INTRODUCTION TO THE DIGITAL DIFFERENTIAL ANALYZER

I. General Description

A DDA is an electronic, digital model of the Vannevar Bush mechanical differential analyzer, circa 1932. A digital differential analyzer is a special purpose digital computer with a semi-fixed program. The program is stored internally and is fixed for a particular problem or computer run. That is, the program cannot modify itself, nor are there any transfer or jump operations. In fact, the DDA has no command structure as such: the program comprises numerical coding which directs the sequencing of pulses from memory to adder section. A magnetic drum, two serial adders, and an elaborate control section make up the computer, which is capable of solving linear and nonlinear differential equations. The DDA has a fixed number of registers which each contain the up-to-date count of one or more sequences of positive or negative pulses. Each register is up-dated once every iteration cycle -- usually one drum revolution time.

The programmer represents the DDA by a fixed number of operational elements called "integrators." Although these elements differ from the analog computer integrator, programming proceeds in a similar fashion to the preparing of an analog computer diagram. The name, integrator, is a carry over from the mechanical differential analyzer. A DDA integrator performs these two operations:

$$y_i = y_{i-1} + \Delta z_i \left(= y_i \, \Delta x_i \right)$$

which approximate the operations:

$$y = \int dy \quad \text{and} \quad dz = y \cdot dx$$

A schematic representation of the DDA integrator as well as the mechanical differential analyzer is shown in Figure 1.

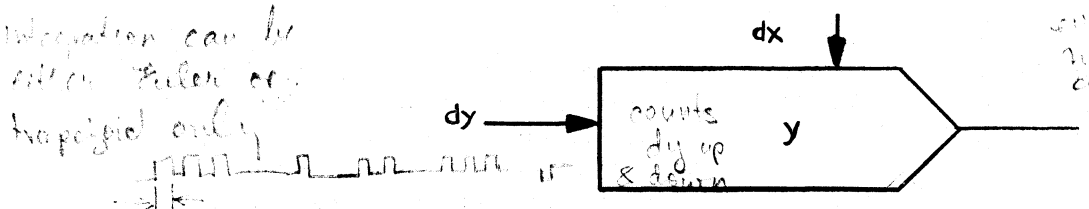


Figure 1

In the mechanical differential analyzer the dx, dy and dz quantities represent differential angular displacements of shafts. By convention the differential notation dx, dy, dz is retained to represent trains of pulses, where the quantity is measured not by the size of the pulses but by the pulse rate (relative to computer iteration rate). Thus "dy" stands for a pulse rate which is a function of time, changing in discrete steps. "dy" does not represent

a differential quantity in the strict sense, nor does it precisely represent a discrete quantity, Δy.

To program a DDA to solve a differential equation, one first solves for highest derivatives and forms this derivative schematically by summing lower order terms which are generated by a series of integrators.

For example, consider the differential equation

$$\ddot{x} = -a\dot{x} - bx$$

$$d\dot{x} = -ad\dot{x} - bdx$$

which is programmed as shown in Figure 2.

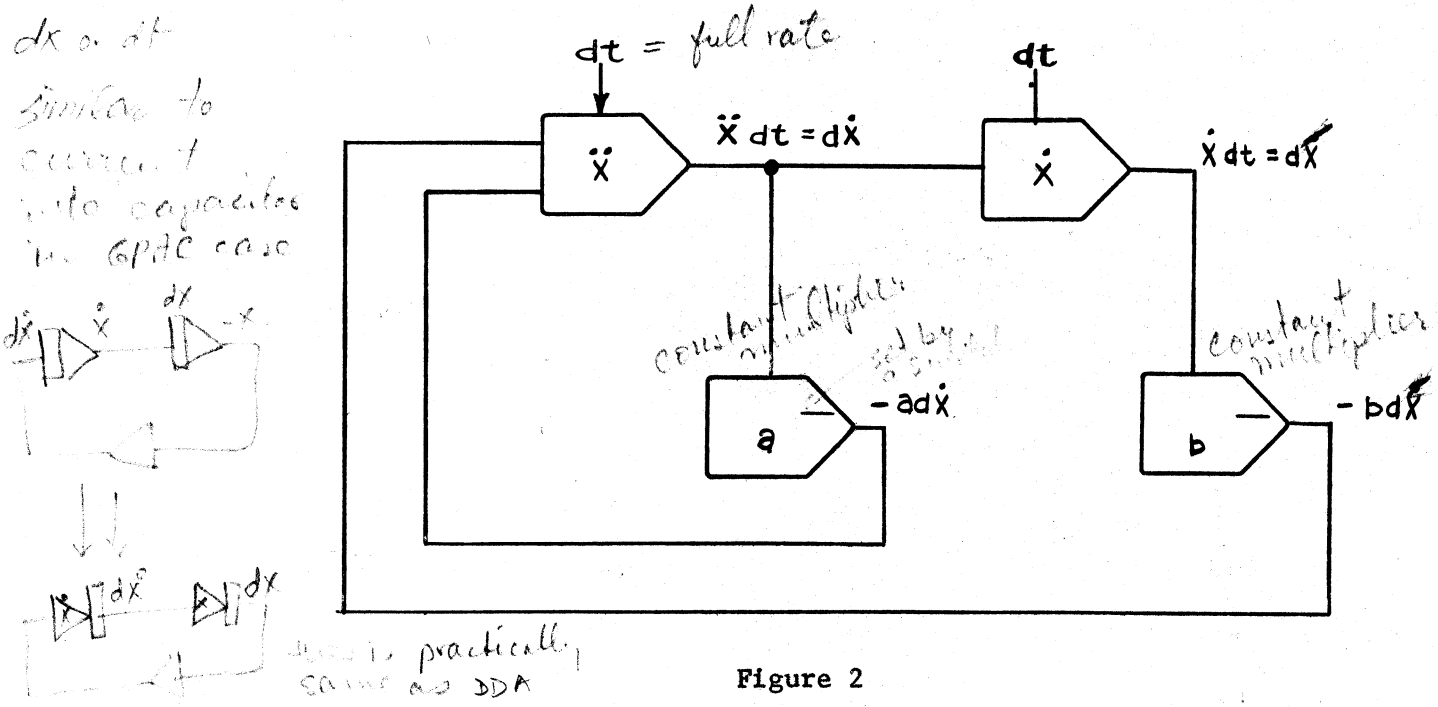


Figure 2

Thus the DDA operational elements are interconnected, just as the operational elements of an analog computer, to form the solution of the differential equation. The major difference is that the DDA elements are fictitious entities which exist only as space on a magnetic drum and are interconnected not by wire but by numerical coding. This numberical coding can be extremely simple, requiring no specific knowledge of the computer or any computer language.

The Bush mechanical differential analyzer utilizes a "wheel and disc" integrator which clearly illustrates the differential analyzer principle. Figure 3.

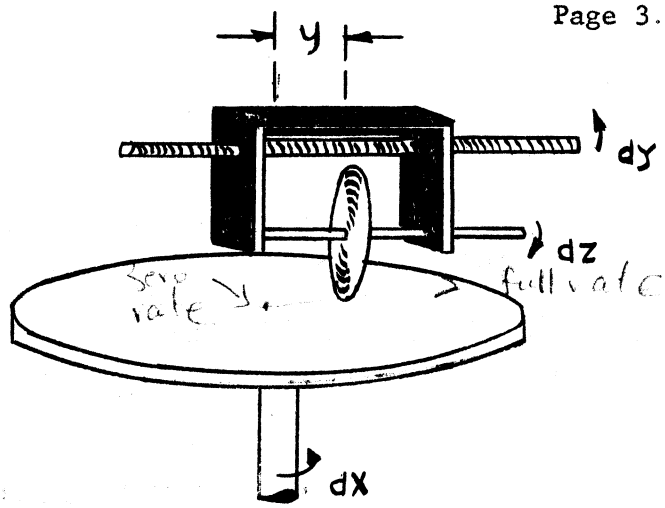


Figure 3

If one were to equip each shaft with a ratchet he would then have a mechanical DDA. Note that a shaft carries no information other than indicating a change by its rotation -- thus even though the shafts interconnect all elements of the differential analyzer, it requires a unit other than a shaft to indicate the value of any parameter such as a revolution counter, or wheel and disc integrator.

The DDA and mechanical differential analyzer operational elements, called integrators, are characterized by the holding of the current value of all parameters in a storage device within the element, and the transmission of only differential changes. The current value of the parameters may be continually monitored by recording devices, but at no time are they actually operated on as in an analog computer (and a digital computer).

Programming the DDA closely parallels analog computer programming, including scaling of variables, except that the range of variables is between $\pm$ 1. Time scaling is different in that scaling is accomplished by assigning a weight to each pulse of the maximummachine pulse rate and adjusting the magnitude scaling of each integrator accordingly. This will be illustrated later.

DDA techniques make the programming of some functions more complicated and others less so. For example, multiplication requires two integrators to form the product d(uv) = vdv + udv. On the other hand only three integrators are needed to generate $d(x^{1/2})$, $d(x^{-1/2})$, and d(lnx) simultaneously. Figure 4.

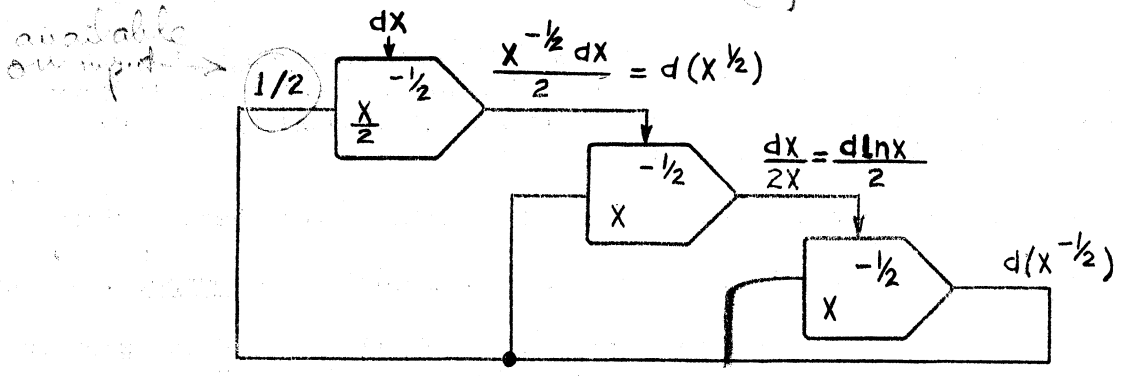


Figure 4

Furthermore: let y = tanx

$$dy = \sec^2 x dx = (1 + \tan^2 x)\, dx = (1 + y^2)\, dx$$

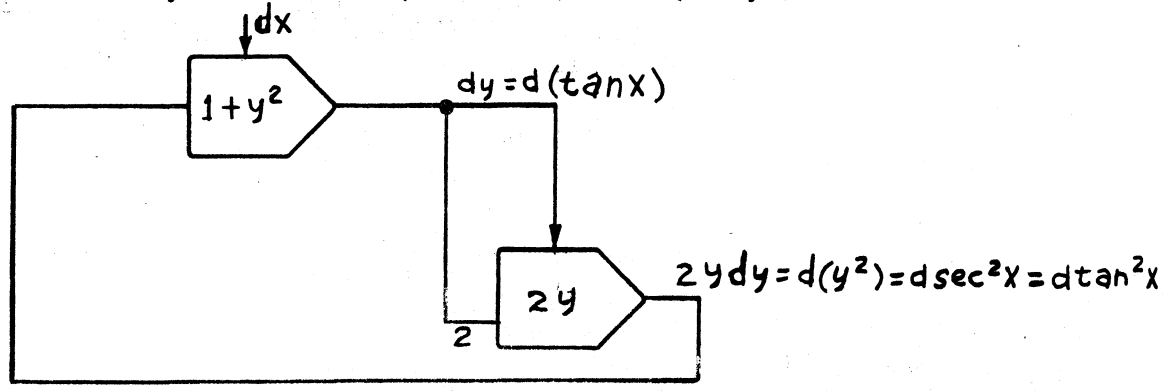


Figure 5

Note that multiplication with DDA may introduce a small but accumulating error. Consider:

$$d(uv) = udv + vdu$$

and

$$u_{n+1} = u_n + du$$

$$v_{n+1} = v_n + dv$$

Then $d(uv)_{n+1} = u_n dv + v_n du + 2\,\underline{dudv}$

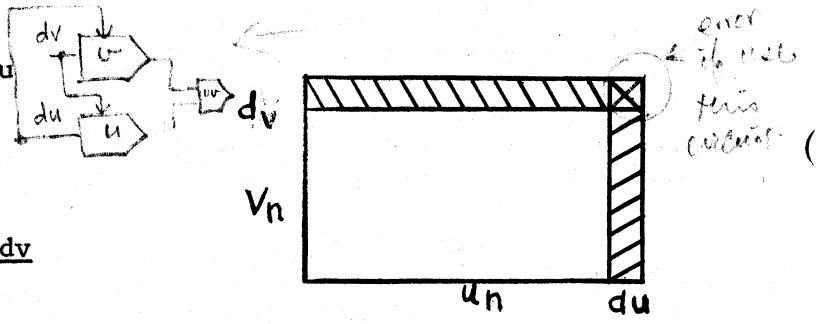


Figure 6

It should be clear from Figure 6 that the correct form is

$$d(uv)_{n+1} = u_n dv + v_n du + \underline{dudv}$$

The dudv term may be completely neglected or added twice. This is unavoidable on some machines, and properly compensated for on others.

In general, the programmer is limited to non-linearities which can be described by differential relationships; such as products, powers, roots, and transcendental functions. Some DDA's are capable of simulating non-linearities like hard and soft limiting, backlash, etc.

For input-output of data, present day DDA's employ these devices:

Program Input	"Live" Input (function generator)	Output (on line)
Paper tape	Curve follower (single variable)	Curve plotter
Flexowriter	Tabulated functions (multi-variable)	Paper tape
Cards		Flexowriter
Keyboard		

II. Operation

The following discussions refer to binary arithmetic operations as well as decimal arithmetic, for both decimal and binary machines exist. The only commercial models now on the market perform binary arithmetic (Litton 20, -40, Packard-Bell TRICE, and the Bendix DA-1, which is an attachment to the G-15 general purpose digital computer). Earlier models, the CRC-105 and Bendix D-12, use binary coded decimal logic.

Since integrators are used to perform all computing functions except output recording, it should suffice to describe the theory of operation of a DDA integrator. Actually, there are a few other special computing elements, the DDA multiplier and the DDA servo, but these involve only simple modifications of the basic integrator and will be mentioned later.

First the integrator will be described functionally, as if parallel arithmetic were performed. Then the operation of the serial DDA will be indicated. Let the integrator be represented by two registers, Y_r and R_r, Figure 7, one dy input pulse rate which increments or decrements the Y_r register, one dx input pulse rate which causes the contents of Y_r to be added to or subtracted from the contents of R_r, and one output pulse rate, called dz.

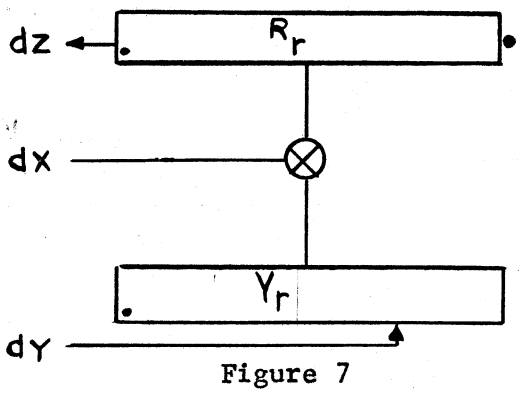


Figure 7

The number if Y_r may be positive or negative and must be less than one in magnitude. The decimal point is at the left end of the register. If Y_r overflows the computer will stop. Initial values of the variable y can be placed in Y_r at the beginning of the problem. The number in R_r may be positive or negative and may increase in either direction until an overflow or underflow occurs. Such an occurrence produces a dz output pulse, either positive or negative. The sum of these pulses in another integrator is proportional to $\int y dx$. Initially R_r is set to a value of one half. The number in R_r represents the fractional remainder of the summation $\sum_n Y_n \Delta x_n + R$

which is equivalent to a fraction of a revolution of the dz shaft in the mechanical differential analyzer. Each positive dx pulse causes subtraction of Y_r from R_r and each negative dx pulse causes subtraction of Y_r from R_r. The overflow of R_r, creating a $\pm$ dz pulse, corresponds to completion of a full clockwise or counter clockwise dz shaft rotation.

The representation of numbers in a DDA is unique and is best described by the table on the next page. The full range of possible numbers lies between plus one and minus one. The number of possible numbers between these extremes depends upon the number of stages in a register. The difference between two successive numbers is the "least count" of a register, or "one" in the last place: 2^{-n} for a binary machine with n-bit registers or 10^{-n} for a decimal machine with n-digit registers. Let this least count be denoted by Δ.

Number		Binary		Decimal
+1.0	.....	0.000000	.....	0.0000
1.0-Δ	.....	1.111111	.....	1.9999
1.0-2Δ	.....	1.111110	.....	1.9998
1.0-3Δ	.....	1.111101	.....	1.9997
*		*		*
0.875	.....	1.111000	.....	1.875
0.75	.....	1.110000	.....	1.75
0.5	.....	1.100000	.....	1.5
0.125	.....	1.001000	.....	1.125
*		*		*
0.0+3Δ		1.000011		1.0003
0.0+2Δ		1.000010		1.0002
0.0+ Δ		1.000001		1.0001
0.0	.....	1.000000	.....	1.0000
0.0- Δ		0.111111		0.9999
0.0-2Δ		0.111110		0.9998
0.0-3Δ		0.111101		0.9997
*		*		*
-0.125	.....	0.111000	.....	0.875
-0.5	.....	0.100000	.....	0.5
-0.75	.....	0.010000	.....	0.25
-0.875	.....	0.001000	.....	0.125
*		*		*

-0.0+3Δ		0.000011		0.0003
-1.0+2Δ		0.000010		0.0002
-1.0+ Δ		0.000001		0.0001
-1.0	.....	0.000000	.....	0.0000

The digit to the left of the point is the sign indicator, one for plus, zero for minus. This is a binary stage and is the same in binary and decimal machines. Note that in both binary and decimal the addition of an increment to the largest number and the subtraction of an increment from the smallest number results in 0.00000 which in some cases may stop the computer but in other applications may be used to advantage. If + 1.0-Δ is considered the largest number possible in Y_r (1.9999 in machine) and -1.0 the smallest (0.0000), then the presence of a plus or minus pulse in the lowest decimal order at the input, dy, can cause Y_r to change from maximum to minimum value or vice versa. This is equivalent, in a sense, to a_n analog computer integrator driving a relay. Such a device is called a DDA servo. It may be used to generate many special functions but has limited value because of the large oscillations it causes.

An integrator may be used for multiplication by a constant simply by not supplying a dy input. For simplicity some DDA's provide constant multipliers which are integrators without provision for dy inputs.

To avoid the error in multiplication described earlier some DDA's have special multiplier units which are simply two integrators connected so as to compute the following:

$$d(uv)_{n+1} = (u_n + du)\, dv + v_n\, du$$

III. Computer Design

It was mentioned before that present day DDA's do not require the construction of individual integrators, each with registers, gates, and adders. By sequencing the operations of each integrator an entire DDA computer can be constructed with one magnetic drum, two serial adders and a few other simple elements. That is, each register is a cell on the drum and each output pulse is also held on a special track on the drum. In one drum revolution the following steps, Figure 8, take place for each integrator in turn:

a. Detect input pulses, dy.

b. Read y from the drum and add (or subtract) in a serial adder.

c. If a dx pulse is present feed the result of step (b) into a serial adder with the contents of R which is read from the drum in proper synchronism.

d. If there is a carry from the highest order in step (c) place a pulse on the dz track.

Actually step (d) for integrator i takes place simultaneously with step (a) for integrator i +1.

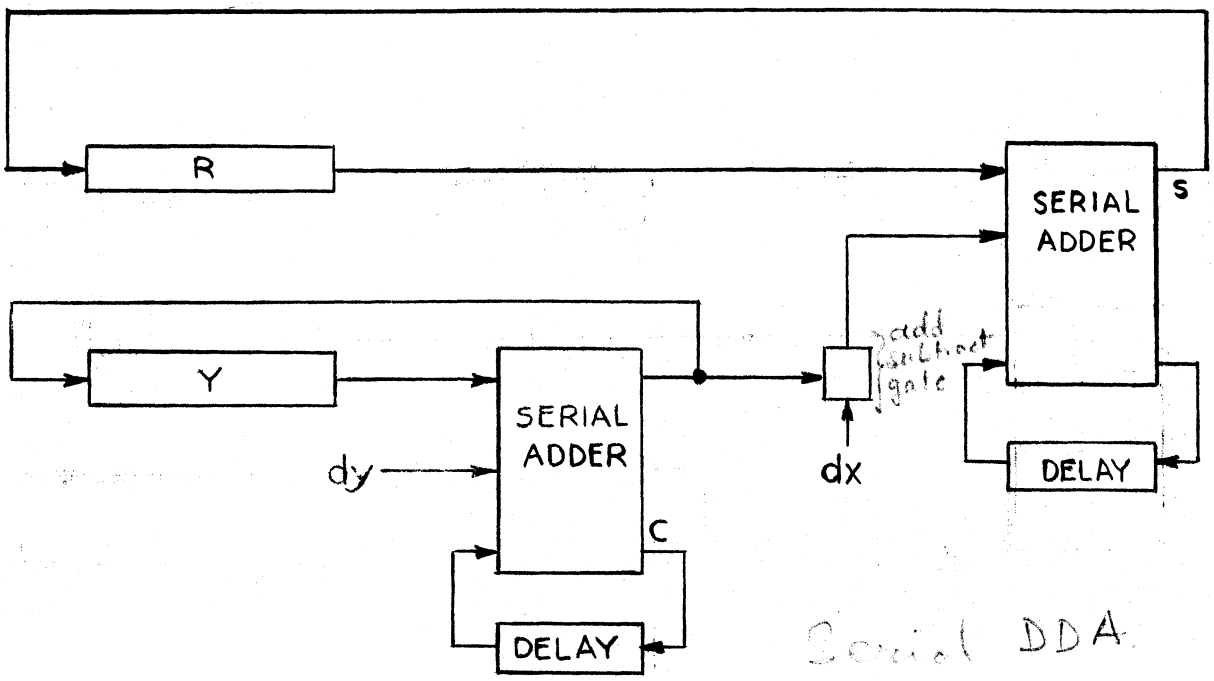


Figure 8

The following is a detailed description of a serial DDA. (Refer to Figure 9)

Present values of Y_i and R_i (i = 1, 2, ... N) are stored on parallel tracks. Each cell Y_i or R_i contains N bits. There are N cells around the drum (N integrators).

There are similarly N cells in tracks L_y and L_x. Cell L_{yi} contains bits in the j^{th} positions (j = 1, 2, ... N) corresponding to dy inputs to integrator i from integrator j. Track L_x contains similar programming information for the dx inputs.

Track Z is not divided into cells but has NxN bit-storage positions around the drum. A pair of read-write heads are separated by N-1 positions with the read head feeding the write head so that a bit read at position r is simultaneously written in position r + N-1 (position numbers increasing as they come under a head). Thus there is a continually circulating and precessing storage on N-1 numbers in track Z. The block of N-1 numbers shifts one position

relative to the read-write head in N pulse times. The numbers on this track are the dz outputs or overflow from the R registers. As each integrator is serviced a new dz is created and is written on the Z track by the write head (overriding the signal from the read head), so that after every N pulse times (p_n) a new dz_r is added to the precessing loop of dz_i's and the old dz_r destroyed.

The N-1 dz output pulses from the previous N-1 integrators all pass under the Z track read head during the processing time of one integrator. Thus coincidence of these dz pulses with programmed information on the L_y track, in the dy decoder, generates a dy input which is added to the value Y_i in the serial adder. This sum or its complement is gated by a plus or minus dx input from the L_x and Z tracks. Thus Y_i + dy is added to R_i. The sum is written on the R track and any overflow in the highest order (pulse time, p_n) is placed on the Z track. Start pulses from track Y and clock pulses generate "s" pulses which occur for each utilized binary place in each Y_i. The clock track also generates the dz output synch pulse, p_n and the basic computing rate, dt, (one pulse every drum revolution).

IV. Scaling

The following discussion will consider scaling in detail and some of the difficulties which occur in programming a DDA. Scaling is very similar to analog computer scaling, buy requires a few simple tricks which are unique to the DDA. Scaling reveals a peculiarity of the DDA, which is most disturbing: accuracy is partially dependent on scaling.

The dynamic range of an analog computer variable (voltage, or shaft position) is determined on the low side by one or more quantities which are not directly under the control of the designer. For a chopper stabilized operational amplifier these are the noise voltage level and the stabilizer offset voltage. The granularity of the "follow-up" potentiometer and, again, noise voltages determined the maximum resolution of a servo-multiplier. The upper limit to dynamic range of an analog variable is directly determined by the designer and is usually some convenient reference level: $\pm$ 10 volts, $\pm$ 30 volts, $\pm$ 100 volts, and 360° of shaft rotation. Therefore, as a practical matter the lower limit may be considered fixed and the upper limit determined by the required dynamic range. The ratio of the upper limit to the lower is the number of significant or distinguishable levels that the variable may assume -- or the resolution.

The DDA variable is a numeric quantity held in a register of n decimal (or binary) stages. The decimal point (or binary point) is always at the extreme left hand end. Thus in contract to the above the upper limit to the dynamic range is fixed at $1.0 - 10^{-n}$ (or $1.0 - 2^{-n}$), or the maximum number ,

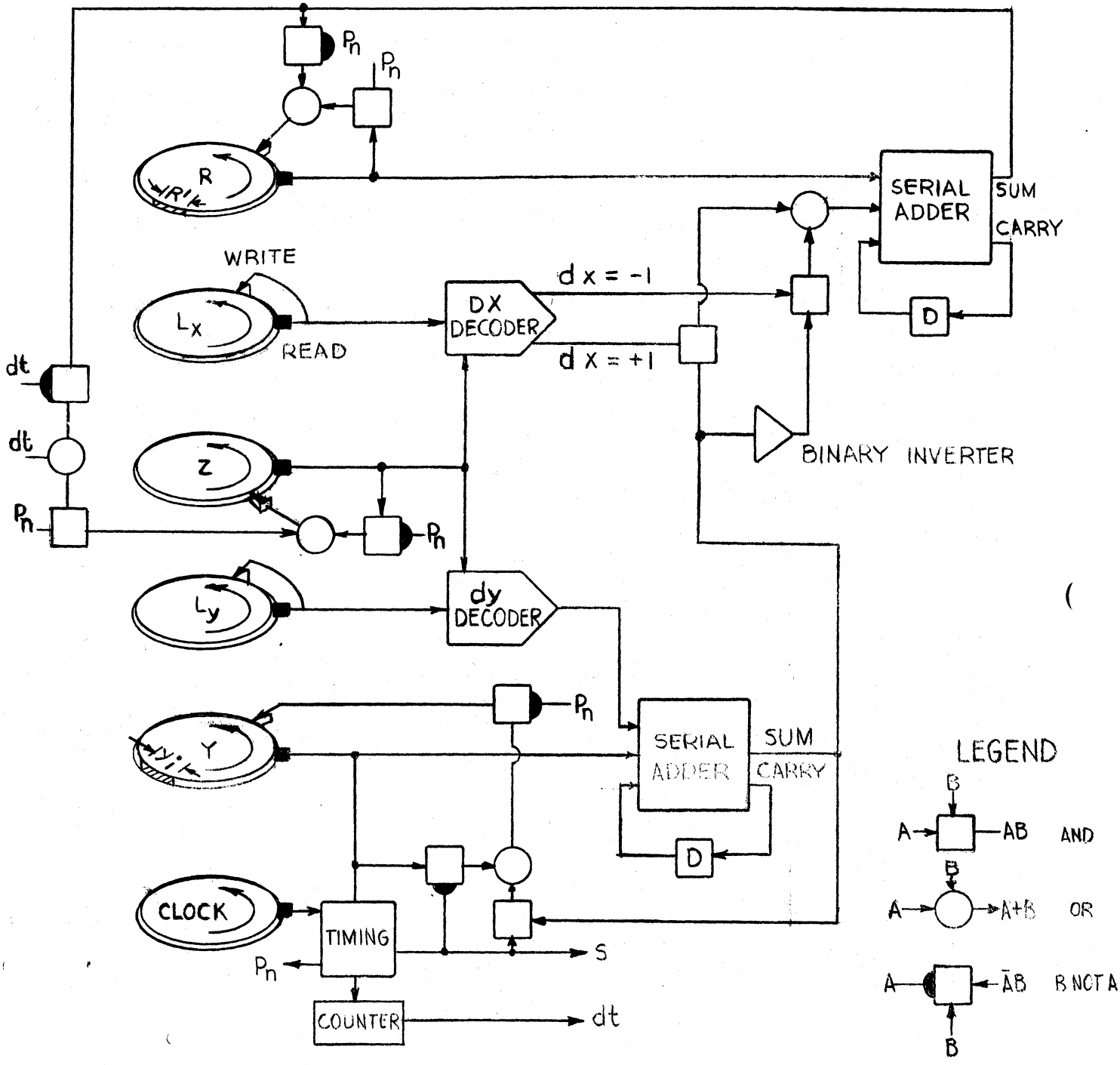
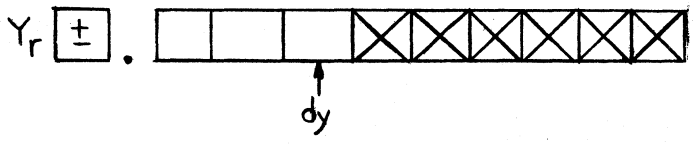


Figure 9

Serial DDA

less than unity, possible with an n-stage register, while the lower dynamic range limit is determined by the length of the register and, therefore, is controlled by the designer. The resolution of the DDA or the number of significant quanta assumed by the variable is, therefore, 10^n (or 2^n). To make the significance of the above contrast clear, note that an analog variable when properly scaled will utilize the full dynamic range of the analog device and a constant voltage near the maximum value, say, 95 volts, will be precise to within the full resolution of the device. Now further note that a DDA variable when properly scaled should approach the maximum limit of the register asthe variable assumes its maximum value, but that this does not mean that the full dynamic range of the device is utilized. In fact, if the variable has a value of 0.9500 the precision of the DDA variable is not determined by the resolution of the device, 10^n, but rather by the scale of the input variables. The scale of an input quantity (dy) to a DDA integrator directly determines the number of decimal (or binary) places of the Y_r register that will be utilized. The remaining stages, to the right, become superfluous. For example, if y has a unity scale factor and one dy pulse is assigned a weight of 10^{-3}, only the first three significant places of Y_r are utilized, as shown.



Thus fluctuations in the variable y smaller than 10^{-3} will not appear in Y_r although they do occur in the computer, and they may be significant. Such fluctuations take place in the R register of the integrator whose output pulse rate is the input dy rate above. Only Y registers can supply output information for recording purposes. Thus, in the above example, y can be recorded to no greater accuracy than 1 part in 1000.

$$\text{Since } Y_1 = \int Y_o \, dx_o + R_o = \int dy_1 + R_o,$$

where R is the fractional remainder of the summation, the true value of y_1 may be ± abcdefghijk, while only ± abcde can be recorded. Figure 10.

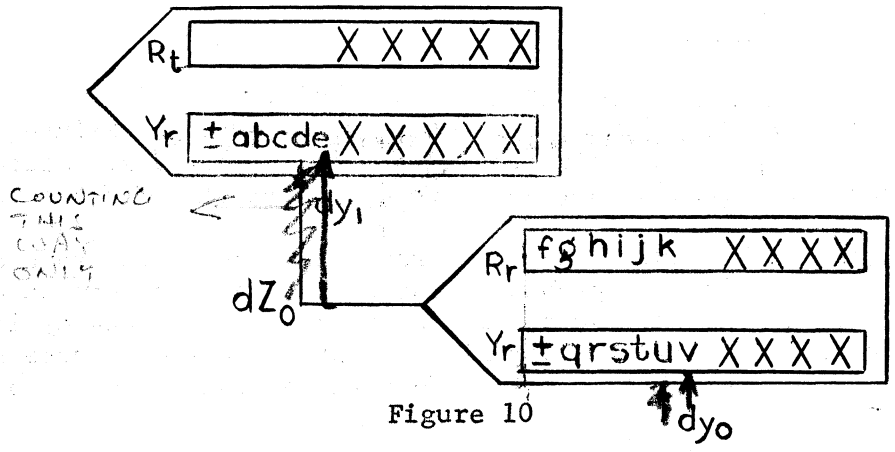


Figure 10

It was mentioned above that the initial value usually assigned to all R registers is 0.5. A more correct procedure would be to compute the correct value for each R register from the precise initial value of the Y register it supplies. For example:

$$Y_1(0) = +2/3 = +.66666\ 666667; \quad R_o = 666667$$
$$Y_1 \qquad R_o$$

Such a procedure becomes exceedingly difficult when one considers that each R_r may supply several Y_r, and each Y_r may have several inputs. In fact the setting of initial conditions on a DDA often requires trial and error methods in order to determine a set of <u>effective</u> initial conditions. *

The preceeding difficulties might be called "scale mark" errors. The scale mark sets the effective length of the y register. Scale mark errors are most likely to effect parameters with large values, limiting them or their derivatives to as little as 2, 3, or 4 place accuracy in a 10 place machine and it may be precisely these variables (altitude, range) which have caused one to look for so-called digital accuracy.

The situation is further complicated by more than one integrator feeding another

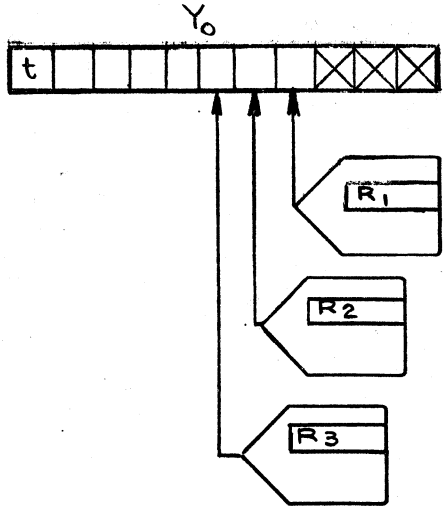


Figure 11

In this case, the true value of y is Y_o plus a combination of R_1, R_2, R_3, i.e.,

$$Y = Y_o + 10^{-5}R_3 + 10^{-6}R_2 + 10^{-7}R_3$$

---

* Cf: <u>Digital Differential Analyzer</u>, by G. Forbes
"On the initial cycle of calculation, the first pulse to each integrator will usually be negative. The numerical order of the integrators, various componsating techniques, and the specific design of the machine will influence this effect.

"The effective initial value of any integrand will not be the same as the value actually filled in coding the problem. The effective value can be approximated by reading all variable integrands for the first ten or twenty cycles and plotting each. A smooth curve making an approximate best fit through these points will sometimes permit determination of the effective initial value to a greater accuracy."

Figure 11 suggests another serious limitation of some DDA's: multiple inputs require different input scale factors. While this can be helpful in some situations it is generally considered a limitation, a burden and an added complication to programming.

EXAMPLE OF SCALING

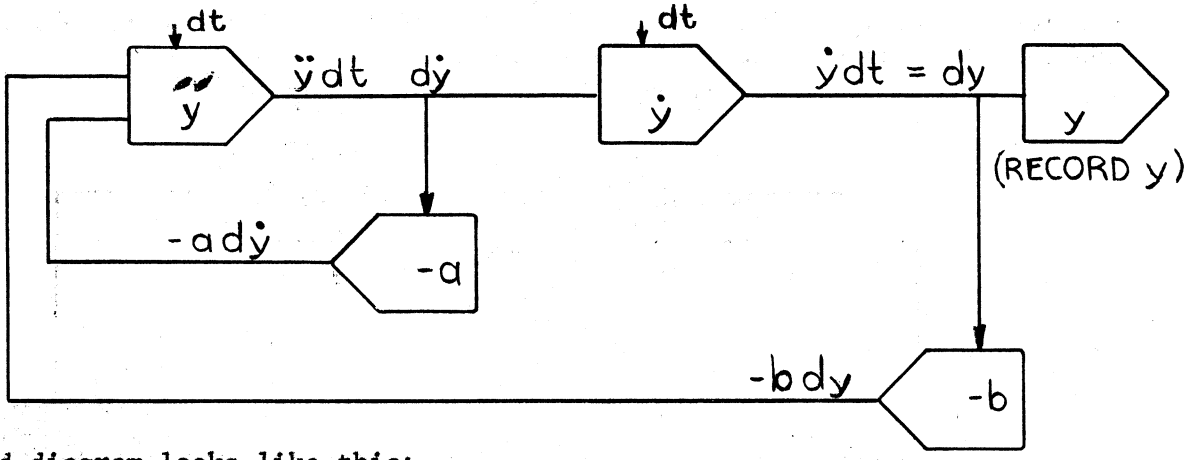
Consider: $\ddot{y} + a\dot{y} + by = 0$

Range of variables $50 < \ddot{y} < 80$ $\quad a = .005$

$+950 > y > -950$ $\quad b = .01$

Let the step in time, $dt = 10^{-4}$ sec; $+100 > y > -100$

Then $d(\dot{y}) = -ad\dot{y} - bdy$



The scaled diagram looks like this:

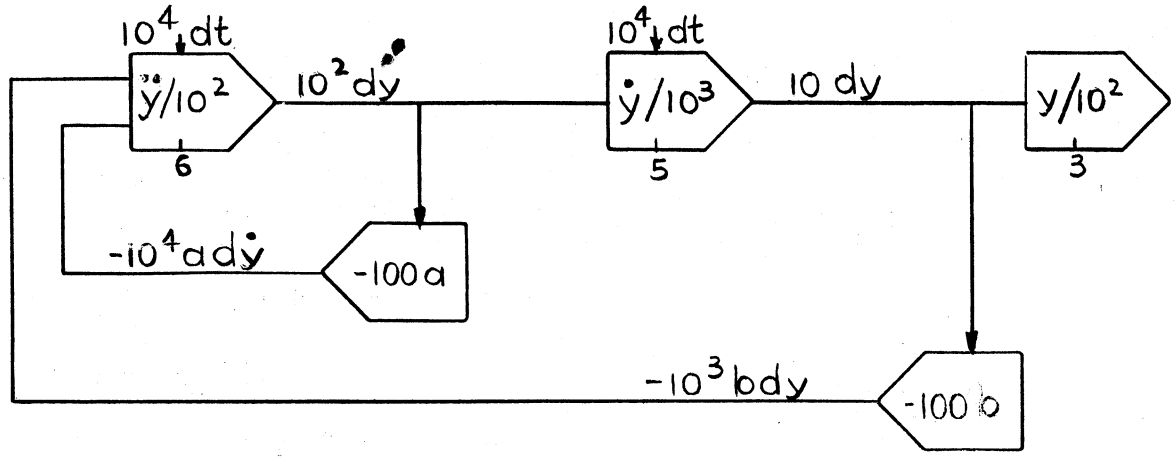


Figure 12

The numbers under the integrators, Figure 12, (except for constant multipliers) are called scale marks. They are equal to the difference between the dy and y scale factors and they represent the effective length of the Y register in each case. Note that the two inputs to the y integrator are at different scale factors. This particular difficulty is handled differently by each DDA but usually involves some scaling restriction of this sort.

Note that the values of a and b could adversely effect the scaling. In this case larger values of a or b would reduce the scale mark of one or more of the top three integrators.

It is of interest to consider this situation in a more general way. Let us choose scale factors S_i such that

$$|y_i \times 10^{S_i}| < 1$$

and let the step in time be $dt = 10^{-\tau}$ so that $10^{\tau}\, dt$ = one unit of time. Now consider an undamped second order system:

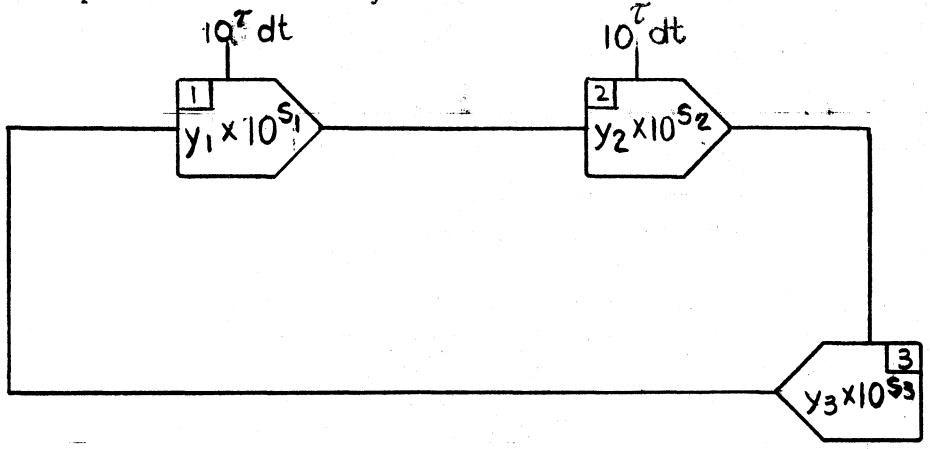


Figure 13

$$y_3 = -\omega^2 = \text{a negative constant}$$

$$dz_1 = y_1 dt \times 10^{S_1 + \tau}$$

$$(y_2 \times 10^{S_2})_{n+1} = (y_2 \times 10^{S_2})_n + (y_1 dt \times 10^{S_1 + \tau}) \times 10^{-k_2}$$

where k_2 = scale mark = $S_1 - S_2 + \tau$

$$dz_2 = y_2 dt \times 10^{S_2 + \tau}$$

$$dz_3 = (y_3 \times 10^{S_3})\, dz_2 = y_x y_2 dt \times 10^{S_2 + S_3 + \tau}$$

$$\text{and then } (y_1 \times 10^{S_1})_{n+1} = (y_1 \times 10^{S_1})_n + (y_x y_2 dt \times 10^{S_2+S_3+\tau}) \times 10^{-k_1}$$

$$\text{where } k_1 = \text{scale mark} = S_2 + S_3 - S_1 + \tau$$

Thus it follows that

$$k_1 + k_2 = 2\tau + S_3$$

Therefore it is concluded that the total available register length, $(k_1 + k_2)$, is determined by the step size (dt) and the maximum loop gain (since $y_3 = -\omega^2$) and $(y_3 \times 10^{S_3} < 1)$. This is comparable to the analog computer situation where the loop gain is equal to $-\omega^2$; except that in the DDA, computer speed is fixed by dt, and the natural frequency of the system (ω) determines the maximum resolution in y, i.e., the size of the dy step, and thus the register length.

Therefore, if the loop gain is >1, i.e., $y_3 > 1$, then S_3 is negative; or, as expected, the resolution in y_i must go down with higher frequency.

Pursuing this approach with higher order and with cross-coupled systems leads to the conclusion that:

(a) For an Nth order system the total available register length is $\sum_{n=1}^{N} = k_n = N\tau + \sum S_i$, where S_i are the scale factors for all constant multipliers in the loop.

(b) Coupling between loops can under certain circumstances introduce scaling limitations which reduce the total available register length.

Going back to the undamped second order systems let the sole requirement be maximum accuracy (or at least full utilization of all registers); let the registers have 10 decimal places and then determine dt from the relation

$k_1 + k_2 = 2\tau + S_3$. Let $\omega^2 < 100$, so $S_3 = -2$ (natural frequency < 1.6 cps).

Then $k_1 + k_2 = 20 = 2\tau - 2$, or $\tau = 11$, $dt = 10^{-11}$.

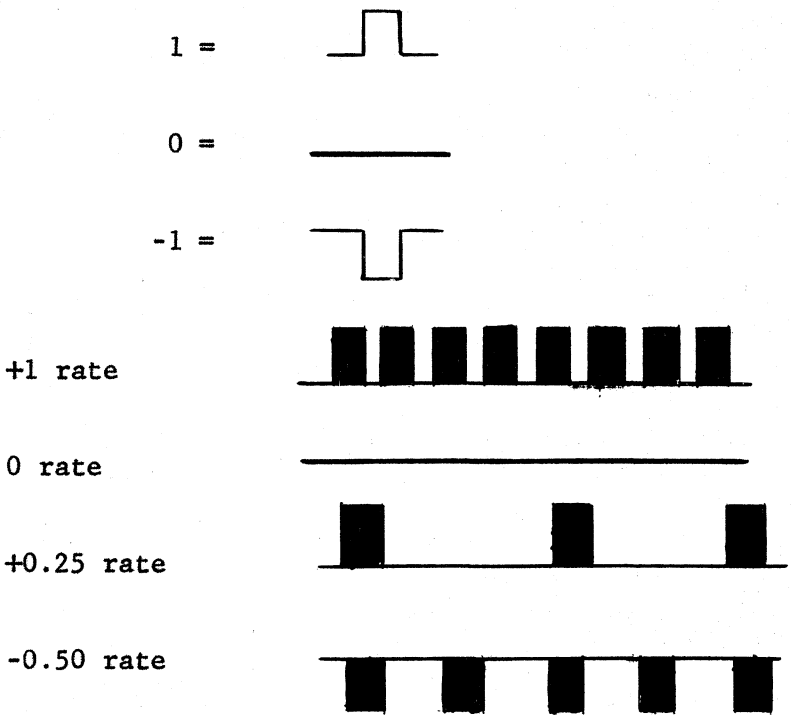
Assuming a machine with a clock rate of 2×10^5 pulses/sec (which is higher than any existing DDA) the solution of the differential equation $y = \omega^2 y$ for one cycle (or about 0.6 seconds of machine time) will require $0.6 \times 10^{11} / 2 \times 10^5 = 0.3 \times 10^6$ seconds, or 83 hours. And this assumes parallel operation

of the DDA integrators. For four decimal accuracy, $k_1 = k_2 = 4$, $\tau = 5$, one cycle would take 0.3 seconds of real time and, therefore, operates twice as fast as real time. Thus a 10^6 increase in resolution costs a 10^6 increase in time of operation.

V. DDA Errors

An adequate error analysis of the DDA has not been made. DDA errors are not generally amenable to analysis by existing numerical techniques, primarily because of the asynchronous nature of the pulse trains. Therefore, the following is limited to a few general remarks.

The two kinds of erros inherent in digital computation are known as truncation error and round-off error. Truncation error is a measure of the degree of accuracy attained in approximating an integral by a summation of small discrete quantities. The ternary output scheme:



The Packard Bell TRICE computer is a parallel DDA in the sense that a number of integrators, rate adders and servos are interconnected at a plug board, and each component computes simultaneously. However, the operations within each component are serial in nature. The addition, $y_i = y_{i-1} + dy$ and $R_i = R_{i-1} + Y_i$ are performed by serial adders -- delay lines being used for storage, in addition to the R, Y and "initial conditions" registers. The basic clock rate for this all solid state computer is quite high: 3 mc. However, the

serially nature of the device results in an iteration rate or effective clock rate of 100 kc. The TRICE uses a ternary output scheme, "trapezoidal" integration, and serial binary logic.

Truncation error can be reduced in only two ways:

(a) By decreasing the size of the step in the independent variable (often called the mesh size), at the expense of speed.

(b) By using a more sophisticated scheme in forming Δy in the summation

$$y_N = y_o + \sum_{n=1}^{N} \Delta y_n$$

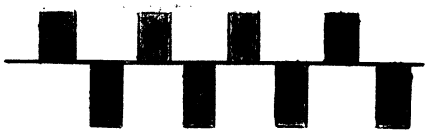
The two schemes presently used by DDA's, very unsophisticated mathematically, are known as rectangular and trapezoidal integration. They correspond to approximating the area under a curve with small rectangles or trapezoids. Clearly the latter is more accurate in general.

While we know what is meant by round-off error when the least significant digits of a large or irrational number are dropped, it is not exactly clear, on the other hand, what is meant by round-off error when dealing with a sequence of pulses. The former type error does not trouble the DDA, but the latter does. Since the magnitude of the rate represented by the sequence of pulses is proportional to the average of the pulses (as described below) clearly there is some error due to changing the rate, but it is not the normal kind of round-off error. It is also clear that this error will be smaller if the number of discrete levels assumed by the pulse train is increased. Thus the so-called ternary pulse system, illustrated below and used by the Bendix DDA is superior to the binary scheme used by the Litton DDA. In the latter scheme the dz output of each integrator at every time interval is a positive or a negative pulse, as shown below

1 = +1 

0 = -1 

Maximum positive rate = +1 rate = 

Zero rate = 

Maximum negative rate = -1 rate =

+ 0.25 rate = 

The rate = $\frac{\text{net positive area}}{\text{total area under pulses}}$ 

Miscellaneous References

"Hybrid Techniques Applied to Optimization Problems," H. Witsenhausen, Proceedings of the Spring Joint Computer Conference - San Francisco, May, 1962.

"On the Continuous Solution of Integral Equations by an Electronic Analog -- Part I," M. E. Fisher, Proceedings of the Cambridge Philosophical Society, Vol. 53, January, 1957, pp. 162-174.

"Proposed Methods for the Analog Solution of Fredholm's Integral Equations," M. E. Fisher, Journal of the Association for Computing Machinery, Vol. 5, October, 1958, pp. 357-369.

"High Speed Memory Analog Computer," Stanley H. Jury and Jack M. Andrews, Industrial and Engineering Chemistry, Vol. 53, No. 11, November, 1961.

"Application of the Analogue Computer in the Study of the Esterification of Terephthalic Acid," J. E. Rijnsdorp, R. Vichnevetsky, and J. C. van de Vusse, Analog Computation Applied to the Study of Chemical Processes, International Seminar - Brussels, November 21-23, 1960.

"Combined Analogue and Digital Computing Techniques for the Solution of Differential Equations," P. A. Hurney, Jr., Proceedings of the Western Joint Computer Conference, presented at San Francisco, February 7-9, 1956.

"Time Multiplexing as Applied to Analog Computation," E. Rawdin, I.R.E. Transactions on Electronic Computer, Vol. EC-8, No. 1, March, 1959, pp. 42-47.

"Application of the Digital Expansion System (D.E.S.) to an Automatic Iteration Problem: (Eigenvalues Search)," R. Vichnevetsky, PCC Report No. 178, Electronic Associates, Inc., Princeton, N. J., March 8, 1962.

"Use of a Combined Analog-Digital System for Re-entry Vehicle Flight Simulation," Dr. Allan Wilson, General Dynamics - Astronautics, San Diego, Calif.

"Combined A-D Simulation in Analysis of Minuteman Control System," J. M. Johnson (Autonetics), Western Simulation Council, Sept. 14, 1961.
"Using a Digital Computer as a Function Generator for an Analog Computer," Otto Reichardt, Midwest Simulation Council, Sept. 18, 1961.

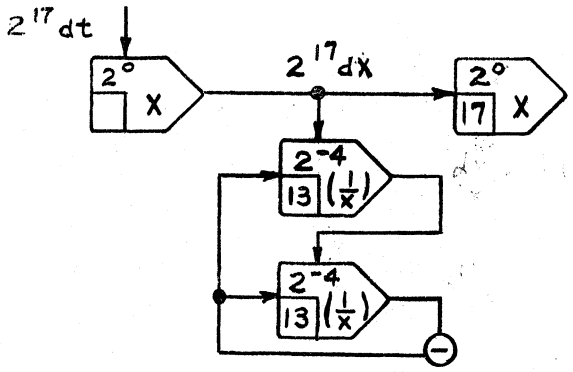
EXPERIENCES WITH A DDA-GP COMPUTER HYBRID SYSTEM *

Introduction

The greatest difficulties in scaling a problem for reasonable running times on a digital differential analyzer are encountered when implicit algebraic operations, such as division, are performed on variables with a wide range of possible values. This is due to a DDA's fixed point operation. It is most obvious when the algebraic functions are generated by solving a differential equation of which they form a solution. Frequently the time scale must be prohibitively slow in order to obtain reasonable accuracy in the functions. For example, suppose the terms x and 1/x are desired where:

$$-1 < x < 1 \qquad 1/15 < x < 1$$

and the coefficient of 2^{17} is assigned to dt, the machine iteration rate. Then the following circuit may be used:



Thus, in order to have the same 17 binary place accuracy in $\frac{1}{x}$ that is now available in x, the coefficient 2^{21} must be assigned to dt, slowing the problem time down by a factor of 16.

Use of implicit methods involving servos are sometimes more effective; however, certain hidden errors, such as the servo getting behind, can occur in these setups.

---

* By John P. Gibbons, NASA-Huntsville, Ala. March 1962

At the MSFC, Computation Division, it was found that using a General Purpose computer to automatically rescale a DDA for different ranges of the variables was quite successful on some problems. Following are descriptions of two sample problems solved and some of the results obtained by using this technique. One criterion used in choosing these examples was that they cannot be done satisfactorily on an analog computer because of the accuracy required.

I. A Celestial Mechanics Problem

A practical problem of this type which frequently occurs is a relatively small body traveling in the earth-moon plane with coordinates x and y. This is described by the equations:

$$\ddot{x} = \frac{-\gamma M_E X}{(X^2 + y^2)^{3/2}} - \frac{\gamma M_L (X - R)}{([X-R]^2 + y^2)^{3/2}} + \omega^2 X + 2\omega\dot{y} \qquad (14\text{-}1)$$

$$\ddot{y} = -\frac{\gamma M_E y}{(X^2 + y^2)^{3/2}} - \frac{\gamma M_L y}{([X-R]^2 + y^2)^{3/2}} + \omega^2 y - 2\omega\dot{x} \qquad (14\text{-}2)$$

Where the coordinate system is rotating at a velocity ω in order to keep the earth and moon fixed. For computer scaling purposes, these equations present the same problems, in principle, as the simple two body problem:

$$\ddot{x} = -\frac{\gamma M X}{(x^2 + y^2)^{3/2}} \qquad (14\text{-}3)$$

$$\ddot{y} = -\frac{\gamma M y}{(x^2 + y^2)^{3/2}} \qquad (14\text{-}4)$$

Since (14-3) and (14-4) may be checked analytically, these were the equations actually solved by direct integration. The DDA circuitry used is shown in Figure 1. The register lengths and binary scale factors of the variables and time are given in terms of j and k, the negative of the binary scale factors for velocity and distance respectively. Since the TRICE operates at 10^5 iterations per second, 2^{17} dt represents approximately real time operation.

Each time an overload in the DDA occurs, operation halts and the contents of the "Y" registers are read into the General Purpose computer. These readings are modified appropriately and read back into the "IC" registers. The DDA is now reset and operation continues. The modifications followed the following logical scheme.

Element Overloaded		Changes Made
(1) 208	$(2^{-j}\dot{x}$ and $2^{-j}\dot{y} < \frac{1}{2})$	$j \rightarrow j - 1$
(2) 101 or 201	$(2^{-j}\dot{x}$ or $2^{-j}\dot{y} > 1)$	$j \rightarrow j + 1$
(3) 106, 107, or 108	$(2^{k-1}\ (\frac{1}{r}) > 1)$	$k \rightarrow k - 1$
(4) 105	$(2^{-k}r > 1)$	$k \rightarrow k + 1$

Reductions in j were limited since each such reduction caused a net loss of one place of accuracy in the velocity terms.

The eliptic trajectory generated by the values:

$$\gamma M = 3.98533 \times 10^5 \quad km^3/sec^2$$

$$\dot{x}_o = 9.35 \text{ km/sec} \qquad \dot{y}_o = 5.5 \text{ km/sec}$$

$$X_o = 0 \qquad Y_o = 6400 \text{ km}$$

$$j_o = 4 \quad k_o = 13 \Rightarrow 2^{11}\, dt \sim 1 \text{ min. per hr.}$$

is shown in Figure 2. At t = 72 hours (approx. 1 2/3 orbits) the system had accumulated errors in r of about 150 km and of less than 2^o in θ. The run took about 5 minutes.

This demonstrates the effectiveness of the rescaling technique. Without it, a run as long as 12 hours would be required to prevent the gravitational acceleration (i.e. the $1/r^3$ term) from completely vanishing before the body reached apogee. To obtain comparable accuracy would require running times longer than real time (72 hours).

On a high speed General Purpose computer (e.g. IBM 7090) with optimum variable interval programming, this problem can be solved in approximately 10 minutes.

II. An Eigenvalue Proglem

One problem received by the MSFC, Computation Division, was to find those positive constant values of W for which $P(o) = \lim_{r\to\infty} P(r) = 0$ in the equation:

$$\frac{d^2P}{dr^2} + \left[\frac{2e^{-r/d}}{r} - \frac{1(L+1)}{r^2}\right] P = WP \qquad (14\text{-}5)$$

where L is an integer ≥ 0, and d is a positive constant.

As $r \to 0$ $e^{-r/d} \to 1 - r/d$ and equation (14-5) becomes

$$\frac{d^2P}{dr^2} + \left[\frac{2}{r} - \frac{L(L+1)}{r^2}\right] P = (W + \frac{2}{d}) P \equiv W^1 P \qquad r \backsim 0 \qquad (14\text{-}6)$$

which has the solution:

$$P = Ne^{-\sqrt{W^1}r} r^{L+1} \sum_{n=o}^{\infty} a_n r^n \qquad (14\text{-}7)$$

where all a_i's are known and N is arbitrary. For convenience, let U be defined by $r^{L+1} U = P$ giving, from eq. (14-5)

$$U^{11} + \frac{(L+1)}{r} U^1 + \frac{2e^{-r/d}}{r} U = W U \qquad (14\text{-}8)$$

A small value of $r = r_o$ (machine time) was chosen and the first two terms in equation (14-7) used to approximate $\frac{U(r_o)}{U^1(r_o)}$. Several runs were then made changing W values in equation (14-8).

The working assumption was that if, with d and L fixed, a setting of W_1 generates k roots of U before divergence and $W_2 < W_1$ generates k + 1 roots, then one eigenvalue of W exists between W_1 and W_2.

The circuit used to solve equation (14-8) is shown in Figure 3. In this case, the General Purpose computer is used both to rescale the TRICE and to "hunt" the eigenvalues.

For purposes of demonstration and as an accuracy check, the following equation was solved on the computer:

$$U^{11} + \frac{(L+1)}{t} U^{1} + \frac{2}{t} U = W\ U \qquad (14\text{-}9)$$

The eigenvalues of this equation are known to be, $W_n = \frac{1}{n^2}$ where n is any positive interger. Since, from a computer viewpoint, generation of $e^{-t/d}$ is several orders of magnitude more accurate than the other integrations in the circuitry, it is reasonable to suppose that the accuracy of equation (14-9) is a good guide to the accuracy of equation (14-8). Actually, for given t_o's, running times, and running rates, equation (14-8) probably gives more accurate determinations of W since it has fewer cycles. Some results of solving both equations (14-8) and 14-9) are shown in Figure 4. The General Purpose computer's rescaling logic was quite simple. Each time t changed by a factor of two dt x 2^k went to dt x 2^{k-1}, from k = 25 to k = 17. An overload in 103 causes this transition.

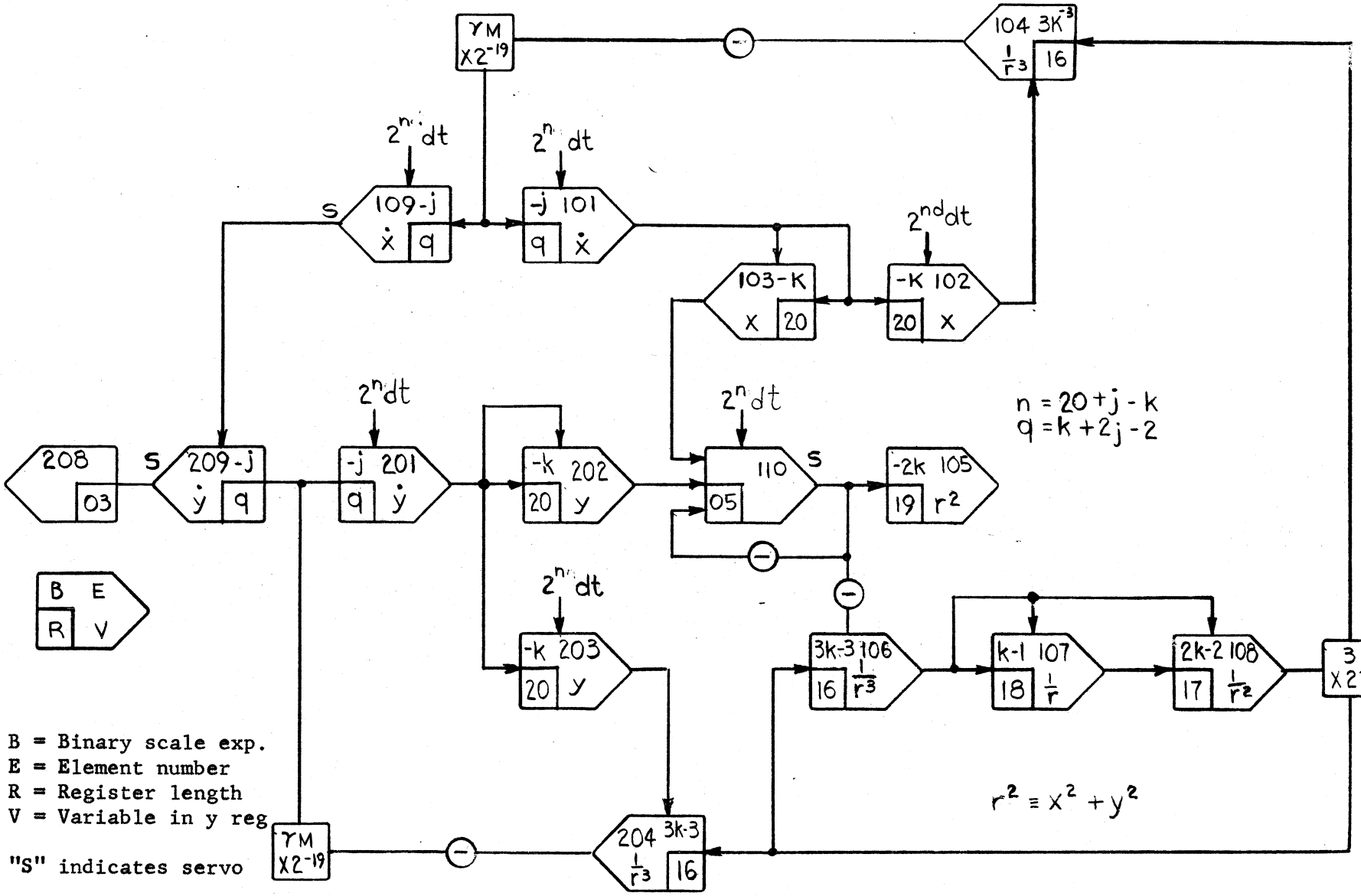

γM X2^-19
104 3K^-3 1/r^3 16
2^n dt
2^n dt
S 109-j ẋ q
-j 101 q ẋ
2^nd dt
103-K X 20
-K 102 20 X
2^n dt
2^n dt
n = 20 + j - k
q = k + 2j - 2
208 03
S 209-j ẏ q
-j 201 q ẏ
-k 202 20 y
110 05
S
-2k 105 19 r^2
B E R V
2^n dt
-k 203 20 y
3k-3 106 16 1/r^3
k-1 107 18 1/r
2k-2 108 17 1/r^2
3 X2^-2
B = Binary scale exp.
E = Element number
R = Register length
V = Variable in y reg
"S" indicates servo
γM X2^-19
204 3k-3 1/r^3 16
r^2 ≡ x^2 + y^2

Fig. 1

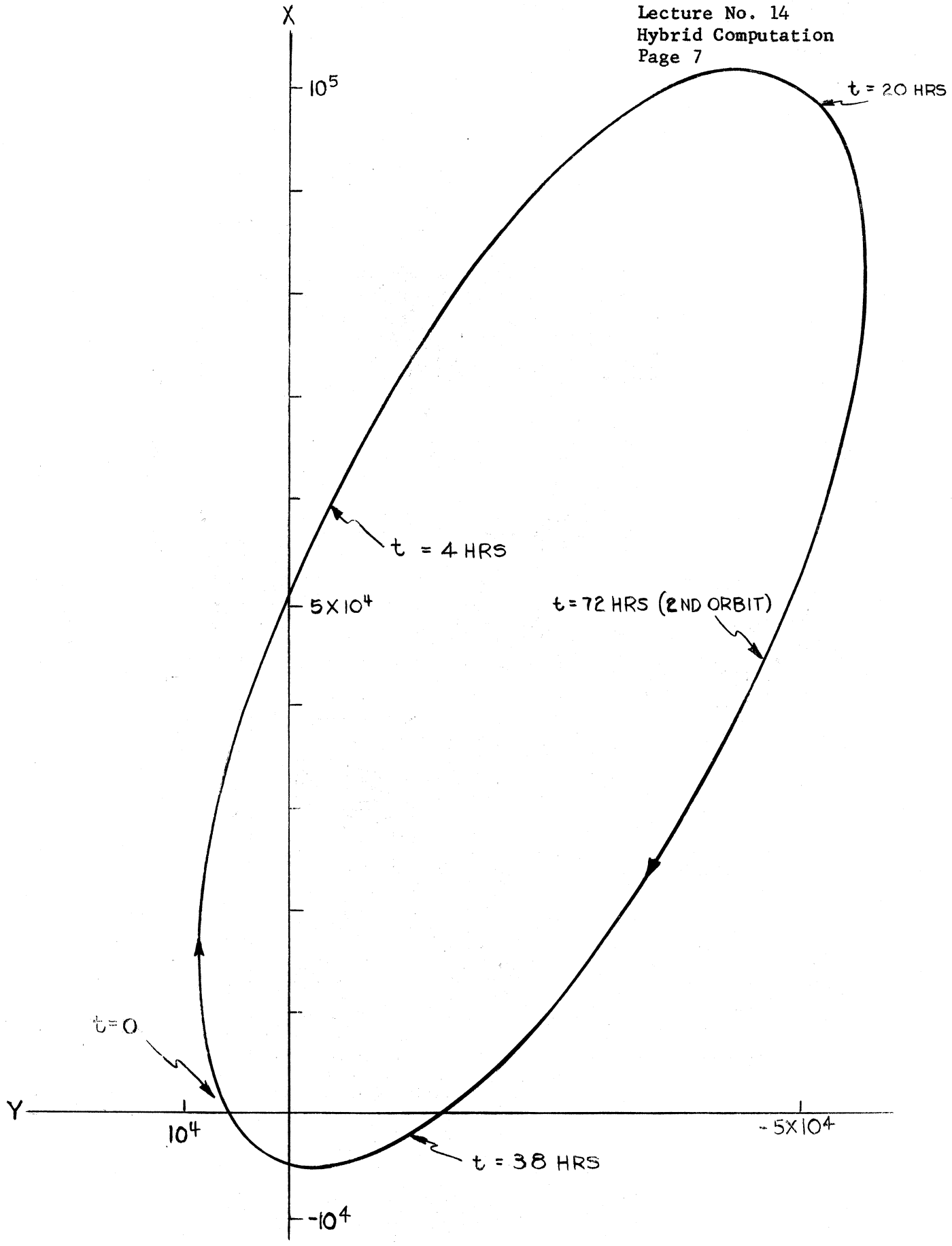
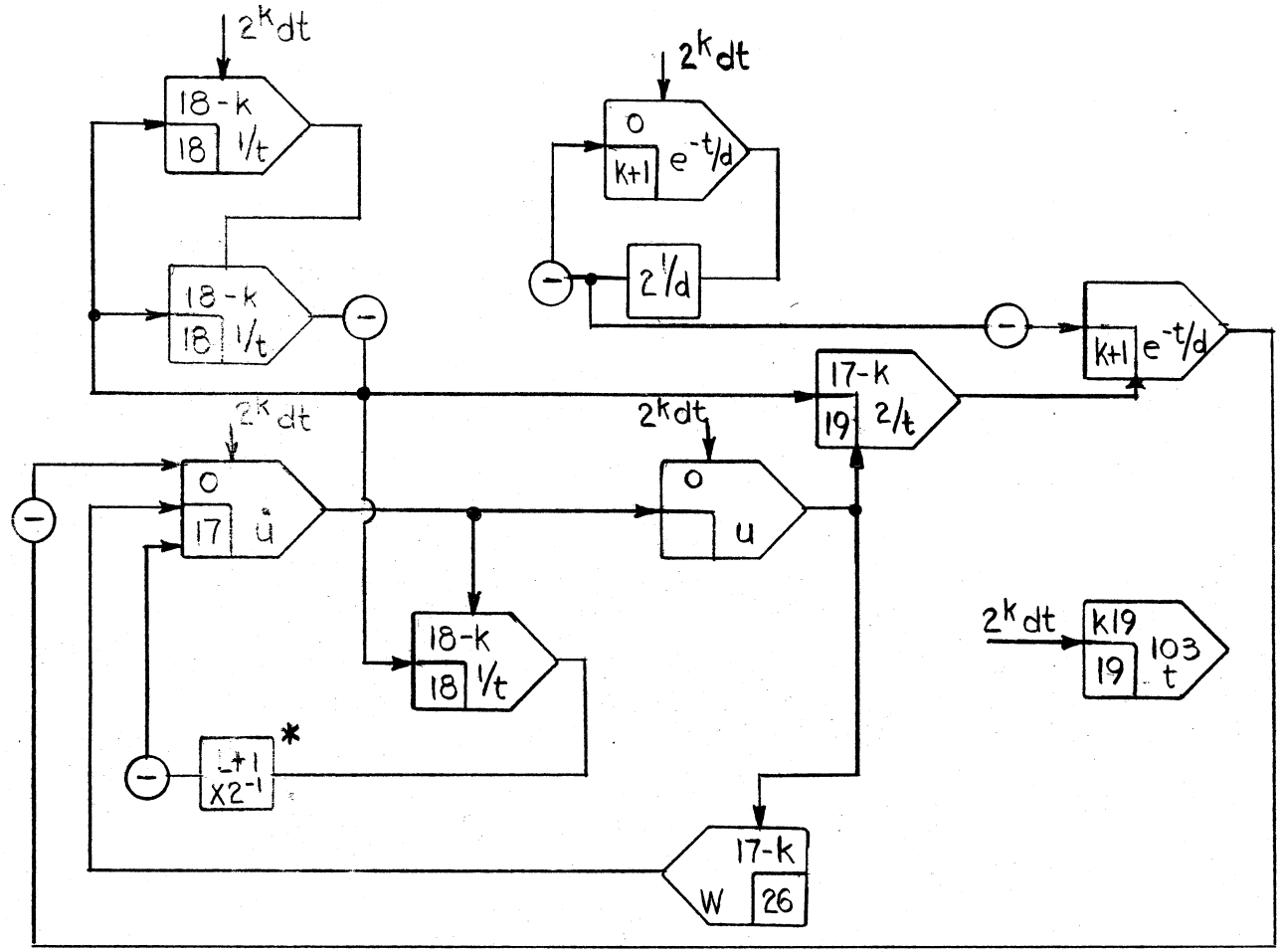

X
10^5
t = 20 HRS
t = 4 HRS
5×10^4
t = 72 HRS (2ND ORBIT)
t = 0
Y
10^4
-5×10^4
t = 38 HRS
-10^4

Fig. 2



* SCALED FOR L=0, 1

[Eq (5) ↔ d = ∞]

Fig. 3

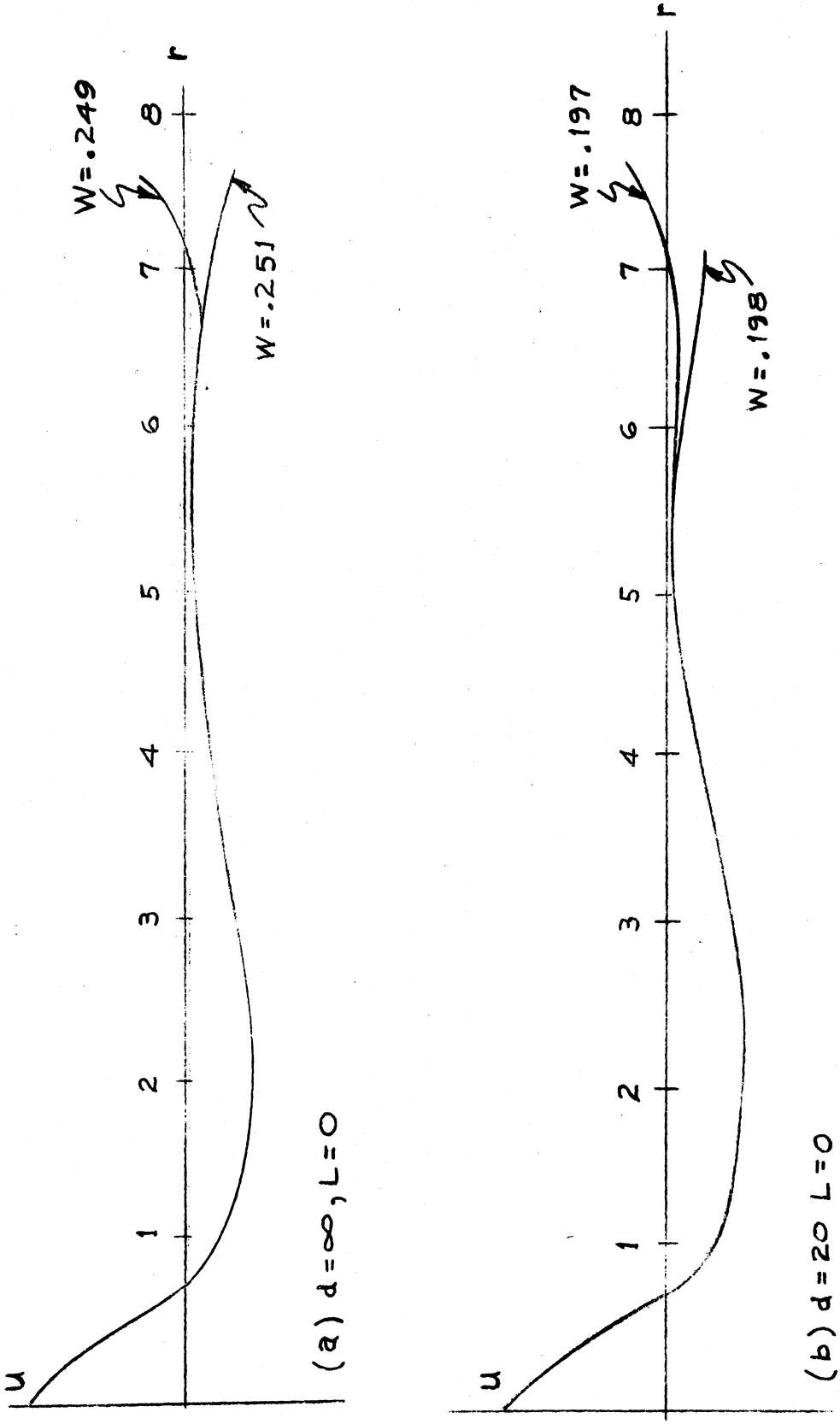


Fig. 4

"DDA Considerations in Combined Computation"*

By taking specific examples and comparing solution times of a 73 integrator DDA having an iteration rate of 623 iterations/sec with the IBM 704, the following generalizations can be made:

(a) For problems requiring 3 to 5 decimal places of accuracy and demanding full integrator capacity of the DDA, the incremental computer has a definite speed advantage. This advantage is further enhanced by the fact that the DDA can readout "on line" (i.e., there is no need to hang up computation processes while reading out results).

(b) Problems requiring greater than 7 decimal places of accuracy, and not large enough to demand full DDA integrator capacity are more rapidly solved on the general purpose computer.

Unfortunately no specific statements can be made regarding the vast "grey" areas which are not included in the above generalizations. The reason for this is that too much depends on the nature of the individual problem.

The east of programming of the DDA is a definite factor when deciding which type of digital computer to use for the solution of differential equations. Another important consideration is that convergence problems are not nearly as critical with the DDA due to the extremely simple integration processes involved. In addition to the above factors, the incremental computer updates all solutions at a high rate, making it desirable in certain control loops involving analog equipment.

From an external standpoint DDA behavior is in many respects similar to a low speed analog computer with sampled outputs. With this in mind many general purpose computer programming techniques developed for analog-digital computation can be applied to the GP-DDA combination. Furthermore, this likeness enables the DDA to aid in analog computer computations. Three promising applications of the DDA in conjunction with analog computers are nonlinear function generation, the performance of multiply operations, and simulation of sampled data systems.

The advantages pointed out are not intended to imply that the DDA has in any sense more over-all utility than analog or general purpose digital computers, but rather that it is a desirable partner in combined computation. Combined computation implies that the problems being solved are so large or complex in nature that they cannot be handled readily on a single computer. In that general purpose digital computers require solution times proportional to the size of the problem, it is seen that if it is possible to allocate portions of its computation to a DDA, an over-all time saving will result.

---

* Excerpt from "DDA Considerations in Combined Computation", by R.J. Leake, IBM Report No. 60-097-275, Oct. 1960, Federal Systems Division, Owego, New York.