AAEC/S16

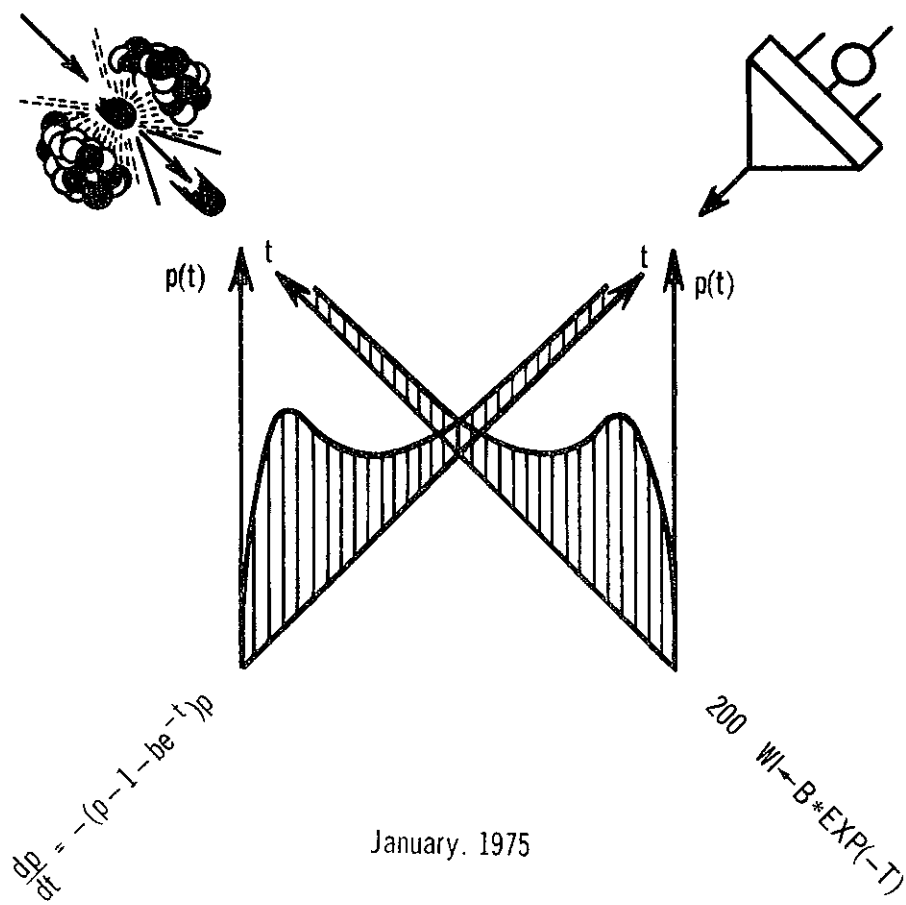
AUSTRALIAN ATOMIC ENERGY COMMISSION
RESEARCH ESTABLISHMENT
LUCAS HEIGHTS

REACTORS, MATHEMATICS AND COMPUTERS

SUMMER SCHOOL

Lectures by

R. P. BACKSTROM
J. M. BARRY
B. E. CLANCY
C. P. GILBERT
D. B. McCULLOCH



January. 1975

AUSTRALIAN ATOMIC ENERGY COMMISSION
RESEARCH ESTABLISHMENT
LUCAS HEIGHTS

REACTORS, MATHEMATICS AND COMPUTERS

SUMMER SCHOOL

Lectures by

R.P. BACKSTROM
J.M. BARRY
B.E. CLANCY
C.P. GILBERT
D.B. McCULLOCH

CONTENTS

CHAPTER 1

PHYSICS OF REACTOR KINETICS

Lecture by

D.B. McCULLOCH

CONTENTS

1.1 NEUTRON CHAIN REACTIONS

1.1.1 Fission

Because of its zero electrical charge, the fundamental particle, the NEUTRON is very favourably placed to interact with atomic nuclei, even in the case of very heavy ones (high atomic weight, A), with large electrical charge (Z).

Some elements high in the Periodic Table, particularly uranium, are capable of interacting with a neutron in such a way that the nucleus splits (or 'fissions') into two more or less equal parts (fission products), with the liberation of a number of further neutrons, and a significant quantity of energy (Figure 1).

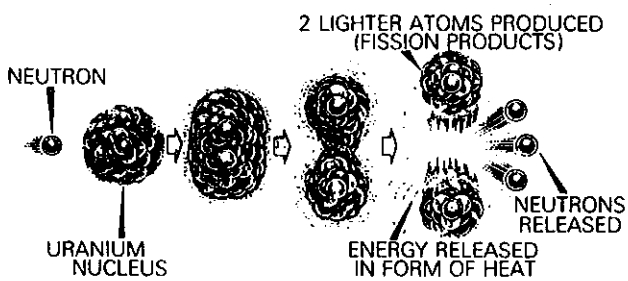


FIGURE 1 Fission of Uranium

The energy release appears because the mass of the two resulting fission product nuclei and the liberated neutrons is in total slightly less than the mass of the original neutron plus target nucleus. The energy equivalent, E, of this mass difference m is given by Einstein's relationship

$$E = mc^2 ,$$

and mostly takes the form of kinetic energy of the fission fragments and neutrons, subsequently appearing as heat as these particles are slowed down in the bulk fissioning material. Some of the energy appears also as gamma rays.

The neutrons liberated in fission arise because stable nuclear configurations for elements at the high end of the periodic table favour a higher neutron-to-proton ratio than is generally required for elements lower down, resulting in a neutron surplus when fission product nuclei are formed.

In addition to the neutrons which are 'boiled off' at the instant of fission, some of the fission product nuclei formed still have too many neutrons to be stable and subsequently emit them by a radioactive decay process with half lives ranging from a few tenths to a few tens of seconds.

These are known as 'delayed neutrons' and are of the order of one per cent of the number released directly ('prompt' neutrons) in the fission process. As we shall see later, they have an extremely important part to play in the dynamic behaviour and the control of nuclear fission reactors.

The energy released in a single fission is about 200 million electron volts. A fission rate of 3×10^{10} per second therefore releases energy at approximately 1 joule per second, *i.e.* a power of 1 watt. This may sound very little, but it should be looked at in the light that complete fissioning of 1 gram of a heavy element would release approximately 1 megawatt day of energy, *i.e.* sufficient to run 1000 - 1 kilowatt electric radiators continuously for 24 hours! Compare this with the energy release for any energy producing chemical process, *e.g.* burning of coal and estimate the equivalent quantities of fuel material required.

To induce fission, a neutron must first be absorbed into the target nucleus. Usually, the probability of absorption for slowly moving neutrons is greater than that for fast neutrons. However, energy considerations inside the compound nucleus formed when the neutron is absorbed, may favour other processes than fission, unless the neutron brings with it at least a certain minimum, or 'fission threshold' energy.

An element is identified by the number of protons in, and hence the electric charge of, its nucleus. In some elements, these protons may be associated with different numbers of neutrons, giving rise to species of the same element having different atomic weights. These are known as *isotopes*.

Of the naturally occurring heavy elements, only the light isotope of uranium $^{235}U_{92}$ undergoes fission with low energy neutrons. This isotope is present to about 0.7 per cent by weight in natural uranium. The abundant uranium isotope $^{238}U_{92}$ (~99.3 per cent), and also $^{232}Th_{90}$ undergo fission only with energetic (fast) neutrons ($E_n \simeq 1$ MeV, $v_n \simeq 10^4$ km s^{-1}).

Uranium can be artificially processed to yield some fractions which contain higher and some which contain lower than natural proportions of the ^{235}U isotope. The former material is favourable to fission reactions, is known as 'enriched uranium', and is widely used as a fuel in nuclear power reactors.

It is because fission is induced by neutrons, and is accompanied by the release of further neutrons, that the possibility of a continuing chain fission reaction exists (Figure 2). Naturally occurring fission chain reactions have not consumed all naturally occurring uranium because processes other than fission compete with fission for the neutrons released by

fission. We shall now examine briefly some of these other processes.

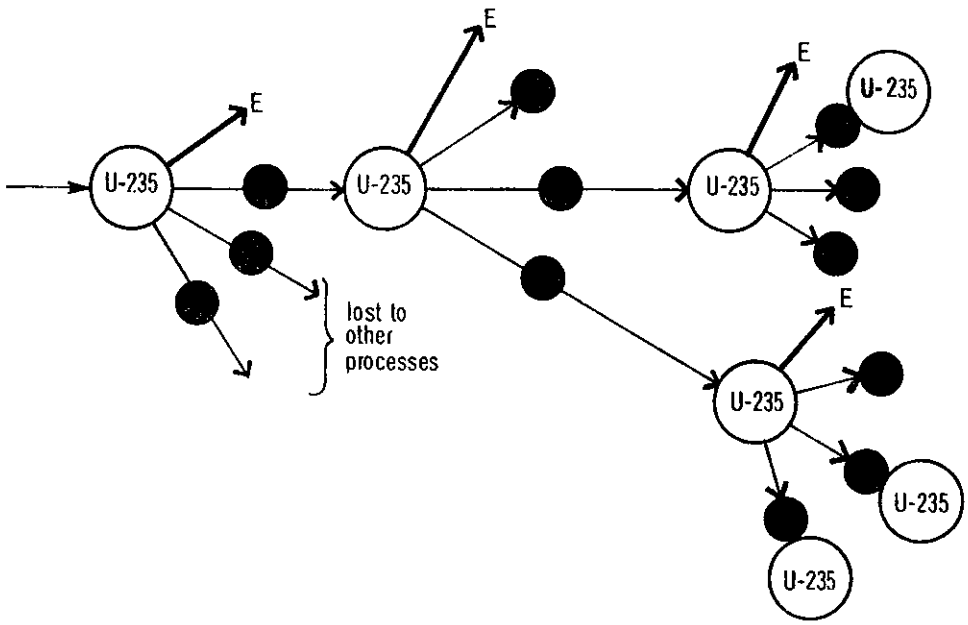


FIGURE 2 Neutron Fission Chain Reaction

1.1.2 Some Other Neutron Interactions with Matter

Elastic Scattering

In this type of interaction, both neutron and interacting nucleus behave rather like hard spheres or billiard balls. Energy and momentum are exchanged essentially as given by the laws of classical mechanics, depending on the mass of the target nucleus and the angle of impact. Successive collisions of this type in *moderating materials* (light atoms of low absorption cross section) are used in thermal neutron reactors to slow neutrons down from the energies at which they are born in fission (max ~ 10 MeV, average ~ 2 MeV) until they approach thermal equilibrium with the molecules of the reactor materials (~ 0.025 eV at room temperature).

Absorption Processes

Inelastic scattering

This process occurs mostly at fairly high neutron energies in interactions with heavier nuclei. It involves absorption into the nucleus of a neutron with energy El, and its re-emission at a lower energy E2, accompanied by a gamma-ray, which carries off the balance of the energy. This process is very effective in transferring neutrons at fission energies to below the

threshold (~ 0.8 MeV) where they would be capable of causing fission in ^{238}U.

Capture

This covers a variety of processes in which a neutron is captured to form either a stable nucleus or one which decays by emission of charged particles and/or γ-rays to give a new product nuclide. The decay may be essentially instantaneous, as for example

$$^{10}B_5(n,\alpha)^{7}Li_3 \ ,$$

which is extensively used in neutron detectors, or may take place with any half life, *e.g.*

$$^{23}Na_{11} + n \rightarrow {}^{24}Na_{11} \xrightarrow[\tau_{\frac{1}{2}} = 15h]{\beta^-,\ \gamma} {}^{24}Mg_{12} \ ,$$

Capture reactions are extensively used in nuclear reactors (a) in the form of absorbing 'control rods' for direct trimming of the fission reaction rate, or for shut-down, and (b) as fillings or coatings of detectors to monitor the neutron flux level.

1.1.3 Neutron Detectors and Reactor Instrumentation

The neutron flux level is of great importance in a reactor, since it determines the rate of fission, and so the power produced in the system.

Because neutrons carry no charge, they are not detected directly, but their presence is deduced from the ionisation produced by the secondary charged particles or γ-rays arising from their interaction with some detecting nucleus.

Detectors fall into two types which are now described.

Activation Detectors

Here a material, usually in the form of foil or wire, giving rise to a radioactive decay species of suitable extended half life is irradiated. Subsequently, it is removed from the reactor and its radioactivity measured. This is an indication of the total neutron dose (that is, time integrated neutron flux) received by the detector at its location in the reactor.

Instantaneous Indicating Detectors

These are designed to monitor the neutron flux level continuously and make available a read-out of its value at any time. They may be designed to operate in either

(i) pulse mode, or

(ii) current mode.

In either case, the ionisation caused by the radiation arising from neutron

fission. We shall now examine briefly some of these other processes.

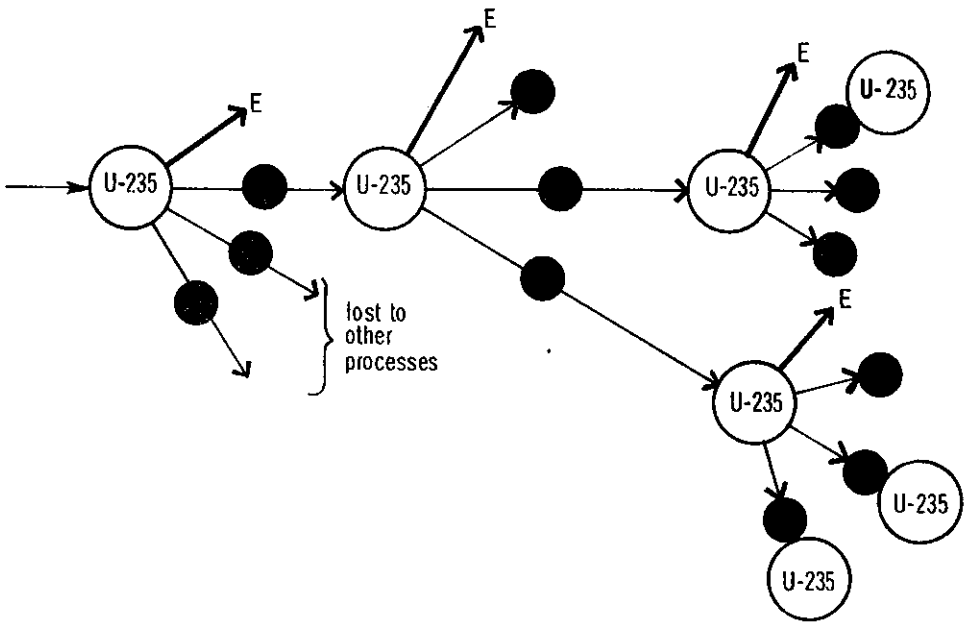


FIGURE 2 Neutron Fission Chain Reaction

1.1.2 Some Other Neutron Interactions with Matter

Elastic Scattering

In this type of interaction, both neutron and interacting nucleus behave rather like hard spheres or billiard balls. Energy and momentum are exchanged essentially as given by the laws of classical mechanics, depending on the mass of the target nucleus and the angle of impact. Successive collisions of this type in *moderating materials* (light atoms of low absorption cross section) are used in thermal neutron reactors to slow neutrons down from the energies at which they are born in fission (max ~ 10 MeV, average ~ 2 MeV) until they approach thermal equilibrium with the molecules of the reactor materials (~ 0.025 eV at room temperature).

Absorption Processes

Inelastic scattering

This process occurs mostly at fairly high neutron energies in interactions with heavier nuclei. It involves absorption into the nucleus of a neutron with energy El, and its re-emission at a lower energy E2, accompanied by a gamma-ray, which carries off the balance of the energy. This process is very effective in transferring neutrons at fission energies to below the

threshold (~ 0.8 MeV) where they would be capable of causing fission in ^{238}U.

Capture

This covers a variety of processes in which a neutron is captured to form either a stable nucleus or one which decays by emission of charged particles and/or γ-rays to give a new product nuclide. The decay may be essentially instantaneous, as for example

$$^{10}B_5(n,\alpha)^{7}Li_3 \ ,$$

which is extensively used in neutron detectors, or may take place with any half life, *e.g.*

$$^{23}Na_{11} + n \rightarrow {}^{24}Na_{11} \xrightarrow[\tau_{\frac{1}{2}} = 15h]{\beta^-,\ \gamma} {}^{24}Mg_{12} \ ,$$

Capture reactions are extensively used in nuclear reactors (a) in the form of absorbing 'control rods' for direct trimming of the fission reaction rate, or for shut-down, and (b) as fillings or coatings of detectors to monitor the neutron flux level.

1.1.3 Neutron Detectors and Reactor Instrumentation

The neutron flux level is of great importance in a reactor, since it determines the rate of fission, and so the power produced in the system.

Because neutrons carry no charge, they are not detected directly, but their presence is deduced from the ionisation produced by the secondary charged particles or γ-rays arising from their interaction with some detecting nucleus.

Detectors fall into two types which are now described.

Activation Detectors

Here a material, usually in the form of foil or wire, giving rise to a radioactive decay species of suitable extended half life is irradiated. Subsequently, it is removed from the reactor and its radioactivity measured. This is an indication of the total neutron dose (that is, time integrated neutron flux) received by the detector at its location in the reactor.

Instantaneous Indicating Detectors

These are designed to monitor the neutron flux level continuously and make available a read-out of its value at any time. They may be designed to operate in either

(i) pulse mode, or

(ii) current mode.

In either case, the ionisation caused by the radiation arising from neutron

interaction in the detector material is measured as an electrical output.

In the pulse mode of operation, the discrete packet of electrical charge arising from each detector event is separately processed and counted. The rate of arrival of the charge pulses is then proportional to the neutron flux at the detector position (Figure 3).

In the current mode of operation, the detector acts as a smoother of the electrical pulses from the ionising events to produce an ionisation current which can be measured with a sensitive direct current monitoring instrument (Figure 4).

Both pulse and ionisation current-type equipment are employed extensively in reactor control and safety instrumentation. Pulse equipment is generally preferable for low power operation and current type at high power.

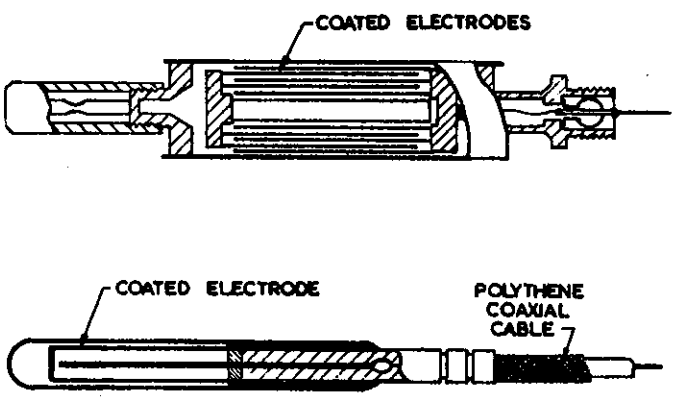


FIGURE 3 *Typical Pulsed-Type Fission Chambers (Schematic)*

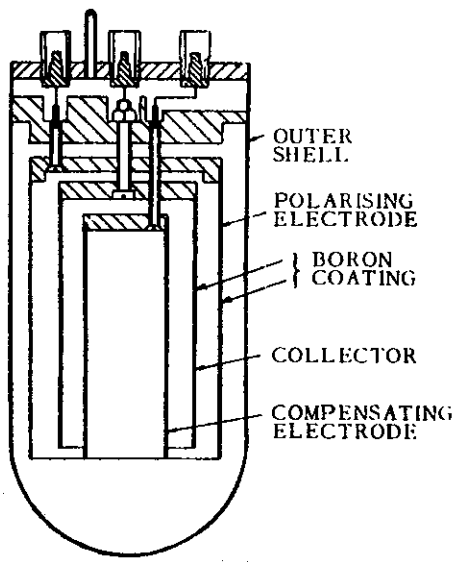


FIGURE 4 *γ-Compensated Current Type Ionisation Chamber (Schematic)*

The MOATA reactor has a pulse channel using a fission detector in which the charged fission product fragments arising from fission in a ^{235}U coating cause ionisation in a gas filling. The ionising events are detected as discrete pulses. The reactor also uses a number of current ionisation chambers, using the $^{10}B_5(n,\alpha)^7Li_3$ reaction discussed in Section 2.2.2. Gas ionisation caused by the resulting α and $^7Li_3$ particles is amplified and registered as an electrical current.

1.1.4 The Fission Chain Reaction

Consider an assembly made up of a number of materials, one of which, *fuel*, is capable of undergoing fission by neutron interaction. The other

materials may include impurities associated with the fuel, *cladding* material to cover the fuel and contain the products formed in fission, a *coolant* material to remove fission heat, a *moderator* material to scatter and so reduce loss of neutrons from the assembly, and to reduce them from the energies at which they are released in fission (*fission neutrons*) towards thermal equilibrium with the moderator atoms (*thermal neutrons*) and *structural* materials forming part of the engineering integrity of the assembly. Such an assembly is potentially a nuclear chain *reactor* (Figure 5).

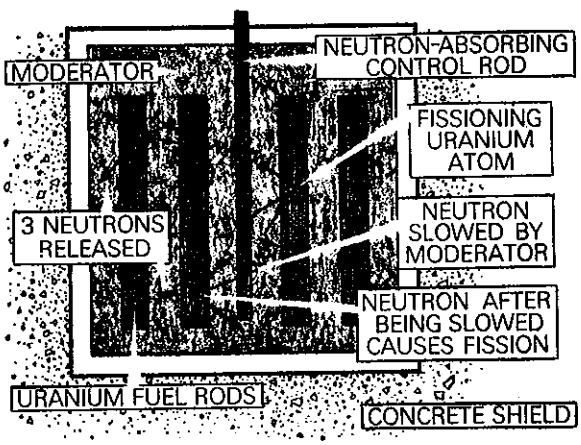


FIGURE 5 Simplified Arrangement of Reactor Showing Chain Reaction

Consider now a fission event within the assembly. On average, $\bar{\nu}$ (approximately 2.5 for ^{235}U) fission neutrons and approximately 200 MeV of energy are released. Of these $\bar{\nu}$ neutrons $\bar{\nu}(1-\beta)$ are released instantaneously and $\beta\bar{\nu}$ over an extended interval following radioactive decay half lives from $\sim$ 0.1 to $\sim$ 55 s. β is called the *delayed neutron fraction*.

The $\bar{\nu}$ neutrons released from an average fission will begin to have collisions with the nuclei of the materials making up the assembly. At each collision there is a probability of

(i) *Elastic scattering* with or without reduction in neutron energy, although on average there will be a steady reduction;

(ii) absorption with re-emission (*inelastic scattering*);

(iii) absorption with formation of a new nuclide and loss of the neutron to the system (*capture*); or

(iv) absorption followed by *fission*.

In continuous competition with all these processes is the probability of being scattered out of and lost to the system (*leakage*).

The probabilities of each of the processes above are dependent on the energy of the neutron and the type of nucleus with which it collides, and are called *cross sections* for the various processes involved.

It is clear from the above description that if the competing processes in the assembly are just balanced so that of the $\bar{\nu}$ neutrons released on average in fission, exactly 1 on average is left to cause a further fission, then the neutron population and hence the fission rate or power of the assembly will be constant. This state is called *critical* (Figure 6).

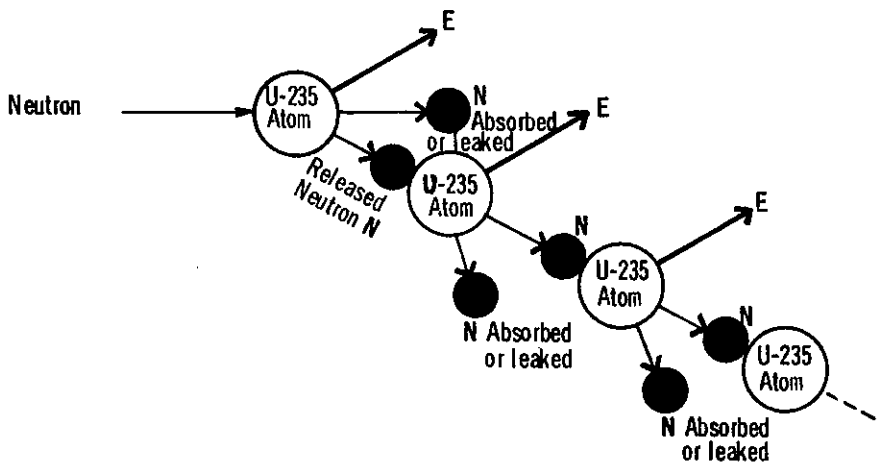


FIGURE 6 Chain Reaction - Critical State

Suppose now that we make some changes to the just critical system, for example, by removing a piece of absorbing material. Now the balance of competing processes is changed and instead of exactly 1 neutron per fission being available to cause a further fission, there will be, say, k where k is a little larger than unity. Consequently, if n_0 neutrons are initially present in the reactor the next generation will have n_0k, the following are n_0k^2, the next n_0k^3, *etc.* That is, the reactor neutron population and hence the fission rate and power level will increase so that in the r^{th} generation $n_r = n_0k^{(r-1)}$. By substituting $t = (r-1)\ell$, where ℓ is the mean lifetime of a neutron generation, then the neutron population, n, after time t, is given by

$$n = n_0k^{t/\ell}$$

whence, taking logarithms (to the base e = 2.718282), we obtain the result

$$n = n_0e^{(\ell n\ k)t/\ell} \qquad (\ell n\ k = \ell og_e k) \quad .$$

Since k normally differs only very slightly from unity, then to a close

approximation

$$\ell n\ k = \ell n[1 + (k-1)] \simeq k-1 ,$$

and then

$$n = n_0 e^{\frac{k-1}{\ell} \cdot t} ,$$

showing that the neutron population or power increases exponentially. (k-1) is sometimes called the *reactivity* of the system and denoted by the symbol, ρ.

This state of the reactor, where the power is diverging exponentially, is called *'super-critical'* (Figure 7). The increase may be arrested and the power maintained at a new constant level simply by making the change necessary to restore k to unity when the required level is reached.

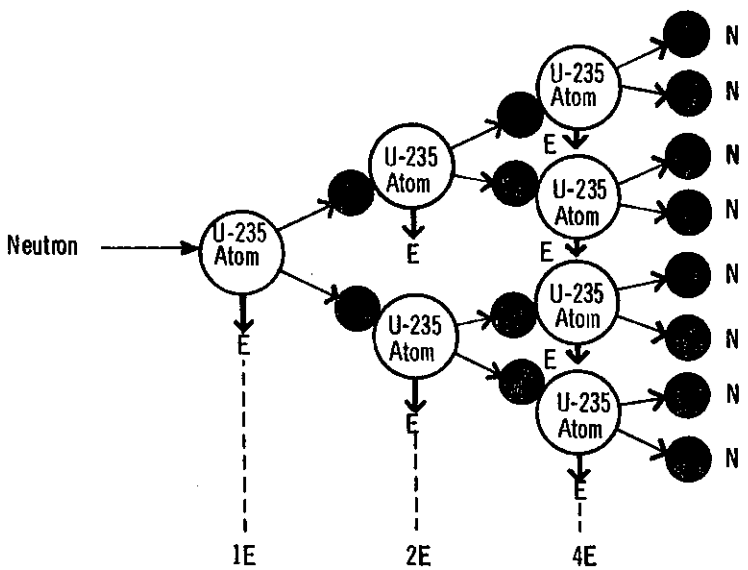


FIGURE 7 Chain Reaction - Supercritical State

By a similar argument to that outlined above for k > 1, if a change is made to the reactor so that k is reduced below unity, the exponent in the neutron population equations becomes negative and we would expect the neutron population to die away exponentially to zero. A reactor in a state for which k < 1 is said to be *subcritical*. In actual reactors, the sub-critical behaviour of the neutron population is very strongly modified by the neutron source which is always present by design or by inherent existence in the reactor materials. With such a source, say of S neutrons per sec, the neutron population in the sub-critical system does not die away to zero, but by following successive generation arguments similar to those above, can be

shown to reach a steady level given by

$$n \propto S(1 + k + k^2 + k^3 + \ldots) = \frac{S}{1-k} \quad .$$

$1/(1-k)$ is now said to be the *multiplication factor* or *multiplication* of the sub-critical reactor.

The behaviour deduced in the preceding two paragraphs for disturbances from the critical state was simplified by neglecting the fact that some of the $\bar{\nu}$ neutrons arising from fission are emitted from fission product fragments at quite long times after the fission occurs. The effect of these delayed neutrons is to slow down very markedly the response of the reactor power to a change in the criticality constant, k. This is very fortunate, since with typical neutron generation life-times ranging from ~ 0.1 μs (10^{-7} seconds) for small fast neutron reactors to ~ 1 ms (10^{-3} seconds) for a large thermal neutron reactor, the formulae just derived show that power increase rates for even small changes in reactivity could be very rapid indeed, and would give rise to severe control problems.

In qualitative terms, a just critical reactor operating at steady power level with neutron density n_0 can be regarded as having its unity criticality constant made up of a *prompt* contribution $(1-\beta)$ plus a delayed part, β. The fission products giving rise to the delayed neutrons are called the *delayed neutron precursors* and are being produced at a constant rate just equal to their rate of depletion by radioactive decay.

If now the criticality constant is increased to $1 + \delta k$, the $1-\beta$ part increases to $(1-\beta)(1+\delta k)$ and the reactor responds immediately to this as described on page 1.7. The rate of production of delayed neutron precursors increases at once corresponding to the new power level, but the rate at which the additional delayed neutrons are released is governed by the radioactive half lives of the precursors as well as by the term $\beta(1+\delta k)$. The net result is that reactor response to a change in k consists of an initial rapid change due to the prompt neutrons, followed by a slower one governed by the delayed neutrons. This is indicated qualitatively in Figures 8a and 8b for increase and decrease in k respectively.

It follows easily that when k is returned to unity to level off at a new desired power, the neutron population does not respond instantaneously, but continues to change according to its 'memory' of the preceding delayed neutron precursor concentrations. Anticipation by the operator based on his experience is therefore necessary if a new power level is to be approached smoothly and without 'over shoot'.

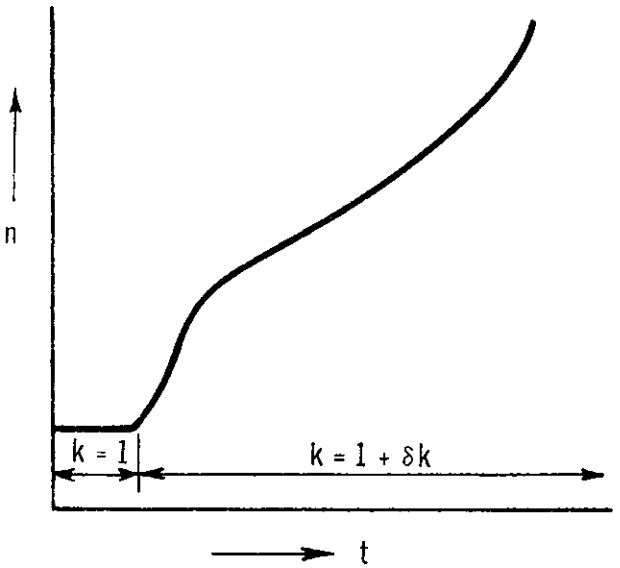


FIGURE 8a Reactor Power Response to Increase in k

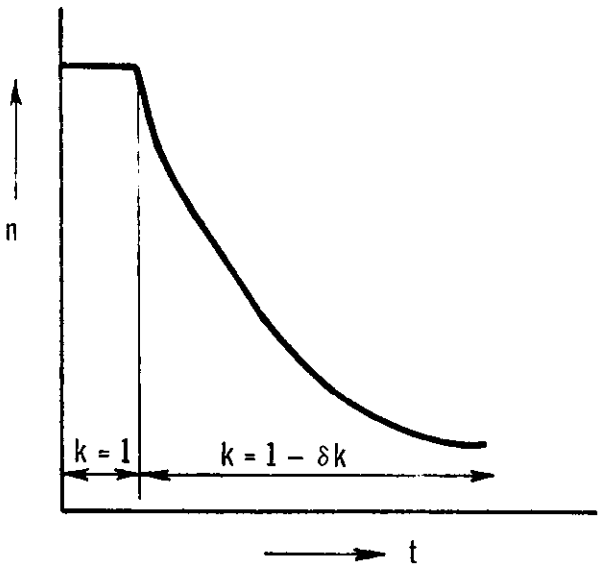


FIGURE 8b Reactor Power Response to Decrease in k

The equations which describe in detail the features of reactor response which we have covered so far are rather cumbersome to handle. Fortunately, they are not necessary for you to understand the physical basis of the reactor problem you will be solving by computer techniques on this course. We must, however, take the above very simplified discussion of reactor behaviour a little further in qualitative terms.

1.1.5 Temperature Effects

Because the competing processes for neutrons in a reactor depend in different ways on the speed of the neutron relative to the nucleus with which it interacts, the balance of the events can be changed by temperature changes occurring as a result of a disturbance from a steady power (critical) condition.

If, then, a disturbance is made so that $k - 1 > 0$, the power will begin to rise. The temperature of the fuel will start to rise, followed more slowly by the coolant, moderator, structural materials, *etc.* If these temperature changes alter the balance of competing neutron processes, such that fission is more favoured, then (k-1) will increase still further, causing the power to rise faster; this will cause a further increase in temperature, (k-1) and so on. This situation is called a *positive temperature or power coefficient of reactivity*; it is clearly unstable and would rapidly lead to disaster if no remedial action were taken. Such a characteristic is usually avoided by design.

If, however, the reactor has a *negative temperature (power) coefficient*

of reactivity, then a positive disturbance to k will be gradually reduced as the temperature rises, and provided the original disturbance is not so great that fuel melting or other damage occurs, the temperature and power will rise until some new higher power level is reached, at which the disturbance is just balanced by the decrease in k due to temperature effects, and again $k \equiv 1$, and the system is just critical at steady power (see Figure 9(a)). This is obviously a self-regulating and stable system which is a desirable feature of real reactors, although the self-regulating characteristics are sometimes deliberately modified in the case of reactors used for generating electricity to enable the reactor power output to follow changing load patterns of the distribution grid.

Because the different reactor components contribute in different amounts to the overall power coefficient of reactivity, and because they heat up at different rates and have different heat capacities, the response for a negative coefficient may not be as simple as described above. For example, the

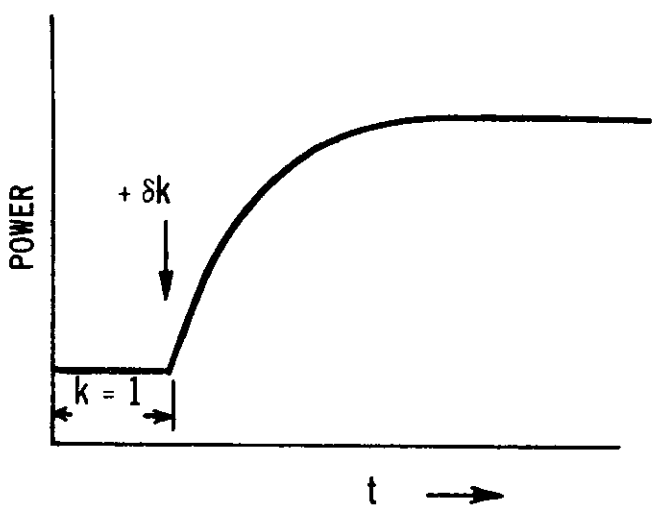


FIGURE 9a

fuel element temperature increases very rapidly, since that is where the heat is mainly produced, and then flows to the moderator or coolant which changes in temperature much more slowly. However, the temperature reactivity coefficient of the fuel is usually smaller than that of the moderator, so that the initial reactivity disturbance is balanced when the moderator has risen only a little in temperature. The power rise is then arrested, but heat

stored in the much hotter fuel elements continues to flow into the moderator, so reducing the reactivity of the system further, and the power falls again as shown in Figure 9(b).

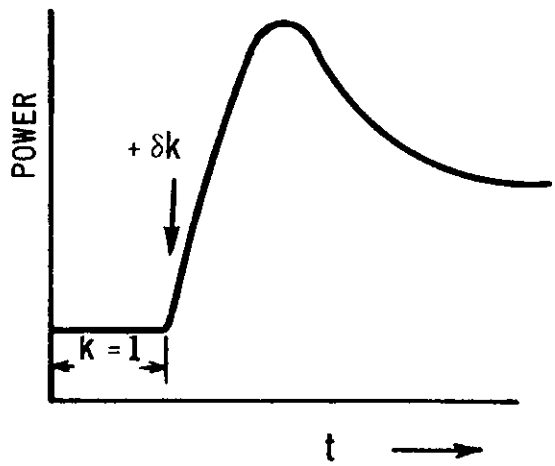


FIGURE 9b

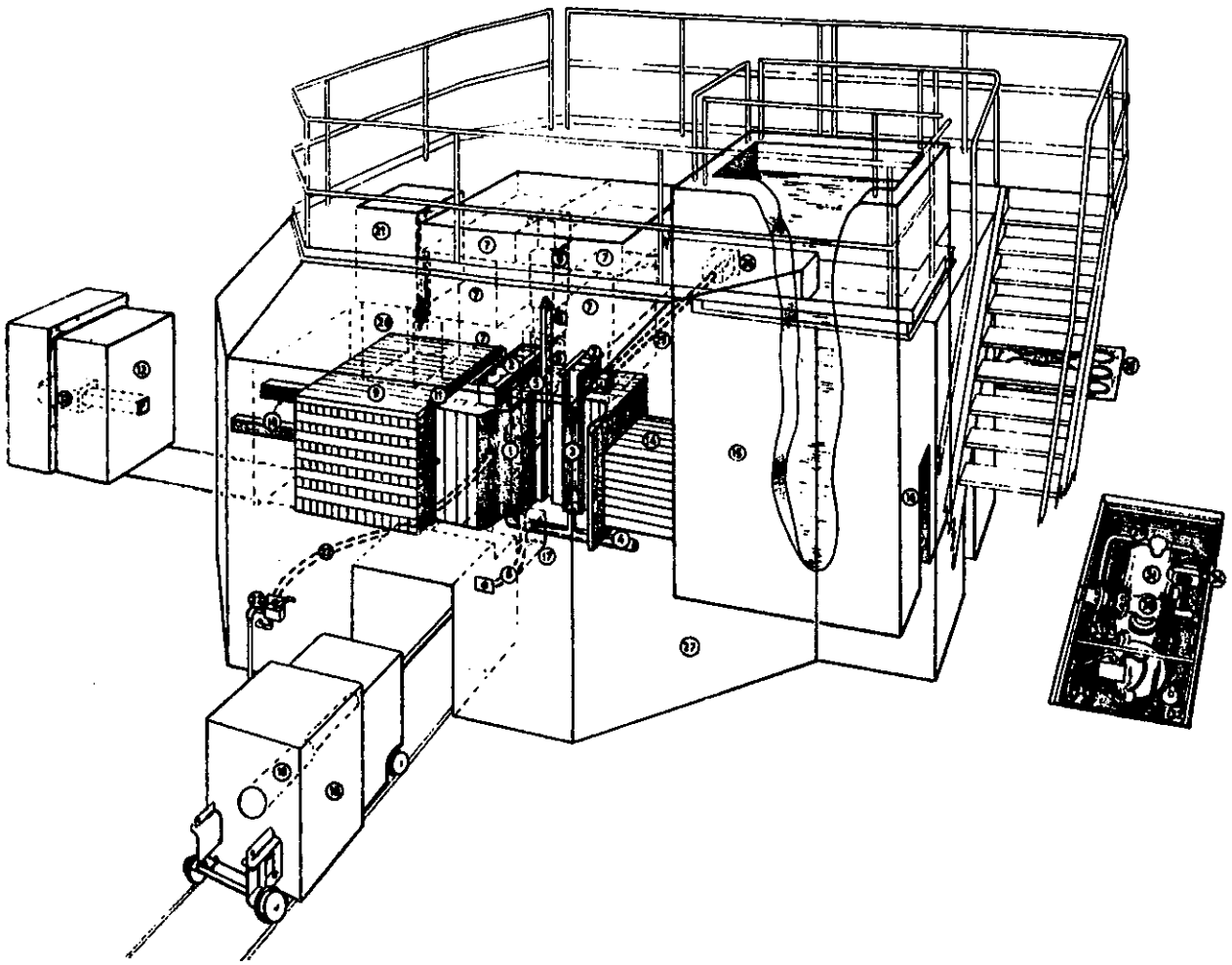
This type of response to a disturbance is not just confined to reactors, but is encountered frequently in the physics of a wide class of systems known as 'negative feedback multiplying systems'. The general response is described by a differential equation which you will find discussed in Chapter 2.1, and which you will be programming for solution by computer methods. Later in the week, you will spend some time looking over the MOATA reactor and seeing a demonstration of its inherent safety following a reactivity disturbance which will give rise to a self-controlling power response, similar to Figure 9(b). The rest of this chapter describes MOATA in sufficient detail to enable you to follow and understand the experimental demonstration you will see.

1.1.6 The MOATA Reactor

MOATA* (Figure 10) is an Argonaut type reactor, based on a design by the Argonne National Laboratory in the United States. Argonaut was intended

---

* *Aboriginal word of North Queensland Ngerikudi tribe, meaning 'firesticks' or 'gentle heat'.*



KEY TO DRAWING

1. GRAPHITE CORE.
2. CORE TANKS.
3. FUEL ELEMENT.
4. MODERATOR/COOLANT INLET.
5. REMOVABLE STRINGERS.
6. CENTRAL STRINGER PLUGS.
7. TOP CLOSURES.
8. NEUTRON SOURCE POSITIONER.
9. THERMAL COLUMN.
10. THERMAL COLUMN STRINGERS.
11. LEAD GAMMA CURTAIN.
12. THERMAL COLUMN DOOR.
13. THERMAL COLUMN DOOR PLUG.
14. SHIELD TANK DUCT.
15. SHIELD TANK.
16. SHIELD TANK OUTER DOOR.
17. HORIZONTAL RADIATION CAVITY.
18. RADIATION CAVITY DOOR.
19. RADIATION CAVITY DOOR PLUG.
20. VERTICAL RADIATION CAVITY.
21. RADIATION CAVITY CLOSURE.
22. PNEUMATIC TUBE.
23. PNEUMATIC TUBE AIR SUPPLY.
24. CONTROL ROD DRUM HOUSING.
25. CONTROL ROD DRIVE SHAFT.
26. CONTROL ROD DRIVE MECHANISM.
27. BIOLOGICAL SHIELD.
28. FUEL & EXPERIMENT STORAGE PIT.
29. PROCESS PIT.
30. DUMP VALVE.
31. DUMP TANK.
32. ION EXCHANGE COLUMN.
33. FLOW RATE REGULATOR.
34. MOTORISED VALVE.

FIGURE 10 The Research Reactor MOATA

originally to be an inexpensive, flexible and safe reactor, suitable for use in universities and similar institutions, the type of core having been the subject of kinetic and transient tests on water-moderated systems. These tests showed this type of core to have strongly self-limiting properties with regard to energy release and power levels reached in reactivity excursions, provided that certain reasonable restrictions on core fuel content are observed. It is therefore an inherently very docile reactor system.

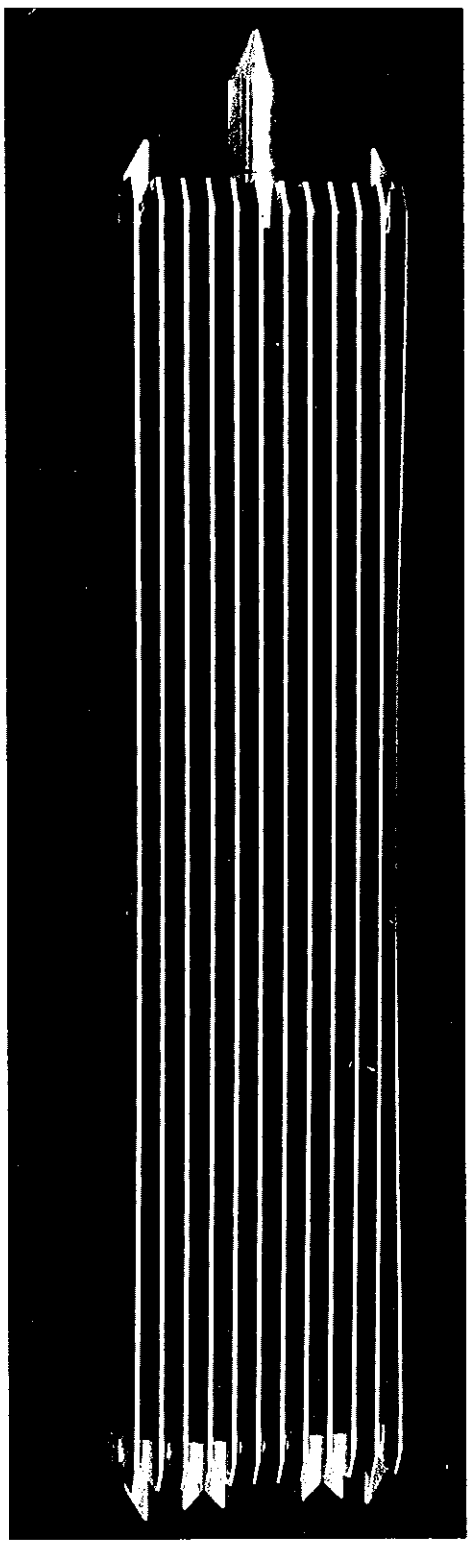


FIGURE 11 A MOATA Fuel Element

The MOATA version of the Argonaut genus can operate at a maximum heat output of 100 kW, when the peak thermal neutron flux is approximately 1.5×10^{12} n cm^{-2} s^{-1}. The reactor is designed to be used for a variety of irradiation experiments and as a source of neutron beams.

MOATA Core

The core of the reactor consists of a 1.3 m cube of graphite into which are set two aluminium core tanks. Six fuel elements are located in each core tank, each fuel element consisting of twelve fuel plates. These fuel plates are of sandwich construction, with a core of uranium-aluminium alloy sheet, clad with pure aluminium. The uranium content is enriched in the thermally-fissile isotope uranium-235 to 90 per cent. Plates are assembled with spacers and bolts into an element as shown in Figure 11. The total core loading is

approximately 3 kg ^{235}U.

Demineralised light water circulates between the fuel plates when the reactor is operating and acts as neutron moderator as well as removing heat generated in the core. Water enters each core tank through a large diameter pipe at its base and leaves at the top.

High-purity nuclear grade graphite acts as an internal neutron reflector between the two core tanks and as an external reflector around the outer surfaces of the tanks. Inserted into the reflector is a one curie, plutonium-beryllium neutron source, which keeps neutron flux measuring instruments on scale when the reactor is shut down. This source can be withdrawn into the biological shield by remote control while the reactor is operating.

Many of the experimental uses of the reactor involve the irradiation of subcritical assemblies, and for this purpose the principal facilities available are cavities into which the assemblies are placed. Six cavities, adjacent to the core, can be provided. Four are situated on the vertical faces of the core, one is situated vertically in the biological shield and one may be formed by removal of the internal reflector graphite from between the core tanks. The four horizontal cavities are filled with graphite at present and thus form thermal columns (regions of relatively pure thermal neutron flux), but the graphite in them is readily removable.

Graphite regions of the reactor contain stringers, which may be removed to provide small irradiation volumes where calibration is carried out of materials used as neutron flux detectors. Access to these stringers and to the cavities, is obtained through plugs or doors in the biological shield, and this can only be done when the reactor is shut down. Rapid access to a high neutron flux region is available by means of the pneumatic tube or 'rabbit', a device similar to the tubes once used in some department stores for sending money to the cashier.

Specimens to be irradiated in the rabbit are inserted in an aluminium tube on the outer face of the biological shield and blown direct to the core by compressed air. Removal is also by compressed air.

Biological Shield

The biological shield is required to protect personnel from intense neutron and gamma radiation associated with the fuel during operation and from high-level, fission-product radiation present even when the reactor is shut down. The MOATA shield consists of layers of heavy concrete immediately surrounding the core, and in the outer faces of the shield. This concrete is made from ilmenite sand (iron and titanium oxides) and steel punchings. The

remainder of the shield volume is filled with ordinary concrete. Total shield thickness is about 2.3 m in any direction from the core.

This shield design is a compromise between two requirements. The core gamma rays require a dense material, such as heavy concrete, in order to be attenuated in a short distance. Nevertheless, neutrons escaping from the core activate any iron in their path, and de-excitation of the iron isotopes produced gives rise to further high energy gamma rays, which in turn require adequate thickness of shielding. Thus any iron, such as that in the heavy concrete, must either be sufficiently shielded on the outer side to remove the iron capture-gammas, or be shielded sufficiently on the core side to absorb and scatter neutrons before they reach the iron.

Plugs and doors in the shield, some of which are rail-mounted, are made either of solid steel, or of steel frames filled with medium density concrete.

The whole-body radiation dose at the shield surface at full power is nowhere greater than a few milliröntgens per hour.

Control System

The control system may be considered in two parts, firstly, the devices used to vary reactor conditions, and secondly, the instruments used to measure reactor conditions. In the first category are the control absorbers or rods, consisting of pieces of neutron-absorbing material arranged to move in a region adjacent to the core tanks. The neutron absorber is boral, a boron carbide-aluminium complex, in the form of sheet, which is welded to stainless-steel spring strip. The spring is wound on a drum mounted on top of the reflector graphite, and the drum is rotated by a shaft passing through the biological shield to a recess on the outer face where the motor and drive mechanisms are housed. A magnetic clutch is interposed between motor and shaft; de-energising of the clutch allows the spring and gravity forces to insert the absorber rapidly to the position of maximum effectiveness. Details of a control rod and its drive mechanism are shown in Figure 12.

MOATA has four such control rods, two being used as safety rods. The latter are fully inserted when the reactor is shut down, and fully withdrawn when it is operating, providing a substantial margin for shutting down whenever necessary. Two rods are used for coarse and fine adjustments when taking the reactor to a critical state. These are, respectively, the shim rod and the regulating rod.

Normal and emergency methods for shutting down the reactor are the same, namely, breaking of magnetic clutch currents and thus fully inserting all rods; by this means rods are inserted in less than half a second. At the

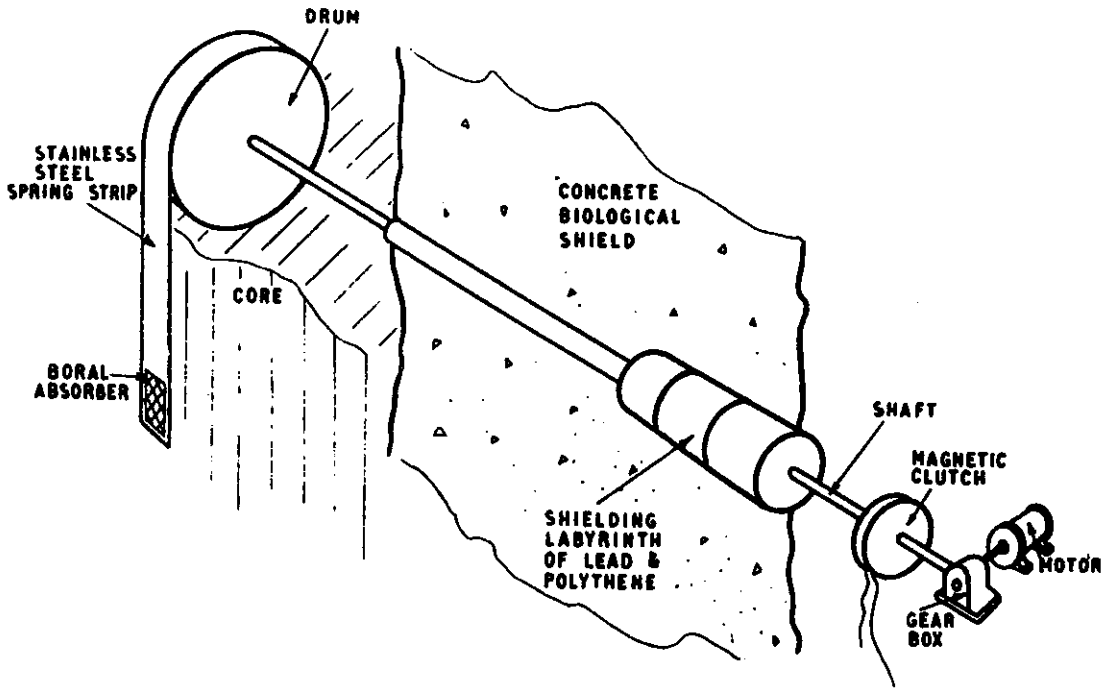


FIGURE 12 Diagram of the MOATA Control Rod Drive Mechanism (Schematic)

same time, the water in the core tanks is dumped and the reactor becomes completely shut down within a few seconds.

Measurement of reactor conditions, the second category of control mechanism, is accomplished by five neutron-detecting chambers arranged on top of the core.

A fission-chamber, with a coating of ^{235}U, is sufficiently sensitive to provide measurable signals when the reactor is shut down. A pulse output is taken from this chamber, and is displayed on the control console as a count-rate and as a rate of change of count-rate, or period.

The neutron flux level in MOATA from shut-down to full power varies over about 8 decades. Because this is much smaller than the range (~ 11 or 12 decades) in a full commercial power producing reactor, high range current type detectors remain quite effective in MOATA from full power down to very low flux levels.

Two similar current-type detectors are used for this wide-range reactor control. They are boron lined γ-compensated ionisation chambers from which mean current outputs are taken for display, one on a logarithmic scale over the full flux range, together with associated flux period information, the other on a switchable-range linear pico-ammeter, allowing accurate control and resetting of reactor power.

Two further entirely separate channels are used as high-flux trip instruments through a safety amplifier, which supplies current for all magnetic clutches. By interrupting the clutch currents, the amplifier shuts down the reactor automatically on reaching a level 25 per cent above rated maximum power.

One problem concerning neutron detecting instruments in reactors is their sensitivity to gamma radiation. In the shut down state fission products in the fuel emit high intensity gamma radiation, which could produce a signal in an ionisation chamber much greater than that due to the source neutrons. The fission chamber does not suffer from this trouble because the pulses due to fission fragments are so much larger than those due to γ-rays, but the log and linear current chambers are sensitive to gamma rays. This is overcome by using compensated chambers, where current due to gamma rays only is cancelled out through use of two similar ionisation volumes in one instrument, but only one of the volumes being neutron sensitive.

It is a principle of safe reactor operation that two independent channels must be operative at any time, and provide information about both flux (or power) and period (or rate of change of flux). Automatic trip circuits are incorporated in the electronic units, which may shut the reactor down at a chosen flux or period limit. The channels provided ensure that flux and period trips are available at any power level from the shut down state to peak power.

An automatic power controller keeps the reactor at a selected power level by appropriate movement of the regulating rod. This instrument relieves the operator of the need to make frequent adjustments to control absorbers, for instance, to compensate for core temperature changes. It also enables reactor power and hence neutron flux at any given point, to be kept more accurately constant over a long period of time than could be attained manually.

A significant contribution to safety of operation is made by the start-up sequence. This is an assembly of switches, arranged so that the operations involved in starting-up the reactor may only be carried out in a prescribed order, no operation of a particular item being possible until the preceding one has been completed satisfactorily. Lights indicate the sequence, and the stage reached at any time.

All electronic instruments and controls are housed in a small control console, which may be operated by a single operator.

Process System

The process system (Figure 13) consists of those units associated with circulation and cooling of the light water moderator. Circulation through core tanks and heat exchanger is maintained by a pump, but water may only rise into the core tanks on closure of the dump valve. The latter operation forms part of the start-up sequence, and is carried out before raising any control absorber.

The dump valve opens rapidly on shut-down, allowing all water in the core tanks to drain within a few seconds into the dump tank, which is the normal storage vessel for the total charge of demineralised light water.

A by-pass loop in the process system provides a filter and mixed-bed ion-exchange column for the water. The former maintains concentration of possible radioactive contaminants at a low level, and the latter keeps pH and conductivity values within a range which minimises corrosive attack on aluminium components in the water circuit.

The heat exchanger is of the shell and tube type cooled on the secondary side by water passing through and rejecting heat to an air cooling tower external to the building. The flow is controllable to permit adjustment of reactor operating temperatures.

The process system also includes instrumentation to monitor primary and

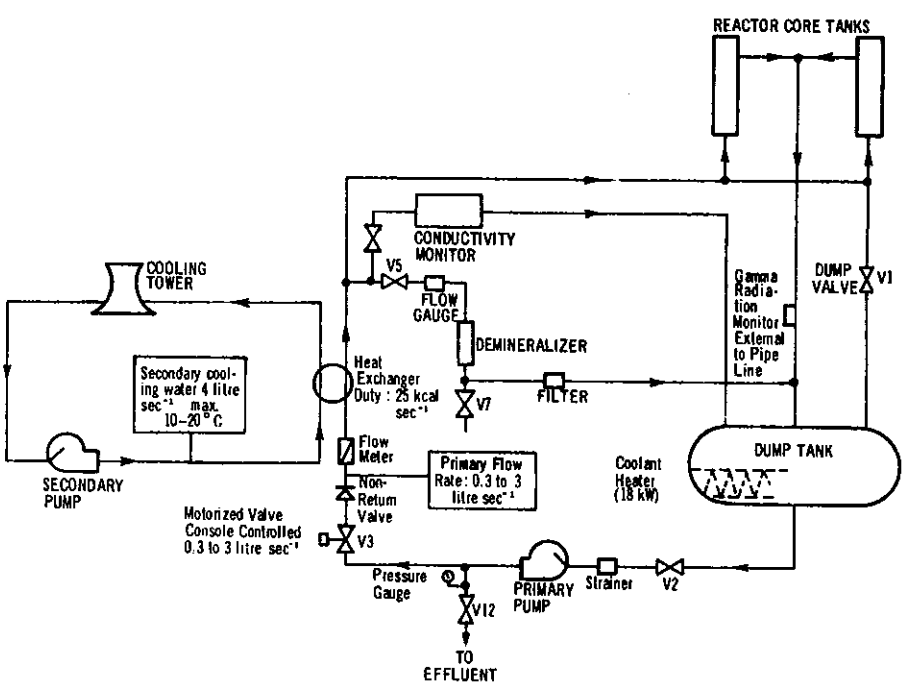


FIGURE 13 MOATA Process System

secondary coolant temperatures and flow, together with pressure switches which monitor water levels in the core tanks, detect interruptions to water circulation, and so on.

Reactor Calibration and Operation

Control characteristics may be described in terms of reactivity, ρ, which was shown in Section 1.1.4 to determine the rate at which changes in neutron population occur. The available reactivity depends on a number of factors, including the quantity of fuel in the core in excess of the critical loading. The absorptive properties of the control absorbers may be expressed in terms of reactivity, and safe operation of a reactor requires available 'negative' reactivity in control absorbers to be considerably greater than the 'positive' reactivity contributed by the fuel excess.

Rate of change of reactor flux obeys an exponential law, the characteristic time or period being dependent on the excess reactivity present. In order to keep the period longer than a certain safe minimum time, excess reactivity of the reactor is limited to a maximum of 0.6 per cent. This limitation brings the reactor into the 'eversafe' category, a sufficiently safe state to permit it to be sited in an open bay of a building without the need for sealed shell containment.

At the initial start-up of MOATA, the period obtained with all control absorbers fully withdrawn gave an immediate indication of excess reactivity acquired from the quantity of fuel loaded. The amount of control absorber insertion needed to restore the critical state gave a measure of the corresponding reactivity worth for that portion of the absorber. By adding further small quantities of fuel, which in turn required greater control absorber insertion to maintain a critical state, it was possible to calibrate most of the length of control absorbers in terms of reactivity. Both regulating rod and shim-rod were calibrated in this way, further calibration being obtained by balancing one rod against the other (*e.g.* withdrawal of one against insertion of the other).

It was not possible to load sufficient fuel to calibrate the shim-rod in its fully-inserted state, so these measurements were made with the reactor subcritical, reactivity worths of rod position changes being related to changes in the subcritical flux level.

Safety rods, which cannot be set in intermediate positions, were calibrated by a 'rod-drop' method. In this technique, the current from an ionisation chamber in the core was followed on a fast pen recorder from the moment current was interrupted to the rod magnetic clutch. The resulting transient

signal enabled calculation of the total reactivity worth of the rod and also provided information on time elapsed between breaking of clutch current and actual fall of the rod. Portions of the shim-rod were also calibrated by the drop method and thus this rod was calibrated by three different methods. Results were found to be in good agreement and Figure 14 shows typical rod calibration curves.

With the reactivity corresponding to every position of shim and regulating rods determined, it becomes possible to measure reactivity worths of other core components. Each component has neutron scattering and absorption properties, so its addition or removal must influence the fission cycle and its reactivity effect is measured simply by obtaining critical settings of the control absorbers, both with and without that particular component in place. The corresponding difference in reactivity of the rods then gives the reactivity of the component.

Neutron flux has been mentioned a number of times in these notes and is a quantity which is often measured. Ionisation detectors, such as ion chambers with coatings of fissile material or proportional counters filled with boron trifluoride gas enable the shape of the reactor flux distribution to be determined quite rapidly and with good accuracy, but do not lend themselves to precise determination of the absolute value of the flux level. For this purpose, activation detectors, such as indium or gold foils are irradiated under standard conditions at a number of standard locations. Their induced radioactivity is then subsequently analysed by absolute counting using 4π-β and β-γ coincidence techniques and allows the actual neutron flux at the irradiation positions to be deduced.

Another aspect of flux measurements is determination of neutron energy spectrum over the reactor. Neutrons are born in the fission process with energies ranging up to about 10 million electron-volts (MeV), and as they collide with moderating material they lose energy. Eventually they become thermalised, taking up the thermal motion of the moderator, energies then being about 0.025 eV.

By activating detectors having selective responses to neutrons of different energies it is possible to build up information on the neutron energy spectrum and its variation over the reactor. This is of importance for many experimental applications of the reactor.

Power calibration of MOATA is a little indirect. Although a high proportion of the energy released in fission ultimately appears in the form of heat, the large thermal capacity of the core, combined with relatively low power

produced, means that it takes a long time to reach thermal equilibrium. Hence measurement of heat added to the circulating coolant is not a reliable guide to total power produced. In any case, reactor users are usually more interested in the neutron flux than thermal power. Thermal power depends on the total fission rate over the core at a given flux level, and thus determination of neutron flux over the fuel region, relative to that at a standard position is the quantity of primary concern. By irradiating gold and copper wires fastened to nylon cords between the fuel plates the flux variations in the fuel can be determined. This enables the integrated core fission rate and hence thermal power to be determined relative to the readings of the reactor installed instrumentation.

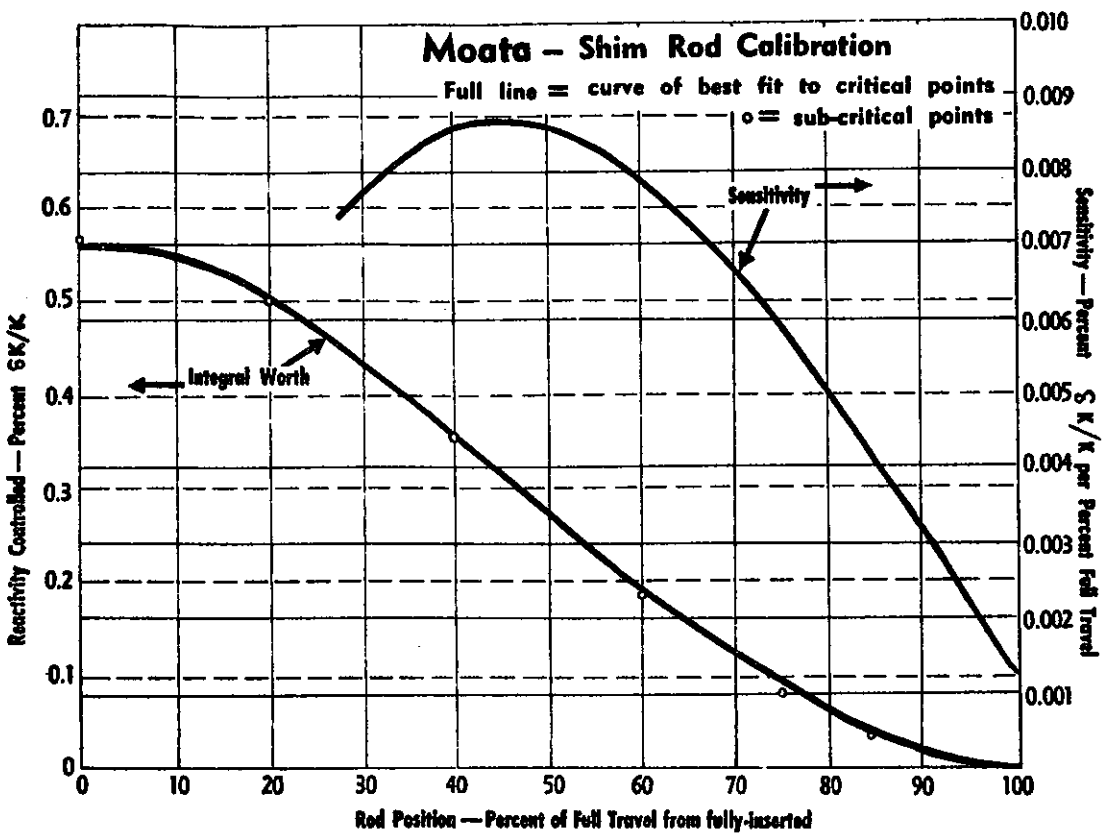


FIGURE 14 Typical Rod Calibration Curves

The reactor operates routinely at all power levels up to 100 kW as required on a daily basis by a wide range of experiments and users. Some typical uses are

(1) Extraction of external neutron beams for nuclear physics measurements, such as cross section determinations.

(2) Production of radioisotopes for a variety of applications embracing experimental, industrial and medical uses.

(3) Neutron irradiations of materials for analysis by activation techniques, investigation of radiation effects on chemical reaction rates, *etc.*

(4) Neutron radiography as a technique for some metallurgical examinations, *etc.*

1.2 ACKNOWLEDGEMENT

The material of Section 1.1.5 is taken largely (with some updating) from the article 'MOATA Reactor' by A.P. Marks, which was published in the October 1962 issue of the AAEC journal 'Atomic Energy in Australia'.

1.3 SUGGESTED FURTHER READING

(1) Any modern text-book available to you on nuclear physics at an introductory level. Concentrate particularly on chapters dealing with interactions of neutrons, γ-rays and low energy charged particles with matter.

(2) Price, W.J. Nuclear Radiation Detection (McGraw Hill)

(3) Murray, R.L. Nuclear Reactor Physics (Prentice-Hall)

(4) Murray, R.L. Introduction to Nuclear Engineering (Prentice-Hall)

(5) Glasstone, S. Principles of Nuclear Reactor Engineering, Chapters I-VI (Van Nostrand), or
Glasstone, S. and Sesonke, A. Nuclear Reactor Engineering, Chapters I-V (Van Nostrand Reinhold Co).

CHAPTER 2

MATHEMATICS OF REACTORS

Lecture by

B. CLANCY

CONTENTS

2.1 INTRODUCTION

There are many aspects of nuclear reactor behaviour which mathematicians find intriguing. In this lecture we will consider the dynamic behaviour of reactors, *i.e.* how they change with time.

In dynamic systems there is always some quantity (or quantities) which describes the important state of the system. In a ship these may be her speed and course; in an electric oscillator the amplitude and frequency may be the state quantities. For a nuclear reactor, the prime quantity of interest is the reactor power.

When an applied mathematician approaches a dynamic system, he first enquires what variables describe the state of the system, how these are related to one another and what forces may produce changes in them. He then tries to write the answers to the questions in the form of equations which provide a mathematical model for the system. Once this is done, the mathematician is likely to have two questions asked of him. They are :

"Is an equilibrium state possible?"

and "Is such a state stable?"

Let us consider just what these questions mean.

Equilibrium. In our context, an equilibrium state is one in which the fundamental state variables can and do remain constant with time. Using our three examples, an equilibrium state would mean :

- The ship's course and speed remain constant - a state preferred by the passengers.
- The oscillator frequency and amplitude don't vary - desirable if the oscillator is part of a radio transmitter.
- The reactor power remains constant - so fuses don't blow nor do house lights grow dim.

To find an equilibrium state, the applied mathematician turns to his mathematical model - the equations for the system - and seeks a solution of the equations in which all time derivatives of his state variables are zero.

Let us look at a simple example where the state variable is a particle velocity and the mathematical model is the familiar equation

$$F = m\frac{dv}{dt} .$$

Asked about an equilibrium state where velocity is constant, our applied mathematician sets to zero the term $\frac{dv}{dt}$, observes that F must then be zero and says

"a body can continue in a state of uniform motion except if it be

acted upon by a force".

I need not tell you the name of our mathematician in this case.

Stability. When a system is operating in an equilibrium state and so should remain there, variations in the outside forces are possible which will tend to push the system away from its equilibrium. The equilibrium is called *stable* if the system responds by trying to revert to the previous equilibrium, and it is called *unstable* if the system responds by going further and further away from that state. An intermediate situation is called *neutral* equilibrium. If you consider the dynamic system of an egg on a flat table, you will find three equilibrium positions, two of which are unstable and the third is either stable or neutral, depending on the disturbing force. For the three earlier examples (as for the egg) we see how desirable it is to operate with stable systems. For nuclear power reactors, a substantial part of the design effort is devoted to establishing a stable system. In this context it is important to notice that the 'systems' we will discuss from here on will usually include a control component. Our ship, for instance, will now have a steering engine and a helmsman. Our reactor will include instrumentation and control devices. The aim of control devices is to make an equilibrium point stable. Even so a system which will be stable without the intervention of a control system (*i.e.* an inherently stable system) is still eminently desirable. For one thing, the control system can then be tuned so as to minimise the size of any movement away from the operating state.

For complex systems, the answer to the stability question can tax the ingenuity of the applied mathematician, and he may only be able to give a partial answer. In these days, he will often employ a computer to give him some feel for the problem by using it to construct a solution to his mathematical equations under various initial conditions. If he is fortunate, he may find that his simulation of the system behaviour will itself provide sufficient information to satisfy the designers.

Hopefully, these general remarks will become more meaningful to you when we consider some specific problems; we shall examine two of these. The first I discuss here and now - the second will be your main problem during this school.

2.2 PROMPT REACTOR DYNAMICS WITH LINEAR FEEDBACK

Under certain conditions the dynamic behaviour of some reactors can be modelled by the equations

$$\frac{1}{p}\frac{dp}{dt} = \frac{\rho}{\ell} \quad \text{or} \quad \frac{d}{dt}(\log_e p) = \frac{\rho}{\ell} ,$$

$$\rho = \rho_{ex} - cE ,$$

$$E = \int_o^t p(t')dt' ,$$

where p is the reactor power,

E is the energy produced by the reactor up to time t,

ℓ is the prompt neutron lifetime,

ρ is the reactivity, and

c is a constant.

The second equation simply expresses the fact that the reactivity has two components: ρ_{ex} the external reactivity produced by control rod movements, and -cE the reactivity compensation produced as a result of temperature changes in the reactor. The model ignores the existence of delayed neutrons and assumes that all the energy produced by the reactor goes into raising its temperature. However, for extremely short times the model has some validity. An equilibrium condition for this reactor implies that the power p is constant so that $\frac{dp}{dt} = 0$; it follows that ρ must also be zero. However if the power p is constant, the energy E must be simply

$$E = p.t,$$

and thus $\rho_{ex} = c.p.t$.

Unless c is zero (*i.e.* there is no feedback), this implies that the external reactivity must be varying continuously. Thus much is straightforward, but the question of stability is more difficult. During the lecture I shall try to suggest the answer to this stability question by actually solving the differential equations with a computer.

2.3 POWER REACTOR SURGE PROBLEM

When considering a nuclear power reactor, one of the first questions which used to come to people's minds was: 'Will it blow up?' You are now sufficiently initiated to rephrase this question to read:

"Can the reactor operate at an equilibrium power level? If so, is the equilibrium stable?"

If many simplifications are made, we can construct a model to show the way the reactor power changes (surges) following a disturbance. This model is expressed by the equations

$$\frac{dp}{dt} = -p(p - 1 - b.e^{-t}) \qquad \text{for } t \geqslant 0 \qquad \ldots (1)$$

$$p = 1 \qquad \text{for } t = 0$$

where p is the reactor power in gigawatts,

t is the time in seconds after the disturbance, and

b is a parameter which characterises the size of the disturbance.

As a preliminary we note that if there is no disturbance and b = 0, then these equations imply that $\frac{dp}{dt} = 0$ at time t = 0. So we are actually starting from an equilibrium state. If b is positive then so is the derivative $\frac{dp}{dt}$, at time t = 0, and hence the power will initially increase. However, when the power has increased to the stage where $p > 1 + b\,e^{-t}$, the derivative $\frac{dp}{dt}$ becomes negative and the power then starts to decrease back towards p = 1. It looks then as if the equilibrium is stable but, as often happens in the real world, there is a snag. We must look closely at the quantity

$$h(b) = \underset{0 \leqslant t \leqslant 10}{\text{Max}}\,[p(t)] \qquad \ldots (2)$$

which is the maximum power in the ten seconds following the disturbance b. It turns out for the particular reactor being modelled that, if

$$h(b) \geqslant 1.2, \qquad \ldots (3)$$

the reactor fuel will start to melt, Equation (1) will no longer be valid and stability cannot be assured. Back to the designers goes the applied mathematician (you!), to be told that a likely size for the disturbance is given by b = 0.5. In addition, they tell you that in an experiment it was found that for a disturbance given by b = 0.3, the maximum power was 1.12 ± 0.01. The real question you have then to answer is : "Will the reactor fuel melt if b = 0.5, given that we may check out analysis against the result h(0.3) = 1.12 ± 0.01?"

To answer this question by pure analysis would be well nigh impossible, but it is quite feasible using a computer. This is your task. All you must do is

(i) write a computer program to solve Equation (1) numerically for any value of b;

(ii) use the program to calculate h(0.3);

(iii) check that your calculated value of h(0.3) is in satisfactory agreement with the experimental result - if it is not, go back to

(i); if it is, go on;

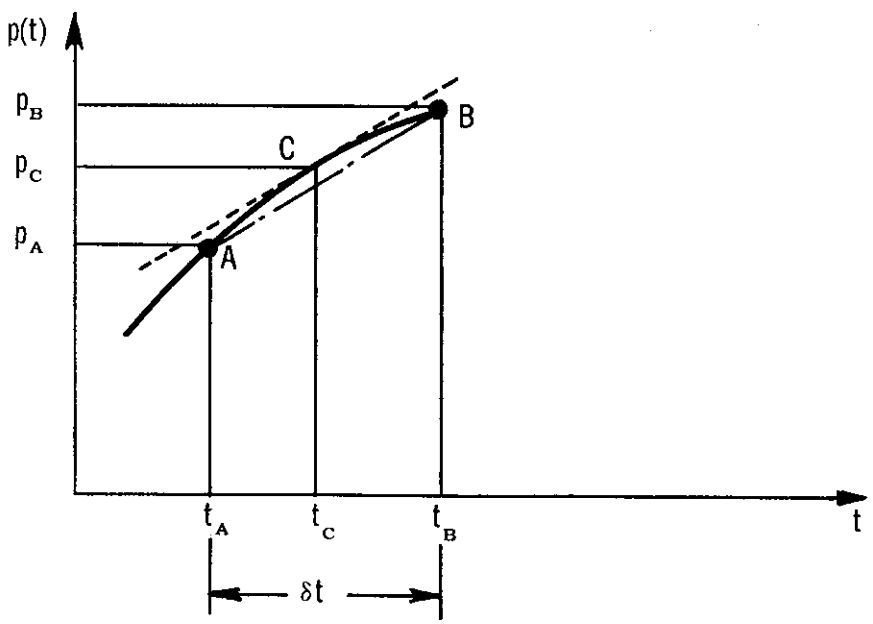
(iv) use the program to calculate h(0.5); and

(v) check condition (3).

2.4 NUMERICAL ANALYSIS

Here we will only consider one aspect of one part of Numerical Analysis - that which is concerned with solution of differential equations - and we will follow a line which introduces ideas as quickly as possible.

Say, by some means or other, we know the solution we seek up to the point A, as sketched, and we wish to extend the solution to the point B corresponding to the given time t_B.



From the sketch we note that the slope of the chord AB equals the slope of the curve at some point C between A and B

$$\frac{p_B - p_A}{\delta t} = \left.\frac{dp}{dt}\right|_C = \left(\frac{dp}{dt} \text{ evaluated at } t = t_C\right) . \qquad \ldots (4)$$

We may assume that the slope of the curve at C is some sort of average of the slopes at A and B

$$\left.\frac{dp}{dt}\right|_C = \left.\frac{dp}{dt}\right|_A (1-\theta) + \left.\frac{dp}{dt}\right|_B \theta \quad , \qquad \ldots (5)$$

where θ is some number between 0 and 1.

If θ were zero, Equation (4) would state

$$\left.\frac{dp}{dt}\right|_C = \left.\frac{dp}{dt}\right|_A$$

which is clearly wrong from our sketch, as is the corresponding statement when $\theta = 1$, *viz*

$$\left.\frac{dp}{dt}\right|_C = \left.\frac{dp}{dt}\right|_B \quad .$$

However, if we choose $\theta = \frac{1}{2}$ then Equation (4) states

$$\left.\frac{dp}{dt}\right|_C = \tfrac{1}{2}\left(\left.\frac{dp}{dt}\right|_A + \left.\frac{dp}{dt}\right|_B\right)$$

and, from the sketch, it would be hard to tell whether this is right or wrong. In any case, it looks to be a close approximation to the answer.

If we leave the value of θ undetermined for the moment, and combine Equations (4) and (5) we get

$$p_B = p_A + \left[\left.\frac{dp}{dt}\right|_A (1-\theta) + \left.\frac{dp}{dt}\right|_B \theta\right] \delta t \quad , \qquad \ldots (6)$$

Now we know $\left.\frac{dp}{dt}\right|_A$ from the differential equation we are trying to solve (Equation (1))

$$\left.\frac{dp}{dt}\right|_A = -\left(p_A - 1 - be^{-t_A}\right) p_A \quad , \qquad \ldots (7)$$

and if we take $\theta=0$ in Equation (6) we get a 'predicted' value for p_B,

$$p_B^* = p_A + \left.\frac{dp}{dt}\right|_A \delta t \quad , \qquad \ldots (8)$$

which we may use to *estimate* $\left.\frac{dp}{dt}\right|_B$,

$$\left.\frac{dp}{dt}\right|_B^* = -\left(p_B' - 1 - be^{-t_B}\ p_B'\right). \qquad \ldots (9)$$

Choosing $\theta=\frac{1}{2}$, corresponding to taking the usual average of the two slopes, and substituting the values given by Equations (7) and (9) into Equation (6) then gives us a 'corrected' estimate of p_B.

Let us see how we use the method just discussed - the so-called Heun method. We may start the method at $t_A = 0$ since we are given $p(o) = 1$. As an illustration we take $b = 0.3$ and $\delta t = 0.1$ and we apply the method using arithmetic performed with a slide rule. We repeat the equations we will use in the form of a *computational procedure*.

$\delta t = 0.1$

Begin

Set $t_A = 0$

Set $p_A = 1$

Set $b = 0.3$

Set $\left.\frac{dp}{dt}\right|_A = -\left(p_A - 1 - be^{-t_A}\right)p_A$ ... (7)

Set $t_B = t_A + \delta t$

Set $p_B^* = p_A + \left.\frac{dp}{dt}\right|_A \delta t$ ... (8)

Set $\left.\frac{dp}{dt}\right|_B^* = -\left(p_B^* - 1 - be^{-t_B}\right)p_B^*$ ... (9)

Set $p_B = p_A + \frac{1}{2}\left(\left.\frac{dp}{dt}\right|_A + \left.\frac{dp}{dt}\right|_B^*\right)\delta t$

Set $t_A = t_B$

if $t_A \geq 10$ finish

Set $p_A = p_B$

go to ——— (7)

The results of our first few slide rule calculations are

t_A	p_A	$1+0.3e^{-t_A}$	$\left.\frac{dp}{dt}\right\|_A$	t_B	p_B^*	$1+0.3e^{-t_B}$	$\left.\frac{dp}{dt}\right\|_B^*$	$\frac{1}{2}\left(\left.\frac{dp}{dt}\right\|_A + \left.\frac{dp}{dt}\right\|_B^*\right)$	p_B
0	1	1.3	0.3	0.1	1.03	1.272	0.250	0.275	1.0275
0.1	1.0275	1.272	0.252	0.2	1.0527	1.246	0.203	0.227	1.0502
0.2	1.0502								

Numerical Analysis is also much concerned with the study of

(i) the error incurred by using an approximation such as the Heun method; and

(ii) the effect of roundoff on the calculation, for example referring to the calculation above

$$p_A = \underline{1.0275X}$$

may have an error in the 6th significant figure (indicated by the X), but

$$p_A - 1 = \underline{0.0275X}$$

would then have an error in the 4th significant figure.

APPENDIX 2A

SOME ANALYSIS FOR THOSE WHO CAN STAND IT

Given the differential equation for the power p(t) of a reactor following a temporary disturbance b,

$$\frac{dp}{dt} = -\left(p-1-be^{-t}\right)p \ , \qquad p(o) = 1 \ ,$$

we ask "what can we do analytically?" Well, mathematical hindsight suggests we change the dependent variable to q given by

$$p = e^{q} \ ,$$

then $$\frac{dp}{dt} = e^{q}\frac{dq}{dt} = p\frac{dq}{dt} \ ,$$

and the differential equation becomes

$$\frac{dq}{dt} = -\left(e^{q}-1-be^{-t}\right) \ , \qquad q(o) = 0 \quad (\text{since } e^{o} = 1) \ .$$

This equation looks worse than the equation we were given, but let us suppose that we know that the power surge is not large so that we may take the first few terms in an expansion of e^{q}

$$e^{q} \simeq 1 + q \ .$$

We then obtain

$$\frac{dq}{dt} = -\left(q - be^{-t}\right) , \qquad q(o) = 0 \ .$$

A solution of the above differential equation is then an approximate solution of the given differential equation (to an extent that we will not pursue) provided q(t) remains small (so that $e^{q} \simeq 1 + q$).

Continuing with the analysis, we multiply the equation above by e^{t} and we obtain

$$\frac{dq}{dt}e^{t} + qe^{t} = b \ .$$

Now we (may) know that

$$\frac{d}{dt}(qe^{t}) = \frac{dq}{dt}e^{t} + qe^{t} \ ,$$

hence the differential equation reduces to

$$\frac{d}{dt}(qe^{t}) = b ,$$

which integrates to give

$$qe^{t} - 0 = bt \qquad (\text{since } q(o) = 0)$$

We thus obtain the approximate solution

$$q(t) = bte^{-t} .$$

In order to find the maximum power surge h(b), we seek the maximum value for p(t) and hence q(t). Now

$$\frac{dq}{dt} = b(e^{-t} - te^{-t})$$

and $$\frac{dq}{dt} = 0 \qquad \text{when } t = 1$$

hence max $q(t) = be^{-1} = 0.368\ b$,

which gives us the result

$$h(b) = e^{0.368\ b} \quad (\text{since } p = e^{q}).$$

Again we must emphasise that the above is only approximate. Nevertheless we find that

$$h(0.3) = 1.117 ,$$

which is in good agreement with the 'experimental' result cited earlier

$$\left(h(0.3) = 1.12 \pm 0.01\right) .$$

The approximation also gives

$$h(0.5) = 1.202 ,$$

which is so close to our critical value h = 1.2 that before we accept the result we would have to pursue detailed analysis of the effect of the approximation $e^{q} \simeq 1+q$ on our final answer. Alternatively we may pursue a numerical approach since we know that simple analysis cannot give us a sufficiently accurate result for the problem in hand.

CHAPTER 3

ACL - PROGRAMMING

Lecture by

J.M. BARRY

CONTENTS

3.1 INTRODUCTION

Each digital computer is capable of obeying a number of basic instructions. These instructions vary for the different computers but their function in scientific computing falls into the following general classes:

(i) The ability to perform the four basic arithmetic operations;

$$+, -, \times, \div.$$

(ii) The ability to perform logical operations (*e.g.* is $\text{Register}_0 \geqslant \text{Register}_1$).

(iii) The ability to transfer data (numbers) between core storage locations and a small set of registers in the computer. To perform arithmetic or logical operations on data, it is necessary to load the numbers into the registers. The cost of registers limits the number used in even the more advanced computers (*e.g.* there are sixteen general purpose registers in the IBM360 computer, and four registers in the NOVA computer). Once the result of an arithmetic operation is complete, it is usual to transfer this result (a number) back to a core storage location until the number is again required. Core storage locations are comparatively less expensive and the NOVA has 16,384 of these, while the IBM360, you will see, has 1,155,072.

(iv) The ability to initiate and complete the transfer of data between the computer and other devices (*e.g.* the reading of numbers from a punched card, the printing of results on a telytype, the 'writing' or 'reading' of data to or from magnetic tape files).

For a scientist to use a computer in its most basic state, he would need to write programs in what is called the basic machine language. It is then necessary for him to be concerned with all the 'housekeeping' tasks, such as remembering the addresses of core locations where he held temporary results *etc.* It is both tedious and time consuming for the scientist to communicate with the computer at such a low level, and the resulting computer program would appear to be unrelated to the mathematical problem he is solving.

Fortunately, there is no longer a need for programmers (scientific or commercial) to concern themselves with such low level programming. Many high level computer 'languages' have been developed to aid the programmer. In scientific programming, FORTRAN is the most widely accepted language, while COBOL dominates the commercial field. There are many other languages that have advantages in special circumstances; one such language is ACL-NOVA, the language to be treated in this Summer School.

By programming in such high level languages, the scientist is free to

devote his attention to the mathematical problem he is solving. The program he writes in the high level language, however, cannot be executed directly on the computer and an intermediate process is necessary before the program instructions are obeyed. There are two ways in which this process can be implemented:

- Compilation.
- Interpretation.

In compilation, the complete set of high level language instructions are translated into a set of machine language instructions by a compiler.

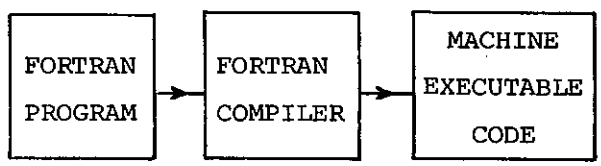


Figure 1 Compilation of a FORTRAN Program

For example, a set of FORTRAN source statements on punched cards is read in through the card reader, and the FORTRAN compiler, which is itself a computer program (usually supplied by the computer manufacturer), checks that each statement obeys the rules of the language (syntax analysis) and generates a set of machine language instructions equivalent to the FORTRAN code. The machine language instructions are then executed and the original FORTRAN statements (the source language) are no longer retained by the computer. For the purpose of the Summer School, compilation of programs will not concern us further because ACL is an interpretive language.

In any interpretive language (*e.g.* ACL), there is no attempt to turn the whole set of source statements into machine code before the program is run. The set of statements is retained in the computer for the duration of the computing task. Another computer program (this time known as the ACL interpreter) takes these statements one at a time from the sequential list (Figure 2). Once the statement has been selected the interpreter will:

(i) Examine the syntax of the statement.

(ii) Determine what operations are necessary, and the order in which they are to be performed.

(iii) Check that all the data necessary to perform the operations are available.

(iv) Perform the operations requested and determine any error conditions; store the intermediate results in core memory; print the final results

requested; read additional data requested.

(v) Determine which statement from the stored list is to be executed next.

There are several advantages to be gained from using an interpretive scheme:

. Because the complete list of statements is retained in the computer, it is possible to make alterations to selected statements, where necessary, without the need to recompile every statement in the program.

. When errors are detected during execution of the program (*e.g.* an attempt to divide by zero), the cause of the error can be quickly traced because the source language is retained in the computer.

To take full advantage of these facilities in the interpretive language ACL, the scientist must be able to communicate directly with the computer. It is no longer satisfactory for him to pre-punch all his commands on cards, read these through the card reader and wait patiently (?) for his results to be returned; he must be in constant communication with the computer throughout the entire operation. This is known as INTERACTIVE computing.

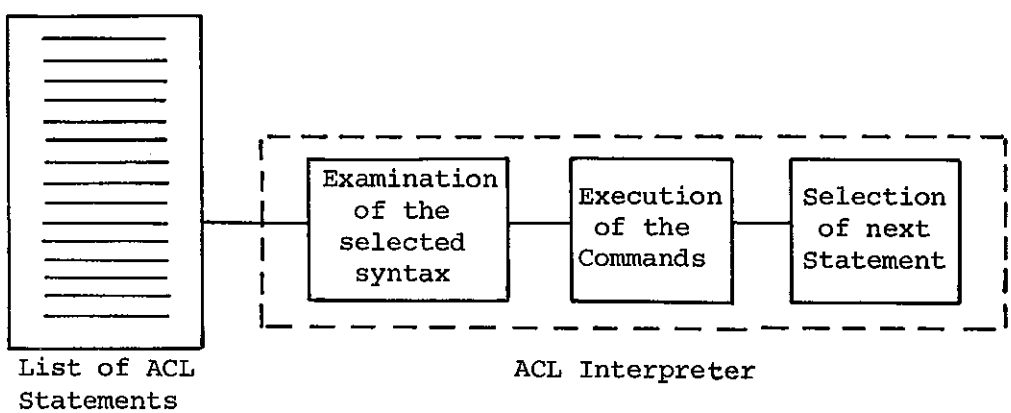


Figure 2 An Interpretive Language

ACL (A Conversational Language) is an interactive system. The scientist enters his program and communicates with the computer through his own teletype. This system provides the user with direct access to a powerful computer, and a convenient method for getting information into it. The user is able to see the result of a complex calculation in a simple understandable form. If the user is able to experiment with the input and see the effect of each change made, then he may quickly begin to understand the behaviour of his complicated mathematical model.

This is the principle of man-machine interaction, or on-line problem solving. The point is that both man and computer can each do some things

much better than the other. A closed loop system allows the special abilities of each to be used to advantage, and the combination may be far more powerful than the sum of the two components (it has been described as the 'two plus two equals five effect').

By comparison, the scientist who has to wait for his results loses 'thinking momentum' and has to collect his thoughts on the problem after every computer run.

The computer network being set up at the Research Establishment will allow powerful interactive computing facilities to be provided at various locations on site. In the meantime, the ACL-NOVA system allows users to solve small and medium sized problems in an interactive manner. The ACL language was designed jointly by Dr. P. Sanger and Mr. N. Bennett, and was implemented on the NOVA computer by Dr. Sanger.

The present lecture is designed to introduce sufficient fundamentals to program a numerical solution to the reactor power surge problem. For more details of the ACL language and the ACL-NOVA system, refer to the publications of Sanger (1971) and Bennett & Sanger (1973).

3.2 STARTING AN ACL SESSION

The ACL-NOVA system currently supports nine teletypewriter terminals. Input into the system is specified by pressing keys on the typewriter-like keyboard. To begin work at a terminal, the terminal must first be 'initialised'. This is done by locating a switch below the keyboard and setting it to the 'on-line' position. (Electric power is supplied to the terminal and the lines of communication to the computer are established.) To complete the initialisation, the user must next press two keys (the CTRL and G) simultaneously. The system responds by typing ACL-NOVA and spacing a few lines. Once the terminal has been initialised, ACL statements may be stored, or immediate statements executed.

3.3 STORED AND IMMEDIATE STATEMENTS

There are two types of statements available in ACL:

(i) The stored statements which are entered and accumulated for later execution. Stored statements are used to compose computer programs and are consequently the more important of the two.

(ii) The immediate statements are obeyed immediately they are entered. They are not 'remembered' by the computer. Immediate statements are used to perform 'one time only' or desk calculator type calculations. They may also be used to control the execution of stored programs, and to perform various editing and 'debugging' functions. An example of an immediate statement is

given below:

Entered Immediate Statement	5+6*9-4⟩	SQR(9)⟩†
ACL-NOVA response	55	3

3.4 IMMEDIATE ARITHMETIC EXPRESSIONS

We shall first consider the way numerical data are represented in the ACL system. All arithmetic and logical operations are carried out on floating point‡ numbers, with approximately seven decimal digit accuracy being achieved. Examples of valid arithmetic constants supported by ACL are:

5, 5., 3.124, .0516, .00016, 5.E+5(=5.0 x 10^5=500000),
-5.31, +8.69, 4.E-1(=4.0 x 10^{-1}=.4)

There are four basic arithmetic operations supported by ACL:

+ addition, - subtraction, * multiplication, / division, *e.g.*

6.+3*4-1.E-1⟩	1.E+2+1⟩
17.9	101

Parentheses may be used to form mathematical expressions. The order in which arithmetic operations are performed follows the normal mathematical rules of operator precedence, *e.g.*

5.*(3-1)⟩	(1-2*(9-3))⟩
10	-11

The unary minus sign can be used in arithmetic expressions, *e.g.*

-5.6⟩	-(8-3)⟩	8*(-1)⟩
-5.6	-5	-8

A secondary mathematical operation of exponentiation (↑) exists, *e.g.*

2^3 is coded as 2↑3⟩ 2.↑3.⟩ and 5+2↑3⟩
8 8 13

Use of the exponentiation operator requires additional care because it is not a direct arithmetic operator (*i.e.* it does not achieve its result through repeated multiplication). Instead, it uses logarithms to achieve its

---

† ⟩ *is used to denote carriage return. The carriage return key is pressed to indicate that a statement is complete. Only when the statement is complete is there any attempt to store or execute it.*

‡ *Remember, a digital computer uses a binary (base 2) system for the representation of numbers. All our decimal numbers are converted to a binary floating point representation inside the computer.*

function. Consequently, simple operations such as 4^2 are more efficiently coded as 4*4 rather than 4↑2. Of course, the exponential operator comes into its own with high order or fractional exponents, *e.g.*

$4^{3/2}$ becomes 4↑1.5⟩ or 4↑(3/2)⟩
8 8

ACL provides a number of basic mathematical functions (a complete list is given in Table 1). Examples of such functions are now shown:

Square Root	Exponential
SQR(7+9)⟩	EXP(1.)⟩
4	2.718282

The immediate arithmetic statements considered so far allow the ACL-NOVA terminal to be used as a desk calculator capable of performing arithmetic operations (and mathematical functions) on specified numbers. The results are immediately printed at the telytype†. There is no recall of any data that is entered into the system. The capacity of our desk calculator is increased if arithmetic constants that are entered are able to be stored away for later recall.

ACL Function	Mathematical Purpose
ABS	absolute value
SIN	trigonometric function (argument in radians)
COS	trigonometric function (argument in radians)
SQR	square root
EXP	exponential
LOG	natural log (base e)
INT	integer part (INT(3.1) is 3; INT(3.9) is 3)
ATN	inverse tan ($\tan^{-1}$), result in radians
DPT	only used with the TYPE statement

TABLE 1 MATHEMATICAL FUNCTIONS AVAILABLE IN ACL

† *The result of an arithmetic expression that is printed at a terminal is in one of two possible formats depending on its magnitude. The number is given in exponent form if it is in the range* $|number| \leq 10^{-4}$ *or* $|number| \geq 10^{3}$ *(e.g. .00001 would appear as 0.1E-3 when printed). All other numbers are printed without an exponent with seven significant figures if necessary. Integers are printed without a decimal point.*

This facility is provided through the use of:

- variables, and
- assignment statements.

The facility is best illustrated through the use of an example. The two immediate statements shown are entered:

```
A←2)
A*4+2)
10
```

The first is accepted but does not produce a visible response at the terminal. The value of 2 is associated with a variable denoted by A (the ACL interpreter assigns a particular core storage location to the variable A and stores the arithmetic constant 2 in that location). When the second statement is entered, the value associated with A is retrieved and used in the evaluation of the expression whose result is then typed. (The value of 2 is still associated with A and available for use in subsequent expressions.) For example,

```
subsequent statements
B←A-5)
B*(A-4))
```

produce the result

```
6 .
```

Both the variables A and B are still defined and this can be demonstrated by typing the following immediate statements:

```
A)
2
B)
-3
```

3.5 VARIABLES

ACL may have three types of variables:

- simple variable;
- singly subscripted variables; and
- doubly subscripted variables.

To start with we shall only consider the simple variable. A simple variable must begin with an alphabetic character and may be followed by up to three alphanumeric characters. (The alphanumeric characters are the set A, B, ...,Z,0,1,...,9; *n.b.* only upper case letters are available on the teletypes.)

Examples of valid ACL simple variables are:

A, A7, RATE, X12B .

3.6 ASSIGNMENT STATEMENTS

The assignment operation in ACL is denoted by the operator ←. In many other languages the equal sign is used, but this may tend to be a little confusing as true mathematical equality is not implied, *e.g.*

A←1)
A←A+2) .

The first statement assigns the value of 1 to the storage location reserved for the variable A. The second statement retrieves the value of A (*i.e.* 1) and increments it by 2 and stores the result (3) back in the location reserved for A.

ACL supports multiple assignments in the one arithmetic statement, and this is an important difference between ACL and many other languages such as FORTRAN. When multiple assignments occur in an expression (assuming first of all that the expression is free of parentheses), then the 'rightmost' assignment arrow is located, the expression to the right of this is evaluated, and the result assigned to the appropriate variable. The process is continued until all the assignments are carried out, *e.g.*

A←3+B←2

B is first assigned the value of 2, the value of 5 is subsequently assigned to A.

In more complicated expressions that contain a number of levels of parentheses plus multiple assignments, the expression is evaluated by searching first for the rightmost pair of opening and closing parentheses. The expression between these parentheses (an expression at zero parentheses level), is then evaluated by searching for the rightmost assignment arrow and proceeding as described in the last paragraph. The parentheses and assignment arrow processing is continued until the whole expression is reduced to zero parentheses level and evaluated; *e.g.*

Y←(A←2*B)↑(B←SQR(Z←9))+X←4)

The rightmost pair of parentheses enclose Z←9 and consequently

Z is first assigned the value 9
B is then assigned the value 3
A is then assigned the value 6

X is then assigned the value 4

Y is finally assigned the value $6^3+4=220$

To perform the same operation in FORTRAN the following five statements are required:

```
Z = 9
B = SQRT(Z)
A = 2*B
X = 4
Y = A**B+X .
```

In coding mathematical expressions in ACL, the choice of the variable names is left to the user. It is generally helpful to the user to select meaningful names, *e.g.*

Area of circle = $\pi \cdot \text{radius}^2$

could be coded as

```
AREA←PIE*R*R)
```

where PIE must previously be initialised to 3.141593.

3.7 TYPING STATEMENTS AT THE TERMINAL

The method of initialising a terminal has already been described and we are now in a position to enter immediate statements. Input begins at column 1, which is taken to be the 'leftmost' position of the teletype carriage that results from pressing the carriage return key (in subsequent examples the carriage return symbol) will be omitted). Input may be up to 72 characters. If input is continued past column 72, the whole line is cancelled and must be entered again.

The terminals attached to the NOVA computer are operated in what is called full duplex mode. This means that a character pressed at the keyboard is sent to the computer, but it will not be printed at the teletype unless it is sent back to the terminal (echoed) by the NOVA computer. By making use of this feature, only characters that are valid in syntax are accepted by the computer and these are echoed at the terminal. For example, if you press the keys 3++3 at the keyboard, the system would only type 3+3 at your terminal. The carriage return will only be accepted when the statement is complete, *e.g.*

```
Y←(A+3*(B+4)
```

requires an additional right parenthesis before the carriage return is

accepted.

This echo facility assists the user by preventing incorrect syntax being typed; however, it will not protect him against errors in the mathematical formulation of the problem being studied.

3.8 OVERVIEW OF ACL PROGRAMMING

At this stage we shall leave the immediate statement and progress to the more important stored statement. It is with the stored statement that computer programs are written. It is best to start with a fairly simple example, to demonstrate the steps involved in developing a computer program and then to study the rules associated with each ACL statement in some detail.

Problem to be solved. $18,000 is borrowed at an interest rate of 10.5% (monthly reducible) and repayments of $200 each month are made. How many years will it take to repay the loan?

Before we can program a digital computer to solve a problem, it is necessary for us to be able to detail logically the steps that are needed to solve the problem, as we would if we were going to tackle the problem with a desk calculating machine, slide rule, or pen and paper. Some people find it a help to draw a flowchart (Figure 3) showing the steps involved, while others prefer to visualise all the steps in their mind. No matter which way it is done, the problem must be broken down into equivalent basic steps.

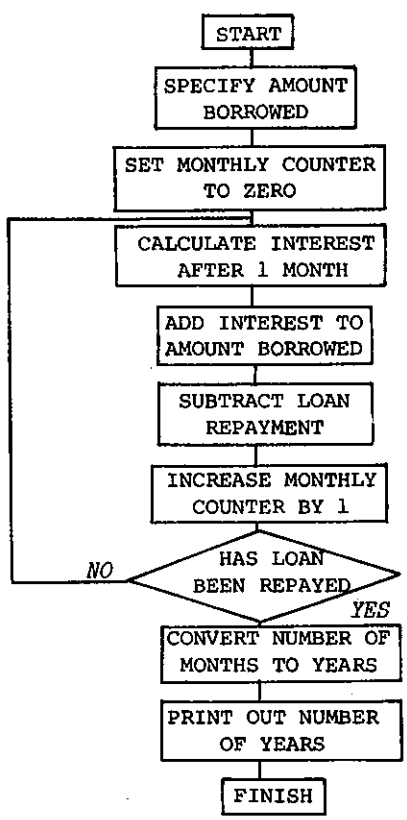


Figure 3 Flow Chart for Compound Interest Problem

The above process could be coded in ACL as indicated below:

```
123|4|56|7|8                                                          72
110|  |  |  |PRIN←18000
120|  |  |  |MCT←0
130|  |  |  |RATE←10.5
140|  |10|  |INT←PRIN*RATE/(100*12)
150|  |  |  |PRIN←PRIN+INT
160|  |  |  |PRIN←PRIN-200
170|  |  |  |MCT←MCT+1
180|  |  |  |IF(PRIN.GT.0) GO TO 10
190|  |  |  |NYRS←MCT/12
200|  |  |  |TYPE 'NO OF YEARS = ', NYRS
210|  |  |  |STOP
```

Each statement in the above example is associated with one logical step in the flow chart (Figure 3). It is possible to combine statements 140, 150 and 160 into the following single stored statement:

```
                                                                      72
140|  |10|  |PRIN←PRIN+PRIN*RATE/(100*12)-200
```

3.9 SEQUENCE AND STATEMENT NUMBERS

Stored statements are distinguished from immediate statements through the use of sequence numbers. Each stored statement has a sequence number in the range 100 to 999. Stored statements are used to build up a program. They are ordered according to their *sequence number* and may be inserted, modified or deleted freely by the user. Stored statements may be typed in any order, and a newly entered statement will replace a previously saved statement with the same sequence number.

In addition to the necessary sequence number, stored statements may have a *statement number* associated with them. In the previous example the only statement that had a statement number in addition to a sequence number was

```
123|4|56|7|8                                                          72
140|  |10|  |INT←PRIN*RATE/(100*12)
    ↑     ↑
sequence  statement
 number    number
```

Statement numbers are two-digit numbers in the range 1 to 99, and are typed in the fifth and sixth field positions in an ACL statement. Statement numbers are usually specified for a statement that is to be the target of a 'GO TO' statement, while sequence numbers are used to keep a program well

ordered.

To relieve the programmer from the routine task of typing in all sequence numbers, ACL has the facility to generate its own sequence numbers. Should the user first press the space bar at the start of a new statement, ACL assumes that the user desires automatic sequence number generation. It then provides the user with a three-digit sequence number, skips one column and is positioned to accept the optional statement number at column 5. The sequence number generated by ACL is determined by the following rules:

(i) In a new program the first sequence number generated is 110.

(ii) Subsequent sequence numbers are formed by increasing the previously entered sequence number by 10.

The first example shows the automatic sequence numbering system in action.

After the sequence number has been given and the carriage is positioned at the fifth column, the system is ready to accept a statement number. If a blank is again given, ACL interprets this as an indication that no statement number is required, and the carriage is then positioned at column 8 to receive the body of the statement. Alternatively, the pressing of a number at column 5 indicates that a statement number of one or two digits is expected (one-digit statement numbers must be followed by a blank).

3.10 TRANSFER OF CONTROL

ACL has one statement to transfer control within a program, namely the GO TO statement. In the previous example we saw the use of the GO TO statement to ensure that the process of monthly interest calculation and repayment was repeated until the loan was repaid. In this example the transfer of control was conditional upon the repayment of the loan not being completed.

We shall consider first the simpler form of the GO TO statement, namely the unconditional GO TO . This takes the form

GO TO { arithmetic statement / arithmetic expression } , *e.g.*

```
(i)   216  GO TO 10
(ii)  216  GO TO 140
(iii) 216  GO TO 5*3-5
(iv)  216  GO TO 12*10+20
(v)   216  GO TO I←5*3-5
(vi)  216  GO TO I←12*10+A←20
(vii) 216  GO TO A↑2+A←100
```

The transfer of control would be unconditional should any of the above

statements be executed. In each case, the following results would have occurred:

(i) Control is passed to a statement with a statement number of 10.

(ii) Control is passed to a statement with a sequence number of 140. (Remember sequence numbers are three-digit numbers, while statement numbers are of one or two digits.)

(iii) Control is passed to a statement with a statement number of 10.

(iv) Control is passed to a statement with a sequence number of 140.

(v) Control is passed to a statement with a statement number of 10, while I is set to 10.

(vi) Control is passed to a statement with a sequence number of 140, while I is set to 140 and A to 20.

(vii) The expression is evaluated as 10100, and because such a value is outside the valid range of sequence and statement numbers, an error diagnostic will be produced when an attempt is made to execute the statement.

The unconditional GO TO statement has rather limited use. It is easy to put a program into a never-ending loop if the GO TO is used on its own. The more useful form of transfer of control is the conditional GO TO. In the first example, the conditional GO TO statement:

```
180     IF(PRIN.GT.0) GO TO 10
```

transferred control back to the statement with the statement number 10 on condition that the principal had not yet been repaid. When the logical test is not satisfied (*i.e.* PRIN$\leqslant$0), control would pass to the next stored statement (in this case the one with a sequence number of 190). The general form of the IF statement is now described:

IF ({a.s.† or a.e.‡} {relational operator} {a.s. or a.e.}) { a.s. or a.e. / TYPE stmnt. / ACCEPT stmnt. / GO TO stmnt. / CALL stmnt. / RETURN stmnt. / CONTINUE stmnt. / STOP stmnt. }

The following relational operators are available:

---

† *a.s. arithmetic statement*
‡ *a.e. arithmetic expression*

.EQ. =

.NE. ≠

.GT. >

.GE. ⩾

.LT. <

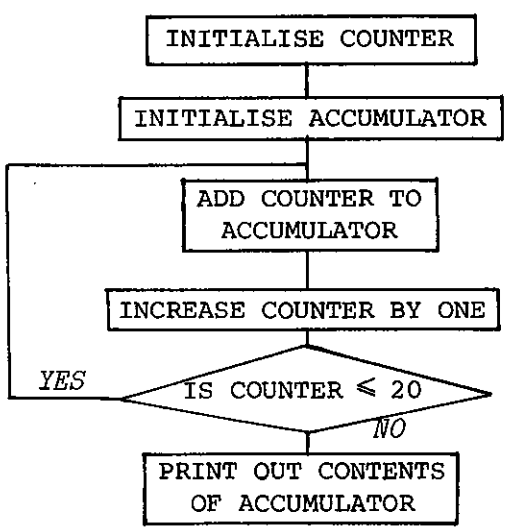
.LE. ⩽

If the logical relation between the two arithmetic statements or expressions enclosed in brackets is satisfied, then the third statement outside the brackets is executed; otherwise control is passed to the next stored statement.

Example (a). We frequently find it necessary to repeat a section of code a given number of times. Suppose our problem is to find the sum of the first 20 integers, *i.e.*

1+2+...+20 .

Ignoring any appeal to mathematical analysis then the following logic would be sufficient:



This could then be coded in ACL as

```
110      C←1
120      SUM←0
130 5    SUM←SUM+C
140      C←C+1
150      IF(C.LE.20) GO TO 5
160      TYPE 'SUM OF FIRST 20 INTEGERS IS ',SUM
170      STOP
```

It is possible to simplify the above by combining statements 140 and 150

as

```
140     IF(C←C+1··LE·20·) GO TO 5
```

Example (b). The following statement is sufficient to test a variable, and should a negative value be detected then its absolute value is substituted†:

```
156     IF(A.LT.0) A← -A
```

3.11 INPUT AND OUTPUT STATEMENTS

The first program given as an example required the determination of the number of years necessary to repay a loan of $18000 at 10.5%, with monthly repayments of $200. The program to perform this particular calculation was provided. After seeing the result we may wish to experiment with the various parameters (principal, interest rate and monthly repayment) that affect the loan. Of course, this could be done by altering statements 110, 130 and 160 in the program. However, a better policy is to design a program that enables the three important quantities to be typed in when the program is run, rather than require a detailed knowledge of the way the program functions to alter statements within it. The program then becomes a general piece of software to be used by any other person interested in the repayment problem, rather than a 'one off' calculation. The way this is done is illustrated in the next example:

```
110 12   ACCEPT PRIN,RATE,REPM
120      MCT←0
130 10   INT←PRIN*RATE/(100*12)
140      MCT←MCT+1
150      IF(PRIN←PRIN+INT-REPM.GT.0) GO TO 10
160      NYRS←MCT/12
170      TYPE 'NO OF YEARS = ',NYRS
180      GO TO 12
```

When any program is loaded, the computer makes no attempt to execute the stored statements until the command RUN is entered as an immediate statement.

The ACCEPT statement suspends execution of the program, and enables the user to type in the value of each variable named in the ACCEPT statement. For the above example the following would appear on the teletype when the command RUN is entered:

---

† *There is of course an easier way of doing this through the ACL supplied absolute value function ABS:*

e.g. *A←ABS(A)*

```
RUN
110     PRIN←
```

ACL gives the sequence number of the accept statement concerned and the name of the variable to be entered. The user then responds by typing in the appropriate value. At the end of the calculation, the teletype listing might appear as

```
RUN
110     PRIN←20000
110     RATE←11.5
110     REPM←200
NO OF YEARS = 27.8333
110     PRIN←
```

The TYPE statement that enables output to be displayed has been partially treated in the examples so far. The value of an expression may be typed out in a format determined by its magnitude, as discussed in Section 4, or entirely in its exponent form, *e.g.*

```
512     TYPE A3
```

causes the value of A3 to be typed in a form determined by its magnitude, while the statement

```
512     TYPE "A3
```

causes the value of A3 to be typed in exponent form, regardless of its magnitude.

When variables are separated by commas, they are printed on the same line with a space between each number. For example, if A1 has the value of 271, and B the value of 6731 in all the following examples, then

```
473     TYPE A1,B
```

causes the line

```
271 6.731000E+03
```

to be typed at the terminal.

To continue output on a new line, a semi-colon should be used as the delimiter, so that

```
423     TYPE A1;B
```

causes the lines

```
271
6.731000E+03
```

to be typed at the terminal. The semicolon delimiter can also be used to space a number of lines, for example

```
483     TYPE ;;;
```

causes the teletypewriter to be spaced three lines.

Literal data (character output or headings) can be printed by enclosing it within apostrophes. For example, the statement

```
671     TYPE ' THIS IS LITERAL DATA '
```

causes the following to be typed

```
THIS IS LITERAL DATA
```

Output may be placed at a particular carriage position by indicating the required position by means of an arithmetic statement or expression enclosed between the < and > symbols. This positional parameter must appear before the required variable or literal data and be separated from it by a comma. For example the statement

```
821     TYPE <27>,'A1 = ',A1
```

causes the following to be printed

```
1                          27|28
                             |A1 =  2.710000E+2
```

When an expression or statement is used between the < and > symbols the position of the carriage is determined dynamically, *e.g.*

```
TYPE <3*X-4>,'*'
```

would type * at the eleventh column if X = 5. Consequently,the TYPE statement may be used to produce a quick graph on the teletype listing of various mathematical functions formulated.

The carriage can be left positioned at column 1 after typing output by using the delimiter colon. This allows output on a given line to be overprinted by successive TYPE statements. For example

```
173 51  TYPE <I>,'*':
            IF(I←I+1··LE·72) GO TO 51
```

would produce a whole line of seventy-two * symbols. (This tends to be a slow process as the carriage is returned to column 1 after each individual * is

typed.)

3.12 STOP AND END STATEMENTS

. Execution of a STOP statement completes the execution of a stored program. The program is still retained.

. The END statement is used to indicate that work has finished at a terminal. It causes all core storage areas used by the terminal to be cleared. It should be the last statement executed before leaving the terminal.

3.13 SUBSCRIPTED VARIABLES

Many mathematical operations require the use of vectors and matrices. ACL supplies a means of handling one or two dimensional arrays. For the simplest array (the one dimensional vector) the ith element of the vector y (represented as v_i in mathematical notation) is coded as V(I) in ACL.

Singly subscripted variables consist of an alphabetic character, which may be followed by an alphanumeric character, followed by an arithmetic statement or expression enclosed in parentheses. The arithmetic statement when evaluated is truncated to an integer value and must be in the range 0 to 65535. The following are examples of valid singly subscripted variables in ACL:

V(12), XI(12), AD(1), AI(J←3), A9(I←B+3)

The following problem demonstrates the use of a vector array. Find the mean and standard deviation of ten numbers to be entered one at a time through an ACCEPT statement.

The mean is defined as $\bar{x} = \frac{\sum_{i=1}^{n} x_i}{n}$

The standard deviation is defined as

Standard Deviation $= \sqrt{\frac{\sum_{i=1}^{n} (x_i - \bar{x})^2}{n-1}}$

```
110      I←1
120 5    ACCEPT X(I)
130      IF(I←I+1.LE.10) GO TO 5
140      I←1
150      SUM←0
160 6    SUM←SUM+X(I)
```

```
170      IF(I←I+1.LE.10) GO TO 6
180      MEAN←SUM/10
190      I←1
200      SUM←0
210 7    SUM←SUM+T*T←X(I)-MEAN
220      IF(I←I+1.LE.10) GO TO 7
230      SDEV←SQR(SUM/9)
240      TYPE MEAN,SDEV
250      STOP
```

Here we use the vector X to store the 10 numbers prior to finding the mean and standard deviation. Before employing vectors in a program *make sure they are really necessary*. If the program required the calculation of the mean only, then there would be no need to retain the ten numbers, because the sum of the 10 number could be accumulated as each number is read in, *e.g.*

```
110      SUM←0
120 5    ACCEPT X
130      SUM←SUM+X
140      IF(I←I+1.LE.10) GO TO 5
150      MEAN←SUM/10
```

A second example involving singly subscripted variables is now given. In an existing ACL program 10 numbers have previously been computed and stored in a vector X. Write the section of code necessary to determine the largest and smallest members of the vector.

One method for doing this is to start at the first element of the array, and because no other elements have yet been tested, to set both the smallest and largest number indicators to the first element. Other elements are then compared in turn, with the two indicators (and replace them where necessary) and when all the vector elements have been tested, the smallest and largest are then available:

```
         :
         :
210      SM←X(I←1)
220      BG←SM
230 5    IF(X(I←I+1).GT.BG) BG←X(I)
240      IF(X(I).LT.SM) SM←X(I)
250      IF(I.LT.10) GO TO 5
260      TYPE 'SMALLEST NO = ',SM;'LARGEST NO = ',BG
270      STOP
```

We shall now deal with the doubly subscripted array. If matrices have not been included in your mathematics course at this stage, then this section can be avoided as it is not necessary for solving the Summer School reactor problem.

The doubly subscripted array starts with an alphabetic character and may be followed by an alphanumeric character, followed by two arithmetic expressions or statements enclosed in parentheses. Each of these statements is truncated to an integer when evaluated. The subscripts are restricted to the range 0 to 255, *e.g.*

```
A(5,3), X1(I,2*I-3), YA(I←J*3,K←5*N+5) .
```

Example. In an existing ACL program two matrices A and B (with n rows and m columns) have previously been computed. Write the section of code necessary to form the sum of these matrices.

Matrix addition is defined as:

$$C = A+B \ ,$$

where the components of the matrix C are defined as

$$C_{ij} = A_{ij} + B_{ij} \ .$$

The ACL code necessary to accomplish this is:

```
          :
210       I←1
220 6     J←1
230 5     C(I,J)←A(I,J)+B(I,J)
240       IF(J←J+1.LE.M) GO TO 5
250       IF(I←I+1.LE.N) GO TO 6
          :
```

3.14 SUBROUTINES

It is often useful to develop sections of code independently of the main program. This allows basic mathematical routines to be perfected and tested in a simple environment. These subsections of code are known as subroutines. Subroutines are frequently written to perform such basic mathematical tasks as

(i) Solution of sets of linear simultaneous equations.

(ii) Multiplication of matrices.

Languages such as FORTRAN place heavy emphasis on the use of subroutines.

In ACL programming however there is only one restricted form of subroutine usage available. The following example demonstrates the use of the CALL statement in ACL to execute a group of stored statements that effectively form a subroutine.

The roots of the quadratic equation $ax^2+bx+c=0$ are determined through the use of a *subroutine*. The *main* section of code will read in the three values for A,B, and C. A CALL statement branches to a subroutine whose function it is to determine the roots. The roots will be returned to the *main* section as the variables X1 and X2. The variable ERR is used on return to indicate whether real roots have been determined.

```
110      ACCEPT A,B,C
120      CALL 800
130      IF(ERR.EQ.0) TYPE 'REAL ROOTS ARE ', X1,X2
140      IF(ERR.NE.0) TYPE 'ROOTS ARE IMAGINARY'
150      STOP
800      DISC←B*B-4*A*C
810      IF(DISC.LT.0) GO TO 5
820      X1←(-B+SQR(DISC))/(2*A)
830      X2←(-B-SQR(DISC))/(2*A)
840      ERR←0
850      RETURN
860 5    ERR←1
870      RETURN
```

The CALL statement is used to pass control to a group of stored statements commencing with the sequence number 800. These subroutine statements evaluate the discriminant (b^2-4ac) and compute the real roots in X1 and X2 if the discriminant is not negative. The absence of real roots is indicated by initialising the variable ERR to a non-zero value (*i.e.* when the discriminant is negative). The RETURN statement passes control back to the main program. For either value of the discriminant, control returns to sequence number 130.

3.15 LIST STATEMENT

Stored statements may be listed (printed out) at a terminal (in sequence number order) when the LIST statement is executed. A single statement, a small group of statements or the entire stored program may be listed:

Example (i)

```
1
LIST
```

causes the whole program to be listed.

Example (ii)

```
1
LIST 67
```

causes the stored statement with the statement number 67 to be listed.

Example (iii)

```
1
LIST 117,421
```

causes all the stored statements, starting from sequence number 117 and finishing with sequence number 421, to be listed.

Some of the teletypewriter terminals have a paper tape reader and paper tape punch attached to them. Input from a terminal can come from either the teletype or the paper tape reader. When output is being printed at a terminal, it will also be punched onto paper tape if the paper tape punch is ON. The paper tape punch is normally OFF.

If you have tested your program and wish to keep a copy of it on paper tape, then you could execute the statement

```
1
LIST:)
```

In this case, to give you time to turn the punch ON, the listing is delayed until a second) is entered as the first character on the next line. When the) is entered at the terminal, five inches of feeder holes are punched at the start of the tape. Five inches of feeder holes are also punched at the end of the listing. For this Summer School we now have a more reliable means of saving programs on disc storage devices, and the paper tape system can be avoided. The means of doing this will form the basis of an additional lecture at this school.

Listing may be terminated at any time by entering the symbol ? from the keyboard.

3.16 SYMBOLS STATEMENT

Execution of the SYMBOLS statement, which takes the form

```
1
SYMBOLS
```

or

```
1
SYMBOLS:
```

causes the values of all the variables defined in the symbol table to be printed out in the form:

```
A1←1.320000E-20
Z13←21
```

```
A2(7)←.01
NA(3,7)←.0823 .
```

If the colon is present in the SYMBOLS statement, it indicates that the output is to be punched onto paper tape. Entering ? from the keyboard will also cause the listing of symbols to be terminated.

The above form of the symbol table listing was chosen so that each line was an immediate arithmetic statement and if the output was punched onto paper tape, it could be later read in to re-initialise the symbol table.

3.17 PROGRAM TRACING

To assist in debugging a program, special tracing facilities are built into the ACL language.

A statement is said to be traced if any symbol table assignments that occur while the statement is being executed are typed at the terminal. These assignments are typed in the form

1	5
{sequence number}	{variable}←{value}

or, for example,

1	5
317	A72←6.13

.

Individual statements may be traced by executing an immediate statement of the form

1	6
TRON	{arith stmt or exprn}

before beginning the execution of a stored program. For example, if the immediate statements

1	6
TRON	190
TRON	220
TRON	230

were executed, followed by the execution of the RUN statement, then the results of statements 190, 220 and 230 would be displayed every time they were executed. When you have finished tracing individual statements you should turn the trace off, using statements of the form

1	7
TROFF	{arith stmt or exprn}

.

If you are really having trouble with your program then you can get a full trace of your program by executing the immediate statement

```
|1    |6
|TRON |
```

before saying RUN.

Every stored statement that is subsequently executed is listed at the terminal and traced. This allows a complete trace of a stored program to be obtained. When you have finished your full trace you should execute the immediate statement.

```
|1     |7
|TROFF |
```

3.18 INTERRUPTING PROGRAM EXECUTION

Execution of a stored program may be interrupted by

(i) The character ? being typed at the terminal while the program is running.

(ii) The result of error condition in the stored program (*e.g.* attempted division by zero).

(iii) The result of executing a PAUSE statement stored as part of the program.

Statements may then be inserted, modified or deleted, or immediate statements may be executed. If the first input character on any line is), then execution continues from the point at which the program was suspended. Execution of the stored program may recommence at a different point by executing an immediate GO TO statement, or it may be restarted by executing a RUN statement.

When the program is interrupted by typing ?, the symbol is not printed at the terminal immediately. The current instruction is completed, and the sequence number of the next statement to be executed is typed to indicate the current state of program execution, *e.g.*

210? indicates that the program has been interrupted and the next statement to be executed was the one with the sequence number 210.

3.19 ERROR CORRECTION

Errors that occur *while a statement is being typed* may be corrected quite simply. For example, to delete the last two characters that were accepted as input type < 2). This would cause the original line of input minus the last two characters to be typed on a new line. These are syntax checked as though they were the original keyboard input. Thus,

```
1
A2←B+SQR(AZ1+C←3)-X3(4,2)-6<2
```

would result in the new line

```
1
A2←B+SQR(AZ1+C←3)-X3(4,2)
```

being typed out. A one- or two-digit number may follow the < symbol.

A statement being entered at the keyboard may also be corrected in *edit mode* (see next section) by typing <<) after the last input character.

Finally, if the whole input line is to be deleted, type <<< after the last input character.

To delete an existing statement from a program, type the sequence number of that statement, and follow it with one blank (to distinguish it from an immediate arithmetic statement) before hitting the carriage return key.

3.20 EDIT STATEMENT

The EDIT statement is used to modify statements that are part of a stored program in what is described as 'edit mode'. The statement takes the form:

EDIT b {arith stmt or exprn} .

When executed, the stored statement with the sequence number specified by the arithmetic statement or expression is printed, and the carriage is returned to the next line at column 1. At this point the statement is ready to be edited. To follow the 'edit mode' procedure, consider a pointer to each character in the original statement (the OS pointer), where initially OS is one.

To copy a character from the original statement, the SPACE key is pressed for each required character and the copied character becomes virtual input from the terminal. If this character is syntactically correct, it is echoed at the terminal and the OS pointer is increased by one.

To insert a new character, the required character should be typed, and it is echoed if it is correct in syntax. If a space character is to be inserted, then the special character ESC should be typed. The OS pointer is not altered in this case.

To jump over a character in the original statement the RUB OUT (or DEL) character should be typed and the OS pointer is simply increased by one, without motion of the carriage.

Once the OS pointer has reached the end of the original statement, further attempts to press SPACE or RUB OUT have no effect. For example, the stored statement

```
|1   |5  |8
|211 |72 |A3←6
```

can be modified by executing the immediate statement

```
|1    |6
|EDIT |211
```

or

```
|1    |6
|EDIT |72
```

If the characters

```
|1
|bbbbbbbb(2)RubOutb2b)
```

are typed, after the statement has first been listed, then the resulting stored statement would be

```
|1   |5  |8
|211 |72 |A(2)←26
```

3.21 CONCLUSIONS

The syntax checking of input and the powerful editing and error correction facilities provided by the ACL-NOVA system allow a user to set up a stored program both simply and easily.

In no time at all, the user can be producing results, and with the interrupt and tracing facilities at his disposal he can also quickly debug his program. By having the opportunity to control the input to a program, the user can gain valuable insight into his calculations.

However, with such ready access to problem solution using interactive computing,one must be careful not to be carried away by the computer. There are times when the only way to discover an error is to THINK - and that is solely our prerogative.

3.22 REFERENCES

Bennett, N.W. & Sanger, P.L. (1973) - The Development of the ACL Language and its implementation ACL-NOVA. Australian Computer Journal, 5 (3) 105-114.

Sanger, P.L. (1971) - ACL-NOVA: A Multi-user Conversational Interpreter for the Nova Computer. AAEC/E221 (Re-issued 1972).

Examples are:

327, 1.231, .0014, -1.45E+12, 3E2.

VARIABLES

Three types of variables may be used - simple variables, singly subscripted variables or doubly subscripted variables.

A simple variable name must begin with an alphabetic character and may be followed by up to three alphanumeric characters. Subscripted variable names consist of an alphabetic character, which may be followed by an alphanumeric character, followed by the subscript (or subscripts) enclosed in parentheses. Singly subscripted variables must have subscripts in the range 0 to 65535. Subscripts for doubly subscripted variables must be in the range 0 to 255.

OPERANDS AND OPERATORS

The basic operands in an arithmetic statement or expression are variables and numbers. The relevant operators are +,-,*,/(divide),↑(power), ←(assignment) and the functions ABS, ATN, COS, DPT, EXP, INT, LOG, SIN and SQR. The mathematical operators have the usual hierarchy of ↑ first, * or / next, and then + or - .

ASSIGNMENTS

If a variable is followed by an assignment arrow, then during statement execution the expression to the right of the assignment arrow is evaluated and its value given to that variable. If a number of assignment arrows occur in an expression then they are processed from right to left.

ARITHMETIC STATEMENTS AND EXPRESSIONS

An expression is termed an arithmetic statement if the first term in the expression is a variable followed by an assignment arrow; otherwise it is an arithmetic expression.

The result of executing an *arithmetic expression* is typed at the terminal in two possible formats depending on its magnitude. The result is given in exponent form if $|\ \text{result}\ | \leqslant 10^{-4}$ or $|\ \text{result}\ | \geqslant 10^{3}$; otherwise it is typed in non-exponent form with seven significant figures.

Examples: (i) 2↑2+AB1←2.453 is an *arithmetic expression* and during execution AB1 would be assigned the value 2.453 and the result 11.453 would be printed at the terminal.

(ii) AX←SQR(C←4*B(1)←1) +3-B←-2 is an *arithmetic statement* and during execution B(1) would first be assigned the value 1, C would then be assigned the value 4, B would be assigned the value -2 and AX would finally be assigned the value 7, but nothing would be printed at the terminal.

SEQUENCE NUMBERS AND STATEMENT NUMBERS

Every statement that is to be stored for later execution must have a sequence number in the range 100 to 999. These statements are stored in the user's work area in an order based upon increasing sequence numbers.

A one- or two-digit statement number in the range 0 to 99 may also be associated with each stored statement.

ERROR CORRECTION

Errors that occur while a statement is being entered at the keyboard may be corrected quite simply. To delete the last two characters that were accepted as input, for example, type < 2). A statement may be corrected in edit mode by typing <<) after the last input character. Finally the whole input line may be deleted by typing <<<) after the last input character.

The EDIT statement allows stored statements to be modified in edit mode. Using this statement, characters may be copied from, inserted into or deleted from an existing statement, or a complete statement copied by changing the sequence number.

STORED PROGRAM EXECUTION

Execution of the RUN statement or the immediate form of the GO TO statement begins stored program execution. Input to a stored program can be read by using the ACCEPT statement, while the IF statement can be used for loop control.

The user can interrupt stored program execution at any time by entering the character ?. At this point, stored statements may be inserted, modified or deleted, or immediate statements executed. If the user subsequently enters) at the column 1 position, then execution continues from the point where the program was suspended. Execution of the stored program may recommence at a different point by executing an immediate GO TO statement, or it may be restarted by executing a RUN statement.

Stored program execution may also be suspended if an error condition occurs, or as the result of certain PAUSE conditions. This allows the user to take corrective action, or to turn local or global trace indicators ON before continuing execution of the stored program.

A STOP statement is used to complete stored program execution, while an END statement should always be the last statement executed when a user has completed his work at a terminal.

INPUT FROM PAPER TAPE OR MAGNETIC DISK

Standard programs may be held on paper tape and may be entered from the paper tape reader once a user has commenced work at a terminal. The value of

variables to satisfy a series of ACCEPT statements in a program may also be read from paper tape. The LIST: and SYMBOLS: options make it very convenient to use input in this form.

It is now preferable to store standard programs on magnetic disk. The means for doing this will be covered in an additional lecture.

APPENDIX 3B

PRACTICE EXAMPLES

QUESTIONS

Before you attempt to code the reactor power surge problem try these practice examples. The answers to the questions are given on p 3B.4 but don't be too hasty to seek out these answers until you have had a go yourself.

Q1. What result would you expect from the following ACL-NOVA immediate statements? Are all the statements valid?

(i) `1.E-5*5./.01`

(ii) `5.E+2+1`

(iii) `5+(2+3*(1+4)`

(iv) `6+2↑3`

(v) `-(3-4)*(-1.)`

(vi) `(3.-4.)↑2`

(vii) `10./(19-(3*6-4.5*2))`

(viii) `3(5-2)`

(ix) `1./(7-(3*2-1)-2)`

(x) `*3./4.`

(xi) `3+4*6↑2`

Q2. Determine the valid simple variables in the following list:

(i) `X`

(ii) `XY`

(iii) `X4Y`

(iv) `3XY`

(v) `RATE`

(vi) `SUM`

(vii) `PRINC`

(viii) `X1Y2`

(ix) `FIVE`

(x) `Rate`

(xi) `6IX`

(xii) `BLOT`

Q3. What is the end result of the following immediate arithmetic statements and expressions entered in the order shown?

(i) `A←3*7-4)`

(ii) `A)`

(iii) `2*B←4+C←5.E+1)`

(iv) `A+B)`

(v) `A←A+B)`

(vi) `4+2*B←6+2)`

(vii) `C←(((B←2)↑(C←4))↑.5)↑.5)`

(viii) `(1.E+1/B←C)*2+C←5)`

(ix) `A←B+C)`

(x) `A←9↑1/2)`

Q4. This question should be tackled at the first practice session on the ACL-NOVA terminal. Stored statements should not be used on this occasion to solve the problem; simply use the terminal as though it was a desk calculator. (Variables may be used to hold arithmetic constants, *e.g.* `Z←5.6`).

(a) Determine the roots of the quadratic equation $ax^2 + bx + c = 0$ for a = 1, b = .9, and c = .2. (The roots are given by

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}.)$$

(b) Consider the series

$$1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \frac{x^5}{5!} + \frac{x^6}{6!} + \frac{x^7}{7!} + \ldots$$

Evaluate the first eight terms of this series for x = .919. What is the result when the terms are summed from right to left? Can you explain the different results? Which one would you expect to be the more accurate? Why?

(c) Are there any advantages in evaluating the above series in the form

$$1 + x(1 + \frac{x}{2}(1 + \frac{x}{3}(1 + \frac{x}{4}(1 + \frac{x}{5}(1 + \frac{x}{6}(1 + \frac{x}{7}))))))$$

Q5. Write each of the following algebraic formulae as ACL statements to calculate the value of y. Use any convenient valid names for the variables, which will be assumed to have been assigned values by previous steps of the program:

(i) $y = \frac{1}{2}(b+c)$

(ii) $y = (a+b)^{2}/3$

(iii) $y = (\frac{1}{a} + \frac{1}{b} + \frac{1}{c})^{-1}$

(iv) $y = 1 + \frac{x}{1!} + \frac{x^2}{2!} + \frac{x^3}{3!}$ (n! = 1x2x...x(n-1) x n)

(v) $y - x = a - \pi y$ (π = 3.141592)

(vi) $\sqrt{y} = u$

(vii) $x = \frac{1}{y} + \frac{1}{b} + \frac{1}{c}$

(viii) $c = 1 + \cfrac{1}{1 + \cfrac{a+b}{y}}$

(ix) $y = e^{x}$

(x) $y = ae^{bx}$

Q6. Write the necessary statements in a program to calculate the variables given by the following expressions. Use any convenient valid names for the variables. You may assume that variables on the right have been assigned values by previous steps of the program and that the values do not require special consideration in evaluating the expressions - for example $a \neq 0$ in (i).

(i) $x = \frac{1}{2a}(-b + \sqrt{b^2-4ac})$

(ii) $s = \sqrt{x^2 + y^2 + z^2}$

(iii) $u = \tanh x = \frac{\frac{1}{2}(e^x - e^{-x})}{\frac{1}{2}(e^x + e^{-x})}$

(iv) $v = \tan x$

(v) $h = 1 - \frac{x^2}{2!} + \frac{x^4}{4!}$

(vi) $y = 1 - e^{-x} - \frac{1}{5} e^{-2x} - \frac{1}{25} e^{-3x}$

(vii) $c = \ln \left| \frac{1}{1+a^3} \right|$

(viii) $g = \left(\frac{\pi}{xy}\right)^{\frac{1}{2}} \sin\left(\frac{xy}{\pi}\right)$

(ix) $y = \left(e^{ax} + e^{-\sqrt{ax}}\right)/3$

(x) $z = \frac{-\tan^{-1}(x/a)}{1 + u/a}$

Try this exercise at the second ACL-NOVA terminal session.

Q7. Write an ACL program that will

(i) Read the four coefficients of a cubic polynomial $f(x) = a+bx+cx^2+dx^3$ by means of the ACCEPT statement.

(ii) Read the estimate x_o of one root of the equation $f(x)=0$ through another ACCEPT statement.

(iii) Improve the estimate of the root by the Newton-Raphson method, *i.e.*

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)} \quad .$$

(iv) The process can be considered to have converged if

$$\frac{|x_{n+1} - x_n|}{|x_n|} < .001$$

(v) Print out the improved estimate of the root.

(vi) Allow only five iterations. If convergence has not been obtained print a message warning of this.

(vii) Repeat from step (i)

When you test your program try the polynomial

$$f(x) = x^3 - 1.5x^2 - 1.5x + 1,$$

and take $x_o = .4$ as your estimate.

Q8. This question is only intended for those students who have completed *Q7* at the second terminal session.

Read in a set of ten numbers. Arrange for these numbers to be sorted into descending order. Print the ordered list out to check your program.

Q9. So far we have been building up experience in ACL programming in order to tackle the reactor power surge problem. Now turn to Dr. Clancy's notes where the mathematical formulation of the problem is given, and develop the ACL program necessary to obtain a numerical solution.

ANSWERS

A1. (i) .005,
(ii) 501,
(iii) invalid syntax,
(iv) 14,
(v) -1,
(vi) log error,
(vii) 1,
(viii) syntax error,
(ix) zero division,
(x) syntax error,
(xi) 147.

A2. Valid variables: X, XY, X4Y, RATE, SUM, X1Y2, FIVE, BLOT;
Invalid variables: 3XY, PRINC, Rate, 6IX.

A3. (i) 17 assigned to A,
(ii) 17 printed,
(iii) 50 assigned to C,
54 assigned to B,
108 printed,
(iv) 71 printed,
(v) 71 assigned to A,
71 printed,
(vi) 8 assigned to B,
20 printed,
(vii) 4 assigned to C,
2 assigned to B,
2 assigned to C,
(viii) 2 assigned to B,
5 assigned to C,
15 printed,
(ix) 7 assigned to A,
(x) 4.5 assigned to A.

A4. (a) Roots of equation are - .5, and - .4.

(b) Summing from left to right gives 2.504263, while from right to left the result is 2.504254. When summing from left to right the largest terms are evaluated and accumulated through addition first, while in summing from left to right the smallest are first considered. Bearing in mind that the NOVA computer operates on floating point numbers with approximately seven decimal digit representation, it follows that round off errors are greater when the larger numbers are summed first, *e.g.* adding 1.E+5 and 1.E-2 produces the result 1.E+5.

A5. (i) Y←.5*(B+C),

(ii) Y←(A+B)*(A+B)/3 or Y←I*(I←A+B)/3

(iii) Y←1/(1/A+1/B+1/C),

(iv) Y←1+X*(1+X*(.5+.1666666*X)),

(v) Y←(X+A)/4.141592,

(vi) Y←U*U,

(vii) Y←1/(X-1/B-1/C),

(viii) Y←(C-1)*(A+B)/(2-C),

(ix) Y←EXP(X),

(x) Y←A*EXP(B*X).

A6. (i) X←(-B+SQR(B*B-4*A*C))/(2*A),

(ii) S←SQR(X*X+Y*Y+Z*Z),

(iii) U←(W-T)/(W+T←1/W←EXP(X)),

(iv) V←SIN(X)/COS(X),

(v) H←1-W*(.5-.0416667*W←X*X),

(vi) Y←1-W*(1+W*(.2+.04*W←EXP(-X))),

(vii) C←-LOG(ABS(1+A*A*A)),

(viii) G←1/SQR(W)*SIN(W←X*Y/3.141592),

(ix) Y←(EXP(W)+EXP(-SQR(W←A*X)))/3,

(x) Z←-ATN(X/A)/(1+U/A).

A7.
```
110 3   ACCEPT A,B,C,D
120     ACCEPT X0
130     I←1
140 6   XN←X0-(A+X0*(B+X0*(C+D*X0)))/(B+X0*(2*C+3*D*X0))
150     IF(ABS(XN-X0)/ABS(XN).LT..001) GO TO 5
160     X0←XN
170     IF(I←I+1..LE.5) GO TO 6
180     TYPE 'ROOT NOT LOCATED IN 5 ITERATIONS'
190     GO TO 3
200 5   TYPE 'ROOT IS',XN
210     GO TO 3
```

A8.
```
110     I←1
120 5   ACCEPT X(I)
130     IF(I←I+1..LE.10) GO TO 5
140     I←1
150 7   J←I+1
160 6   IF(X(J).LE.X(I)) GO TO 2
170     TEMP←X(J)
```

```
180      X(J)←X(I)
190      X(I)←TEMP
200 2    IF(J←J+1..LE.10) GO TO 6
210      IF(I←I+1..LE.9) GO TO 7
220      I←1
230      TYPE X(I)
240      IF(I←I+1..LE.10) GO TO 8
250      STOP
```

APPENDIX 3C

LIST OF ACL STATEMENTS

A. IMMEDIATE STATEMENTS

Comments

Arithmetic statements

Arithmetic expressions

RUN

GO TO {arith stmt or exprn}

LIST [:] [␣ arith stmt or exprn[,arith stmt or exprn]]

SYMBOLS [:]

CLEAR [␣ variable[,variable]...]

SPACE

FTRON

FTROFF

TRON [␣ arith stmt or exprn]

TROFF [␣ arith stmt or exprn]

TYPE [␣ { [: [:] ... ; [;] ...] Operand [{ : [:] ... ; [;] ... } operand]... | : [:] ... ; [;] ... }]

where the operands take the form, [<arith stmt or exprn>,]

{ 'characters' ["] arith stmt or exprn }

PB ␣ {arith stmt or exprn}

PA ␣ {arith stmt or exprn}

STOP

END

SUSPEND [:]

EDIT ␣ {arith stmt or exprn}

System Messages

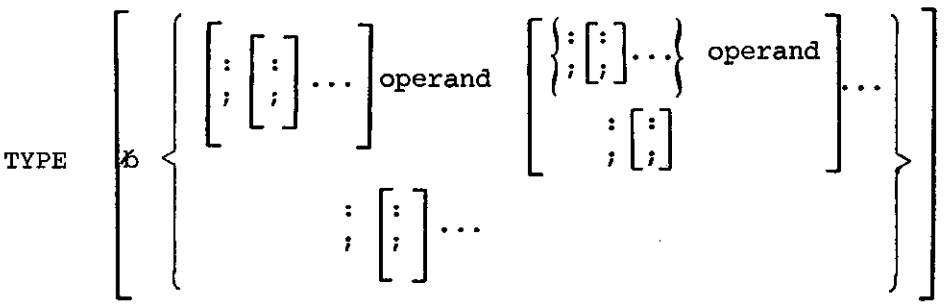
B. STORED STATEMENTS

Comments

Arithmetic statements

Arithmetic expressions

GO␣TO␣ {arith stmt or exprn}



where the operands take the form, [<arith stmt or exprn>,]

{ 'characters' }
{ ["] arith stmt or exprn }

ACCEPT ↓ {variable[,variable] ...}

CALL ↓ {arith stmt or exprn}

RETURN

CONTINUE

STOP

IF ({arith stmt or exprn} { .EQ. / .NE. / .GT. / .GE. / .LT. / .LE. } {arith stmt or exprn}) ↓ { arith stmt or exprn / TYPE stmt / ACCEPT stmt / GO TO stmt / CALL stmt / RETURN stmt / CONTINUE stmt / STOP stmt }

PAUSE

END

CHAPTER 4

LOADING AND SAVING ACL PROGRAMS ON IBM360 DISK STORAGE

Lecture by

R.P. BACKSTROM

4.1 PROCEDURE

To save an ACL program on IBM360 disk storage, enter

[#SAVE PROGNAME,INT/ACCTNMBR]

followed by carriage return. PROGNAME may be replaced by any one to eight character program name, provided that the first letter is alphabetic and the rest are alphanumeric (*i.e.* one of A-Z or 0-9). INT represents the three initials of the user (as contained on his IBM360 job card) and, for the purposes of this Summer School, will be SSK. ACCTNMBR is the user's account number (also contained on his IBM360 job card) and is an eight character field containing a Division and Section code, followed by a five digit employee number. The Summer School account number is AM290060.

To save both the program statements and the ACL symbol table, enter

[#SAVES PROGNAME,INT/ACCTNMBR]

In either case, the response from the IBM360 computer will be

[-PROGNAME-SAVED AT 10.20 AM ON 74.315]

indicating the time of day at which the program was saved (if the program was being saved for the first time), or

[-PROGNAME-REPLACED AT 10.20 AM ON 74.315]

(if it was replacing an earlier version).

To avoid confusion between different users' saving programs under the one 'Summer School' account, it is recommended that program names commence with the three initials of the particular user. In this way, replacement of other people's programs and data can be avoided. For example, if William J. Smith wished to save his program, PLOT1, he should enter

[#SAVE WJSPLOT1,SSK/AM290060]

To delete programs no longer required from IBM360 disk storage, enter CNTRL,BELL to clear the work area and then enter a normal SAVE command *e.g.*

[#SAVE WJSPROG1,SSK/AM290060]

This 'null' program SAVE request is interpreted as a DELETE request. The response will be either

[-WJSPROG1-DELETED AT 12.43 PM ON 74.315, or]

[-WJSPROG1-NOT LOCATED IN ACL LIBRARY]

depending on whether or not the program WJSPROG1 existed. This tidying procedure may be required from time to time because disk storage is not unlimited.

To load programs from the ACL library, enter

[#LOAD PROGNAME,INT]

followed by carriage return. In the case of the Summer School, this will be

[#LOAD WJSDATA4,SSK]

Loading does not require the specification of an account number to enable various ACL programs to be shared by other users. For saving ACL programs, however, the account number requirement gives users protection against accidental replacement or deletion of programs.

CHAPTER 5

ANALOGUE AND HYBRID COMPUTERS

Lecture by

C.P. GILBERT

CONTENTS

5.1 INTRODUCTION

5.1.1 Dynamic Systems

Many advances in science and engineering are possible only because of our ability to use mathematical equations to describe the behaviour of complicated systems.

Here we are concerned with what are known as *dynamic* systems, *i.e.* those that vary with time, which are often described using differential equations. A simple example of a dynamic situation is the movement of a ball bouncing on an uneven surface and, if necessary, equations could be formulated to describe this motion and solutions obtained.

A nuclear reactor is a much more complicated dynamic system, but equations are available to describe how the power and temperature vary with time following a disturbance. Solutions to these equations can help us find out if the fuel would melt or not, and would thus assist the designers to ensure that fuel melting could not occur in an actual reactor. However, differential equations of this type are not easy to solve in practice.

5.1.2 Analogues

When dynamic systems are examined in detail, one important property emerges. Many electrical, mechanical, thermal and other systems can be described by equations of the same *form*, although the actual numbers involved may be different in each case. Figure 1 shows simple examples of three systems, each with energy decaying away. The current i in an inductive circuit, the velocity v of a flywheel with a brake, and the temperature θ of a cup of hot water which is cooling down, can all be described by equations of the form:

$$\frac{dx}{dt} = -Kx \quad . \qquad \ldots(1)$$

Systems which resemble each other in this way are called *analogues* of one another, and if each is given an equivalent disturbance they will all behave in exactly the same manner, although probably at different speeds.

Thus although the three systems are different in physical form, their *dynamic* properties are identical. There is then the possibility that we can examine one of them (that happens to be convenient), in order to find out how the others behave. This can assist us with the solution of complicated differential equations and, in particular, with our reactor problem.

5.1.3 Simulation

One way to examine the reactor behaviour would be to do an experiment,

that is,purposely inject the disturbance and then measure the reactor power and temperature as they vary with time. Although this can be done on a small experimental reactor like MOATA, with a full size power reactor the experiment would be slow, very expensive and probably dangerous. However, if we could find some sort of analogue of the reactor (*i.e.* another physical system, or 'model', having the same dynamic behaviour), then it would be much simpler, safer and cheaper to do the same experiment on the analogue. This idea has been found to be so successful in some applications that, instead of looking for convenient analogues in a haphazard way, special pieces of equipment have been built solely for this purpose.

These Analogue Computers, as they are called, consist of a number of units which can be put together like building blocks to form analogues of different systems; accurate measurements can then be made on the resulting model. The process of doing an experiment on a computer model instead of on the real system is known as *Simulation*.

5.1.4 Computing Operations

As will become clear, addition and integration are the most important processes that the units of an analogue computer have to perform, and a number of methods are available.

Figure 2a shows how *addition* can be performed using a liquid; if the contents of the smaller containers are emptied into a sufficiently large container, the final volume of liquid is the sum of the initial volumes:

$$V = v_1 + v_2 + v_3 + v_4 + v_5 \quad .$$

The more important process of *integration* can be achieved as shown in Figure 2b. The height H of the fluid in a container of base area A is the integral, with respect to time, of the fluid flow F (volume/sec) as determined by the tap:

$$H = \frac{1}{A} \int_0^t F \, dt \quad .$$

These simple analogues would be of no use in practice; they are inaccurate, slow, unsuitable for interconnection, and probably wet. However, there are much better analogue processes available; the most important of them uses an electronic amplifier and is described in the following section.

5.2 OPERATIONAL AMPLIFIER CIRCUITS

While it is not necessary to understand this section in detail to follow the rest of the lecture, you should be clear about the overall

behaviour of a potentiometer (Figure 3b), of an adder (Figure 3c), and of an integrator (Figure 4c).

An operational amplifier has the following properties:

- Very high voltage amplification, or 'gain' K (say 10^6).
- Negative gain (a *positive* input produces a *negative* output, and *vice versa*).
- Works at d.c. as well as at a.c., for all frequencies up to perhaps 10^6 Hz.

For computing purposes, such amplifiers are used in a feedback circuit which exchanges the high voltage gain for other more desirable properties. The circuit of Figure 3a is arranged so that the voltages v_1, v_2, v_3 and V_o are all of the order of a few volts. Then, if $K = 10^6$, the amplifier input voltage u is equal to $-V_o/K$, which never exceeds a few microvolts and can usually be neglected. For instance, if the combined effect of all the inputs is positive, u tries to go positive: this causes V_o to go negative by a very much larger amount, which opposes the rise in u because of R_f. Finally u ends up very slightly positive, causing a negative output V_o.

If we assume that u is zero, the current i is the sum of the input currents:

$$i = \frac{v_1}{R_1} + \frac{v_2}{R_2} + \frac{v_3}{R_3} \quad .$$

This current cannot enter the amplifier, but is drawn through R_f by V_o, and so

$$i = -\frac{V_o}{R_f} \quad .$$

Eliminating i,

$$-\frac{V_o}{R_f} = \frac{v_1}{R_1} + \frac{v_2}{R_2} + \frac{v_3}{R_3}$$

leading to

$$V_o = -\left[v_1 \frac{R_f}{R_1} + v_2 \frac{R_f}{R_2} + v_3 \frac{R_f}{R_3} \right] \quad .$$

Thus the output is minus the sum of the input voltages. The resistance ratios $\frac{R_f}{R_1}$ are normally fixed at convenient values, such as 1 or 10, and variable coefficients are introduced using potentiometers. (A potentiometer, represented by a circle as shown in Figure 3b, is simply a device for

reducing the size of a voltage by an amount which can be set very accurately.) It is a pity that the amplifier gives a reversal in sign, but it is unavoidable, and causes little difficulty.

The complete *adder* circuit is conventionally drawn as shown in Figure 3c, the values of the resistance ratios being marked *only* if they are other than unity. Then

$$V_o = -[0.3\ v_1 + 1.6\ v_2 + v_3] \quad . \qquad \ldots(2)$$

Note that all voltages are measured with respect to earth, although the earth connection itself is omitted.

In the *integrator* circuit of Figure 4a, a feedback capacitor C is used. Assuming as before that u = o,

$$i = \frac{v}{R} \quad .$$

Again, the current is constrained to flow into the feedback component, but in this case the voltage is proportional to the integral of the current i with respect to time t, namely

$$V_o = -\frac{1}{C} \int_o^t i\ dt,$$

and substituting for i shows that

$$V_o = -\frac{1}{CR} \int_o^t v\ dt \quad .$$

A constant value of v causes V_o to change at a constant rate; a sine input gives a cosine output and so on (Figure 4b). Unless otherwise shown, the time constant CR can be assumed to be unity, and the integrator circuit is conventionally drawn as shown in Figure 4c, for which

$$V_o = -0.6 \int_o^t v_1\ dt \quad , \qquad \ldots(3)$$

or

$$-\frac{dV_o}{dt} = 0.6\ v_1 \quad .$$

Accuracies better than 0.1 per cent can be obtained without difficulty for adders and integrators of the type shown.

5.3 ELECTRONIC ANALOGUE COMPUTERS

An electronic analogue computer consists of a number of operational amplifiers which can be used for addition, integration, multiplication and

a range of other functions; facilities are provided which permit the interconnection and switching of the computing circuits, and which allow accurate measurements to be made on them. The *problem variables* in which we are interested (flux, velocity, temperature or force, for instance) are all represented in the computer by *voltages*. These voltages may vary quite slowly, and can then be read on a voltmeter, or they may change so quickly that an oscilloscope is required to observe them.

A medium-sized machine might have about 100 amplifiers, including perhaps 30 integrators, and could thus perform 30 integrations at the same time.

Consider the circuit of Figure 5. The extra input on top of the integrator is inverted, and supplies a *fixed* voltage b to the output as an 'initial condition' before the integration starts, but has no other effect. When switch A is closed, this *defines* the instant which the computer regards as t = 0; at this time the output V = b. We have now made the integrator input equal to V, and so the circuit obeys the equation

$$-\frac{dV}{dt} = V \qquad \text{or} \qquad \frac{dV}{dt} = -V \qquad \ldots(4)$$

This is basically the same as Equation (1), which describes the systems of Figure 1, and so the circuit of Figure 5 is simply one more analogue, having the same dynamic properties as the other three systems. [Returning to Equation (4), analysis shows that if we let $V = ke^{-t}$, where k is an unknown constant, then differentiating we get

$$\frac{dV}{dt} = -ke^{-t} = -V \ .$$

This demonstrates that $V = ke^{-t}$ is a solution of Equation (4). Since we have made V = b at t = o, substitution shows that k = b, and so the solution is $V = be^{-t}$.]

The circuit of Figure 5 'solves' Equation (4) by producing a voltage proportional to be^{-t} each time switch A is closed. V starts off positive and, *via* the integrator, forces itself to get smaller. As it does so, it falls more slowly, reaching zero exponentially.

Switches such as A, and many other controls required by the computer, are usually omitted from the computing circuit - their presence is assumed.

Summarising, should we wish to examine one of the systems of Figure 1, possibly with a very complicated series of disturbances, the simplest, cheapest and most accurate way of doing the experiment would be to apply a voltage representing the disturbances to the circuit of Figure 5.

5.4 PROBLEM SOLUTION

In the previous section we started with a computer circuit and *analysed* its behaviour. The usual process is the other way round - we are given an equation and have to *design* a circuit which will solve it, resulting in the process illustrated in Figure 6. The equivalence between the physical system and the analogue circuit is very marked, and examination of the behaviour of the latter, used as a working model, provides considerable insight into the operation of the original system. In fact, one major advantage of analogue computers is that they form a means of learning, and in some cases simulators behave so much like the original system that they are used to train operators.

A simplified equation describing a reactor has been given as

$$\frac{dp}{dt} = - [p - 1 - be^{-t}] \, p \quad ,$$

which we will examine, in a more convenient form, as

$$\frac{dp}{dt} = - qp \quad , \qquad \ldots(5)$$

where

$$q = [p - 1 - be^{-t}] \quad ,$$

and

$$p = 1 \text{ at } t = 0 \quad .$$

We wish to find out how large b may be without causing p to exceed 1.2.

Consider the circuit of Figure 7. Apart from a reversal in sign, integrator 1 is connected as in Figure 5, so we know it produces $-be^{-t}$, which can be used to represent the input disturbance. Adder 2 combines the terms which make up q, and multiplier 3 forms the product qp, which Equation (5) shows to be equal to $-\frac{dp}{dt}$. Thus, integrator 4 provides p. This circuit is an analogue of the reactor described by Equation (5) and, once started with the correct initial conditions, it will behave in exactly the same way as the reactor. Unit 5 is a comparator which triggers when its total input passes through zero. It is used to indicate if p exceeds a peak of 1.2, and so tell us if the fuel would have started to melt or not.

As shown, the circuit would be adjusted and observed by an operator. He would take a number of solutions with a range of values of b until he found, by trial and error, the value which just failed to trigger the comparator. Voltages would be plotted on a chart, or displayed on an

oscilloscope, and the wanted result (the size of b) is the final setting of the potentiometer and would be read on a digital voltmeter.

Notice that all the computing operations occur simultaneously (in parallel) not in sequence as in a digital computer, and that the solution arises at a definite speed, *i.e.* the same speed as the reactor in our case. By using smaller capacitors in the integrators, the solution can be up to 10^4 times faster and, if many integrations are involved, the overall operation is faster than can be achieved by a digital computer. However, the high speed cannot be properly utilised by a human operator.

A major drawback to pure analogue computers is their inability to store information, and in general their usefulness is limited to the solution of ordinary differential and straightforward algebraic equations. However, within this field they provide valuable insight as well as useful answers.

5.5 HYBRID COMPUTERS

A recent development, whose full impact has not yet been felt, is the Hybrid Computer. This consists of an analogue computer, a general purpose digital computer, and an interface. The latter provides the facilities required for the two machines to cooperate effectively (Figure 8).

The analogue computer allows high speed, parallel computation. The digital computer can be programmed to:

- Perform the scaling calculations and check the analogue circuit.
- Act as a very high speed operator, who readjusts the computer before each solution, as determined by the preceding solution.
- Perform parts of the computation which the analogue computer finds difficult.

(One might also express the same idea by saying that the analogue computer becomes one of the peripherals upon which the digital computer can call when required.)

In the problem of Figure 7 for instance, having written an executive program, the operator would only have to type in the permissible power peak (1.2 in our case), and the computer would then print out the size of disturbance b which just avoids this peak. The program would have gone through almost the same trial and error process as the human operator using the circuit, but would do it perhaps 10^5 times faster.

There are few problems which a hybrid computer cannot profitably undertake, but much more experience is required in using these machines on large problems. We are doing our best to gain that experience now.

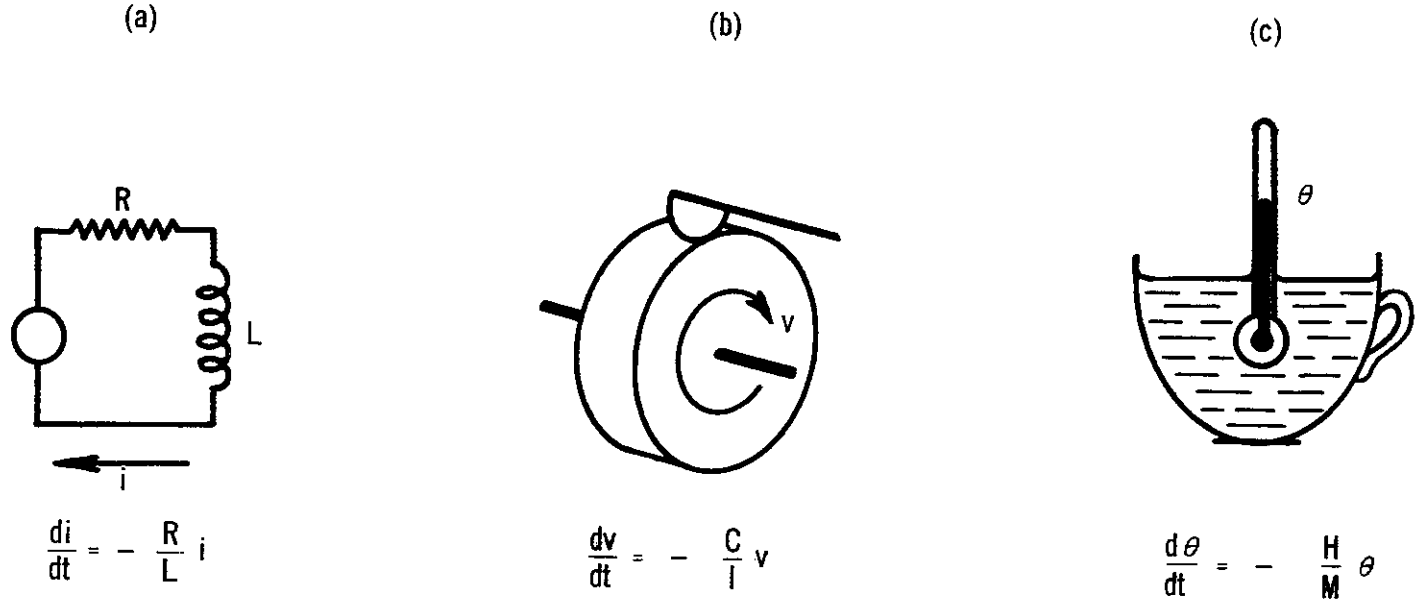


FIGURE 1. (a) Current i in an inductive circuit (b) Velocity v of a flywheel with a brake (c) Temperature θ of a cup of hot water

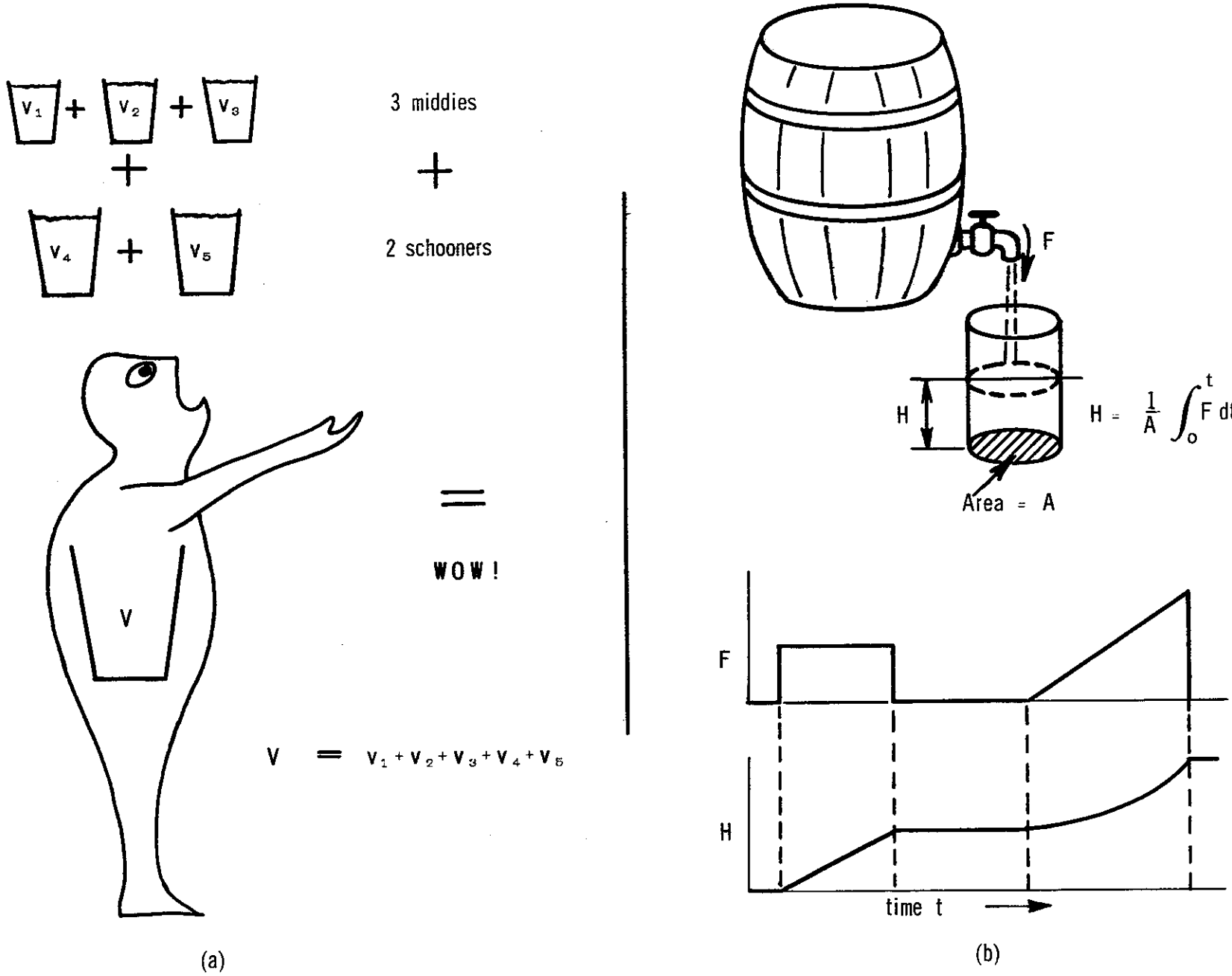
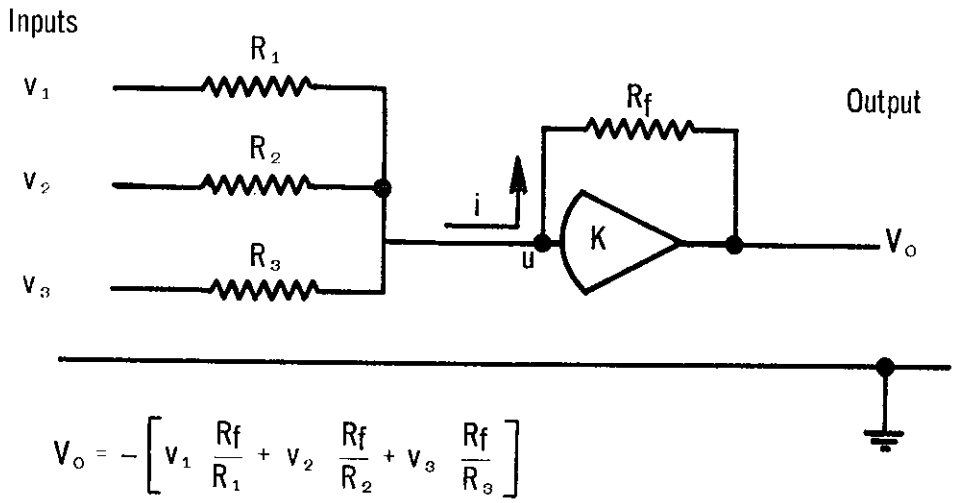
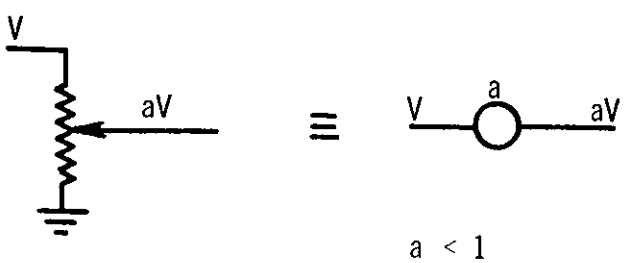


FIGURE 2. Liquid analogues for (a) addition, using volumes, and (b) integration



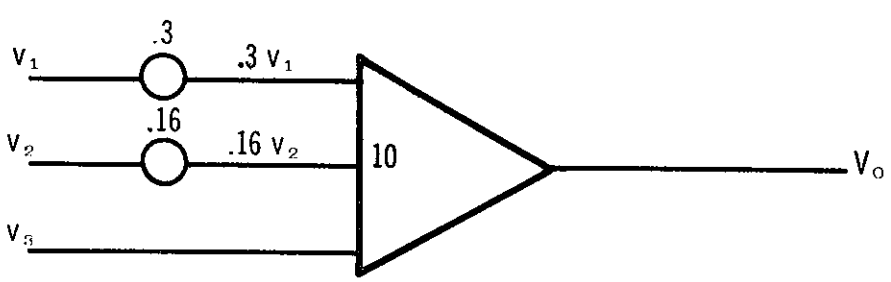
(a)

$$V_o = -\left[v_1 \frac{Rf}{R_1} + v_2 \frac{Rf}{R_2} + v_3 \frac{Rf}{R_3} \right]$$



(b)

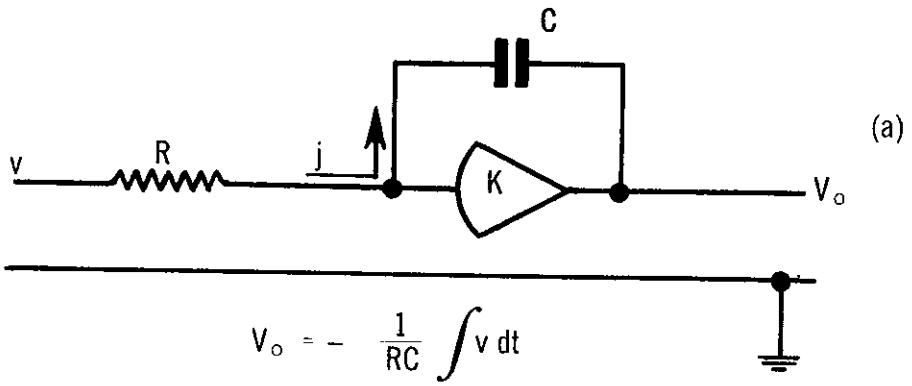
$a < 1$

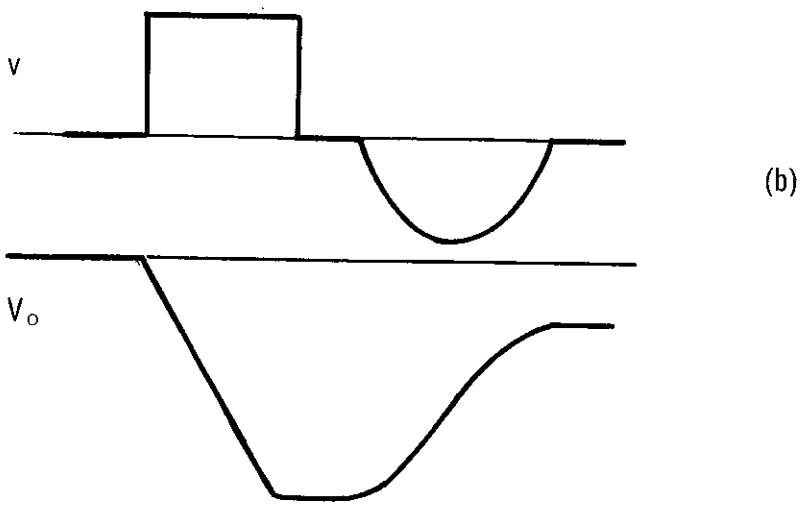


(c)

$$V_o = -\left[.3\, v_1 + 1.6\, v_2 + v_3 \right]$$

FIGURE 3. A circuit for addition: (a) Circuit details (b) Potentiometers (c) Circuit using conventional symbols: the whole of the circuit (a) is contained within the triangle





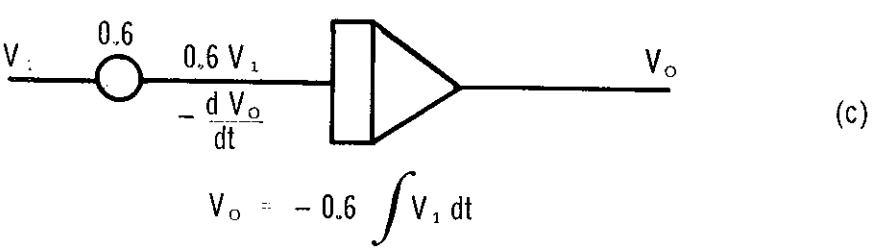


FIGURE 4. A circuit for integration: (a) Circuit details (b) Typical waveforms (c) Circuit using conventional symbols: the whole of circuit (a) is contained within the special triangular symbol. More than one input circuit is permitted

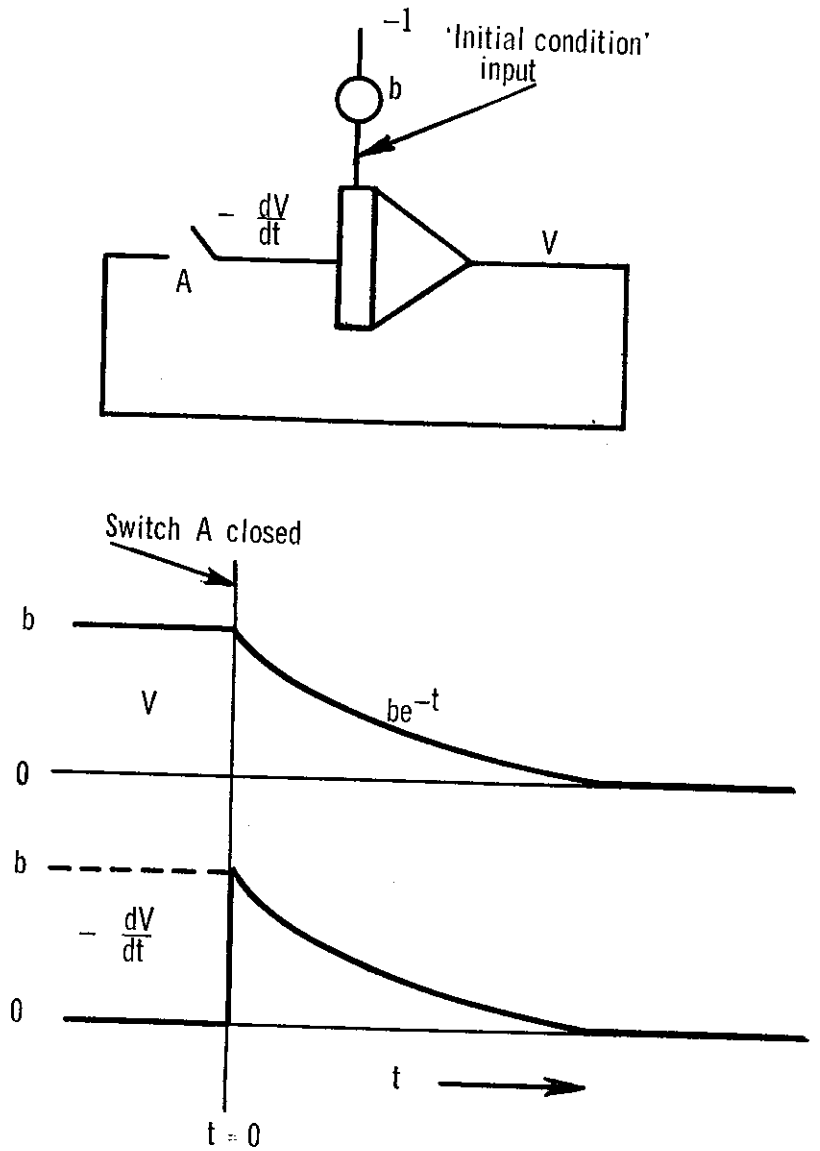


FIGURE 5. A circuit for the solution of $\frac{dV}{dt} = -V$ and typical waveforms. The input via b only provides the starting voltage

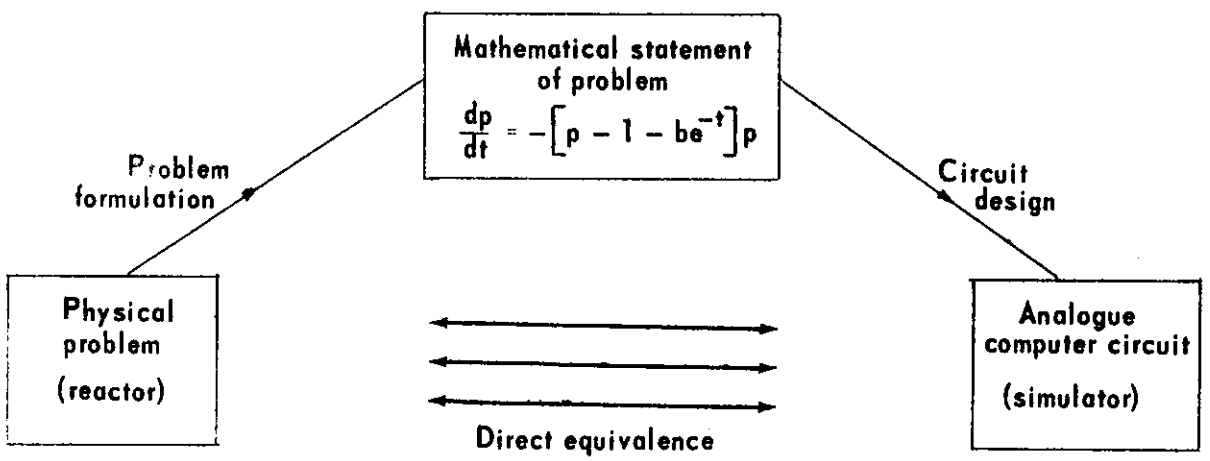


FIGURE 6. Pictorial representation of the analogue method of problem solving

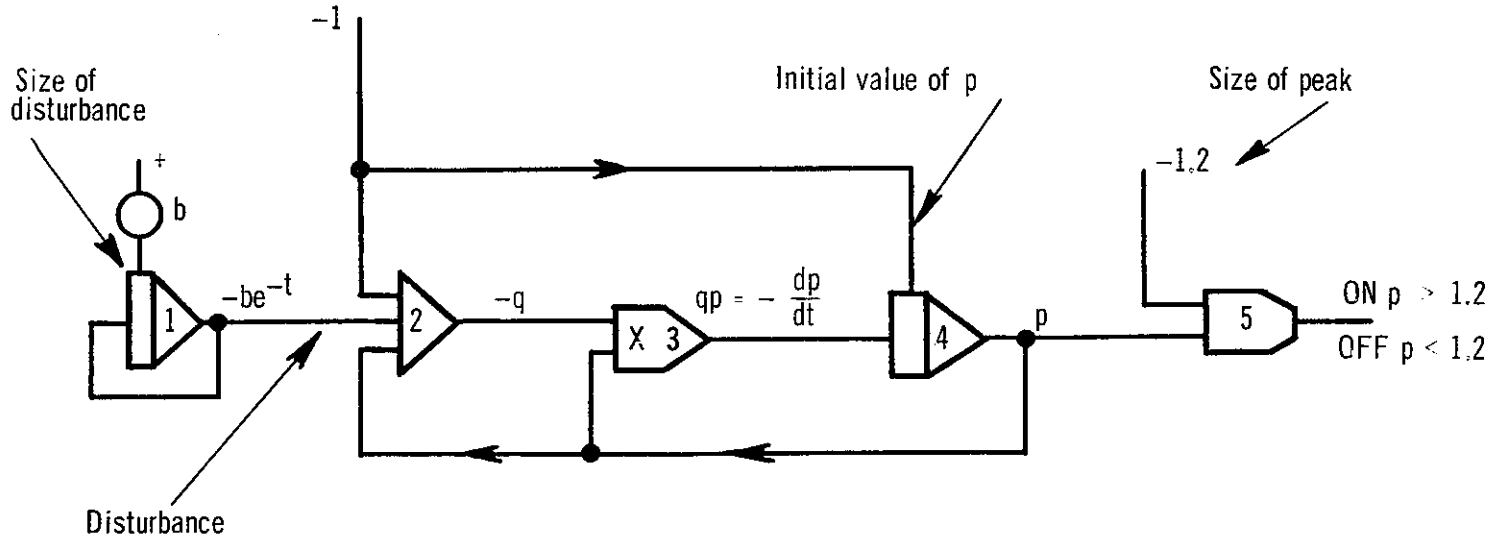


FIGURE 7. A circuit for the solution of $\frac{dp}{dt} = -\left[p - 1 - be^{-t}\right]p$ with an indication when p>1.2

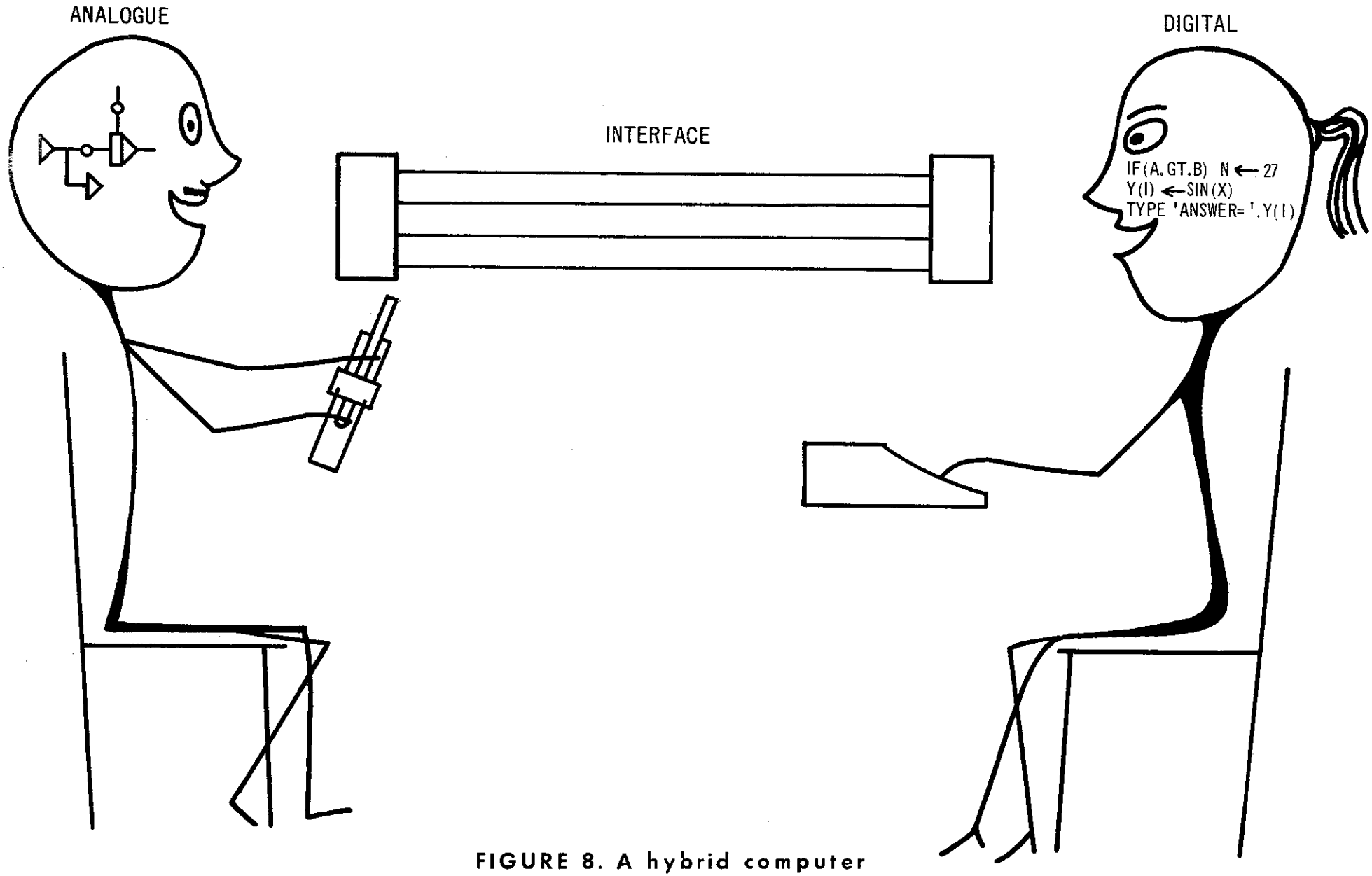


FIGURE 8. A hybrid computer