Bendix Computer

Copy No. 4 Report No.

Description of

BENDIX D-12

DIGITAL DIFFERENTIAL ANALYZER

Bendix Computer

DIVISION OF BENDIX AVIATION CORPORATION
5630 ARBOR VITAE STREET, LOS ANGELES 45, CALIFORNIA

Description of BENDIX D-12

DIGITAL DIFFERENTIAL ANALYZER

bу

Lowell S. Michels

BENDIX COMPUTER DIVISION
Bendix Aviation Corporation
5630 Arbor Vitae Street
Los Angeles 45, California

March 13, 1954

INTRODUCTION TO DIFFERENTIAL ANALYZERS PART I

Many problems arising in scientific and engineering applications are easily expressed as ordinary differential equations or simultaneous sets of such equations. When these equations are sufficiently complex that they cannot be readily solved by ordinary analytic means, some means of approximation must be used to analyze the system represented by the equation(s). One such means is the electronic differential analyzer.

Two types of differential analyzers are in use:

- (1) Analog differential analyzers;
- (2) Digital differential analyzers.

Mechanical analog computing components have been known for some time: from the time of Lord Kelvin in 1876. They were not developed to any useful degree, however, until a group at M.I.T. headed by Dr. Vannevar Bush developed the first differential analyzer in the early 1930's. During World War II, electronic and electro-mechanical analog computers were developed to a rather advanced degree.

Immediately after the war a group at Northrop Aircraft, Inc. realized the limitations of analog differential analyzers and developed the first digital type analyzers. Because the techniques are relatively new, few people, even in the computer

field, understand the techniques and this brochure is an attempt to explain them to a sufficient extent so that prospective users will realize their usefulness.

A digital differential analyzer is an electronic computer which solves differential equations by numerical integration. This does not, however, rule out the possibility of the use of this ty pe of computer for solving other types of equations, because many such equations may be expressed in the form of differential equations. Problems falling under this category include such calculations as: roots of transcendental equations; roots of simultaneous algebraic equations; integral equations; certain partial differential equations.

These numerical solutions are accomplished by means of a network of interconnected operational entities, called integrators,
each one of which may sequentially perform quadratures in
accordance with a fixed program. Each integrator may also perform miscellaneous operations not directly connected with integration, such as decision, adding and servo operations.

Although the integrators are operational entities, they are not actually physically separable, one from another, but are said to be time separable. The same electronic circuits perform the operations for all of the integrators in time sequence. In the Bendix D-12 it takes about 167 microseconds to perform the operations of one integrator, or .01 second to perform the operations of all 60 of its integrators. The computer is

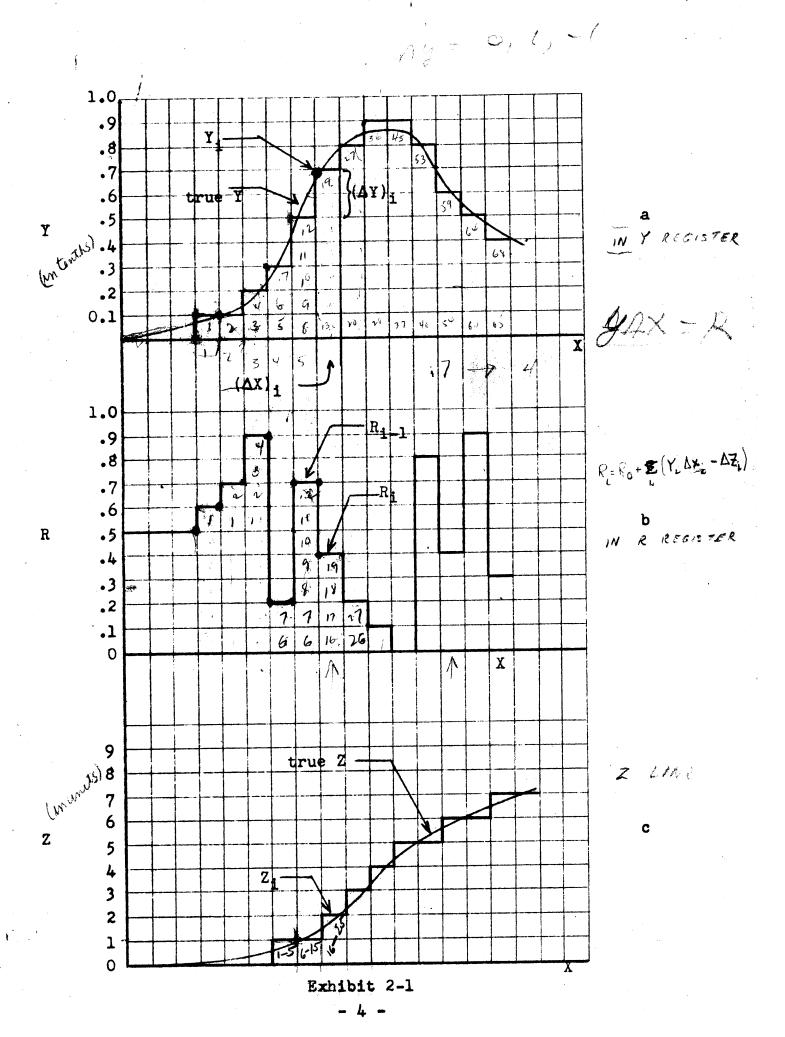
therefore said to have an iteration rate of 100 per second* on all of its integrators.

Since the integrators are not physically separable, the interconnections between integrators are not made physically; i.e.,
by means of wires or plug boards. These interconnections are
made by electronic coding means which involve time delay (or
memory) circuits. Moreover, the information concerning each
integrator is stored in a memory after processing while the
circuits process the other integrators, and is taken from
storage only when the circuits are ready to process that
particular integrator again. The time of storage of the information for one integrator (.01 second) is called one machine
cycle.

The type of memory storage used is the magnetic drum, and the information regarding the integrators is recorded sequentially about the circumference of the drum.

A digital differential analyzer, then, is a group of operational entities, called integrators, which are interconnected so as to perform iterated cycles resulting in numerical solutions of differential equations.

^{*}If less than 30 integrators are required in a given problem, the iteration rate may be increased to 200 per second.



PART II

METHOD OF INTEGRATION

Each of the operational entities in a digital differential analyzer is called an <u>integrator</u> because of the type of operation which it may perform. This operation is the performance of quadratures, in accordance with a fixed program which results in an approximation to integration. In the simplest case this program is one of rectangular integration, in which the area under a curve is approximated by summing the areas of a series of rectangles, as shown in exhibit 2-la.

Each integrator performs a repeated cycle of operations. In any cycle, say the i-th, the integrator receives a primary incremental input, $(\Delta X)_{\underline{i}}$, a secondary incremental input, $(\Delta Y)_{\underline{i}}$, and emits an incremental output, $(\Delta Z)_{\underline{i}}$. Each of these increments is in the form of an electronic signal, but mathematically each may be considered as a change in the arithmetic value of one of the variables, X, Y or Z respectively.

Associated with each integrator are two arithmetic registers. One of these, the \underline{Y} register, serves to accumulate the secondary incremental inputs $(\Delta Y)_1$ from cycle to cycle. On the i-th cycle, the value so accumulated may be given by the equation:

2-1
$$Y_i = Y_0 + \sum_{j=1}^{i} (\Delta Y)_j$$

where Y_0 is an initial value of Y. (See exhibit 2-la.) The range of Y_i may be considered to be in the interval $-1 \le Y_i \le +1$.

The (\(\Delta \times \) increment, although an electronic signal, may be considered to take on one of three mathematical values: a positive value; a negative value; or zero. When it is not zero, its absolute value is considered to be one on an appropriate scale. Depending upon which of the values it assumes on the i-th cycle, it is used to signal the integrator that the value in the second register, called the R register, should be respectively increased or decreased by the value held in the Y register, or remain unchanged. The resultant value held in the R register would be expressed by

$$R_{\mathbf{i}} = R_{\mathbf{o}} + \sum_{j=1}^{\mathbf{i}} Y_{j} (\Delta X)_{j},$$

except that the value so held must be in the interval $0 \le R_i < +1$. It is apparent that the R register will overflow; i.e., an attempted carry will occur into the position in the register which would have been occupied by the units digit. This occurs periodically as the register attempts to attain a value greater than one as Y_i is added to it. (See the top of exhibit 2-2.) Therefore, equation 2-2 must be modified.

Shown in exhibit 2-lb is a representation of the actual value held in the R register. Notice that the value held in the Y register, Y_i of exhibit 2-la, is successively added to the value in the R register at each increment, $(\Delta X)_j$, but when the R value tries to exceed +1, the register overflows and the resulting value being held is obtained by subtracting one from the value which would otherwise have been held.

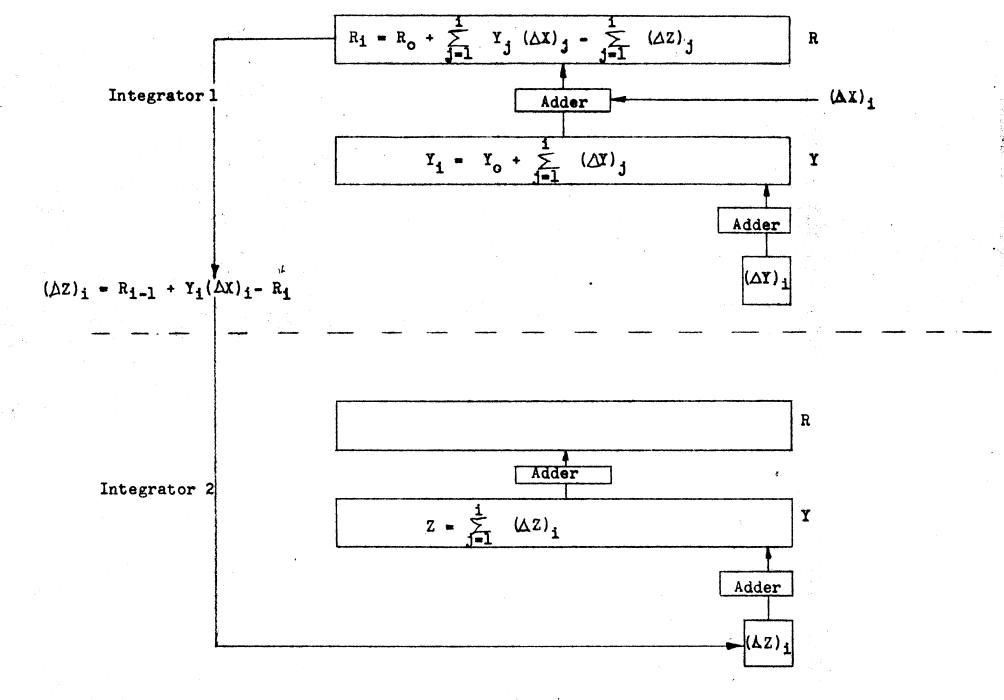


Exhibit 2-2

This overflow, or attempted carry, of the R register may be considered as the $(\Delta Z)_i$ output of the integrator. Since the R register will not overflow every time a $(\Delta X)_i$ occurs, the output of the integrator will sometimes be zero. Likewise, since either $(\Delta X)_i$ or Y_i may be negative, the overflow actually may become an "attempted borrow" which is considered as a negative overflow, and $(\Delta Z)_i$ is negative. Thus, $(\Delta Z)_i$ may take on three values similar to $(\Delta X)_i$. The secondary incremental input $(\Delta Y)_i$ also may take on a similar three values, but, as will be seen later, it may take on other values as well.

The incremental outputs of any of the various integrators in the computer may be used as the primary and/or secondary incremental input to any other integrator(s) including itself. Hence, in exhibit 2-2 the output of the first integrator is used as the secondary input to a second integrator. From equation 2-1 it can be seen that the Y register of this integrator will hold the value

$$z_{\mathbf{i}} = z_{\mathbf{o}} + \sum_{\mathbf{j}=1}^{\mathbf{i}} (\Delta z)_{\mathbf{j}},$$

which is represented by the step curve in exhibit 2-lc. The height of this curve is just the summation of the overflows of R in exhibit 2-lb.

The modification of equation 2-2 to take into account the overflows of the R register takes the form of a subtraction of the summation of all overflows.

$$R_{\mathbf{i}} = R_{\mathbf{0}} + \sum_{\mathbf{j}=1}^{\mathbf{i}} Y_{\mathbf{j}} (\Delta X)_{\mathbf{j}} - \sum_{\mathbf{j}=1}^{\mathbf{i}} (\Delta Z)_{\mathbf{j}}.$$

The value of $(\Delta Z)_i$ is given by

2-5
$$(\Delta Z)_{i} = R_{i-1} + Y_{i} (\Delta X)_{i} - R_{i},$$

where R_{i-1} is the value held in the R register before Y_i was added to it, or subtracted from it, or neither.

The incremental outputs of a number of integrators may be accumulated and simultaneously used as the secondary inputs to any integrator. It is apparent, then, that $(AY)_i$ may have any value $-n \le (AY)_i \le +n$, where n is the number of integrators whose outputs are used as incremental inputs.

Although the incremental values (ΔX)_i and (ΔZ)_i were said to take on the values plus one, minus one, or zero and (ΔY)_i was assumed to take on values between -n and +n, these concepts are only assumed when talking in terms of machine language. Mathematically they may be considered as very small changes, h, in the respective variables, X, Z and Y, where the h's in general are different for each variable. These incremental changes may be made very small, in which case they approach infinitesimals. Hence, as an expedient, the infinitesimal notation dX, dY and dZ is often used. Furthermore, if dZ is considered very small, the value of the R register, being a fraction of dZ, is even smaller and equation 2-5 may be approx-

imated by

$$2-6$$
 $dZ = YdX$.

This is the equations which will be assumed to express the mathematical operation of an individual integrator. Correspondingly, equation 2-3 in the new notation is

$$Z = \int_{X_0}^{X} Y dX.$$

A schematic representation for each integrator is given by a five sided block, which is used in a schematic information flow diagram or "road map" showing the interconnections between integrators. In this case exhibit 2-2 would be replaced by a schematic diagram similar to exhibit 2-3.

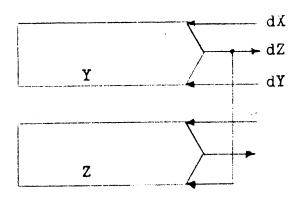


Exhibit 2-3

The X, Y and Z variables are normally renamed to correspond to variables defined by the problem.

In review, the "infinitesimals", dX, dY and dZ, are in reality finite small increments of the three variables. Each change

or increment of X and Z may take on the mathematical values +h, -h, or zero where the h's, in general, will be different for each variable, while the dY increments may take on values in the interval -nh≤ dY≤+nh and are accumulated in the Y register of each integrator to give the value Y. The operation performed by the integrator then may be expressed by equation 2-6, where the X, Y and Z variables are renamed to correspond to variables defined by the problem.

PART III

OTHER OPERATIONS OF INTEGRATORS

As stated in Part I, integration is not the only type of operation than can be performed by integrators. These other operations have nothing to do with integration as such, but the name "integrator" will be retained in the interests of uniformity.

In all of these operations the integrator receives the same two inputs, dY and dX, and has an output, dZ. The method of operation of the integrators when performing these other operations is different from that explained in Part II and is determined by the type of desired result. None of these special operations makes any use of the R register; the dZ output is not the overflow of this register.

There are basically three types of special operations which an integrator may perform: servo; decision; adder operation.

3.1 Servos and Decision Integrators

The operation of an integrator to perform either servo or decision operation is essentially the same, differing only in a mathematical sense as determined mainly by the initial conditions set into the Y register of the integrator at the start of a problem. The value of the dZ incremental output on any given cycle (i) is determined by the value of the dX increment and the value held in the Y register in that cycle, according to the following table:

(a) If $2 > |Y_i| \ge 1$, $(\Delta Z)_i = 0$

(b) If $0 > Y_{i} > -1$, $(\triangle Z)_{i} = -(\triangle X)_{i}$

(c) If $0 < Y_i < 1$, $(AZ)_i = (AX)_i \rightarrow$

(d) If $Y_{i} = 0$, $(\Delta Z)_{i} = 0$

Exhibit 3-1

The main difference between the operation of the integrator when operating as a servo and when operating as a decision integrator is the relative value about which the Y register value is operating. As the name implies, a servo should always have an output which tends to reduce an error value to zero. This error value is actually the value which the integrator holds in its Y register. This indicates that the Y register value will operate around zero, and in reality the Y register value is usually very small. The conditions (b), (c) and (d) are the only ones used by a servo under these circumstances, and the dX input is always positive every iteration.

A decision integrator, on the other hand, does not necessarily have its Y value near zero. It can use the fact that the output increments essentially reverse sign as the integrand passes through zero (conditions b and c), or it can use the condition that the output increments become zero when the integrand exceeds one in absolute value.

The sample problem of Part VI contains a decision integrator for generating the absolute value of a problem variable.

3.2 Adders

An adder generates an incremental output which is the sum of

very similar to a servo in that the output increment is determined in the same manner as in a servo (conditions given in exhibit 3-1). The difference between the two is that an adder has the negative of its own output fed into itself as a dY input. This is done automatically (including the multiplication of the output increment by -1) when the given integrator is coded as an adder. The dX input to an adder is always positive.

The resulting operation of an adder is as follows: if the value in the Y register is slightly positive on a given iteration, the output increment (from exhibit 3-1) will be positive. On the next iteration this positive increment will be multiplied by -1 and used as a dY incremental input. This resulting negative increment will be added algebraically to the Y register value, making this value less positive. This will continue until the value in the Y register is reduced to zero, at which time the output increments will become zero. Likewise, if the value in the Y register is negative, the output increments will be negative and the Y register value will be increased until it is zero.

The term "adder" is readily apparent when it is considered that other dY incremental inputs are also present on each iteration (the series of increments of the two variables which are being added together). These increments will tend to make the Y register value other than zero--positive when the

increments are positive and negative when they are negative.

Hence, when one of the incoming increments is positive, the adder will have one, and only one, positive output increment; and when there is a negative input increment, a negative output increment will result.

It can be seen, then, that whenever either of the two (or more) dy inputs have non-zero increments, the output of the adder is a non-zero increment and the number of incremental outputs of the adder is the sum of the incremental inputs from all dy sources.

PART IV

SPECIAL FEATURES OF THE BENDIX D-12

In the foregoing parts the general operation of the computer has been explained in sufficient detail to show how the computer is able to solve problems: an example problem shown in Fart VI will serve further to clarify the method of working problems. However, no mention has been made of inputoutput and other finer points about the internal operation of the computer. In this part the method of communicating with the machine will be explained and certain internal features pointed out.

4.1 Numbering System

It might have been assumed by the reader when looking at exhibit 2-1 that the decimal numbering system is used in the computer. Since many computers use the binary system, this assumption might reasonably have been false. However, the reader is hereby reassured that if he were to use the computer he would find that the decimal numbering system is used exclusively. The internal operations of the computer are accomplished using excess-three coded decimal digits. Furthermore, although it is definitely helpful for the operator to know the excess-three code when using the computer, it is not necessary.

4.2 Output Multipliers

Although output multipliers are used to increase the efficiency of decimal scaling, and scaling is presented in a later section, the reasons for output multipliers can be given without explaining scaling.

To do this, assume that the known maximum value which will ever be attained in the Y register is 0.2; i.e., a 2 in the left-most position of the Y register next to the decimal point. Then as the dX incremental inputs to the integrator cause this value to be added to the value in the R register there results a situation similar to that shown in exhibit 4-1 with a non-zero output every fifth dX increment.

Exhibit 4-1

The outputs occur whenever the value in the R register tries to exceed one.

Now assume that the R register value cannot exceed .5, so that when its value tries to exceed this amount an output occurs and the resulting value will be the sum modulo .5. This is illustrated in exhibit 4-2.

Exhibit 4-2

Notice that in every five iterations (or non-zero dX incremental inputs) there are two non-zero dZ outputs, whereas in exhibit 4-1 there was only one. In this case the output multiplier is said to be 2, because the frequency of non-zero outputs is doubled over that of the normal operation where the output multiplier is said to be one.

There is one other possible output multiplier: five. In this case the value in the R register cannot exceed .2, as shown in exhibit 4-3. Hence, if Y = .2 is successively added to the value in the R register, the R register value does not change and there is an output every iteration.

Exhibit 4-3

Since there is a non-zero output every iteration, the frequency of output is five times the case where the output multiplier is one.

The net result of the output multipliers of two and five, then, is to increase the frequency of non-zero output over that when an output multiplier of one is used. Because of this there results a definite advantage in scaling (this will be more apparent later) and a greater amount of precision may be obtained in a given problem with the same running time.

4.3 Ternary Transfer

In order to point out one feature of the Bendix D-12, it is necessary to explain a system of incremental transfer used in other differential analyzers. As was stated, the dX and dZ increments may take on three values including zero. This is called ternary transfer of information. Other types of differential analyzers use what is called a binary transfer system in which the increments may not assume the value zero. This is done by using a method of adding Y to R such that the R register is induced to overflow (either positively or negatively) every cycle. Since there is no zero, if a variable is not changing or is changing slowly, the system of transfer must adopt a series of plus-one-minus-one increments to show a zero rate. (This "plus-one-minus-one" is sometimes referred to as "one-zero".)

It is apparent in such a case that the last digit of each Y register is not completely significant. Its oddness or

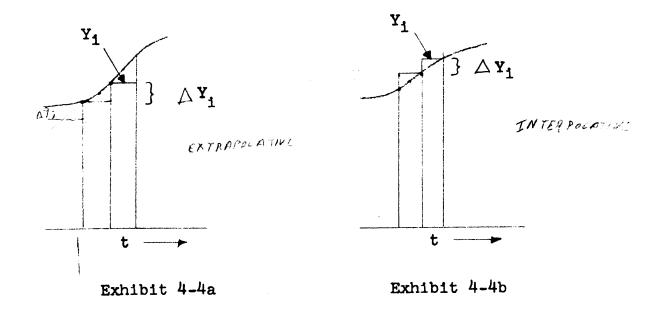
eveness is completely predetermined by whether the number of the cycle is odd or even, and by the oddness or evenness of the number of dY inputs coded for the particular integrator.

It has been found that such a system induces certain errors other than round-off or truncation errors; hence, the ternary transfer system has been adopted in the Bendix computer, and no more will be said about binary transfer.

4.4 Truncation Correction

In any numerical method for solving differential equations—especially non-linear equations—many integrations are being performed, each of which depends upon the outcome of other integrations. Physically, then, it is impossible to have exact up-to-date data for each integration. In any cycle it will be possible to receive the results of certain integrations to be used in others, but not the other way around. The starting point of the process must use old data.

The two cases can be represented by two curves shown in exhibit 4-4. The first is the case for the starting integrators which use old data, and the latter the case for other integrators using the results of the first integrators.



These curves are in contrast to the condition shown in exhibit 2-la in which the true Y was assumed to be straddled by the Y₁ values. Here it is seen that the starting integrator uses the height of the curve at the front of the rectangle as the value of Y₁, when, for the best approximation, it should use the average of this value and that at the back of the rectangle. The other case shows that the Y₁ value being used is the height of the curve at the back of the rectangle, where the average of this height and that at the beginning of the strip should actually be used.

Now, since the integrator in the first case does not know the neight of the curve at the back of the rectangle, it must make a guess at the height of the curve in the middle. The best guess is to use the height of the curve at the beginning of the rectangle and add to it one-half of the $(\Delta Y)_i$ on that cycle. This is called extrapolation and an integrator that does this is called an extrapolative integrator.

Since the integrator in the second case may obtain the height of the curve at the beginning of the rectangle by subtracting $(\Delta Y)_i$ from the height at the end, it can just as easily obtain the average height of the curve in the rectangle by subtracting one-half of the amount. An integrator which uses this scheme is called an <u>interpolative</u> integrator.

The rectangular areas shown in exhibit 4-4 are now modified to correspond to those shown in exhibit 2-la, in that the true Y value is actually straddled by the tops of the rectangles. Another interpretation of this process is that the rectangular areas are actually replaced by trapezoidal areas as shown in exhibit 4-5, where the top of the trapezoid is a straight line chord of the curve. Therefore, the term trapezoidal integration is used to distinguish this process.

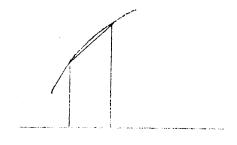


Exhibit 4-5

dz-Ydx

Equation 2-6 may be considered part of the Taylor series where only the first order term is used and all others are truncated. Because finite increments are used, it would be better if the second term could be used. It can be shown by rigorous mathematics that trapezoidal integration results in this. For that reason, it is sometimes called second order integration. Also, since the correction prevents the truncation of the second order term it is sometimes called truncation correction.

In order to prevent the necessity of having to refill the computer to start a problem over from the beginning, a separate register is provided in the arithmetic section of each integrator to hold the initial conditions of the integrand. At any time, under control of the operator, these may be automatically copied into the Y registers and computations restarted.

An incremental input may be coded to the initial condition register (Y₁ register) of any integrator in a similar manner to coding inputs to the Y registers. The computer then may be coded to automatically reset the new initial conditions and restart computations using these new conditions.

4.6 Input-Output

There are a number of means for inserting information into and taking information out of the computer, and the choice of which ones are used is dependent upon the desired result. Of course, there are two basic types of input and output: first, the information put in and taken out of the computer when it is not computing; second, information input and output during computations.

4.6.1 Typewriter Input and Output

An electric typewriter is the principal means by which the operator communicates with the computer. In preparing a problem for solution on the computer, a coding form is prepared which gives a set of data to be typed on the typewriter. As this data is typed, a punched paper tape is prepared containing this information. A paper tape reader then extracts the information from the tape and enters it into the computer, the computer interpreting the information as a problem which it is to work.

As the problem proceeds in the solution, the computer will have been programmed to type out answers on the typewriter.

These answers will consist of the values contained in the Y registers of designated integrators; hence, are the values of some of the variables of the problem.

Although only the selected variables of the problem will be typed out automatically by the computer, any Y register value may be typed out on the typewriter at any point in the problem by means of manual control. This allows the operator to monitor the problem when in the process of trouble shooting the program when the problem is first started.

4.6.2 Manual Fill and Monitoring

The method of filling and of monitoring described above is sufficient for filling the computer, monitoring, and trouble shooting and correcting the program; but frequently it is easier and quicker to make minor changes in the program and to view the operations of the computer by a readier means.

A small display oscilloscope is provided as an integral part of the operator's console to examine the information channels of the computer at any time during the operation of the computer, and a small manual keyboard is supplied with which the operator may enter information into the computer without using the typewriter.

4.6.3 Memory Read-in and Read-out

If the running time of a problem is very long, and it is desired to run it in two or more parts, perhaps shutting the computer down between parts, the entire information of the

memory may be taken out of the computer and stored on punched tape. (This is distinguished from the automatic programming tape described above.) When the problem is ready to be resumed, the information on this tape may be copied back into the memory automatically and computations continued.

4.6.4 Plotter Output

With the exception of the automatic type-out as described in 4.5.1, all of the above sections are concerned with getting information into the machine in preparation to working a problem, whereas input and output during computations are concerned now.

A second method of output from the computer is the automatic graph plotter. As any designated variable changes during the running of the problem, its incremental changes can be fed out of the computer to one of the axes of an incremental type graph plotter. Increments of a second variable are fed to the other axis and a plot of one variable versus another is obtained.

4.6.5 Function Input

There are two methods for inserting into the computations information of an empirical nature such as experimental data.

If such information is in the form of a curve, first or second

of the independent variable may be punched onto paper tape and this entered into computation in an incremental manner. This method is called <u>incremental input</u>. The same method may be used to enter information in the form of tables but the second method is of such a nature as to provide means for entering the tabular values into the integrands directly. This second method is called <u>non-incremental input</u>.

PART V

PROGRAMMING

Although it is apparent that the subject of programming is much too large to cover in a few short paragraphs, and, indeed, many of the facets must come by experience, those readers most interested may be able to obtain some background through a brief discussion. With this and the sample problem in Part VI, the reader will be able to do some sample programming and get a better feel for the subject.

There are three basic steps in preparing a problem for solution on the computer:

- (1) Mapping;
- (2) Scaling;
- (3) Coding.

5.1 Mapping

In Part II reference was made to the schematic block diagram which represents the information flow between integrators (see exhibit 2-3). The process of mapping is the drawing of this information flow diagram for all of the integrators used in the problem.

This information flow diagram will be determined by the particular problem being solved, and the relations which express the operation of the various integrators (equation 2-6 and exhibit 3-1).

5.2 Scaling

In general, scaling consists in assigning mathematical values to each increment of all of the series of increments being transferred from integrator to integrator. For any series of increments the scale value is expressed as the reciprocal of the value of any one increment. This scale value is also the number of increments which, if accumulated, would give one unit of the variable concerned.

As a result, associated with the dX and the dY inputs to each integrator is a scale factor; and in the case of the dY increments, since they are actually being accumulated, the scale factor gives the number of increments necessary to produce a change of one unit in the Y value. The scaling for each individual integrator consists in calculating the scale factor for the dZ output increments.

One other scale factor must be defined-that of the value held in the Y register of an integrator. Normally the value so held will be considered by the machine to be less than one in absolute value; i.e., the decimal point will be before the most significant digit. If such is the case, since the problem value need not necessarily be less than one, the true problem value must be multiplied by a suitable quantity to make it less than one. Also, since the problem value may be quite small, it may be desirable to multiply it by a value greater than one. In either case, the quantity by which the problem value is multiplied is called the scale value of the Y register.

All of these scale factors, viz., the dX, dY, Y and dZ scale factors, are practically always expressed as an integer (usually 1, 2 or 5, but not always) multiplied by an integral power of ten, where the integral part usually arises through the use of the output multipliers.

The relationships between the four basic scale factors and the output multiplier associated with each integrator may be expressed by established formulas. Since the dX and dY scale factors will be given for a particular integrator, excepting the case where they might subsequently be changed, and the Y scale factor and the output multiplier will, to a large extent, be determined by the maximum range of the Y value, the dZ scale factor is the only unknown for the particular integrator. Therefore, the following formula will express this scale factor as a function of the known quantities:

$$S_{dZ} = S_{dX} MS_{Y} \times \times \times$$

where: S_{dZ} is the scale factor of the dZ output,

 S_{dX} is the scale factor of the dX input,

M is the output multiplier,

Sy is the scale factor of the Y value.

Furthermore, if

 I_{dY} is the integral part of the dY scale factor,

Iv is the integral part of the Y scale factor,

N <u>is the number of significant decimal digits</u> of the fractional part of the Y register which will be used,

is the exponent of ten in the dY scale factor, and E_Y is the exponent of ten in the Y scale factor, then $I_Y = I_{dY}, \quad \text{and} \quad \text{and} \quad \text{for} \quad P$ $N = E_{dY} - E_Y$

5.3 Coding

In section 4.6.1 it was mentioned that filling of the machine was accomplished automatically when information from a punched paper tape is copied into the computer. This information was put on the tape when the coding form was typed on the typewriter keyboard. Coding consists in preparing this form for typing, and will probably be done simultaneously with the typing, although the information for the coding will be written on the programming sheet.

No attempt will be made here to explain all of the meaning of this form, but a sample form is shown in exhibit 5-2.

	YREGISTER	OPCERTION	04 174	5 77		
PARTICULA JANGERORI	· *	INST. TO INTE	RG. 2N/	2017	dy	* y
11	00000	31212	90 dx	16	17	18
12	0000000	21212	90	11		
13	00000	31212	90	19	20	21
14	0005000	21212	90	13		
15	0005000	21112	14	13		
16	0217995	111-1	12			
17	00161435	11111	15			
18	03162831	11111	14			
19	06434593	111-1	14			
20	01375565	111-1	15			
21	-0480014	11111	12			

etc.

Exhibit 5-2

The first column is the number of the particular integrator for which the information on the same line is intended. The next group of numbers gives the initial condition to be filled into the Y register, while the third group is the operational code group which gives operation instructions to the integrator. The following successive columns give the integrator numbers whose outputs are used as inputs to the particular integrator. The first column is the dX input, the second is the variable initial condition input, and the last few (maximum of eight) are the dY inputs. In the form shown no inputs are coded for the variable initial condition; hence, the column reserved for coding such inputs is blank.

A better understanding of programming will be obtained when Part VI is read.

PART VI

SAMPLE PRUBLEM

In order to show more clearly the method which would be used on the computer in preparing a problem for solution, a sample problem is programmed completely, the program is explained in some detail, and the typed results given.

A typical well-behaved ordinary differential equation which gives the mathematical relation for a system of damped oscillations with non-linear damping is given by:

6-1
$$u'' + 0.5 u' \cdot |u'| + 0.4 u' + u = 0$$

where u' and u'' are the first and second derivatives respectively of u in respect to an independent variable, x_*

If equation 6-1 is transformed so that u" is expressed in terms of the rest of the equation, and the differential of the equation is taken, there results:

6-2
$$du'' = -0.5 d(u' \cdot | u' |) -0.4 du' - du$$

The "road-map" for the interconnections of the integrators is shown in figure 6-1. The scaling is also shown.

If it is assumed that in integrator 11 the Y register holds a value corresponding to u", and this is integrated with respect to dx, then du' = u"dx is the result, and the incremental outputs are the increments of u'. These are then used as the dX and dY inputs to integrator 12, as the dX input to 13, and the dY inputs to integrators 14 and 15. As the

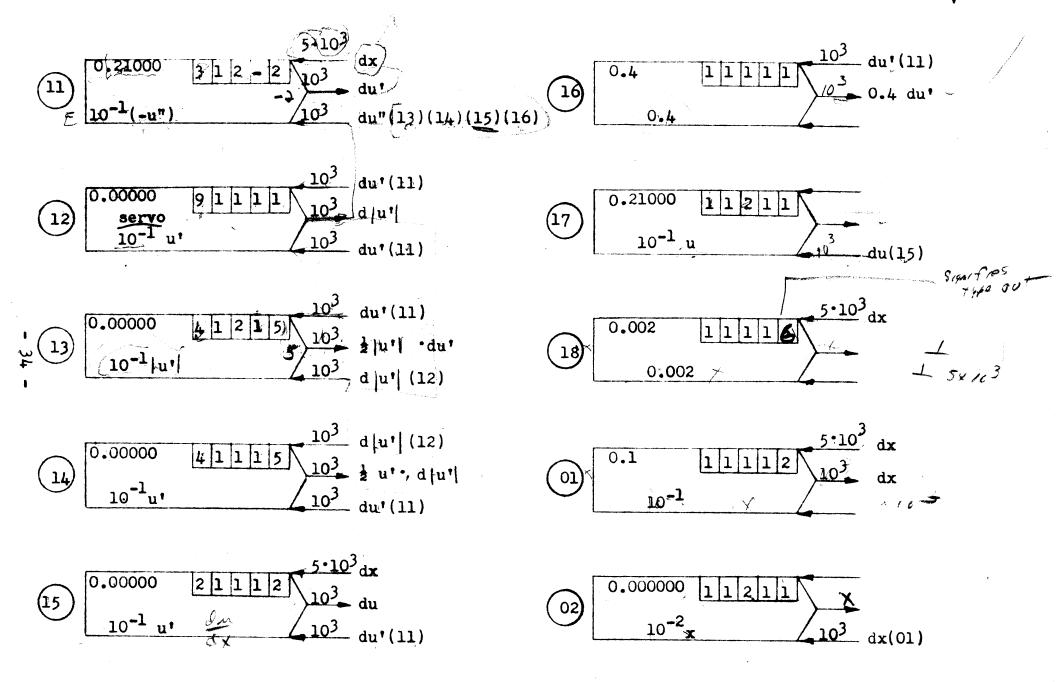


Exhibit 6-1

result, the Y registers of 12, 14 and 15 will contain u'.

Now, integrator 12 is a decision integrator, and one use of a decision integrator is hereby presented. Since the absolute value of u' is required, the differentials of this variable are produced by integrator 12. If the Y register of 12 is positive (u' positive), then the incremental changes of u' will be passed through unchanged and will become the dZ output increments of 12. Since, when u' is positive, the differentials of u' are identical to the differentials of |u|, the output of 12 is what is desired. Likewise, when u' is negative the negative of the differentials of u' occur as outputs of 12. Since, when u' is negative, the derivative of u' is the negative of the derivative of |u|, the outputs of 12 are correct in this case also. Thus, the outputs of 12 are the differentials of the absolute value of u'.

Integrators 13 and 14 work in conjunction with one another. Since |u'| and u' must be multiplied together, the differential of this product must be generated by 13 and 14. The differential of the product of two variables, u and v, is given by the formula $d(uv) = u \ dv + v \ du$. The two terms on the right of this formula require one integrator each, and in the case in question the two integrators are 13 and 14.

In integrator 15 the integration of u' with respect to dx is accomplished to produce du which is accumulated in integrator 17 in order to be typed out.

Integrator 16 is used to multiply du' by 0.4 to produce the next to the last term of equation 6-2. The output of this integrator plus the outputs of integrators 13, 14 and 15, which produce the other two terms of equation 6-2, are used in dY inputs to integrator 11. Together all of these inputs form -du" from equation 6-2.

Without taking into account the scaling of this problem, it appears on the surface that the multiplication of d(u'.|u'|) by 0.5 has not been done; but, as will be seen later, this is automatically accomplished in scaling. Along the same line, the signs might be mentioned. Since all of the signs on the right of equation 6-2 are negative and the integrators produce the positive of these terms, the input increments to ll will not be du" but -du" and -u" will be held in the register. Unless the output of ll was reversed -du' would result; hence, this output has a sign reversal, as will be seen later.

Before the other integrators are explained, consider the scaling of the integrators mentioned thus far. The scale value of the independent variable (machine time, an artificial variable having a positive increment each iteration) is chosen fairly arbitrarily as 5·10³, as seen by the number near the dX input of integrator 11. It is known (or assumed) that u'' will not exceed five in absolute value at any point during the running of the problem. This results in a Y scale factor in 11 to be 10⁻¹ so that when u'' is five, the Y register of this integrator will contain 0.5. From the earlier discussion about

output multipliers, it can be seen that a multiplier of two can be used if the Y value will not exceed 0.5. Using such a multiplier, the dX scale factor, Y register scale factor, and the output multiplier are multiplied together to produce 10³ as the dZ output scale factor. The other scaling relationships are also satisfied by this scaling; namely, the integral part of the Y scale factor equals the integral part of the dY scale factor, and the number of decimal digits in the Y register will be 3 - (-1), the difference between the exponential parts of the dY and the Y scale factors.

Identical scaling is evidenced in integrator 15.**

The output scale factor of decision integrator is almost always identical to the scale factor of the dX input; hence, in integrator 12 the scale of dZ is the same as the scale of dX, or 10³.

The scaling of 12 and 13 is not so readily apparent. If the normal procedure is followed in these integrators, the dZ scale factor should be 5.10^2 , and indeed it is for what would normally be the expected output. But as will be seen, a one-half is multiplied by the expected differential output, so the multiplication must have been done through scaling. If there are 500 increments per unit of u'. | u' | , as repre-

sented by the expected scale factor of 13 and 14, then there are 1000 or 10^3 increments of 1/2 u¹. | u¹|. This results in the 10^3 scale factor on the outputs of 13 and 14.

The five characters in the upper right hand corner of each integrator block are the five coding digits described under section 5-3. Two of them are rather apparent and have been discussed for each integrator: the right one being the output multiplier of the integrator, and the second from the right being the sign reversal code. If this character is negative there is a sign reversal on the output (integrator 11), otherwise it is a one.

In the third of the five positions is put a 2 in any integrator which is to be typed out during computations, and a one other-wise. Hence, in this problem the integrands of 11, 13, 17, and 02 will be typed out.

The second digit position is concerned with selecting integrators whose Y values are to be reset periodically during computations. Since this sample problem does not use variable initial condition, all of the characters in this position are one's.

The first of the digits is the mode or type of operation performed by the integrator. The character put in this

position is determined by the following chart.

- 1 used when the mode is relatively immaterial (integrators 16 and 17)
- 2 interpolative integrator

**

- 3 extrapolative integrator
- 4 integrator doing multiplication as 13 and 14
- 5 adder
- 9 servo decision integrator

The method used by the operator to select the extrapolative or interpolative mode is the following. If the dY inputs to an integrator come from higher numbered integrators, the integrator is an extrapolator; if they come from integrators preceding it (lower numbers), the integrator is an interpolator.

The purpose of integrators 18, 01 and 02 may now be discussed. It is desirable in this problem to type out the variables already mentioned at equal intervals of the independent variable, x. Integrator 17 will determine the interval of typeout. If an output multiplier of 5 is put in any integrator (the last character of the operational code of integrator 18), whenever that integrator has a non-zero output all of the designated variables will be typed out. In integrator 18, as 0.002 is added to the R register each iteration, it will take 500 iterations for it to have a non-zero output. Since 5000 increments of dx are equivalent to a change of

one unit in x, the typeout will occur at intervals of 0.1 of x. Incidentally, if the typeout was to occur at intervals of any other variable, the dX of integrator 18 could be made the increments of the desired variable.

It is desirable to accumulate the independent variable, x, in an integrator in order that it may be typed out also. If, however, dx is used as the dY input to an integrator directly, since its scale factor is $5 \cdot 10^3$, the value held in the Y register will not be x but five times x. Hence, integrator Ol is used to change the scale factor of dx, and its output is used as the dY input to O2.

The explanation of the program is complete except, possibly, the selection of integrator numbers. In general, the only criterion was to select the numbers in a logical order, proceeding from higher derivatives to lower ones. However, since printing will occur in numerical order, to get the desired format on the printed output page, the numbers of Ol and O2 were chosen so that x would occur in the first column of the printed page.

In the upper left-hand corner of each block is the initial conditions of each integrand. The first digit of this number is the units digit and is zero in all of the integrators of this problem. Next come the significant digits the number of which is determined by the exponential part of the dY and Y scale factors. The last digit is a round-off digit. This digit is never changed during the problem, but must be put in initially.

In all of the integrators in this problem this digit is zero. However, since there is no input to a constant multiplier this digit is the same as any other digit, hence no special difference is made in 16, 18, etc.

The coding for this problem is shown in exhibit 6-2.

٠,٥٠	021000	312-2	90 1	13 14 15	16//51
TT				17 14 17	10,0,
12:	م در 000000 م	91111	11	T1	
13 ¹	(A) 000000(45)	41215	15)11 55	12/3	
14	000000	41115	12	11	
15	000000	21112	90	11	
15 16	04	11111	11		
17	021000	11211		15	
18	0002	11116	90		
01	01	11112	90		
02	000000	11211	4	01	
_					

Exhibit 6-2

Since the coding sheet is practically a recopy of the numbers written on the "road-map", little explanation is necessary.

The first column, of course gives the integrator number. A space is always left after this number, and a second space is left if the initial condition of the integrand is positive, which it is in all integrators of this problem; if any of them were negative, a minus sign would have been put in this position. The initial condition of the integrand is next copied from the programming sheet, giving the units digit, the significant digits, and the round-off digit. Since there are a total of nine digits that could be filled into the register, one space is left for each unused digit position. Thus in integrator 11 three more spaces are left. One extra space is left to separate the

integrand from the operational code. The five digit operational code is then typed verbatum from the programming sheet, and one more space left. The dX source is then typed, including the digits 90 if the machine independent variable is the source of the dX input. One more space is left and the source of the variable initial condition input is given (there are none for this problem). After that the sources of the dY inputs are given, a space separating each additional input.

The last character typed is a period (below the last integrator number) which signifies the end of the coding. The tape, when being read by the computer will automatically stop at this point.

x•10 ³	-u"•10 ³	u'·103	u·10 ³
00000 00100 00200 00300 00400 00500 00600	02100 01987 01824 01623 01400 01170 00943	00000 -00205 -00396 -00568 -00719 -00848 -00953	02100 02090 02060 02011 01947 01868 01778
00700 00800 00900 01000 01100 01200 01300	00726 00527 00349 00191 00055 -00063 -00162	-01037 -01099 -01143 -01170 -01182 -01181	01678 01571 01459 01343 01225 01107 00990
02000	-00519	- 00 9 03	00250

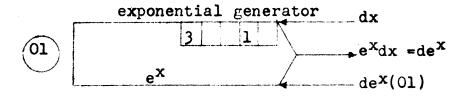
Exhibit 6-3

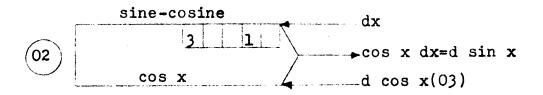
In exhibit 6-3 is shown the table of values which was actually typed out by the computer. Notice that the variables were typed

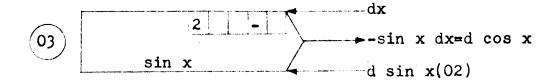
out by numerical order by integrator. The method of labeling the columns is purely arbitrary, but in this case the decimal point of all of the numbers was considered at the right, i.e., all of the numbers are whole numbers. Therefore, to obtain the actual problem values, one must divide the values by the dY scale factor. The columns are then labeled as the pertinent variable times the dY scale factor.

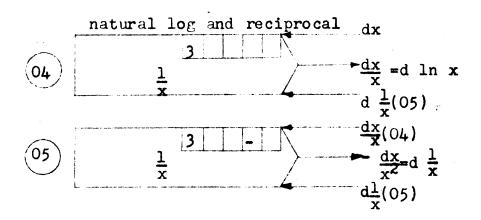
PART VII FUNCTION GENERATORS

In this part is presented some sample integrator hook-ups which will result in the generation of certain often used functions. No explanation is included but the variables are expressed quite explicitly and no difficulty should be encountered in understanding the hook-ups. The mode and sign digits of the operational code are presented, also.









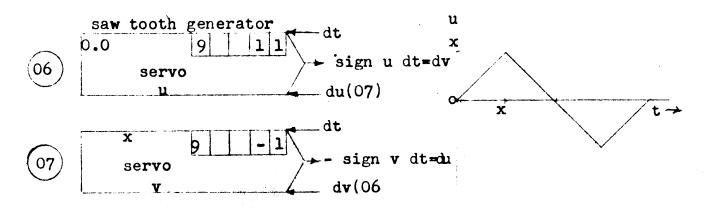


Exhibit 7-1 - 44 -