THE INVESTIGATIONS LEADING TO THE SPECIFICATION OF A DIGITAL COMPUTER FOR POWER SYSTEM OPERATIONAL STUDIES.

By

D.G. WONG, B.Sc., B.E.

A thesis presented in support of an application for the degree of Master of Engineering Science, in the Department of Electrical Engineering of the University of Sydney.

May, 1959.

THE INVESTIGATIONS LEADING TO THE SPECIFICATION OF A DIGITAL COMPUTER FOR POWER SYSTEM OPERATIONAL STUDIES.

C O N T E N T S.

INTRODUCTION.

This thesis is an account of investigations carried out within the Electrical Engineering Department of the University of Sydney while the author was under the employment of the Snowy Mountains Hydro Electric Authority.

When the project was commenced in November, 1956, the digital differential analyser "ADA" was under construction in the Electrical Engineering Department, and it was understood that another digital differential analyser was to be built for the Snowy Mountains Hydro Electric Authority. This second computer was to be specifically designed for the solution of a System Operational Problem, which had previously taken many "man-years" of manual calculations for its solution.

The starting point of the project was then to gain familiarity with the system Operational Problem. This was carried out by studying the methods used in a series of manual calculations carried out by Engineers of the Authority. The object of the calculations was to derive figures corresponding to the optimum performance of the system. The method proposed for carrying this out was very simple and straight-forward. A set of system parameters was assumed; a set of operating instructions were derived from fundamental principles, (such as the minimisation of spill) and operating experience and the performance of the system was calculated from a set of records (of inflows and other data) covering a period of 50 years. By inspection of the results, modifications to the parameters or to the operating instructions would be made, and the calculations would be repeated. The operating instructions were based on the known state of the storages, the known parameters of the system and the current values of the monthly inflows. No knowledge of future inflows was assumed.

The second aim of the project was to investigate the capabilities of the digital differential analyser, which had been accredited with many of the outstanding advantages of both analogue and digital computers. It had been stated that the programming was extremely simple, as it resembled that of the analogue computer or the mechanical differential analyser. It was also claimed

that the computer possessed a great deal of flexibility (in the form of special integrator functions), as its logical organisation was entirely digital.

For the solution of the System Operational Problem, certain features of the digital differential analyser had to be investigated. Firstly, it was necessary to know whether the computer could be programmed to satisfy the system equations and the operating instructions. Secondly, it was necessary to know whether the input and output facilities of ADA were sufficient for the problem in question. Finally, some estimates of the computational times were also necessary, as the calculations were to be repeated many times.

It was assumed that after the computer had been installed (in Cooma), it would be used by all divisions of the Authority. Some of the problems to which the D.D.A. could profitably be applied included the stability of surge tanks, the cooling of concrete structures and the analysis of stress distributions. It was therefore necessary to analyse in detail the capabilities of the D.D.A. for the solution of various forms of differential equations. Investigations into the use of the special integrator functions were also carried out, as these are necessary in the analysis of systems described by non-linear or discontinuous functions.

After familiarity had been gained with both the System Operational Problem and the capabilities of ADA, it was suspected that the digital differential analyser was not the best kind of computer for the purpose in question. The problem was not the solution of a differential equation. Moreover, it was found that in most cases, the problem consisted of carrying out a sequence of arithmetic operations on whole numbers. At this time, the writer had no experience with general purpose digital computers and no conclusions could immediately be drawn. The use of the general purpose digital computer SILLIAC was available to engineers of the Authority, and the suggestion was made that an attempt should be made to program SILLIAC for the solution of the problem.

The first step was to obtain an exact specification of the problem. The parameters of the system

and the system equations had to be determined. When this was not possible, the required information was estimated as closely as possible, but this did not affect the writing and the checking of the program. Although the final problem was to cover a period of 50 years, the trial program was written to cover a period of only one year. It was realised, however, that the 50 years problem was only a repetition of the single year problem, apart from a number of running totals which would be required. It was therefore necessary to provide a set of inflow figures covering a period of only one year. These figures were chosen so that as many different conditions would be met,and the program would be tested as thoroughly as possible.

When considerations were given to the organisation of the program, it was immediately realised that the storage capacity of SILLIAC was insufficient to store all the parameters and data for the complete (50 years) problem. Although it might have been possible to store the data for the single year problem, the solution of the problem did not require all this information at the same time; the calculations were to be carried out on a monthly basis, and it was therefore decided to repeatedly enter sets of monthly figures. The program was therefore organised to call in a set of monthly figures, to carry out the computation on these figures, to output the required information, to store the required information, and then to call in the next set of figures. This was repeated until 12 months' figures had been processed, when the annual totals of the relevant information was also punched out. Although the program was written to process only one year's information, it was realised that there were no difficulties in storing and punching out the running totals of the annual figures.

The writing of the program was a success, and the results of the trial solution to the problem are presented in the thesis.

During the writing of the program, the suitability of general purpose digital computers for the solution of the operational problems was realised. This fact was stressed in an official report to the Authority. The main reason for this conclusion was that the problem

had been formulated as a sequence of simple arithmetic operations, and the programming of a general purpose digital computer to carry out these operations was very straightforward.

Even after coming to the above conclusion, there still remained the fundamental advantage of the digital differential analyser for the solution of ordinary differential equations. It was found that the D.D.A. was extremely easy to program, and it was quite probable that problems involving the solution of ordinary differential equations would occasionally arise.

To overcome the last obstacle to the recommendation for the construction of a general purpose digital computer instead of a digital differential analyser, it was decided to determine whether it was feasible for a G.P. digital computer to be programmed to operate as a D.D.A., so that the D.D.A. principles of programming could still be applied.

A program for SILLIAC was written to simulate the operation of ADA. The program is such that the mapping and scaling required for the solution of an ordinary differential equation is identical to that required for its solution on ADA. Although the coding of the problem is different, the method used is straightforward and presents no difficulties. The main reason for the choice of a different coding system lies in the fact that the five binary digits of characters on tapes for SILLIAC and ADA are read in the reverse sequence. In particular, a 5'th bit character on a tape for SILLIAC does not necessarily correspond to a 5'th bit character for ADA.

The SILLIAC simulation program was only developed to the stage when the feasibility of the techniques used were indicated. A D.D.A. check program was written to check the simulation of the integrator functions, but the investigations were not carried to the stage when it would be possible to state the relative computational speeds of the simulation program compared with the speed of ADA.

During the course of the above investigations, attention was drawn to the fact that a D.D.A. simulation

program had been written for the IBM 704 computer. This program, which is known as DIDAS, has been used with a great deal of success. Accurate integration formulae are used, and this results in higher computational speeds as a smaller number of iterations would be necessary for the same problem.

The experience gained with the SILLIAC simulation program and the information about DIDAS left little doubt that very effective D.D.A. simulation programs can be written for general purpose digital computers. This conclusion helped to eliminate the objection to general purpose digital computers on the grounds that they were difficult to program for the solution of differential equations.

A paper in the March 1957 issue of the I.R.E. Transactions on Electronic Computers described the logical design of the LGP-30, a serial binary digital computer manufactured by Librascope Inc. Many considerations including the availability of design led to the decision to base the Authority's computer upon the specification of this simple but extremely effective machine. Departures from the specification of the LGP-30 are briefly described in the thesis. A discussion of some of the orders and a brief description of the manual controls are also presented.

At the time of writing it can be said that an advanced stage has been reached in the design, construction and testing of the Authority's computer (which will be known as SNOCOM). It is not the intention of the author to describe any of these aspects in detail. This thesis is concerned primarily with the investigations leading to the specification of SNOCOM and only a brief description of the specification is given in the final chapter.

CHAPTER 1. POWER SYSTEM OPERATION.

1.1 INTRODUCTION.

1.1.1 Thermal Systems.

1.1.2 Combined Hydro-Thermal Systems.

1.2 A SYSTEM OPERATIONAL PROBLEM OF THE SNOWY MOUNTAINS AUTHORITY.

1.2.1 General.

1.2.2 "Controllability" of Hydro Stations.

1.2.3 Utilisation of Hydro Power.

1.2.4 Overall System Operation.

1.2.5 System Equations.

1.2.6 Operating Instructions.

1.2.7 Manual Calculations.

CHAPTER I. POWER SYSTEM OPERATION.

1.1 INTRODUCTION.

1.1.1 Thermal Systems.

Papers on the fundamental concept of incremental cost loading were first published by Steinberg & Smith in 1934. (Ref. 1). These authors dealt mainly with power station aspects but their methods soon found wide application through the power industry.

The economic operation of interconnected power stations was firstly carried out by pre-calculating the loading schedules directly from the production cost curves. However, on the occurrence of a forced plant outage, these pre-calculated schedules were found to be very inflexible. The need for the rapid calculation of the optimum loadings soon led to the development of the loading slide-rule.

When tie-line losses were found to be appreciable, optimum loading was achieved when the plant incremental rate plus a suitable incremental loss term (or penalty factor) was the same for each station. Network analysers and special-purpose computers were used in the calculation of these penalty factors which were then transferred to the loading slide-rule.

With the development of both analogue and digital computing equipment, a more accurate account of transmission losses has been made possible.

Starting with the work of George (Ref. 2), various "loss formulas" have been developed. With certain restrictions on the generator voltages, phase-angles and reactive powers, the total electric network loss was expressed as a function of the individual power inputs from the generating stations. The "B coefficients" used in the loss formulas were used throughout the normal range of a load cycle, but new sets of coefficients were required as system operating conditions were appreciably altered. Incremental transmission loss formulas were derived from the formula for

the total electric network loss.

The curves for the incremental cost of power production at each station can be reasonably approximated by a series of straight lines. The incremental cost of generated power and the incremental transmission losses are used in "load co-ordinating equations" which determine when the incremental cost of delivered power is the same for all stations. (Ref. 2-8).

The solution of the load co-ordinating equations may be solved on either analogue or digital computers, (Ref. 9-15). Analogue techniques have been used very successfully in automatic economic dispatching and load control systems. Examples of economic dispatch computers include "GEDA" - the Goodyear Electronic Differential Analyser used in the Ohio Edison System and the "Early Bird" used in the power system of the Southern Co. Power Pool. Various system tie-line megawatt indications are fed into the computer continuously by telemeter; the computer determines the desired generation from each plant and feeds this information back into the automatic load-frequency control equipment. The use of digital computers has been found to be a valuable tool for the solution of the load-coordinating equations in connection with power system planning. (Ref.32).

During the last decade, much effort has been directed towards the derivation of accurate methods of accounting for transmission losses. Other sources of inaccuracies in economic dispatching systems include those introduced by the representation of the incremental production cost curves and inaccuracies resulting from inability to hold generation at exact desired output. (Ref.16). Although incremental maintenance costs represent only 10% or less of the total incremental production costs, this item has recently been given some attention. (Ref.17). Investigations into the magnitude of the increase in production cost incurred because of routine deviations from the optimum schedule, have shown that a schedule determined from a loading slide rule utilizing carefully estimated penalty factors result in additional production costs, and these extra costs can be rationally used in justifying automatic dispatching equipment. (Ref.18).

Recently, both analogue and digital computers have been used for exact economic dispatch. (Ref.19-21). This new approach to loss minimisation in electric power systems requires no assumptions and the approximations implicit in the B-type constants are avoided.

1.1.2 Combined Hydro-Thermal Systems.

The use of the load-co-ordinating equations discussed above apply only to thermal stations. As the actual operating costs of a hydro station are only a small fraction of those for a thermal station, the application of the co-ordinating equations would result in the immediate utilisation of all available water. However, this is not desirable, as the economic value of a hydro station is far in excess of the resultant saving in fuel.

Hydro stations in a predominantly thermal system are usually designed to operate as peak load stations, as they have many advantages over thermal stations when used for peak load operation. Savings are obtained by the avoidance of banking losses and starting up and shutting down losses which would be incurred by low capacity factor operation of plant. A further most important advantage of having a percentage of hydro-generation in a thermal system is the rapidity with which hydro-electric generator sets can be brought into operation in case of emergency. (Ref.22-25).

With the advent of nuclear power, most systems will become predominantly thermal and this greatly strengthens the case for peak load hydro power and pumped storage schemes. The use of pumped storage generating stations will enable nuclear stations to operate at full load, with the surplus output being stored by means of pumping and subsequently utilised in the form of peak-load power.

The co-ordination of steam generation and incremental transmission losses has been treated quite thoroughly in the literature (Ref.2-21). However, the co-ordination of hydro and steam generation for maximum overall economy is much more involved and only a relatively few papers of a fundamental nature are available. (Ref.26-29, 64). The difficulties introduced by hydro stations lie in the fact that operation

at any time during a period of pondage operation (i.e. periods of drawdown and refill) affects the results for the whole of that period of pondage operation, whereas for thermal stations the operation from instant to instant is largely independent of the operation at any other time.

From investigations on a "mathematical model" of a combined hydro and thermal system, Watchorn has suggested basic co-ordinating equations for combined hydro-thermal systems. (Ref.29). Various equations were investigated and the one which gave the best results was chosen; however, a rigorous basis for these co-ordinating equations was not established.

An extremely promising approach to the problem of economic hydro-thermal operation lies in the use of the calculus of variations (Ref.27, 30). The minimisation of annual operating cost was treated as a problem in the calculus of variations by Cypser. (Ref.27). A high-speed digital computer was used to evaluate the relative advantages of small changes in all control variables and act accordingly in gradually seeking better modes of operation. The amount of data to be processed by the computer in problems of this type was found to be relatively large. Apart from the specification of all plant characteristics, it is necessary for the computer to store the inflows for the entire period under consideration. (For example the optimised operation of a 15 plant system over a period of 50 years using monthly inflows would require 9,000 storage locations for the monthly inflows alone).

The optimisation procedure was demonstrated using a model of a system with two hydro plants with storage and one high-cost thermal plant. Transmission losses were not included.

The linearisation of Cypser's equations has been carried out by Carey and the linearised equations have been applied to short-range load allocation. (Ref.28).

An earlier paper by Chandler, Dandeno, Glimn and Kirchmayer considered the problem of short-range economic operation of a combined hydro-thermal system

when the net heads of the hydroelectric plants could be considered constant. (Ref.26). A recent paper by Glimn and Kirchmayer has extended this work and several methods of treating the head variations are considered. (Ref.64). Using the calculus of variations, equations equivalent to those of Cypser (Ref.27) are derived and a sample problem is considered with reference to a mathematical model using an IBM 650.

When this thesis project was commenced, engineers of the Snowy Mountains Hydro Electric Authority had already spent many "man-years" of manual calculations for the solution of various System Operational Problems. The object of the calculations was to derive figures corresponding to the optimum performance of the system. The method proposed for carrying this out was very simple and straight-forward. A set of system parameters was assumed; a set of operating instructions were derived from the fundamental principles, (such as the minimisation of spill) and operating experience, and the performance of the system was calculated from a set of records (of inflows and other data) covering a period of 50 years. By inspection of the results, modifications to the parameters or to the operating instructions would be made, and the calculations would be repeated. The operating instructions were based on the known state of the storages, the known parameters of the system and the current values of the monthly inflows. No knowledge of future inflows was assumed. A fundamental difference between the above method and that using the calculus of variations lies in the fact that the latter method requires the knowledge of all inflows during the entire period under consideration. The former method has the advantage that the operating instructions together with the subsequent results may be used by the load dispatcher and power station superintendents for the economic operation of the system on a monthly, daily or hourly basis.

Having established the <u>seasonal</u> use of water by using either the "monthly system operational method" or the calculus of variations, the correct manner of operation during any day can be determined in a manner similar to that employed for a thermal station. (Ref. 32). "Incremental cost rates" of a hydro station may be obtained from the "incremental water rates" by placing an arbitrary "value" on the water available to

the station. The "value" used must be that which results in the desired seasonal use of water.

In the past, considerable effort has been directed at the co-ordination of steam generation taking into account transmission losses. (Ref.2-21). The transportation problem of supplying fuel to these stations has been treated by techniques of linear programming (Ref.31). More recently papers on the economic operation of combined hydro-thermal systems have been published (Ref.26-29, 64). High-speed digital computers have been used in all of the above investigations. As the specification of the problems and the mathematical techniques for their solution become established, it is not too optimistic to believe that computer programs will be written for the economic co-ordination of hydro-thermal systems taking into account the inter-related effects of transmission losses, transportation and fuel charges and pondage restrictions. As operating experience of the system is gained, weather and load forecasts may also be taken into account. The ultimate development would of course be some form of central computer-controller which receives all system parameters by telemeter and automatically adjusts the operation of all units in the system so that the optimum over-all system operation is maintained. With the rapid advances made in our present age of "automation, computation and control", it is likely that the realisation of this ultimate development may take place in the very near future.

1.2 A SYSTEM OPERATIONAL PROBLEM OF THE SNOWY MOUNTAINS AUTHORITY.

1.2.1 General.

The specification of the computer which was to be built for the Snowy Mountains Hydro-Electric Authority was to be based upon the requirements of a certain System Operational problem. This problem had been described in some detail by J. Kelly - an engineer from the Authority. (Ref.35). For this reason, a detailed description of this problem will not be given in this thesis. Some of the salient points will, however, be mentioned.

The object of the study was to determine the annual energy production of the Authority's stations, the spillage loss and the monthly fluctuations of the reservoirs over a fifty year period.

All input data was in the form of monthly requirements of energy production and monthly inflows to the stations and reservoirs.

The water flow throughout the system must be controlled whenever possible to satisfy given operating instructions. These operating instructions must be obtained so that the results of the study represent the optimum performance of the system. If the study is carried out on a monthly basis, the operating instructions are based upon the energy requirements, the state of the storages at the beginning of the month and the known inflows and outflows during the month.

1.2.2 "Controllability" of Hydro Stations.

Each power station is classified as being uncontrolled, semi-controlled or fully controlled, according to the size and nature of the storage from which it draws its water supply.

The storage which supplies an uncontrolled station (sometimes called "run-of-the-river" stations); is small, and is only sufficient to take into account the daily fluctuations of the electrical load imposed

upon the power station. The energy generated by the station over a long period of time depends on the amount of water which flows into the storage and the installed capacity of the station. During drought periods, the output from an uncontrolled station may be very small, while during flood seasons, the station may not be able to utilise all available water (by operating 24 hours each day) and hence water will flow over the spillway and will be wasted.

Semi-controlled stations are supplied from water storages of moderately large capacity. These storages are sufficiently large to account for monthly fluctuations of load on the power stations they control. However, they have insufficient capacity to guarantee the operation of the stations during long drought periods.

A storage which supplies a semi-controlled station is generally operated to "marks"; the drawn down filling rates of the storage are controlled whenever possible so that the storage level at the end of each month will be at a predetermined level or "mark". This is carried out to take full advantage of the predicted monthly inflows to the storage throughout the year, and so prevent the storage from either spilling or drying up. For example, at the beginning of the snow-melt months, the mark would generally be fairly low; during these months, the mark is gradually raised so that at the beginning of the dry season, the storage is full but is not spilling. The storage is then drawn down again during the dry season so that it is nearly empty at the beginning of the next snow-melt period.

Fully-controlled stations are supplied from water storages which are sufficiently large to safeguard the operation of the station during long drought periods. Adaminaby Reservoir of the Snowy Mountains Scheme will have a storage capacity of 3,800,000 acre feet, and this is sufficient to safeguard the power generation of the entire Snowy Mountains Scheme.

1.2.3 Utilisation of Hydro Power.

The power stations of the Snowy Mountains Scheme have been designed to operate (as peak load stations) in conjunction with the existing and future

thermal stations of New South Wales and Victoria. The storages of the scheme are sufficiently large to guarantee an output from the stations even during long drought periods, and hence the stations have a "firm power rating".

For any specified load duration curve, the amount of water available and the combined operation of the stations will determine the capacity factors at which each station is to operate. A station which operates at a capacity factor which is less than or equal to the desired capacity factor is said to generate "demand energy". If the station operates at a greater capacity factor, (to take full advantage of the available water), the additional energy generated will replace the operation of thermal stations with the consequent saving in fuel, and hence this additional energy is known as "Fuel-replacement energy". During flood seasons, a station may operate for 24 hours each day (i.e. at 100% capacity factor); any water which flows over the spillway can be expressed in terms of additional energy which could be produced if the additional installed capacity of the station had been available. This energy equivalent of the spill is called"spillage loss".

1.2.4 Overall System Operation.

As the cost of the Snowy Mountains Scheme is to be financed entirely from the sale of electricity, it is essential that a study of the combined operation of the system is carried out, to ensure that the economic potential of the scheme is developed to its maximum advantage.

Monthly records (extending over 50 years) of the inflows to the various storages, together with a series of specified load curves are used in the study to determine the fluctuations of the storages and the demand energy, the fuel-replacement energy and the spillage loss for each station. In the study the stations are treated in the order of their "controllability". To make full use of the energy produced by the uncontrolled stations, these stations are allowed the greatest possible latitude as to which portion of the load curve they are to occupy. The semi-controlled stations are treated after the uncontrolled stations; and finally the fully-

controlled stations are treated so that the stations do produce the total demand energy required from the scheme.

Hydro-stations with firm power ratings allow large savings to be made by putting off the installation of thermal plant. The energy generated in excess of the firm output (i.e. the fuel-replacement energy) is therefore not worth as much as the demand energy produced by the stations. The results of the study may be used to advantage. A knowledge of the total fuel-replacement energy may be used to increase the firm rating of the scheme and hence to increase its worth. Again, the results of the study may be used to determine the most economical design and operation of the system and to be a guide for future development. For example, if a particular storage is found to spill consistently, it may be an economical proposition to increase the installed capacity of this station, and hence to increase the firm rating of the combined system.

1.2.5 System Equations.

Some of the system equations which are used in the study are as follows:-

(a) The Fundamental Storage equation accounting for inflows, outflows and spill.
(b) Surface area - storage equations (for the calculation of evaporation loss).
(c) Level-storage equations (for the calculation of head variations).
(d) Flow-level equations (for the calculation of the maximum flow through tunnels).
(e) Flow-energy-head equations (for the calculation of energy generation).

Not all of the above equations were necessary for all power stations, as in many cases, effects of evaporation loss and head variation were neglected.

Equations (b) and (c) are actually empirical relationships, but may be represented approximately by algebraic equations.

1.2.6 Operating Instructions.

An example of the operating instructions for the Tumut Pond-Adaminaby Reservoir combination will now be presented. These were not described in any great detail in the original notes by Kelly and they are presented to illustrate some of the basic principles (such as the minimisation of spill) upon which these operating instructions are based.

OPERATING INSTRUCTIONS FOR TUMUT POND-ADAMINABY RESERVOIR.

A discussion of a possible set of operating operations for the Tumut Pond - Adaminaby Reservoir combination will now be presented.

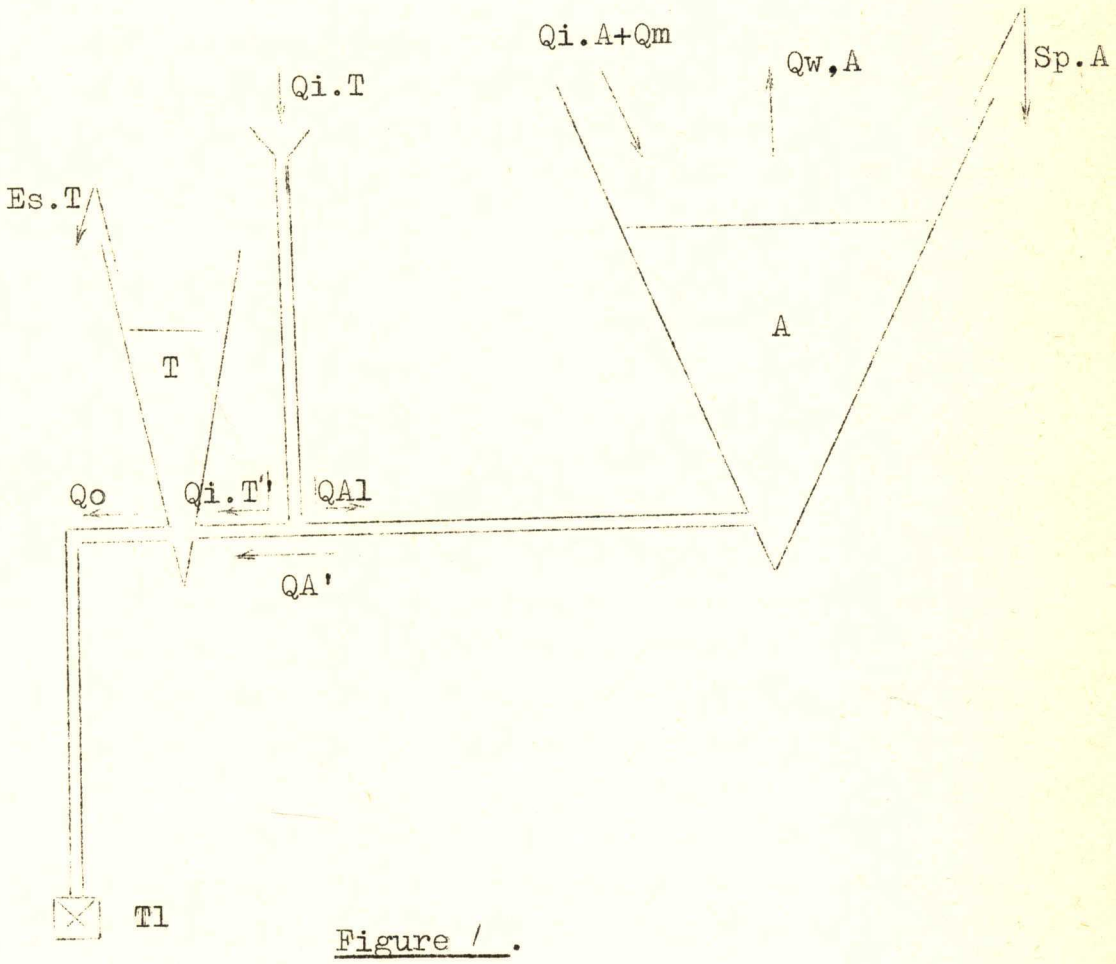


Figure / .

Figure / is a schematic drawing of the system under consideration. The notation used is as follows:-

Qi.A+ Qm	=	Inflow into Adaminaby Reservoir.
Qw.A	=	Evaporation loss from Adaminaby Reservoir.
Sp.A	=	Spill at Adaminaby.
Qi.T	=	Inflow into Tumut Pond which may be diverted partially or entirely into Adaminaby.
Qi.T'	=	Actual Inflow into Tumut Pond.
QAl	=	Water diverted from Tumut Pond into Adaminaby Reservoir.
	=	QiT - QiT'.
Es.T	=	Spillage loss of the T stations.
Qo	=	Required outflow from Tumut Pond.
QA'	=	Water diverted from Adaminaby into Tumut Pond.
S.A. max.	=	Maximum storage of Adaminaby Reservoir.
S.T. max.	=	Maximum storage of Tumut Pond.

So.A = Adaminaby storage at the beginning of the month.

So.T = Tumut Pond storage at the beginning of the month.

QA = Maximum flow from Adaminaby to Tumut Pond.

The system operation study may be started by assuming initial values for the state of the storages. From these values the water levels at Tumut Pond and Adaminaby Reservoir and the water surface area of Adaminaby Reservoir are calculated. The next step in the study is to calculate Qo, QA and Qw.A. The first of these quantities Qo depends on (1) the energy required from the entire Snowy System, (2) the energy generated in the uncontrolled and semi-controlled stations, (3) the uncontrolled inflow into the T stations and (4) the water level of Tumut Pond (i.e. or the total head on the T stations). The second quantity QA depends on the difference in levels of Adaminaby Reservoir and Tumut Pond, and, of course, if the level of Adaminaby is below that of Tumut Pond then QA = 0. The last quantity Qw.A depends on the known rate of evaporation and the surface area.

The inflows Qi.T and Qi.A + Qm are known.

The object now is to determine the values of QA' and QA1 (or Qi.T') so as to obtain the optimum performance of the system.

With Tumut Pond, three conditions are of interest, viz.,

$$So.T - Qo + Qi.T + QA < S.T\ max. \qquad .....(T.1)$$

$$\left.\begin{aligned} So.T - Qo + Qi.T &\leqslant S.T\ max \\ S.T.\ max &\leqslant So.T - Qo + QiT + QA \end{aligned}\right\} \qquad .....(T.2)$$

$$S.T.\ max < So.T - Qo + Qi.T \qquad .....(T.3)$$

With Adaminaby Reservoir, there are also three conditions of interest; these are:-

$$So.A - Qw.A + Qi.A + Qm < S.A\ max \qquad .....(A1)$$

$$So.A - Qw.A + Qi.A + Qm - QA \leqslant S.A\ max. \qquad .....(A2)$$

$$S.A\ max < So.A - Qw.A + Qi.A + Qm - QA \qquad .....(A3)$$

With the combined system there are therefore nine possible conditions. Each of these conditions must be

examined and the operating instructions obtained so as to produce the most economical operation of the system. These operating instructions are based on some fundamental principles which are:-

(a). To cause the minimum amount of spill.
(b). To store water rather than generate fuel replacement energy.
(c). To generate fuel replacement energy rather than to cause a reservoir to spill.
(d). To maintain as large a head on station T1 as possible so as to conserve water.

Now, if condition T1 exists (together with either A1, A2 or A3), it is necessary to make $Qi.T' = Qi.T$ and $QA' = QA$ to be consistent with principle (d).

Again, if condition T2 exists, it is necessary to make $Qi.T' = Qi.T$ to be consistent with principle (d).

If conditions T2 and A1 exist, the flow from Adaminaby to Tumut Pond is restricted so as to just fill Tumut Pond; i.e. $QA' = S.T_{max} - (So.T - Qo + Qi.T)$. By restricting this flow the additional water is stored in Adaminaby and this is consistent with principle (b).

If conditions T2 and A2 exist, the flow from Adaminaby to Tumut Pond is restricted to the larger of the two quantities $S.T_{max} - (So.T - Qo + Qi.T)$ and $(So.A - Qw.A + Qi.A + Qm) - S.A_{max}$. If the former quantity is the larger, at the end of the month Tumut Pond will be full and this is consistent with principles (b) and (d). If the latter quantity is the larger, at the end of the month both Tumut Pond and Adaminaby Reservoir will be full and some fuel replacement energy will be generated by the T stations. This is consistent with principles (c) and (d).

If conditions T2 and A3 exist, the flow from Adaminaby should not be restricted (i.e. $QA' = QA$) so as to be consistent with principles (a) and (c).

If conditions T3 and A1 exist, QA' is made equal to zero and the inflow Qi.T is utilised - (1) by filling Tumut Pond and (2) by filling Adaminaby with the additional water and (3) by generating fuel replacement energy. This is carried out by making QA1 equal to the smaller of the two quantities (So.T - Qo + Qi.T) - S.T max and S.A max - (So.A - Qw.A + Qi.A + Qm). If the former quantity is the smaller, all additional water is stored in Adaminaby; this is consistent with principle (b). If the latter quantity is the smaller, at the end of the month both Tumut Pond and Adaminaby Reservoir will be full and some fuel replacement energy will be generated. This is consistent with principle (c).

It is assumed that the additional water at Tumut Pond may be completely diverted to Adaminaby. A limit imposed by the difference in the water levels and the tunnel dimensions may be considered, but it was thought that this was unnecessary because of the relative magnitudes of the quantities involved.

If conditions T3 and A2 exist, the flow from Adaminaby to Tumut Pond is adjusted so that at the end of the month Adaminaby is full, i.e., QA' is made equal to (So.A - Qw.A + QiA + Qm) - S.A max and Qi.T' is made equal to Qi.T. Some fuel replacement energy is generated and this is consistent with principle (c).

If conditions T3 and A3 exist, the flow from Adaminaby should be unrestricted (i.e. QA' = QA and QiT' = Qi.T) so as to be consistent with principles (a) and (c).

1.2.7 Manual Calculations.

The manual calculations required to carry out the System Operation for one year are shown on pages 25 to 26. These calculations were actually carried out to check the SILLIAC program described in Chapter 3. They are presented here as an example of the calculations which had previously taken many man-years for their solution, and upon which the specification of a computer was to be based.

The figures ringed in blue pencil represent the problem data. The figures ringed in green are the monthly inflows which must be filled in from records for the year concerned. The figures ringed in red are the results required from the study.

With only one exception the complete study may be expressed in terms of a sequence of simple arithmetic operations which are used to obtain figures in one column from those in other columns of the table. These arithmetic operations are as follows:-

(1). Addition
(2). Subtraction
(3). Multiplication
(4). Division
(5). Taking the square root of a number.
(6). Taking the cube root of a number.
(7). Taking the smallest of several numbers.
(8). Taking the largest of several numbers.
(9). Taking the difference of two numbers but recording zero if the difference is negative.

To obtain column 177, a set of instructions must be obeyed. These correspond to the "operating instructions" which are discussed on pages 20 to 23 .

NOTATION : $S(1,2)$ = Smaller of Columns 1 & 2

$S(21,25,26,27,28)$ = Smallest of columns 21, 25, 26, 27, 28.

$L(29,32,34)$ = Largest of columns

	1	2	3	4	5	6	7	8	9	10
	M4						M5H			
	INFLOW	MAX. DEMAND	DEMAND OUTPUT	MAX. OUTPUT	SPILL	F.R.E.	INFLOW	MAX. DEMAND	DEMAND OUTPUT	MAX. OUTPUT
					(1-4) if +ve	1-2-5 if +ve				
	Data	Data	$S(1,2)$	Data			Data	Data	$S(7,8)$	Data
SYMBOL	Ei M4	Edm M4	Eo M4	Em M4	Es M4	Ef M4	Ei M5H	Edm M5H	Eo M5H	Em M5H
UNITS	E	E	E	E	E	E	E	E	E	E
JAN.	.000100	.000163	.000100	.000750	0	0	.000050	.000087	.000050	.000400
FEB.	.000300	.000163	.000163	.000750	0	.000137	.000200	.000087	.000087	.000400
MAR.	.000900	.000179	.000179	.000750	.000150	.000571	.000600	.000095	.000095	.000400
APR.	.000900	.000179	.000179	.000750	.000150	.000571	.000600	.000095	.000095	.000400
MAY	.000900	.000179	.000179	.000750	.000150	.000571	.000600	.000095	.000095	.000400
JUNE	.000900	.000194	.000194	.000750	.000150	.000556	.000600	.000104	.000104	.000400
JULY	.000100	.000194	.000100	.000750	0	0	.000050	.000104	.000050	.000400
AUG.	.000001	.000194	.000001	.000750	0	0	.000001	.000104	.000001	.000400
SEPT.	.000050	.000181	.000050	.000750	0	0	.000010	.000097	.000010	.000400
OCT.	.000050	.000181	.000050	.000750	0	0	.000040	.000097	.000040	.000400
NOV.	.000050	.000181	.000050	.000750	0	0	.000040	.000097	.000040	.000400
DEC.	.000050	.000163	.000050	.000750	0	0	.000040	.000087	.000040	.000400
TOTAL			.001295		.000600	.002406			.000707	

	51	52	53	54	55	56	57	58	59	60
	M1B									M
	TOTAL INFLOW (49+50)	MAX. DEMAND	DEMAND OUTPUT	MAX. OUTPUT	SPILL (51-54) if +ve	F.R.E. (51-52-55) if +ve	U/C INFLOW	CONTR'D INFLOW 0.500 x (51)	TOTAL INFLOW (57+58)	MAX. DEMAND
		Data	$S(51,52)$	Data			Data			Data
SYMBOL	Ei M1B	Edm M1B	Eo M1B	Em M1B	Es M1B	Ef M1B	Eiu M2L		Ei M2L	Edm M2L
UNITS	E	E	E	E	E	E	E	E	E	E
JAN.	.0000795	.000255	.0000795	.000900	0	0	.000090	.0000397	.0001297	.000290
FEB.	.000255	.000255	.000255	.000900	0	0	.000090	.0001275	.0002175	.000290
MAR.	.000271	.000281	.000271	.000900	0	0	.000300	.0001355	.0004355	.000320
APR.	.0002655	.000281	.0002655	.000900	0	0	.000300	.0001328	.0004328	.000320
MAY	.0010316	.000281	.000281	.000900	.0001316	.000619	.000300	.0005158	.0008158	.000320
JUNE	.0010321	.000307	.000307	.000900	.0001321	.000593	.000300	.000516	.000816	.000349
JULY	.000281	.000307	.000281	.000900	0	0	.000049	.0001435	.0001925	.000349
AUG.	.000114	.000307	.000114	.000900	0	0	.000001	.000057	.000058	.000349
SEPT.	.0000992	.000286	.0000992	.000900	0	0	.000010	.0000496	.0000596	.000320
OCT.	.0000859	.000286	.0000859	.000900	0	0	.000040	.0000429	.0000829	.000320
NOV.	.0000859	.000286	.0000859	.000900	0	0	.000040	.0000429	.0000829	.000320
DEC.	.0000715	.000255	.0000715	.000900	0	0	.000040	.0000358	.0000758	.000290
TOTAL			.0022025		.0002637	.001212				

columns 29, 32, 34 | 40^{x} = last month's value of column 40 | "(1-4) if +ve" : if (1) > (4), record difference; if (1) ≤ (4), " zero

10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
MSH			M2H											
MAX. OUTPUT	SPILL	F.R.E	INFLOW	MAX. DEMAND	DEMAND OUTPUT	MAX. OUTPUT	SPILL	F.R.E.	INFLOW	MARKS	OUTPUT TO GAIN MARK	52-49	60-57	84-80
	7-10 if +ve	7-8-11 if +ve					13-16 if +ve	13-14-17 if +ve			$40^{x}+19-20$ if +ve	if +ve	if +ve	if +ve
Data			Data	Data	S'(13,14)	Data			Data	Data				
Em MSH	Es MSH	Ef MSH	Ei m2H	Edm m2H	Eo m2H	Em M2H	Es M2H	Ef m2H	Qi. K	Sc. K	Sm. K	Ec M1B	Ec m2L	Ec M3
E	E	E	E	E	E	E	E	E	V	V	V	E	E	E
·000400	0	0	·000010	·000013	·000010	·000100	0	0	00010	·00376	·00027	·000200	·000200	·000400
·000400	0	·000113	·000050	·000013	·000013	·000100	0	·000037	00050	·00354	·00072	·000050	·000200	·000400
·000400	·000200	·000305	·000200	·000014	·000014	·000100	·000100	·000086	·00050	·00336	·00085	·000050	·000020	·000400
·000400	·000200	·000305	·000200	·000014	·000014	·000100	·000100	·000086	·00050	·00314	·00113	·000050	·000020	·000040
·000400	·000200	·000305	·000200	·000014	·000014	·000100	·000100	·000086	00050	·00293				0
·000400	·000200	·000296	·000200	·000016	·000016	·000100	·000100	·000084	·01100	·00218				0
·000400	0	0	·000010	·000016	·000010	·000100	0	0	·00030	·00131	·00314	·000300	·000300	·000700
·000400	0	0	·000001	·000016	·000001	·000100	0	0	·00040	·00074	·001029	·000306	·000348	·000740
·000400	0	0	·000010	·000013	·000010	·000100	0	0	·00020	·00074	0			
·000400	0	0	·000010	·000013	·000010	·000100	0	0	·00100	·00240	0			
·000400	0	0	·000010	·000013	·000010	·000100	0	0	·00100	·00415	0			
·000400	0	0	·000010	·000013	·000010	·000100	0	0	00100	·00393	0			
	·000800	·001324			·000132		·000400	·000379						

60	61	62	63	64	65	66	67	68	69	70	71	72	73	74
M2L									GUNGARLIN					
MAX. DEMAND	DEMAND OUTPUT	MAX. OUTPUT	SPILL	F.R.E.	INFLOW	MARKS	OUTPUT TO GAIN MARK	(84-80-81)	9.50 x (68)	S(67,69)	$78^{x}+65-70$	(71-77)	(70+72)	AQUEDUCT CAPACITY
	S(59,60)		(59-62) if +ve	(59-60-63)) if +ve			$78^{x}+65-66$ if +ve	if +ve				if +ve		
Data		Data			Data	Data								Data
Edm m2L	Eo m2L	Em m2L	Es m2L	Ef m2L	Qi. G	Sc. G	Sm. G		S'	S''		S^{x}	SR.G	
E	E	E	E	E	V	V	V	E	V	V	V	V	V	V
·000290	·0001297	·000600	0	0	·00010	·00496	·00044	·0001004	·0009538	·00044	·00496	0	·00044	·00200
·000290	·0002175	·000600	0	0	·00010	·00462	·00044	0	0	0	·00506	0	0	·00200
·000320	·000320	·000600	0	·0001155	·00080	·00428		0	0	0	·00586	·00021	·00021	·00200
·000320	·000320	·000600	0	·0001128	·00300	·00394		0	0	0	·00865	·00300	·00300	·00200
·000320	·000320	·000600	·0002158	·000280	·00300	·00359		0	0	0	·00865	·00300	·00300	·00200
·000349	·000349	·000600	·000216	·000251	·00300	·00326		0	0	0	·00865	·00300	·00300	·00200
·000349	·0001925	·000600	0	0	·00010	·00292	·00283	·0002553	·002425	·002425	·003325	0	·002425	·00200
·000349	·000058	·000600	0	0	·00010	·00258	·00127	·000606	·005757	·00127	·00258	0	·00127	·00200
·000320	·0000596	·000600	0	0	·00020	·00369	0			0	·00278	0	0	·00200
·000320	·0000829	·000600	0	0	·00100	·00431	0			0	·00378	0	0	·00200
·000320	·0000829	·000600	0	0	·00100	·00565	0			0	·00478	0	0	·00200
·000290	·0000768	·000600	0	0	·00100	·00530	·00048	·0003899	·003704	·00048	·00530	0	·00048	·00200
	·0022079		·0004318	·0007593										

if (1) > (4), record difference
if (1) ≤ (4), " zero

UNITS : $E = 10^5$ MWM
$V = 10^4$ th. ac. ft.

KOSCIUSKO

21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
OUTPOT TO GAIN MARK $40^x+19-20$ if +ve	52-49 if +ve	60-57 if +ve	84-80 if +ve	11.0 x (22)	22.0 x (23)	9.5 x (24)	15.8 x (44)	S(21,25,26 27,28)	Data	30-1-7 -13-49 -57-80 if +ve	3.28 x (31)	Data	15.8 x (33)	L(29, 32, 34)	$40^x+19-35$	Data
Sm.K	Ec M1B	Ec M2L	Ec M3	S/M1B	S/M2L	S/M3	SdmM1A	SR.K'	Emin.t	Emin s/c	Smin Kos	Emin M1A	Smin M1A	SR.K"		SKmin
V	E	E	E	V	V	V	V	V	E	E	V	E	V	V	V	V
·00027	·000200	·000200	·000400	·00220	·00440	·00380	·00251	·00027	·000246	0	0	·000015	·000237	·00027	·00376	·00041
·00072	·000050	·000200	·000400	·00055	·00440	·00380	·00251	·00055	·000246	0	0	000015	·000237	·00055	·00371	·00041
·00085	·000050	·000020	·000400	·00055	·00044	·00380	·00278	·00044	·000321	0	0	000022	·000348	·00044	·00377	·00041
·00113	·000050	·000020	·000040	·00055	·00044	·00038	·00278	·00038	·000321	0	0	·000022	·000348	·00038	·00389	·00041
			0					0	·000321	0	0	·000022	·000348	·000348	·004042	·00041
			0					0	·000385	0	0	·000038	·00060	·00060	·014443	·00041
·00314	·000300	·000300	·000700	·00330	·00660	·00665	·003081	·003081	·000385	·000128	·000442	·000038	·00060	·003081	·001369	·00041
·001029	·000306	·000348	·000740	·003366	·007656	·00703	·003081	·001029	·000385	·000379	·001243	·000038	·00060	·001243	·000526	·00041
0								0	·000345	·000165	·000541	·000025	·000395	·000541	·000185	·00041
0								0	·000345	·000105	·000344	·000025	·000395	·000395	·001015	·00041
0								0	·000345	·000105	·000344	·000025	·000395	·000395	·00162	·00041
0								0	·000246	·000006	·00002	·000015	·000237	·000237	·002383	·00041

71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87
GARLIN									M3							
$78^x+65-70$	(71-77) if +ve	(70+72)	AQUEDUCT CAPACITY Data	DIVERSION G. To M3 S(73,74)	$78^x+65-75$	MAX. STORAGE Data	STATE OF STORAGE S(76,77)	SPILL (76-77) if +ve	U/C INFLOW Data	CONTR'D INFLOW FROM K. 2.31 x (59)	CONTR'D INFLOW FROM G. .1052 x (75)	TOTAL INFLOW 80+81+82	MAX. DEMAND Data	DEMAND OUTPUT S(83,84)	MAX. OUTPUT Data	SPILL 83-86 if +ve
	S^x	SR.G		SR.G'		S.G max.	S.G	Sp.G	Eiu M3			Ei. M3	Edm M3	Eo M3	Em. M3	Es. M3
V	V	V	V	V	V	V	V	V	E	E	E	E	E	E	E	E
·00496	0	·00044	·00200	·00044	·00496	·00565	·00580 ·00496	0	·000215	·0002996	·0000463	·0005609	·000615	·0005609	·002650	0
·00506	0	0	·00200	0	·00506	·00565	·00506	0	·000215	·0005024	0	·0007174	·000615	·000615	·002650	0
·00586	·00021	·00021	·00200	·00021	·00565	·00565	·00565	0	·000276	·001006	·0000221	·0013041	·000676	·000676	·002650	0
·00865	·00300	·00300	·00200	·00200	·00665	·00565	·00565	·00100	·000636	·0009998	·0002104	·0018462	·000676	·000676	·002650	0
·00865	·00300	·00300	·00200	·00200	·00665	·00565	·00565	·00100	·001000	·0018845	·0002104	·0030949	·000676	·000676	·002650	·0004449
·00865	·00300	·00300	·00200	·00200	·00665	·00565	·00565	·00100	·001000	·001885	·0002104	·0030954	·000741	·000741	·002650	·0004454
·003325	0	·002425	·00200	·00200	·00375	·00565	·00375	0	·000041	·0004447	·0002104	·0006461	·000741	·0006961	·002650	0
·00258	0	·00127	·00200	·00127	·00258	·00565	·00258	0	·000001	·000134	·0001336	·0002686	·000741	·0002686	·002650	0
·00278	0	0	·00200	0	·00278	·00565	·00278	0	·000050	·0001377	0	·0001877	·000680	·0001877	·002650	0
·00378	0	0	·00200	0	·00378	·00565	·00378	0	·000050	·0001915	0	·0002415	·000680	·0002415	·002650	0
·00478	0	0	·00200	0	·00478	·00565	·00478	0	·000050	·0001915	0	·0002415	·000680	·0002415	·002650	0
·00530	0	·00048	·00200	·00048	·00530	·00565	·00530	0	·000050	·0001751	·0000505	·0002756	·000615	·0002756	·002650	0
				·01040				·00300						·0058559		·0008903

37	38	39	40	41	42	43	44	45	46	47	48	49	50
						MIA						MIB	
Data	KOS. FAILS BY 37-36 if +ve	MAX. STORAGE Data	STATE OF STORAGE $S\left[\begin{matrix}39,\\ L(36,3)\end{matrix}\right]$	36-39 if +ve	RELEASE 35+41	INFLOW ·0634 × (42)	MAX. DEMAND Data	DEMAND OUTPUT S(43,44)	MAX. OUTPUT Data	SPILL (43-46) if +ve	F.R.E (43-44 -47) if +ve	u/c INFLOW Data	CONTR'D INFLOW ·0909 × (42)
SKmin		SKmax	S.K	SR.K^x	SR.K	Ei MIA	Edm MIA	Eo MIA	Em MIA	Es MIA	Ef MIA	Eiu MIB	
V	V	V	V	V	V	E	E	E	E	E	E	E	E
·00041	0	·00415	·00393 ·00376	0	·00027	·0000171	·000159	·0000171	·000600	0	0	·000055	·0000245
·00041	0	·00415	·00371	0	·00055	·0000349	·000159	·0000349	·000600	0	0	·000205	·000050
·00041	0	·00415	·00377	0	·00044	·0000279	·000176	·0000279	·000600	0	0	·000231	·000040
·00041	0	·00415	·00389	0	·00038	·0000241	·000176	·0000241	·000600	0	0	·000231	·0000346
·00041	0	·00415	·004042	0	·000348	·0000221	·000176	·0000221	·000600	0	0	·001000	·0000316
·00041	0	·00415	·00415	·010293	·010892	·0006906	·000195	·000195	·000600	·0000906	·000405	·000042	·0009901
·00041	0	·00415	·001369	0	·003081	·0001953	·000195	·000195	·000600	0	·0000003	·000007	·000280
·00041	0	·00415	·000526	0	·001243	·0000788	·000195	·0000788	·000600	0	0	·000001	·000113
·00041	·000225	·00415	·00041	0	·000541	·0000343	·000179	·0000343	·000600	0	0	·000050	·0000492
·00041	0	·00415	·001015	0	·000395	·000025	·000179	·000025	·000600	0	0	·000050	·0000359
·00041	0	·00415	·00162	0	·000395	·000025	·000179	·000025	·000600	0	0	·000050	·0000359
·00041	0	·00415	·002383	0	·000237	·000015	·000159	·000015	·000600	0	0	·000050	·0000215
								·0006942		·0000906	·0004053		

87	88	89	90	91	92	93	94	95	96	97	98	99	100
SPILL 83-86 if +ve	F.R.E. 83-84 -87 if +ve	JINDA. LOCAL INFLOW Data	RELEASE FROM m3 9.50 × (83)	TOTAL INFLOW To JINDA. 89+90 +79+	DEMAND ENERGY FROM u/c s/c STATIONS 3+9+15 +45+53 +60+85								
Es. m3	Ef. m3	ΔJ		Qi. J	Eot.u/c s/c								
E	E	V	V	V	E								
0	0	·00100	·005329	·006329	·0009472								
0	·0001024	·00100	·006815	·007815	·0013854								
0	·0006281	·01000	·012389	·022389	·0015829								
0	·0011702	·00629	·017539	·027653	·0015736								
·0004449	·0019774	·01000	·029402	·039403	·0015871								
·0004454	·001909	·01000	·029406	·039406	·001906								
0	0	·00864	·006613	·016964	·0015306								
0	0	·01000	·002552	·012552	·0005224								
0	0	·00100	·001783	·002783	·0004608								
0	0	·01000	·002294	·012294	·0005353								
0	0	·01000	·002294	·012294	·0005353								
0	0	·01000	·002618	·012618	·0005379								
·0008903	·0057837			·212499									

UNITS $E = 10^5$ MWM
$V = 10^4$ th.ac.ft

COLUMN 177 : if 175 is non-zero, 177 = 0
if 181 > 168, 177 = 161
if 170 > 190, 177 = 161
if 170 ≤ 190 ≤ 169 and 181
if 167 ≤ 181 ≤ 168 and 169
if 170 ≤ 190 ≤ 169 and 167

JINDABYNE

	101	102	103	104	105	106	107	108	109	110	111	112	113	114	115	116
	MAX. DEMAND ALL M STATIONS Data	(128-125) if +ve	(101-125) -92 if +ve	4·22 × (162)	4·22 × (103)	$\sqrt[3]{122^x}$	·00259 × (106)	SURFACE AREA (107) − ·000112	RATE OF EVAP. Data	EVAP. LOSS (108) × (109)	MARKS Data	OUTPUT TO GAIN MARK 122^x +91 −110 −111 if +ve	$S'_{112}^{(104,105)}$	MIN. ENERGY FROM M6-7 Data	114-125 if +ve	MIN. RELEASE 4·22 × (115)
SYMBOL	Em Ji	E Je	E Ji	S Je	S Ji	−	−	A.J	Kev	Qev. J	Sc. J	Sm. J	S'	E min J		S min J
UNITS	E	E	E	V	V	−	−	10^4 th. ac	ft	V	V	V	V	E	E	V
JAN.	·003820	·003300	·002802 8		·01182 8	·4948	·00128 2	·00117	·484	·00056 6	·11700	·00986 3	009863	·001720	·001650	·006963
FEB.	·003820	·002000	·001064 6		·00449 3	·4891	·001267	·001155	·650	·00075 1	·11100	·013064	·004493	·001720	·00035	·001477
MAR.	·004000	·00067	0		0	·4927	·001276	·001164	·480	·000559	·10700		0	·002020	0	0
APR.	·004000	0	0	0	0	·4948	·001282	·00117	·195	·000228	·10200		0	·002020	0	0
MAY	·004000	·00359	·002332 9		·00984 5	·4948	·001282	·00117	·050	·000059	·09860	·061843	·009845	·002020	·00194	·008187
JUNE	·004950	·00401	·002964		·01250 8				0	0	·09360	·066906	·012508	·002320	·00224	·009453
JULY	·004950	·00401	·003339 4		·01409 2				0	0	·08800	·050064	·014092	·002320	·00224	·009453
AUG.	·004950	·00401	·004347 6	·016922		·4948	·001282	·00117	·030	·000035	·08400	·049617	·016922	·002320	·00224	·009453
SEPT.	·004030	·00341	·003499 2	·014390		·4887	·001266	·001154	·080	·000092	·09180	·027586	·014390	·002020	·00194	·008187
OCT.	·004030	·00341	·003414 7	·014390		·4718	·001222	·00111	·080	·000089	·10900	·008201	·008201	·002020	·00194	·008187
NOV.	·004030	·00341	·003414 7	·014390		·4776	·001237	·001125	·100	·000112	·11800	·003182	·003182	·002020	·00194	·008187
DEC.	·003820	·00329	·003202 1		·013513	·4835	·001252	·00114	·280	·000319	·12110	·004194	·004194	·001720	·00164	·006921
TOTAL																

TUMUT POND —

	151	152	153	154	155	156	157	158	159	160	161	162	163	164	165	166
	·313 × (150)	TOTAL HEAD ·2313 + (151)	T.P. LEVEL ·1285 + (152)	0·824 × (149) ÷ (152)	MIN. STORAGE ADAm. Data	$\sqrt[3]{191^x}$	·0404 × (156)	ADAm. LEVEL ·3526 + (157)	(158−153) if +ve	$\sqrt{159}$	MAX. E-T FLOW ·3041 × (160)	·00652 × 191^x	SURFACE AREA ·00149 + (162)	EVAP. LOSS (109) × (163)	INFLOW Data	INFLOW Data
SYMBOL	−	H	LT	Qo	S.Amin	−	−	LA	−	−	QA	−	A.A	Qev. A	Qi.T	Qi. A
UNITS	−	10^4 ft	10^4 ft	V	V	−	−	10^4 ft	−	−	V	−	10^4 th. ac	V	V	V
JAN.	·019796	·251096	·379596	·005225	·00620	·71138	·028740	·381340	·0017444	·04176	·012699	·002347	·003837	·001857	·00200	·00500
FEB.	·020620	·251920	·380420	·003911	00620	·71368	·028833	·381433	·001013	·03182	·009676	·002370	·003860	·002509	·00500	·00200
MAR.	·020620	·251920	·380420	·001028	·00620	·71659	·028950	·381550	·001130	·03362	·010224	·002399	·003889	·001867	·00800	·03000
APR.	·020620	·251920	·380420	·001058	·00620	·72811	·029416	·382016	·001596	·03995	·012149	·002517	·004007	·000781	·00020	·00100
MAY	·020620	·251920	·380420	·001014	·00620	·72811	·029416	·382016	·001596	·03995	·012149	·002517	·004007	·000200	·00200	·00500
JUNE	·020620	·251920	·380420	·004153	·00620	·72811	·029416	·382016	·001596	·03995	·012149			0	·05000	·01000
JULY	·020620	·251920	·380420	·005381	·00620	·72811	·029416	·382016	·001596	·03995	·012149			0	·00020	·00100
AUG.	·020620	·251920	·380420	·008680	·00620	·72789	·029407	·382007	·001587	·03984	·012115	·002514	·004004	·000120	·00020	·00100
SEPT.	·020620	·251920	·380420	·004446	·00620	·72549	·029310	·381910	·001490	·03860	·011738	·002490	·003980	·000318	·00020	·00100
OCT.	·020620	·251920	·380420	·008966	·00620	·72562	·029315	·381915	·001495	·03867	·011760	·002491	·003981	·000318	·00020	·00100
NOV.	·020620	·251920	·380420	·008977	·00620	·72287	·029204	·381804	·001384	·03720	·011313	·002463	·003953	·000396	·00020	·00100
DEC.	·020620	·251920	·380420	·009119	·00620	·72003	·029089	·381689	·001269	·03562	·010832	·002434	·003924	·001099	·00020	·00100
TOTAL																

if 170 ≤ 190 ≤ 169 and 181 < 167 , 177 = 174
if 167 ≤ 181 ≤ 168 and 169 < 190 , 177 = 172
if 170 ≤ 190 ≤ 169 and 167 ≤ 181 ≤ 168 , 177 = L(174, 172)

COLUMN 137 if $144^{x} < 133$, 137 = 0
COLUMN 161 if $191^{x} < 155$, 161 = 0

113	114	115	116	117	118	119	120	121	122	123	124	125	126	127	128	129	130
												M6-7					
$S'_{(104,105)\,112}$	MIN. ENERGY FROM M6-7 Data	114-125 if +ve	MIN. RELEASE 4.22 × (115)	L(113, 116)	$122^{x}+91-110-117$	MIN. STORAGE Data	JINDA. FAILS BY 119-118 if +ve	MAX. STORAGE Data	STATE OF STORAGE $S\left[{}^{121}_{L(118,119)}\right]$	(118-121) if +ve	RELEASE (117+123)	u/c INFLOW Data	CONTR'D INFLOW .237 × (124)	TOTAL INFLOW (125+126)	MAX. DEMAND Data	DEMAND OUTPUT S(127, 128)	MAX. OUTPUT Data
S'	E min J		Smin J	S''		SJ min		SJ max	S. J	S^{*}	SR. J			Ei. J	Em Je	Eo J	Em J
V	E	E	V	V	V	V	V	V	V	V	V	E	E	E	E	E	E
.009863	.001720	.001650	.006963	.009863	.11700	.00610	–	.12110	.12110 .11700	0	.009863	.000070	.0023375	.0024075	.003370	.0024075	.010800
.004493	.001720	.00035	.001477	.004493	.119571	.00610	–	.12110	.119571	0	.004493	.001370	.0010648	.0024348	.003370	.0024348	.010800
0	.002020	0	0	0	.141401	.00610	–	.12110	.12110	.020301	.020301	.003000	.0048113	.0078113	.003670	.003670	.010800
0	.002020	0	0	0	.148525	.00610	–	.12110	.12110	.027425	.027425	.006000	.0064997	.0124997	.003670	.003670	.010800
.009845	.002020	.00194	.008187	.009845	.150598	.00610	–	.12110	.12110	.029448	.039343	.000080	.0093243	.0094043	.003670	.003670	.010800
.012508	.002320	.00224	.009453	.012508	.147998	.00610	–	.12110	.12110	.026888	.039396	.000080	.0093369	.0094169	.004090	.004090	.010800
.014092	.002320	.00224	.009453	.014092	.123972	.00610	–	.12110	.12110	.002872	.016964	.000080	.0040205	.0041005	.004090	.004090	.010800
.016922	.002320	.00224	.009453	.016922	.116695	.00610	–	.12110	.116695	0	.016922	.000080	.0040105	.0040905	.004090	.004090	.010800
.014390	.002020	.00194	.008187	.014390	.104996	.00610	–	.12110	.104996	0	.014390	.000080	.0034104	.0034904	.003490	.003490	.010800
.008201	.002020	.00194	.008187	.008201	.109000	.00610	–	.12110	.109000	0	.008201	.000080	.0019436	.0020236	.003490	.0020236	.010800
.003182	.002020	.00194	.008187	.008187	.112995	.00610	–	.12110	.112995	0	.008187	.000080	.0019403	.0020203	.003490	.0020203	.010800
.004194	.001720	.00164	.006921	.006921	.118373	.00610	–	.12110	.118373	0	.006921	.000080	.0016403	.0017203	.003370	.0017203	.010800
																.0373765	

163	164	165	166	167	168	169	170	171	172	173	174	175	176	177	178	179	180
	TUMUT POND — ADAMINABY RESERVOIR																
SURFACE AREA .00149 + (162)	EVAP. LOSS (109) × (163)	INFLOW Data	INFLOW Data	$182^{x}+165-154$	$182^{x}+165-154+161$	$191^{x}+166+137-164$	$191^{x}+166+137-164-161$	167-181 if +ve	181-167 if +ve	190-169 if +ve	169-190 if +ve	S(171, 173)	(165-175)	See Notes Above.	$182^{x}+176+177-154$	MIN. STORAGE T.P. Data	T.P. FAILS BY (179-178) if +ve
A.A	Qev. A	Qi. T	Qi. A									QA1	Qi. T'	QA'		ST min	
10^{4} th. ac	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V	V
.003837	.001857	.00200	.00500	.000775	.013474	.367064	.354365	0	.003565	.018936	0	0	.00200	.003565	.00434	.00043	–
.003860	.002509	.00500	.00200	.005429	.015105	.366880	.357204	.001089	0	.01412	0	.001089	.003911	0	.00434	.00043	–
.003889	.001867	.00800	.03000	.011312	.021536	.400048	.389824	.006972	0	0		0	.00800	.010224	.021536	.00043	–
.004007	.000781	.00020	.00100	.003482	.015631	.390136	.377987	0	.000858	0	.004136	0	.00020	.004136	.007618	.00043	–
.004007	.000200	.00200	.00500	.005326	.017475	.394689	.382540	.000986	0	0	.008689	0	.00200	.008689	.014015	.00043	–
	0	.05000	.01000	.050187	.062336	.39986	.387711		0	0		0	.05000	.012149	.062336	.00043	–
	0	.00020	.00100	–.000841	.011308	.390831	.378682	0	.005181	0	.004831	0	.00020	.005181	.00434	.00043	–
.004004	.000120	.00020	.00100	–.004140	.007975	.39033	.378215	0	.008480	0	.00433	0	.00020	.008480	.00434	.00043	–
.003980	.000318	.00020	.00100	.000094	.011832	.386296	.374558	0	.004246	0	.000296	0	.00020	.004246	.00434	.00043	–
.003981	.000318	.00020	.00100	–.004426	.007334	.386489	.374729	0	.008766	0	.000489	0	.00020	.008766	.00434	.00043	–
.003953	.000346	.00020	.00100	–.004437	.006876	.382074	.370761	0	.008777	.003926	0	0	.00020	.008777	.00434	.00043	–
.003924	.001099	.00020	.00100	–.004579	.006253	.377017	.366185	0	.008919	.008993	0	0	.00020	.008919	.00434	.00043	–
												.001089		.083132			

128	129	130	131	132	133	134	135	136	137	138	139	140
M6-7					TANTANGARA							
MAX. DEMAND Data	DEMAND OUTPUT $S(127,128)$	MAX. OUTPUT Data	SPILL (127-130) if +ve	FRE (127-128) -131 if +ve	MIN. STORAGE Data	$\sqrt[3]{144^x}$	$0.219 + (134)$	$\sqrt[2]{(135)}$	FLOW TANT. TO ADAM. $.00516 \times (136)$	$.0136 \times (144)^x$	SURFACE AREA $.000365 + (138)$	EVAP. LOSS $(109) \times (139)$
Em Je	Eo J	Em J	Es J	Ef J	S.Ta min	—	—	—	Qm	—	A.Ta.	Qev.Ta
E	E	E	E	E	V	—	—	—	V	—	10^4 ha ac	V
.003370	.0024075	.010800	0	0	.00500	.3583	.5773	.7598	.003921	.0006256	.0009906	.000479
.003370	.0024348	010800	0	0	.00500	.3492	.5682	.7538	.003890	.0005794	.0009444	.000614
.003670	.003670	010800	0	.0041413	.00500	.3659	.5849	.7647	.003946	.0006664	.0010314	.000495
.003670	.003670	010800	.0016997	.007130	.00500	.3572	.5762	.7591	.003917	.0006196	.0009846	.000192
.003670	.003670	010800	0	.0057343	.00500	.3488	.5678	.7536	.003889	.0005773	.0009423	.000047
.004090	.004090	010800	0	.0053269	.00500	.3406	.5596	.7480	.003860			0
.004090	.004090	010800	0	.000105	.00500	.3322	.5512	.7425	.003831			0
.004090	.004090	010800	0	.000000 5	.00500	.3233	.5423	.7364	.003800	.0004600	.000825	.000025
.003490	.003490	010800	0	.000000 4	.00500	.3142	.5332	.7302	.003768	.0004216	.0007866	.000063
.003490	.0020236	010800	0	0	.00500	.3113	.5303	.7281	.003757	.0004103	.0007753	.000062
.003490	.0020203	010800	0	0	.00500	.3084	.5274	.7261	.003747	.0003991	.0007641	.000076
.003370	.0017203	010800	0	0	.00500	.3287	.5477	.7401	.003819	.0004831	.0008481	.000237
	.0373765		.0016997	.0223439					.046145			

178	179	180	181	182	183	184	185	186	187	188	189	190
$182^x + 176 + 177 - 154$	MIN. STORAGE T.P. Data	T.P. FAILS BY (179-178) if +ve	MAX. STORAGE T.P. Data	T.P. STATE OF STORAGE $S\left[\begin{matrix}181\\L(178,179)\end{matrix}\right]$	(178-181) if +ve	$.2932 \times (183)$	MAX. OUTPUT Data	(185-147)	FRE $S(184,186)$	SPILL (184-186) if +ve	$191^x + 166 + 137 - 164 + 175 - 177$	MAX. STORAGE ADAM. Data
	ST min		S.T max	S.T			Em.T		Ef.T	Es.T		S.A max
V	V	V	V	V	V	E	E	E	E	E	V	V
.00434	.00043	—	.004434	.00400 .00434	0	0	.01010		0	0	.363499	.38600
.00434	.00043	—	.004434	.00434	0	0	.01010		0	0	.367969	.38600
.021536	.00043	—	.004434	.00434	.017196	.0050419	.01010	.0096859	.0050419	0	.389824	.38600
.007618	.00043	—	.004434	.00434	.003278	.0009611	.01010	.0096766	.0009611	0	.38600	.38600
.014015	.00043	—	.004434	.00434	.009675	.0028367	.01010	.0096901	.0028367	0	.38600	.38600
.062336	.00043	—	.004434	.00434	.057996	.0170044	.01010	.008730	.008730	.0082744	.387711	.38600
.00434	.00043	—	.004434	.00434	0	0	.01010		0	0	.385650	.38600
.00434	.00043	—	.004434	.00434	0	0	.01010		0	0	.381850	.38600
.00434	.00043	—	.004434	.00434	0	0	.01010		0	0	.382050	.38600
.00434	.00043	—	.004434	.00434	0	0	.01010		0	0	.377523	.38600
.00434	.00043	—	.004434	.00434	0	0	.01010		0	0	.373297	.38600
.00434	.00043	—	.004434	.00434	0	0	.01010		0	0	.368098	.38600
									.0175697	.0082744		

137	138	139	140	141	142	143	144	145	146	147	148	149	150
TANTANGARA									TUMUT - ADAMINABY				
FLOW TANT. TO ADAM. ·00516 × (136)	·0136 × (144)^x	SURFACE AREA ·000365 + (138)	EVAP. LOSS (109) × (139)	INFLOW Data	144^x +141 −140−137	MAX. STORAGE Data	STATE OF STORAGE Σ(142,143)	SPILL (142−143) if +ve	TOTAL SNOWY DEMAND Data	ENERGY FROM A. GROUP 146−129 −92	u/c INFLOW Data	ENERGY FROM T.P RELEASE (147−148)	$\sqrt{182^x}$
Qm	—	A.Ta.	Qev.Ta	Qi.Ta		S.Ta max	S.Ta	Sp.Ta	TSD	EAT		EA	—
V	—	10^4 ha ac	V	V	V	V	V	V	E	E	E	E	—
·003921	·0006256	·0009906	·000479	·00100	·04260	·04900	·04600 ·04260	0	·005146	·0017913	·000199	·0015923	·063246
·003890	·0005794	·0009444	·000614	·01500	·053096	·04900	·04900	·004096	·005146	·0013258	·000130	·0011958	·065879
·003946	·0006664	·0010314	·000495	·00100	·045559	·04900	·045559	0	·005667	·0004141	·000100	·0003141	·065879
·003917	·0006196	·0009846	·000192	·00100	·04245	·04900	·042450	0	·005667	·0004234	·000100	·0003234	·065879
·003889	·0005773	·0009423	·000047	·00100	·039514	·04900	·039514	0	·005667	·0004099	·000100	·0003099	·065879
·003860			0	·00100	·036654	·04900	·036654	0	·007366	·001370	·000100	·001270	·065879
·003831			0	·00100	·033823	·04900	·033823	0	·007366	·0017454	·000100	·0016454	·065879
·003800	·0004600	·000825	·000025	·00100	·030998	·04900	·030998	0	·007366	·0027536	·000100	·0026536	·065879
·003768	·0004216	·0007866	·000063	·00300	·030167	·04900	·030167	0	·005400	·0014592	·000100	·0013592	·065879
·003757	·0004103	·0007753	·000062	·00300	·029348	·04900	·029348	0	·005400	·0028411	·000100	·0027411	·065879
·003747	·0003991	·0007641	·000076	·01000	·035525	·04900	·035525	0	·005400	·0028444	·000100	·0027444	·065879
·003819	·0004831	·0008481	·000237	·01000	·041469	·04900	·041469	0	·005146	·0028878	·000100	·0027878	·065879
·046145								·004096		·0202660			

187	188	189	190	191	192	193	194	195	196	197	198	199	200
FRE. Σ(184,186)	SPILL (184−186) if +ve	191^x +166 +137−164 +175−177	MAX. STORAGE ADAM. Data	STATE OF STORAGE − ADAM. Σ(189,190)	SPILL −ADAM (189−190) if +ve								
Ef.T	Es.T		S.Amax	S.A	Sp.A								
E	E	V	V	V	V								
0	0	·363499	·38600	·36000 ·363499	0								
0	0	·367969	·38600	·367969	0								
·0050419	0	·389824	·38600	·38600	·003824								
·0009611	0	·38600	·38600	·38600	0								
·0028367	0	·38600	·38600	·38600	0								
·008730	·00827744	·387711	·38600	·38600	·001711								
0	0	·385650	·38600	·385650	0								
0	0	·381850	·38600	·381846	0								
0	0	·382050	·38600	·382050	0								
0	0	·377523	·38600	·377723	0								
0	0	·373297	·38600	·373297	0								
0	0	·368098	·38600	·368098	0								
0175697	00827744				005535								

DGW

CHAPTER 2. PROGRAMMING OF THE DIGITAL DIFFERENTIAL ANALYSER "ADA".

2.1 BASIC CONCEPTS OF THE DIGITAL DIFFERENTIAL ANALYSER.

2.2 PROGRAMMING PRINCIPLES.

2.2.1 Block diagrams and Mapping (with examples).

2.2.2 Scaling.

2.2.3 Coding (with example).

2.3 A BRIEF DISCUSSION OF SELECTED EXAMPLES.

2.3.1 A differential equation with 2 point boundary conditions.

2.3.2 Partial differential equations.

2.3.3 Interpolation formulae.

2.3.4 Fourier Analysis.

2.3.5 Time Lag (or delay) Function.

2.3.6 The evaluation of a double integral.

2.3.7 Miscellaneous.

2.4 D.D.A. REQUIREMENTS OF THE SYSTEM OPERATIONAL PROBLEM.

2.4.1 Arithmetical Operations on Whole Numbers.

2.4.2 Remarks on D.D.A. Requirements.

CHAPTER 2. PROGRAMMING OF THE DIGITAL DIFFERENTIAL ANALYSER "ADA".

2.1. BASIC CONCEPTS OF THE DIGITAL DIFFERENTIAL ANALYSER.

The digital differential analyser is a special-purpose digital computer designed for the solution of ordinary differential equations. It combines the advantages of both analogue and digital systems, as the programming of the computer is very similar to that of the mechanical or electronic differential analyser, while the internal organisation of the computer is entirely digital.

Unlike the mechanical differential analyser which consists of a number of individual units such as integrators, adders, gear boxes etc., in ADA, there is no physical entity which can be described as an integrator. Integrators in ADA can best be described as operational entities.

At present, ADA has a capacity of 50 integrators, each of which is associated with 4 numerical registers stored on the surface of a magnetic drum. All the integrators share a common arithmetic and control section. The registers (which have a capacity of 6 decimal digits plus sign) are taken into this arithmetic and control section in turn, they are processed and then rewritten back on the drum.

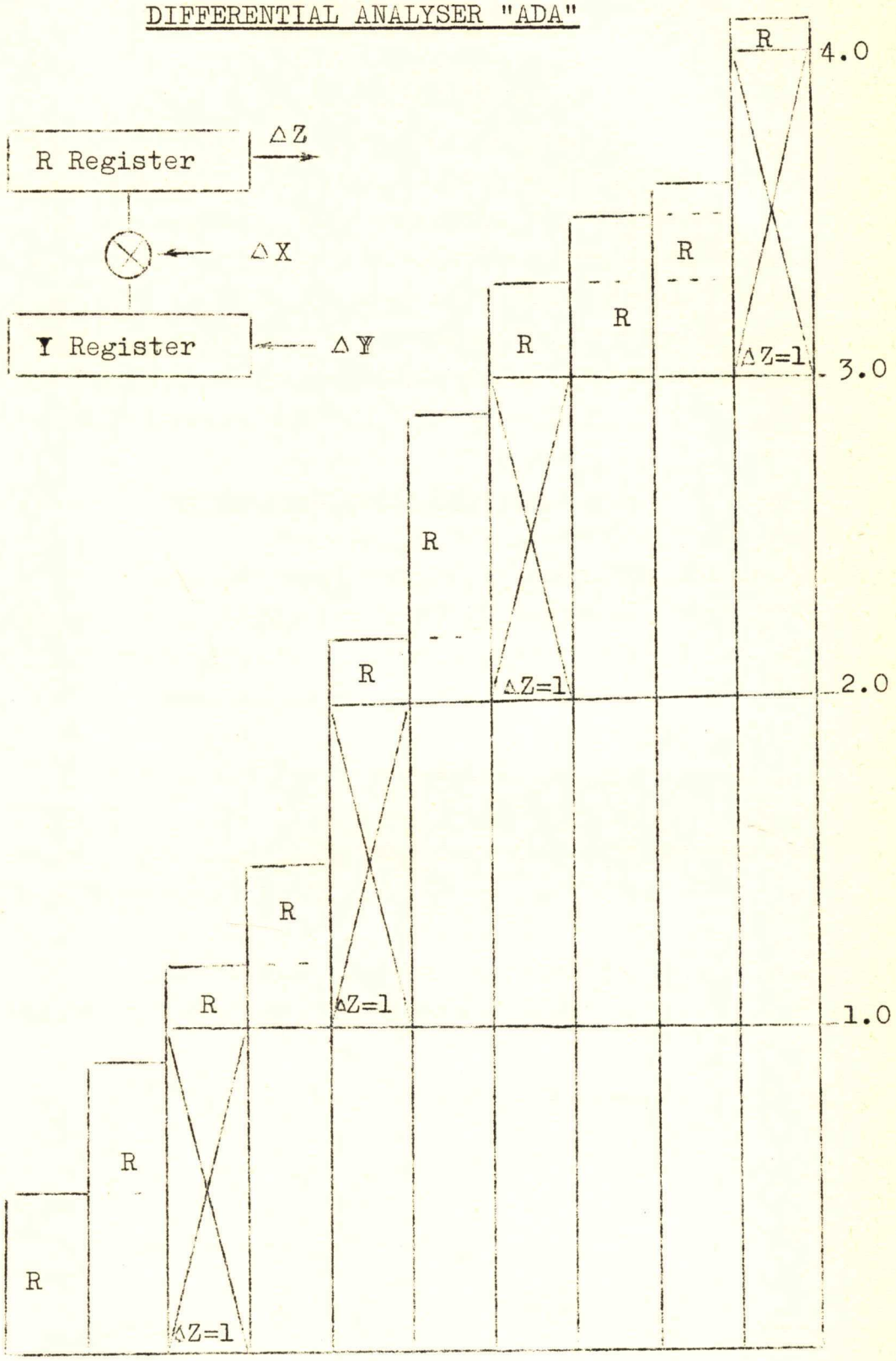
In an analogue computer, the outputs of all units are to be considered as continuous variables, while in the digital differential analyser, the outputs are made to vary by small increments. These increments can only take 3 values, which are numerically equal to + 1, 0 or - 1 times the value of unity in the least significant decimal digit of the appropriate registers.

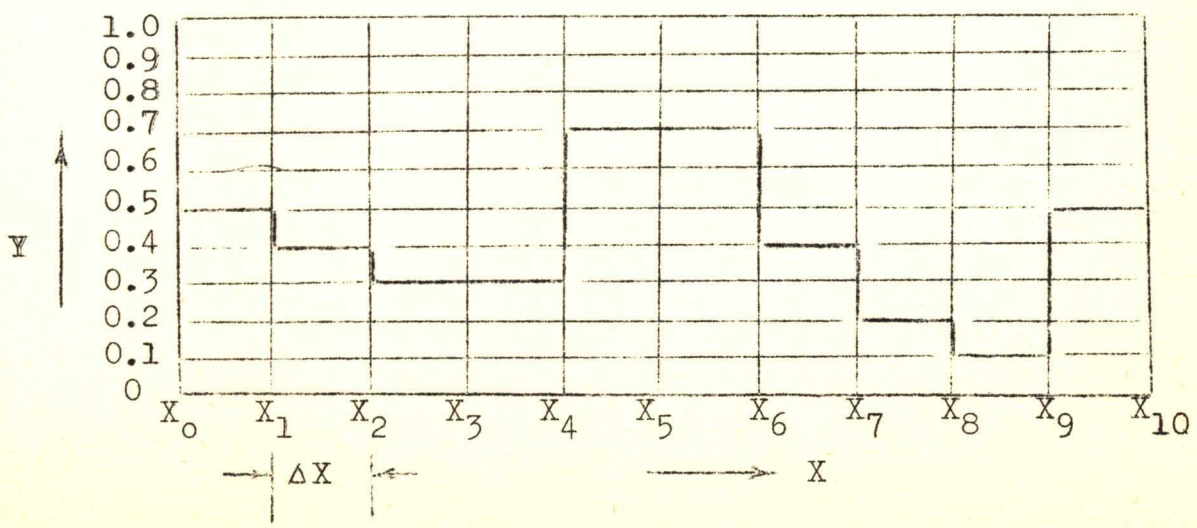
The basic integration process, which involves 2 of the 4 registers associated with each integrator, may be considered to take place in several stages. These registers known as the Y and R registers are shown in the simplified schematic on page 30. In the first stage, all Δy increments are added into the Y register, thus

forming the current value of the integrand. The R and Y registers are connected by a gate which is controlled by the Δx input. Whenever Δx is + 1, 0 or - 1, the current value of the Y register multiplied by + 1, 0 or - 1 respectively is added to the current value of the R register. It is apparent that if the capacity of the R register were made very large, it could represent the integral of y with respect to x. However, this is not done as the integrator would not produce an <u>incremental</u> output which is essential if it is to be used as a Δx or Δy input to other integrators in the normal differential analyser method of interconnections. Instead, the capacity of the R register is made equal to that of the Y register, so that in the final stage of the integration process, the overflows can be used to represent the incremental output.

The integration process is illustrated on page 30. The contributions to the integral in the first 3 intervals are 0.5, 0.4 and 0.3 respectively. If R were initially zero, and the capacity of R & Y were limited to 0.999999, the final values of R would be 0.5, 0.9 and 0.2. The 0.2 value would be associated with a + 1 output.

DIAGRAM ILLUSTRATING THE BASIC INTEGRATION PROCESS OF THE DIGITAL DIFFERENTIAL ANALYSER "ADA"





2.2 PROGRAMMING PRINCIPLES.

The preparation of problems for their solution on a digital differential analyser involves four steps:- block-diagram representation, mapping, scaling and coding. Block-diagram representation is not always necessary and is often included under mapping. However, for new and complex problems, simplification is often achieved by representing the problem initially in this form, and appropriate emphasis is placed upon the main program interconnections.

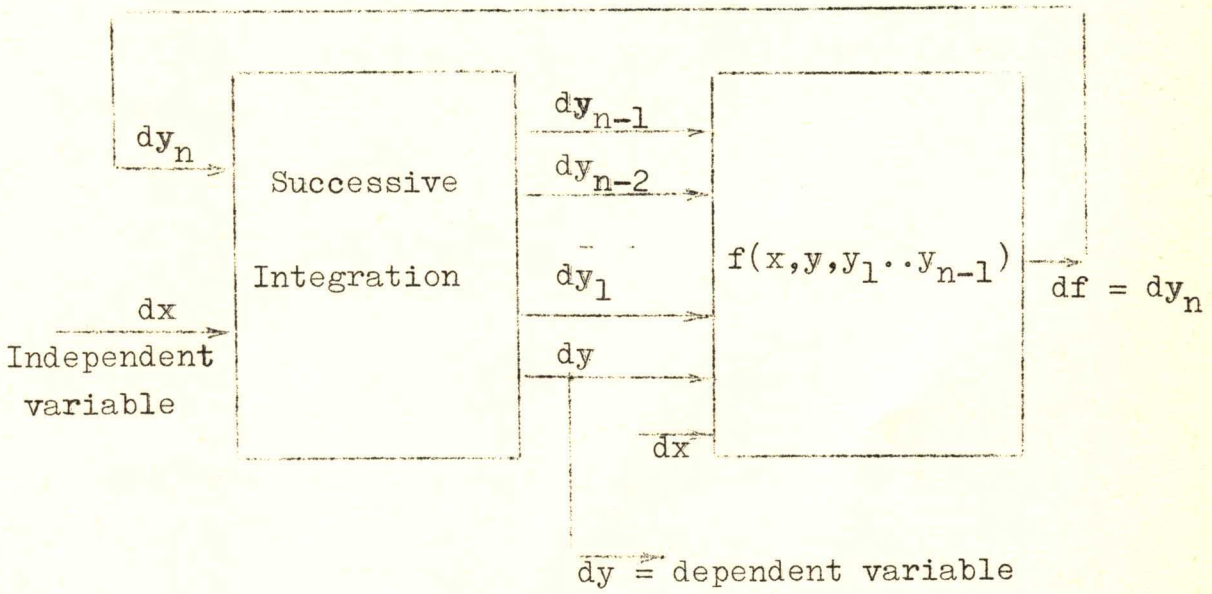
2.2.1 Block Diagrams & Mapping.

For the solution of an ordinary differential equation, if possible it is often desirable to express the highest derivative explicitly in terms of the other variables. Such an equation would then take the form:-

$$y_n = f(x, y, y_1, y_2, \ldots . y_{n-1})$$

where $y_1, y_2, \ldots . y_n$ are the 1st, 2nd....n'th derivatives of y with respect to x.

The block diagram representation of the solution of this equation is as follows:-



The block marked "successive integration" represents a chain of integrators which generate the lower derivatives and the dependent variable from the highest derivative and the independent variable by

successive integration. The block marked $f(x_1\ y_1\ y_1 \cdots y_{n-1})$ represents a group of integrators which are required to generate increments of the function $f(x, y, y_1, \cdots y_{n-1})$ from the changes of the variables $x, y, y_1, \cdots y_{n-1}$. These increments which also represent increments of the highest derivative are fed back to the integrators which carry out the successive integration. This important link which "closes the loop" would have been lost in the detailed integrator interconnection but it is appropriately emphasised in this block diagram representation.

It is to be noted from the above diagram that all interconnections are represented by changes (or increments) of the variables. It would now be appropriate to stress the fact that the transfer of information between integrators of a digital differential analyser can only take place in increments, (e.g. the +1, 0, or -1 increments in "ADA"). This is quite different from the "whole number" operation of general-purpose digital computers, and to stress this point D.D.A.'s are often referred to as "incremental computers".

The term "mapping" is used for the specification of the integrator interconnections. Before this can be done it is often necessary to express the equations in "incremental form". This involves either taking increments of each term or multiplying throughout by dx, the independent variable. For Bessel's equation:-

$$y'' + \frac{1}{x} y' + y = 0$$

the incremental form would become

$$dy' + d(\frac{1}{x} y') + dy = 0$$

when increments of each term are taken, and

$$dy' + \frac{1}{x} dy + y\ dx = 0$$

when the equation is multiplied by dx. In general, there does not seem to be very much difference between the complexity of the integrator interconnections based on either incremental form. In the above example, the

latter form may be preferred because of the elimination of the second derivative and also because the $d(\frac{1}{x} y')$ term in the former equation will produce an extra term when it is expanded using the udv + vdu form of multiplication.

In some cases, the mapping of a problem depends largely on the behaviour of the functions throughout the desired range of the independent variables. For example, if Bessel's equation is to be solved for x within the range $0 \leqslant x \leqslant 1$, the term $\frac{1}{x}$ would tend to infinity in the region of $x = 0$ and the scaling for the generation of $\frac{1}{x}$ would be impossible. To overcome this difficulty, the equation is multiplied through by x and solved in the implicit form:-

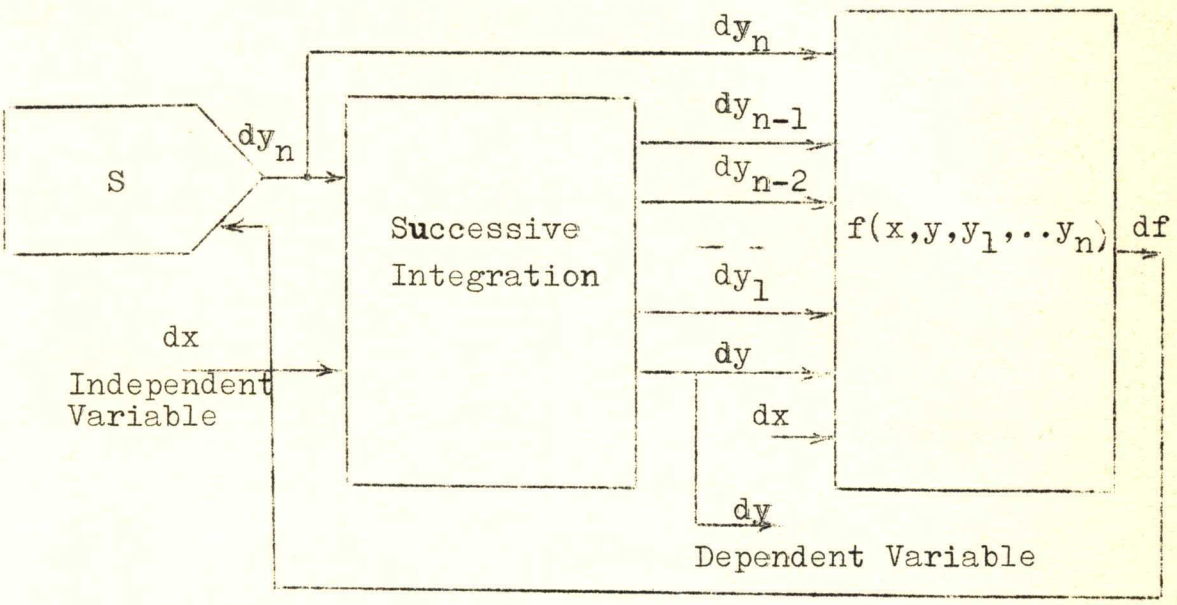
$$x\,dy' + dy + xy\,dx = 0.$$

as will be described later, a "servo" integrator will be used in this application.

When the highest derivative cannot be conveniently expressed in terms of the other variables, or when it is undesirable to do so because of scaling difficulties (as in the above example), "servo" integrators may be used to solve the equation in its implicit form. The term "servo" is applied to "decision" integrators when they are used in special "feedback" interconnections. The variations of a function from its desired value are fed into the integrand of a decision integrator which continually adjusts one variable of the function so that these variations are minimised. For the implicit equation:-

$$f(x, y, y_1, \ldots y_n) = 0$$

the block diagram representation of the D.D.A. solution using a servo integrator is as follows:-



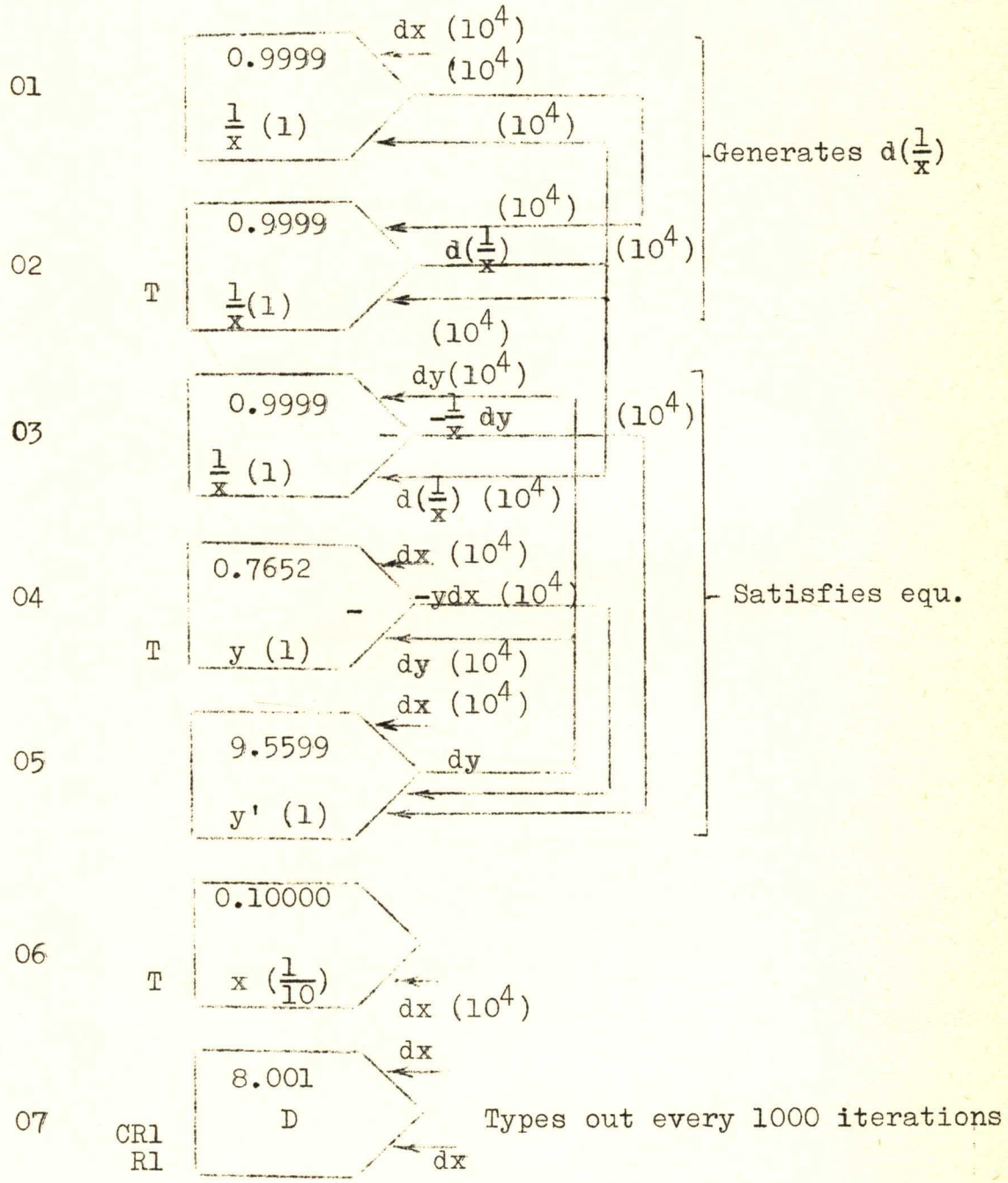
The integrator marked "S" is the decision integrator used in the above "servo" application.

Examples of the mapping required for the solution of differential equations is shown on pages 35 and 36. The map on page 36 illustrates the use of the servo interconnection. Other maps for the generation of functions and for the solution of various forms of differential equations are given in the appendix.

EXAMPLE:- Solution of Bessel's equation for the tabulation of $J_o(x)$ for x = 1.0 (0.1) 10.0

Incremental Form :- $dy' = -\frac{1}{x}\,dy - y\,dx$

Initial Conditions:- x = 1.0000, y = 0.7652, y' =-0.4401



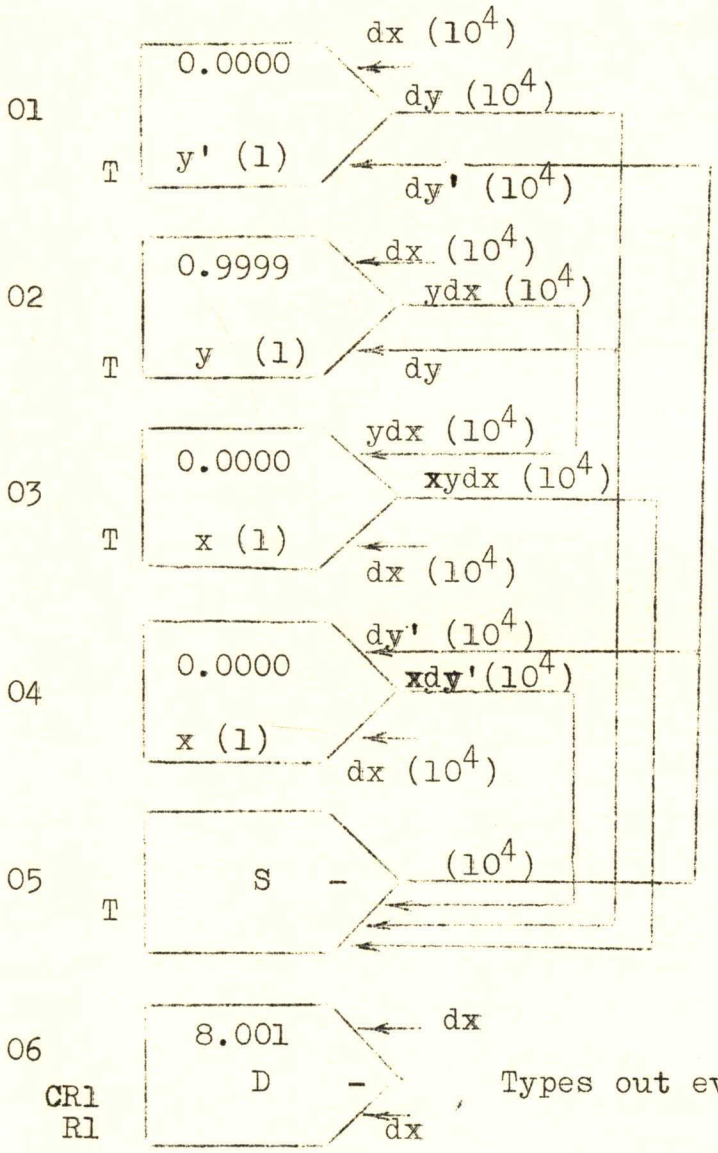
Coding:-

```
@01@60/@02/@@001100@09999 0000@
@02@01/@02/@@006150@09999 0000@
@03@05/@02/@@006100@09999 0000@
@04@60/@05/@@006150@07652 0000@
@05@60/@03/04/@@001150@95599 0000@
@06@@60/@@001150@010000
                          000@
@07@60/@60/@@169100@8001 00000@
```

EXAMPLE:- Solution of Bessel's equation for the tabulation of $J_0(x)$ for x = 0.0 (0.1) 1.0

Incremental Form:- $x\,dy' + dy + xy\,dx = 0$

Initial Conditions:- $x = 0,\ y = 1.0,\ y' = 0$



Types out every 1000 iterations

Coding:-

```
@01@60/@05/@@001150@00000 0000@
@02@60/@01/@@001150@09999 0000@
@03@02/@60/@@001150@00000 0000@
@04@05/@60/@@001100@00000 0000@
@05@60/@01/03/04/@@009150@00000
                                  0000@
@06@60/@60/@@169100@8001 00000@
```

2.2.2 Scaling.

Scaling consists of assigning scale factors to problem variables so that the corresponding numbers may be satisfactorily processed by the computer to the required accuracy. The scale factor relationships which must exist between the computer variables are derived in the appendix and are reproduced below.

$$(1) \quad M K_Y |Y_p| < 1$$

$$(2) \quad K_z = M K_Y K_x$$

$$(3) \quad K_y = 10^N K_Y$$

where M = multiplier
K_Y = scale factor of the integrand
$|Y_p|$ = maximum absolute value of the problem variable
K_z = scale factor of δz
K_x = " " " δx
K_y = " " " δy
N = integer such that $1 \leqslant N \leqslant 6$

Equation 1 ensures that the problem variables are correctly fitted to the numerical range of the machine. The term on the left side of the equation must be made as close to "1" as possible. In this way integrands work near their maximum capacity and maximum accuracy is obtained. Equation 2 satisfies the scale-factor relationship between the inputs and output of each integrator. Equation 3 ensures that increments are added to the desired least significant place of the integrands. The relationship between the machine variables will correspond to the relationship between the problem variables when the above relationships are satisfied for each integrator used in the problem solution. Scale factors, accuracy and computing time are inter-related, and the object of optimum scaling is of course to produce the maximum accuracy in the shortest time.

2.2.3 Coding.

Coding is the process of converting the scaled integrator maps (such as those on pages 35 and 36), into a form which can be entered into the machine. The input mechanism of the machine has been designed to accept information in a certain form or code and all problems must be converted to this standard form before it can be punched on paper tape and then read into the computer by the paper tape reader. From the coding sheets shown in the appendix, it can be seen that the coding for each integrator consists of the integrator number, ΔX integrator, ΔY integrator(s), ΔI integrator, fixed instructions and initial conditions.

Examples of coding are shown at the bottom of pages 35 and 36, and on page 39. This latter example represents a program written to check all functions of the digital differential analyser "ADA".

TEST PROGRAM.

```
@06@60/@@@201100@00070 0000@
@07@60/@@@701100@00070 0000@        Partial & Complete Reset
@08@@06/@@001150@000 000000@
@09@@07/@@001150@000 000000@

@10@70/@@@001100@0499999 00@
@11@10/@60/@@101200@00 00000000@
@12@10/@60/@@101400@00 00000000@
@13@10/@60/@@101100@00 00000000@
@14@10/@60/@@101300@00 00000000@    Integration Mode
@15@@11/@@001150@0000 00000@
@16@@12/@@001150@0000 00000@
@17@@13/@@001150@0000 00000@
@18@@14/@@001150@0000 00000@

@25@60/@@@001100@01000 0000@
@26@60/@@@002100@01000 0000@
@27@60/@@@003100@01000 0000@
@28@60/@@@006100@01000 0000@
@29@60/@@@007100@01000 0000@
@30@60/@@@008100@01000 0000@       Multiplier
035@@25/@@001150@0000 00000@
@36@@26/@@001150@0000 00000@
@37@@27/@@001150@0000 00000@
@38@@28/@@001150@0000 00000@
@39@@29/@@001150@0000 00000@
@40@@30/@@001150@0000 00000@

@41@10/@10/@@004100@8750 00000@
@42@@41/@@001150@0000              Decision Integrator
                    00000@

@00@60/@@@004100@01000 0000@
@01@60/@@@009100@01000 0000@
@02@@00/25/@@001150@00000 0000@
@03@@01/26/@@001150@00000 0000@       Y Input
@31@60/@@@004100@01000 0000@
@32@60/@@@009100@01000 0000@       Sign Reversal
@33@@06/31/@@001150@00000 0000@
@34@@07/32/@@001150@00000 0000@

@05@@@60/@201150@00000 0000@       Addition into I

@04@@60/@@001150@00000 0000@       Counter

@43@60/@60/@@119100@89 00000000@   Resets every 2 iterations
@44@60/@60/@@229100@8801 00000@      "     " 200   "
@45@60/@60/@@389100@8501 00000@    Types out every 500   "
@46@60/@60/@@449100@8501 00000@    Calls in tape every 500  "

@19@@@@401150@0012345 0
.0123456 0
.0234567 0
.0345678 0
.0456789 0                         Call in
.0567890 0
.0678901 0                         Empirical Data
.0789012 0
.0890123 0
.0901234 0
.....
```

2.3 A BRIEF DISCUSSION OF SELECTED EXAMPLES.

2.3.1 Differential equations with 2-point boundary conditions.

A method has been described for the solution of 2-point boundary value problems on analogue computers. (Ref.65). The problem is converted into two initial value problems; the analogue computer solves both of these and combines the results to yield a solution of the original boundary value problem.

Another approach is possible using the decision integrator and reset control functions of a digital differential analyser (Ref.39). Unknown conditions at one point are assumed and the computation is carried out towards the second point. The difference between the actual and desired conditions at this point are used to modify the assumed initial conditions. This procedure is carried out automatically using decision integrators and reset controls and the solution is made to converge upon that which satisfies both conditions to the required accuracy.

A program was written to verify the above technique. The problem chosen was the solution of the differential equation:-

$$\frac{d^2y}{dx^2} + \frac{dy}{dx} - 6y = 0$$

with the conditions $x = 0, y = 3; \quad x = 1, y = 7.5$

The formal solution of this very simple problem is of course

$$y = e^{2x} + 2e^{-3x}$$

The results calculated from this equation were to be used to check the computer program.

2.3.2 Partial Differential Equations.

When the derivatives of a partial differential equation are replaced by their finite-difference-approximations, the equation is transformed into either systems of algebraic equations or systems of ordinary differential equations. In either case the resultant

equations may be programmed for their solution on differential analysers (Ref.41,51,52).

To illustrate the above application of D.D.A.'s, the following one dimensional heat flow problem was formulated and programmed for its solution on "ADA":-

An insulated metal rod 1 metre long is initially at 50°C. At t = 0, its ends are suddenly brought to 0°C and 20°C and held at these temperatures. If the diffusivity α^2 = 1.77 cm^2/sec, find the temperature distribution at any later time.

The partial differential equation for heat flow in one dimension is:-

$$\frac{\partial u}{\partial t} = \alpha^2 \frac{\partial^2 u}{\partial x^2}$$

where u is temperature (°C), t is time (sec), x is distance (cm) and α^2 is the diffusivity (cm^2/sec). If the rod is divided into 10 equal segments and the temperatures at the 9 equally spaced stations denoted by u_j (j = 1, 2,..9), the partial differential equation may be approximated by the following set of 9 simultaneous ordinary differential equations

$$\frac{du_j}{dt} = \alpha^2 \frac{u_{j+1} - 2u_j + u_{j-1}}{(\delta x)^2}$$

where δx = 10 cm.

There have been many contributions to the technique of solving partial differential equations on both analogue and digital machines. It is not the author's intention to discuss these in detail in this thesis. The above example was given to illustrate the possible extensions of the use of D.D.A.'s using the methods of numerical analysis.

2.3.3 Interpolation formulae.

ADA is provided with means for filling any integrand with values from the input tape. If linear interpolation is used to introduce an empirical function,

the first differences of the function are entered into the integrand of an integrator whose primary incremental input is the independent variable.

If the intervals between tabular values of an empirical function are very long, some higher order interpolation formula may be necessary to provide the required accuracy.

Any interpolation formula may be used. However, it is desirable to arrange the integrator interconnections so that it is necessary to input only tabular values of differences of the highest order (i.e. to input only 1 value for each interval of interpolation). The lower order differences can be determined from the preceding values of the next higher order differences during the computation.

The above remarks do not apply if the empirical function contains a point of discontinuity. At a point of discontinuity, the appropriate values of all the differences must be reset into the chain of integrators representing the interpolation formula.

Because of the desirability of entering only one value for each interval of integration, the use of Newton's forward difference interpolation formula has been found to result in the simplest integrator interconnections.

Figure 2 represents the integrator interconnections required to enter empirical functions using Newton's forward difference interpolation formula correct to second differences. Using the normal notation of numerical analysis this formula is as follows:-

$$f(x_o + \theta\ \delta x) = f_o + \theta\ \Delta f_o + \frac{1}{2!}\ \theta(\theta-1)\ \Delta^2 f_o + \text{----}$$

Considering the variation of θ from 0 to 1, the above equation may be expressed in incremental form as

$$df = \Delta fo\ \ d\theta + \Delta^2\ fo\ (\theta - \tfrac{1}{2})\ d\theta + \text{----}$$

It may be seen from the figure that only the second differences are entered from the tape. The I register of integrator 2 contains the quantity $\Delta f_o + \Delta^2 f_o\ \theta$.

When $\theta = 1$, this quantity is equal to f_1 since:-

$$\Delta f_1 = \Delta f_o + \Delta^2 f_o$$

Hence, the first differences are evaluated during the computation from the preceding values of the second differences. The new values of the differences are reset into the integrands at regular intervals. The above technique may be extended to higher order differences.

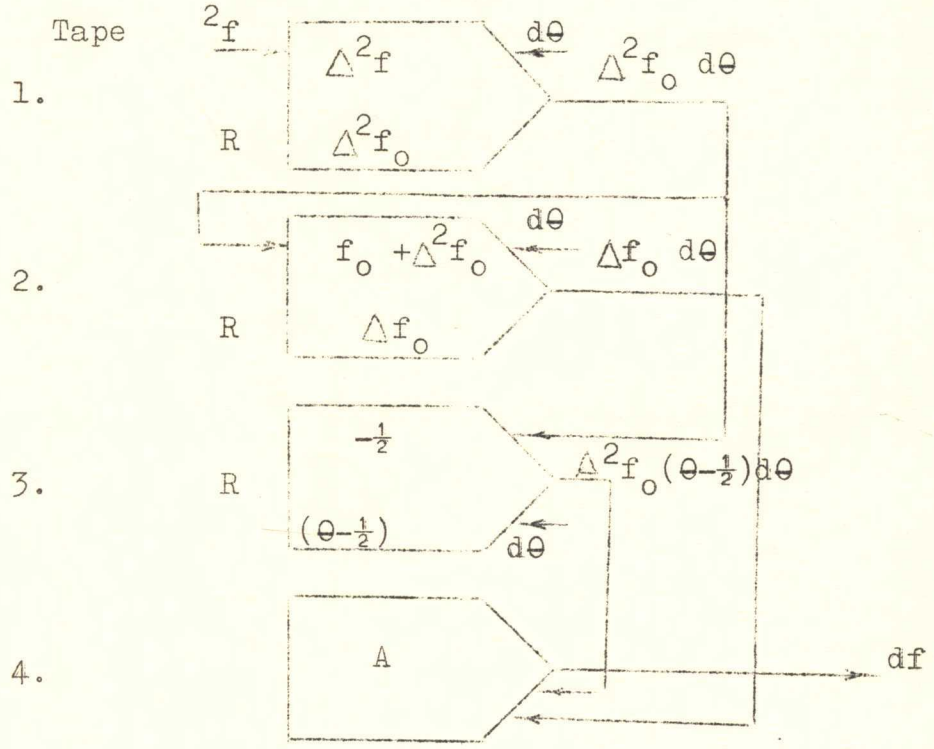


Fig 2.

Programs for Newton's formula to second and third differences and Bessel's formula to second differences have been written for the sub-tabulation of functions.

2.3.4 Fourier Analysis.

The fourier coefficients of a periodic function which is either generated within the computer or entered using an interpolation formula may be evaluated by a D.D.A. program. The coefficients

$$a_o = \frac{1}{\pi} \int_0^{2\pi} f(x)\, dx$$

$$a_n = \frac{1}{\pi} \int_0^{2\pi} f(x) \cos n x\, dx$$

$$b_n = \frac{1}{\pi} \int_0^{2\pi} f(x) \sin nx\, dx$$

may be evaluated by generating sin n x, cos n x within the computer and carrying out the appropriate integrations over one complete cycle.

2.3.5 Time Lag (or delay) Function.

The representation of true time lag involves the generation of the function f (t - τ) when the function f(t) is known.

There are several methods for the representation of time lags on analogue computers (Ref.42,53). These methods which are based on mathematical approximations, can also be used for the representation of time lags on the digital differential analyser.

An entirely different method utilising the initial conditions registers can also be used on digital differential analysers. This method involves three stages:-

(a) the evaluation of finite differences of the function.

(b) the storage and transfer of the differences down a chain of integrators (and thus introducing the delay)

(c) the regeneration of the function using some interpolation formula.

2.3.6 The Evaluation of a Double Integral.

The moment of inertia of the area in the first quadrant bounded by the parabola $y^2 = 1 - x$ and the coordinate axes, about an axis perpendicular to the xy plane at (1,0) is represented by the double integral.

$$\int_0^1 \int_0^{1-y^2} \left\{ (x-1)^2 + y^2 \right\} dx\, dy$$

and this has the value $\frac{44}{105}$. ($\div$ 0.419)

The evaluation of a double integral on a D.D.A. requires the approximation of the double integral by the

sum of a number of single integrals. Such an approximation for the above example is:-

$$\sum_{y = y_1}^{y = y_m} \left[\int_0^{1-y^2} \left\{ (x-1)^2 + y^2 \right\} dx \right] \Delta y$$

where for example $y_1, y_2 \ldots y_m = 0, 0.01, 0.02 \ldots 0.99$.

and $\Delta y = 0.01$.

To illustrate the procedure, a program was written to evaluate the 100 single integrals represented by above expression. For a computing time of 3 minutes, a value differing by only 1% of the correct value was obtained.

2.3.7 Miscellaneous.

A wide variety of problems received some attention during the course of investigations on the capabilities of the D.D.A. Some of these are listed below:-

Differential equations.

Highest derivative known explicitly
linear with constant coefficients
highest derivative not known explicitly
simultaneous d.e.'s
d.e. of Bessel, Tschebyschef, Riccati, La Guerre, Hermite
heat flow in 1 and 2 dimensions
wave equation
transients in multiple mesh circuits

Algebraic equations.

simultaneous linear
implicit with 2 or 3 variables

Algebraic functions.

forms of multiplication and division
integral and non-integral powers
exponential, logarithm
reciprocal, square root
variable power of a variable

Trigonometric and Inverse Trigonometric functions.

sin, cos, tan, cosec, sec, cot.
$\sin^{-1}$, $\cos^{-1}$, $\tan^{-1}$, cosec^{-1}, $\sec^{-1}$, $\cot^{-1}$

Hyperbolic functions.

sinh, cosh, tanh, cosech, sech, coth

Functions of a complex variable.

complex integration
complex exponential
complex sine and cosine

Transformation of co-ordinates.

polar to cartesian
cartesian to polar

Programs for checking operation of plotter.

epicyclic curve
equiangular spiral
lissajous' curves

Interpolation formulae.

linear
Newton's to second differences
Newton's to third differences
Bessel's to second differences

Generation of functions using decision integrators.

Absolute value
maximum and minimum limits
ramp
triangular and saw-tooth waves
square waves (Y register value only)
dead band
hysterisis or backlash
capacity limiter
time lag or delay function

Test Programs.

Programs for checking the operation of all machine functions.

D.D.A. REQUIREMENTS OF THE SYSTEM OPERATIONAL PROBLEM.

In October, 1956, it was suggested that the specification of the D.D.A. for the Snowy Mountains Authority should be based on the computer requirements of certain system operational studies (Ref.34). These studies were described as:-

> "The detailed monthly step by step storage operational studies required to determine the most favourable methods of storage and hydro plant operation and for the assessment of the relative merits of the various alternative forms of development being contemplated."

A brief discussion of these problems was given in Chapter I and an example of the manual calculations are presented on pages 25 to 26.

When the author joined the Authority in November 1956, his immediate responsibility was to examine the problem in detail and to determine the number of integrators required by the DDA for its solution. Details of the integrator interconnections were presented in an official report (Ref.36), but only some of the more salient points will be discussed in this thesis.

ARITHMETIC OPERATIONS ON WHOLE NUMBERS.

In the section on basic concepts of the D.D.A., it was stressed that the D.D.A. was an incremental computer and that the only possible transfer between integrators was in the form of +1, 0 or -1 increments. When arithmetical operations on whole numbers are required, these must take place one increment at a time. It must therefore be expected that the computing time is very long even for low accuracy calculations.

In the system operational problem, many expressions were to be evaluated from whole numbers. An example of these is:-

$$S_m = Q_i + S_o - S_c$$

where Q_i = inflow during month

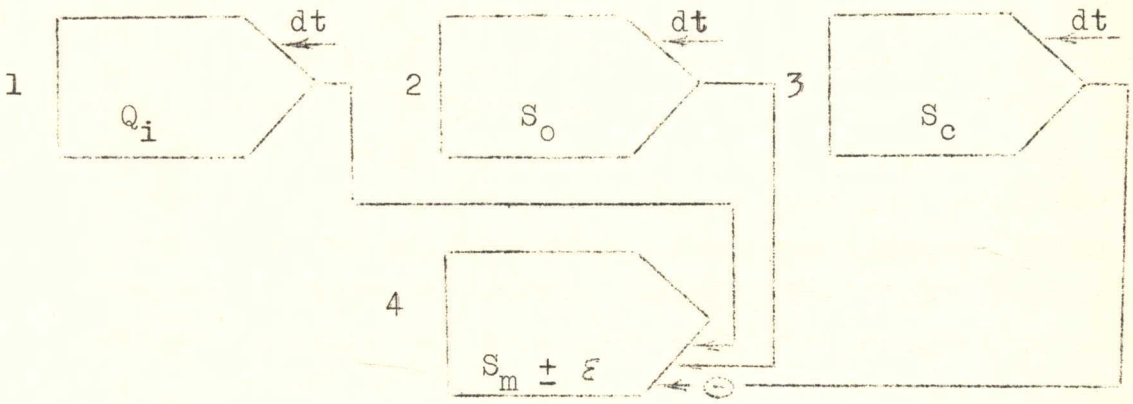
S_o = storage at end of last month

S_c = storage mark at end of month

and S_m = release determined by mark operation.

The 3 quantities Q_i, S_o, and S_c were either held in or entered into the integrands of integrators. The quantity S_m had to be evaluated before the operation for the month could commence as it was used in operating instructions to determine the actual release.

Consider now the following interconnections:-

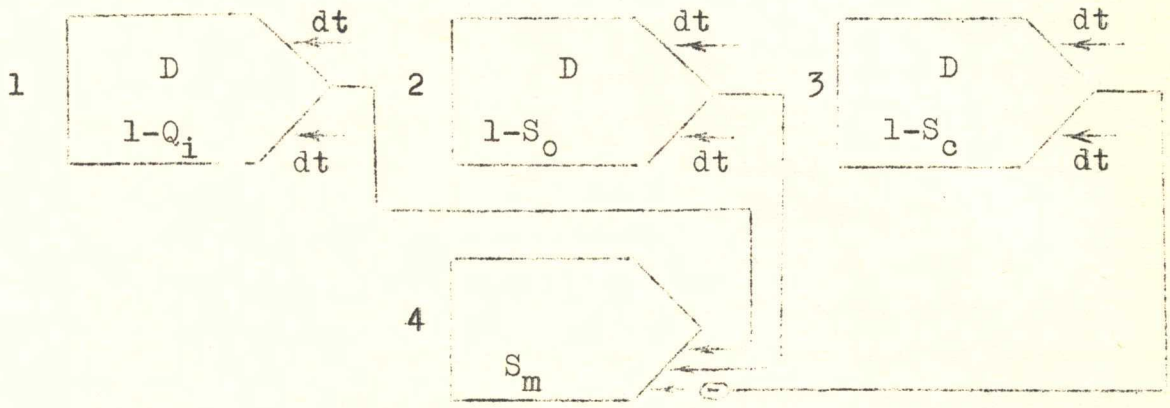


where Q_i, S_o, S_c, S_m have been suitable scaled so that each is less than unity.

After 1000 increments of the independent variable dt have entered integrators 1, 2 and 3, 1000 $Q_i \pm 1$, 1000 $S_o \pm 1$ and 1000 $S_c \pm 1$ increments respectively will be produced at the outputs. These cannot be specified exactly as they depend upon the actual values of Q_i, S_o and S_c and the initial values of the R registers. If these increments are added into the 3rd decimal place of the integrand of the 4th integrator (with the appropriate sign reversal as shown), after 1000 iterations the value of this integrand will become $S_m \pm 0.003$ Operating at 50 iterations per second, this simple arithmetic operation will take 20 seconds to perform.

The inaccuracy mentioned above may be eliminated

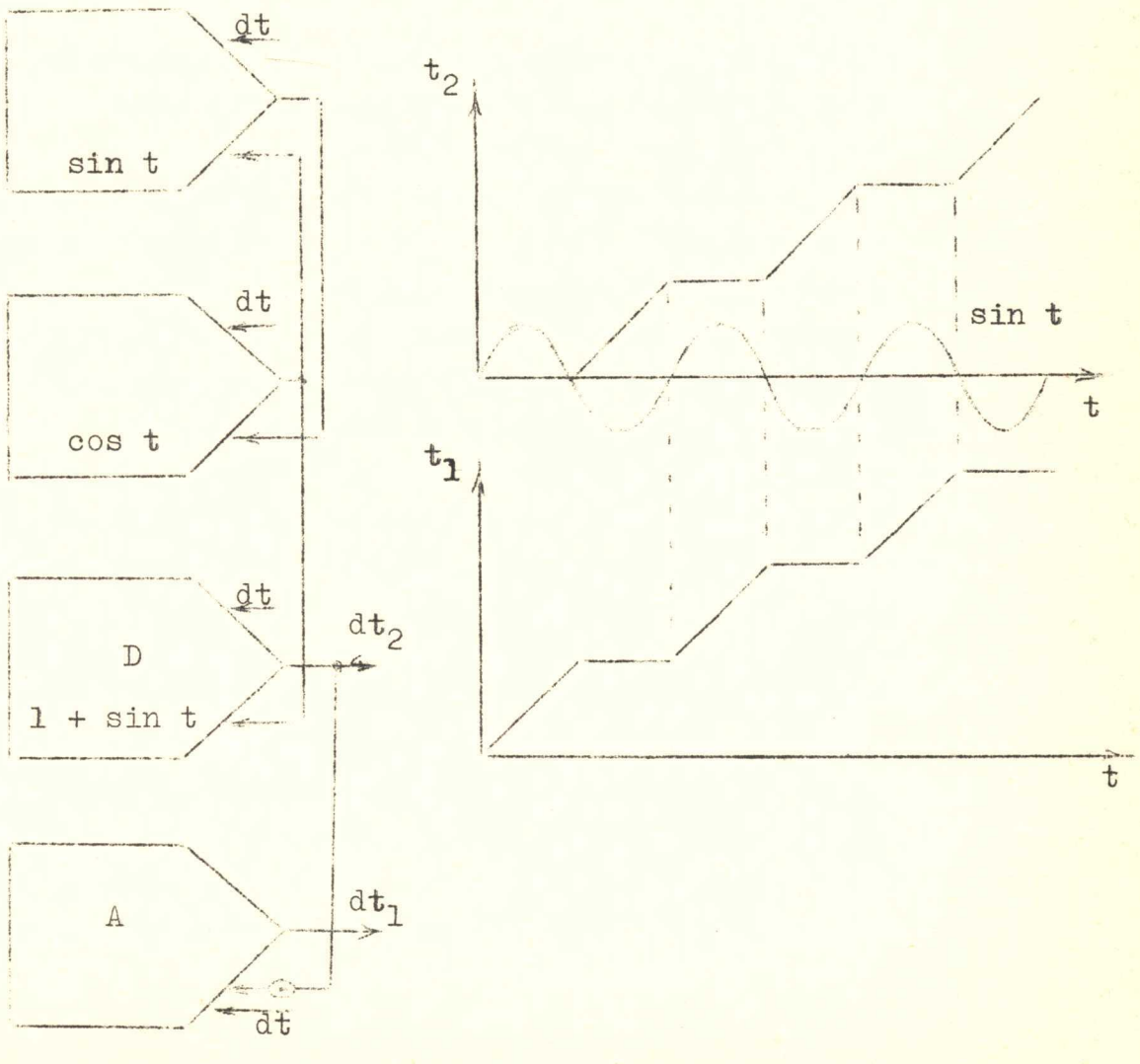
by the use of decision integrators as shown below:-



If for example $Q_i = 0.295$, the integrand of integrator 1 is initially 0.705 and after 295 dt inputs, integrator 1 will stop conducting. Integrators 2 and 3 operate similarly and after 1000 iterations, S_m will be formed to 3 figure accuracy in integrator 4.

In many operating instructions in the problem, it was necessary to take the minimum or maximum of several numbers. To carry out this operation, increments of all numbers are passed into an array of decision integrators which block the passage of all increments except those from the maximum or minimum number. The first program written to carry out this operation was designed so that the integrands of the decision integrators were set up by the increments as they passed through the array. Accuracy considerations for near-equal numbers would have made this program almost useless. A second program designed to set up the integrands before the passage of increments was proposed. Detailed descriptions of these programs will not be given in this thesis. At the time of writing, it was felt that an undue amount of emphasis had been placed on the "ease" of D.D.A. programming and too much effort had been directed towards the extension of the use of the D.D.A. into applications for which the computer had not been designed. Complications to the solution of the system operational problem were introduced by the use of the D.D.A. because this computer had then to be programmed to carry out arithmetic operations on whole numbers. It may however be appropriate to mention at this stage that when the above investigations were carried out little was known of the capabilities of the D.D.A. and the author had no experience with general-purpose digital computers.

In the system operational problem, a number of modes of operation were possible for each station and the correct mode was selected according to pre-assigned operating instructions. It had been suggested that as decision integrators could carry out switching functions they could be easily programmed to carry out this selection. This is not strictly correct because of the inherent limitation that all operations in a D.D.A. take place on increments rather than on whole numbers. This difficulty was overcome by arranging for the computation to take place in 2 phases. In the first phase, arithmetic operations such as taking the smallest of 5 numbers were carried out to select the correct mode of operation. In the second phase, the operation for the month was carried out with all variables under the influence of system parameters such as installed capacities and maximum storages. At the end of this phase, the required results such as end-of-month storage, spill and energy production were produced and typed out. The integrator interconnection required to carry out this 2 phase computation is as follows:-



The variable dt_1 is used as the independent variable for all integrators in the phase 1 computation. During this phase $dt_1 = dt$ (the machine independent variable) and $dt_2 = 0$. Similarly the integrators in the phase 2 computation have dt_2 as their independent variable and during this phase $dt_2 = dt$ and $dt_1 = 0$.

The number of integrators required for the solution of the entire system operational problem was found to be about 380. Of these only 65 were normal integrators and these were used mainly for the generation of functions. The others were used as input integrators, type-out integrators, decision integrators, adders and constant multipliers. The number of empirical functions (which were to be entered into the computer) was found to be about 40; and the number of output variables which were to be typed out was about 50. When the required number of integrators was known investigations were carried out to determine whether the construction of such a large computer was feasible. The use of an English Electric "Deuce"-type magnetic drum was proposed, and the logical design of a magnetic core memory for the integrator-output-store was commenced. However from an analysis of the utilisation of the integrator functions, it was suspected that the D.D.A. was not well suited for the solution of this particular problem, which had been formulated, not as a system of ordinary differential equations but as a long sequence of simple arithmetic operations on whole numbers. At this stage, no experience with general-purpose digital computers had been obtained and no immediate recommendations could be made. When SILLIAC was made available to engineers of the Authority it was decided to program this computer for the solution of the system operational problem. The investigations using SILLIAC will be described in the next chapter.

CHAPTER 3. INVESTIGATIONS USING THE GENERAL PURPOSE DIGITAL COMPUTER SILLIAC.

3.1 THE SOLUTION OF THE SYSTEM OPERATIONAL PROBLEM ON SILLIAC.

3.1.1 Summary.

3.1.2 A Brief Description of the Program.

3.1.3 Advantages over the D.D.A. solution.

3.1.4 Program Structure.

3.1.5 Examples of Sub-routines.

3.1.6 Silliac Results.

3.2 THE SIMULATION OF A DIGITAL DIFFERENTIAL ANALYSER ON A GENERAL-PURPOSE DIGITAL COMPUTER.

3.2.1 Introduction.

3.2.2 A Brief Description of the Simulation Routine.

3.2.3 Coding of Fixed Instructions.

3.2.4 Program Structure.

3.2.5 Tape Preparation.

3.2.6 An Example.

CHAPTER 3. INVESTIGATIONS USING THE GENERAL PURPOSE DIGITAL COMPUTER SILLIAC.

3.1. THE SOLUTION OF THE SYSTEM OPERATIONAL PROBLEM ON SILLIAC.

3.1.1 Summary.

A program for SILLIAC was written for the System Operational Problem mentioned in Chapters 1 & 2. The object of writing this program was to obtain some indication of the relative merits of SILLIAC and a digital differential analyser for solving this problem.

Some of the results of a trial study extending over a period of only one year are presented on pages 63 and 64.

The inflows used in the study were chosen so that many different conditions would be met and the computer would test the program as thoroughly as possible. In many cases, values were assumed for the constants and data used in the study, but these were estimated as carefully as possible, and the actual values do not affect the writing and checking of the program.

Calculations using a slide-rule, four figure logarithms and a Facet desk calculating machine were carried out to check the SILLIAC program. These manual calculations which were presented earlier in Chapter 1 of this thesis were found to agree extremely well with the SILLIAC results.

The conclusion was reached that SILLIAC was ideally suited for the solution of the system operational problem.

3.1.2 A Brief Description of the Program.

The SILLIAC program uses 842 orders excluding subroutines and parameters. Because of the relative ease of presentation in this thesis, the "flow diagram" of the program has been represented by the "program structure" on pages 57 to 59; this is simply an itemised list of the specification of all sections of the program. Examples of the detailed coding of these sections are given on pages 60

to. 62. Branching in the flow diagram is represented by instructions to jump from one section to another.

All load curve data and system parameters were arranged to be stored permanently in the memory. The program then called in the monthly inflows, carried out the computation and then punched out the end of month storages. This procedure was repeated until 12 months' records had been processed, when the annual totals of demand energy, fuel replacement energy, spillage loss and various flows in the system were also punched out. The inflows for the first month of the next year were then called in and the process was repeated until the complete analysis (covering the 50 year period) had been carried out.

The SILLIAC results for one year's operation is shown on page 63.

The program has the feature that if any of the storages fail, the amount by which the storage fails is punched out together with the minimum storage to distinguish it from other storages which may fail in the same month. The results on page 63 show a failure of Kosciusko Reservoir.

Details of the monthly operation may be obtained by changing one order of the program. When this is done the results are in the form shown on page 64. Although these results are not normally required, they were found to be particularly useful in using the manual calculations to check the program.

Each step in the manual calculations had been specified in the heading of each column in the example on pages 25 and 26. The routines required to carry out these steps are extremely simple. However, a program based on the compilation of these simple routines would have been extremely long and inefficient. Most of the programming effort was directed at the organisation of the program so that similar operations were carried out by the same group of orders or subroutines.

For each year of the analysis it was estimated that the computer had to perform 15,000 orders (of which 400 were multiplications or divisions) and 60 integral

roots. From these figures, the computing time for 50 years of the analysis, would be about 3 minutes.

The number of monthly inflows was 15 and each of these was punched to 5 figure accuracy. Allowing for space and carriage return characters etc., it was estimated that the computer had to input 60,000 characters for the complete 50 year problem. With a tape reader speed of 200 characters/sec., the input time would be about 5 minutes.

The output quantities include the monthly storage values for 6 reservoirs, annual totals of various flows and the demand energy, fuel-replacement energy and spillage loss for 9 groups of stations. Each of these was punched to 5 figure accuracy. Allowing additional characters for signs, decimal points, carriage returns and figures to specify the year and month it was estimated that the number of output characters would be about 1000/year. With an output punch speed of 60 characters/second, the 50,000 output characters for the 50 year problem would take about 15 minutes.

A summary of the estimated times for the 50 year problem is as follows:-

Input Time	:	5 minutes
Computing Time	:	3 minutes
Output Time	:	15 minutes

3.1.3 Advantages over the D.D.A. solution.

The system operational problem had been formulated as a sequence of arithmetical operations on whole numbers. One advantage of SILLIAC over ADA lies in the ease with which SILLIAC can perform the basic arithmetical operations. The addition of two whole numbers in a D.D.A. is a very tedious process as each increment of the number must be counted out.

Another advantage lies in the extensive use of subroutines. In G.P. computers, the same subroutine may be repeatedly used by the main program. No equivalent technique exists for the D.D.A. method of solution, and it is often necessary to repeat groups of integrators for the generation of a particular function required by the problem equations.

Less punching of input data will be involved for the SILLIAC program. All load curve data and storage marks may be punched once and stored in the computer's memory. This is not possible with the D.D.A., and as each quantity is required every month, 12 x 50 = 600 values must be punched together with their destination integrators.

Optimum scaling for the D.D.A. may prove troublesome, as computational time increases with accuracy requirements. In particular, to obtain an additional decimal place, the computational time is increased 10 times. With SILLIAC this is not the case. The arithmetical operations will be carried out to full accuracy, but only the significant portions of the answers will be punched out.

PROGRAM STRUCTURE.

1. Make $S_1 = 0$ for checking
   $S_1 = \frac{1}{2}$ " computation

2. Input constants.

3. Clear locations for annual outputs.

4. Input monthly inflows.

5. From counter, determine year, month and season.

6. Transfer appropriate load curve data into temporary storage.

7. Transfer appropriate storage marks into temporary storage.

8. From operating instructions, find the desired release from Kosciusko.

9a. If the end of month storage is less than the minimum storage punch out amount by which Kosciusko fails & make the end of month storage equal to the minimum storage.

9b. If the desired end of month storage is greater than the maximum storage, increase the release from Kosciusko so that end of month storage is the maximum storage.

10. Determine energy equivalent of inflows to M1A, M1B and M2L.

11. From operating instructions, find the desired diversion from Gungarlin to M3.

12. If the desired end of month storage is greater than the maximum storage, increase the diversion from Gungarlin.

13. If the desired diversion is greater than the aqueduct capacity, the actual diversion is made equal to the aqueduct capacity.

14. Determine end of month storage and spill at Gungarlin.

15. Determine energy equivalent of total inflow to M3.

16. Determine total inflow to Jindabyne.

17. Determine demand energy, fuel replacement energy and spillage loss for stations M4, M5H, M2H, M1A, M1B, M2L, and M3. Add these values to the annual totals.

18. Determine the total demand energy for the month from the above stations.

19. From operating instructions determine the desired release from Jindabyne.

20. If the end of month storage is less than the minimum storage punch out amount by which Jindabyne fails & make the end of month storage equal to the minimum storage.

21 If the desired end of month storage is greater than the maximum storage, increase the release from Jindabyne so that the end of month storage is the maximum storage.

22. Determine energy equivalent of inflow to M6 - 7, the demand energy, fuel replacement energy and spillage loss for station M6 - 7, and add these to the annual totals.

23. Determine diversion from Tantargara to Adaminaby, end of month storage and spill at Tantargara.

24. From the operating instructions, find the desired diversion from Adaminaby Reservoir to Tumut Pond or from Tumut Pond to Adaminaby Reservoir.

25. If the end of month storage of Tumut Pond is less than the minimum storage, print out amount by which Tumut Pond fails.

26. If the desired end of month storage of Tumut Pond is greater than the maximum storage, calculate the fuel replacement energy and spillage loss of the T stations. Add these quantities to the annual totals.

27. Calculate end of month storage and spill at Adaminaby.

28. Punch out year, month and end of month storages.

29. If $S_1 = 0$ go to step 33,
if $S_1 = \frac{1}{2}$ go to step 30.

30. Check for end of year.

If yes, go to step 33,
if no, go to step 31.

31. Check for end of operation.

If yes, go to step 33,
if no, go to step 32.

32. Add 1 to counter and go to step 4.

33. Punch out "annual" totals.

34. Check for end of operation.

If yes, stop,
if no, go to step 35.

35. Add 1 to counter and go to step 3.

DETERMINATION OF "YEAR", "MONTH" AND "SEASON" FROM COUNTER.

Location a contains an integer "n" which represents a certain date in the following way:-

n = 05 x 12 + 0 = 60 represents January '05
n = 05 x 12 + 11= 71 " December '05
n = 06 x 12 + 0 = 72 " January '06
n = 06 x 12 + 11= 83 " December '06 etc.etc.

It is required to determine the year ('05 or '06 in the above examples) and the month (1,2.....12) which the integer n represents, and to record these quantities in locations y and m respectively. The four seasons Summer (Dec. Jan. Feb.), Autumn (Mar. Apr. May), Winter, (June, July, Aug.) and Spring (Sept. Oct. Nov.)are represented by the numbers 0, 1, 2, and 3; this quantity representing the season of the year is to be determined and recorded in location s.

```
0       51      aF      ┐-Integer division
        00      1F      │
1       66      "12"    │ Integer 12
        10      1F      ┘
2       40      mF        (m) = 0,1,.....10 or 11
        F5      mF
3       40      mF        (m) = 1,2,.....11 or 12
        S5      F             = "months" Jan. -to- Dec.
4       40      yF        (y) = "year"
        F5      aF
5       40      bF        Location b temporary storage
        51      bF      ┐-Integer division
6       00      1F      │
        66      "12"    │ Integer 12
7       10      1F      ┘
        40      bF        (b) = 1,2,...11, 0 = months Jan.
8       51      bF      ┐- Integer division     to Dec.
        00      1F      │
9       66      "3"     │ Integer 3
        10      1F      ┘
10      S5      F
        40      sF        (s) = "season" (0,1,2 or 3)
```

DETERMINATION OF DEMAND ENERGY, FUEL-REPLACEMENT ENERGY AND SPILLAGE LOSS.

Seven values of Ei are given in locations a to a + 6
" " " Edm " " b to b + 6
" " " Em " " c to c + 6
" " " Eo " " d to d + 6
" " " Ef " " e to e + 6
" " " Es " " f to f + 6

For each set of Ei, Edm and Em values, corresponding values of Eo, Ef and Es are to be computed (according to the equations shown below), and then added to the totals in the locations shown above. The seven values of Eo are to be added together and recorded in location g.

If Ei $\leq$ Edm then Eo = Ei, Ef = Es = 0

If Edm $<$ Ei $\leq$ Em Eo = Edm, Ef = Ei - Edm, Es = 0

If Ei $>$ Em then Eo = Edm, Ef = Em - Edm and Es = Ei - Em

0	41	6F	counter
	41	gF	
1	L5	"a"	
	L4	6F	
2	42	5L	
	L5	"b"	
3	L4	6F	
	42	6L	
4	L5	"c"	
	L4	6F	
5	42	7L	
	L5	(F)	
6	40	7F	temporary storage for Ei
	L5	(F)	
7	40	8F	" " " Edm
	L5	(F)	
8	40	9F	" " " Em
	L5	7F	
9	L0	9F	
	36	14L	
10	41	12F	" " " Es
	L5	7F	
11	L0	8F	
	32	15L	
12	41	11F	" " " Ef
	L5	7F	

13	40	10F	temporary storage for Eo
	26	17L	
14	40	12F	
	L5	9F	
15	L0	8F	
	40	11F	
16	L5	8F	
	40	10F	
17	L5	"d"	
	L4	6F	
18	42	23L	
	42	24L	
19	L5	"e"	
	L4	6F	
20	42	25L	
	42	26L	
21	L5	"f"	
	L4	6F	
22	42	27L	
	42	28L	
23	50	F	waste
	L5	(F)	form Eo
24	L4	10F	
	40	(F)	
25	50	F	waste
	L5	(F)	form Ef
26	L4	11F	
	40	(F)	
27	50	F	waste
	L5	(F)	form Es
28	L4	12F	
	40	(F)	
29	L5	gF	form running total of Eo
	L4	10F	
30	40	gF	
	F5	6F	add "1" to counter
31	40	6F	
	L0	"7"	test for end
32	36	33L	
	26	1L	repeat for next set

3.1.6(a) Silliac Results.

YEAR	MONTH	KOSCIUSKO	GUNGARLIN	JINDABYNE	THANTHANGARA	TUMUT POND	ADAMINABY
05	01	0037.6	0059.6	1170.0	0425.9	0043.4	3634.9
05	02	0037.1	0050.6	1195.7	0490.0	0043.4	3679.6
05	03	0037.7	0056.5	1211.0	0455.5	0043.4	3860.0
05	04	0038.9	0056.5	1211.0	0424.4	0043.4	3860.0
05	05	0040.4	0056.5	1211.0	0395.1	0043.4	3860.0
05	06	0041.5	0056.5	1211.0	0366.5	0043.4	3860.0
05	07	0013.6	0037.5	1211.0	0338.2	0043.4	3856.4
05	08	0005.2	0025.8	1166.9	0309.9	0043.4	3818.4
		← KOSCIUSKO FAILS BY 2.2 th. ac.ft					
-0002.2		0004.1					
05	09	0004.1	0027.8	1049.9	0301.6	0043.4	3820.5
05	10	0010.1	0037.8	1090.0	0293.4	0043.4	3777.2
05	11	0016.2	0047.8	1129.9	0355.2	0043.4	3733.0
05	12	0023.8	0053.0	1183.7	0414.6	0043.4	3681.0

STATE OF STORAGE

YEAR	ΣSRG	ΣSpG	ΣQIJ	ΣQm	ΣSpTo	ΣQAI	ΣQM	ΣSpH
05	0104.0	0030.0	2125.0	0461.4	0040.9	0010.8	0831.2	0055.3

	M4	M5R	M2H	M1H	M4R	M10L	M3	M6	T1-2-5-6
ΣE_s	00129	00070	00013	00069	00220	00220	00585	13737	02026
ΣE_f	00240	00132	00037	00040	00121	00075	00578	02234	01756
ΣE_s	00060	00080	00040	00009	00026	00043	00089	00169	00827

3.1.6(b) Silliac Results.

YEAR	MONTH	KOSCIUSKO	GUNGARLIN	JINDABYNE	TANTANGARA	TUMUT P.	ADAMINABY
05	05	0040.4	0056.5	1211.0	0395.1	0043.4	3860.0

	SR.G'	Sp.G	Qi. J	Qm	Sp.Ta	QA1	QH'	Sp.A
05	0020.0	0010.0	0394.0	0038.8	0000.0	0000.0	0086.8	0000.0

MAY 1905

	M4	M1SH	M12H	M11A	M11B	M12L	M13	M16-7	T1-2-5-6
E_o	00017	00009	00001	00002	00028	00032	00067	00367	00040
E_f	00057	00030	00008	00000	00061	00028	00197	00573	00283
E_s	00015	00020	00010	00000	00013	00021	00044	00000	00000

05 06 0041.5 0056.5 1211.0 0366.5 0043.4 3860.0

05 0020.0 0010.0 0394.0 0038.6 0000.0 0000.0 0121.4 0017.1

JUNE 1905

00019	00010	00001	00019	00030	00034	00074	00409	00137
00055	00029	00008	00040	00059	00025	00190	00532	00873
00015	00020	00010	00009	00013	00021	00044	00000	00827

3.2. THE SIMULATION OF A DIGITAL DIFFERENTIAL ANALYSER ON A GENERAL-PURPOSE DIGITAL COMPUTER.

3.2.1 Introduction:

After the suitability of general purpose digital computers for the solution of the system operational problems had been established, it was decided to program SILLIAC to simulate the operation of a digital differential analyser. D.D.A.'s were unquestionably superior for the solution of ordinary differential equations. In a large organisation like the Snowy Mountains Authority, it was not unreasonable to expect that problems involving these equations would arise from time to time. Although several problems such as surge tank stability studies had been mentioned, it was not known whether the frequency of the occurrence of these problems could still justify the installation of a digital differential analyser. To overcome this last obstacle to the recommendation for a general purpose digital computer, it was necessary to know whether it was feasible to write DDA simulation routines on G.P. computers so that the differential analyser methods of programming could still be applied.

The operation of analogue computers had been simulated by a program written for a Datatron 204 at the Jet Propulsion Laboratory in Pasadena, California, (Ref.66). This routine is known as DEPI for Differential Equations Pseudo-lode Interpreter. The preparation time for a particular problem using DEPI was found to be much less than that for either a standard digital program or an analogue program. However, it was found that the computing time for DEPI was about 10 times that required by an analogue machine. It is to be noted however that the Datatron 204 is a medium speed machine and this ratio would be reduced or even reversed on some high speed computers.

DIDAS, a Digital Differential Analyser Simulator is now available to members of the SHARE organisation of IBM 704 users, (Ref.67). This routine computes in automatic floating point arithmetic and the scaling problem, which can be time consuming in the programming for conventional differential analysers is

thereby eliminated. DIDAS uses a more accurate integration formula than the conventional analyser, thus permitting the use of a larger step-size with a corresponding reduction in running time or greater accuracy obtained with an equivalent step-size. Using the 4096 word core store of the IBM 704 computer, systems of differential equations requiring in the neighbourhood of 300 integrators may be solved. The computing time for a 60 integrator problem is about 7 iterations per second. This is considerably slower than most conventional DDA's having an equivalent number of integrators; however, much larger step sizes are possible using DIDAS (up to 1000 times larger) to obtain the same desired accuracy.

A letter from George R. Slayton, the originator of DIDAS has indicated that DIDAS has been used successfully at the Lockheed Aircraft Corporation's laboratories and has in fact replaced the use of Lockheed's conventional digital differential analyser.

3.2.2 Brief Description of the Simulation Routine.

The program written for SILLIAC was a direct simulation of ADA. No attempt was made to use a higher order integration formula or floating point techniques. 240 storage locations are reserved for the 4 registers of each of the 60 integrators. The allocation of storage for these Registers is as follows:-

Location	100 - 159	:	I register
"	160 - 219	:	Y "
"	220 - 279	:	Y* "
"	280 - 339	:	R "

The fixed instructions of the DDA program are entered into SILLIAC as sexadecimal numbers, and stored in locations 340 to 459. The coding of these fixed instructions is specified on pages 68 to 71 .

The initial conditions which are in the form of decimal numbers are entered by subroutine N2, and the Y registers of the integrators are printed out to the required accuracy by appropriately setting the program parameter of subroutine P14.

Again because of the relative ease of presentation, the flow diagram of the simulation routine has been represented in the form shown on pages 72 to 75. The actual punching of the tape was carried out in the sequence shown on page 76. An example is given on page 77 for the coding of a particular problem.

CODING OF FIXED INSTRUCTIONS.

If n integrators are used ($n \leq 60$), the sexadecimal characters A to J must be entered into each of the locations 340 to 340 + n - 1 and the characters A' to J' must be entered into each of the locations 400 to 400 + n - 1.

Locations 340 - 399.

A B C D E F G H I J

A: spare
B-C: ΔX integrator
D-E: ΔI "
F: reset command
G: reset control
H: multiplier
I: integration mode
J: type out

Locations 400 - 459.

A' B' C' D' E' F' G' H' I' J'

A': Y & I register accuracy
B'-C': 1st ΔY integrator
D'-E': 2nd "
F'-G': 3rd "
H'-I': 4th "
J': integrator output store

Sexadecimal character A:

Spare ; fill in zero (0)

Sexadecimal characters B - C: ΔX integrator (with sign)

These two characters are the sexadecimal equivalents of the integrator number. For example, for integrator 35 (decimal) and BC characters would be 23 (sexadecimal).

If a ΔX input is required from the independent variable the ΔX integrator number is 60; i.e. the B-C characters are 3N.

If there is no connection to the ΔX input, then the integrator number is made equal to 61; i.e. the B-C characters are 3J.

For a negative input from any integrator, one (1) is placed in the most significant digit of the first sexadecimal character. e.g. for a negative input from integrator 35, the coding for the B - C characters would be K3 (i.e. 1010,0011) while that for a positive input from integrator 35 is 23 (i.e. 0010,0011).

The above convention is also used for the numbers of the ΔI and the ΔY integrators.

Sexadecimal characters D - E:

ΔI integrator (with sign)

Sexadecimal character F: Reset command.

F = 0 no reset command.

1)	complete reset	(R & Y registers are reset by reset control 1
2)		" 2
3)		" 3
4)		" 4
6)	partial reset	(Y register (only) is reset by reset control 1
7)		" 2
8)		" 3
9)		" 4

Sexadecimal character G: Reset control.

This instruction is used for integrators which determine when integrators are to be reset or when integrands are to be typed-out.

G = 0 not a reset control
1 1st reset control
2 2nd "
3 3rd "
4 4th "
5 type-out control
6 1st reset control as well as a type-out control
7 2nd "
8 3rd "
9 4th "

Sexadecimal character H: Multiplier.

H = 1 multiplier is 1
2 " 2
3 " 5
4 decision integrator
6 multiplier is -1
7 " -2
8 " -5
9 decision integrator with negative output.

Sexadecimal character I - integration mode.

I = 1 rectangular
2 interpolative
3 extrapolative
4 multiplicative

Sexadecimal character J - type out.

J = 0 integrand is not to be typed-out.
5 integrand is to be typed-out.

Sexadecimal character A' - Y & I register accuracy.

A' = 1 One decimal digit accuracy. i.e. increments are added into the first place after the decimal point.
A' = 2 Two decimal digit accuracy.
A' = 7 Seven " " " (limit)
We limit accuracy to 7 decimal digits ; constant 0.00000001 is used here and due to round off, binary equivalent of this constant is slightly in error. The % error increases as this constant decreases.

Sexadecimal characters B' - C'

1st ΔY integrator (with sign)

Sexadecimal characters D' - E'

2nd ΔY integrator (with sign)

Sexadecimal characters F' - G'

3rd ΔY integrator (with sign)

Sexadecimal characters H' - I'

4th ΔY integrator (with sign)

Sexadecimal character J' : integrator output store.

This is used as a temporary storage and the initial value is made equal to zero (J' = 0).

Only the last 2 bits are used. The first 2 bits of J' are spare and shall be kept equal to 00 all the time. The representation of +1, -1 and 0 outputs is as follows:-

J' = 0010	represents	+1 output
J' = 0001	"	-1 "
J' = 0000	"	0 "

PROGRAM STRUCTURE OF THE D.D.A. SIMULATION ROUTINE.

START ROUTINES

1. Call in DDA Program:-
   (a) Coded instructions (sexadecimal characters) including ΔX, ΔY, ΔI integrators, Y & I register accuracy and fixed instructions.

   (b) Initial conditions (Decimal fractions).

2. Transfer the initial conditions into the Y & Y* registers.

3. Set the initial values of the R registers.

4. Clear iteration counter.

5. Clear integrator counter.

COMPUTE ROUTINE

6. Transfer into temporary storage the coded instructions of the integrator specified by the integrator counter.

7. From the accuracy digit, compute and place in temporary storage, the numerical value of each ΔY or ΔI increment.

8. From the integration mode digit, compute and place in temporary storage, the numerical value of $n\ \Delta Y$ (where $n = \frac{1}{2}$, 1 or $1\frac{1}{2}$). This is the amount which is to be added to or subtracted from the Y* register whenever a ΔY increment occurs.

9. From the first ΔY integrator number:-

   if there is no connection, go to step 13,
   if the input is zero, go to step 10.

Depending on the value of the input (whether +1 or -1) and on the sign of the input, add or subtract the values obtained in steps 7 & 8 to/from the Y & Y* registers respectively.

10. From the second ΔY integrator number:-

if there is no connection, go to step 13,
if the input is zero, go to step 11.

Depending on the value of the input (whether +1 or -1) and on the sign of the input, add or subtract the values obtained in steps 7 & 8 to/from the Y & Y* registers respectively.

11. From the third ΔY integrator number:-

if there is no connection, go to step 13,
if the input is zero, go to step 12.

Depending on the value of the input (whether +1 or -1) and on the sign of the input, add or subtract the values obtained in steps 7 & 8 to/from Y & Y* registers respectively.

12. From the fourth ΔY integrator number:-

if there is no connection, go to step 13,
if the input is zero, go to step 13.

Depending on the value of the input (whether +1 or -1) and on the sign of the input, add or subtract the values obtained in steps 7 & 8 to/from the Y & Y* registers respectively.

13. From the ΔI integrator number:-

if there is no connection, go to step 14,
if the input is zero, go to step 14.

Depending on the value of the input (whether +1 or -1) and on the sign of the input, add or subtract the value obtained in step 7 to/from the I register.

14. From the multiplier digit:-

if the integrator is a decision integrator, go to step 19. Record the output sign. Determine and place in temporary storage the maximum capacity of the R register.

15. From the ΔX integrator number:-

if there is no connection, go to step 21,
if the input is zero, record a zero output and go to step 20.

Depending on the value of the input (whether +1 or -1) and on the sign of the input, add or subtract the value of the Y* register to/from the value of the R register and place the result in temporary storage.

16. From the values obtained in steps 14 and 15, determine the integrator output and the new value of the R register.

17. Make the value of the Y* register equal to the value of the Y register.

18. Go on to step 20.

19. Decision integrator. From the ΔX integrator number:-

if there is no connection, go to step 21,
if the input is zero, record a zero output and go to step 20,
if the Y register exceeds the normal integrator range, record a zero output and go to step 20.

From the value of the input (whether +1 or -1), the sign of the input, the output sign and the integrand sign, determine the integrator output.

20. Store new output by adding (new output - old output) to the old output.

21. Reset and Print Control. From the reset control digit:-

if the integrator is not used as a reset control, go to step 25,
if the integrator output is not +1, go to step 25,
if the integrator is used only as a reset control, go to step 24,
if the integrator is used both as a reset and print control, set the switch (step 23) so that step 24 will follow step 22 and then go to step 22.
if the integrator is used only as a print control, set the switch (step 23) so that step 25 will follow step 22 and then go to step 22.

22. Print Routine. Obtain type-out digits of all integrators in turn. If type-out is not required go to the next integrator. If type-out is required, obtain the accuracy digit and type out the Y register to the required accuracy; then go to the next integrator. Having treated all integrators, punch a CR/LF character.

23. Switch. Transfer control to step 24 or 25 depending on the setting of the switch carried out in step 21.

24. Reset Routine. Obtain the reset command digits of all integrators in turn. If the integrator is not required to be reset go to the next integrator, If the reset command digit does not correspond to the reset control digit obtained in step 21, go to the next integrator. Transfer contents of the I register into the Y & Y* registers. If a partial reset is required go to the next integrator. If a complete reset is required, set the R register to a value corresponding to the multiplier digit of the integrator. Having treated all integrators go on to step 25.

MAIN PROGRAM CONTROL

25. Add 1 to integrator counter.

26. Detect whether all integrators have been treated. If not, return to step 6. If so go on to step 27.

27. Add 1 to iteration counter.

28. Detect whether the sufficient number of iterations have been carried out.

If not, return to step 5. If so go on to step 29.

29. Final stop. OFF.

TAPE PREPARATION FOR D.D.A. SIMULATION ROUTINE.

```
DOI         X12
 00         6K
S parameters
 00         20K
parameters
 00         K6
start routines
 00         K7
input decimal fractions subroutine
 00         K8
input sexadecimal numbers subroutine
 00         K9
transfer coded instructions
 00         KK
add increments into Y, Y* and I
 00         KS
subroutine 1
 00         KN
subroutine 2
 00         KJ
determine integrator output
 00         KF
reset and print and program control
 00         KL
subroutine P14
 26         pN          start program
decimal fractions
sexadecimal numbers
(Silliac must stop at this point so that the DDA tape
 may be placed in the reader.  The program must contain
 the order 20 1019N, so that on starting more tape will
 be read).
 00         58K
No. of integrators
No. of iterations
 26         qN          start program
coded instructions          (sexadecimal numbers)
initial conditions          (decimal fractions)
```

CODING FOR THE SOLUTION OF BESSEL'S EQUATION USING THE SIMULATION ROUTINE.

Problem

To solve $y'' + \frac{1}{x} y' + y = 0$

given $x = 1$ when $y = 0.7652$ $y' = -0.4401$

and to type out x & y for x = 1.0 (0.1) 11.0

Coding for the solution of the problem on ADA

```
@01@60/@02/@@001100@09999 0000@
@02@01/@02/@@006150@09999 0000@
@03@05/@02/@@006100@09999 0000@
@04@60/@05/@@006150@07652 0000@
@05@60/@03/04/@@001150@95599 0000@
@06@@60/@@001150@010000
                          000@
@07@60/@60/@@169100@8001 00000@
```

Coding for the solution of the problem using the simulation routine

Fixed Instructions:-

```
03N3J00110
0013J00615
0053J00610
03N3J00615
03N3J00115
03J3J00115
03N3J16910

4023J3J3J0
4023J3J3J0
4023J3J3J0
4053J3J3J0
403043J3J0
53N3J3J3J0
33N3J3J3J0
```

Initial conditions:-

```
+09999
+09999
+09999
+07652
-04401
+01
-1999N
```

CHAPTER 4. THE SPECIFICATION OF THE GENERAL PURPOSE DIGITAL COMPUTER SNOCOM.

CHAPTER 4. THE SPECIFICATION OF THE GENERAL PURPOSE DIGITAL COMPUTER SNOCOM.

4.1. REASONS FOR BASING THE SPECIFICATION ON THE LGP-30.

After it had been decided that a general purpose digital computer should be built for the Authority, the immediate aim of the project was then to establish the specification of the computer. This could be carried out in three possible ways:-

(1) to establish a completely new design,
(2) to copy a computer of proven design,
or (3) to modify and improve upon a computer of proven design.

The only justification of the first alternative could be that no computer of proven design was able to carry out the required calculations. This, of course, was not the case, and therefore the development of a completely new design was not carried out.

The choice of the computer to copy was influenced a great deal by the availability of the logical design. Although the specification of most digital computers had been published, in general, only scant details of their logical design were made available. However, there was one exception. In the March 1957 issue of theIRE Transactions on Electronic Computers, there appeared a paper entitled "The Logical Design of a Simple General Purpose Computer" written by Stanley P. Frankel. The paper described the logical design which is used in MINAC (constructed at the California Institute of Technology) and the LGP-30 (manufactured by Librascope Inc.). These machines are serial binary digital computers which make use of magnetic drum bulk storage and circulating registers. The integrated design of the computer has been carefully arranged to minimise circuit hardware (Ref.47,48,49).

We were very impressed by the simplicity and neatness of this design, and this led to the immediate decision to base our computer upon this simple but extremely effective machine.

Apart from the availability of the logical design, the choice of the LGP-30 had several other advantages:-

(1) the experience gained from the magnetic drum in ADA would greatly assist the design of the circuitry associated with the magnetic drum on the new machine,

(2) the order code of the LGP 30 is very simple (involving only 16 orders) and the machine would be very easy to program,

(3) the minimal design would result in a relatively cheap computer,

(4) the same method of construction as that used in ADA could be used.

4.2 DEPARTURES FROM THE SPECIFICATION OF THE LGP-30.

4.2.1 Input-Output.

The LGP 30 makes use of Flexowriter for the input and output of information. This machine operates at only 10 characters per second. From experience gained with the SILLIAC studies, it was quite apparent that this input and output speed was far too slow. It was therefore decided to improve upon the specification of the LGP 30 by installing a Ferranti transistorised high-speed tape reader capable of operating at 300 characters per second, and a Teletype high speed tape punch capable of operating at 50 characters per second. It was also decided to increase the computational speed by installing a Bryant magnetic drum capable of running at 6,000 RPM (compared with 3,600 RPM for the drum of the LGP 30).

It was also decided to design the logic for a monitor printer. The logic was to be arranged so that the computer, the punch and the printer could all be in operation simultaneously. This would considerably increase the effective output speed.

The input logic was arranged so that the computer and the tape reader could also be simultaneously in operation. More flexibility was built into the input

order, so that either a complete word, a single character or several characters could be entered into the computer by the execution of an input order. This flexibility was arranged by using control characters on the input tape.

4.2.2 Storage & Computing Speeds.

The bulk memory of the LGP 30 consists of 64 tracks of the drum. Information in each track consists of 64 words (each of 32 binary digits). At the time of writing the bulk memory of SNOCOM consists of 64 tracks each of 32 words. Difficulties in stacking 2048 bits on each track have necessitated the reduction of the packing density by a factor of two. This however has meant that both our storage and computational speed have also been reduced by factors of two. With optimum coding, the time taken for all orders other than multiplication and division is 2.8 m.sec., and that for a multiplication or division order is 20 m.sec. Although the computing speeds have been halved, with the drum running at 6000 RPM the mean access time is still 5 m.sec.

Optimum coding is carried out by appropriately selecting the operand addresses. This is possible as the sector numbers of the drum are interleaved so that numbers in sequence always lie 9 sectors apart.

4.3 THE SNOCOM ORDER CODE.

The SNOCOM order code is presented on pages 114 to 116. A brief discussion of some of the orders will now be given:-

4.3.1 Multiply & Shift Orders.

It is to be noted that two forms of multiplication are available. the "7" multiply order is used for the multiplication of numbers considered as "fractions" while the "6" multiply order is used for the multiplication of numbers considered as "integers". As the accumulator is used to hold both the multiplier and the product, with the suitable choice of constants for the multiplicand, "7" and "6" multiply orders can be used as right and left "shift" orders. A list of these constants with the corresponding number of shifts is shown on page 123 A

4.3.2 External Program Control.

By suitable programming the conditional transfer order can be made to depend on the setting of the external transfer switch Z_0, instead of the sign of the accumulator. This facility can be used to advantage, when it is required to use the same program tape to carry out two related but different operations. Such a need arose while programming the System Operational Problem on SILLIAC. Details of the monthly operation of the system were required for checking the program but only the annual totals of some variables were required from the study. After the monthly details were found to be correct, it was necessary to change one order on the tape before the program could be checked in its final form. With the Z_0 switch on SNOCOM, manual control of a program is possible, and the two different forms of output mentioned above could be obtained by entering the same program twice but with the Z_0 switch in different positions. A further discussion of the Z_0 switch is presented in the appendix.

4.3.3 The Return Address Order.

The "return address" order is used for "planting" the (return) "link" between a subroutine and the main program. The order immediately following the return address order is usually an unconditional transfer order which transfers control to the first order in the subroutine. The address of the second order after the return address order is planted into the last order of the subroutine (also an unconditional transfer order) which transfers control back to the main program. If the addresses of the first and last orders of a closed subroutine are m and n respectively, this subroutine will be entered by the following orders:-

```
3     n    Plants link

+     m  →To first order of subroutine

       ←From last  "     "     "
```

4.3.4 Break Point Orders.

The "stop" or "break point" orders are made contingent on the track number of the order and the position of the break point switches. If the track number is "zero", the stop is unconditional; if it is "one", the order is a waste order and if the track number contains a "one" in the binary place corresponding to any of the break-point switches which is down, the computer will stop. A list of these orders is given on page 120.

4.3.5 Output Orders.

The track number of output orders determines the character which is printed or punched. If the most significant bit is zero, the character corresponding to the other 5 bits is punched. If the most significant bit is "one", a character will be printed if the other 5 bits correspond to any of the 19 combinations which are decoded. These are:- 0,1,2,3,4,5,6,7,8,9,+,-,N,J, F,L, space, full-stop and carriage return/line feed. A list of the output orders is given on pages 117 to 119.

4.3.6 Input Orders.

The effect of an input order is to shift the accumulator 4 (binary) places to the left and to read the next (non-fifth bit) character into the least significant end of the accumulator. When a "stroke" (/) character is first decoded by the input logic, the above procedure is repeated until the second control character "number" (≠≠) is presented by the input reader. With the use of these control characters, complete words can be entered on the execution of a single input order.

4.4 MANUAL CONTROLS.

4.4.1 Initial Filling.

The manual controls for SNOCOM were designed so that information could be entered into a machine which was either completely cleared or full of "rubbish". To do this, three operations are necessary:-

1. To fill the accumulator from the tape reader (→ A)

2. To transfer the contents of the accumulator into the instruction register (A → R)

3. To execute the instruction held in the instruction register (Ex. R)

As an example, suppose that we wish to fill the word /abcdefgh## into location xyz (hexadecimal). The input tape would contain the following words:-

/OONOxyzO## /abcdefgh##

and the manual operations would be as follows:-

1.	Fill Accumulator	(→ A)
2.	Transfer A to R	(A → R)
3.	Fill Accumulator	(→ A)
4.	Execute R	(Ex.R)

Steps 1 & 2 put the "hold" order into the R register. The word /abcdefgh## is entered into the accumulator in step 3, so that on the execution of the hold order in step 4, this word will be filled into location xyz in the main store.

4.4.2 Starting the Program.

The manual controls also include facilities for starting and stopping the computation.

A push-button for clearing the C register (or counter) is also provided. This of course transfers control to track 00, sector 00 when computation is started. It was intended that tracks 00 to 03 were to be filled with input routines and commonly used constants. After this information had been correctly filled, the throwing of a "block record" switch would prevent its possible mutilation whenever erratic programs were put into the machine. The input program starting from track 00, sector 00 may take one of two forms:-

1. A "Hexadecimal Input" Routine,

2. A "Decimal Order Input" Routine.

An example of the former is described on pages 126 to 128.

At the time of writing, it is not known whether the block record switch will safeguard against switching transients (e.g. those produced by switching the machine on and off). If the block record switch cannot be relied upon and it is necessary to frequently "pedal" in the input routine, it is essential that this routine should be as short as possible. A "bootstrap" routine consisting initially of only 3 orders is described on pages /3/ to /32.

4.5 CONCLUDING REMARK.

At the time of writing, it can be said that an advanced stage has been reached in the design, construction and testing of SNOCOM. This extremely effective machine will undoubtedly prove invaluable not only in power system operational studies but also in a wide range of other investigations.

CONCLUDING REMARKS.

This thesis describes a number of investigations which were partly responsible for changing the specification of a computer which was to be built for the Snowy Mountains Authority from a digital differential analyser to a general-purpose digital computer.

As the computer was to be used for the solution of a particular system operational problem, firstly it was necessary to obtain an exact specification of this problem. This was briefly described in Chapter 1, together with introductory remarks concerning the nature of the problem in relation to other system operational studies.

At the beginning of the project it was understood that a D.D.A. would be built for the Authority. It was therefore necessary to examine in detail the capabilities of the D.D.A. and to program it for the solution of the system operational problem. These investigations are briefly described in Chapter 2.

When it was suspected that a general purpose digital computer would be more suitable, the problem was programmed for its solution on SILLIAC and estimates of the computing time were obtained. To maintain the advantages of the D.D.A. for the solution of ordinary differential equations, a D.D.A. simulation routine was written for SILLIAC. The SILLIAC investigations were briefly described in Chapter 3.

The reasons for basing the Authority's computer upon the specification of the LGP-30 are given in Chapter 4. A discussion of some important aspects of this specification is also presented.

From the investigations described in the thesis, a number of very general conclusions may be drawn:-

1. Digital differential analysers are special-purpose digital computers designed for the solution of ordinary differential equations, and are programmed with the relative ease of analogue computers. With the provision of special integrator functions, a wide range of "discontinuity nonlinearities" may be generated, and the computer may be programmed for the automatic solution of

a number of problems which would normally require manual operation on a mechanical differential analyser. The D.D.A. is an incremental computer, and for this reason it is extremely inefficient for carrying out simple arithmetic operations on whole numbers.

2. General-purpose digital computers are an invaluable tool for planning purposes when frequently a number of alternatives have to be examined in detail. Special-purpose computers may only be applied to problems whose specifications have been firmly established, and the grounds for their justification may be evaluated by simulation routines on general purpose digital computers.

As the system operational problem had been formulated as a sequence of arithmetic operations on whole numbers and not as a system of ordinary differential equations, it is obvious that a G.P. digital computer will be far superior to a D.D.A. for the solution of this particular problem. The merit of this thesis does not lie wholly on this simple and obvious conclusion, but rather on the investigations which were necessary to change the specification from a D.D.A. to a G.P. digital computer.

ACKNOWLEDGMENTS.

The work described in this thesis was carried out within the Electrical Engineering Department of The University of Sydney while the author was under the employment of the Snowy Mountains Hydro Electric Authority. The help of Professor D.M. Myers, Professor of Electrical Engineering, and the support of the Snowy Mountains Authority are gratefully acknowledged.

The author wishes to thank Dr. M.W. Allen for providing many enlightening discussion periods on computer techniques, for directly supervising most of the computer studies, and for being largely responsible for the specification, design and construction of SNOCOM.

The author gratefully acknowledges the encouragement and interest of Dr. H.K. Messerle, who supervised earlier research work carried out by the author, and who has continued to provide a number of helpful suggestions.

Thanks are also due to Messrs. K.E. Johnson and J. Kelly of the Snowy Mountains Authority, for supplying much of the information concerning the "System Operation" of the Snowy Mountains Scheme.

REFERENCES.

(1) THE THEORY OF INCREMENTAL RATES. Steinberg & Smith.
Part 1, Electrical Engineering, March, 1934,
Part II, " " , April, 1934.

(2) INTRASYSTEM TRANSMISSION LOSSES. E.E. George.
AIEE Transactions (Electrical Engineering) Vol.62, March, 1943, pp.153-58.

(3) CO-ORDINATION OF FUEL COST & TRANSMISSION LOSS BY USE OF THE NETWORK ANALYSER TO DETERMINE PLANT LOADING SCHEDULES. E.E. George, H.W. Page & J.B. Ward.
AIEE Trans. Vol.68, Pt.11, 1949, p.1152-63.

(4) TOTAL AND INCREMENTAL LOSSES IN POWER TRANSMISSION NETWORKS. J.B. Ward, J.R. Eaton, H.W. Hale.
AIEE Transactions, Vol.69, Pt.1, 1950. pp.626-32.

(5) TRANSMISSION LOSSES & ECONOMIC LOADING OF POWER SYSTEMS. L.K. Kirchmayer & G.H. McDaniel.
G.E. Review, Vol.54 No.10, Oct. 1951. p.39-46.

(6) ANALYSIS OF TOTAL AND INCREMENTAL LOSSES IN TRANSMISSION SYSTEMS. L.K. Kirchmayer, G.W. Stagg.
AIEE Transactions, Vol.70, Pt.1, 1951. pp.1197-1204.

(7) CO-ORDINATION OF INCREMENTAL FUEL COSTS AND INCREMENTAL TRANSMISSION LOSSES BY FUNCTIONS OF VOLTAGE PHASE ANGLES. W.R. Brownlee.
Power Apparatus & Systems, June, 1954, p.529.

(8) A GENERAL TRANSMISSION LOSS EQUATION. E.D. Early, R.E. Watson, G.L. Smith.
AIEE Transactions, Vol.74, 1955, AIEE paper No.55-90.

(9) AUTOMATIC DIGITAL COMPUTER APPLIER TO GENERATION SCHEDULING. A.F. Glimn, L.K. Kirchmayer, R. Habermann Jr., and R.W. Thomas.
AIEE Transactions, Part III-B, October 1954, p.1267.

(10) A COMPUTER FOR ECONOMIC SCHEDULING AND CONTROL OF POWER SYSTEMS. C.D. Morrill & J.A. Blake.
AIEE Transactions, Part III, December 1955, p.1136.

(11) AN INCREMENTAL COST OF POWER-DELIVERED COMPUTER. E.D. Early, W.E. Phillips and W.T. Shreve.
AIEE Transactions, Part III, June, 1955, p.529.

(12) A NEW AUTOMATIC DISPATCHING SYSTEM FOR ELECTRIC POWER SYSTEM. K.N. Burnett, D.W. Halfhill & B.R. Shepard.

Trans Amer. Inst. Elect. Engrs III, Vol.75, 1049-56 (1956 = Pwr Apparatus Syst., No.27 (Dec. 1956)). E.E. Abstr. No.1778/1957.

(13) THE "EARLY BIRD" GOES AUTOMATIC. E.J. Kompass.

Control Eng., Vol.3 pp.77-83, Dec. 1956.

(14) AUTOMATIC ECONOMIC DISPATCHING AND LOAD CONTROL - OHIO EDISON SYSTEM. R.H. Travers.

Trans Amer. Inst. Elect. Engrs III, Vol.76, 291-301 (1957) = Pwr Apparatus Syst., No.30 (June, 1957. E.E. Abstr. No.5744/1957.

(15) OPERATING EXPERIENCE WITH GEDA AUTOMATIC ECONOMIC DISPATCHING - OHIO EDISON SYSTEM. R.H. Travers.

Trans. Amer. Inst. Elect. Engrs III, Vol.77, 407-10 (1958) = Pwr Apparatus Syst. No.36 (June, 1958). E.E. Abstr. No.4516/1958.

(16) ACCURACY CONSIDERATIONS IN ECONOMIC DISPATCHING OF POWER SYSTEMS. I. A.F. Glimn, L.K. Kirchmayer, V.R. Peterson and G.W. Stagg.

Trans Amer. Inst. Elect. Engrs III, Vol.75, 1125-37 (1956) = Pwr Apparatus Syst., No.27 (Dec. 1956). E.E. Abstr. No.3264/1957.

(17) INCREMENTAL MAINTENANCE COSTS OF STEAM-ELECTRIC GENERATING STATIONS. M.J. Steinberg.

Trans Amer. Inst. Elect. Engrs III, Vol.76, 1251-5 (1957) = Pwr Apparatus Syst., No.34 (Feb. 1958). E.E. Abstr. No.3314/1958.

(18) ECONOMIC CONSIDERATIONS IN GENERATION SCHEDULING FOR THE SOUTHWESTERN PUBLIC SERVICE COMPANY SYSTEM. R.W. Thomas, L.K. Kirchmayer and J.R. Wilson.
Trans Amer. Inst. Elect. Engrs. III, Vol.76, 1545-55 (1957) = Pwr Apparatus Syst. No.34 (Feb. 1958). E.E. Abstr. No.2349/1958.

(19) A NEW APPROACH TO LOSS MINIMISATION IN ELECTRIC POWER SYSTEMS. J.F. Calvert, T.W. Sze.

Power Apparatus & Systems, Feb. 1958 pp.1439-1446.

(20) EXACT ECONOMIC DISPATCH - DIGITAL COMPUTER SOLUTION. R.B. Shipley & M. Hochdorf.

Trans Amer. Inst. Elect. Engrs III, Vol.75, 1147-53 (1956) = Pwr Apparatus Syst., No.27 (Dec.1956) E.E. Abstr. No.3265/1957.

(21) ELECTRIC ANALOG CIRCUITS FOR EXACT ECONOMIC DISPATCH. R.B. Shipley.

Trans Amer. Inst. Elect. Engrs III, Vol.76, 869-74 (1957) = Pwr Apparatus Syst., No.33 (Dec. 1957). E.E. Abstr. No.2771/1958.

(22) INTEGRATED SYSTEM OPERATION - ECONOMY. J.M. Telfer.
Symposium on Power System Operation, Electrical Engineering Dept. University of Sydney, January, 1955.

(23) MODERN TRENDS IN THE HYDRO-ELECTRIC POWER FIELD. W. Diesendorf.
The Journal of the Institution of Engineers, Australia Apr-May, 1958. p.117.

(24) HYDRO GENERATION. J.B. Kirkwood.
Symposium on Power System Economy, Electrical Engineering Dept., University of Sydney, August, 1954.

(25) SYMPOSIUM ON "POWER SYSTEM OPERATION" SESSION II. HYDRO-ELECTRIC POWER STATION OPERATION. A.C.H. Frost.
The University of Sydney, January, 1955.

(26) SHORT-RANGE ECONOMIC OPERATION OF A COMBINED THERMAL AND HYDROELECTRIC POWER SYSTEM. W.G. Chandler, P.L. Dandeno, A.F. Glimn, L.K. Kirchmayer.
AIEE Trandsactions Vol.72, Pt.3, 1953 p.1057.

(27) COMPUTER SEARCH FOR ECONOMICAL OPERATION OF A HYDRO-THERMAL ELECTRIC SYSTEM. R.J. Cypser.
AIEE Transactions, Part III-B, October, 1954 p.1260.

(28) SHORT-RANGE LOAD ALLOCATIONAL HYDRO-THERMAL ELECTRIC SYSTEM. John J. Carey.
AIEE Transactions, Part III-B, October, 1954, p.1105.

(29) CO-ORDINATION OF HYDRO AND STEAM GENERATION. C.W. Watchorn.
AIEE Transactions Part III, April, 1955. p.142.

(30) THE CALCULUS OF VARIATIONS AS APPLIED TO STORAGE PROBLEMS. B.A. Bolt.
The Statistical Society of New South Wales. Symposium on Operations Research, August, 1957.

(31) AN APPLICATION OF LINEAR PROGRAMMING TO A TRANSPORTATION PROBLEM. N.B. Heal.
The Statistical Society of New South Wales, Symposium on Operations Research, August, 1957.

(32) AN APPLICATION OF DIGITAL COMPUTERS TO POWER SYSTEM PLANNING. J.B. Kirkwood, N.B. Heal and R.C. Moffat.
The Institution of Engineers, Australia, Electrical & Mechanical Engineering Transactions, May, 1959, p.17-23.

(33) PROPOSED ELECTRIC HYDRAULIC ANALOGUE MODEL FOR STUDYING HYDRO-ELECTRIC SYSTEM OPERATION. (Report)
Holdaway & Hogan S.M.H.E.A. library 621.312134
Physics Section SE2.

(34) TENTATIVE SPECIFICATIONS FOR DIGITAL DIFFERENTIAL ANALYSER. S.M.H.E.A. report 2/10/56. J.B. Kirkwood.

(35) SYSTEM OPERATION. J. Kelly.
S.M.H.E.A. report 19/2/57.

(36) A DIGITAL DIFFERENTIAL ANALYSER STUDY OF "SYSTEM OPERATION" FOR THE SNOWY MOUNTAINS HYDRO ELECTRIC AUTHORITY. D.G. Wong.
S.M.H.E.A. Report, March, 1957.

(37) THE C.S.I.R.O. DIFFERENTIAL ANALYSER. D.M. Myers, W.R. Blunden.
J. of the I.E. Aust., Oct-Nov. 1952.

(38) DDA - WHAT IS IT? R.N. Goldman.
Electrical Engineering, July, 1958, pp.592-594.

(39) OPERATION MANUAL DIGITAL DIFFERENTIAL ANALYSER MODEL D-12 BENDIX COMPUTER, Division of Bendix Aviation Corporation 5630 Arbor Vitae St. Los Angeles 45, California.
April, 1954.

(40) DIGITAL DIFFERENTIAL ANALYSERS. (An applications manual for digital and Bush type differential analysers), George F. Forbes, B.S. Part I - Elements (second edition 1955). Part II - Applications (third edition 1956). Copies may be obtained from George F. Forbes 10117 Bartee Avenue, Pacoima, California.

(41) AN ELECTRONIC DIFFERENTIAL ANALYSER AS A DIFFERENCE ANALYSER. Louis B. Wadel.
Journal of the Association of Computing Machinery, Vol.1 No.3 July, 1954.

(42) SIMULATION OF TIME DELAYS (WITH BIBLIOGRAPHY). The Simulation Council Newsletter.
Instruments & Automation, March, 1957.

(43) SIMULATION OF DISCONTINUITY NONLINEARITIES. Don H. Schuster.
Instruments & Automation, April, 1957.

(44) A DECIMAL ADDITION - SUBTRACTION UNIT. M.W. Allen.
Proc. I.E.E. Part B Supplement, Convention on Digital Computer Techniques, April, 1956, p.138.

(45) A.D.A. - A TRANSISTOR DECIMAL DIGITAL DIFFERENTIAL ANALYSER. M.W. Allen.
J. Instn Engrs, Australia, Vol.29, No.10-11, 255-62 (Oct.-Nov., 1957). E.E. Abstr. No.2766/1958.

(46) A STUDY OF "SYSTEM OPERATION" USING "SILLIAC". D.G. Wong.
S.M.H.E.A. Report, September, 1957.

(47) THE LIBRASCOPE GENERAL PURPOSE COMPUTER LGP-30. S. Frankel and J. Cass.
= Instruments & Automation, Vol.29 pp.264-270, Feb.1956.

(48) USEFUL APPLICATIONS OF A MAGNETIC DRUM COMPUTER. S. Frankel.
Elec. Eng. Vol.75, pp.634-639, July, 1956.

(49) THE LOGICAL DESIGN OF A SIMPLE GENERAL PURPOSE COMPUTER. Stanley P. Frankel.
I.R.E. Transactions on Electronic Computers, March, 1957.

(50) HIGHER MATHEMATICS FOR ENGINEERS AND PHYSICISTS. I.S. & E.S. Sokolnikoff.
Second Edition. (McGraw-Hill) 1941.

(51) NUMERICAL ANALYSIS. (Book). D.R. Hartree, F.R.S.
Oxford Clarendon Press.

(52) CALCULATING INSTRUMENTS AND MACHINES.(Book). D.R. Hartree.
(University of Illinois Press).

(53) ANALOG COMPUTER TECHNIQUES. Clarence L. Johnson.
(McGraw-Hill) 1956.

(54) LOGICAL DESIGN OF DIGITAL COMPUTERS. (BOOK) M. Phister.
(Wiley, 1958).

(55) ARITHMETIC OPERATIONS IN DIGITAL COMPUTERS. (Book)
R.K. Richards Ph.D., 1955. (D. Van Nostrand Company,Inc.)

(56) THE PREPARATION OF PROGRAMS FOR AN ELECTRONIC DIGITAL COMPUTER. (Book) Second Edition. M.V. Wilkes, D.J. Wheeler, S. Gill. (Addison-Wesley, 1957).

(57) DIGITAL COMPUTER PROGRAMMING.(Book). D.D. McCraken.
(Wiley, 1957).

(58) ENGINEERING GEOLOGY FOR THE SNOWY MOUNTAINS SCHEME. D.G. Moye.
The Journal of The Inst. of Engrs, Australia, Oct.-Nov. 1955, p.287.

(59) THE 330-kV TRANSMISSION SYSTEM IN NEW SOUTH WALES. W. Diesendorf and J.J. Hurley.

The Journal of The Institution of Engineers, Australia, Oct.-Nov. 1955. p.273.

(60) THE UPPER TUMUT WORKS. D.E. Campbell, I.L. Pinkerton, A.N.G. Bray and A.C.H. Frost.

The Journal of the Inst. of Engineers, Australia, Jan.-Feb. 1956. p.1.

(61) THE SNOWY MOUNTAINS SCHEME AND THE APPLICATION OF SCIENTIFIC SERVICES. T.D.J. Leech.

The Journal of the Inst. of Engineers, Aust. p.99. March, 1958.

(62) THE INVESTIGATION, DESIGN, CONSTRUCTION AND COMMISSIONING OF THE GUTHEGA PROJECT OF THE SNOWY MOUNTAINS SCHEME. H.E. Dann, I.K. Pinkerton, E.G. Warrell, R.H. Blake, A.C.H. Frost and W.M. Shellshear.

J. Instn Engrs, Australia, Vol.29, No.6, 129-58 (June, 1957). E.E. Abstr. No.493/1958.

(63) A SURVEY OF DOMESTIC ELECTRONIC DIGITAL COMPUTING SYSTEMS. Martin H. Weik - (United States Department of Commerce Office of Technical Services).

Ballistic Research Laboratories, Report No.971, Dec. 1955. Aberdeen Proving Ground, Mel. Price $4.75.

(64) ECONOMIC OPERATION OF VARIABLE-HEAD HYDROELECTRIC PLANTS. A.F. Glimn, L.K. Kirchmayer.

Power Apparatus & Systems, Dec. 1958. p.1070.

(65) ON AN ANALOGUE SOLUTION OF TWO POINT BONDARY VALUE PROBLEMS. D. Greenspan, B. Ulrick, S. Matsuno.

The Review of Scientific Instruments, Vol.28, No.12 December, 1957, p.1040-2.

(66) A METHOD OF SIMULATING A DIFFERENTIAL ANALYSER ON A DIGITAL COMPUTER. F. Lesh.

External Publication No.412, Jet Propulsion Laboratory, California Institute of Technology, Pasadena, California, 30th Sept. 1957.

(67) DIDAS. A DIGITAL DIFFERENTIAL ANALYSER SIMULATOR. George R. Slayton.

(68) DIFFERENTIAL ANALYSER SOLUTION OF HYDRAULIC PROBLEMS IN HYDROELECTRIC SYSTEMS. H.K. Messerle.

La Houille Blanche, Dec.1956, No.6, p.813-836.

A P P E N D I X

DIGITAL DIFFERENTIAL ANALYSER SOLUTION OF ORDINARY DIFFERENTIAL EQUATIONS

1. Highest derivative known explicitly in terms of the lower derivatives and the variables

If the highest derivative y_n of an n'th order ordinary differential equation can be conviently expressed in terms of the other variables, the equation may be written in the form:-

$$y_n = f(x, y, y_1, y_2 - - - - y_{n-1}) \quad ---- \quad (1)$$

where $y_1, y_2 \; - - - \; y_{n-1}, y_n$ are the 1st, 2nd $- - - -$ (n-1)'th, n'th derivatives of y with respect to x.

For its solution on a digital differential analyser, equation 1 must firstly be represented in "incremental form". When this is done the following equation is obtained:-

$$dy_n = df(x, y, y_1, y_2, - - - \; y_{n-1}) \quad --- \quad (2)$$

where dy_n and $df(x, y, y_1, y_2, - - - - y_{n-1})$ represent "increments of y_n" and "increments of $f(x, y, y_1, y_2, - - - y_{n-1})$" respectively.

(It may be noted that the procedure for expressing equation 1 in incremental form is equivalent to differentiating equation 1 with respect to x and then multiplying through by dx).

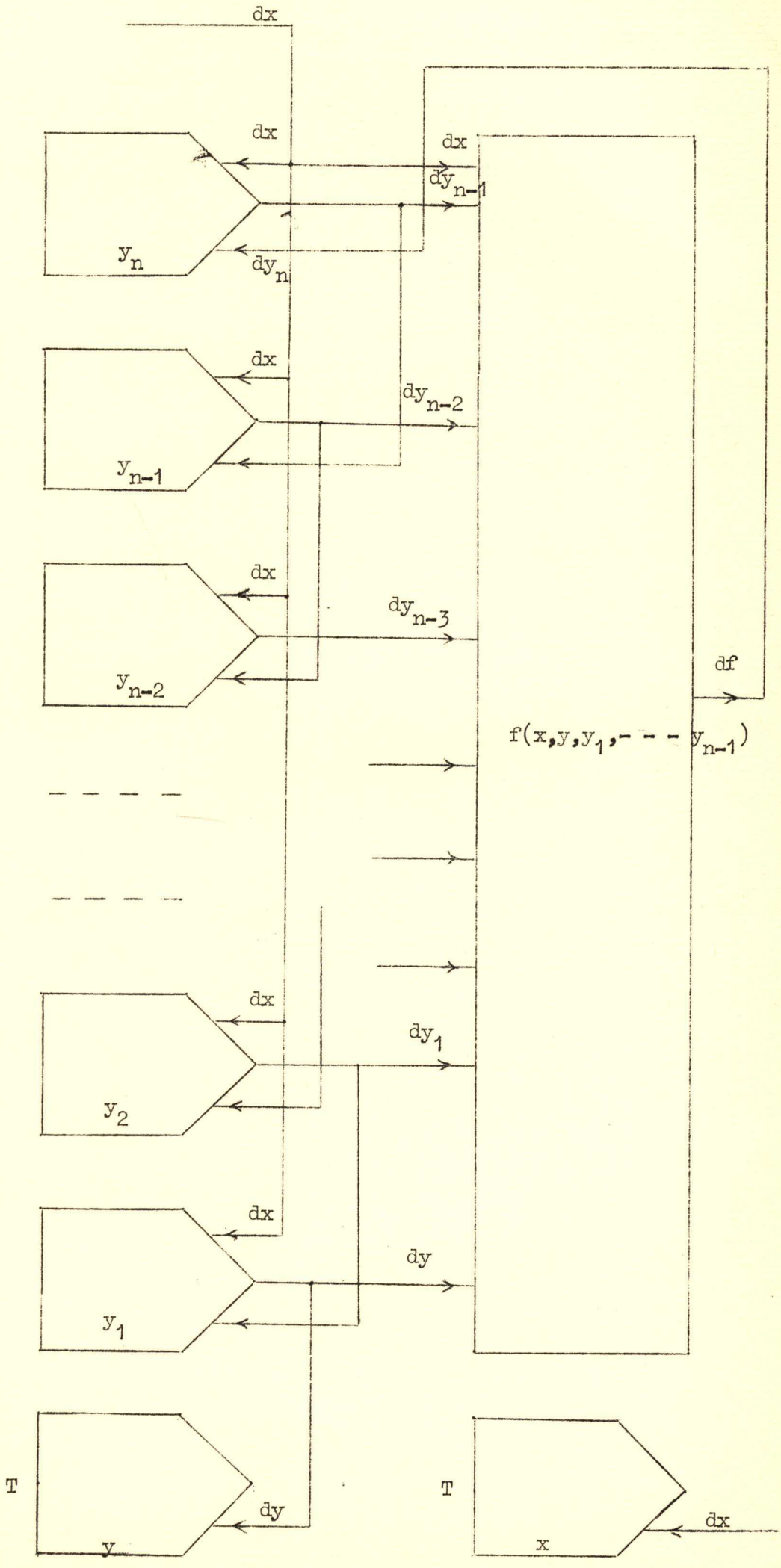
The mapping required for the digital differential analyser solution of equation 2 is shown in figure 1. The block marked $f(x, y, y_1, - - - \; y_{n-1})$ represents a number of integrators which are required to generate increments of the function $f(x, y, y_1, y_2 - - y_{n-1})$ from the changes in the variables $x, y, y_1, y_2, - - - \; y_{n-1}$. These increments, which also represent increments of the highest derivative (from equation 2), are accumulated in the integrand of an integrator whose primary incremental input is the independent variable dx. The output of this integrator is therefore dy_{n-1}, because:-

$$dy_{n-1} = dy_n \cdot dx \quad --- \quad (3)$$

The other derivatives and the dependent variable are similarly obtained by the successive integration of dy_{n-1}.

Increments of the independent variable x and the dependent variable y may be accumulated in the integrands of integrators so that they may be typed out.

DIFFERENTIAL EQUATION: $y_n = f(x,y,y_1,y_2,- - - y_{n-1})$



2. Linear Differential Equation with Constant Coefficients.

A linear differential equation with constant coefficients is an equation of the form:-

$$a_n y_n + a_{n-1} y_{n-1} + - - - - - a_1 y_1 + a_o y = f(x) \quad ----- \quad (4)$$

where a_n, a_{n-1}, - - - - a_1, a_o are constants.

Solving equation 4 for y_n we obtain:-

$$y_n = \frac{-1}{a_n} \left\{ a_{n-1} y_{n-1} + a_{n-2} y_{n-2} + - - - - - a_1 y_1 + a_o y - f(x) \right\} \quad --- \quad (5)$$

Equation 5 is a special case of equation 1. Hence the mapping for the digital differential analyser solution of equation 5 will follow the method discribed in section 1.

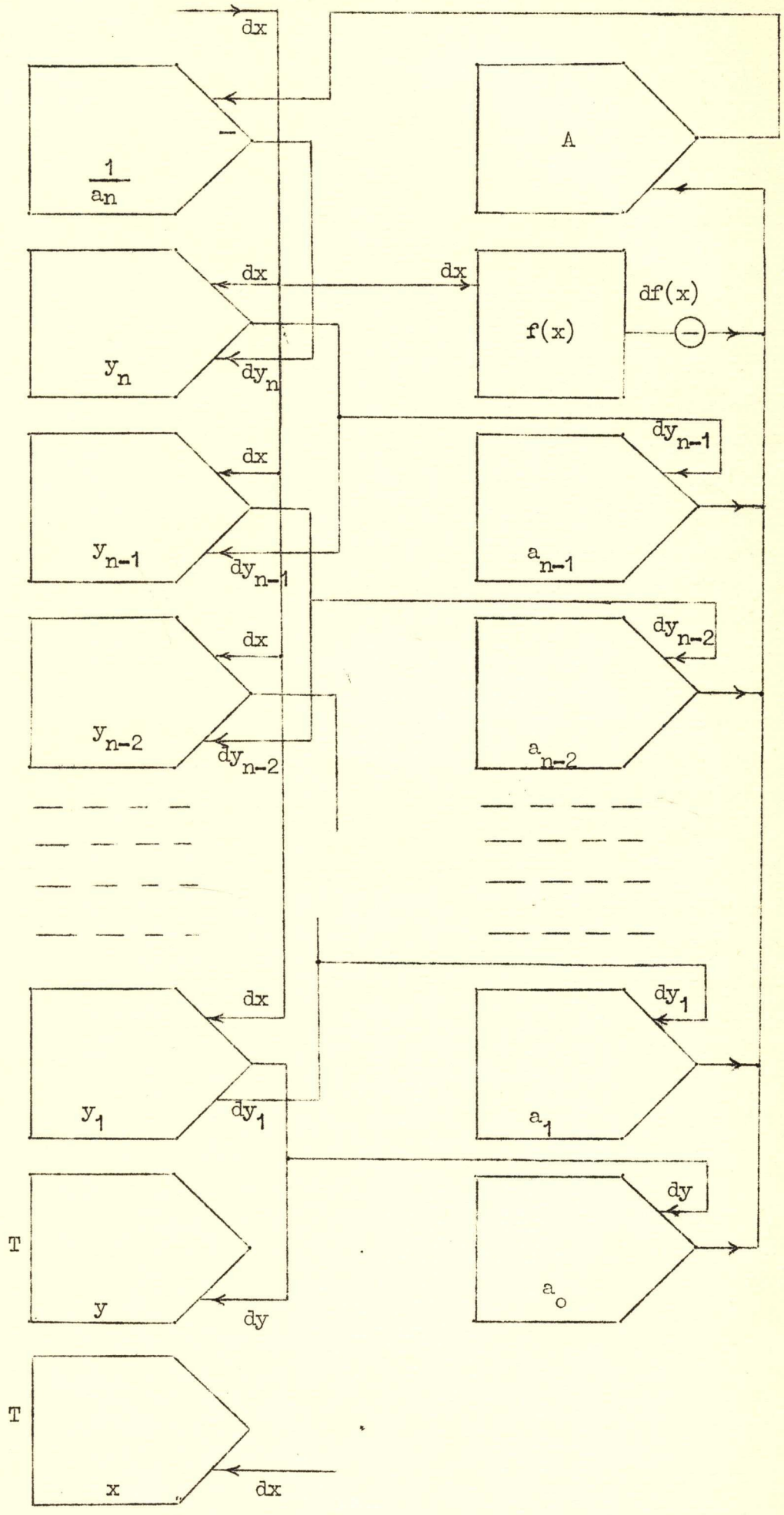
Expressing equation 5 in incremental form, we obtain:-

$$dy_n = \frac{-1}{a_n} \left\{ a_{n-1} dy_{n-1} + a_{n-2} dy_{n-2} + - - - a_1 dy_1 + a_o dy - df(x) \right\} \quad --- \quad (6)$$

The mapping required for the d.d.a. solution of equation 6 is shown in figure 2. The block marked f(x) represents a number of integrators required to generate increments of the function, i.e. df(x), from increments of the independent variables (i.e. dx). n integrators are used for the successive integration of y_n; n+1 integrators are used as constant multipliers for the generation of the terms on the right hand side of equation 6, and the adder is used to combine these terms to form the highest derivative, y_n.

LINEAR DIFFERENTIAL EQUATION WITH CONSTANT COEFFICIENTS

$$a_n y_n + a_{n-1} y_{n-1} + - - - + a_1 y_1 + a_0 y = f(x)$$



XX.

3. Highest Derivative not known explicitly in terms of the lower derivatives and the variables

An ordinary differential equation of the n'th order may be represented by the implicit equation:-

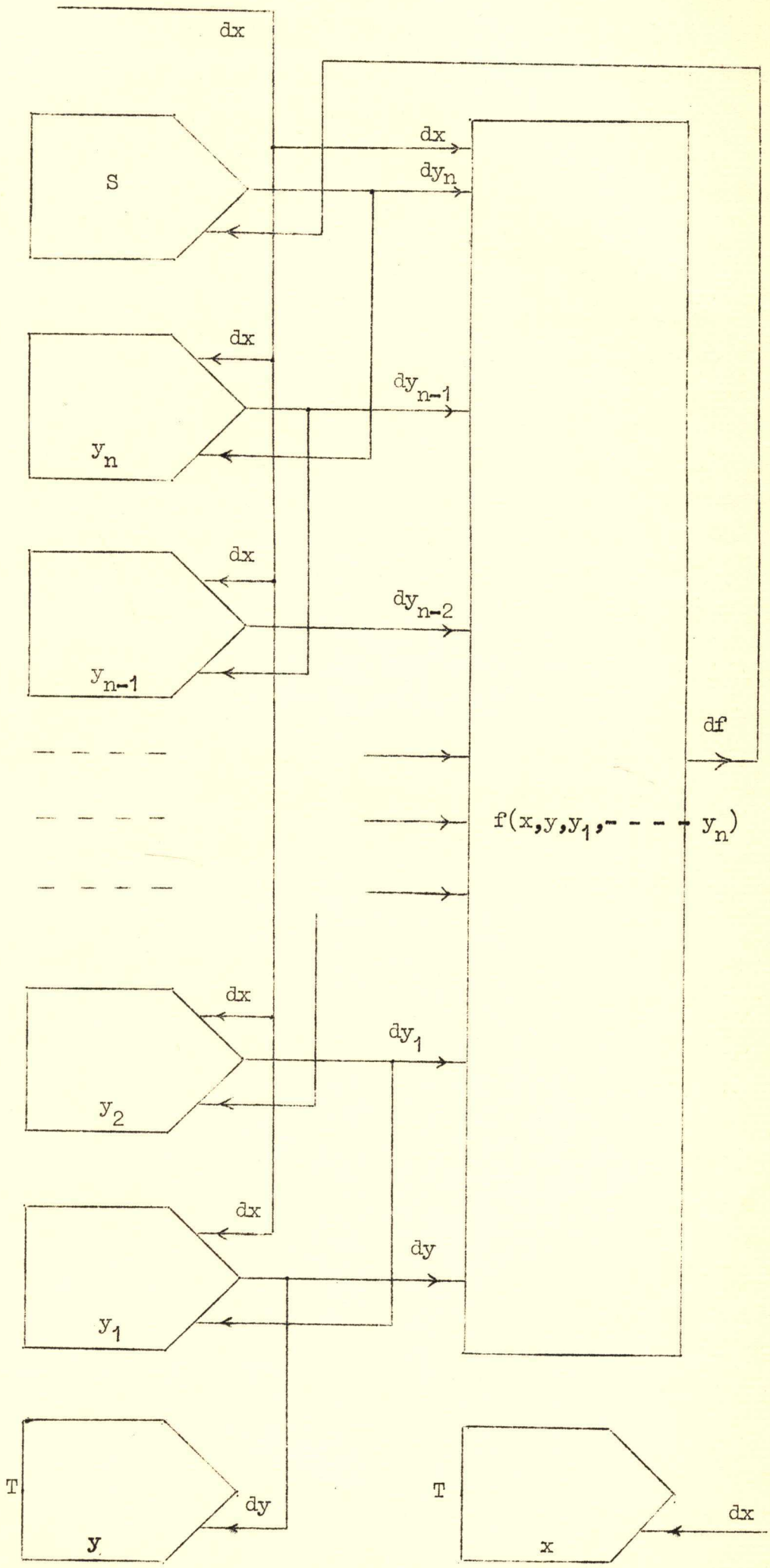
$$f(x, y, y_2, - - - - y_n) = 0 \qquad --- \qquad (7)$$

We now consider the case when equation 7 cannot be solved for the highest derivative y_n.

The mapping required for the d.d.a. solution of equation 7 is shown in figure 3. The block marked $f(x, y, y_1, y_2, - - - - y_n)$ represents a group of integrators which are required to generate increments of the function $f(x, y, y_1, y_2, - - - y_n)$ from variations of the variables $x, y, y_1, y_2, - - - y_n$. These increments are accumulated in the integrand of a servo integrator. The servo adjusts the value of the highest derivative y_n so that equation 7 is always satisfied. The highest derivative is integrated successively in a chain of integrators to generate the lower derivatives and the dependent variable.

The use of a servo integrator is analogous to the use of an operator (i.e. using a human servo) when solving equations like equation 7 on a mechanical differential analyser. The function $f(x, y, y_1, y_2, ---- y_n)$ is generated within the mechanical differential analyser by a series of integrators, adding units and ratio boxes. The value of the function is displayed on a plotting table or a counter. During the computation, the operator must continuously adjust the angular position of the shaft representing the highest derivative y_n so that the value of the function, as displayed by the plotting table or counter, is always zero.

DIFFERENTIAL EQUATION $f(x,y,y_1, \text{---} y_n) = 0$



4.

4. Simultaneous Differential Equations

A set of simultaneous ordinary differential equations of the n'th order, involving two dependent variables, may be represented by the equations:-

$$f_1(x, y, z, y_1, y_2, \text{---} y_n, z_1, z_2 \text{---} z_n) = 0 \qquad \text{---(8a)}$$

$$f_2(x, y, z, y_1, y_2, \text{---} y_n, z_1, z_2, \text{---} z_n) = 0 \qquad \text{---(8b)}$$

where $y_1, y_2 \text{---} y_n$ are the derivatives of y with respect to x and $z_1, z_2, \text{---} z_n$ are the derivatives of z with respect to x.

The digital differential analyser solution of equations 8a and 8b is represented by the integrator interconnections shown in Fig. 4. The block marked f_1 represents a group of integrators required to generate the function f_1. The increments of this function are accumulated in the integrand of a servo integrator which is used to generate the derivative y_n so that equation 8a is satisfied. The derivative y_n is successively integrated in a chain of integrators. Similarly, the block marked f_2 represents another group of integrators required to generate the function f_2. Another servo integrator is used to generate the derivative z_n so that equation 8b is satisfied. The derivative z_n is also successively integrated in a chain of integrators.

Two special cases of equations 8a and 8b are represented by the equations:-

$$y_n = f_3(x, y, z, y_1, y_2, \text{---} y_{n-1}, z_1, z_2, \text{---} z_n) \qquad \text{---(9a)}$$

$$z_n = f_4(x, y, z, y_1, y_2, \text{---} y_n, z_1, z_2, \text{---} z_{n-1}) \qquad \text{---(9b)}$$

$$f_1(x, y, y_1, y_2 \text{---} y_n) = F_1(x, y, z) \qquad \text{---} \quad (10a)$$

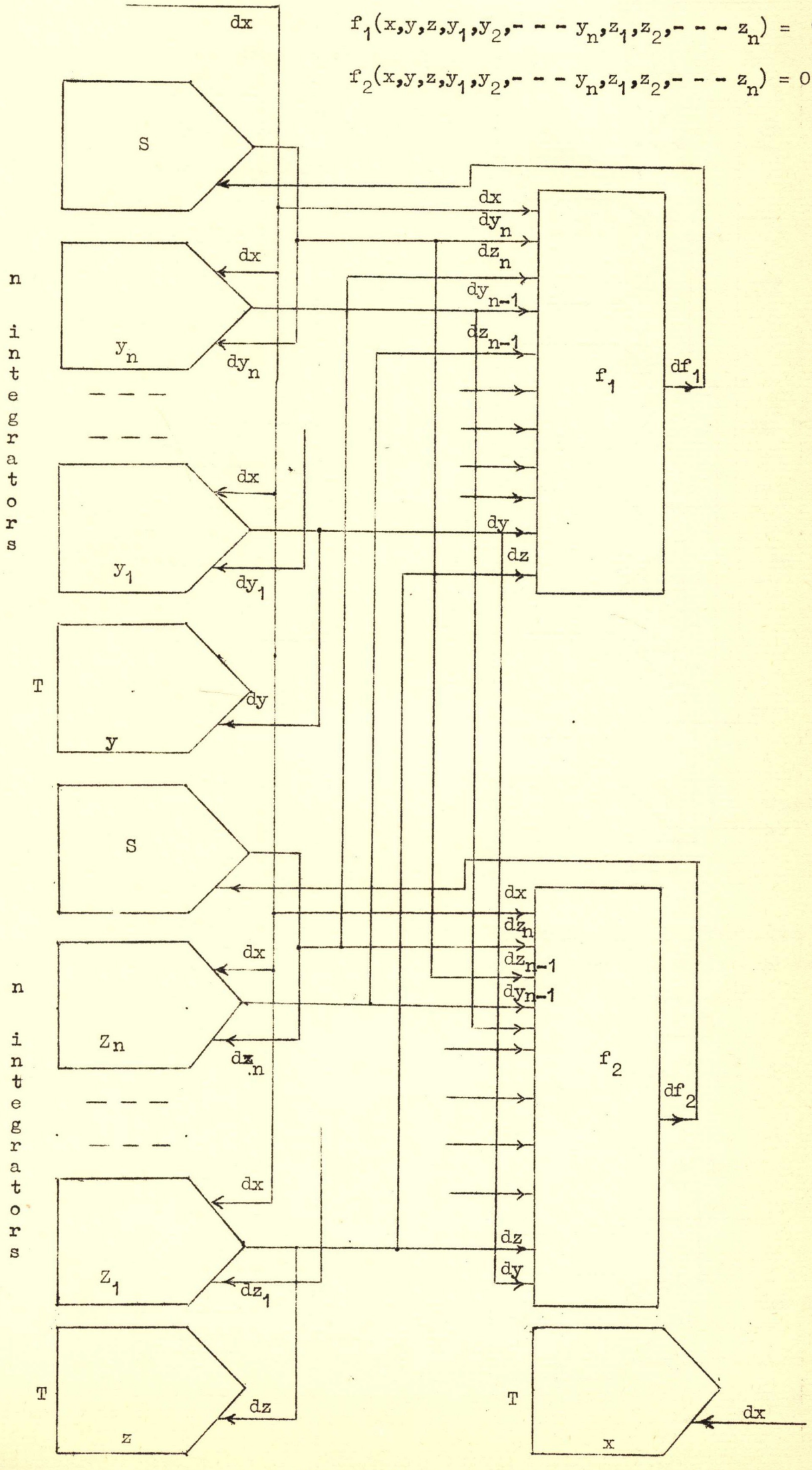
$$f_2(x, z, z_1, z_2 \text{---} z_n) = F_2(x, y, z) \qquad \text{---} \quad (10b)$$

The integrator interconnections for the solution of these equations may be readily obtained by extensions of the procedures discussed in previous sections. For this reason, further discussion of the d.d.a. solution of these equations will not be given.

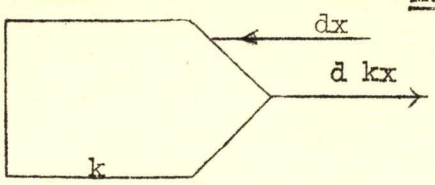
SIMULTANEOUS DIFFERENTIAL EQUATIONS

$f_1(x,y,z,y_1,y_2,- - - y_n,z_1,z_2,- - - z_n) = 0$

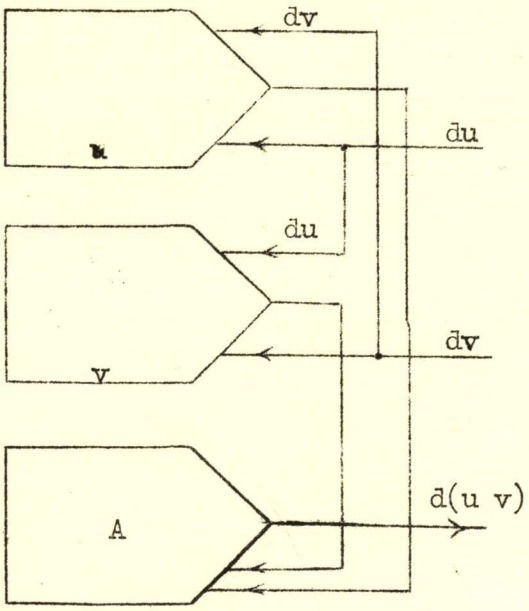
$f_2(x,y,z,y_1,y_2,- - - y_n,z_1,z_2,- - - z_n) = 0$



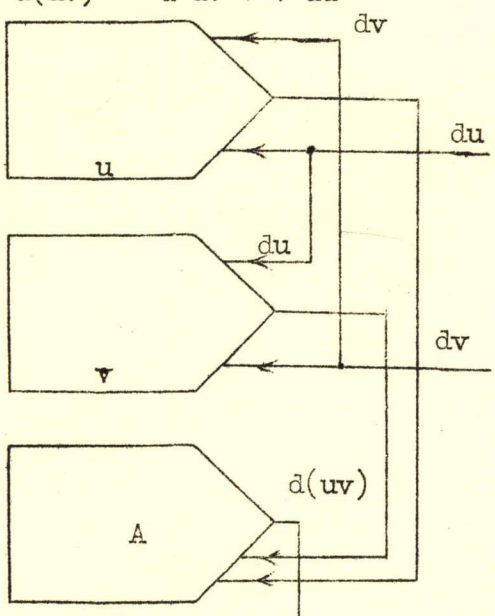
Multiplication

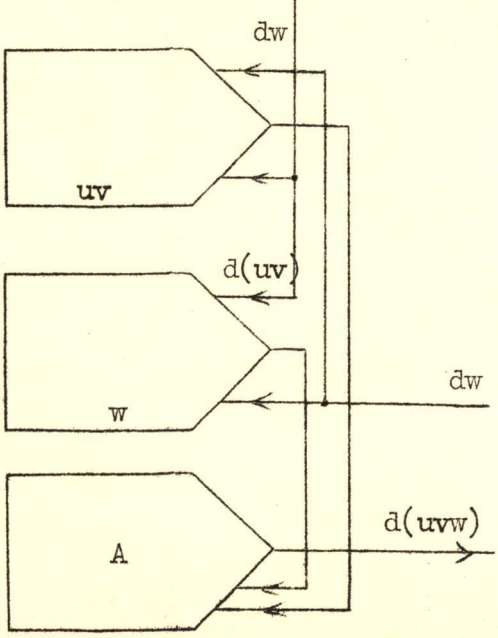


Multiplication by a constant

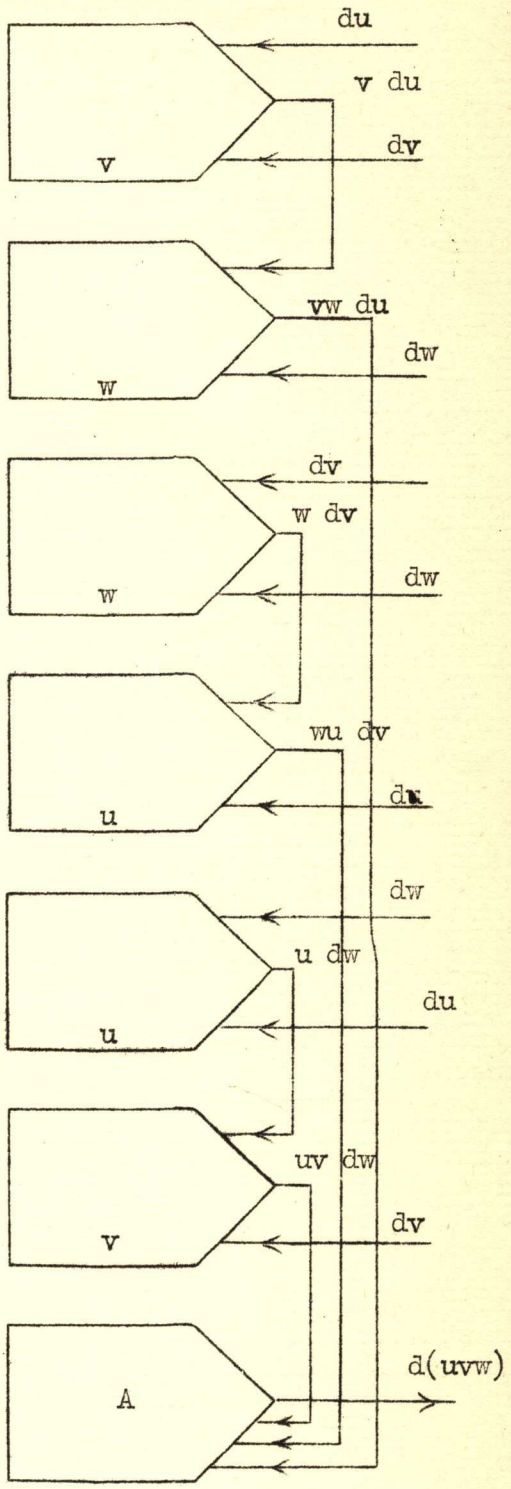


Multiplication of 2 variables
$d(uv) = u\,dv + v\,du$

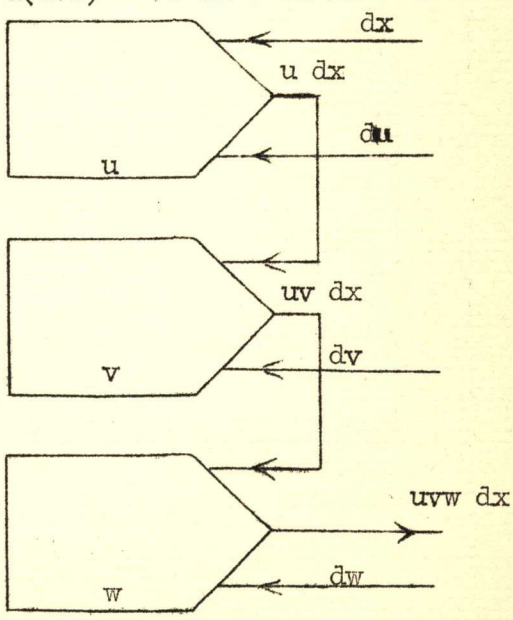




Multiplication of 3 variables
$d(uvw) = uv\,dw + w\,d(uv)$

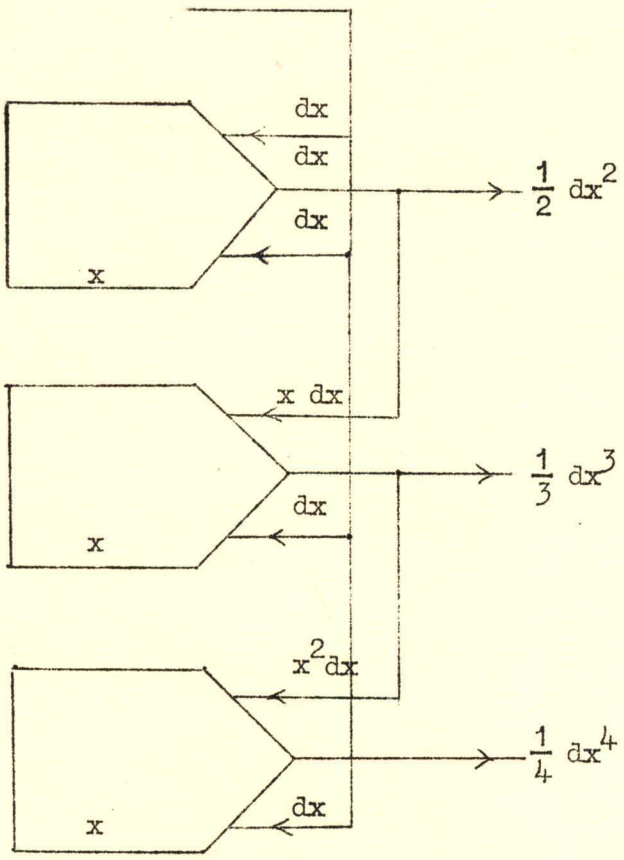


Multiplication of 3 variables
$d(uvw) = vw\,du + wu\,dv + uv\,dw$

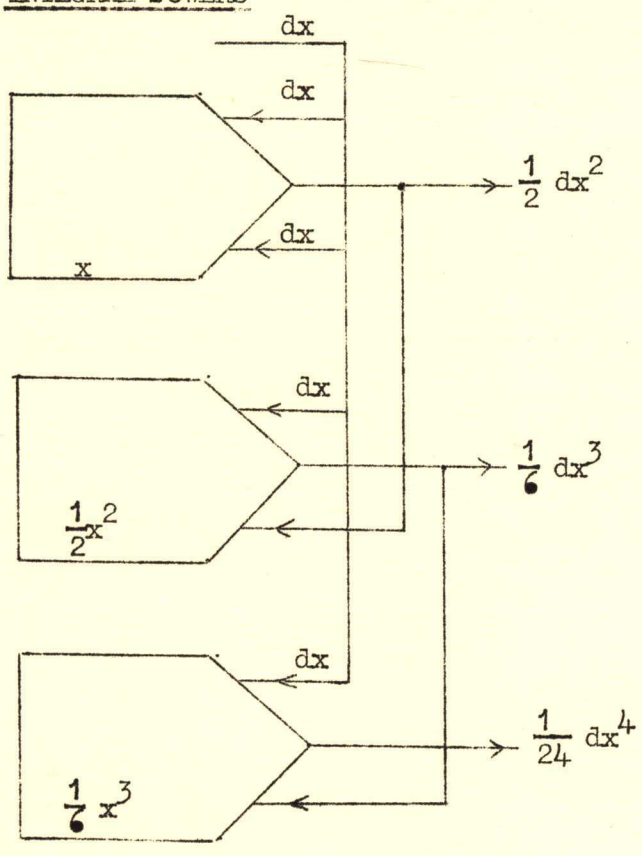


uvw dx form of multiplication

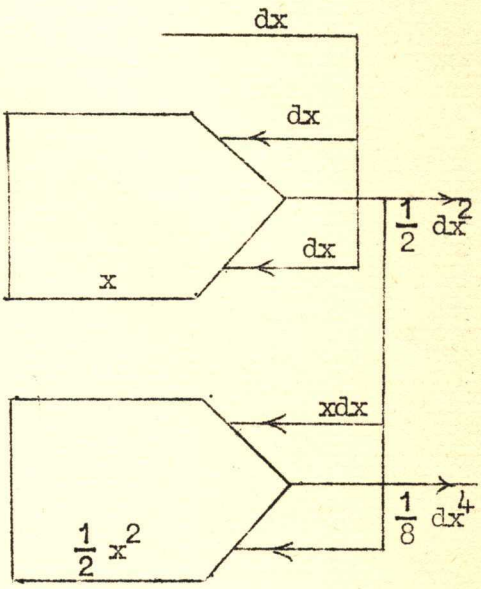
ALGEBRAIC FUNCTIONS - POWER OF A VARIABLE



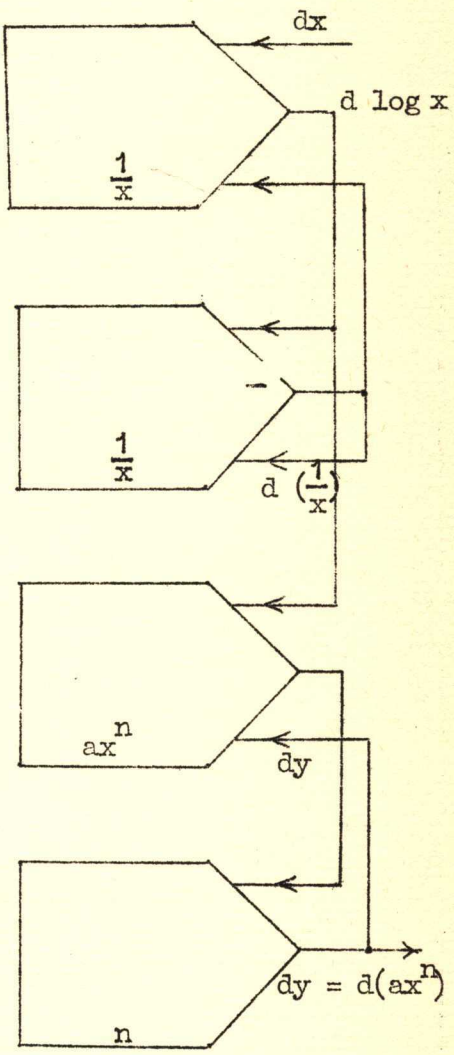
INTEGRAL POWERS



INTEGRAL POWERS

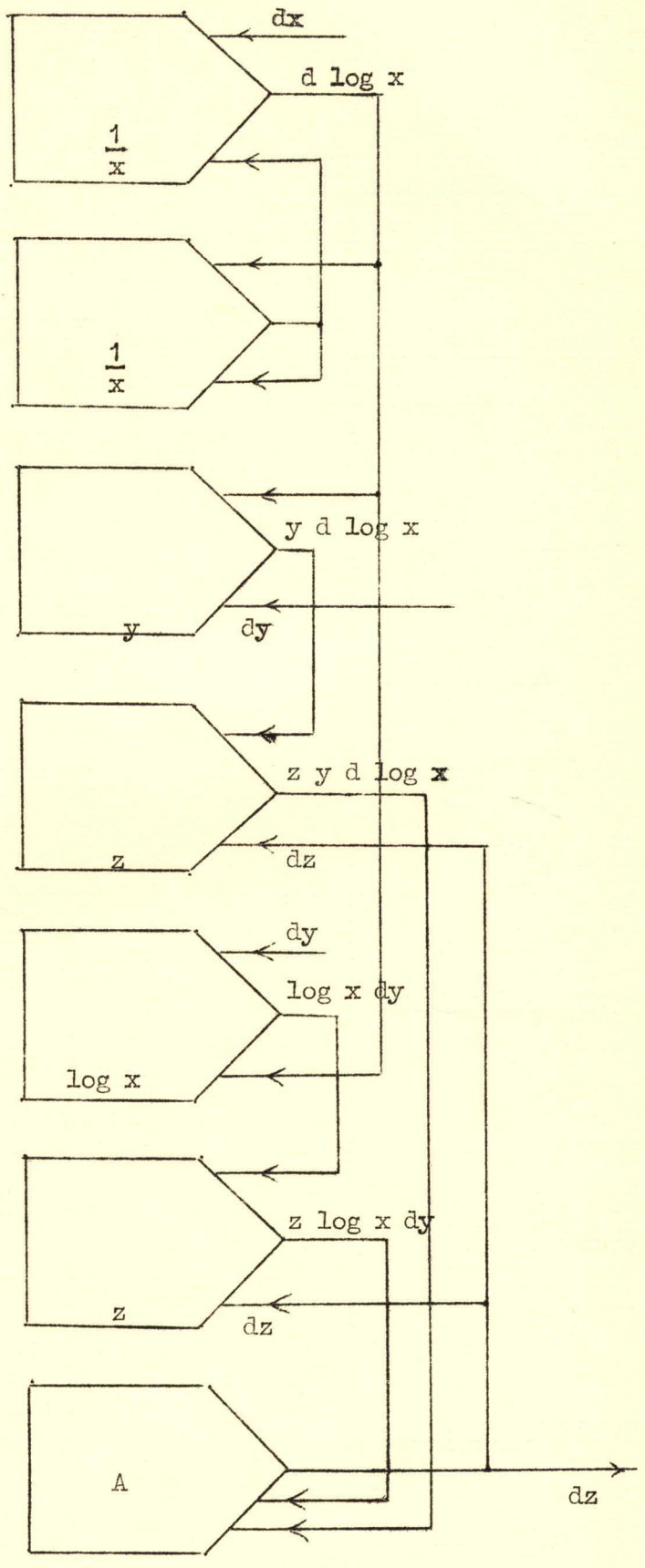


INTEGRAL POWERS



INTEGRAL AND NON-INTEGRAL POWERS

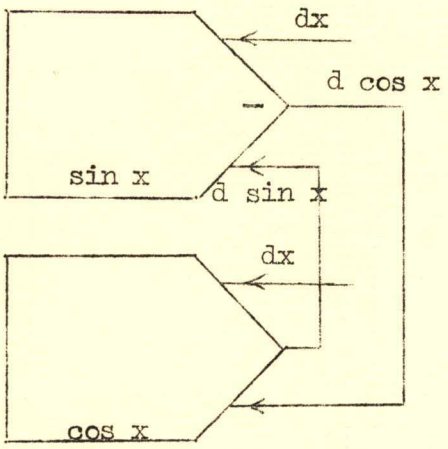
$y = a x^n$



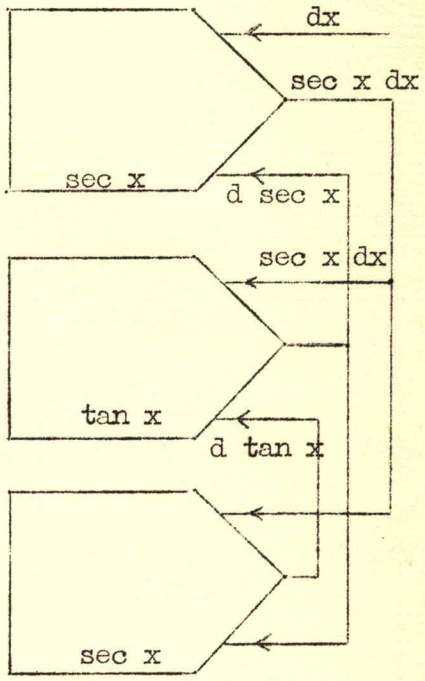
VARIABLE POWER OF A VARIABLE

$z = x^y$

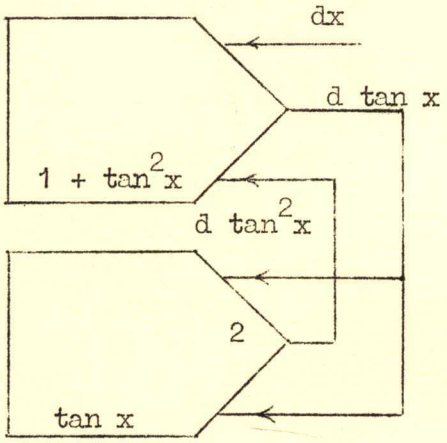
A TRIGONOMETRIC FUNCTIONS



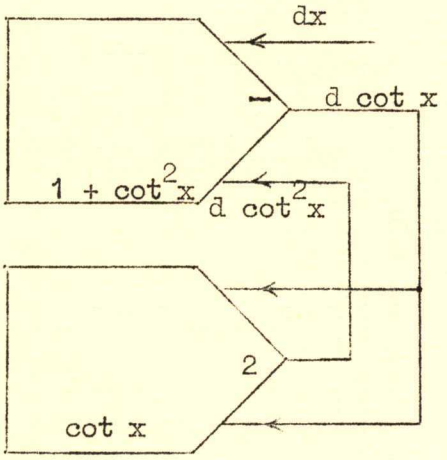
sin x & cos x



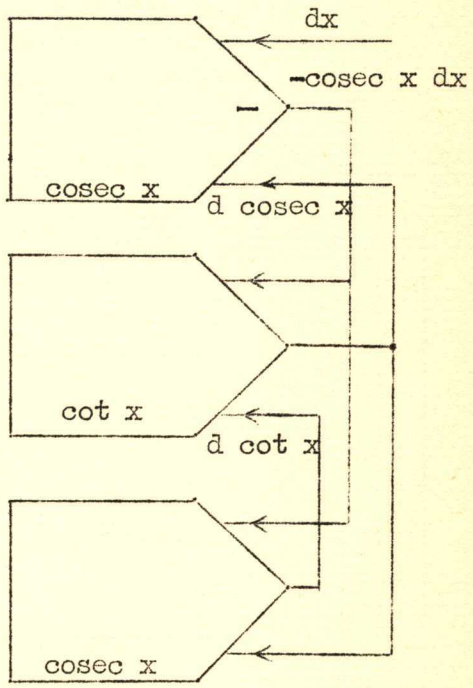
Tan x & Sec x



Tan x, $\tan^2 x$ & $\sec^2 x$

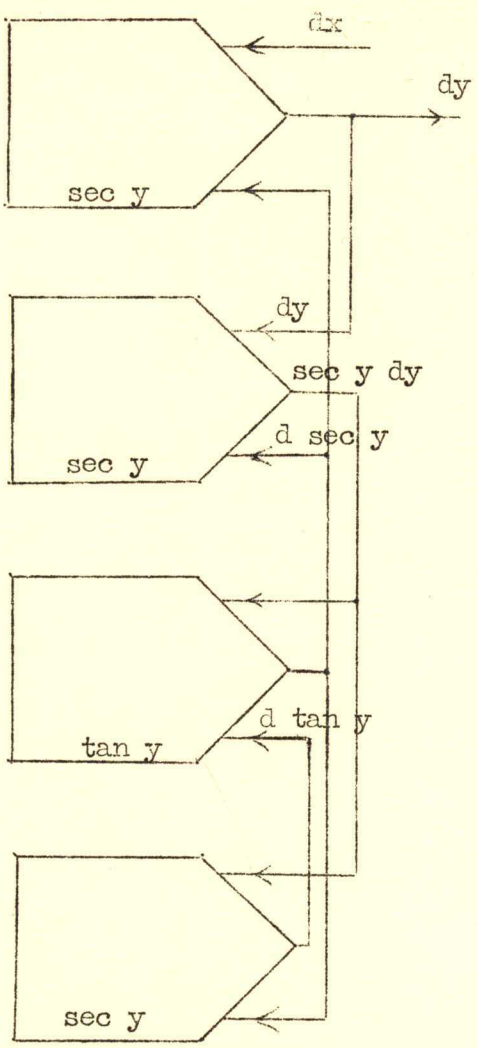


Cot x, $\cot^2 x$ & $\operatorname{cosec}^2 x$

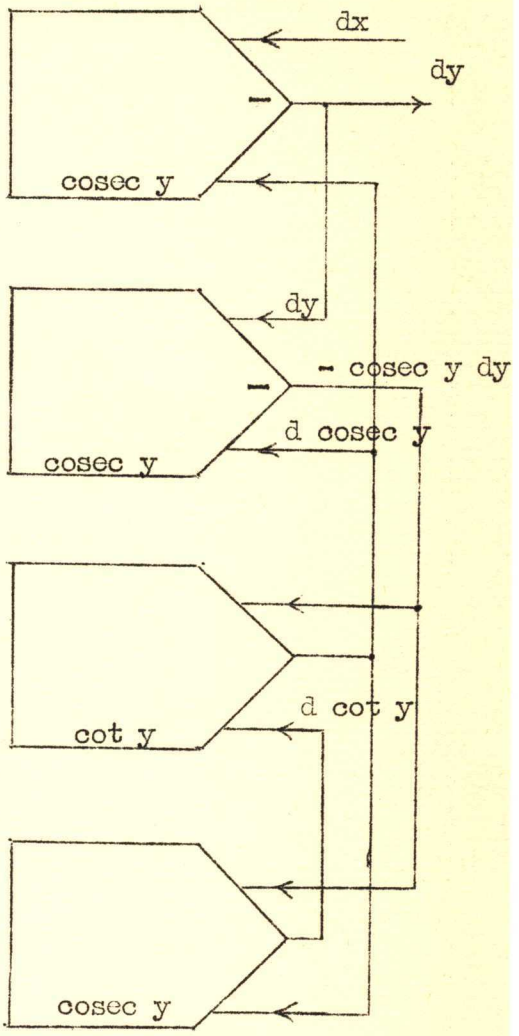


Cot x & Cosec x

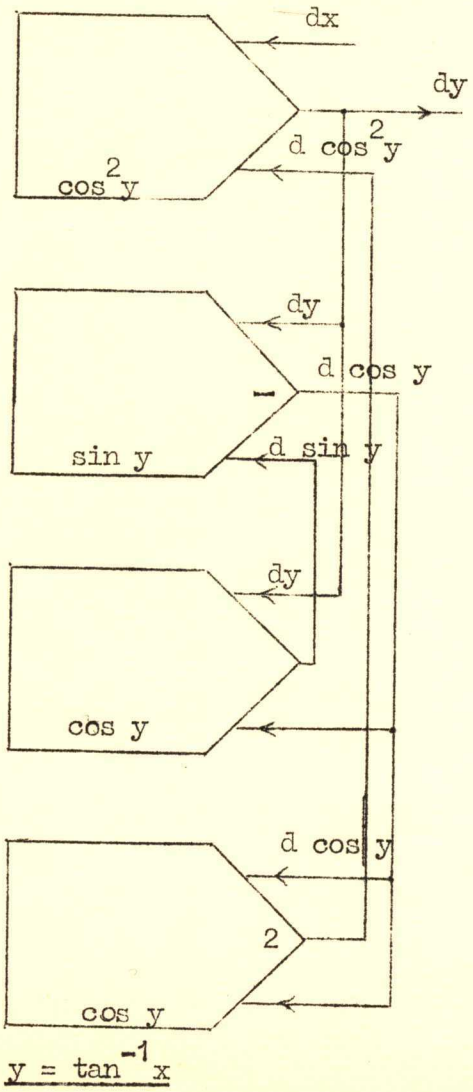
INVERSE TRIGONOMETRIC FUNCTIONS



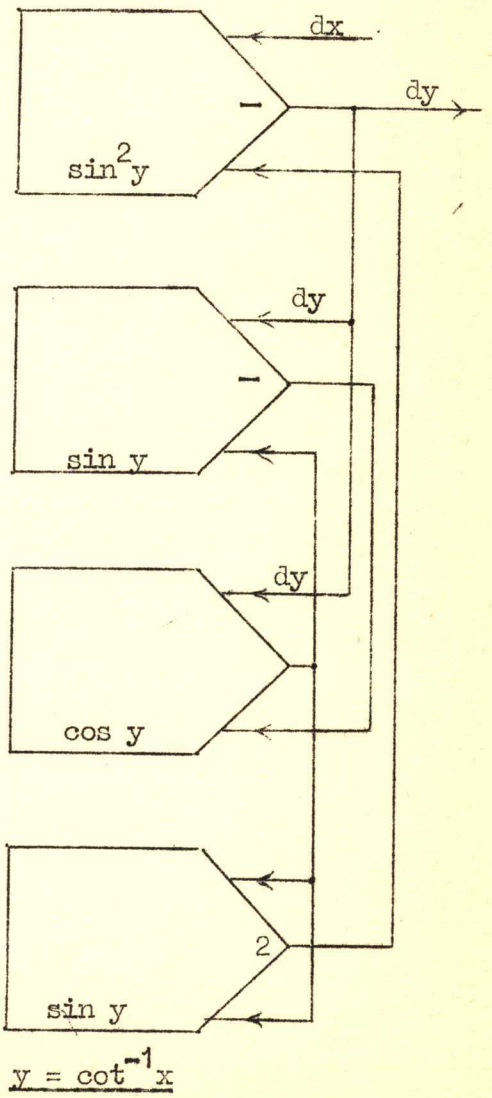
$y = \sin^{-1} x$



$y = \cos^{-1} x$

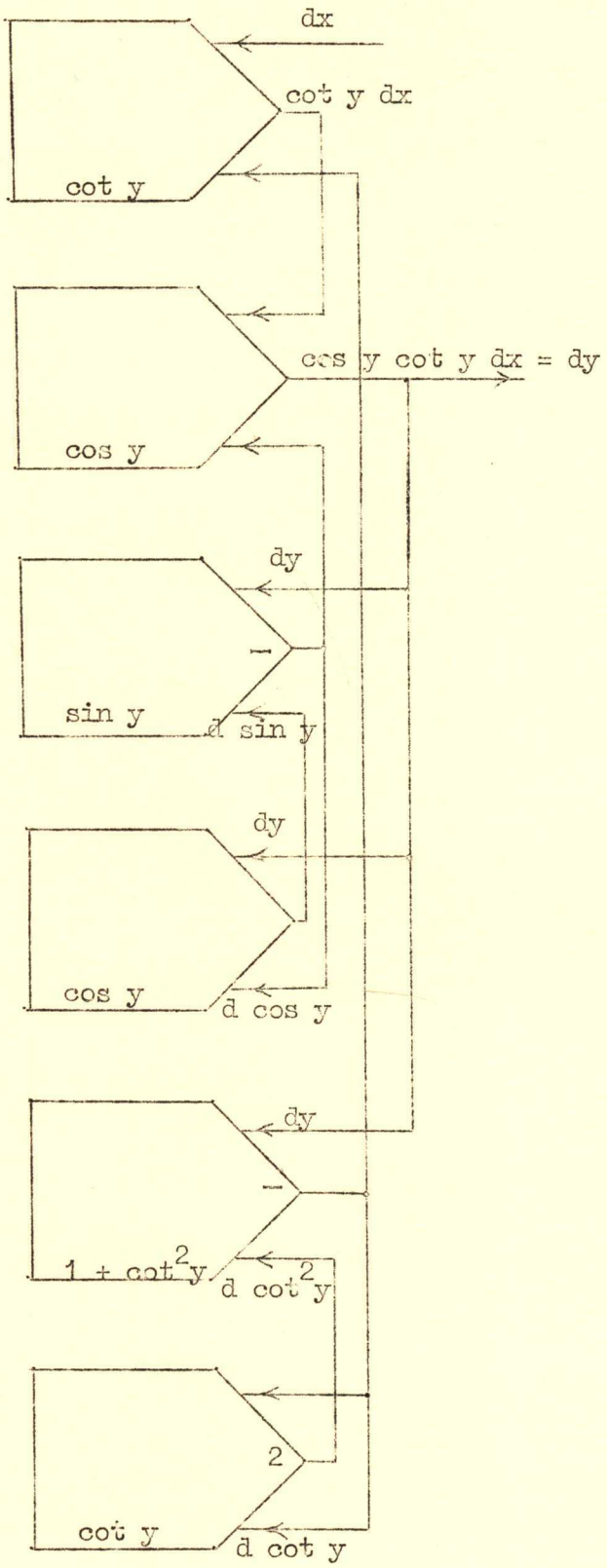


$y = \tan^{-1} x$

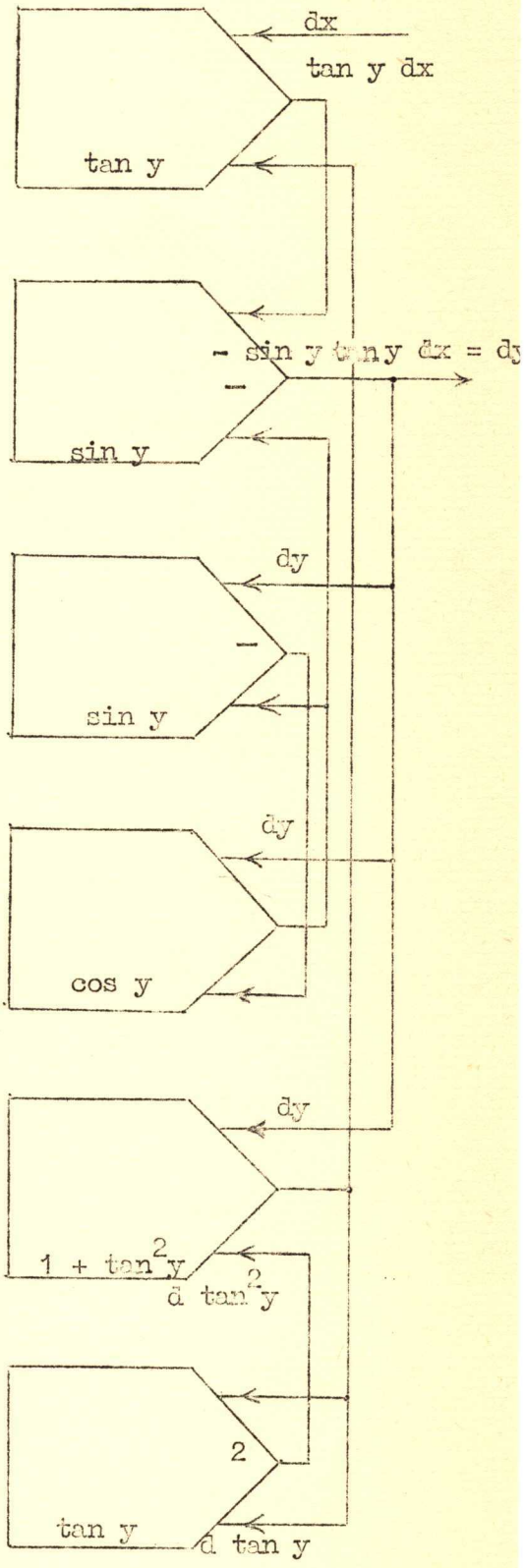


$y = \cot^{-1} x$

B INVERSE TRIGONOMETRIC FUNCTIONS

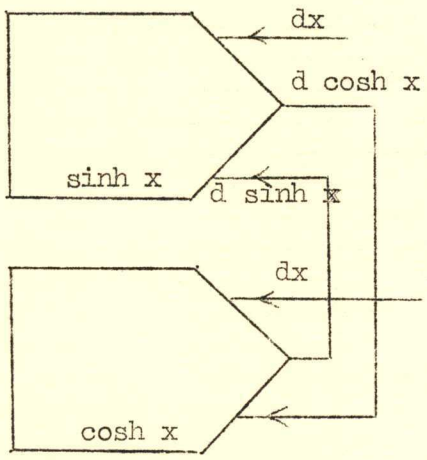


$y = \sec^{-1} x$

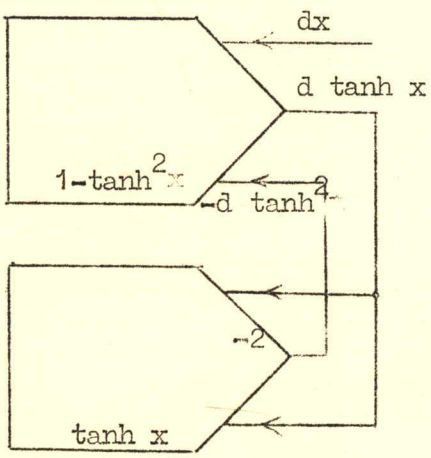


$y = \operatorname{cosec}^{-1} x$

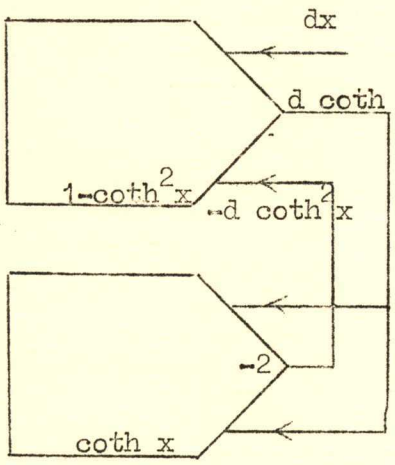
HYPERBOLIC FUNCTIONS



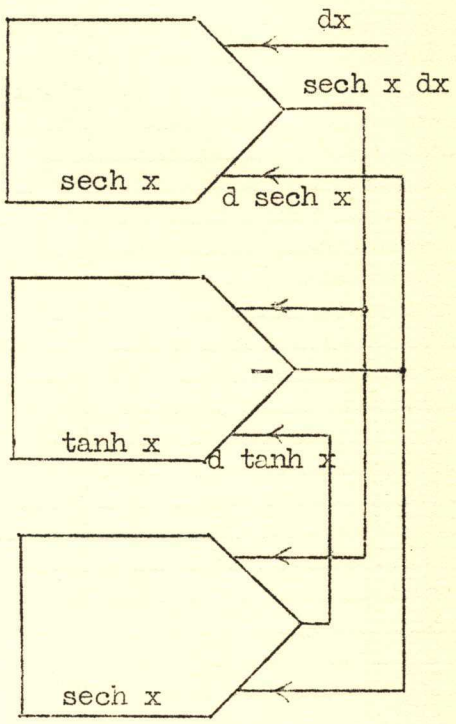
SINH x — COSH x



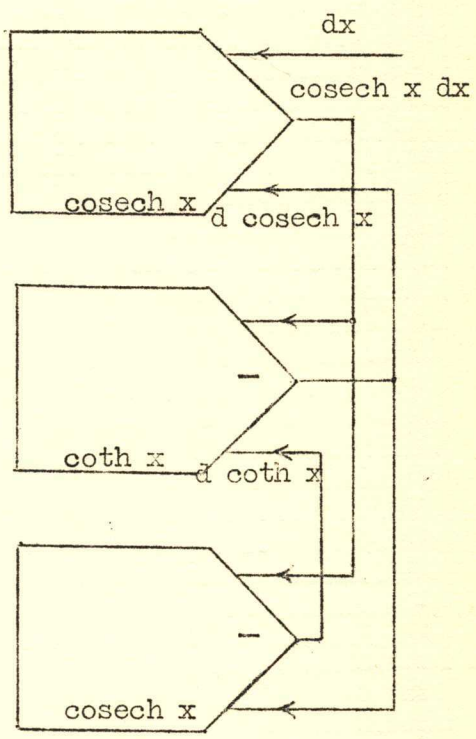
TANH x, $TANH^2x$, $SECH^2x$



COTH x, $COTH^2x$, $COSECH^2x$



TANH x — SECH x



COTH x — COSECH x

A.D.A. CODING SHEET

INTEG. No.	ΔX INTEG.	ΔY INTEGRATORS	ΔI INTEG.	FIXED INSTRUCTIONS	INITIAL CONDITIONS
@ 0 0 @	@	@	@	0 @	0 0 @
@ 0 1 @	@	@	@	0 @	0 0 @
@ 0 2 @	@	@	@	0 @	0 0 @
@ 0 3 @	@	@	@	0 @	0 0 @
@ 0 4 @	@	@	@	0 @	0 0 @
@ 0 5 @	@	@	@	0 @	0 0 @
@ 0 6 @	@	@	@	0 @	0 0 @
@ 0 7 @	@	@	@	0 @	0 0 @
@ 0 8 @	@	@	@	0 @	0 0 @
@ 0 9 @	@	@	@	0 @	0 0 @
@ 1 0 @	@	@	@	0 @	0 0 @
@ 1 1 @	@	@	@	0 @	0 0 @
@ 1 2 @	@	@	@	0 @	0 0 @
@ 1 3 @	@	@	@	0 @	0 0 @
@ 1 4 @	@	@	@	0 @	0 0 @
@ 1 5 @	@	@	@	0 @	0 0 @
@ 1 6 @	@	@	@	0 @	0 0 @
@ 1 7 @	@	@	@	0 @	0 0 @
@ 1 8 @	@	@	@	0 @	0 0 @
@ 1 9 @	@	@	@	0 @	0 0 @
@ 2 0 @	@	@	@	0 @	0 0 @
@ 2 1 @	@	@	@	0 @	0 0 @
@ 2 2 @	@	@	@	0 @	0 0 @
@ 2 3 @	@	@	@	0 @	0 0 @
@ 2 4 @	@	@	@	0 @	0 0 @
@ 2 5 @	@	@	@	0 @	0 0 @
@ 2 6 @	@	@	@	0 @	0 0 @
@ 2 7 @	@	@	@	0 @	0 0 @
@ 2 8 @	@	@	@	0 @	0 0 @
@ 2 9 @	@	@	@	0 @	0 0 @
@ 3 0 @	@	@	@	0 @	0 0 @
@ 3 1 @	@	@	@	0 @	0 0 @
@ 3 2 @	@	@	@	0 @	0 0 @
@ 3 3 @	@	@	@	0 @	0 0 @
@ 3 4 @	@	@	@	0 @	0 0 @
@ 3 5 @	@	@	@	0 @	0 0 @
@ 3 6 @	@	@	@	0 @	0 0 @
@ 3 7 @	@	@	@	0 @	0 0 @
@ 3 8 @	@	@	@	0 @	0 0 @
@ 3 9 @	@	@	@	0 @	0 0 @
@ 4 0 @	@	@	@	0 @	0 0 @
@ 4 1 @	@	@	@	0 @	0 0 @
@ 4 2 @	@	@	@	0 @	0 0 @
@ 4 3 @	@	@	@	0 @	0 0 @
@ 4 4 @	@	@	@	0 @	0 0 @
@ 4 5 @	@	@	@	0 @	0 0 @
@ 4 6 @	@	@	@	0 @	0 0 @
@ 4 7 @	@	@	@	0 @	0 0 @
@ 4 8 @	@	@	@	0 @	0 0 @
@ 4 9 @	@	@	@	0 @	0 0 @
@ 5 0 @	@	@	@	0 @	0 0 @
@ 5 1 @	@	@	@	0 @	0 0 @
@ 5 2 @	@	@	@	0 @	0 0 @
@ 5 3 @	@	@	@	0 @	0 0 @
@ 5 4 @	@	@	@	0 @	0 0 @
@ 5 5 @	@	@	@	0 @	0 0 @
@ 5 6 @	@	@	@	0 @	0 0 @
@ 5 7 @	@	@	@	0 @	0 0 @
@ 5 8 @	@	@	@	0 @	0 0 @
@ 5 9 @	@	@	@	0 @	0 0 @

DERIVATION OF THE SCALING EQUATIONS FOR ADA.

The scale factors (K_Y, K_x, K_y, K_z) relating the computer values (Y, dX, dY, dZ) to the problem values (Y_p, dX_p, dY_p, dZ_p) are defined by the following equations:-

$$Y = K_Y Y_p \quad .....(1)$$

$$dX = K_x dX_p \quad .....(2)$$

$$dY = K_y dY_p \quad .....(3)$$

$$dZ = K_z dZ_p \quad .....(4)$$

Now, dX, dY and dZ are outputs of single integrators, and are thus limited to the values +1, 0 and -1. Hence, if $K_x = 10^3$, a dX of +1 corresponds to a dX_p of 10^{-3}.

The absolute value of the integrand of an integrator has the maximum value of $\frac{1}{M}$, where M is the integrator output multiplier of 1, 2 or 5; i.e. K_Y and M must be chosen so that:-

$$1 > M K_Y |Y_P|_{max} \quad .....(5)$$

The equation of an integrator is:-

$$dZ = M Y dX \quad .....(6)$$

Hence, by introducing the scale factor relations (equations 1, 2, 3 and 4), we have:-

$$K_z dZ_p = M K_Y Y_p K_x dX_p \quad .....(7)$$

The problem equation corresponding to equation 6 is:-

$$dZ_p = Y_p dX_p \quad .....(8)$$

Hence, to satisfy equations 7 and 8, we have:-

$$K_z = M K_Y K_x \quad .....(9)$$

The Y register of an integrator accumulates the running total of the secondary incremental inputs (dY) multiplied by a predetermined factor (h), so that:-

$$Y = Y_o + h\sum dY \quad .....(10)$$

where Y_o is the initial value of the integrand.

By introducing the scale factor relations, we have:-

$$K_Y\ Y_p = K_Y\ Y_{op} + h\sum K_y\ dY_p \quad .....(11)$$

The problem equation corresponding to equation 10 is:-

$$Y_p = Y_{op} + \sum dY_p \quad .....(12)$$

Hence, to satisfy equations 11 and 12 :-

$$K_Y = h\ K_y \quad .....(13)$$

The factor (h) is made equal to unity in the least significant decimal place of the register, i.e. :-

$$h = 10^{-N} \quad .....(14)$$
$$(1 \leqslant N \leqslant 7)$$

where N is the required number of significant decimal digits of the register. Hence, from equations 13 and 14 :-

$$K_Y = 10^{-N}\ K_y \quad .....(15)$$

From equation 15, we have :-

$$N = \log_{10} K_y - \log_{10} K_Y \quad .....(16)$$

The necessary scaling relationships for an integrator have now been established and are restated below:-

$$1 > M\ K_Y\ |Y_p|_{max} \quad (M = 1, 2 \text{ or } 5) \quad .....****$$

$$K_z = M\ K_Y\ K_x \quad (M = 1, 2 \text{ or } 5) \quad .....****$$

$$N = \log_{10} K_y - \log_{10} K_Y \quad (1 \leqslant N \leqslant 7) \quad .....****$$

ORDER CODE
of the
GENERAL-PURPOSE DIGITAL COMPUTER "SNOCOM".

Code	Instruction (LGP-30)	Instruction (SNOCOM)	Effect
0001	B m	1 m	Bring. Clear the accumulator and add the contents of location m to it.
1110	A m	F m	Add the contents of m to the contents of the accumulator and retain the result in the accumulator.
1111	S m	L m	Subtract the contents of m from the contents of the accumulator and retain the result in the accumulator.
0111	M m	7 m	Multiply the number in the accumulator by the number in memory location m, terminating the result at 30 binary places.
0110	N m	6 m	Multiply the number in the accumulator by the number in m, retaining the least significant half of the product.
0101	D m	5 m	Divide the number in the accumulator by the number in m retaining the rounded quotient in the accumulator.
1001	E m	9 m	Extract or logical product, i.e. clear the contents of the accumulator to zero in those bit positions occupied by zeros in m.
1010	U m	+ m	Transfer control to m unconditionally, i.e. get the next instruction from m.

Code	Instruction (LGP-30)	Instruction (SNOCOM)	Effect
1011	T m	- m	Test or conditional transfer. Transfer control to m only if accumulator is negative. (N.B. By correct programming, this conditional transfer order can be made to depend on the setting of an external transfer switch).
1100	H m	N m	Hold. Store contents of accumulator in m retaining the number in the accumulator.
1101	C m	J m	Clear. Store contents of accumulator in m and clear the accumulator.
0010	Y m	2 m	Store only the address part of the word in the accumulator in memory location m leaving the rest of the work undisturbed in memory.
0011	R m	3 m	Return address. Add "one" to the address held in counter register (C) and record in the address portion of the instruction in memory Location m. (The counter register normally holds the address of the next instruction to be executed.)
0000	Z t	0 t	Stop. Unconditional stop if t = 0. Waste order if t = 1. Otherwise contingent on B.P. switch settings on Control panel.
1000	P t	8 t	Punch or print the character set up in the track number part of the address. If the weight 32 (sixth) bit is 0 then punch, but if it is 1 then print. (Note 5 bits only are decoded for output).

Code	Instruction (LGP-30)	Instruction (SNOCOM)	Effect
0100	I	4	Input. Shift the contents of the accumulator 4 (binary) places to the left, read the character assembled in the input flip-flops into the 4 least significant places of the accumulator and then assemble the next character from the tape reader into the input flip-flops. If the first character read is the control character / (stroke) then repeat the above procedure until the second control character ## (number) is found.

N.B. The address part of the instructions shown above is denoted by m when it refers to a memory location, and by t when only the track number is significant. The address part of an input order has no significance.

PUNCH ORDERS

D.O.I. ORDER	BINARY EQUIV. OF TRACK NO.	"SILLIAC" FIGURES	REPRODUCER LETTERS	HEXADECIMAL ORDER
/8 0000##	000000	0	P	/00800000##
/8 0100##	000001	1	Q	/00800200##
/8 0200##	000010	2	W	/00800400##
/8 0300##	000011	3	E	/00800600##
/8 0400##	000100	4	R	/00800800##
/8 0500##	000101	5	T	/00800+00##
/8 0600##	000110	6	Y	/00800N00##
/8 0700##	000111	7	U	/00800F00##
/8 0800##	001000	8	I	/00801000##
/8 0900##	001001	9	O	/00801200##
/8 1000##	001010	+	K	/00801400##
/8 1100##	001011	-	S	/00801600##
/8 1200##	001100	N	N	/00801800##
/8 1300##	001101	J	J	/00801+00##
/8 1400##	001110	F	F	/00801N00##
/8 1500##	001111	L	L	/00801F00##
/8 1600##	010000	Delay	Delay	/00802000##
/8 1700##	010001	##	D	/00802200##
/8 1800##	010010	CR/LF	CR/LF	/00802400##
/8 1900##	010011	(	B	/00802600##
/8 2000##	010100	letter shift	letter shift	/00802800##
/8 2100##	010101	,	V	/00802+00##
/8 2200##	010110	)	A	/00802N00##
/8 2300##	010111	/	X	/00802F00##
/8 2400##	011000	space	space	/00803000##
/8 2500##	011001	=	G	/00803200##
/8 2600##	011010	.	M	/00803400##
/8 2700##	011011	figure shift	figure shift	/00803600##
/8 2800##	011100	'	H	/00803800##
/8 2900##	011101	:	C	/00803+00##
/8 3000##	011110	x	Z	/00803F00##
/8 3100##	011111	erase	erase	/00803F00##

PUNCH ORDERS

D.O.I. ORDER	BINARY EQUIV. OF TRACK NO.	"E.E.D." REPRODUCER FIGURES	LETTERS	HEXADECIMAL ORDER
/8 0000##	000000	0	Ω	/00800000##
/8 0100##	000001		1	/00800200##
/8 0200##	000010	5	2	/00800400##
/8 0300##	000011	3	3	/00800600##
/8 0400##	000011	4	4	/00800800##
/8 0500##	000101		5	/00800+00##
/8 0600##	000110		6	/00800N00##
/8 0700##	000111		7	/00800F00##
/8 0800##	001000	8	8	/00801000##
/8 0900##	001001		9	/00801200##
/8 1000##	001010		+	/00801400##
/8 1100##	001011		-	/00801600##
/8 1200##	001100		N	/00801800##
/8 1300##	001101	@	J	/00801+00##
/8 1400##	001110		F	/00801N00##
/8 1500##	001111		L	/00801F00##
/8 1600##	010000	1	Delay	/00802000##
/8 1700##	010001		##	/00802200##
/8 1800##	010010	6	CR/LF	/00802400##
/8 1900##	010011		(	/00802600##
/8 2000##	010100	3		/00802800##
/8 2100##	010101		,	/00802+00##
/8 2200##	010110	8	)	/00802N00##
/8 2300##	010111	/	/	/00802F00##
/8 2400##	011000	2	space	/00803000##
/8 2500##	011001		=	/00803200##
/8 2600##	011010	7	.	/00803400##
/8 2700##	011011	figure shift	figure shift	/00803600##
/8 2800##	011100	4	'	/00803800##
/8 2900##	011101		:	/00803+00##
/8 3000##	011110	9	X	/00803N00##
/8 3100##	011111	letter shift	letter shift	/00803F00##

PRINT ORDERS.

D.O.I. ORDER	BINARY EQUIVALENT OF TRACK NO.	CHARACTER PRINTED	HEXADECIMAL ORDER
/8 3200##	100000	0	/00804000##
/8 3300##	100001	1	/00804200##
/8 3400##	100010	2	/00804400##
/8 3500##	100011	3	/00804600##
/8 3600##	100100	4	/00804800##
/8 3700##	100101	5	/00804+00##
/8 3800##	100110	6	/00804N00##
/8 3900##	100111	7	/00804F00##
/8 4000##	101000	8	/00805000##
/8 4100##	101001	9	/00805200##
/8 4200##	101010	+	/00805400##
/8 4300##	101011	-	/00805600##
/8 4400##	101100	N	/00805800##
/8 4500##	101101	J	/00805+00##
/8 4700##	101110	F	/00805N00##
/8 4700##	101111	L	/00805F00##
/8 5000##	110010	CR/LF	/00806400##
/8 5600##	111000	space	/00807000##
/8 5800##	111010	.	/00807400##

BREAK POINT ORDERS

LGP-30 symbol : Z

SNOCOM symbol : 0

There are 5 break point switches on the control panel. These are designated BP02, BP04, BP08, BP16 and BP32.

The computed will stop on a break point order if any of these switches is down and the corresponding binary digit of the track number is a "1".(The sector number has no significance).

For example, the D.0.1. order /0 2600## will stop the computer if any one of the 3 switches BP16, BP08 or BP02 is down, as the binary equivalent of the track number is 011010 (i.e. 1's occur in the weight 16, weight 8 and weight 2 positions).

The computer will also stop unconditionally if the track number is zero.

Apart from helping to detect the unconditional stop, the least significant binary digit of the track number has no significance.

ABBREVIATED LIST OF BREAK POINT ORDERS.

D.0.I ORDER	HEXADECIMAL ORDER	EFFECT
/0 0000##	/00000000##	Unconditional stop
/0 0100##	/00000200##	Waste order
/0 0200##	/00000400##	Computer stops if BP02 is down
/0 0400##	/00000800##	" " " BP04 " "
/0 0800##	/00001000##	" " " BP08 " "
/0 1600##	/00002000##	" " " BP16 " "
/0 3200##	/00004000##	" " " BP32 " "

THE EXTERNAL TRANSFER SWITCH Z_0.

Manual control of a program in the computer is made possible by the external transfer switch Z_0.

The conditional transfer control order (LGP-30 symbol: "T"; SNOCOM symbol:"-") has the same effect as an unconditional transfer control order (LGP-30 symbol "U"; SNOCOM symbol "+") provided the sign (t) digit position of the accumulator is occupied by a "1", or if the corresponding digit of the instruction is a "1" and the external transfer switch Z_0 is closed.

The underlined words "or" and "and" in the above sentence must be given special attention. The normal conditional transfer control order will have the hexadecimal form

/00-0xyz0##

where xyz is a hexadecimal address. The sign digit of the order is "0", and therefore the effect of the order will depend only upon the sign digit of the accumulator.

The order

/80-0xyz0##

has a "1" in the sign digit position. Because of the significance of the "or" in the original statement of the order, if the above order is to depend only on the setting of the external transfer switch, it is necessary to ensure that the accumulator is positive when this order is obeyed. This can be carried out by preceding the order with a "clear accumulator" order.

The effect of the conditional transfer order and the Z_0 switch is summarised extremely well by equation 19 of Frankel's paper:-

$$\bar{Q}_4' = Q_1\bar{Q}_2Q_3 \ (A + RZ_0) \ t +$$

When the required conditions are satisfied, this equation transforms a conditional transfer control order to an unconditional transfer control order in $\emptyset 3$. It is to be

noted that these 2 orders are distinguished only by the setting of the Q_4 flip-flop.

The Z_o switch can be used to advantage when it is required to use the same program tape to carry out two related but different operations. Branching of the program can be made contingent on the Z_o switch setting by the following orders:-

/00J0 abc0≠≠ "clears accumulator"

/80-0 xyz0≠≠ "contingent on Z_o switch"

where abc is the address of any temporary storage and xyz is the address to which control is transferred if the Z_o switch is closed.

If the program requires 3 branches instead of only 2, the Z_o switch can be used in connection with the unconditional stop order. Consider the following orders:-

/00J0abc0≠≠

/80-0 xyz0≠≠

/0000 0000≠≠

/80-0 pqr0≠≠

If Z_o is initially closed, control will be transferred to xyz after the computer obeys the second order. If Z_o is initially open, the computer will stop on obeying the third (unconditional stop) order. If Z_o is now closed, on restarting control will be transferred to pqr on obeying the fourth order. If Z_o is left open the next order in sequence will be obeyed.

CONSTANTS USED WITH "7" & "6" ORDERS

FOR RIGHT & LEFT SHIFTS

CONSTANT	CONSTANT IN HEXADECIMAL	"M" ORDER RIGHT SHIFT NO. OF PLACES	"N" ORDER LEFT SHIFT NO. OF PLACES
1 @ o	/80000000##		31
1 @ 1	/40000000##	1	30
1 @ 2	/20000000##	2	29
1 @ 3	/10000000##	3	29
1 @ 4	/08000000##	4	27
1 @ 5	/04000000##	5	26
1 @ 6	/02000000##	6	25
1 @ 7	/01000000##	7	24
1 @ 8	/00800000##	8	23
1 @ 9	/00400000##	9	22
1 @10	/00200000##	10	21
1 @11	/00100000##	11	20
1 @12	/00080000##	12	19
1 @13	/00040000##	13	18
1 @14	/00020000##	14	17
1 @15	/00010000##	15	16
1 @16	/00008000##	16	15
1 @17	/00004000##	17	14
1 @18	/00002000##	18	13
1 @19	/00001000##	19	12
1 @20	/00000800##	20	11
1 @21	/00000400##	21	10
1 @22	/00000200##	22	9
1 @23	/00000100##	23	8
1 @24	/00000080##	24	7
1 @25	/00000040##	25	6
1 @26	/00000020##	26	5
1 @27	/00000010##	27	4
1 @28	/00000008##	28	3
1 @29	/00000004##	29	2
1 @30	/00000002##	30	1
1 @31	/00000001##		

CONVERSION TABLE
FROM TRACK & SECTOR NUMBERS
TO HEXADECIMAL ADDRESSES.

TRACK	FIRST 2 HEX. CHAR. SECTOR 00-15	SECTOR 16-31
00	00	01
01	02	03
02	04	05
03	06	07
04	08	09
05	0+	0-
06	0N	0J
07	0F	0L
08	10	11
09	12	13
10	14	15
11	16	17
12	18	19
13	1+	1-
14	1N	1J
15	1F	1L
16	20	21
17	22	23
18	24	25
19	26	27
20	28	29
21	2+	2-
22	2N	2J
23	2F	2L
24	30	31
25	32	33
26	34	35
27	36	37
28	38	39
29	3+	3-
30	3N	3J
31	3F	3L
32	40	41
33	42	43
34	44	45
35	46	47
36	48	49
37	4+	4-
38	4N	4J
39	4F	4L
40	50	51
41	52	53
42	54	55
43	56	57
44	58	59
45	5+	5-
46	5N	5J
47	5F	5L
48	60	61
49	62	63
50	64	65
51	66	67
52	68	69
53	6+	6-
54	6N	6J
55	6F	6L
56	70	71
57	72	73
58	74	75
59	76	77
60	78	79
61	7+	7-
62	7N	7J
63	7F	7L

SECTOR	THIRD HEX. CHAR.
00	0
01	1
02	2
03	3
04	4
05	5
06	6
07	7
08	8
09	9
10	+
11	-
12	N
13	J
14	F
15	L
16	0
17	1
18	2
19	3
20	4
21	5
22	6
23	7
24	8
25	9
26	+
27	-
28	N
29	J
30	F
31	L

"SNOCOM" SECTOR NUMBER SEQUENCE.

SECTOR NUMBER	BINARY EQUIVALENT
0	00000
25	11001
18	10010
11	01011
4	00100
29	11101
22	10110
15	01111
8	01000
1	00001
26	11010
19	10011
12	01100
5	00101
30	11110
23	10111
16	10000
9	01001
2	00010
27	11011
20	10100
13	01101
6	00110
31	11111
24	11000
17	10001
10	01010
3	00011
28	11100
21	10101
14	01110
7	00111
0	00000

HEXADECIMAL INPUT ROUTINE. 1.

The following routine will be stored in locations 0000 to 0015. Hence, control may be transferred to the first order of the routine simply by clearing the C register.

The routine will input words coded in hexadecimal, and will store them in consecutive storage storage locations beginning at location 0400. (Sector 00 in track 04 was chosen so that the 0-3 block record switch could be used to prevent the possible mutilation of this hexadecimal input routine).

A word with a "1" in the waste digit position signifies the end of the program being filled. When the hexadecimal input routine detects the end of the program, it will store the last word and then transfer control to location 0400. (It is to be noted that the waste digit position of all words recorded in and played back from the store is always "0". However the waste digit position of the accumulator when filled from the tape reader may be "0" or "1").

The program utilises the temporary storage locations 6300 and 6301. Some temporary storage is necessary as the routine requires the modification of addresses; this is not possible for orders in track 00 as it was anticipated that the 0-3 block record switch would put on after the routine had been successfully "pedalled" into the computer.

HEXADECIMAL INPUT ROUTINE IN DECIMAL FORM.

LOCATION	ORDER	
0000	1 0013	
0001	N 6300	
0002	1 0014	
0003	J 6301	
0004	4 0000	
0005	+ 6300	
0006	6 0015	
0007	- 0400	
0008	1 6300	
0009	F 0012	
0010	J 6300	
0011	+ 0004	
0012	1 @ 27	/00000010##
0013	N 0400	
0014	+ 0006	
0015	1 @ 0	/80000000##

The orders in locations 0000 to 0003 fill the temporary storage as follows:-

LOCATION	ORDER
6300	N (0400)
6301	+ 0006

The integer multiply order shifts the waste digit of the word filled in the accumulator into the sign digit position so that the following conditional transfer order is able to detect the end of the program being filled.

The orders in locations 0008, 0009 and 0010 are used to augment by unity the address of the hold order in location 6300.

INITIAL FILLING OF THE HEXADECIMAL INPUT ROUTINE.

The hexadecimal input routine in hexadecimal form is shown in the right hand column of the orders shown below (apart from the last order which is introduced for another purpose to be described later).

The routine may be pedalled into the computer if each order is preceded by a "hold" or "hold & clear" order as shown below.

The pedalling procedure is as follows:-

(1) → A & Operate "Fill Accumulator"

(2) A → R & Operate "Transfer A to R"

(3) → A & Operate "Fill Accumulator"

(4) Ex R & Operate "Executes R"

The first two operations places the "hold & clear" order in the R register. The third operation places the word to be filled into the accumulator so that when the fourth operation is carried out the "hold & clear" order is obeyed and the appropriate word is filled into the store.

```
/00J00000##        /001000J0##
/00J00010##        /00N07F00##
/00J00020##        /001000F0##
/00J00030##        /00J07F10##
/00J00040##        /00400000##
/00J00050##        /00+07F00##
/00J00060##        /006000L0##
/00J00070##        /00-00800##
/00J00080##        /00107F00##
/00J00090##        /00F000N0##
/00J000+0##        /00J07F00##
/00J000-0##        /00+00040##
/00J000N0##        /00000010##
/00J000J0##        /00N00800##
/00J000F0##        /00+00060##
/00J000L0##        /80000000##
/00000000##        /00000000##
```

The two unconditional stop orders at the bottom of the two columns are used in the bootstrap filling of the hexadecimal input routine. This will be described under "Bootstrap Routine".

BOOTSTRAP ROUTINE. 1.

The following bootstrap routine will repeatedly input two words and obey the first while the second is in the accumulator. The decimal coding of the routine using locations 6327 to 6331 is as follows:-

LOCATION	ORDER
6327	4 0000
6328	N 6330
6329	4 0000
6330	( )
6331	+ 6327

The bootstrap routine is used to fill a program into the store when each order of the program is preceded by an appropriate "hold" or "hold & clear" order. For example, if the input tape contained the order pair:

/00J00020## /001000F0##

the bootstrap routine will place the second order /001000F0## into memory location 002 (hexadecimal).

The bootstrap routine obviates the necessity for pedalling the hexadecimal input routine using the initial filling push buttons on the control panel.

The execution of the bootstrap routine can be terminated if the first order of a pair of orders is an unconditional stop or an unconditional transfer control order. In the first case, the computer will stop and in the second, control will be transferred to the address specified by the order.

INITIAL FILLING OF THE BOOTSTRAP ROUTINE.

The hexadecimal form of the bootstrap routine is shown below in the right hand column of orders. The four "hold" orders which precede each of these is used for

initial filling in the usual way.

/OONO7L-0≠≠	/00400000≠≠
/OONO7LNO≠≠	/OONO7LFO≠≠
/OONO7LJO≠≠	/00400000≠≠
/OONO7LLO≠≠	/OO+07L-0≠≠

The procedure is as follows:-

1. (a) → A & Operate
   (b) A → R & Operate
   (c) → A & Operate
   (d) Ex . R & Operate

2. Repeat 1.

3. Repeat 1.

4. Repeat 1.

5. (a) A → R & Operate
   (b) Ex . R & Operate
   (c) Compute & Operate

Steps 1 to 4 fill the program into locations 6327, 6328, 6329 and 6331.

Step 5 transfers control to the first word of the program in location 6327.

BOOTSTRAP ROUTINE 2.

The following bootstrap routine will repeatedly input 2 words and obey the first word while the second is in the accumulator. The first 2 words on the tape must be the following order pair:-

N 0004 + 0000

In hexadecimal form these orders are:-

/00N00040## /00+00000##

The decimal coding of the routine using locations 0000 to 0002 is as follows:-

LOCATION	ORDER
0000	4 0000
0001	N 0003
0002	4 0000

The bootstrap routine is used to fill a program into the store when each order of the program is preceded by an appropriate "hold" or "hold & clear" order. For example if the 3rd or 4th words on the input tape were:-

/00J07LL0## /00+07L-0##

the routine will place the order /00+07L-0## into memory location 7LL (hexadecimal).

The execution of the bootstrap routine can be terminated if the first order of a pair of orders is an unconditional stop or an unconditional transfer control order. In the first case, the computer will stop and in the second, control will be transferred to the address specified by the order.

INITIAL FILLING OF THE BOOTSTRAP ROUTINE.

The hexadecimal form of the bootstrap routine is shown below in the right hand column of orders. The 3 "hold" orders which precede each of these is used for initial filling in the usual way.

/OONOOOOO≠≠	/OO4OOOOO≠≠
/OONOOO1O≠≠	/OONOOO3O≠≠
/OONOOO2O≠≠	/OO4OOOOO≠≠

The procedure is as follows:-

1. Initial Set.

2. (a) → A & Operate
   (b) A → R & Operate
   (c) → A & Operate
   (d) Ex R & Operate

3. Repeat 2.

4. Repeat 2.

5. Clear C.

6. Compute & Operate.

Steps 1 to 4 fill the 3 orders into locations 0000, 0001 and 0002.

Steps 5 and 6 transfer control to 0000.

After the execution of the first 3 orders the order /OONOOO4O≠≠ will be held in location 0003 and the order /OO+OOOOO≠≠ will be held in the accumulator.

After the execution of the order in 0003 /OO+OOOOO≠≠ will be held in 0004.

On the execution of the order in 0004 control will be transferred to 0000.

The decimal coding of the relevant orders in the store are now:-

LOCATION	ORDER
0000	4 0000
0001	N 0003
0002	4 0000
0003	( )
0004	+ 0000