NASA TECHNICAL NOTE

NASA TN D-3042

NASA TN D-3042

DESIGN OF REAL TIME COMPUTERS UTILIZING COUNTING TECHNIQUES

by George J. Moshos

Lewis Research Center

Cleveland, Ohio

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION • WASHINGTON, D. C. • OCTOBER 1963

DESIGN OF REAL TIME COMPUTERS UTILIZING COUNTING TECHNIQUES

By George J. Moshos

Lewis Research Center
Cleveland, Ohio

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

For sale by the Clearinghouse for Federal Scientific and Technical Information
Springfield, Virginia 22151 – Price $3.00

DESIGN OF REAL TIME COMPUTERS UTILIZING COUNTING TECHNIQUES

by George J. Moshos

Lewis Research Center
Cleveland, Ohio

SUMMARY

The design of a class of special purpose computing machines, which compute by counting, is systematically developed. The basis of the design philosophy is to limit the basic building elements to three fundamental units and to develop the method of synthesis such that these three building elements are represented as operational units. In particular, the three basic building elements are (1) the binary rate multiplier, which is a means of scaling down a pulse stream to some specified fraction, (2) the counter, and (3) the anticoincidence circuit, which is a means of separating pulses arriving at the counter simultaneously. The computational errors, that is, rounding-off and truncation errors, introduced into the machines when these elements are treated as operational units are identified. The method of synthesis is explicitly stated, and a wide variety of machines obtained directly from this synthesis are presented. Finally, a series of machines is presented for interpolation and extrapolation of a function, which is available only as empirical data.

INTRODUCTION

Computers are usually divided into two broad categories, analog and digital. Analog computers represent variables as physical quantities. The solution of a problem in an analog computer is attained by constraining a physical model of the problem to be solved and measuring the variables. The ability to program a wide variety of problems is achieved by having functional components available (e.g., adders, multipliers, integrators) and interconnecting them by means of a patchboard. The resulting interconnection is scaled to match the desired equation. On the other hand, digital computers represent variables as discrete quantities. The usual method of solution of a problem in a digital computer is attained by sequencing instructions through the fetch-execute cycle of its

control unit. Another class of digital computers, known as incremental computers, combines the parallel functional simplicity and speed of analog computers with the ability of attaining computational precision, which is not dependent on precision of measurements. Such computers attain a speed advantage over conventional general purpose computers by transmitting and processing only partial words in a number of parallel arithmetic organs rather than the whole words needed by the fetch-execute cycle. Moreover, the digital nature of these computers permits the problem solution to be repeated exactly and therefore does not possess the drift characteristic of analog computers. The incremental computer, which has commanded the most interest in the literature, is the digital differential analyzer, that is, DDA. This computer can be viewed as a digital analogy of an analog computer. The usual design practice in each of these machines is to permit them to solve a large spectrum of problems. When a computer need arises for a special purpose application, this versatility provided by general purpose computers is felt as an added cost factor.

A class of incremental techniques, known in the industry as countup-countdown techniques, has been used in real time control. Data is represented in these techniques by a unitary code. For example, the number 28 is represented by 28 pulses. A function may be represented by counting the sequence of pulses in a forward-backward counter or converting them directly into an analog quantity (e.g., by a stepping motor) for analog processing. Consequently, when a real time application deals with continuous smoothly varying functions, countup-countdown techniques offer a simplicity of hardware that is economically more attractive than computing systems designed to handle a large spectrum of problems.

The purpose of this study is to investigate countup-countdown techniques with the objective of demonstrating that they can, in fact, be used to generate a wide variety of nontrivial functions. This will be done by displaying a circuit that will generate each function. However, since the techniques on which this study is based are described in the literature only in an ad hoc manner (refs. 1 to 5), only specific circuits will be permitted as the basic building elements. In particular, the fundamental units to be permitted are (1) the binary rate multiplier (BRM), which is a means of scaling down a pulse stream to some specified fraction, (2) the counter, and (3) the anticoincidence circuit, which is a means of separating pulses arriving at the counter simultaneously. In order to strengthen the previous argument the explicit use of whole word adders and subtractors will be avoided completely. A succinct recapitulation of the purpose of this study is to develop and demonstrate systematically the versatility of techniques based on counting for solving sophisticated and practical special purpose computer design problems.

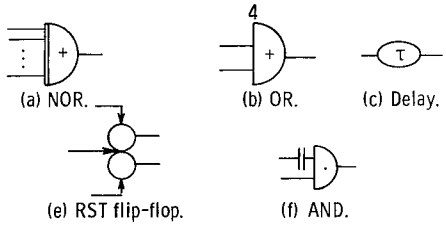


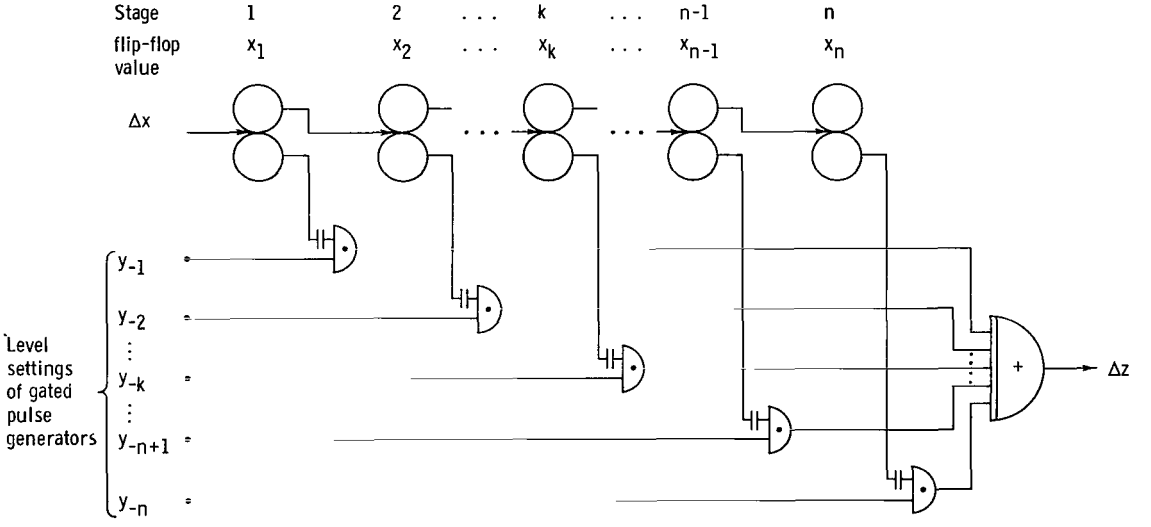
Figure 1. - Basic logic elements.

The method of synthesis herein will be to describe the principal building elements as operational units and then proceed by operational techniques to show how to fabricate the various machines. In particular, a first-order difference equation can be represented by a counter, and approximate integration can be attained by using a counter in cascade with a binary rate multiplier.

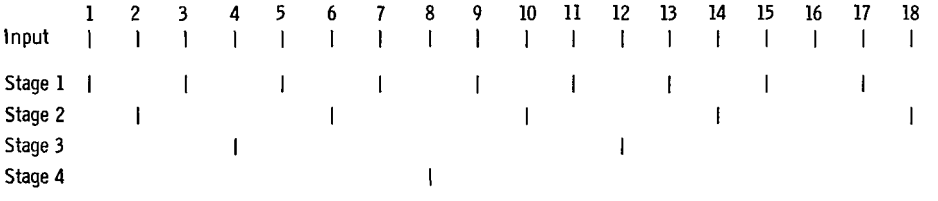
The principal design elements can be described in terms of the logic elements shown in figure 1. The NOR element shown in figure 1(a) may have various numbers of inputs. The parallel lines shown on one of the inputs in figure 1(f) are included to indicate that the AND circuit is intended to act as a pulse gate dependent on the level setting of the other line.

BINARY RATE MULTIPLIER

A binary rate multiplier is a means of scaling down a pulse stream to some specified fraction. A logic diagram of a BRM, which is built of standard logic elements, is shown in figure 2(a). This circuit is described in detail in several of the references (e.g.,



(a) Logic diagram.



(b) Timing diagram.

Figure 2. - Binary rate multiplier.

refs. 2 and 5); consequently, a brief description here will suffice. The input pulse stream is applied directly to the binary counter whose value is denoted by $x_n x_{n-1} \cdot \cdot \cdot x_2 x_1$. Each flip-flop of the counter is operated as a trigger. For every two input pulses to a trigger two output pulses are produced; one pulse when the flip-flop makes a 0 to 1 transition called an α pulse and one when the flip-flop makes a 1 to 0 transition called a β pulse. The β pulse is used to trigger the successive stage of the counter. The α pulses are gated through AND gates and mixed through a NOR element to produce the desired fraction of the input pulses. This simple mixing technique may be used because the α pulses from the various stages are separated in time from each other. This timing factor is shown in figure 2(b).

The quantitative relation of a BRM may be expressed as

$$\Delta z = y\Delta x \tag{1}$$

where y is the binary number represented by the settings of the AND gates; that is, $y = .y_{-1}y_{-2} \cdot \cdot \cdot y_{-n}$, Δx is the number of input pulses and Δz is the number of output pulses. If y remains constant over a Δx interval of 2^n pulses, where n is the number of stages of the BRM, then this relation is exact. However, if Δx is less than 2^n pulses, this multiplicative relation is only approximate, where the difference between the actual output and that given by equation (1) is the rounding-off error in the multiplication. It can be demonstrated that this approximation can be improved by increasing the number of stages and that it varies with the starting conditions of the BRM counter.

TABLE I. - EXAMPLES OF p-SEQUENCES

Pulse	p-Sequence		
	Two stage	Three stage	Four stage
1	10	100	1000
2	01	010	0100
3	10	100	1000
4	00	001	0010
5		100	1000
6		010	0100
7		100	1000
8		000	0001
9			1000
10			0100
11			1000
12			0010
13			1000
14			0100
15			1000
16			0000

Because of the approximate nature of equation (1) when Δx is less than 2^n pulses, the specific output sequence will be calculated in demonstrating specific machines. For these calculations, the pulse stream shown in figure 2(b) may be displayed in mathematical vector form. This will be called the p-sequence. Each position of the vector in this sequence represents the possible output at a particular pulse time from a stage of the BRM. The p-sequence for a two-, three-, and four-stage BRM is displayed in table I.

The p-sequences given in the table assume that the BRM counter starting value is zero. If another starting value is used, its associated p-sequence can be easily obtained. Moreover, if an interval greater than 2^n pulses is used, the

p-sequence can be obtained by repeating the p-sequence given in the table.

The sequence of output pulses may be calculated by multiplying bit by bit the p-sequence with the respective values of the level settings of the AND gates. This process is illustrated by two examples as follows:

Example A:

$$\begin{pmatrix} 100 \\ 010 \\ 100 \\ 001 \\ 100 \\ 010 \\ 100 \\ 000 \end{pmatrix} * \begin{pmatrix} 01010101 \\ 10111011 \\ 11101111 \end{pmatrix} = \begin{pmatrix} 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \\ 0 \end{pmatrix}$$

Example B:

$$\begin{pmatrix} 100 \\ 010 \\ 100 \\ 001 \\ 100 \\ 010 \\ 100 \end{pmatrix} * \begin{pmatrix} 1010101 \\ 0100010 \\ 0001000 \end{pmatrix} = \begin{pmatrix} 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \\ 1 \end{pmatrix}$$

The first matrix, in each of these examples, is the p-sequence. The next matrix represents successive values of the gate settings. When these two matrices are multiplied together by conventional matrix multiplication, the actual result of the BRM output is developed along the diagonal of the resultant matrix. This restricted form of matrix multiplication is notated by an asterisk and by displaying the result as a vector on the right side. The first element of the resultant vector represents the output of the BRM after the first input pulse to the BRM is applied; the second element of the resultant vector represents the output of the BRM after the second input pulse is applied, etc.

The expected output value of example A after the first input pulse is given by equation (1) and is equal to 3/8 of a pulse, since $y = 3/8$ and $\Delta x = 1$. The expected output value after the second input pulse is the expected output value of the first input pulse plus the expected output value resulting from the second pulse. Therefore, the expected output value of example A is 35/8 pulses for the eight input pulses. As shown by actual com-

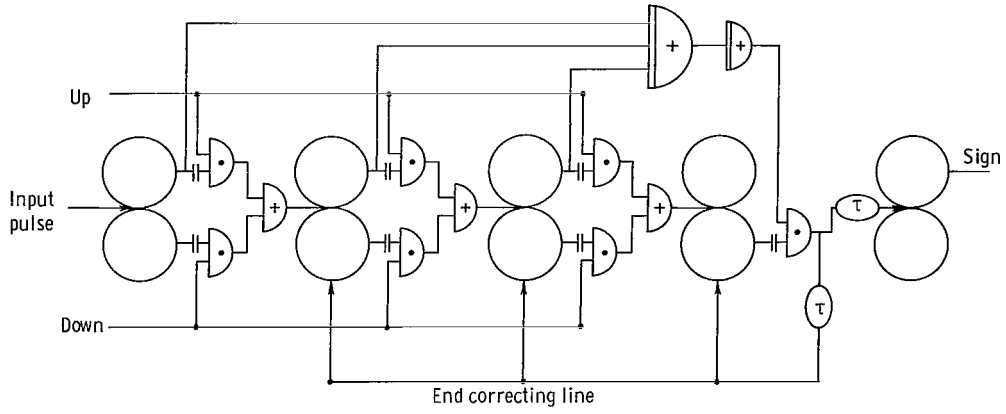


Figure 3. - Signed four-stage bidirectional counter.

putation, however, the BRM yields zero output pulses. On the other hand, the expected value of example B by equation (1) is 21/8 pulses for the seven input pulses. The preceding computation yields seven output pulses. Both of these examples are pathological cases in the use of the BRM. The approximate nature of equation (1) can ordinarily be expected to yield more realistic results. Some of these results are presented in this study.

The method of synthesis to be presented necessitates that the BRM operate on signed quantities. In particular, the level setting of the AND gates and the counter input pulses must be signed quantities, and the BRM is to yield a signed pulse output. If the output pulses are accumulated in a counter, the sign of the pulse will determine the direction of counting. If the output pulses are used to drive a stepping motor, the sign of the pulse will determine the direction the stepping motor is to turn. Throughout this discussion the signs of various quantities are considered available through level logic. Consequently, the output sign can be obtained from the input signs by an exclusive OR circuit.

COUNTER

The purpose of the counter in the machines that will be considered is twofold: (1) to accumulate the pulses arriving at the counter in order to display the total number of counts and (2) to set the levels of the BRM's. In the first application, the counting sequence can be any desired sequence for a terminal device. In many real time applications the output pulses may not be accumulated directly but are converted to an analog quantity for analog processing (e. g., by a stepping motor). In the second application, the counting sequence must be compatible with the BRM. This general requirement can be met by the circuit displayed in figure 3.

A number is represented in this counter by magnitude plus sign. As had been stated

TABLE II. - DOWN-COUNTING SEQUENCE

Input pulse	
-1	+1
+111	-111
+110	-110
+101	-101
+100	-100
+011	-011
+010	-010
+001	-001
+000	-000
+111 → +001 → -001	-111 → -001 → +001

TABLE III. - UP-COUNTING SEQUENCE

Input pulse	
-1	+1
-000	+000
-001	+001
-010	+010
-011	+011
-100	+100
-101	+101
-110	+110
-111	+111

TABLE IV. - COUNTER SIGN CONTROL

Sign input pulse	Sign counter	Line activated
+	+	Up
+	-	Down
-	+	Down
-	-	Up

earlier, the signs are represented by level logic. The counter counts down in magnitude when the input pulse and counter are opposite in sign and counts up in magnitude when the counter and input pulse have the same sign. The circuit is designed so that the α pulses are used to count down and the β pulses used to count up. There are two representations of zero, that is, minus zero and positive zero. When the counter value is at +00 . . . 0 and a -1 pulse arrives, the counter is set to -00 . . . 01. This end correction is accomplished in three steps. The normal sequence first changes the counter value to +11 . . . 1. The magnitude is then corrected in the second step to +0 . . . 01. Finally, the sign is changed to -00 . . . 01. The sign is changed last so that the β pulses generated when the magnitude is corrected do not propagate to the successive stages of the counter. In a similar manner to that just given, the counter is set to +00 . . . 01 when the counter value is -00 . . . 0 and a +1 pulse arrives. The down-counting sequence for a three-stage counter is given in table II.

The up-counting sequence utilizing the β pulses of the flip-flops is given in table III.

Since the signs of both the pulse output of the BRM and counter value are to be processed by level logic, the activation of the up-down line is accomplished by an exclusive OR circuit. This is obvious from table IV.

ANTICOINCIDENCE CIRCUIT

Pulses arriving at a counter simultaneously must first be separated before they are entered into the counter. The circuit that accomplishes this task is called an anticoincidence circuit. Fundamentally, this circuit necessitates storing each pulse as it arrives. Each stored pulse is then presented to the counter according to a fixed program.

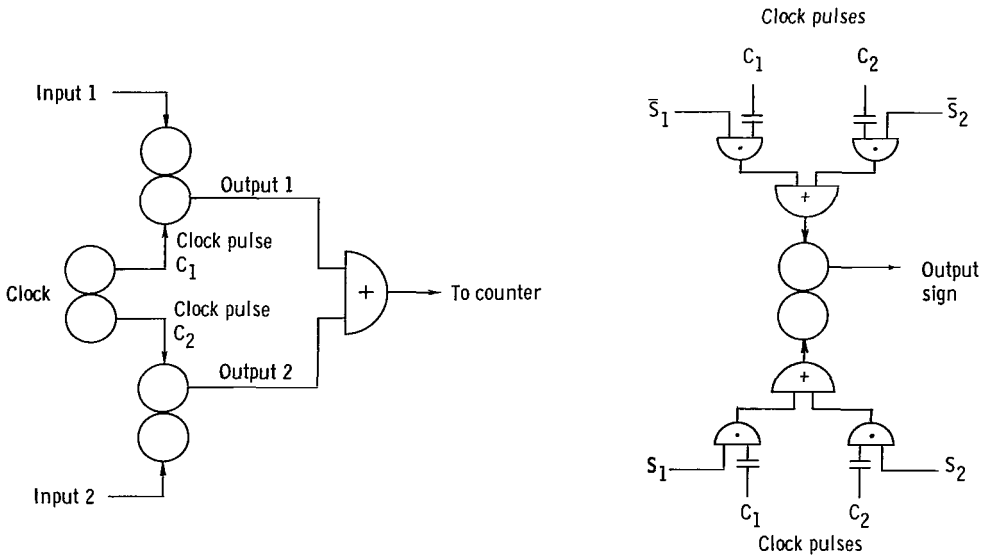


Figure 4. - Anticoincidence circuit.

The circuit configuration that can accomplish this task for two inputs is shown in figure 4.

The operation of the circuit given in figure 4 is as follows: If a pulse from input 1 exists, it is stored in flip-flop 1. If a pulse from input 2 exists, it is stored in flip-flop 2. It will be noted that these two inputs can arrive simultaneously. Two pulses are emitted by the clock that are separated from each other. If flip-flop 1 had been set by input 1, it is reset by the clock pulse C_1, which in turn generates an output pulse. If flip-flop 1 had not been set, no output pulse will appear in the output. Flip-flop 2 is reset and an output is similarly generated by clock pulse C_2. Since clock pulses C_1 and C_2 are separated, the corresponding output pulses are also separated.

If either input arrives at the same moment as its associated clock pulse, this circuit may fail. However, it is usually the case that the inputs are time dependent on the clock, and natural time lags prevent them from arriving simultaneously. In any event, it is sufficient for this study just to state that this circuit can be expanded to adjust for such failures since the manner by which this can best be accomplished depends intimately on the particular application.

Since the sign of a pulse is processed by level logic, the signs need not be stored before they are presented to the anticoincidence circuit. When a pulse is presented to the counter, however, its sign must also be presented. This may be simply accomplished by shifting the sign level to a flip-flop by the separated clock pulses C_1 and C_2. This circuit is also shown in figure 4 where S's and $\bar{S}$'s are the sign levels of the pulses and their complements, respectively.

If more than two inputs arrive at the counter simultaneously, a need arises for a circuit other than a simple clock to separate the stored pulses. The design of such a

TABLE V. - PULSES GENERATED BY SEVERAL COUNTING SEQUENCES

Counting sequence	Pulses generated	Counting sequence	Pulses generated	Counting sequence	Pulses generated
000	$--\alpha$	000	$--\alpha$	000	$--\alpha$
001	$-\alpha\beta$	001	$-\alpha-$	001	$-\alpha-$
010	$--\alpha$	011	$--\beta$	011	$\alpha--$
011	$\alpha\beta\beta$	010	$\alpha--$	111	$--\beta$
100	$--\alpha$	110	$--\alpha$	110	$-\beta-$
101	$-\alpha\beta$	111	$-\beta-$	100	$\beta--$
110	$--\alpha$	101	$--\beta$		
111	$\beta\beta\beta$	100	$\beta--$		

multiphase clock is outside the scope of this study. Nevertheless, it is instructive to digress and indicate some techniques by which this can be accomplished. First, it is noted that a simple binary counting sequence such as the leftmost sequence shown in table V will serve this purpose. It will be noted, however, that, while this sequence can generate more than two steps, the α and β pulses from the various flip-flops are not separated, so consequently cannot both be used.

All the sides of the flip-flops could be used if the counting sequence utilized a unit distance code. Such a code would guarantee that not more than one flip-flop would change state at any step of the counting sequence. Consider the Gray code counting sequence given by the middle sequence in table V. This counting sequence can be used to separate as many as six inputs arriving at the counter simultaneously; however, this requires the counter itself to go through eight steps. An example of a counting sequence that may be used to handle six inputs and yet go through only six steps in the counting sequence is given by the rightmost sequence in table V.

SCHEMATIC REPRESENTATION

The three circuits described in this section are the principle design elements; however, in describing the machines promised in this report, these three circuits will be represented as operational units. The advantages to be gained by using operational units rather than these circuits are twofold. First, the method of synthesis can be more clearly presented. Secondly, a considerable hardware reduction can usually be realized when the composite machine is considered because these simplifications arise when all the features of these basic circuits are not required. A checklist of the features, which when removed would simplify the basic circuits, would include (1) sign control of BRM may not be required, (2) counter may not be required both to countup and countdown,

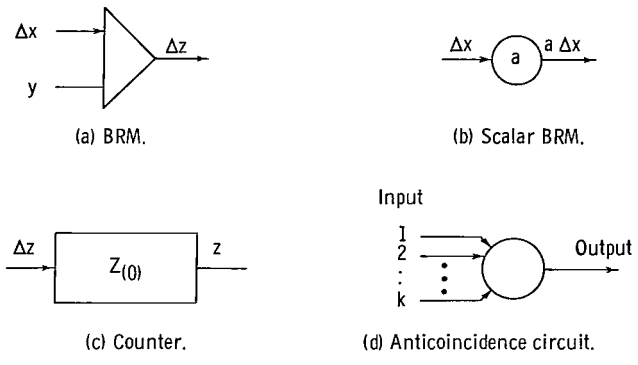


Figure 5. - Schematic diagrams of principle design elements.

(3) two BRM's may receive the same input pulse stream with the result that one BRM counter may be used with two sets of AND gates, and (4) the level setting of the BRM may be constant with the result that a scaling circuit (see ref. 5) rather than a BRM may be used. These three circuits are, however, sufficient as principal design elements.

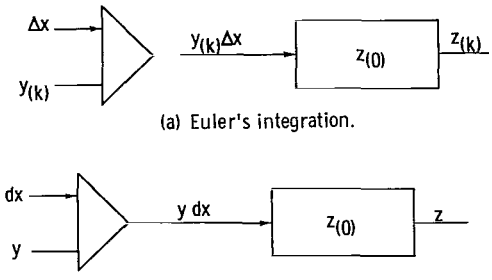
The three principal design elements as operational units are presented in figure 5. The BRM is represented by the schematic diagram shown in figure 5(a). The value y in this diagram is less than 1 and is obtained from level logic. The quantities Δx and Δz are the input and output pulse streams, respectively. The input-output relation for this diagram is expressed by equation (1). Alternately, the BRM is represented by the schematic diagram shown in figure 5(b) when the value of y remains constant. In these cases the BRM may be replaced by a scaling circuit in the final design. Figure 5(c) presents the schematic diagram of the counter. The quantity Δz is an input pulse stream and z is the output that may be used in level logic. When the counter is used to set the levels of the gated pulse generators of a BRM, a scale reduction of 2^{-n} is implied by the connection. At times this scale reduction will be shown explicitly by the same diagram shown in figure 5(b). The initial conditions of a counter may be shown explicitly by inserting it in the box, that is, $z_{(0)}$. If the counter value is considered to be a function of iterative steps, the input-output relation may be expressed by the first-order difference equation:

$$z_{(k)} = z_{(k-1)} + \Delta z_{(k-1)} \tag{2}$$

The value of $z_{(k)}$ in equation (2) in terms of the initial condition of the counter is

$$z_{(k)} = z_{(0)} + \sum_{i=0}^{k-1} \Delta z_{(i)} \tag{3}$$

Figure 5(d) represents the schematic diagram of an anticoincidence circuit. This circuit accepts multiple pulse inputs and produces a single pulse output. The design of this circuit is such as to permit the input pulses to arrive simultaneously. At times, however, it will be convenient to use this schematic diagram for multiple pulse inputs even if the pulses are known to be separated. This usage, therefore, should permit a corresponding



(a) Euler's integration.

(b) Integration model.

Figure 6. - Integrator.

simplification in the final design. Finally, the convention is adopted in each of these schematic diagrams to use an arrowhead on a pulse logic input or output line and not to use an arrowhead on a level logic input or output line.

Approximate integration can be attained by using a counter in cascade with a BRM. This configuration is shown in figure 6(a). If the output of the BRM is considered as a function of iterative steps, the output at iterative step k is $y_{(k)}\Delta x$. Entering this output directly into the counter yields the first-order difference equation:

$$z_{(k)} = z_{(k-1)} + y_{(k-1)}\Delta x \tag{4}$$

If $z_{(0)}$ represents the initial value of the z counter, equation (4) may be expressed as

$$z_{(k)} = z_{(0)} + \sum_{i=0}^{k-1} y_{(i)}\Delta x \tag{5}$$

This equation is recognized as Euler's (rectangular) integration. A model of this process, which is convenient for machine synthesis, is presented in figure 6(b). The deviation of the results given by the model from that given by equation (5) is the truncation error.

SYNTHESIS (DIFFERENTIAL EQUATION)

The method of synthesis that will be applied in this section is to express the function to be generated as the solution to a differential equation. It has been demonstrated with analog techniques that a wide variety of functions can be generated by utilizing only integrating units and adders (e. g., refs. 6 and 7). For example, with a mechanical differential analyzer, the basic units are a ball-disk integrator and a differential. In the synthesis of countup-countdown machines, the integrator model of figure 6 and the anticoincidence circuit, which permits the summation of two pulse streams, will serve as these operational units. It should be reemphasized that the principal design elements were recognized as entities only for purposes of synthesis, and that the fabrication of the actual machines may permit circuit simplification, which may result in reductions of the hardware requirements.

The first step in this synthesis is to express the function to be generated as a

differential equation such that the highest order derivative is isolated; that is,

$$\frac{d^m y}{dx^m} = f\left(\frac{d^{m-1} y}{dx^{m-1}}, \ldots, \frac{dy}{dx}, y, x\right) \tag{6}$$

The independent variable in equation (6) is represented by a clock. The next step in the synthesis is to assume that a circuit has been designed to generate the highest order derivative. Integrators may then be used to reduce successively the order of the derivative according to the equation

$$\frac{d^{k-1} y}{dx^{k-1}} = \int \frac{d^k y}{dx^k}\, dx \tag{7}$$

A circuit for generating the highest order derivative, whose existence had previously been assumed, may now be developed by the constraint defined by the right side of equation (6). The preceding process terminates the design of the basic configuration for generating the function.

The schematic diagram of the machine just designed must then be scaled in order to (1) match exactly the defining equation and (2) accommodate the range of variables in a finite machine. In particular, the counters are used in the machine configuration to handle magnitudes that have a finite excursion based on the number of stages. Therefore, when a bidirectional counter is used, its magnitude must be such that

$$|\text{Counter value}| \leq 2^n - 1 \tag{8}$$

Since the level setting of a BRM must be less than 1, when a counter is used for this purpose its scale will be reduced accordingly; that is,

$$[\text{Level setting of BRM}] = 2^{-n}\,[\text{Counter value}] \tag{9}$$

Finally, the scale of both sides of the defining equation, that is, equation (6), must be the same.

This procedure may be reduced to a finite number of steps. A synopsis of these steps is as follows:

Step 1. - Isolate the highest order derivative in the differential equation as shown in equation (6).

Step 2. - Assume the highest order derivative has been generated in a counter.

Step 3. - Generate each successive lower derivative by using an integrating unit.

Step 4. - Constrain the independent variable, the function, and its derivatives according to the right side of equation (6) and connect its output directly to the counter representing the highest order derivative.

Step 5. - Assign arbitrary constants to the independent variable and its highest order derivative.

Step 6. - Write constraint equations at each counter based on the maximum excursion and number of stages.

Step 7. - Write the constraint equation based on the defining equation.

Step 8. - Calculate scale factors to satisfy the equations of steps 6 and 7.

The application of this procedure is illustrated in the following sections in the design of specific countdown-countup machines. The first two examples will be machines for generating the exponential function and for generating the sine-cosine functions. These two machines will be illustrated in detail.

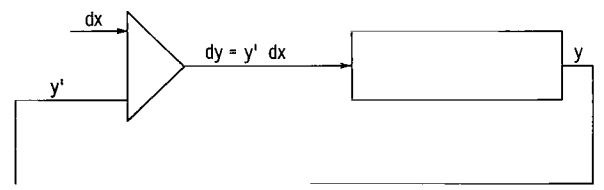
EXPONENTIAL FUNCTION

The differential equation

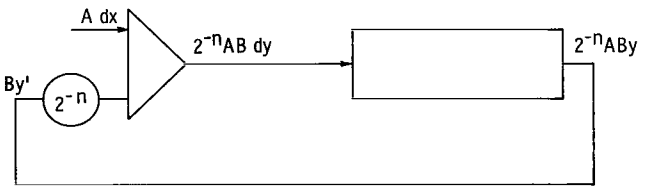
$$y' = y, \quad y(0) = 1 \tag{10}$$

whose solution is

$$y = e^x \tag{11}$$



(a) Basic design.



(b) Scaled design.

Figure 7. - Exponential function generator.

will be used to design the circuit for generating the exponential function. The design solution for this circuit is presented in figure 7. The detailed procedure for this synthesis is as follows:

Step 1. - The defining differential equation given by equation (10) is in the desired form.

Step 2. - Assume a circuit has been designed to generate the highest order derivative. This is represented by the line labeled y' in figure 7(a).

Step 3. - The function y is generated by integrating y'.

Step 4. - Since by equation (10) the assumed highest order derivative y' is equal to y, then y is directly connected to the line y'. This completes the basic circuit shown in figure 7(a).

Step 5. - This is the first step in the design of the scaled circuit shown in figure 7(b). The arbitrary constants A and B are assigned as scale factors to the independent variable and the highest order derivative, respectively. The interpretation of A is "A counts per unit of x." The interpretation of B is "B counts per unit of y'." Note that the scale factor of the counter in figure 7(b) is reduced by 2^{-n} when it is used to set the levels of the BRM. If the value of the counter is By' counts, the level setting of the BRM is $2^{-n}By'$.

Step 6. - Constraint equations are written for the counter; that is,

$$\left|2^{-n}ABy\right|_{max} \leq 2^{n} - 1$$

Step 7. - The mechanization of the defining equation is justified by the following constraint equation:

$$By' = 2^{-n}ABy$$

Step 8. - From the equation of step 7 it can be calculated that

$$A = 2^{n} \text{ counts/unit of } x$$

The calculation of B depends on the maximum excursion of the variable y according to the equation

$$B\left|y\right|_{max} \leq 2^{n} - 1$$

This essentially completes the schematic design of the circuit for generating e^{x}.

TABLE VI. - SCALE FACTORS AND INITIAL CONDITIONS OF EXPONENTIAL MACHINE

n	A	B	Initial counter value
5	32	10	10
6	64	20	20
7	128	40	40
8	256	80	80

Some choices for A and B based on the equation of step 8 are given in table VI. This series of machines has been simulated on a computer in order to illustrate some typical results that can be obtained by machines of this type. These results are presented in figure 8. The difference equation for the exponential machine may be obtained directly from figure 9, which is identical to the circuit shown in figure 7 but labeled according to Euler's integration. The scale factor associated with the counter is "B pulses per unit of y." When the counter is at iterative step $k-1$, its

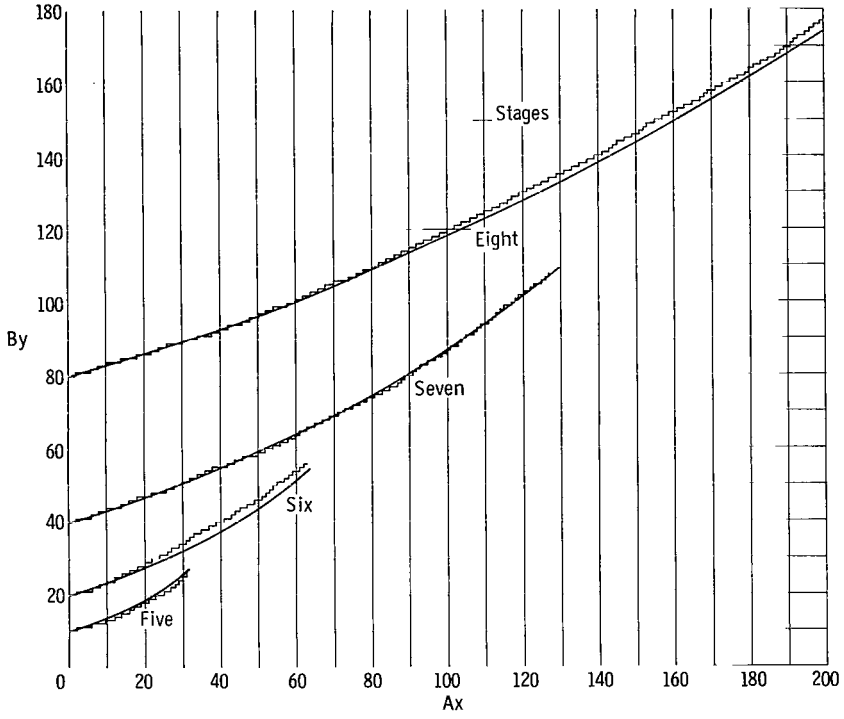


Figure 8. - Exponential machine output.

value is at $y_{(k-1)}$. During this iteration, $2^n \Delta x$ pulses arrive at the BRM counter, and the BRM puts out $By_{(k-1)} \Delta x$ pulses. These output pulses are added to the counter to form the counter value for iterative step k. Mathematically the value of the counter may be expressed by the difference equation

$$By_{(k)} = By_{(k-1)} + By_{(k-1)} \Delta x \tag{12}$$

If each clock pulse is taken as an iterative step, $\Delta x = 2^{-n}$ and equation (12) may be rewritten as

$$y_{(k)} = (1 + 2^{-n})\, y_{(k-1)} \tag{13}$$

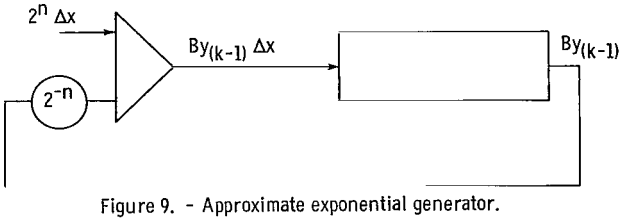


Figure 9. - Approximate exponential generator.

Solving equation (13) in terms of the initial conditions of y (i.e., y(0) = 1) yields

$$y_{(k)} = (1 + 2^{-n})^k \tag{14}$$

This solution is shown in figure 8.

The difference between the differential-equation solution and the difference-equation solution is the truncation error of the pro-

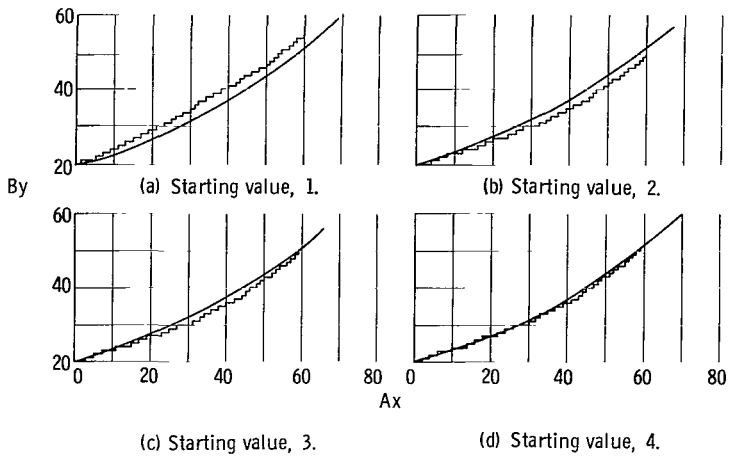


Figure 10. - Output of six-stage exponential machine.

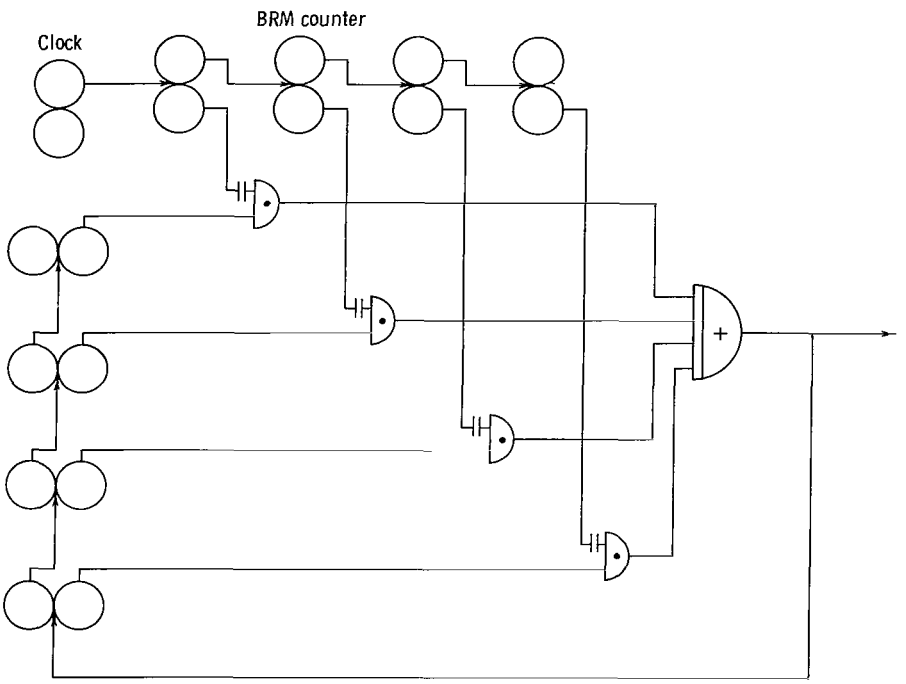
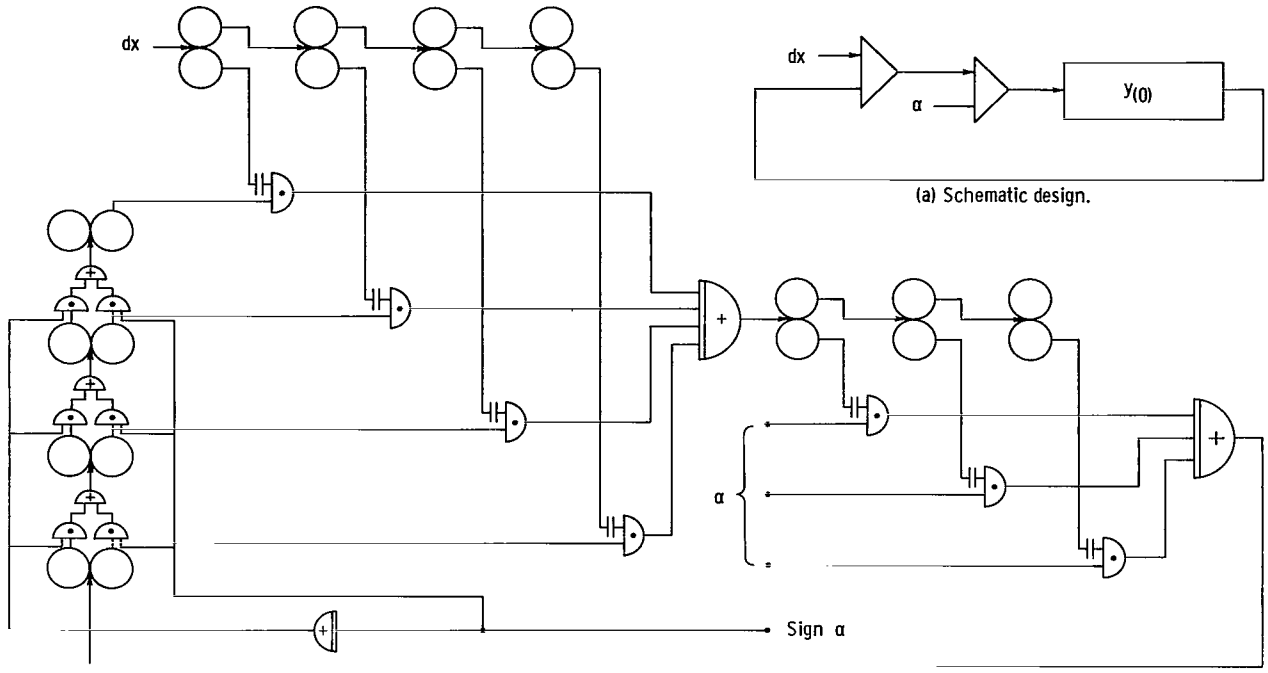


Figure 11. - Logic diagram of exponential machine.

cess. The difference between the actual output and the difference-equation solution is the error due to round off. The results shown in figure 8 illustrate that the round-off error is dependent on the number of stages of the BRM. Figure 10 shows this error for the six-stage BRM for a number of different starting values of the BRM counter. This figure illustrates that the round-off error is dependent on the starting value of the BRM counter. By these simple means, measurable improvement can usually be attained in the total error.

Finally, consider a configuration of this machine built out of standard logic elements (see ref. 5) for generating the function. This is shown for a four-stage system in figure 11.



(a) Schematic design.

(b) Logical design.

Figure 12. - General exponential function machine.

Since the exponential function is a monotonically increasing function, the counter shown in this circuit is a simple forward counter.

The synthesis and analysis of a countup-countdown machine for the generation of the general exponential function

$$y = y_0 e^{\alpha x} \tag{15}$$

from the differential equation

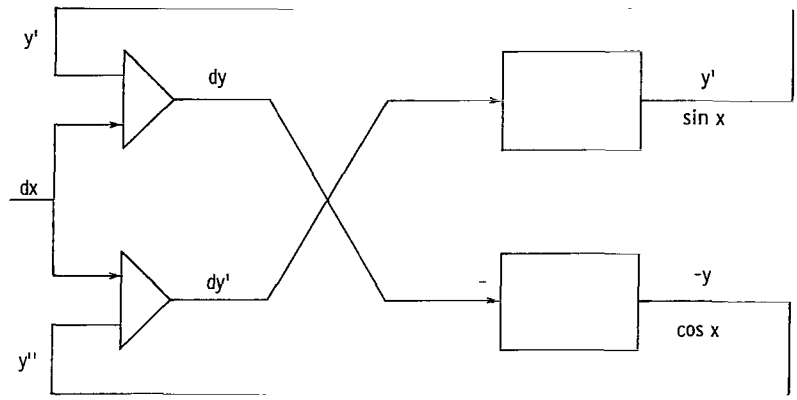
$$y' = \alpha y, \quad y(0) = y_0 \tag{16}$$

follows with only minor modification the design of the machine for generating e^x. The schematic diagram for this machine is given in figure 12(a), and the logical design is given in figure 12(b).

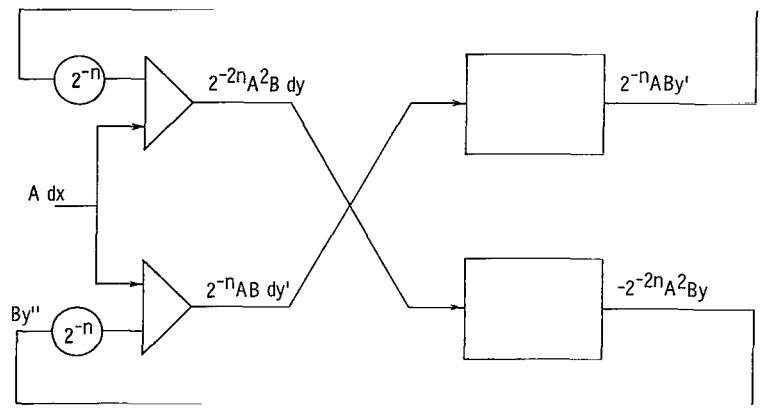
SINE-COSINE GENERATOR

The differential equation

$$y'' = -y, \quad y'(0) = 0, \quad y(0) = 1 \tag{17}$$



(a) Basic design.



(b) Scaled design.

Figure 13. - Sine-cosine generator.

is used to design the sine-cosine generator. The basic schematic diagram and the scaled schematic diagram are shown in figures 13(a) and (b), respectively, and may be developed systematically as follows:

Step 1. - The defining differential equation given by equation (17) is in the desired form.

Step 2. - Assume that a circuit has been designed to generate the highest order derivative. This is represented by the line labeled y'' in figure 13(a).

Step 3. - The function y' is generated by integrating y'', and the function y is generated by integrating y'.

Step 4. - Since y enters the differential equation negatively, the pulses arriving at the y counter have a sign change with the result that the output of the counter is -y corresponding to y'' (see defining equation).

Step 5. - Arbitrary constants A and B are assigned as scale factors to the independent variable and the highest order derivative, respectively. The interpretation of A is "A counts per unit of x" (i.e., per radian). The interpretation of B is "B counts

TABLE VII. - SCALE FACTORS AND INITIAL CONDITIONS FOR SINE-COSINE MACHINE

n	A	B	Counter	
			Sine	Cosine
3	8	4	0	-4
4	16	8	0	-8
5	32	16	0	-16
6	64	32	0	-32

per unit of y." If y' counter is set to 0 and y counter is set to -1 (note that this corresponds to the cosine being +1), y' will countup to generate the sine and y will countdown to generate the cosine.

Step 6. - Constraint equations are written for each counter; that is,

$$2^{-n}AB\left|y'\right|_{max} \leq 2^{n} - 1$$

$$2^{-2n}A^{2}B\left|y\right|_{max} \leq 2^{n} - 1$$

Step 7. - The constraint equation is written to justify the defining equation

$$By'' = -2^{-2n}A^{2}By$$

Step 8. - From the equations in steps 6 and 7, the scale factors can be chosen:

$$A = 2^{n},\ B\left|y\right|_{max} < 2^{n} - 1$$

Some choices for A and B are given in table VII based on the equation of step 8. This series of machines has been simulated on a computer, and the results plotted in figure 14.

The truncation error associated with this circuit may be calculated by solving the difference equations associated with this circuit. Calling the values of the sine and cosine counters at the k iterative step "$BS_{(k)}$" and "$BC_{(k)}$", respectively, gives the difference equations at these two counters:[1]

$$BS_{(k)} = BS_{(k-1)} + BC_{(k-1)}\,\Delta x \tag{18}$$

$$BC_{(k)} = BC_{(k-1)} - BS_{(k-1)}\,\Delta x \tag{19}$$

[1]Eqs. (18) and (19) may be expressed more concisely as

$$\frac{1}{(\Delta x)^{2}}\,\nabla^{2}S_{k} = -S_{k}$$

where $\nabla^{2}S_{k}$ is the second backward difference. Therefore, the second derivative in this machine is approximated by the second backward difference.

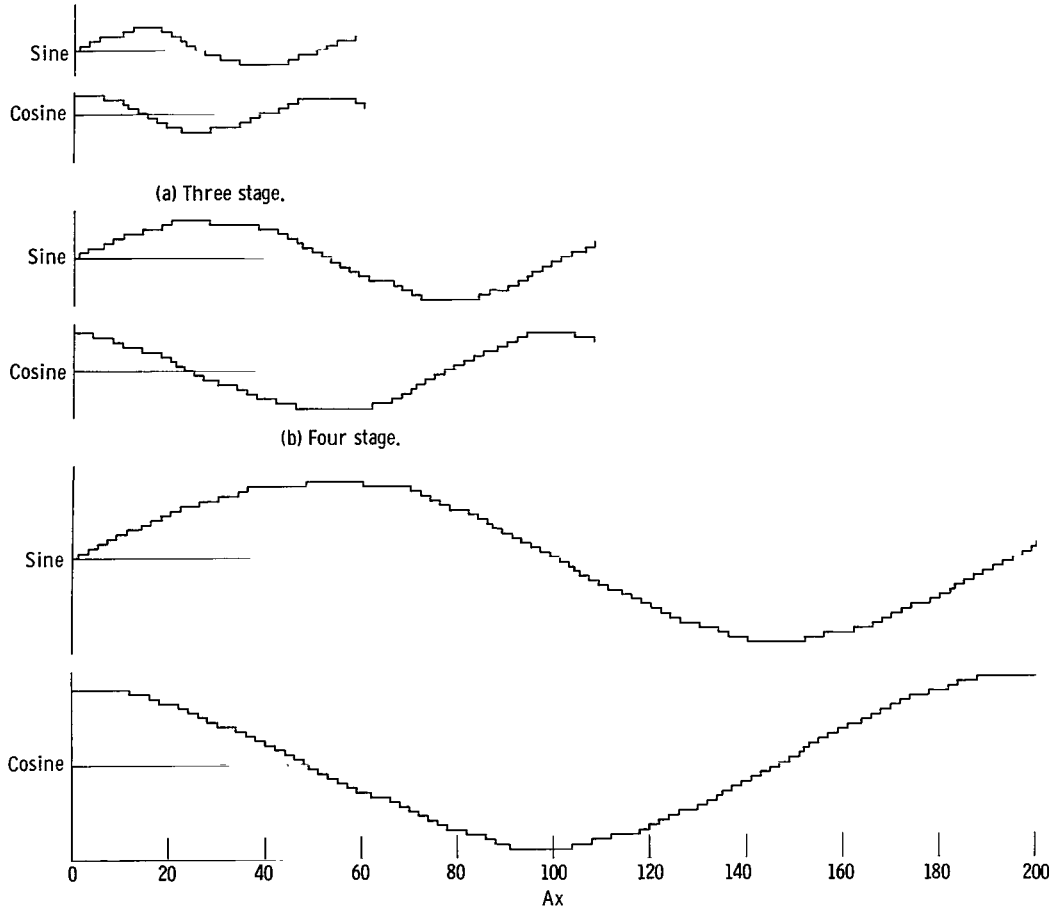


Figure 14. - Output of sine-cosine generator.

If each clock pulse is taken as an iterative step, equations (18) and (19) may be written in matrix form as

$$\begin{pmatrix} S_{(k)} \\ C_{(k)} \end{pmatrix} = \begin{pmatrix} 1 & 2^{-n} \\ -2^{-n} & 1 \end{pmatrix} \begin{pmatrix} S_{(k-1)} \\ C_{(k-1)} \end{pmatrix} \tag{20}$$

An approximate solution of equation (20) for large n in terms of the initial conditions

$$\begin{pmatrix} S_{(0)} \\ C_{(0)} \end{pmatrix} = \begin{pmatrix} 0 \\ 1 \end{pmatrix} \tag{21}$$

may be written as

$$S_{(k)} = (1 + k2^{-2n-1}) \sin 2^{-n}k \tag{22}$$

$$C_{(k)} = (1 + k2^{-2n-1}) \cos 2^{-n}k \tag{23}$$

A quantitative evaluation of this circuit is complicated by the fact that it is used to generate two functions. One method that seems especially well suited for testing such a circuit is to plot one output function with respect to the other function rather than with respect to the independent variable. For the sine-cosine generator this is called the "circle test" since the resultant figure for a perfect sine-cosine generator would be a circle. Moreover, it is possible to study the errors due to round-off independent of those due to truncation by comparing the actual output to equation (20) output. For the sine-cosine generator a composite plot of the solution to the difference equation may be simply obtained by expressing equations (22) and (23) in polar coordinates, with the result that

$$\rho = 1 + 2^{-n-1}\theta \tag{24}$$

The difference between this equation and the circle represents the truncation error of the process and is seen to increase as the spiral of Archimedes. The results plotted in figure 14 are compared with the plot of equation (24) in figure 15 by this method.

OTHER DIFFERENTIAL EQUATION MACHINES

The hyperbolic-sine - hyperbolic-cosine machine is based on the differential equation

$$y'' = +y \tag{25}$$

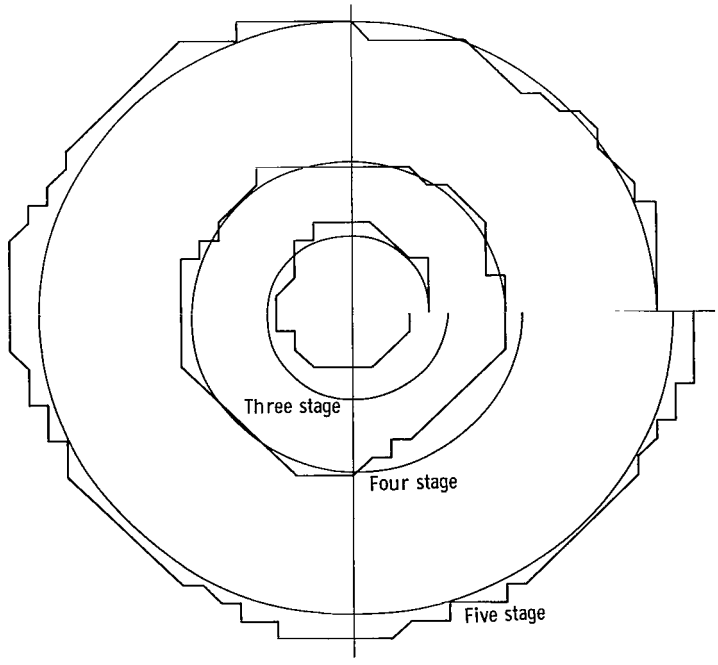
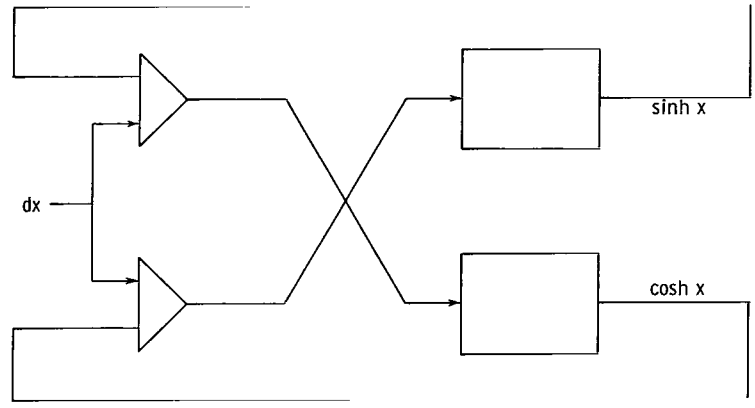
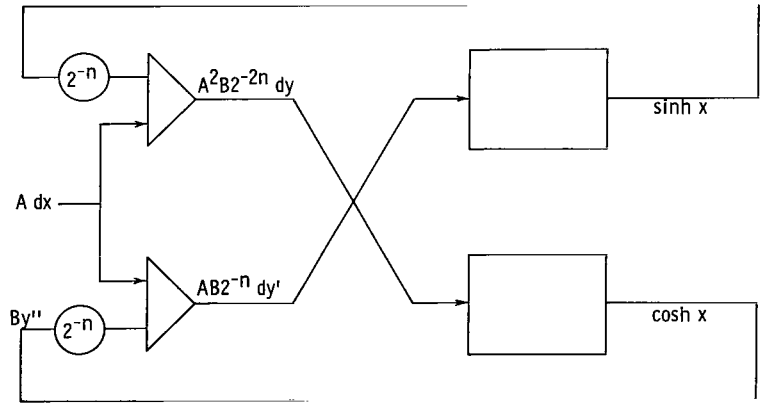


Figure 15. - Sine-cosine circle test.



(a) Basic design.



(b) Scaled design.

Figure 16. - Hyperbolic-sine - hyperbolic-cosine generator.

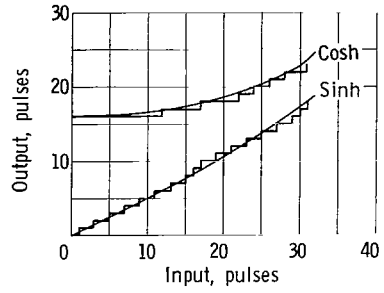


Figure 17. - Output of five-stage hyperbolic-sine - hyperbolic-cosine machine.

```
p-Sequence  Hyperbolic sine x
10000
01000    00000000000000000000000000000011111
10000    00000000000000001111111111111100000
00100    00000001111111100000011111111100000
10000    00011110000111100111100001111001111
01000    01100110011001101001100110011010001
10000    Output of hyperbolic sine
00010    00000000000100000100010101010010100
10000
01000
10000
00100
10000    Hyperbolic cosine x
01000
10000    11111111111111111111111111111111111
00001    00000000000000000000000000000000001
10000    00000000000000000000000011111111110
01000    00000000000000000001111110000111110
10000    00000000000011111110000110011000110
00100    Output of hyperbolic cosine
10000    10101010101010111010101010111010
01000
10000
00010
10000
01000
10000
00100
10000
01000
10000
00000
```

Figure 18. - p-Sequence calculations for hyperbolic-sine - hyperbolic-cosine machine.

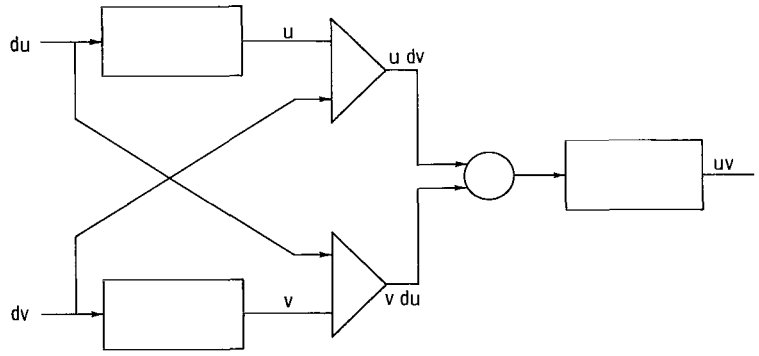


Figure 19. - Product machine.

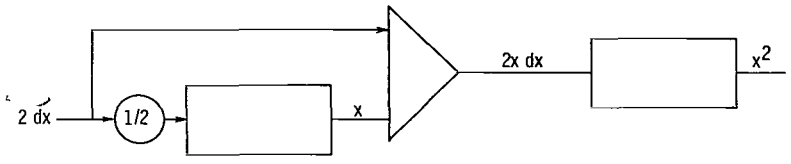


Figure 20. - Square machine.

The schematic diagram for this machine is similar to the sine-cosine generator except that all outputs from the BRM's are added to the counters. The basic circuit and the scaled circuit for this machine are shown in figure 16. It will be noted from this diagram that the values of A and B are defined by the equation set (26) as follows:

$$\left.\begin{aligned} By'' &= 2^{-2n}A^2By \\ A &= 2^n \\ B|y_{max}| &\le 2^n - 1 \\ B|y'_{max}| &\le 2^n - 1 \end{aligned}\right\} \qquad (26)$$

The output for a hyperbolic-sine - hyperbolic-cosine generator is plotted in figure 17 for a five-stage system where B is chosen equal to 16. It is instructive to display the p-sequence calculation from which these results were obtained. These are shown in figure 18.

A series of other useful machines will be illustrated in this section. In particular, if two pulse streams du and dv are given, the product uv may be generated by using the equation

$$duv = u\,dv + v\,du \qquad (27)$$

The basic design for this product machine is shown in figure 19.

The machine for generating the square of a function is shown schematically in figure 20. This machine will generate the function

$$y = x^2 \qquad (28)$$

and is based on the equation

$$y' = 2x \qquad (29)$$

The square machine is utilized as a subassembly in the machine for generating the reciprocal of a function; that is,

$$y = 1/x \qquad (30)$$

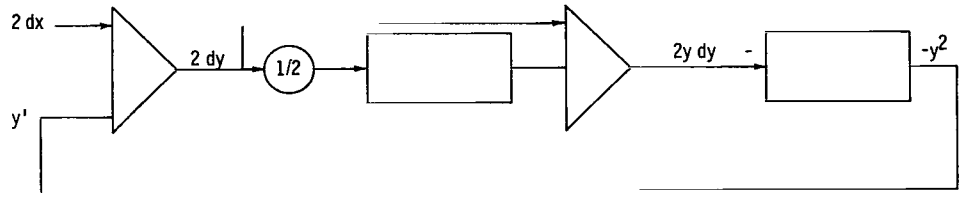


Figure 21. - Reciprocal machine.

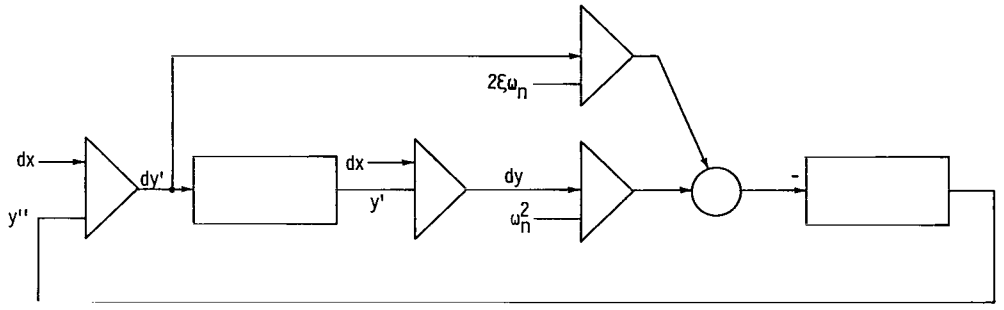


Figure 22. - Second-order differential-equation machine.

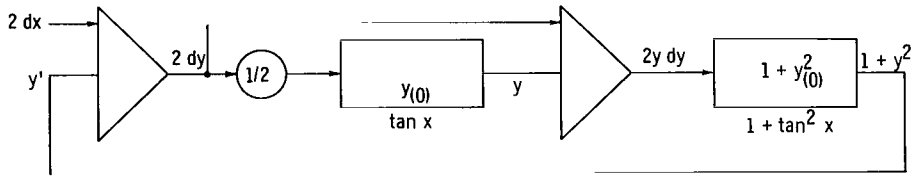


Figure 23. - Tangent machine.

The basic design for this reciprocal machine based on the differential equation

$$y' = -y^2 \tag{31}$$

is shown in figure 21.

The machine for generating the solution to the second-order differential

$$y'' + 2\xi\omega_n y' + \omega_n^2 \, y = 0 \tag{32}$$

is shown in figure 22.

The tangent machine is shown schematically in figure 23. This machine is based on the differential equation

$$y' = 1 + y^2 \tag{33}$$

A similarity will be noted between this machine and that of the reciprocal machine.

The square-root machine is based on the solution of the differential equation

$$y' = 1/2y \tag{34}$$

It will be noted by this equation that it will form a subassembly of the reciprocal machine; that is, the differential equation

$$dz/dy = -2z^2 \tag{35}$$

is used in order to form the function

$$z = 1/(2y) \tag{36}$$

The basic design of this machine is shown in figure 24.

SYNTHESIS (DIFFERENCE EQUATION)

Consider the iterative process of successive substitution in the functional equation

$$x_{(k+1)} = \varphi[x_{(k)}] \tag{37}$$

where $\varphi(x)$ is chosen such that the fixed points of $\varphi(x)$ (i.e., the points x_i where $x_i = \varphi(x_i)$) are the roots of $f(x) = 0$. One simple form of $\varphi(x)$ might be $x - f(x)$, which leads to the iterative process

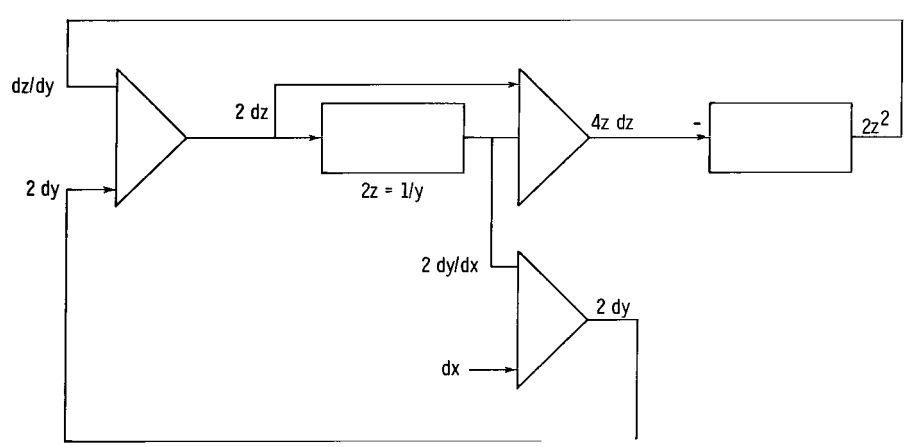


Figure 24. - Square-root machine.

$$x_{(k+1)} = x_{(k)} - f[x_{(k)}] \tag{38}$$

A more general form is $x - g(x)f(x)$ which leads to the iterative process

$$x_{(k+1)} = x_{(k)} - g[x_{(k)}]f[x_{(k)}] \tag{39}$$

A restriction on $g(x)$ in this latter form is that it has no zeros that are not zeros of $f(x)$ and that the multiplicity of its poles at the zeros of $f(x)$ be less than the multiplicity of the zeros of $f(x)$ at these points. With these restrictions, it can be readily seen that the iterative equation (eq. (39)) has fixed points at the zeros of $f(x)$ (i.e., $x = x - 0$). The function $g(x)$ in equation (39) is chosen so that the process converges.

The basic equation of a counter immediately suggests a method for generating an equation of the form of equation (39). This method of synthesis is simply to generate a pulse stream equal to $g(x)f(x)$ and feed it into a counter. This procedure may be outlined as follows:

Step 1. - Write the defining equation in the implicit form $g(x)f(x) = 0$.

Step 2. - Assume a counter with the value of x and generate a pulse stream equal to $g(x)f(x)p$, where $g(x)f(x)$ is the level setting of a BRM and p is the input to the BRM counter for converting this level setting into a pulse stream.

Step 3. - Feed back the pulse stream generated in step 2 into the x counter.

Step 4. - Assign an arbitrary constant to each variable represented in the machine.

Step 5. - Write constraint equations and calculate the scale factors such that these equations are satisfied.

DIVIDE ALGORITHM

The machine for generating x such that

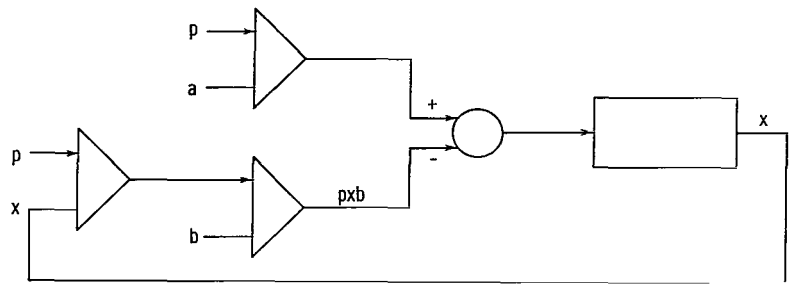
$$x = a/b \tag{40}$$

may be designed as follows:

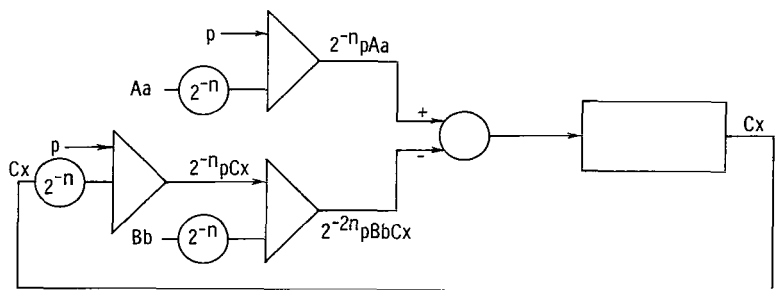
Step 1. - One way in which equation (40) may be rewritten to put it into the desired form is

$$xb - a = 0 \tag{41}$$

Step 2. - By assuming a counter value representing x, a pulse stream equal to $xb - a$ may be generated (see fig. 25(a)).



(a) Basic design.



(b) Scaled design.

Figure 25. - Divide algorithm.

Step 3. - The pulse stream generated in step 2 is fed into the counter representing x.

Step 4. - The schematic of figure 25(a) is redrawn in figure 25(b), and each variable is assigned an arbitrary constant.

Step 5. - The constraint equations may be written directly from figure 25(b), in which an iteration is taken at every clock pulse:

$$\left.\begin{aligned} C\left|x_{max}\right| &\le 2^n - 1 \\ A\left|a_{max}\right| &\le 2^n - 1 \\ B\left|b_{max}\right| &\le 2^n - 1 \end{aligned}\right\} \tag{42}$$

$$Cx_{(k+1)} = Cx_{(k)} - 2^{-2n}BbCx_{(k)} + 2^{-n}Aa \tag{43}$$

Suppose for the sake of argument that

$$A = B = C = 2^n \tag{44}$$

Equation (44) implies that

$$\left.\begin{aligned} x_{max} &\leq 1 - 2^{-n} \\ a_{max} &\leq 1 - 2^{-n} \\ b_{max} &\leq 1 - 2^{-n} \end{aligned}\right\} \tag{45}$$

and equation (43) may be rewritten as

$$x_{(k+1)} = x_{(k)} - 2^{-n}bx_{(k)} + 2^{-n}a \tag{46}$$

If equation (46) converges to a fixed point x,

$$x = x - 2^{-n}bx + 2^{-n}a \tag{47}$$

If $\mathcal{E}_{(k)}$ is the iterative truncation error at iterative step k, that is,

$$\mathcal{E}_{(k)} = x - x_{(k)} \tag{48}$$

then from equations (46) and (47)

$$\mathcal{E}_{(k+1)} = (1 - 2^{-n}b)\mathcal{E}_{(k)} \tag{49}$$

This may be written in terms of $\mathcal{E}_{(0)}$ as

$$\mathcal{E}_{(k)} = (1 - 2^{-n}b)^k\mathcal{E}_{(0)} \tag{50}$$

This process will converge if

$$\lim_{k\to\infty} \mathcal{E}_{(k)} = 0 \tag{51}$$

which implies the condition

$$\left|1 - 2^{-n}b\right| < 1 \tag{52}$$

for convergence. Therefore, by equation (52) the process is seen to converge. If the sign of the outputs of the BRM are reversed, however, such as shown in figure 26, an analysis will show that the process will not converge.

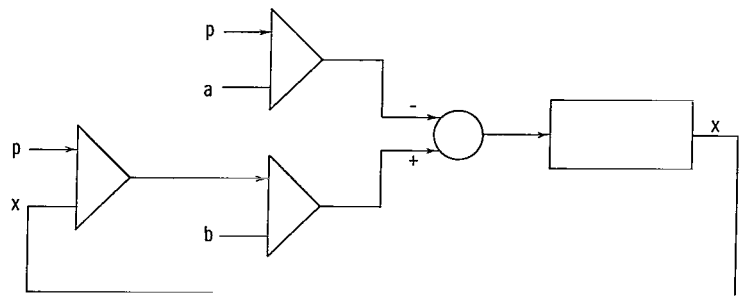


Figure 26. - Nonconvergent divide algorithm.

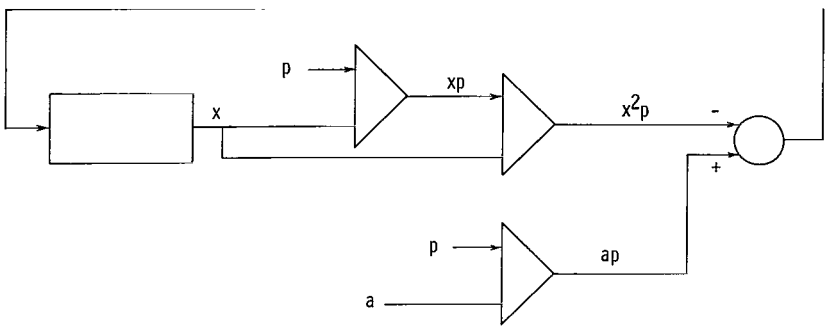


Figure 27. - Iterative-process square-root machine.

OTHER DIFFERENCE-EQUATION MACHINES

The square-root machine, that is,

$$x = \sqrt{a} \tag{53}$$

may be designed by finding the zeros of the equation

$$x^2 - a = 0 \tag{54}$$

This machine is shown schematically in figure 27. If the scale factor of x and a are both taken as 2^n, an analysis similar to that of the divide algorithm shows that the iterative process

$$x_{(k+1)} = x_{(k)} - 2^{-n}x_{(k)}^2 + 2^{-n}a \tag{55}$$

is generated by the machine. The iteration truncation error for this equation may be written as

$$\mathcal{E}_{(k+1)} = \left\{1 - 2^{-n}[x + x_{(k)}]\right\}\mathcal{E}_{(k)} \tag{56}$$

where x is the solution. A sufficient condition for this process to converge is that

$$\left|1 - 2^{-n}[x + x_{(k)}]\right| < 1 \tag{57}$$

Since

$$x_{max} < 1 \tag{58}$$

with the scale factor chosen, the process converges; however, had

$$x_{(k+1)} = x_{(k)} - 2^{-n}[a - x^2_{(k)}] \tag{59}$$

been chosen as the iterative process, the process would not converge.

A product machine may also be designed by using this method of synthesis. In particular, if

$$x = ab \tag{60}$$

the product may be found by the iterative process

$$x_{(k+1)} = x_{(k)} - p[x_{(k)} - ab] \tag{61}$$

This machine is shown in figure 28. If the scale factors for x, a, and b are all taken as 2^n, the machine generates the iterative process

$$x_{(k+1)} = x_{(k)} - 2^{-n}[x_{(k)} - ab] \tag{62}$$

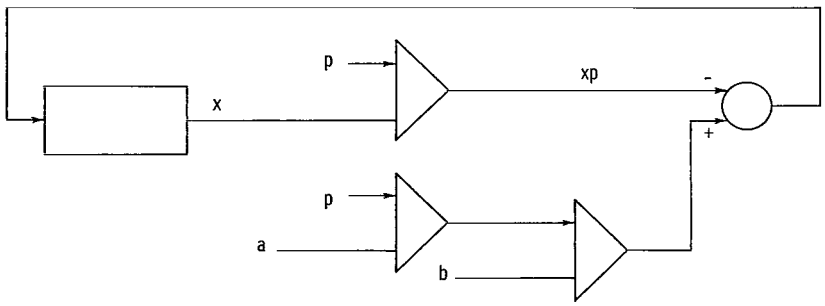


Figure 28. - Iterative-process product machine.

The iterative truncation error for this machine in terms of $\mathcal{E}_{(0)}$ is

$$\mathcal{E}_{(k)} = (1 - 2^{-n})^k \mathcal{E}_{(0)} \tag{63}$$

SYNTHESIS (REGENERATIVE CIRCUIT)

Consider the schematic diagram shown in figure 29. The value of K is bounded such that

$$|K| \leq 1 - 2^{-n} \tag{64}$$

The output equation for this circuit may be written as

$$dz = K(dx + dz)$$

$$\frac{dz}{dx} = \frac{K}{1 - K} \tag{65}$$

As $K \to 1$ in equation (65), the ratio $dz/dx \to \infty$; however, equation (64) fixes an upper bound on this ratio such that

$$\frac{K}{1 - K} = 2^n - 1 \tag{66}$$

Therefore, by using this regenerative circuit, the BRM may act as an amplifier. If large amplifications are to be considered, however, other factors must be taken into account. In particular, suppose a dx pulse arrives at the BRM and this generates a dz pulse. The dz pulse is delayed by a gated pulse generator and is fed back to the BRM. This in turn may generate another dz pulse. This process may be continued depending on the value of K; however, each time a dz pulse is recirculated, the pulse shape deteriorates. For example, if leading edge logic is used, the rise time of each pulse will be increased until the dz pulse is not sharp enough to be utilized. Moreover, enough time must be allowed between the dx pulses to permit the maximum number of dz pulses. The maximum value of K usually utilized in these circuits will in general be less than that permitted by equation (64).

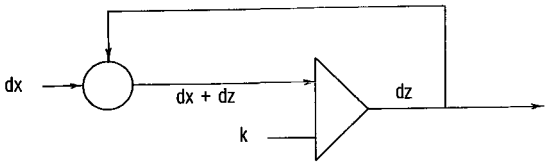


Figure 29. - Regenerative circuit.

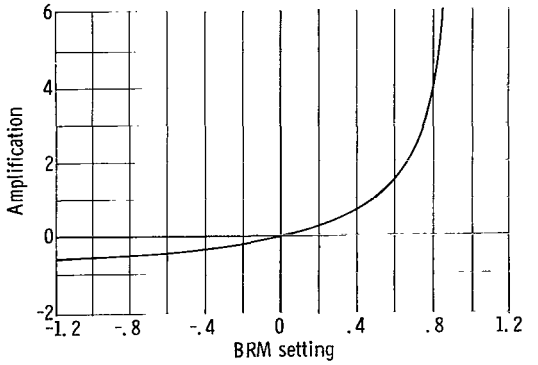


Figure 30. - Amplification of regeneration circuit.

This point is illustrated by the plot of equation (65) in figure 30. If $-1 < K \leq 1/2$, at most, 1 pulse will be fed back with each input pulse. If $1/2 < K < 1$, more than 1 pulse will be fed back with each input pulse. Therefore, by fixing the upper value of K, the maximum number of feedback pulses may be restricted.

The method of synthesis in this section is similar to that used in the synthesis by differential equation. In this section, however, the differential equation will involve the highest order derivative on both sides of the equation; that is,

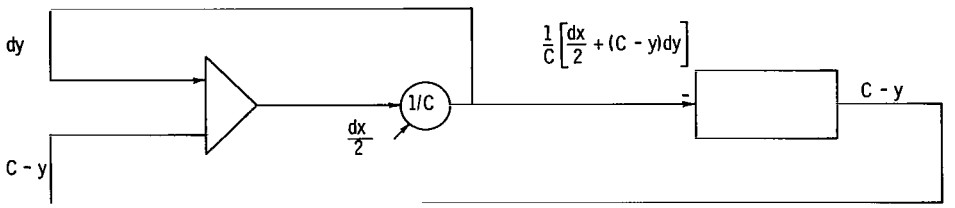
$$\frac{d^m y}{dx^m} = f\left(\frac{d^m y}{dx^m}, \ldots, y, x\right) \tag{67}$$

In general, the equation is written in this form when the highest order derivative cannot be isolated. In particular, if the highest order derivative has a nonconstant coefficient, it may be written as equation (67) by adding and subtracting a constant from this coefficient (see refs. 8 and 9). The design of the circuit based on equation (67) implies the use of the regenerative circuit, since the generation of the highest order derivative involves itself.

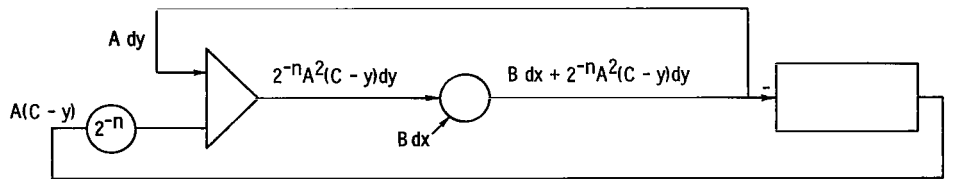
SQUARE-ROOT MACHINE

Consider the generation of the square root

$$y = \sqrt{x} \tag{68}$$



(a) Basic design.



(b) Scaled design.

Figure 31. - Regenerative-circuit square-root machine.

from the equation

$$y \frac{dy}{dx} = 1/2 \tag{69}$$

The first step of this technique is to write equation (69) as

$$(y - C + C) \frac{dy}{dx} = 1/2 \tag{70}$$

and then isolate the highest order derivative as shown in the following equation:

$$dy = \frac{1}{C}\left[\frac{dx}{2} + (C - y)dy\right] \tag{71}$$

The basic and scaled circuit for generating equation (71) is shown in figures 31(a) and (b), respectively. Following the same technique used to derive a machine based on a differential equation gives the constraint equations:

$$A\, dy = B\, dx + 2^{-n}A^2(C - y)dy \tag{72}$$

(defining equation for the machine)

$$A\left|C - y\right|_{max} \leq 2^n - 1 \tag{73}$$

(the counter limiting equation). From these equations, the scale factors may be computed by

$$\left.\begin{aligned} A &= 2^n/C \\ B &= 2^{n-1}/C^2 \end{aligned}\right\} \tag{74}$$

where C is chosen such that it satisfies the inequality

$$\left|1 - \frac{y}{C}\right|_{max} \leq 1 - 2^{-n} \tag{75}$$

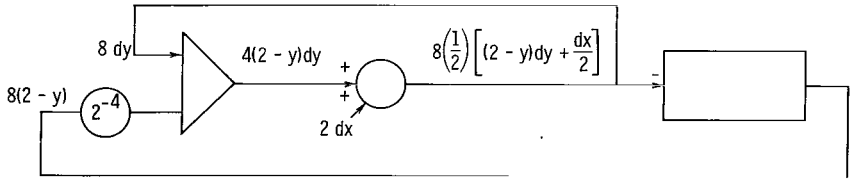


Figure 32. - Scaled diagram for four-stage square-root machine.

Based on this design, figure 32 gives the computed scales and circuit for a four-stage regenerative-circuit square-root machine. The output of this machine together with that of the desired output is shown in figure 33.

OTHER REGENERATIVE CIRCUIT MACHINES

The natural-logarithm machine

$$y = \ln x \tag{76}$$

may be designed as a regenerative circuit by the equation

$$dy = \frac{dx}{C} + \left(1 - \frac{x}{C}\right)dy \tag{77}$$

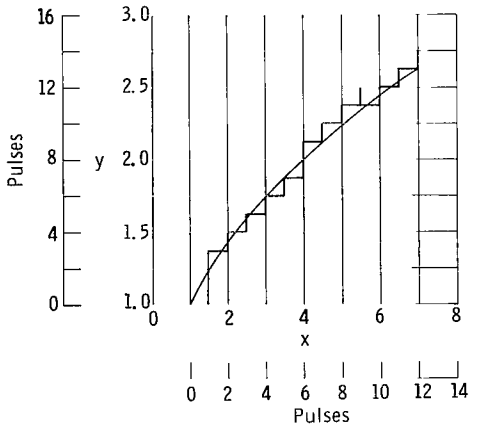


Figure 33. - Output of regenerative-circuit square-root machine.

The circuit for generating this function is shown in figure 34.

The quotient machine

$$z = x/y \tag{78}$$

may be designed as a regenerative circuit by the equation

$$dz = \frac{1}{C}[(y + C)dz + z\,dy - dx] \tag{79}$$

The schematic circuit for this function is shown in figure 35.

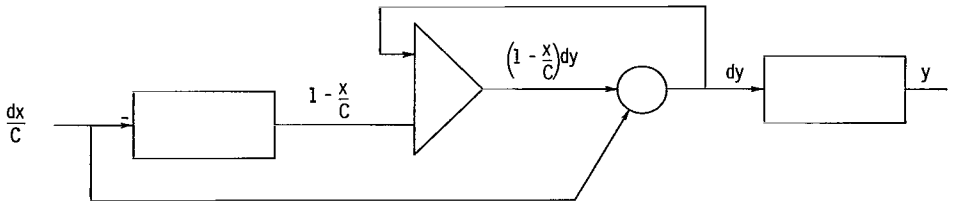


Figure 34. - Regenerative-circuit logarithm machine.

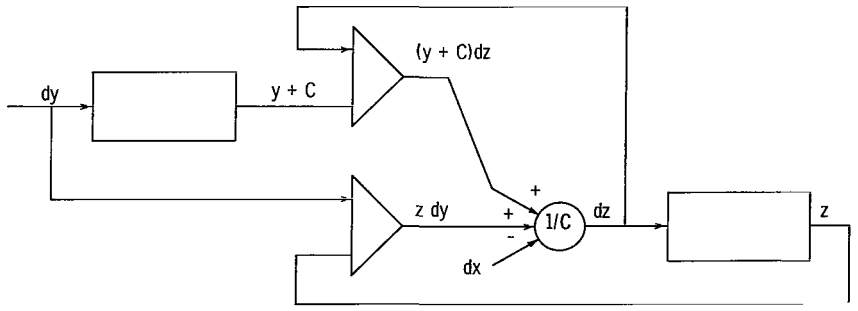


Figure 35. - Regenerative-circuit quotient machine.

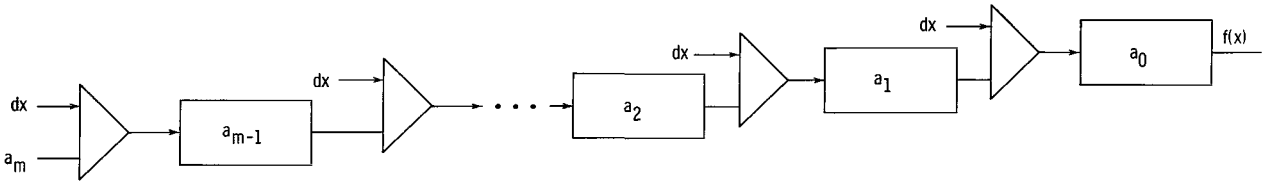


Figure 36. - General polynomial machine.

PIECEWISE POLYNOMIAL MACHINES

When a function to be generated is available only as empirical data (e.g., such as in sampled data systems), the design must be based on generating an approximation function. Various classes of approximation functions and techniques for obtaining approximations have received considerable attention in the literature (e.g., see ref. 10). A particularly convenient form for approximating a continuous function is that of a polynomial. The general polynomial

$$f(x) = \frac{a_m x^m}{m!} + \frac{a_{m-1}}{(m-1)!} x^{m-1} + \cdots + \frac{a_2}{2!} x^2 + a_1 x + a_0 \tag{80}$$

may be generated by the circuit shown in figure 36. However, it is usually the case that all the data are not immediately available for generating the function over its entire

range, or if it is available, the polynomial needed to meet the accuracy requirements is of excessively high degree. In these applications the requirements of the problem may be met by using a series of relatively low degree polynomials, where each polynomial is used to fit data only in a restricted range. Such machines are called piecewise polynomial machines.

Because of their widespread use in applications (see refs. 4, and 11 to 13) and also because they can be realized with relatively simple circuits, this section is devoted exclusively to describing a series of machines for generating piecewise polynomials arising from finite-difference techniques. These machines are grouped into two broad categories based on applications, that is, interpolation or extrapolation. Each machine of this series will generate a low order polynomial fitted to data available at equal intervals of the argument. In passing from one segment into the next new data are introduced. The form of the data in each case is simply generated from the empirical data.

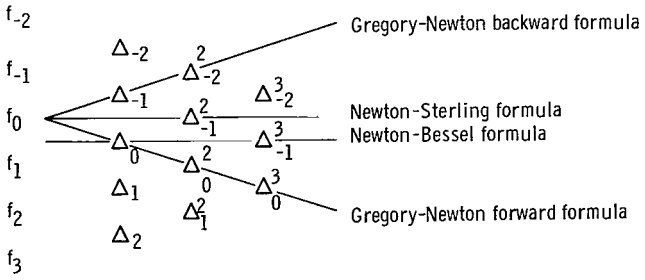
In order to facilitate the description of the machines in the next two sections, ordinary difference notation will be used. In particular, ω_n is the value of the independent variable, where the value of the function is obtained; that is,

$$f(\omega_n) = f_n \tag{81}$$

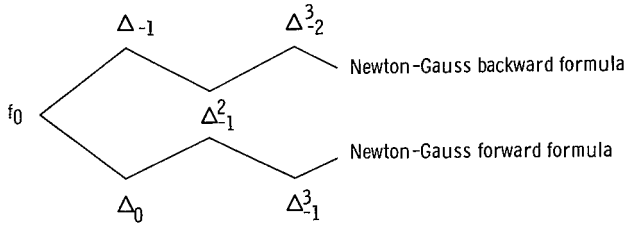
The quantity $\delta\omega$ represents the spacing of the independent variable, and $\Delta_n, \Delta_n^2, \ldots, \Delta_n^m$ are the successive differences that may be obtained from lower order differences as follows:

$$\left.\begin{aligned} \Delta_n &= f_{n+1} - f_n \\ \Delta_n^2 &= \Delta_{n+1} - \Delta_n \\ &\vdots \\ \Delta_n^m &= \Delta_{n+1}^{m-1} - \Delta_n^{m-1} \end{aligned}\right\} \tag{82}$$

In formulating the approximation formula that passes through the given points, it is convenient to display these various differences in tabular form as shown in figure 37(a). From this difference table, alternate forms of the approximation formula may be derived dependent on the differences that are utilized, that is, on the path through the difference table. The paths of some of these formulas are shown in figure 37(b). The form of the formula that will be utilized in the following discussion will be such that, in each case, the value of the variable x will vary from 0 to 1 in the interval of interest.



(a) Direct-path difference formulas.



(b) Broken-path difference formulas.

Figure 37. - Difference table and paths of difference formulas.

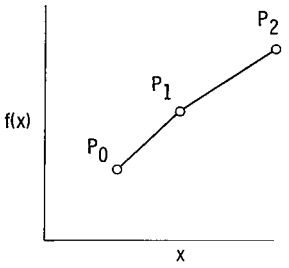


Figure 38. - Scheme for piecewise linear interpolation.

INTERPOLATION

A piecewise linear interpolator can be obtained by passing a linear polynomial through successive pairs of points of the function to be generated. This scheme is illustrated in figure 38. A linear polynomial is generated that passes through the points P_0 and P_1. At the point P_1, the point P_2 is added to the scheme and a linear polynomial is generated that passes through the points P_1 and P_2. This procedure may be expressed in terms of ordinary differences by the Gregory-Newton interpolation formula; that is,

$$f(\omega_n + x\delta\omega) = f_n + \Delta_n x \tag{83}$$

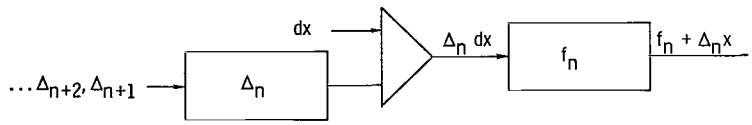
The first derivative of this formula is

$$\delta\omega f'(\omega_n + x\delta\omega) = \Delta_n \tag{84}$$

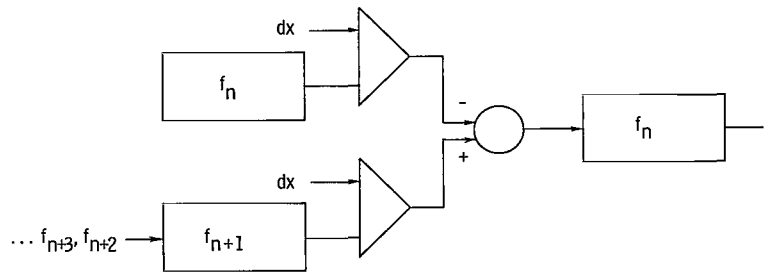
This interpolation formula may be generated by the linear polynomial generator shown in figure 39(a).

The corresponding equation and its first derivative for the next interpolation interval is given by

$$f(\omega_{n+1} + x\delta\omega) = f_{n+1} + \Delta_{n+1} x \tag{85}$$



(a) First difference input data.



(b) Function values input data.

Figure 39. - Machines for piecewise linear interpolation.

$$\delta\omega f'(\omega_{n+1} + x\delta\omega) = \Delta_{n+1} \tag{86}$$

At the end of the first interval the values of the function and its derivative given by equations (83) and (84) are

$$f(\omega_n + \delta\omega) = f_n + \Delta_n = f_{n+1} \tag{87}$$

$$\delta\omega f'(\omega_n + \delta\omega) = \Delta_n \tag{88}$$

The corresponding values of these two quantities needed at the start of the next interval are given by equations (85) and (86) and are

$$f(\omega_{n+1}) = f_{n+1} \tag{89}$$

$$\delta\omega f'(\omega_{n+1}) = \Delta_{n+1} \tag{90}$$

By direct computation it may be verified that, in order to proceed from one interval to the next, the quantity Δ_n^2 (which is $\Delta_{n+1} - \Delta_n$) needs to be added to the setting of the BRM, and the output (i. e., the end point of the interpolation interval) need not be modified. Since adders have been excluded as basic design elements, however, the same result may be attained by transferring Δ_{n+1} as the setting of the BRM (since $\Delta_{n+1} = \Delta_n + \Delta_n^2$). Consequently, the circuit shown in figure 39(a) may be used for piecewise linear interpolation of a function by transferring successive first difference as settings for the BRM in order to proceed from one interval to the next.

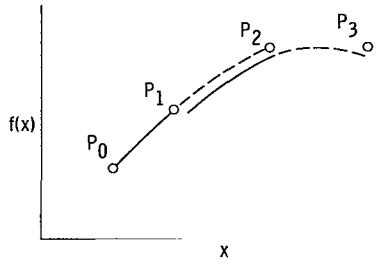


Figure 40. - Scheme for piecewise quadratic interpolation (back interval).

The circuit just derived is well suited for an application in which an incremental encoder is used to generate the input data. If an absolute encoder is used to generate the primary data, the preceding circuit may be adapted for this input by using the defining equation for first differences, that is, equation (82). This circuit is shown in figure 39(b). As in the previous case, only one new piece of information must be transferred into the circuit in order to proceed from one interpolation interval to the next. In this case, however, before the new information, that is, f_{n+2}, is transferred as the setting of the lower BRM, the value of the lower BRM, that is, f_{n+1}, must be transferred to the upper one.

Two piecewise quadratic interpolators will be derived. The scheme for the first one, which will be called the back interval quadratic interpolator, is illustrated in figure 40. A quadratic polynomial is generated that passes through points P_0, P_1, and P_2. This polynomial is used to generate the curve between points P_0 and P_1. The point P_3 is then added to the scheme and a quadratic polynomial is derived that passes through the points P_1, P_2, and P_3. This polynomial is then used to generate the curve between P_1 and P_2.

This procedure may also be conveniently expressed by the Gregory-Newton quadratic interpolation formula; that is,

$$f(\omega_n + x\delta\omega) = f_n + \Delta_n x + \frac{x(x - 1)}{2} \Delta_n^2 \tag{91}$$

The successive derivatives for this formula are

$$\delta\omega f'(\omega_n + x\delta\omega) = \left(\Delta_n - \frac{1}{2} \Delta_n^2\right) + \Delta_n^2 x \tag{92}$$

$$(\delta\omega)^2 f''(\omega_n + x\delta\omega) = \Delta_n^2 \tag{93}$$

The corresponding equation and its derivatives for the next interpolation interval are

$$f(\omega_{n+1} + x\delta\omega) = f_{n+1} + \left(\Delta_{n+1} - \frac{1}{2} \Delta_{n+1}^2\right)x + \frac{\Delta_{n+1}^2}{2} x^2 \tag{94}$$

$$\delta\omega f'(\omega_{n+1} + x\delta\omega) = \Delta_{n+1} - \frac{1}{2} \Delta_{n+1}^2 + \Delta_{n+1}^2 x \tag{95}$$

$$(\delta\omega)^2 f''(\omega_{n+1} + x\delta\omega) = \Delta_{n+1}^2 \tag{96}$$

Consequently, the corrections to be added to the second derivative, the first derivative, and the function at the end of the interval in order to proceed to the next interval are Δ_n^3, $-\frac{1}{2}\Delta_n^3$, and 0, respectively. From this formulation, however, addition is required in order to proceed from one interval into the next. A formulation of this process that leads to the elimination of the explicit adder is to splinter the polynomial given by equation (91) into the following two polynomials:

$$f_a(\omega_n + x\delta\omega) = f_n + \Delta_n x + \frac{\Delta_n^2}{2}x^2 \tag{97}$$

and

$$f_b(\omega_n + x\delta\omega) = -\frac{\Delta_n^2}{2}x \tag{98}$$

where

$$f(\omega_n + x\delta\omega) = f_a(\omega_n + x\delta\omega) + f_b(\omega_n + x\delta\omega) \tag{99}$$

The first and second derivatives of equation (97) are

$$(\delta\omega)f_a'(\omega_n + x\delta\omega) = \Delta_n + \Delta_n^2 x \tag{100}$$

and

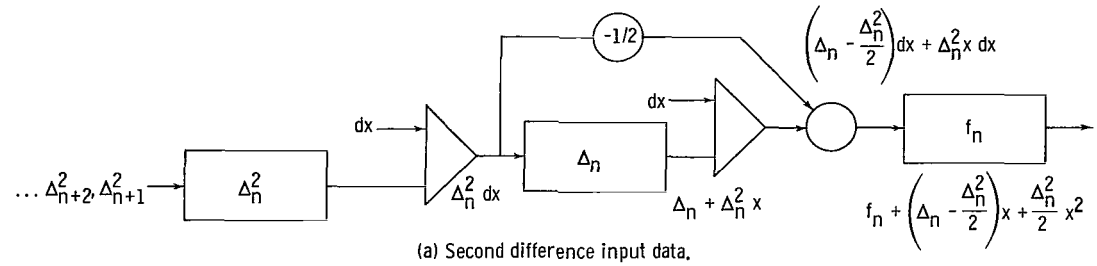
$$(\delta\omega)^2 f_a''(\omega_n + x\delta\omega) = \Delta_n^2 \tag{101}$$

The corresponding splintering of equation (94) yields

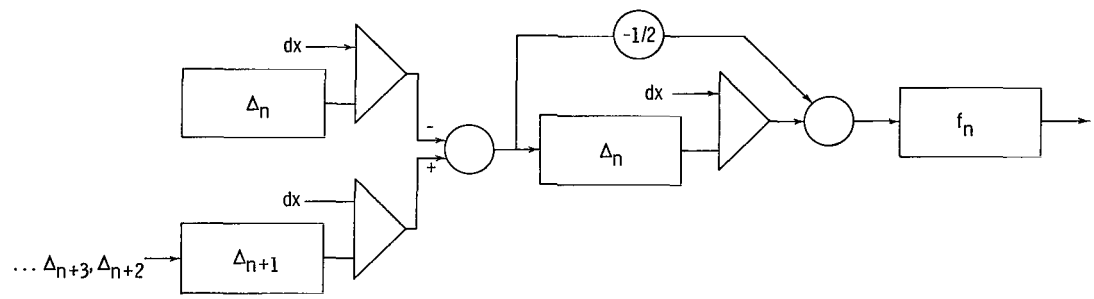
$$f_a(\omega_{n+1} + x\delta\omega) = f_{n+1} + \Delta_{n+1}x + \frac{\Delta_{n+1}^2}{2}x^2 \tag{102}$$

$$(\delta\omega)f_a'(\omega_{n+1} + x\delta\omega) = \Delta_{n+1} + \Delta_{n+1}^2 x \tag{103}$$

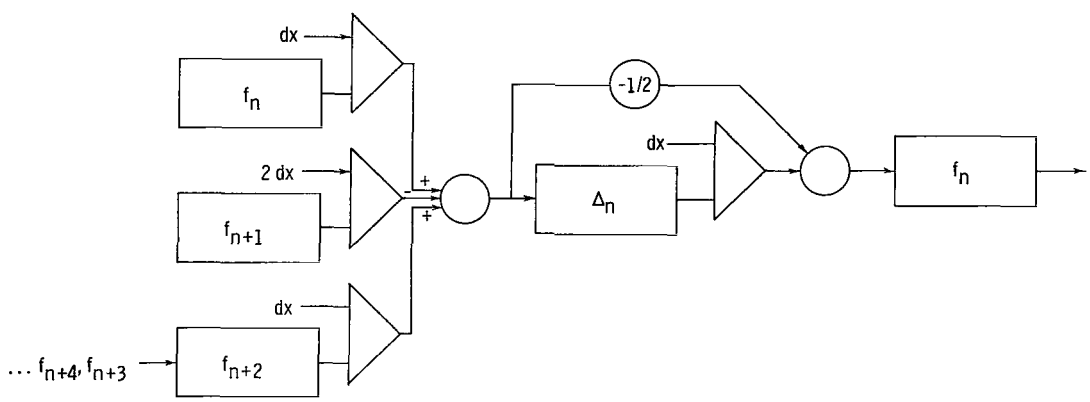
$$(\delta\omega)^2 f_a''(\omega_{n+1} + x\delta\omega) = \Delta_{n+1}^2 \tag{104}$$



(a) Second difference input data.



(b) First difference input data.



(c) Function values input data.

Figure 41. - Machines for piecewise quadratic interpolation (back interval).

Consequently, no correction needs to be added to the first derivative in generating the function f_a. Equations (97) and (98) may then be used to design the circuit shown in figure 41(a). It will be noted that, in order to proceed from one interval to the next, only new second difference data need be transferred to set the levels of the leftmost BRM.

Based on the defining equation for second differences, that is, equation (82), the machines shown in figures 41(b) and (c) may be obtained directly from the machine shown in figure 41(a). In these machines, previous first differences and values of the function are transferred directly from the lower BRM's to the upper ones before a new first difference and a value of the function, respectively, are transferred into the lower one in order to proceed from one interval into the next.

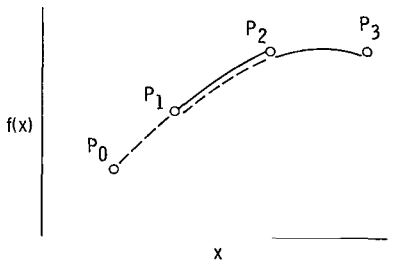
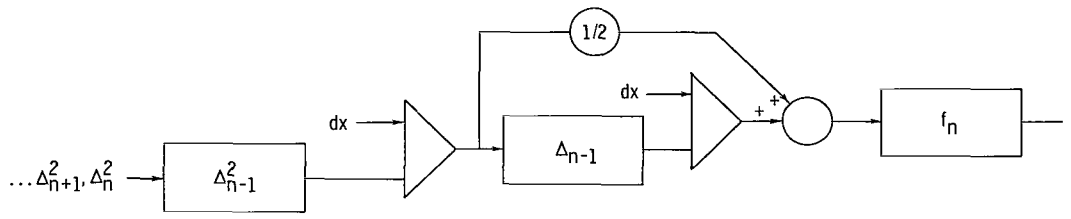
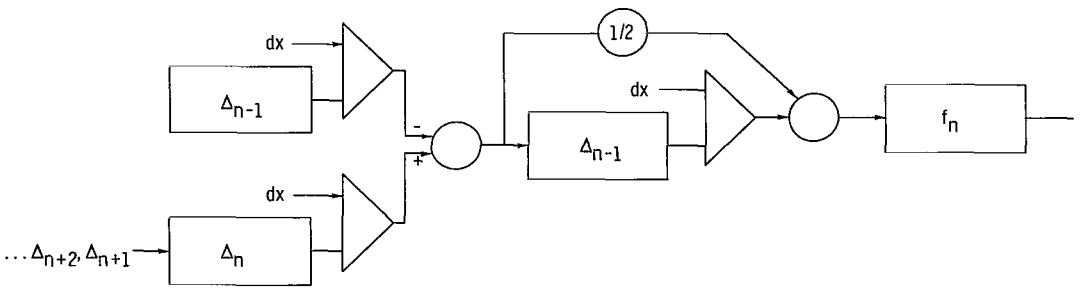


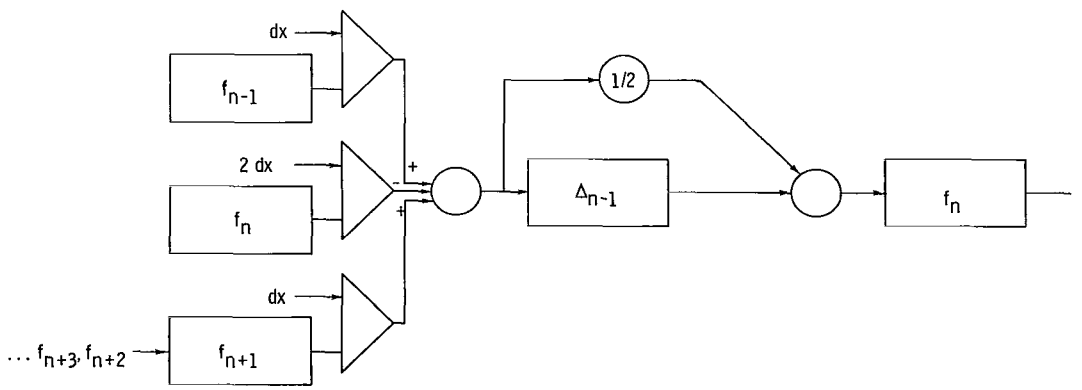
Figure 42. - Scheme for piecewise quadratic interpolation (front interval).



(a) Second difference input data.



(b) First difference input data.



(c) Function values input data.

Figure 43. - Machines for piecewise quadratic interpolation (front interval).

The scheme for piecewise front interval quadratic interpolation illustrated in figure 42 may be derived by use of the Newton-Gauss interpolation formula given in the following equation:

$$f(\omega_n + x\delta\omega) = f_n + x\,\Delta_{n-1} + \frac{x(x+1)}{2}\,\Delta^2_{n-1} \tag{105}$$

If equation (105) is implemented directly, the first derivative and second derivative must be corrected by adding $\frac{1}{2}\Delta^3_{n-1}$ and Δ^3_{n-1}, respectively, to these quantities in order to proceed from one interval to the next. The explicit need for an adder may be avoided in a manner similar to that used in the previous discussion by splintering equation (105) into the following pair of equations:

$$f_a(\omega_n + x\delta\omega) = f_n + x\,\Delta_{n-1} + \tfrac{1}{2}x^2\,\Delta^2_{n-1} \tag{106}$$

$$f_b(\omega_n + x\delta\omega) = \tfrac{1}{2}\Delta^2_{n-1}x \tag{107}$$

Based on this pair of equations, the circuit shown in figure 43(a) may be derived directly. The machines shown in figures 43(b) and (c) are adapted from the circuit shown in figure 43(a) by using the definition of the second difference given in equation (82).

A cubic interpolator may be obtained from the Newton-Gauss interpolation; that is,

$$f(\omega_n + x\delta\omega) = f_n + x\,\Delta_n + \tfrac{1}{2}x(x-1)\Delta^2_{n-1} + \tfrac{1}{6}x(x-1)(x+1)\Delta^3_{n-1} \tag{108}$$

In this case, however, the discussion will be limited to using this formula for central interval interpolation only. This scheme is illustrated in figure 44. The points P_0, P_1, P_2, and P_3 are used to generate an interpolation formula for interpolating between P_1 and P_2. The point P_4 is then added to the scheme and the points P_1, P_2, P_3, and P_4 are used to interpolate between points P_2 and P_3.

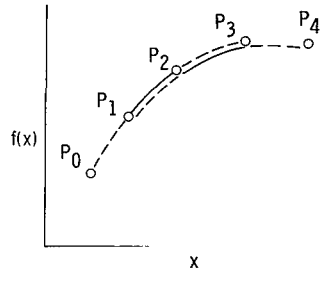
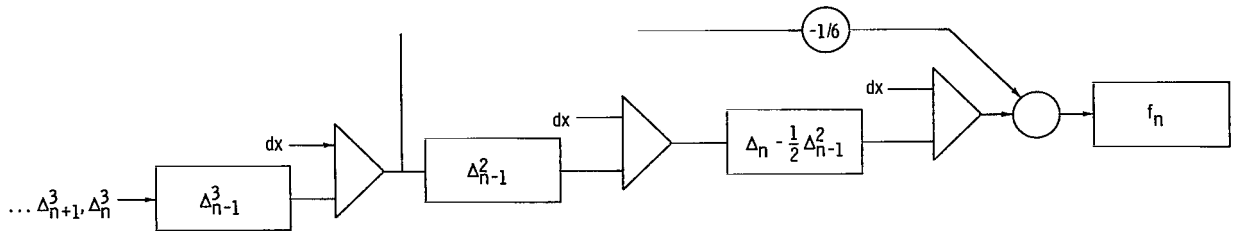


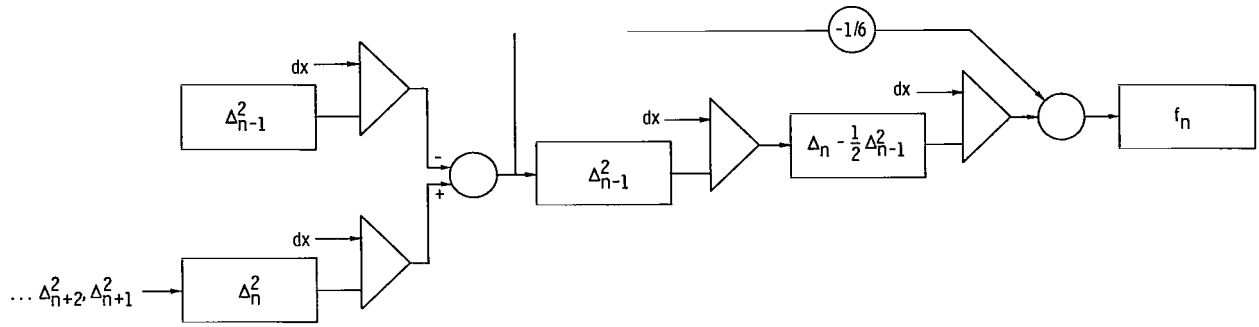
Figure 44. - Scheme for piecewise cubic interpolation (central interval).

Equation (108) may be applied directly to yield a central interval cubic interpolator. In this case, however, the third, second, and first derivatives must be corrected by adding Δ^4_{n-1}, 0, and $\frac{1}{6}\Delta^4_{n-1}$, respectively, to these quantities in order to proceed from one interval to the next.

A configuration may be obtained that conforms with the design practice of not using an adder by splintering equation (108)



(a) Third difference input data.



(b) Second difference input data.

Figure 45. - Machines for piecewise cubic interpolation (central interval).

into the following pair of equations:

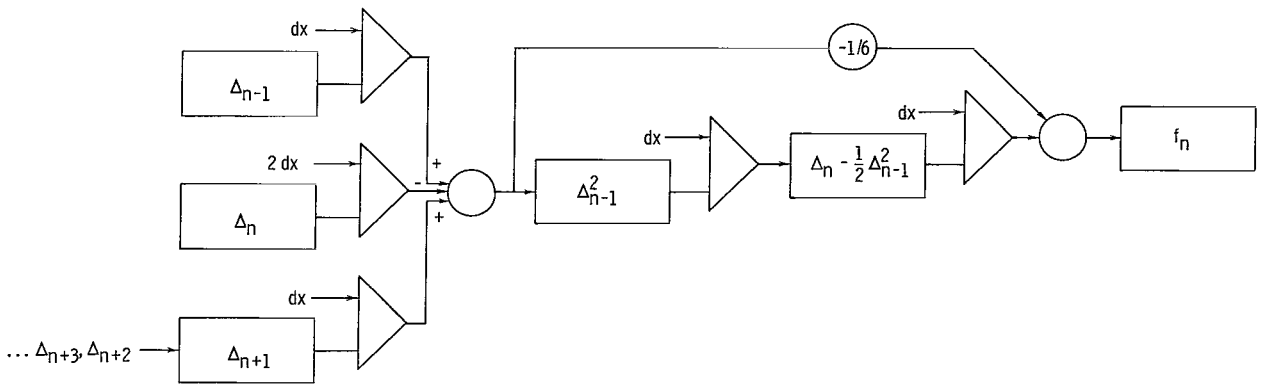
$$f_a(\omega_n + x\delta\omega) = f_n + x\left(\Delta_n - \frac{1}{2}\Delta^2_{n-1}\right) + \frac{x^2}{2}\Delta^2_{n-1} + \frac{x^3}{6}\Delta^3_{n-1} \tag{109}$$

$$f_b(\omega_n + x\delta\omega) = -\frac{x}{6}\Delta^3_{n-1} \tag{110}$$

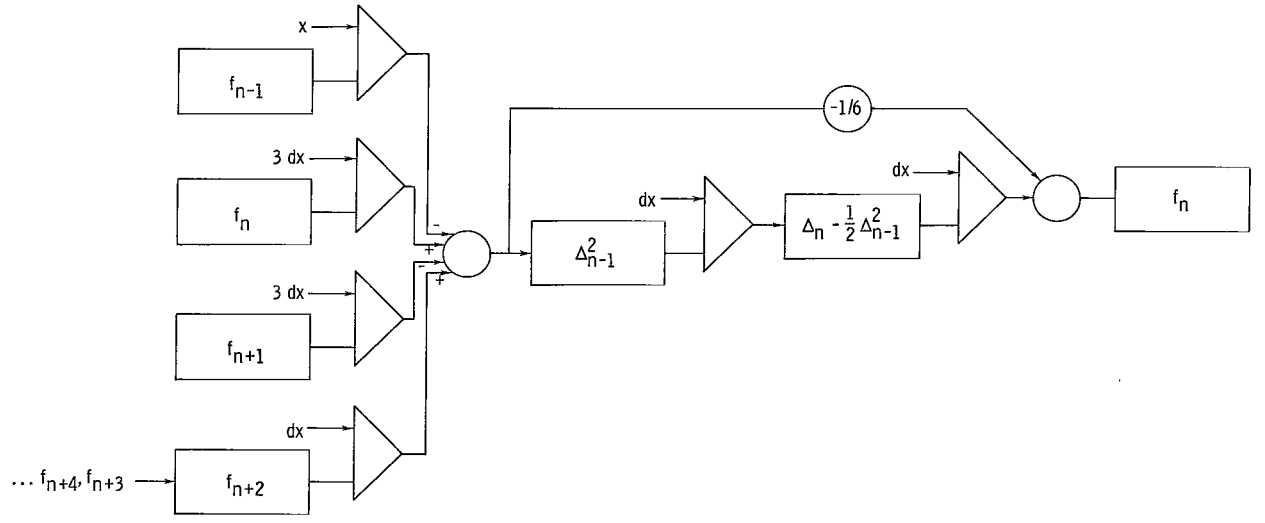
Based on this pair of equations, a circuit may be obtained such that the function, its first derivative, and its second derivative need not be changed in order to proceed from one interval to the next. This circuit is shown in figure 45(a). The circuits presented in figures 45(b) to (d) are modifications of this circuit based on the definition of the third difference.

EXTRAPOLATION

Extrapolation presents an added problem in that the output of the machine must also be corrected in order to proceed from one interval to the next. This is illustrated in



(c) First difference input data.



(d) Function values input data.

Figure 45. - Concluded.

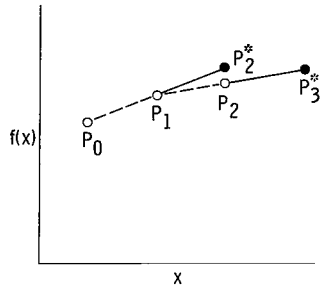


Figure 46. - Scheme for piecewise linear extrapolation.

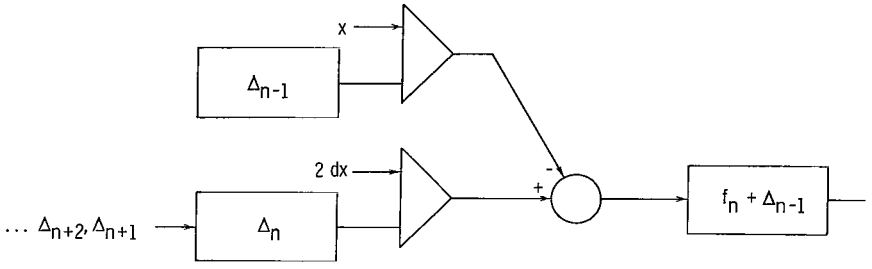


Figure 47. - Machine for piecewise linear extrapolator.

figure 46 for linear extrapolation. A linear polynomial through P_0 and P_1 is used to extrapolate the values from P_1 to P_2^*. The point P_2 is then added to the scheme, and a linear polynomial through P_1 and P_2 is used to extrapolate the next interval. The predicted value P_2^* and the new value P_2 can be expected to be different. Consequently, the output must be corrected for this new value P_2. In order to avoid putting a jump in the output function at this point, the scheme that will be employed is to put in the correction linearly over the entire next interval. This scheme (as well as that of quadratic extrapolation described next) is closely related to the Porter-Stoneman digital filters (see ref. 14) and may be extended accordingly.

The Gregory-Newton backward finite difference formula may be used to design the linear extrapolation machine; that is,

$$f(\omega_n + x\delta\omega) = f_n + \Delta_{n-1}x \tag{111}$$

The corresponding formula for extrapolating the next interval is

$$f(\omega_{n+1} + x\delta\omega) = f_{n+1} + \Delta_n x \tag{112}$$

If these formulas are applied directly, the circuit must be corrected at the end of the interval by adding Δ_{n-1}^2 to both the function and its first derivative in order to proceed into the next interval. This, however, would cause a jump in the output function. This jump may be avoided by putting the correction term in the output in a linear manner over the entire next interval. The resulting polynomial has the property that its initial value corresponds to the end point of equation (111), and its final value corresponds to the end point of equation (112). A polynomial that satisfied these constraints may be written as follows:

$$f(\omega_{n+1} + x\delta\omega) = (f_n + \Delta_{n-1}) + (\Delta_n + \Delta_{n-1}^2)x \tag{113}$$

The second difference in equation (113) may be eliminated by using the defining equation given by equation (82). This substitution yields the following equivalent equation:

$$f(\omega_{n+1} + x\delta\omega) = (f_n + \Delta_{n-1}) + (2\Delta_n - \Delta_{n-1})x \tag{114}$$

Equation (114) may be implemented to yield the linear extrapolator shown in figure 47.

The scheme for quadratic extrapolation is shown in figure 48. The quadratic equation through points P_0, P_1, and P_2 is used to extrapolate the data to the point P_3^*. The point P_3 is then added to the scheme and can, in general, be expected to be different from P_3^*. The quadratic equation through the points P_1, P_2, and P_3 is then used to

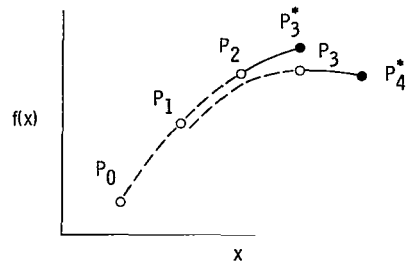


Figure 48. - Scheme for piecewise quadratic extrapolation.

extrapolate to the point P_4^*. In order to avoid putting a jump in the output when new information is added to the scheme, the correction may be put into the output in a linear manner over the entire next interval in the same manner as that employed for linear extrapolation.

The Gregory-Newton backward difference formula forms the basis of the quadratic extrapolation. This formula may be written as follows:

$$f(\omega_n + x\delta\omega) = f_n + x\,\Delta_{n-1} + \frac{x(x+1)}{2}\,\Delta^2_{n-2} \tag{115}$$

The corresponding formula for the next interval is

$$f(\omega_{n+1} + x\delta\omega) = f_{n+1} + x\,\Delta_n + \frac{x(x+1)}{2}\,\Delta_{n-1} \tag{116}$$

If equations (115) and (116) are implemented directly, the quantities Δ^3_{n-2}, $3\Delta^3_{n-2}/2$, and Δ^3_{n-2} must be added to the output function, its derivative, and its second derivative, respectively, in order to proceed from one interval to the other. As was indicated earlier, the jump in the output function can be avoided by putting in the correction over the entire next interval. A polynomial that satisfies these constraints (i. e., has the end of eq. (115) as its initial point and the end of eq. (116) as its final point) may be written as follows:

$$f(\omega_{n+1} + x\delta\omega) = f_n + \Delta_{n-1} + \Delta^2_{n-2} + \left(\Delta_n + \frac{\Delta^2_{n-1}}{2} + \Delta^3_{n-2}\right)x + \frac{\Delta^2_{n-1}}{2}x^2 \tag{117}$$

Substituting the difference relation given by equation (82) into equation (117) yields

$$f(\omega_{n+1} + x\delta\omega) = f_n + \Delta_{n-1} + \Delta^2_{n-2} + \left(\Delta_{n-1} + \frac{5}{2}\Delta^2_{n-1} - \Delta^2_{n-2}\right)x + \frac{\Delta^2_{n-1}}{2}x^2 \tag{118}$$

Equation (118) may be splintered into the two equations

$$f_a(\omega_{n+1} + x\delta\omega) = f_n + \Delta_{n-1} + \Delta^2_{n-2} + \Delta_{n-1}x + \frac{\Delta^2_{n-1}}{2}x^2 \tag{119}$$

$$f_b(\omega_{n+1} + x\delta\omega) = \left(\frac{5}{2}\Delta^2_{n-1} - \Delta^2_{n-2}\right)x \tag{120}$$

to yield the circuit shown in figure 49.

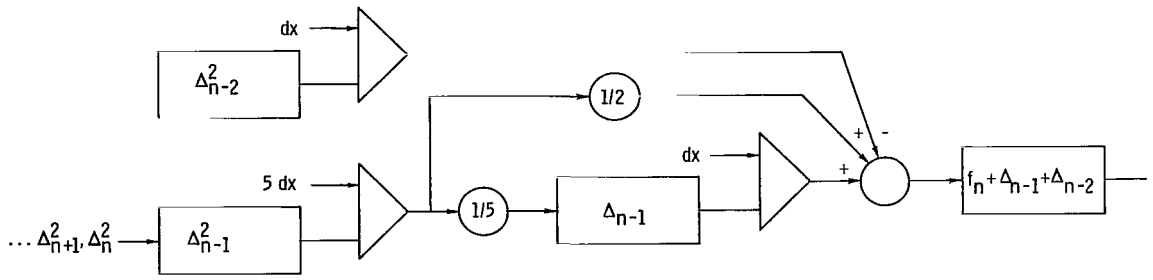


Figure 49. - Machine for piecewise quadratic extrapolator.

The circuits shown in figures 47 and 49 may be readily expanded by use of finite difference relations (as was done for interpolators) to yield circuits that accept functional values and first differences as the primary source of data.

CONCLUDING REMARKS

A handful of circuits has been reported in the literature that were designed to meet the needs of special purpose digital computer problems arising from real time applications. The organization of these circuits is to utilize simple counting techniques as the basis for computing. This results in a simplicity of hardware that make them attractive for such special purpose applications. The design of these circuits has been examined in this study with the objective of (1) "explaining" the circuits and (2) generalizing the design philosophy such that new circuits may be admitted with the same organization. In order to be specific the principal design elements were limited to three fundamental units. The elements are (1) the binary rate multiplier, which is a means of scaling down a pulse stream to some specified fraction, (2) the counter, and (3) the anticoincidence circuit, which is a means of separating pulses arriving at a counter simultaneously.

These design elements are represented as operational units that may be used to describe the machines. Operational techniques are then used as the method of synthesis. In particular, a counter is utilized to represent a first-order difference equation and a counter in cascade with a BRM is utilized to represent approximate integration. The computational error, that is, rounding-off and truncation errors, introduced into the machines as a result of treating the principal design elements as operational units are identified. Having identified these two sources of errors permitted us to obtain better results experimentally by two methods: (1) increasing the number of stages and (2) changing the round-off error by changing the starting value of the BRM counter.

The method of synthesis is presented in three parts: (1) expressing the function to be generated as a differential equation, (2) expressing it as the fixed point of an iterative process, and (3) expressing it in terms of a regenerative circuit that is presented. The

method of synthesis is explicitly stated and is satisfactory in that all known circuits may be directly obtained from it. A wide variety of other functions is also obtained using these synthesis techniques. Many of these examples are illustrated, and in some cases actual experimental results were obtained and discussed with the machine.

A series of machines is presented for interpolation and extrapolation of a function that is available only as empirical data. In particular, the function is generated over its entire range by a sequence of low order polynomials. Finite-difference techniques are used to describe the polynomials. The order of the polynomial is limited to a cubic for interpolation, and a quadratic for extrapolation since these seem to be the important cases in practice. Nevertheless, these techniques can be easily extended to include higher order polynomials.

Lewis Research Center,
National Aeronautics and Space Administration,
Cleveland, Ohio, July 21, 1965.

REFERENCES

1. Harris, J. N.: A Programmed Variable-Rate Counter for Generating the Sine Function. IRE Trans. on Electronic Computers, vol. EC-5, no. 1, Mar. 1956, pp. 21-26.

2. Gordon, B. M.: Adapting Digital Techniques for Automatic Controls - I. Electrical Mfg., vol. 54, no. 5, Nov. 1954, pp. 136-143.

3. Gordon, B. M.: Adapting Digital Techniques for Automatic Controls - II. Electrical Mfg., vol. 54, no. 6, Dec. 1954, pp. 120-125; 298; 300.

4. Arnstein, W.; Mergler, H. W.; and Singer, B.: Digital Linear Interpolation and the Binary Rate Multiplier. Control Eng., vol. 11, no. 6, June 1964, pp. 79-83.

5. Mergler, H. W.: *Digital Control Systems Engineering.* Vols. I and II. Case Inst. Tech., 1961.

6. Braun, E. L.: Digital Computer Design: Its Logic, Circuitry, and Synthesis. Academic Press, Inc., 1963.

7. Soroka, W. W.: Analog Methods in Computation and Simulation. McGraw-Hill Book Co., Inc., 1954.

8. Amble, O.: On a Principle of Connexion for Bush Integrators. J. Sci. Instr., vol. 23, no. 12, Dec. 1946, pp. 284-287.

9. Michel, J. G. L.: Extensions on Differential Analyzer Technique. J. Sci. Instr., vol. 25, no. 10, Oct. 1948, pp. 357-361.

10. Whitaker, E.; and Robinson, G.: Calculus of Observations. D. Van Nostrand, Inc., 1944.

11. Moshos, G. J.: Analog Interpolation for Automatic Control. J. Assoc. Computing Machinery, vol. 2, no. 2, Apr. 1955, pp. 83-91.

12. Mergler, H. W.: A Digital-Analog Machine Tool Control System. Proc. of Western Joint Computer Conf., AIEE-IRE-ACM, 1954, pp. 46-49.

13. Huskey, H. D., ed.: Computer Handbook. McGraw-Hill Book Co., Inc., 1962.

14. Ragazzini, J. R.; and Franklin, G. F.: Sampled-Data Control Systems. McGraw-Hill Book Co., Inc., 1958.

NASA-Langley, 1965 E-2881

3/18/85

"The aeronautical and space activities of the United States shall be conducted so as to contribute . . . to the expansion of human knowledge of phenomena in the atmosphere and space. The Administration shall provide for the widest practicable and appropriate dissemination of information concerning its activities and the results thereof."

—NATIONAL AERONAUTICS AND SPACE ACT OF 1958

NASA SCIENTIFIC AND TECHNICAL PUBLICATIONS

TECHNICAL REPORTS: Scientific and technical information considered important, complete, and a lasting contribution to existing knowledge.

TECHNICAL NOTES: Information less broad in scope but nevertheless of importance as a contribution to existing knowledge.

TECHNICAL MEMORANDUMS: Information receiving limited distribution because of preliminary data, security classification, or other reasons.

CONTRACTOR REPORTS: Technical information generated in connection with a NASA contract or grant and released under NASA auspices.

TECHNICAL TRANSLATIONS: Information published in a foreign language considered to merit NASA distribution in English.

TECHNICAL REPRINTS: Information derived from NASA activities and initially published in the form of journal articles.

SPECIAL PUBLICATIONS: Information derived from or of value to NASA activities but not necessarily reporting the results of individual NASA-programmed scientific efforts. Publications include conference proceedings, monographs, data compilations, handbooks, sourcebooks, and special bibliographies.

Details on the availability of these publications may be obtained from:

SCIENTIFIC AND TECHNICAL INFORMATION DIVISION

NATIONAL AERONAUTICS AND SPACE ADMINISTRATION

Washington, D.C. 20546