THE MARCONI REVIEW

Volume XXXVI *No. 190* *Third Quarter 1973*

The Marconi Review is published four times a year
Annual subscription £3. Single copies 75p. Postage paid to any part of the world
Copyright reserved by G.E.C.-Marconi Electronics Limited, Chelmsford, Essex, England

EDITOR C. P. COOPER, B.Sc.Tech., C.Eng., F.I.E.E.

G.E.C.-Marconi Electronics Limited, Baddow Research Laboratories
West Hanningfield Road, Great Baddow, Essex, England

COMPARISON OF SYSTEM SIMULATIONS BY DIGITAL AND ANALOGUE COMPUTERS

by BERNARD de NEUMANN and BARRY PETTICAN

Investigations into the performance capabilities and behaviour of complex systems (for example a servo system) are generally not easily carried out in the form of an analytical study because the calculations involved may be extremely laborious, and the characteristics of the component parts of the system, such as non-linearities may not readily lend themselves to solution by simple mathematical methods. In these cases it is usual to resort to some method of simulation to facilitate study of the system characteristics of interest. This article compares the relative merits of analogue and digital computer simulation and gives examples of the use of each.

Definition of Simulation

In order to perform a simulation of the behaviour of a system* it is necessary to have a model of that system which reflects its essential properties, and which is amenable to experiments that would be impossible, impractical, or too costly to perform on the actual system.

Simulation is the operation and exploration of a model with the purpose of obtaining insight into the behaviour of the system under consideration.

Introduction and Background

The main purpose of this article is to compare the utilities of electronic digital computer simulations and electronic analogue computer simulations. Simulation, in the widest sense of the word has been a pastime of mankind since the dawn of civilisation, and whilst it is our main aim to discuss only

* A system is an organized or self-organizing set of purposive, possibly abstract, components which interact one with another, and possibly as a whole with the system's environment.

simulation by computer we will mention a few examples from outside this limited field when it is expedient.

Early work in the field of analogue computation was carried out by Lord Kelvin during the latter part of the 19th century when he attempted to solve a second-order differential equation by using a mechanical method of integration. The basis of the mechanical integrator he employed is illustrated in Fig. 1; it can be seen that the variable quantities are displacements, either linear or rotational, of different parts of the mechanism. The sum or difference of two shaft displacements may be evaluated with

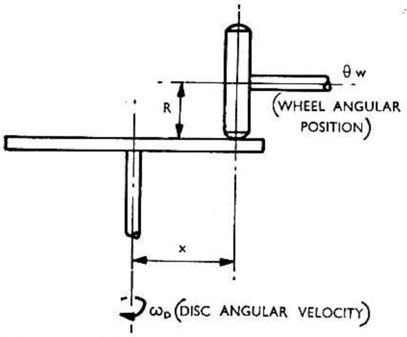


Fig. 1. Wheel and disc integrator.

the aid of a differential gearbox and this device was also put into use by Lord Kelvin in attempting to solve his problem. Consider the second-order differential equation for simple harmonic motion,

$$d^2y/dt^2 + y = 0 \quad \text{i.e. } d^2y/dt^2 = -y.$$

If a displacement representing d^2y/dt^2 is applied as an input to a mechanical integrator, the output displacement is proportional to dy/dt; if this is now connected to a second integrator stage the resulting output is proportional to y. A sign reversal, in this case a reversal of direction of movement gives $-y$. This represents the right-hand side of the equation to be solved. Kelvin's original work attempted the solution by continually updating the input, in an attempt to make it equal to the output from the system obtained from the preceding input. It was not for some time that he realised the equation could be continuously solved by having a direct mechanical connection between input and output, because this compels d^2y/dt^2 to equal minus y.

Kelvin was unsuccessful in producing a solution in this way because of the frictional problems and lack of torque associated with the mechanisms he was employing, despite modifications which were made to the mechanism of the integrator by his brother George Thompson in an attempt to overcome the difficulties.

The idea of closing a loop in the manner conceived by Kelvin is fundamental to many analogue and digital computer simulations.

At about the same time Charles Babbage was designing a machine to evaluate mathematical tables by successive difference methods. This device, which employed different sizes of toothed wheels to perform additions and subtractions, was an elementary mechanical digital computer, and within its limited capabilities was a satisfactory, but slow, method of automatic calculation.

The main impetus to the field of analogue and digital computation was obtained from the military requirements of the Second World War. The United States Army required gunnery tables and attempts were made to produce them by simulating ballistic trajectories with the aid of the mechanical differential analyser. This contained sophisticated versions of the mechanical integrators employed by Lord Kelvin which were able to work satisfactorily under closed-loop conditions. However, there were disadvantages, the principal ones being its slowness and noise-corrupted results, and consequently the Army planned to supersede it with an all electronic sequence-controlled computer, the ENIAC.

The ENIAC was not completed until 1946, too late for the war, and so was used for a variety of other tasks including work on the H-bomb. The ENIAC, being the first computer of its kind, had a number of deficiencies, particularly its small store and its clumsy programming by plug boards. ENIAC did, however, demonstrate that it was possible to assemble usefully very complex equipment (ENIAC had 20,000 valves and required a 200 kW power supply), and spurred Johnny von Neumann and his collaborators to design the first stored program digital computer, the EDVAC.(1) The EDVAC had its internal logical structure specified in terms of the McCulloch-Pitts neuron—a "theoretical" neuron that was postulated in a famous analysis of brain function.(2) One can thus rather loosely say that the electronic digital computer derives from a simulation of the brain.

The introduction of the general-purpose high-speed digital computer had a deleterious effect on mechanical methods of computation and they never recovered. In parallel with the work on ENIAC, Norbert Wiener, the father of cybernetics, was working on the problem of producing automatic fire-control predictors for anti-aircraft batteries, and it is from these devices that the modern electronic analogue computers are derived.

Simulation provides a means for extending the realm of experimentation, and facilitates exploration of the performance of an engineering system without having the hardware available. It enables ideas to be tested which may result in catastrophic failures or incur great expense if tested on the real system. Any system may be simulated providing the physical and mathematical laws governing the behaviour of its component parts are understood; of course simulation can be abused, and may produce bad predictions if care is not taken to ensure that all the relevant physical laws and dependencies are taken into consideration. Indeed simulation, as a technique, is probably most useful when it is used to analyse the performance of a system which consists of many "well understood" parts which interact in a complicated manner. Basically simulation enables one to explore the question "What would be the result of . . . . " in a manner which is often cheaper than experimenting on the actual system, if it already exists.

Scope of Simulation

Analogue simulation as a field includes scale model testing such as wind-tunnel experiments on models of high velocity projectiles, or the determination of a ship's hull stability characteristics from models in large water tanks. An early example of this was the use of the orrery, which consisted of a model of the solar system, with the principal bodies mounted on arms which were driven by a complicated gearing arrangement, and which was used to predict eclipses. A further example derives from wartime research and involved testing Barnes Wallis's ingenious idea of a bouncing bomb; the bomb's feasibility was initially tested by launching marbles on to the water surface of a butt in his garden, and later on a grander scale in the water tanks at the Hydraulic Research Station at Teddington. However, we will confine ourselves to the consideration of simulation on devices that have been constructed specifically for the solution of mathematically defined problems, and which operate on principles which are in general independent of the physical nature of the problem. Such devices go back at least as far as 1876 when Lord Kelvin designed and built a tide predictor that was based upon his mechanical integrator.

Digital simulation has been utilized extensively since 1944 when it was used in the development of the atomic bomb, and has developed into a large and profitable pastime. The colourful term "Monte-Carlo method" stems from 1944 and is often applied to a digital simulation which depends upon stochastic information. This represents a departure from its original meaning; formerly it was the code name given by Johnny von Neumann to a particular technique used in solving the neutron transport equations for the Manhattan project to build an atomic bomb, however this different meaning seems to have become almost universally accepted.

In the early days of the "super" (hydrogen bomb) project, research effort was directed at the approach advocated by Edward Teller. Stan Ulam and Johnny von Neumann had the task to independently simulate the reaction. However, both simulations demonstrated that the proposed device was not feasible and it was thus abandoned. Later Stan Ulam conceived the idea which was to form the basis of the "super" and the efficacy of his scheme was verified by further simulations.

Nowadays the Meteorological Office's computers simulate the genesis and subsequent motion of the barotropic vortices of the earth's atmosphere, and the resulting information is used to produce weather forecasts which are of great economic and social benefit to the community. Multi-sector mathematical models of the economy have been constructed and refined for the past forty years and the more realistic of these are used by government advisers to simulate the effects, and determine the most expedient, of various policies that have been designed to steer the economy towards a particular goal, usually in the shortest possible time. The Road Research Laboratory has utilized computer simulation in its analysis of different types of road intersections and traffic light strategies, in an effort to increase traffic flow. The effect on the tidal flow of the River Thames of the Maplin project (to build an airport, dock and industrial complex off Foulness in the estuary) and the proposed Thames barrage has been simulated by both digital and analogue methods.

There are many other instances where simulation has been profitably applied such as in geophysics, astro-physics, mathematical biology, oceanography, nuclear reactor design, hydrodynamics, communications engineering, and the Apollo and Concorde projects probably could not have functioned without it. Further examples occur in war-gaming and business-gaming where an individual tests his abilities and strategies against simulated adversaries, and also in "world dynamics" where the effects on the earth's future ecological stability, of such things as a reduction in the birthrate in 1980 say, or the discovery of a new pollution free fuel, or the availability of an inexhaustible supply of food are simulated. Indeed it is difficult to imagine that our present society could exist without the tremendous assistance that is derived from these simulations, which are generally only possible with the availability of digital computers. Possible future applications are in the fields of genetic and molecular engineering where, for example, it would be desirable to simulate the effects of possible changes in the structure of DNA, or determine the characteristics of a new chemical compound.

The tremendous demand for digital computer simulations has made it essential for "high level" simulation languages to be implemented on digital computers. Instances of these languages are CSMP (Continuous System Modelling Program), SLANG (Simulation Language), PANSY (Program for the Analysis of Non-linear Systems), SPADE (Simulation Program for the Analysis of Differential Equations) and MODSIM (Modulated System Simulation). SPADE (See Appendix 1) and MODSIM are the languages that are used by the Marconi Company on its ICL System 4–70 and were developed by the Mathematical Physics Group of the GEC-Marconi Electronics Research Laboratories. All of these languages enable a user to specify in a simple form the system which he wishes to analyse, and the range of conditions over which he desires the simulation.

Fundamentals of Digital Simulation

In order to implement any task on a digital computer it is necessary to define the task in terms of an available computer language such as FORTRAN or ALGOL. It was found that simulation tasks were often only clumsily programmed in terms of these languages, and so special simulation languages and programs evolved and have become the basic entities for digital simulation of complex dynamical systems. These languages allow the user to specify his problem in a simulation-polarized manner; that is in terms of "built-in" idealised abstract building-blocks such as integrators, summers, limiters and the like which occur in "non-ideal" form in practice. An example is the SPADE language, which was developed within the Marconi Company, and which allows fairly general FORTRAN-like statements to be written in the definition of the problem; SPADE has all the usual FORTRAN functions available, as well as a selection of logical operators, the "standard" non-linearities and a selection of other "black boxes" (see Appendix 1).

Most digital simulations of dynamical phenomena rely on the ability of the computer to integrate numerically possibly large sets of simultaneous differential equations. These may be characterized by the vector differential equation:

$$\mathbf{y}' = \mathbf{f}(\mathbf{y},t) \text{ with } \mathbf{y}(0) = \mathbf{y}_0, \tag{1}$$

which may be solved numerically by a variety of different methods. The most elementary numerical method for solving equation (1) is to use the approximation:

$$\mathbf{y}_{n+1} = \mathbf{y}_n + \Delta t \mathbf{f}(\mathbf{y}_n, n\Delta t),$$

which is an application of the elementary approximation:

$$y + \Delta y = y + (dy/dx)\,\Delta x.$$

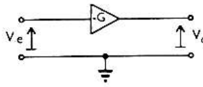
Generally the numerical solution of equation (1) is quite straight-forward, however, there exists a number of pitfalls to trap the unwary which we will not go into here. It is to combat these that some highly sophisticated techniques have been designed in recent years. As one might expect the most efficient (fastest) technique varies from problem to problem and it is because of this characteristic that a spectrum of techniques is usually made available within a simulation program.

The power of a digital simulation program is really only limited by the size of the computer on which it is implemented. For example, on the Marconi 4–70, SPADE could in theory handle up to the order of 10,000 simultaneous differential equations, though it is doubtful that the simulation would be completed in a sensible time. Roughly the time to simulate on a digital computer goes up linearly with the number of integrations, whereas it remains constant on an analogue computer.

Within the Marconi Company there exists a further simulation program, MODSIM which is used for simulating communications systems, mainly in the frequency domain, by utilizing the Fast Fourier Transform*. This program evolved from a "Monte Carlo" simulation of a multi-channel telephone link, and is used to great effect for the prediction of the performances of communications systems.

It is possible to use MODSIM to simulate the steady-state periodic behaviour of non-linear systems (see Appendix 2). The technique used has proved to be a very fast and accurate means of obtaining results from mildly non-linear systems, such as transistor amplifiers(3) and phaselock loops.

Fig. 2.



Fundamentals of Electronic Analogue Simulation

The analogue computer has one basic unit which may be adapted to provide a number of different characteristics. The basic unit is a high gain d.c. amplifier. The essential qualities of the amplifier are (i) a high voltage gain of the order of 10^6, (ii) a high input impedance (greater than $10M\Omega$), (iii) a low output impedance (a few tens of ohms), (iv) a low drift rate or d.c. offset (to achieve this, chopper stabilised amplifiers are often used) and (v) a wide bandwidth, since this will impose a restriction on the maximum operating frequency of the computer system.

Diagrammatically the amplifier may be represented as Fig. 2.

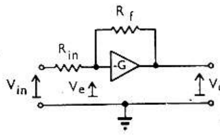
Since the amplifier has a very high gain represented by G the input voltage V_e will tend to zero for all values of V_0 within the linear operating region, i.e. the amplifier input is a *virtual earth*. The amplifier also provides a polarity reversal (inversion) between input and output. Consider the amplifier connected as shown in Fig. 3. If the input impedance is very high no effective current will flow into the amplifier, and Ohm's law gives:

$$(V_{in} - V_e)/R_{in} = (V_e - V_0)/R_f. \tag{2}$$

Also:

$$V_0 = -G\ V_e,$$

Fig. 3.



i.e.

$$V_e = -V_0/G.$$

* The Fast Fourier Transform is a highly efficient technique for calculating the Discrete Fourier Transform

$$x_m = (1/N) \sum_{n=0}^{N-1} \sigma^{nm} X_n, \text{ where } \sigma = \exp(j2\pi/N)$$

and

$$m = 0, 1, 2 \ldots, (N-1),$$

when N is composite (that is decomposable into a large number of integer factors.)

Substituting in equation (2) gives:

$$[V_{in}+(V_0/G)]/R_{in} = -[(V_0/G)+V_0]/R_f$$

$$V_{in}/R_{in} = -\{V_0/R_f+V_0/(GR_f)+V_0/(GR_{in})\}.$$

If G is very large:

$$V_{in}/R_{in} \approx -V_0/R_f,$$

that is:

$$V_0/V_{in} = -R_f/R_{in}.$$

Providing the input impedance and the gain of the amplifier are very high, the voltage gain of the arrangement in Fig. 3 is given by the ratio of the feedback and input resistor values.

Fig. 4.

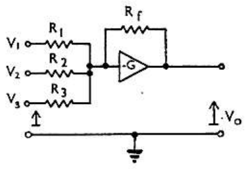
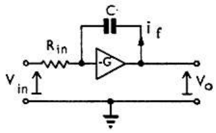


Fig. 5.



By similar argument, for a number of input resistors as shown in Fig. 4 it can be shown that

$$-V_0 = V_1R_f/R_1+V_2R_f/R_2+V_3R_f/R_3.$$

When all resistor values are equal the arrangement provides a voltage summing device, i.e.,

$$-V_0 = V_1+V_2+V_3.$$

Consider the case where the feedback resistor R_f is replaced by a capacitor C as in Fig. 5.

Assuming infinite amplifier gain and input impedance:

$$V_{in}/R_{in} = -i_f$$

and since the amplifier input is a virtual earth:

$$i_f = C(dV_0/dt),$$

or

$$V_{in}/R_{in} = -C(dV_0/dt),$$

or

$$V_0 = -(CR_{in})^{-1}\int V_{in}dt.$$

Thus the output voltage is directly proportional to the time integral of the input voltage.

These two amplifier configurations, Figs. 4 and 5, are the basic building blocks of all electronic analogue computer models.

It can be seen from the previous calculations that the accuracy of an electronic analogue computer is dependent on the quality of its operational amplifiers and the accuracy of the input and feedback component values. The low output impedance of the amplifiers enables several other amplifier configurations to be connected to the output without affecting the operational characteristics of the device.

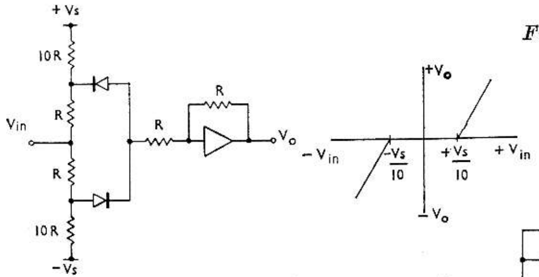


Fig. 6. Dead zone characteristics.

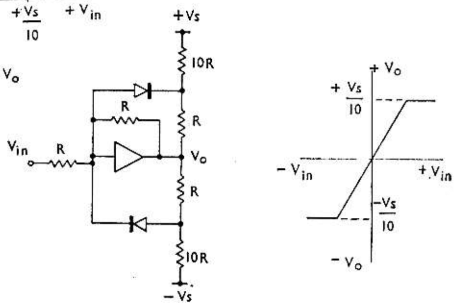


Fig. 7. Voltage limiter.

More sophisticated networks may be connected to the input and used as feedback components on the amplifiers (to provide more complicated transfer functions) and non-linear functions may be generated by the application of non-linear components to the input or feedback circuits. An example of a non-linear component which may be used in this way is a diode. This is frequently used for the generation of dead zones or voltage limiters as shown in Figs. 6 and 7.

The resistor chains producing the "break point" voltages for the non-linear characteristics shown in Figs. 6 and 7 are frequently produced by employing potentiometers built in to the computer, the diodes being connected to the wipers of the potentiometers.

In order that a problem may be set up on an analogue computer it is necessary that the relationships governing the behaviour of the system to be investigated can be expressed mathematically; or alternatively, a block diagram of the system can be drawn representing the system as a number of simple linear or non-linear functional blocks, which may then be translated into an equivalent arrangement expressible in the form of simple transfer functions of the types obtainable with analogue amplifier configurations. An example of a block diagram for part of a hydraulic-mechanical servo system is shown in Fig. 8; and Fig. 9 shows the analogue computer diagram for simulation of the same system. All of the amplifier output voltages in the simulation are representative of a particular physical quantity existing in the real system, for example, the output

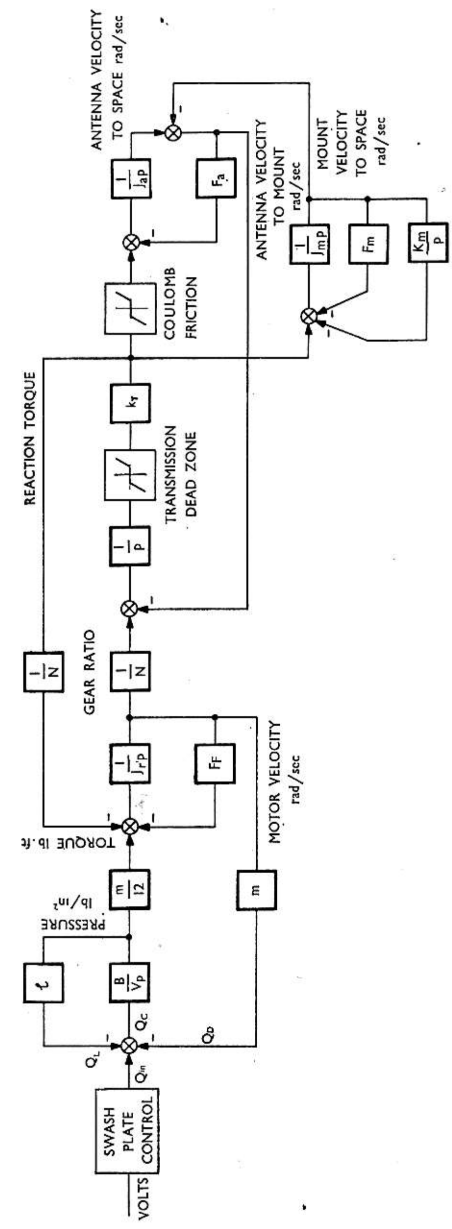


Fig. 8. Block diagram of hydraulic drive, antenna and mount.

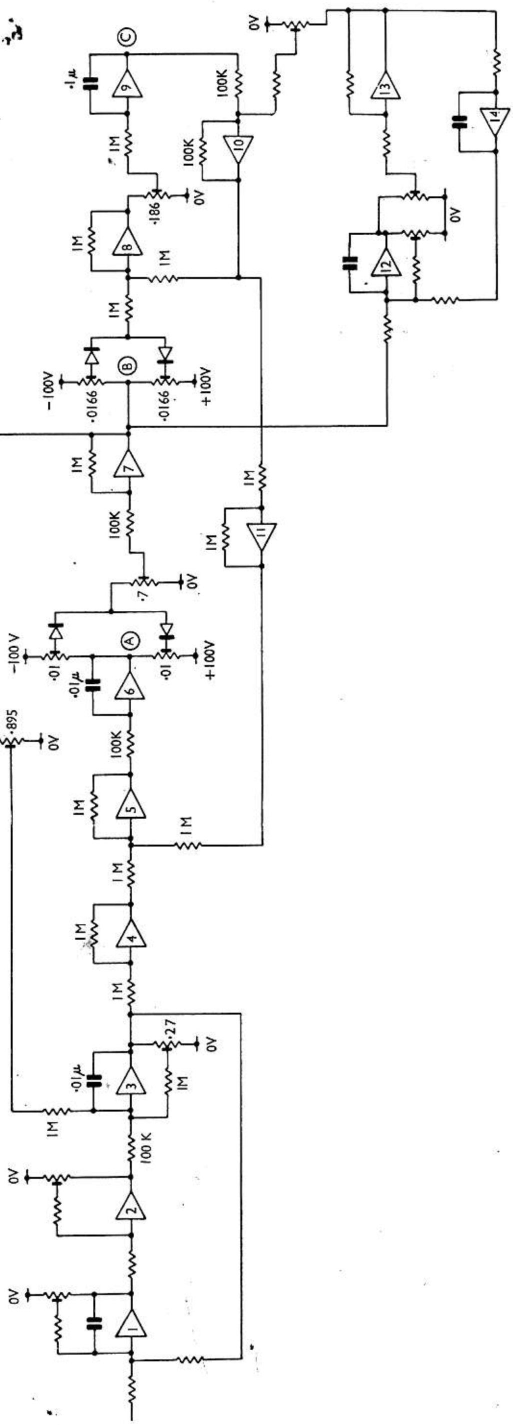


Fig. 9. Electronic analogue simulation of Fig. 8.

voltage of amplifier 3 in Fig. 9 represents motor shaft velocity and the voltage of amplifier 9 represents torque available to accelerate the load. The question then arises as to how much velocity, torque etc. is represented by a particular value of voltage at these points. The scale factors to be applied at the various points are easily calculated where purely linear systems are to be simulated since all the quantities will have the same relationships with each other irrespective of the input signal levels applied.

This is not the case where non-linear functions are included in the model; then a scale factor must be allocated to one of these functions first, all other scale factors being determined from that for the chosen non-linearity. In the example in Fig. 9 the transmission dead zone was scaled so that 1 volt was equivalent to 1 minute of arc. A typical linear operating range of output voltage for analogue computer amplifiers is $\pm$ 100 volts, and care must be taken to ensure that no amplifier output will saturate during the simulation. In some cases therefore, considerable juggling of scale factors and amplifier gains and time constants is necessary to produce a model which works satisfactorily over a wide dynamic range.

The interconnections between inputs and outputs of the amplifier blocks comprising the model are normally made by inserting wire links on some type of patch panel. In some computer systems a number of detachable patch panels is provided so that several different models may be constructed and interchanged on the computer with comparative ease. In this type of machine the operational components for example, potentiometers, diodes and close-tolerance resistors and capacitors are built in to the machine and it is necessary to re-set potentiometers when interchanging patch panels. To maintain accuracy, provision is made to set up the values of the potentiometer ratios with the wiper connected to any input and loading resistors used in the model so that loading effects are taken into account. It is usual for the independent variable in an analogue computer model to be real time, but a problem may be arranged so that the computer works in scaled time, for example, time constants in the model may be made, say, 1,000 times larger than reality so that one second on the analogue computer compares with one millisecond on the real system. This facility is useful when a model is to be made of a fast system which may possess components having wider bandwidths than the amplifier in the analogue computer.

Inputs to the model may be injected directly from a waveform generator and may take the form of sine-waves, square-waves or ramps. Outputs may be monitored on an oscilloscope or some form of chart recorder. If required a transfer function analyser may be directly connected to the model to obtain the gain and phase response of any desired part of the arrangement.

Mention has been made of two non-linear functions which may be produced with the aid of suitable input or feedback components. In addition special auxiliary units may be used in conjunction with the operational amplifier to obtain other non-linear functions; examples of these are analogue multipliers, sine/cosine resolvers and diode function-generators; the latter using biassed diodes to switch in different input and feedback resistor values to vary the overall gain of the arrangement according to different input or output voltages. In this way a non-linear input/output characteristic is obtained by constructing it with a number of straight line approximations.

Analogue computers vary in size from small portable units, with as few as five operational amplifiers, which are frequently used for educational purposes, to large installations containing many hundreds of amplifiers and occupying four or five free standing cabinets.

Comparisons Between Digital and Analogue Methods of Simulation

First of all it is appropriate to discuss the versatility of the main types of computer. The analogue computer is a rather specialised piece of equipment which has only a low adaptability compared with the digital computer. This is not to say that the analogue computer is not useful, it is often ideal for simulating continuous dynamic systems in which accuracy plays a small part, but this is only a very restricted field of the realm of the digital computer. The digital computer can in principle solve any problem that the analogue computer can handle, though sometimes at greater direct cost. Digital computers are however extremely versatile. In 1936 Alan Turing showed that there exists an axiomatic finite-state machine which can in principle perform any given algorithmic procedure.(4) He also demonstrated the mathematical existence of a finite-state automaton which could simulate the behaviour of any other particular finite-state automaton.* On the basis of these theorems one might be tempted to think that the digital computer can in principle solve any problem. This is not the case, and Turing also showed that it is impossible to construct a digital automaton that could determine infallibly whether another digital automaton would ever complete its execution of an arbitrary algorithm.

The versatile algorithmic ability of the digital computer has made it increasingly indispensable to business operations during the past 25 years and it has now become relatively easy to obtain access via a terminal, so that, providing the computer system is programmed to work with one of the high level simulation languages, no additional computer hardware is needed to produce a simulation of any type of system that the user wishes

* It is the existence of this theorem which allows FORTRAN, ALGOL and other computer languages to be implemented on digital computers and which makes digital computers such powerful simulators.

to investigate. It is also possible (within the restrictions imposed by the computer streaming system) for a number of users to work with a large number of complicated simulations simultaneously. The digital computer does not simulate in real time and therefore it is necessary to re-enter the program for each set of parameter changes in the model, in order to determine their effects. This may be rather expensive if the model is complicated and consumes a large amount of computer time, and for this reason it is necessary to exercise care and selectivity in the system variations investigated wherever this problem occurs.

Since all variables in the digital simulation are stored as numbers which represent the instantaneous state of the system at a point in time, there are no problems associated with unintentional saturation or limiting as may occur during the running of an analogue simulation. Output data is obtained as a table of dependent variables displayed at selected points of the independent variable. It is then often possible to examine the output variables with the aid of an interactive video display, and analyse and plot interesting results as desired from the display.

Digital computer simulations are isolated from the external environment and numerical integration methods, although basically approximations, will not suffer from the drift problems occurring in analogue computer integrators. Values of the variables occurring in the simulations may be stored indefinitely within the computer and conditions occurring at any instantaneous value of the independent variable may be precisely repeated at a later date if required.

A digital computer used within the context of this article cannot be interfaced with actual hardware although certain computers which attempt to combine the advantages of both analogue and digital systems may be interfaced via D—A and A—D converters, with either hardware or other analogue computer amplifiers. In this case the overall system is described as hybrid.

It is necessary to be able to describe the performance of any particular component within the digital simulation in the form of a mathematical or physical law although in some cases, if a direct law is not available, it is possible to make the computer interpolate over a discretely measured characteristic.

Analogue computers need a large number of expensive operational amplifiers to enable complex simulations to be investigated, and for this reason unless continuous use can be made of a large analogue computer system it may not be an economic proposition: the technique of employing detachable patch panels to enable more than one simulation to be set up and interchanged with others quickly, may be advantageous in certain circumstances, but the analogue computer cannot compete with the digital system in terms of simultaneous usage. The analogue computer works in real time or transformed time and the effect of changing any parameter may be immediately observed; there is also no cost penalty involved in carrying out investigations with a large number of different system constants where this is desired. The independent variable is always time although it is possible to use another quantity providing it may be equated to time for the purposes of the simulation.

Since all independent variables in the electronic analogue computer are represented by voltages, care must be taken, as previously mentioned, to ensure that the maximum voltage that an operational amplifier can produce within its linear range is not exceeded during the simulation period. Any output may be continuously monitored and output waveforms displayed on a storage oscilloscope or similar directly-connected recording device.

Analogue computers are not always isolated from the external environment and the electronic integrators will be sensitive to temperature changes causing varying drift rates and therefore errors. Electronic integrators will store their output values for only a limited time before capacitor leakage current causes significant errors. It is therefore difficult to repeat precisely a particular set of conditions at a later date. In a large number of applications however, the repeatability accuracy will be sufficient for the problem under consideration.

An analogue computer may be interfaced directly with actual system hardware providing that only electrical signals cross the interface. It may also be interfaced with a device whose internal characteristics are not fully defined, subject to the same conditions, with regard to signals crossing the interface.

An Example of an Electronic Analogue Simulation

This example considers the construction of an electronic analogue model used in simulating the performance of a proposed servo control system for pointing a radar tracking antenna. The simulation considered here is purely that of the hydraulic drive, the antenna and its mounting.

An analogue model was chosen for the following reasons:

(i) It would be possible to directly interface the analogue model with the hardware used to control the real system if required—with the additions of a model of the swash plate control loop and some device giving an electrical feedback of antenna position or velocity.

(ii) The problem was principally concerned with optimising the frequency response of the proposed servo system. This is easily carried out on an analogue simulation by connecting a transfer function analyser to obtain a direct readout of gain and phase shift at each input frequency chosen.

(iii) A large number of variations of servo loop compensation terms could be tested in the minimum amount of time since the effects of changes could be immediately observed.

The first stage in producing the model was to construct the block diagram shown in Fig. 8 using the available information as to the values of the constants listed in Table 1. The block diagram was produced by

TABLE 1

Constants for system block diagram of Fig. 8

Symbol	Meaning	Value and Units
B	fluid bulk modulus	$2{\cdot}45 \times 10^5$ lbs/in.^{2}
V	fluid volume	$6{\cdot}25$ in.^{3}
m	displacement constant	$0{\cdot}7$ in.^{3}/radian.
J	rotor inertia	$0{\cdot}001$ lb.ft sec.^{2}
F_F	viscous damping	$0{\cdot}027$ lb.ft/rad/sec.
Q_{in}	input oil flow	in.^{3}/sec.
Q_L	leakage flow	in.^{3}/sec.
Q_C	compressibility flow	in.^{3}/sec.
Q_D	displacement flow	in.^{3}/sec.
N	gear ratio	59
K_T	transmission compliance	$2{\cdot}18 \times 10^6$ lb.ft/rad.
J_a	antenna inertia	167 lb. ft sec.^{2}
F_a	viscous drag	310 lb.ft/rad/sec.
J_m	Mount inertia	5,280 lb.ft sec.^{2}
F_m	mount damping	34,400 lb.ft/rad/sec.
k_m	mount compliance	35×10^6 lb.ft/rad.

Transmission dead zone — ± 1 minute of arc.
Final bearing coulomb friction — ± 150 lb.ft.
p denotes the Laplace operator.

considering the basic laws of physics and mechanics applied to each part of the arrangement. Care must be taken to ensure the authenticity of the block diagram since any errors made at this stage will be carried through the whole exercise and give rise to false information in the results obtained.

The next stage was to construct an analogue circuit diagram (Fig. 9) based on the arrangement of the block diagram. If comparison is made between the two diagrams of Figs. 8 and 9 the similarities in the arrangements should be immediately obvious. At each stage where a physical integration occurs (for example, the application of a torque to a mass with inertia will give rise to acceleration which when integrated gives a velocity) the analogue model diagram contains an electronic integrator. Each closed loop in the model corresponds to a closed loop in the original block diagram.

The transfer function of each loop in the electronic analogue model is designed to have the same transfer function as the corresponding loop in the original block diagram. For example, consider the reaction torque feedback loop in the block diagram, ignoring the transmission deadzone, this has the transfer function:

$$\{K_T/N^2p\}\{1/F_F\}\{1/[1+(J/F_F)p]\} = \{23{,}200/p\}\{1/[1+(0.001/0.027)p]\}$$

The electronic analogue loop has the transfer function:

$$\{1000/p\}\{0{\cdot}7\times 10\times 0{\cdot}895/0{\cdot}27\}\{1/[1+(0{\cdot}01/0{\cdot}27)p]\},$$

$$= \{23{,}200/p\}\{1/[1+(0{\cdot}01/0{\cdot}27)p]\}$$

The scaling at each point in the model is based on that selected for the transmission dead zone. This was 1 volt = 1 minute of arc.

The transmission compliance is

$$2{\cdot}18\times 10^6 \text{ lb.ft/rad.}$$

or

$$635 \text{ lb.ft/minute of arc}$$

1 Volt at (A) will produce 7 volts at (B) a torque analogue point in the simulation, i.e.

$$7 \text{ volts} = 635 \text{ lb.ft.}$$

or

$$1 \text{ volt} = 90{\cdot}8 \text{ lb.ft.}$$

Since the coulomb friction was taken as $\pm$ 150 lb.ft, then the setting of the potentiometers at (B) will be that which gives a dead zone of 150/90·8 volts = 1·65 volts, i.e.

potentiometers set to 0.0165 per unit.

The scaling of the output velocity at (C) will be obtained by considering the value of the viscous drag term, that is 310 lb.ft/rad/sec. 1·65 volts at (C) results from 1·65 volts appearing on the right hand side of the coulomb friction simulation which represents a useful torque of 150 lb.ft.

Therefore at (C) 1·65 volts = 150/310 rads/sec.

= 0·485 rads/sec.

At (C) 1 volt = 0·293 rads/sec.

The scaling at every other point may be calculated in a similar manner.

Example of Digital Simulation

The following set of differential equations is associated with a spacecraft re-entry problem.

$$V' = 9{\cdot}295\cos\alpha - 32{\cdot}2\sin\gamma$$
$$-0{\cdot}00056022\ \{0{\cdot}129+0{\cdot}0151632\ (0{\cdot}965+5{\cdot}1\ \alpha)^2\}V^2$$

$$\begin{aligned}
\gamma' &= 9{\cdot}295 \sin \alpha - 32{\cdot}2 \cos \gamma + 0{\cdot}00056022\ (0{\cdot}965 + 5{\cdot}1\ \alpha)\ V^2 \\
y' &= -0{\cdot}00009421\ (0{\cdot}215y + 0{\cdot}44\ \alpha - 0{\cdot}026)\ V^2 \\
\theta' &= y \\
\alpha &= \theta - \gamma
\end{aligned}$$

with the initial conditions:
$V = 200$, $y = -0{\cdot}0204$, $\gamma = 0$, and $\theta = 0{\cdot}0525$ at $t = 0$.

The derivation of these equations which represent the physical laws governing re-entry need not concern us. A simulation of this particular re-entry would be intractable by electronic analogue computer solely because of their inherently low absolute accuracy, caused by noise and other imperfections. This would cause severe problems in the calculation of α say, which is the difference of two small numbers. On the other hand there are no real problems for the digital simulator in this case.

A complete problem definition may be written down directly in the SPADE language (see Appendix 1) as follows:

```
Y1  = 200
Y3  = −0·0204
Y4  = 0·0525
C5  = 9·295
C6  = 32·2
D1  = COS(X6)*C5−SIN(Y2)*C6−X9*(0·129+0·051632*X7*X7)
D2  = (SIN(X6)*C5−COS(Y2)*C6+X9*X7)/Y1
D3  = −0·00009421*X8*(0·215*Y3−0·026+0·44*X6)
D4  = Y3
X6  = Y4−Y2
X7  = 0·965+5·1*X6
X8  = Y1*Y1
X9  = 0·00056022*X8
*
TTLE RE-ENTRY PROBLEM
INTG KMVR
TIME TMAX = 30, DELT = 0·1
PRNT Y1, Y2, Y3, Y4
*
```

where:
$D1 \equiv V'$
$D2 \equiv \gamma'$
$D3 \equiv y'$
$D4 \equiv \theta'$
$X6 \equiv \theta - \gamma \equiv \alpha$

The program prints the date, time, and a copy of the input statements, and then tabulates the values of model time and the output variables, with a title at the head of each page, see Table 2. The output variables may also be plotted with the aid of the CALCOMP plotter (Figs. 10 and 11).

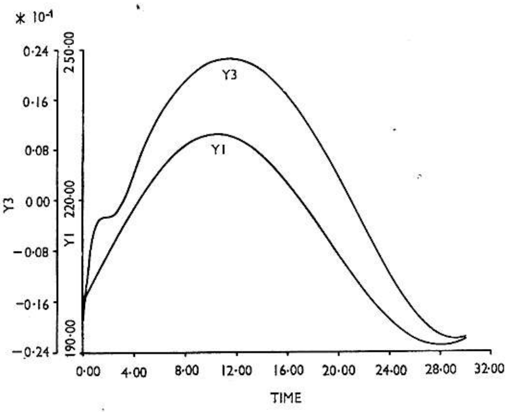


Fig. 10. Re-entry problem Y1 and Y3.

TABLE 2

SPADE ANALOGUE SIMULATION PROGRAM
DATE: 29/08/72 TIME: 16/08/06

PROBLEM INPUT STATEMENTS

```
00000010  Y1 = 200
00000020  Y3 = −0·0204
00000030  Y4 = 0·0525
00000040  C5 = 9·295
00000050  C6 = 32·2
00000060  D1 = COS(X6)*C5−SIN(Y2)*C6−X9*(0·129+0·051632*X7*X7)
00000070  D2 = (SIN(X6)*C5−COS(Y2)*C6+X9*X7)/Y1
00000080  D3 = 0·00009421*X8*(0·215*Y3−0·026+0·44*X6)
00000090  D4 = Y3
00000091  X6 = Y4−Y2
00000092  X7 = 0·965+5·1*X6
00000093  X8 = Y1*Y1
00000094  X9 = 0·00056022*X8
00000100  *
00000110  TTLE RE-ENTRY PROBLEM
00000120  INTG KMVR
00000130  TIME TMAX = 30, DELT = 0·1
00000140  PRNT Y1, Y2, Y3, Y4
00000150  *
```

RE-ENTRY PROBLEM

TIME	Y1	Y2	Y3	Y4
0·0000E 00	2·0000D 02	0·0000D 00	−2·0400D−02	5·2500D−02
1·0000E−01	2·0047D 02	−2·0073D−03	−1·7762D−02	5·0594D−02
2·0000E−01	2·0093D 02	−3·9336D−03	−1·5350D−02	4·8940D−02
3·0000E−01	2·0141D 02	−5·7688D−03	−1·3173D−02	4·7516D−02
4·0000E−01	2·0188D 02	−7·5050D−03	−1·1233D−02	4·6298D−02
5·0000E−01	2·0236D 02	−9·1360D−03	−9·5297D−03	4·5261D−02
6·0000E−01	2·0284D 02	−1·0658D−02	−8·0560D−03	4·4384D−02
7·0000E−01	2·0332D 02	−1·2067D−02	−6·8016D−03	4·3643D−02
8·0000E−01	2·0380D 02	−1·3364D−02	−5·7532D−03	4·3017D−02
9·0000E−01	2·0429D 02	−1·4548D−02	−4·8946D−03	4·2486D−02
1·0000E 00	2·0477D 02	−1·5621D−02	−4·2078D−03	4·2032D−02
1·1000E 00	2·0525D 02	−1·6586D−02	−3·6736D−03	4·1639D−02
1·2000E 00	2·0573D 02	−1·7446D−02	−3·2718D−03	4·1293D−02
1·3000E 00	2·0622D 02	−1·8205D−02	−2·9823D−03	4·0981D−02

and so on.

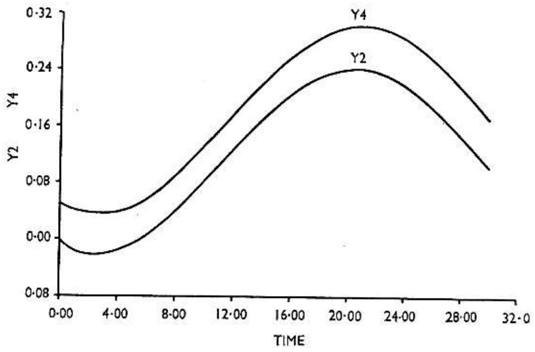


Fig. 11. Re-entry problem Y2 and Y4.

Concluding Remarks

It has been demonstrated that both digital and analogue simulations have certain advantages over each other in certain respects. However the versatility of the digital computer has made it increasingly common and accessible, and it is basically for this reason that digital simulation is often the more cost effective of the alternatives.

References

1 John von Neumann: "First Draft of a Report on the EDVAC", Moore School of Electrical Engineering, University of Pennsylvania, June 30th, 1945 (unpublished).

2 W. S. McCulloch and W. Pitts: "A logical Calculus of the Ideas Immanent in Nervous Activity", *Bulletin of Mathematical Biophysics* 5, pp. 115–133, 1943.

3 T. B. M. Neill: "Improved Method of Analysing Non-Linear Electrical Networks", *Electronic Letters* 5, pp. 13–15, 1969.

4 A. M. Turing: "On Computable Numbers with an Application to the Entscheidungsproblem", *Proc. London Math. Soc.*, Vol. 42, pp. 230–265, 1936.

APPENDIX 1

THE SPADE SIMULATION PROGRAM

SPADE is a program which simulates a continuous system by evaluating the time response for a given set of initial conditions and inputs.

SPADE simulation can be applied to a wide variety of problems. Typical examples are, (i) in control engineering, a study of the effectiveness of a control system, (ii) in applied mathematics or mechanical engineering, a solution of the response of a system or model for which the equations of motion can be written down but not solved in closed form, or (iii) in mathematics, the solution of the initial value problem for a set of ordinary differential equations. SPADE can slso be used to evaluate the transient response of an electrical network after reduction of the problem to the equivalent differential equations.

The SPADE input language enables data to be prepared directly from either a block diagram or the differential equations. The system is represented by a set of structure statements (similar to FORTRAN assignment statements) which specify the functional blocks and their interconnections as well as the initial conditions.

Provision is made for *delay* and *hysteresis* and such analogue devices as *integrators* and *limiters* together with the usual FORTRAN functions including sine, cosine, exponential and absolute value; and additional SPADE devices such as *random number generators* and *arbitrary non-linear functions* are also available. Structure statements are built up using these functions and SPADE variables and operators in a free format; the statements may be written in any order (with the exception of certain initial conditions) since the program sorts them into a logically correct sequence before execution.

SPADE control statements specify the time step, the total run time and which of the variables are to be printed, as well as various other quantities. The values of the output variables are tabulated with the current value of time at each time step, and separate programs are available which can process this output information and (i) plot selected variables on the CALCOMP graph plotter in a format specified by the user, or (ii) transfer selected variables via a disc to the interactive displays of the Marconi Myriad computer for processing by MIDAS (Myriad Interactive Data Analysis System).

SPADE provides a range of integration procedures, from simple rectangular or trapezoidal integration to complex predictor-corrector methods with automatic selection of order and time step to meet a specified error criterion. The variable step Kutta-Merson method is the most generally useful but in some cases a simpler or more complicated method gives better results. A special method suitable for stiff systems is also available.

The Spade Input Language

The input statements required by the SPADE program are written in two sections, each terminated by a line (card) containing a single asterisk. The first section contains the structure of the system to be simulated and the second section contains the control statements to specify parameters for the run.

Structure Statements

The SPADE structure statements are a set of assignments which when executed in the correct order calculate the derivatives of the dependent variables with respect to the independent variable for given values of the dependent and independent variables.

The statements are built up from SPADE operands, operators and functions, as listed below.

Spade Operands

Dependent variables are designated by Y*n*.†

Assignments to these variables are taken as initial values, and evaluated once only, before commencement of the run. Any subset of these variables may be used and any variable to which an initial value is not assigned is initialised to zero.

Derivatives of the dependent variables are designated by D*n*.

These are the derivatives of the corresponding Y variables with respect to the independent variable. (D27 = (d/dt)Y27). For each Y used the corresponding D must be assigned.

The *independent variable* is designated by T or TIME.

Auxiliary variables are designated by X*n*.

These variables are used to simplify the coding, avoid the repetition of sub-expressions and also to allow the values of sub-expressions to be available as output.

Assigned constants are designated by C*n*.

These are calculated once only and then used in other expressions. For example, if the value of π is required the function arc-tangent (ATN) can be used, for example C5 = 4*ATN(1). Constant assignment expressions and expressions assigning initial values to Y variables may only contain constants and those assigned constants which were assigned earlier in the data. This is because constant assignments are not sorted before execution.

† *n* is a positive integer.

For constants any valid FORTRAN number may be used provided that it contains no internal spaces.

Spade Operators

The following operators shown in Table 3 may be used; alternatives to the conventional symbols for "or", "and" and "not" are given because these symbols cannot be entered from a teletype.

TABLE 3

(	left parenthesis	¬	not
)	right parenthesis	?	not
,	comma	>	greater than
:	equivalent	<	less than
#	implies	+	plus
=	equal	—	minus
|	or	*	times
@	or	/	divide
∧	and	%	divide with overflow check
&	and	!	to the power of

Spade Functions

Each SPADE function is represented by a three letter code such as SIN, COS. Most of the usual FORTRAN functions are provided (note that the name is not the same in cases where the FORTRAN name is longer than three letters) as well as special functions with application to simulation. The simple functions are listed in Table 4.

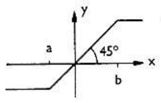
TABLE 4

SIN	sine	INT	integer part
COS	cosine	SQT	square root
ATN	arc-tangent	LOG	natural logarithm
TAN	tangent	DBS	decibels (20 $\log_{10}$)
EXP	exponential	TNH	hyperbolic tangent
ABS	absolute value	SGN	sign (—1, 0, +1)

The mathematical definitions of the special SPADE functions are given below:

Limiter (two parameters)

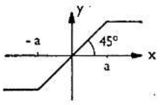
$y = $ LIM (x, a, b): $\quad y = a,\ x < a;$
$\quad y = x,\ a \leq x \leq b;$
$\quad y = b,\ x > b.$



Limiter (one parameter)

$y = \mathrm{LM1}\ (x, a)$:

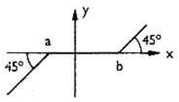
$y = -a,\ x < -a;$
$y = x,\ -a \le x \le a;$
$y = a,\ x > a.$



Dead zone (two parameters)

$y = \mathrm{DZN}\ (x, a, b)$:

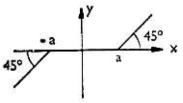
$y = x - a,\ x < a;$
$y = 0,\ a \le x \le b;$
$y = x - b,\ x > b.$



Dead zone (one parameter)

$y = \mathrm{DZ1}\ (x, a)$:

$y = x + a,\ x < -a;$
$y = 0,\ -a \le x \le a;$
$y = x - a,\ x > a.$



Slope (Acceleration) Limiter

$y = \mathrm{SLL}\ (x, a)$:

z = last output, t_0 = last time, $p = a(t - t_0)$;
$y = z - p,\ x < z - p;$
$y = x,\ z - p \le x \le z + p;$
$y = z + p,\ x > z + p.$

Delay
$y = \mathrm{DLY}\ (x, a)$: $y = 0,\ t < a;\ y(t) = x(t - a),\ t \ge a.$

Memory
$y = \mathrm{OLD}\ (x)$: y = last value of x.

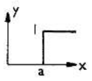
Re-settable flip-flop

$y = \mathrm{RST}\ (a, b, c)$:

z = last output;
$y = 0,\ a > 0;$
$y = 1,\ a \le 0,\ b > 0;$
$y = 0,\ a \le 0,\ b \le 0,\ c > 0,\ z = 1;$
$y = 1,\ a \le 0,\ b \le 0,\ c > 0,\ z = 0;$
$y = 0,\ a \le 0,\ b \le 0,\ c \le 0,\ z = 0;$
$y = 1,\ a \le 0,\ b \le 0,\ c \le 0,\ z = 1.$

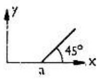
Step Input
$y = \mathrm{STP}\ (a)$:

$y = 0,\ t < a;$
$y = 1,\ t \ge a.$

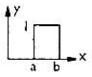


Ramp Input
$y = \mathrm{RMP}\ (a)$:

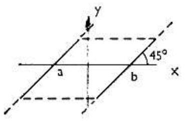
$y = 0,\ t < a;$
$y = t - a,\ t \ge a.$



Pulse Input
$y = \mathrm{PLS}\ (a, b)$: $y = 1,\ a \leqslant t \leqslant b$;
$y = 0$ otherwise.



Hysteresis
$z =$ last output;
$y = \mathrm{HYS}\ (x, a, b)$: $y = x-a,\ x<z+a$;
$y = x-b,\ x>z+b$;
$y = z,\ z+a \leq x \leq z+b$.

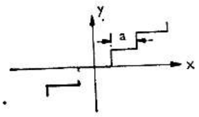


Function Switch
$y = \mathrm{IFF}\ (x, a, b, c)$: $y = a,\ x<0;\ y = b,\ x = 0;\ y = c,\ x>0$.

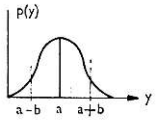
Input Switch
$y = \mathrm{SWT}\ (x, a, b)$: $y = a,\ x \leq 0;\ y = b,\ x>0$.

Arc Tangent with two Arguments
$y = \mathrm{AT2}\ (x_1, x_2)$: $y = \arctan\ (x_1/x_2)$.

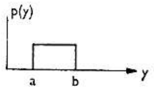
Quantizer
$y = \mathrm{QNT}\ (x, a)$: $y = ka,\ (k-\frac{1}{2})\ a<x \leq (k+\frac{1}{2})a$;
$k = \ldots\ -3, -2, -1,\ 0,\ 1,\ 2,\ 3\ \ldots$



Noise Generator with Normal Distribution
$y = \mathrm{GSS}\ (a, b)$:
Normal distribution of variable y with mean a and standard deviation b.



Noise Generator with Uniform Distribution
$y = \mathrm{RAN}\ (a, b)$:
Uniform distribution of variable y between a and b.



Maximum
$y = \mathrm{MAX}\ (x_1, x_2, x_3, \ldots x_n)$: $y =$ largest argument, n is arbitrary.

Minimum
$y = \mathrm{MIN}\ (x_1, x_2, x_3, \ldots x_n)$: $y =$ smallest argument, n is arbitrary.

Linear Interpolation on Arbitrary Set of Data with Index n
$y = \mathrm{AFN}\ (x, n)$.

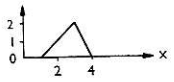
The data for the function are given in the form:

$$An = x_1, y_1, x_2, y_2, x_3, y_3 \ldots\ldots x_k, y_k$$

The x and y values are the data points which define the function arranged in order of increasing x. Continuation lines are indicated by a comma after the last number on the preceding card.

Example:

$$A1 = 0, 0, 1, 0, 2, 1, 3, 2, 4, 0, 5, 0$$



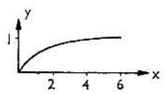
Aitken Interpolation of Order m *on Arbitrary Set of Data with Index n*

$$y = \text{GFN}(x, n, m)$$

The data are presented in the same form as for AFN (x, n)

Example:

$$G5 = 0, 0, 1, 0{\cdot}5, 2, 0{\cdot}75, 3, 0{\cdot}90, 4, 0{\cdot}97, 5, 0{\cdot}99, 6, 1$$



Some care is required in deciding whether to use linear or higher order interpolation. Although the higher order method would seem at first to be more accurate this is only true for smooth curves. Linear interpolation gives much more reliable results for a function with discontinuities.

Operation Hierarchy

Parentheses are used to specify that the expressions within these are to be evaluated first, in the same way as normal algebraic or FORTRAN notation. Where it is not overruled by parentheses the order in which operations are performed is as follows:

Operation	*Hierarchy*
Function evaluation (SIN, COS)	1st
Exponentiation (!)	2nd
Multiplication and division (*, /, %)	3rd
Addition and subtraction (+, −)	4th
Comparison of value (>, =, <)	5th
Not (¬, ?)	6th
And (∧, &)	7th
Or (|, @)	8th
Implies (#)	9th
Equivalent (:)	10th

Operators with the same hierarchy are performed from left to right, thus $a*b/c$ is evaluated as $(a*b)/c$.

APPENDIX 2

FAST FOURIER TRANSFORMS IN NON LINEAR SYSTEM SIMULATION

The MODSIM program can simulate non-linear systems which can be described by an equation of the form:

$$Z(D)\mathbf{v}(t) = \mathbf{v}_i(t) + K\int_0^t H(t-\tau)\mathbf{f}[\mathbf{v}(\tau)]d\tau,$$

where $Z(D)$ is a matrix linear differential operator of order n,
$H(\tau)$ is a matrix of impulse responses of order n,
$\mathbf{v}$ and $\mathbf{v}_i$ are n-dimensional column vectors,
$\mathbf{f}$ is an n-dimensional function of an n-vector,
K is a scalar,
and n is an integer > 0.

This is done by Fourier transforming to:

$$Z(\omega)\mathbf{V}(\omega) = \mathbf{V}_i(\omega) + KH(\omega)\mathscr{F}\mathbf{f}[\mathscr{F}^{-1}\,\mathbf{V}(\omega)],$$

where $\mathscr{F}$ is the Fourier transform operator.

The Picard iteration:

$$\mathbf{V}_{k+1}(\omega) = Z^{-1}(\omega)\,\mathbf{V}_i(\omega) + Z^{-1}(\omega)\,KH(\omega)\,\mathscr{F}\mathbf{f}[\mathscr{F}^{-1}\mathbf{V}_k(\omega)], \qquad (4)$$

with

$$\mathbf{V}_0(\omega) = Z^{-1}(\omega)\,\mathbf{V}_i(\omega),$$

is then performed numerically until $\mathbf{V}_{k+1}$ differs by less than a desired amount from $\mathbf{V}_k$. This is achieved by using the Fast Fourier Transform as an approximation to the Fourier Transform $\mathscr{F}$(3). The iteration, equation (4), will not always converge, however it is always possible to modify the original equation so that convergence will occur. The above method is a very fast and useful means of finding the frequency characteristics of a mildly non-linear system, and hence of simulating such a system.