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 machine 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

\[
\frac{dx_i}{dt} = f_i(x_1, x_2, \cdots, x_n, t) \quad i = 1, 2, \cdots, n
\]

(1)

are expressed by these codes without considering the order or sequence 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 the 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

\[
\begin{align*}
x_{n+1} &= x_n + 2h \dot{x}_n, \\
x_{n+1} &= x_n + \left( \frac{h}{2} \right) (\dot{x}_n + \dot{x}_{n+1}), \quad h \equiv dx
\end{align*}
\]

(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>
<thead>
<tr>
<th>component</th>
<th>pseudo code</th>
<th>function</th>
</tr>
</thead>
<tbody>
<tr>
<td>integrator ((a_0=0))</td>
<td>\pm 0 a_0 a_2 b_0 b_1 b_2 1 e_0 e_1 e_2 \ r \ \pm \int b dt \</td>
<td></td>
</tr>
<tr>
<td>adder with multiplier ((a_0=1))</td>
<td>\pm 1 a_0 a_2 b_0 b_1 b_2 1 e_0 e_1 e_2 \ r \ \pm b e</td>
<td></td>
</tr>
<tr>
<td>division operator ((a_0=2))</td>
<td>\pm 2 a_0 a_2 b_0 b_1 b_2 1 e_0 e_1 e_2 \ r \ \pm b / c</td>
<td></td>
</tr>
<tr>
<td>function generator ((a_0=3))</td>
<td>4 a_0 a_2 b_0 b_1 b_2 1 e_0 e_1 e_2 \ r \ implicit type function generator, finds the root (x) of (f(x)=0)</td>
<td></td>
</tr>
<tr>
<td>function generator ((a_0=5))</td>
<td>5 a_0 a_2 b_0 b_1 b_2 0 0.0 f \ r \ explicit type function generator (\pm f(b)), (f): function No. (0, 1, 2, \ldots)</td>
<td></td>
</tr>
<tr>
<td>delay ((a_0=6))</td>
<td>6 a_0 a_2 b_0 b_1 b_2 1 e_0 e_1 e_2 \ r \ \beta(i-\ \delta t), \ \sum(c_i)=300</td>
<td></td>
</tr>
<tr>
<td>relay ((a_0=7))</td>
<td>7 0 r b_0 b_1 b_2 1 e_0 e_1 e_2 \ r' \ output: b \ for (r) \geq 0; \ c \ for (r) &lt; 0</td>
<td></td>
</tr>
<tr>
<td>independent variable ((a_0=800))</td>
<td>8 00 0 0 0 0 \ r \ c: printing designation (printed per (\delta t)</td>
<td></td>
</tr>
<tr>
<td>constants (potentiometer) ((a_0=9))</td>
<td>9 a_0 a_2 b_0 b_1 b_2 d_0 d_1 d_2 \ r \ constants c: number of the constants (c \leq 50)</td>
<td></td>
</tr>
</tbody>
</table>

\[ \pm a_0 a_2 b_0 b_1 \times c_0 e_1 e_2 \ r \]

the first 3 digits \(a_0 a_2\) (component a) represent the output component, the first of which \(a_0\), shows the feature of the component and the second and third digits \(a_2\) its number. The digits \(b_0 b_1\) and \(c_0 e_1 e_2\) represent input components, the latter of which are 000 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 approximation \(x^{(i)}\), the \(i+1\)-th one will be calculated by

\[
x^{(i+1)} = x^{(i)} - \frac{f(x^{(i)})}{f'(x^{(i)})}.
\]

The input \(b\) is the \(f(x^{(i)})/f'(x^{(i)})\) above, whereas 8-code \((a_0=8)\) 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 deferred to later.

**A.** 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 limited to 9.

**a.** By the column designation \( a_4a_2 \), the printed form appears as follows:

\[
\begin{array}{cccccc}
00 & 01 & 02 & 03 & 04 & 05 \\
06 & 07 & \cdots & \cdots & \cdots & 10 \\
\end{array}
\]

The digit following \( b \) must be \( n-1 \), when one wants of print \( n \) data following \( b \). For example, by 001 000 3 \( d_1d_2d_3 \) one has the following printed form:

\[
(003) (002) (001) (000)
\]

When \( d_1d_2d_3 = 0000 \), 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 \( a_3 \)-part.

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

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>
<thead>
<tr>
<th>( -000 )</th>
<th>100</th>
<th>1</th>
<th>801</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>000</td>
<td>000</td>
<td>0</td>
<td>000</td>
<td>0</td>
</tr>
<tr>
<td>001</td>
<td>000</td>
<td>1</td>
<td>002</td>
<td>0</td>
</tr>
<tr>
<td>002</td>
<td>000</td>
<td>1</td>
<td>001</td>
<td>0</td>
</tr>
<tr>
<td>100</td>
<td>001</td>
<td>1</td>
<td>002</td>
<td>0</td>
</tr>
<tr>
<td>800</td>
<td>000</td>
<td>0</td>
<td>040</td>
<td>0</td>
</tr>
<tr>
<td>801</td>
<td>000</td>
<td>0</td>
<td>001</td>
<td>0</td>
</tr>
<tr>
<td>900</td>
<td>800</td>
<td>0</td>
<td>020</td>
<td>5</td>
</tr>
<tr>
<td>901</td>
<td>000</td>
<td>2</td>
<td>020</td>
<td>9</td>
</tr>
</tbody>
</table>

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 appeared 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 \(a_n(\leq 7)\) to the code of the least \(a_0(=0)\). If a code \(a\) which contains those inputs \(b\) or \(c\) 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 input \(b\) 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 \(a_n=0\) (integrator) is read at first time, the process repeated again from the largest \(a_n\) which is not yet translated. Finishing the translation of all pseudo codes consisting of \(a_n\) 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 until 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.