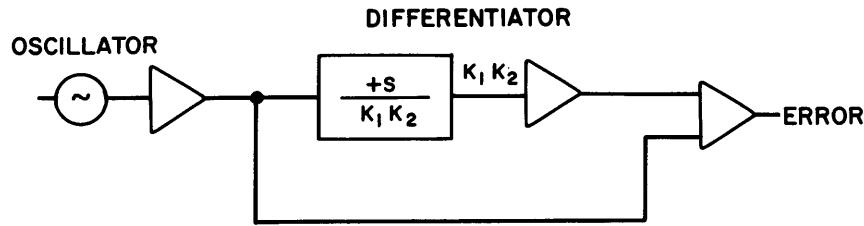
but in practice α is adjusted to a slightly different value due to amplifier and potentiometer phase shift. The error of the circuit is strongly dependent on α causing its setting to be critical. C must be selected so that

α is large but not close to unity. This is due to the stability of the loop composed of amplifiers 1 and 2, and pot 3. A good choice for C is

$$C = \frac{0.3}{K_1 K_2}$$

The circuit will give good performance for $10 \leq K_1 K_2 \leq 1000$. The lower limit is necessary in order to limit the size of C and thus avoid stability problems. It follows that the setting for α is roughly 0.7.

The test circuit for adjusting α is (the block denotes the differentiator circuit).

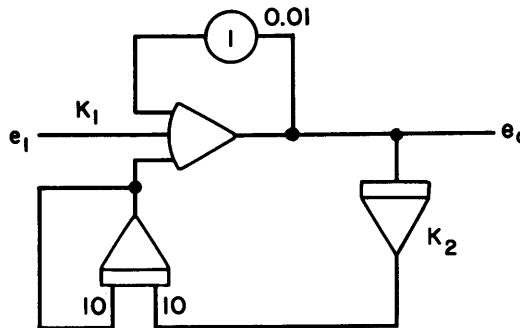


The oscillator is set to the highest frequency of interest and is adjusted until the error is minimum, using a scope. In practice $\alpha > 0.7$.

An excellent low-frequency differentiation circuit is

This circuit works well for

$$1 \leq \frac{K_1}{K_2} \leq 100. \text{ If desired,}$$



pot 1 can be adjusted precisely with the previous test circuit, in which the integrator input is set to

$$\frac{K_2}{K_1} .$$

TIME DELAY (Continuous)

Sometimes a program for time delay is needed for solving a problem. The time-domain definition of time delay is

It has the transfer function

$$e_0(t) = e_1(t - T).$$

$$\frac{e_0}{e_1} = -K_1 s \left(\frac{.1s + 1}{.001s^2 + .01s + K_2} \right)$$

and at low frequency

$$\frac{e_0}{e_1} = -\frac{K_1}{K_2} s .$$

This has the transfer function

$$\frac{e_0}{e_1} = e^{-sT}.$$

There is no convenient way of generating a continuous time delay with only analog computer elements. How-

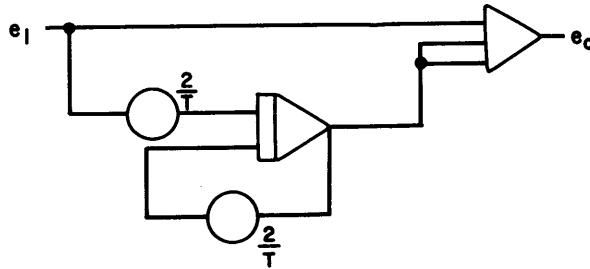
ever, various approximations can be made; most of which are due to Padé. The simplest (first order) is

$$\frac{e_0}{e_1} = \frac{1 - \frac{T}{2}s}{1 + \frac{T}{2}s}$$

which works reasonably well for a delay of $\phi \leq 0.6$ where ϕ is expressed in radians. Since $\phi = \omega T$, where ω is the frequency,

$$T \leq \frac{0.6}{\omega} .$$

That is, at very low frequencies, the approximation will provide relatively large time delays with reasonable accuracy. The program is

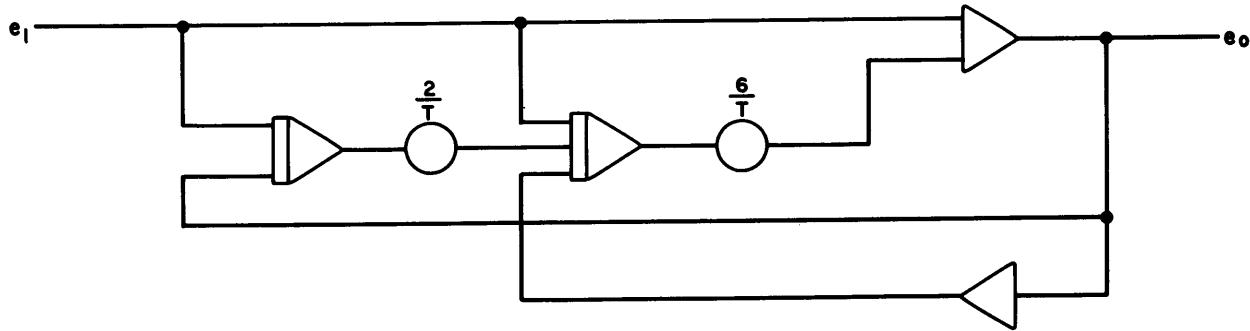


In practice, the highest significant frequency in the simulation is estimated in order to determine an upper bound for T .

A second-order approximation which is reasonably accurate for $\phi \leq 1.8$ is

$$\frac{e_0}{e_1} = \frac{1 - \frac{Ts}{2} + \frac{T^2 s^2}{12}}{1 + \frac{Ts}{2} + \frac{T^2 s^2}{12}} .$$

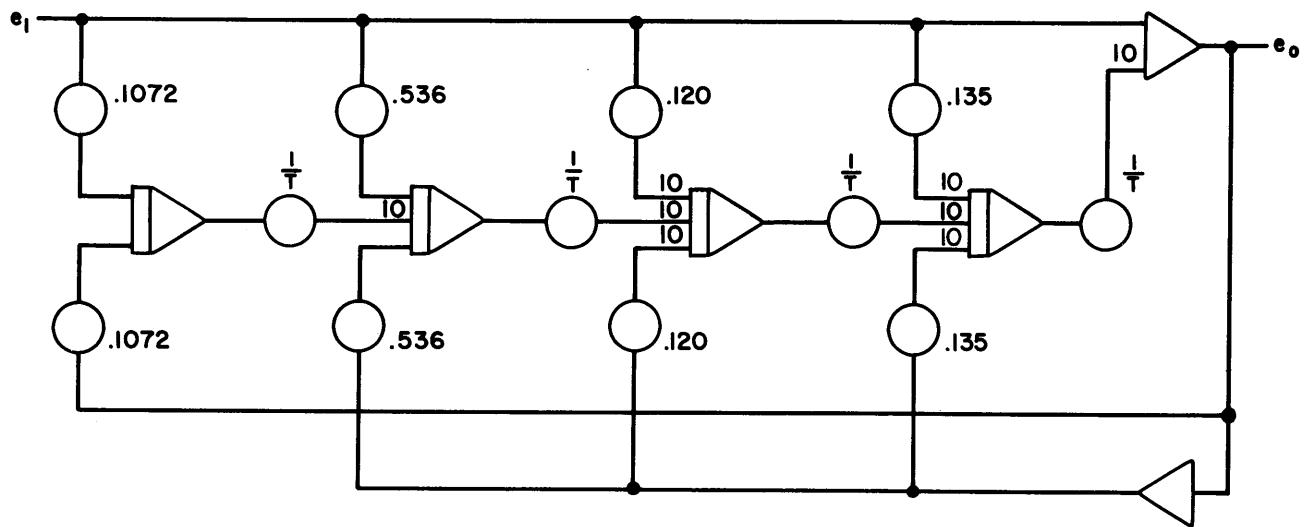
The program is



A fourth-order approximation which is reasonably accurate for $\phi \leq 7.5$ is

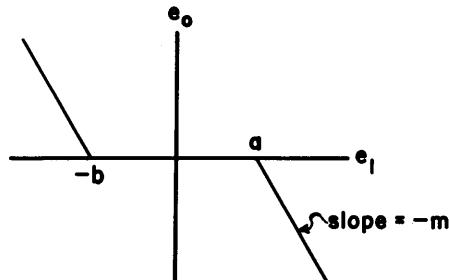
$$\frac{e_0}{e_1} = \frac{1 - \frac{1}{2}(Ts) + \frac{3}{28}(Ts)^2 - \frac{1}{84}(Ts)^3 + \frac{1}{1680}(Ts)^4}{1 + \frac{1}{2}(Ts) + \frac{3}{28}(Ts)^2 + \frac{1}{84}(Ts)^3 + \frac{1}{1680}(Ts)^4} .$$

The program is

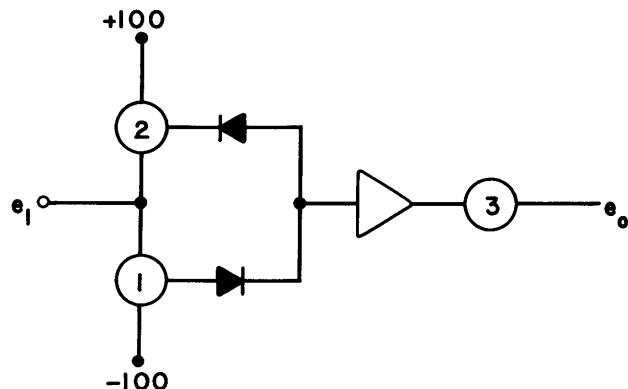


DEAD-ZONE¹

The dead-zone operation relates e_1 , e_0 by the sectionally linear function



The analog program is

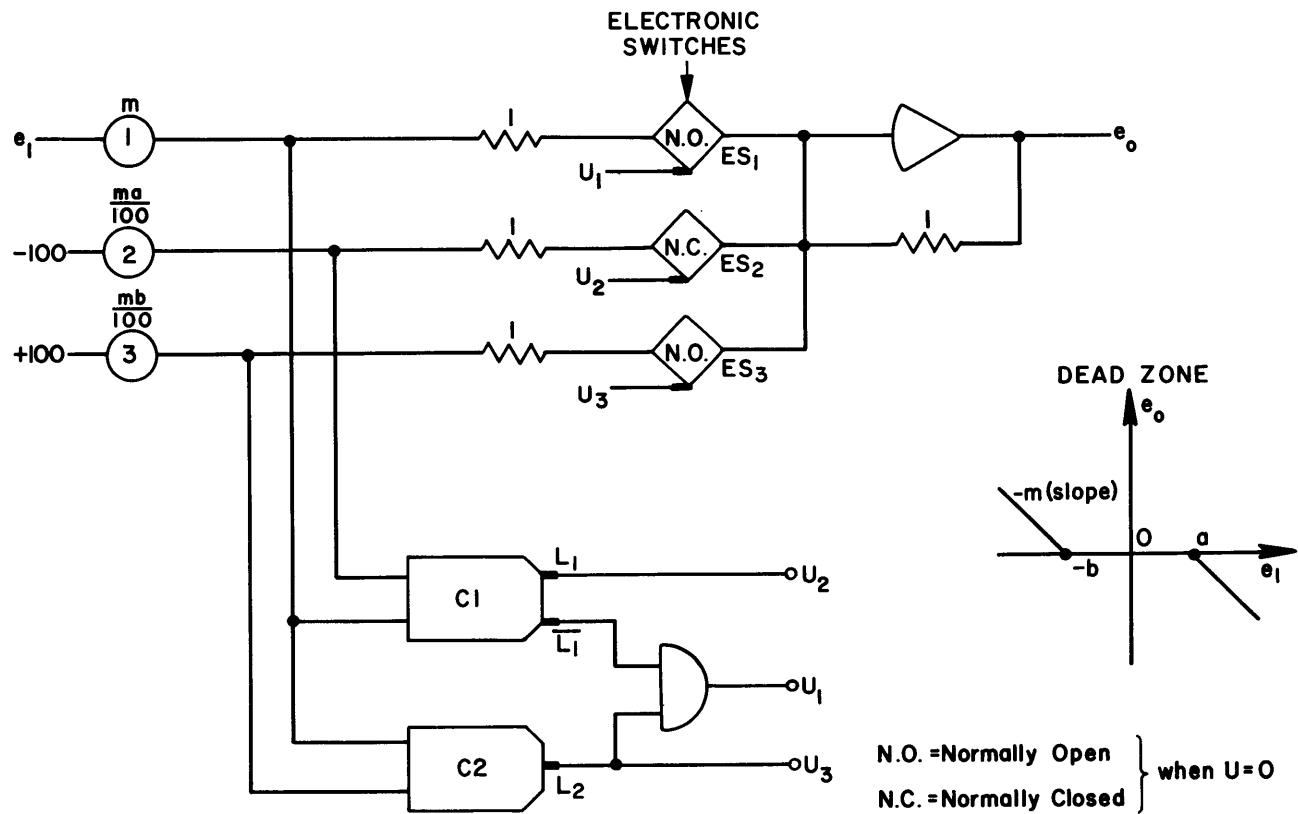


¹ In the following circuits note that mode relays and electronic switches are operated by logical zeroes. The circuits have been shown with electronic switches for mode control. Function relays can be used instead of electronic switches, if high-speed operation is not required.

Pot 1 is set so that its diode will not conduct until $e_1 \geq a$. Pot 2 is set so that its diode will not conduct until $e_1 \leq -b$. Thus, for $-b \leq e_1 \leq a$, neither diode will conduct and $e_o = 0$. Also if pot 3 is set so the total gain is m then

$$\begin{aligned} e_o &= -m(e_1 - a), \quad e_1 \geq a \\ &= 0 \quad , \quad -b \leq e_1 \leq a \\ &= -m(e_1 + b), \quad e_1 \leq -b. \end{aligned}$$

Note that pot 3 cannot be set to m exactly, since the source impedance due to the pot-diode networks is not zero. If digital logic is used, the program is



me_1	L_1	\bar{L}_1	L_2	U_1	U_2	U_3	ES_1	ES_2	ES_3	e_o
$-mb \leq me_1 \leq ma$	0	1	1	1	0	1	O	O	O	0
$+, > ma$	1	0	1	0	1	1	C	C	O	$-m(e_1 - a)$
$-, < -mb$	0	1	0	0	0	0	C	O	C	$-m(e_1 + b)$

Pots 2 and 3 are used to set the breakpoints for the functions. Their outputs are used both to set the comparator references and to provide a biased input to the amplifier as required. The amplifier output will be zero when all three switches are open. Logical variables L_1 , L_2 are defined by

$$L_1 = \text{sgn}(e_1 - a) \quad (\text{output of C1})$$

$$L_2 = \text{sgn}(e_1 + b) \quad (\text{output of C2})$$

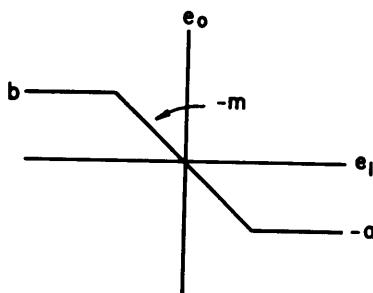
ES_1 is open when $\bar{L}_1 = L_2 = 1$, which is equivalent to

$$-b \leq e_1 \leq a.$$

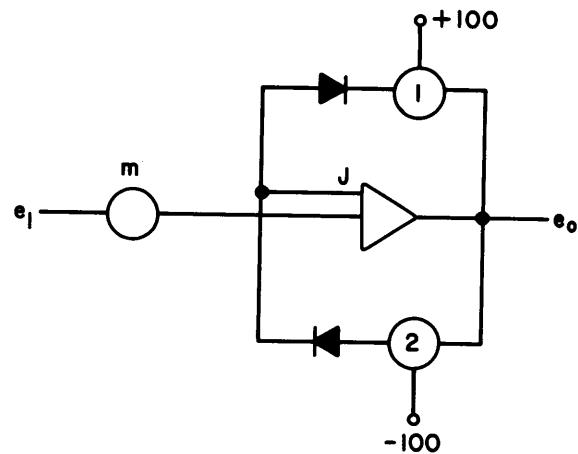
Since pot 1 is set to m, e_o will have the desired functional relationship to e_1 . Normally closed and normally open contacts of the electronic switches are used to provide the logic level inversion required.

SATURATION

The saturation function is

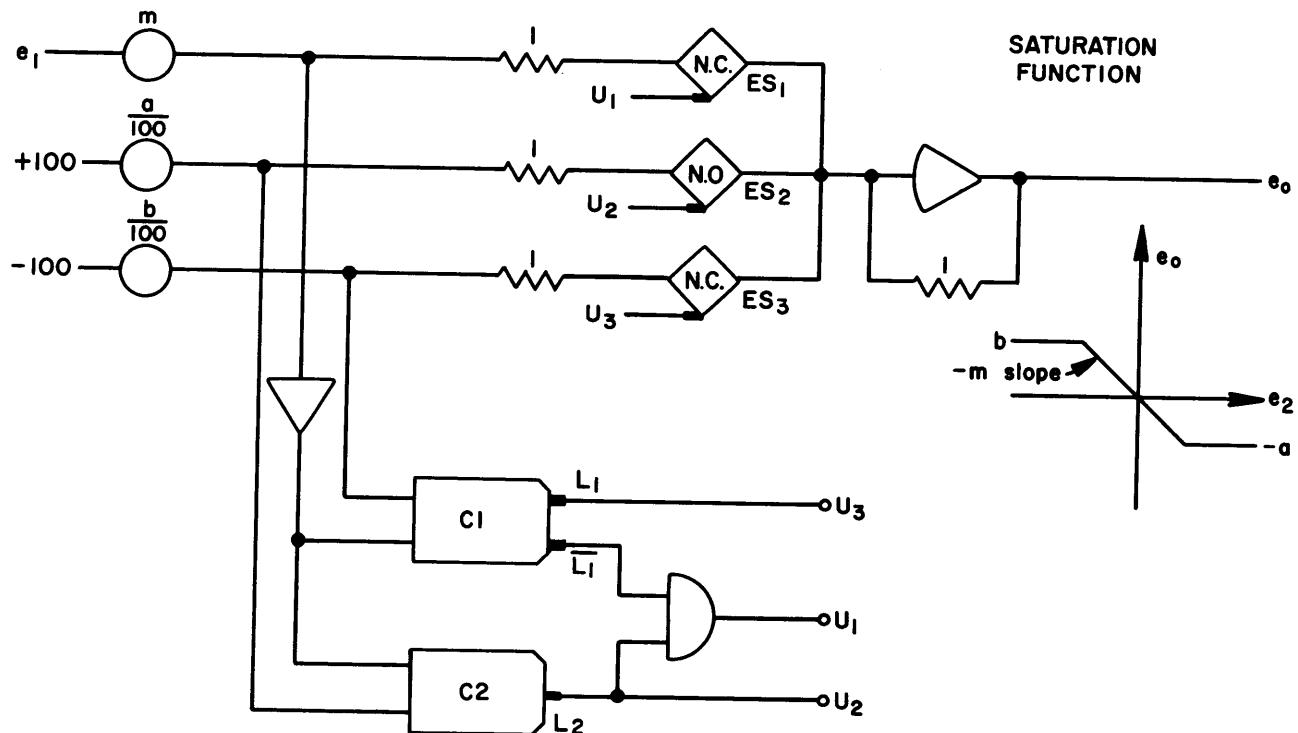


The analog program is



Pots 1 and 2 are set to limit at $-a$ and b respectively. This circuit gives a "soft" limit. That is, e_o does not remain constant at the saturation level, but continues to increase with increasing e_1 due to the non-zero feedback resistance when the diodes are conducting.

The program with digital logic is



$\frac{e_1}{\frac{b}{m} \leq e_1 \leq \frac{a}{m}}$	L_1	\bar{L}_1	L_2	U_1	U_2	U_3	ES_1	ES_2	ES_3	$\frac{e_o}{-me_1}$
$+, e_1 > \frac{a}{m}$	0	1	0	0	0	0	O	C	O	$-a$
$-, e_1 < \frac{b}{m}$	1	0	1	0	1	1	O	O	C	b

The operation of the electronic switches is governed by the logical equations

$$ES_1 = \overline{L_1} \cdot L_2 \quad \left(\frac{-b}{m} \leq e_1 \leq \frac{a}{m} \right)$$

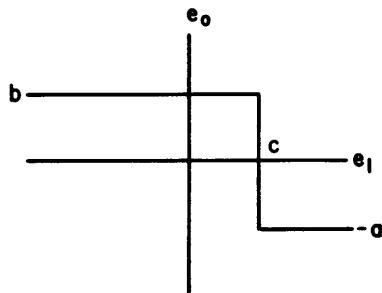
$$ES_2 = \overline{L_2} \quad (e_1 \geq \frac{a}{m})$$

$$ES_3 = L_1 \quad (e_1 \leq \frac{-b}{m})$$

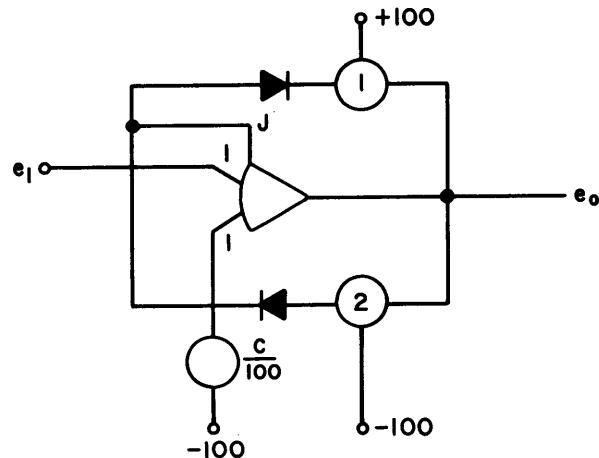
The normally closed or open contacts of the electronic switches are used to provide the logic level inversions where required. The logic program will provide hard-limiting; i.e., the slope after saturation will be zero. This is not true for the purely analog program.

BINARY (bang-bang)

The binary function is

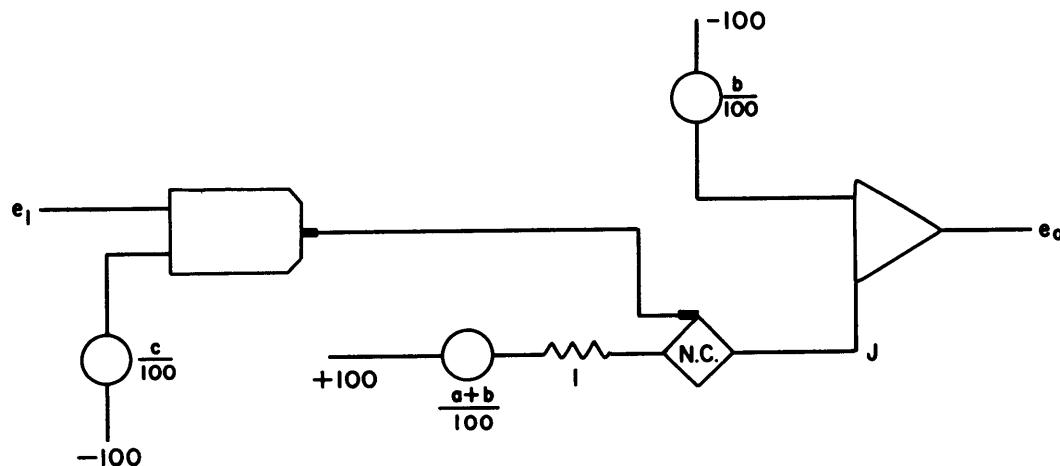


The analog program is



Pots 1 and 2 are set to limit at $-a$ and b respectively. As in the "soft" saturation function, the output in the limiting region has a non-zero slope.

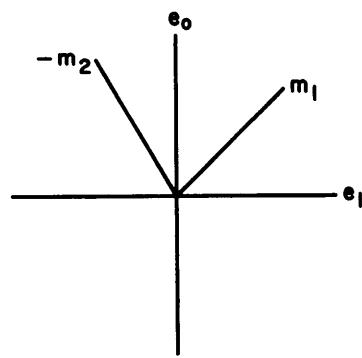
The program with logic is



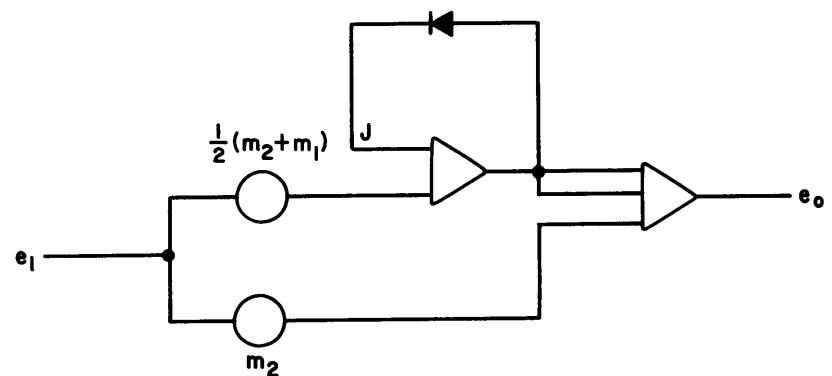
The NC (normally closed) contacts of the switch are used.

ABSOLUTE VALUE (variable slope)

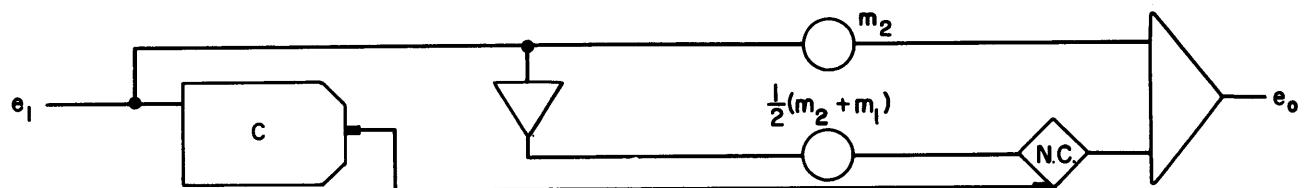
The absolute value function with variable slope is



The analog program is

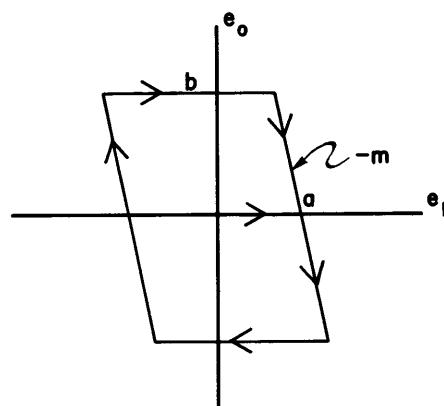


The program with logic is

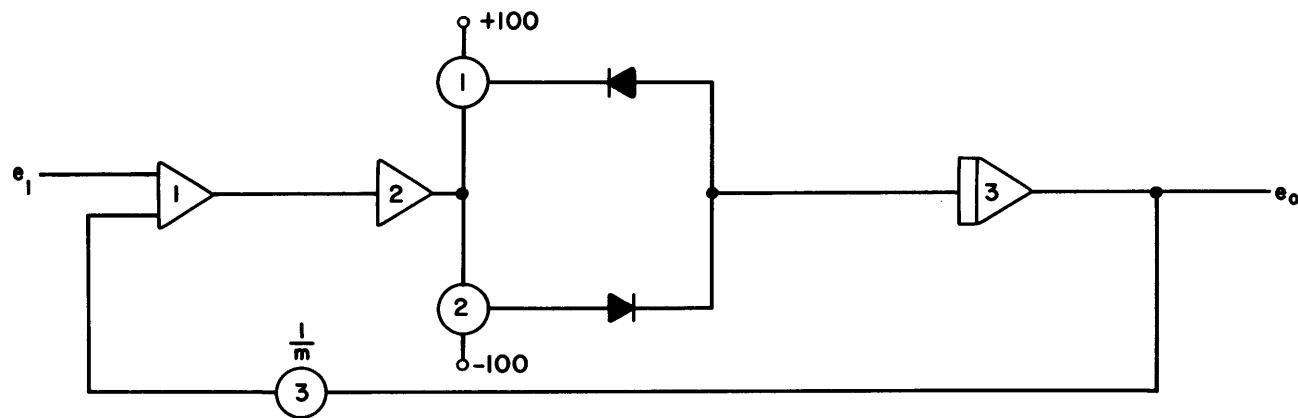


BACKLASH

The backlash function is



The analog program is



Pot 3 is set to $\frac{1}{m}$ where m is the desired slope. Pots 1 and 2 are set to limit at $-b$ and a respectively. The limiting network and amplifier 3 comprise a dead-zone circuit whose output is an integrator rather than a summer. Integrator drift will be about $50\mu\text{v/sec}/\mu\text{fd}$, so it is necessary to consider two factors when selecting the integrator capacitor:

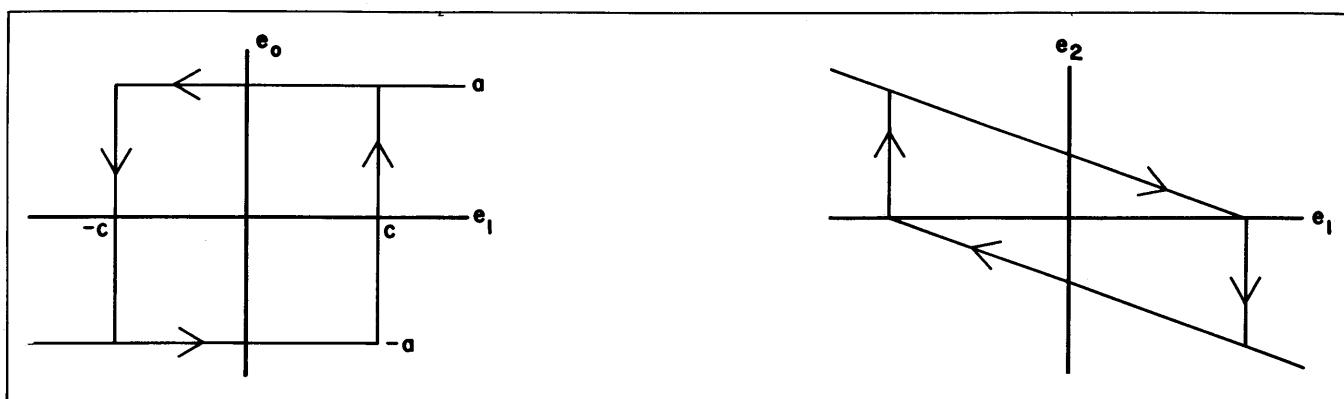
1 - Sufficiently small IC time constant to permit adequate tracking

2 - Sufficiently large C to ensure that the integrator drift is permissible.

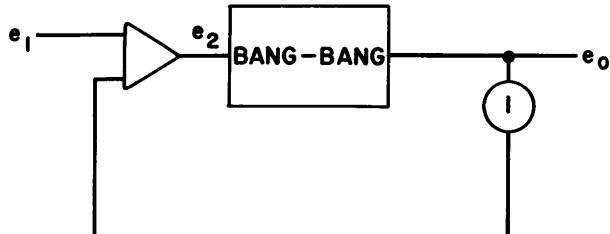
The logic program for this function is obtained by replacing the analog dead-zone circuit with the equivalent logic program.

BISTABLE FLIP-FLOP

The bistable flip-flop function is



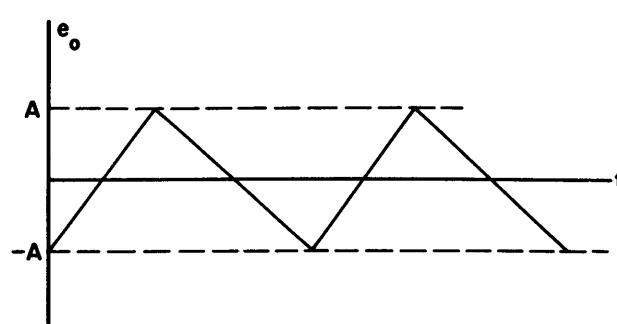
The analog program is



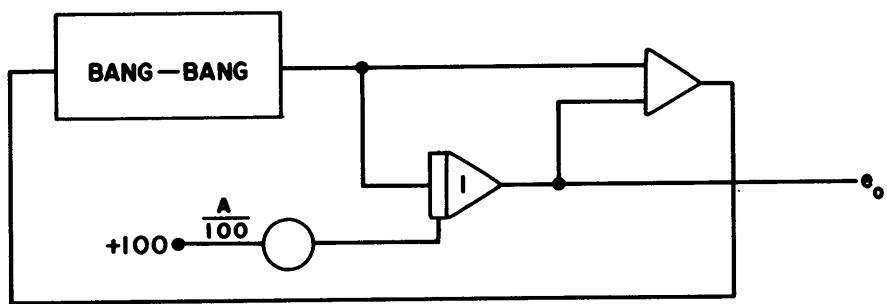
Pot 1 adjusts the value of e_1 which results in a change of state, namely C.

DITHER

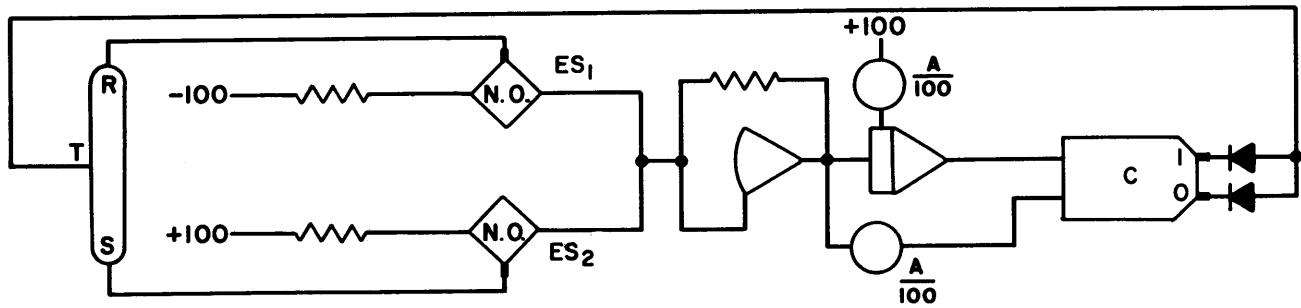
The dither function is



The analog program is



In the bang-bang circuit the limits are set to A , $-A$. The logic program is



In both the analog and the logic programs, the gain of integrator 1 determines the dither frequency.

CHAPTER 11

SIMULATION OF CONSTANT COEFFICIENT TRANSFER FUNCTIONS

The simulation of control systems requires programs for constant coefficient transfer functions of the type

$$T(s) = \frac{P(s)}{Q(s)}$$

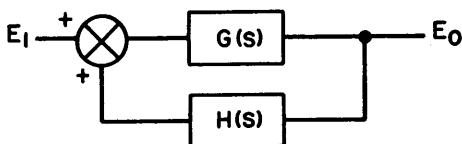
where P, Q , have polynomial form:

$$P(s) = \sum_{i=0}^n a_i s^i, \quad Q(s) = \sum_{j=0}^m b_j s^j.$$

If $n > m$ differentiation is implied. Differentiators are treated in Chapter 10. Here, it is assumed that $n < m$. For physically realizable systems the a_i, b_j are all real numbers. It is well known that any polynomial with real coefficients can be represented as a product of quadratic factors. Further, in many practical problems the physical model is composed of dominant second or first order transfer functions. For these reasons only cases for which $m \leq 2$ are considered. Programs for specific higher order transfer functions can be derived easily with the approach outlined in Chapter 3. The material here is intended to comprise a dictionary for second or first order programs. With this information, generating a program for a complex system diagram with many blocks is easy. The program is written for each block and the system is represented by making appropriate interconnections between blocks.

In developing transfer functions using only pots, amplifiers, and integrators, the following fundamental characteristics of feedback or closed-loop systems are of interest.

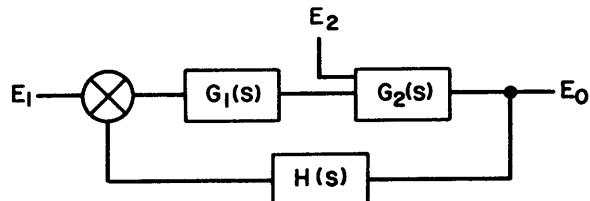
General Closed-Loop System



The transfer function of this system is

$$\frac{E_0}{E_1} = \frac{G}{1 - HG}$$

where $G(s)$ is the forward path transfer function and $H(s)$ is the feedback path transfer function.



The transfer functions of this system are

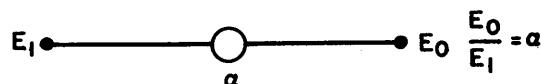
$$\frac{E_0}{E_1} = \frac{G_1 G_2}{1 - HG_1 G_2}$$

$$\frac{E_0}{E_2} = \frac{G_2}{1 - HG_1 G_2} = \frac{1}{G_1} \left(\frac{E_0}{E_1} \right)$$

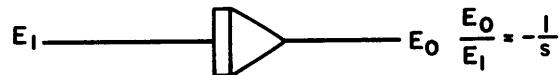
That is, the response to inputs other than E_1 is equal to the normal response divided by the transfer function preceding the actual input.

Specific transfer function programs are given below.

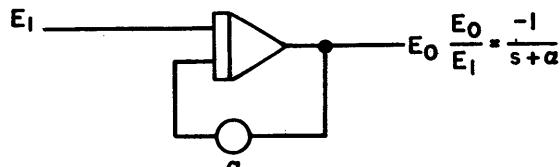
$$T(s) = \alpha:$$



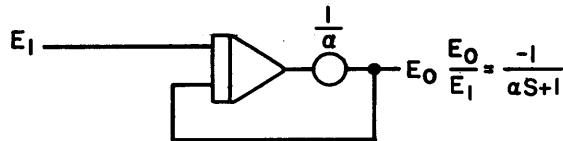
$$T(s) = \frac{1}{s}:$$



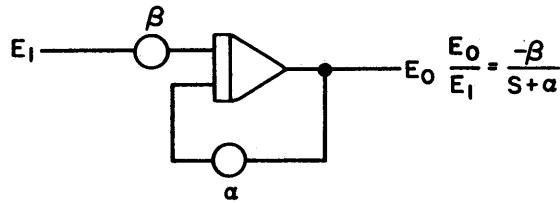
$$T(s) = \frac{1}{s + \alpha}:$$



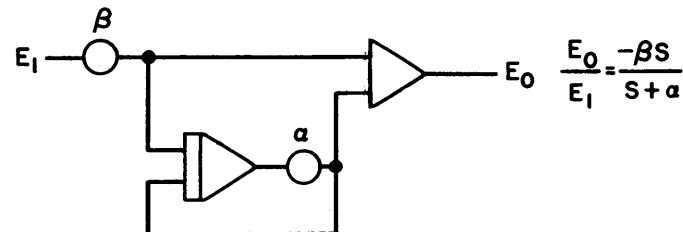
$$T(s) = \frac{1}{\alpha s + 1} :$$



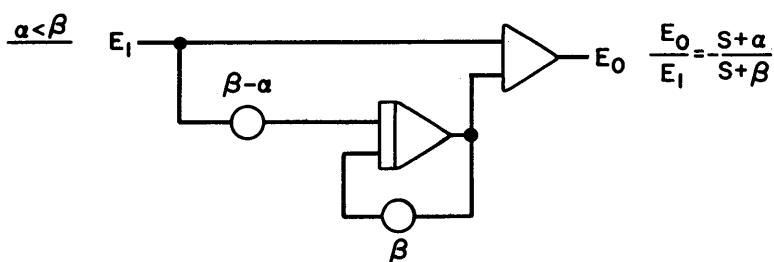
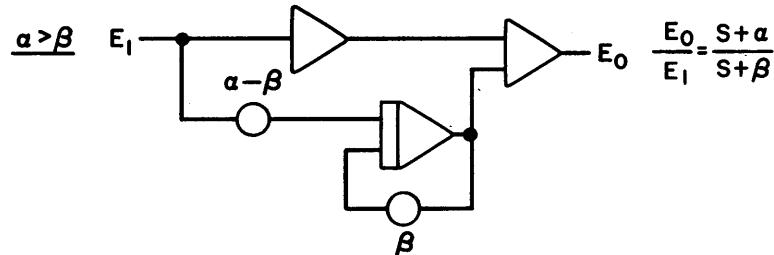
$$T(s) = \frac{\beta}{s + \alpha} :$$



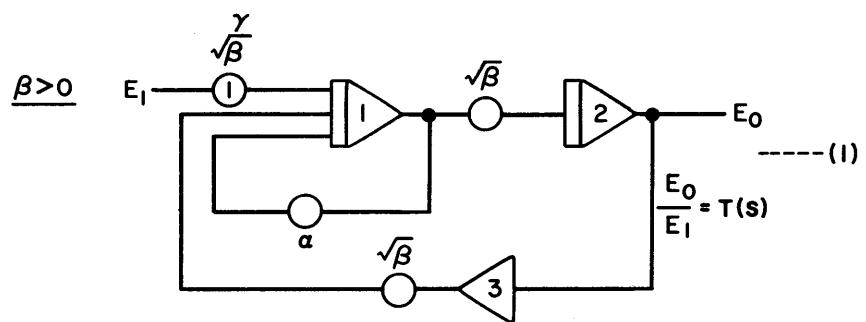
$$T(s) = \frac{\beta s}{s + \alpha} :$$



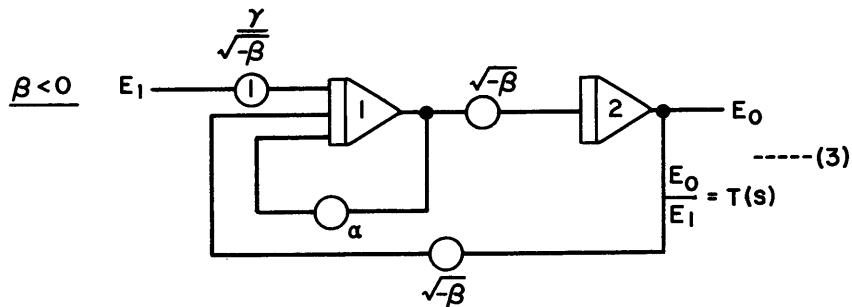
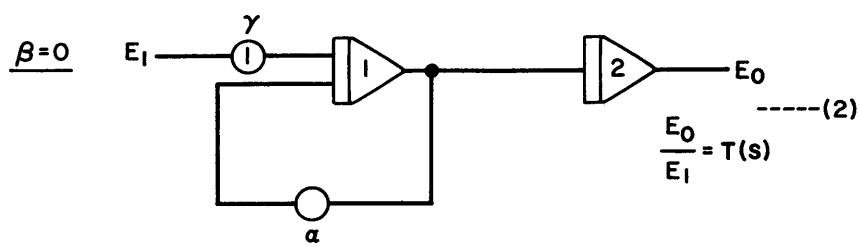
$$T(s) = \frac{s + \alpha}{s + \beta} :$$



$$T(s) = \frac{\gamma}{s^2 + \alpha s + \beta} :$$



$$T(s) = \frac{\gamma}{s^2 + \alpha s + \beta} :$$

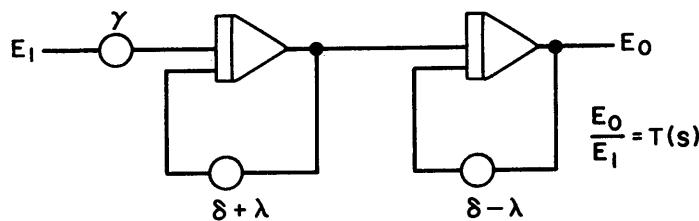


For $\alpha, \beta > 0$, $\alpha^2 > 4\beta$, another program is preferable.

Let $\delta = \frac{\alpha}{2}$, $\lambda^2 = \delta^2 - \beta$. Then

$$T(s) = \frac{\gamma}{(s + \delta + \lambda)(s + \delta - \lambda)}$$

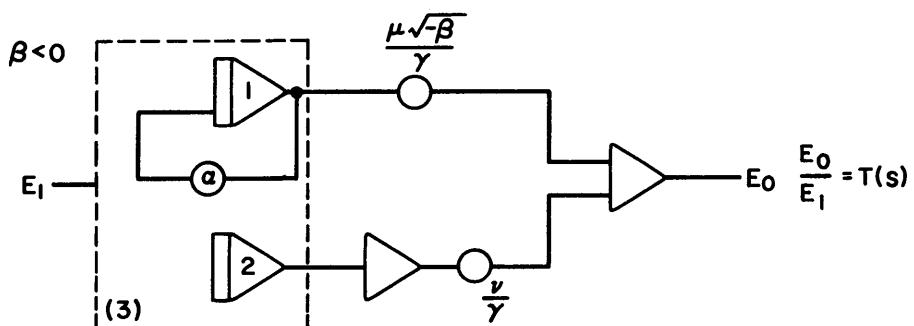
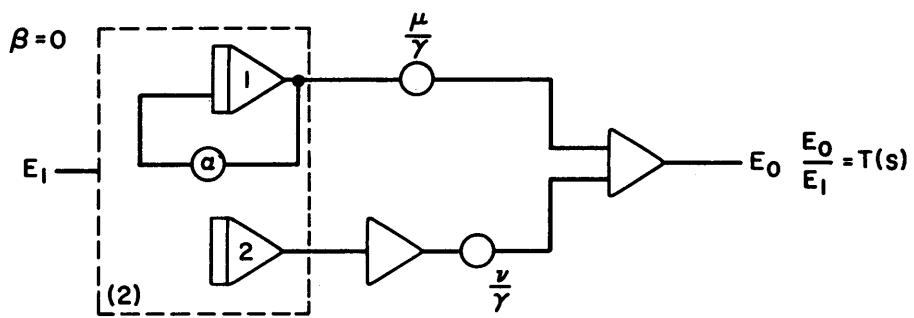
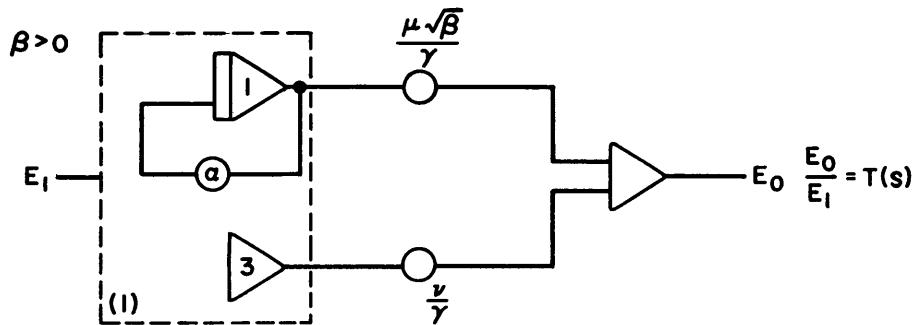
and the program is



Note that the condition $\alpha, \beta > 0$ ensures $(\delta + \lambda), (\delta - \lambda) > 0$.

$$T(s) = \frac{\mu s + \nu}{s^2 + \alpha s + \beta} :$$

The program for this function uses types (1), (2), & (3) of $T(s) = \frac{\gamma}{s^2 + \alpha s + \beta}$, shown above, with the addition of two pots and one or two summers. In each of (1), (2), (3), pot 1 is set with $\gamma = 1$

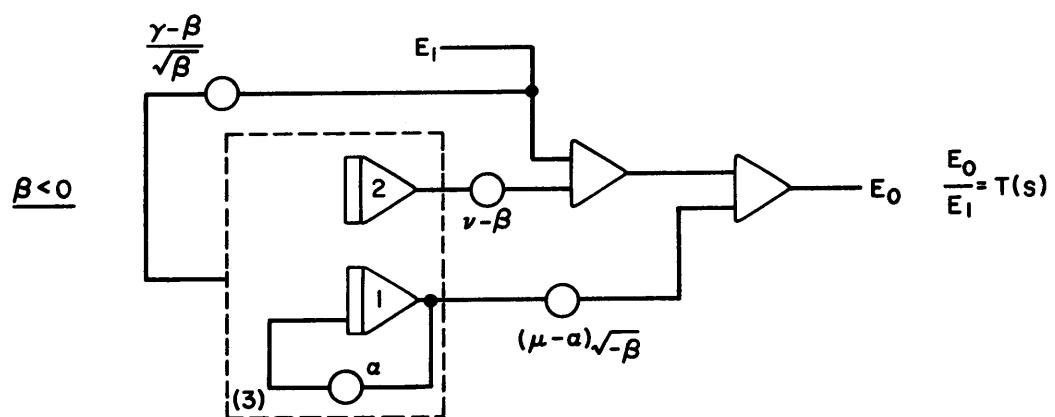
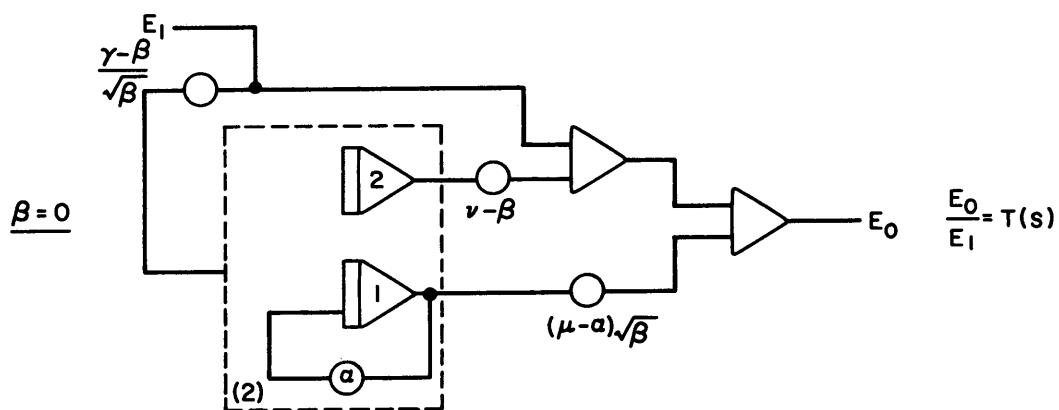
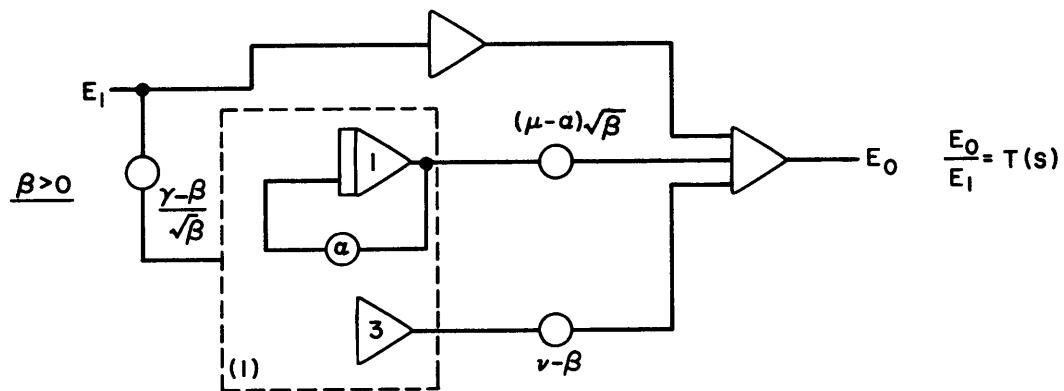


$$T(s) = \frac{s^2 + \mu s + \nu}{s^2 + \alpha s + \beta}$$

For convenience $T(s)$ is simplified

$$T(s) = 1 + \frac{(\mu - \alpha)s + (\nu - \beta)}{s^2 + \alpha s + \beta}$$

and the programs are (where again extra circuitry has been added to (1), (2), (3) above and in each pot 1 is set with $\gamma = 1$).





CHAPTER 12

CONTROL SYSTEM SIMULATION

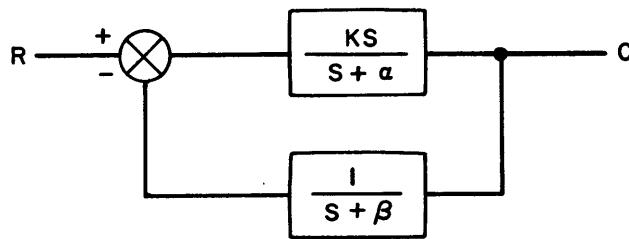
A model of a control system is usually represented by a block diagram. The block diagram contains constant coefficient transfer functions, non-linearities, and frequently, special differential equations. The easiest way to generate the complete computer program for the model is to

1 - derive a program for each block

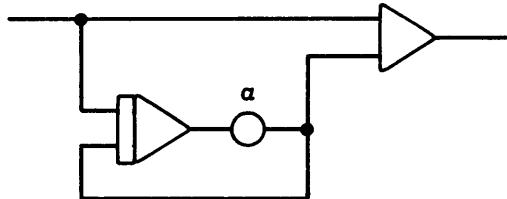
2 - interconnect the blocks.

This method is called block programming and is simple using information developed in Chapter 4. Some illustrative examples are shown below.

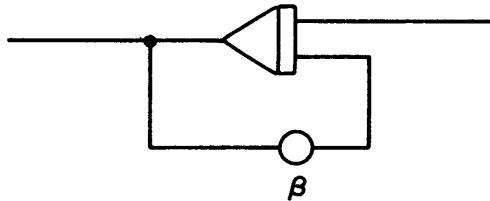
MODEL:



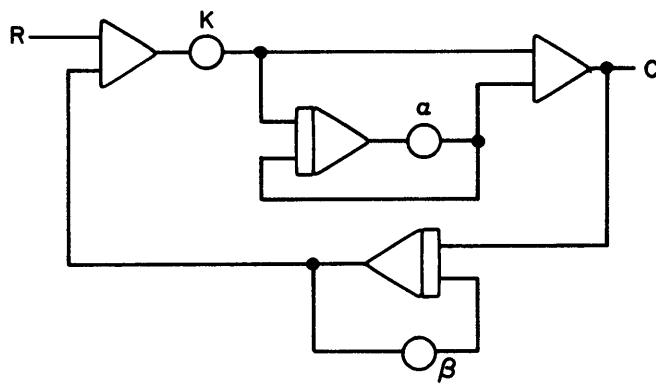
$$-\frac{s}{s + \alpha} :$$



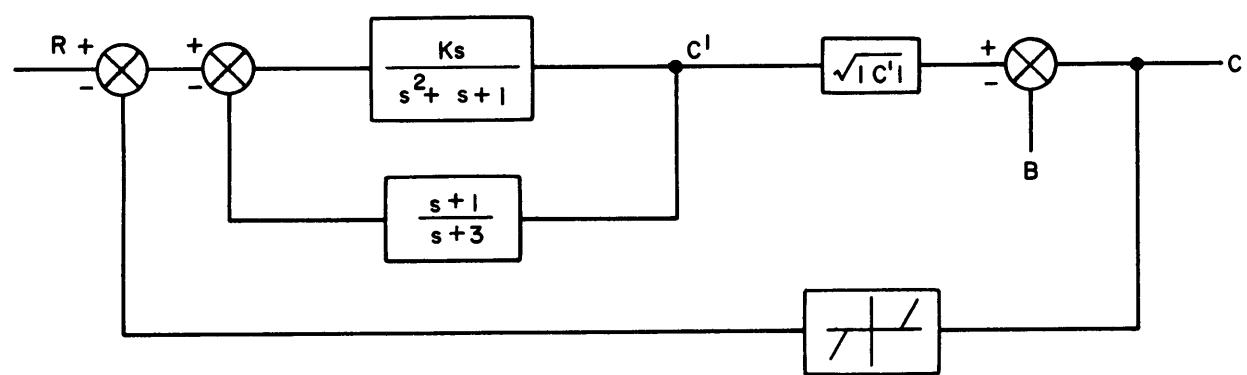
$$-\frac{1}{s + \beta} :$$



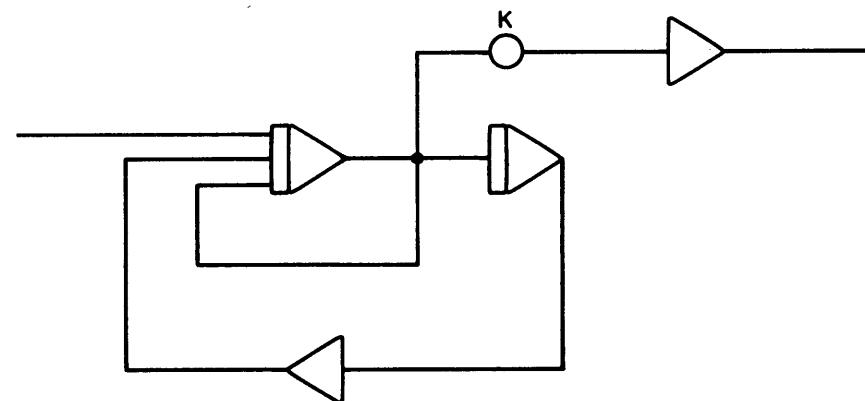
PROGRAM:



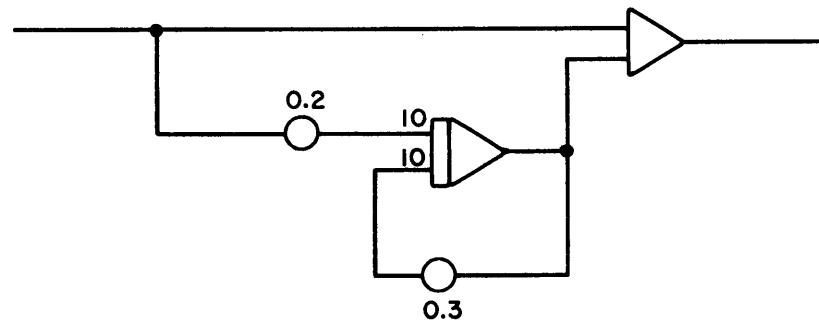
MODEL:



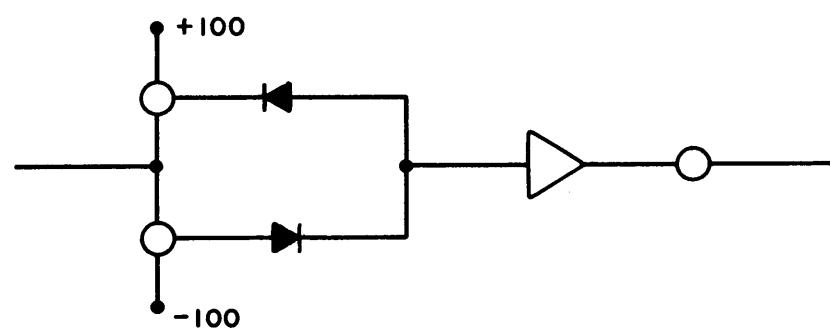
$$\frac{Ks}{s^2 + s + 1} :$$



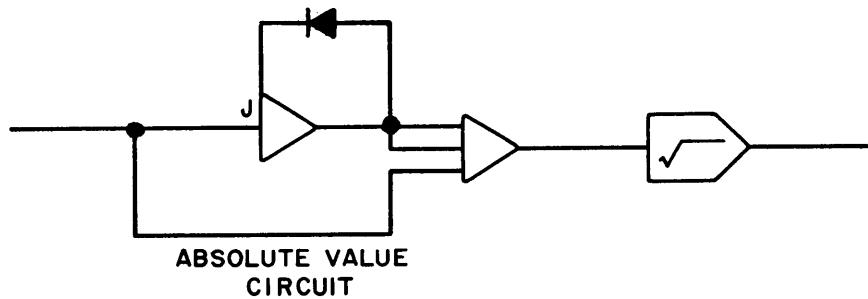
$$- \frac{s+1}{s+3}, \beta > \alpha :$$



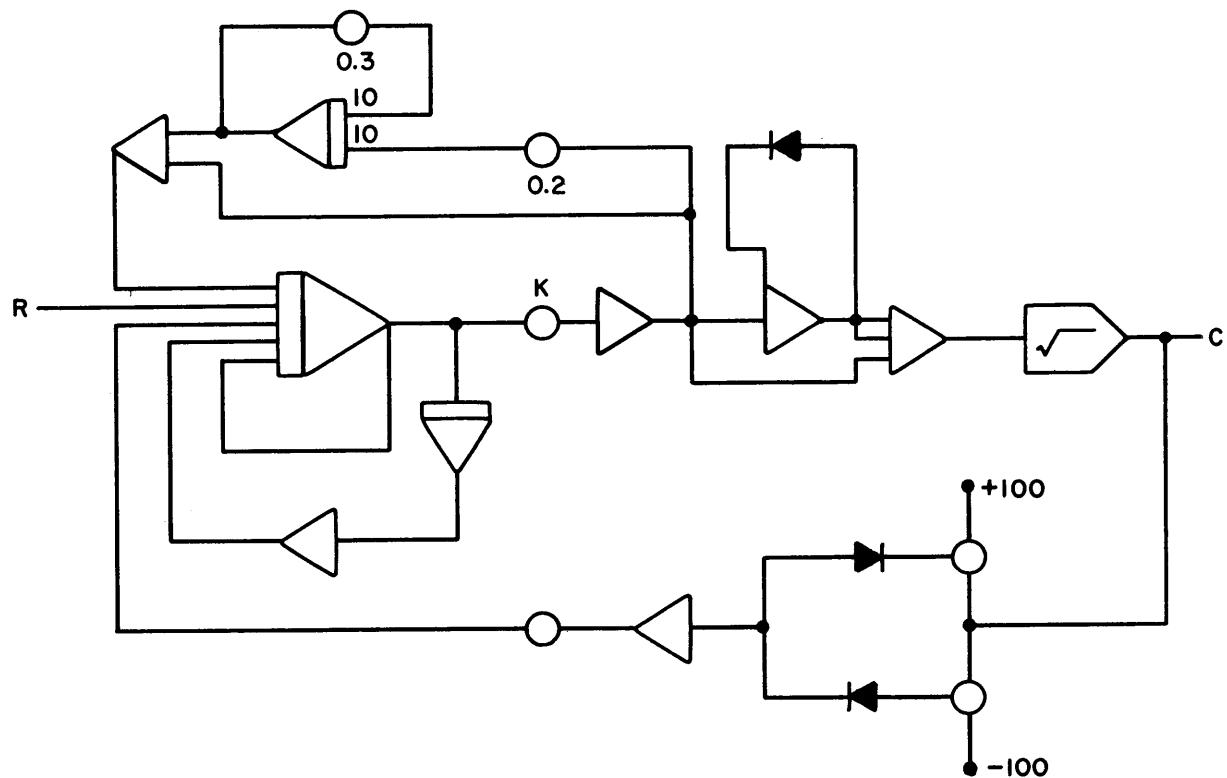
$\frac{1}{7} :$



$\sqrt{I^+ I^-}$:



PROGRAM:



Note that no pot is required for the parameter, B, because this parameter is used to translate the dead-zone function.



CHAPTER 13

FUNDAMENTALS OF VECTOR ANALYSIS

The development of vector operators given below is within the framework of Euclidean (x, y, or x, y, z) space. Due to the invariance property of vectors and vector operators under transformation to other coordinate systems of engineering interest, it is reasonable to proceed in this manner. Sometimes we will show the representation of the vector operator under discussion in other coordinate systems for the purpose of illustration. Indeed, the main motivation for using vector notation is its non-dependence on a particular coordinate system.

DEFINITIONS

(A · B) is the scalar (dot) product of two vectors.

$$A \cdot B = |A| \cdot |B| \cos(A, B)$$

(A x B) is the vector (cross) product of two vectors.

If

$$C = A \times B$$

then

$$|C| = |A| \cdot |B| \sin(A, B).$$

Note that

$$(C \times (A \times B)) = 0$$

∇ is called the vector operator del,

$$\nabla \triangleq \frac{\partial}{\partial x}, \frac{\partial}{\partial y}, \frac{\partial}{\partial z}$$

(Here ordered n-tuples are used instead of the customary i j k unit vector notation, i.e. $\frac{\partial}{\partial x}, \frac{\partial}{\partial y} \triangleq \frac{\partial}{\partial x} i + \frac{\partial}{\partial y} j$).

If $\phi = \phi(x, y, z)$ is a scalar-valued function, then

$$\nabla \phi = \frac{\partial \phi}{\partial x}, \frac{\partial \phi}{\partial y}, \frac{\partial \phi}{\partial z}$$

is a vector-valued function also known as grad ϕ .

∇ has different forms in other coordinate systems.

Example: In cylindrical coordinates

$$\nabla = \frac{\partial}{\partial r}, \frac{1}{r} \frac{\partial}{\partial \theta}, \frac{\partial}{\partial z}$$

For a vector $V = V_x, V_y, V_z$;

$$\nabla \times V \triangleq \frac{\partial V_z}{\partial y} - \frac{\partial V_y}{\partial z}, \frac{\partial V_x}{\partial z} - \frac{\partial V_z}{\partial x}, \frac{\partial V_y}{\partial x} - \frac{\partial V_x}{\partial y}$$

is a vector-valued function known as curl V .

$$\nabla \cdot \nabla = \nabla^2 = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2} + \frac{\partial^2}{\partial z^2} \quad \text{is known as the Laplacian.}$$

SUMMARY OF OPERATOR ALGEBRA (greek letters are scalar-valued functions, capital letters are vector-valued functions)

$$\nabla \cdot (\alpha V) = \alpha (\nabla \cdot V) + V \cdot (\nabla \alpha)$$

$$\nabla \times (\alpha V) = \alpha (\nabla \times V) + (\nabla \alpha) \times V$$

$$\nabla \cdot (V \times U) = U \cdot (\nabla \times V) - V \cdot (\nabla \times U)$$

$$\nabla (V \cdot U) = V \times (\nabla \times U) + (V \cdot \nabla) U + U \times (\nabla \times V) + (U \cdot \nabla) V$$

$$\nabla \times (V \times U) = (U \cdot \nabla) V - (V \cdot \nabla) U - U (\nabla \cdot V) + V (\nabla \cdot U)$$

$$\nabla \times (\nabla \alpha) = 0$$

$$\nabla \cdot (\nabla \times V) = 0$$

$$\nabla \times (\nabla \times A) = \nabla (\nabla \cdot A) - \nabla^2 A \quad (\text{this defines } \nabla^2 \text{ which is not a true vector operation}).$$

$$A \times B = -B \times A$$

$$A \cdot (B \times C) = (A \times B) \cdot C \quad (\text{triple scalar product})$$

DIFFERENTIAL OPERATIONS

The remainder of what follows will be derived for two dimensions but may be readily extended to three.

Divergence:

If $f(x, y)$ is a scalar-valued function then the family $f = c$ of curves in the plane are level lines for f . If

f is thought of as a potential then these curves are equipotential lines.

$$df = \frac{\partial f}{\partial x} dx + \frac{\partial f}{\partial y} dy = 0$$

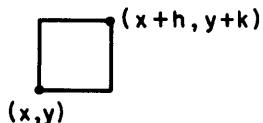
$$\nabla f \cdot (dx, dy) = 0$$

where (dx, dy) is a vector element of arc along the curve $f = c$. (This is so because we constrain the values of x, y so that $f = c$). Thus we see that

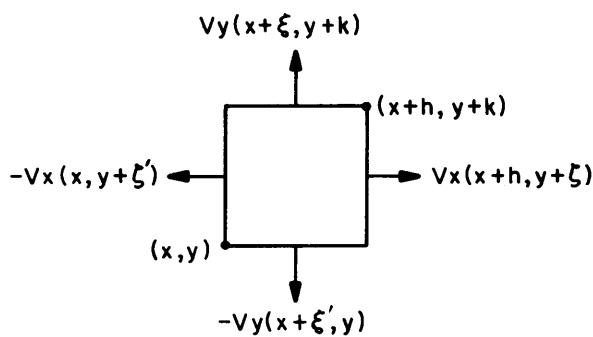
$$\nabla f \perp (dx, dy),$$

and the family of curves obtained from the differential tangent field, ∇f , are orthogonal to the family of level lines and represent a flux (or flow). $\nabla f \cdot N \triangleq \frac{df}{dN}$ is called the directional derivative of f in the N direction (N is a unit vector) at the point (x, y) . The meaning is obvious. $\nabla \cdot V = \frac{\partial V_x}{\partial x} + \frac{\partial V_y}{\partial y}$ is a scalar-valued function known as the divergence of the vector-valued function, V . It is commonly written as $\text{div } V$.

Consider a small rectangle in the x, y -plane



We think of V as a flux, and determine the net emergence of flux from the rectangle,



We have

$$\begin{aligned} \text{Outflow} &= \left[V_x(x+h, y+\xi) - V_x(x, y+\xi') \right] k \\ &+ \left[V_y(x+\xi, y+k) - V_y(x+\xi', y) \right] h \end{aligned}$$

$$\begin{aligned} \frac{\text{Outflux}}{\text{unit area}} &= \frac{V_x(x+h, y+\xi) - V_x(x, y+\xi')}{h} \\ &+ \frac{V_y(x+\xi, y+k) - V_y(x+\xi', y)}{k} \end{aligned}$$

In the limit as $h, k \rightarrow 0$ we have

$$\frac{\text{Outflux}}{\text{unit area}} = \frac{\partial V_x}{\partial x} + \frac{\partial V_y}{\partial y} = \nabla \cdot V = \text{div } V$$

Thus if ϕ is a conservative potential field

$$\nabla \cdot \nabla \phi = 0 \text{ or } \nabla^2 \phi = 0$$

At a point x, y ; $\nabla^2 \phi \geq 0$ implies a local source in the case of incompressible flow. Note that with compressible steady flow $\nabla^2 \phi \neq 0$, in general, without the presence of sources or sinks. In transient physical processes $\nabla^2 \phi$ is a measure of the change of energy from one form to another. For example, in heat transfer processes heat is stored locally by the distributed heat capacity of the medium in which the process occurs. Thus, if T is the temperature at (x, y) , the local conductive heat transfer is $k \nabla T$ where k is the thermal conductivity, and the local heat stored by the heat capacity of the medium is $c \Delta T$ where c is distributed heat capacity. The loss of thermal flux (heat flow) from the conductive process equals the heat flux required for heat storage and this time-varying relationship is given by

$$\nabla \cdot k \nabla T = c \frac{\partial T}{\partial t}$$

or

$$\nabla^2 T = \frac{c}{k} \frac{\partial T}{\partial t} = \frac{1}{K} \frac{\partial T}{\partial t} \quad (\text{Diffusion equation})$$

where K is the thermal diffusivity constant.

In the case of the small oscillations of a uniform membrane, if $w(x, y)$ is defined to be deflection, then the potential energy stored by deformation of the membrane is a function of ∇w while the local kinetic energy of motion due to the mass of the membrane is a function of $\frac{\partial w}{\partial t}$. Thus the time-varying transfer of energy from one state to the other is described by

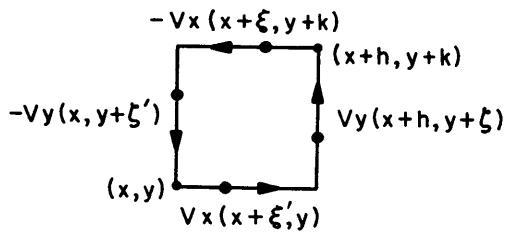
$$\nabla^2 w = \frac{1}{c^2} \frac{\partial^2 w}{\partial t^2} \quad (\text{wave equation})$$

where c is the wave propagation velocity.

Curl:

If we think of the vector-valued function, V , as a velocity field, we can interpret $\text{curl } V$ in a convenient

way. Consider the tangential velocity around the sides of a small rectangle.



The circulation (spin) around the boundary is given by

$$\frac{V_y(x+h, y+\xi) - V_y(x, y+\xi')}{h} + \frac{V_x(x+\xi', y) - V_x(x+\xi, y+k)}{k}$$

In the limit as $h, k \rightarrow 0$ we have

$$\frac{\text{circulation}}{\text{unit area}}(\text{spin}) = \frac{\partial V_y}{\partial x} - \frac{\partial V_x}{\partial y} = \nabla \times V = \text{curl } V.$$

To visualize the result in three-dimensions, we think of an infinitesimal paddle wheel located at the point (x, y, z) . This paddle wheel is angularly oriented so that its shaft has the greatest rate of spin. Using the

righthand rule, the shaft then points in the direction of $\nabla \times V$ and its angular velocity is proportional to the magnitude of $\nabla \times V$.

If $\nabla \times V = 0$, V is said to be an irrotational field.
If $V = \nabla \times A$, V is said to be a solenoidal field.
If $V = \nabla \cdot \phi$ then V is irrotational. ($\nabla \times \nabla \cdot \phi = 0$).
If $V = \nabla \times A$ then V has zero divergence ($\nabla \cdot \nabla \times A = 0$).

VECTOR TRANSFORMATION THEOREMS

Theor 1: (Divergence or Gauss theorem)

Let R be a region over which a vector valued function v is defined, then (differentiability assumed)

$$\int_R \nabla \cdot v = \int_{\partial R} v \cdot N$$

where n is a unit exterior normal to ∂R (boundary of R).

Theor 2: (Stokes' theorem)

Let R be a 3-dimensional region over which a vector valued function V is defined, and let σ be a 2-dimensional subregion of R , then (differentiability assumed)

$$\int_{\sigma} N \cdot (\nabla \times V) = \int_{\partial \sigma} V \cdot ds$$

where N is a unit positive (sense to be determined by $\partial \sigma$) normal to σ and ds is a direction vector which lies in $\partial \sigma$.



CHAPTER 14

PARTIAL DIFFERENTIAL EQUATIONS – PART 1

This section treats the class of partial differential equations which can be represented

$$\nabla \cdot K \nabla \phi = \Xi \phi, \quad \phi = \phi(x_1, \dots, x_n, t)$$

where the operation, Ξ , is defined by one of

$$\Xi \phi = 0 \quad (\text{Laplace equation})$$

$$\Xi \phi = k \quad (\text{Poisson equation})$$

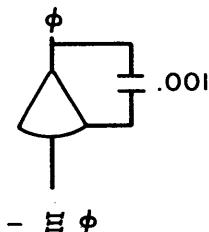
$$\Xi \phi = k\phi \quad (\text{Helmholtz equation})$$

$$\Xi \phi = C(X) \frac{\partial \phi}{\partial t} \quad (\text{Diffusion equation})$$

$$\Xi \phi = M(X) \frac{\partial^2 \phi}{\partial t^2} \quad (\text{Wave equation})$$

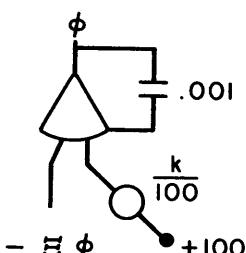
The analog computer is used to integrate $\Xi \phi$, in other words to generate Ξ^{-1} . This will be discussed before considering the generation of $\nabla \cdot K \nabla \phi$. The block programs for these cases are

Laplace:



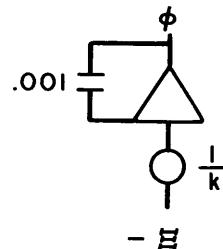
The high-gain amplifier forces $\Xi \phi = 0$ when connected to the remainder of the program. The capacitor is used to reduce high-frequency gain which may result in instability.

Poisson:



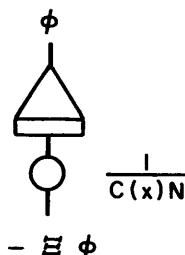
The high-gain amplifier ensures $\Xi \phi - k = 0$.

Helmholtz:

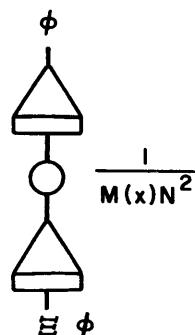


The capacitor is added to prevent algebraic loop instability.

Diffusion: (N is the time scale factor)



Wave: (N is the time scale factor)



With the blocks for $\Xi \phi$ available, all that remains is to find how to represent $\nabla \cdot K \nabla \phi$; the Laplacian of ϕ . The form of ∇ will depend on the coordinate system (see Chapter 13). To illustrate the concept we will derive the Laplacian for one dimension first and then develop it for both two-dimensional Euclidean and Riemannian manifolds. The extension to three dimensions will be obvious. For simplicity it will be assumed that K is not a function of time. For the case where K is a function of time, the pot set to K is replaced by a multiplier.

ONE DIMENSION

$$\nabla \cdot K\nabla\phi \equiv \frac{\partial}{\partial x} \left[K(x) \frac{\partial}{\partial x} \phi(x, t) \right].$$

The derivatives will be replaced by finite differences:

Since

$$\frac{\partial}{\partial x} f(x, t) = \lim_{\Delta x \rightarrow 0} \frac{f(x + \Delta x, t) - f(x, t)}{\Delta x}$$

then, for small Δx

$$\frac{\partial}{\partial x} f(x + \frac{\Delta x}{2}, t) = \frac{f(x + \Delta x, t) - f(x, t)}{\Delta x}$$

Adopting the notations

$$x_{n+\frac{1}{2}} = \frac{x_n + x_{n+1}}{2}$$

and

$$f(x_n, t) = f_n \text{ for any } n$$

we have, for finite differences

$$\frac{\Delta f\left(\frac{x_2 + x_1}{2}, t\right)}{\Delta x} = \frac{f(x_2, t) - f(x_1, t)}{x_2 - x_1}$$

For any n

$$\begin{aligned} \frac{\Delta f\left(\frac{x_{n+1} + x_n}{2}, t\right)}{\Delta x} &= \frac{\Delta f\left(x_{n+\frac{1}{2}}, t\right)}{\Delta n} = \\ &= \frac{\Delta f_{n+\frac{1}{2}}}{\Delta x} = \frac{f(x_{n+1}, t) - f(x_n, t)}{x_{n+1} - x_n} = \\ &= \frac{f_{n+1} - f_n}{x_{n+1} - x_n} \end{aligned}$$

Thus

$$\begin{aligned} [K\nabla\phi]_{x_{n+\frac{1}{2}}} &= K(x_{n+\frac{1}{2}}) \cdot \frac{\phi(x_{n+1}, t) - \phi(x_n, t)}{x_{n+1} - x_n} \\ &= K_{n+\frac{1}{2}} \cdot \frac{\phi_{n+1} - \phi_n}{x_{n+1} - x_n} \end{aligned}$$

Similarly

$$[K\nabla\phi]_{x_{n-\frac{1}{2}}} = K_{n-\frac{1}{2}} \cdot \frac{\phi_n - \phi_{n-1}}{x_n - x_{n-1}}$$

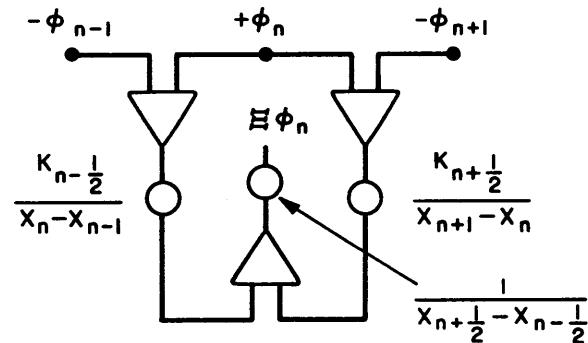
Using the same approximation

$$[\nabla \cdot K\nabla\phi]_{x_n} = \frac{[K\nabla\phi]_{x_{n+\frac{1}{2}}} - [K\nabla\phi]_{x_{n-\frac{1}{2}}}}{x_{n+\frac{1}{2}} - x_{n-\frac{1}{2}}} = \Xi\phi_n$$

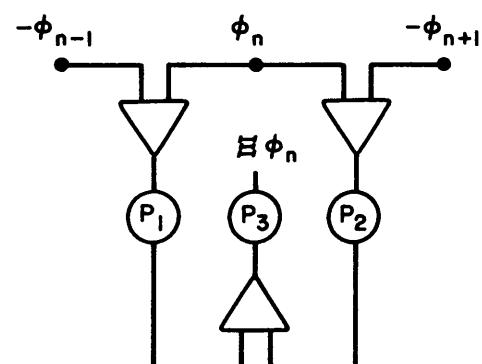
Therefore, by substituting the different equations previously obtained, we have

$$\begin{aligned} \frac{K_{n+\frac{1}{2}}}{x_{n+1} - x_n} [\phi_{n+1} - \phi_n] - \frac{K_{n-\frac{1}{2}}}{x_n - x_{n-1}} [\phi_n - \phi_{n-1}] &= \\ &= \left(x_{n+\frac{1}{2}} - x_{n-\frac{1}{2}} \right) \Xi\phi_n \end{aligned}$$

The program which generates $\Xi\phi_n$ is



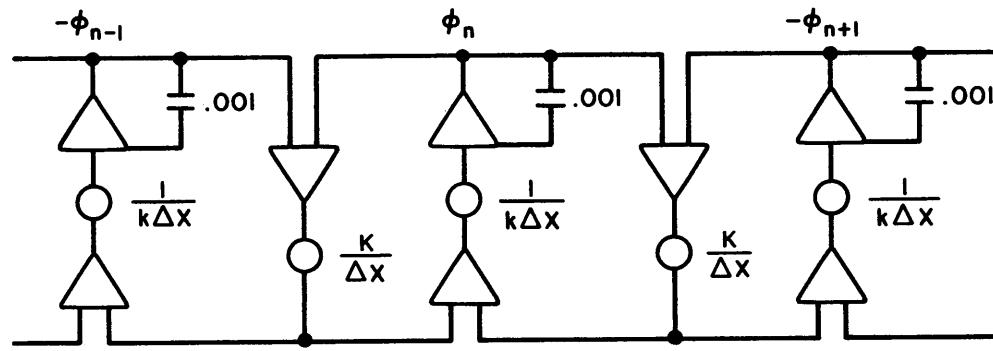
If $x_{n+1} - x_n = \Delta x$, for all n , then the program reduces to



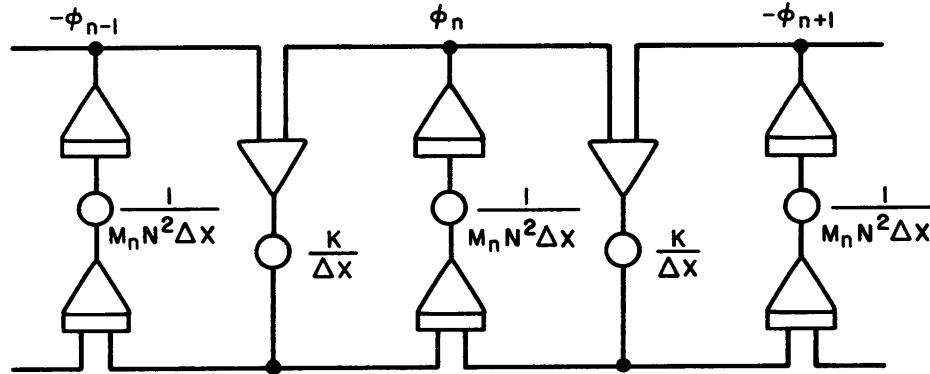
$$P_1 = \frac{K_{n-\frac{1}{2}}}{\Delta x}, \quad P_2 = \frac{K_{n+\frac{1}{2}}}{\Delta x}, \quad P_3 = \frac{1}{\Delta x}.$$

Finally, we have the program for all five types of equations above. Two are given below as examples.

Helmholtz:



Wave:



ONE-DIMENSIONAL BOUNDARY CONDITIONS

Boundary conditions of two types are commonly encountered:

$$\phi(\alpha, t) = f(t) \quad [\text{Type 1}]$$

$$K(\alpha) \nabla \phi(\alpha, t) = f(t) \quad [\text{Type 2}]$$

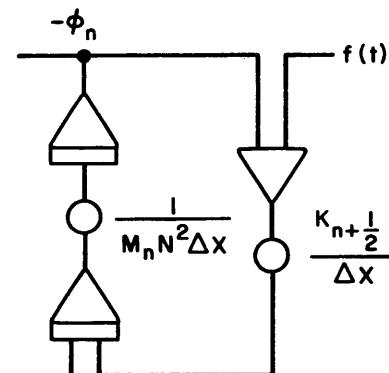
Sometimes both types are specified simultaneously as a mixed boundary condition. These are approximated by

$$\phi_n = f(t), \quad x_n = \alpha$$

$$K_{n+\frac{1}{2}} \nabla \phi_{n+\frac{1}{2}} = f(t), \quad x_{n+\frac{1}{2}} = \alpha$$

The programs are shown below for the Wave equation, but are typical for all $\Xi\phi$.

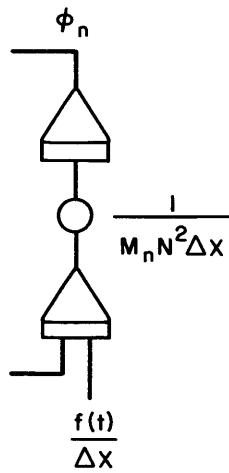
Type 1:



Type 2:

TWO-DIMENSIONAL EUCLIDEAN MANIFOLD

A two-dimensional Euclidean manifold is commonly called an x, y-coordinate system. Here



$$\nabla \cdot K \nabla \phi = \left(\frac{\partial}{\partial x}, \frac{\partial}{\partial y} \right) \cdot \left(K(x, y) \frac{\partial \phi}{\partial x}, K(x, y) \frac{\partial \phi}{\partial y} \right)$$

and

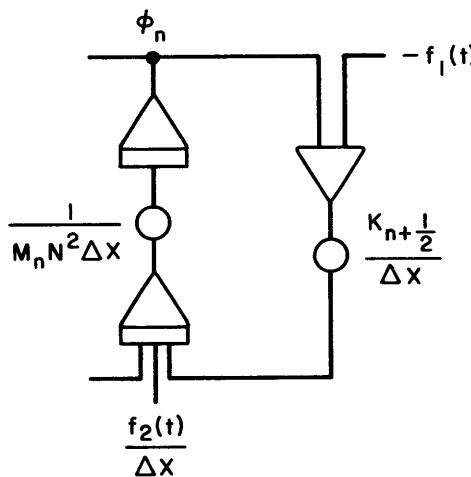
$$\nabla \cdot K \nabla \phi = \frac{\partial}{\partial x} K \frac{\partial \phi}{\partial x} + \frac{\partial}{\partial y} K \frac{\partial \phi}{\partial y} = \Xi \phi$$

As for the one-dimensional case, finite difference approximations are made for the derivatives: we will adopt the notation

$$\phi_{n, m} \stackrel{\Delta}{=} \phi(x_n, y_m)$$

$$\frac{K_{n+\frac{1}{2}, m}}{x_{n+\frac{1}{2}} - x_{n-\frac{1}{2}}} \left(\phi_{n+1, m} - \phi_{n, m} \right) - \frac{K_{n-\frac{1}{2}, m}}{x_{n+\frac{1}{2}} - x_{n-\frac{1}{2}}} \left(\phi_{n, m} - \phi_{n-1, m} \right)$$

Mixed boundary conditions are represented by



$$+ \frac{K_{n, m+\frac{1}{2}}}{y_{m+\frac{1}{2}} - y_{m-\frac{1}{2}}} \left(\phi_{n, m+1} - \phi_{n, m} \right)$$

$$- \frac{K_{n, m-\frac{1}{2}}}{y_{m+\frac{1}{2}} - y_{m-\frac{1}{2}}} \left(\phi_{n, m} - \phi_{n, m-1} \right) = \Xi \phi_{n, m}$$

If $x_{n+1} - x_n = \Delta x$, $y_{m+1} - y_m = \Delta y$, for all n, m , then the equation reduces to

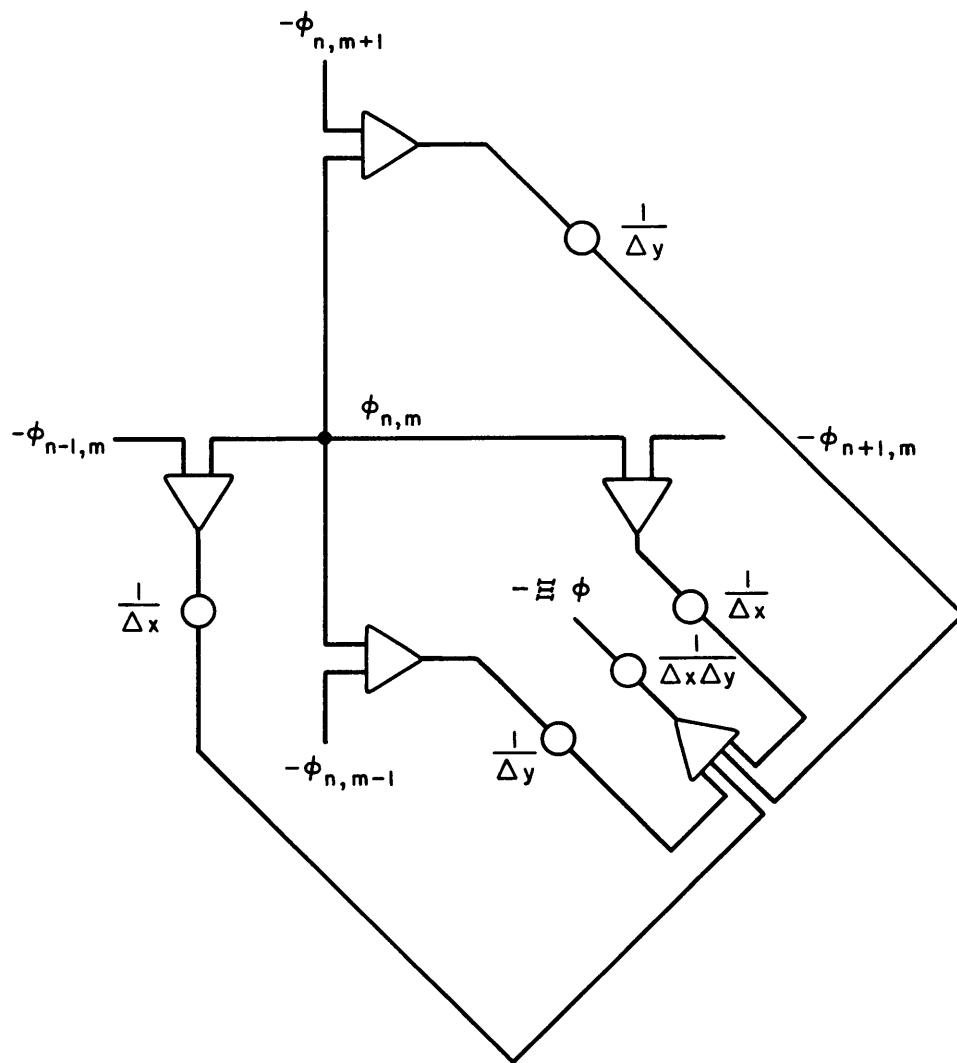
$$\frac{K_{n+\frac{1}{2}, m}}{\Delta x} \left(\phi_{n+1, m} - \phi_{n, m} \right) + \frac{K_{n-\frac{1}{2}, m}}{\Delta x} \left(\phi_{n-1, m} - \phi_{n, m} \right)$$

$$+ \frac{K_{n, m+\frac{1}{2}}}{\Delta y} \left(\phi_{n, m+1} - \phi_{n, m} \right) \frac{K_{n, m-\frac{1}{2}}}{\Delta y} \left(\phi_{n, m-1} - \phi_{n, m} \right)$$

$$= \Delta x \Delta y \Xi \phi_{n, m}.$$

This discussion can be extended to higher-dimension cases by treating each of the boundary points in the same fashion.

The program is



TWO-DIMENSIONAL RIEMANN MANIFOLD

Let the metric for the manifold be given by ($ds = \text{arc length}$)

$$ds^2 = g_{ij} dx^i dx^j.$$

In what follows we will use summation notation (i.e., we sum on repeated indices).

$$g = |g_{ij}| = g_{i\alpha} G^{i\alpha} \quad (\text{no sum on } i)$$

so that G^{ij} is the cofactor of g_{ij} in the determinant $|g_{ij}|$.

We define

$$g^{ij} = \frac{G^{ij}}{g} .$$

Note that

$$g^{ij} g_{ik} = \delta^j_k .$$

It can be shown, that if ϕ is a scalar function,

$$\nabla^2 \phi = \frac{1}{\sqrt{g}} \left(\frac{\partial}{\partial x^i} \right) \left(\sqrt{g} g^{ij} \frac{\partial \phi}{\partial x^j} \right)$$

Thus, we have a general expression for $\nabla^2 \phi$ in non-orthogonal curvilinear coordinates. We could treat the non-orthogonal case. However, that treatment is much more complicated than for the orthogonal coordinates. Since the latter is of greater practical interest, we will restrict our attention to

$$g_{ij} = 0, \quad i \neq j,$$

$$g^{ij} = 0, \quad i \neq j$$

Thus, in three dimensions,

$$ds^2 = g_{11}(dx^1)^2 + g_{22}(dx^2)^2 + g_{33}(dx^3)^2$$

$$g = g_{11} g_{22} g_{33}$$

$$g^{ii} = \frac{1}{g_{ii}} \text{ (no sum on i)}$$

We denote

$$\phi(x^i, x^j, x^k); x^\alpha = A_\alpha, \alpha=i, j, k, A_\alpha \text{ are constant}$$

by $\phi_{i, j, k}$.

Consider the diffusion equation

$$\nabla \cdot K \nabla \phi = \Xi \phi$$

We will need to replace the space derivatives with finite differences.

$$\begin{aligned} \nabla \cdot K \nabla \phi &= \frac{1}{\sqrt{g}} \frac{\partial}{\partial x^i} (K \sqrt{g} g^{ij} \frac{\partial \phi}{\partial x^j}) \\ &= \Xi \phi, \quad K = K(x^1, x^2, x^3) \end{aligned}$$

$$\frac{\partial}{\partial x^i} (K \sqrt{g} g^{ij} \frac{\partial \phi}{\partial x^j}) = \sqrt{g} \Xi \phi$$

Now

$$g^{ij} = 0, \quad i \neq j$$

$$\sqrt{g} g^{11} = \frac{\sqrt{g}}{g_{11}} = \frac{\sqrt{g_{22} g_{33}}}{\sqrt{g_{11}}}$$

$$\sqrt{g} g^{22} = \frac{\sqrt{g_{11} g_{33}}}{\sqrt{g_{22}}}$$

$$\sqrt{g} g^{33} = \frac{\sqrt{g_{11} g_{22}}}{\sqrt{g_{33}}}$$

and

$$\begin{aligned} &\frac{\partial}{\partial x^1} \left(\frac{K \sqrt{g_{22}} \sqrt{g_{33}}}{\sqrt{g_{11}}} \frac{\partial \phi}{\partial x^1} \right) \\ &+ \frac{\partial}{\partial x^2} \left(\frac{K \sqrt{g_{11}} \sqrt{g_{33}}}{\sqrt{g_{22}}} \frac{\partial \phi}{\partial x^2} \right) \\ &+ \frac{\partial}{\partial x^3} \left(\frac{K \sqrt{g_{11}} \sqrt{g_{22}}}{\sqrt{g_{33}}} \frac{\partial \phi}{\partial x^3} \right) = \sqrt{g} \Xi \phi \end{aligned}$$

When approximated with finite differences

$$\frac{\partial}{\partial x^1} \left(\frac{K \sqrt{g_{22}} \sqrt{g_{33}}}{\sqrt{g_{11}}} \frac{\partial \phi}{\partial x^1} \right) \rightarrow \frac{1}{\Delta x^1 \Delta x^2 \Delta x^3}$$

$$\cdot \left[\begin{bmatrix} \frac{\Delta x^2 \Delta x^3}{\Delta x^1} & \left(\frac{K \sqrt{g_{22}} \sqrt{g_{33}}}{\sqrt{g_{11}}} \right)_{i+1/2, k, j} \end{bmatrix} \right]$$

$$\cdot \left[\begin{bmatrix} \phi_{i+1, j, k} - \phi_{i, j, k} \end{bmatrix} + \left[\frac{\Delta x^2 \Delta x^3}{\Delta x^1} \left(\frac{K \sqrt{g_{22}} \sqrt{g_{33}}}{\sqrt{g_{11}}} \right) \right] \right]$$

$$\left[\begin{bmatrix} \phi_{i-1/2, j, k} - \phi_{i, j, k} \end{bmatrix} \cdot \left[\begin{bmatrix} \phi_{i-1, j, k} - \phi_{i, j, k} \end{bmatrix} \right] \right]$$

and similarly for the other terms. Let us consider the one-dimensional projection to determine the significance of these terms. From the definition of the metric, clearly

$$\Delta x^1 \sqrt{g_{11}}$$

$$\Delta x^2 \sqrt{g_{22}}$$

$$\Delta x^3 \sqrt{g_{33}}$$

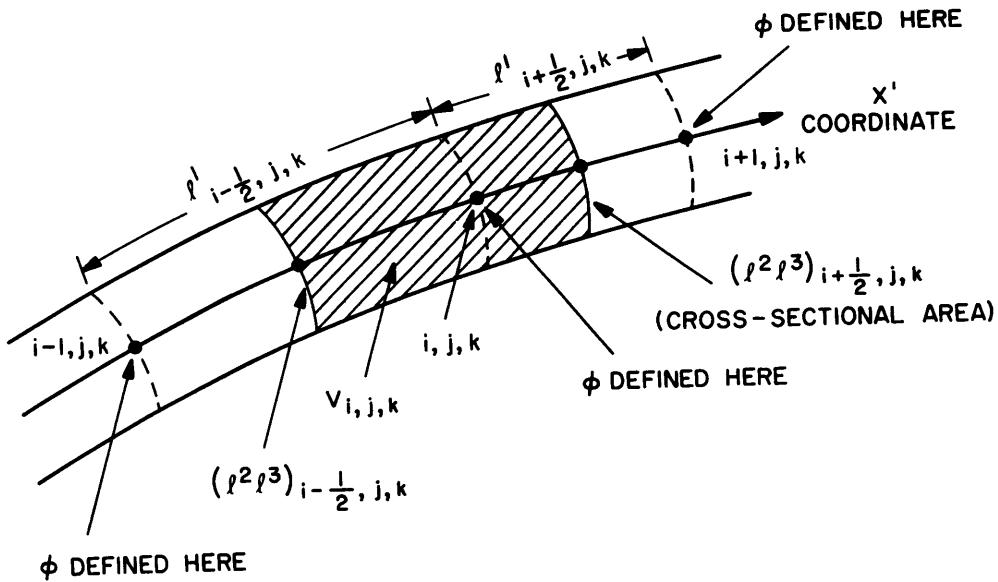
are lengths in the x^1, x^2, x^3 directions respectively. We denote these by l^1, l^2, l^3 respectively. $\Delta x^1 \Delta x^2 \Delta x^3 \sqrt{g}$ is the volume $V = l^1 l^2 l^3$.

In general, the value of the lengths and the volume for increments in x^1, x^2, x^3 depends on location. The one-dimensional projection of the problem is

$$\left[\left(\frac{l^2 l^3 K}{l^1} \right)_{i+1/2, j, k} (\phi_{i+1, j, k} - \phi_{i, j, k}) \right]$$

$$+ \left[\left(\frac{l^2 l^3 K}{l^1} \right)_{i-1/2, j, k} (\phi_{i-1, j, k} - \phi_{i, j, k}) \right]$$

$$= V_{i, j, k} \Xi \phi_{i, j, k}$$



Thus, for example, we see that, in the Diffusion equation

$$\left(\frac{l^2 l^3 K}{l^1}\right)_{i-\frac{1}{2}, j, k} \text{ and } \left(\frac{l^2 l^3 K}{l^1}\right)_{i+\frac{1}{2}, j, k}$$

are lumped conductivities defined at appropriate locations and $V_{i, j, k}$ is a lumped capacity, also defined at an appropriate location. The extension to the non-projected problem in three-dimensions is obvious. It is interesting to note that the theoretical result agrees with our intuition.

ASYMMETRIC LATTICE

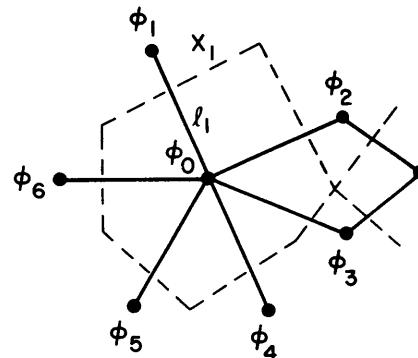
The set of space points over which the operator

$$\nabla \cdot K \nabla \phi$$

is defined is called a lattice. In arbitrary regions one cannot find a metric in any simple way. Thus, the approach developed for the Riemann manifold is not applicable. If a Euclidean system is used, then the boundary must be composed of a collection of orthogonal line segments. This has two disadvantages: the lattice density must be high on the boundary in order to represent the boundary. The lattice density on the boundary determines the lattice density in the interior of the region, leading to an exorbitant equipment requirement for the program. The asymmetric lattice representation of $\nabla \cdot K \nabla \phi$ will allow one to avoid both of these difficulties.

Consider a lattice whose only restriction is that a set of perpendiculars to the branches (one for each) can

be found such that the perpendiculars in each polygon enclosed by a branch loop intersect at only one point.



Referring to the figure above,

1. Let a point function ϕ_α be defined over the lattice
2. Consider one point of the lattice where we arbitrarily choose ϕ_0 and denote by ϕ_α the ϕ 's at branch connected neighbors
3. Define l_i to be the length of the branch between lattice points associated with ϕ_0, ϕ_i .
4. Define r_i to be the length of the \perp to l_i
5. Define the symmetry ratio $Y_i = \frac{r_i}{l_i}$

Now consider the representation of $\nabla^2 \phi$ at the lattice point 0 given by

$$0(\phi_0) \triangleq \sum_{\alpha} \frac{Y_i}{A_0} (\phi_i - \phi_0)$$

Where α is the index set of the neighbors. We will choose the Y_i and A_0 such as to minimize error. Now for the continuous medium (where we denote the location of lattice point i by S_i) each $\phi(S_i)$ is given in terms of $\phi(S_0)$ and the derivatives of ϕ at S_0 by

$$\phi(S_i) = \phi(S_0) + \sum_{n=1}^{\infty} \frac{1}{n!} (x_i \frac{\partial}{\partial x} + y_i \frac{\partial}{\partial y})^n \phi(S_0)$$

We have assigned a Cartesian metric to the domain so that ℓ_i is the pair (x_i, y_i) . Thus the representation $0(\phi_0)$ is given by

$$\sum_{\alpha} Y_i \sum_{n=1}^{\infty} \frac{1}{n!} (x_i \frac{\partial}{\partial x} + y_i \frac{\partial}{\partial y})^n$$

or

$$\sum_{n=1}^{\infty} \frac{1}{n!} \sum_{\alpha} Y_i (x_i \frac{\partial}{\partial x} + y_i \frac{\partial}{\partial y}).$$

Thus

$$\begin{aligned} 0(\phi_0) &= \sum_{\alpha} Y_i x_i \phi_{xo} + \sum_{\alpha} Y_i y_i \phi_{yo} \\ &\quad + \frac{1}{2} \sum_{\alpha} Y_i x_i^2 \phi_{xxo} + \frac{1}{2} \sum_{\alpha} Y_i y_i^2 \phi_{yyo} \\ &\quad + \sum_{\alpha} Y_i x_i y_i \phi_{xyo} + \text{higher order terms} \end{aligned}$$

Now we require

$$\sum_{\alpha} Y_i x_i = 0, \quad \sum_{\alpha} Y_i y_i = 0 \quad (1)$$

(1), above, is the x and y projection of $\sum_{\alpha} Y_i \ell_i$.

$$\sum_{\alpha} Y_i \ell_i = \sum_{\alpha} r_i$$

$\sum_{\alpha} r_i$ projected on x, y is zero since the polygon formed by the r_{α} is closed. Thus, (1) is automatically satisfied. Evidently our choice of Y_i was suitable.

Define

$$\beta_x = \frac{1}{2} \sum_{\alpha} Y_i x_i^2, \quad \beta_y = \frac{1}{2} \sum_{\alpha} Y_i y_i^2, \quad \beta_{xy} = \sum_{\alpha} Y_i x_i y_i$$

Then 0 can be expressed

$$\begin{aligned} 0(\phi_0) &= \beta_x \phi_{xxo} + \beta_y \phi_{yyo} + \beta_{xy} \phi_{xyo} \\ &\quad + \text{higher order terms.} \end{aligned}$$

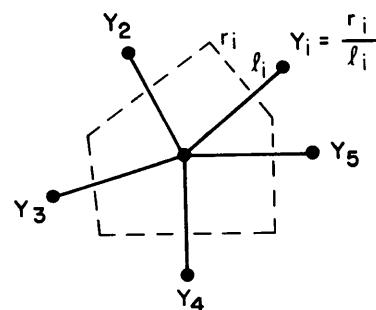
We want to reduce this to canonical form. The result above is independent of an orthogonal transformation on (x, y) . Note that there exists an orthogonal transformation such that $\beta_{xy} = 0$. Henceforth, without loss of generality, we will assume $\beta_{xy} = 0$.

Thus,

$$0(\phi_0) = \beta_x \phi_{xxo} + \beta_y \phi_{yyo} + \text{higher order terms.}$$

$$\begin{aligned} 0(\phi_0) &= \left(\frac{\beta_x + \beta_y}{2} \right) (\phi_{xxo} + \phi_{yyo}) \\ &\quad + \left(\frac{\beta_x - \beta_y}{2} \right) (\phi_{xxo} - \phi_{yyo}) + \text{higher order terms.} \end{aligned}$$

We see that if β_x is significantly different in magnitude from β_y , then 0 is a poor approximation of ∇^2 . Thus we require the polygon formed by the perpendiculars to have relatively good symmetry properties. The higher order terms can be reduced by increasing the lattice density. One can get an intuitive feeling for the symmetry properties of the polygon by observing that the system has a physical interpretation in terms of moments of inertia as shown below.



Y_i = Mass at ℓ_i

All that remains is to find the normalizing constant A_o . We choose

$$A_o = \frac{\beta_x + \beta_y}{2}$$

$$A_o = \frac{1}{4} \sum_{\alpha} Y_i (x_i^2 + y_i^2) = \frac{1}{2} \sum_{\alpha} Y_i l_i^2$$

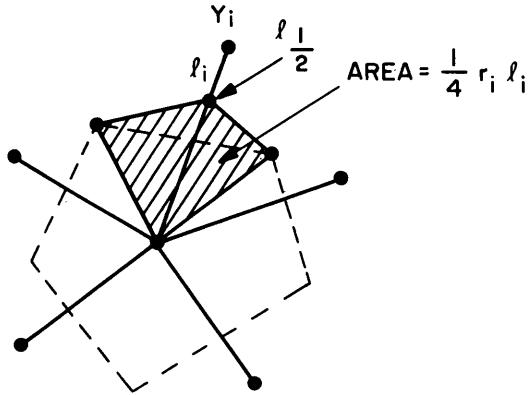
$$A_o = \frac{1}{4} \sum_{\alpha} r_i l_i .$$

Interpretation:

If ℓ_{ij} is a branch from ϕ_i to ϕ_j , then the lumped flux from polygon i to polygon j corresponding to $\nabla\phi$ is $\frac{r_{ij}}{\ell_{ij}} [\phi_i - \phi_j]$ so that $Y_{ij} = \frac{r_{ij}}{\ell_{ij}}$ is a lumped admittance. A_j is the area associated with lattice point j .

MacNeal, R. H., "An Asymmetrical Finite Difference Network", Quarterly Appl Math, October 1953)

Geometrically,



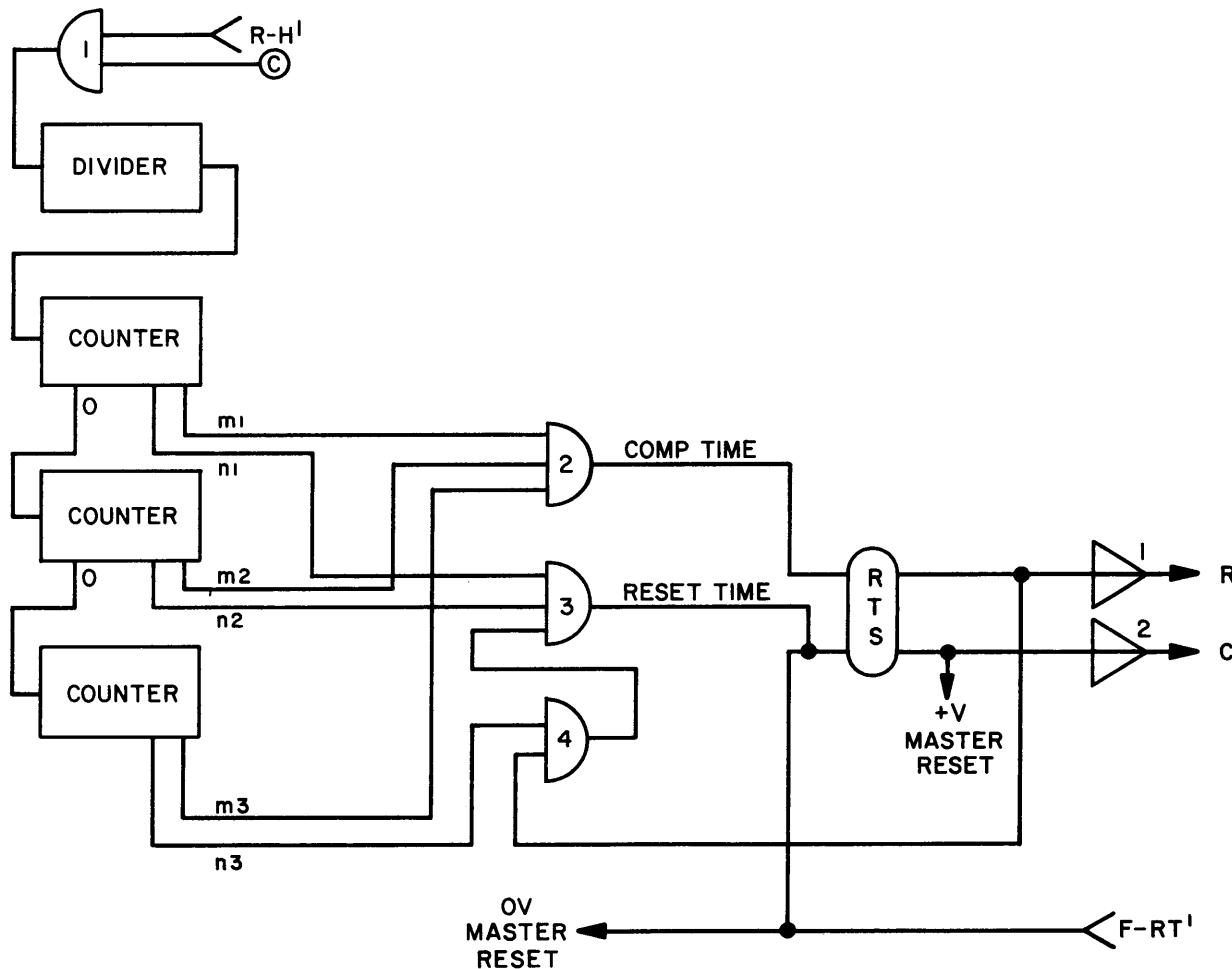


CHAPTER 15

BASIC ITERATIVE PROGRAMMING

REPETITIVE OPERATION

In Repetitive Operation (RO) the entire problem on the computer is solved repetitively. This is useful for determining the influence of parameters on the solution. Potentiometers and switches are operated manually while the computer solves the problem repeatedly. The RO mode cycle is R, C, R, C, etc. The length of the R (initial condition or reset) and C (compute) intervals can be set approximately by the controls at the console. Usually the C interval is adjusted so that the problem solution time is long enough for one to observe the significant part of the solution. The R interval is set so that all integrators return to correct initial values. The RO controls are calibrated only roughly. For a more precise setting of the IC intervals, the program below can be used with the RO selector switch at the console set to EXTERNAL:



The output from the F-RT' terminal resets the divider, counters, and clock, because the logical function (GATE 1) + F-RT' is connected to the 0V reset terminal. Note that RH=F-RT'. The clock is off as long as the F-RT' terminal is a logical 1. The F-RT' terminal is a logical 1 when the main machine mode is R. The FF is reset whenever

$$(GATE\ 1) + F-RT' = 1$$

When the FF is reset all the integrators will be in the R (IC) mode since the R output of the FF is connected through a driver to the R terminal. Each time the FF goes to the S state, the divider and counters will be reset momentarily due to the connection of the inverted output of Gate 2 to the +V terminal. The FF will be set when Gate 2 = 1. The logical equations for the gates are,

$$G1 = m_1 \cdot m_2 \cdot m_3$$

$$G2 = n_1 \cdot n_2 \cdot n_3$$

Thus, the reset and compute intervals are,

$$R = 100m_3 + 10m_2 + m_1$$

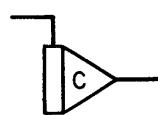
$$C = 100n_3 + 10n_2 + n_1$$

times the period of the divider output.

Note: The connection to patchboard C terminal turns on mode switch compute light during C portion of cycle.

ITERATIVE OPERATION

In iterative operation (IO), results obtained during or at the end of one solution of the problem are used to change parameters or the circuit configuration (switching) for the next solution. The term IO has an obvious derivation from mathematics. In IO, integrators and DAC's are used for memory. Thus, in particular, there is a need for integrators with opposite or complementary logic. These are called complementary integrators. They must be in the R(IC)-mode when the problem is being solved and in the Hold-mode while the normal integrator is in the R-mode. The C-mode can be used instead of the H-mode if the integrator has no compute input, but only an initial condition input.



The chart below shows the relationship between integrator modes

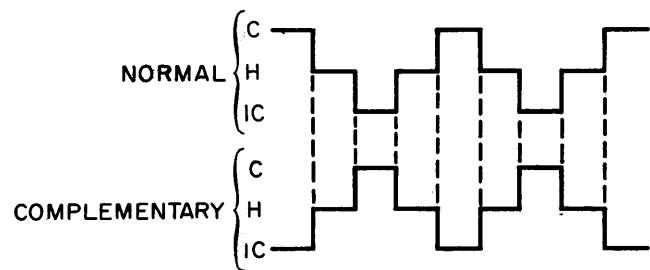
Normal
Integrator

IC
H
C

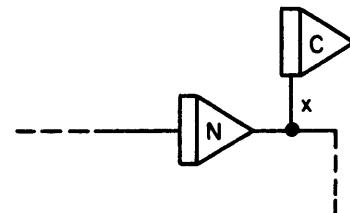
Complementary
Integrator

C
H
IC

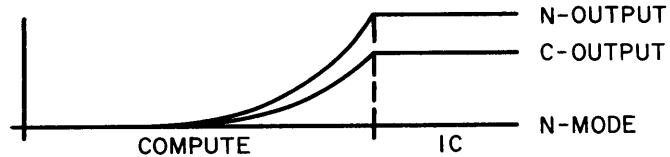
The mode duty cycle for integrators in IO is



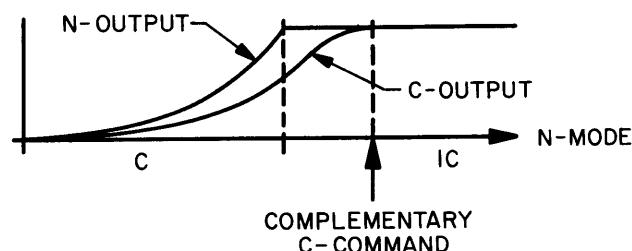
The insertion of the hold interval avoids timing problems between mode switches for normal and complementary integrators. It also has another important purpose as shown below. Suppose a complementary integrator is tracking a problem variable, x,



and x is rapidly changing. Then if the C-integrator goes to compute when the N-integrator goes to initial condition, the C-integrator will not store the final value of the N-integrator due to the time constant of the IC-circuit:



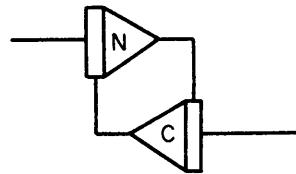
This problem can be solved if the C-mode command to the C-integrator is delayed:



Without the insertion of the hold interval this cannot be accomplished. In some problems, both the N-, and C-integrators are used for computation, in sort of a bang-bang fashion. For example, if there are two sub-simulations, each of which depends on the final values of the other, then both types of integrators are used for equation solution, not merely memory. In this

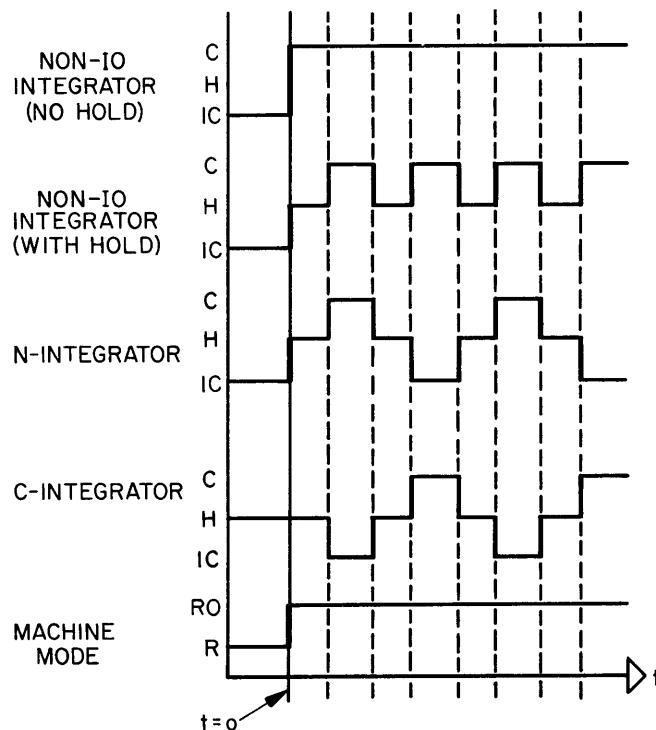
case delayed commands for both types are necessary. If all the variables are sufficiently slow, then the need for delayed commands is obviated. The implementation of mode duty cycles and delayed commands will be discussed presently.

The programmer needs a switch at the console to start IO. This is the RO mode switch when the RO selector switch is in external. It follows that there must be a "safe" mode when not in IO. This is the standby or S-mode and is activated by the reset switch for iterative integrators. Consider the circuit

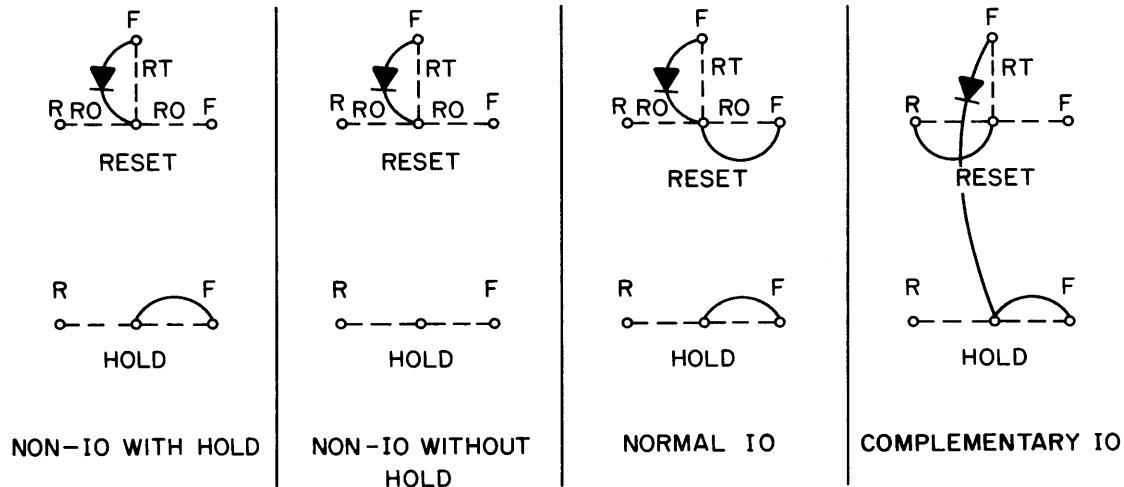


It is plain that the N-integrator must be in IC in the S-mode. The C-integrator cannot also be in IC in the S-mode because the two integrators would form a positive feedback system resulting in instability. The C-integrator cannot be in compute since it may have an initial compute input and consequently would integrate to overload. Thus, the C-integrator must be in hold during the S-mode. Finally, there may be non-IO integrators in the program and these must remain in

Compute or Hold during the iterative cycle. The foregoing can be summarized by



The patching diagram for IO integrator control is



The F-RT terminal is energized when the machine mode is Reset, putting N-integrators in IC and C-integrators in Hold. The F-Hold terminal will be energized when the Hold button is pushed, putting both N-integrators and C-integrators in Hold. There is no conflict with the RO C-, R-terminals since these terminals are disconnected from the patchboard in any mode except RO. When the machine mode is RO, the problem Hold or H-terminal on the patchboard will be connected to

the F-Hold terminal. Consequently, to put both the N- and C-integrators in Hold, it is necessary that the H-terminal be energized and that neither of the RO C-, R-terminals be energized by C, R inputs from the patchboard respectively. The diagram above shows patching for various integrators with mode relays. If electronic mode control is used, then the relays are replaced with electronic switches as shown in Chapter 6.

The logic program for basic IO control is shown below.

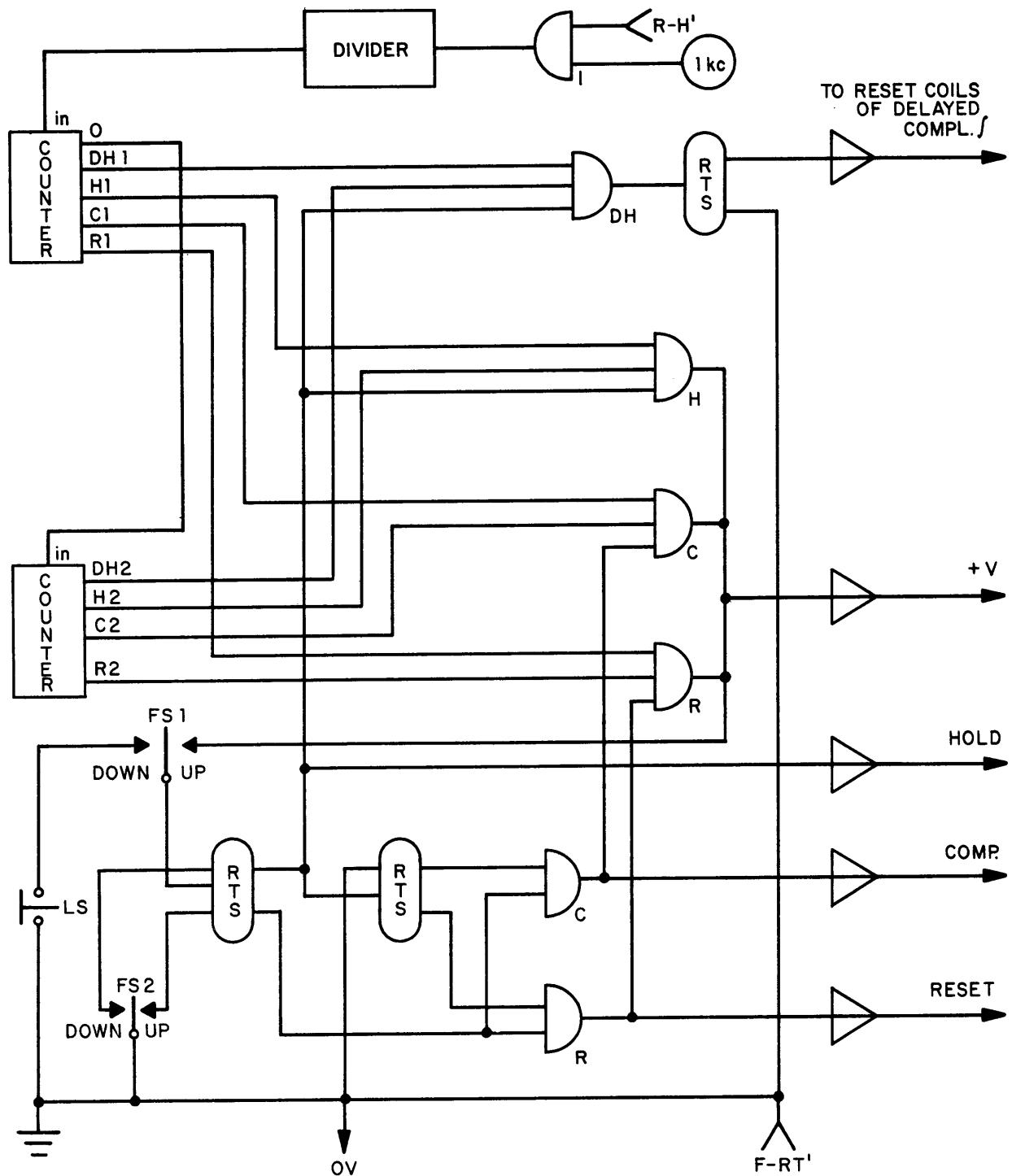


Fig. 1. Iterative Operation Logic Control

Assume FS1 (Function Switch No. 1) is up and FS2 is down. When the machine is in the Reset mode, the output of the F-RT' terminal will be a logical 1. This output is used to reset the divider and counters and to place both FF1 and FF2 in the R-state. The clock is off. FF1 and FF2 form a two-bit binary counter. (The IO cycle H, C, H, IC corresponds to the counter output 00, 01, 10, 11.) Gate H is ready. When the RO mode switch is activated the decade counters will start and both N, C-integrators will be in Hold. At $10H_2 + H_1$

the coincidence at Gate H will reset the decade counters and advance the binary counter to 01. (The use of the output of Gate DH will be discussed later.) The output of the binary counter will ready Gate C, thereby putting the N-integrators in Compute, and the C-integrators in IC by energizing the C-terminal through its driver. At $10C_2 + C_1$ the coincidence at Gate C will reset the decade counter and advance the binary counter to 10. The output of the binary counter will ready gate 1 and

put all integrators in Hold. At $10H_2 + H_1$ the coincidence at Gate H will reset the decade counters and advance the binary counter to 11. The output of the binary counter will ready Gate R, put the C-integrators in Compute, and put the N-integrators in Reset by energizing the R-terminal through its driver. At $10R_2 + R_1$ the coincidence at Gate R will reset the decade counters and advance the binary counter to 00. The circuit is now ready to go through the same cycle again.

If FS2 is up, then the IO cycle will start with C (i.e., it will skip the first and only the first H interval).

Delayed Hold commands are provided for the delayed complementary integrators by gate DH and the associated FF. The logical equation for gate DH is,

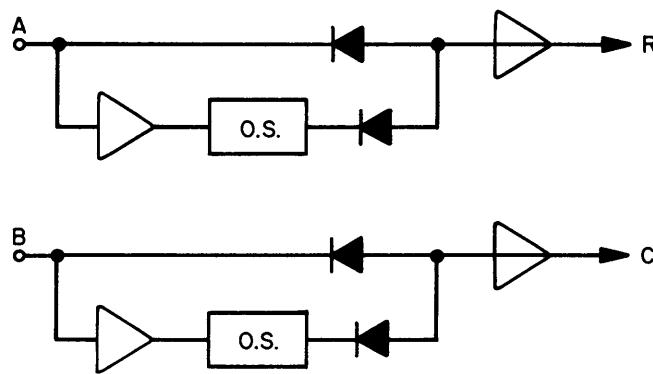
$$DH = H \cdot DH_1 \cdot DH_2 .$$

Note that

$$10DH_2 + DH_1 < 10H_2 + H_1$$

so that the FF will be toggled before the end of each Hold interval. This causes the Reset period of the delayed complementary integrators to bracket the Compute interval of the normal integrators.

Delayed Hold can be obtained by other means. Delayed Hold can be generated for both N-, and C-integrators by delaying the deactivation of the C, R inputs, respectively. The circuits for obtaining these delays are shown in the following diagrams. The inverters are those associated with the preceding gates.

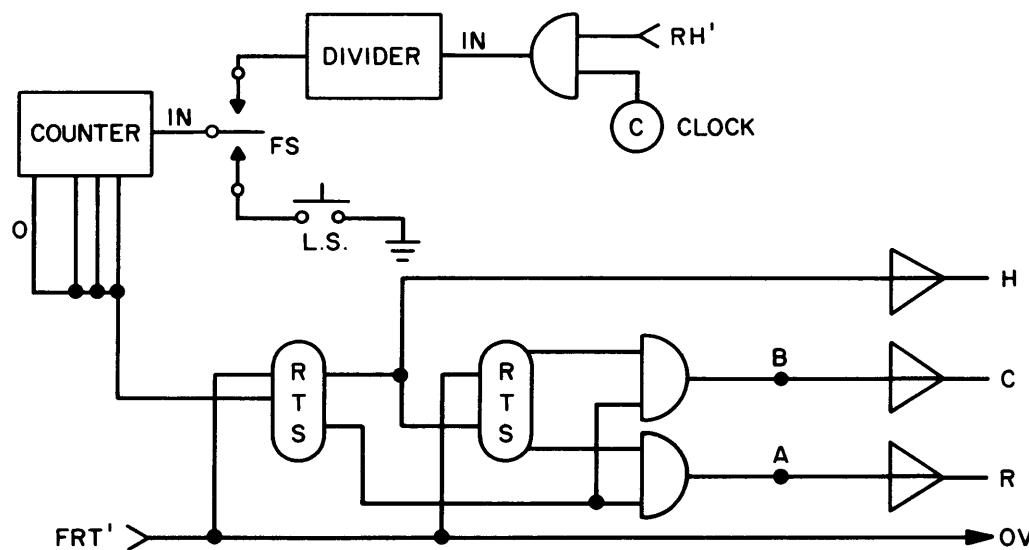


If FS1 is down, the logic switch found on the counter module can be used to single-step the computer through the iterative program.

If the machine Hold switch is depressed, all integrators, IO and non-IO, will go into Hold. This happens because the R, C inputs at the patchboard are disconnected from the F-RO and R-RO terminals in the integrator modules and the main machine mode takes precedence. The clock input to the divider is also stopped because the RH terminal output is a logical 0. This saves the state of the divider, counters, and two-bit binary counter. Consequently, the iterative program can be put in Hold at any time while preserving its status. To return to the iterative program after being in Hold, it is only necessary to activate the RO mode switch.

Sometimes it is desirable to control either the C or the IC-interval or both by an event in the program. This is accomplished by using comparator outputs instead of decade counter outputs as inputs to the appropriate gates.

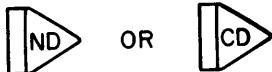
A simpler but less flexible IO logical control program is



This program will permit only an IO cycle composed of 10 time intervals. As before, the RH' output is used to gate the clock.

MODE CONTROL NOTATION

When delayed commands are to be used for an integrator it will be denoted by:



depending on its type

Integrators can be controlled separately by pairs, not under general IO control.

In this case the following notation is used:



R = LOGICAL EXPRESSION (IC RELAY)
H = LOGICAL EXPRESSION (HOLD RELAY)

For example, $R = L_1 \cdot L_2$. The drivers are still used as logical and power buffers. If a resistor rather than a capacitor is used for the feedback element of an integrator amplifier, then the mode control relays can still be used. The notation is

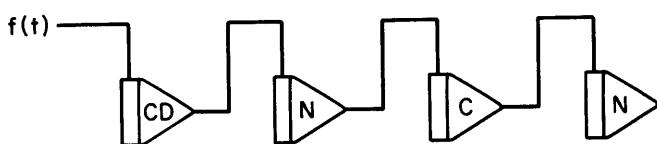


R=LOGICAL EXPRESSION
H=LOGICAL EXPRESSION

Thus, a summer can be made to act as a sampler in a sampled data system by controlling the hold relay.

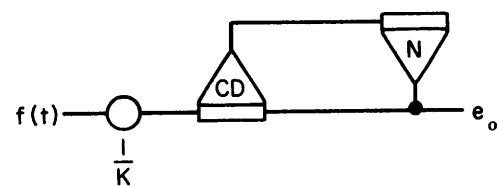
MEMORY CHAIN

A memory chain program is



This circuit will store a sequence of values of f(t). The first integrator requires a delayed command to obtain the final value of f when the program which generates f goes into Hold. The chain can be any length desired.

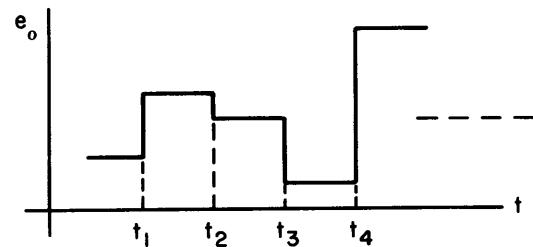
ACCUMULATOR



This program will compute the sum of n successive values of f:

$$e_o = \frac{1}{K} \sum_{i=1}^n f(t_i).$$

The pot is for scaling. The output, e_o , as a function of time is



Note that the complementary integrator has two IC inputs. The second is obtained by patching a resistor between the output of the normal integrator and the LJ terminal of the C-integrator.

AUTOMATIC RESCALING

Sometimes it is necessary to rescale a problem during the computer solution in order to meet the accuracy requirement imposed on the simulation. This occurs because of the wide range of variables involved. One good example of this situation can be found in the simulation of a nuclear reactor start-up, where the neutron level may rise through six decades of power. A technique is presented here for automatic rescaling. Although only a simple example is used for illustration, the technique is applicable to the reactor start-up simulation and many other similar problems.

Suppose it is necessary to generate a function, $f(t)$, where

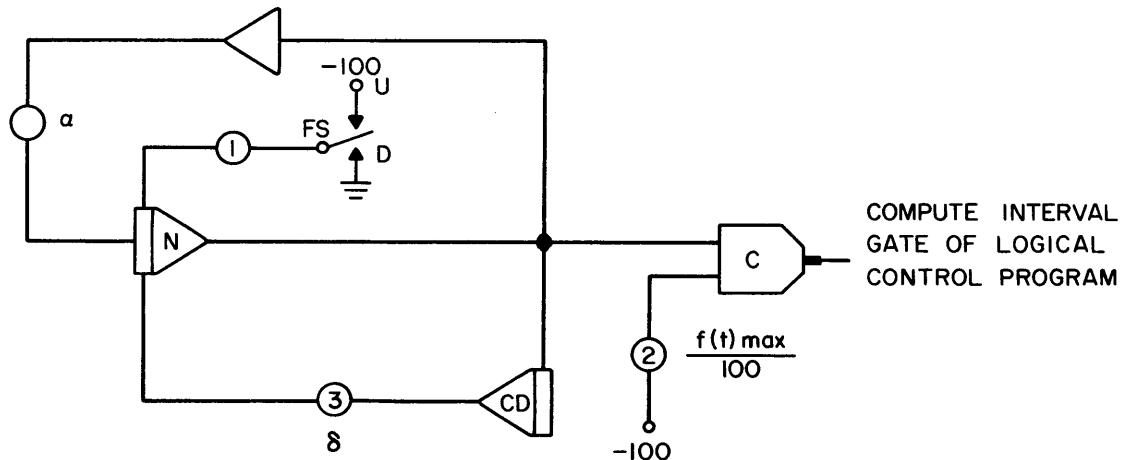
$$f(t) = K_n e^{\alpha t}$$

$$\alpha > 0$$

$$K_n = \text{amplitude scale factor},$$

and as the function exponentially increases and reaches some limit, $f(t)_{\max}$, it is required to perform ampli-

tude rescaling and proceed with the simulation. The following program automatically fulfills these requirements.



The integrator required for the generation of $f(t)$ is a normal integrator. The compute interval is controlled by a comparator. The function switch is in the Up position only at the start of the simulation and the initial IC of the integrator is entered from potentiometer 1. The switch could be replaced by a function relay energized by the FR bus. The IO logic program (Fig. 1 or equivalent) goes through its cycle and waits in the Compute mode until an external signal commands it to go into the Hold mode. Note that the comparator output is used instead of the counter output to activate Gate C (Fig. 1). This signal occurs when the output of the integrator reaches a limit set on potentiometer 2 and the comparator is triggered. A complementary integrator with a delayed command learns this value and feeds it to the IJ terminal of the normal integrator through potentiometer 3 and a resistor R.

As the logic goes to the IC mode, a new initial condition is established and the entire process is repeated.

The amplitude scale factor K_n is given by the following equation

$$K_n = K_1 \frac{(0.05\delta)^n}{R} \quad n = 0, 1, 2, 3----$$

where R is in megohms, δ is the setting of potentiometer 3 and n is the number of scale changes.

FUNCTIONS OF A DEPENDENT VARIABLE

This technique is not highly accurate, but is useful if precision is not a requirement. Suppose we want to generate

$$f(x), \quad x = x(t)$$

If f is the solution of an ordinary differential equation

$$L(D)f(t) = c(t)$$

Where $L(D)$ is a differential operator, then

$$f(x) = \int^x L^{-1}(D)c(t)dt.$$

We see that time can be used as a dummy integration variable. Suppose $g(x)$ is the solution of

$$M(d)g(x) = b(t).$$

That is

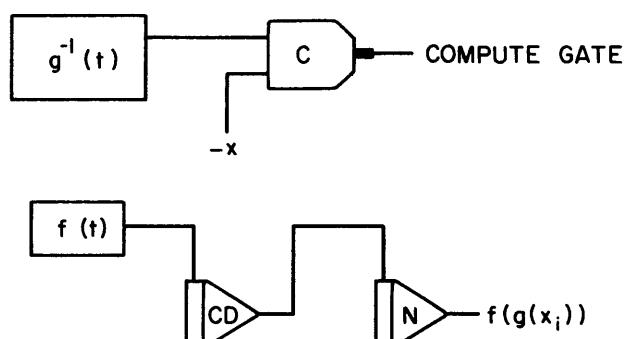
$$g(x) = \int^x M^{-1}(D)b(t)dt.$$

Then $f\{g(x)\}$ can also be generated, using time as a dummy variable, from the simultaneous solution of

$$f(g(x)) = \int^{g(x)} L^{-1}(D)c(t)dt$$

$$g(x) = \int^x M^{-1}(D)b(t)dt.$$

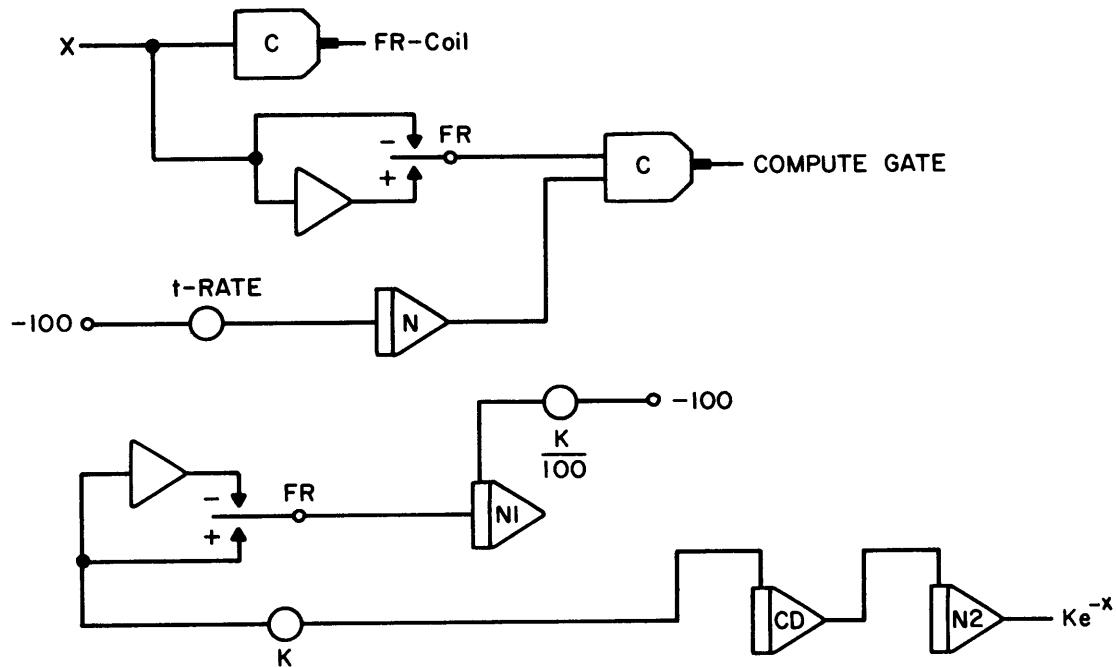
The program for the generation of $f(g(x))$ is,



The comparator is energized when $g^{-1}(t) = x$, so that $t = g(x)$. Therefore, the output of the f circuit at this time will be

$$f(g(x_2)).$$

Example: $f(x) = Ke^{-x}$, $-b \leq x \leq b$

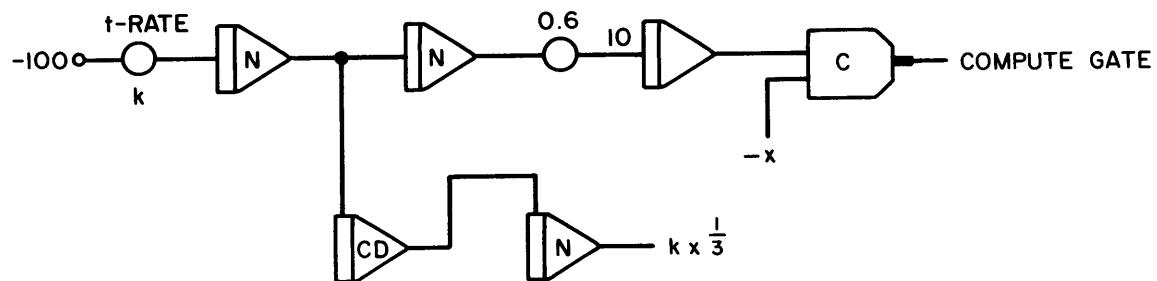


The comparator will activate when $t = x$. The output of integrator N1 is Ke^{-t} so that the output of integrator N2 is Ke^{-x} . If these IO integrators compute at high speed, say with $0.001 \mu\text{fd}$ capacitors and electronic mode control, the output of N2 is a closed stair-

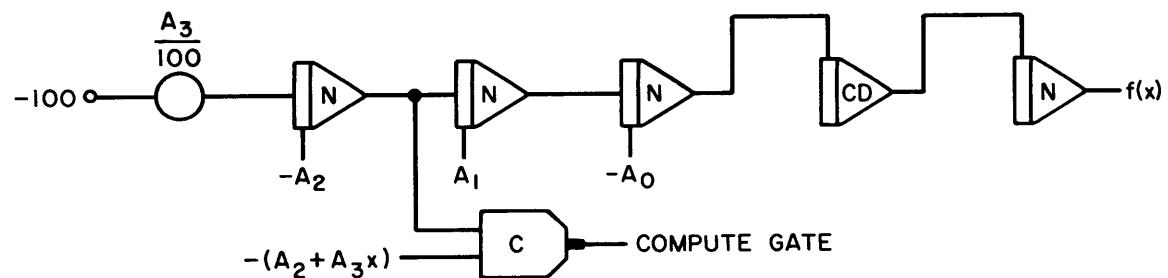
case approximation to $Ke^{-x(t)}$ for slowly varying $x(t)$. This IO program can be used as a subroutine for a non-IO main program.

The same general comments apply to the following examples.

Example: $f(x) = x^{\frac{1}{3}}$, $x \geq 0$

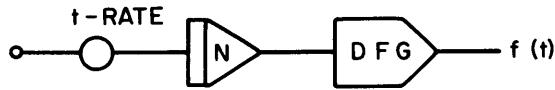


Example: $f(x) = A_0 + A_1 x + \frac{A_2}{2!} x^2 + \frac{A_3}{3!} x^3$, $x \geq 0$.



PERIODIC FUNCTION GENERATION

Suppose a non-IO program requires a periodic function. We can use an IO integrator and a function generator to provide the periodic function. The function sub-program is



Here the N-integrator is under regular IO control. The main program will see the output of the DFG as a periodic function because its non-IO integrators are in Hold during the time the N-integrator is in IC and Hold. In fact, the Hold can be eliminated if there is no other IO calculation to be performed. The non-IO integrator must have the logical control



$$R = F - RT \\ H = F - H$$

so that it will be in Hold, not IC, when the IO integrator is in reset.

NEWTON'S INTERPOLATION FORMULA

$$f_i \triangleq f(x_i)$$

$$f[x_0] \triangleq f_0$$

$$f[x_0, x_1] \triangleq \frac{f_1 - f_0}{x_1 - x_0}$$

$$f[x_0, x_1, x_2] = \frac{f[x_1, x_2] - f[x_0, x_1]}{x_2 - x_0}$$

and generally

$$f[x_0, x_1, \dots, x_n] =$$

$$\frac{f[x_1, x_2, \dots, x_n] - f[x_0, x_1, \dots, x_{n-1}]}{x_n - x_0}$$

We have

$$f(x) = f[x_0] + (x-x_0)f[x_0, x] \quad \text{----- (1)}$$

$$f[x_0, x] = f[x_0, x_1] + (x-x_1)f[x_0, x_1, x] \quad \text{----- (2)}$$

and generally

$$f[x_0, \dots, x_{n-1}, x] = f[x_0, \dots, x_n] + (x-x_n)f[x_0, \dots, x_n, x].$$

Substituting (2) in (1)

$$f(x) = f[x_0] + (x-x_0)f[x_0, x_1] + (x-x_0)(x-x_1)f[x_0, x_1, x]$$

and

$$f(x) = f[x_0] + (x-x_0)f[x_0, x_1] + \dots + (x-x_0)\dots(x-x_{n-1})f[x_0, \dots, x_n] + E(x)$$

where

$$E(x) = (x-x_0)\dots(x-x_n)f[x_0, \dots, x_n, x].$$

If f has polynomial form, then for sufficiently large n , $E=0$. A good approximation for f , in any event, is obtained by assuming $E=0$. The result is called Newton's interpolation formula. This formula can be used to provide a continuous representation of a function defined at n points. The intervals x_i, x_{i-1} need not be equal. However, to simplify our derivation we will assume that these intervals are equal. It should be remembered that the continuous memory to be developed can be used for unequal sample intervals with different values for coefficients in the computer program. We let $x_0=0$ and assume all $x_i - x_{i-1} = a$.

Then

$$f[x_0, \dots, x_n] = \frac{(E-1)^n f_0}{n! a^n}, \quad E^n f_0 \triangleq f_n$$

and

$$f(x) = f_0 + \sum_{m=1}^n \frac{(E-1)^m f_0}{m! a^m} \prod_{i=0}^{m-1} (x-x_i)$$

so that

$$f\left(\frac{x}{a}\right) = f_0 + \sum_{m=1}^n \frac{(E-1)^m f_0}{m!} \prod_{i=0}^{m-1} \left(\frac{x}{a} - i\right) \quad (3)$$

For a specific number, k , of $E^k f_0$ a representation of $f\left(\frac{x}{a}\right)$ can be obtained for (3). For $n=3$, we have

$$f\left(\frac{x}{a}\right) = f_0 + (E-1)f_0\left(\frac{x}{a}\right) + \frac{(E-1)^2 f_0}{2}\left(\frac{x}{a} - 1\right) + \frac{(E-1)^3 f_0}{6}\left(\frac{x}{a}\right)\left(\frac{x}{a} - 1\right)\left(\frac{x}{a} - 2\right).$$

collecting terms

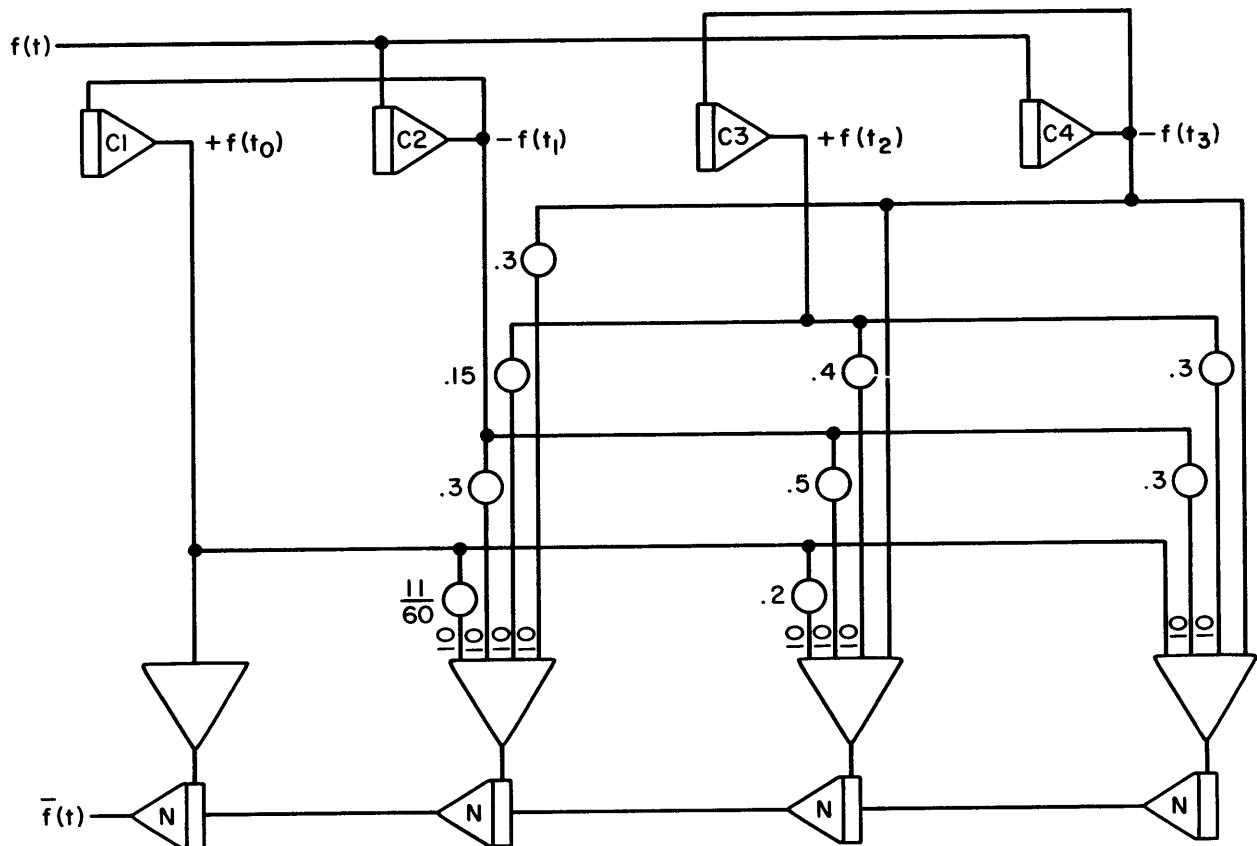
$$f(\frac{x}{a}) = f_0 + \left[\frac{(E-1)^2 f_0}{2} + \frac{2(E-1)^3 f_0}{6} \right] \left(\frac{x}{a} \right) - \\ + \left[\frac{(E-1)^2 f_0}{2} - \frac{3(E-1)^3 f_0}{6} \right] \left(\frac{x}{a} \right)^2 - \\ + \frac{(E-1)^3 f_0}{6} \left(\frac{x}{a} \right)^3$$

$$f(x) = f_0 + \left(\frac{1}{3} f_3 - \frac{3}{2} f_2 + 3f_1 - \frac{11}{6} f_0 \right) x \\ + \left(-\frac{1}{2} f_3 + 2f_2 - \frac{5}{2} f_1 + f_0 \right) x^2 \\ + \left(\frac{1}{6} f_3 - \frac{1}{2} f_2 + \frac{1}{2} f_1 - \frac{1}{6} f_0 \right) x^3$$

$$f(x) = f_0 + \left(\frac{1}{3} f_3 - \frac{3}{2} f_2 + 3f_1 - \frac{11}{6} f_0 \right) x \\ + \left(-f_3 + 4f_2 - 5f_1 + 2f_0 \right) \frac{x^2}{2!} \\ + \left(f_3 - 3f_2 + 3f_1 - f_0 \right) \frac{x^3}{3!}$$

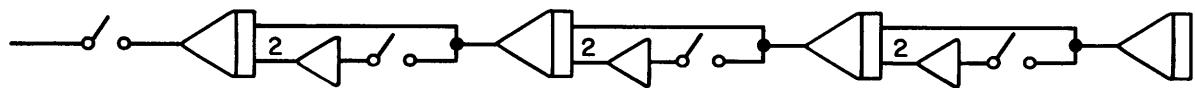
CONTINUOUS MEMORY

In the analog computer we can retain a continuous representation, \bar{f} , of f by sampling f at n points (not necessarily uniformly spaced). \bar{f} can be generated later on command with the interpolation formula. The computer program which implements this operation from four uniformly sampled values of f is shown below.



Comparators can be used in conjunction with one ramp integrator to de-energize the R-relays of amplifiers 1, 2, 3, 4 in the proper time sequence. The integrators are switched from the initial condition mode to the compute mode when it is desired to "playback" $\bar{f}(t)$.

If, instead of $\bar{f}(t)$, it is desired to playback $\bar{f}(T-t)$, the summers and switches are added to the integrator chain as shown below.



The procedure is the same as before except that the switches are energized at $t=T$.

Although only the circuit for four uniform samples of f is shown, the technique can be extended to the case of n non-uniform samples. The pot coefficients can be calculated previously with a digital computer or desk calculator.



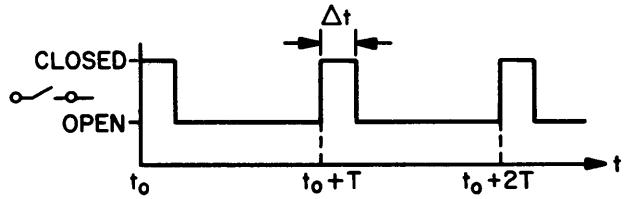
CHAPTER 16

SAMPLED-DATA SYSTEM SIMULATION

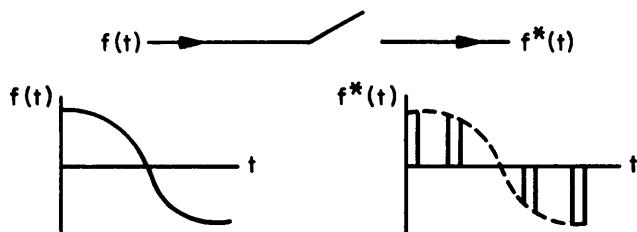
Sampled-data systems result from the utilization of telemetry, digital control, multiplexed controllers and other devices of a similar nature in a control system. The analog computer is a useful tool for the study, evaluation and synthesis of both linear and non-linear sampled-data systems.

THE SAMPLER

The single-rate sampler is a switch whose periodic duty cycle is shown below.



If $f(t)$ is sampled by this switch, the output function is denoted $f^*(t)$.



In most sampled-data systems, Δt is small compared to T . In classical analysis Δt is assumed to approach zero with finite area, so that $f^*(t) = f(0), f(T), \dots$. The introduction of this assumption may or may not significantly alter system behavior. When the analog computer is used for analysis, the sampler can be simulated exactly. The H-relays for the integrator amplifiers can be used to represent samplers. The duty cycle of these relays can be automatically implemented with the digital logic.

The sampling operation produces sidebands.

$$Lf^* = \frac{1}{T} \sum_{n=-\infty}^{\infty} F(s + jn\omega_s),$$

(L indicates Laplace transform)

where $\omega_s = \frac{2\pi}{T}$. In order that the sidebands do not overlap, we must have

$$Ff = 0, |\omega| \geq \frac{\pi}{T}.$$

(The operator F is the Fourier transform.)

If there is sideband overlap it is not possible (with techniques considered here) to recover $f(t)$ from $f^*(t)$. Of course, in practice, it is usually not necessary to recover $f(t)$ exactly. Consequently, judgment is required to determine the extent of allowable sideband overlap. This is one question which is conveniently answered with the analog computer.

FUNCTION RECOVERY

An ideal filter for function recovery has the Fourier transform

$$\begin{aligned} T(\omega) &= A(\omega)e^{j\phi(\omega)} \\ A(\omega) &= 1, |\omega| \leq \omega_c \\ &= 0, |\omega| > \omega_c \\ \phi(\omega) &= -k\omega. \end{aligned}$$

Thus, if $F(\omega)$ is the Fourier spectrum of $f(t)$, and

$$F(\omega) = 0, |\omega| \geq \omega_c$$

then the Fourier transform of the recovered function, $F'(t)$, is

$$F'(\omega) = T(\omega)F^*(\omega) = F(\omega)e^{-jk\omega}$$

so that

$$f'(t) = \frac{1}{T} f(t - k).$$

$T(\omega)$ represents an ideal low-pass filter and a "cardinal data hold" if $k = 0$.

The impulse response of the ideal low-pass filter is

$$T(t) = \frac{\omega_c}{\pi} \frac{\sin [\omega_c(k-t)]}{\omega_c(k-t)}.$$

The filter is not physically realizable due to the precursor (response for $t < 0$). Ideally $\omega_1 < \omega_c < (\omega_s - \omega_1)$ where $\omega_s \geq 2\omega_1$. ω_s should be sufficiently greater than $2\omega_1$ so that an adequate practical filter can be found.

The foregoing applies to piecewise continuous functions. Special filter techniques such as clamping can be used for discontinuous functions¹. Hold circuits are useful approximations of the ideal low-pass filter.

Zero-order hold filter:

Let the zero-order filter satisfy the relationship

$$f(nT + t) = f(nT), \quad 0 \leq t < T.$$

That is, the hold circuit senses the value of f at $t = nT$ and holds this value until a new value is sensed at $t = nT + T$. Suppose $f(t) = u_0(t)$ so that $F(s) = \frac{1}{s}$.

Then

$$F^*(s) = \sum_{n=0}^{\infty} e^{-nsT} = \frac{1}{1 - e^{-sT}}$$

If the transfer function for the zero-order hold circuit is $T(s)$, then

$$T(s)F^*(s) = \frac{1}{s}$$

$$\Rightarrow T(s) = \frac{1}{sF^*(s)} = \frac{1 - e^{-sT}}{s}$$

Thus

$$T(j\omega) = \frac{1 - e^{-j\omega T}}{j\omega}$$

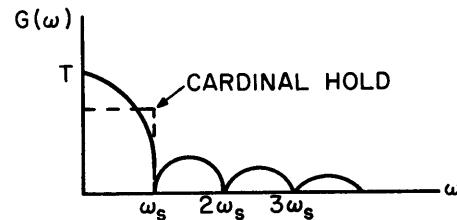
$$T(j\omega) = Te^{\frac{-j\omega T}{2}} \left[\begin{matrix} \frac{j\omega T}{2} & \frac{-j\omega T}{2} \\ e^{\frac{j\omega T}{2}} & -e^{\frac{-j\omega T}{2}} \end{matrix} \right] \frac{2j(\frac{\omega T}{2})}{2j(\frac{\omega T}{2})}$$

¹M. Schwartz, "Information Transmission, Modulation, and Noise," McGraw-Hill, 1959.

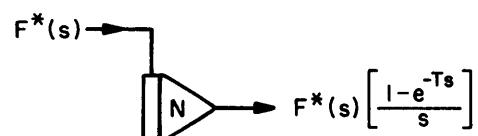
$$T(j\omega) = Te^{\frac{-j\omega T}{2}} \left[\begin{matrix} \sin\left(\frac{\omega T}{2}\right) \\ \frac{\omega T}{2} \end{matrix} \right] = G(\omega)e^{j\phi(\omega)}$$

$$G(\omega) = T \left[\begin{matrix} \sin\left(\frac{\omega T}{2}\right) \\ \frac{\omega T}{2} \end{matrix} \right]$$

$$\phi(\omega) = -\frac{\omega T}{2}$$



A computer circuit which simulates a practical zero-order hold filter is shown below. Integrator modes are controlled by the IO logic.



If an ideal hold filter is to be simulated, then the main computation must be interrupted while the memory loads.

First-order hold filter:

For the first-order hold filter, we have the predictor

$$f(nT + t) = f(nT) + \frac{t}{T} [f(nT) - f(nT - T)], \quad 0 \leq t < T.$$

If $f(nT) = 1$ for all n , then the output $o(t)$ from the filter is

$$\begin{aligned} o(t) &= 1 + \frac{t}{T}, \quad 0 \leq t < T \\ &= 1, \quad t \geq T. \end{aligned}$$

Thus, if the transfer function for the filter is $T(s)$, then

$$O(s) = T(s) F^*(s)$$

$$F^*(s) = \frac{1}{1 - e^{-sT}}$$

$$O(s) = \left(\frac{1}{s} + \frac{1}{Ts^2} \right) (1 - e^{-sT})$$

$$\rightarrow T(s) = \left(\frac{1}{s} + \frac{1}{Ts^2} \right) (1 - e^{-sT})^2$$

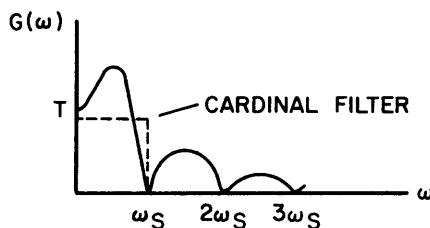
It can be shown that if $T(j\omega)$ is expressed

$$T(j\omega) = G(\omega)e^{j\phi(\omega)}$$

then

$$G(\omega) = \sqrt{\omega^2 + 1} \quad \left[\frac{\sin \frac{\omega T}{2}}{\frac{\omega T}{2}} \right]^2$$

$$\phi(\omega) = -\omega T + \tan^{-1} \omega$$



K-th order hold filter:

The predictor for the k-th order hold filter is

$$f(nT + t) = \sum_{m=0}^k \frac{t^m}{m!} \Delta^m f(nT), \quad 0 \leq t < nT$$

where Δ is an operator such that

$$\Delta f(nT) = \frac{1}{T} f(nT - T).$$

Practical low-pass filters:

Recovery can also be accomplished with a variety of low-pass filters. The analog simulation of the sampled-data system usually will provide a means of deciding the filtering requirements. Various types of filters may be tried in the simulation.

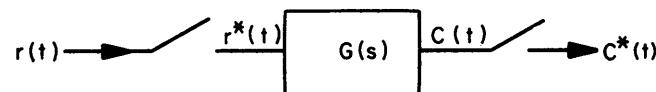
RIPPLE

Various responses in the sampled-data system may contain components that are either completely or partially hidden by the sampling process. These are called ripple and may have a bearing on system stability. In classical analysis, the system response between sampling instants is examined either by the

introduction of fictitious samplers or the use of the modified Z-transform. The analog computer is an effective tool for evaluation of ripple since the continuous time response of all elements of the system is available for observation.

PULSE TRANSFER FUNCTION

In what follows, it will be assumed that all samplers are synchronized and have the same period T . In order to find the pulse transfer function of an element of a sampled-data system, consider the simple system below



We have

$$C(s) = G(s) R^*(s)$$

$$C^*(s) = \frac{1}{T} \sum_{n=-\infty}^{\infty} C(s + jn\omega_s)$$

$$C^*(s) = \frac{1}{T} \sum_{n=-\infty}^{\infty} G(s + jn\omega_s) R^*(s + jn\omega_s)$$

Note that $R^*(s + jn\omega_s)$ is periodic with period ω_s . (Consider the sideband structure.)

Thus

$$R^*(s + jn\omega_s) = R^*(s), \text{ for all } n$$

$$C^*(s) = R^*(s) \frac{1}{T} \sum_{n=-\infty}^{\infty} G(s + jn\omega_s) \dots \text{Equ. 1}$$

$$C^*(s) = G^*(s) R^*(s)$$

From the relationship between the Laplace and Z-transforms, it follows

$$G(z) = \frac{C(z)}{R(z)}$$

Where $G(z)$ is the pulse transfer function and is the Z-transform of $g(t)$, the impulse response. For convenience, we will use $F(s)$, $F(z)$ to denote the Laplace and Z-transforms of $f(t)$, respectively, although $F(s)$ and $F(z)$ have different functional form in general.

PULSE TRANSFER FUNCTION OF VARIOUS LOOPS

For convenience, we define

$$Z\{F(s)\} \triangleq Z\{L^{-1} F(s)\} .$$

The system



(see equation (1) above). Thus

$$\frac{C(z)}{R(z)} = Z \{G_1(s)G_2(s)\} .$$

It is convenient to denote $Z \{G_1(s)G_2(s)\}$ by $G_1G_2(z)$. It is important to note that $G_1G_2(z) \neq G_1(z)G_2(z)$.

satisfies

$$C(s) = G_1(s)G_2(s)R^*(s)$$

$$C^*(s) = [G_1(s)G_2(s)R^*(s)]^* .$$

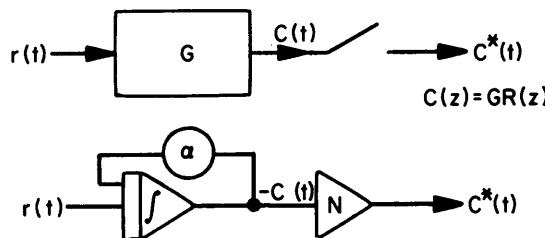
$$C^*(s) = [G_1(s)G_2(s)]^* R^*(s)$$

The computer block programs for the simulation of various simple sampled-data systems together with the closed-loop pulse transfer function are shown in the examples below. In each case the sampling function is implemented with mode relays under IO logic

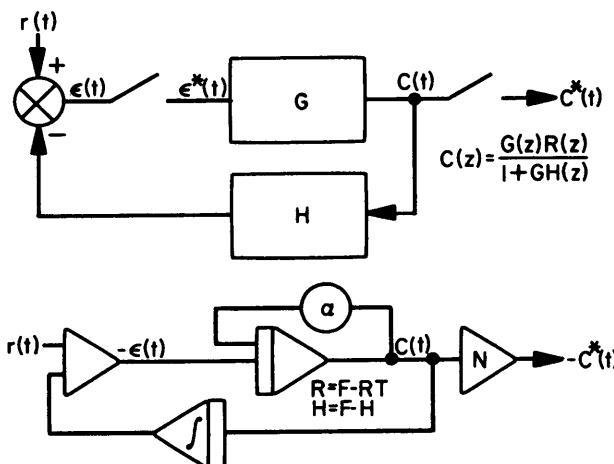
control. In all cases $G = \frac{1}{s + \alpha}$, $H = \frac{1}{s}$.

(The symbol \int for an integrator means it is not under IO control. The normal IO logical program is used for the sampling amplifiers).

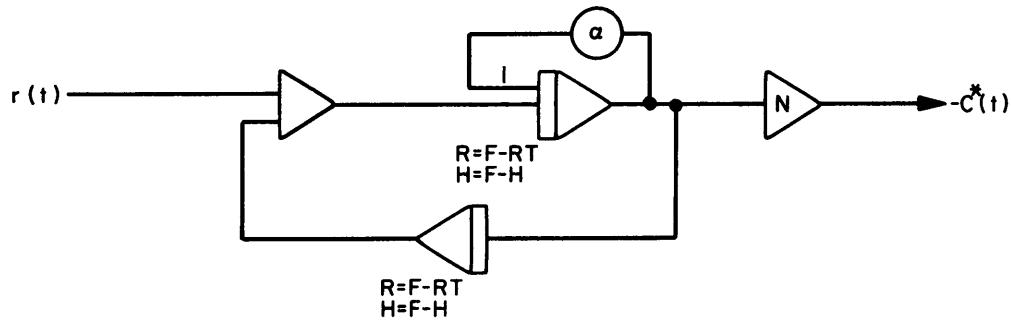
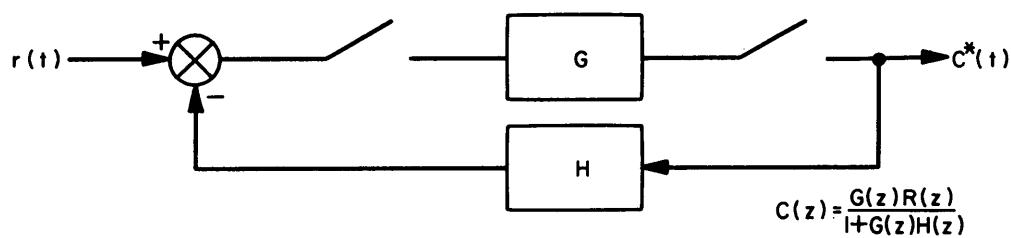
Example:



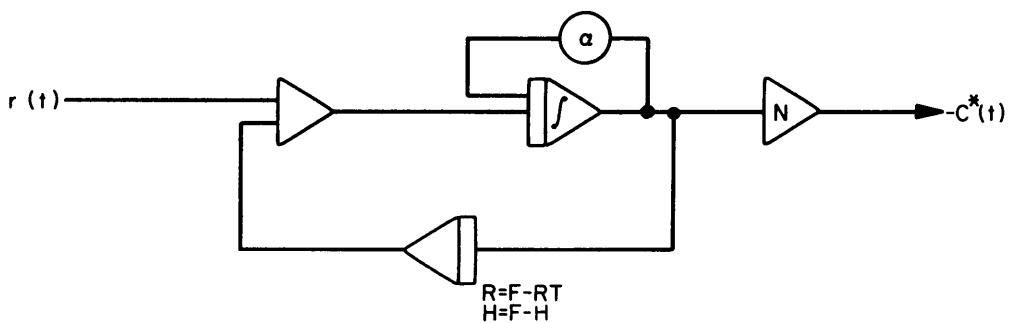
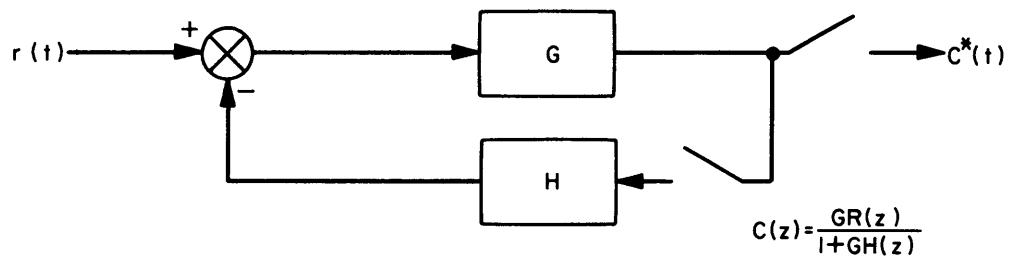
Example:



Example:



Example:



DESIGN TECHNIQUES

Root-locus:

Root-locus plotting techniques apply without modification in the z -plane. The region of stability is $|z| < 1$. The interpretation of the locus is somewhat different. The family of circles

$$|z| = \sigma = \alpha T$$

represent constant damping, α , (not damping ratio) for the response

$$f^* = [e^{-\alpha t} \cos \omega t] *$$

The family of radii

$$\text{Arg } z = \theta = \omega T$$

represent constant frequency for the above response.

Thus, ζ , the damping ratio is given by

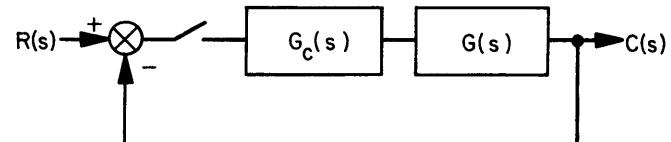
$$\zeta = \sin^{-1} \frac{\alpha}{\sqrt{\alpha^2 + \omega^2}}$$

or equivalently

$$\zeta = \sin^{-1} \frac{\sigma}{\sqrt{\sigma^2 + \theta^2}}$$

Pole-zero cancellation techniques:

Suppose the desired pulse transfer function for the controller



(where $G_c(s)$ is the compensating function) is given by $T(z)$. Then since

$$T(z) = \frac{C(z)}{R(z)} = \frac{G_c G(z)}{1 + G_c G(z)}$$

we have

$$G_c G(z) = \frac{T(z)}{1 - T(z)} .$$

This technique is useful when G_c is realizable.

Digital Compensation:

A digital controller can be used instead of a compensation network above. This controller can be simulated with the analog computer if it is sufficiently simple. If not, then a hybrid system may be used with the analog computer simulating the remaining sampled-data system.

MULTIRATE-ASYNCHRONOUS SYSTEMS

Multirate and asynchronous sampled-data systems are at best difficult to analyze with classical techniques. The analog computer, with its asynchronous control structure finds powerful application for such problems.

CHAPTER 17

PARTIAL DIFFERENTIAL EQUATIONS – PART 2

PROPAGATION OPERATOR

There is a class of equations which can be solved with the similarity transformation

$$\phi(R) = (T^{-1} \Lambda^x T) \phi(\partial R)$$

where T is a suitably defined integral transform over ∂R (boundary of the region, R) and propagation into R is along the x^i coordinate of R . That is, the surfaces $x^i = \text{constant}$ are "concentric" to ∂R . $(T^{-1} \Lambda^x T)$ is called the propagation operator and Λ the propagation function.

That $T^{-1} \Lambda^x T$ exists, can be shown if R is homogeneous in the boundary coordinates. That is, usually R can be non-homogeneous only in x^i . We will use this approach to find Green's functions for the solution of differential equations where we know the required integral transforms over ∂R .

First we derive Λ for a few simple cases for the purpose of illustration.

Consider the cases:

$$u_{yy} = -u_{xx} \quad (1)$$

$$u_{yy} = u_x \quad (2)$$

$$u_{yy} = u_{xx} \quad (3)$$

We will use the Fourier transform and its inverse. We claim (for propagation along the x coordinate)

$$\begin{aligned} u(x, y) &= T^{-1} \Lambda^x G(\omega) \\ &= \frac{1}{2\pi} \int_{-\infty}^{\infty} \Lambda^x G(\omega) e^{j\omega y} d\omega. \end{aligned}$$

Then,

$$u_{yy} = \frac{1}{2\pi} \int_{-\infty}^{\infty} -\omega^2 \Lambda^x G(\omega) e^{j\omega y} d\omega$$

$$u_{xx} = \frac{1}{2\pi} \int_{-\infty}^{\infty} \ln^2 \Lambda^x G(\omega) e^{j\omega y} d\omega$$

$$u_x = \frac{1}{2\pi} \int_{-\infty}^{\infty} \ln \Lambda^x G(\omega) e^{j\omega y} d\omega$$

For (1) above

$$u_{yy} + u_{xx} = \frac{1}{2\pi} \int_{-\infty}^{\infty} (-\omega^2 + \ln^2 \Lambda) G(\omega) d\omega = 0$$

$$\Lambda^x e^{j\omega y} G(\omega) d\omega = 0$$

for arbitrary G . A sufficient condition for the vanishing of the integral is

$$\ln^2 \Lambda = \omega^2$$

$$\Lambda = e^{-|\omega|}$$

and

$$u(x, y) = T^{-1} e^{-|\omega| x} G(\omega)$$

$$u(0, y) = T^{-1} G(\omega)$$

so that

$$G(\omega) = (T) u(0, y).$$

Consequently

$$u(x, y) = (T^{-1} e^{-|\omega| x} T) u(0, y)$$

and $(T^{-1} e^{-|\omega| x} T)$ is the desired propagation operator.

For (2) we have

$$\frac{1}{2\pi} \int_{-\infty}^{\infty} (-\omega^2 - \ln \Lambda) e^{j\omega y} G(\omega) d\omega = 0$$

so that for arbitrary G

$$\Lambda = e^{-\omega^2}$$

and the operator is

$$T^{-1} e^{-\omega^2 x} T.$$

For (3) we have

$$-\omega^2 = \ln^2 \Lambda$$

$$\ln \Lambda = \pm j\omega$$

$$\Lambda_1 = e^{j\omega}, \quad \Lambda_2 = e^{-j\omega}$$

Let

$$u(x, y) = T^{-1} \Lambda_1^x G_1(\omega) + T^{-1} \Lambda_2^x G_2(\omega)$$

$$= T^{-1} e^{j\omega x} G_1(\omega) + T^{-1} e^{-j\omega x} G_2(\omega)$$

Then

$$u_x(x, y) = T^{-1} j\omega e^{j\omega x} G_1(\omega) - T^{-1} j\omega e^{-j\omega x} G_2(\omega)$$

$$u(0, y) = T^{-1} G_1(\omega) + T^{-1} G_2(\omega)$$

$$u_x(0, y) = T^{-1} j\omega G_1(\omega) - T^{-1} j\omega G_2(\omega)$$

$$G_1(\omega) + G_2(\omega) = (T) u(0, y)$$

$$G_1(\omega) - G_2(\omega) = \frac{1}{j\omega} (T) u_x(0, y)$$

$$G_1(\omega) = \frac{1}{2} \left[(T) u(0, y) + \frac{1}{j\omega} (T) u_x(0, y) \right]$$

$$G_2(\omega) = \frac{1}{2} \left[(T) u(0, y) - \frac{1}{j\omega} (T) u_x(0, y) \right]$$

$$u(x, y) = \left(T^{-1} \frac{e^{j\omega x}}{2} (T) \right) u(0, y)$$

$$+ \left(T^{-1} \frac{e^{j\omega x}}{2j\omega} (T) \right) u_x(0, y)$$

$$+ \left(T^{-1} \frac{e^{-j\omega x}}{2} (T) \right) u(0, y)$$

$$- \left(T^{-1} \frac{e^{-j\omega x}}{2j\omega} (T) \right) u_x(0, y)$$

$$u(x, y) = \left(T^{-1} \cos \omega x (T) \right) u(0, y) + \left(T^{-1} \frac{\sin \omega x}{\omega} (T) \right) u_x(0, y).$$

Let's examine one more case

$$\frac{\partial^2}{\partial y^2} (EI \frac{\partial^2 u}{\partial y^2}) + m \frac{\partial^2 u}{\partial x^2} = 0.$$

Let EI, m be constants and define

$$k = \frac{m}{EI}$$

so that

$$u_{yyyy} + k u_{xx} = 0$$

then

$$\omega^4 + k \ln^2 \Lambda = 0$$

$$\ln^2 \Lambda = \frac{j\omega^2}{\sqrt{k}}, \quad \frac{-j\omega^2}{\sqrt{k}}$$

$$\Lambda_1 = \frac{e^{j\omega^2 x}}{\sqrt{k}}, \quad \Lambda_2 = \frac{e^{-j\omega^2 x}}{\sqrt{k}}$$

and we see from the last case

$$u(x, y) = \left(T^{-1} \cos \left(\frac{\omega^2 x}{\sqrt{k}} \right) (T) \right) u(0, y)$$

$$+ \left(T^{-1} \left[\frac{\sin \left(\frac{\omega^2 x}{\sqrt{k}} \right)}{\omega^2} \right] (T) \right) u_x(0, y)$$

SUMMARY FOR POLAR COORD'S (r, θ).

$$(T) f(\theta) = \int_0^{2\pi} f(\xi) e^{-jn\xi} d\xi = F(n)$$

$$T^{-1} F(n) = \frac{1}{2\pi} \sum_{n=-\infty}^{\infty} F(n) e^{jn\theta} = f(\theta)$$

For $\nabla^2 \phi = 0$ we have

$$u(r, \theta) = \left(T^{-1} r^{-|n|} (T) \right) u(0, \theta)$$

For $\nabla^2 \phi + k^2 \phi = 0$ we have

$$u(r, \theta) = \left(T^{-1} \left[\frac{J_n(kr)}{J_n(k)} \right] (T) \right) u(0, \theta)$$

HIGHER DIMENSIONS

We can derive similar results in higher dimension with the use of suitably defined transforms on ∂R.

GREEN'S FUNCTIONS

We proceed to use the propagation operator to obtain Green's functions.

Case I: $u_{xx} + u_{yy} = 0$.

$$u(x, y) = \left(T^{-1} e^{-|\omega| x} T \right) u(0, y)$$

$$= \frac{1}{2\pi} \left(\int_{-\infty}^{\infty} e^{-|\omega| x} e^{j\omega y} d\omega \right).$$

$$\int_{-\infty}^{\infty} u(0, \zeta) e^{-j\omega \zeta} d\zeta$$

$$= \frac{1}{2\pi} \left(\int_{-\infty}^{\infty} u(0, \zeta) d\zeta \right) \int_{-\infty}^{\infty} e^{-|\omega| x} e^{j\omega(y-\zeta)} d\omega$$

Now

$$\int_{-\infty}^{\infty} e^{-|\omega| x} + j\omega(y-\zeta) d\omega =$$

$$\int_0^{\infty} e^{j\omega(y-\zeta)} e^{-\omega x} d\omega$$

$$+ \int_{-\infty}^0 e^{+\omega x} e^{j\omega(y-\zeta)} d\omega$$

$$= \frac{1}{x+j(\zeta-y)} + \frac{1}{x-j(\zeta-y)} = \frac{2x}{x^2+(y-\zeta)^2}$$

so that

$$u(x, y) = \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{(x)u(0, \zeta)}{x^2+(y-\zeta)^2} d\zeta.$$

Case II: $u_{yy} = u_x$

$$u(x, y) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-\omega^2 x} e^{j\omega y} d\omega$$

$$\cdot \left(\int_{-\infty}^{\infty} u(0, \zeta) e^{-j\omega \zeta} d\zeta \right)$$

$$= \frac{1}{2\pi} \int_{-\infty}^{\infty} u(0, \zeta) d\zeta$$

$$\cdot \left(\int_{-\infty}^{\infty} e^{-\omega^2 x} e^{j\omega(y-\zeta)} d\omega \right).$$

Now

$$\begin{aligned} \int_{-\infty}^{\infty} e^{-\omega^2 x} e^{j\omega(y-\zeta)} d\omega &= \int_{-\infty}^{\infty} e^{-\omega^2 x} \cos \omega(y-\zeta) d\omega \\ &+ j \int_{-\infty}^{\infty} e^{-\omega^2 x} \sin \omega(y-\zeta) d\omega \\ &= 2 \int_0^{\infty} e^{-\omega^2 x} \cos \omega(y-\zeta) d\omega \\ &= \frac{1}{2\sqrt{\pi x}} e^{-\frac{(y-\zeta)^2}{4x}} \end{aligned}$$

so that

$$u(x, y) = \frac{1}{4\pi\sqrt{\pi}} \int_{-\infty}^{\infty} e^{-\frac{(y-\zeta)^2}{4x}} \frac{u(0, \zeta)}{\sqrt{x}} d\zeta.$$

Case III: $u_{yy} = u_{xx}$

$$u(x, y) = \left(T^{-1} \cos \omega x T \right) u(0, y)$$

$$+ \left(T^{-1} \frac{\sin \omega x}{\omega} T \right) u_x(0, y)$$

$$\text{Let } f(x, y) = \left(T^{-1} \cos \omega x T \right) u(0, y)$$

$$g(x, y) = \left(T^{-1} \frac{\sin \omega x}{\omega} T \right) u_x(0, y)$$

Then

$$f = \frac{1}{2\pi} \int_{-\infty}^{\infty} \cos \omega x e^{j\omega y} d\omega$$

$$\cdot \left(\int_{-\infty}^{\infty} u(0, \zeta) e^{-j\omega \zeta} d\zeta \right)$$

$$= \frac{1}{2\pi} \int_{-\infty}^{\infty} u(0, \zeta) d\zeta$$

$$\cdot \left(\int_{-\infty}^{\infty} \cos \omega x e^{j\omega(y-\zeta)} d\omega \right)$$

Now

$$\int_{-\infty}^{\infty} \cos \omega x e^{j\omega(y-\zeta)} d\omega = \int_{-\infty}^{\infty} \cos \omega x \cos \omega(y-\zeta) d\omega$$

$$\int_{-\infty}^{\infty} \cos \omega x e^{j\omega(y-\zeta)} d\omega = \Pi \left[\delta(x-y+\zeta) + \delta(-x-y+\zeta) \right]$$

and

$$\begin{aligned} f(x, y) &= \frac{1}{2} \int_{-\infty}^{\infty} u(0, \zeta) \left[\delta(x-y+\zeta) + \delta(-x-y+\zeta) \right] d\zeta \\ f(x, y) &= \frac{1}{2} [u(0, y-x) + u(0, y+x)] . \end{aligned}$$

Also

$$\begin{aligned} g(x, y) &= \frac{1}{2\pi} \int_{-\infty}^{\infty} \frac{\sin \omega x}{\omega} e^{j\omega y} d\omega \\ &\cdot \left(\int_{-\infty}^{\infty} u_x(0, \zeta) e^{-j\omega \zeta} d\zeta \right) \\ &= \frac{1}{2\pi} \int_{-\infty}^{\infty} u_x(0, \zeta) d\zeta \int_{-\infty}^{\infty} \frac{\sin \omega x}{\omega} [\cos \omega(y-\zeta) \\ &+ j \sin \omega(y-\zeta)] d\omega \end{aligned}$$

for $y-\zeta > 0$:

$$\begin{aligned} \int_{-\infty}^{\infty} \frac{\sin \omega x}{\omega} \cos \omega(y-\zeta) d\omega &= \pi, \quad y - \zeta < x \\ &= 2\pi, \quad y - \zeta = x \\ &= 0, \quad y - \zeta > x \end{aligned}$$

for $y-\zeta < 0$:

$$\begin{aligned} \int_{-\infty}^{\infty} \frac{\sin \omega x}{\omega} \cos \omega(y-\zeta) d\omega &= \pi, \quad \zeta - y < x \\ &= 2\pi, \quad \zeta - y = x \\ &= 0, \quad \zeta - y > x \end{aligned}$$

for the first case ($y-\zeta > 0$)

$$\int = \pi, \quad y > \zeta > y - x$$

for the second case ($y-\zeta < 0$)

$$\int = \pi, \quad y + x > \zeta > y$$

so in order to meet both conditions

$$\int = \pi, \quad y + x > \zeta > y - x$$

and

$$g(x, y) = \frac{1}{2} \int_{y-x}^{y+x} u_x(0, \zeta) d\zeta .$$

Finally,

$$\begin{aligned} u(x, y) &= \frac{1}{2} [u(0, y-x) + u(0, y+x)] \\ &+ \frac{1}{2} \int_{y-x}^{y+x} u_x(0, \zeta) d\zeta . \end{aligned}$$

Case IV:

Consider $u_{yy} = u_x$ in the region $0 \leq y \leq \pi, 0 \leq x$.

$$Tf(y) = \int_0^\pi f(\zeta) \sin n\zeta d\zeta = F(n)$$

$$T^{-1} F(n) = \frac{2}{\pi} \sum_{n=1}^{\infty} F(n) \sin ny$$

$$u(x, y) = T^{-1} \Lambda^x G(n) = \frac{2}{\pi} \sum_{n=1}^{\infty} \Lambda^x (\sin ny) G(n)$$

$$u_{yy} = \frac{2}{\pi} \sum_{n=1}^{\infty} \Lambda^x (-n^2) (\sin ny) G(n)$$

$$u_x = \frac{2}{\pi} \sum_{n=1}^{\infty} \Lambda^x \log \Lambda (\sin ny) G(n)$$

We have

$$\log \Lambda = -n^2$$

$$\Lambda = e^{-n^2}$$

$$u(x, y) = T^{-1} e^{-n^2 x} G(n)$$

$$u(0, y) = T^{-1} G(n)$$

$$G(n) = (T) u(0, y)$$

$$u(x, y) = (T^{-1} e^{-n^2 x} T) u(0, y) .$$

$$u(x, y) = \frac{2}{\pi} \sum_{n=1}^{\infty} e^{-n^2 x} \sin ny$$

$$\cdot \left(\int_0^\pi u(0, \zeta) (\sin n\zeta) d\zeta \right)$$

$$= \frac{2}{\pi} \int_0^\pi u(0, \zeta) d\zeta$$

$$\cdot \left(\sum_{n=1}^{\infty} e^{-n^2 x} (\sin ny) \sin n\zeta \right)$$

or to write out a few terms

$$u(x, y) = \frac{2}{\pi} \left\{ e^{-x} \sin y \int_0^\pi u(0, \zeta) (\sin \zeta) d\zeta + e^{-4x} \sin 2y \cdot \left(\int_0^\pi u(0, \zeta) (\sin 2\zeta) d\zeta \right) \right.$$

$$\left. + e^{-9x} \sin 3y \int_0^\pi u(0, \zeta) (\sin 3\zeta) d\zeta + \dots \right\}$$

All of these techniques can be implemented using repetitive operation. The idea is to evaluate at high speed the equations for fixed values for the coordinate parameters. Then by changing these parameters slowly the solutions can be found over the region, R, of definition for the equation.

EXAMPLES

$$u_{xx} + u_{yy} = 0:$$

$$u(x, y) = \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{x u(0, \zeta)}{x^2 + (y - \zeta)^2} d\zeta$$

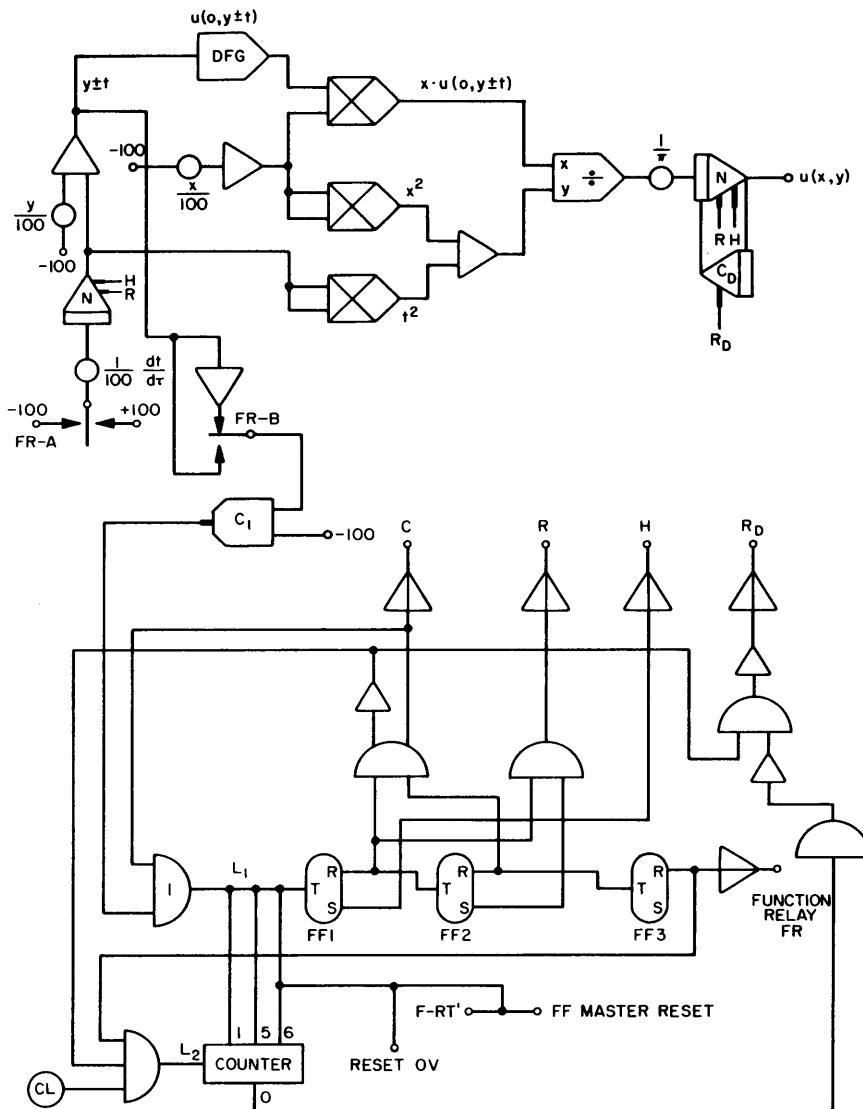
$$\text{Let } t = \zeta - y$$

$$u(x, y) = \frac{1}{\pi} \int_{-\infty}^{\infty} \frac{x u(0, y+t)}{x^2 + t^2} dt$$

$$u(x, y) = \frac{1}{\pi} \int_0^{\infty} \frac{x u(0, y+t)}{x^2 + t^2} dt$$

$$+ \int_0^{\infty} \frac{x u(0, y-t)}{x^2 + t^2} dt$$

The program is:

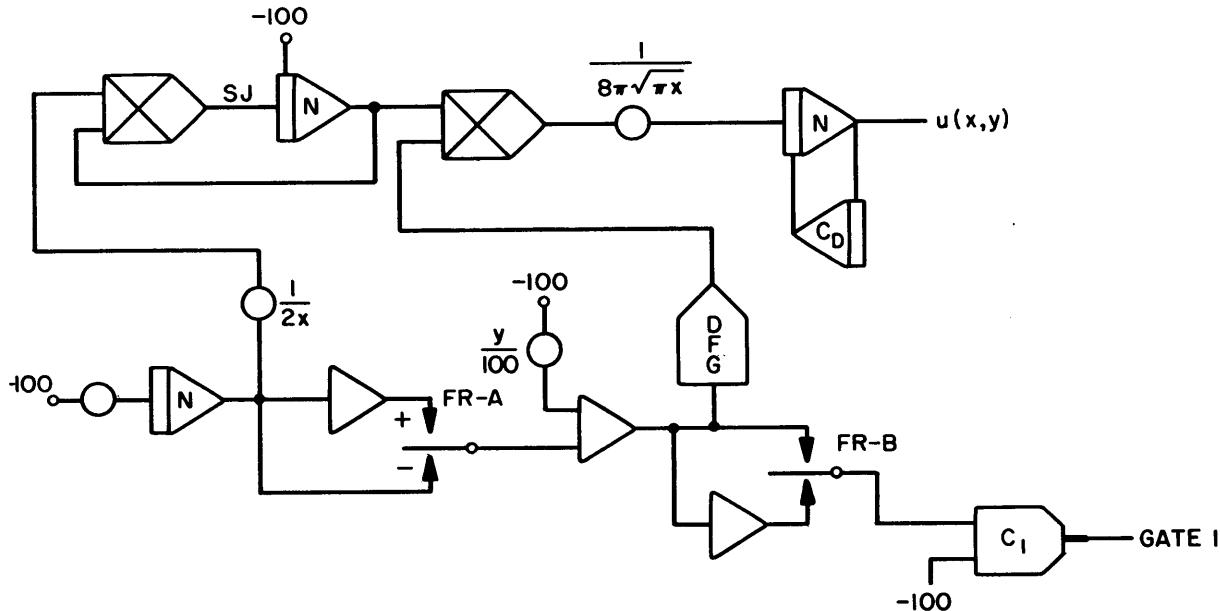


The FF's act as an up-counter with successive trigger inputs to FF1. When the machine is in reset the counter output is zero and the integrators are in the standby mode. (The RO selector switch is in external). When the RO pushbutton is activated, the N-integrators go to compute. As the counter is incremented, the mode sequence is

COUNTER	MODE
3 2 1	FF
0 0 0	C
0 0 1	H
0 1 0	IC
0 1 1	H
1 0 0	C
1 0 1	H

Note that $(\overline{FF3}) = 0$ when the counter has reached 101. When this occurs there are no more trigger inputs to FF1 and the program stops. Before this, the trigger for FF1 is provided by the clock (L_2) when the machine is not in compute and by the comparator (C_1) when the machine is in compute. There are only two compute intervals. The function relay (FR) is energized (+) during the first compute interval and de-energized (-) during the second compute interval.

The analog program is



Both the logic program and the principle of operation of the complete program are the same as for the previous example.

The first and second integrals are calculated during the first and second compute intervals, respectively. The comparator is energized when $y \pm t$ reaches its maximum absolute value for each calculation. It then terminates the compute interval. The final output from the accumulator is the function $u(x, y)$ for the particular x, y chosen for the run.

$$u_{yy} = u_x:$$

$$u(x, y) = \frac{1}{2\pi} \int_{-\infty}^{\infty} e^{-\frac{(y-\zeta)^2}{4x}} \frac{u(0, \zeta)}{2\sqrt{\pi x}} d\zeta$$

$$\text{Let } t = \zeta - y.$$

$$u(x, y) = \frac{1}{4\pi\sqrt{\pi x}} \int_{-\infty}^{\infty} e^{\frac{-t^2}{4x}} u(0, \zeta) d\zeta$$

$$u(x, y) = \frac{1}{4\pi\sqrt{\pi x}} \left(\int_0^{\infty} e^{\frac{-t^2}{4x}} u(0, y+t) dt + \int_0^{\infty} e^{\frac{-t^2}{4x}} u(0, y-t) dt \right)$$

CHAPTER 18

CORRELATION ANALYSIS

PROBABILITY THEORY

Sample space:

A sample space, S , is the collection of all outcomes of an experiment.

Probability:

$P(S)$ is a function defined over S . P maps S into the interval $0 \leq P(S) \leq 1$. P is not necessarily univalent. The integral (or sum) over the range of P is unity. In case S is finite, P is said to assign a probability to each point of S . In case S is infinite, P is said to define a probability density over S . Suppose S_α is a nested collection of subsets of S such that $S_i \subset S_{i+1}$ for all i .

The extremum of $S_\alpha = S$ and the infimum of S_α is empty. The measure of the range of P for the monotone sequence S_α is the probability distribution for S with respect to the collection S_α .

EXPECTATION AND VARIANCE

The expectation of S , $E(S)$, is the measure of $SP(S)$. The variance, $V(S)$, is the measure of

$$S^2 P(S) - E^2(S)$$

Example:

Suppose S is given by $-\infty < x < \infty$ and

$$P(x) = (2\pi)^{-\frac{1}{2}} \exp \left[-\frac{(x - \mu)^2}{2\sigma^2} \right].$$

Then

$$E(x) = \int_{-\infty}^{\infty} x P(x) dx = \mu$$

$$V(x) = \int_{-\infty}^{\infty} x^2 P(x) dx - \mu^2 = \sigma^2$$

Let the S_α be defined by $-\infty < x < x_\alpha$, then

$$\phi(x_\alpha) = \int_{-\infty}^{x_\alpha} P(x) dx$$

is the distribution function.

STATIONARY RANDOM PROCESSES

If the statistics of a process are time-independent, the process is said to be stationary.

Suppose $f(t)$ is a random function defined for $-\infty < t < \infty$.

Let the domain of f be denoted by D . We define an ordered countable set $\{D_i\}$ so that

$$D_i : x_i \leq t < x_{i+1}, x_{i+1} - x_i > \lambda.$$

$$\cup D_i = D$$

If $f_i = f(D_i)$, then the collection f_α is an ensemble. If f is stationary and λ is sufficiently large, the ensemble will be ergodic. (That is, the statistical properties of the members of the ensemble are identical.) Henceforth, all ensembles considered will be assumed ergodic.

FIRST AND SECOND PROBABILITY DENSITIES

Let S be the sample space consisting of all possible values for the ensemble $\{f_i(t_i + \Delta)\}$ where $\Delta < \lambda$.

$P(S)$ is said to be the first probability density. $P(S)$ is independent of Δ , since f is stationary. The ensemble is ergodic so that

$$E(S) = \lim_{\delta \rightarrow \infty} \frac{1}{2\delta} \int_{-\delta}^{\delta} f(t) dt$$

$$V(S) = \lim_{\delta \rightarrow \infty} \frac{1}{2\delta} \int_{-\delta}^{\delta} f^2(t) dt - E(S)^2$$

Let S_T be the sample space consisting of all possible pairs of values $(f_i(t_i + \Delta), f_i(t_i + \Delta + T))$ for the ensemble where $\Delta + T < \lambda$. $P_T(S_T)$ is said to be the second probability density. $P_T(S_T)$ depends only on T , not on Δ . The ensemble is ergodic so that

$$E_T(S_T) = \lim_{\delta \rightarrow \infty} \frac{1}{2\delta} \int_{-\delta}^{\delta} f(t)f(t + T) dt.$$

(It can be shown that this equation is valid for $T > \lambda$.)

AUTOCORRELATION FUNCTION

$E_T(S_T)$, above, is the autocorrelation function for f and is denoted by $\phi_{ff}(T)$. That is,

$$\phi_{ff}(T) = \lim_{\delta \rightarrow \infty} \frac{1}{2\delta} \int_{-\delta}^{\delta} f(t)f(t+T)dt$$

The autocorrelation function has the properties:

- 1) $\phi_{ff}(T) = \phi_{ff}(-T)$
- 2) $\phi_{ff}(0)$ is the average power of f .
- 3) $|\phi_{ff}(T)| \leq \phi_{ff}(0)$

Proof:

$$|\phi_{ff}(T)| \leq \lim_{\delta \rightarrow \infty} \frac{1}{2\delta} \int_{-\delta}^{\delta} |f(t)| \cdot |f(t+T)| dt$$

$$\begin{aligned} &\leq \lim_{\delta \rightarrow \infty} \frac{1}{2\delta} \int_{-\delta}^{\delta} \frac{1}{2} \left\{ |f(t)|^2 + |f(t+T)|^2 - [|f(t)| - |f(t+T)|]^2 \right\} dt \\ &\leq \phi_{ff}(0) - \lim_{\delta \rightarrow \infty} \frac{1}{4\delta} \int_{-\delta}^{\delta} [|f(t)| - |f(t+T)|]^2 dt \end{aligned}$$

- 4) $f \in P_\nu \implies \phi_{ff} \in P_\nu$ (is periodic with period ν)

Proof: $f(t+T) = f(t+\nu)$

- 5) ϕ_{ff} contains no phase information about f .
- 6) For every f there exists a unique ϕ_{ff} ; the converse is not true.
- 7) If $f \in P_\nu$ for any ν , then $\lim_{T \rightarrow \infty} \phi_{ff}(T) = 0$
- 8) If f can be represented by

$$f = \sum_{n=-\infty}^{\infty} A_n \sin(\omega_n t + \phi_n),$$

then

$$\phi_{ff}(T) = \sum_{n=-\infty}^{\infty} \phi_{nn}(T)$$

$$\text{where } \phi_{nn} = \lim_{\delta \rightarrow \infty} \frac{1}{2\delta} \int_{-\delta}^{\delta} A_n^2 \sin(\omega_n t + \phi_n) \cdot (\sin(\omega_n t + T + \phi_n)) dt$$

Proof:

$$\lim_{\delta \rightarrow \infty} \frac{1}{2\delta} \left(\int_{-\delta}^{\delta} A_n \sin(\omega_n t + \phi_n) \right).$$

$$A_m \sin(\omega_m t + T + \phi_m) dt = 0, m \neq n.$$

- 9) If $f, f' \in bd$, then $\phi_{f'f'}(T) = -\phi_{ff}''(T)$.

Proof:

$$\begin{aligned} \phi_{f'f'}(T) &= \lim_{\delta \rightarrow \infty} \frac{1}{2\delta} \int_{-\delta}^{\delta} f'(t)f'(t+T) dt \\ &= \lim_{\delta \rightarrow \infty} \frac{1}{2\delta} \left\{ \left[f(t)f'(t+T) \right]_{-\delta}^{\delta} - \int_{-\delta}^{\delta} f_1(t)f''(t+T) dt \right\} \\ &= -\frac{d^2}{dT^2} \lim_{\delta \rightarrow \infty} \frac{1}{2\delta} \int_{-\delta}^{\delta} f_1(t)f_2(t+T) dt \end{aligned}$$

CROSS-CORRELATION FUNCTION

Suppose f, g are two stationary random functions. f, g can be decomposed (as above) into two ergodic ensembles f_α, g_α . The union of these ensembles is a composite ensemble of pairs $(f, g)_\alpha$. Let S_T be the sample space consisting of all possible pairs of values $(f_i(t_i + \Delta), g_i(t_i + \Delta + T))$ for the composite ensemble, where $\Delta + T < \lambda$ (λ is the ensemble duration). Since the ensemble is ergodic

$$E_T(S_T) = \lim_{\delta \rightarrow \infty} \frac{1}{2\delta} \int_{-\delta}^{\delta} f(t)g(t+T) dt$$

E_T is the cross-correlation function for the pair (f, g) and is denoted by $\phi_{fg}(T)$.

It can be shown that $\phi_{fg}(T)$ is valid for $T > \lambda$. The cross-correlation function has the properties:

1) Generally $\phi_{fg}(T) \neq \phi_{fg}(-T)$.

$$\text{Consider } g(t) = \frac{1}{\epsilon}, -\epsilon \leq t \leq \epsilon$$

$$= 0, |t| > \epsilon$$

2) $\phi_{fg}(T) = \phi_{gf}(-T)$

3) Max $\phi_{fg}(T)$ does not necessarily occur for $T=0$.

$$4) |\phi_{fg}(T)| \leq \sqrt{\phi_{ff}(0) \phi_{gg}(0)}$$

Proof:

$$\phi_{fg}^2(T) \leq \lim_{\delta \rightarrow \infty} \frac{1}{2\delta} \int_{-\delta}^{\delta} f^2(t)g^2(t+T)dt \leq$$

$$\left[\lim_{\delta \rightarrow \infty} \frac{1}{2\delta} \int_{-\delta}^{\delta} f^2(t)dt \right].$$

$$\left[\lim_{\delta \rightarrow \infty} \frac{1}{2\delta} \int_{-\infty}^{\infty} g^2(t+T)dt \right]$$

AUTOCORRELATION FOR A SUM OF FUNCTIONS

Henceforth we will denote

$$\lim_{\delta \rightarrow \infty} \frac{1}{2\delta} \int_{-\delta}^{\delta} f(t)dt$$

by Mean $\{f(t)\}$, or just $M\{f\}$ wherever convenient. Let $f(t) = f_1(t) + f_2(t)$.

$$\begin{aligned} \phi_{ff}(T) &= \text{Mean} \left\{ [f_1(t) + f_2(t)] [f_1(t+T) + f_2(t+T)] \right\} \\ &= M\{f_1(t)f_1(t+T)\} + M\{f_1(t)f_2(t+T)\} \\ &\quad + M\{f_2(t)f_1(t+T)\} + M\{f_2(t)f_2(t+T)\} \end{aligned}$$

Thus,

$$\phi_{ff} = \phi_{f_1 f_1} + \phi_{f_1 f_2} + \phi_{f_2 f_1} + \phi_{f_2 f_2}.$$

The general rule for a linear combination can be deduced easily.

EXAMPLES OF AUTOCORRELATION FUNCTIONS

Example 1 (sinewave):

Let $f = A \sin(\omega t + \psi)$, then

$$\phi_{ff} = \text{Mean} \left\{ A^2 \sin(\omega t + \psi) \sin(\omega t + \omega T + \psi) \right\}$$

Since $f \in P_\omega$,

$$\phi_{ff} = \frac{\omega A^2}{2\pi} \int_{-\alpha}^{\alpha} \sin(\omega t + \psi) \sin(\omega t + \omega T + \psi) dt$$

$$\phi_{ff} = \frac{A^2}{2\pi} \int_{\alpha\omega + \psi}^{\alpha\omega + \psi + 2\pi} \sin(\zeta) \sin(\zeta + \omega T) d\zeta$$

$$\phi_{ff} = \frac{A^2}{4\pi} \int_0^{2\pi} \cos \omega T dt = \frac{A^2}{2} \cos \omega T$$

Example 2 (general random function):

Suppose f is defined by

- a) $f(n\delta + t) = f(n\delta), 0 \leq t < \delta$
- b) $P[f(n\delta) = x] = P(x)$
- c) $E[f(n\delta)] = 0$

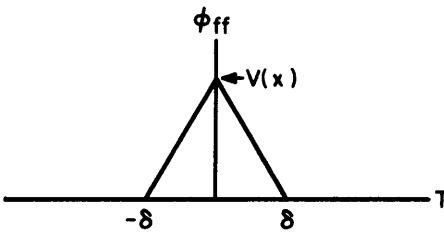
then

$$\begin{aligned} E[f(\zeta)f(\zeta + T)] &= \int_{-\infty}^{\infty} [xP(x)] \left[x(1 - \left| \frac{T}{\delta} \right|) \right] dx + \\ &\quad \iint_{-\infty}^{\infty} [xP(x)] [yP(y)] \left| \frac{T}{\delta} \right| dx dy, |T| < \delta \\ &= \iint_{-\infty}^{\infty} [xP(x)] [yP(y)] dx dy, |T| > \delta. \end{aligned}$$

But $\iint_{-\infty}^{\infty} xyP(x)P(y) = 0$, so that

$$\phi_{ff} = \left(1 - \left|\frac{T}{\delta}\right|\right) \int_{-\infty}^{\infty} x^2 P(x) dx = V(x) \left(1 - \left|\frac{T}{\delta}\right|\right), T \leq \delta$$

$$= 0, T > \delta$$



Example 3 (binary noise):

Binary noise is a special case of (2), above, where

$$P(x) = \frac{1}{2} u_1 (x - x_0) + \frac{1}{2} u_1 (x + x_0).$$

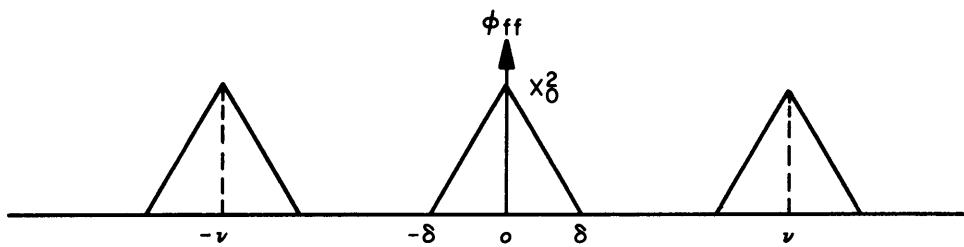
That is f takes on values of $x_0, -x_0$ with probabilities of $\frac{1}{2}$.

Then $V(x) = x_0^2$ and

$$\phi_{ff} = x_0^2 \left(1 - \left|\frac{T}{\delta}\right|\right)$$

Example 4 (periodic binary noise):

Suppose f is defined as in (3), above, except that $f(\zeta) = f(\zeta + \nu)$ for large ν . Then ϕ_{ff} is periodic.



POWER SPECTRAL DENSITY

The operator F denotes the Fourier transform, so that $Fg = G(\omega)$.

$$\Phi_{gg}(\omega) = \lim_{\delta \rightarrow \infty} \frac{1}{2\lambda} \left[\int_{-\lambda}^{\lambda} g(\zeta) e^{-j\omega\zeta} d\zeta \right]^2$$

$$\Phi_{gg}(\omega) = F\phi_{gg} = \lim_{\delta \rightarrow \infty} \int_{-\delta}^{\delta} e^{-j\omega T} dT$$

$$\cdot \left[\lim_{\delta \rightarrow \infty} \frac{1}{2\lambda} \int_{-\lambda}^{\lambda} g(t)g(t+T) dt \right]$$

$$\Phi_{gg}(\omega) = \lim_{\delta \rightarrow \infty} \frac{1}{2\lambda} \left\{ \int_{-\lambda}^{\lambda} g(t) dt \right. \\ \left. \cdot \left[\lim_{\delta \rightarrow \infty} \int_{-\delta}^{\delta} g(t+T) e^{j\omega T} dT \right] \right\}$$

$$\Phi_{gg}(\omega) = \lim_{\delta \rightarrow \infty} \frac{1}{2\lambda} \left\{ \int_{-\lambda}^{\lambda} g(t) e^{j\omega t} dt \right.$$

$$\left. \cdot \left[\lim_{\delta \rightarrow \infty} \int_{-\delta}^{\delta} g(\zeta) e^{-j\omega\zeta} d\zeta \right] \right\}$$

$\Phi_{gg}(\omega)$ is said to be the power spectral density. We have

$$\phi_{gg}(T) = \int_{-\infty}^{\infty} \Phi_{gg}(\omega) e^{j\omega T} dT.$$

WHITE NOISE

White noise has constant power spectral density of arbitrary magnitude (say A). Thus, the autocorrelation function for white noise is

$$\phi_{gg} = Au_1(T),$$

where u_1 is the unit impulse.

PERIODIC WHITE NOISE

If f satisfies the requirements for white noise for some interval δ , where δ is large, and if $f(t+\delta) = f(t)$, then

$$\phi_{ff}(T) = A \sum_{n=-\infty}^{\infty} u_1(T-n\delta).$$

FINITE CORRELATION FUNCTION

If f, g are defined only for $0 \leq t < \infty$, then

$$\phi_{fg} = \lim_{\delta \rightarrow \infty} \frac{1}{\delta} \int_0^\delta f(t) g(t+T) dt.$$

ϕ_{fg} can be approximated by a sum:

$$\phi_{fg} \approx \text{Mean} \sum_{m=1}^N f(m \Delta t) g(m \Delta t + T) \quad (1)$$

In this case Δt must be small enough that the high-frequency components of f, g are adequately represented (see Chapter 16). N must be large enough that the mean of the sum is a good approximation to ϕ_{fg} .

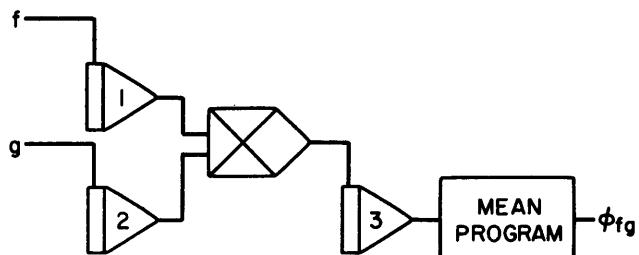
CORRELATION PROGRAMS

Simple computer programs for (1) require that

$$\Delta t > T. \quad (2)$$

To obtain the best approximation, $\Delta t = T + \epsilon$, where ϵ is the minimum time required for sampling in the computer program. (2) imposes a limitation on the magnitude of T . That is if T is large, then so is Δt . Thus, for large T , relatively few samples are made of f, g . This restriction is not unduly severe, because ϕ_{fg} cannot be determined accurately for large T together with finite histories for f, g .

If (1) is to be implemented directly, the program is



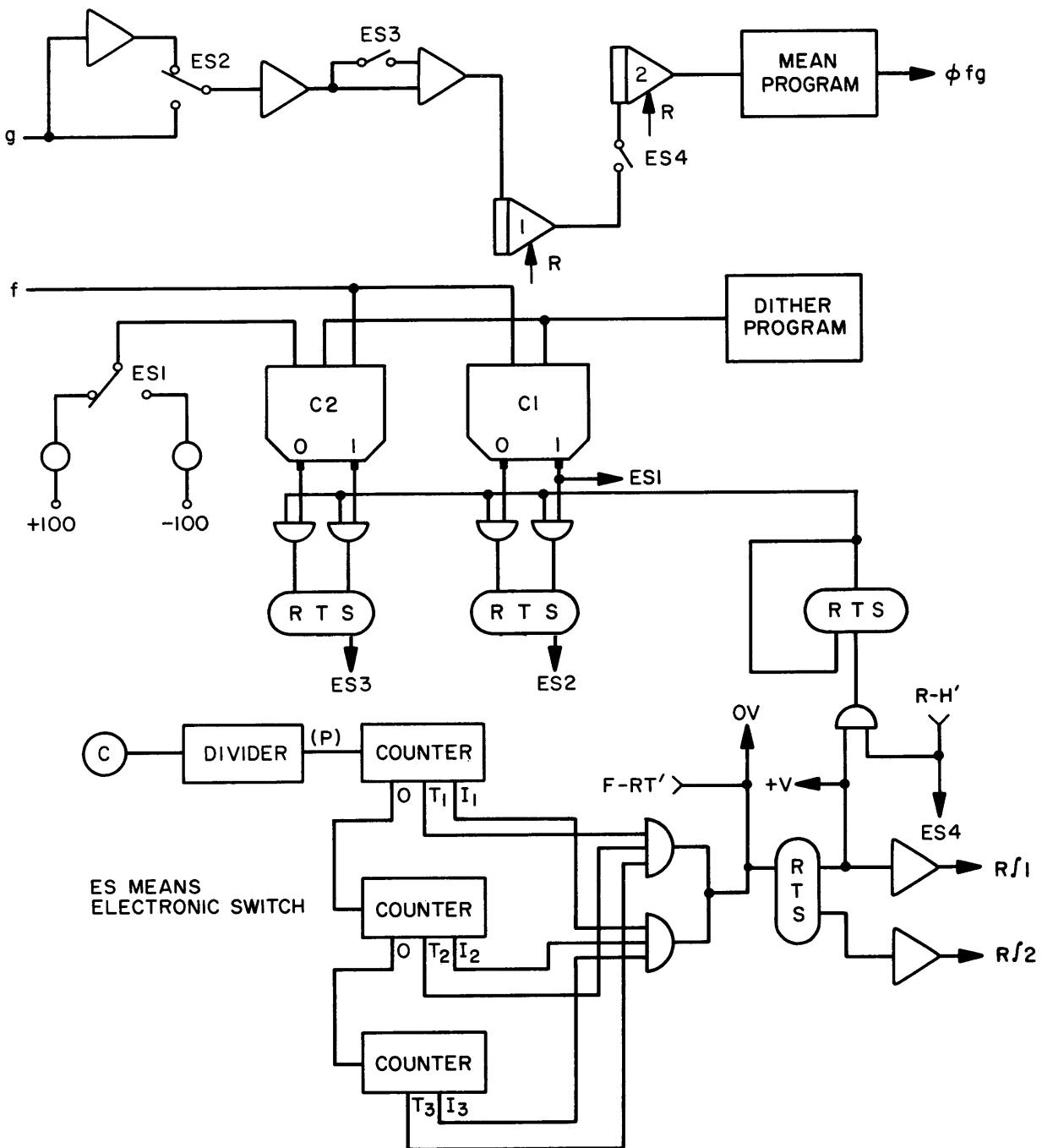
Integrator 1 samples f and then after an interval, T , integrator 2 samples g . Integrator 3 is in hold until both samples are made and then it samples the product $f(t_i) g(t_i + T)$. The output of integrator 3 is averaged by a Mean Program (Chapter 3) which in turn generates ϕ_{fg} . If T is to be specified accurately, the integrator capacitors must be as small as possible to minimize the effect of the IC time constant.

The total sampling time can be reduced by replacing integrator 1 with a quantizer whose input is f together with dither (see reference). f will be represented by a two-bit word (output of the quantizer).

The table below defines the quantizing function

<u>OUTPUT</u>	<u>BINARY OUTPUT</u>	<u>DECIMAL OUTPUT</u>
$-2 \leq x < -1$	00	-2
$-1 \leq x < 0$	01	-1
$0 \leq x < 1$	10	+1
$1 \leq x \leq 2$	11	+2

Except for scaling the program is



The Dither Program is found in Chapter 10 and the Mean Program in Chapter 3.

The correlation parameter, T , and the reset interval for integrator 1 are given by $(10T_3 + 10T_2 + T_1) P$ and $(10I_3 + 10I_2 + I_1)P$, respectively, where P is the period of the output of the divider.

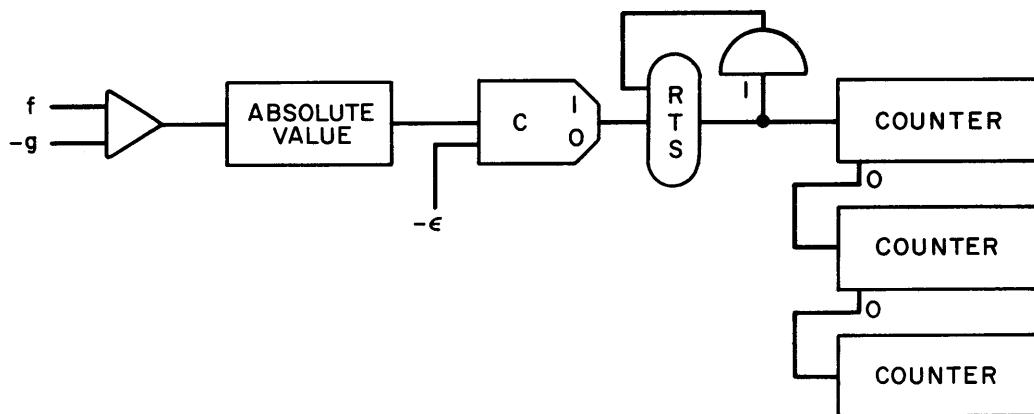
Reference: Korn, G. A., "Hybrid-Computer Techniques for Measuring Statistics from Quantized Data", Simulation, April 1965.

CHAPTER 19

ON-LINE DATA ANALYSIS PROGRAMS

An analog computer can be used to advantage for on-line data analysis. Programs are given in this section which correspond to the more frequent operations on incoming data. The analog computer is used for both processing (analysis) and editing. The principles developed in other chapters can be used for processing. The programs here are for simple editing.

The program which measures error frequency is



ERROR FREQUENCY

Suppose an error is defined to occur whenever

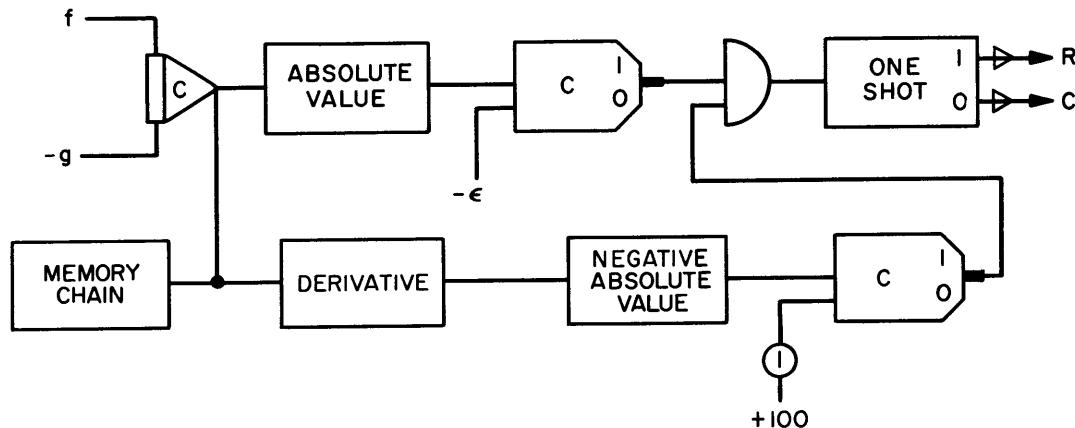
$$|f(t) - g(t)| > \epsilon .$$

ERROR PEAK VALUE

Suppose, with the above definition of error, it is required to find a sequence of peak values of error,

$$\begin{aligned} \text{Max } & |f - g| \\ \text{Max } & |f - g| . \end{aligned}$$

The program is



All the programs indicated will be found in various chapters of this manual. The integrators in this program are under iterative control from the C, R logical control inputs. Pot 1 is set to a small value to provide positive zero derivative detection.

TIME BETWEEN EVENTS

Suppose two events, E_1 , E_2 are defined by

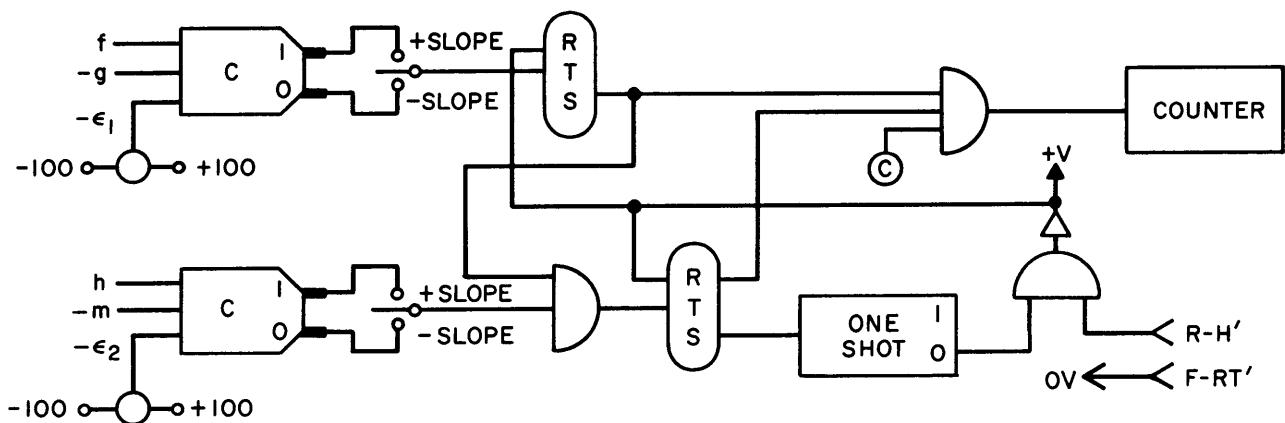
$$E_1: f(t) - g(t) = \epsilon_1$$

$$\dot{f}(t) - \dot{g}(t) \geq 0$$

$$E_2: h(t) - m(t) = \epsilon_2$$

$$\dot{h}(t) - \dot{m}(t) \geq 0$$

A program which will determine a sequence of time lapses between E_1 , E_2 (not E_2 , E_1) is



CHAPTER 20

SYSTEM OPTIMIZATION

System optimization is a procedure for generating a response, y_s , of a known experimental system so that it matches, as closely as possible, some desired response y_d . y_d may come from tape or may be the response of a known but more complex system. In what follows we will assume that either an external device generates y_d or else we know the computer program for it. A measure of the accuracy of fit of y_s needs to be established. This is called the Criterion Function (CF). The CF has different forms depending on whether dynamic or static optimization is required. The exact form of the CF in either case is a matter of technical judgment and will depend on the particular situation. The CF for the dynamic problem will need to take account of the transient behavior of both y_s and y_d over some interval of time. Typical choices for the CF are

STATIC

$$F = \frac{1}{2} (y_s - y_d)^2$$

DYNAMIC

$$f = \int_0^T [y_s(t) - y_d(t)]^2 dt.$$

(Note that the CF is defined so that it is non-negative)

The optimization problem amounts to finding maxima or minima of the CF. Since y_s is a function of its parameters, say C_1, \dots, C_n , so is the CF. Thus the CF can be viewed as a surface over the parameter plane. In general the CF will have several maxima and minima. We will restrict our attention to finding the nearest minimum from some initial point in the parameter plane. Once techniques are developed for the minimum problem, they can be applied with modifications, which will be obvious, to the maximum problem.

We will develop two methods for finding a minimum: steepest descent and relaxation. The static and dynamic optimization problems will be treated separately.

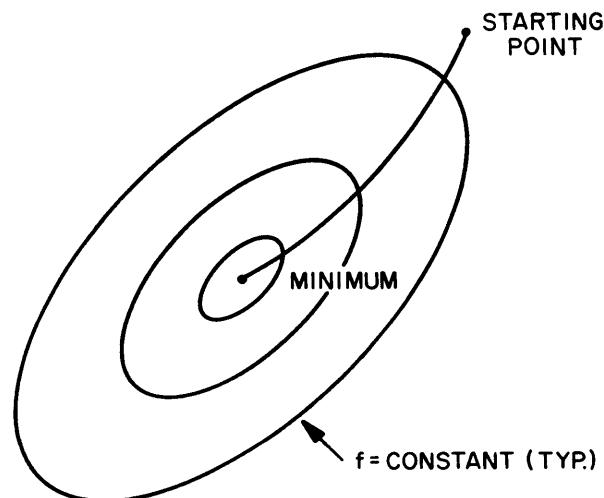
STATIC PROBLEM

The family of curves $f = \text{constant}$ on the surface for f , the CF, are constant elevation contour lines. Differentiating

$$df = \sum_{j=1}^n \frac{\partial f}{\partial C_j} dC_j = \nabla f \cdot d\bar{C} = 0$$

where \bar{C} is the parameter vector -- i.e., $\bar{C} = [C_1, C_2, \dots, C_n]$ (Remember that the CF is a function of the parameters of y_s .)

We see that ∇f is perpendicular to $d\bar{C}$ and $d\bar{C}$ is a differential vector which is tangential to the space curve $f = \text{constant}$. The projection of the curve on the parameter plane is identical geometrically to the space curve because it is at constant elevation over the plane. The projection of ∇f on the parameter plane is perpendicular to the curve. It is evident that ∇f corresponds to the slope of the surface f and $\nabla f = 0$ at the minimum. To get to the minimum from some point on the surface, then, all we need to do is follow $-\nabla f$. (If we wanted to find the maximum we would follow $+\nabla f$.) The projection of the path on the parameter plane is



If the system is static (algebraic) we can change the parameters of y_s while following $-\nabla f$ without introducing transients in the system. Now if

$$f = \frac{1}{2} (y_s - y_d)^2$$

then

$$\nabla f = (y_s - y_d) \nabla y_s.$$

If we let $e = y_s - y_d$, then \bar{C} can be determined by (since we follow $-\nabla f$)

$$\bar{C} = -e \int \nabla y_s dt.$$

or in scalar form

$$C_1 = -e \int \frac{\partial y_s}{\partial C_1} dt.$$

$$\vdots$$

$$C_n = -e \int \frac{\partial y_s}{\partial C_n} dt.$$

In practice, it is usually advisable to use gain factors, K_k , so that

$$C_1 = -K_1 e^{\int \frac{\partial y_s}{\partial C_1} dt}$$

$$\vdots$$

$$\vdots$$

$$\vdots$$

$$C_n = -K_n e^{\int \frac{\partial y_s}{\partial C_n} dt}.$$

K is chosen so that the procedure converges but at the same time is stable. When $\nabla f = 0$, the integrals will be stationary and \bar{C} will be the parameter vector (point on the parameter plane) corresponding to the minimum of f . ∇y_s is generated by differentiating the generating equation for y_s with respect to the parameters C_i .

DYNAMIC PROBLEM

Except under special circumstances, ∇f cannot be generated on an instantaneous basis. Even if it could, the parameters could not be changed during the determination of f . Since the calculation of f is carried out over a finite time interval, varying the parameters would lead to a result which is not the CF.

Some definitions for f will permit ∇f to be computed as often as is f . For example, if

$$f = \frac{1}{2} \int_0^T (y_s - y_d)^2 dt$$

then

$$\nabla f = \int_0^T (y_s - y_d) \nabla y_s dt.$$

As soon as we know the components of y_s , we can use a steepest-descent method. After each solution a new position in the parameter plane is chosen by taking a finite step along $-\nabla f$. Now ∇y_s can be generated by the parameter-influence equation method. The differential equation for each component of ∇y_s is obtained by differentiation of the generating equation for y_s . For example, if

$$C_1 \ddot{y}_s + C_2 y_s = 0, \quad \dot{y}_s(0) = A, \quad y_s(0) = A,$$

the equations for $\frac{\partial y_s}{\partial C_1}, \frac{\partial y_s}{\partial C_2}$ can be obtained by differentiating first with respect to C_1 and then C_2 .

$$\frac{\partial}{\partial C_1} : C_1 \frac{d^2}{dt^2} \left(\frac{\partial y_s}{\partial C_1} \right) + C_2 \frac{\partial y_s}{\partial C_1} = -\ddot{y}_s,$$

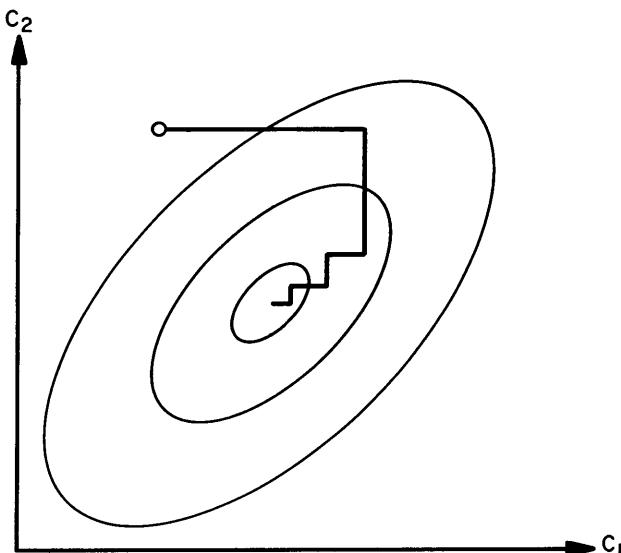
$$\frac{d}{dt} \left(\frac{\partial y_s}{\partial C_1} \right) = 0, \quad \frac{\partial y_s}{\partial C_1} = 0$$

$$\frac{\partial}{\partial C_2} : C_1 \frac{d^2}{dt^2} \left(\frac{\partial y_s}{\partial C_2} \right) + C_2 \frac{\partial y_s}{\partial C_2} = -y_s,$$

$$\frac{d}{dt} \left(\frac{\partial y_s}{\partial C_2} \right) = 0, \quad \frac{\partial y_s}{\partial C_2} = 0.$$

These parameter-influence equations can be solved with an analog program. Their non-homogeneous terms are obtained from outputs of the generating program for y_s . It will be noticed that they have identical form except for the non-homogeneous terms. Due to equipment limitations, it is usually necessary to solve them one at a time, switching in the correct non-homogeneous inputs from the y_s program in a suitable way. This leads to the use of the relaxation method.

With the relaxation method we change only one parameter, C_i , until the component, $(\nabla f)_i$, corresponding to the parameter is zero or nearly so. The procedure is applied to all the parameters on a cyclic basis. An example of the resulting path in a two dimensional parameter plane is



This method obviously is ideal for use with the parameter-influence equations. Nothing needs to be changed in the influence equation program except the non-homogeneous term.

In the event it is not convenient to use the parameter-influence method, successive values of f can be calculated for a sequence of values of C_i . This is continued until two values of f bracket $(\nabla f)_i = 0$. Then the procedure is applied varying C_{i+1} and so on. This will also lead to the minimum for f . In either case the subscript indexing and switching of non-homogeneous terms is done with a digital logic program. In the latter case, successive values of f are retained by a two or three word memory chain and comparisons made by comparators.

With either method the step size must be determined by a gain factor which ensures proper convergence.

CHAPTER 21

MEDICAL APPLICATIONS

Originally published by: Beckman Instruments, Inc.
Author: Hiroshi Hara

ANALOG SIMULATION OF VENTRICULAR PUMP ACTION

Due to the complexity and non-linearity involved, the building of an analog model of blood circulation is one of the challenging problems in biomedical science. This example is by no means intended to cover the subject, but illustrates how iterative programming is conveniently used in simulating ventricular pump action.

The ventricle has an inlet valve (tricuspid valve) and an outlet valve (pulmonary valve). A cardiac cycle consists of two periods called the diastole and the systole, as shown in Figure 1. During the diastole, the heart muscle relaxes and blood flows into the ventricle as the inlet valve opens. At the completion of the diastole, the systole begins. During this period the heart muscle contracts and as the ventricular pressure exceeds the arterial pressure, the outlet valve opens, expelling blood from the ventricle. An electric analog of the ventricle is shown in Figure 2. The analogous quantities are indicated below.

Blood Pressure	~	Voltage
Blood Flow	~	Current
Blood Volume	~	Charge

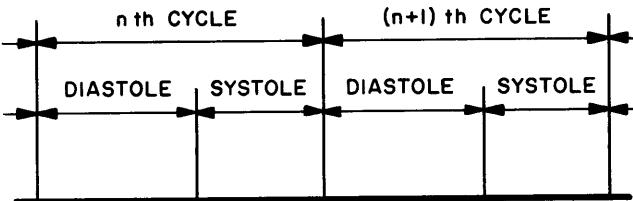


FIGURE 1

The contraction and relaxation of the muscular wall of the ventricle is simulated by decrease and increase, respectively, of capacitance c_1 in some prescribed manner. If we let $E = \frac{1}{c_1}$ = elastance of the ventricle, the pump action is assumed to be described by the following equations:

$$[E_d]_n = A - (A - E_{n-1}^s) e^{-\frac{t}{\tau}} + B [q_d]_n \quad (1)$$

$$[E_s]_n = E_n^d + \psi_n t, \quad \psi_n = f(q_n^d) \quad (2)$$

where

$$[E_d]_n = \text{Elastance during } n \text{ th diastole}$$

$$[E_s]_n = \text{Elastance during } n \text{ th systole}$$

$$E_n^d = \text{Elastance at the end of } n \text{ th diastole}$$

$$E_{n-1}^s = \text{Elastance at the end of } (n-1) \text{ th systole}$$

$$[q_d]_n = \text{Blood volume during } n \text{ th diastole}$$

ψ_n = Slope of $[E_s]_n$ to be determined as a function of q_n^d , the blood volume at the end of n th diastole.

A, B, and τ are positive constants.

Since the static work done by the heart at a particular value of q_n^d is known experimentally (Starling's Curve), ψ_n must be found as a function of q_n^d such that experimental results are duplicated. The work done during the systole is

$$W_s = k \int_0^{T_{\text{systole}}} v_2 i_{12} dt,$$

where v_2 = arterial pressure

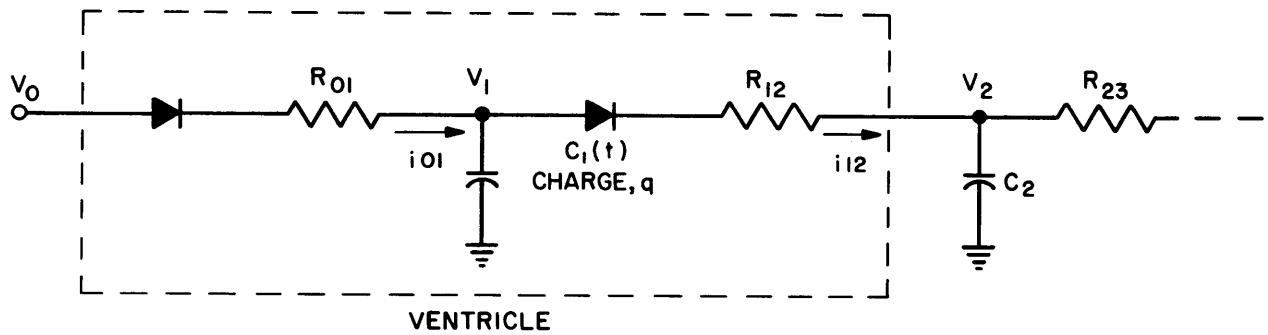
i_{12} = blood flow out of ventricle

k = constant

and consequently for a certain q_n^d , ψ_n must be such that $[W_s]_n = [W_s]$ Starling. In other words, iteration with respect to ψ_n is required.

The passive circuit shown in Figure 2, equations (1) and (2), and an iteration circuit are implemented as shown in Figure 3. Note that two iterative cycles are required to simulate one cardiac cycle. The distinction between the diastole and systole is made by a function relay (Relay KO) which in turn is controlled by one of the memory integrators in a chain of four. The action of this circuitry is shown by the timing diagram in Figure 4. Since the compute intervals for the diastole and systole differ, the comparator which controls the compute interval of the iterative control program must, necessarily, require two different reference voltages which are switched alternately by relay KO. The function of the M2 - M1 memory chain is to make

ELECTRIC ANALOG OF A VENTRICLE



V_0 : VENOUS PRESSURE

V_1 : VENTRICULAR PRESSURE

V_2 : ARTERIAL PRESSURE

R_{01} : INLET VALVE RESISTANCE

R_{12} : OUTLET VALVE RESISTANCE

$C_1 : \text{VENTRICLE CAPACITANCE} = \frac{1}{E}, \text{ WHERE } E = \text{ELASTANCE}$

q : BLOOD VOLUME

$W_s : \text{WORK DONE BY THE VENTRICLE DURING SYSTOLE} =$

$$k \int v_2 i_{12} dt, \text{ WHERE } k = \text{CONSTANT}$$

FIGURE 2

the final value of q (blood volume) at the end of the n th diastole available throughout the n th systole. (\bar{M} denotes complementary memory.) Note that relay K_0 prevents this memory chain from learning the end systolic blood volume.

In order to determine the required functional relationships between q_n^d and ψ_n , a simple iteration circuit is

provided. With some ψ_n at the output of M_3 , systolic work is computed, and this is compared to the output of $FG\ 2$ which follows Starling's curve. The error is repeatedly used to up-date ψ_n until the desired ψ_n is obtained. The above procedure is repeated for various values of v_0 until enough information is obtained to set up $FG\ 1$. With $FG\ 1$ in the circuit, a ventricular model is ready to be used as a part of a circulation model which will not be discussed here.

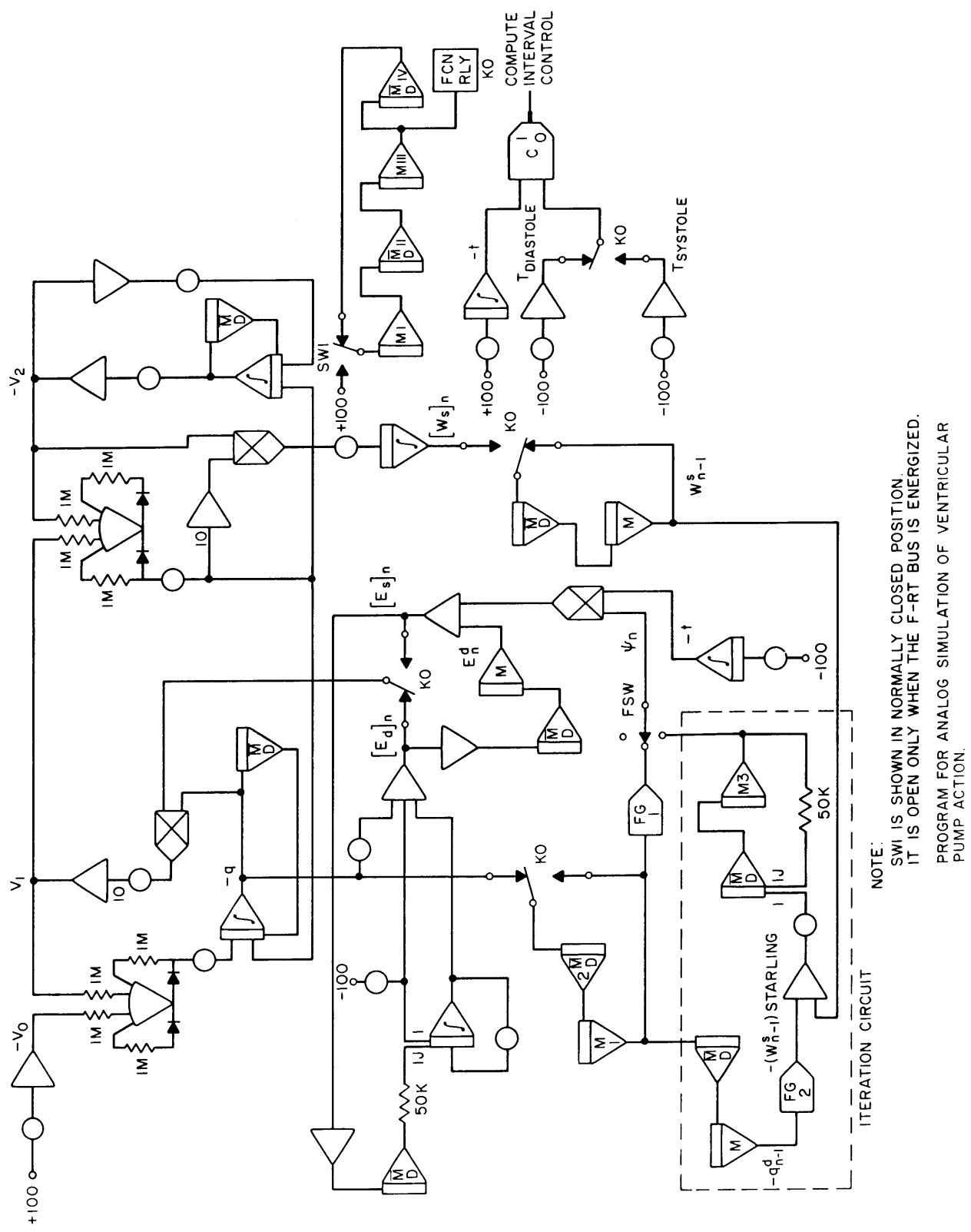


FIGURE 3

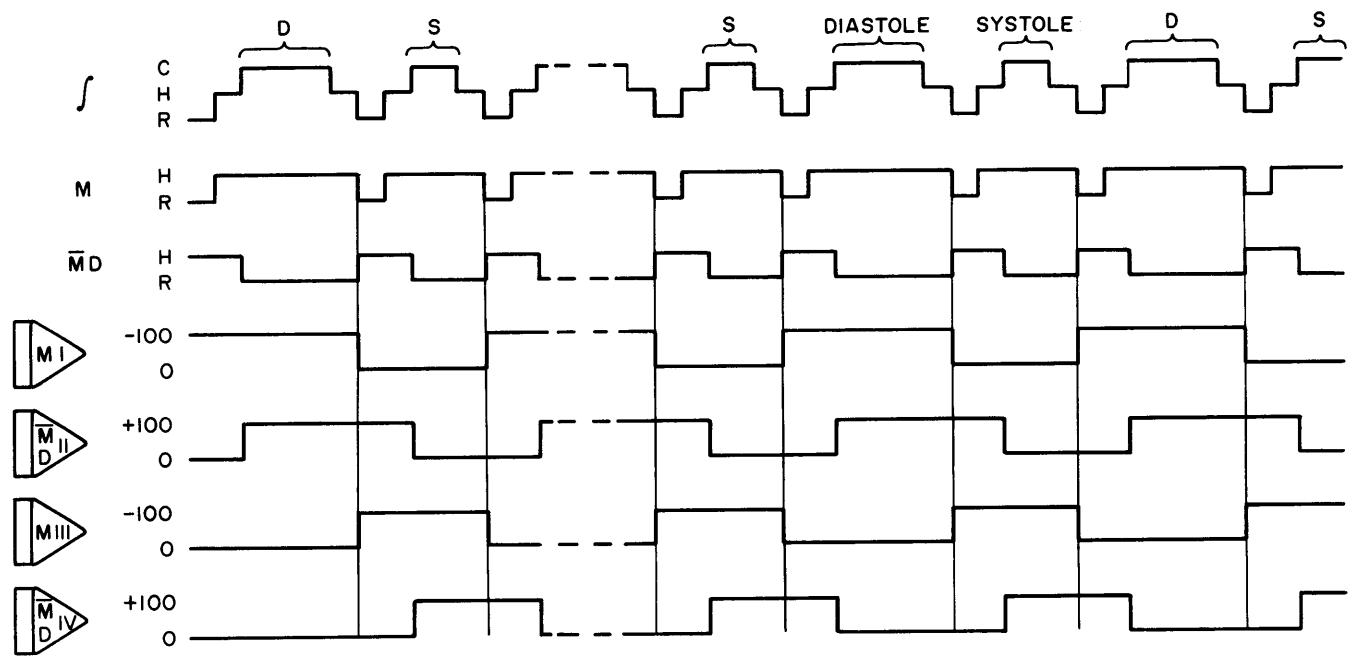


FIGURE 4

CHAPTER 22

A PRACTICAL APPROACH TO ADAPTIVE CONTROL

Originally published in: CONTROL ENGINEERING, May 1966
 Authors: J. W. Banham Jr. and W. L. Smith

Controllers are linear elements that are often required to operate in nonlinear systems. Thus they cannot be expected to provide optimum performance over a wide range of system operating conditions. However, through a linear representation of the nonlinear system a controller can be designed with adaptive features that does provide optimum compensation for transient system requirements. The following text describes the design of a controller and its conversion to the "adaptive" type (one whose dynamic parameters are continuously established as a function of transients in the system). The approach described has been breadboarded and tested for a shipboard system but the design techniques are applicable to other control systems.

The maneuverability of a naval ship depends to a large degree upon the transient response characteristics of the steam generating and machinery systems. Most marine steam generators use forced draft; air for combustion is supplied to the fuel oil burners by variable speed blowers. The forced draft blowers in the machinery plant of a high speed combat ship are generally powered by direct drive steam turbines and regulated by controlling the steam flow to the turbines. The rotational inertial characteristics of these machines, over a wide range of operating speeds, contribute materially to the "sluggishness" of process response to changes in applied load. Designers interested in improving the system response must optimize the controller parameters of those system loops or sub-loops with the longer time constants.

SYSTEM DESIGN STUDY

In a conventional automatic combustion control system, controller settings are usually determined empirically at the load condition of maximum open loop sensitivity. Because controllers are linear elements and the forced draft blowers are nonlinear, the situation is considerably less than optimum over a wide range of operating loads.

a) Nonlinearities in the blower

For any type of rotating machinery, the angular acceleration of the rotor assembly is proportional to the applied differential torque and inversely proportional to the polar moment of inertia. This linear relationship may be expressed by the simple differential equation:

$$\ddot{\omega} = \frac{1}{J} (T_a - T_r)$$

where

$$\begin{aligned}\omega &= \text{angular acceleration, rad/sec}^2 \\ J &= \text{polar moment of inertia, lb-ft-sec}^2 \\ T_a &= \text{driving torque developed by the prime mover, lb-ft} \\ T_r &= \text{load torque imposed by the blower, lb-ft}\end{aligned}$$

Several factors influence the applied torques. The torque developed by a steam turbine is a function of the turbine steam rate and its rotational speed. A typical turbine performance graph illustrating the functional relationship is shown in Figure 1. The parabolic curve on this map represents the design load resistance for the turbine-fan unit. For this particular set of performance data, it was found practical to represent the driving torque by the algebraic expression

$$T_a = \left(\frac{G_s}{G_{smax}} \right) T_{max} - m \left(\frac{G_s}{G_{smax}} \right) \left(\frac{\omega}{\omega_{max}} \right)$$

where

$$\begin{aligned}G_s/G_{smax} &= \text{turbine steam flow, fraction of rating} \\ T_{max} &= \text{maximum locked rotor torque, lb-ft} \\ m &= \text{a constant, lb-ft} \\ \omega/\omega_{max} &= \text{angular velocity, fraction of rating}\end{aligned}$$

The steam flow to the turbine nozzles varies with the steam supply pressure and temperature and the steam valve flow coefficient and port area. The forced draft blowers considered in this study were equipped with a V-ported regulating throttle valve for which the characteristic flow-lift curve is as illustrated in Figure 2. The supply steam state is reasonably constant throughout the load range. The load torque varies directly with the blower system resistance and with the square of the rotational speed.

$$T_r = \beta \left(\frac{\omega}{\omega_{max}} \right)^2$$

LINEAR REPRESENTATION OF THE NONLINEAR SYSTEM

All of the elements of the combustion air flow control loop of a marine steam generating system are characterized by some nonlinearity. The following analysis illustrates one method of representing the forced draft blower nonlinearities by a set of linear functions.

In the basic acceleration equation

$$\ddot{\omega} = \frac{1}{J} (T_a - T_r)$$

the torque terms are functions of the turbine steam flow, G_s and the turbine-fan speed, ω , as shown in the text of this article. The accelerating torque, T_a , was found to be a function of both steam flow and speed; the graphical relationship appears in Figure 1. In this study, the analytical relation between these variables was established by curve-fitting to manufacturer's performance curves, but can also be developed by a detailed derivation of momentum equations applied to the turbine wheel. For a fixed blower air system resistance, the load torque varies with the square of the rotational speed. The constant of proportionality, β , is established by this resistance.

Steam flow to the turbine is regulated by a throttling valve characterized by the arbitrary function

$$G_s = F(L)$$

where L represents valve lift.

Sufficient information is now available to permit the development of a set of transfer functions which describes the open loop frequency response of the turbine-blower unit to small perturbations about a fixed operating level. The general perturbation response is given by

$$\Delta\omega = \frac{1}{J} \int (\Delta T_a - \Delta T_r) dt$$

If a small disturbance is applied at a particular steady state initial condition (L_0 , G_{so} , ω_0), the nonlinear

torque terms may be reduced to linear functions expressed by the following equations:

$$\Delta T_a = \Delta G_s (T_{max} G_{so} - m G_{so} \omega_0)$$

$$\Delta T_r = 2\beta \omega_0 \Delta \omega$$

G_s is a function of valve lift as shown in Figure 2; a tangent constructed at the point (L_0 , G_{so}) has a slope given by

$$\Delta G_s = \frac{d}{dL} F(L) \Big|_{L=L_0} \Delta L$$

Substitution of these linearized functions into the acceleration equation and applying a Laplace operator yields

$$\begin{aligned} \Delta\omega = \frac{1}{Js} & \left[\frac{d}{dL} F(L) \Big|_{L=L_0} L (T_{max} G_{so} \right. \\ & \left. - m G_{so} \omega_0) - 2\beta \omega_0 \Delta \omega \right] \end{aligned}$$

For any particular steady state operating level the terms $d/dL F(L) \Big|_{L=L_0}$, $T_{max} G_{so}$, $m G_{so} \omega_0$ and $2\beta \omega_0$ are constants; the above equation may thus be expressed as a transfer function:

$$\frac{\Delta\omega}{\Delta L}(S) = \frac{\frac{d}{dL} F(L) [T_{max} G_{so} - m G_{so} \omega_0]}{1 + [J/2\beta \omega_0] S}$$

This transfer function defines the response of shaft speed to valve lift for small perturbations given initial conditions defined by L_0 , G_{so} , and ω_0 .

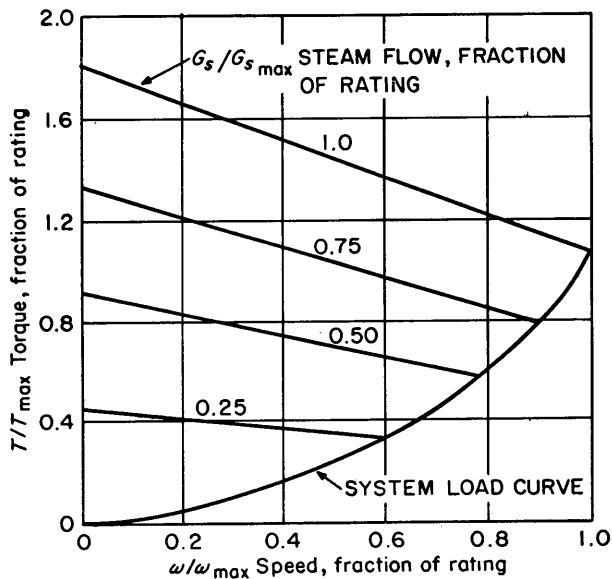


Figure 1. Forced draft blower turbine performance curves.

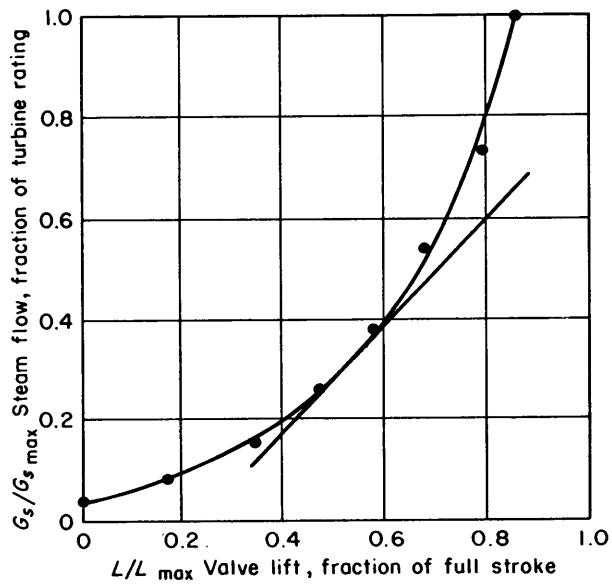


Figure 2. Turbine valve steam regulating characteristics. The arbitrary function $G_s = F(L)$ used in Figures 3 and 7 is illustrated. Construction of a tangent at $(G_s/G_{s\max} = 0.30, L/L_{\max} = 0.53)$ illustrates

the graphical approach to the linear representation of a nonlinear system. The tangent slope represents a valve gain of 1.12 percent of steam flow per percent of valve lift at a point corresponding to 53 percent of lift.

where β is a constant for a particular system resistance, lb-ft. In the design study, the coefficient β was permitted to vary with the effective fan discharge area (established by the number of burner air registers in use in the steam generator).

This information led to the model of the system shown in Figure 3. The problem at this point was to devise a control law which would provide optimum compensation for the nonlinear system over a wide range of operating speeds. For this design, the system is initially described in the frequency domain as a continuous series of linear systems, each similar in form to the others but having different parameters.

b) Linear representation of the nonlinear system

One approach to the linear representation of a nonlinear system is to expand the nonlinear elements of the system by means of the Taylor series, then disregard all but the linear terms. This produces a linear system which approximates the real system response to small perturbations about a fixed operating point. Because the steam admission valve characteristic is an essentially arbitrary function, a graphical approach was used. In this method, all of the nonlinear terms are represented as linear functions by constructing tangents at selected operating points along the curves which represent the functions of the algebraic variables. The slope of each tangent curve is taken as the gain of the element under consideration, Figure 2. The resulting linear model is illustrated in Figure 4. The development of the values of the parameters associated with this model is detailed in the box on page 22-2.

A linear model like the one in Figure 4 leads to a set of transfer functions instead of a single function. Each function in the set represents the open loop frequency response of the system to a small signal sinusoidal disturbance superimposed on a particular steady state initial condition. The validity of the analytical model constructed in this fashion was confirmed by comparison with actual frequency response data obtained from the operating units subjected to pulse test analysis at two load conditions, (Ref. 1). The family of transfer functions thus obtained was used as the basis for a series of control system designs based on classical frequency response techniques applicable to linear systems.

DESIGN OF THE CONTROL SYSTEM

The following conditions were used as criteria for the dynamic performance of the closed-loop system:

- There must be no offset error in the steady state.
- No overshoot in the transient response to a "ramp" input disturbance is to be permitted.
- The time integral of the system absolute error is to be minimized.

The conventional control loop employs a proportional-plus-integral controller which receives as its feedback the output signal from a combustion air flow

DEVELOPMENT OF CONTROLLER DESIGN

Conventional controllers used for regulating forced draft blowers are simple proportional-plus-integral devices characterized by fixed gain and reset rate over the entire operating range. The output signal of a linear controller of this type is given by

$$\theta = \gamma e - \alpha \frac{1}{\tau} \int e dt$$

where e = error between input and feedback signals
 γ = controller proportional gain

$$\frac{1}{\tau} = \text{controller reset rate}$$

An adaptive feature was incorporated in such a controller by servoing the proportional gain and reset rate (integral crossover frequency) settings as required to maintain an optimum combination of closed loop system response and stability over the designed range of forced draft blower operation. These controller parameters are shown plotted against various loop operating levels in Figure 6. Curve fitting of these graphical functions produced the following algebraic expressions:

$$\gamma' = \frac{\gamma}{\omega/\omega_{\max}}$$

where γ' = variable controller gain

$$\gamma = \text{optimum controller gain at } \omega = \omega_{\max}$$

ω/ω_{\max} = steady state loop operating level

$$\frac{1}{\tau'} = \frac{1}{\tau_0} + \frac{1}{\tau_{\max}} \left(\frac{1}{\omega/\omega_{\max}} \right)$$

where $\frac{1}{\tau'} = \text{variable controller reset rate, rad/sec}$

$$\frac{1}{\tau_0} = \text{optimum controller reset rate}$$

$\text{at } \omega/\omega_{\max} = 0$

$$\frac{1}{\tau_{\max}} = \text{optimum controller reset rate}$$

$\text{at } \omega/\omega_{\max} = 1$

The output of the adaptive air flow controller may therefore be expressed by the equation

$$\theta' = \bar{\gamma} \left(\frac{1}{\omega/\omega_{\max}} \right) e + \frac{1}{\tau_0} \int e dt$$

$$+ \frac{1}{\tau_{\max}} \left(\frac{1}{\omega/\omega_{\max}} \right) \int e dt$$

It is to be noted that the controller parameters have been expressed as functions of the control feedback signal (in this particular system, the blower rotational speed). To produce maximum loop gain during any transient disturbance, the controller gain and reset rate are automatically varied as a function of either the command signal ϕ or the feedback signal ω , whichever of the two is least. An arrangement which produces this effect is illustrated in the computer schematic, Figure 7. Using the notation indicated on this diagram, the controller equation becomes

$$\theta' = \bar{\gamma} \left(\frac{1}{\rho/\rho_{\max}} \right) e + \frac{1}{\tau_0} \int e dt$$

$$+ \frac{1}{\tau_{\max}} \left(\frac{1}{\rho/\rho_{\max}} \right) \int e dt$$

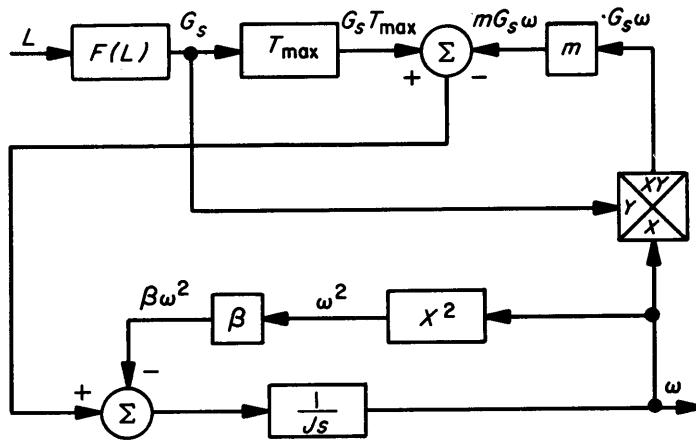


Figure 3. Forced draft blower block diagram.

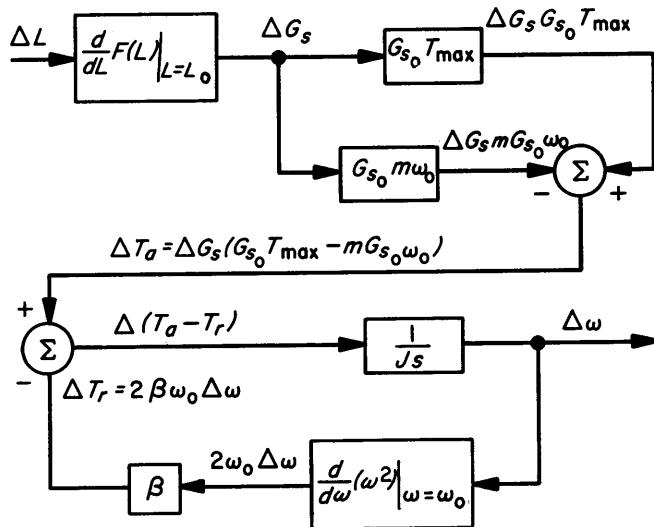


Figure 4. Block diagram of forced draft blower perturbation model at (L_0, ω_0) .

transmitter. For the "adaptive" design, it was elected to utilize the same basic controller concept but to vary the proportional gain and integral cross-over frequency to maintain optimum performance.

The conventional design was initiated by selecting controller parameters which would give the entire control loop a gain margin of six db and a phase margin of 35 deg; these margin goals were derived from previous experience. This conventional procedure was repeated for each of the forced draft blower transfer functions at the various operating load levels. The result, detailed in the box on page 22-4, was a set of empirically optimum controller settings which could be correlated

with the loop input or output to develop analytic functions that would specify the adaptive nature of the design.

The air flow controller thus obtained was simulated together with the forced draft blowers and air flow system on an analog computer. The transient response to input disturbances comparable to the effects of ship maneuvering were then studied. Typical results are shown in Figure 5. Note that a small amount of overshoot occurred which violates one of the system design criteria. This observation led to a repetition of the design procedure, this time using gain and phase margins of 9 db and 45 deg, respectively. Figure 6

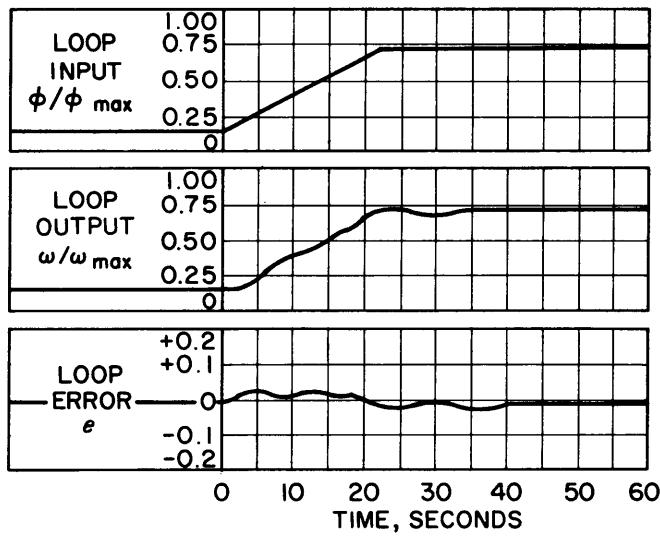


Figure 5. System transient response with initial optimization criteria.

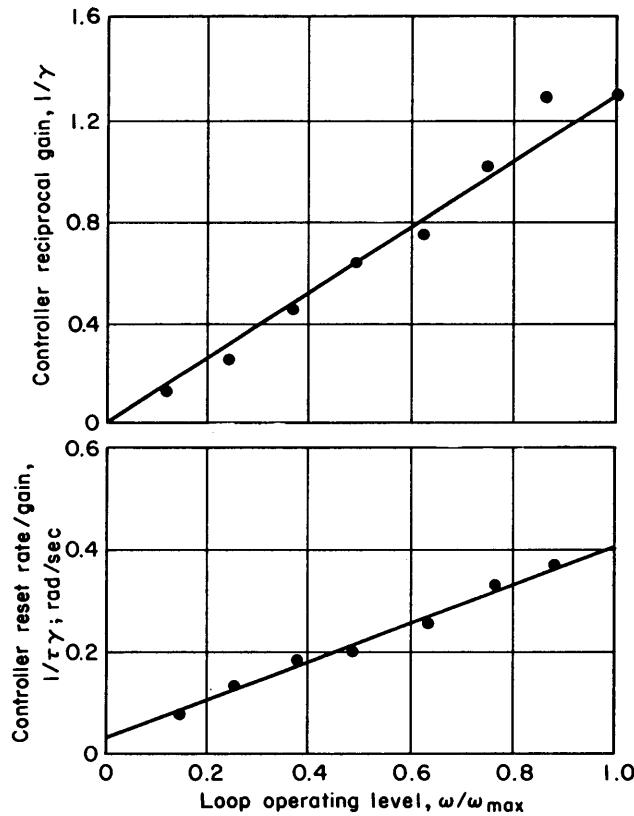


Figure 6. Controller parameters (servoed controller settings) as a function of loop operating level in the steady state. Open loop gain and phase margins are 9 db and 45 deg.

illustrates the relationship between the controller parameters and the loop operating level for performance with the revised open-loop margins.

A revised analog computer program was then prepared; a simplified version of the basic program is shown in Figure 7. Transient response of this simulated system appears in the chart recordings of Figure 8A; the corresponding response of the simulated adaptive control system appears for comparative purposes in Figure 8B. It is evident from these recordings that the adaptive system is considerably superior to the conventional system. Inspection of the error recordings discloses a reduction of five to one in the time integral of absolute error. At the same time there is neither overshoot nor permanent offset. Thus the design criteria have been met.

FROM SIMULATION OF ACTUAL TESTS

Results obtained from the analog computer studies supported a decision to breadboard a pneumatic control system from standard hardware. The basic components used in the adaptive controller were pneumatic proportional and integral amplifiers with servo-driven gain settings. Two of these units provided the necessary proportional-plus-reset control mode. The gains of these two devices were automatically varied as a function of either the input to the closed air flow control loop or its output, whichever produced the greatest loop gain during any transient disturbance. In the steady state, these variables are equal and produce the same gain in the open loop.

Tests of the loop were conducted by imposing "ramp" inputs which corresponded to those used in the computer simulation, thus aiding in an evaluation of control system performance. The recorded results of one of these transients are shown in Figure 8C. The transient performance of the closed loop system closely paralleled that of the simulated system, see Figure 8B.

The air flow control loop comprises one of the open-loop elements of the steam generator's automatic combustion control system. The principal benefit of the adaptive air flow control system when used as a part of the overall steam generator control system is its ability to reduce the deviation of controlled steam pressure and to eliminate the possibility of incomplete combustion (and accompanying smoke) during maneuvering. Although hardware requirements for systems of this type are presently more complex, and less reliable than for conventional systems, adaptive design techniques become attractive when applied to systems constructed from encapsulated or modular design control circuitry. The U. S. Navy is currently investigating adaptive systems of the solid state pneumatic (pure fluid) and solid state electronic types for use in the control of shipboard main propulsion and auxiliary machinery.

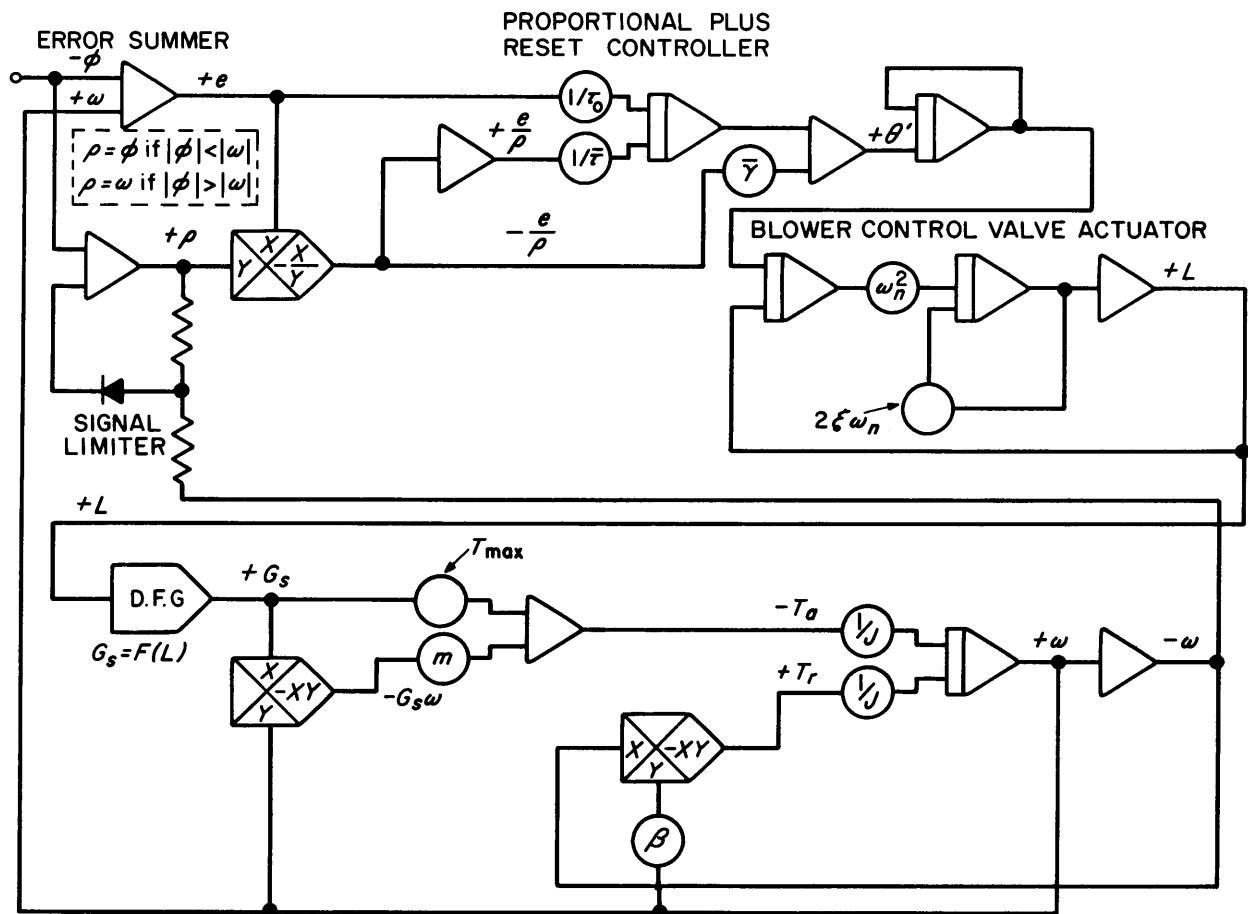


Figure 7. Simplified analog computer program used to study conventional and adaptive controller responses.

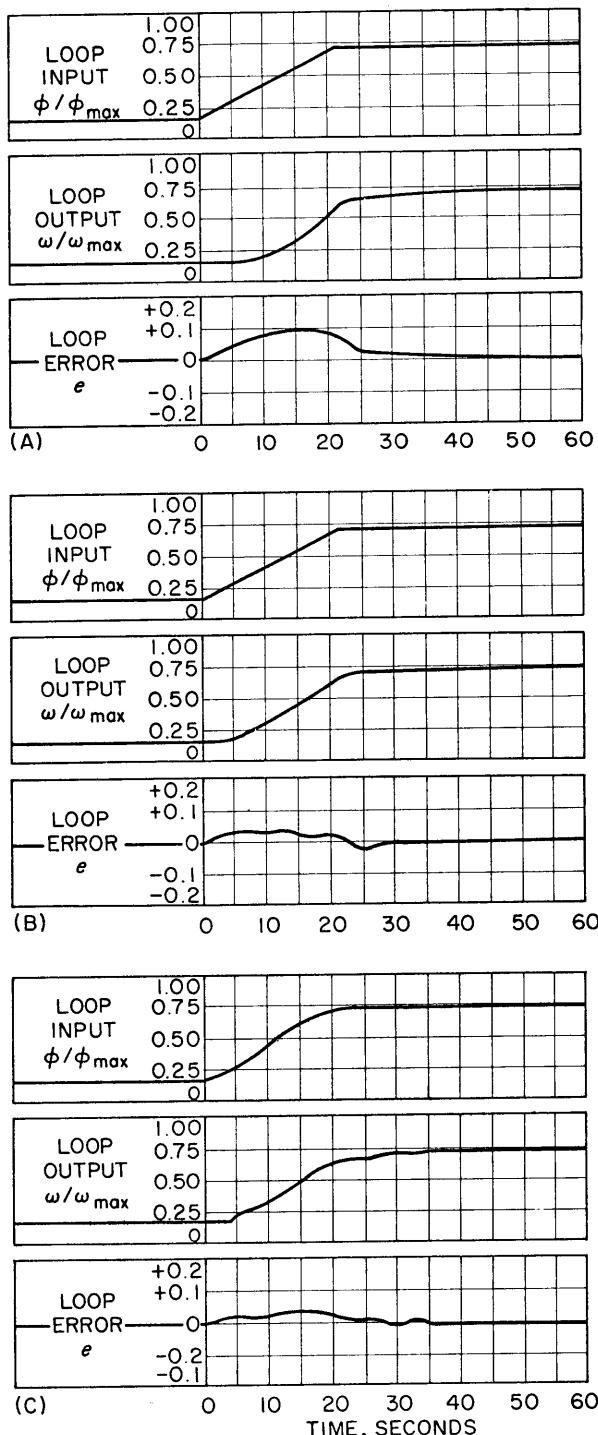


Figure 8. Transient response recordings. A-simulated conventional controller. B-simulated adaptive controller. C-breadboard adaptive controller.

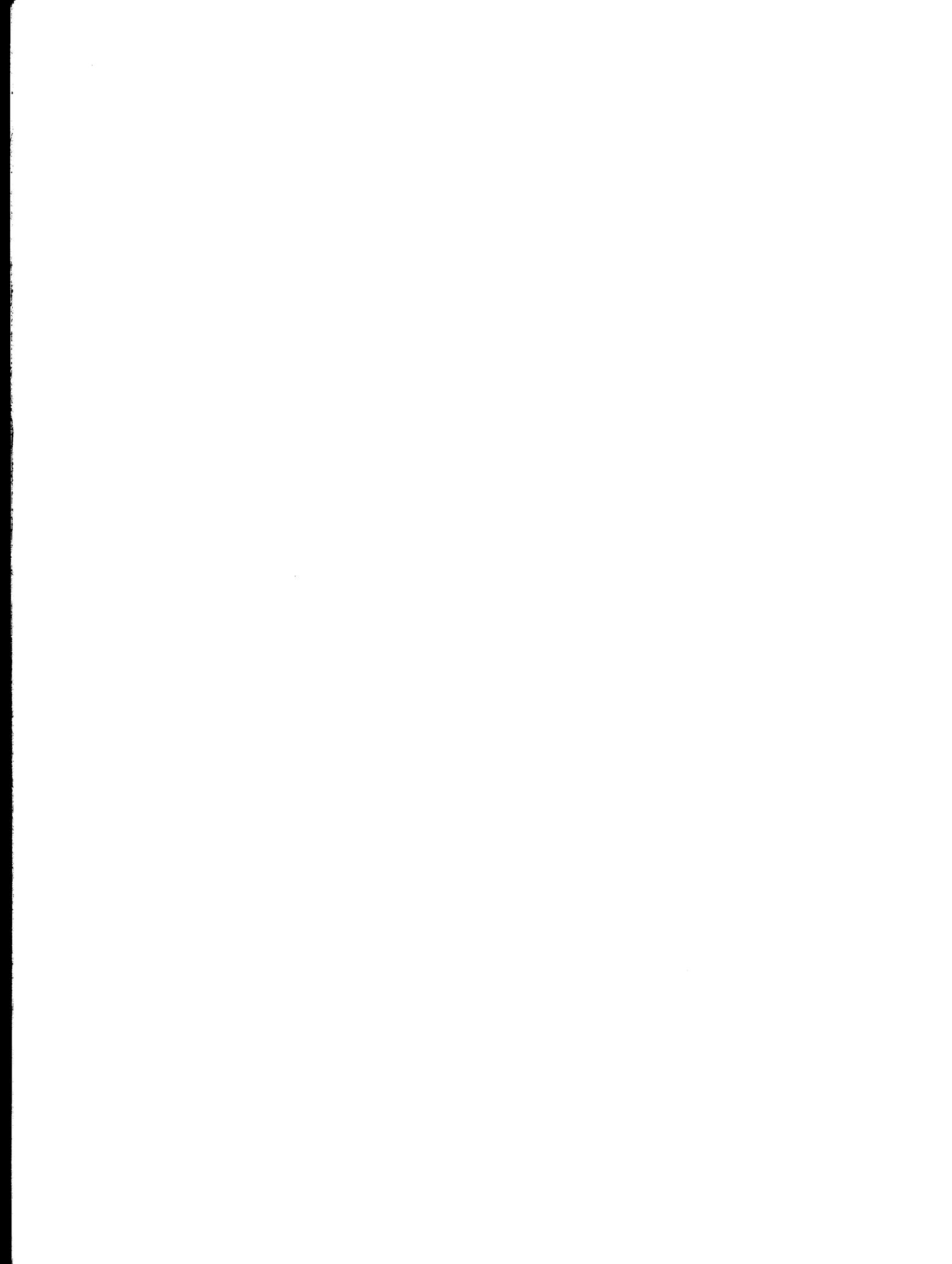
REFERENCES

1. OBTAIN PROCESS DYNAMICS BY PULSE TESTING, J. W. Banham, Control Engineering, April 1965.
2. NONLINEAR AUTOMATIC CONTROL, J. E. Gibson, McGraw-Hill, New York, 1963.

APPENDIX I

Glossary of Abbreviations

C	Capacitor	IO	Iterative Operation
C	Compute	J	Junction
C	Clock	K	Relay
CK	Check	L	Logical Variable
D or DN	Down	M	Meter
DCU	Decimal Counting Unit	NC	Normally Closed
DFG	Diode Function Generator	NO	Normally Open
DVM	Digital Volt Meter	O	Output
ES	Electronic Switch	P or POT	Coef. Potentiometer
Fb	Feedback	R	Reset
FF	Flip-flop	R	Resistor
F-RT	Forward-Reset Real Time	R-H	Reverse-Hold
FR	Function (Operational) Relay	RO	Repetitive Operation
FS	Function Switch	RTS	Reset-Toggle-Set
H	Hold	S	Set
I or IC	Initial Condition	SJ	Summing Junction
IJ	Initial Junction	T	Toggle
IN	Input	TL	Trunk Line
INV	Inverter	U	Up



APPENDIX II

The symbology used throughout this handbook is in conformity with the objectives of Simulation Councils, Inc., namely, to encourage the use of uniform analog computer graphics. This article was prepared by SCI, and appeared in the March 1966 issue of *SIMULATION*. Permission has been granted to reprint this article.

Presenting...

Uniform graphics for *SIMULATION*

The symbols and the methods of laying out analog and hybrid computer diagrams presented here are advocated to alleviate the confusion caused by the uncoordinated invention of new symbols and diagramming practices. The increasing use of hybrid techniques and equipment has aggravated an already bad situation to the point that it is often no longer possible for one worker in our field to read another's diagram. Usually this is because symbols are devised and diagrams are drawn to include details peculiar to a particular kind of equipment. Such a wiring, or "patching," diagram is of course necessary for setting up and checking out an actual simulation, but hardware-peculiar details are only confusing to those with other kinds of equipment. With few exceptions, the use of a simplified signal-flow diagram to illustrate technical articles is much more effective.

With the foregoing in mind an SCI committee composed of

GEORGE BURGIN
JOE HUSSEY
HANS JORGENSEN
GRANINO KORN
JOHN MCLEOD

selected the symbols and offers the following suggestions for their use. Primary considerations were current usage, clarity, and simplicity. We devised no new symbols and, unless there were overriding indications to the contrary, we adopted those already in widest use. Clarity and simplicity, we believe, will be enhanced by the choice of unique shapes to represent different components, and the elimination of all unnecessary details in diagrams.

There was no intent on the part of our committee to set up standards for the industry. However, all diagrams appearing in *SIMULATION* will be prepared according to the committee's recommendations (as they may be modified from time to time), and we hope that these recommendations will prove attractive to others. Suggestions for modifications and additions are solicited.

General rules

The following methods and symbols are recommended for the illustration of technical articles prepared for publication. Unless the purpose of the article is to describe the use of a particular kind of equipment, and the hardware details are pertinent to the subject, such illustrations should not be "hardware-peculiar." In other words, the objective should be to show signal flow, rather than "patching" details.

The primary, or overall, system diagram should show only the essential signal flow. Where it is necessary to show details, separate diagrams should be made and referenced to the primary diagram by enclosing the detailed area of the primary diagram in dotted lines with suitable notation.

The direction of signal flow should be indicated by arrowheads except where the shape of the symbols makes the direction of flow obvious. Primary signal flow (with the exception of feedback loops) should be from left to right, and, if practical, each "line" of symbols should be made to read like the mathematical relation it represents.

The choice of whether to end a line and label it (preferably with the symbol of the variable that the signal represents) when it reaches the right-hand side of the diagram, and then indicate its continuation with the same label as it enters again at the left-hand side, instead of drawing in the connection, should be made on the basis of clarity; if a line returning the signal from right to left will cross many other lines and be hard to follow, it should not be drawn in.

If a diagram involves a number of identical circuits, only one should be shown in detail, while the others should be indicated by boxes with appropriate notation.

Components should not be numbered unless they are referred to by number in the text.

All amplifier gains should be shown just outside the amplifier at the point where the input enters. Unity gains should not be labeled.

Always apply the test of clarity and simplicity. Ask yourself: "Is this the most understandable way to diagram this for those unfamiliar with the hardware, and less familiar with the subject, than I?"

ELEMENT	SYMBOL	FUNCTION	NOTES
Signal flow	Analog 	Binary 	Distinction as to kind of signal is made only at beginning and end of line, and then only where there are two or more kinds of signals to be shown on one diagram. Arrowheads should be omitted only if direction of signal flow is obvious.
	Digital word 	Mechanical 	
	Connections 	No connection 	
Summers	Amplifier 	$x = -(10a + 5b + c)$	No arrows; shape clearly indicates direction of flow. No label on unity-gain input. "N" indicates where number of amplifier should be placed if, and only if, referred to in text.
	Summing amplifier (with high-gain input) 	$x = -(10a + 5b + \mu c)$	Label high-gain input "SJ" or "G" only if pertinent; the point at which c enters (at top or bottom of symbol) indicates a gain of μ .
	High-gain amplifier 	$x = -\mu(10a + 5b + c)$	Note that the numbers at the inputs are now only relative gains; μ , an unspecified large number which is a function of the particular amplifier, is the overall gain.
	Inverting amplifier 	$x = -a$	When amplifiers are used only to change signal sign they should be smaller. Orientation should be that which makes for greatest graphical simplicity, i.e., they may be shown in vertical, or in feedback, paths.
Integrators	Standard 	$x = -a - \int_0^t (10b + 5c + d) dt$	Standard integrator mode is that of basic problem. The point at which a enters (at top or bottom of symbol) indicates that it sets the initial value of x, so it is redundant to label it IC.
	Two-mode 	RESET when $R=1$ COMPUTE when $R=0$	Arrows indicating mode control should be shown only when two or more integrators are operating in different modes simultaneously. The mode-control input should then be labeled or coded to indicate the controlling mode.
	Three-mode 	RESET when $R=1$ or $R=S=1$ HOLD when $S=1, R=0$ COMPUTE when $R=S=0$	
	Track-hold 	TRACK when $R=1$ HOLD when $R=0$	When used in this way the size of the integrating capacitor may be important, as it affects the tracking lag. In such cases the value should be indicated as shown. The quantity tracked or held is $-(a + b + c)$.

ELEMENT	SYMBOL	FUNCTION	NOTES
Integrators (continued)	Memory pair 	Integrator 1 in Track for $S=1$ Integrator 2 in Track for $R=1$	Integrator 1 in TRACK for $S=1$, integrator 2 is TRACK for $R=1$. In certain iterative problems y is required to have some value for the first iteration; thus an "initial" initial condition must be furnished as shown.
Potentiometers	Two-terminal 	$x = ka$ (load connected) $a = 1, b = 0$ (load connected)	The mathematical relationship shown for the three-terminal pot is given only to define k . It will not usually hold during problem solution because if b were always zero a two-terminal pot could be used. If it is desirable to define k in any other way, its meaning should be clearly indicated.
	Three-terminal 		
	Servo-set 	$x = ka$	This should be used only if the fact that N is servo-set is significant to the solution of the problem, as, for instance, when it must be automatically reset after each run of an iterative problem.
	Digital 	$x = k^*a$	As shown, the symbol indicates a digital pot with integral (committed) inverting amplifier. The symbol for non-inverting digital pots should not include the amplifier. *Indicates quantized value.
Function generators	Arbitrary 	$x = f(a)$	The number N should be used only when the function generator is referred to by number in the text, or when it is desirable to explain what the function f is, either by a footnote, or by a separate graph.
	Mathematical functions (typical) 	$x = \sqrt{a}$	In cases where the function generated can be represented by a standard mathematical symbol, the f should be replaced by the symbol.
	Multiplier 	$x = \frac{ab}{\alpha}$ $x = \frac{\alpha a}{b}$	These symbols should be used for all analog multipliers and dividers; if the kind is significant to the solution of the problem, it should be so stated in the text and/or noted on the diagram. Because equipment differs, the sign of the output for specified signs of both inputs should be given, otherwise inversion will be assumed.
	Divider 	$\alpha = \text{reference voltage}$	The number N should be used only if the component is referenced in the text, or if more than one channel of a multichannel device is used. In the latter case the subscript, in this case i , identifies the channel. In many cases, it will be found that a drawing can be simplified and clarified by drawing the same multichannel device in more than one signal flow path. In this case the number would be the same, but the subscript would be different for each channel.
	Polar to rectangular 	$x = R \cos \theta$ $y = R \sin \theta$	If a resolver has additional outputs they should be shown only if used.
	Rectangular to polar 	$\theta = \arctan y/x$ $R = \sqrt{x^2 + y^2}$	
	Comparators With binary output 	$U = 1 (a < b)$ $U = 0 (a > b)$ $\bar{U} \neq U$	The symbol for a relay comparator can, of course, be made by combining either of these symbols with that of a relay.
	With true and false binary outputs 		

ELEMENT	SYMBOL	FUNCTION	NOTES
Relay		$U_a = x \quad (U=0)$ $\tilde{U}_a = x \quad (U=1)$	If the diagram is drawn as shown, the designations of arms and contacts are superfluous. However, it is often desirable, for clarity, to show the relay coil in one place on a diagram and the contacts in signal-flow paths elsewhere. In this case, N_{aa} would designate the normally Open ($U=0$) contact of channel a, relay N; N_{aa} the Arm of channel a of relay N, etc.
Digital inverter		$x = \tilde{a}$	The tilde (~) is used instead of a bar to indicate negation. This is recommended because of the many other meanings of a bar over a variable.
Digital/analog switch		$U_a = a \quad (U=1)$	ON when binary signal, U, is digital "one."
Limiters	<p>Feed-back</p> <p>Bridge</p>	$x = -a \quad (l < a < u)$ $x = l \quad (a < l)$ $x = u \quad (a > u)$	u = larger value at which output is limited l = smaller value at which output is limited
Converters	<p>Analog-to-digital</p> <p>Digital-to-analog</p>	$x^* = a$ $x = a^*$	*Indicates quantized value.
Diode	Solid state 	$x = a \quad (a > x)$ $x = 0 \quad (a < x)$	
Diode gates	<p>AND</p> <p>NAND</p> <p>OR</p> <p>NOR</p>	<p>AND $x = a \cdot b \cdot c$</p> <p>NAND $x \neq a \cdot b \cdot c$</p> <p>OR $x = a + b + c$</p> <p>NOR $x \neq a + b + c$</p>	<p>Output is high (logic 1) if and only if all inputs are high.</p> <p>Output is low (logic 0) if and only if all inputs are high.</p> <p>Output is high (logic 1) if and only if at least one input is high (i.e., any or all of the inputs are high).</p> <p>Output is low (logic 0) if and only if at least one input is high (i.e., any or all of the inputs are high).</p>
Flip-flop	Typical 		As there are many kinds of flip-flops, symbolic representation of the operation of any but the simplest is not advised; if the operation is not obvious, an explanation in the text or a footnote with the diagram is recommended.
Black box		This is Black Box number N, to be used if no symbol is given here, or in case of doubt! It can have any number of inputs and outputs, but the nature of the signals should be indicated and all signal-flow directions should be designated by arrowheads. Here there are three analog inputs giving rise to one analog and one binary output.	The function of a "black box" can often be made obvious by properly labeling the inputs and outputs. However, if it is not obvious, the function should be explained in the text or by a footnote with the diagram. If internal details are of interest, they should be shown in an appropriately labeled auxiliary diagram.

102687275