HHEC-S 17

AAEC/S17

AAEC/S17

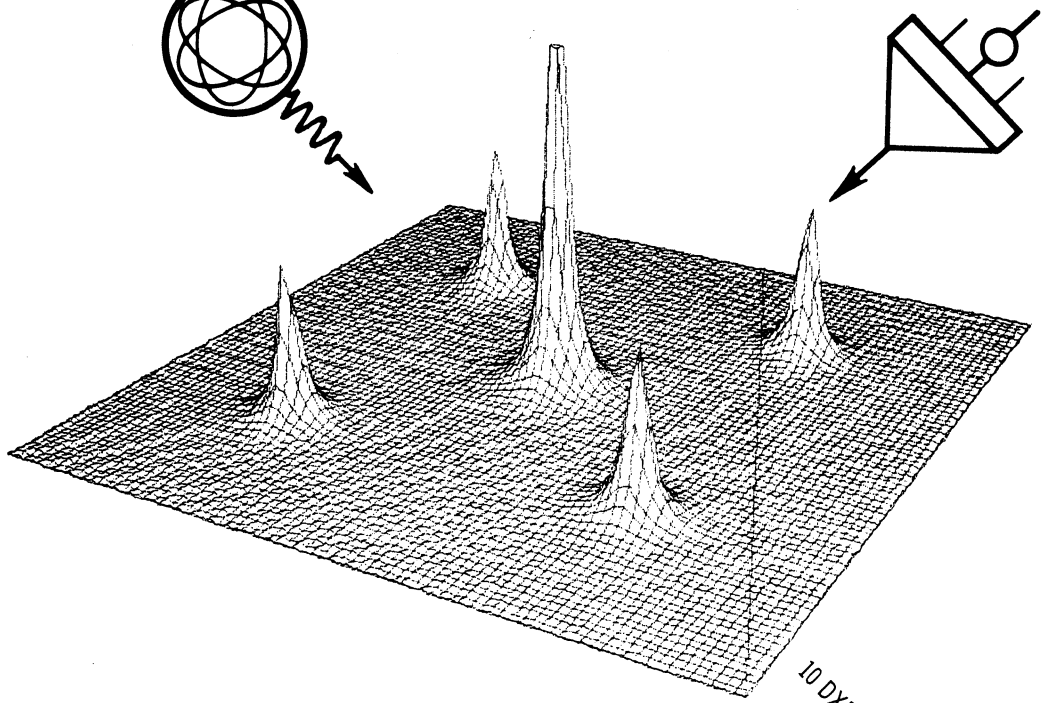
AUSTRALIAN ATOMIC ENERGY COMMISSION
RESEARCH ESTABLISHMENT
LUCAS HEIGHTS

SUMMER SCHOOL, 1976

DOWN BUT NEVER OUT – THE MATHEMATICS AND COMPUTATION OF EXPONENTIALS ARISING IN THE FIELDS OF PHYSICS, CHEMISTRY, BIOLOGY, ...

Edited by

P.J.F. Newton



$\frac{dx}{dt} + \lambda x = 0$

10 DXDT = –ALAM*X

December 1975

ISBN 0 642 99725

COVER DESIGN

The theme of the design is exponentials in action. The somewhat tall and challenging mountain peaks depicted as the centre piece is simply a plot of the number of neutrons around five sources emitting neutrons into an absorbing block of material. Each mountain consists of an exponential curve rotated about a central axis.

AUSTRALIAN ATOMIC ENERGY COMMISSION
RESEARCH ESTABLISHMENT
LUCAS HEIGHTS

SUMMER SCHOOL 1976

DOWN BUT NEVER OUT - THE MATHEMATICS AND COMPUTATION OF EXPONENTIALS ARISING IN THE FIELDS OF PHYSICS, CHEMISTRY, BIOLOGY,...

Edited by

P.J.F. Newton

ABSTRACT

These notes are for a Summer School which will introduce mathematically minded year 12 High School students to scientific computing covering a variety of scientific disciplines. All of the disciplines concentrate on examples that follow the basic exponential behaviour of two coupled first order differential equations.

The various problems pursued are from the disciplines of mathematics, physics, chemistry, biology, and include consideration of other exponential processes such as competing population problems such as between sharks and little fishes.

Much of the course is devoted to electronic computing. The student
(a) will set up a digital computer for the least squares problem, and
(b) will use an analogue computer to study competing exponential processes.

National Library of Australia card number and ISBN 0 642 99725

The following descriptors have been selected from the INIS Thesaurus to describe the subject content of this report for information retrieval purposes. For further details please refer to IAEA–INIS–12 (INIS: Manual for Indexing) and IAEA–INIS–13 (INIS: Thesaurus) published in Vienna by the International Atomic Energy Agency.

[1] COMPUTER CALCULATIONS; FUNCTIONS; MATHEMATICAL MODELS; MATHEMATICS

[2] LEAST SQUARES FIT; MATHEMATICS; NON LINEAR PROBLEMS; DIAGRAMS

[3] COMPUTER CALCULATIONS; FORTRAN; MATHEMATICS; PROGRAMMING

[4] AMPLIFIERS; ANALOG COMPUTERS; DIFFERENTIAL EQUATIONS; HYBRID COMPUTERS; MATHEMATICAL MODELS; POPULATION DYNAMICS; SIMULATION

[5] PROGRAMMING; LANGUAGES; COMPUTERS

[6] CAPTURE; CHAIN REACTIONS; CLADDING; FISSION; FUEL ELEMENTS; MOATA REACTOR; RADIOACTIVITY; SCATTERING

[7] CHEMICAL REACTIONS; DECOMPOSITION; KINETICS; OXIDATION; REDUCTION

[8] BIOLOGICAL CELLS; BIOLOGICAL RADIATION EFFECTS; SURVIVAL CURVES; TARGET THEORIES; MATHEMATICAL MODELS; PROBABILITY

CONTENTS

CHAPTER 1

THE EXPONENTIAL FUNCTION

Lecture by

J.P. POLLARD

ABSTRACT

Mathematical properties of the exponential function, $y(x) = e^x$ are studied in order to set the stage for material to be presented at the Summer School. Nowadays static display of strange relationships for their own intrinsic beauty is not enough, we need to pursue the relationships into computational action. Following the trend, practical digital computer and electronic calculator computation of exponentials is introduced.

CONTENTS

1.1 THE POWER AND EXPONENTIAL FUNCTION

A general class of power function is given by the expression

$$y(x) = p^x, \qquad \ldots(1.1)$$

e.g. $y(3) = p^3 = ppp$ and $y(0) = p^0 = 1$,

where p is a number, or base, chosen to suit the problem and needs of the user. A simple example is given by the familiar antilog function arising when $p = 10$,

$$y(x) = 10^x \qquad \ldots(1.2)$$

so that, taking logarithms (to the base 10), we have

$$\log y = x. \qquad \ldots(1.3)$$

It so happens that, in computation, the base $p = 10$ is convenient and often used. However, in the physical world one particular value of p arises in a natural way corresponding to

$$p = e = 2.71828\ldots \quad , \qquad \ldots(1.4)$$

a number as highly regarded in the mathematical world as the number

$$\pi = 3.14159\ldots \qquad \ldots(1.5)$$

(Digression...

Having met π as a really special number, you may not like to see a competitor enter the field. Well don't worry! π and e are blood cousins through the strange but beautiful relationship

$$e^{\pi\sqrt{-1}} = -1, \quad \text{(derived by Euler).)} \qquad \ldots(1.6)$$

Let us move on to investigate the properties of the power function given by equation (1.1). Some obvious properties we notice are that

$$\left.\begin{array}{l} y(x+z) = p^{x+z} = p^x p^z = y(x)y(z) \\ \text{and} \quad y(x) = \left(p^{\frac{x}{n}}\right)^n = [y(\frac{x}{n})]^n \end{array}\right\} \qquad \ldots(1.7)$$

then a ratio required in calculating the function derivative is simplified in the manner

$$\frac{y(x+\delta x) - y(x)}{\delta x} = \frac{y(x)y(\delta x) - y(x)}{\delta x}$$

$$= y(x)\left[\frac{y(\delta x) - 1}{\delta x}\right]$$

$$= \ y(x)\,\left[\frac{p^{\delta x} - 1}{\delta x}\right] \quad .$$

From the definition of a derivative we have

$$\frac{dy}{dx} = \lim_{\delta x \to 0} \left[\frac{y(x+\delta x) - y(x)}{\delta x}\right] \quad ,$$

hence $$\frac{dy}{dx} = y(x) \lim_{\delta x \to 0} \left[\frac{p^{\delta x} - 1}{\delta x}\right] \quad .$$

We find that a particular value of p(=e) exists such that

$$\lim_{\delta x \to 0} \left[\frac{e^{\delta x} - 1}{\delta x}\right] = 1 \quad , \qquad \ldots(1.8)$$

from which our central result is obtained for the so-called exponential function,

$$\boxed{\begin{array}{l} \dfrac{dy}{dx} = y(x) \\ \text{or } \dfrac{de^{x}}{dx} = e^{x} \ , \\ \textit{i.e.}\ e^{x} \text{ is its own derivative.} \end{array}} \qquad \ldots(1.9)$$

Of almost equal importance is the integral result obtained from equation (1.9). We have

$$\int_0^x \frac{dy}{dx}\,dx = \int_0^x y(x)\,dx \quad ;$$

but

$$\int_0^x \frac{dy}{dx}\,dx = \int_0^x dy = [y(x)]_0^x = y(x) - y(0) = y(x) - 1 \quad ,$$

hence we have

$$\begin{array}{|l|} \displaystyle\int_0^x y(x)\,dx = y(x) - 1 \ , \\ \text{or } \displaystyle\int_0^x e^{x}\,dx = e^{x} - 1 \ \text{(definite integral)} \end{array} \ ,$$

$$\boxed{\begin{array}{l}\text{or } \displaystyle\int_0^x e^x dx = e^x \text{ (indefinite integral)} \\ \textit{i.e.} \quad e^x \text{ is its own integral.}\end{array}} \quad , \qquad \ldots(1.10)$$

A quick look at figure 1.1 will help us to visualise properties (1.9) and (1.10)

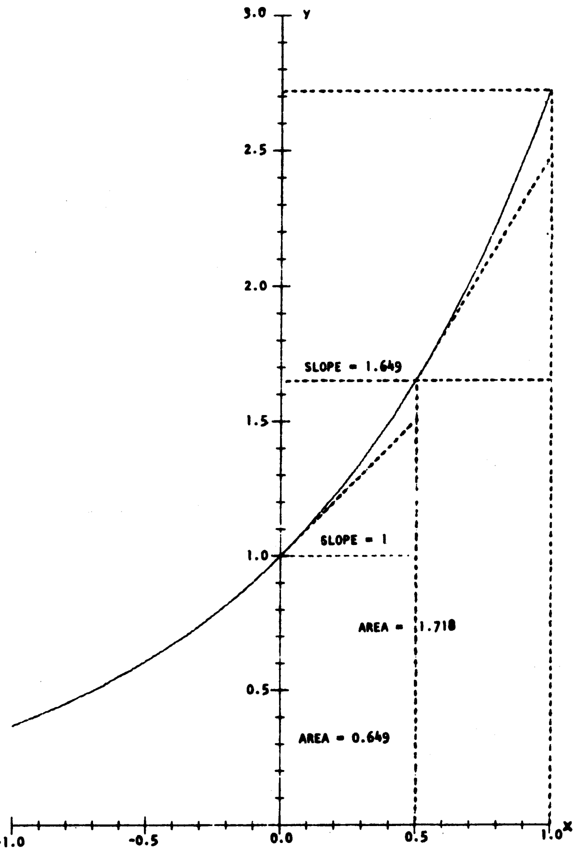


Figure 1.1 The function e^x

Changing to the variables required for this Summer School, a slightly more general exponential function is

$$x(t) = x_0 e^{\lambda t} , \qquad \ldots(1.11)$$

where x_0 and λ are constants. The rule for differentiating a function of a function shows us that

$$\frac{dx}{dt} = x_0 \frac{de^{\lambda t}}{d\lambda t} \; \frac{d\lambda t}{dt} = x_0 e^{\lambda t} \lambda = \lambda x(t) \quad .$$

Similarly we calculate the indefinite integral

$$\int x(t) dt = x_0 \int e^{\lambda t} \; \frac{d\lambda t}{\lambda} = x_0 \; \frac{e^{\lambda t}}{\lambda} = \frac{x(t)}{\lambda} \quad .$$

Collecting results we have

$$\boxed{\begin{array}{l} \dfrac{de^{\lambda t}}{dt} = \lambda e^{\lambda t} \\ \hline \displaystyle\int e^{\lambda t}dt = \frac{e^{\lambda t}}{\lambda} \\ \hline \text{if } \dfrac{dx}{dt} = -\lambda x \\ \text{with } x(0) = x_0 , \\ \text{then } x(t) = x_0 e^{-\lambda t} \end{array}} \qquad \ldots(1.12)$$

where the last results are obtained by considering equation (1.11) with λ negative.

Before we proceed much further, we will need to know more about the number e.

1.2 ESTIMATING THE NUMBER e

At present, all we know about the number e is contained in the expression

$$\lim_{\delta x \to 0} \left[\frac{e^{\delta x} - 1}{\delta x}\right] = 1 \quad .$$

Now if we consider the above equation for sufficiently small values of δx, we have

$$e^{\delta x} - 1 \simeq \delta x$$

then $$e^{\delta x} \simeq 1 + \delta x$$

and, taking logs,

$$\delta x \log e \simeq \log(1+\delta x)$$

$$\log e \simeq \frac{1}{\delta x} \log(1+\delta x)$$

$$\simeq \log(1+\delta x)^{\frac{1}{\delta x}}$$

$$e \simeq (1+\delta x)^{\frac{1}{\delta x}} \quad .$$

Instead of the above let us introduce

$$n = 1/\delta x \quad (n\to\infty \text{ as } \delta x\to 0) \qquad \ldots(1.13)$$

then $$e \simeq (1+\frac{1}{n})^n$$

and, in fact $e = \lim_{n\to\infty} (1+\frac{1}{n})^n$ . ...(1.14)

Let us tabulate a few estimations of e using the approximation (1.13). We obtain the results

$$(1+\frac{1}{2})^2 = 2.25$$
$$(1+\frac{1}{5})^5 = 2.489$$
$$(1+\frac{1}{10})^{10} = 2.594$$
$$(1+\frac{1}{20})^{20} = 2.653 \quad .$$

This method does not seem to be very practical since we need to go much further to obtain e to 4 figures; nevertheless someone has calculated

$$(1+ \frac{1}{10000})^{10000} = 2.7182 \quad .$$

Even this is not entirely correct for the figures stated, since the value to 6 figures is

$$\boxed{e = 2.71828} \quad ; \qquad ...(1.15)$$

but, in section 1.3, we will produce our own estimates from approximations for e^x when $x=1$.

1.3 APPROXIMATIONS FOR e^x

1.3.1 The Derivative Approach

We have already established the derivative property of the exponential function $y(x) = e^x$; it is

$$\frac{dy}{dx} = e^x \quad \text{(equation (1.9))}.$$

Let us assume

$$y(x) = e^x = a_0 + a_1x + a_2x^2 + a_3x^3 + a_4x^4 + \ldots, \qquad ...(1.16)$$

a power series expansion with coefficients a_0, a_1, a_2, ... to be determined, then

$$y(0) = e^0 = 1 = a_0 + 0 + 0 + \ldots$$

$$\therefore \quad a_0 = 1 \quad .$$

Differentiating equation (1.16),

$$\frac{dy}{dx} = e^x = a_1 + 2a_2x + 3a_3x^2 + 4a_4x^3 + \ldots ,$$

hence equating coefficients of like powers of x (they must be the same)

$$a_1 = a_0$$
$$\therefore \quad a_1 = 1$$
$$2a_2 = a_1$$
$$\therefore \quad a_2 = 1/2$$
$$3a_3 = a_2$$
$$\therefore \quad a_3 = 1/(2\times3)$$
$$4a_4 = a_3$$
$$\therefore \quad a_4 = 1/(2\times3\times4)$$

and so on. If we define the factorial function as

$$n! = 1\times2\times3\times \ldots \times(n-1)\times n \text{ (the product of the first n integers),} \qquad \ldots(1.17)$$

then we obtain the important result

$$\boxed{e^x = 1 + x + \frac{x^2}{2!} + \frac{x^3}{3!} + \frac{x^4}{4!} + \ldots} \qquad \ldots(1.18)$$

which works for all values of x although far too many terms may be required for large values of x. For this latter situation, invoking properties of exponents (equation (1.7)), we take

$$\boxed{e^x = e^m \left(e^{\frac{x-m}{n}}\right)^n} \qquad \ldots(1.19)$$

where n and m are integers chosen to make $(\frac{x-m}{n})$ sufficiently small. The computation process consists of first calculating $e^{\frac{x-m}{n}}$, then multiplying the result by itself n times, and finally further multiplying by e, m times. Of course, the actual choice for m and n is somewhat arbitrary. As an example

$$e^{3\cdot2} = e^3\left(e^{\frac{0\cdot2}{4}}\right)^4 = (2.71828)^3 \left[1+0.05 + \frac{1}{2}(0.05)^2 + \frac{1}{6}(0.05)^3 + \frac{1}{24}(0.05)^4\right]^4$$

$$= 20.0855(1.051271)^4 = 20.0855(1.22140)$$

$$= 24.5324, \text{ which compares well with the exact result } 24.5325.$$

Using our power series expansion (1.18), we have

$$e = 1 + 1 + \frac{1}{2!} + \frac{1}{3!} + \frac{1}{4!} + \frac{1}{5!} + \frac{1}{6!} + \frac{1}{7!}$$ (+ terms we will ignore)

$$e = 2.7183 \qquad \ldots(1.20)$$

which is a far more practical approach than that adopted in section 1.2.

1.3.2 <u>The Integral Approach</u>

The integral property we require is given by equation (1.10), namely

$$\int_0^x e^x\,dx = e^x - 1\ ;$$

we estimate the integral numerically using Simpson's rule.

(Digression ...

Simpson's rule in its simplest form states that

$$\int_0^{2h} f(x)\,dx \simeq \frac{h}{3}[f(0) + 4f(h) + f(2h)].$$

For example

$$\int_0^{2h} x^2\,dx \simeq \frac{h}{3}[0^2 + 4h^2 + (2h)^2] = \frac{8}{3}h^3$$

which in this instance is exact since

$$\int_0^{2h} x^2\,dx = [\frac{1}{3}x^3]_0^{2h} = \frac{1}{3}(2h)^3 = \frac{8}{3}h^3.)$$

We choose h = x/2; then

$$\int_0^x e^x\,dx \simeq \frac{x}{6}[e^0 + 4e^{\frac{x}{2}} + e^x] = e^x - 1\ .$$

Simplifying the above equation and noting that $e^0=1$, we have

$$x + 4xe^{\frac{x}{2}} + xe^x = 6e^x - 6\ ,$$

and then

$$(6-x)e^x - 4xe^{\frac{x}{2}} - (6+x) = 0\ . \qquad \ldots(1.21)$$

Now if we define $z=e^{\frac{x}{2}}$, equation (1.21) becomes a quadratic equation in z,

$$(6-x)z^2 - 4xz - (6+x) = 0$$

with the positive solution

$$z = e^{\frac{x}{2}} = \frac{1}{2(6-x)} [4x + \sqrt{(4x)^2+4(6-x)(6+x)}\ ]$$

and we obtain an approximation for e^x

$$\boxed{e^x \simeq \{\frac{1}{6-x} [2x + \sqrt{3x^2+36}\ ]\}^2} \quad , \qquad \ldots(1.22)$$

which is valid provided Simpson's rule gives a reasonable estimate of $\int_0^x e^x dx$. It is found that the approximation holds to 5 figures for the interval

$$\boxed{-\frac{1}{2} \leqslant x \leqslant \frac{1}{2}} \qquad \ldots(1.23)$$

and powering similar to the process described in the previous section (equation (1.19)) may be used for numbers outside the interval. (The interested reader should verify that the approximation (1.22) exactly satisfies the exponential property

$$e^{-x} = 1/e^x ,$$

which explains why the interval (1.23) extends into the negative region.)

(Digression ...

If you own a simple electronic calculator that has a square root button, then approximation (1.22) enables you to calculate exponentials with relative ease.)

As examples we calculate

$$e^{3\cdot 7} = e^4\ e^{-0\cdot 3}$$

$$= (2.71828)^4 (0.740817);$$

hence $\quad e^{3\cdot 7} = 40.447$, which is correct to 5 figures,

and $\quad e = (e^{\frac{1}{2}})^2 = (1.64874)^2$

giving $\quad e = 2.7183$. $\qquad \ldots(1.25)$

1.3.3 Other Approaches

Other approaches are used on a digital computer but we will not investigate them except to say that they are usually part of a package of routines made available by the machine manufacturer. For example, on the IBM360 in FORTRAN, we simply code

```
        Y=EXP(X)
```

to return the exponential of X in the storage location Y.

(Digression ...

If you have time you might like to produce a table of e^x for x = -0.5, -0.4, ..., 0, 0.1, 0.2, ..., 0.5 using both the approximation (1.22) and the machine supplied routine to verify that the approximation is valid to 5 figures.)

1.4 THE NATURAL LOGARITHM

The natural logarithm, denoted $\log_e y$ or $\ell n\ y$, inverts the relationship given by the exponential

$$y = e^x , \qquad \ldots(1.26)$$

to give $\ell n\ y = x$, ...(1.27)

just as the ordinary logarithm, denoted log y, inverts the relationship given by the antilog expression

$$y = 10^z , \qquad \ldots(1.28)$$

to give $\log y = z$ . ...(1.29)

Conversion between the two types of logarithms is easy. First we note that, from equations (1.28) and (1.29),

$$y = 10^{\log y} \quad (\text{and also } y = e^{\ell n\ y}) ;$$

hence, taking natural logs, we obtain

$$\ell n\ y = (\log y)\ (\ell n\ 10) = 2.30259 \log y$$

giving the result

$$\boxed{\log y = \ell n\ y/2.30259} \ . \qquad \ldots(1.30)$$

As for the exponential function, there are many possible approaches for obtaining approximations for the natural logarithm. An approximation that is inverse to our approximation (1.22) is obtained from equation (1.21). By solving for x rather than $e^{x/2}$, we obtain

$$\boxed{\ell n\ y \simeq \frac{6(y-1)}{y+4\sqrt{y}+1}} \qquad \ldots(1.31)$$

which is accurate to 5 figures for the interval

$$\boxed{(0.60653=)\ e^{-\frac{1}{2}} \leqslant y \leqslant e^{\frac{1}{2}}\ (=1.6487)}\ . \qquad \ldots(1.32)$$

As an example of a possible method of attack when the above interval is exceeded, we have

$$\ell n(286.5)=\ell n(100\times2.865)=2\ell n(10)+4\ell n(2.865)^{\frac{1}{4}}=2(2.30259)+4\ell n(1.30101)$$

(using successive square roots)

$$= 4.60518 + 4(0.263140)$$

hence $\ell n(286.5)=5.6577$, which is correct to 5 figures.

Alternatively

$\ell n(286.5)=16\ell n(286.5)^{1/16}=16\ell n(1.424198)$ (using successive square roots) $= 5.6577$.

We now turn to an important property of ℓn y, namely

$$\boxed{\int\frac{dy}{y} = \ell n\ y}\ . \qquad \ldots(1.33)$$

We readily verify this result from equation (1.9) since

$$\int\frac{dy}{y} = \int dx = x = \ell n\ y\ .$$

1.5 THAT STRANGE BUT BEAUTIFUL RELATIONSHIP

This section is only of interest to the advanced reader. For the few that are determined to continue, you might recall from section 1.1 that the Euler relationship (1.6) was exposed for admiration

$$e^{\pi\sqrt{-1}} = -1\ . \qquad \ldots(1.34)$$

Written alternatively as

$$(e^{\frac{\pi}{8}\sqrt{-1}})^{8} = -1\ , \qquad \ldots(1.35)$$

the question arises as to whether we could take $x = \frac{\pi}{8}\sqrt{-1}$ in our approximation (1.22) and obtain the result (1.34). (We have at least scaled the x to perhaps an appropriate size!) Well if we take the bold step we obtain the answer

$$(e^{\frac{\pi}{8}\sqrt{-1}})^{8} = -1.00000 - 0.00002\sqrt{-1}\ . \qquad \ldots(1.36)$$

This somewhat surprising result that our approximation (1.22) holds for imaginary arguments (that is x = something times $\sqrt{-1}$) really means, to those that know the deeper Euler relationship

$$e^{x\sqrt{-1}} = \cos x + \sqrt{-1}\sin x \quad , \qquad \ldots(1.37)$$

that we could derive approximations for cos x and sin x from our exponential approximation (1.22). We leave that as an exercise for the advanced reader.

CHAPTER 2

MATHEMATICS OF A LEAST SQUARES PROCESS

Lecture by

B.E. CLANCY

ABSTRACT

The problem of determining the equation of a curve which best describes a set of experimental measurements is discussed and a technique for fixing the equation when the curve is a straight line is described.

CONTENTS

2.1 LINEAR RELATIONSHIP BETWEEN EXPERIMENTAL VARIABLES

When analysing the results of scientific experiments in the light of a theory, a problem which constantly arises is that of reconciling theory and experiment. During this Summer School a fair part of your time will be involved with just this problem.

A common situation is where the theory says that two measurable quantities are connected by a linear relationship. If we call the two quantities x and t, the theory then says

$$x = mt + b \qquad \ldots(2.1)$$

where m and b are fixed constants.

For example, suppose that x is the length in metres of a metal bar and t is the temperature of the bar in degrees Celsius. The usual theories of heat and of the properties of metals assert that equation (2.1) holds when b is the length of the bar at 0°C and m is a coefficient of expansion for the metal. We can construct many other theories for which equation (2.1) is their mathematical statement.

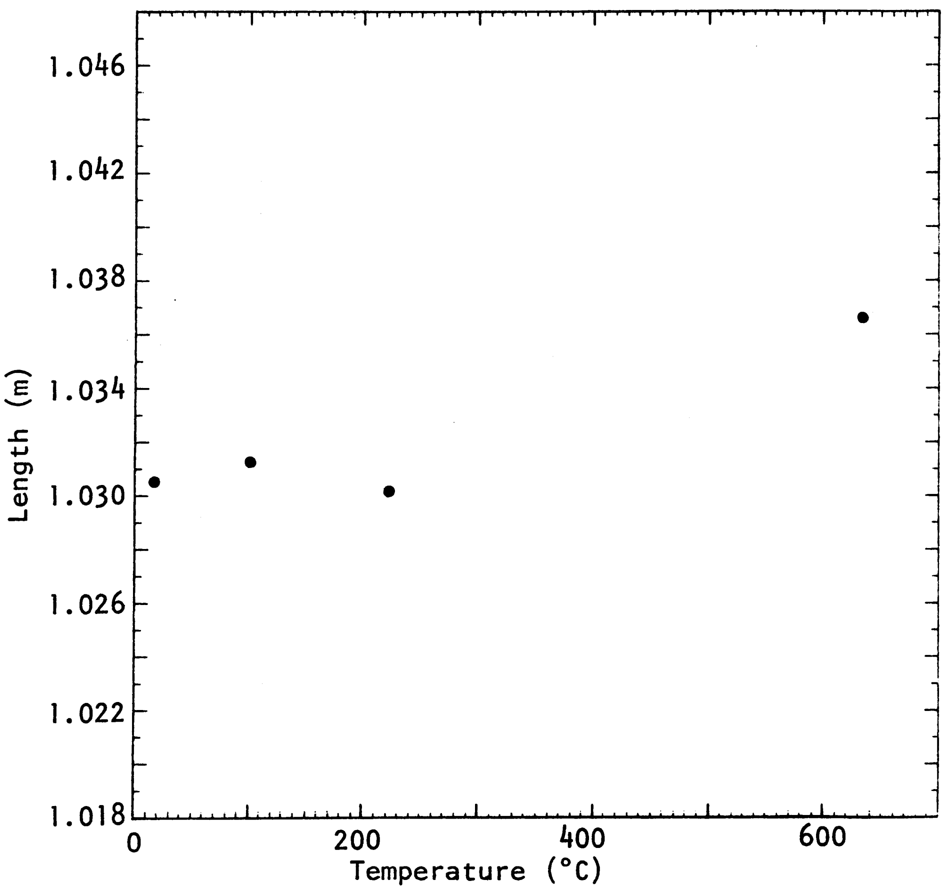


Figure 2.1

Let us suppose that an experiment is carried out to determine the values of the constants m and b in equation (2.1). In our example, the experiment might consist of heating or cooling the bar by some means, waiting until all points on the bar were at the same temperature and then simultaneously measuring that temperature and the length of the bar. This would give a pair of values, say t_1, x_1, for the two variables t and x. After repeating the experiment a number of times for different temperatures, and recording the results, we would have a set of pairs t_1, x_1 t_2, x_2 t_3, x_3 ... *etc.*

The theory asserts that if these pairs are used as Cartesian coordinates of points, then the points will all lie on a single straight line. This will be the situation if the theory is correct and if our experimental technique is perfect. Unfortunately, we aren't perfect beings and the result of the experiment is likely to be a set of points like those in figure 2.1.

Instinctively we would think, after inspecting these results, that

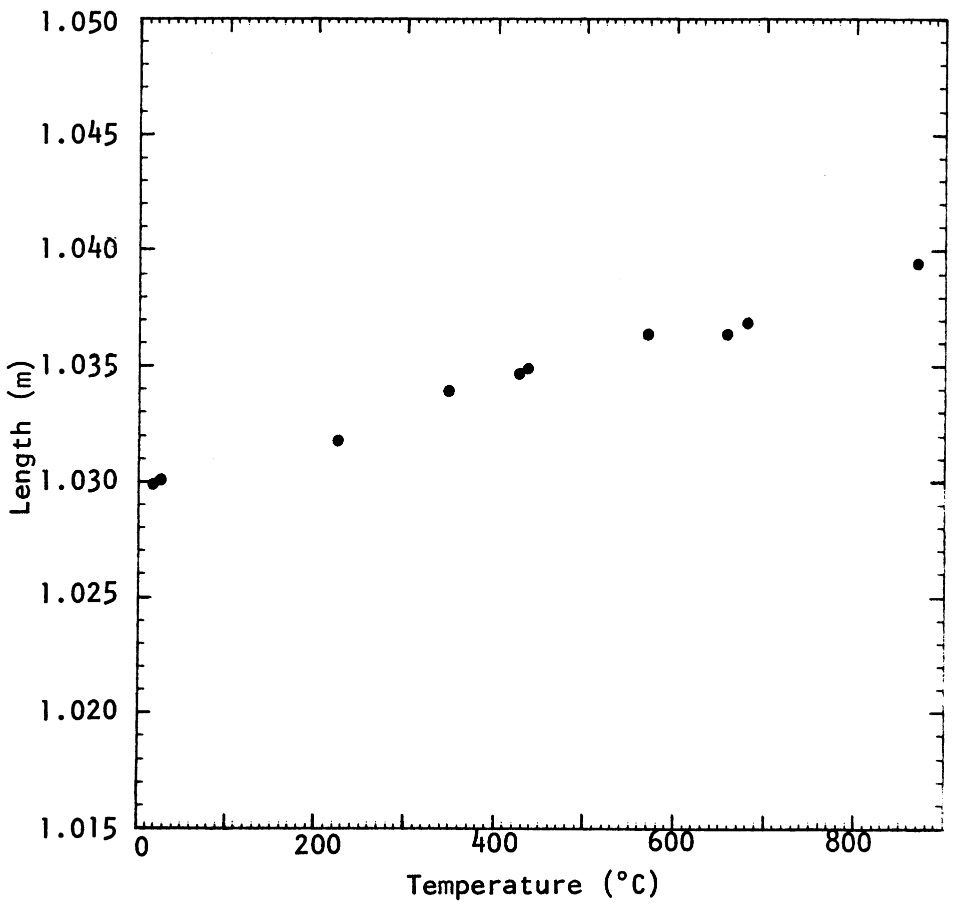


Figure 2.2

the theory was probably all right and that had our experimental technique been far better, then the points would all lie exactly on a single line. Unfortunately, what happens if the experiment is repeated with more care and with more complex and expensive equipment is that we achieve a set of points similar to those of figure 2.2.

We see that the situation is much improved - there does seem to be a line $x = mt + b$ which nearly passes through the points. However, we can't draw the line by simply joining the points.

2.2 THE LINE OF BEST FIT

Faced with the situation in figure 2.2, we have to find some way of constructing a line which best seems to fit the set of points. We can do this by constructing what are sometimes called 'eyeball fits', *i.e.* we simply put a rule across the points and shift it around until we are more or less satisfied that the line is good enough. Unfortunately, no two people's eyeball fits ever agree and some more objective procedure is desirable. Such a procedure we now describe.

2.3 LINEAR LEAST SQUARES FITTING

We suppose that the line of best fit we are seeking has the equation

$$x = mt + b,$$

and calculate an error e_i for each of our N experimental points by the rule

$$e_i = mt_i + b - x_i \quad .$$

A convenient measure of the total inaccuracy in the fit is the total squared error E, where

$$E = \sum_{i=1}^{N} e_i^2$$

$$E = \sum_{i=1}^{N} (mt_i + b - x_i)^2 \quad . \qquad \ldots(2.2)$$

Squaring the individual errors removes the distinction between positive and negative errors, and also has the effect of enhancing the contribution of large individual errors while reducing the contribution of small ones.

The least squares fitting procedure asserts that the line of best fit is that which makes the squared error E of equation (2.2) a minimum;

our remaining problem is to establish a procedure for finding the values of m and b which do this.

If the coefficient b is held fixed, then E varies with m and the calculus tells us that E will be stationary if the derivation $\frac{dE}{dm}$ is zero. We find also that this stationary point is a true minimum. If, on the other hand, m is held fixed, then E varies with b and we can show that the corresponding stationary point where $\frac{dE}{db}$ is zero is also a minimum. In equation (2.2) we have a situation where both m and b can vary independently, but we can show that the minimum value of E occurs when both derivatives are simultaneously zero. The derivatives, denoted in this situation by $\frac{\partial E}{\partial m}$ and $\frac{\partial E}{\partial b}$, are given by

$$\frac{\partial E}{\partial m} = \sum_{i=1}^{N} [2t_i(mt_i + b - x_i)]$$

$$= 2m\,\Sigma(t_i^2) + 2b\,\Sigma(t_i) - 2\,\Sigma(t_i\,x_i),$$

$$\frac{\partial E}{\partial b} = \sum_{i=1}^{N} [2(mt_i + b - x_i)]$$

$$= 2m\,\Sigma(t_i) + 2bN - 2\,\Sigma(x_i) \quad .$$

If we equate these to zero, we get a pair of simultaneous equations for m and b which have the solution

$$m = \frac{\{N\ (t_i x_i) - \Sigma(t_i)\Sigma(x_i)\}}{\Delta}$$

$$b = \frac{\{\Sigma(t_i^2)\Sigma(x_i) - \Sigma(t_i)\Sigma(t_i x_i)\}}{\Delta} \qquad \ldots(2.3)$$

where $\Delta = N\Sigma(t_i^2) - [\Sigma(t_i)]^2$

2.4 A SIMPLE EXAMPLE

Let us see what this procedure gives for a simple example. Suppose we have four pairs of values (t_i, x_i) as follows:

(0.0, 1.2) (1.0, 1.9) (2.0, 3.1) (3.0, 3.8) .

These points are plotted in figure 2.3 and you should first try for an eyeball fit to the points.

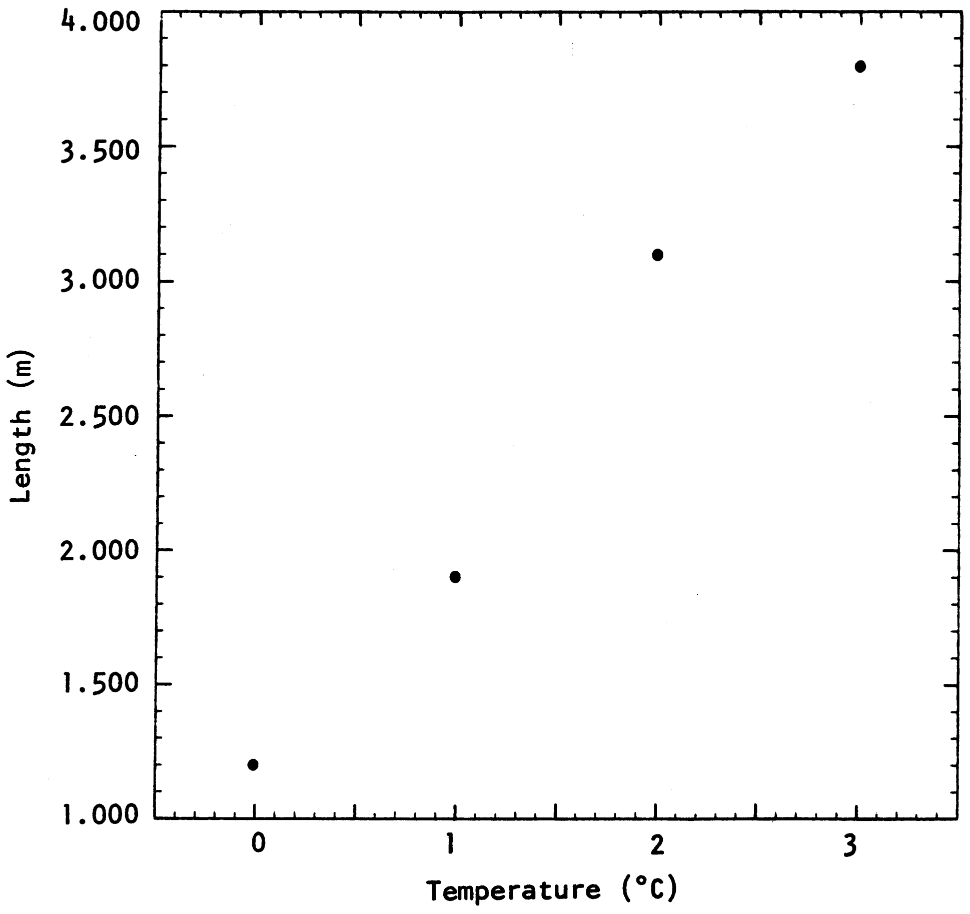


Figure 2.3

To carry out the least squares fit in this case, we can form the table below:

(t_i)	(x_i)	(t_i^2)	(t_ix_i)	N=4
0.0	1.2	0.0	0.0	
1.0	1.9	1.0	1.9	
2.0	3.1	4.0	6.2	
3.0	3.8	9.0	11.4	

We then proceed as follows;

$$\Sigma t_i = 6.0 \quad \Sigma x_i = 10.0 \quad \Sigma t_i^2 = 14.0 \quad \Sigma t_i x_i = 19.5$$

$$\Delta = N\Sigma(t_i^2) - [\Sigma(t_i)]^2$$

$$= 4(14.0) - (6.0)^2 = 20.0$$

$$m = \frac{N\Sigma(t_ix_i) - \Sigma(t_i)\Sigma(x_i)}{\Delta}$$

$$= \frac{4(19.5) - (6.0)(10.0)}{20.0} = 0.9$$

$$b = \frac{\Sigma(t_i^2)\Sigma(x_i) - \Sigma(t_i)\Sigma(t_i x_i)}{\Delta}$$

$$= \frac{(14.0)(10.0) - (6.0)(19.5)}{20.0} = 1.15$$

The line of best fit from a least squares analysis is thus

$$x = (0.9)t + 1.15$$

If you construct this line on figure 2.3 you will see how well (or how poorly) it agrees with your eyeball fit.

The arithmetic necessary for fitting a line to these four points is not too complex. In a real experiment you may expect to have scores of pairs of values and the arithmetic would be far more tedious. Fortunately, digital computers exist to relieve us from the tedium.

2.5 NON-LINEAR RELATIONSHIPS

In the real world, measurable quantities are not always connected by linear relationships. During this Summer School, much of your time

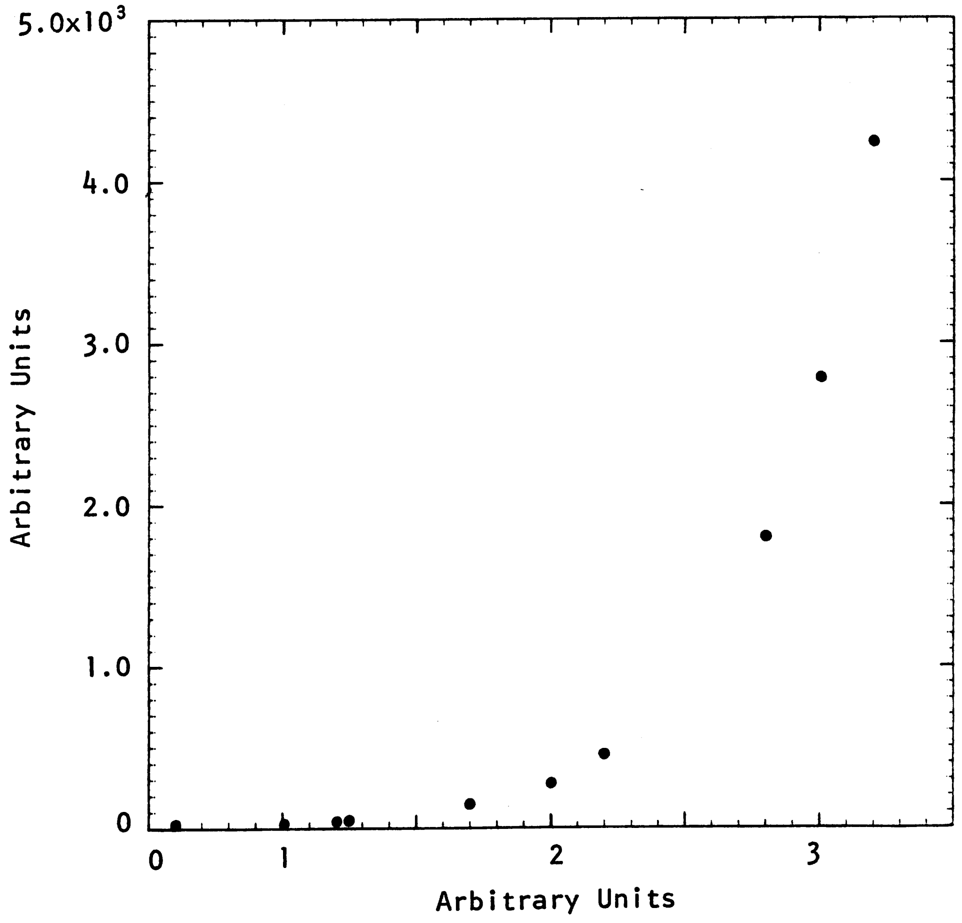


Figure 2.4

will be involved with quantities which are connected by a particular non-linear function - the exponential function. The appropriate theories will then assert that the t and x variables are connected by the equation

$$x = ce^{mt} , \qquad \ldots(2.4)$$

with c and m being constants. We can see that an experiment to determine values of the coefficients m and c could consist of collecting pairs of values (t_i, x_i) and hoping that they lie on a curve whose equation has the form of equation (2.4).

Let us imagine that we have done such an experiment and plotted the points to get the situation in figure 2.4.

It is not easy to see, from this plot, whether or not the points lie on or even close to a curve with the required shape. This problem is easily overcome by changing our method of analysis. We take the natural logarithm of each x value, call it z say, and plot the points whose coordinates are (t_i, z_i). Instead of the theory simply being given by equation (2.4), we must now couple it with the equation

$$z = \ell n\ (x)$$

so that

$$z = \ell n\ (ce^{mt})$$

$$= \quad mt + \ell n\ (c) \qquad \ldots(2.5)$$

which is a linear relationship.

I have done *essentially* this with the data plotted in figure 2.4 and from it produced figure 2.5. In this figure we see that the points tend to lie on a straight line.

Figure 2.5 was actually drawn using a computer-generated logarithmic scale on the vertical axis, where the distance between two points is proportional to the difference of their logarithms. Graph paper ruled in this fashion is called semi-logarithmic paper and its use saves the trouble of looking up the logarithms of the individual numbers to be plotted. It can thus be used when attempting an 'eyeball fit' to pairs of numbers which are believed to be related to one another by equation (2.4).

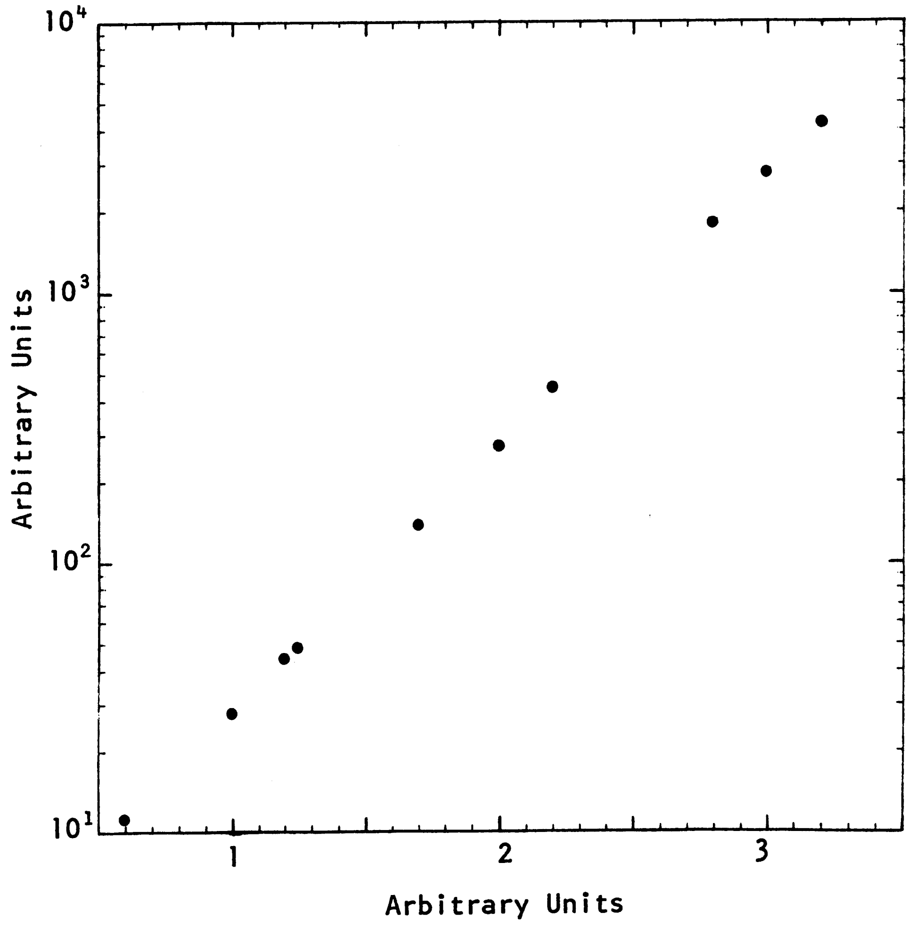


Figure 2.5

2.6 <u>LEAST SQUARES FITTING TO EQUATION (2.4)</u>

When analysing an experiment for which the theory is expressed by equation (2.4), the disadvantages of using eyeball fits to the data are the same as those involving a linear relationship like equation (2.1). It is more satisfactory to use a least squares technique to construct a curve of best fit. The simplest technique, and the one which you will be using during the Summer School, is to use the logarithms of the experimental y values and change the theory so that it is expressed by equation (2.5). The technique already developed for finding a line of best fit then applies.

The results of our experiment are again the set of N pairs of numbers (t_1, x_1), (t_2, x_2), ... For each number x_i, we compute the natural logarithm

$$z_i = \ln (x_i)$$

to get the set of pairs (t_1, z_1), (t_2, z_2) ... Exactly as in section 2.3, we then form the sums:

$$\sum_{i=1}^{N}(t_i) \quad \sum_{i=1}^{N}(t_i^2) \quad \sum_{i=1}^{N}(z_i) \quad \sum_{i=1}^{N}(t_i\, z_i) \quad ,$$

and compute the slope m and intercept b of the line of best fit

$$z = mt + b \qquad \ldots(2.6)$$

from the formula

$$\left.\begin{aligned}\Delta &= N\Sigma(t_i^2) - [\Sigma(t_i)]^2 \\ m &= \frac{\{N\Sigma(t_iz_i) - \Sigma(t_i)\Sigma(z_i)\}}{\Delta} \\ b &= \frac{\{\Sigma(t_i^2)\Sigma(z_i) - \Sigma(t_i)\Sigma(t_iz_i)\}}{\Delta}\end{aligned}\right\} \qquad \ldots(2.7)$$

Since z stands for ℓn (x), equation (2.6) is simply

$$\ell n\ (x) = mt + b$$

and this becomes, after taking anti-logarithms,

$$x = e^{mt+b}$$

or $x = c\ e^{mt}$, the theoretical curve

with $c = e^{b}$ . ...(2.8)

The coefficient m for the curve of best fit remains just as it is found in equation (2.7), and the combination of equations (2.7) and (2.8) lets us compute the remaining coefficient c.

CHAPTER 3

F O R T R A N

Lecture by

J.M. BARRY

ABSTRACT

An introductory course in FORTRAN programming for Summer School students designed to demonstrate the mathematical potential of the language, while introducing sufficient basic I/O skills to write programs.

CONTENTS

3.1 INTRODUCTION

Each digital computer is capable of obeying a number of basic instructions. These instructions vary for different computers but they have many attributes in common:

(i) The ability to perform the four arithmetic operations (+, -, x, ÷).

(ii) The ability to perform logical operations (is $A \geqslant B$).

(iii) The ability to perform 'housekeeping' instructions (*e.g.* moving numbers from core store to registers where arithmetic and logical operations may be performed on them).

For a programmer to communicate with the computer at this most fundamental level, it would be necessary for him to develop programs in the basic machine language of the computer at his disposal. In the early days of computing, it was necessary for scientists and mathematicians to concern themselves with the intricacies of binary coding. The long delays and inconvenience of this form of man-machine communication accelerated the growth of programming languages that could be used more readily by the problem solver. Many languages (FORTRAN, ALGOL, PLI, APL, ACL *etc.*) have been developed for scientific, commercial and other applications. FORTRAN is chosen as the vehicle for problem solving at this Summer School owing to its world-wide acceptability as a scientifically oriented programming language. There are no computers that obey programs written in FORTRAN directly. It is necessary for 'high level' programs in languages such as FORTRAN to be translated into an appropriate set of machine language instructions. This process is known as *compilation*.

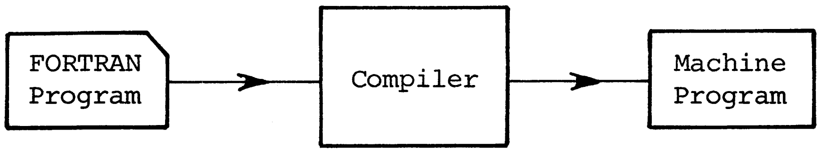


Figure 3.1 - Compilation of FORTRAN program

The FORTRAN source statements are translated to a set of machine language instructions by a FORTRAN compiler (figure 3.1). The compiler is itself a program (usually supplied by the machine manufacturer) that first checks to ensure the FORTRAN statements obey the 'rules' of the language (syntax analysis), and then supplies a set of machine instructions that will implement what the programmer has specified. When the compilation process is completed, the machine instructions generated may be *executed*. The finer details of this process and the way it is implemented on the IBM360 will not be our concern at this Summer School as we are primarily interested in using the computer as a tool for mathematical problem solving.

3.2 OVERVIEW OF FORTRAN PROGRAMMING

Let us first consider the steps involved in solving a sample problem, and the FORTRAN program that could be developed to carry them out. When this is done we shall examine the various FORTRAN statements in closer detail.

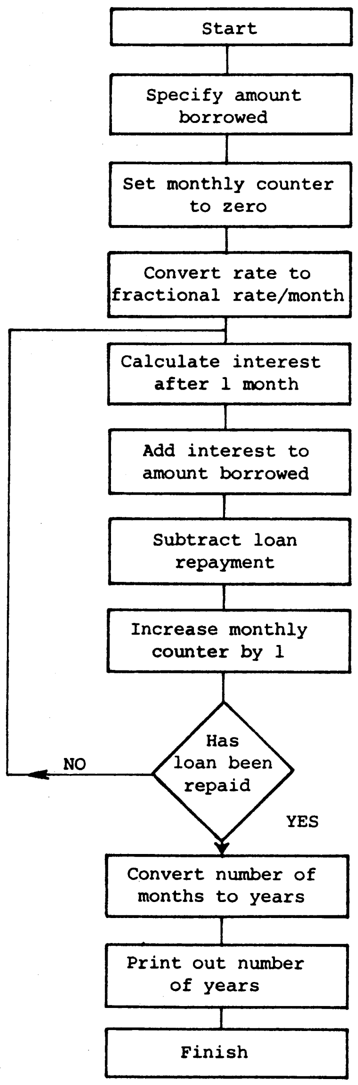

Start
Specify amount borrowed
Set monthly counter to zero
Convert rate to fractional rate/month
Calculate interest after 1 month
Add interest to amount borrowed
Subtract loan repayment
Increase monthly counter by 1
Has loan been repaid
NO
YES
Convert number of months to years
Print out number of years
Finish

Figure 3.2 - Flow Chart for Compound Interest Problem

Problem If $18,000 is borrowed at a rate of 7% (monthly reducible) and repayments of $200 each month are made, then 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, in much the same way 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 helpful to draw a flowchart (figure 3.2) showing the steps involved, while others prefer to visualise all the steps in their mind.

From this flow chart the following program can be coded. At this point we will not concern outselves with the formal rules for coding but just look at the end product (figure 3.3).

```
1   5 6 7                                                          7273   80
C        PRØGRAM BY J.M. BARRY TØ DETERMINE THE NUMBER ØF
C        YEARS NECESSARY TØ REPAY A LØAN.
C        THE PRINCIPAL BØRRØWED, INTEREST RATE AND MØNTHLY REPAYMENT
C        ARE TØ BE READ FRØM A PUNCHED DATA CARD.
         READ (1,100)PRINC,RATE,PAYMNT
         MNCNTR=0
         FRATEM=RATE/(100•*12•)
      1  ADPRIN=PRINC*FRATEM
         PRINC=PRINC+ADPRIN
         PRINC=PRINC-PAYMNT
         MNCNTR=MNCNTR+1
         IF(PRIN•GT•0•) GØ TØ 1
         YEARS=MNCNTR/12•
         WRITE(3,101)YEARS
    100  FØRMAT(3F10•3)
    101  FØRMAT(' NØ ØF YEARS = ',F10•3)
         STØP
         END
```

The data card necessary for this problem would be

```
          10|       20|       30|
   18000.        7.       200.
```

Figure 3.3 - Sample program for compound interest problem

3.3 PUNCHING OF CARDS

To assist in the punching of cards, programmers usually use a standard coding sheet representing the 80 columns available on a punched card. Each line of the sheet represents a new card which may contain only one statement.

```
1   5 6 7                                                   7273    80
C        J. SMITH STATEMENT EXAMPLE JAN 1976                THIS
C        THE ABØVE IS A CØMMENT                             SECTION
         X=A+B                                              NOT
    50   Y=9•-C                                             USED
         P R INC = RATE * PRINC/100• + PRINC                IN
             SUM = A+B+C+D+E+F+G+H+                         FORTRAN
      1    Ø+P+Q+R                                          PROGRAMS
```

Statements can be punched from columns 7 to 72. To assist the programmer to recall aspects of a program, a comment card (denoted by a C in column 1) may be placed anywhere within the punched deck. These are ignored by the FORTRAN compiler. Normally we shall commence our programs with a comment card to assist with the identification of the program.

Columns 1 to 5 inclusive can be used if desired to assign a statement label in the form of a number in the range 1 to 99999 (there is no need to choose labels in ascending order).

Blanks may be inserted within a statement to make it more readable, and may be considered as being removed in the compilation process. Should a statement be too long to fit on one card, it is continued from column 7 of a subsequent card provided column 6 of this card contains a continuation character (any character other than a blank or zero will suffice as a continuation character).

The character set available within the FORTRAN system consists of

(i) 26 capital letters A,B,C,...Z ;

(ii) 10 numerals 0,1,2,...,9;

(iii) 10 special characters +,-,*(multiplication),/(division) ,•,÷,(,),=,4; and

(iv) a blank (usually written ƀ if its presence is to be emphasised for punching).

3.4 ARITHMETIC CONSTANTS

We will treat three different types of constants sufficient for handling data (numbers) in most scientific problems.

3 (i) *INTEGER* (or fixed point) constants

- a whole number without a decimal point whose absolute value is $\leqslant 2^{31} - 1$ (2147483647).

Valid integer constants 0 -5 +357 7005192

Invalid integer constants 27• 5,132 9812735997

(ii) *REAL* (or floating point) single precision constants.

- up to 7 decimal digits with a decimal point, with or without an exponent. The absolute magnitude is approximately 10^{-78} to 10^{75}.

Valid real single precision constants

+0. 7.91 5.3E+2(=$5.3\text{x}10^2$)

5.3E2($5.3\text{x}10^2$) -.051E-03 ($-.051\text{x}10^{-3}$)

Invalid real single precision constants 1 3,471.2 1.E

(iii) *REAL* (or floating point) double precision constants.

- similar to (ii) but up to 16 digits are possible with a D exponent being necessary instead of E. Double precision constants will not be necessary for the Summer School problems.

The reason for the careful distinction drawn between the three types of constants is electronic rather than mathematical. The electronic 'hardware' necessary for *INTEGER* arithmetic operations is less sophisticated and consequently faster than that used for *REAL* arithmetic. By performing those operations that require no decimal point in integer mode, considerable time savings can be made.

3.5 VARIABLES

A variable is a symbolic name used to identify a data item that will occupy a location (one word) of core storage. The actual address of this location is assigned by the compilation process. If we move a number into a variable it will replace the previous contents of that location.

```
TIME=0•
```

This places zero in the location reserved for TIME. When a transfer is made from a location, the previous contents remain unaltered.

```
X=TIME
```

This assigns the contents of the location reserved for TIME to that reserved for X without altering the contents of the location associated with TIME.

The '=' operation should be interpreted as the assignment of the result of the right hand expression to the left hand location. Consequently, an expression such as

```
        A=A+1.
```

does not yield any algebraic result but rather is interpreted as increasing the old value associated with A by 1. to give a new result also called A.

Variable names may have up to 6 characters (special characters are not permitted) the first of which must be alphabetic such as

```
        TIME , X3B , I5 , T .
```

Variables like constants take an *INTEGER* or *REAL* form. Unless the programmer provides specifications to the contrary, all variables commencing with I,J,K,L,M or N are *INTEGER* variables, while the remainder are single precision *REAL* variables.

Variables may also be subscripted in FORTRAN. Such variables may be used to represent vectors or matrices which you probably have encountered in your mathematics courses.

V(3) is the FORTRAN representation of the vector component v_3 ,

A(3,4) is the FORTRAN representation of the matrix element a_{34}.

(Further consideration of SUBSCRIPTED variables will be delayed until section 3.12.)

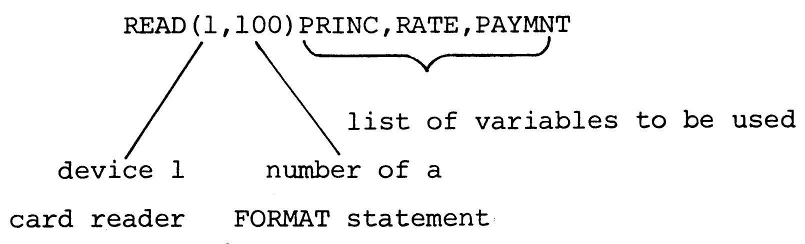
3.6 INPUT AND OUTPUT

One way of assigning values to variables is through the direct use of an arithmetic expression:

```
        X=6.3
```

Should one wish to alter the data on which the program is to operate without changing the program itself, then a READ statement is needed.

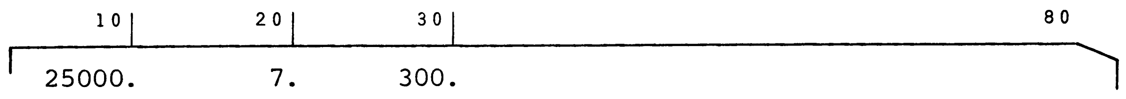
The read statements initiate the reading of data cards (these are separate from the program cards) and the transfer of the numbers on these cards to the variables in the READ lists.



Numbers are read from device 1 under the control of an editing (FORMAT) statement. The supplied FORMAT statement (100) will describe the way the punched data card is laid out. In this case

```
100 FØRMAT(3F10•3)
```

indicates that 3 numbers are punched on the data card satisfying the format code F10.3.



F10.3 signifies *REAL* constants without any exponent (F), 10 columns being kept for each number. Should there be no decimal point punched on the card, one will automatically be assumed to exist 3 digits to the left of column 10. (When the decimal point is punched, the second parameter is ignored.)

The output statement WRITE, functions in a similar manner.

```
     WRITE(3,101)YEARS
 101 FØRMAT(' NØ ØF YEARS = ',F10•3)
```

would display on the printer (device 3) output of the form

```
NØ ØF YEARS = 26•314
```

3.7 ARITHMETIC OPERATIONS AND EXPRESSIONS

Five arithmetic operations are available to FORTRAN users:

(i)	addition	+	*e.g.* A+B
(ii)	subtraction	-	*e.g.* A-B
(iii)	multiplication	*	*e.g.* A*B
(iv)	division	/	*e.g.* A/B
(v)	exponentiation	**	*e.g.* A**3 (A^3)

Expressions may be enclosed within parentheses as in normal algebra.

$(a+b)\ (c+d)$	(A+B)*(C+D)
$(a+b)^2$	(A+B)**2
$\frac{a}{bc}$	A/(B*C)

Parentheses are necessary to prevent two operations from appearing next to each other (should such a combination be possible)

```
X*-Y  must be coded X*(-Y)
```

The sequence of operations in expressions is determined from the following hierarchy and is consistent with normal mathematics.

(i) **

(ii) */ left to right precedence

(iii) +- left to right precedence.

Consequently, the expression

```
X+(Y/A)-(3.*U)+P*(S**4)/3.
```

could have been correctly abbreviated to

```
X+Y/A-3.*U+P*S**4/3.
```

The integer variables or constants deserve special mention. Division of one integer by another results in the truncation of any fractional remainder.

```
I=9
K=I/2
```

would result in K taking the value 4. This property can often be exploited to the programmer's advantage in the testing for even integers;

```
K=I-I/2*2
```

would assign 1 to K if I is odd, and 0 if I is even.

Expressions should consist of variables or constants all in the same mode *i.e.* all *REAL* or all *INTEGER*. There is one exception to this rule in that the exponent of a *REAL* variable or constant may be *INTEGER*. The following are permitted forms of exponentiation:

```
V**2          V**A
(-V)**.45     V**(-I)
V**(-2)       I**3
```

The mode of a variable on the left hand side of an arithmetic assignment need not be the same as that of the expression on the right.

```
A=I+1
```

The compiler will arrange for the right hand side to be evaluated in

INTEGER mode and the result to be converted to the *REAL* mode before it is stored away. Because of truncation in *INTEGER* division, great care should be exercised in using this type of arithmetic.

3.8 SUPPLIED MATHEMATICAL FUNCTIONS

As there are a number of special mathematical functions or operations that are common to many problems, the FORTRAN compiler provides these as part of the normal system. To calculate the exponential function $x=e^t$, all we need do is code

```
X=EXP(T)
```

To use a supplied mathematical function, it is only necessary to follow the function name by an argument enclosed in parentheses. The result will be returned as though the function name itself designated a variable in the program. The argument may be a variable, constant or arithmetic expression

```
A=EXP(A-C)+SQRT(15.)
```

A list of frequently required functions follows:

Mathematical Function	Function Name (Argument)
square root,	SQRT(X)
exponential, e^x	EXP(X)
natural logarithm, log x (or ℓn x)	ALOG(X)
sine of an angle (in radians), sin x	SIN(X)
cosine of an angle (in radians), cos x	COS(X)
tangent of an angle (in radians), tan x	TAN(X)
arctangent (result in radians), $\tan^{-1}x$	ATAN(X)
absolute value (real numbers), $\|x\|$	ABS(X)

Functions other than those supplied through the compiler are often necessary, so FORTRAN allows a programmer to name and define his own special functions (section 3.13).

3.9 TRANSFER OF CONTROL

Execution of a program will commence at the first executable statement and proceed through subsequent instructions in order, unless a transfer of control statement is encountered. The simplest means of transfer of control is through an 'unconditional GØ TØ' statement.

```
56 READ(1,9)X
   WRITE(3,11)X
   GØ TØ 56
```

This section of program would cause cards to be read and printed with no escape mechanism until the supply of punched data cards was exhausted in which case an error condition would cause the program to fail. Clearly such a statement alone would be of limited use.

There is an extension of this statement, known as the 'computed GØ TØ', which gives a little more choice in the statement to which the branch is to be made.

```
GØ TØ (71,56,1,9),I
```

If I=1 control passes to statement 71
If I=2 control passes to statement 56
If I=3 control passes to statement 1
If I=4 control passes to statement 9

For any other value of I, control would pass to the next sequential statement in the program.

The most useful form of the transfer of control statement is the 'logical IF' as demonstrated in our first sample program.

```
IF(PRINC•GT•0•)GØ TØ 1
```

If PRINC is greater than zero, then control will pass to the statement labelled 1. The logical IF statement can be considered to be of the form

```
IF (logical expression) executable statement
```

The logical expression can take one of two values only, .TRUE. or .FALSE. In a logical IF, the statement appended will be executed only if the logical expression returns a .TRUE. result. When it is .FALSE., the appended statement is ignored and control will pass to the next statement.

```
   IF(A•LT•B)GØ TØ 56
   WRITE(3,11)B
56 A=B*C
```

If $A < B$, then A will be recalculated as the product of B and C. For $A \geqslant B$ the value of B will be printed first.

While the appended statement is frequently a 'GØ TØ' statement, it may be any executable statement other than another 'logical IF' or a 'DØ' statement (section 3.10).

```
        IF(A•LT•0•)A=-A
```

This would be sufficient to replace A with its absolute value although the coding would be somewhat slower than using the alternative statement

```
        A=ABS(A)
```

Logical expressions are most frequently formed by two arithmetic expressions and a relational operator.

A•EQ•B	a is equal to	b	$a = b$
A•NE•B	a is not equal to	b	$a \neq b$
A•GT•B	a is greater than	b	$a > b$
A•GE•B	a is greater than or equal to	b	$a \geqslant b$
A•LT•B	a is less than	b	$a < b$
A•LE•B	a is less than or equal to	b	$a \leqslant b$

e.g.

```
        IF(A+B•LE•C+SQRT(X**2+Y**2))A=1.
```

Frequently we wish to carry out more than one logical test at a time. This can be done by combining logical expressions with one of the following logical operators:

•AND•	both expressions must be .TRUE. to return a .TRUE. result
•ØR•	result a .TRUE. if either expression is .TRUE.

```
        READ(1,100)A,B,C
   100  FØRMAT(3F10•3)
        IF(A+B•GE•C•AND•A+C•GE•B•AND•B+C•GE•A)WRITE(3,157)A,B,C
   157  FØRMAT('A,B, AND C ARE PØSSIBLE SIDES OF A TRIANGLE',3F10•3)
        STØP
        END
```

The above program will read three values for A,B, and C respectively from a punched data card (not shown here) and will test whether the values A,B and C are capable of being the lengths of the sides of a triangle. As before, if the combined logical expression is .FALSE. then control will pass to the next statement.

3.10 LOOPS

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 code would be sufficient

```
      .
      .
      .
      ISUM=0
      I=1
    5 ISUM=ISUM+I
      I=I+1
      IF(I•LE•20)GØ TØ 5
      .
      .
      .
```

In this example, ISUM is chosen as a variable name to accumulate the sum of the integers (integer variables start with I,J,K,L,M or N). It is first necessary to initialise this to zero and the counter (I) to 1. Two statements are then necessary to increase the counter and to test it to determine whether the loop is complete, and transfer control back if it is not. Because scientific programming is often repetitive in this way, FORTRAN supplies a 'DO' statement to allow operations such as the above to be quickly coded as

```
      ISUM=0
      DØ  5 I=1,20
    5 ISUM=ISUM+I
```

The 'DO' statement specifies the last statement in the series of statements to be repeated (5), an *INTEGER* variable to act as the counter (I), and two *INTEGER* constants or variables to act as the initial and final values for which the loop is executed.

```
      DØ  2 J=N,M
```

will cause all statements between itself and including one with a label 2 to be repeated (M-N+1) times. It is necessary for N and M to have previously been assigned values, either as the left hand side of an arithmetic assignment, or through a READ command. FORTRAN requires that $N \geqslant 1$, while $M \geqslant N$. When a 'DO' loop is completed, the 'DO' variable (J in the above example) is regarded as being undefined.

It is at times necessary to nest one 'DO' loop inside another. Suppose we have 100 punched data cards with one number on each card, and that our aim is to find the average of each group of 10 and print that average out. The following is a complete program capable of doing this.

```
C       PRØGRAM BY J. SMITH
C       TØ READ 100 NUMBERS AND
C       FIND AND PRINT THE AVERAGE ØF EACH GRØUP ØF 10
        DØ  1 I=1,10
        SUM=0•
        DØ  2 J=1,10
        READ(1,15)X
     2  SUM=SUM+X
        AVG=SUM/10•
     1  WRITE(3,16)AVG
    15  FØRMAT(F10•3)
    16  FØRMAT(' AVERAGE FØR GROUP ØF 10 = ',F10•3)
        STØP
        END
```

```
      5.32  )
      8.61  }   data cards
       .    )
       .
       .
```

The loops function so that the inner counter will vary the most rapidly, *i.e.*

```
I 1 1 1 ...  1 2 2 2 ... 2 ... 10 10
J 1 2 3 ... 10 1 2 3 ... 10 ... 9 10
```

The last statement in a 'DO' loop can be any executable statement other than a transfer of control. A dummy statement CONTINUE, which does not perform any machine function, is provided as a way around this restriction.

```
    DØ 27 I=1,N
    .
    .
    .
    IF(X•GT•27•35)GØ TØ 95
 27 CØNTINUE
    .
    .
    .
 95 X=X+7•
    .
    .
    .
```

3.11 STOP AND END STATEMENTS

The STOP and END statements serve two different purposes.

(i) The END statement provides an indication to the compiler that all the FORTRAN statements that precede it form a complete and separate program or subprogram in their own right.

(ii) The STOP statement is translated by the FORTRAN compiler as part of the machine program to be executed. When the program is executed and the STOP statement encountered, execution of it will cease and the computer will switch to the next job waiting.

3.12 ARRAYS OF VARIABLES

Many mathematical operations require the use of vectors and matrices. FORTRAN supplies a means of handling 1,2,3 or higher dimensional arrays. For the simplest array (the 1 dimensional vector), the i^{th} element of the vector v (v_i) is represented in FORTRAN as V(I). Elements of an array or vector are capable of being used in FORTRAN in the same way ordinary variables are employed.

```
V(I)=0.          the ith element of V is set to zero
A=V(I)+C(J)-D(3)
V(I-1)=V(3*I-7)
```

The subscripts used to refer to vector or array elements must be *INTEGER* and greater than zero. They may be constants, variables or expressions. The FORTRAN compiler reserves one location (word) for non-subscripted variables to be stored in. As subscripted variables take one location for each array element, it is necessary for the programmer to specify to the compiler the maximum number of elements associated with each array. This is done through a non-executable statement, the 'DIMENSION' statement that must precede the first use of the array it is defining.

```
    DIMENSIØN V(15)
    DØ  1 I=1,8
  1 V(2*I-1)=0.
```

The 'DIMENSION' statement would tell the compiler that V is a vector (1 dimensional) array requiring 15 storage locations. The supplied statements would set all the odd components of V to zero. The next example demonstrates how a vector may be used to calculate the mean and standard deviation of a set of 10 numbers. These numbers are read from 10 cards (*i.e.* 1 number per card).

$$\bar{X} = \frac{\sum_{i=1}^{10} x_i}{10}$$

$$\text{Standard deviation} = \sqrt{\frac{\sum_{i=1}^{10} (X_i - \bar{X})^2}{9}}$$

```
C      J. SMITH CALCULATE MEAN AND STANDARD DEVIATIØN
C      ØF 10 NUMBERS
       DIMENSIØN X(10)
       DØ  1 I=1,10
     1 READ(1,53)X(I)
    53 FØRMAT(F10•3)
       SUM=0
       DØ  2 I=1,10
     2 SUM=SUM+X(I)
       AVG=SUM/10•
       SUMSQ=0
       DØ  3 I=1,10
     3 SUMSQ=SUMSQ+(X(I)-AVG)**2
       SDEV=SQRT(SUMSQ/9•)
       WRITE(3,541)AVG,SDEV
   541 FØRMAT(' MEAN AND STANDARD DEVIATIØN ',2F10•3)
       STØP
       END
```

Here we use the vector X to store 10 numbers prior to finding the mean and standard deviation. Before employing vectors in a program, make sure they are really necessary. In a previous example (section 3.10), the mean of a set of numbers was required. There was no need in that case to retain the 10 numbers as the accumulated sum of each number as it was read was sufficient. When the standard deviation is sought, the numbers must be retained at least up to the point at which the mean is determined.

The next example demonstrates a program that computes the vector sum $\underset{\sim}{s}$ of two vectors $\underset{\sim}{u}$ and $\underset{\sim}{v}$

$$\underset{\sim}{s} = \underset{\sim}{u} + \underset{\sim}{v} .$$

For $\quad \underset{\sim}{u} = (3,5,2)$

and $\quad \underset{\sim}{v} = (4,2,7)$

then $\quad \underset{\sim}{s} = (3+4,\ 5+2,\ 2+7)$

$\qquad\quad = (7,7,9)$

Mathematically we say that the i^{th} component of $\underset{\sim}{s}$ is formed by

$$s_i = u_i + v_i \qquad 1 \leqslant i \leqslant 3$$

The program will read the three pairs of data from separate punched cards as shown

```
          u        v
                                
          10|      20|
---------------------------------------------
          3•       4•
          5•       2•
          2•       7•
```

into two vector arrays (U and V), compute the vector sum in S and print out each component of S on a separate line.

```
      DIMENSIØN S(3),U(3),V(3)
C     FIRST READ IN THE DATA
      DØ  1 I=1,3
    1 READ(1,100)U(I), V(I)
  100 FØRMAT(2F10•3)
C     NØW FØRM THE VECTØR SUM
      DØ  2 I=1,3
    2 S(I)=U(I)+V(I)
C     WRITE ØUT HEADING AND RESULTS
      WRITE(3,101)
  101 FØRMAT(' VECTØR S ')
```

```
      DØ  3 I=1,3
    3 WRITE(3,102)S(I)
  102 FØRMAT(2F10•3)
      STØP
      END
```

(In this example, it would have been possible to perform the vector addition operation without the use of subscripted variables (how? see appendix 3A for a solution). Such an operation, however, is frequently a small part of a much larger program where it is necessary to store the data in subscripted variables.)

When arrays of higher order than the one dimensional vector treated so far are needed, the 'DIMENSION' statement informs the compiler of the number of dimensions (*i.e.* the number of subscripts) and the total storage for the array.

```
      DIMENSIØN A(5,5)
```

This informs the compiler that A is a matrix (2 dimensional array) requiring 25 locations for storage

```
      DIMENSIØN A(5,5),B(5,5),C(5,5)
      .
      .
      .
      DØ  1 I=1,5
      DØ  1 J=1,5
    1 C(I,J)=A(I,J)+B(I,J)
      .
      .
      .
```

In this case, two matrices A and B are summed and the result stored in a new matrix C.

3.13 <u>SUBPROGRAMS</u>

We have met (section 3.8) the special mathematical functions supplied through the FORTRAN compiler. The user is able to supply two types of subprograms of his own when necessary:

- FUNCTION subprogram.
- SUBROUTINE subprogram.

Need for subprograms arises

(i) when the same mathematical function or procedure is required at many points in a program;

(ii) in larger programs where it pays to write and test sections of the code independently; and

(iii) when more than one person is responsible for developing the code.

The <u>FUNCTION subprogram</u> returns a single value as its result and is usually used to perform mathematical operations similar to $\sqrt{\ }$, or function evaluation. The user supplied function is best demonstrated by an example. Suppose we wish to evaluate a cubic polynomial for various values of x:

$$f(x) = 1 + 1.5x + 3.2x^2 + 6x^3 ,$$

which for speed of computation is best written as

$$f(x) = 1 + x\ (1.5 + x\ (3.2 + 6x)) \quad .$$

Then we might use the coding ...

```
      .                              ⎫
      .                              ⎪
      .                              ⎪
Y = F(X)+6•                          ⎪
      .                              ⎪
      .                              ⎪
      .                              ⎪
Z = F(X-1•)                          ⎬  Main or calling program
      .                              ⎪
      .                              ⎪
      .                              ⎪
STØP                                 ⎪
END                                  ⎭
FUNCTIØN F(A)                        ⎫
F = 1.+A*(1.5+A*(3.2+6••*A))         ⎬  FUNCTION subprogram
RETURN                               ⎪
END                                  ⎭
```

In the main program, the function is invoked by naming the function and enclosing in parentheses a constant, variable, or expression for which the cubic polynomial is to be evaluated. The FUNCTION subprogram is defined through the use of the 'FUNCTION' statement and an appropriate name 'F' (in this case) by which the function is to be known. An argument list corresponding to that in the main program is also required. The argument names in the function are only dummy ones and need not be the same as those in the main program (all the other variables and labels are local to the function and are in no way associated with labels or variables in the main program). When the above function is invoked twice by the main program, the values X and X-1• respectively are transferred into the location set aside for A. The function *must* return one value through the assignment of an arithmetic expression to the function name as in

```
F = 1•+A*(1•5+A*(3•2+6••A))
```

The 'RETURN' is a transfer of control from the function back to the main program from where control was originally passed. The END statement is once again a signal to the compiler that this is the end of a logically independent set of FORTRAN statements.

The SUBROUTINE subprogram is the more powerful version of a subprogram and usually performs more involved operations than those for which the FUNCTION is designed. Typical tasks for which subroutines are used would include finding the roots of equations, multiplication or inversion of matrices, and solving sets of linear equations. Unlike the function subprogram, the subroutine is not restricted to returning one result as part of an arithmetic expression. The SUBROUTINE and the main program communicate through the argument list only. The following code shows the use of a subroutine QUAD to determine real roots of a quadratic equation $ax^2 + bx + c = 0$. The coefficients of the equation to be solved are supplied as arguments to the subroutine, while the subroutine is responsible for returning the two roots and is an indication as to whether real roots were possible.

```
 1 READ(1,5)C1,C2,C3
   CALL QUAD(C1,C2,C3,X1,X2,IER)
   IF(IER•EQ•0)WRITE(3,57)X1,X2
   IF(IER•NE•0)WRITE(3,59)
 5 FØRMAT(3F10•3)
57 FØRMAT(' RØØTS ØF QUADRATIC ARE ',2F10•3)
59 FØRMAT(' NØ REAL RØØTS EXIST ')
   GØ TØ 1
   END
   SUBRØUTINE QUAD(A,B,C,R1,R2,K)
   DISC=B*B-4••A*C
```

```
  IF(DISC•LT•0) GØ TØ 5
  DISC=SQRT(DISC)
  R1=(-B+DISC)/(2•*A)
  R2=(-B-DISC)/(2•*A)
  K=0
  RETURN
5 K=1
  RETURN
  END
```

The subroutine is invoked, through a 'CALL' statement, by naming the subroutine and supplying a list of variables through which values are to be transferred to and from the subroutine. The main program passes the three coefficients of the quadratic while the subroutine will return the roots in X1 and X2 and an indication (K=1 or 0) to the sign of the discriminant. Once again the code within the subroutine is independent of the calling program.

3.14 ERRORS IN PROGRAMMING

The FORTRAN compiler will inform us in no uncertain terms of any syntactical errors we make in coding a program. Such errors are easy to detect and correct. The computer is a totally obedient servant; provided we ask it to perform a task in the language it understands, it will obey us without question. Therefore the hardest errors to identify are the ones we make in specifying the logic or steps involved in solving our problem. All programs should be considered guilty of containing bugs until proven innocent ('debugged').

Too often the poor computer is blamed for an error in the program that should have been found and removed by the programmer when he was 'debugging' his code.

garbage in   implies   garbage out

This adage is certainly true but the programmer and, in particular, the scientific programmer may find it difficult to recognise the output of a program for what it is. It is advisable to test programs thoroughly before placing any confidence on their output. This is often done by comparing the computed solution with a known mathematical or physical solution. When agreement is satisfactory we may then proceed to use our program for all the cases we are interested in.

Unlike the more commercially oriented programmer, the problems faced by a mathematical programmer are three-fold. As most commercial tasks are well defined, errors in the computer output are directly due to the program or incorrect data on which it operated. The scientific problem solver is solving a mathematical model of some real physical system. When this model was developed, many assumptions (and probably simplifications) were made. Just how valid were these and are they the source of errors? Were the errors caused by the type of numerical technique chosen to solve the model? Or were the errors due to the coding of these techniques?

3.15 PRACTICE EXAMPLES

Before you attempt to code the least squares problem described in chapter 2, try these practice examples. The answers to the questions are given in section 3.16, but don't be too hasty to seek out these answers until you have had a go yourself.

Q1. (a) In the list below, which items are variables or constants?

(b) What is the mode (integer or real) of each variable or constant in the list?

(c) Are any invalid?

<u>List</u> (1) 1., (2) ABC, (3) I4, (4) 14, (5) -0.0001E-10, (6) INKSTAIN, (7) FIVE, (8) 6IX, (9) e, (10) 0, (11) BØS, (12) A*B, (13) 5,312.6, (14)+5.E-03, (15) BLOT

Q2. Write each of the following algebraic formulae as a FORTRAN statement to calculate y. Use any convenient real names for the variables, which will be assumed to have been assigned values by previous steps of the program.

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

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

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

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

(5) $y-x = a-\pi y$ $(\pi=3.141592)$

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

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

What values would be stored in the variable on the left of the following arithmetic statements, given that A=3?

(8) I=A

(9) I=A/2.

(10) U=A/2.

Q3. Write the necessary statements of portion of a program to calculate the variables given by the following expressions. Use any convenient 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 calculating the expressions - for example $a \neq 0$ in (1).

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

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

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

(4) $v = \tan x$

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

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

(7) $c = \ell n \left|\frac{1}{1+a^3}\right|$

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

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

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

Q4. Write a program that will

(1) Read the four coefficients of a cubic polynomial $f(x)=a+bx+cx^2+dx^3$ from a punched card in FORMAT (4F10.3).

(2) Read the estimate x_0 of a root of the equation f(x)=0 from a second card (FORMAT(F10.3)).

(3) Improve the estimate of the root by the Newton-Raphson method

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

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

$$\frac{|x_{n+1} - x_n|}{|x_n|} < 0.001$$

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

(6) Allow only 5 iterations. If convergence has not been achieved, print a message warning of this.

(7) Repeat from (1).

Q5. (For advanced students only)
Read in a set of 10 numbers punched one per card (FORMAT(F10.3)). Write code that will sort these numbers in descending order.

3.16 ANSWERS AND TYPICAL CODING

Q1. (1) real constant, (2) real variable, (3) integer variable, (4) integer constant, (5) real constant, (6) invalid variable name as more than 6 characters, (7) real variable, (8) invalid variable name as first character is not alphabetic, (9) invalid variable name as e is a lower case letter, (10) integer constant, (11) real variable, (12) invalid since an expression is not a variable, (13) invalid as comma is not permitted, (14) real constant, (15) real variable.

Q2.
```
 (1)        Y = 0.5*(B+C)
 (2)        Y = 0.3333333*(A+B)*(A+B)
 (3)        Y = 1./(1./A+1./B+1./C)
 (4)        Y = 1.+X*(1.+X*(0.5+0.1666667*X))
 (5)        Y = (X+A)/4.141592
 (6)        Y = U*U
 (7)        Y = 1./(X-1./B-1./C)
 (8)        I = 3
 (9)        I = 1
(10)        U = 1.5
```

Q3.

```
   (1)        X = (-B+SQRT(B*B-4.*A*C))/(2.*A)
   (2)        S = SQRT(X*X+Y*Y+Z*Z)
   (3)        W1 = EXP(X)
              W2 = 1./W1
              U = (W1-W2)/(W1+W2)
   (4)        V = TAN(X)
   (5)        W1 = X*X
              H = 1.-W1*(0.5-0.04166667*W1)
   (6)        W1 = EXP(-X)
              Y = 1.-W1*(1.+W1*(0.2+0.04*W1))
   (7)        C = -ALOG(ABS(1.+A*A*A))
   (8)        W1 = X*Y/3.141592
              G = SQRT(1./W1)*SIN(W1)
   (9)        W1 = A*X
              Y = (EXP(W1)+EXP(-SQRT(W1)))*0.3333333
  (10)        Z = -ATAN(X/A)/(1.+U/A)
```

Q4.

```
C     J. SMITH
C     RØØT ØF CUBIC EQUATIØN
C     READ CØEFFICIENTS FRØM ØNE PUNCHED CARD
    9 READ(1,56)A,B,C,D
   56 FØRMAT(4F10•3)
C     READ IN ESTIMATE FØR RØØT
      READ(1,57)XN
   57 FØRMAT(F10•3)
C     LØØP TO IMPRØVE ESTIMATE
      DØ  1 I=1,5
      XNP1=XN-(A+XN*(B+XN*(C+D*XN)))/(B+XN(2.*C+XN*3.*D))
C     NØW ASK IS PRØCESS CØNVERGING
      IF(ABS((XNP1-XN)/XN)•LT••001)GØ TØ 2
    1 XN=XNP1
C     RØØT HAS NØT BEEN FØUND
      WRITE(3,58)
   58 FØRMAT(' RØØT NØT FØUND WITHIN 5 ITERATIØNS ')
C     TRY ANØTHER SET ØF CØEFFICIENT
      GØ TØ 9
C     RØØT LØCATED WITHIN PRESCRIBED BØUNDS
```

```
     2 WRITE(3,59)XNP1
    59 FØRMAT (' RØØT ØF CUBIC = ',F10•3)
C      TRY ANØTHER SET ØF CØEFFICIENTS
       GØ TØ 9
       END
```

Sample data cards:

```
column  10|      20|        30|        40|
_____________________________________________________
       5.6     3.7        2.         -46.}
       1.2                                }
```

Q5.

```
C      J. SMITH
C      SØRTING PRØBLEM
       DIMENSIØN X(10)
C      SET UP LØØP TØ READ 10 NUMBERS
       DØ  1 I=1,10
     1 READ(1,9)X(I)
     9 FØRMAT (F10•3)
       DØ  2 I=1,9
       N=I+1
       DØ  2 J=N,10
       IF(X(J)•LE•X(I))GØ TØ 2
       TEMP=X(J)
       X(J)=X(I)
       X(I)=TEMP
     2 CØNTINUE
       .
       .
       .
```

CHAPTER 4

ANALOGUE COMPUTING AND DYNAMICS

Lecture by

C.P. GILBERT

ABSTRACT

The related concepts of analogues and simulation are described, and the electronic analogue computer is introduced as the most convenient means of building simulators. The use of such computers for the solution of differential equations is illustrated by examples having a decaying exponential type of response, and the uses of hybrid computers are briefly mentioned. Behaviour characterised by increasing exponentials is described, mainly with reference to population growth.

CONTENTS

4.1 INTRODUCTION

4.1.1 Dynamic Systems

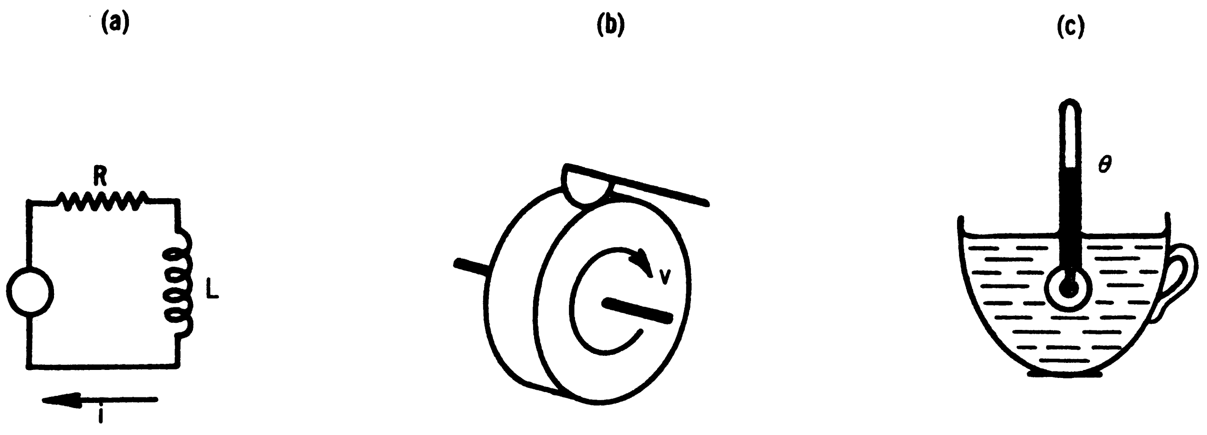
Many advances in science and engineering are possible only because of our ability to use mathematical equations to describe the behaviour of complicated systems.

Here we are concerned with what are known as *dynamic* systems, *i.e.* those that vary with time, which are usually described using differential equations. An 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.

While these lectures will concentrate on systems having an exponential response, we must remember that the methods are normally applied to much more complicated systems, such as a complete nuclear reactor.

4.1.2 Analogues

When dynamic systems are examined in detail, one important property emerges. Many electrical, mechanical, biological and other systems can be described by equations of the same *form,* although the actual numbers involved may be different in each case. Figure 4.1 shows simple examples



$$\frac{di}{dt} = -\frac{R}{L}i \qquad \frac{dv}{dt} = -\frac{C}{I}v \qquad \frac{d\theta}{dt} = -\frac{H}{M}\theta$$

Figure 4.1 (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

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:

$$\frac{dx}{dt} = -Kx \quad . \qquad \ldots(4.1)$$

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. As we shall see, this can assist us with the solution of very complicated sets of differential equations.

4.1.3 Simulation

One way to examine the behaviour of a nuclear reactor, say, would be to do an experiment. A disturbance would be purposely injected by some means, and the reactor power and temperature would be measured as they varied with time. Unfortunately, with a *full-size* power reactor the experiment would be slow, very expensive and possibly 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.*

4.1.4 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 4.2(a) 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:

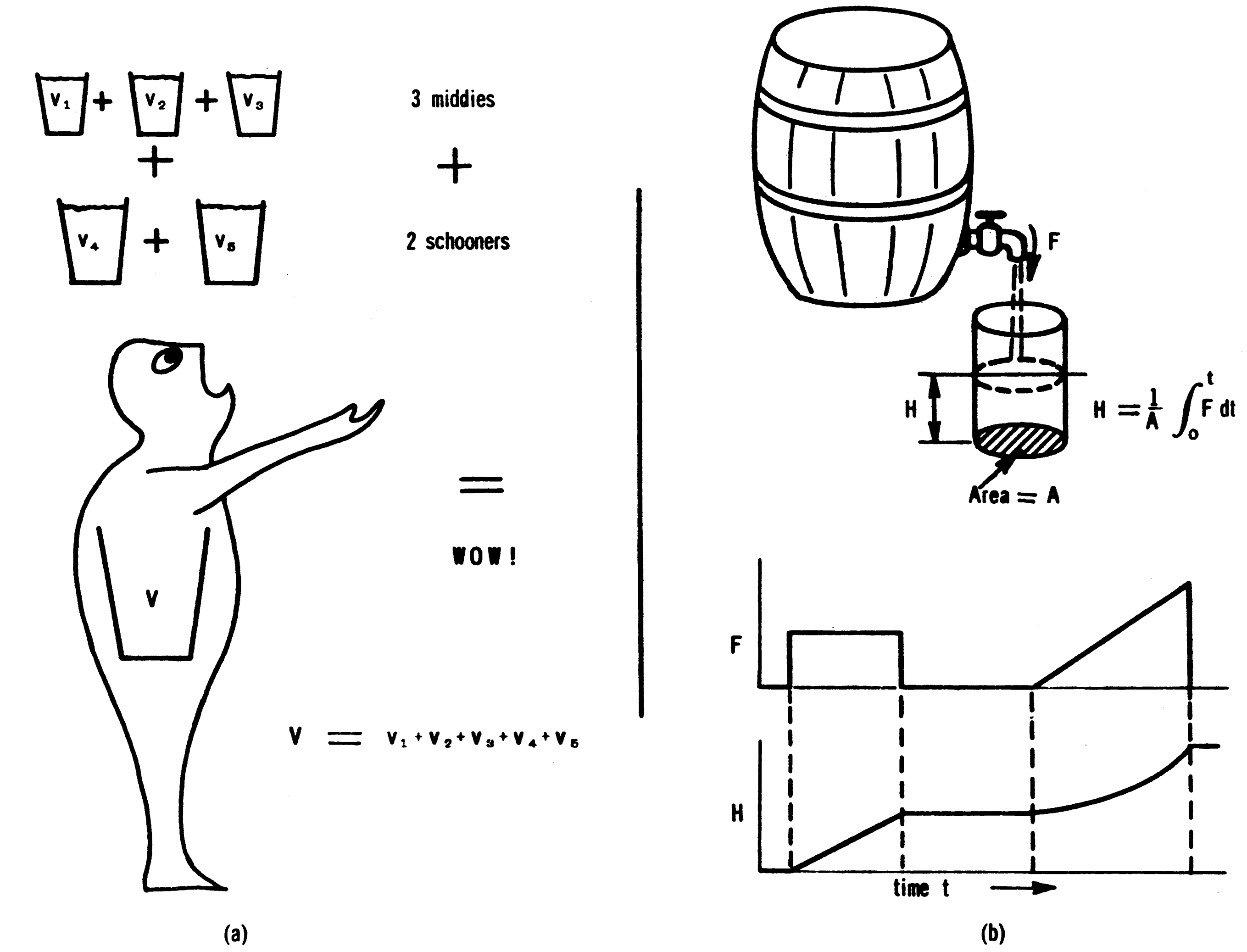


Figure 4.2 Liquid analogues for (a) addition, using volumes, and (b) integration

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

The more important process of *integration* can be achieved as shown in figure 4.2(b). 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:

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

These simple analogues would be of little use in practice; they are inaccurate, slow, unsuitable for interconnection, and probably wet. However, there are much better analogue processes available; the most useful of them uses an electronic amplifier and is described in the following section.

4.2 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 4.3(b)), of an adder (figure 4.3(c)), and of an integrator (figure 4.4(c)).

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 4.3(a) 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 can never exceed a few microvolts and so 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

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

Thus the output is minus the sum of the input voltages. The resistance ratios R_f/R_i 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 4.3(b), is simply a device for reducing the size of a voltage by an amount which can be set very accurately.)

The complete *adder* circuit is conventionally drawn as shown in figure 4.3(c), the values of the resistance ratios being marked *only* if they are other than unity. Then

$$V_o = -[0.3\ v_1 + 1.6\ v_2 + v_3] \quad . \qquad \ldots(4.2)$$

Note that all voltages are measured with respect to earth, although the earth connection itself is omitted. It is a pity that the amplifier gives a reversal in sign, but it is unavoidable, and causes little difficulty.

In the *integrator* circuit of figure 4.4(a), a feedback capacitor C is used. Assuming as before that the amplifier input u = 0,

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

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

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

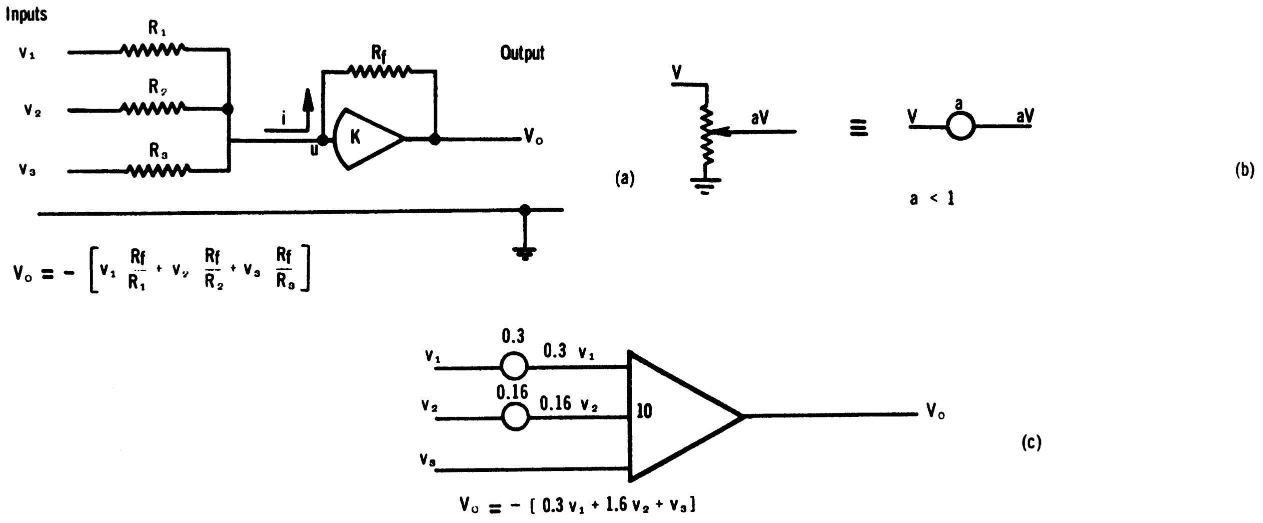


Figure 4.3 A circuit for addition: (a) Circuit details (b) Potentiometers (c) Circuit using conventional symbols: the whole of the circuit (a) is contained within the triangle

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 4.4(b)). Unless otherwise shown, the time constant CR can be assumed to be unity, and the integrator circuit is conventionally drawn as shown in figure 4.4(c), for which

$$V_o = - 0.6 \int_o^t v_1 \, dt \quad , \qquad \ldots(4.3)$$

or

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

If more than one input is applied to the circuit of figure 4.4(c), their sum is integrated.

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

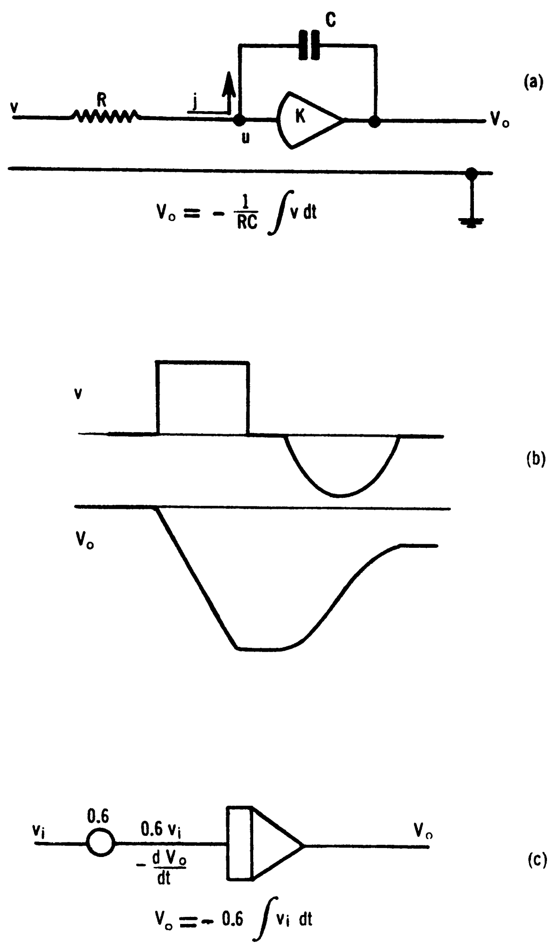


Figure 4.4 A circuit for integration: (a) Circuit details (b) Typical waveforms (c) Circuit using conventional symbols: the whole of circuit (a) is contained within the special trianglar symbol. More than one input circuit is permitted

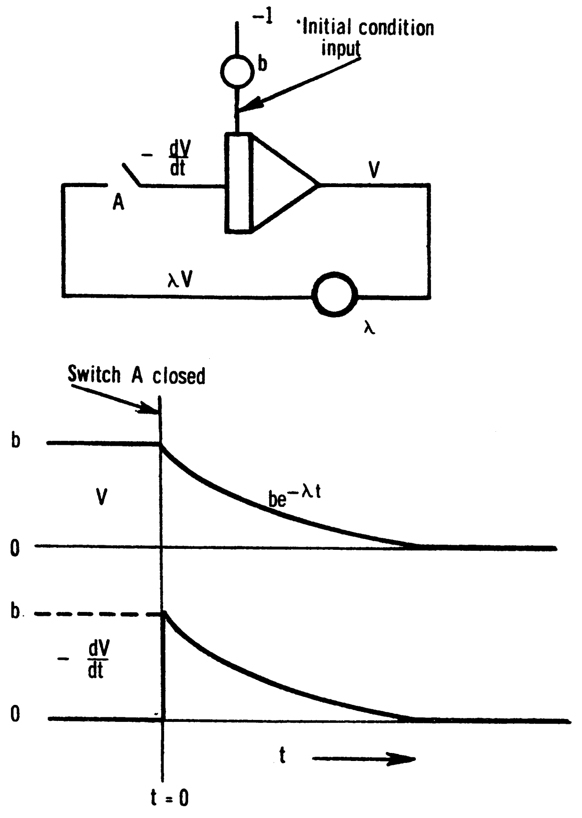


Figure 4.5 A circuit for the solution of $\frac{dV}{dt} = -\lambda V$ and typical waveforms. The input via b only provides the starting voltage

4.3 ELECTRONIC ANALOGUE COMPUTERS

4.3.1 General

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, concentration, 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.

4.3.2 Equation Solution

Consider the circuit of figure 4.5. 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. Using a potentiometer, we have now made the integrator input equal to λV, and so the circuit obeys the equation

$$-\frac{dV}{dt} = \lambda V \qquad \text{or} \qquad \frac{dV}{dt} = -\lambda V \quad . \qquad \ldots(4.4)$$

This is basically the same as equation 4.1, which describes the systems of figure 4.1, and so the circuit of figure 4.5 is simply one more analogue, having the same dynamic properties as the other three systems. As you know from a previous lecture, the solution to equation (4.4) is an exponential: if we let $V = ke^{-\lambda t}$, where k is an unknown constant, then differentiating we get

$$\frac{dV}{dt} = -\lambda k e^{-\lambda t} = -\lambda V \quad .$$

This demonstrates that $V = ke^{-\lambda t}$ is *a* solution of equation (4.4). Since we have made V = b at t = 0, substitution shows that k = b, and so *the* solution is $V = be^{-\lambda t}$.

The circuit of figure 4.5 'solves' equation (4.4) by producing a voltage proportional to $be^{-\lambda t}$ each time switch A is closed. V starts off positive and, *via* the integrator, forces itself to get smaller. As it does so, its rate of change also gets smaller, which is precisely what equation (4.4) tells us in a more compact way.

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 4.1, possibly with a very complicated series of disturbances, the simplest and most accurate way of doing the experiment would be to apply a voltage representing the disturbances to the circuit of figure 4.5.

4.4 PROBLEM SOLUTION

To obtain the above solution we started with a computer circuit and *analysed* its behaviour. The usual process is the other way round - we are given a set of equations and have to *design* a circuit which will solve them, resulting in the process illustrated in figure 4.6. 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.

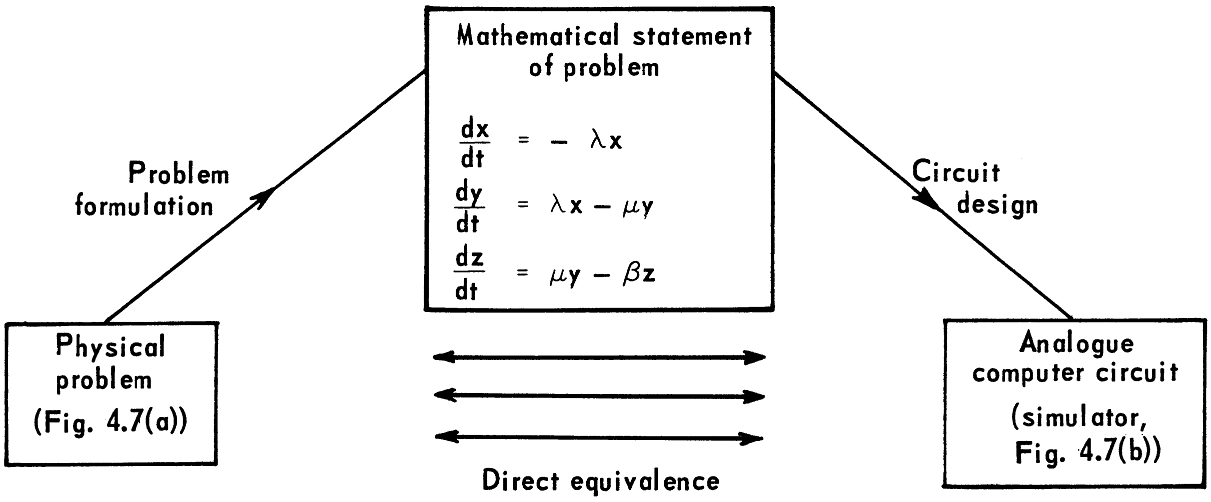

Mathematical statement of problem
$\frac{dx}{dt} = -\lambda x$
$\frac{dy}{dt} = \lambda x - \mu y$
$\frac{dz}{dt} = \mu y - \beta z$
Problem formulation
Circuit design
Physical problem (Fig. 4.7(a))
Analogue computer circuit (simulator, Fig. 4.7(b))
Direct equivalence

Figure 4.6 Pictorial representation of the analogue method of problem solving

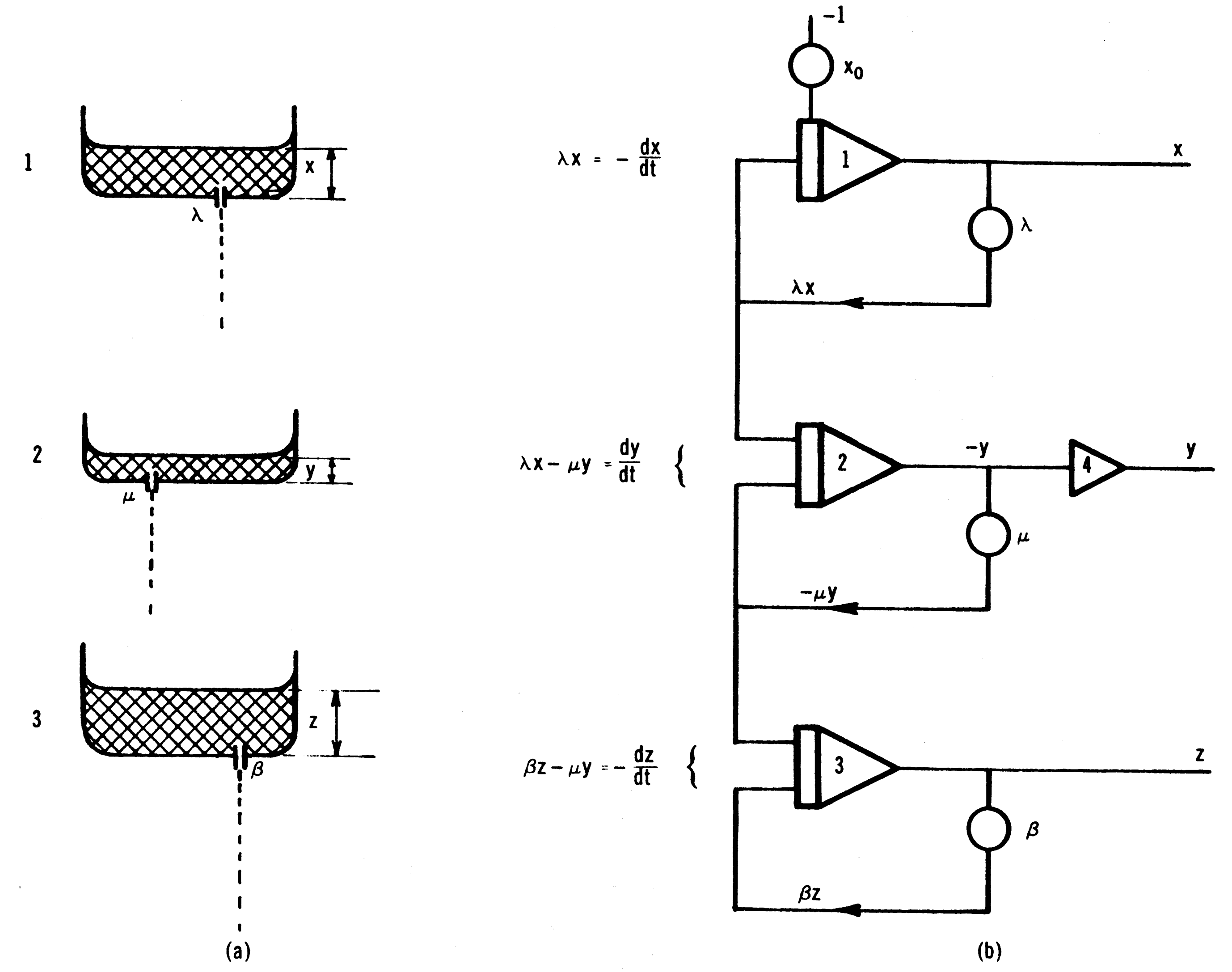


Figure 4.7 (a) A problem in hydraulics
(b) A simulator to represent the problem

As an example of the process of figure 4.6, consider the series of tanks in figure 4.7(a), each having a drain hole through which it leaks into the next tank. The depth of water in the first tank is x, and the surface moves (or the depth changes) with velocity dx/dt, which depends on the flow of water in and out. For the first tank the inflow is zero, although the experiment starts (the plug is pulled out) with an initial depth x_o. The outflow depends on the size of the hole, denoted by λ, and the depth (*i.e.* pressure) of water. Thus

velocity of surface = inflow - outflow

i.e. $$\frac{dx}{dt} = 0 - \lambda x \quad ,$$

or $$\frac{dx}{dt} = -\lambda x \quad . \qquad \ldots(4.5)$$

and, by comparison with equation (4.4), we know that x will fall exponentially.

However, the second tank, which starts empty, has an inflow from tank 1 as well as the normal outflow; its rate of change of depth is thus

$$\frac{dy}{dt} = \lambda x - \mu y \quad . \qquad \ldots(4.6)$$

Similarly for the third tank,

$$\frac{dz}{dt} = \mu y - \beta z \qquad \ldots(4.7)$$

and one could go on indefinitely. This system of tanks gives a very clear idea of how one radioactive material decays into another, which itself decays (at a different rate) into a third, because the equations describing that situation are identical to equations (4.5) to (4.7), *i.e.* the two systems are analogous.

Our simple exponential solution only fits the first tank, the change in depth in the others being complicated by their varying inflow. However, we have successfully *formulated* the problem of figure 4.7(a), and can now go on to design the computer circuit.

From figure 4.5 we know that we can solve equation (4.5), using integrator 1 of figure 4.7(b) to represent tank 1. The potentiometer introduces λ, the size of the drain hole (or the decay constant of a radionuclide).

Consider now equation (4.6). Let us assume that a signal representing $-\mu y$ is available; then with the existing λx signal we can make up dy/dt. This is integrated (and inverted) in integrator 2 to give $-y$, and so we can supply the wanted $-\mu y$ signal from the potentiometer. The circuit of integrator 3 solving equation (4.7) can be found in exactly the same way, except that the signs are all reversed, and so we have developed an analogue computer circuit to simulate the water levels in the three tanks. An inverting amplifier (4) allows y to be viewed the right way up.

Notice from the demonstration 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 levels in the tanks 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 much faster than can be achieved by a digital computer. However, the high speed cannot be properly utilised by a human operator.

4.5 HYBRID COMPUTERS

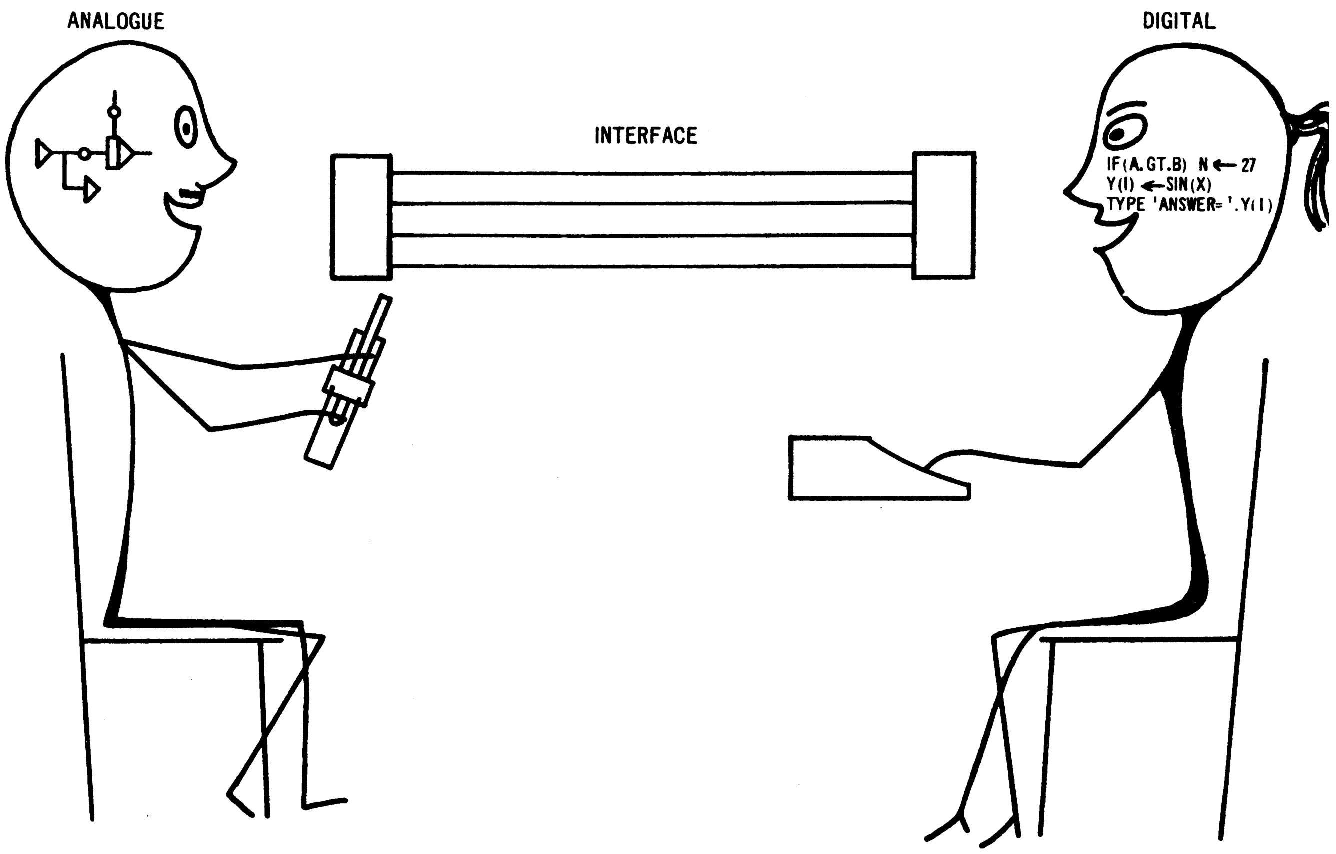
A fairly recent development, whose full impact has not yet been felt, is the Hybrid Computer. This consists of an analogue computer, a general purpose digital computer, and an interface. The latter provides the facilities required for the two machines to cooperate effectively (figure 4.8).

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, which 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.)

As an example, suppose we wanted to find the value of λ for an experimental result thought to be an exponential. The operator could use the circuit of figure 4.5 and, by comparing the output with the wanted curve, he could adjust the potentiometer to get a better fit.



```
IF(A.GT.B) N←27
Y(I) ←SIN(X)
TYPE 'ANSWER='.Y(I)
```

Figure 4.8 A hybrid computer

After a number of trial and error solutions he could get a reasonable match and the value of λ would be given by the potentiometer setting. This process would be tedious and probably inaccurate.

However, with the digital section of the hybrid computer performing the comparison and resetting the analogue section, many trial solutions would be performed in one second, and an accurate result could be obtained very quickly.

4.6 EXPONENTIALS AND EXPERIENCE

4.6.1 Increasing Exponentials

So far we have talked mainly about quantities which decay exponentially ($e^{-\lambda t}$) since these are very common in practice. However, it is quite possible for λ itself to be negative (corresponding to a drain hole which squirts water *into* a tank), and so we end up with e^{kt} where k is positive, giving an increasing exponential of the type shown in figure 4.9. Whereas before we thought in terms of a 'halving-time', now we must think of a 'doubling time' T.

Clearly such a response cannot continue indefinitely; it *must* stop somewhere. Fortunately such transients are very rare, although as you know, a nuclear reactor can theoretically behave in this way (until it melts) if it is wrongly designed and carelessly operated.

Compound interest on a sum of money gives exponential growth - 5% compound interest leads to a doubling time of about 14 years. Another situation involving growing exponentials is dealt with in the next section.

4.6.2 Populations

Consider a colony of 100 grubs. Given sufficient food and space, an average of 20 eggs are produced by each grub per month, of which 10 eggs hatch out and produce grubs which survive to the egg laying stage. Remembering that the original grubs die at the end of the month, the grub population increases by a factor of ten each month.

Months	0	1	2	6	12	n
Population N	10^2	10^3	10^4	10^8	10^{14}	$10^{(n+2)}$

(If the grubs are each 1 millimetre long, 10^{14} of them placed end-to-end would stretch for 10^8 kilometres! The moon is only 4×10^5 km away.)

The population curve can be fitted by the equation $N = 100\, e^{2.3t}$ where t is in months (figure 4.9), and so the original differential equation must have been

$$\frac{dN}{dt} = 2.3N \quad .$$

Thus the grub population is expanding exponentially with a doubling time of about nine days.

The most frightening thing about such an increase is its insidious speed. If you only observe the past, it is difficult to realise just how quickly things will move in the future, and any delay can turn a difficult situation into an impossible one.

Fortunately for us, natural populations run out of food or space, or reach some other limitation, possibly of the type discussed next, a predator-prey situation.

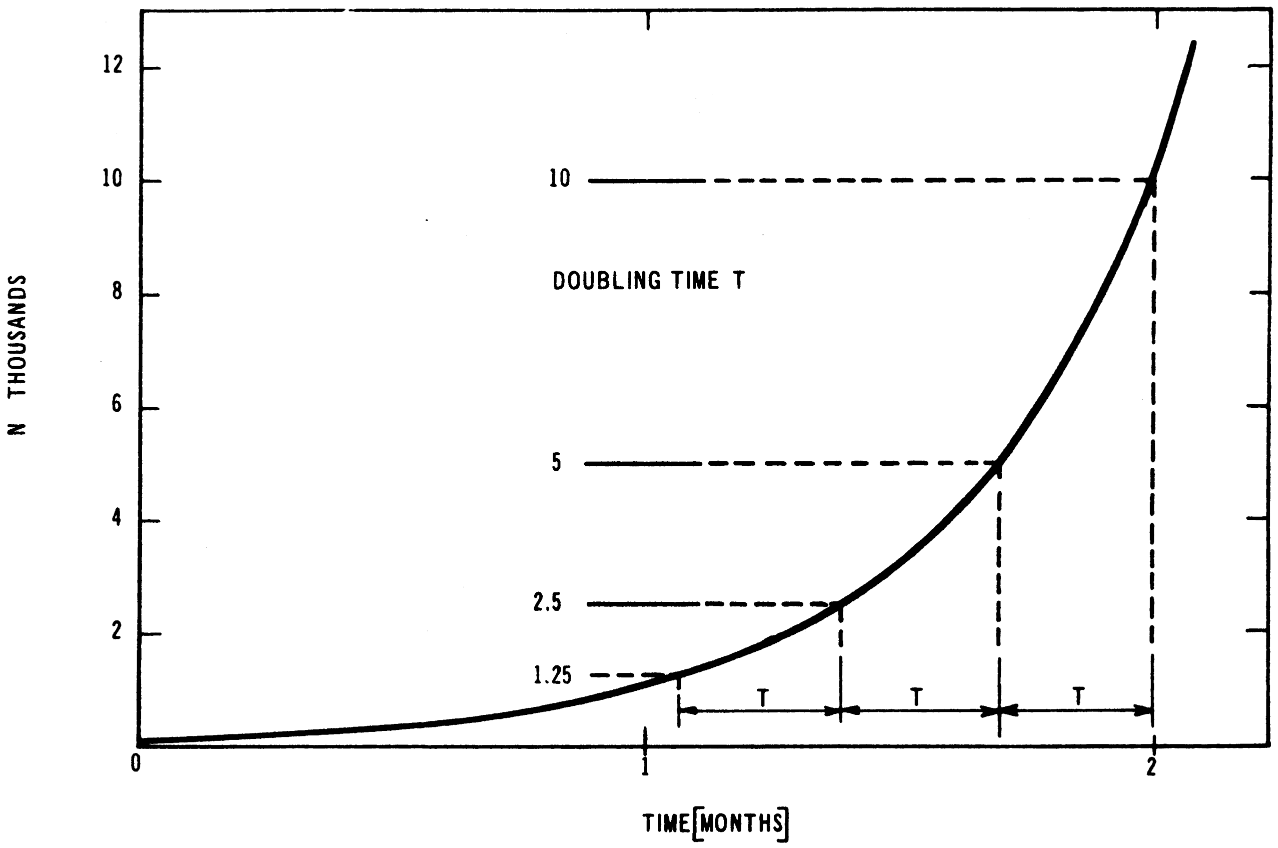


Figure 4.9 The exponential $N=100e^{2\cdot 3t}$ showing the growth of a population of grubs

4.6.3 Sharks and Little Fishes

The fish population is similar to the grub population: there is plenty of food and the oceans are large, so their numbers would increase exponentially, were it not for the sharks who live on fish. The rate of change of the fish population F is

$$\frac{dF}{dt} = k_1F - k_2F.S$$

The first term on the right hand side represents the normal growth rate, and the second the rate at which fish are eaten. The latter depends on both the number of fish *and* the number of sharks, and introduces a product term which is new to us, but which computers can handle easily. For the sharks,

$$\frac{dS}{dt} = -k_4S + k_3F.S \quad .$$

The first term accounts for the sharks who die or who leave the area simply because there are too many other sharks around already; the second term represents the shark's birthrate, which again depends upon the product F.S. (the number of shark parents, and the food supply). While this model is oversimplified it has some interesting properties, including cyclic, or oscillatory behaviour, alternating between famine and plenty (for the sharks).

4.6.4 The Really Super Important Problem

The only population which has been encouraged to expand unchecked is the human population. For over 300 years it has been growing *more* than exponentially, *i.e.* initially with a doubling time of 250 years, but now, at a level of over 3,500 million, with a doubling time of only 33 years.

Not only this, but the human race is using up unrenewable resources (oil, coal, metals, *etc.*), and generating pollution at rates which are also growing exponentially, both because of the rising number of people and because of a rising material standard of living. Clearly this type of growth cannot continue indefinitely or there will be no room left, insufficient food and virtually no raw materials.

In view of what we know of exponentials, it is clear that the human race must manage its affairs better in the future by finding ways of limiting both the population and its usage of the world's resources to levels that our planet can sustain. Unless this is done, nature will apply one of her own traditional methods of limitation - famine and disease, probably preceded by war.

Also we know that every delay in coming to grips with the problem makes matters worse-in fact a delay of only 33 years doubles the size of the problem.

The past four or five generations have worked hard to provide better material standards of living, and so have helped to increase the population and accelerate the use of natural resources. The present generation is continuing to do this, due to sheer inertia and bewilderment, but at least it has realised that a serious problem exists. It will be the responsibility of the next generation to start dealing with these formidable difficulties.

CHAPTER 5

INTERACTIVE COMPUTING WITH ACL

Lecture by

R.P. BACKSTROM

ABSTRACT

Interactive computing, in which man and machine work closely together in setting up a computation, is described in terms of a locally developed language ACL implemented to support many users of a NOVA minicomputer.

CONTENTS

CORRIGENDUM

AAEC/S17 SUMMER SCHOOL, 1976

Section 3.3, p.3.4.

Second line of sub-section (iii) should read

,˙,',(,),=,$;   and

5.1 INTRODUCTION

ACL is a high-level language designed specifically to suit scientific problems. The language was developed at the Australian Atomic Energy Commission Research Establishment, Lucas Heights, New South Wales [Bennett & Sanger 1973], and has been implemented by Dr P.L. Sanger on a NOVA computer presently supporting 18 terminals [Sanger 1972]. This is referred to as the ACL-NOVA system. The NOVA computer is also linked into the on-site computer network (referred to as the Dataway) and therefore has access to the IBM360 Model 65 central computer. This allows ACL programs developed by users at the terminals to be saved on IBM360 disk storage for later recall to the NOVA computer.

The ACL language consists of immediate statements and stored statements. Stored statements are translated and saved in the user's work area within the NOVA computer from which they may be recalled and executed under user program control. Each stored statement has a sequence number in the range 100 to 999 and may also have a statement number in the range 0 to 99. Stored statements are used to build up a stored program. They are ordered according to their sequence number and may be inserted, modified or deleted by the user. Stored statements may be typed in any order; newly entered statements replace previously saved statements with the same sequence number.

An immediate statement which does not have a sequence number is translated and executed when typed and is then discarded. These statements are used to perform one-time or 'desk calculator' type calculations, to control the execution of a stored program and to perform various editing and debugging functions.

Work space within the NOVA computer is allocated dynamically, thus allowing large programs to be given extra storage areas when necessary. The work space thus obtained is subsequently returned to the system when the user completes work at a terminal.

5.2 TERMINAL INPUT

To begin an ACL session at a terminal, the user should hold the CNTRL key and then press the letter G. The system responds with the message ACL-NOVA and spaces a few lines. Once the terminal has been initialised in this way, statements may be stored or immediate statements executed.

Input begins from column 1, which is taken to be the leftmost position of the teletype carriage that results from pressing the Carriage Return key; it may consist of up to 72 characters. If input is continued past column 72, the whole line is cancelled and must be entered again.

A stored statement consists of a 3-digit sequence number starting in column 1, an optional statement number starting in column 5, and the statement itself starting in column 8, as shown in the following layout:

```
|1      |5  |7|8                                       72|
110     1     ACCEPT A,B
120           C←A+B
130     24    TYPE 'SUM = ',C
```

Immediate statements begin in column 1 as follows:

```
|1                                                     72|
EXP(2.30259)
A←3.141593*R*R
RUN
```

The terminals attached to the NOVA computer are operated in full-duplex mode; this means that a character pressed at the terminal keyboard is sent to the computer, but it is not printed out unless it is accepted by the NOVA and 'echoed' under program control by the ACL-NOVA system. Therefore, only characters which are valid at each point are 'echoed' at the terminal. For example, if a user pressed the keys SQR2 at the keyboard, the NOVA computer would recognise SQR as the start of the function 'square root' and would then expect an opening bracket, (. The 2 would be rejected and, instead a 'bell' character would be given to indicate that an invalid character had been pressed.

The full-duplex mode feature is also used to simplify the task of entering stored statements from a terminal. If the user gives a space at column 1, the system responds with a sequence number followed by a space. This sequence number is ten greater than the last sequence number specified. In the same way, if a space is entered at column 5, the system responds with three spaces. Thus for a stored statement, a new sequence number and no statement number may be generated by pressing the space bar twice.

Stored statements may be deleted by entering the sequence number, a space and carriage return. Statement numbers may be altered (or omitted) by entering the sequence number, a space, a new statement number (or space to omit the statement number), one more space and carriage return.

5.3 ARITHMETIC OPERATIONS

All arithmetic operations are performed using single-precision floating-point numbers which give between 6- and 7-digit accuracy. For input, the numbers may have free format, that is they may be integers, may contain a decimal point and may also contain an exponent. Examples include:

257, 1.704, -.00193, 1.E-7 and 4E47

The operations available under ACL are +, -, * (multiplication), / (division) and ↑ (power). The usual hierarchy applies, *i.e.* ↑ first, * or / next and then + or -. The following functions are also available in ACL arithmetic expressions:

Function	Description
ABS	Absolute Value
ATN	Arctan (answer in radians)
COS	Cosine (argument in radians)
DPT	Decimal Point Function - see under TYPE statement
EXP	Exponential
INT	Integer Part (INT(4.7)=4, INT(-1.5)=-2)
LOG	Natural Logarithm (base e)
SIN	Sine (argument in radians)
SQR	Square Root (gives +ve or zero result)

Note that there is no TAN (tangent) function. This may, however, be constructed using SIN(X)/COS(X).

5.4 VARIABLES

Three kinds of variables are supported by ACL: simple variables, singly subscripted and doubly subscripted variables.

A simple variable name must begin with an alphabetic character (*i.e.* upper case A to Z) and may be followed by up to three alphanumeric characters (*i.e.* upper case A to Z or 0 to 9). Subscripted variable names consist of an alphabetic character which may be followed by an alphanumeric character, followed by the subscript (or subscripts separated by a comma) enclosed in brackets. Singly subscripted variables must have subscripts in the range 0 to 65535; doubly subscripted variables must have subscripts in the range 0 to 255.

Examples of valid variable names are:

A, AREA, X1, X(I), Z3(I,7) and Z3

It is recommended that subscripted variables not be used unless every value needs to be stored separately. For example, to total n numbers, the total may be obtained by starting with zero and adding each number to the total as it is entered into the computer. This will conserve the work space allocated and allow more users simultaneous access to ACL-NOVA.

5.5 ASSIGNMENTS

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

```
X2←F-E+0*X1←(F←-B/D)+E←SQR(B*B-4*A*C)/(D←2*A)
```

D is evaluated first, then E, then the value of F and finally X1 and X2. This expression in fact gives the real roots of the quadratic equation $Ax^2 + Bx + C = 0$. If B*B-4*A*C happens to be negative, however, ACL will issue an error message because it cannot find the square root of a negative number. The program is then suspended while the user makes some corrections (if necessary) to the coefficients and presses carriage return to resume processing of the above expression. The user may decide not to continue with that calculation, but type RUN to restart the program, this time entering different coefficients.

5.6 INPUT AND OUTPUT STATEMENTS

Numerical values may be entered into an ACL program using the

ACCEPT statement followed by a list of variables separated by commas, for example:

```
        110       ACCEPT  A,B,C
```

When this statement is executed, ACL will print out the sequence number of the ACCEPT statement and the name of the variable to be entered, thus:

```
        110  A←
```

and will wait until a numerical value (or indeed any valid arithmetic expression) is typed followed by a carriage return. The values for B and C will then be requested in a similar manner.

The TYPE statement provides the means by which printed output can be produced from an ACL program. The format of numerical output is determined by the magnitude of the numbers, and will be in exponent form if in the range $|\text{number}| \leqslant 10^{-4}$ or $|\text{number}| \geqslant 10^{3}$. All other values appear with up to seven significant figures in decimal notation (except that integers will be printed without a decimal point).

The simplest example of a TYPE statement would be one in which the value of a single variable is typed at the left hand margin of the page, for example, using:

```
        200       TYPE  X1
```

If the value of X1 is required in exponent form regardless of its magnitude, we would code:

```
        200       TYPE "X1
```

The values of several variables may be typed on the one line (with one blank separating each number) as follows:

```
        300       TYPE I,X(I),Y,Z
```

Headings or descriptions of results may also be printed using the TYPE statement, provided the message is enclosed in single quotation marks, for example:

```
        340       TYPE 'THE TOTAL IS ', TOT
```

would produce the line:

```
        THE TOTAL IS 4.70319
```

If the output is required to be positioned exactly on the line, the number of positions to leave at the left of the output variable may be specified as an arithmetic expression enclosed within the symbols < and >. Thus to leave ten blanks before printing the value of X, we could use:

```
        400       TYPE <10>,X
```

The expression between the symbols < and > may be variable, thus providing a way of producing graphs of properly scaled functions. For example:

```
        500       TYPE <20+20*SIN(I←I+.2)>,'*'
```

would calculate a value between 0 and 40, space that many blanks and then print the *. In this way, a rough sine curve can be depicted on the teletype.

The TYPE statement may also contain colons (:) and semicolons (;) as delimiters. A colon is used to print the output, and issue a carriage return but not a line feed, enabling another TYPE statement to overprint the last line (where a complicated line could perhaps not be described in a single TYPE statement). For example:

```
        610       TYPE  A,B,C,D,E,F,G,H,I,J,K,L:
        620       TYPE  <40>,M,N,O,P,Q,R
```

would print the first twelve values, give a carriage return, space 40 blanks and then print the other six values (provided that all these numbers are small enough to fit on a teletype with only 72 print positions). Semicolons are used to space to the next line before printing the rest of the output, for example:

```
        500       TYPE A;B;;
```

would print the values of A and B at the left hand margin of successive lines and leave 1 blank line (not two) after printing the value of B.

To produce a column of figures of varying numerical value with the decimal point aligned, the DPT function may be used in the positional descriptor. DPT(X) will give a value which is the character position of the real (or virtual) decimal point. This means that to position a number so that the decimal point is in column 10, the expression 10-DPT(X) will give the number of spaces to leave so that the alignment is made. The TYPE satement would then be:

```
        600       TYPE <10-DPT(X)>,X
```

5.7 ERROR CORRECTION AND THE EDIT STATEMENT

Errors which occur while a statement is being typed may be corrected quite simply. For example, to delete the last seven characters that were accepted as input, type <7. This would cause the original line of input minus the last seven characters to be typed on a new line and the rest of the line could then be typed in.

A statement being entered may also be edited by typing << and a carriage return (see section 5.8).

To delete the entire line before final acceptance by the computer, type <<<.

5.8 EDIT STATEMENT

This statement is used to modify statements which are part of a stored program in what is termed 'edit mode'. The statement is of the form, for example:

```
EDIT  190
```

Statement 190 is then printed and the carriage returned to the left on the next line. To follow the 'edit mode' procedure, consider a pointer to each character in the original statement. This pointer begins at the first character and moves to the next character each time the SPACE key is pressed (but copying each original character as it goes).

To insert a new character, press that character instead of the SPACE key (unless a SPACE is required in which case a key marked ESC is used). To delete a character from the original line, press the DELETE (or RUBOUT) key. In this case, the input pointer will move one position to the right. The rest of the line (if syntactically correct) can then be copied by typing SPACE the required number of times.

Special care must be used to edit characters inside quotation marks in a type statement. For example, the following would not delete the last three characters:

```
160       TYPE 'THE TOTAL EQUALS<3
```

because the < and the 3 would be considered part of the message. To overcome this problem, close off the message with a single quote and, this time, delete four characters, thus

```
160       TYPE 'THE TOTAL EQUALS'<4
```

In fact, the carriage return entered after the 3 above would have been accepted without giving a line feed. Eventually, ACL would decide that this line had more than 72 characters and delete the entire line anyway (much to the surprise of the user perhaps).

5.9 LIST STATEMENT

Stored statements may be listed at a terminal in sequence number order by executing the LIST statement as an immediate statement. A single statement, a range of statements or the entire program may be listed. Examples for these three include the following:

```
LIST 300
LIST 400,900
LIST
```

If you wish to keep a copy of your program on paper tape, execute the statement:

```
LIST:
```

Since the paper tape punch is normally OFF, turn it ON and give another carriage return. Five inches of leader tape is punched, then the program and finally another length of blank trailer tape. Turn the punch OFF and tear off the paper tape.

If you do not wish to wait for the entire listing, a question mark will stop the output at the end of the current line. A question mark is used generally to interrupt the program while it is executing and to suspend program flow.

5.10 SYMBOLS STATEMENT

This statement is very similar to LIST except that its only forms are:

```
SYMBOLS   or   SYMBOLS:
```

the difference being that ':' signifies paper tape output. The values contained in the symbol table are printed out in the following form:

```
A←4.709
B(1)←9.704327E+06
B(2)←-2
X(4,11)←0
```

As in the LIST statement, the symbol table listing may be terminated by pressing question mark.

5.11 RUN STATEMENT

This statement causes the program to start executing at the lowest numbered sequence number regardless of whether execution had been previously interrupted.

5.12 PROGRAM TRACING

To assist in debugging a program, special tracing facilities are built into the ACL language. Individual statements may be traced by executing an immediate statement of the form:

```
TRON  240
```

Any number of individual statements may be marked for tracing; all may be traced by saying:

```
TRON
```

When program execution resumes, all symbol table entries are printed out (along with the originating statement if using TRON), thus greatly helping to spot errors (or 'bugs') in the program.

After having found the trouble (if any), individual trace requests may be dropped by typing:

```
TROFF  240
```

To turn all tracing off, use: TROFF.

5.13 TRANSFER OF CONTROL

ACL has two main statements for transferring control within a program, namely the GO TO statement and the IF statement.

The GO TO statement is an unconditional branch statement, which means that when executed (either as an immediate statement or as a stored statement), control will always pass to the statement whose sequence number or statement number is evaluated from the expression on the right hand side. For example, the following are all valid GO TO statements:

```
200  GO TO 1
300  GO TO 110
400  GO TO J
500  GO TO K←K+100
```

If the expression evaluates to an integer in the range 0 to 99, a statement number branch is performed; if it evaluates to an integer in the range 100 to 999, a sequence number branch is performed.

The IF statement (see appendix 5A for the full description) provides a way of conditionally branching. For example, if some looping operation was to be performed 100 times, an IF statement such as the following could be used:

```
          480  IF(J←J+1..LE.100) GO TO 2
```

Note the 'double dots' in the above syntax. The first dot is a decimal point and the second is part of the 'less than or equal to' test. To avoid these 'double dots', we could say instead:

```
          480  IF(J←1+J.LE.100) GO TO 2
```

If the relation between the two expressions is true, the right hand statement (in this case a GO TO) is executed. Otherwise the next ACL statement following the IF statement is executed.

5.14 SUBROUTINE CALLS

ACL provides subroutine calls to groups of statements considered as subroutines in the following way:

```
          140  CALL 800
```

The return address is remembered and execution then passes to the statement whose sequence number is 800. When a RETURN statement is executed, the program returns to the statement after the CALL.

Subroutines may be nested to any depth and they may all make symbol table references to any symbol. In other words, subroutines should not use the same variable names as outer level subroutines unless logically correct to do so.

5.15 SAVING ACL PROGRAMS ON IBM360 DISK STORAGE

ACL programs developed at a terminal may be saved on IBM360 disk storage and later reloaded into the NOVA computer when required. This is possible because the NOVA computer is linked to the IBM360 computer *via* the Dataway and another intermediate computer (a PDP9L) which is connected on one side to the Dataway and on the other to a channel of the IBM360 computer.

Although ACL programs may be saved on paper tape and later reloaded *via* the paper tape reader on the teletypes, this is a time-consuming process even at ten characters per second. The loading and saving time *via* the Dataway is a matter of seconds even for the largest programs containing hundreds of statements.

To save an ACL program on IBM360 disk storage, enter:

```
[#SAVE PROGNAME,INT/ACCTNMBR]
```

followed by carriage return. PROGNAME is the program name and may consist of up to eight characters provided the first letter is alphabetic (*i.e.* one of A-Z) and the rest are alphanumeric (*i.e.* one of A-Z and 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). The Summer School account number to be used is AM290060.

To avoid confusion between different Summer School 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 can be avoided. For example, if Carole Ann Stuart wished to save her program, EXP, she should type:

```
[#SAVE CASEXP,SSK/AM290060]
```

To save both the ACL program and also the current contents of the symbol table, type: SAVES instead of :SAVE as shown below:

```
[#SAVES CASDATA,SSK/AM290060]
```

In either case, the IBM360 computer response will be:

```
[-PROGNAME-SAVED AT 09.30AM ON 75.287]
```

indicating the time and day of the year on which the program was saved (if the program was being saved for the first time, or:

```
[-PROGNAME-REPLACED AT 09.30AM ON 75.287]
```

(if it was replacing an earlier version). The same area on disk is used when replacing programs, so that it does not use up the disk space to replace a program many times during its development.

5.16 RELOADING ACL PROGRAMS

To reload an ACL program from the IBM360 disk storage into the NOVA computer, enter:

```
[#LOAD PROGNAME,INT]
```

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

```
[#LOAD CASPROG1,SSK]
```

for example. Loading does not require the specification of an account number so that various users can share ACL programs. For saving, however, the account number requirement (and also the fact that each Summer School user names his programs beginning with his initials) gives protection against accidental replacement of programs on disk.

5.17 DELETION OF ACL PROGRAMS FROM DISK STORAGE

To delete programs no longer required on IBM360 disk storage, enter CNTRL/G to re-initialise the ACL work area and then enter a normal SAVE request, for example:

```
[#SAVE CASCUBIC,SSK/AM290060]
```

This 'null' program SAVE request is interpreted as a DELETE request. The response from the IBM360 computer will be either:

```
[-CASCUBIC-DELETED AT 04.30PM ON 75.287],
```

or

```
[-CASCUBIC-NOT LOCATED IN ACL LIBRARY]
```

depending on whether or not the program CASCUBIC was currently stored on disk.

5.18 CONCLUSIONS

ACL-NOVA provides a most useful interactive computing facility, enabling users to set up and test programs very simply and quickly. The extremely simple concept of syntax checking statements character by character as they are entered guards the user against trivial typing mistakes, which even large-scale computer systems seem unable to do.

With the ability to trace program flow, interrupt execution, change variables and then resume execution (from where it was interrupted or from some other statement), the user can gain valuable insight into the mathematical significance of his calculations.

The connection to the IBM360 computer also offers great time-savings in being able to SAVE and LOAD any size ACL program in a matter of seconds.

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 a programming error is to THINK.

5.19 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 Report E221. (Reissued 1972).

APPENDIX 5A

LIST OF ACL STATEMENTS

5A1 *IMMEDIATE STATEMENTS*

Arithmetic statement or expression

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

EDIT {arith stmt or exprn}

RUN

GO TO {arith stmt or exprn}

PB {arith stmt or exprn}

PA {arith stmt or exprn}

TRON [arith stmt or exprn]

TROFF [arith stmt or exprn]

FTRON

FTROFF

SPACE

CLEAR [variable[,variable]...]

SYMBOLS [:]

SUSPEND [:]

STOP

END

$$\text{TYPE}\left[\left[\begin{Bmatrix}:\\;\end{Bmatrix}\ldots\right]\text{operand}_1\left[\begin{bmatrix}\begin{Bmatrix}:\\;\end{Bmatrix}\ldots\\ ,\ \text{operand}_2\end{bmatrix}\ \text{operand}_1\right]\ldots\right]$$

where the operands take the form:

$$\left[\text{[<arith stmt or exprn>,]}\begin{Bmatrix}\text{'character string'}\\ \text{["] arith stmt or exprn}\end{Bmatrix}\right]$$

except that operand$_2$ may not be null.

5A2 *STORED STATEMENTS*

C character string (Comment only)

Arithmetic statement or expression.

GO TO {arith stmt or exprn}

ACCEPT {variable[,variable] ...}

CALL {arith stmt or exprn}

RETURN

CONTINUE

$$\text{IF(\{arith stmt or exprn\}} \left\{\begin{array}{l}\text{.EQ.}\\ \text{.NE.}\\ \text{.LT.}\\ \text{.LE.}\\ \text{.GE.}\\ \text{.GT.}\end{array}\right\} \text{\{arith stmt or exprn\})} \left\{\begin{array}{l}\text{arith stmt or exprn}\\ \text{GO TO stmt}\\ \text{ACCEPT stmt}\\ \text{CALL stmt}\\ \text{RETURN stmt}\\ \text{CONTINUE stmt}\\ \text{TYPE stmt}\\ \text{STOP stmt}\end{array}\right\}$$

TYPE (see Part 5A1)

PAUSE

STOP

APPENDIX 5B

A SAMPLE ACL PROGRAM

The following is a short ACL program designed to find the divisors of integers containing any number of digits. The trial divisors should not, however, be greater than about 16 million because loss of accuracy will then occur in the divisions.

An array D(N) is used to hold the digits of the number and the division is performed from left to right in a manner similar to a long division calculation by hand.

```
110        TYPE 'ENTER NO. OF DIGITS IN DIVIDEND.'
120        ACCEPT ND
130        TYPE ;'ENTER DIVIDEND, ONE DIGIT AT A TIME.'
140        N←Ø
150   1    N←N+1
160   2    ACCEPT D(N)
170        IF(D(N)*(D(N)-9).GT.Ø) GO TO 2
180        IF(INT(D(N)).NE.D(N)) GO TO 2
190        IF(ND←ND-1..GT.Ø) GO TO 1
200        TYPE ;'ENTER INITIAL DIVISOR AND INCREMENT.'
210        ACCEPT P,I
220        TYPE ;
230   3    J←1
240        R←Ø
250   4    R←INT(P*(Q-INT(Q←(1Ø*R+D(J))/P))+.5)
260        IF(J←J+1..LE.N) GO TO 4
270        P←P+I
280        IF(R.NE.Ø) GO TO 3
290        TYPE 'DIVISOR = ',P-I
300        GO TO 3
```

CHAPTER 6

EXPONENTIALS AND REACTORS

Lecture by

D.B. McCULLOCH

ABSTRACT

Physical processes important to the behaviour of nuclear reactors are briefly outlined, leading to a description of the 100 kW research reactor, Moata.

The simple equations for neutron-induced artificial radioactivity are derived, and applied to a Moata irradiation experiment in which a target foil is identified by measurement of the resulting radioactive half-life.

CONTENTS

6.1 INTRODUCTION

Physical phenomena, whose behaviour in terms of some basic variable such as time or distance can be described by the exponential function, are so widespread that the exponential is one of the most important and frequently used expressions in physical analysis. In simple terms, we may say that whenever an observed quantity changes by a fixed ratio in a fixed interval of time or space, regardless of where in absolute terms the time or space interval is chosen, then the variation of that observed quantity is exponential.

In the atomic energy field, the time dependance of the strength of a radioactive source, or of the power level of a nuclear reactor following an adjustment to its control system, are examples of exponential behaviour. The attenuation of a beam of radiation such as gamma rays passing through matter, or of the neutron intensity as one moves away from a neutron source in a diffusing medium such as graphite, are other examples; but here the exponential variation is with distance rather than time. The full list of exponentially varying phenomena associated with atomic energy would be almost endless.

As a practical application of the theory you will be studying on this course, you will be attempting to identify an element by determining the radioactive half-life (*i.e.* the characteristic time interval over which the induced radioactivity falls by a factor of 2) following an irradiation in the 100 kW research reactor Moata. A description of the reactor and how it is operated, and the way in which you will use it for your experiments is appropriate. We shall of course give due attention to those aspects where exponential behaviour comes into play; but first we will need to look at some basic neutron reactions with matter, and the principles of the neutron fission chain reaction on which the operation of all reactors depends.

6.2 SOME NEUTRON INTERACTIONS WITH MATTER

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).

Such neutron interactions, particularly the process known as fission, form the basis for design and operation of all nuclear reactors. Some awareness of all these mechanisms is necessary to understand the principles of the Moata reactor, which you will be meeting later in the course.

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 E1, 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 of one which decays by emission of charged particles and/or gamma rays to give a new product nuclide. The decay may be essentially instantaneous, as for example

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

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

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

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 shutdown, and (b) as fillings or coatings of detectors to monitor the neutron flux level.

Fission

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 6.1) This is the fundamental process on which all nuclear reactors depend.

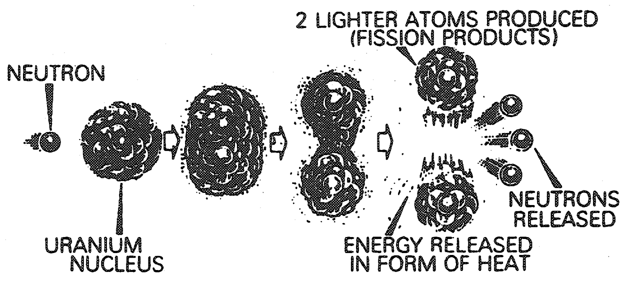

2 LIGHTER ATOMS PRODUCED (FISSION PRODUCTS)
NEUTRON
NEUTRONS RELEASED
URANIUM NUCLEUS
ENERGY RELEASED IN FORM OF HEAT

Figure 6.1 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 play an extremely important part 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 one 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 $^{238}U_{92}$ ($\sim$ 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 sec^{-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; it is known as 'enriched uranium', and is widely used as a fuel in nuclear power reactors.

6.3 FISSION CHAIN REACTIONS AND REACTORS

Because fission is induced by neutrons, and is accompanied by the release of further neutrons, the possibility of a continuing chain fission reaction exists (figure 6.2). Naturally occurring fission chain reactions have not consumed all naturally occurring uranium because processes other than fission (as described in section 6.2) compete with fission for the neutrons released by fission. By suitable design, however, the effect of these competing processes can be reduced sufficiently to allow a self-sustaining fission chain reaction to proceed.

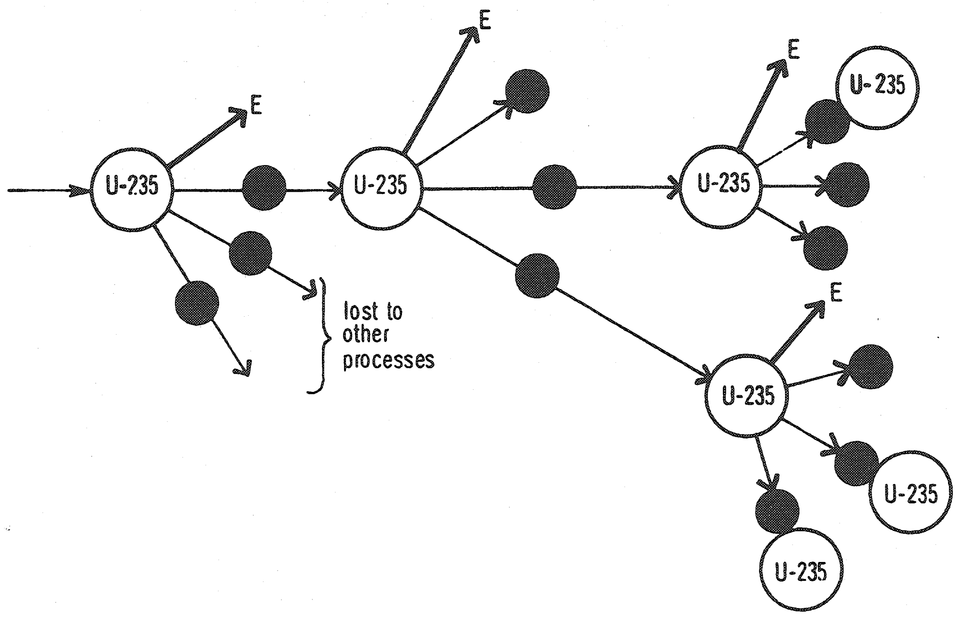


Figure 6.2 Neutron fission chain reaction

Consider now 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 6.3).

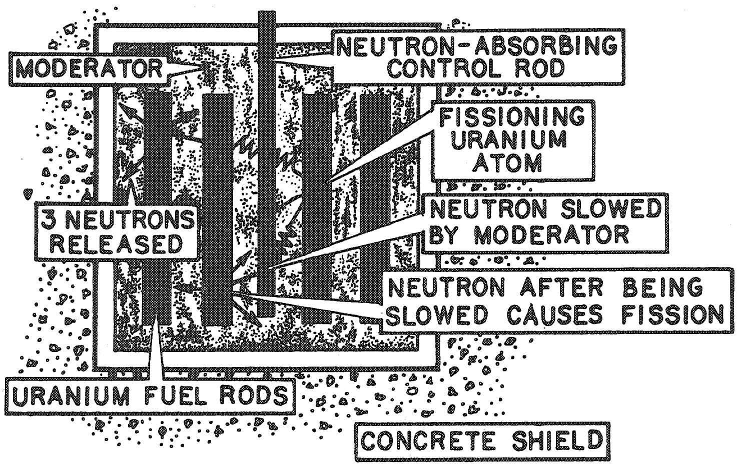

NEUTRON-ABSORBING CONTROL ROD
MODERATOR
FISSIONING URANIUM ATOM
3 NEUTRONS RELEASED
NEUTRON SLOWED BY MODERATOR
NEUTRON AFTER BEING SLOWED CAUSES FISSION
URANIUM FUEL RODS
CONCRETE SHIELD

Figure 6.3 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 ~ 0.1 to ~ 55 sec. 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 6.4).

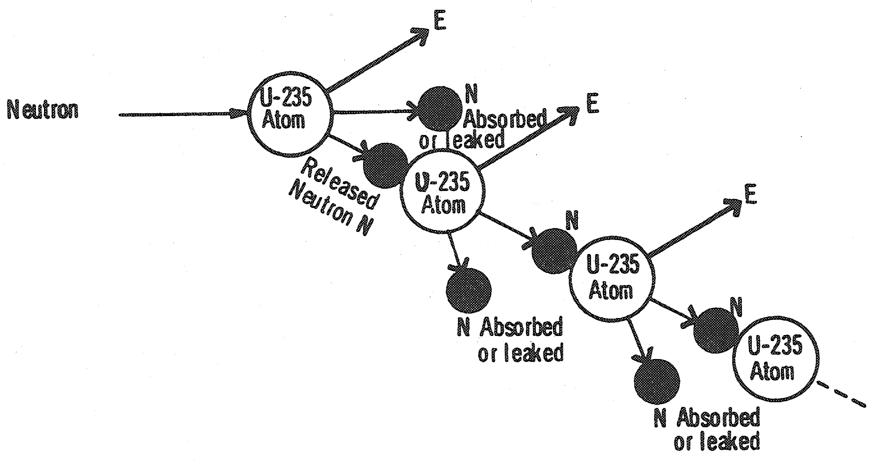


Figure 6.4 Chain reaction - critical state

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*. Since, apart from statistical fluctuations, the time interval between successive generations is constant, we have a situation where the neutron population is always changing by a fixed ratio in a given interval of time, regardless of where the time intervals are chosen; that is, the reactor neutron population, and hence the fission rate and power level, is increasing exponentially. In the r^{th} generation, $n_r = n_0k^{(r-1)}$, and 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

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

Taking logarithms (to the base e = 2.718282), we then obtain the result

$$n = n_0 e^{(\ell n\, k)t/\ell} \qquad (\ell n\, k = \log_e k) \quad ,$$

and since k normally differs only very slightly from unity, to a close approximation

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

whence

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

which is the usual way in which the exponential growth of the reactor neutron population with time is expressed. (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 '*supercritical*' (figure 6.5). 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.

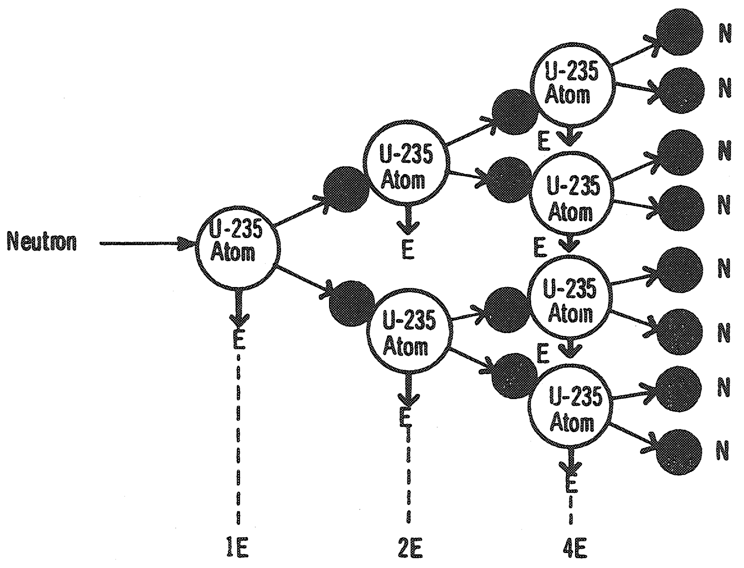


Figure 6.5 Chain reaction - supercritical state

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 subcritical 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 second, the neutron population in the subcritical 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 subcritical 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 lifetimes ranging from ~ 0.1 μsec (10^{-7} seconds) for small fast neutron reactors to ~ 1 msec (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 6.7. 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 6.6(a) and 6.6(b) 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'.

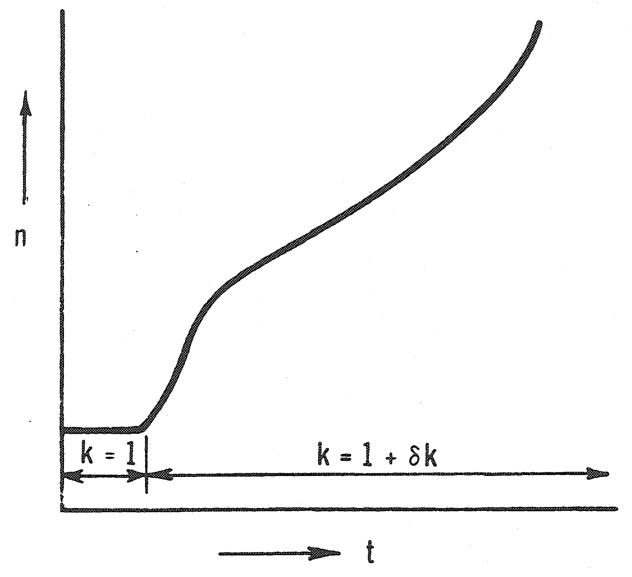


Figure 6.6(a) Reactor power response to increase in k

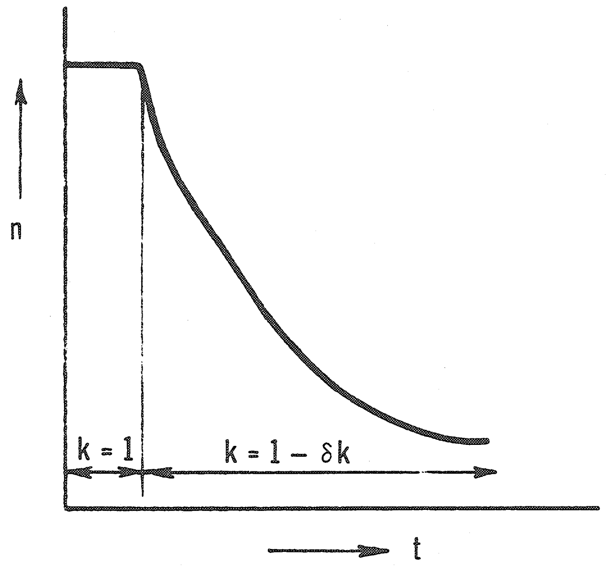
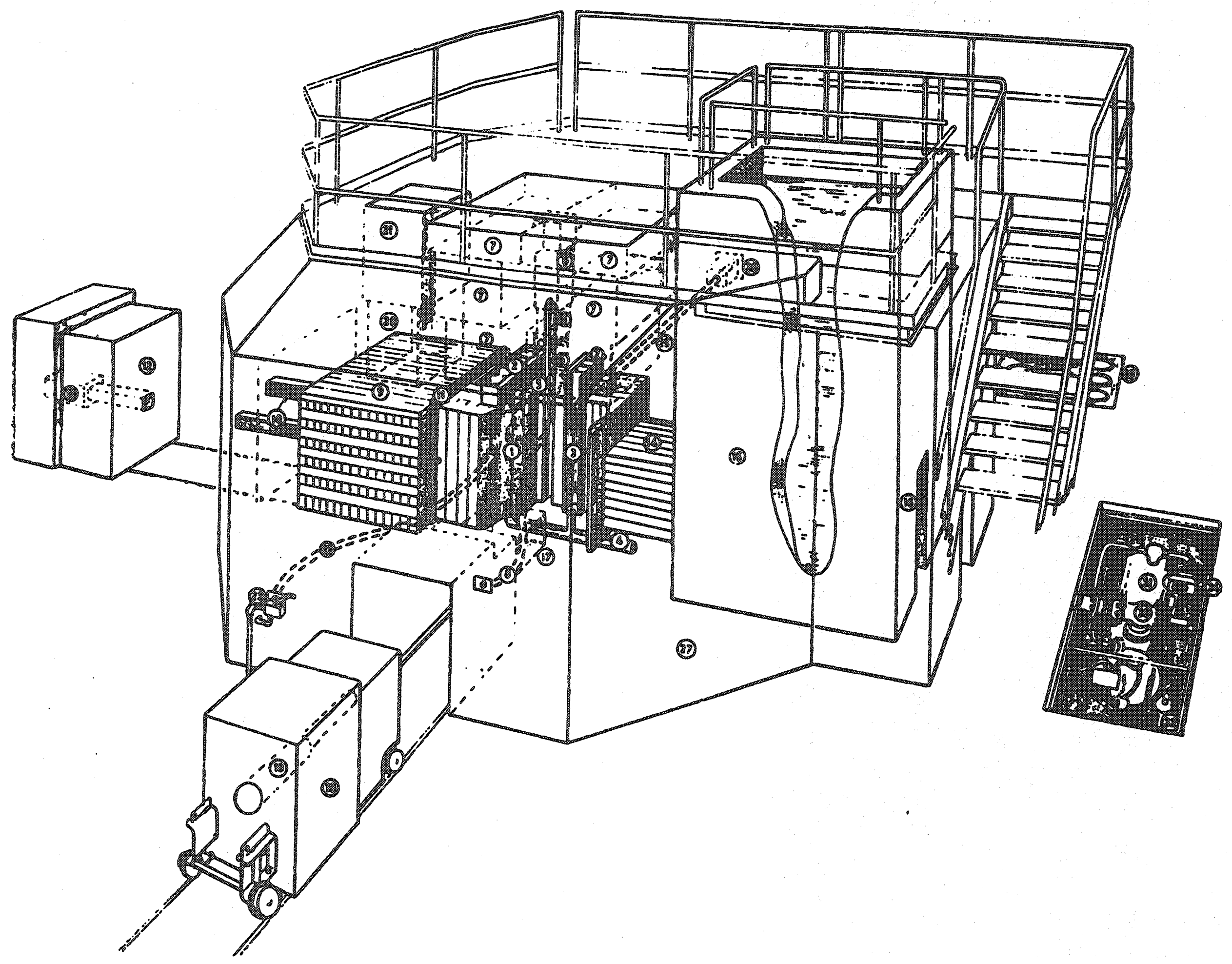


Figure 6.6(b) Reactor power response to decrease in k

6.4 THE MOATA REACTOR

Moata - Aboriginal word of North Queensland Ngerikudi tribe meaning 'firesticks' or 'gentle heat' - is basically a reactor (figure 6.7) of the Argonaut type as developed by the Argonne National Laboratory in the United States of America. The Argonaut reactor was intended 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



KEY TO DRAWING

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

Figure 6.7 The research reactor Moata

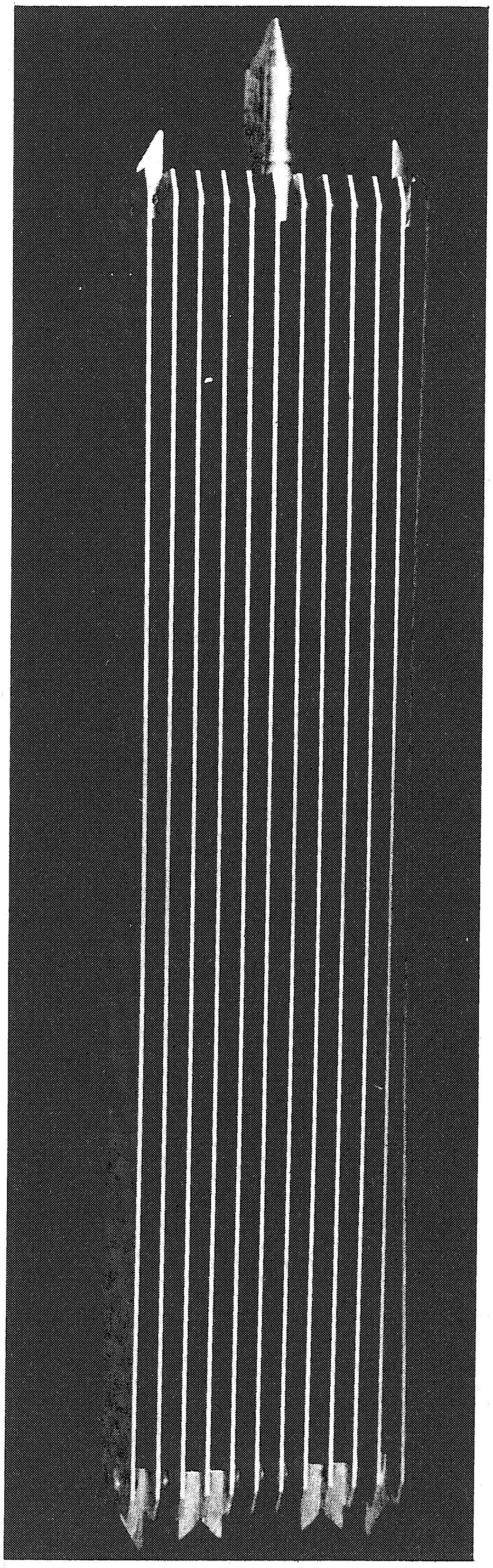


Figure 6.8 A Moata fuel element

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 inherently a very docile reactor system.

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 x 10^{12} n cm^{-2} sec^{-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 6.8. 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.

Moata offers a variety of irradiation facilities including six large cavities. Four of these are horizontally situated against 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 millirem 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 6.9.

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.

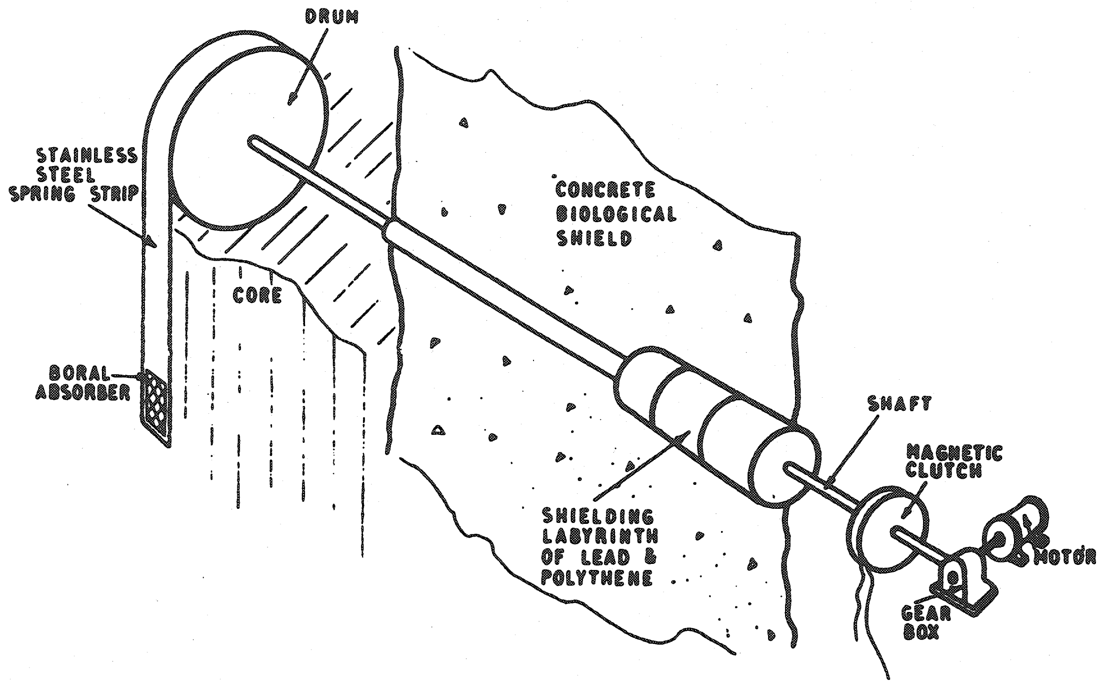

DRUM
STAINLESS STEEL SPRING STRIP
CORE
BORAL ABSORBER
CONCRETE BIOLOGICAL SHIELD
SHIELDING LABYRINTH OF LEAD & POLYTHENE
SHAFT
MAGNETIC CLUTCH
MOTOR
GEAR BOX

Figure 6.9 Diagram of the Moata control rod drive mechanisms (schematic)

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

The second category of control requirement, the measurement of reactor conditions, is achieved by five neutron detecting chambers arrayed on top of the core.

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 gamma rays arising from their interaction with some detecting nucleus. The resulting ionisation is measured as an electrical output, either as discrete pulses whose rate of arrival is proportional to the neutron flux at the detector location, or smoothed into a continuous current which is measured by a sensitive direct current monitoring device. Pulse-mode operation is usually preferred for low neutron flux levels, and current mode for high power operation. The reactions most generally favoured for reactor instrumentation are fission, in which the fission product fragments cause the electrical output or $^{10}B(n,\alpha)^{7}Li$, in which α-particles are the primary source of electrical output. Either of these reactions can be incorporated into detectors designed for operation in pulsed or current modes. Some typical detectors are illustrated schematically in figures 6.10 and 6.11.

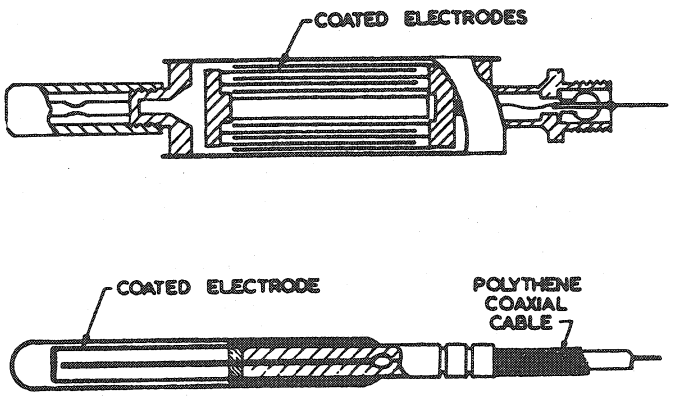


Figure 6.10 Typical pulsed-type fission chambers (schematic)

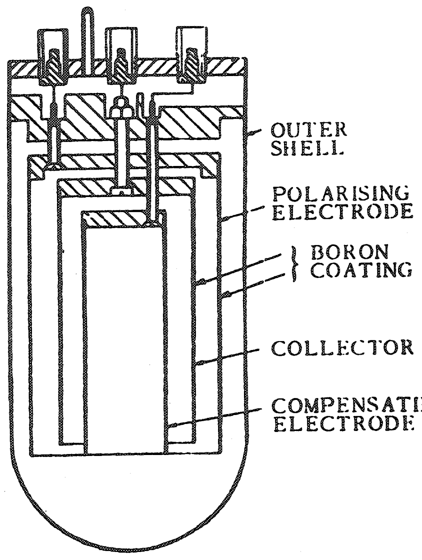


Figure 6.11 γ-Compensated current type ionisation chamber (schematic)

In Moata, 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 from shutdown 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 gamma 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 picoammeter, allowing accurate control and resetting of reactor power.

Two other 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 shutdown 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 gamma 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, with 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 able to 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 down the reactor at a chosen flux or period limit. The channels provided ensure that flux and period trips are available at any power level from the shutdown 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 startup 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 6.12) 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 startup sequence, and is carried out before raising any control absorber.

The dump valve opens rapidly on shutdown, 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 bypass 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.

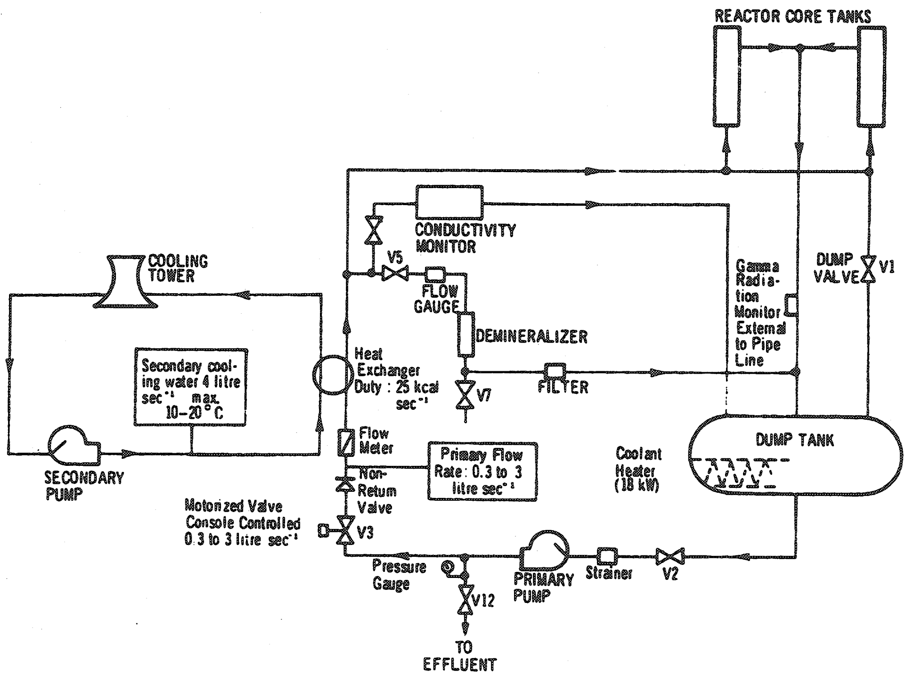

REACTOR CORE TANKS
CONDUCTIVITY MONITOR
COOLING TOWER
V5
FLOW GAUGE
DEMINERALIZER
Gamma Radiation Monitor External to Pipe Line
DUMP VALVE
V1
Secondary cooling water 4 litre sec⁻¹ max. 10–20 °C
Heat Exchanger Duty : 25 kcal sec⁻¹
V7
FILTER
SECONDARY PUMP
Flow Meter
Non-Return Valve
Primary Flow Rate: 0.3 to 3 litre sec⁻¹
Coolant Heater (18 kW)
DUMP TANK
Motorized Valve Console Controlled 0.3 to 3 litre sec⁻¹
V3
Pressure Gauge
PRIMARY PUMP
Strainer
V2
V12
TO EFFLUENT

Figure 6.12 Moata process system

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 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 Operation

Control characteristics may be described in terms of reactivity, ρ, which was shown in section 6.3 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, and is limited to bring the reactor into an 'eversafe' category. This together with its low power level and consequent small total fission product inventory, allows it to be sited in an open building without necessity for sealed shell containment.

Each day, before startup, a comprehensive schedule of tests is carried out to ensure that all instrumentation and safety equipment is operating correctly. The reactor is then taken to the required power level for the experiments to be carried out by a programmed sequence of steps, which cannot be violated without causing automatic shutdown. These steps are:

(1) Start coolant circulating pumps and close dump valve. This allows the coolant to fill the core tanks to operating level, which is maintained by continuous flow and overspill.

(2) Raise first safety rod.

(3) Raise second safety rod.

(4) Raise fine control rod to around the centre of its operating range.

(5) Withdraw coarse control rod until reactor is supercritical ($k>1$) and the neutron flux (power) is increasing with a suitable doubling time ($\sim$ 20 sec usually).

(6) Withdraw neutron source.

(7) As required power level is approached, gradually move coarse control rod inwards until just critical state ($k\equiv1$) is obtained at required power level.

(8) Switch to auto-controller, which maintains pre-set demand power by automatic adjustments to the fine control rod position.

Reactor Uses

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 determination.

(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.*

(5) Analysis of mineral ore samples for uranium content. Fissions in a small sample (a few grams) of the ore during a short irradiation (~ 1 min) at the centre of the reactor give rise to emission of delayed neutrons which can be counted (for ~ 40 sec) after the sample is removed from the reactor. The uranium content can then be determined by comparison with the count-rates obtained from standard uranium samples under similar conditions. The samples and standards are transferred by a pneumatic system under automatic control, which provides a rapid, accurate and reliable service for uranium ore assay.

6.5 A RADIOACTIVITY EXPERIMENT USING MOATA

Build up of a Radioactive Species

Some isotopes of most elements when irradiated in a neutron flux can capture a neutron to form a new isotope with a higher ratio of neutrons to protons in its nucleus. The new nucleus may be stable, in which case it is usually formed in an excited state and returns to its ground state by emission of one or more gamma rays. Alternatively, the new nucleus may be unstable, and decay by emission of charged particles (α, β^+ or β^-) to form an isotope of a different element, which again may be stable, or can undergo further radioactive decay until a stable nuclear configuration is reached.

Since all the radioactive processes which can arise in this way each have their own characteristic decay constant or half life, analysis of the general case of radiation emission following neutron irradiation of a substance can be quite complex. However, there are some cases in which the product nucleus shows a simple decay to a final stable form with a single characteristic lifetime. We shall restrict ourselves to this form of irradiation and decay for your experiment and its analysis. Some examples, with decay constants and half-lives are given in table 6.1.

TABLE 6.1

SOME EXAMPLES OF NEUTRON-INDUCED RADIOACTIVITY

Target Nucleus	Product Nucleus	Half life, τ	Decay Constant, λ (sec^{-1})
^{53}V	^{54}V	55.0 sec	1.26×10^{-2}
^{164}Dy	$^{165}Dy^m$	1.25 min	9.24×10^{-3}
^{27}Al	^{28}Al	2.30 min	5.02×10^{-3}
^{51}V	^{52}V	3.77 min	3.06×10^{-3}
^{103}Rh	$^{104}Rh^m$	4.40 min	2.63×10^{-3}
^{65}Cu	^{66}Cu	5.10 min	2.27×10^{-3}
^{59}Co	$^{60}Co^m$	10.50 min	1.10×10^{-3}
^{127}I	^{128}I	25.00 min	4.62×10^{-4}
^{115}In	$^{116}In^m$	54.00 min	2.14×10^{-4}

Consider a sample of N_0 atoms of a naturally occurring isotope with neutron capture cross section σ (cm^2), and placed in a neutron flux of ϕ ($cm^{-2}\ sec^{-1}$). Provided $\sigma\phi$ is not too large, and the irradiation is not too prolonged (as is usually the case in practice), we can neglect the change in N_0 as the irradiation proceeds, and consider the new isotope produced by neutron capture to be formed at a constant rate given by $N_0\sigma\phi$.

If the new isotope is stable, and does not itself capture neutrons, then it is easy to see that it will build up linearly with time, and the quantity present after time t(sec) will be simply $N_0\sigma\phi t$.

If however, the new isotope is unstable and has a characteristic decay constant, λ sec^{-1}, it will decay as a competing process with its formation, and consequently its build up will be slower. Radioactive decay is a statistical process, and the probable number of radioactive nuclei which will decay in an infinitesimally short time interval δt, is proportional to the number of nuclei present and the characteristic decay constant λ. If N nuclei of the new isotope are present at time t from the start of the irradiation, then in usual differential notation we now have

$$\frac{dN}{dt} = N_0\sigma\phi - \lambda N \ , \qquad \ldots(6.1)$$

where $\frac{dN}{dt}$ is the net rate of increase in the new isotope, the first term on the right hand side of the equation is the constant rate of formation by neutron capture, and the second term on the right hand side represents its loss by radioactive decay.

It is easy to see from this equation, since the terms on the right hand side are of opposite sign, that at some time after the start of irradiation dN/dt can become zero. The new isotope formation and decay rates are then just balanced, and no further increase in the quantity of the new isotope present will occur. This situation is called *saturation*.

Equation (6.1) is quite easy to solve with some rearrangement and manipulation as follows:

$$\frac{dN}{dt} + \lambda N = N_0\sigma\phi \ .$$

Now multiply both sides by $e^{\lambda t}$, whence

$$e^{\lambda t}\frac{dN}{dt} + \lambda N e^{\lambda t} = N_0\sigma\phi \ e^{\lambda t} \ .$$

Inspection of the left hand side now shows that this is just the differential with respect to time of the product $Ne^{\lambda t}$ and the equation becomes

$$\frac{d}{dt}(Ne^{\lambda t}) = N_0\sigma\phi \ e^{\lambda t} \ .$$

This is now easy to integrate, and we have

$$Ne^{\lambda t} = \frac{N_0\sigma\phi}{\lambda} \ e^{\lambda t} + \text{const.}$$

which, by rearrangement, gives

$$N = \frac{N_0\sigma\phi}{\lambda} + Ce^{-\lambda t} \ . \qquad \ldots(6.2)$$

As is usual in integration problems, the constant C is arbitrary until defined by some boundary condition. In this case, we know that at commencement of the irradiation, the number of new nuclei present is zero, *i.e.* N = 0 at t = 0. Substitution in equation (6.2) thus gives

$$C = -\frac{N_0\sigma\phi}{\lambda} \quad ,$$

and

$$N = \frac{N_0\sigma\phi}{\lambda} \quad (1-e^{-\lambda t}) \qquad ...(6.3)$$

which is the standard equation for the buildup of nuclei of a single radioactive decay species in a reactor irradiation. The disintegration rate D, immediately on removal from the reactor, is given simply by λN, or

$$D = N_0\sigma\phi \ (1-e^{-\lambda t}) \ \text{sec}^{-1} \qquad ...(6.4)$$

In practice, activities are usually expressed in curies per gram of starting material, whence

$$S = \frac{0.6 \ \sigma\phi}{3.7\text{x}10^{10}A} \quad (1-e^{-\lambda t}) \quad \text{Ci g}^{-1} \quad , \qquad ...(6.5)$$

where 1 Ci is a disintegration rate of 3.7 x 10^{10} per second,
A is the atomic weight, and
σ is now expressed in barns (10^{-24} cm^2).

Alternatively, in terms of half-life ($T_{\frac{1}{2}}$) rather than λ,

$$S = \frac{0.6 \ \phi\sigma}{3.7\text{x}10^{10}A} \quad (1-e^{\frac{-0.691t}{T_{\frac{1}{2}}}}) \qquad ...(6.6)$$

Reference to exponential tables shows that for

$t/T_{\frac{1}{2}}$ =	1	2	3	4	5
$(1-e^{\frac{-0.691t}{T_{\frac{1}{2}}}})$ =	0.5	0.75	0.87	0.94	0.97

This indicates that 75% of the saturation activity is obtained after an irradiation of two half-lives, and 87% after three. Subsequent buildup is very slow and, in consequence, there is usually little point in irradiations lasting more than two to three half-lives of the wanted product isotope.

The Moata Experiment

A small sample of a suitable element will be placed in the Moata 'rabbit' facility (section 6.4) and pneumatically transferred to the core region. After a short irradiation at a power level chosen to give suitable activity for handling and counting, the sample will be fired out and transferred to a shielded counting facility for measurement of its induced radioactivity.

The total counts obtained in successive short time intervals, τ, will be displayed and recorded on paper tape. After following the radioactive decay for a few half-lives, you will then process the count data, and analyse them by the least squares fitting method as described in chapter 2, to derive a value for the half-life of the radioactive species produced in the irradiation. Each group will have a different material and should be able to identify the starting substance from the list of reactions given in table 6.1.

6.6 FURTHER READING SUGGESTIONS

(1) Any modern text-book available to you on nuclear physics at an introductory level. Concentrate particularly on chapters dealing with interactions of neutrons, gamma 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-IV (Van Nostrand), or
Glasstone, S. & Sesonke, A. Nuclear Reactor Engineering, Chapters I-V (Van Nostrand Reinhold Co).

CHAPTER 7

'e' IN CHEMISTRY

Lecture by

A.J. EKSTROM

ABSTRACT

The use of 'e' in chemistry is illustrated by the derivation of rate laws for a unimolecular decomposition reaction and a bimolecular, second order, oxidation-reduction reaction. Particular features of the two resulting expressions are briefly discussed.

CONTENTS

7.1 INTRODUCTION

By the time you read this chapter, you will have some appreciation of the mathematical significance of e and the methods used to determine its value of 2.71828.... It remains, therefore, to show how e appears, quite naturally, in various areas of scientific specialisation which are dependent on simple mathematics for the analysis and understanding of the results obtained by the experimenter in his laboratory.

7.2 A FIRST ORDER RATE LAW IN CHEMICAL KINETICS

Perhaps the best way to illustrate the importance and usefulness of e in chemistry is to consider the field of chemical kinetics; this is an area of specialisation concerning itself with the study of the rates (how fast) and mechanisms of chemical reactions. For example, the compound hydrogen peroxide (H_2O_2) is chemically unstable, decomposing on

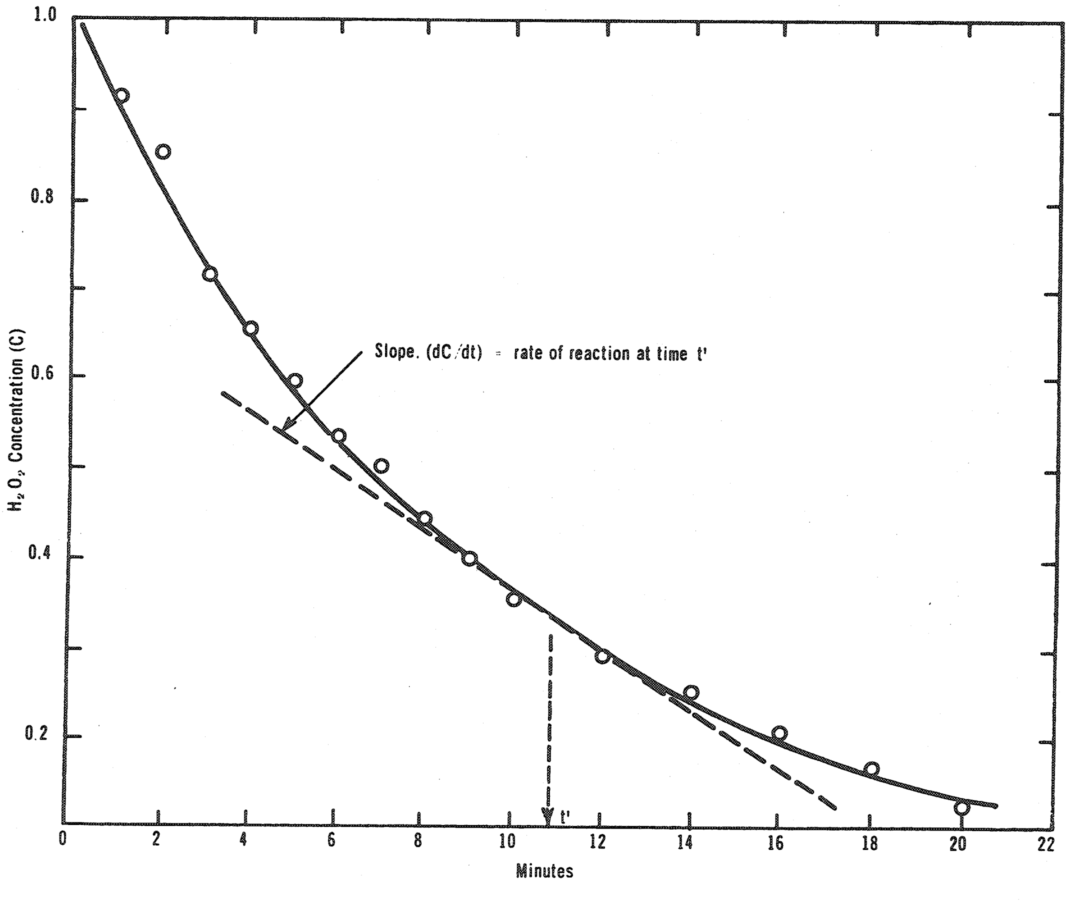


Figure 7.1 Hydrogen peroxide concentration as a function of time

standing to form, eventually, oxygen and water. If we conducted an experiment in which we measured the hydrogen peroxide concentration as a function of time, we would obtain results similar to those illustrated in figure 7.1. You can see that the *rate* of reaction, represented by the slope of the curve (d[C]/dt) at any time t, decreases with time, and is in fact proportional to the concentration ([C]) of hydrogen peroxide at any time (table 7.1). We can thus write the mathematical expression

$$- \frac{d[C]}{dt} \propto [C] \qquad \ldots(7.1)$$

or

$$- \frac{d[C]}{dt} = k\,[C] \quad , \qquad \ldots(7.2)$$

where d[C]/dt represents again the rate of change of the hydrogen peroxide concentration at any time t, [C] represents the hydrogen peroxide concentration, and k is the proportionality constant, commonly called the *rate constant*.

TABLE 7.1

TYPICAL EXPERIMENTAL RESULTS FOR THE DECOMPOSITION OF HYDROGEN PEROXIDE

Time (Minutes)	Hydrogen Peroxide Concentration (C) (moles per litre)	d[C]/dt (M min^{-1})	(d[C]/dt)/C (min^{-1})
0	1.0	0.110	0.110
1	0.93	0.095	0.102
2	0.85	0.083	0.098
3	0.71	0.073	0.102
4	0.66	0.064	0.097
5	0.59	0.056	0.095
6	0.53	0.050	0.094
7	0.50	0.050	0.100
8	0.44	0.047	1.107
9	0.40	0.043	0.108
10	0.35	0.038	0.108
12	0.29	0.026	0.090
14	0.25	0.028	0.112
16	0.21	0.021	0.100
18	0.17	0.017	0.100
20	0.12	0.010	0.083
70	0.001	0.0001	0.100

Equation (7.2) is a simple differential equation which can be slightly reorganised and integrated. Thus:

$$-\frac{d[C]}{[C]} = k\ dt \qquad \ldots(7.3)$$

and

$$-\int\frac{d[C]}{[C]} = \int k\ dt \quad . \qquad \ldots(7.4)$$

Recalling that

$$\int\frac{dx}{x} = \ell n\ x \quad , \qquad \ldots(7.5)$$

equation 7.4 becomes

$$\ell n\ C = -kt + a \quad , \qquad \ldots(7.6)$$

where a is a constant. If we now assume that at t = 0, *i.e.* at the start of the reaction, $C = C_0$ then we can readily show that

$$a = \ell n\ C_0 \qquad \ldots(7.7)$$

and hence that

$$\ell n \left\{\frac{[C]}{[C_0]}\right\} = -kt \quad . \qquad \ldots(7.8)$$

At this point, it is useful to recall the simple relationship

$$\log_{10}100 = 2$$

or

$$10^2 = 100$$

and that, by analogy, equation (7.8) can be written as:

$$\frac{[C]}{[C_0]} = e^{-kt} \qquad \ldots(7.9)$$

or

$$[C] = [C_0]\ e^{-kt} \quad . \qquad \ldots(7.10)$$

Thus, we have shown that the concentration of hydrogen peroxide, [C], is given by the product of the initial concentration $[C_0]$ and the exponential term e^{-kt}. We have thus arrived at a simple exponential decay law from an experimental observation and some elementary mathematical operations.

You might note that equation (7.8) predicts that a plot of $\ell n \frac{[C]}{[C_0]}$ against t should be a straight line of slope -k. You may wish to convince yourself of this fact using the data summarised in table 7.1.

A special situation arises when $[C] = 0.5\ [C_0]$, *i.e.* when exactly one half of the initial hydrogen peroxide concentration has decomposed. In this case, equation (7.8) becomes

$$\frac{\ln\ [0.5\ C_0]}{[C_0]} = -\ kt_{0.5} \qquad \ldots(7.11)$$

or $$t_{0.5} = \frac{\ln 2}{k} \qquad \ldots(7.12)$$

i.e. $$t_{0.5} = \frac{0.69}{k}\ . \qquad \ldots(7.13)$$

The time $t_{0.5}$ is often termed the halflife of the reaction and, according to equation 7.13, is seen to be a constant for a particular reaction which is independent of the initial concentration C_0.

It may be noted that equations similar to (7.8) and hence (7.10) above, are found to describe a wide variety of physical processes amongst which the rate of decay of radioactive elements, the absorption of light by absorbers and the variation of atmospheric pressure with altitude are only a few examples. You may wish to examine a textbook of Physical Chemistry in the library for further examples.

7.3 A SECOND ORDER RATE LAW

Returning now to our examination of chemical kinetics, the simple rate law (equation 7.1) is, in fact, rarely observed since most chemical reactions involve at least two reacting species as for example, the oxidation of iron (II) ions by cobalt (III) ions

$$\mathrm{Fe(II)} + \mathrm{Co(III)} \rightarrow \mathrm{Fe(III)} + \mathrm{Co(II)}\ . \qquad \ldots(7.14)$$

For convenience, we will call the initial iron (II) concentration $[A_0]$, the initial cobalt (III) concentration $[B_0]$ and the concentration at any time of one of the products, *e.g.* of iron (III), is x. We thus obtain the following boundary conditions:

	Fe(II) +	Co(III) →	Fe(III) +	Co(III)
at t = o	A_0	B_0	0	0
at t = t	(A_0-x)	(B_0-x)	x	x

When we attempt to measure the rate of appearance of the Fe(III) in this reaction, we find that the rate is proportional to the product of the concentrations of Fe(II) and Co(III), *i.e.*,

$$\frac{dx}{dt} = k[A_0-x][B_0-x] \qquad \ldots(7.15)$$

and hence (by employing partial fractions and integration) we have

$$kt = \frac{1}{A_0-B_0} \ln \frac{[B_0][A_0-x]}{[A_0][B_0-x]} , \qquad \ldots(7.16)$$

or, in exponential form, that

$$x = \left[\frac{1 - \exp(kt(A_0-B_0))}{1 - \alpha\exp(kt(A_0-B_0))}\right] A_0 \qquad \ldots(7.17)$$

where $\alpha = A_0/B_0$

Rate laws similar to those given by equation (7.1) are frequently termed 'first order' rate laws because they contain only a first order concentration dependency; on the other hand, rate laws similar to (7.15) are naturally termed 'second order' since they contain the product of two concentration terms.

Two special cases of a second order rate law arise. Firstly, we may arrange our experimental conditions such that at the start of the reaction the concentration of A is very much larger than that of B, *i.e.* $A_0 >> B_0$. Under these circumstances, the term (A_0-x) in equation (7.15) will effectively remain constant at the value A_0 during the reaction and hence it can be incorporated into the proportionality constant k. The reaction can then be described mathematically by an equation similar to (7.1) above and, under these circumstances, the reaction is commonly termed 'pseudo-first-order'.

The second special case arises if we make the initial concentrations of A and B equal, *i.e.* if $A_0 = B_0$. In that case, equation (7.15) becomes

$$\frac{dx}{dt} = k[A_0-x]^2 \qquad \ldots(7.18)$$

which, on integration, gives

$$kt = \left[\frac{1}{A_0} \quad \frac{x}{A_0-x}\right] . \qquad \ldots(7.19)$$

You may note that, in this case, an expression is obtained which does not contain an exponential term.

7.4 <u>THE INCENTIVE FOR STUDYING CHEMICAL KINETICS</u>

In conclusion it is perhaps of interest to say something about the reasons for the study of chemical kinetics. Firstly, the kinetics of reactions provides information on the mechanisms of reactions and this helps us to understand exactly how chemical transformations occur. Secondly, as you will probably know already, the whole chemical industry is based on a very wide variety of chemical reactions and, since time is money, there are real incentives to find the conditions under which chemical reactions occur as rapidly and as efficiently as possible.

We hope to discuss some examples of these aspects during the lectures which you will attend during your visit to the Australian Atomic Energy Commission.

CHAPTER 8

SURVIVAL CURVES FOR IRRADIATED CELLS

Lecture by

D.K. GIBSON

ABSTRACT

The subject of the lecture is the probability of survival of biological cells which have been subjected to ionising radiation. The basic mathematical theories of cell survival as a function of radiation dose are developed. A brief comparison with observed survival curves is made.

CONTENTS

8.1 INTRODUCTION

In this lecture I am going to present some of the mathematical descriptions of radiation damage to living cells. The exponential function will be seen to appear often enough to justify the inclusion of this topic under the general title of this Summer School.

8.2 DOSE

For the purposes of this lecture we need not be too concerned by the measurement of radiation dose nor the units used. For instance, we could well think of measuring dose by, say, minutes of exposure to a constant intensity X-ray flux. However, for comparing radiations of different types and from different sources, the unit 'rad' is used. When we say that a certain piece of material has received a dose of R rads, we mean that 100R ergs of energy have been absorbed by each gram of the material. This unit was developed because it seemed that it was the absorbed energy density which determined the amount of radiation damage.

Although this is roughly true for many types of radiation, it is very inaccurate for others. For example, slow neutrons, which interact through the displacement of nuclei, prove to be far more damaging than X-rays, which interact mainly through ionisation, for the same amount of energy deposited. For this reason another unit is often used; this unit is called the 'rem' and it is directly proportional to the biological damage produced. The rem and the rad are related by an empirical factor called the quality factor. One particular type of radiation has been chosen arbitrarily to have a quality factor of 1. 250 kV X-rays were chosen as the normalising standard, probably because many experiments had been carried out using hospital X-ray machines.

8.3 CELL SURVIVAL EXPERIMENTS

Before we delve into the mathematics of radiation damage, we should look at what sort of experiments the theories are attempting to explain. In principle, the experiments are as follows.

A certain type of cell is grown in a culture medium which provides the right nutrients and conditions for growth and reproduction. Some of the cells are drawn off and exposed to a measured amount of radiation. They are then 'plated out' onto a Petri dish, well provided with growth medium, and given time to reproduce. After a certain time, those cells which are sufficiently unaffected by the radiation will undergo five or more divisions. Cells that do not undergo divisions are considered dead; in actual fact it is 'reproductive death' that is measured, as some cells may be alive but unable to divide.

By dividing five or more times, the cells have proved that they are alive and have also formed colonies large enough to be seen easily under a microscope. The colonies are counted and the number compared with a control sample of unirradiated cells. By subjecting batches of samples to varying amounts of radiation, a 'survival curve' can be built up, that is, a plot of the fraction of surviving cells as a function of radiation dose.

In this type of experiment there are many possible parameters; these include the type of cell, the type of radiation, the dose rate, the degree of oxygenation of the cells, and the addition of radiation protective chemicals.

It is the general form of the cell survival curves that we are going to look at mathematically.

8.4 EXPONENTIAL SURVIVAL

If we have N cells, each with a probability pdR of being killed by a radiation dose dR, we may write:

$$dN = -N\,pdR$$

$$\frac{dN}{N} = -\,pdR$$

which gives

$$\ell n\,\frac{N}{N_0} = -\,pR\quad ,$$

where N_0 is the original value of N, at R = 0.

Therefore we can write for the surviving fraction, S,

$$S = \frac{N}{N_0} = e^{-pR}\quad . \qquad \ldots(8.1)$$

This is simple exponential loss of particles, arising because dN is proportional to N. A cell cannot be killed more than once, just as a radioactive nucleus cannot decay more than once.

The applicability of equation (8.1) to real cell survival curves will be discussed, together with other models, in section 8.8. Suffice it to say here that although some cells exhibit a straight exponential survival curve others, particularly mammalian cells, do not. Hence we must look at some more complex models.

We can think of equation (8.1) in a slightly different way, which will be useful later on. The surviving fraction is the same as the probability of any one cell surviving a radiation dose R. We can write for the probabilities of a cell surviving and being killed, respectively:

$$P_s(R) = e^{-pR} ,$$

$$P_k(R) = 1-e^{-pR} . \qquad ...(8.1a)$$

8.5 SOME PROBABILITY THEORIES

As the following theories of cell killing depend on the probabilities of various arrangements of targets being hit by small quanta of radiation, behaving like projectiles, we will first develop the necessary formulae.

The Binomial Distribution

Think of a number of targets, each with a probability, p, of being hit. On the average we can say that each target will be hit a certain number of times but, in reality, some targets will not be hit at all, whereas others will suffer multiple hits. We want to find the chances of a target being hit, 0, 1, 2 ... times.

The problem is best understood by looking at a simple case. Let us think of three shots being fired at four targets. The chance of any one shot hitting any one target is $P_H = 1/4$. The probability of the shot missing the target is $P_M = 1 - P_H = 3/4$. The probability of a particular target being hit three times is:

$$P_3 = (P_H)^3 = \frac{1}{64} .$$

To obtain the probability of our chosen target being hit twice only (P_2), we must sum three probabilities:

$$\begin{aligned} P_2 &= P_H . P_H P_M + P_H P_M P_H + P_M P_H P_H , \\ &= 3 P_H P_H P_M = 3 \times (\tfrac{1}{4})^2 \times \tfrac{3}{4} = \tfrac{9}{64} . \end{aligned}$$

Hence, the probability that our target will be hit twice only is three times the probability of receiving two hits followed by a miss. The factor of three obviously accounts for the fact that the order of the hits and misses is unimportant, and there are three ways of arranging two objects (hits) in three boxes (shots).

We can generalise and write that the probability of one target being struck h times out of n shots, with the probability per shot being p, is

$$P(h) = p^h (1-p)^{n-h} \frac{n!}{h!(n-h)!} \qquad ...(8.2)$$

where the number of ways of arranging h things in n boxes is given by the binomial coefficient or combinatorial factor n!/(h! (n-h)!).

	n!	= product of integers 1 to n,
except	0!	is taken to be 1,
e.g.	4!	= 1.2.3.4 = 24

The Poisson Distribution

We can consider the cases where the number of shots fired is very large, but the probability of any one target being hit by any one shot is extremely small. That is, n is large and p is small, but

$$\lambda = np$$

the average number of hits per target is of moderate magnitude.

We will see what happens to P(h) as $n \to \infty$ while λ remains constant. Firstly the probability of no hits is

$$\begin{aligned} P(0) &= (1-p)^n \quad , \\ &= (1-\frac{\lambda}{n})^n \quad , \\ &= 1 - n\frac{\lambda}{n} + \frac{n(n-1)}{2!}\frac{\lambda^2}{n^2} - \frac{n(n-1)(n-2)}{3!}\frac{\lambda^1}{h^3} + \quad \ldots \quad , \\ &= 1 - \lambda + \frac{n-1}{n}\frac{\lambda^2}{2!} - \frac{(n-1)(n-2)}{n^2}\frac{\lambda^3}{3!} + \quad \ldots \quad , \\ &\to 1 - \lambda + \frac{\lambda^2}{2!} - \frac{\lambda^3}{3!} \text{ as } n \to \infty \quad , \\ &= e^{-\lambda} \quad . \end{aligned}$$

Now consider the ratio of two adjacent P(h)'s:

$$\begin{aligned} \frac{P(h)}{P(h-1)} &= \frac{p^n(1-p)^{n-h}}{p^{n-1}(1-p)^{n-h+1}} \; \frac{n!}{h!(n-h)!} \; \frac{(h-1)!(n-h+1)!}{n!} \quad , \\ &= \frac{p}{1-p} \; \frac{n-h+1}{h} \quad , \\ &= \frac{\lambda}{1-\frac{\lambda}{n}} \; \frac{1+\frac{1-h}{n}}{h} \\ &\to \frac{\lambda}{h} \text{ as } n \to \infty \quad . \end{aligned}$$

Therefore, knowing P(0), we can calculate P(1), *etc* .

$$P(0) = e^{-\lambda} \quad ,$$

$$P(1) = \frac{\lambda}{1} e^{-\lambda} = \lambda\, e^{-\lambda} \quad ,$$

$$P(2) = \frac{\lambda}{2} \lambda\, e^{-\lambda} = \frac{\lambda^2}{2} e^{-\lambda} \quad ,$$

$$P(3) = \frac{\lambda}{3} \frac{\lambda^2}{2} e^{-\lambda} = \frac{\lambda^3}{3.2.} e^{-\lambda} \quad ,$$

and clearly

$$P(h) = \frac{\lambda^h}{h!} e^{-\lambda} \quad . \qquad \ldots(8.3)$$

This is called the Poisson distribution and it gives the probability of recording 0, 1, 2, ... , h, ... hits, assuming a very large number of shots, each with a small probability of success.

It is clear the total probability

$$\begin{aligned}\sum_{n=0}^{\infty} P(h) &= e^{-\lambda} + \lambda e^{-\lambda} + \frac{\lambda^2}{2!} e^{-\lambda} + \ldots \quad , \\ &= e^{-\lambda}(1 + \lambda + \frac{\lambda^2}{2!} + \ldots) \quad , \\ &= e^{-\lambda}\, e^{\lambda} \quad , \\ &= 1 \quad ,\end{aligned}$$

and this is as it should be.

8.6 MULTI-HIT THEORY OF RADIATION DAMAGE

In the following discussions we must think of the absorption of radiation as a series of discrete active events. Not all the radiation will be dissipated in active events, but a given dose can be correlated with so many active events per unit mass. An active event corresponds to the target hits referred to in section 8.5.

Let us now assume that each cell contains a vital region, or target, which must be hit a given number of times in order to inactivate it and lead to cell death. The probability that any one target will be struck by any one quantum of dose is constant and very small, but there are a large number of dose quanta. We may therefore apply the Poisson formula (equation 8.3) to find the probability of any one target receiving h hits.

If we take for the average number of hits $\lambda = pR$, and assume that cells with more than ν hits are killed, we can write for the survival curve

$$S(R,\nu) = \sum_{h=0}^{\nu} \frac{(pR)^h e^{-pR}}{h!} \quad ,$$

$$= e^{-pR} \sum_{h=0}^{\nu} \frac{(pR)^h}{h!} \quad . \qquad \ldots(8.4)$$

Clearly if $\nu = 0$ (*i.e.* if one hit is enough to kill the cell)

$$S = e^{-pR} \quad ,$$

the case of exponential survival derived in section 8.4.

Another limiting case is $\nu = \infty$, the case of indestructible cells, where we have

$$S = e^{-pR} \sum_{h=0}^{\infty} \frac{(pR)^h}{h!} \quad ,$$

$$= e^{-pR}\, e^{pR} \quad ,$$

$$= 1 \quad .$$

8.7 MULTI-TARGET SINGLE-HIT THEORY

Another possibility is that the cell contains t targets, b of which must be hit once in order for the cell to die. As an analogy one could think of an insect initially with six legs, which could stagger along with reasonable success with five, four, three legs, but with two, one or no legs would fall over and starve.

In this model, once a target is hit it is unaffected by further hits. In this case, the probability of hitting a target is not constant but falls off exponentially, as given by equation (8.1a), because there are fewer targets left as irradiation proceeds. Using this variable probability in equation 8.2 we find that the probability of b targets being hit is

$$P(b) = \left(1 - e^{-pR}\right)^b \left(e^{-pR}\right)^{t-b} \frac{t!}{b!\,(t-b)!} \quad .$$

If all cells that have more than ν targets hit die, we have a survival curve

$$S(R,\nu,t) = \sum_{b=0}^{\nu} \frac{t!}{b!\,(t-b)!} \left(1-e^{-pR}\right)^b \left(e^{-pR}\right)^{t-b} \quad .$$

$$\ldots(8.5)$$

In a commonly used form of equation (8.5), it is assumed that all t targets must be inactivated to kill the cell. That is $\nu = t - 1$, and

$$S(R,t) = \sum_{b=0}^{t-1} \frac{t!}{b!(t-b)!} \left(1-e^{-pR}\right)^{t} \left(e^{-pR}\right)^{t-b} ,$$

$$= \sum_{b=0}^{t} \frac{t!}{b!(t-b)!} \left(1-e^{-pR}\right)^{t} \left(e^{-pR}\right)^{t-b} - \frac{t!}{t!0!} e^{0} \left(1-e^{-pR}\right)^{t} ,$$

$$= \left(1-e^{-pR} + e^{-pR}\right)^{t} - \left(1-e^{-pR}\right)^{t} ,$$

$$= 1 - \left(1-e^{-pR}\right)^{t} . \qquad \text{...(8.5a)}$$

The first term in equation (8.5) (*i.e.* $b = 0$) is

$$S = e^{-pRt} ,$$

which is the survival curve if only one inactivated target leads to cell death. Also in equation (8.5a), if the number of targets available and necessarily inactivated for death approaches infinity (*i.e.* $t \to \infty$) we get again $S = 1$.

8.8 DISCUSSION OF SURVIVAL CURVES

The three types of survival curves are shown in figures 8.1, 8.2 and 8.3.

Pure exponential survival (figure 8.1) is found experimentally in the case of simple organisms such as viruses and bacteria. Mammalian cells irradiated with X-rays produce curves with a shoulder very much like the curves in figures 8.2 and 8.3. The experimental data is not good enough to distinguish between the two target theories.

If the survival curve has a straight portion, it can be extrapolated back to the abscissa and the value of the intercept is called the extrapolation number. The extrapolation number is often taken to give an idea of the number of targets which must be inactivated. This can be seen in the case of equation (8.5a), where we have t targets all of which must be inactivated to produce death. We have

$$S = 1 - \left(1-e^{-pR}\right)^{t}$$

and, since in the asymptotic region pR is large, we may write

$$S = 1 - \left(1-te^{-pR}\right) ,$$

$$= te^{-pR} ,$$

$$\therefore \ln S = \ln t - pR .$$

Thus equation (8.5a) approaches a straight line plot (on a log-linear graph), with slope $-p$ and extrapolation number t.

Similarly it can be shown that the more general equation (8.5) will extrapolate back to

$$\frac{t!}{n!(t-n)!} \quad .$$

However, the approach to the linear form can be so slow that, in practice, no meaningful data can be obtained from an extrapolation number. This is illustrated in figure 8.3 where the curve

$$S = \sum_{b=0}^{3} \frac{10!}{b!(10-b)!} \left(1-e^{pR}\right)^{b} \left(e^{-pR}\right)^{10-b}$$

has been calculated and plotted over 20 orders of magnitude. The extrapolation numbers calculated from pairs of points are shown along the curve between the two points used for their calculation. It can be seen that the apparent extrapolation numbers slowly approach the analytical value of 120. However, at survival values of about 10^{-6}, below which it is not very practical to go in real experiments, the apparent extrapolation number is about 20, bearing no relation to the expected value. The value of 10 shown for the same curve plotted over a smaller range in figure 8.2 looks convincing but it has no meaning at all.

It is reasonable to expect that simple cells such as bacteria would be inactivated by the loss of one target, whereas in a more complex cell, the function of one target would to some extent be duplicated by another potential target.

For radiation such as α-particles, which deposit energy very densely along their tracks, mammalian cells exhibit straight exponential survival with no shoulder. It is believed, in this case, that the density of 'hits' for this type of radiation is so large that, in the event of a hit, all targets in a cell will be struck together.

It is obviously attractive to think that the DNA molecules might be the vital target or targets. There is quite a lot of evidence to support this idea. However, the true picture is bound to be complicated by the variety of other mechanisms for radiation damage of cells. One important factor is the production of active chemical radicals which cause further chemical changes to the cell.

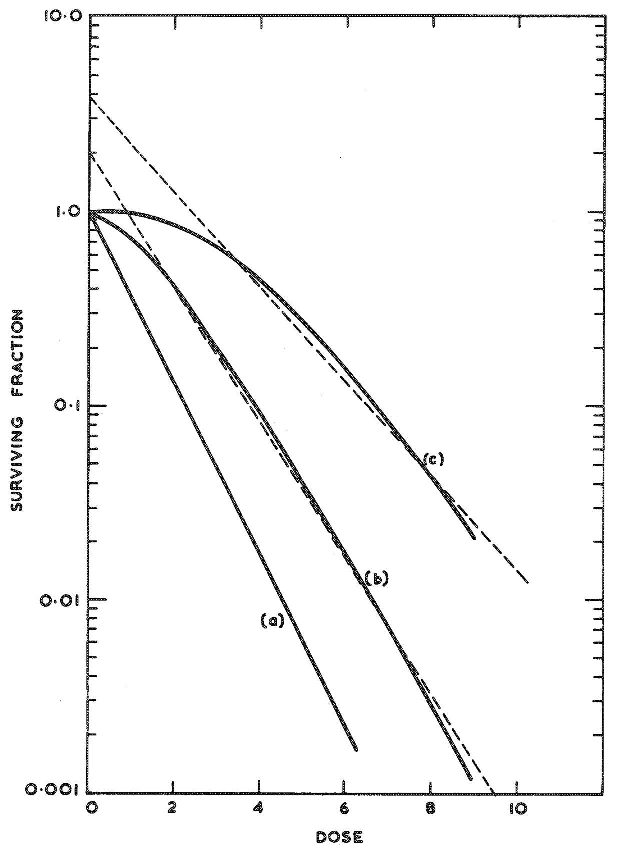


Figure 8.1 Theoretical survival curves:
(a) Pure exponential survival.
(b) Single-target multi-hit model with v=3, equation 8.4 (i.e. two hits sufficient for inactivation). (c) Single-target multi-hit model with v=3 (i.e. four hits necessary for inactivation). Note that the straight lines drawn through (b) and (c) have been put in to clarify the continuously curving nature of the survival curves. Their extrapolation number has no significance.

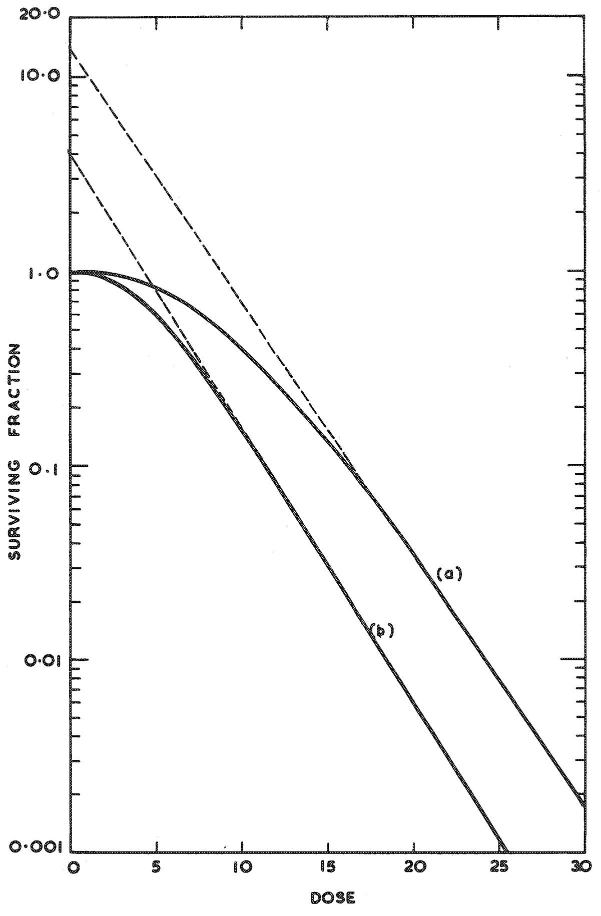


Figure 8.2 Theoretical survival curves: Two curves on the multi-target single-hit model.
(a) t=10; v=3, equation 8.5. Out of ten possible targets in each cell, at least four must be hit for inactivation of cell. This has an apparent extrapolation number of 14, but this is completely spurious as there is insufficient range in this graph. (see figure 8.3)
(b) t=4; v=3, equation 8.5a. All four possible targets must be hit. The extrapolation number is the correct one, i.e. 4).

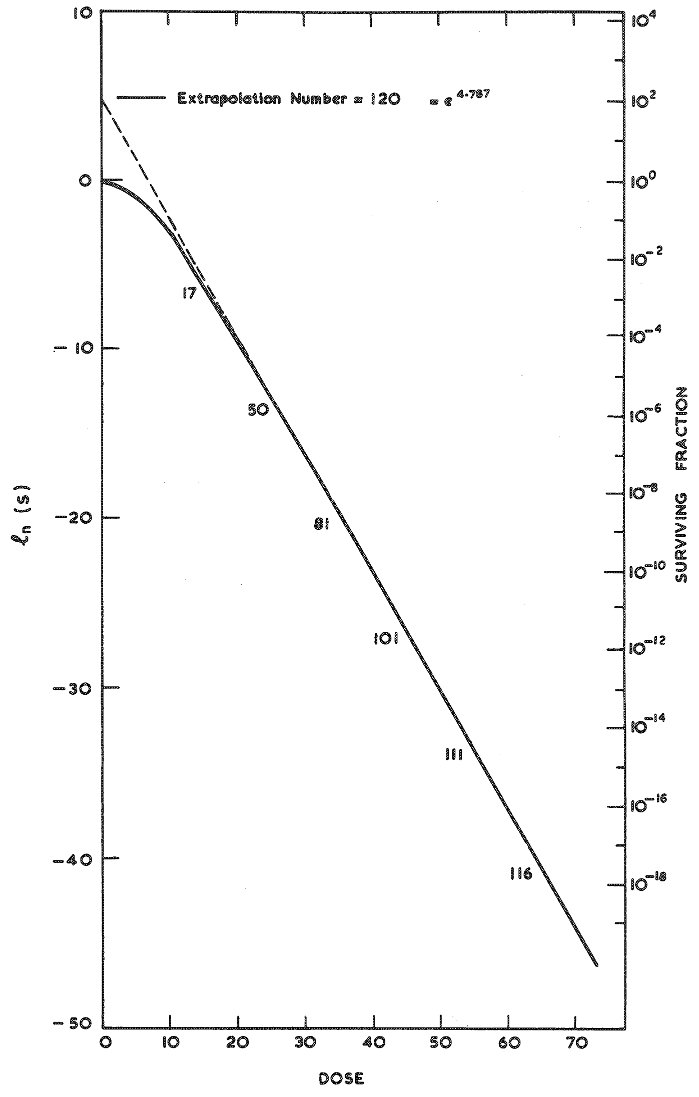


Figure 8.3 Theoretical survival curve, calculated on multi-target, single-hit model. This curve is the same as curve (a) in figure 8.2, only it covers a much greater range. The numbers below the curve show the value of the extrapolation number obtained from points around that region. It can be seen to approach the correct value of 120, but only at exceedingly small survival values. Clearly, extrapolation from survival of 10^{-6}, obtainable by experiment, would yield meaningless values (as in figure 8.2, curve(a)).