TA 165
.M41
.I591
no.E-
364

Engineering Memorandum E-364

# A PROGRAM FOR TRANSLATION OF MATHEMATICAL EQUATIONS FOR WHIRLWIND I

by

J. H. Laning Jr. and N. Zierler

January, 1954

INSTRUMENTATION LABORATORY

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

Cambridge 39, Mass.

Engineering Memorandum E-364

# A PROGRAM FOR TRANSLATION of MATHEMATICAL EQUATIONS FOR WHIRLWIND I

by

J. H. Laning Jr. and N. Zierler

January, 1954

INSTRUMENTATION LABORATORY
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

J. H. Laning Jr.
J. H. Laning Jr.

N. Zierler
N. Zierler

✓

T-J5
I 60e

## ACKNOWLEDGEMENT

This report was prepared under the auspices of DIC Project 7139 sponsored by the Armament Laboratory of the Wright Air Development Center, through USAF Contract AF33(616)-2039.

The Publication of this report does not constitute approval by the Air Force of the findings or the conclusions contained therein. It is published only for the exchange and stimulation of ideas.

## ABSTRACT

An interpretive program for Whirlwind I is described that will accept algebraic equations, differential equations, etc. expressed on Flexowriter punched paper tape in ordinary mathematical notation(within certain limits imposed principally by the Flexowriter) as input and automatically provide the desired solutions.

INDEX

## 1. Introduction

Nearly everyone who has had a problem to solve on a large-scale digital computing machine has probably felt that it would be indeed convenient if one could give the machine his problem in ordinary mathematical language with perhaps a suggestion for a method of solution. We have been motivated by a sympathy for this feeling in writing a program for Whirlwind I that will constitute, we hope, a useful step in the indicated direction. The effect of our program is to create a computer within a computer and the purpose of the present report is to describe this computer and provide a programmer's manual for its use.

The input to the computer is in the form of a paper tape punched on a Flexowriter, which is a specially designed typewriter that produces a coded pattern of holes in the tape for each character (including backspace, carriage return, etc.) that is typed. The problem to be solved is typed on the Flexowriter in mathematical form using the conventions and notations discussed below. As the resulting tape is fed to the computer the coded characters are translated by means of our program into a set of standard Whirlwind instructions. After the translation is completed, the computer carries out the indicated instructions to solve the problem, presenting the answers requested by the programmer in typewritten form. Examples of typical problems appear throughout the report.

We have attempted to stay as close as possible to ordinary mathematical notation; however, because of the limitations of the Flexowriter and of our program itself, certain typographical conventions must be observed. Programming rules are discussed throughout the report and are summarized in a later section.

## 2. Basic Operations

All of the lower case letters are available as variables, and equations of the following form are used:

$$x = \quad ,$$

Here x stands for any lower case letter, and the blank between the equals sign and the comma stands for a mathematical expression involving lower case letters, numbers, plus and minus signs with their usual significance, parentheses, and slanting slashes / indicating division. Examples of such equations are:

$$\begin{aligned} a &= 5, \\ y &= -6.3a, \\ b &= 0.0053\,(a-y)\,/2ay, \end{aligned} \qquad (1)$$

The computer regards the right hand side of an equation as a set of computational operations to be performed. The result of the series of computations indicated by the right hand side of a single equation is always a number and after the computation is finished this number is stored in the registers assigned to the letter that appears on the left hand side of the equation. Equations are carried out in the order in which they are written. Thus, in the above example the value 5 is first stored in the space in storage assigned to the variable a; it is then called forth in the second equation as the number stored in the registers assigned to a, multiplied by -6.3, and the result is stored in the space in storage assigned to the variable y. The computer then proceeds to compute the right hand side of the third equation, making use of the numbers stored in the registers assigned to the symbols a and y, and store the resulting number in the space assigned to the variable b. Because of this modus operandi such equations as

$$\begin{aligned} n &= n + 2, \\ w &= -\,w, \end{aligned} \qquad (2)$$

are meaningful. The first of these adds the number 2 to the contents of the storage assigned to the variable n, subsequently storing the result in the same location, and thereby causes the current value of n to be increased by 2; the second causes the current value of w to be replaced by its negative.

It should be observed that a letter variable must appear on the left hand side of an equation so that it may be assigned a definite value before it appears on the right hand side of an equation. This principle is illustrated in the set of equations (1) where the variable a is first set to the value 5, y is then assigned a value depending on a numerical constant and the current value of a, and so forth. On the other hand a program beginning

$$\begin{aligned} n &= m - 1, \\ m &= 1, \end{aligned} \tag{3}$$

would be incorrect since m would have no preassigned value when the computer attempted to find the value of n. The meaning of this restriction is, of course, that methods of solution must be explicitly stated in general.

Basically, the computer performs the elementary operations of addition, subtraction, multiplication and division. Certain conventions of a typographical or interpretational nature should be mentioned in connection with these operations.

## A. Parentheses

Only standard parentheses are available typographically. No braces or square brackets may be used. No more than four parentheses may be open in a single equation at any one time, thus,

$$x = a(b + c(d + e(f + g(h + i)(j + k)^2)^{-1})), \tag{4}$$

is allowed while

$$x = a(b + c(d + e(f + g(h + i(j + k)^2))^3)), \tag{5}$$

is not. Stated in another way, the number of left parentheses "(" minus the number of right parentheses ")" lying to the left of any point of the equation must not exceed four; otherwise use of parentheses is unrestricted.

## B. Division

A little reflection will show that an expression such as

$$a + 2b/c + d \tag{6}$$

is mathematically ambiguous. The computer of course, assigns a particular interpretation to such an expression. The rules by which it does so can be most conveniently explained as follows: a plus or minus sign has the effect of separating any mathematical expression into two terms to the left and the right of it. This separation with respect to a plus or minus sign occuring within parentheses only applies, of course, to the total quantity within the parentheses. Thus, if an expression such as that above is written without use of parentheses, the plus and minus signs separate the expression into separate terms. In the case cited, these terms are the quantity a, the quantity 2b divided by c, and the quantity d. If in the above example, one had wished that the result should be the sum of a and the quantity 2b divided by (c + d), one would have had to include the quantity c + d within parentheses:

$$a + 2b/(c + d) \tag{7}$$

If, on the other hand, one had wished to consider this quantity as the ratio of the sum a + 2b to the sum c + d, then both of these quantities should have been included in the parentheses.

$$(a + 2b)/(c + d) \tag{8}$$

If it is merely kept in mind that pluses and minuses act as separating symbols, no confusion can arise. As a further example, one can write an expression such as

$$cd/a(b + c) + d \tag{9}$$

The computer interprets this expression as the sum of the quantity d and the ratio whose numerator is the product cd and whose denominator is the product of a with the quantity b + c.

One more form of mathematical expression involving division is worthy of comment. The use of more than one division within a single mathematical term is admissible and is correctly interpreted by the computer. Thus a/b/c is properly interpreted as the product ac divided by b. On the other hand the expression

$$a/(b + c/d) \tag{10}$$

is interpreted as the ratio of a to the quantity which is formed by adding to b the ratio c over d. In the latter case, the quantity within the parentheses is itself interpreted as a single mathematical expression to be evaluated by the computer. In effect, an expression contained within parentheses is isolated from the rest of the equation and the normal rules of multiplication and division apply without regard to the remainder of the equation.

### C. Numerical Quantities

Although it has been assumed implicitly in writing the above material it is worth stating explicitly that the computer correctly interprets numbers in ordinary decimal notation. No standard form is assumed for numerical quantities by the computer. Thus it interprets correctly such quantities as 0.012, 012, .012, 48, 321 etc., without ambiguity.

### D. Exponents

Upper case numbers on the Flexowriter appear as exponents; there is an upper case minus sign but no upper case plus sign. The computer will handle positive (unsigned) and negative integral exponents. Some examples are:

$$\begin{aligned} a &= 5^{2}, \\ b &= (a - 2)^{-2}, \\ c &= (ab^{3})^{-5}, \\ d &= a(b + c)^{2}/(c - a)a^{-3}, \end{aligned}$$

## 3. The Number System

Numbers are stored and arithmetic operations are carried out by the computer in the 24, 6 number system. In this system numbers appear in the machine essentially in the form $a \cdot 2^{b}$ where a is a signed 24 place binary number with $1/2 \leq a < 1$ or $a = 0$ and b is a signed 6 place binary number. Thus, it is useless to supply more than 8 significant decimal figures, although in general no difficulty is caused by supplying more - roundoff occurs automatically. If numbers, including, of course, numbers produced as the result of computations, exceed $2^{63} - 2^{39}$(or, roughly, $9.2233995 \cdot 10^{18}$) in absolute value, an alarm will result. Any integer up to the preceding number can be written explicitly: e.g., 9223314040000000000 is a permissible number. The same number could also be written $9.22331404 \cdot 10^{18}$. It should be noted, however, that the computer first reads in a number as an integer, placing the decimal point only after the number is completely read in. This means that no number is allowed which, without benefit of decimal point, would be interpreted as an integer exceeding $2^{63} - 2^{39}$. Otherwise, numbers may be written in quite arbitrary decimal form as has been illustrated to some extent in the preceding examples.

## 4. Output

The delayed printer is used for output. The current value of any number of letter variables may be printed out by inserting the instruction PRINT followed by the desired letters, followed by a period. No computation may be performed during a print instruction (e.g., PRINT 3a is not admissible) and commas may, but need not be, inserted between letters. Thus, PRINT x,y. and PRINT xy. would both cause the current values of x and y to be printed out. The form of the print out is best explained by examples. If a = 5 currently the instruction PRINT a. would cause a = +.50000000 + 01 to be printed out. The second number; i.e., the +01, is interpreted as the power of 10 to be multiplied by the first number. Thus, if a = .05 we should have a = +.50000000-01 and if a = - $\pi$ we should have a = -.31415927 + 01. When a PRINT instruction is performed the first and last characters to appear as output are carriage returns. If more than 6 variables are called for on a single PRINT instruction, 6 will be placed on each line, except possibly for the last, of course, and there will be no line space between lines; for example, the instruction PRINT a, b, c, d, e, f, g, h. will produce:

```
a = +. .        b = +. .        c = +. .        d = +. .        e = +. .      f = +. .

g = +. .        h = +. .        i = +. .
```

while the instructions PRINT a, b, c. PRINT d, e, f. PRINT g, h, i. will produce

```
            a = +. .        b = +. .        c = +. .

            d =  +. .       e = +. .        f = +. .

            g = +. .        h = +. .        i = +. .
```

where the dots stand for appropriate numbers.

## 5. <u>The Instructions SP, CP, SR, and CR</u>

It has been stated that equations are carried out in the order in which they are written. It is clearly desirable to be able to interrupt this order from time to time. We may wish to make the order in which the operations are performed depend on the result of a previous computation. For this purpose, or for purposes of identification in general, equations may be numbered with decimal integers between 1 and 100 inclusive. Thus we may write

```
3  x = 2,

1  y = 0.5 (x + 2/x),
```

The first equation is then identified as equation number 3, the second as equation number 1. The equation number always precedes the equation to which it refers.

The effect of the instruction SP followed by an integer from 1 to 100 and a comma is to "transfer control" to the equation with the specified number and to proceed on from that equation. For example, SP 3, would cause equation 3 to be executed next and then the equation or instruction following equation 3 in the order of writing.

In the following we use the number 5 for purposes of illustration; it is understood that any number that has been assigned to an equation may be used. The effect of the instruction CP 5, is the same as that of SP 5, if the last computed quantity is negative while CP 5, is ignored if the last computed quantity is positive.

For example, if n has the value 3 then in the following program

```
  e = n - 5,
  CP 5,
  x = 3 ab
 5y = ax,
```

when the instruction CP 5, is reached the equation x = 3 ab, will be skipped since the quantity last computed before the CP 5, was e = n - 5 = 3 - 5 = -2. If n had the value 5 in the above example then e = n - 5 = 0. In this case the CP may or may not behave like an SP since 0 may appear as +0 or -0. This ambiguity need cause no difficulty in general. If, for example, it is desired to go thru a certain process starting with equation 1 exactly 10 times we may write:

```
        n = 0,
  1     .
        .
        .
        .
        .
        n = n + 1,
        e = n - 9.5,
        CP 1,
```

The quantity n is thus used as a counter, and e will then have the successive values -8.5, -7.5, -6.5, . . . , - 0.5, +0.5 and we obtain the desired result. Further examples will be given below.

The instruction SR 5, causes equation 5 to be executed next and then, after equation 5, the equation or instruction following the SR 5,. The instruction CR 5, bears the same relation to SR 5, that CP 5, does to SP 5,.

An SP, CP, SR, or CR may not itself be assigned an equation number.

## 6. STOP

The instruction STOP must be placed at the end of every program. Its effect is to stop the computer. This instruction may be given an equation number so that an SP or CP may be used to stop the computation.

## 7. The Stop Character

Flexowriter character number 49 is the "stop code" or "stop character". This character is used by our program in the following way. It may happen, for example, that a fairly large amount of data will be processed in several ways at several different times. Rather than duplicate the long data tape onto the processing program tape in each instance one may simply place a stop character at the end of the data tape. The effect will be to stop the computer temporarily to allow a program tape to be inserted into the tape reader. Computation will then be resumed when the WWI restart button is pushed. Other situations in which this feature may be useful will undoubtedly occur to programmers.

## 8. Ignored Characters

Except when they occur between the digits of a number the following characters are ignored:, space, back space, tabulate and carriage return. Any of these characters or any combination of them, inserted between digits is interpreted as a multiplication sign. Thus x = 23, (i.e., no characters between the 2 and the 3) would set x to the value twenty-three while x = 2 3, (i.e., 2 followed by space followed by 3) would cause x to assume the value 6.

The nullify character is always ignored* so that if a space or other erroneous character is accidently inserted between digits an error may be avoided by simply nullifying the character.

It should be pointed out that superfluous characters are not always apparent from a print. If the combination "space, backspace" or the combination "shift to upper case, shift to lower case" is inserted between digits, the combination, in either case would be interpreted as a multiplication sign. Error may again be avoided in these cases by nullifying the superfluous characters.

## 9. The Character $\cdot$ Denoting Multiplication

The lower dot of the colon has been filed off on at least one of the Flexowriter units available at DCL. This character, or the colon itself, may be used to denote multiplication; e.g., $5\cdot 10^4$ has the value 50,000 as has also $5\ 10^4$ where only one or more spaces or tabulates is inserted between the "5" and the "$10^4$".

*Except in an SP, SR, CP, CR, or STOP instruction. If a typographical error occurs in these instructions, the whole instruction must be nullified.

## 10. Examples

Suppose it is desired to print a table of values of cosine x for x = 0, .1, .2, . . . , 1 radian and that we decide to use up to the $x^{10}$ term in the power series for cosine x. This could be accomplished by the following program:

```
        x = 0,
1       z = 1 - x^2/2 + x^4/2·3·4 - x^6/2·3·4·5·6
            + x^8/2·3·4·5·6·7·8 - x^10/ 2·3·4·5·6·7·8·9·10,
        PRINT x, z.
        x = x + .1,
        e = x - 1.05,
        CP 1,
        STOP
```

We might prefer an iterative scheme to produce the same result. The following scheme has the advantage that the number of terms in the series may be adjusted by changing a single number:

```
        x = 0,
1       y = 10,
        z = 1,
2       z = 1 - z x^2/y (y - 1),
        y = y - 2,
        e = 1 - y,
        CP 2,
        PRINT x, z.
        x = x + 0.1,
        a = x - 1.05,
        CP 1,
        STOP
```

## 11. Function Subroutines

It is clearly desirable to build into the computer subroutines that compute automatically certain frequently used functions. This process has been begun for our computer and at present the following functions are available:

| Designation | Function |
|---|---|
| $F^1$ | square root |
| $F^2$ | sine |
| $F^3$ | cosine |
| $F^4$ | tangent |
| $F^5$ | $\text{sine}^{-1}$ |
| $F^6$ | $\text{cosine}^{-1}$ |
| $F^7$ | $\text{tangent}^{-1}$ |
| $F^8$ | secant |
| $F^9$ | cosecant |
| $F^{10}$ | cotangent |
| $F^{11}$ | absolute value |
| $F^{12}$ | signum |
| $F^{13}$ | exponential ($e^x$) |
| $F^{14}$ | negative exponential ($e^{-x}$) |
| $F^{15}$ | sinh |
| $F^{16}$ | cosh |
| $F^{17}$ | tanh |
| $F^{18}$ | sech |
| $F^{19}$ | csch |
| $F^{20}$ | coth |
| $F^{21}$ | $\log_{10}$ |
| $F^{22}$ | $\log_2$ |
| $F^{23}$ | $\log_e$ |

The sine of x may be obtained by writing simply $F^2(x)$ and such an expression may be treated arithmetically in the same way as a variable in an equation, for example:

$$x = 2(F^1(y)\ F^2(z) + F^{11}(z)),$$

would set x equal to 2( $\sqrt{y}$ sine z + |z| ). Subject to the usual restriction on the use of multiple parentheses ( no more than 4 open at a time), computation may be carried out inside the parentheses containing the argument of a function. For example,

$$x = F^3 (x + 2/y(a + b) \; z^{-1} ),$$

would set x equal to cosine (x + 2 z/y(a + b)). Further, functions of functions are permitted. For example,

$$x = F^4(y + F^6(z) ),$$

would set x equal to tan $(y + \cos^{-1} z)$ or

$$x = F^1 (F^{11}(a) ),$$

would set x equal to $\sqrt{|a|}$ .

## 12. Additional Variables and Variable Indices

Subscript numbers are not available on the standard Flexowriter so additional symbols for variables are obtained as follows: Any lower case letter followed by a vertical slash followed by an upper case integer between 0 and 1023 inclusive may be used as a variable in precisely the same way the lower case letters alone have been used above. We shall call such variables "superscript variables." Thus $x|^5 = 2n$, $a|^2 = a|^7 + b|^6$, PRINT a, $x|^3$, c, $d|^{75}$. are legitimate expressions. Any lower case letter may be used as a variable index for superscript variables in the combination lower case letter followed by vertical slash, followed by lower case letter. For example, when the expression x|n is encountered, the current value of n is taken as the index: if n = 6 currently then x|n is at this time treated as $x|^6$.

If, in the expression x|n, n has a non-integral value greater than 0.5, the computer rounds n off to the nearest integer. Negative values of n are not permitted in this connection, nor are values between zero and one half inclusive, with the exception that the value zero is permissible if assigned by the specific equation n = 0, .

The roundoff property can often be used to advantage; e. g. , in interpolation in a set of tabular data. Suppose that we are given 10 values $f(x_1)$, $f(x_2), \ldots,$ $f(x_{10})$ of a function at a set of equally spaced points $x_1$, $x_2, \ldots, x_{10}$. The tabular values can be read in by the equations

```
$f\big|^{1} =$            ,
$f\big|^{2} =$            ,
$\cdots\cdots\cdots\cdots$
$f\big|^{10} =$           ,
```

We also require, for the purpose, the set of integers 1 to 10, read in as follows:

```
        $n = 0,$
1       $n = n + 1,$
        $k\big|n = n,$
        $e = n - 9.5,$
        CP 1,
```

setting $k\big|^{1} = 1$, $k\big|^{2} = 2$, etc.

To find $f(x)$, $(x_1 < x < x_{10})$, by linear interpolation, the following program is used ($h$ = the difference between $x_k$ and $x_{k+1}$):

```
$j = 0.5 + (x - x\big|^{1}) / h,$
$m = k\big|j,$
$n = m + 1,$
$d = j + 0.5 - m,$
$y = d\, f\big|n + (1 - d)\, f\big|m,$
```

giving $y$ as the desired result.

## Printing Superscript Variables

The general format of the print-out of superscript variables is the same as for ordinary variables. The instruction PRINT $y\big|^{5}$. will cause

$$y\big|^{5} = \ldots$$

to be printed out while the instruction PRINT $b\big|i$. will cause

$$b\big|^{3} = \ldots$$

to be printed if 3 is the current value of i.

13. SP n, etc..

It is already possible to change conditionally the order in which operations are performed by using the CP instruction. It is also possible to program such a change by using the following instruction: SP followed by any lower case letter followed by a comma. If, for example, the variable u has the value 15, when the instruction SP u, is reached by the control, then this instruction will have the effect of SP 15,. The instructions CP, SR and CR may similarly be used with lower case letters. For non-integer values of the index, the same restrictions and round-off properties hold as for variable superscript indices.

14. Variables Assigned to the Drum

Superscript variables are ordinarily assigned to MS (magnetic core storage) and there is room for a total of 250 such variables. If additional variables are desired they may be obtained in two ways. The instruction ASSIGN followed by a lower case letter followed by a vertical slash followed by an upper case integer between 1 and 1023 inclusive followed by a comma causes space on the drum to be selected for variables. For example, ASSIGN $a|^{1000}$, automatically reserves space for variables $a|^{1}$ through $a|^{1000}$ inclusive to the drum and takes up in the storage table for variables in MS the space of only one ordinary superscript variable. Once this assignment has been made, the superscript variables in question may be treated as ordinary superscript variables.

Special provision has been made for assigning tables of values to the drum. Suppose, for example, that we wish to store the values of a function g at the values $t_1$, $t_2$, $t_3$, $t_4$ of its argument. Let these respective values be assigned the symbols $g|^{1}$, $g|^{2}$, $g|^{3}$, $g|^{4}$ and suppose, for definiteness, that $g|^{1} = 2$, $g|^{2} = 4$, $g|^{3} = 6$, $g|^{4} = 0$. We already have one method of assigning these variables to the drum:

ASSIGN $g\Big|^{4}$,

$g|^{1} = 2,\ g|^{2} = 4,\ g|^{3} = 6,\ g|^{4} = 0,$

Exactly the same thing may be accomplished by writing

$g|N = 2, 4, 6, 0,$

Any lower case letter may be used in this way but only the upper case letter N may follow the vertical slash. Each value, including the last, must be followed by a comma.

Once variables have been assigned to the drum in this fashion, they may be treated as ordinary variables in exactly the same way as variables assigned to the drum (but not given initial values) by the instruction ASSIGN.

The $a\Big|N$ type of instruction may be used in another way to assign variables (with their initial values) to the drum. Suppose that one wished to set $a|^{1} = 1$, $a|2 = 1.5$, $a|^{3} = 2$, $a|^{4} = 2.25$, $a|^{5} = 2.5$, $a|^{6} = 3.5$, $a|^{7} = 4.5$,. This may be done by writing

$a|N = 1.0\ (.5)\ 2.\ (0.25)\ 2.5\ (1)\ 4.5,$

That is, one writes after the equals sign, the value of $a|^{1}$, then the increment in parentheses, then the value up to which one is to go using this increment, and so on, terminating the instruction with a comma. It is understood that the "a" in the above example may be replaced by any lower case letter but only N may follow the slash.

As another example consider the little routine given in the discussion of interpolation in section 13 for assigning $k|^{1} = 1$, $k|^{2} = 2, \ldots, k|^{10} = 10$. This could be replaced by

$k|N = 1\ (1)\ 10,$

It should be noted that the operations multiplication, addition, etc., involving variables on the drum will in general take about 30 milliseconds longer per variable on the drum than the same operations involving MS variables. If variables on the drum are used sufficiently often in a program to make a significant difference in the computer time used, and if there is space for the variables in MS then the variables should be assigned, or re-assigned to MS.

For example, suppose one wishes to introduce 100 values $f|^1$ thru $f|^{100}$ of a function. If they are assigned by writing out

```
f|^1 = ...,
f|^2 = ...,
..........,
f|^100 = ...,
```

then 100 unnumbered equations, the maximum, have been used. Hence we might assign the values to the drum by

```
y|N = ...,...,...,
```

and then re-assign them to MS (if they were to be used sufficiently often to make it worth the trouble) by writing something like

```
            n = 0,
1           n = n + 1,
          f|n = y|n,
            e = n - 99.5,
            CP 1,
```

It is difficult to say in advance just how many variables may be assigned to the drum for a particular problem. There is a maximum of 3072 (which may be increased at a later date) but the drum is shared by the equations and the drum variables of a program so that in no case could 3072 drum variables actually be used. In any case, if storage is exceeded the automatic post mortem (described below) will inform the programmer where and how the trouble arose.

## 15. Differential Equations

Provision has been made for the solution of ordinary differential equations by the method of Gill - a variation of the 4th order Runge-Kutta technique. In order to make use of this provision, certain conventions must be observed.

1) the letter t must be used for the independent variable.

2) the letter h must be used for the increment in t.

3) the upper case letter D is used to denote d/dt.

Any ordinary or superscript variable (other than t and h) may be used for the dependent variables and for auxiliary variables. A given nth order system must be reduced to a system of n first order differential equations. Suppose then, this reduction having been done, we let $y_1, y_2, .., y_n$ denote the n dependent variables. Our system is of the form:

$$dy_i/dt = f_i(t, y_1, y_2, \ldots, y_n), \quad i = 1, 2, .., n.$$

The way in which the solution is to be accomplished is perhaps best explained by an example. Suppose that there are two differential equations, that we take 2 as the initial value for t and that h = 0.5. Suppose further, that we wish to print out the solutions for t = 2, 3, 4, up to 10, that the initial values of the $y_i$ are 0, and that

$$f_1(t, y_1, y_2) = y_2+1, \quad f_2(t, y_1, y_2) = -y_1,$$

Our program may then be as follows:

```
           t = 2,
           h = 0.5,
          y|^1 = 0,
          y|^2 = 0,
1         PRINT  t, y|^1, y|^2.
           e = 9.7 - t,
           CP 3,
2          D y|^1 = y|^2 + 1,
           D y|^2 = - y|^1,
           e = -e,
           CP 2,
           SP 1,
3          STOP
```

It should be observed that t is automatically increased by the value h upon completion of the last equation that starts with a D. Of course, any or all of the values of t, h, and the dependent variables may be reset during the computation. One important restriction on the form of a program should be noted. The carrying forward of the solution a single step of length h actually involves going through the part of the program that starts with the first D equation and ends with the last D equation a total of 4 times. These 4 sub-steps take place automatically but it is essential that all relevant auxiliary comput ation take place between the first and last D equations. Suppose,for example, that we were to change the part of the preceding program between equations 1 and 2 as follows:

$e = 9.7 - t$,
CP 3,
2 $a = y|^{2} + 1$,
$D\,y|^{1} = a$,

This change would appear to be innocuous since "a" seems to be appropriately reset before each entry into the differential equations. However, upon completion of the second D equation in each of the first 3 sub-steps, computation would be resumed with the first D equation ( and not equation 2 ) so that "a" would not have the current value of $y|^{2} + 1$.

Finally, it has been stated that a total of 250 MS superscript variables may be used; this is not true for programs involving differential equations. If there are k equations that begin with a D in a program then 250 - 2k MS superscript variables may be used.

## 16. Post Mortems

Certain automatic post-mortem features have been included but this part of the computer is essentially still in the design stage. Roughly speaking, if a program exceeds storage in any one of several possible ways the computer will tell us where and how storage was exceeded.

If an arithmetic alarm occurs, the programmer will be told the number of the equation in which the alarm occured and the number of the equation that preceded the alarm. It should be noted here that equations not assigned numbers by the programmer are automatically assigned numbers from 101 to 200 by the computer in the order in which they are written.

The programmer may obtain the values of any variables he desires after an arithmetic alarm by writing the appropriate PRINT instruction as equation 100.

## 17. Rules

1. Every equation must end with a comma.
2. The nullify character is always ignored except when it occurs between the S and P of SP (similarly for SR, CR, CP, STOP).
3. Except when they occur between the digits of a number or in SP, etc., the following characters are ignored: space, back space, tabulate and carriage return. Any of these characters inserted between numbers is interpreted as a multiplication sign.
4. Every PRINT instruction must end with a period.
5. No more than 4 open parentheses may be written in a single equation before one is closed.
6. The instructions SP, CP, SR, CR must be terminated by a comma and <u>may not be numbered.</u>
7. PRINT instructions may be numbered.
8. The instruction STOP may be numbered and may appear only at the end of a program.
9. The instruction ASSIGN must be terminated by a comma.
10. Each assignment of a group of variables to the drum (by either the ASSIGN or the $x|N$ type of instruction) uses up the space ordinarily occupied by one superscript variable in the table for superscript variables in MS. With this understanding a total of 250 superscript variables may be assigned to MS.
11. A maximum of 100 numbered equations and 100 unnumbered equations may be used in any one problem.

## 18. Temporary Rules and Restrictions

As of the date at which this report is written the following restrictions are in effect. It is expected that they will be removed in the immediate future.

1. An SR or CR instruction may not refer to an equation that makes use of a subroutine.
2. The $x|N$ is not yet in operation.

All other portions of the program are believed to be operating satisfactorily but exhaustive tests have not been made. Information relative to any troubles encountered in the use of this program would be appreciated by the authors.

## INITIAL DISTRIBUTION

Commander
Tactical Air Command
Langley Air Force Base, Va.
Attn: Command Adjutant

Air University Library
Maxwell Air Force Base
Maxwell, Ala.
Attn: Req. No. CR-3250

Bureau of Aeronautics
Room 1W82
Navy Department
Washington 25, D.C.
Attn: Aer EL-71

Commander
Operation Development Force
U.S. Atlantic Fleet
U.S. Naval Base
Norfolk 11, Va.
Attn: L.P. Dunn

DCS/Development
Headquarters, USAF
The Pentagon Building
Washington 25, D.C.
Attn: Armament Division

Commander
Air Force Armament Center
Eglin Air Force Base, Fla.
Attn: ACGL, Technical Library

Commander (2 copies)
Wright Air Development Center
Wright-Patterson AFB, Ohio
Attn: WCRRB, Dr. J.E. Clemens

Mr. Melvin Jenney
Patent Office
Building 5-118, M.I.T.

M.I.T., Lincoln Laboratory
Box 30
Cambridge 39, Mass.
Attn: Mary Granese
Librarian

Naval Air Development Center
Johnsville, Penn.
Attn: Air Force Development
Field Representative, WCOLE

Commander
Air Force Cambridge Research Center
230 Albany Street
Cambridge 39, Mass.
Attn: CRQST-2, Electronics

University of Chicago
Air Weapons Research Center
Museum of Science and Industry
Chicago 37, Ill.
Attn: CSDAF

Weapons Systems Evaluation Group
Office, Secretary of Defense
Washington 25, D.C.
Attn: Col. C.G. Dodge, Amor,
Exec. Secretary

Commander
Air Research and Development Center
P.O. Box 1395
Baltimore 3, Md.
Attn: RDDDR-1, Col R.E. Jarmon

Commander (3 copies)
Wright Air Development Center
Wright-Patterson AFB, Ohio
Attn: WCOSR, Mr. S. Sheppard

Commandant
Air Force Institute of Technology
Wright-Patterson AFB, Ohio
Attn: MCLI, Electrical Engineering Dept.

Commander
Air Research and Development Command
P.O. Box 1395
Baltimore 3, Md.
Attn: RDDD, Gen'l L.I. Davis

Air Force Development Field Representative
Naval Ordnance Test Station
Inyokern, China Lake, California
Attn: Maj. J.B. Goudy

INITIAL DISTRIBUTION (con't)

Director of Operations
Deputy of Chief of Staff Operations
Headquarters, USAF
Washington 25, D.C.
Attn: Dr. L.A. Brothers
Operations Analysis Division

Commander
Aberdeen Proving Ground, Md.
Attn: Ballistics Research Lab.

Commander
Wright Air Developm^nt Center
Wright-Patterson Al ʻhio
Attn: WCSB

Commander
Strategic Air Command
Offutt Air Force Base, Nebraska
Attn: Col. W.R. McKelvey
Tactical Requirements Div.
D/ Ops.

Commander
Strategic Air Command
Offutt Air Force Base, Nebraska
Attn: Col. J. V. Dickson
Armament-Electronics Div.
D/ Mat.

Director of Requirements
Headquarters USAF
Washington 25, D.C.
Attn: AFDRQ
Lt. Col. Herman Dorfman

Commander
Tactical Air Command
Langley Air Force Base, Virginia
Attn: TNAMR, Maj. E. T. Eden

Air Force Development
Field Representative
Naval Air Test Center
Patuxent River, Md.
Attn: Maj. J.E. Foley

Commander
Wright Air Development Center
Wright-Patterson AFB, Ohio
Attn:WCLGS-3
Capt. W.W. McKenna

Director, Naval Research Laboratory
Anacostia
Washington 20, D.C.
Attn: Code 1110

Petroleum Reference Laboratory
Pennsylvania State College
State College, Penn.
Attn: Prof. M.R. Fenske

Chief, Bureau of Ships
Department of the Navy
Washington 25, D.C.
Attn: Code 822

Commander
Air Force Missile Test Center
Patrick Air Force Base
Cocoa, Florida
Attn: Technical Library

President
Marine Corps Equipment Board
Marine Corps School
Quantico, Virginia
Attn: Col. C.O. Bierman
Air Secretary

Commander
Wright Air Development Center
Wright-Patterson AFB, Ohio
Attn: WCLGS-3
Lt. A. Scoville

Commander
Wright Air Development Center
Wright-Patterson AFB, Ohio
Attn: WCLGL-4
M.S.J. Graebner

Commander
Wright Air Development Center
Wright-Patterson AFB, Ohio
Attn: WCLOR, W.J. Sen

U.S. Naval Development Center
Johnsville, Penn.
Attn: Major Rall, AMCEFO