The Programmed Digital Differential Analyzer

MASARU TAKATA*

This programmed digital differential analyzer described in the following is a simulation of an analogue computer on a digital machine and it is used for solving the initial value problems of ordinary differential equations. The user of the analogue computer may do without the knowledge about the numerical methods. It is necessary for him only to make the connections between the components used. There, the order of the arithmetic operations necessary for the computation in the case of the digital macnine does not matter at all. Whereas, it is the main task for the programmer of the digital computer.

In this programmed digital differential analyzer (abbreviated as DDA in the following), the programming is completed by only to make the connections as it is done in the case of an analogue computer. We suppose the pseudo components of the analogue computer which are represented by the corresponding numerical codes shown in the table. The functions $f_i(x, t)$ of the right hand side of the differential equations

$$\frac{dx^i}{dt}=f_i(x_1, x_2, \cdots, x_n, t) \quad i=1, 2, \cdots, n \tag{1}$$

are expressed by these codes without considering the order or sequense of the arithmetic operations. These three address type pseudo codes are then translated to the machine codes which are the object programme of the function subroutines of $f_i(x, t)$. The order of the arithmetic operations properly judged by the DDA system.

Since tte floating arithmetic is used, the scaling is unnecessary. The data such as the initial conditions are read in by the usual input routine. The numerical integration formulas used are the predictor-corrector type

$$\left.\begin{aligned} x_{n+1}&=x_{n-1}+2h\dot{x}_n\,, \\ x_{n+1}&=x_n+(h/2)(\dot{x}_n+\dot{x}_{n+1})\,, \quad h\equiv\Delta x \end{aligned}\right\} \tag{2}$$

which are programmed as the master routine and the linkage between the master and the function subroutine is automatically made by the system after the completion of the translation. Somewhat detailed descriptions are made in the following.

The machine used is JEIDAC-101 or NEAC-2203 which has 2000 words drum memory with a mean access time of 3 ms. Since one word is composed of 11 decimal digits with sign part, the pseudo code has the same number of digits, and it has the following form as shown in the Table 1,

* Faculty of Engg., Tokyo Univ., Tokyo

TABLE 1.

component	pseudo code	input b		input c		function
integrater	$\pm 0\, a_1a_2$	$b_0b_1b_2$	1	$c_0c_1c_2$	r	$\pm\int bcdt$
($a_0=0$)	$\pm 0\, a_1a_2$	$b_0b_1b_2$	0	0 0 0	r	$\pm\int bdt$
adder with multiplier	$\pm 1\, a_1a_2$	$b_0b_1b_2$	1	$c_0c_1c_2$	r	$\pm bc$
($a_0=1$)	$\pm 1\, a_1a_2$	$b_0b_1b_2$	0	0 0 0	r	$\pm b$
division operator ($a_0=2$)	$\pm 2\, a_1a_2$	$b_0b_1b_2$	1	$c_0c_1c_2$	r	$\pm b/c$
(undefined) ($a_0=3$)						undefined
function generator ($a_0=4$)	$+4\, a_1a_2$	$b_0b_1b_2$	1	$c_0c_1c_2$	r	implicit type function generator, finds the root x of $f(x)=0$
function generator ($a_0=5$)	$\pm 5\, a_1a_2$	$b_0b_1b_2$	0	0 0 f	r	explicit type function generator $\pm f(b)$, f: function No. (0, 1, 2, $\cdots$)
delay ($a_0=6$)	$+6\, a_1a_2$	$b_0b_1b_2$	1	$c_0c_1c_2$	r	$b(t-c\Delta t)$, $\sum(c)_i=300$
relay	$\pm 7\, 0\ r$	$b_0b_1b_2$	1	$c_0c_1c_2$	r'	output: b for $(r)\geqq 0$, : c for $(r)<0$
($a_0=7$)	$\pm 7\, 0\ r$	$b_0b_1b_2$	1	$c_0c_1c_2$	r'	output: b for $(r)=0$, (r, r': 1−6) : c for $(r)\neq 0$
independent variable ($a_0=800$)	$+8\, 0\, 0$	0 0 0	0	$c_0c_1c_2$	r	c: printing designation (printed per $c\Delta t$)
constants (potentiometer) ($a_0=8$)	$+8\, a_1a_2$	0 0 0	0	$c_0c_1c_2$	r	constants c: number of the constants $c\leq 50$
printer ($a_0=9$)	$+9\, a_1a_2$	$b_0b_1b_2$		$d_0d_1d_2$	d_3	print (b) a_1a_2: designation of the column d_0d_1: unmber of digits of integer part d_2d_3: total number of digits to be printed

$$\pm a_0a_1a_2\ b_0b_1b_2 \times c_0c_1c_2\ r$$

the first 3 digits $a_0a_1a_2$ (component a) represent the output component, the first of which a_0, shows the feature of the component and the second and third digits a_1a_2 its number. The digits $b_0b_1b_2$ and $c_0c_1c_2$ represent input components, the latter of which are 0 0 0 if the $\times$ part is 0. As one may easily understand the meaning for the case of $a_0=0, 1, 2, 5, 8$, the expression will be confined to the cases of $a_0=4, 7, 9$.

***a*$_0$=4.** The root of x for the equation $f(x)=0$ will be obtained by Newton-Raphson method, namely from the i-th aproximation $x^{(i)}$, the $i+1$-th one will be calculated by

$$x^{(i+1)}=x^{(i)}-f(x^{(i)})/f'(x^{(i)})\ .$$

The input b is the $f(x^{(i)}/f'(x^{(i)})$ above, whereas 8-code ($c_0=8$) is usually assigned to the input c as the initial estimation which is, subsequently

in the course of calculation, replaced by the value of x at the preceding time. The explanation of the translation of this code will be defered to later.

$A_0=7$. The output of the relay 70 r is either b or c depending upon the condition of the content of the component which has the same r as its tail with that of code 70 r. The number of the relays are limitted to 9.

$a_0=9$. By the column designation a_1a_2, the printed form appears as follows:

00 01 02 03 04 05
06 07·····10
.....

The digit following b must be $n-1$, when one wants of print n data following b. For example, by 901 000 3 $d_0d_1d_2d_3$ one has the following prindted form:

(003) (002) (001) (000)

When $d_0d_1d_2d_3=0000$, the printed numbers are in floating form used in the machine.

The only one restriction in the programming is to arrange the order of the pseudo codes in the ascending order of the a_0-part.

Example. $\dot{x}=yz$, $\dot{y}=-zx$, $\dot{z}=-\frac{1}{2}xy+z$. (Initial conditions are omitted here).

The integrators 00, 01 and 02 are assigned to z, y and x respectively of which outputs are all required in the form of fixed type with two digits of integer part and nine digits of total. To the constant $\frac{1}{2}$, the code 801 is assigned and the adder with multiplier is necessary for z. The programme is as shown in Table 2. The end letter '$0Z$' shows the end of the programme.

TABLE 2.

−000	100	1	801	0
000	000	0	000	0
−001	000	1	002	0
002	000	1	001	0
100	001	1	002	0
800	000	0	040	0
801	000	0	001	0
900	800	0	020	5
901	000	2	020	9
				0z

The numerical data such as initial values or constants etc., following these pseudo codes are read in by the initial routine.

The system of this DDA consists of three parts;

(1) the routine which reads the pseudo codes and constructs two tables necessary for the translation.

(2) the routine which finds the order of the

numerical procedure required for DDA codes and produces the object programme.

(3) the numerical integration routine.

The first one above reads the DDA codes and stores them in the appropriate memories, constructing the tables in which the number of the components appeard in the programme and the actual address assigned to each component is stored. These tables are referred at every time of the translation and are modified if necessary.

The translation is executed as follows. The processor reads the pseudo codes stored in the memory in reversed order from the one of the largest $a_0(\leq 7)$ to the code of the least $a_0(=0)$. If a code a which contains those inputs b_0 or c_0 as 0, 6 or 8, the pseudo codes consist of such a a are translated immediately and the code a is registered in the table, where all the translated codes a are memorized. The code, which has the iuput b or c other than 0, 6 or 8-code or registered one, is skipped over momentarily and the content of the unsolved code counter U is raised by one. If the content of the counter U is not zero when the pseudo code $a_0=0$ (integrator) is read at first time, the process repeated again from the largest a_0 which is not yet translated. Finishing the translation of all pseudo codes consisting of a_0 other than 0, then the translation of the 0-codes are executed. Finally the link orders to the master routine are planted in the top and tail of the object programme, and then the type routine is compiled.

The situation is not so simple as above for the case of 4-code. When the 4-code appears, its input part b, namely $f(x^{(i)})/f'(x^{(i)})$, may not be obtained untill the output 4-code itself is calculated, since the 4-code namely x, is included in the calculation of $f(x)$ or $f'(x)$. The processor finds its existence by recognising the fact that the content of the counter U of the present sweep is the same as that of the preceding one. Then the first 4-code is compiled by the following procedure. The processor produces at first such a programme that transfers the content of the memory, corresponding to 8-code written in the c-part of that 4-code, to the memory assigned to x, the content of that 4-code, as its initial guess. Memorizing the address of the order of the object programme now produced, which is just preceding to the upper end of the loop calculation x, the processor translates the pseudo codes now solvable by adding that 4-code to the list of the already translated codes. After the completion of the programme calculation f/f', machine codes for the convergence test are produced. This procedure is repeated enough, if more 4-codes are there.

The number of steps of this processor is about 1200, excluding 430 steps for the master (integration) and the printing routines. The time

required to read this programme is about one and a half minutes by the photo-tape reader. As an example of the processing time we may quote the one which required one minute and thirty five seconds to produce about 240 steps of the object programme.

This system was successively used for the problems of chemical reactions or the simulation of the control processes. This is also applicable for the calculation of the algebraic expressions which is required to be listed in tabular form for the equidistant arguments.