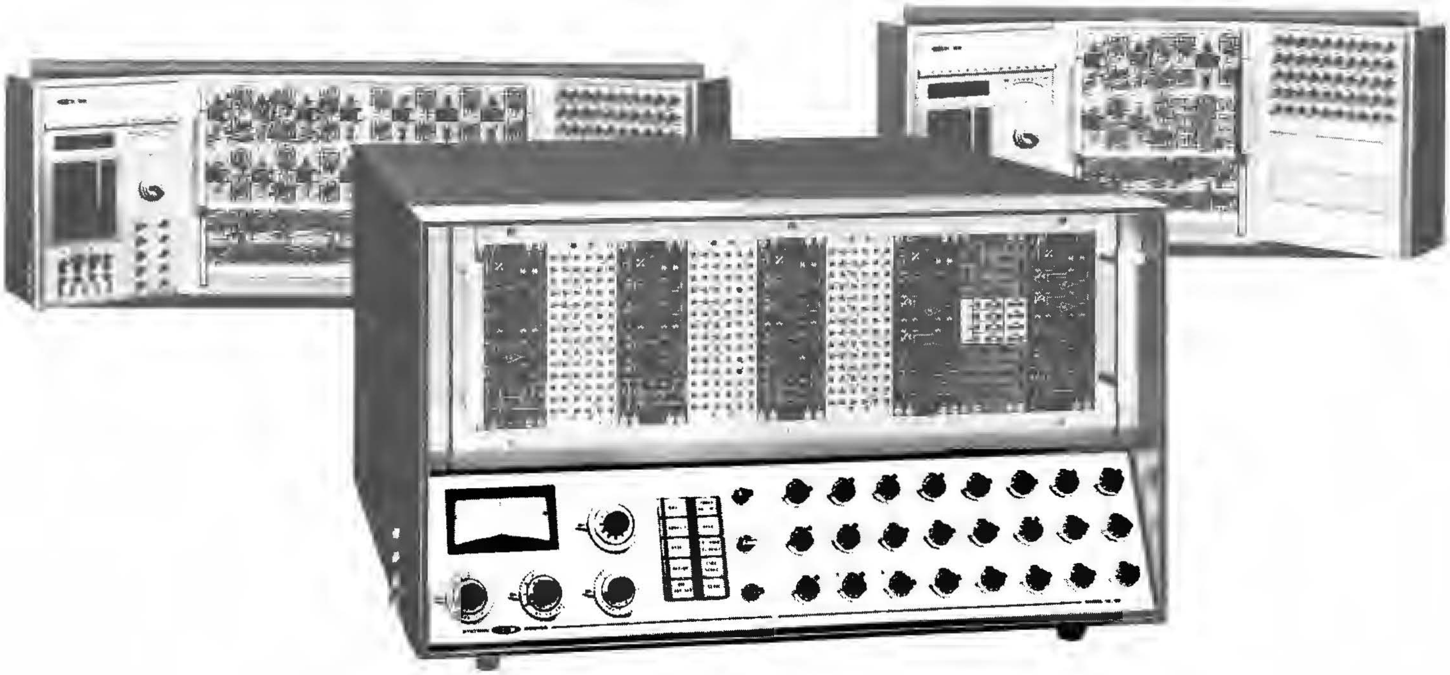
BLOCK PROGRAMMING FOR PHYSICAL SYSTEMS



SYSTRON DONNER
CORPORATION

Prepared by:

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

and

Edward M. Billinghurst

Chief Engineer - Analog Computers
SYSTRON-DONNER CORPORATION

Additional copies of this handbook may be obtained from:

Marketing Department
Analog Computers
SYSTRON-DONNER CORPORATION
888 Galindo Street
Concord, California 94520 U.S.A.

BLOCK PROGRAMMING

A Valuable Aid to Simulation of Lumped-Parameter Physical Systems

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

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

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

MATHEMATICAL OPERATOR (O)

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

Summer

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

Inverter - -1 Sign Changer

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

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

Variable Multiplier

Arbitrary Function - f(x)

Analytical Function Trigonometric, Log, Hyperbolic, etc.

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

ELEMENT

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

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

BLOCK

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

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

DRIVING SOURCE

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

SYSTEM STABILITY

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

NETWORK

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

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

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

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

TABLE I. Single-element Computing Blocks

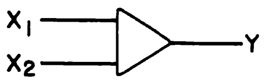
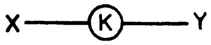
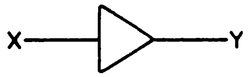
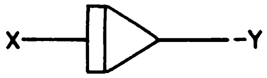
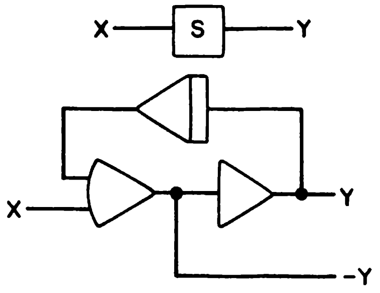
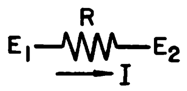
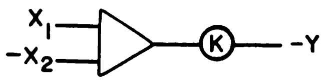
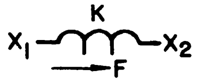
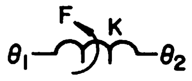
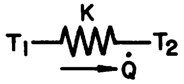
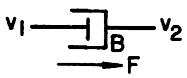
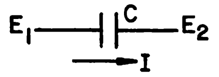
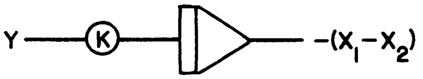
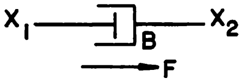
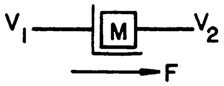
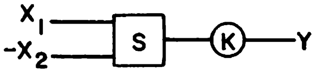
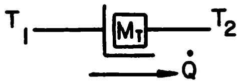
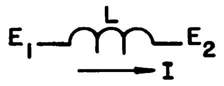
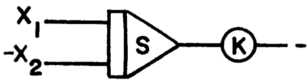
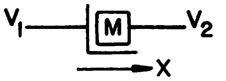
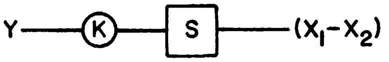
PHYSICAL ELEMENT	SCHEMATIC SYMBOL	MATHEMATICAL RELATIONSHIP	PROGRAM BLOCK
		Summer Σ $Y = -(X_1 + X_2)$	
		Constant Multiplier (POT) $Y = KX$	
		Inverter $Y = -X$	
		Integrator $Y = \frac{X}{s}$	
		Differentiator $Y = sX$	
Resistor		$I = \frac{(E_1 - E_2)}{R}$	
Linear Spring		$F = K(X_1 - X_2)$	
Torsional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

TABLE I (Continued)

PHYSICAL ELEMENT	SCHEMATIC SYMBOL	MATHEMATICAL RELATIONSHIP	PROGRAM BLOCK
Electrical Capacitor		$(E_1 - E_2) = \frac{I}{sC}$	
Viscous Damper		$(X_1 - X_2) = \frac{F}{sB}$	
Mechanical Mass		$(v_1 - v_2) = \frac{F}{sM}$	
Thermal Mass		$(T_1 - T_2) = \frac{\dot{Q}}{sM_T}$	
Electrical Inductor		$I = \frac{(E_1 - E_2)}{sL}$	
Mechanical Mass		$X = \frac{(v_1 - v_2)}{s}$	

COMPUTER PROGRAM MECHANIZATION COMMENTS

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

a) assigning a position in the mechanization diagram to all necessary variables and operators.

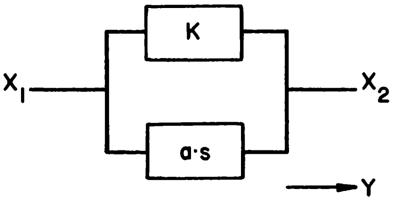
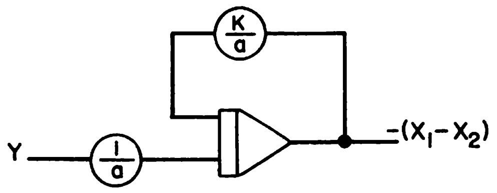
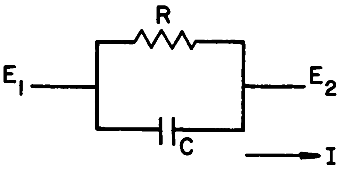
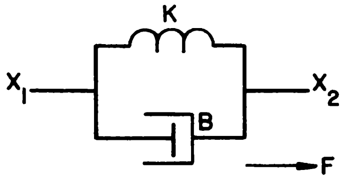
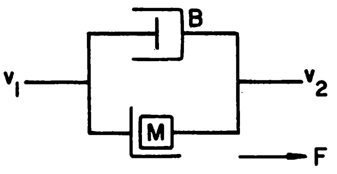
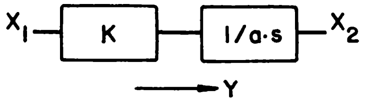
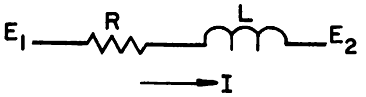
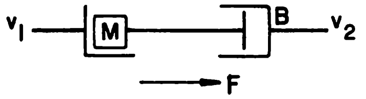
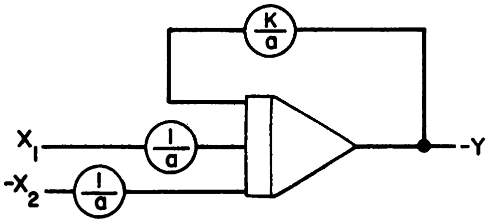
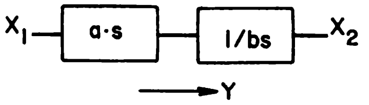
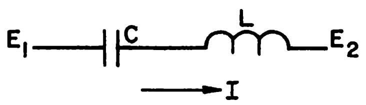
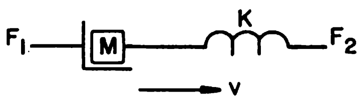
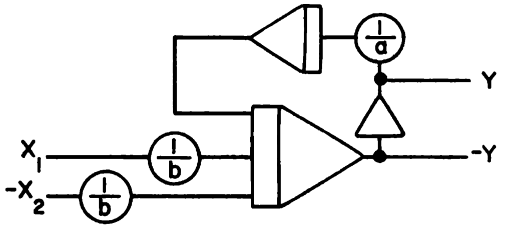
b) accomplishing the necessary interconnections so that each output has its proper inputs.

c) checking to see that all input/output relationships in the mechanization are satisfied.

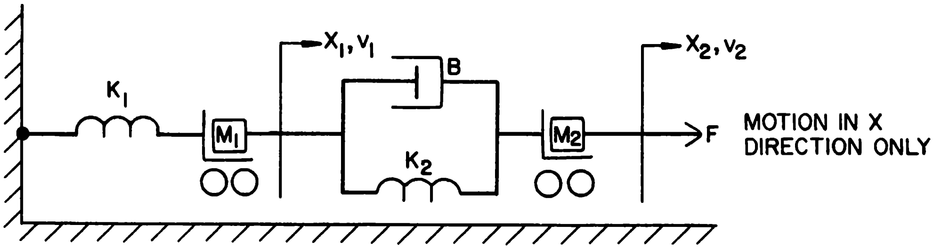
EXAMPLES OF NETWORK PROGRAMMING

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

TABLE II. Two-Element Computing Blocks

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

MASS - SPRING - DAMPER SYSTEM



Element - force equations

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

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

Damper B: $F_D = B (v_2 - v_1)$

Initial values at $t = 0$.

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

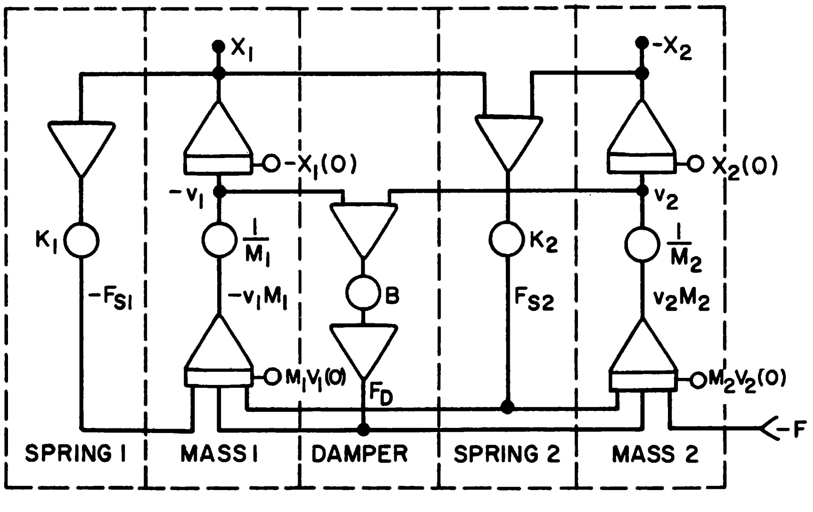
Forces acting on masses

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

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

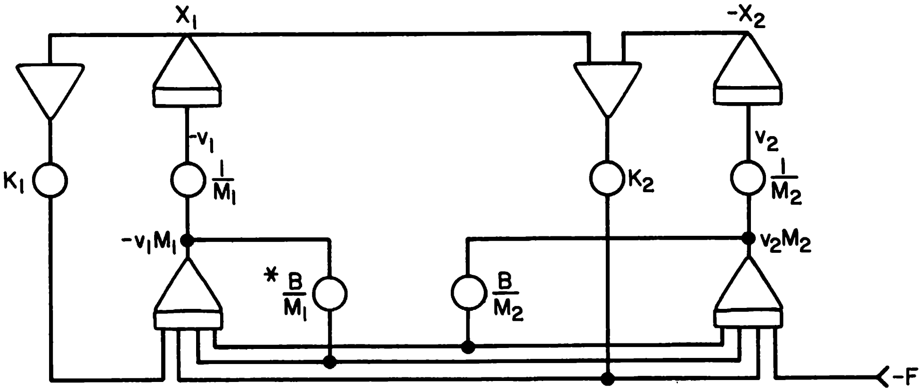
Steps to mechanize

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



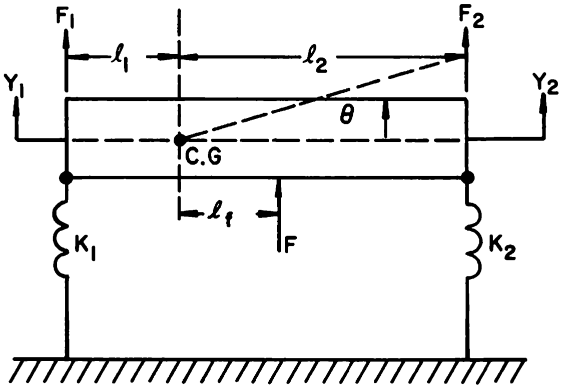
The above mechanization makes each parameter available as an individual adjustment. The programming is easily accomplished by generating the forces imposed on the elements and then operating on the forces to obtain velocity and displacement. Any or all of the initial conditions, $X_1(0)$, $X_2(0)$, $v_1(0)$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

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



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

Rigid Bar Supported on Springs

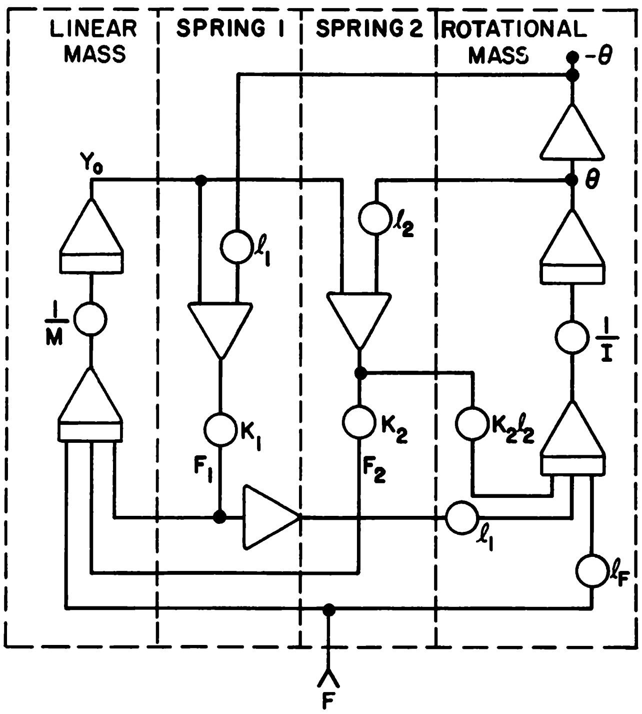


Rigid bar of mass M, moment of inertia about C. G. of I. Assume θ small such that $\theta \approx \sin\theta$. Vertical motion only. Y_0 is displacement of C.G.

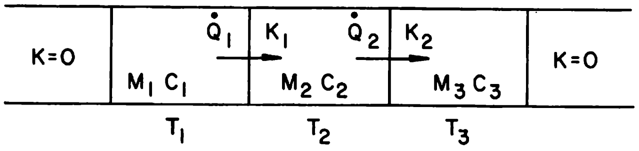
$$\dot{\theta} = \frac{F_2\ell_2 - F_1\ell_1 + F\ell_F}{sI} \qquad \dot{Y}_0 = \frac{F_1 + F_2 + F}{sM}$$

$$F_1 = -K_1 Y_1 \qquad Y_1 = Y_0 - \ell_1\theta$$

$$F_2 = -K_2 Y_2 \qquad Y_2 = Y_0 + \ell_2\theta$$



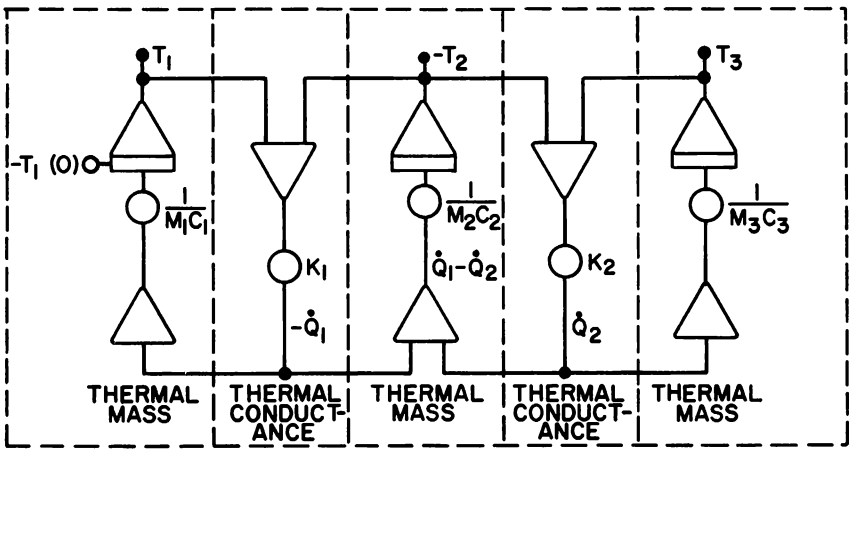
One-Dimensional Heat Transfer

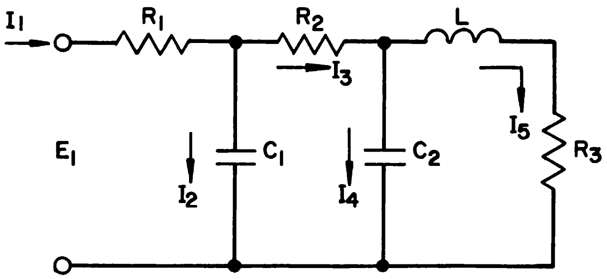


K - THERMAL CONDUCTIVITY
M - MASS
C - SPECIFIC HEAT

$$\dot{Q}_1 = K_1 (T_1 - T_2) \qquad \dot{Q}_2 = K_2 (T_2 - T_3)$$

$$sT_1 = \frac{\dot{Q}_1}{M_1 c_1} \qquad sT_2 = \frac{\dot{Q}_1 - \dot{Q}_2}{M_2 c_2} \qquad sT_3 = \frac{\dot{Q}_2}{M_3 c_3}$$



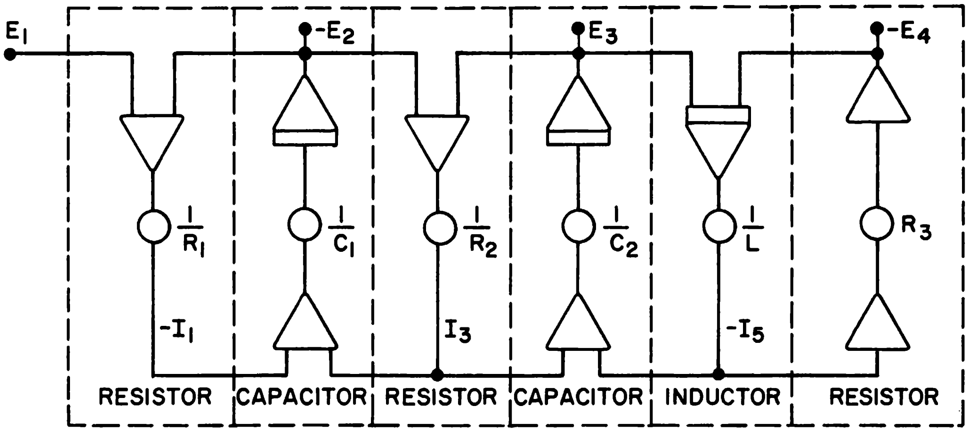


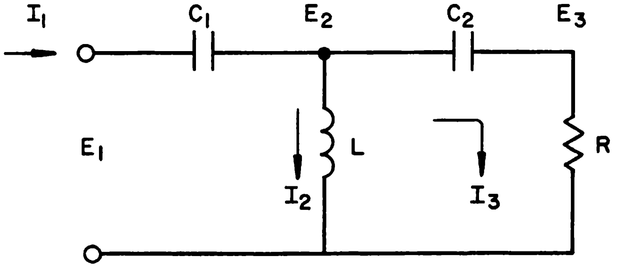
$$I_1 = \frac{E_1 - E_2}{R_1} \qquad I_2 = I_1 - I_3$$

$$I_3 = \frac{E_2 - E_3}{R_2} \qquad I_4 = I_3 - I_5$$

$$I_s = \frac{E_3 - E_4}{sL} \qquad E_2 = \frac{I_2}{sC_1}$$

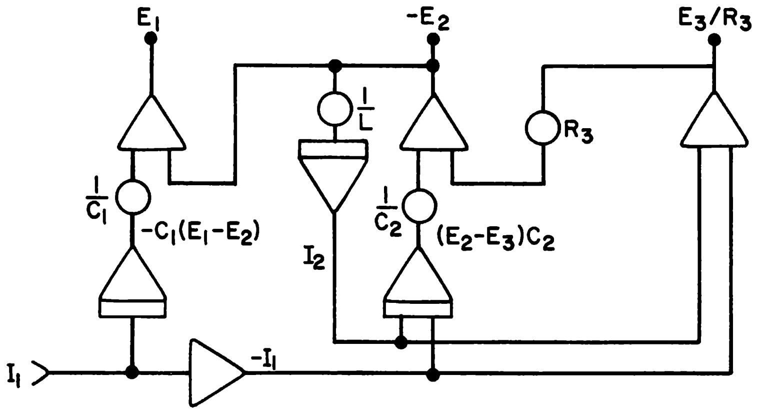
$$E_4 = I_5 R_3 \qquad E_3 = \frac{I_4}{sC_2}$$



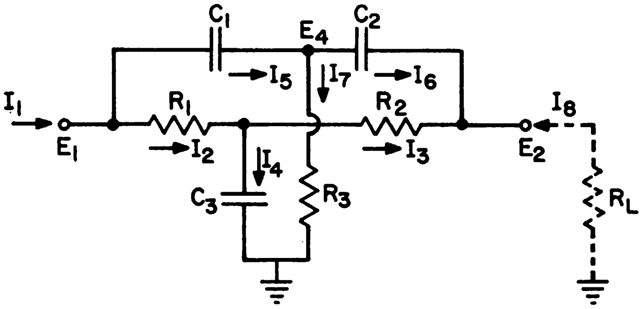


$$E_1 - E_2 = \frac{I_1}{sC_1} \qquad E_2 - E_3 = \frac{I_3}{sC_2} \qquad E_3 = I_3 R_3$$

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



TWIN-T NETWORK



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

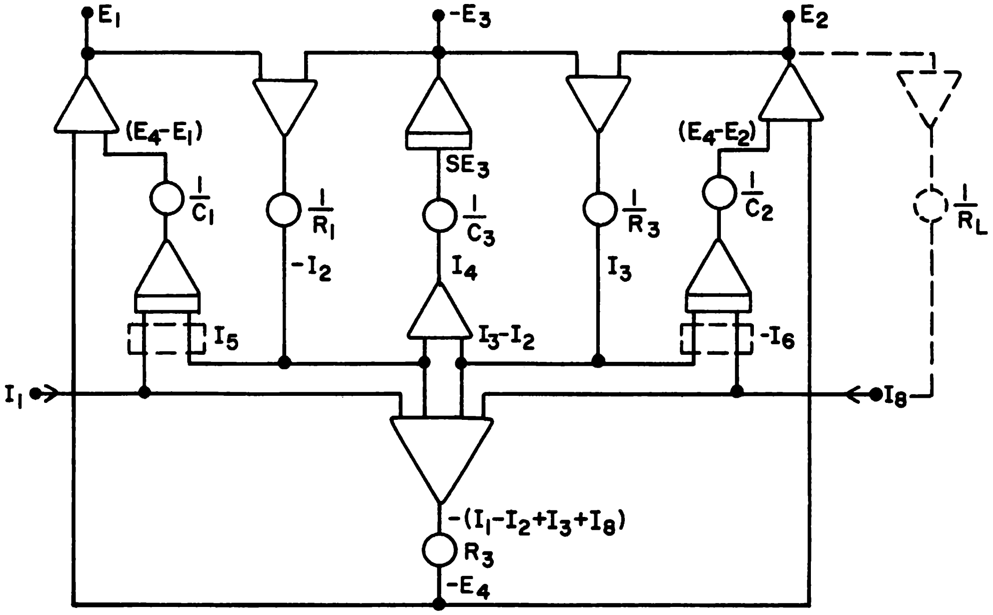
$$-I_3 = \frac{E_2 - E_3}{R_2}$$

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

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

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

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



SCALING

The mechanization diagrams associated with the preceding examples show the computing element interconnections without indicating potentiometer settings, amplifier input resistor values, and integrator capacitor values. These are unscaled mechanizations. Scaling is required to relate the computer voltages to the physical problem variables.

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

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

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

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

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

$$V = a \cdot X$$

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

$$t_c = Nt_p, \; dt_c^m = N^m dt_p^m$$

It follows that

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

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

The general procedure for scaling is:

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

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

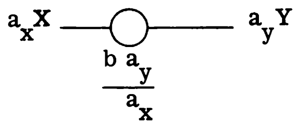
POTENTIOMETER

Physical equation: $Y = bX$

Scale factors $V_x = a_x X, \; V_y = a_y Y$

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

Program



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

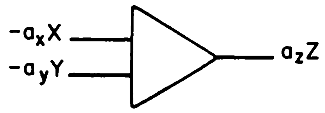
SUMMER

Physical equation: $Z = X + Y$

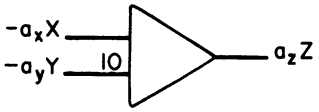
Scale factors $V_z = a_z Z, \; V_x = a_x X, \; V_y = a_y Y$

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

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



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



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

INTEGRATOR

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

Physical equation: $Y = \int X dt_p$

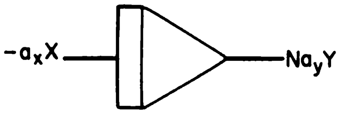
Scale factors $V_y = a_yY,\ V_x = a_xX,\ t_c = Nt_p$

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

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

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

and the program is



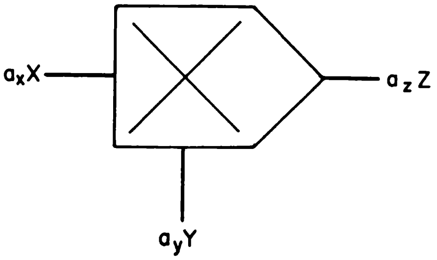
MULTIPLIER

Physical equation: $XY = Z$

Scale factors: $V_x = a_xX,\ V_y = a_yY,\ V_z = a_zZ$

Scaled equation: $\frac{V_xV_y}{100} \ \frac{100a_z}{a_xa_y} = V_z$

Program:



It must be true that

$$\frac{100a_z}{a_xa_y} = 1.$$

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

$$a_z = \frac{a_xa_y}{100}$$

DIVIDER

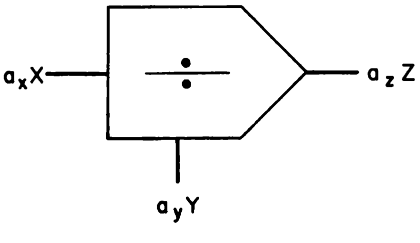
Physical equation: $\frac{X}{Y} = Z$

Scale factors $V_x = a_xX,\ V_y = a_yY,\ V_z = a_zZ$

Scaled equation: $100\frac{V_x}{V_y} \cdot \frac{a_za_y}{100a_x} = V_z$

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

Program:



SQUARE ROOT

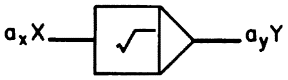
Physical equation: $Y = \sqrt{X}$

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

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

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

Program:



FUNCTION GENERATOR

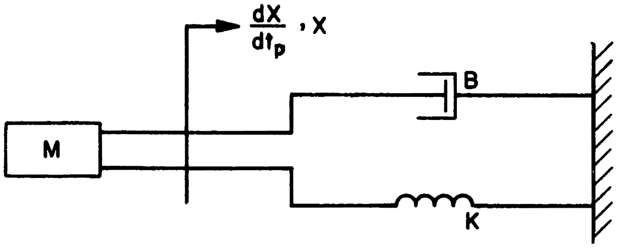
Physical equation: $Y = f(x)$

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

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

EXAMPLE:

Mass-spring-damper system



Physical equations: Spring Force, $F_s = KX$

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

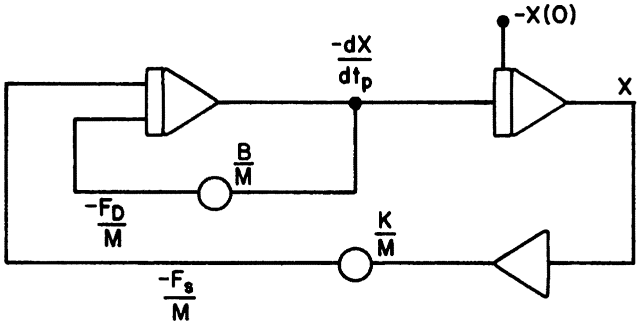
Mass Acceleration, $\frac{d^2X}{dt_p^2} = \frac{-(F_s + F_D)}{M}$

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

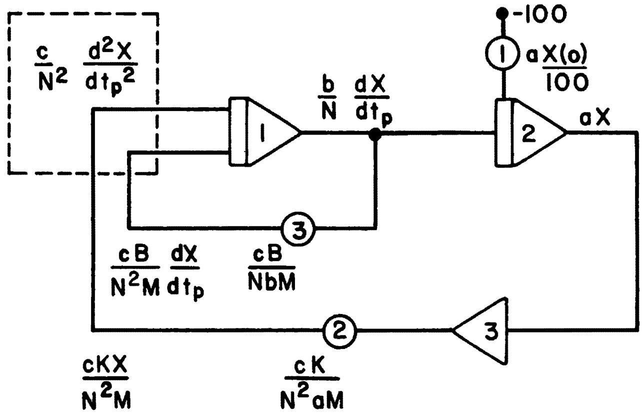
$\frac{dX}{dt_p}(0) = 0, \ X(0) = .95$

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

The unscaled program is:



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


$\frac{c}{N^2}\frac{d^2X}{dt_p^2}$
-100
1
$\frac{aX(o)}{100}$
$\frac{b}{N}\frac{dX}{dt_p}$
1
2
aX
3
$\frac{cB}{N^2M}\frac{dX}{dt_p}$
$\frac{cB}{NbM}$
2
3
$\frac{cKX}{N^2M}$
$\frac{cK}{N^2aM}$

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

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

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

   a = 100

2. Potentiometer 1 setting.

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

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

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

$$\frac{b}{N} = \frac{100\ (\text{volts})}{(dX/dt_p)\text{max.}} \qquad (4)$$

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

4. Potentiometer 2 setting;

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

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

c = 100 will satisfy both conditions.

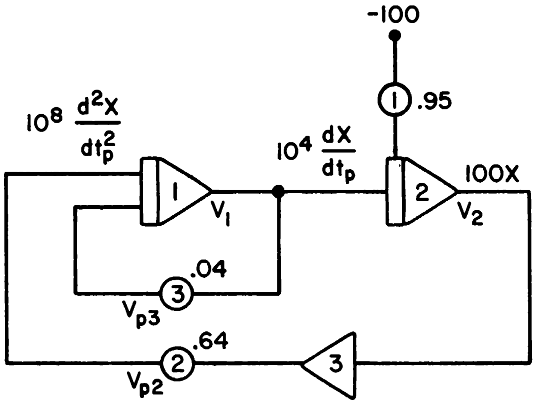
P2 = .64

5. Potentiometer 3 setting:

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

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

The final program with numerical values is:



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

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

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

GENERAL COMMENTS ON TIME SCALING:

The following points concerning time scaling are of particular importance:

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

2. The most convenient way of changing all integrator gains simultaneously is by changing the capacitors by a fixed factor. Integrator capacitors can be simultaneously changed by means of the computer Time Scale control[1].

3. An event which occurs in a fixed real-time interval can occur in several different computer time intervals depending on the choice of integrator capacitors, that is, depending on the time scale factor.

4. Events of identical character which occur in different real-time intervals can all be made to occur in the same computer time interval by appropriate time scaling.

Bibliography of Selected References:

Jackson, A. S., *Analog Computation,* McGraw-Hill Book Co., New York, 1960

Johnson, C. L., *Analog Computer Techniques,* 2nd edition, McGraw-Hill Book Co., New York, 1963.

Karplus, W. J., *Analog Simulation,* McGraw-Hill Book Co., New York, 1958.

Korn, G. A., and T. M. Korn, *Electronic Analog and Hybrid Computers,* McGraw-Hill Book Co., New York, 1964.

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